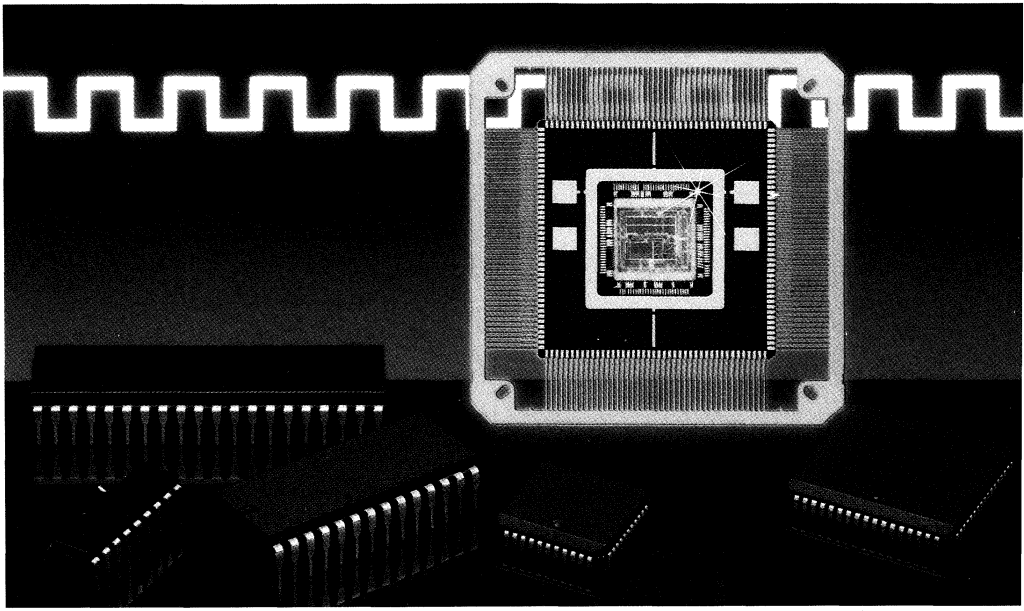
SIEMENS



Microprocessors Support Components

Data Catalog

Contents

SIEMENS

Microprocessor Components

Microprocessor Support Components

Data Catalog 1990

Edition 10.90

Published by Siemens AG, Bereich Halbleiter, Marketing-Kommunikation
Balanstraße 73, D-8000 München 80.

As far as patents or other rights of third parties are concerned, liability is only assumed for components per se, not for applications, processes and circuits implemented within components or assemblies.

The information describes the type of component and shall not be considered as assured characteristics.

Terms of delivery and rights to change design reserved.

For questions on technology, delivery and prices please contact the Offices of Siemens Aktiengesellschaft in the Federal Republic of Germany and Berlin (West) or the Siemens Companies and Representatives worldwide.

Due to technical requirements components may contain dangerous substances. For information on the type in question please contact your nearest Siemens Office, Components Group.

Siemens AG is an approved CECC manufacturer.

Contents

Contents

Type Survey for further Data Catalogs

8-bit Single-Chip Microcontrollers

SAB 8035/8048, incl. Ext. Temp. Range

SAB 80512K, CMOS, Rom-less Version

SAB 80512/80532, incl. Ext. Temp. Range

SAB 80513/8352-5, SAB 80513-16/8352-5-16, incl. Ext. Temp. Range

SAB 80515/80535

SAB 80515/80535, incl. Ext. Temp. Range

SAB 80515K, Rom-less Version

SAB 80C515/SAB 80C535, CMOS, incl. Ext. Temp. Range

SAB 80C517/80C537, SAB 80C517-16/SAB 80C537-16, CMOS, incl. Ext. Temp. Range

SAB 8051A/8031A, SAB 8051A-16/SAB 8031A-16

SAB 8051A/8031A, incl. Ext. Temp. Range

SAB 8052A/8032A

SAB 8052A/8032A, incl. Ext. Temp. Range

SAB 8052B/8032B, SAB 8052B-16/8032B-16, SAB 8032B-20

SAB 80C52/80C32, CMOS, incl. Ext. Temp. Range

SAB 83515-4, incl. Ext. Temp. Range

16-bit Single-Chip Microcontrollers

SAB 80C166/83C166, CMOS

PC-Peripherie Components

SAB 82C171, CMOS	Color Palette
SAB 82C176, CMOS	Color Palette
SAB 82C206, CMOS	Integrated Peripheral Controller
SAB 82C211, CMOS	CPU/Bus Controller of Siemens PC-AT™ Chipset
SAB 82C212, CMOS	Page/Interleave Memory Controller of Siemens PC-AT™ Chipset
SAB 82C215, CMOS	Data/Address Buffer of Siemens PC-AT™ Chipset

System Components

SAB 16C550A	Universal Asynchronous Receiver/Transmitter with FIFOs
SAB 7201A	Multi-Protocol Serial Communication Controller
SAB 8155/SAB 8155-2	RAM, stat., with I/O and Timer
SAB 82257	High-Performance DMA Controller for 16-bit Microcomputer Systems
SAB 82258A	Advanced DMA Controller (ADMA) for 16-/32-bit Microcomputer Systems
SAB 82C258A	Advanced DMA Controller (ADMA) for 16-/32-bit Microcomputer Systems
SAB 8237A/SAB 8237A-5	Programmable DMA Controller
SAB 82C37A-5/SAB 82C37A-8	CMOS, Programmable DMA Controller
SAB 82C37B-5/SAB 82C37B-8	CMOS, Programmable DMA Controller
SAB 82C50/SAB 16C450, CMOS	Universal Asynchronous Receiver/Transmitter
SAB 82C51A, CMOS	CMOS, Programmable Communications Interface
SAB 82C552/SAB 82C551	Advanced Peripheral Interface Controller with FIFOs
SAB 82511	Token Bus Modem (TBM)
SAB 82C53, CMOS	Programmable Interval Timer
SAB 82C54, CMOS	Programmable Interval Timer
SAB 82C55A-2, CMOS	Programmable Peripheral Interface
SAB 82556	Universal System Interface Controller
SAB 8256A/SAB 8256A-2	Programmable Multifunction Controller (MUART)
SAB 8259A/SAB 8259A-2	Programmable Interrupt Controller
SAB 82C59A-2, CMOS	Programmable Interrupt Controller

Memory Components

HYB 41256-10/-12/-15	262, 144-Bit Dynamic RAM
HYB 514256B-60/-70/-80	256K x 4-bit Dynamic RAM
HYB 514256BL-60/-70	LOW POWER VERSION
HYB 511000B-60/-70/-80	1M x 1-bit Dynamic RAM
HYB 511000BL-60/-70	LOW POWER VERSION
HYB 514100-80/-10	4M x 1-bit Dynamic RAM
HYB 514400-80/-10	1M x 4-bit Dynamic RAM
HYM 91000SL-60/-70/-80 HYM 91000L-60/-70/-80 HYM 91000SL-60/-70 HYM 91000LL-60/-70/-80	1,048,576 x 9-bit Dynamic RAM Module
HYM 94000S-80/-10	4,194,304 x 9-bit Dynamic RAM Module
HYM 39500S-80	256K x 9-bit Dynamic RAM Module
HYM 362500S-80	256K x 36-bit Dynamic ROM Module
HYM 365120S-80	512K x 36-bit Dynamic RAM Module
HYM 361020S-80/-10	1M x 36-bit Dynamic RAM Module
HYM 362020S-80/-10	2M x 36-bit Dynamic RAM Module

General Information

General Information

Type designation code for ICs

IC type designations are based on the European Pro Electron system. The code system is explained in the Pro Electron brochure D 15, edition 1985, available at:

Pro Electron, Avenue Louise, 430 (B. 12)
B-1060 Brussels, Belgium

Mounting instructions

Plastic Package

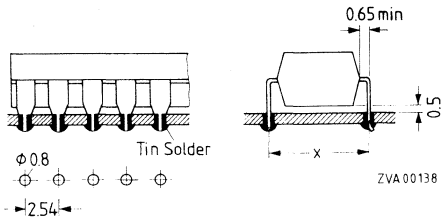
The 90° pins fit into holes with a diameter of 0.7 to 0.9 mm, spaced 2.54 mm apart. See spacing x in figure 1.

The bottom of the package will not touch the PC board after insertion because the pins have shoulders just below the package **(see figure 1)**.

After insertion of the package into the PC board it is advisable to bend the ends of two pins at an angle of approx. 30° to the board so that the package does not have to be pressed down during soldering. Plastic packages are soldered on that side of the PCB facing away from the package.

The maximum permissible soldering temperature is 350 °C (max. 3 s) for hand soldering and 260 °C (max. 10 s) for dip soldering and wave soldering.

Figure 1



Dimensions in mm

Plastic packages (SO and PLCC) for surface mounting (SMD)

Iron soldering:	soldering temperature 350 °C for max. 3 s; minimum distance between package and soldering point 1.5 mm package temperature max. 150 °C; no mechanical stress on the pins
Vapor phase soldering:	soldering temperature 215 °C, max. soldering time 40 s
Wave soldering: (pins and package are dipped into the tin bath)	soldering temperature 260 °C, max. soldering time 8 s

Storage, pretreatment before processing

The components are to be stored in a dry environment. When solder methods causing solder heat shock stresses are used (reflow soldering where the component is dipped into the solder bath, vapor-phase soldering) it is recommendable to subject IC's in plastic packages to a 24-hour drying phase at 125 °C.

Other points to note

Ensure that no current is able to flow between the solder bath or soldering iron and the PCB. It is advisable to ground the pins that are to be soldered as well as the solder bath or soldering iron.

When the pins are being prepared and inserted in a PCB, circuits should be protected against static charging. Under no circumstances should the components be removed or inserted while the operating voltage is switched on.

The increase in chip temperature during the soldering process results in a temporary increase in electrostatic sensitivity of integrated circuits. Special precautions should therefore be taken against line transients, e.g. through the switching of inductances on magnetic chutes, etc.

Processing guidelines for ICs

Integrated circuits (ICs) are **e**lectro**s**tatic-**s**ensitive (**ESS**) devices. The requirement for greater packing density has led to increasingly small structures on semiconductor chips with the result that today every IC, whether bipolar, MOS, or CMOS, has to be protected against electrostatics.

MOS and CMOS devices generally have integrated protective circuits and it is hardly possible any more for them to be destroyed by purely static electricity. On the other hand, there is acute danger from **e**lectro**s**tatic **d**ischarges (**ESD**).

Of the multiple of possible sources of discharge, charged devices should be mentioned in addition to charged persons. With low-resistive discharges it is possible for peak power amounting to kilowatts to be produced.

For the protection of devices the following principles should be observed:

a) Reduction of charging voltage, below 200 V if possible.
Means which are effective here are an increase in relative humidity to ≥ 60 % and the replacement of highly charging plastics by antistatic materials.

b) With every king of contact with the device pins a charge equalization is to be expected. This should always be highly resistive (ideally $R = 10^6$ to $10^8\,\Omega$).

All in all this means that ICs call for special handling, because uncontrolled charges, voltages from ungrounded equipment or persons, surge voltage spikes and similar influences can destroy a device. Even if devices have protective circuits (e.g. protective diodes) on their inputs, the following guidelines for their handling should nevertheless be observed.

Identification

The packing of ESS devices is provided with the following label by the manufacturer:

Scope

The guidelines apply to the storage, transport, testing, and processing of all kinds of ICs, equipped and soldered circuit boards that comprise such components.

Handling of devices

1. ICs must be left in their containers until they are processed.
2. ICs may only be handled at specially equipped work stations. These stations must have work surfaces covered with a conductive material of the order of 10^6 to $10^9\ \Omega$/cm.
3. With humidity of > 50 % a coat of pure cotton is sufficient. In the case of chargeable synthetic fibers the clothing should be worn close-fitting. The wrist strap must be worn snugly on the skin and be grounded across a resistor of 50 to 100 Ω.
4. If conductive floors, $R = 5 \times 10^4$ to $10^7\ \Omega$ are provided, further protection can be achieved by using so-called MOS chairs and shoes with a conductive sole ($R \approx 10^5$ to $10^7\ \Omega$).
5. All transport containers for ESS devices and assembled circuit boards must first be brought to the same potential by being placed on the work surface or touched by the operator before the individual devices may be handled. The potential equalization should be across a resistor of 10^6 to $10^8\ \Omega$.
6. When loading machines and production devices it should be noted that the devices come out of the transport magazine charged and can be damaged if they touch metal, e.g. machine parts.

Example 1) Conductive (black) tubes.
The devices may be destroyed in the tube by charged persons or come out of the tube charged if this in emptied by a charged person.
Conductive tubes may only be handled at ESS work stations (high-resistance work-station and person grounding).

Example 2) Anti-static (transparent) tubes.
The devices cannot be destroyed by charged persons in the tube (there may be a rare exception in the case of custom ICs with unprotected gate pins).
The devices can be endangered as in 1) when the tube is emptied if the latter, especially at low humidity, is no longer sufficiently anti-static after a long period of storage (> 1 year).

In both cases damage can be avoided by discharging the devices across a grounded adapter of high-resistance material ($\approx 10^6$ to $10^8\ \Omega$/cm) between the tube and the machine.

The use of metal tubes – especially of anodized aluminum – is not advisable because of the danger of low-resistance device discharge.

Storage

ESS devices should only be stored in identified locations provided for the purpose. During storage the devices should remain in the packing in which they are supplied. Thestorage temperature should not exceed 60 °C.

Transport

ESS devices in approved packing tubes should only be transported in suitable containers of conductive or longterm anti-static-treated plastic or possibly unvarnished wood. Containers of high-charging plastic or very low-resistance materials are likewise unsuitable.

Transfer cars and their rollers should exhibit adequate electrical conductivity ($R < 10^6\ \Omega$). Sliding contacts and grounding chains will not reliably eliminate charges.

Incoming inspection

In incoming inspection the above guidelines should be observed. Otherwise any right for refund or replacement if devices fail inspection may be lost.

Material and mounting

1. The drive belts of machines used for the processing of the devices, in as much as they come into contact with them (e.g. bending and cutting machines, conveyor belts), should be treated with anti-static spray (e.g. anti-static spray 100 from Kontaktchemie). It is better, however, to avoid the contact completely.
2. If ESS devices have to be soldered or desoldered manually, soldering irons with thyristor control cannot be used. Siemens EMI-suppression capacitors of the type B 81711-B31 ... -B36 have propen very effective against line transients.
3. Circuit boards fitted and soldered with ESS devices are always to be considered as endangered.

Electrical tests

1. The devices should be processed with observation of these guidelines. Before assembled and soldered circuit boards are tested, remove any shorting ring.
2. Test sockets must not be conducting any voltage when individual devices or assembled circuit boards and inserted or withdrawn, unless works' specifications state otherwise. Ensure that the test devices do not produce any voltage spikes, either when being turned on and off in normal operation or if the power fuse blows or other fuses respond.
3. Signal voltages may only be applied to the inputs of ICs when or after the supply voltage is turned on. They must be disconnected before or when the supply voltage is turned off.
4. Observe any notes and instructions in the respective data books.

Packing of assembled PC boards or flatpack units

The packing material should exhibit low volume conductivity:
$10^5\ \Omega/cm < \rho < 10^{10}\ \Omega/cm$.

In most cases - especially with humidity of > 40 % - this requirement is fulfilled using simple currugated board. Better protection is obtained with bags of conductive polyethylene foam (e.g. RCAS 1200 from Richmond of Redlands, California).

It must always be ensured that boards do not touch.

In special cases it may be necessary to provide potection against strong electric fields, such as can be generated by conveyor belts for example. For this purpose a sheath of aluminum foil is recommended, although direct contact between the film and the PCB must be avoided. Cardboard boxes with an aluminum-foil lining, such as those used for shipping of our devices, are available from Laber of Munich.

Ultrasonic cleaning of ICs

In incoming inspection the above guidelines should be observed. Otherwise any right for refund or replacement if devices fail inspection may be lost.

The following recommendation applies to plastic packages. For cavity packages (metal and also ceramic) separate regulations have to be observed.

Freon and isopropyl alcohol (trade name: propanol) can be used as solvents. These solvents can also be used for plastic packages because they do not eat into the plastic material.

An ultrasonic bath in double halfwave operation is advisable because of the low component stress.

The ultrasonic limits are as follows:

sound frequency	f > 40 kHz
exposure	t > 2 min
alternating sound pressure	p > 0.29 bar
sound power	N > 0.5 W/cm²/liter

Data classification

Maximum ratings

Maximum ratings are absolute ratings; exceeding only one of these values may cause irreversible damage to the integrated circuit.

Characteristics

The listed characteristics are ensured over the operating range of the integrated circuit. Typical characteristics specify mean values expected over the production spread. If not otherwise specified, typical characteristics will apply at T_A = 25 °C and for the given supply voltage.

Operating range

In the operating range the functions given in the circuit description will be fulfilled.

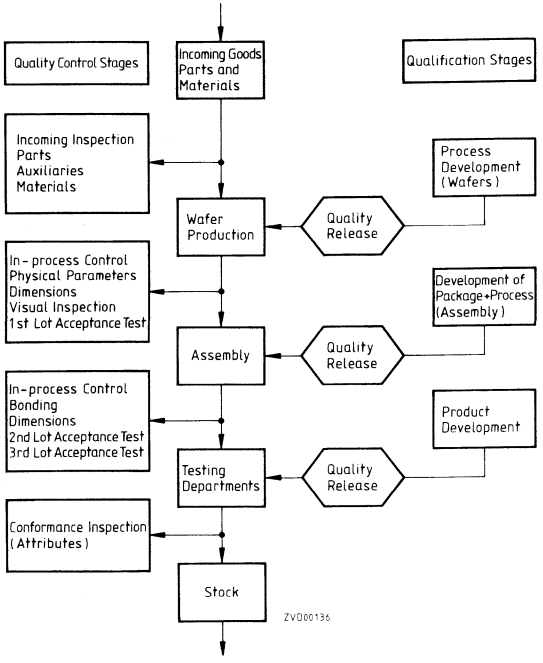
Quality assurance system

The high quality and reliability of integrated circuits from Siemens is the result of a carefully arranged production which is systematically checked and controlled at each production stage.

The procedures are subject to a quality assurance system; full details are given in the brochure "Siemens Quality Assurance System - Integrated Circuits" (SQS-IC).

Figure 4 shows the most important stages of the "SQS-IC". A quality assurance (QA) department which is independent of production and development, is responsible for the selected control measures, acceptance procedures, and information feedback loops. This department has state-of-the-art test and measuring equipment at its disposal, works according to approved methods of statistical quality control, and is provided with facilities for accelerated life and environmental tests used for both qualification and routine monitoring tests.

Figure 4



The latest methods and equipment for preparation and analysis are employed to achieve continuity of quality and reliability.

Conformance

Each integrated circuit is subjected to a final test at the end of the production process. These tests are carried out by computer-controlled, automatic test systems because hundreds of thousands of operating conditions as well as a large number of static and dynamic parameters have to be considered. Moreover, the test systems are extremely reliable and reproducible. The quality assurance department carries out a final check in the form of a lot-by-lot sampling inspection to additionally ensure this minimum percent defectives as well as the acceptable quality level (AQL). Sampling inspection is performed in accordance with the inspection plans of DIN 40080, as well as of the identical MIL-STD-105 or IEC 410.

The table shows the results of such sampling inspections performed with hundreds of thousands of ICs in 1985. These results correspond to the average outgoing quality (AOQ), and are specified as defectives per million (DPM).

	Inoperatives AOQ (DPM)	**Sum of electrical defectives AOQ (DPM)**	**Sum of mechanical defectives AOQ (DPM)**
SSI/MSI ≤ 1000 gate functions	40	200	100
LSI/MSI ≤ 1000 gate functions	120	400	200

Reliability

Measures taken during development

The reliability of ICs is already considerably influenced at the development stage. Siemens has, therefore, fixed certain design standards for the development of circuit and layout, specifying e.g. minimum width and spacing of conductive layers on a chip, dimensions and electrical parameters of protective circuits for electrostatic charge, etc. An examination with the aid of carefully arranged programs operated on large-scale computers, guarantees the immediate identification and elimination of unintentional violations of these design standards.

In-process control during production

The manufacturing of integrated circuits comprises several hundred production steps. As each step is to be executed with utmost accuracy, the in-process control is of outstanding importance. Some processes require more than a hundred different test measures. The tests have been arranged such that the individual process steps can be reproduced continuously.

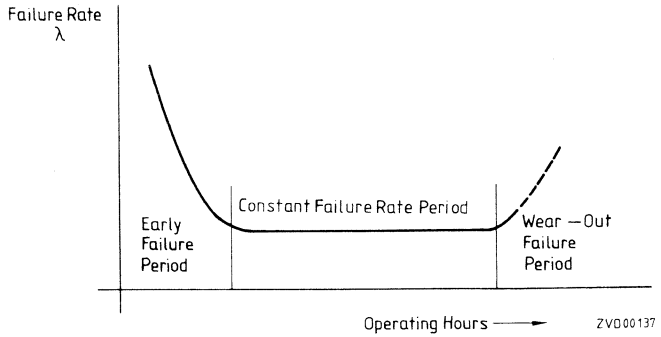
The decreasing failure rates reflect the never ending effort in this direction; they have been reduced considerably despite an immense rise in the IC's complexity.

So in 1985 the typical random failure rates estimated for accelerated life tests with almost 2 million ICs of all complexities are found to be around 80 fit.

Reliability monitoring

The general course of the ICsfailurerateversustimeis'shown by a so-called "bathtub" curve (**figure 5**). The failure rate has its peak during the first few operating hours (early failure period). After the early failure period has decayed, the "constant" failure rate period starts during which the failures may occur at an approximately uniform rate. This period ends with a repeated rise of the curve during the wear-out failure period. For ICs, however, the latter period usually lies far beyond the service life specified for the individual equipment.

Figure 5



Reliability tests for ICs are usually destructive examinations. They are, therefore, carried out with samples. Most failure mechanisms can be accelerated by means of higher temperatures. Due to the temperature dependence of the failure mechanisms, it is possible to simulate future operational behavior within a short time by applying high temperatures; this is called life test.

The acceleration factor B for the life test can be obtained from the Arrhenius equation

$$B = \exp \frac{E_A}{k}\left[\frac{1}{T_1} - \frac{1}{T_2}\right]$$

where T_2 is the temperature at which the life test is performed, T_1 is the assumed operating temperature, and k is the Boltzmann constant.

Important for factor B is the activation energy E_A. It lies between 0.3 and 1.3 eV and differs considerably for individual failure mechanisms.

For all Siemens ICs, the reliability data from life tests is converted to an operating temperature of T_A = 40 °C, assuming an average activation energy of 0.4 eV. The acceleration factor for life tests at 125 °C is thus 24, compared with operational behavior. This method considers also failure mechanisms with low activation energy, i.e. which are only slightly accelerated by the temperature effect.

Various reliability tests are periodically performed with IC types that are representative of a certain production line - this is described in the brochure "SQS-IC". Such tests are e.g. humidity test at 85 °C and 85 % relative humidity, pressure cooker test, as well as life tests up to 1000 hours and more. Test results are available in the form of summary reports.

Summary of Types (incl. ordering codes)

Summary of Types

Type	Ordering Code	Package	Description	Page
8-/16-Bit Microprocessors				
SAB 80186-1-N	Q67120-C306	PL-CC-68	16-bit Microprocessor, 10 MHz	
SAB 80186-N	Q67120-C250	PL-CC-68	16-bit Microprocessor, 8 MHz	
SAB 80188-1-N	Q67120-C299	PL-CC-68	8-bit Microprocessor, 10 MHz	
SAB 80188-N	Q67120-C252	PL-CC-68	8-bit Microprocessor, 8 MHz	
SAB 80286-16-N	Q67120-C479	PL-CC-68	16-bit Microprocessor, 12,5 MHz	
SAB 8085AH-P	Q67120-C122	P-DIP-40	8-bit Microprocessor, 3 MHz	
SAB 8085AH-2P	Q67120-C124	P-DIP-40	8-bit Microprocessor, 5 MHz	
SAB 8086-1-P	Q67120-C141	P-DIP-40	16-bit Microprocessor, 10 MHz	
SAB 8086-2-P	Q67120-C142	P-DIP-40	16-bit Microprocessor, 8 MHz	
SAB 8086-P	Q67120-C116	P-DIP-40	16-bit Microprocessor, 5 MHz	
SAB 8088-1-N	Q67120-C321	PL-CC-44	8-bit Microprocessor, 10 MHz	
SAB 8088-1-P	Q67120-C249	P-DIP-40	8-bit Microprocessor, 10 MHz	
SAB 8088-2-N	Q67120-C302	PL-CC-44	8-bit Microprocessor, 8 MHz	
SAB 8088-2-P	Q67120-C213	P-DIP-40	8-bit Microprocessor, 8 MHz	
SAB 8088-N	Q67120-C301	PL-CC-44	8-bit Microprocessor, 5 MHz	
SAB 8088-P	Q67120-C106	P-DIP-40	8-bit Microprocessor, 5 MHz	

Summary of Types (cont'd)

Type	Ordering Code	Package	Description	Page
32-Bit Microprocessor				
SAB-R2000A-12-A	Q67120-C552	C-PGA-145	RISC Processor, 12,5 MHz	
SAB-R2000A-16-A	Q67120-C494	C-PGA-145	RISC Processor, 16,67 MHz	
SAB-R2000A-20-A	Q67120-C517	C-PGA-145	RISC Processor, 20 MHz	
SAB-R2010A-12-QJ	Q67120-C553	CL-CC-84	Floating-Point Coprocessor, 12,5 MHz	
SAB-R2010A-16-QJ	Q67120-C495	CL-CC-84	Floating-Point Coprocessor, 16,67 MHz	
SAB-R2020-16-N	Q67120-C556	PL-CC-68	Write Buffer, 16,67 MHz	
SAB-R3000-16-A	Q67120-C554	C-PGA-145	RISC Processor, 16,67 MHz	
SAB-R3000-25-A	Q67120-C514	C-PGA-145	RISC Processor, 25 MHz	
SAB-R3000-25-QF	Q67120-C496	C-QFP-172	RISC Processor, 25 MHz	
SAB-R3010-16-QJ	Q67120-C555	CL-CC-84	Floating-Point Coprocessor, 16,67 MHz	
SAB-R3010-20-QJ	Q67120-C565	CL-CC-84	Floating-Point Coprocessor, 20 MHz	
SAB-R3010-25-QJ	Q67120-C497	CL-CC-84	Floating-Point Coprocessor, 25 MHz	
SAB-R3020-25-N	Q67120-C557	PL-CC-68	Write Buffer, 25 MHz	

Summary of Types (cont'd)

Type	Ordering Code	Package	Description	Page
32-Bit Microprocessor				
SAB-R2000A-12-A	Q67120-C552	C-PGA-145	RISC Processor, 12,5 MHz	
SAB-R2000A-16-A	Q67120-C494	C-PGA-145	RISC Processor, 16,67 MHz	
SAB-R2000A-20-A	Q67120-C517	C-PGA-145	RISC Processor, 20 MHz	
SAB-R2010A-12-QJ	Q67120-C553	CL-CC-84	Floating-Point Coprocessor, 12,5 MHz	
SAB-R2010A-16-QJ	Q67120-C495	CL-CC-84	Floating-Point Coprocessor, 16,67 MHz	
SAB-R2020-16-N	Q67120-C556	PL-CC-68	Write Buffer, 16,67 MHz	
SAB-R3000-16-A	Q67120-C554	C-PGA-145	RISC Processor, 16,67 MHz	
SAB-R3000-25-A	Q67120-C514	C-PGA-145	RISC Processor, 25 MHz	
SAB-R3000-25-QF	Q67120-C496	C-QFP-172	RISC Processor, 25 MHz	
SAB-R3010-16-QJ	Q67120-C555	CL-CC-84	Floating-Point Coprocessor, 16,67 MHz	
SAB-R3010-20-QJ	Q67120-C565	CL-CC-84	Floating-Point Coprocessor, 20 MHz	
SAB-R3010-25-QJ	Q67120-C497	CL-CC-84	Floating-Point Coprocessor, 25 MHz	
SAB-R3020-25-N	Q67120-C557	PL-CC-68	Write Buffer, 25 MHz	

8-/16-Bit Microprocessors

SAB 8085AH
8-Bit Microprocessor

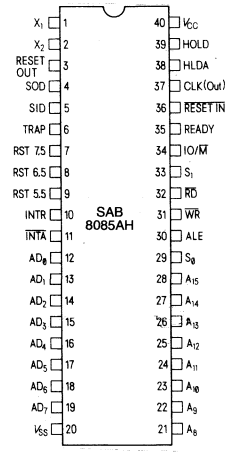
SAB 8085AH (3 MHz)

- Single +5V Power Supply with ±10% Voltage Margins
- 30% Less I_{CC} than SAB 8085A
- 100% Software Compatible with SAB 8080A
- 1.3 µs Instruction Cycle (SAB 8085AH); 0.8 µs (SAB 8085AH-2)
- On-Chip Clock Generator (with External Crystal, LC or RC Network)
- On-Chip System Controller; Advanced Cycle Status Information Available for Large System Control

SAB 8085AH-2 (5 MHz)

- Four Vectored Interrupt Inputs (one is Non-Maskable) Plus an SAB 8080A Compatible Interrupt
- Serial In/Serial Out Port
- Decimal, Binary and Double Precision Arithmetic
- Direct Addressing Capability to 64K Bytes of Memory

Pin Diagram



Pin Names

A_8–A_{15}	Address Bus	$\overline{INTA}$	Interrupt Acknowledge
AD_0–AD_7	Mux. Address/ Data Bus	RST 5.5, 6.5, 7.5	Restart Interrupts
ALE	Address Latch Enable	TRAP	Trap
S_0, S_1, IO/$\overline{M}$	Machine Cycle Status	$\overline{RESET\ IN}$	Reset in
		RESET OUT	Reset out
$\overline{RD}$	Read Control		
$\overline{WR}$	Write Control	X_1, X_2	Crystal/Clock Input
READY	Ready	CLK	Clock Output
HOLD	Hold	SID	Serial Input Data
HLDA	Hold Acknowledge	SOD	Serial Output Data
INTR	Interrupt Request	V_{CC}	+5V
		V_{SS}	Ground (OV)

SAB 8085AH is a complete 8 bit parallel Central Processing Unit (CPU). Its instruction set is 100% software compatible with the SAB 8080A microprocessor. Its high level of system integration allows a minimum of three IC's [SAB 8085AH (CPU), SAB 8156 (RAM/IO) and SAB 8355/SAB 8755A (ROM/PROM/IO)] while maintaining total system expandability. The SAB 8085AH-2 is a faster version of the SAB 8085AH.

The SAB 8085AH incorporates all of the features that the SAB 8224 (clock generator) and SAB 8228 (system controller) provided for the SAB 8080A, thereby offering a high level of system integration.

The SAB 8085AH uses a multiplexed data bus. The address is split between the 8 bit address bus and the 8 bit data bus. The on-chip address latches of SAB 8155/SAB 8156/SAB 8355/SAB 8755A memory products allow a direct interface with the SAB 8085AH.

SAB 8085AH is implemented in +5V advanced N-channel, silicon gate Siemens MYMOS technology and is a selected version of the standard SAB 8085A.

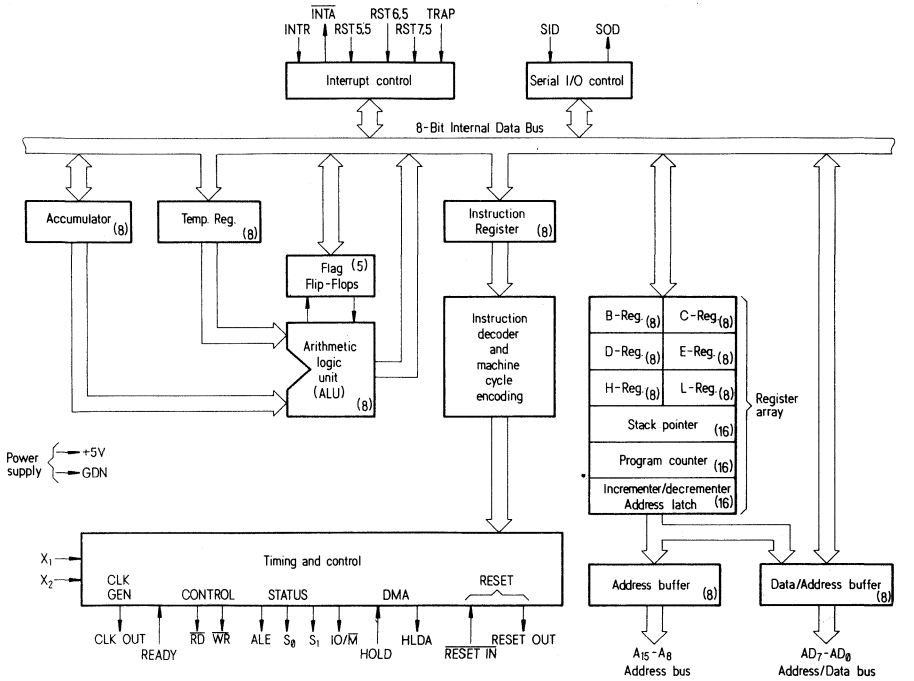
Pin Definitions and Functions

Symbol	Number	Input (I) Output (O)	Function
X_1, X_2	1, 2	I	X_1 AND X_2 – Are connected to a crystal, LC, or RC network to drive the internal clock generator. X_1 can also be an external clock input from a logic gate. The input frequency is divided by 2 to give the processor's internal operating frequency.
RESET OUT	3	O	RESET OUT – Reset Out indicates CPU is being reset. Can be used as a system reset. The signal is synchronized to the processor clock and lasts an integral number of clock periods.
SOD	4	O	SERIAL OUTPUT DATA LINE – The output SOD is set or reset as specified by the SIM instruction.
SID	5	I	SERIAL INPUT DATA LINE – The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.
TRAP	6	I	TRAP – Trap interrupt is a nonmaskable RESTART interrupt. It is recognized at the same time as INTR or RST 5.5-7.5. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt (see following table).
RST 5.5 RST 6.5 RST 7.5	9 8 7	I	RESTART INTERRUPTS – These three inputs have the same timing as INTR except they cause an internal RESTART to be automatically inserted. The priority of these interrupts is ordered as shown in the following table. These interrupts have a higher priority than INTR. In addition, they may be individually masked out using the SIM instruction.
INTR	10	I	INTERRUPT REQUEST – Is used as a general purpose interrupt. It is sampled only during the next to the last clock cycle of an instruction and during Hold and Halt states. If it is active, the Program Counter (PC) will be inhibited from incrementing and an $\overline{\text{INTA}}$ will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.
$\overline{\text{INTA}}$	11	O	INTERRUPT ACKNOWLEDGE – Is used instead of (and has the same timing as) $\overline{\text{RD}}$ during the instruction cycle after an INTR is accepted. It can be used to activate an SAB 8259A Interrupt chip or some other interrupt port.
AD_0–AD_7	12–19	I/O	MULTIPLEXED ADDRESS/DATA BUS – Lower 8 bits of the memory address (or I/O address) appear on the bus during the first clock cycle (T state) of a machine cycle. It then becomes the data bus during the second and third clock cycles.
A_8–A_{15}	21–28	O	ADDRESS BUS – The most significant 8 bits of the memory address or the 8 bits of the I/O address, 3-stated during Hold and Halt modes and during RESET.

<table>
<tr><th>Symbol</th><th>Number</th><th>Input (I)
Output (O)</th><th>Function</th></tr>
<tr><td>$S_\emptyset$, S_1, and IO/$\overline{M}$</td><td>29, 33
34</td><td>O</td><td>MACHINE CYCLE STATUS –
<table>
<tr><th>IO/$\overline{M}$</th><th>S_1</th><th>$S_\emptyset$</th><th>Status</th></tr>
<tr><td>$\emptyset$</td><td>$\emptyset$</td><td>1</td><td>Memory write</td></tr>
<tr><td>$\emptyset$</td><td>1</td><td>$\emptyset$</td><td>Memory read</td></tr>
<tr><td>1</td><td>$\emptyset$</td><td>1</td><td>I/O write</td></tr>
<tr><td>1</td><td>1</td><td>$\emptyset$</td><td>I/O read</td></tr>
<tr><td>$\emptyset$</td><td>1</td><td>1</td><td>Opcode fetch</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Opcode fetch</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Interrupt Acknowledge</td></tr>
<tr><td>*</td><td>$\emptyset$</td><td>$\emptyset$</td><td>Halt</td></tr>
<tr><td>*</td><td>X</td><td>X</td><td>Hold</td></tr>
<tr><td>*</td><td>X</td><td>X</td><td>Reset</td></tr>
</table>
* = 3-state (high impedance)
X = unspecified
S_1 can be used as an advanced R/$\overline{W}$ status. IO/$\overline{M}$, $S_\emptyset$ and S_1 become valid at the beginning of a machine cycle and remain stable throughout the cycle. The falling edge of ALE may be used to latch the state of these lines.</td></tr>
<tr><td>ALE</td><td>30</td><td>O</td><td>ADDRESS LATCH ENABLE – It occurs during the first clock state of a machine cycle and enables the address to get latched into the on-chip latch of peripherals. The falling edge of ALE is set to guarantee setup and hold times for the address information. The falling edge of ALE can also be used to strobe the status information. ALE is never 3-stated.</td></tr>
<tr><td>$\overline{WR}$</td><td>31</td><td>O</td><td>WRITE CONTROL – A low level on $\overline{WR}$ indicates the data on the Data Bus is to be written into the selected memory or I/O location. Data is set up at the trailing edge of $\overline{WR}$. 3-stated during Hold and Halt modes and during RESET.</td></tr>
<tr><td>$\overline{RD}$</td><td>32</td><td>O</td><td>READ CONTROL – A low level on $\overline{RD}$ indicates the selected memory or I/O device is to be read and that the Data Bus is available for the data transfer, 3-stated during Hold and Halt modes and during RESET.</td></tr>
<tr><td>READY</td><td>35</td><td>I</td><td>READY – If READY is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If READY is low, the CPU will wait an integral number of clock cycles for READY to go high before completing the read or write cycle. READY must conform to specified setup and hold times.</td></tr>
<tr><td>$\overline{RESET\ IN}$</td><td>36</td><td>I</td><td>RESET IN – Sets the Program Counter to zero and resets the Interrupt Enable and HLDA flip-flops. The data and address buses and the control lines are 3-stated during RESET and because of the asynchronous nature of RESET, the processor's internal registers and flags may be altered by RESET with unpredictable results. $\overline{RESET\ IN}$ is a Schmitt-triggered input, allowing connection to an RC network for power-on RESET delay. The CPU is held in the reset condition as long as $\overline{RESET\ IN}$ is applied.</td></tr>
<tr><td>CLK</td><td>37</td><td>O</td><td>CLOCK – Clock output for use as a system clock. The period of CLK is twice the X_1, X_2 input period.</td></tr>
</table>

Symbol	Number	Input (I) Output (O)	Function
HLDA	38	O	HOLD ACKNOWLEDGE – Indicates that the CPU has received the HOLD request and that it will relinquish the bus in the next clock cycle. HLDA goes low after the Hold request is removed. The CPU takes the bus one half clock cycle after HLDA goes low.
HOLD	39	I	HOLD – Indicates that another master is requesting the use of the address and data buses. The CPU, upon receiving the hold request, will relinquish the use of the bus as soon as the completion of the current bus transfer. Internal processing can continue. The processor can regain the bus only after the HOLD is removed. When the HOLD is acknowledged, the Address, Data $\overline{RD}$, $\overline{WR}$, and IO/$\overline{M}$ lines are 3-stated.
V_{CC}	40		POWER SUPPLY (+5V)
V_{SS}	20		GROUND (OV)

Functional Block Diagram



Interrupt Priority, Restart Address, and Sensitivity

Name	Priority	Address Branched To[1] When Interrupt Occurs	Type Trigger
TRAP	1	24H	Rising edge AND high level until sampled
RST 7.5	2	3CH	Rising edge (latched)
RST 6.5	3	34H	High level until sampled
RST 5.5	4	2CH	High level until sampled
INTR	5	see Note 2	High level until sampled

NOTES
1. The processor pushes the PC on the stack before branching to the indicated address.
2. The address branched to depends on the instruction provided to the CPU when the interrupt is acknowledged.

Functional Description

The SAB 8085AH is a complete 8-bit parallel central processor. It is designed with N-channel depletion loads and requires a single +5 volt supply. Its basic clock speed is 3 MHz (SAB 8085AH) or 5 MHz (SAB 8085AH-2), thus improving on SAB 8080A's performance with higher system speed. Also it is designed to fit into a minimum system of three IC's: The CPU (SAB 8085AH), a RAM/IO (SAB 8156), and a ROM or EPROM/IO chip (SAB 8355 or SAB 8755A).

The SAB 8085AH has twelve addressable 8-bit registers. Four of them can function only as two 16-bit register pairs. Six others can be used interchangeably as 8-bit registers or as 16-bit register pairs. The SAB 8085AH register set is as follows:

Mnemonic	Register	Contents
ACC or A	Accumulator	8 bits
PC	Program Counter	16-bit address
BC, DE, HL	General-Purpose Registers; data pointer (HL)	8 bits × 6 or 16 bits × 3
SP	Stack Pointer	16-bit address
Flags or F	Flag Register	5 flags (8-bit space)

The SAB 8085AH uses a multiplexed Data Bus. The address is split between the higher 8-bit Address Bus and the lower 8-bit Address/Data Bus. During the first T state (clock cycle) of a machine cycle the low order address is sent out on the Address/Data bus. These lower 8 bits may be latched externally by the Address Latch Enable signal (ALE). During the rest of the machine cycle the data bus is used for memory or I/O data.

The SAB 8085AH provides $\overline{RD}$, $\overline{WR}$, S_0, S_1, and IO/$\overline{M}$ signals for bus control. An Interrupt Acknowledge signal ($\overline{INTA}$) is also provided. HOLD and all Interrupts are synchronized with the processor's internal clock. The SAB 8085AH also provides Serial Input Data (SID) and Serial Output Data (SOD) lines for simple serial interface.

In addition to these features, the SAB 8085AH has three maskable, vector interrupt pins and one nonmaskable TRAP interrupt.

Interrupt and Serial I/O

The SAB 8085AH has 5 interrupt inputs: INTR, RST 5.5, RST 6.5, RST 7.5, and TRAP. INTR is identical in function to the SAB 8080A INT. Each of the three RESTART inputs, 5.5, 6.5, and 7.5, has a programmable mask. TRAP is also a RESTART interrupt but it is nonmaskable.

The three maskable interrupts cause the internal execution of RESTART (saving the program counter in the stack and branching to the RESTART address) if the interrupts are enabled and if the interrupt mask is not set. The non-maskable TRAP causes the internal execution of a RESTART vector independent of the state of the interrupt enable or masks (see table above).

There are two different types of inputs in the restart interrupts. RST 5.5 and RST 6.5 are **high level-sensitive** like INTR (and INT on the SAB 8080) and are recognized with the same timing as INTR. RST 7.5 is **rising edge-sensitive.**

For RST 7.5, only a pulse is required to set an internal flip-flop which generates the internal interrupt request. The RST 7.5 request flip-flop remains set until the request is serviced. Then it is reset automatically. This flip-flop may also be reset

by using the SIM instruction or by issuing a $\overline{\text{RESET IN}}$ to the SAB 8085AH. The RST 7.5 internal flip-flop will be set by a pulse on the RST 7.5 pin even when the RST 7.5 interrupt is masked out.

The status of the three RST interrupt masks can only be affected by the SIM instruction and $\overline{\text{RESET IN}}$.

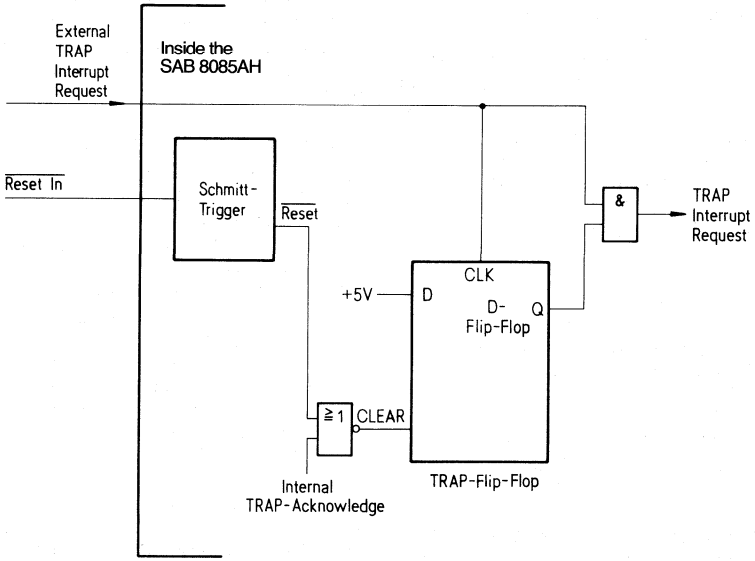
The interrupts are arranged in a fixed priority that determines which interrupt is to be recognized if more than one is pending as follows: TRAP – highest priority, RST 7.5, RST 6.5, RST 5.5, INTR – lowest priority. This priority scheme does not take into account the priority of a routine that was started by a higher priority interrupt. RST 5.5 can interrupt an RST 7.5 routine if the interrupts are re-enabled before the end of the RST 7.5 routine.

The TRAP interrupt is useful for catastrophic events such as power failure or bus error. The TRAP input is recognized just as any other interrupt but has the highest priority. It is not affected by any flag or mask. The TRAP input is both **edge and level sensitive.** The TRAP input must go high and remain high until it is acknowledged. It will not be recognized again until it goes low, then high again. This avoids any false triggering due to noise or logic glitches. The following figure illustrates the TRAP interrupt request circuitry within the SAB 8085AH. Note that the servicing of any interrupt (TRAP, RST 7.5, RST 6.5, RST 5.5, INTR) disables all future interrupts (except TRAPs) until an EI instruction is executed.

The TRAP interrupt is special in that it disables interrupts, but preserves the previous interrupt enable status. Performing the first RIM instruction following a TRAP interrupt allows you to determine whether interrupts were enabled or disabled prior to the TRAP. All subsequent RIM instructions provide current interrupt enable status. Performing a RIM instruction following INTR or RST 5.5 – 7.5 will provide current Interrupt Enable status, revealing that Interrupts are disabled.

The serial I/O system is also controlled by the RIM and SIM instructions. SID is read by RIM, and SIM sets the SOD data.

TRAP and RESET in Circuit



Driving the X_1 and X_2 Inputs

You may drive the clock inputs of the SAB 8085AH or SAB 8085AH-2 with a crystal, an LC tuned circuit, an RC network, or an external clock source. The driving frequency must be at least 1 MHz, and must be twice the desired internal clock frequency; hence, the SAB 8085AH is operated with a 6 MHz crystal (for 3 MHz clock), and the SAB 8085AH-2 can be operated with a 10 MHz crystal (for 5 MHz clock).

If a crystal is used, it must have the following characteristics:

Parallel resonance at twice the clock frequency desired

C_L (load capacitance) $\leq$ 30 pf
C_s (shunt capacitance) $\leq$ 7 pf
R_s (equivalent shunt resistance) $\leq$ 75 Ohms
Drive level: 10 mW
Frequency tolerance: $\pm$.005% (suggested)

Note the use of the 20 pF capacitor between X_2 and ground. This capacitor is required with crystal frequencies below 4 MHz to assure oscillator startup at the correct frequency. A parallel-resonant LC circuit may be used as the frequency-determining network for the SAB 8085AH, providing that its frequency tolerance of approximately $\pm$10% is acceptable. The components are from the formula:

$$f = \frac{1}{2\pi\sqrt{L\,(C_{ext} + C_{int})}}$$

To minimize variations in frequency, it is recommended that you choose a value for C_{ext} that is at least twice that of C_{int}, or 30 pF. The use of an LC circuit is not recommended for frequencies higher than approximately 5 MHz.

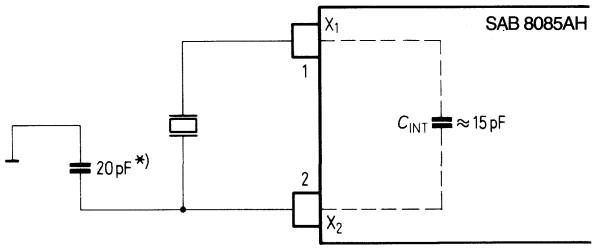
An RC circuit may be used as the frequency-determining network for the SAB 8085AH if maintaining a precise clock frequency is of no importance. Variations in the on-chip timing generation can cause a wide variation in frequency when using the RC mode. Its advantage is its low component costs. The driving frequency generated by the circuit shown is approximately 3 MHz. It is not recommended that frequencies greatly higher or lower than this be attempted.

The following figures show the recommended clock driver circuits. Note in D and E that pullup resistors are required to assure that the high level voltage of the input is at least 4V.

For driving frequencies up to and including 6 MHz you may supply the driving signal to X_1 and leave X_2 opencircuited (Figure D). If the driving frequency is from 6 MHz to 10 MHz, stability of the clock generator will be improved by driving both X_1 and X_2 with a pushpull source (Figure E). To prevent self-oscillation of the SAB 8085AH, be sure that X_2 is not coupled back to X_1 through the driving circuit.

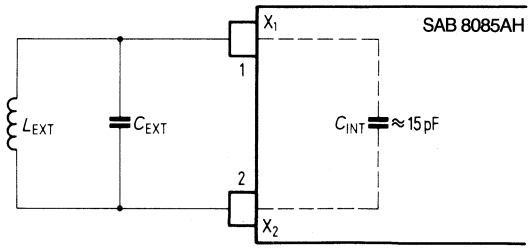
Clock Driver Circuits

A) Quartz Crystal Clock Driver

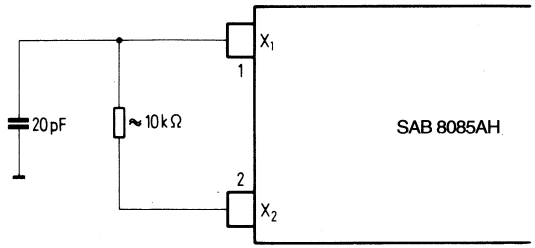


*) 20 pF Capacitors required for Crystal Frequency $\leq$ 4 MHz only.

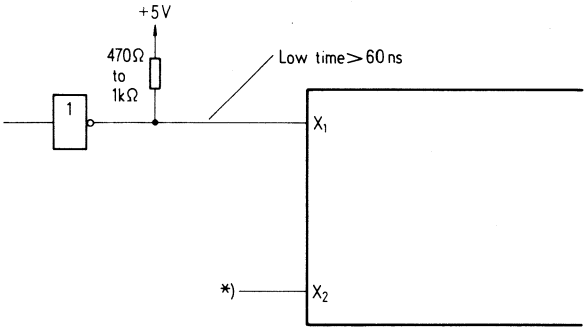
B) LC Tuned Circuit Clock Driver



C) RC Circuit Clock Driver

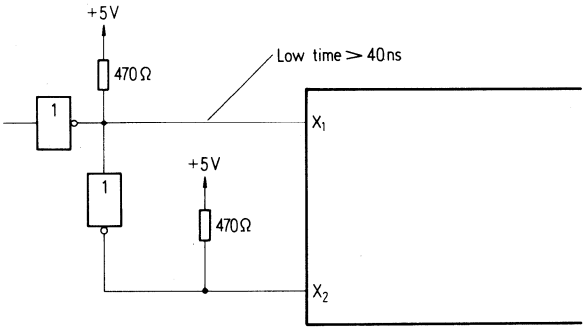


D) 1 – 6 MHz Input Frequency External Clock Driver Circuit



*) X_2 left floating

E) 1 – 10 MHz Input Frequency External Clock Driver Circuit



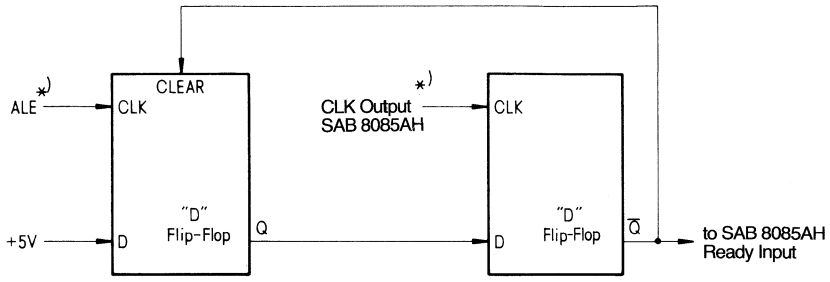
Generating Wait State

If your system requirements are such that slow memories or peripheral devices are being used, the circuit shown in the following figure may be used to insert one WAIT state in each SAB 8085AH machine cycle.

The D flip-flops should be chosen so that
- CLK is rising edge-triggered
- CLEAR is low-level active.

As in the SAB 8080, the READY line is used to extend the read and write pulse lengths so that the SAB 8085AH can be used with slow memory. HOLD causes the CPU to relinquish the bus when it is through with it by floating the Address and Data Buses.

Generation of a Wait State for SAB 8085AH CPU



*) ALE and CLK (Out) should be buffered if CLK Input of Latch Exceeds SAB 8085AH IOL or IOH.

System Interface

The SAB 8085A family includes memory components, which are directly compatible to the SAB 8085AH CPU. For example, a system consisting of the three chips, SAB 8085AH, SAB 8156, and SAB 8355 will have the following features:

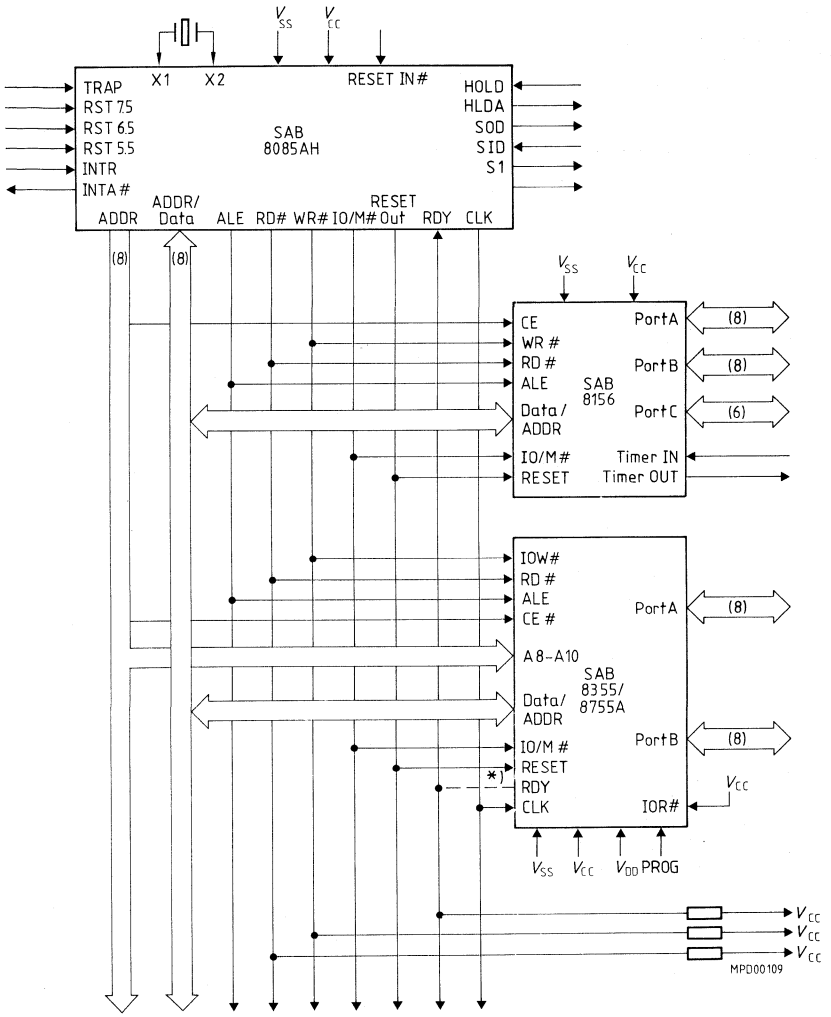
- 2K Bytes ROM
- 256 Bytes RAM
- 1 Timer/Counter
- 4 8-bit I/O Ports
- 1 6-bit I/O Port
- 4 Interrupt Levels
- Serial In/Serial Out Ports

This minimum system, using the standard I/O technique is as shown in the following figure.

In addition to standard I/O, the memory mapped I/O offers an efficient I/O addressing technique. With this technique, an area of memory address space is assigned for I/O address, thereby, using the memory address for I/O manipulation. The figure on page 11 shows the system configuration of Memory Mapped I/O using SAB 8085AH.

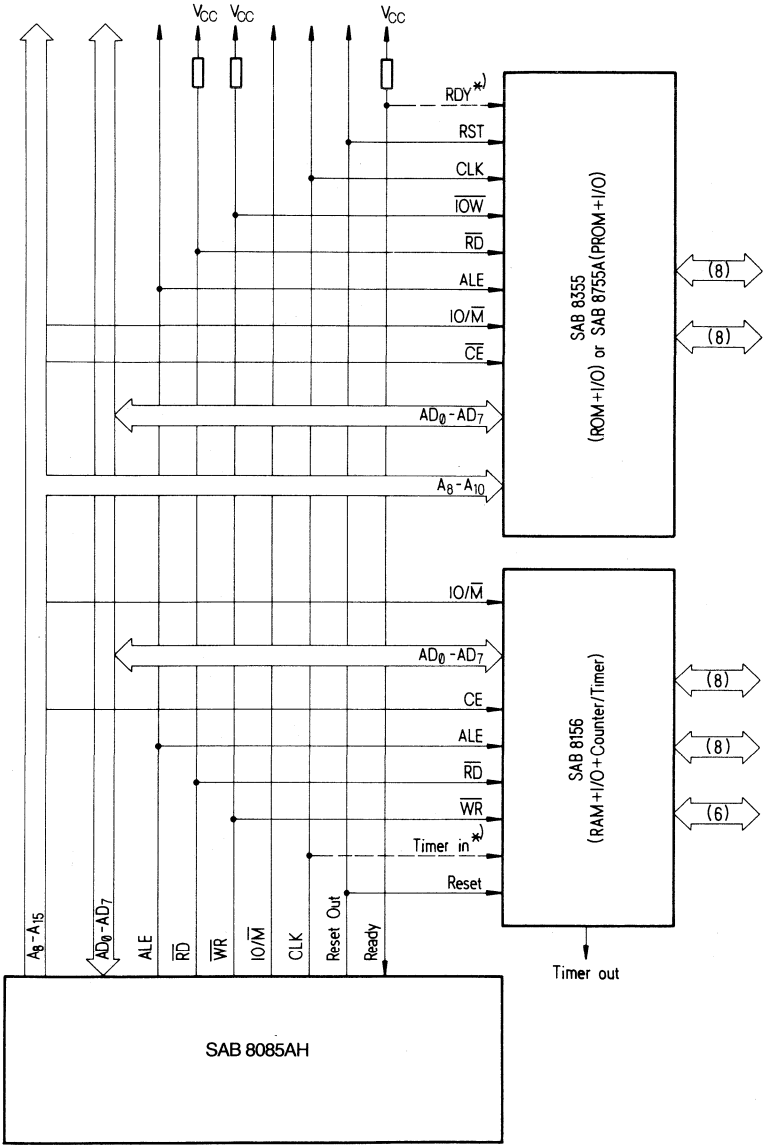
The SAB 8085AH CPU can also interface with the standard memory that does **not** have the multiplexed address/data bus. It will require a simple SAB 8282 (8-bit latch) as shown in the figure on page 12.

SAB 8085AH Minimum System (Standard I/O Technique)



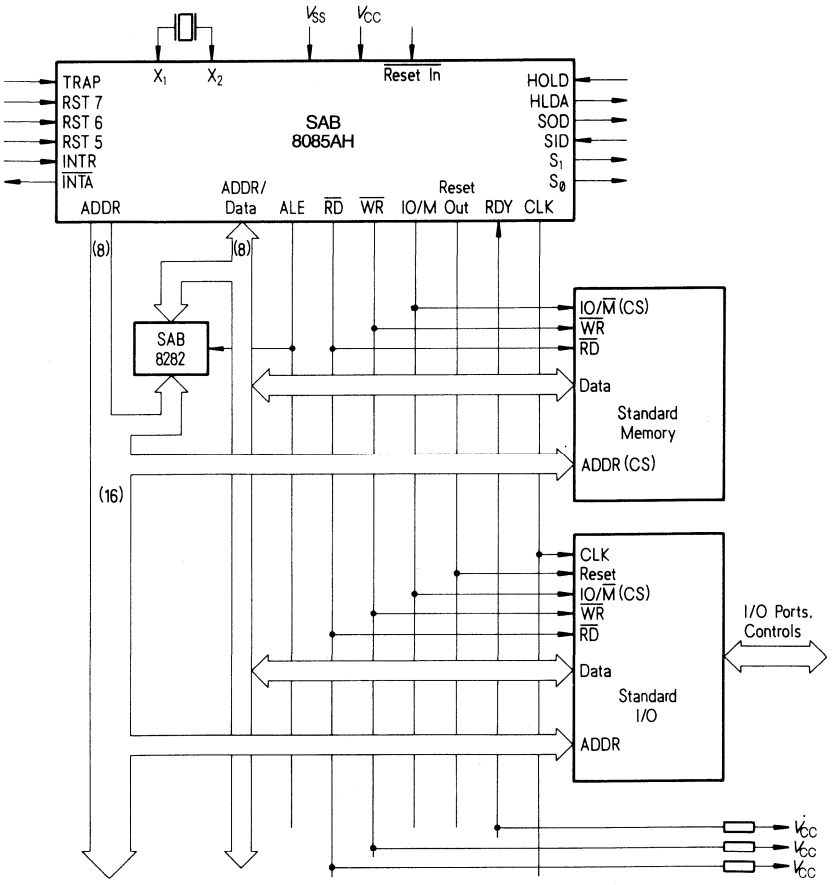
*) Optional Connection

SAB 8085AH Minimum System (Memory Mapped I/O)



*) Optional Connection

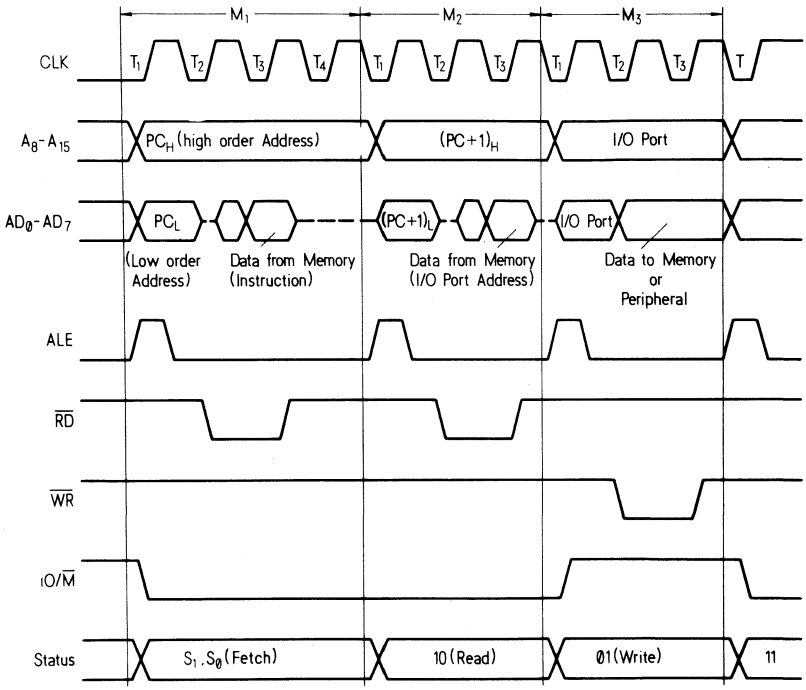
SAB 8085AH System (Using Standard Memories)



Basic System Timing

The SAB 8085AH has a multiplexed Data Bus. ALE is used as a strobe to sample the lower 8-bits of address on the Data Bus. The following figure shows an instruction fetch, memory read and I/O write cycle (as would occur during processing of the OUT instruction). Note that during the I/O write and read cycle that the I/O port address is copied on both, the upper and lower half of the address.

SAB 8085AH Basic System Timing



There are seven possible types of machine cycles. Which of these seven takes place is defined by the status of the three status lines ($IO/\overline{M}$, S_1, S_0) and the three control signals ($\overline{RD}$, $\overline{WR}$, and $\overline{INTA}$); (see following table). The status lines can be used as advanced controls (for device selection, for example), since they become active at the T_1 state, at the outset of each maching cycle. Control lines $\overline{RD}$ and $\overline{WR}$ become active later, at the time when the transfer of data is to take place, so are used as command lines.

SAB 8085AH Machine Cycle Chart

Machine Cycle		Status			Control		
		$IO/\overline{M}$	S1	SØ	$\overline{RD}$	$\overline{WR}$	$\overline{INTA}$
Opcode Fetch	(OF)	Ø	1	1	Ø	1	1
Memory Read		Ø	1	Ø	Ø	1	1
Memory Write		Ø	Ø	1	1	Ø	1
I/O Read	(IOR)	1	1	Ø	Ø	1	1
I/O Write	(IOW)	1	Ø	1	1	Ø	1
Acknowledge for INTR	(INA)	1	1	1	1	1	Ø
Bus Idle	(BI): DAD	Ø	1	Ø	1	1	1
	ACK of RST, TRAP	1	1	1	1	1	1
	HALT	TS	Ø	Ø	TS	TS	1

Ø = Logic "Ø"; 1 = Logic "1"; TS = High Impedance

A machine cycle normally consists of three T states, with the exception of OPCODE FETCH, which normally has either four or six T states (unless WAIT or HOLD states are forced by the receipt of $\overline{READY}$ or HOLD inputs). Any T state must be one of ten possible states, as summarized in the following table.

SAB 8085AH Machine State Chart

Machine State	Status and Buses				Control		
	S1, SØ	$IO/\overline{M}$	A_8–A_{15}	$AD_Ø$–AD_7	$\overline{RD}$, $\overline{WR}$	$\overline{INTA}$	ALE
T_1	X	X	X	X	1	1	1 [1]
T_2	X	X	X	X	X	X	Ø
T_{WAIT}	X	X	X	X	X	X	Ø
T_3	X	X	X	X	X	X	Ø
T_4	1	Ø [2]	X	TS	1	1	Ø
T_5	1	Ø [2]	X	TS	1	1	Ø
T_6	1	Ø [2]	X	TS	1	1	Ø
T_{RESET}	X	TS	TS	TS	TS	1	Ø
T_{HALT}	Ø	TS	TS	TS	TS	1	Ø
T_{HOLD}	X	TS	TS	TS	TS	1	Ø

Ø = Logic "Ø"; 1 = Logic "1"; TS = High Impedance; X = Unspecified.
[1]) ALE not generated during 2nd and 3nd machine cycles of DAD instruction.
[2]) $IO/\overline{M}$ = 1 during T_4–T_6 of INA machine cycle.

Instruction Set Summary

Mnemonic	Instruction Code D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Operations Description
MOVE, LOAD, AND STORE									
MOVr1 r2	0	1	D	D	D	S	S	S	Move register to register
MOV M.r	0	1	1	1	0	S	S	S	Move register to memory
MOV r.M	0	1	D	D	D	1	1	0	Move memory to register
MVI r	0	0	D	D	D	1	1	0	Move immediate register
MVI M	0	0	1	1	0	1	1	0	Move immediate memory
LXI B	0	0	0	0	0	0	0	1	Load immediate register Pair B & C
LXI D	0	0	0	1	0	0	0	1	Load immediate register Pair D & E
LXI H	0	0	1	0	0	0	0	1	Load immediate register Pair H & L
STAX B	0	0	0	0	0	0	1	0	Store A indirect
STAX D	0	0	0	1	0	0	1	0	Store A indirect
LDAX B	0	0	0	0	1	0	1	0	Load A indirect
LDAX D	0	0	0	1	1	0	1	0	Load A indirect
STA	0	0	1	1	0	0	1	0	Store A direct
LDA	0	0	1	1	1	0	1	0	Load A direct
SHLD	0	0	1	0	0	0	1	0	Store H & L direct
LHLD	0	0	1	0	1	0	1	0	Load H & L direct
XCHG	1	1	1	0	1	0	1	1	Exchange D & E, H & L Registers
STACK OPS									
PUSH B	1	1	0	0	0	1	0	1	Push register Pair B & C on stack
PUSH D	1	1	0	1	0	1	0	1	Push register Pair D & E on stack
PUSH H	1	1	1	0	0	1	0	1	Push register Pair H & L on stack
PUSH PSW	1	1	1	1	0	1	0	1	Push A and Flags on stack
POP B	1	1	0	0	0	0	0	1	Pop register Pair B & C off stack
POP D	1	1	0	1	0	0	0	1	Pop register Pair D & E off stack
POP H	1	1	1	0	0	0	0	1	Pop register Pair H & L off stack
POP PSW	1	1	1	1	0	0	0	1	Pop A and Flags off stack
XTHL	1	1	1	0	0	0	1	1	Exchange top of stack, H & L
SPHL	1	1	1	1	1	0	0	1	H & L to stack pointer
LXI SP	0	0	1	1	0	0	0	1	Load immediate stack pointer
INX SP	0	0	1	1	0	0	1	1	Increment stack pointer
DCX SP	0	0	1	1	1	0	1	1	Decrement stack pointer
JUMP									
JMP	1	1	0	0	0	0	1	1	Jump unconditional
JC	1	1	0	1	1	0	1	0	Jump on carry
JNC	1	1	0	1	0	0	1	0	Jump on no carry
JZ	1	1	0	0	1	0	1	0	Jump on zero
JNZ	1	1	0	0	0	0	1	0	Jump on no zero
JP	1	1	1	1	0	0	1	0	Jump on positive
JM	1	1	1	1	1	0	1	0	Jump on minus
JPE	1	1	1	0	1	0	1	0	Jump on parity even
JPO	1	1	1	0	0	0	1	0	Jump on perity odd
PCHL	1	1	1	0	1	0	0	1	H & L to program counter

Instruction Set Summary (Cont'd)

Mnemonic	Instruction Code D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Operations Description
CALL									
CALL	1	1	0	0	1	1	0	1	Call unconditional
CC	1	1	0	1	1	1	0	0	Call on carry
CNC	1	1	0	1	0	1	0	0	Call on no carry
CZ	1	1	0	0	1	1	0	0	Call on zerro
CNZ	1	1	0	0	0	1	0	0	Call on no zerro
CP	1	1	1	1	0	1	0	0	Call on positive
CM	1	1	1	1	1	1	0	0	Call on minus
CPE	1	1	1	0	1	1	0	0	Call on parity even
CPO	1	1	1	0	0	1	0	0	Call on parity odd
RETURN									
RET	1	1	0	0	1	0	0	1	Return
RC	1	1	0	1	1	0	0	0	Return on carry
RNC	1	1	0	1	0	0	0	0	Return on no carry
RZ	1	1	0	0	1	0	0	0	Return on zero
RNZ	1	1	0	0	0	0	0	0	Return on no zero
RP	1	1	1	1	0	0	0	0	Return on positive
RM	1	1	1	1	1	0	0	0	Return on minus
RPE	1	1	1	0	1	0	0	0	Return on parity even
RPO	1	1	1	0	0	0	0	0	Return on parity odd
RESTART									
RST	1	1	A	A	A	1	1	1	Restart
INPUT/OUTPUT									
IN	1	1	0	1	1	0	1	1	Input
OUT	1	1	0	1	0	0	1	1	Output
INCREMENT AND DECREMENT									
INR r	0	0	D	D	D	1	0	0	Increment register
DCR r	0	0	D	D	D	1	0	1	Decrement register
INR M	0	0	1	1	0	1	0	0	Increment memory
DCR M	0	0	1	1	0	1	0	1	Decrement memory
INX B	0	0	0	0	0	0	1	1	Increment B & C registers
INX D	0	0	0	1	0	0	1	1	Increment D & E registers
INX H	0	0	1	0	0	0	1	1	Increment H & L registers
DCX B	0	0	0	0	1	0	1	1	Decrement B & C
DCX D	0	0	0	1	1	0	1	1	Decrement D & E
DCX H	0	0	1	0	1	0	1	1	Decrement H & L
ADD									
ADD r	1	0	0	0	0	S	S	S	Add register to A
ADC r	1	0	0	0	1	S	S	S	Add register to A with carry
ADD M	1	0	C	0	0	1	1	0	Add memory to A
ADC M	1	0	0	0	1	1	1	0	Add memory to A with carry
ADI	1	1	0	0	0	1	1	0	Add immediate to A
ACI	1	1	0	0	1	1	1	0	Add immediate to A with carry
DAD B	0	0	0	0	1	0	0	1	Add B & C to H & L
DAD D	0	0	0	1	1	0	0	1	Add D & E to H & L
DAD H	0	0	1	0	1	0	0	1	Add H & L to H & L
DAD SP	0	0	1	1	1	0	0	1	Add stack pointer to H & L

Instruction Set Summary (Cont'd)

Mnemonic	Instruction Code D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Operations Description
SUBTRACT									
SUB r	1	0	0	1	0	S	S	S	Subtract register from A
SBB r	1	0	0	1	1	S	S	S	Subtract register from A with borrow
SUB M	1	0	0	1	0	1	1	0	Subtract memory from A
SBB M	1	0	0	1	1	1	1	0	Subtract memory from A with borrow
SUI	1	1	0	1	0	1	1	0	Subtract immediate from A
SBI	1	1	0	1	1	1	1	0	Subtract immediate from A with borrow
LOGICAL									
ANA r	1	0	1	0	0	S	S	S	And register with A
XRA r	1	0	1	0	1	S	S	S	Exclusive OR register with A
ORA r	1	0	1	1	0	S	S	S	OR register with A
CMP r	1	0	1	1	1	S	S	S	Compare register with A
ANA M	1	0	1	0	0	1	1	0	And memory with A
XRA M	1	0	1	0	1	1	1	0	Exclusive OR memory with A
ORA M	1	0	1	1	0	1	1	0	OR memory with A
CMP M	1	0	1	1	1	1	1	0	Compare memory with A
ANI	1	1	1	0	0	1	1	0	And immediate with A
XRI	1	1	1	0	1	1	1	0	Exclusive OR immediate with A
ORI	1	1	1	1	0	1	1	0	OR immediate with A
CPI	1	1	1	1	1	1	1	0	Compare immediate with A
ROTATE									
RLC	0	0	0	0	0	1	1	1	Rotate A left
RRC	0	0	0	0	1	1	1	1	Rotate A right
RAL	0	0	0	1	0	1	1	1	Rotate A left throught carry
RAR	0	0	0	1	1	1	1	1	Rotate A right through carry
SPECIALS									
CMA	0	0	1	0	1	1	1	1	Complement A
STC	0	0	1	1	0	1	1	1	Set carry
CMC	0	0	1	1	1	1	1	1	Complement carry
DAA	0	0	1	0	0	1	1	1	Decimal adjust A
CONTROL									
EI	1	1	1	1	1	0	1	1	Enable interrupts
DI	1	1	1	1	0	0	1	1	Disable Interrupt
NOP	0	0	0	0	0	0	0	0	No-operation
HLT	0	1	1	1	0	1	1	0	Halt
NEW SAB 8085AH INSTRUCTIONS									
RIM	0	0	1	0	0	0	0	0	Read Interrupt Mask
SIM	0	0	1	1	0	0	0	0	Set Interrupt Mask

NOTES
1. DDS or SSS: B 000, C 001, D 010, E 011, H 100, L 101, Memory 110, A 111.
2. Two possible cycle times (6/12) indicate instruction cycles dependent on condition flags.

Absolute maximum ratings *)

Ambient Temperature Under Bias	0 to 70°C
Storage Temperature	−65 to +150°C
Voltage on any Pin with Respect to Ground	−0.5 to +7 V
Power Dissipation	1.5 Watt

D.C. Characteristics

T_A = 0 to 70°C; V_{CC} = 5 V ±10%; V_{SS} = 0 V; (unless otherwise specified)

Symbol	Parameter	Limit Values		Units	Test Conditions
		Min.	Max.		
V_{IL}	Input Low Voltage	−0.5	+0.8	V	–
V_{IH}	Input High Voltage	2.0	V_{CC}+0.5		
V_{OL}	Output Low Voltage	–	0.45		I_{OL} = 2 mA
V_{OH}	Output High Voltage	2.4	–		I_{OH} = −400 µA
I_{CC}	Power Supply Current	–	120	mA	–
I_{IL}	Input Leakage		±10	µA	$0 \lesssim V_{IN} \lesssim V_{CC}$
I_{LO}	Output Leakage				$0.45\,V \leqslant V_{OUT} \leqslant V_{CC}$
V_{ILR}	Input Low Level, RESET	−0.5	+0.8	V	–
V_{IHR}	Input High Level, RESET	2.4	V_{CC}+0.5		
V_{HY}	Hysteresis, RESET	0.25			

*) Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

A.C. Characteristics

T_A = 0 to 70°C; V_{CC} = 5 V ± 10%; V_{SS} = 0 V

Symbol	Parameter	Limit Values				Units
		SAB 8085AH-2 [2]		SAB 8085AH [2]		
		Min.	Max.	Min.	Max.	
t_{CYC}	CLK Cycle Period	320	2000	200	2000	
t_1	CLK Low Time (Standard CLK Loading)	80	–	40	–	
t_2	CLK High Time Standard CLK Loading)	120		70		
t_r, t_f	CLK Rise and Fall Time		30	–	30	
t_{XKR}	X_1 Rising to CLK Rising	30	120	30	100	
t_{XKF}	X_1 Rising to CLK Falling		150		110	
t_{AC}	A_8–A_{15} Valid to Leading Edge of Control [1]	270	–	115	–	
t_{ACL}	$A_{\emptyset}$–A_7 Valid to Leading Edge of Control	240				
t_{AD}	$A_{\emptyset}$–A_{15} Valid to Valid Data In	–	575	–	350	
t_{AFR}	Address Float After Leading Edge of $\overline{READ}$ ($\overline{INTA}$)		0		0	
t_{AL}	A_8–A_{15} Valid Before Trailing Edge of ALE [1]	115	–	50	–	
t_{ALL}	$A_{\emptyset}$–A_7 Valid Before Trailing Edge of ALE	90				
t_{ARY}	READY Valid from Address Valid	–	220	–	100	ns
t_{CA}	Address (A_8–A_{15}) Valid After Control	120	–	60	–	
t_{CC}	Width of Control Low ($\overline{RD}$, $\overline{WR}$, $\overline{INTA}$)	400		230		
t_{CL}	Trailing Edge of Control to Leading Edge of ALE	50		25		
t_{DW}	Data Valid to Trailing Edge of $\overline{WRITE}$	420		230		
t_{HABE}	HLDA to Bus Enable	–	210	–	150	
t_{HABF}	Bus Float After HLDA					
t_{HACK}	HLDA Valid to Trailing Edge of CLK	110	–	40	–	
t_{HDH}	HOLD Hold Time	0		0		
t_{HDS}	HOLD Setup Time to Trailing Edge of CLK	170		120		
t_{INH}	INTR Hold Time	0		0		
t_{INS}	INTR, RST, and TRAP Setup Time to Falling Edge of CLK	160		150		
t_{LA}	Address Hold Time After ALE	100		50		
t_{LC}	Trailing Edge of ALE to Leading Edge of Control	130		60		
t_{LCK}	ALE Low During CLK High	100		50		

Notes see next page.

A.C. Characteristics (continued)

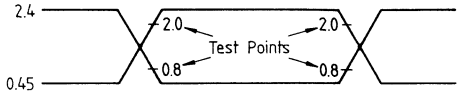
Symbol	Parameter	Limit Values				Units
		SAB 8085AH [2]		SAB 8085AH-2 [2]		
		Min.	Max.	Min.	Max.	
t_{LDR}	ALE to Valid Data During Read	–	460	–	270	
t_{LDW}	ALE to Valid Data During Write		200		120	
t_{LL}	ALE Width	140	–	80	–	
t_{LRY}	ALE to READY Stable	–	110	–	30	
t_{RAE}	Trailing Edge of $\overline{READ}$ to Re-Enabling of Address	150	–	90	–	
t_{RD}	$\overline{READ}$ (or $\overline{INTA}$) to Valid Data	–	300	–	150	
t_{RV}	Control Trailing Edge to Leading Edge of Next Control	400	–	220	–	ns
t_{RDH}	Data Hold Time After $\overline{READ}$ $\overline{INTA}$ [7]	0		0		ns
t_{RYH}	READY Hold Time					
t_{RYS}	READY Setup Time to Leading Edge of CLK	110		100		
t_{WD}	Data Valid After Trailing Edge of $\overline{WRITE}$	100		60		
t_{WDL}	LEADING Edge of WRITE to Data Valid	–	40	–	20	

NOTES

1. A_8–A_{15} address Specs apply to IO/$\overline{M}$, $S_\emptyset$, and S_1 except A_8–A_{15} are undefined during T_4–T_6 of OF cycle, whereas IO/$\overline{M}$, $S_\emptyset$, and S_1 are stable.
2. **Test conditions**: t_{CYC} = 320 ns (SAB 8085AH)/200 ns (SAB 8085AH-2); C_L = 150 pF.
3. For all output timing where C_L = 150 pF use the following correction factors:
   25 pF $\leqslant C_L <$ 150 pF: −0.10 ns/pF
   150 pF $< C_L \leqslant$ 300 pF: +0.30 ns/pF
4. Output timings are measured with purely capacitive load.
5. All timings are measured at output votage V_L = 0.8V, V_H = 2.0V, and 1.5V with 20 ns rise and fall time on inputs.
6. To calculate timing specifications at other values of t_{CYC} the following table should be used.
7. Data hold time is guaranteed under all loading conditions.

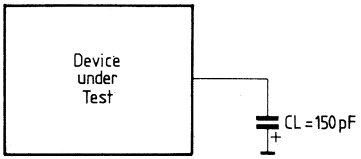
A.C. Testing

Input, Output Waveform



A.C. Testing: Input are driven at 2.4 V for a Logic "1" and 0.45 V for a Logic "0".
Timing Measurements are made at 2.0 V for a Logic "1" and 0.8 V for a Logic "0".

Load Circuit



C_L = 150 pF
C_L = includes JIG Capacitance

Bus Timing Specification as a t_{CYC} Dependent

SAB 8085AH

Symb.	Min.	Max.
t_{AL}	(1/2) T − 45	
t_{LA}	(1/2) T − 60	
t_{LL}	(1/2) T − 20	−
t_{LCK}	(1/2) T − 60	
t_{LC}	(1/2) T − 30	
t_{AD}	−	(5/2 + N) T − 225
t_{RD}		(3/2 + N) T − 180
t_{RAE}	(1/2) T − 10	
t_{CA}	(1/2) T − 40	
t_{DW}	(3/2 + N) T − 60	
t_{WD}	(1/2) T − 60	−
t_{CC}	(3/2 + N) T − 80	
t_{CL}	(1/2) T − 110	
t_{ARY}	−	(3/2) T − 260
t_{HACK}	(1/2) T − 50	−
t_{HABF}	−	(1/2) T + 50
t_{HABE}		(1/2) T + 50
t_{AC}	(2/2) T − 50	
t_1	(1/2) T − 80	−
t_2	(1/2) T − 40	
t_{RV}	(3/2) T − 80	
t_{LDR}	−	(4/2) T − 180

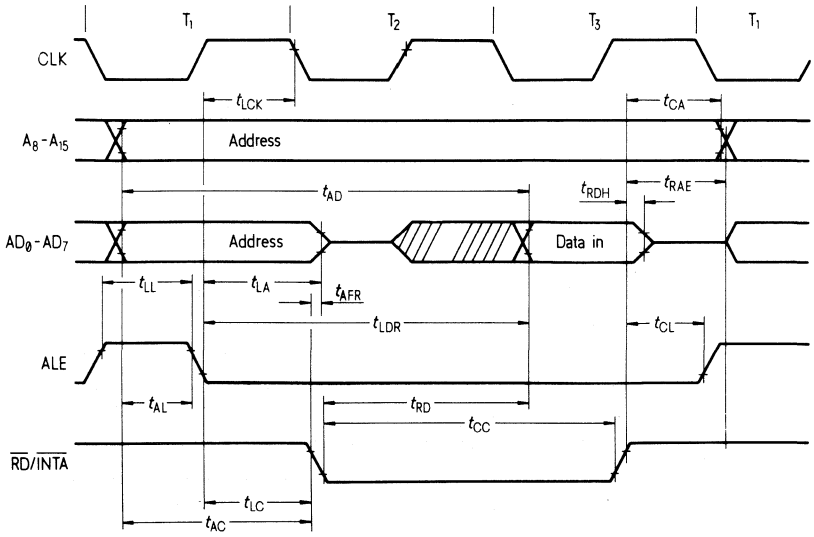
SAB 8085AH-2

Symb.	Min.	Max.
t_{AL}	(1/2) T − 50	
t_{LA}	(1/2) T − 50	
t_{LL}	(1/2) T − 20	−
t_{LCK}	(1/2) T − 50	
t_{LC}	(1/2) T − 40	
t_{AD}	−	(5/2 + N) T − 150
t_{RD}		(3/2 + N) T − 150
t_{RAE}	(1/2) T − 10	
t_{CA}	(1/2) T − 40	
t_{DW}	(3/2 + N) T − 70	−
t_{WD}	(1/2) T − 40	
t_{CC}	(3/2 + N) T − 70	
t_{CL}	(1/2) T − 75	
t_{ARY}	−	(3/2) T − 200
t_{HACK}	(1/2) T − 60	−
t_{HABF}	−	(1/2) T + 50
t_{HABE}		(1/2) T + 50
t_{AC}	(2/2) T − 85	
t_1	(1/2) T − 60	−
t_2	(1/2) T − 30	
t_{RV}	(3/2) T − 80	
t_{LDR}	−	(4/2) T − 130

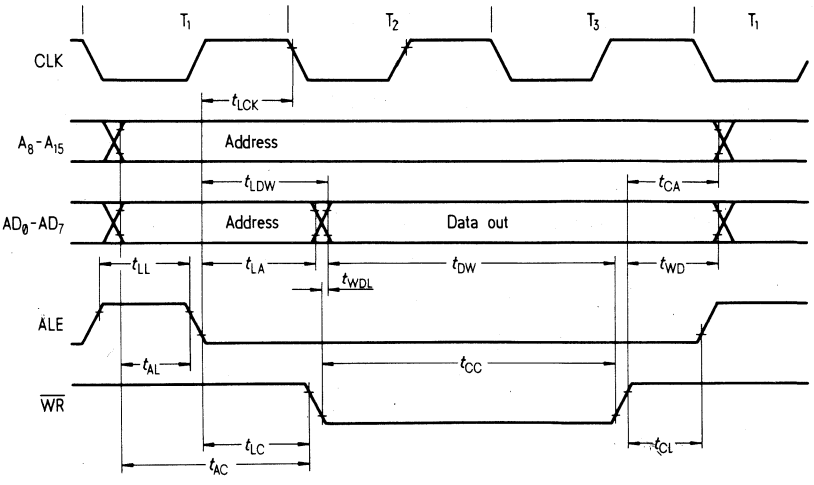
N is equal to the total WAIT states. T = t_{CYC}.

Waveforms

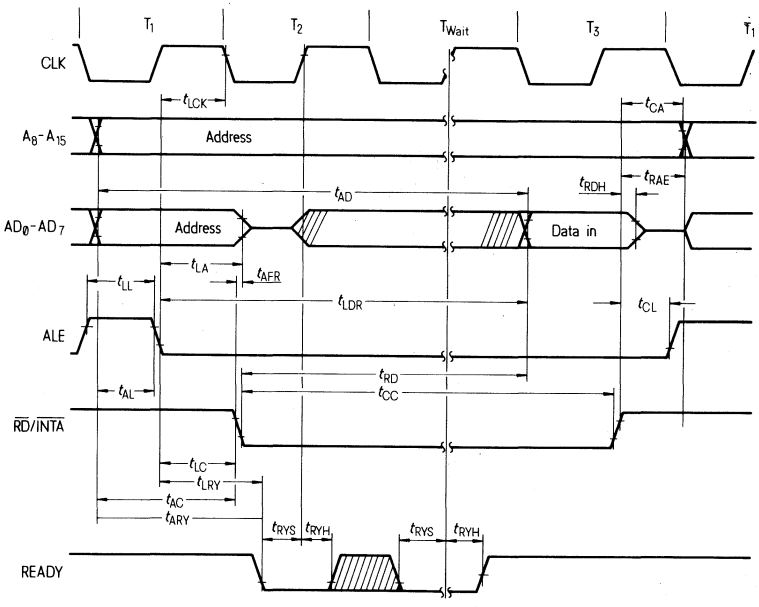
Read



Write

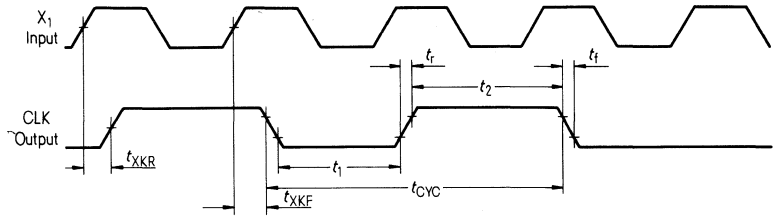


Read Operation with Cycle (Typical) – Same Ready Timing Applies to Write

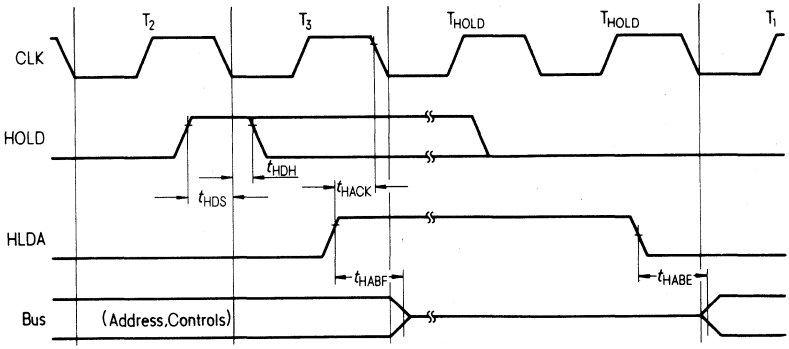


READY must remain stable during setup and hold times.

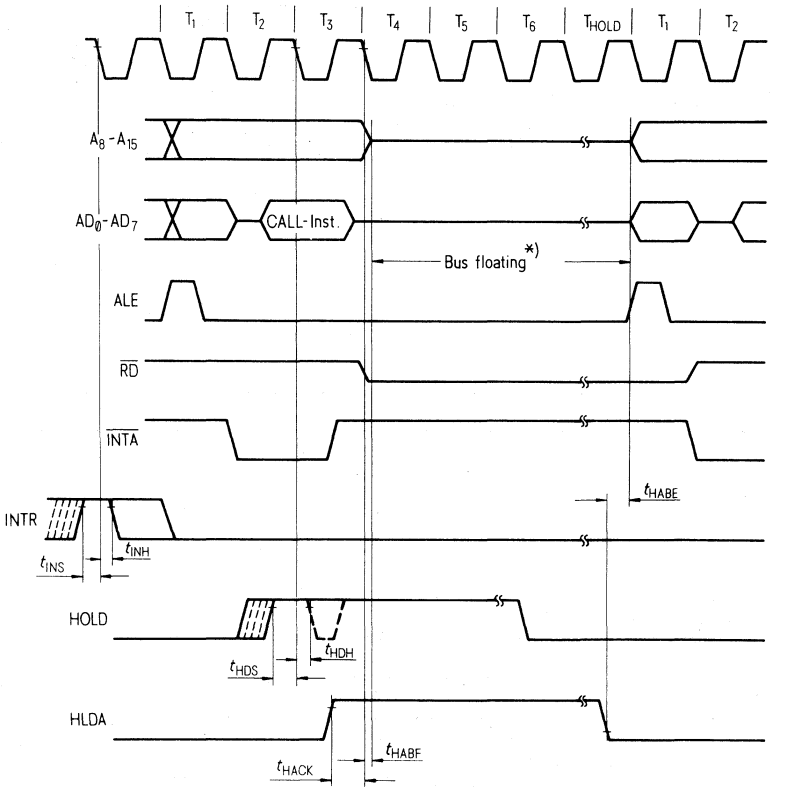
Clock


X1
Input
CLK
Output
tr
tf
t2
tXKR
t1
tXKF
tCYC

Hold


T2
T3
THOLD
THOLD
T1
CLK
HOLD
tHDH
tHDS
tHACK
HLDA
tHABF
tHABE
Bus
(Address,Controls)

Interrupt and Hold



*) IO/$\overline{M}$ is also floating during this time.

Ordering Information

Component	Description	Ordering Number
	8-Bit Microprocessor	
SAB 8085AH-P	3 MHz, 1.3 µs, (plastic)	Q 67120-C122
SAB 8085AH-2-P	5 MHz, 0.8 µs, (plastic)	Q 67120-C124

SAB 8086
16-Bit Microprocessor

SAB 8086-2 8 MHz
SAB 8086-1 10 MHz

SAB 8086 5 MHz

- Direct addressing capability to 1 Mbyte of memory
- Assembly language compatible with SAB 8080 / SAB 8085
- 14-word by 16-bit register set with symmetrical operations
- 8 and 16-bit signed and unsigned arithmetic in binary or decimal including multiply and divide
- Bit, byte, word and block operations
- 24 operand addressing modes
- Clock rates up to 10 MHz (SAB 8086-1)
- Compatible with industry standard 8086
- 40-pin plastic dual-in-line package (P-DIP-40)

Figure 1 Pin Diagram

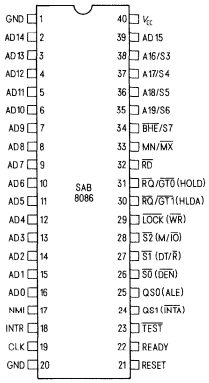


Figure 2 Pin Names

AD0–15	Address/Data	A16–19	Address
$\overline{S}$0–2	Status	S3–7	Status
INTR	Interrupt Request	$\overline{BHE}$	Bus High Enable
CLK	Clock	HOLD	Hold
QS0–1	Queue Status	HLDA	Hold Acknowledge
$\overline{TEST}$	Test for Busy	$\overline{WR}$	Write
READY	Ready	DT/$\overline{R}$	Bus Driver Transmit/ Receive
RESET	Chip Reset		
MN/$\overline{MX}$	Minimum/Maximum Mode	$\overline{DEN}$	Bus Driver Enable
		ALE	Address Latch Enable
$\overline{RD}$	Read	$\overline{INTA}$	Interrupt Acknowledge
$\overline{RQ}$/$\overline{GT}$0–1	Request/Grant	NMI	Non-Maskable Interrupt
$\overline{LOCK}$	Bus Lock		
M/$\overline{IO}$	Memory/IO	GND	Ground
		V_{CC}	+5 V

SAB 8086 is a new-generation, high-performance 16-bit microprocessor implemented in +5 V depletion load, N channel, silicon gate Siemens MYMOS technology, packaged in a 40-pin plastic dual-in-line package (P-DIP-40). It is 100 percent compatible with the industry standard 8086. With features like string handling, 16-bit arithmetic with multiply and divide it significantly increases system performance. It is highly suited for multiprocessor applications in various configurations.

Pin Definitions and Functions

The following pin definitions are for SAB 8086 systems in **either minimum or maximum mode.** The "Local Bus" in these descriptions is the direct multiplexed bus interface connection to the SAB 8086 (without regard to additional bus buffers).

<table>
<tr><th>Symbol</th><th>Pin</th><th>Input (I)
Output (O)</th><th>Function</th></tr>
<tr><td>AD0–AD15</td><td>2–16
39</td><td>I/O</td><td>ADDRESS DATA BUS
These lines constitute the time multiplexed memory I/O address (T1) and data (T2, T3, T4) bus. A0 is analogous to $\overline{BHE}$ for the lower byte of the data bus, pins D7 to D0. It is low during T1 when a byte is to be transferred on the lower portion of the bus in memory or I/O operations. Eight-bit oriented devices tied to the lower half would normally use A0 to condition chip select functions. These lines are active high and float to tristate OFF during interrupt acknowledge and local bus "hold acknowledge".</td></tr>
<tr><td>A16/S3
A17/S4
A18/S5
A19/S6</td><td>35–38</td><td>O</td><td>ADDRESS/STATUS
During T1 these are the four most significant address lines for memory operations. During I/O operations these lines are low. During memory and I/O operations, status information is available on these lines during T2, T3, TW and T4. The status of the interrupt enable flag bit (S5) is updated at the beginning of each CLK cycle.
A17/S4 and A16/S3 are encoded as follows:
<table>
<tr><th>A17/S4</th><th>A16/S3</th><th>Characteristics</th></tr>
<tr><td>0 (low)</td><td>0</td><td>Alternate Data</td></tr>
<tr><td>0</td><td>1</td><td>Stack</td></tr>
<tr><td>1 (high)</td><td>0</td><td>Code or None</td></tr>
<tr><td>1</td><td>1</td><td>Data</td></tr>
<tr><td>S6 is 0 (low)</td><td></td><td></td></tr>
</table>
This information indicates which relocation register is presently being used for data accessing.
These lines float to tristate OFF during local bus "hold acknowledge".</td></tr>
<tr><td>$\overline{BHE}$/S7</td><td>34</td><td>O</td><td>BUS HIGH ENABLE/STATUS
During T1 the bus high enable signal ($\overline{BHE}$) should be used to enable data onto the most significant half of the data bus, pins D15 to D8. Eight-bit oriented devices tied to the upper half of the bus would normally use $\overline{BHE}$ to condition chip select functions. $\overline{BHE}$ is low during T1 for read, write, and interrupt acknowledge cycles when a byte is to be transferred on the high portion of the bus. The S7 status information is available during T2, T3, and T4. The signal is active low, and floats to tristate OFF in "hold". It is low during T1 for the first interrupt acknowledge cycle.</td></tr>
<tr><td>$\overline{RD}$</td><td>32</td><td>O</td><td>READ strobe indicates that the processor is performing a memory or I/O read cycle, depending on the state of the S2 pin. This signal is used to read devices which reside on the SAB 8086 local bus. $\overline{RD}$ is active low during T2, T3 and TW of any read cycle, and is guaranteed to remain high in T2 until the SAB 8086 local bus has floated. This signal floats to tristate OFF in "hold acknowledge".</td></tr>
</table>

Pin Definitions and Functions (cont'd)

Symbol	Pin	Input (I) Output (O)	Function
READY	22	I	**READY** is the acknowledgement from the addressed memory or I/O device that it will complete the data transfer. The RDY signal from memory I/O is synchronized by the SAB 8284B clock generator to form READY. This signal is active high. The SAB 8086 READY input is not synchronized. Correct operation is not guaranteed if the setup and hold times are not met.
INTR	18	I	**INTERRUPT REQUEST** is a level triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. A subroutine is vectored to via an interrupt vector lookup table located in system memory. It can be internally masked by software reseting the interrupt enable bit. INTR is internally synchronized. This signal is active high.
$\overline{\text{TEST}}$	23	I	The **$\overline{\text{TEST}}$** input is examined by the "wait" instruction. If this input is low execution continues, otherwise the processor waits in an "idle" state. This input is synchronized internally during each clock cycle on the leading edge of CLK.
NMI	17	I	**NON-MASKABLE INTERRUPT** is an edge triggered input which causes a type 2 interrupt. A subroutine is vectored to via interrupt vector lookup table located in system memory. NMI is not maskable internally by software. A transition from a low to high initiates the interrupt at the end of the current instruction. This input is internally synchronized.
RESET	21	I	**RESET** causes the processor to immediately terminate its present activity. The signal must be active high for at least four clock cycles. It restarts execution, as described in the Instruction Set Description, when RESET returns low. RESET is internally synchronized.
CLK	19	I	The **CLOCK** provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing.
MN/$\overline{\text{MX}}$	33	I	**MINIMUM/$\overline{\text{MAXIMUM}}$**: indicates which mode the processor is to operate in. The two modes are discussed in the following sections.
V_{CC}	40		**POWER SUPPLY** (+5 V)
GND	1, 20		**GROUND** (0 V)

Pin Definitions and Functions (cont'd)

The following pin definitions are for the SAB 8086/8288 system in **maximum mode** (i. e. MN/$\overline{MX}$ = GND). Only the pin functions which are unique to maximum mode are described; all other pin functions are as already described.

<table>
<tr><th>Symbol</th><th>Pin</th><th>Input (I)
Output (O)</th><th>Function</th></tr>
<tr><td>$\overline{S2}$, $\overline{S1}$, $\overline{S0}$</td><td>26–28</td><td>O</td><td>These STATUS lines are encoded as follows:
<table>
<tr><th>$\overline{S2}$</th><th>$\overline{S1}$</th><th>$\overline{S0}$</th><th>Characteristics</th></tr>
<tr><td>0 (low)</td><td>0</td><td>0</td><td>Interrupt Acknowledge</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Read I/O Port</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Write I/O Port</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Halt</td></tr>
<tr><td>1 (high)</td><td>0</td><td>0</td><td>Code Access</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Read Memory</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Write Memory</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Passive</td></tr>
</table>
Status is active during T4, T1, and T2 and is returned to the passive state (1,1,1) during T3 or during TW when READY is high. This status is used by the SAB 8288A bus controller to generate all memory and I/O access control signals. Any change by $\overline{S2}$, $\overline{S1}$, or $\overline{S0}$ during T4 is used to indicate the beginning of a bus cycle, and the return to the passive state in T3 or TW is used to indicate the end of a bus cycle.
These signals float to tristate OFF in "hold acknowledge".</td></tr>
<tr><td>$\overline{RQ}$/$\overline{GT0}$, $\overline{RQ}$/$\overline{GT1}$</td><td>30–31</td><td>I/O</td><td>The $\overline{REQUEST}$/$\overline{GRANT}$ pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bidirectional with $\overline{RQ}$/$\overline{GT0}$ having higher priority than $\overline{RQ}$/$\overline{GT1}$. $\overline{RQ}$/$\overline{GT}$ has an internal pullup resistor so may be left unconnected. The request/grant sequence is as follows (see figure 14):
1. A pulse of 1 CLK wide from another local bus master indicates a local bus request ("hold") to the SAB 8086 (pulse1).
2. During the CPU's next T4 or T1 a pulse 1 CLK wide from the SAB 8086 to the requesting master (pulse 2) indicates that the SAB 8086 has allowed the local bus to float and that it will enter the "hold acknowledge" state at the next CLK. The CPU's bus interface unit is disconnected logically from the local bus during "hold acknowledge".
3. A pulse 1 CLK wide from the requesting master indicates to the SAB 8086 (pulse 3) that the "hold" request is about to end and that the SAB 8086 can reclaim the local bus at the next CLK.
Each master-master exchange of the local bus is a sequence of 3 pulses. There must be one dead CLK cycle after each bus exchange. Pulses are active low.
If the request is made while the CPU is performing a memory cycle, it will release the local bus during T4 of the cycle when all the following conditions are met:
1. Request occurs on or before T2.
2. Current cycle is not the low byte of a word (on an odd address).
3. Current cycle is not the first acknowledge of an interrupt acknowledge sequence.
4. A locked instruction is not currently executing.</td></tr>
</table>

Pin Definitions and Functions (cont'd)

<table>
<tr><th>Symbol</th><th>Pin</th><th>Input (I)
Output (O)</th><th>Function</th></tr>
<tr><td>$\overline{\text{LOCK}}$</td><td>29</td><td>O</td><td>The $\overline{\text{LOCK}}$ output indicates that other system bus masters are not to gain control of the system bus while $\overline{\text{LOCK}}$ is active low. The $\overline{\text{LOCK}}$ signal is activated by the "LOCK" prefix instruction and remains active until the completion of the next instruction. This signal is active low, and floats to tristate OFF in "hold acknowledge".</td></tr>
<tr><td>QS1, QS0</td><td>24–25</td><td>O</td><td>The QUEUE STATUS is valid during the CLK cycle after which the queue operation is performed.
QS1 and QS0 provide status to allow external tracking of the internal SAB 8086 instruction queue.
<table>
<tr><th>QS1</th><th>QS0</th><th>Characteristics</th></tr>
<tr><td>0 (low)</td><td>0</td><td>No Operation</td></tr>
<tr><td>0</td><td>1</td><td>First Byte of Op Code from Queue</td></tr>
<tr><td>1 (high)</td><td>0</td><td>Empty the Queue</td></tr>
<tr><td>1</td><td>1</td><td>Subsequent Byte from Queue</td></tr>
</table></td></tr>
</table>

Pin Definitions and Functions (cont'd)

The following pin definitions are for the SAB 8086 **minimum mode** (i. e. MN/$\overline{MX}$ = V_{CC}). Only the pin functions which are unique to minimum mode are described; all other pin functions are as described before.

Symbol	Pin	Input (I) Output (O)	Function
M/$\overline{IO}$	28	O	This **STATUS LINE** is logically equivalent to S 2 in the maximum mode. It is used to distinguish a memory access from an I/O access. M/$\overline{IO}$ becomes valid in the T4 preceding a bus cycle and remains valid until the final T4 of the cycle (M = high, $\overline{IO}$ = low). M/$\overline{IO}$ floats to tristate OFF in local bus "hold acknowledge".
$\overline{WR}$	29	O	**$\overline{WRITE}$** strobe indicates that the processor is performing a write memory or write I/O cycle, depending on the state of the M/$\overline{IO}$ signal. $\overline{WR}$ is active for T2, T3 and TW of any write cycle. It is active low, and floats to tristate OFF in local bus "hold acknowledge".
$\overline{INTA}$	24	O	**$\overline{INTA}$** is used as a read strobe for interrupt acknowledge cycles. It is active low during T2, T3 and TW of each interrupt acknowledge cycle.
ALE	25	O	**ADDRESS LATCH ENABLE** is provided by the processor to latch the address into the SAB 8282A/SAB 8283A address latch. It is a high pulse active during T1 of any bus cycle. Note that ALE is never floated.
DT/$\overline{R}$	27	O	**DATA TRANSMIT/$\overline{RECEIVE}$** is needed in minimum system that desires to use a SAB 8286A/SAB 8287A data bus transceiver. It is used to control the direction of data flow through the transceiver. Logically DT/$\overline{R}$ is equivalent to $\overline{S1}$ in the maximum mode, and its timing is the same as for M/$\overline{IO}$. (T=high, $\overline{R}$=low). This signal floats to tristate OFF in local bus "hold acknowledge".
$\overline{DEN}$	26	O	**$\overline{DATA\ ENABLE}$** is provided as an output enable for the SAB 8286A/SAB 8287A in a minimum system which uses the transceiver. $\overline{DEN}$ is active low during each memory and I/O access and for $\overline{INTA}$ cycles. For a read or $\overline{INTA}$ cycle it is active from the middle of T2 until the middle of T4, while for a write cycle it is active from the beginning of T2 until the middle of T4. $\overline{DEN}$ floats to tristate OFF in local bus "hold acknowledge".
HOLD HLDA	30–31	I O	**HOLD** indicates that another master is requesting a local bus "hold". To be acknowledged, HOLD must be active high. The processor receiving the "hold" request will issue **HLDA** (high) as an acknowledgement in the middle of T4 or T1. Simultaneous with the issuance of HLDA the processor will float the local bus and control lines. After HOLD is detected as being low, the processor will lower HLDA, and when the processor needs to run another cycle, it will again drive the local bus and control lines. HOLD is not an asynchronous input. External synchronization should be provided if the system cannot otherwise guarantee the setup time. The same rules as for $\overline{RQ}$/$\overline{GT}$ apply regarding when the local bus will be released.

Figure 3 Functional Block Diagram

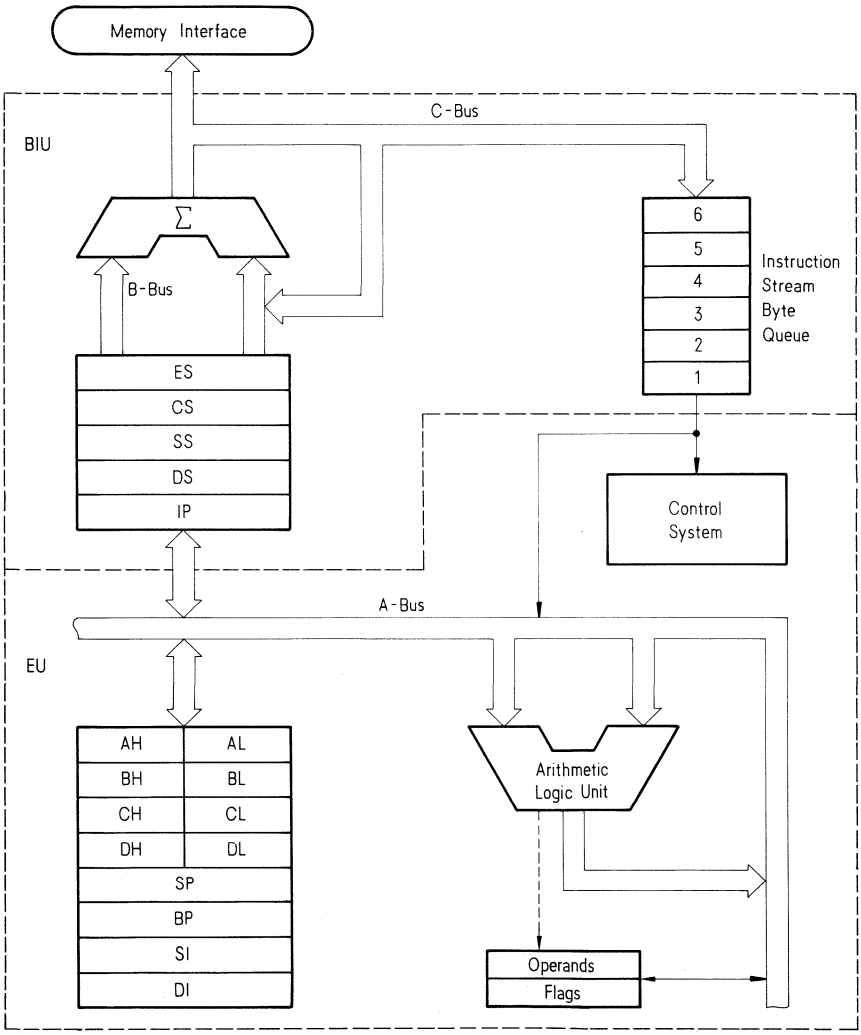
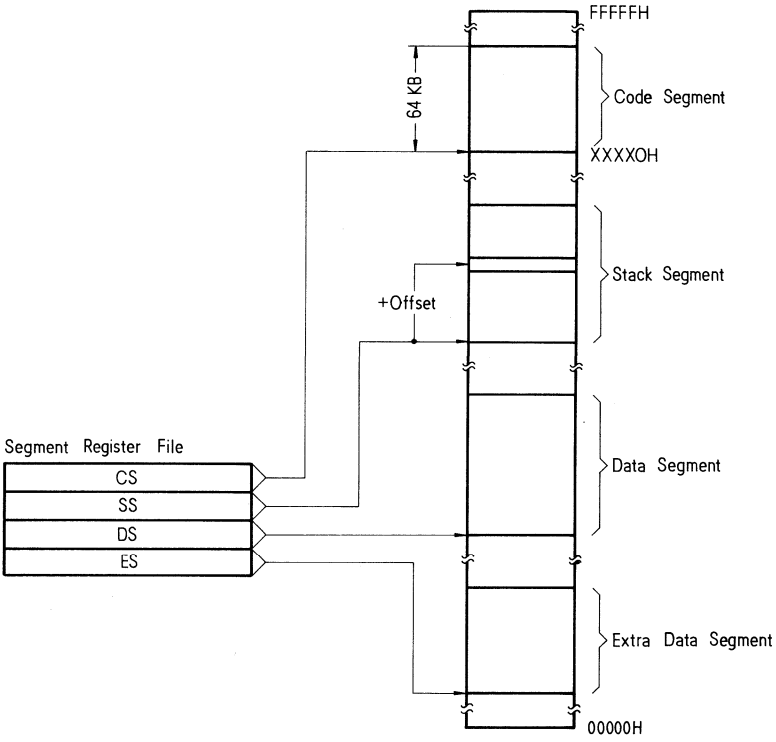


Figure 4 Memory Organization



Functional Description

The internal functions of the SAB 8086 processor are partitioned logically into two processing units. The first is the Bus Interface Unit (BIU) and the second is the Execution Unit (EU) as shown in the block diagram of figure 3.

The bus interface unit provides the functions related to instruction fetching and queuing, operand fetch and store, and address relocation. The overlap of instruction pre-fetching provided by this unit serves to increase processor performance through improved bus bandwidth utilization. Up to 6 bytes of the instruction stream can be queued while waiting for decoding and execution.

The instruction stream queuing mechanism allows the BIU to keep the memory utilized very efficiently. Whenever there is space for at least 2 bytes in the queue, the BIU will attempt a word fetch memory cycle. This greatly reduces „dead time" on the memory bus.

The execution unit receives pre-fetched instructions from the BIU queue and provides un-relocated operand addresses to the BIU. Memory operands are passed through the BIU for processing by the EU, which passes results to the BIU for storage.

The processor provides a 20-bit address to memory which locates the byte being referenced. The memory is logically organized as a linear array of 1 million bytes, addressed as 00000(H) to FFFFF(H). The memory can further be logically divided into code, data, alternate data, and stack segments of up to 64 Kbytes each, with each segment falling on 16-byte boundaries (see figure 4).

Minimum and Maximum Modes

The requirements for supporting minimum and maximum mode in SAB 8086 systems are sufficiently different that they cannot be met efficiently with 40 uniquely defined pins. Consequently, the SAB 8086 is equipped with a strap pin (MN/$\overline{MX}$) which defines the system configuration.

The definition of a certain subset of the pins changes dependent on the condition of the strap pin.

When MN/$\overline{MX}$ pin is strapped to GND, the SAB 8086 treats pins 24 through 31 in maximum mode. An SAB 8288A bus controller interprets status information coded into $\overline{S0}$, $\overline{S1}$, $\overline{S2}$ to generate bus timing and control signals.

When the MN/$\overline{MX}$ pin is strapped to V_{CC}, the SAB 8086 generates bus control signals itself on pins 24 through 31, as shown in parentheses in figure 1.

Bus Operation

The SAB 8086 has a combined address and data bus commonly referred to as a time multiplexed bus.

Each processor bus cycle consists of at least four CLK cycles. These are referred to as T1, T2, T3 and T4 (see figure 5). The address is emitted from the processor during T1 and data transfer occurs on the bus during T3 and T4. T2 is used primarily for changing the direction of the bus during read operations. In the event that a "NOT READY" indication is given by the addressed device, "wait" states (TW) are inserted between T3 and T4. Each inserted wait state is of the same duration as a CLK cycle. Periods can occur between SAB 8086 bus cycles. These are referred to as "idle" states (Ti) or inactive CLK cycles. The processor uses these cycles for internal housekeeping.

During T1 of any bus cycle the ALE (Address Latch Enable) signal is emitted (by either the processor or the SAB 8288A bus controller, depending on the MN/$\overline{MX}$ strap). At the trailing edge of this pulse, a valid address and certain status information for the cycle may be latched.

Status bits $\overline{S0}$, $\overline{S1}$, and $\overline{S2}$ are used, in maximum mode, by the bus controller to identify the type of bus transaction according to the following table:

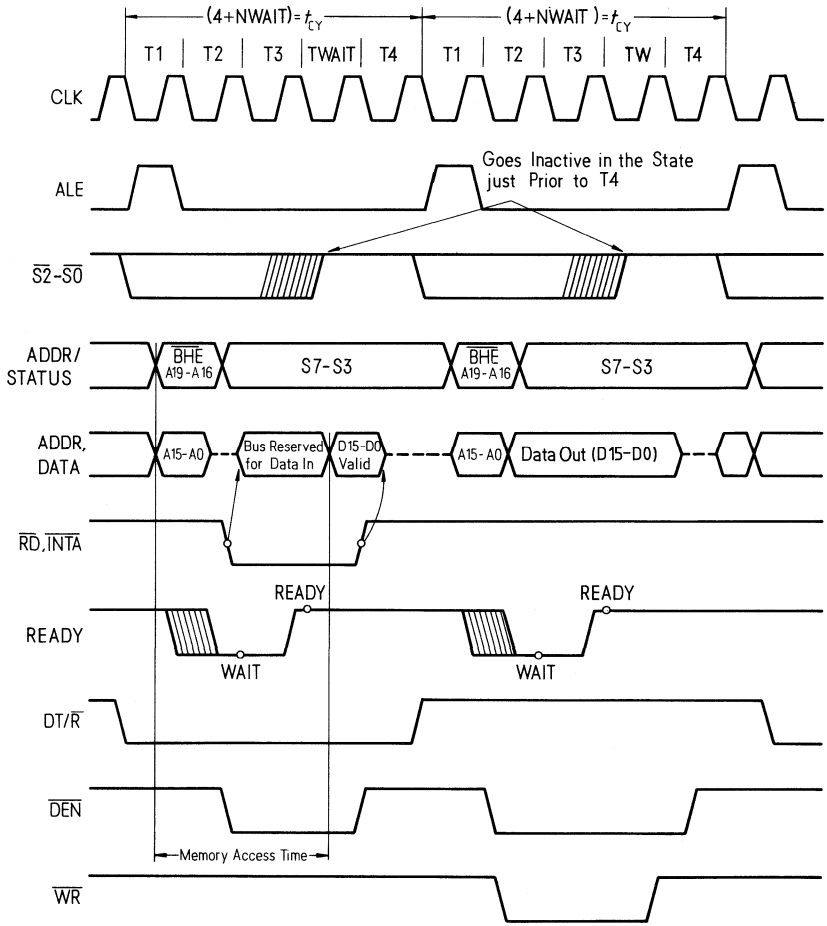
$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	Characteristics
0 (Low)	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt
1 (High)	0	0	Instruction Fetch
1	0	1	Read Data from Memory
1	1	0	Write Data to Memory
1	1	1	Passive (no bus cycle)

Status bits S3 through S7 are multiplexed with high-order address bits and the $\overline{BHE}$ signal, and are therefore valid during T2 through T4. S3 and S4 indicate which segment register (see Instruction Set Summary) was used for this bus cycle in forming the address, according to the following table:

S4	S3	Characteristics
0 (Low)	0	Alternate Data (extra segment)
0	1	Stack
1 (High)	0	Code or None
1	1	Data

S5 is a reflection of the PSW interrupt enable bit. S6 = 0 and S7 is a spare status bit.

Figure 5 Basic System Timing



I/O Addressing

In the SAB 8086, I/O operations can address up to a maximum of 64 K I/O byte registers or 32 K I/O word registers.
The I/O address appears in the same format as the memory address on bus lines A15 to A0. The address lines A19 to A16 are zero in I/O operations.
The variable I/O instructions which use register DX as a pointer have full address capability while the direct I/O instructions directly address one or two of the 256 I/O byte locations in page 0 of the I/O address space.

System Components

SAB 8282A	Octal Latch
SAB 8283A	Octal Latch (inverting)
SAB 8284B	Clock Generator and Driver
SAB 8286A	Octal Bus Transceiver
SAB 8287A	Octal Bus Transceiver (inverting)
SAB 8288A	Bus Controller
SAB 8289	Bus Arbiter
SAB 8259A	Programmable Interrupt Controller

Typical Applications

SAB 8086 is a general-purpose 16-bit microprocessor which can be used for applications ranging from process control to data processing. Figures 6 and 7 show typical system configurations for SAB 8086 familiy components.

Figure 6 Minimum Mode SAB 8086 Typical System Configuration

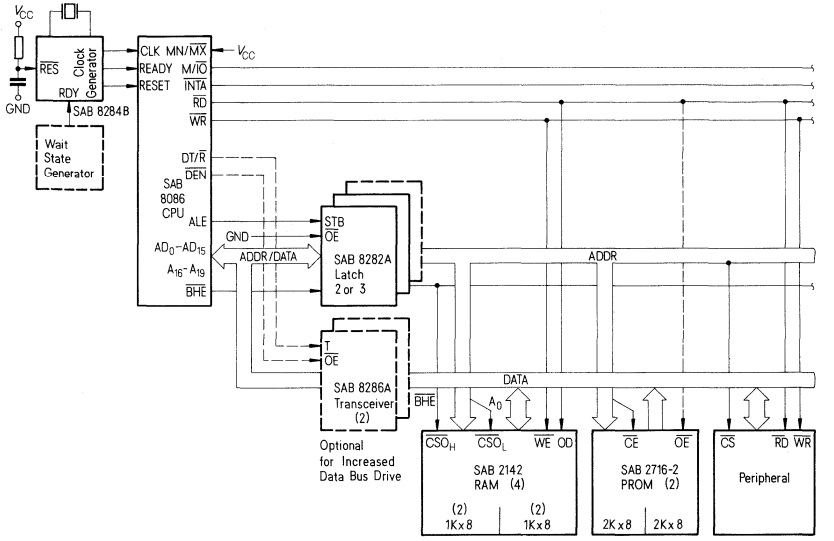
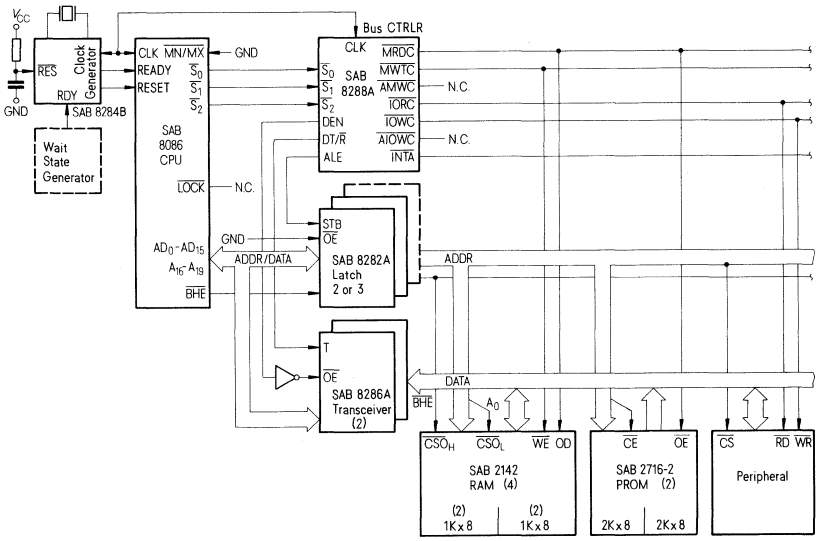


Figure 7 Maximum Mode SAB 8086 Typical System Configuration



Instruction Set Summary

Data Transfer

MOV = Move:	**76543210**	**76543210**	**76543210**	**76543210**
Register / memory to / from register	100010dw	mod reg r/m		
Immediate to register/memory	1100011w	mod 000 r/m	data	data if w=1
Immediate to register	1011w reg	data	data if w=1	
Memory to accumulator	1010000w	addr-low	addr-high	
Accumulator to memory	1010001w	addr-low	addr-high	
Register/memory to segment register	10001110	mod 0 reg r/m		
Segment register to register/memory	10001100	mod 0 reg r/m		
PUSH = Push:				
Register/memory	11111111	mod 110 r/m		
Register	01010 reg			
Segment register	000 reg 110			
POP = Pop:				
Register/memory	10001111	mod 000 r/m		
Register	01011 reg			
Segment register	000 reg 111			
XCHG = Exchange:				
Register/memory with register	1000011w	mod reg r/m		
Register with accumulator	10010 reg			
IN = Input from:				
Fixed port	1110010w	port		
Variable port	1110110w			

OUT = Output to:	76543210	76543210	76543210	76543210
Fixed port	1110011w	port		
Variable port	1110111w			
XLAT = Translate byte to AL	11010111			
LEA = Load EA to register	10001101	mod reg r/m		
LDS = Load pointer to DS	11000101	mod reg r/m		
LES = Load pointer to ES	11000100	mod reg r/m		
LAHF = Load AH with flags	10011111			
SAHF = Store AH into flags	10011110			
PUSHF = Push flags	10011100			
POPF =Pop flags	10011101			

Arithmetic
ADD = Add:

Reg./memory with register to either	000000dw	mod reg r/m		
Immediate to register/memory	100000sw	mod 000 r/m	data	data if s: w=01
Immediate to accumulator	0000010w	data	data if w=1	

ADC = Add with carry:

Reg./memory with register to either	000100dw	mod reg r/m		
Immediate to register/memory	100000sw	mod 010 r/m	data	data if s:w=01
Immediate to accumulator	0001010w	data	data if w=1	

INC = Increment:

Register/memory	1111111w	mod 000 r/m
Register	01000reg	
AAA = ASCII adjust for add	00110111	
DAA = Decimal adjust for add	00100111	

SUB = Subtract:	76543210	76543210	76543210	76543210
Reg./memory and register to either	0 0 1 0 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 1 0 1 r/m	data	data if s:w=01
Immediate from accumulator	0 0 1 0 1 1 0 w	data	data if w=1	

SBB = Subtract with borrow:				
Reg./memory and register to either	0 0 0 1 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 0 1 1 r/m	data	data if s:w=01
Immediate from accumulator	0 0 0 1 1 1 0 w	data	data if w=1	

DEC = Decrement:	76543210	76543210	76543210	76543210
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 1 r/m		
Register	0 1 0 0 1 reg			
NEG = Change sign	1 1 1 1 0 1 1 w	mod 0 1 1 r/m		

CMP = Compare:				
Register/memory and register	0 0 1 1 1 0 d w	mod reg r/m		
Immediate with register/memory	1 0 0 0 0 0 s w	mod 1 1 1 r/m	data	data if s:w=01
Immediate with accumulator	0 0 1 1 1 1 0 w	data	data if w=1	
AAS = ASCII adjust for subtract	0 0 1 1 1 1 1 1			
DAS = Decimal adjust for subtract	0 0 1 0 1 1 1 1			
MUL = Multiply (unsigned)	1 1 1 1 0 1 1 w	mod 1 0 0 r/m		
IMUL = Integer multiply (signed)	1 1 1 1 0 1 1 w	mod 1 0 1 r/m		
AAM = ASCII adjust for multiply	1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0		
DIV = Divide (unsigned)	1 1 1 1 0 1 1 w	mod 1 1 0 r/m		
IDIV = Integer divide (signed)	1 1 1 1 0 1 1 w	mod 1 1 1 r/m		
AAD = ASCII adjust for divide	1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0		
CBW = Convert byte to word	1 0 0 1 1 0 0 0			
CWD = Convert word to double word	1 0 0 1 1 0 0 1			

Logic	76543210	76543210	76543210	76543210
NOT = Invert	1 1 1 1 0 1 1 w	mod 0 1 0 r/m		
SHL/SAL = Shift logical/arithmetic left	1 1 0 1 0 0 v w	mod 1 0 0 r/m		
SHR = Shift logical right	1 1 0 1 0 0 v w	mod 1 0 1 r/m		
SAR = Shift arithmetic right	1 1 0 1 0 0 v w	mod 1 1 1 r/m		
ROL = Rotate left	1 1 0 1 0 0 v w	mod 0 0 0 r/m		
ROR = Rotate right	1 1 0 1 0 0 v w	mod 0 0 1 r/m		
RCL = Rotate through carry flag left	1 1 0 1 0 0 v w	mod 0 1 0 r/m		
RCR = Rotate through carry flag right	1 1 0 1 0 0 v w	mod 0 1 1 r/m		
AND = And:				
Reg./memory and register to either	0 0 1 0 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 0 0 r/m	data	data if w=1
Immediate to accumulator	0 0 1 0 0 1 0 w	data	data if w=1	
TEST = And function to flags, no result:				
Register/memory and register	1 0 0 0 0 1 0 w	mod reg r/m		
Immediate data and register/memory	1 1 1 1 0 1 1 w	mod 0 0 0 r/m	data	data if w=1
Immediate data and accumulator	1 0 1 0 1 0 0 w	data	data if w=1	
OR = Or:				
Reg./memory and register to either	0 0 0 0 1 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 0 0 1 r/m	data	data if w=1
Immediate to accumulator	0 0 0 0 1 1 0 w	data	data if w=1	
XOR = Exclusive Or:				
Reg./memory and register to either	0 0 1 1 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 1 0 r/m	data	data if w=1
Immediate to accumulator	0 0 1 1 0 1 0 w	data	data if w=1	

String Manipulation	**76543210**	**76543210**	**76543210**
REP = Repeat	1111001z		
MOVS = Move byte/word	1010010w		
CMPS = Compare byte/word	1010011w		
SCAS = Scan byte/word	1010111w		
LODS = Load byte/word to AL/AX	1010110w		
STOS = Store byte/word from AL/A	1010101w		

Control Transfer

CALL = Call:			
Direct within segment	11101000	disp-low	disp-high
Indirect within segment	11111111	mod 010 r/m	
Direct intersegment	10011010	offset-low	offset-high
		seg-low	seg-high
Indirect intersegment	11111111	mod 011 r/m	

JMP = Unconditional jump:			
Direct within segment	11101001	disp-low	disp-high
Direct within segment short	11101011	disp	
Indirect within segment	11111111	mod 100 r/m	
Direct intersegment	11101010	offset-low	offset-high
		seg-low	seg-high
Indirect intersegment	11111111	mod 101 r/m	

RET = Return from CALL:	**76543210**	**76543210**	**76543210**
Within segment	11000011		
Within seg. adding immediate to SP	11000010	data-low	data-high
Intersegment	11001011		
Intersegment adding immediate to SP	11001010	data-low	data-high
JE/JZ = Jump on equal/zero	01110100	disp	
JL/JNGE = Jump on less/not greater or equal	01111100	disp	
JLE/JNG = Jump on less or equal/not greater	01111110	disp	
JB/JNAE = Jump on below/not above or equal	01110010	disp	
JBE/JNA = Jump on below or equal/ not above	01110110	disp	
JP/JPE = Jump on parity/parity even	01111010	disp	
JO = Jump on overflow	01110000	disp	
JS = Jump on sign	01111000	disp	
JNE/JNZ = Jump on not equal/not zero	01110101	disp	
JNL/JGE = Jump on not less/greater or equal	01111101	disp	
JNLE/JG = Jump on not less or equal/ greater	01111111	disp	
JNB/JAE = Jump on not below/above or equal	01110011	disp	
JNBE/JA = Jump on not below or equal/above	01110111	disp	
JNP/JPO = Jump on not parity/parity odd	01111011	disp	
JNO = Jump on not overflow	01110001	disp	
JNS = Jump on not sign	01111001	disp	
LOOP = Loop CX times	11100010	disp	
LOOPZ/LOOPE = Loop while zero/equal	11100001	disp	
LOOPNZ/LOOPNE = Loop while not zero/equal	11100000	disp	
JCXZ = Jump on CX zero	11100011	disp	

INT = Interrupt	**76543210**	**76543210**
Type specified	11001101	type
Type 3	11001100	
INTO = Interrupt on overflow	11001110	
IRET = Interrupt return	11001111	

Processor Control

CLC = Clear carry	11111000	
CMC = Complement carry	11110101	
STC = Set carry	11111001	
CLD = Clear direction	11111100	
STD = Set direction	11111101	
CLI = Clear interrupt	11111010	
STI = Set interrupt	11111011	
HLT = Halt	11110100	
WAIT = Wait	10011011	
ESC = Escape (to external device)	11011xxx	mod x x x r/m
LOCK = Bus lock prefix	11110000	

Notes:

AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value.
Greater = more positive;
Less = less positive (more negative) signed values
if d = 1 then "to" reg; if d = 0 then "from" reg
if w = 1 then word instruction; if w = 0 then byte instruction
if s:w = 01 then 16-bits of immediate data from the operand
if s:w = 11 then an immediate data byte is sign-extended to form the 16-bit operand
if v = 0 then "count" = 1; if v = 1 then "count" in (CL)
x = don't care
z is used for string primitives for comparsion with ZF FLAG

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp high is absent
if mod = 10 then DISP = disp-high: disp low

if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) +DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)

* except if mod = 00 and r/m = 110 then EA = disp-high:disp-low.

Segment Override Prefix

0 0 1 reg 1 1 0

REG is assigned according to the following table

16-bit (w=1)	8-bit (w=0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instruction which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:

FLAGS = X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

Absolute Maximum Ratings

Ambient temperature under bias	0 to 70°C
Storage temperature	−65 to +150°C
Voltage on any pin with respect to ground	−1.0 to +7V
Power dissipation	2.5 W

Note:
Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

SAB 8086: T_A = 0 to 70°C, V_{CC} = 5 V ±10%
SAB 8086-1/8086-2: T_A = 0 to 70°C, V_{CC} = 5 V ± 5%

Parameter	Symbol	Limit values		Unit	Test conditions
		min.	max.		
Input low voltage	V_{IL}	−0.5	+0.8	V	–
Input high voltage	V_{IH}	2.0	V_{CC}+0.5	V	–
Output low voltage	V_{OL}	–	0.45	V	I_{OL} = 2.5 mA
Output high voltage	V_{OH}	2.4	–	V	I_{OH} = −400 µA
Power supply current SAB 8086 SAB 8086-2 SAB 8086-1	I_{CC}	 – – –	 340 350 360	 mA mA mA	All outputs open T_A = 25°C
Input leakage current	I_{LI}	–	±10	µA	0V ≤ V_{IN} ≤ V_{CC}
Output leakage current	I_{LO}	–	±10	µA	0.45 V ≤ V_{OUT} ≤ V_{CC}
Clock input low voltage	V_{CL}	−0.5	+0.6	V	–
Clock input high voltage	V_{CH}	3.9	V_{CC}+1.0	V	–
Capacitance of input buffer (all inputs except AD0 to AD15, $\overline{RQ}/\overline{GT}$)	C_{IN}	–	15	pF	f_c = 1 MHz
Capacitance of I/O buffer (AD0 to AD15, $\overline{RQ}/\overline{GT}$)	C_{IO}	–	15	pF	f_c = 1 MHz

AC Characteristics for SAB 8086/8086-2

SAB 8086: T_A = 0 to 70°C, V_{CC} = 5 V ± 10%
SAB 8086-2: T_A = 0 to 70°C, V_{CC} = 5 V ± 5%

Minimum Complexity System (figures 8, 9, 12, 15)
Timing Requirements

Parameter	Symbol	Limit values				Unit	Test conditions
		SAB 8086		SAB 8086-2			
		min.	max.	min.	max.		
CLK cycle period SAB 8086	t_{CLCL}	200	500	125	500	ns	–
CLK low time	t_{CLCH}	118	–	68	–	ns	–
CLK high time	t_{CHCL}	69	–	44	–	ns	–
CLK rise time	t_{CH1CH2}	–	10	–	10	ns	from 1.0 to 3.5V
CLK fall time	t_{CL2CL1}	–	10	–	10	ns	from 3.5 to 1.0V
Data in setup time	t_{DVCL}	30	–	20	–	ns	–
Data in hold time	t_{CLDX}	10	–	10	–	ns	–
RDY setup time into SAB 8284A [1] [2]	t_{R1VCL}	35	–	35	–	ns	–
RDY hold time into SAB 8284A [1] [2]	t_{CLR1X}	0	–	0	–	ns	–
READY setup time into SAB 8086	t_{RYHCH}	118	–	68	–	ns	–
READY hold time into SAB 8086	t_{CHRYX}	30	–	20	–	ns	–
READY inactive to CLK [3]	t_{RYLCL}	−8	–	−8	–	ns	–
HOLD setup time	t_{HVCH}	35	–	20	–	ns	–
INTR, NMI, $\overline{\text{TEST}}$ setup time [2]	t_{INVCH}	30	–	15	–	ns	–
Input rise time (except CLK)	t_{ILIH}	–	20	–	20	ns	from 0.8 to 2.0V
Input fall time (except CLK)	t_{IHIL}	–	12	–	12	ns	from 2.0 to 0.8V

[1]) Signal at SAB 8284B shown for reference only.
[2]) Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
[3]) Applies only to T2 state (8 ns into T3).

Timing Responses

Parameter	Symbol	Limit values				Unit	Test conditions
		SAB 8086		SAB 8086-2			
		min.	max.	min.	max.		
Address valid delay	t_{CLAV}	10	110	10	60	ns	
Address hold time	t_{CLAX}	10	–	10	–	ns	
Address float delay	t_{CLAZ}	t_{CLAX}	80	t_{CLAX}	50	ns	
ALE width	t_{LHLL}	$t_{CLCH}-20$	–	$t_{CLCH}-10$	–	ns	
ALE active delay	t_{CLLH}	–	80	–	50	ns	
ALE inactive delay	t_{CHLL}	–	85	–	55	ns	
Address hold time to ALE inactive	t_{LLAX}	$t_{CHCL}-10$	–	$t_{CHCL}-10$	–	ns	
Data valid delay	t_{CLDV}	10	110	10	60	ns	C_L = 20 to 100 pF for all SAB 8086 outputs (in addition to SAB 8086 self-load)
Data hold time	t_{CHDX}	10	–	10	–	ns	
Data hold time after WR	t_{WHDX}	$t_{CLCH}-30$	–	$t_{CLCH}-30$	–	ns	
Control active delay 1	t_{CVCTV}	10	110	10	70	ns	
Control active delay 2	t_{CHCTV}	10	110	10	60	ns	
Control inactive delay	t_{CVCTX}	10	110	10	70	ns	
Address float to READ active	t_{AZRL}	0	–	0	–	ns	
$\overline{RD}$ active delay	t_{CLRL}	10	165	10	100	ns	
$\overline{RD}$ inactive delay	t_{CLRH}	10	150	10	80	ns	
$\overline{RD}$ inactive to next address active	t_{RHAV}	$t_{CLCL}-45$	–	$t_{CLCL}-40$	–	ns	
HLDA valid delay	t_{CLHAV}	10	160	10	100	ns	
$\overline{RD}$ width	t_{RLRH}	$2t_{CLCL}-75$	–	$2t_{CLCL}-50$	–	ns	
$\overline{WR}$ width	t_{WLWH}	$2t_{CLCL}-60$	–	$2t_{CLCL}-40$	–	ns	
Address valid to ALE low	t_{AVAL}	$t_{CLCH}-60$	–	$t_{CLCH}-40$	–	ns	
Output rise time	t_{OLOH}	–	20	–	20	ns	from 0.8 to 2.0V
Output fall time	t_{OHOL}	–	12	–	12	ns	from 2.0 to 0.8V

Maximum Mode System (using SAB 8288A bus controller) (figures 10 to 14)
Timing Requirements

Parameter	Symbol	Limit values				Unit	Test conditions
		SAB 8086		SAB 8086-2			
		min.	max.	min.	max.		
CLK cycle period SAB 8086	t_{CLCL}	200	500	125	500	ns	–
CLK low time	t_{CLCH}	118	–	68	–	ns	–
CLK high time	t_{CHCL}	69	–	44	–	ns	–
CLK rise time	t_{CH1CH2}	–	10	–	10	ns	from 1.0 to 3.5V
CLK fall time	t_{CL2CL1}	–	10	–	10	ns	from 3.5 to 1.0V
Data in setup time	t_{DVCL}	30	–	20	–	ns	–
Data in hold time	t_{CLDX}	10	–	10	–	ns	–
RDY setup time into SAB 8284A [1] [2]	t_{R1VCL}	35	–	35	–	ns	–
RDY hold time into SAB 8284A [1] [2]	t_{CLR1X}	0	–	0	–	ns	–
READY setup time into SAB 8086	t_{RYHCH}	118	–	68	–	ns	–
READY hold time into SAB 8086	t_{CHRYX}	30	–	20	–	ns	–
READY inactive to CLK [4]	t_{RYLCL}	−8	–	−8	–	ns	–
Setup time for recognition (INTR, NMI, $\overline{TEST}$) [2]	t_{INVCH}	30	–	15	–	ns	–
$\overline{RQ}/\overline{GT}$ setup time	t_{GVCH}	30	–	15	–	ns	–
$\overline{RQ}$ hold time into SAB 8086	t_{CHGX}	40	–	30	–	ns	–
Input rise time (except CLK)	t_{ILIH}	–	20	–	20	ns	from 0.8 to 2.0V
Input fall time (except CLK)	t_{IHIL}	–	12	–	12	ns	from 2.0 to 0.8V

[1]) Signal at SAB 8284B or SAB 8288A shown for reference only.
[2]) Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
[3]) Applies only to T3 and wait states.
[4]) Applies only to T2 state (8 ns into T3).

Timing Responses

Parameter	Symbol	Limit values				Unit	Test conditions
		SAB 8086		SAB 8086-2			
		min.	max.	min.	max.		
Command active delay [1]	t_{CLML}	10	35	10	35	ns	C_L = 20 to 100 pF for all SAB 8086 outputs (in addition to SAB 8086 self-load)
Command inactive delay [1]	t_{CLMH}	10	35	10	35	ns	
READY active to status passive [3]	t_{RYHSH}	–	110	–	65	ns	
Status active delay	t_{CHSV}	10	110	10	60	ns	
Status inactive delay	t_{CLSH}	10	130	10	70	ns	
Address valid delay	t_{CLAV}	10	110	10	60	ns	
Address hold time	t_{CLAX}	10	–	10	–	ns	
Address float delay	t_{CLAZ}	t_{CLAX}	80	t_{CLAX}	50	ns	
Status valid to ALE high [1]	t_{SVLH}	–	20	–	20	ns	
Status valid to MCE high [1]	t_{SVMCH}	–	20	–	20	ns	
CLK low to ALE valid [1]	t_{CLLH}	–	20	–	20	ns	
CLK low to MCE high [1]	t_{CLMCH}	–	20	–	20	ns	
ALE inactive delay [1]	t_{CHLL}	4	15	4	15	ns	
Data valid delay	t_{CLDV}	10	110	10	60	ns	
Data hold time	t_{CHDX}	10	–	10	–	ns	
Control active delay [1]	t_{CVNV}	5	45	5	45	ns	
Control inactive delay [1]	t_{CVNX}	10	45	10	45	ns	

[1]) Signal at SAB 8284B or SAB 8288A shown for reference only.
[2]) Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
[3]) Applies only to T3 and wait states.
[4]) Applies only to T2 state (8 ns into T3).

Timing Responses (cont'd)

Parameter	Symbol	Limit values				Unit	Test conditions
		SAB 8086		SAB 8086-2			
		min.	max.	min.	max.		
Address float to READ active	t_{AZRL}	0	–	0	–	ns	
$\overline{RD}$ active delay	t_{CLRL}	10	165	10	100	ns	C_L = 20 to 100 pF for all SAB 8086 outputs (in addition to SAB 8086 self-load)
$\overline{RD}$ inactive delay	t_{CLRH}	10	150	10	80	ns	
$\overline{RD}$ inactive to next address active	t_{RHAV}	$t_{CLCL}-45$	–	$t_{CLCL}-40$	–	ns	
Direction control active delay [1]	t_{CHDTL}	–	50	–	50	ns	
Direction control inactive delay [1]	t_{CHDTH}	–	30	–	30	ns	
$\overline{GT}$ active delay	t_{CLGL}	0	85	0	50	ns	
$\overline{GT}$ inactive delay	t_{CLGH}	0	85	0	50	ns	
$\overline{RD}$ width	t_{RLRH}	$2t_{CLCL}-75$	–	$2t_{CLCL}-50$	–	ns	
Output rise time	t_{OLOH}	–	20	–	20	ns	from 0.8 to 2.0V
Output fall time	t_{OHOL}	–	12	–	12	ns	from 2.0 to 0.8V

[1]) Signal at SAB 8284B or SAB 8288A shown for reference only.
[2]) Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
[3]) Applies only to T3 and wait states.
[4]) Applies only to T2 state (8 ns into T3).

AC Characteristics for SAB 8086-1

T_A = 0 to 70°C, V_{CC} = 5V ± 5%

Minimum Complexity System (figures 8, 9, 12, 15)
Timing Requirements (preliminary)

Parameter	Symbol	Limit values		Unit	Test conditions
		min.	max.		
CLK cyde period	t_{CLCL}	100	500	ns	–
CLK low time	t_{CLCH}	53	–	ns	–
CLK high time	t_{CHCL}	39	–	ns	–
CLK rise time	t_{CH1CH2}	–	10	ns	from 1.0 to 3.5V
CLK fall time	t_{CL1CL2}	–	10	ns	from 3.5 to 1.0V
Data in setup time	t_{DVCL}	5	–	ns	–
Data in hold time	t_{CLDX}	10	–	ns	–
RDY setup time into SAB 8284A[1)2)]	t_{R1VCL}	35	–	ns	–
RDY hold time into SAB 8284A[1)2)]	t_{CLR1X}	0	–	ns	–
READY setup time into SAB 8086	t_{RYHCH}	53	–	ns	–
READY hold time into SAB 8086	t_{CHRYX}	20	–	ns	–
READY inactive to CLK[3)]	t_{RYLCL}	–10	–	ns	–
HOLD setup time	t_{HVCH}	20	–	ns	–
INTR, NMI, $\overline{\text{TEST}}$ setup time [2)]	t_{INVCH}	15	–	ns	–
Input rise time (except CLK)	t_{ILIH}	–	20	ns	from 0.8 to 2.0V
Input fall time (except CLK)	t_{ILHIL}	–	12	ns	from 2.0 to 0.8V

[1)] Signal at SAB 8284B shown for reference only.
[2)] Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
[3)] Applies only to T2 state (8 ns into T3).

Timing Responses SAB 8086-1 (preliminary)

Parameter	Symbol	Limit values		Unit	Test conditions
		min.	max.		
Address valid delay	t_{CLAV}	10	50	ns	
Address hold time	t_{CLAX}	10	–	ns	
Address float delay	t_{CLAZ}	10	40	ns	
ALE width	t_{LHLL}	t_{CLCH}-10	–	ns	
ALE active delay	t_{CLLH}	–	40	ns	
ALE inactive delay	t_{CHLL}	–	45	ns	
Address hold time to ALE inactive	t_{LLAX}	t_{CHCL}-10	–	ns	
Data valid delay	t_{CLDV}	10	50	ns	
Data hold time	t_{CHDX}	10	–	ns	
Data hold time after WR	t_{WHDX}	t_{CLCH}-25	–	ns	
Control active delay 1	t_{CVCTX}	10	50	ns	C_L = 20 to 100 pF for all SAB 8086 outputs (in addition to SAB 8086 self-load)
Control active delay 2	t_{CHCTV}	10	45	ns	
Control inactive delay	t_{CVCTX}	10	50	ns	
Address float to READ active	t_{AZRL}	0	–	ns	
$\overline{RD}$ active delay	t_{CLRL}	10	70	ns	
$\overline{RD}$ inactive delay	t_{CLRH}	10	60	ns	
$\overline{RD}$ inactive to next address active	t_{RHAV}	t_{CLCL}-35	–	ns	
HLDA valid delay	t_{CLHAV}	10	60	ns	
$\overline{RD}$ width	t_{RLRH}	$2t_{CLCL}$-40	–	ns	
$\overline{WR}$ width	t_{WLWH}	$2t_{CLCL}$-35	–	ns	
Address valid to ALE low	t_{AVAL}	t_{CLCH}-35	–	ns	
Output rise time	t_{OLOH}	–	20	ns	from 0.8 to 2.0V
Output fall time	t_{OHOL}	–	12	ns	from 2.0 to 0.8V

Maximum Mode System (using SAB 8288A bus controller) (figures 10-14)
Timing Requirements SAB 8086-1 (preliminary)

Parameter	Symbol	Limit values		Unit	Test conditions
		min.	max.		
CLK cycle period	t_{CLCL}	100	500	ns	–
CLK low time	t_{CLCH}	53	–	ns	–
CLK high time	t_{CHCL}	39	–	ns	–
CLK rise time	t_{CH1CH2}	–	10	ns	from 1.0 to 3.5V
CLK fall time	t_{CL2CL1}	–	10	ns	from 3.5 to 1.0V
Data in setup time	t_{DVCL}	5	–	ns	–
Data in hold time	t_{CLDX}	10	–	ns	–
RDY setup time into SAB 8284A[1])[2])	t_{R1VCL}	35	–	ns	–
RDY hold time into SAB 8284A[1])[2])	t_{CLR1X}	0	–	ns	–
READY setup time into SAB 8086	t_{RYHCH}	53	–	ns	–
READY hold time into SAB 8086	t_{CHRYX}	20	–	ns	–
READY inactive to CLK[3])	t_{RYLCL}	–10	–	ns	–
Setup time for recognition (INTR, NMI, $\overline{TEST}$)[2])	t_{INVCH}	15	–	ns	–
$\overline{RQ}/\overline{GT}$ setup time	t_{GVCH}	12	–	ns	–
$\overline{RQ}$ hold time into SAB 8086	t_{CHGX}	20	–	ns	–
Input rise time (except CLK)	t_{ILIH}	–	20	ns	from 0.8 to 2.0V
Input fall time (except CLK)	t_{IHIL}	–	12	ns	from 2.0 to 0.8V

[1]) Signal at SAB 8284B or SAB 8288A shown for reference only.
[2]) Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
[3]) Applies only to T2 state (8 ns into T3).

Timing Responses SAB 8086-1 (preliminary)

Parameter	Symbol	Limit values		Unit	Test conditions
		min.	max.		
Command active delay[1])	t_{CLML}	10	35	ns	C_L = 20 to 100 pF for all SAB 8086 outputs (in addition to SAB 8086 self-load)
Command inactive delay[1])	t_{CLMH}	10	35	ns	
READY active to status passive[2])	t_{RYHSH}	–	45	ns	
Status active delay	t_{CHSV}	10	45	ns	
Status inactive delay	t_{CLSH}	10	55	ns	
Address valid delay	t_{CLAV}	10	50	ns	
Address hold time	t_{CLAX}	10	–	ns	
Address float delay	t_{CLAZ}	10	40	ns	
Status valid to ALE high[1])	t_{SVLH}	–	20	ns	
Status valid to MCE high[1])	t_{SVMCH}	–	20	ns	
CLK low to ALE valid[1])	t_{CLLH}	–	20	ns	
CLK low to MCE high[1])	t_{CLMCH}	–	20	ns	
ALE inactive delay[1])	t_{CHLL}	4	15	ns	
Data valid delay	t_{CLDV}	10	50	ns	
Data hold time	t_{CHDX}	10	–	ns	
Control active delay[1])	t_{CVNV}	5	45	ns	
Control inactive delay[1])	t_{CVNX}	10	45	ns	

[1]) Signal at SAB 8284B or SAB 8288A shown for reference only.
[2]) Applies only to T3 and wait states.

Timing Responses SAB 8086-1 (cont'd)
(preliminary)

Parameter	Symbol	Limit values		Unit	Test conditions
		min.	max.		
Address float to READ active	t_{AZRL}	0	–	ns	C_L = 20 to 100 pF for all SAB 8086 outputs (in addition to SAB 8086 self-load)
$\overline{RD}$ active delay	t_{CLRL}	10	70	ns	
$\overline{RD}$ inactive delay	t_{CLRH}	10	60	ns	
$\overline{RD}$ inactive to next address active	t_{RHAV}	t_{CLCL}-35	–		
Direction control active delay[1])	t_{CHDTL}	–	50	ns	
Direction control inactive delay[1])	t_{CHDTH}	–	30	ns	
$\overline{GT}$ active delay	t_{CLGL}	0	45	ns	
$\overline{GT}$ inactive delay	t_{CLGH}	0	45	ns	
$\overline{RD}$ width	t_{RLRH}	$2t_{CLCL}$-40	–	ns	
Output rise time	t_{OLOH}	–	20	ns	from 0.8 to 2.0V
Output fall time	t_{OHOL}	–	12	ns	from 2.0 to 0.8V

[1]) Signal at SAB 8284B or SAB 8288A shown for reference only.
[2]) Applies only to T3 and wait states.

Figure 8 Bus Timing – Minimum Mode System

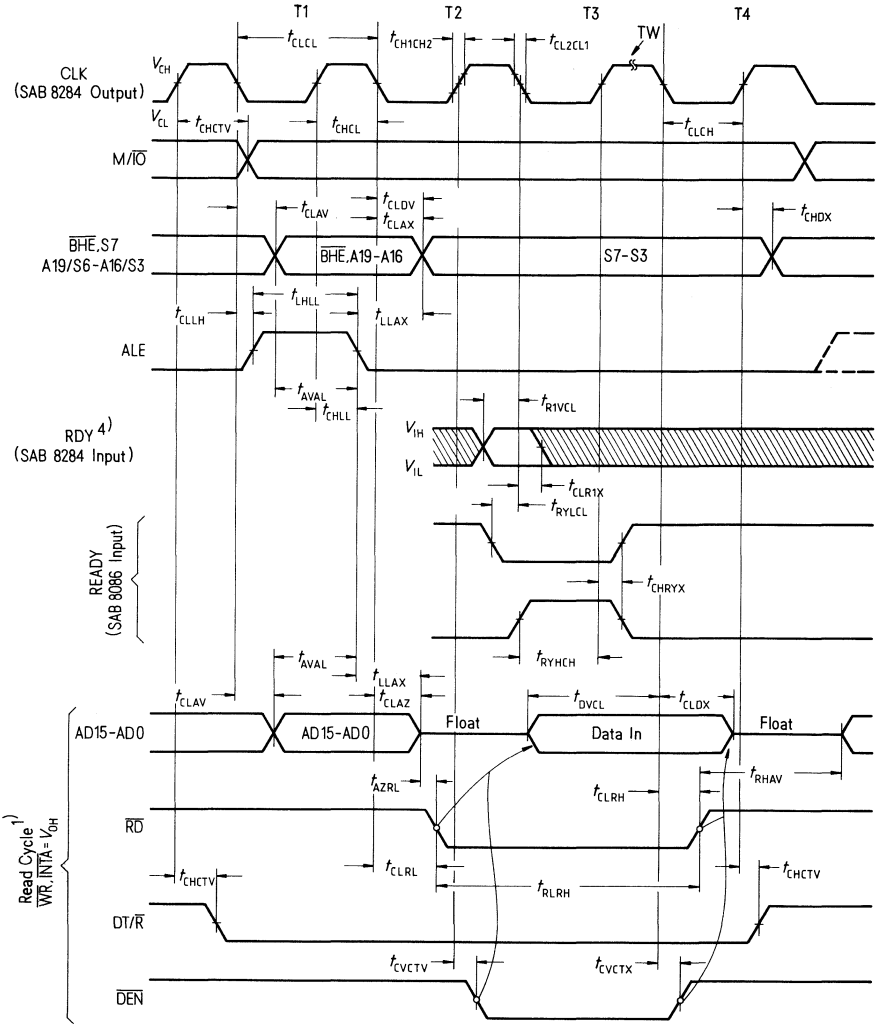
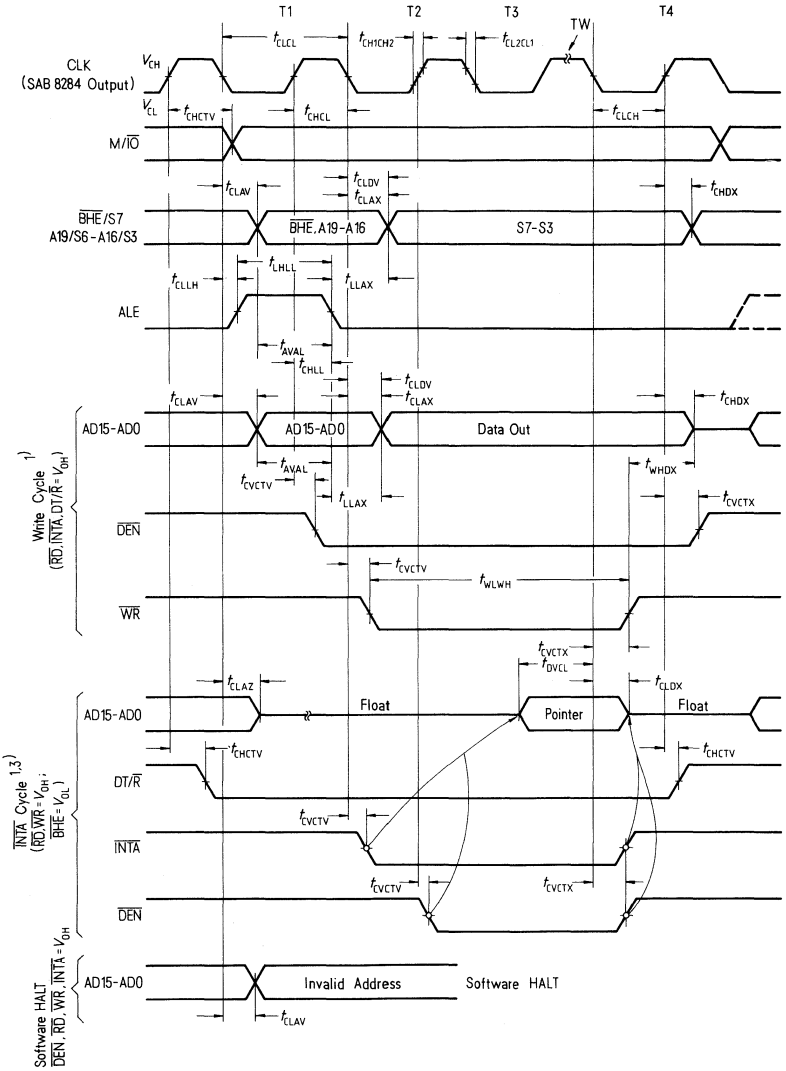


Figure 9 SAB 8086 Bus Timing – Minimum Mode System (cont'd)



1) All signals switch between V_{OH} and V_{OL} unless otherwise specified.
2) RDY is sampled near the end of T2, T3, TW to determine if TW machines states are to be inserted.
3) Two INTA cycles run back to back. The SAB 8086 local ADDR/DATA bus is floating during both INTA cycles. Control signals shown for second INTA cycle.
4) Signals at SAB 8284B are shown for reference only.
5) All timing measurements are made at 1.5 V unless otherwise noted.

Figure 10 SAB 8086 Bus Timing – Maximum Mode System (using SAB 8288A)

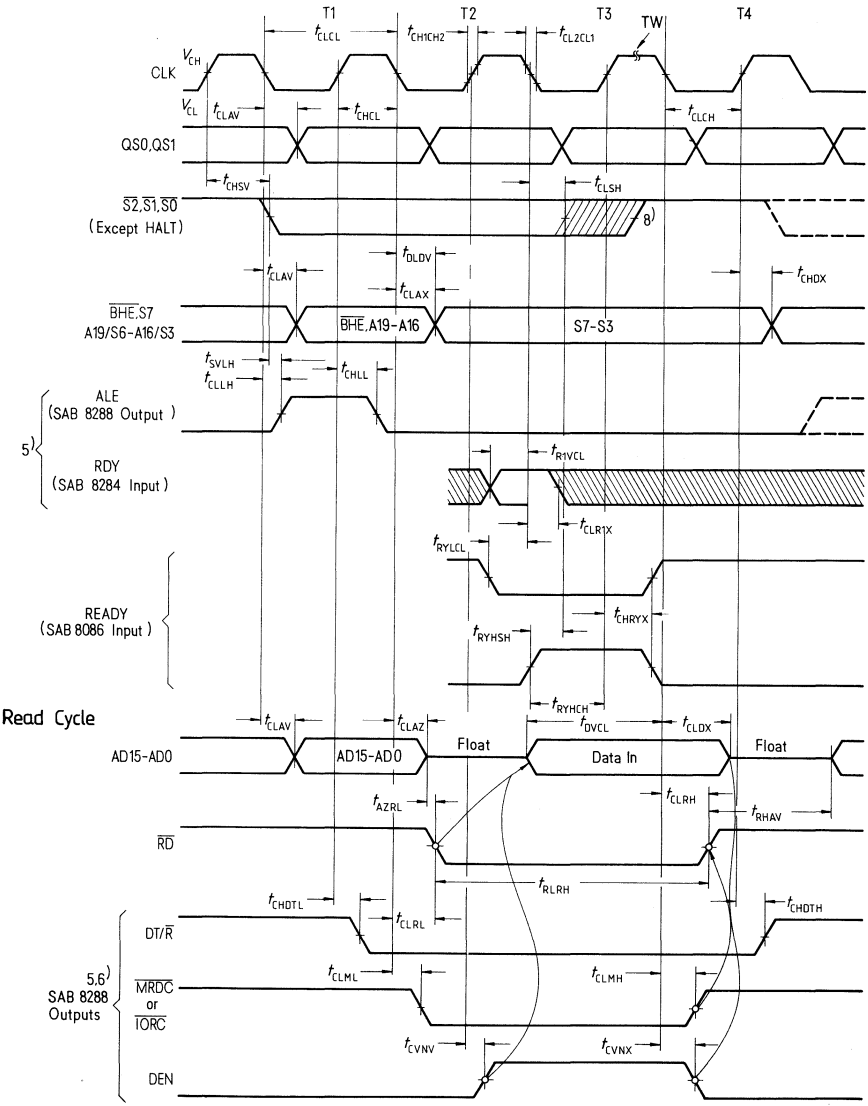
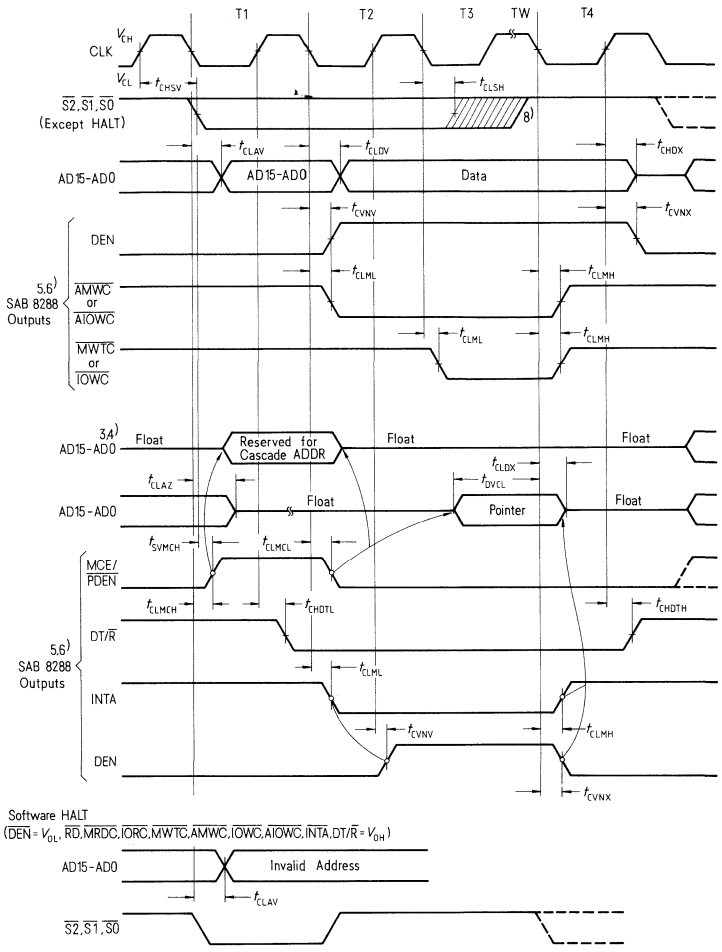
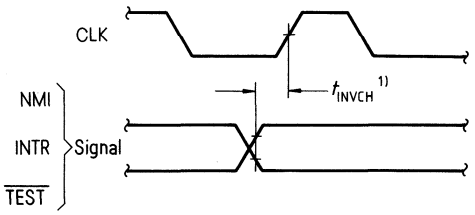


Figure 11 SAB 8086 Bus Timing – Maximum Mode System (using SAB 8288A) (cont'd)



[1]) All signals switch between V_{OH} and V_{OL} unless otherwise specified.
[2]) RDY is sampled near the end of T2, T3, TW to determine if TW machines states are to be inserted.
[3]) Cascade address is valid between first and second INTA cycle.
[4]) Two INTA cycles run back-to-back. The SAB 8086 local ADDR/DATA bus is floating during both INTA cycles. Control for pointer address is shown for second INTA cycle.
[5]) Signals at SAB 8284B or SAB 8288A are shown for reference only.
[6]) The issuance of the SAB 8288A command and control signals ($\overline{MRDC}$, $\overline{MWTC}$, $\overline{AMWC}$, $\overline{IORC}$, $\overline{IOWC}$, $\overline{AIOWC}$, $\overline{INTA}$ and DEN) lags the active high SAB 8288A DEN.
[7]) All timing measurements are made at 1.5 V unless otherwise noted.
[8]) Status inactive in state just prior to T4.

Figure 12 Asynchronous Signal Recognition



1) Setup requirements for asynchronous signals only to guarantee recognition at next CLK

Figure 13 Bus Lock Signal Timing (Maximum Mode only)

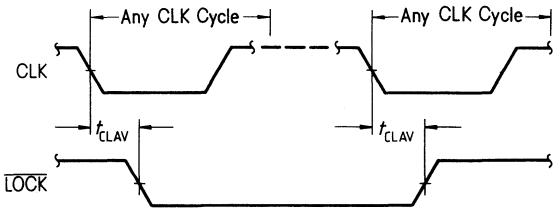
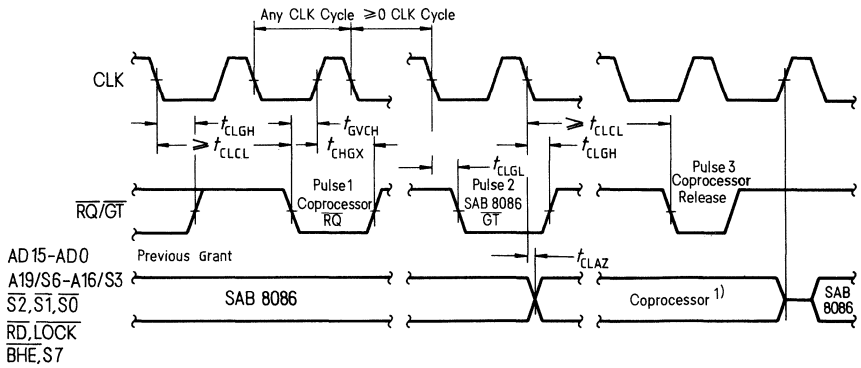
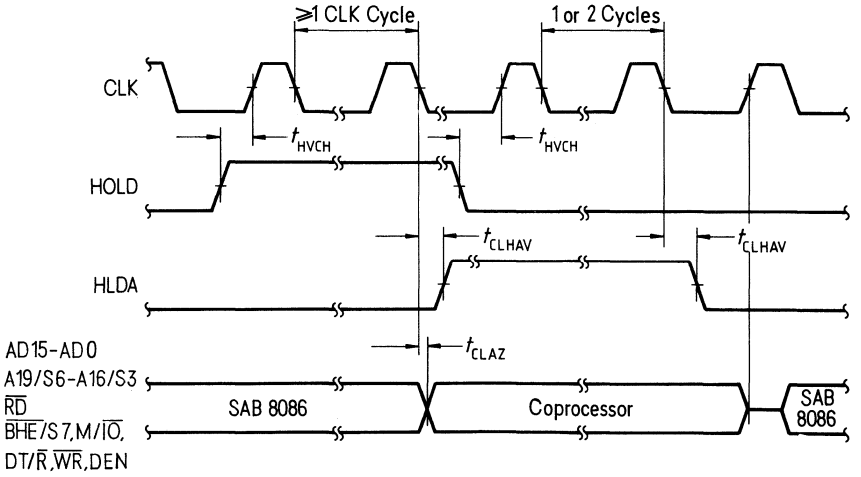


Figure 14 Request/Grant Sequence Timing (Maximum Mode only)



1) The coprocessor may not drive the buses outside the region shown without risking contention

Figure 15 Hold/Hold Acknowledge Timing (Minimum Mode only)



Ordering Information

Type	Ordering code	Description
SAB 8086-P	Q67120-C116	16-bit microprocessor – 5 MHz (plastic)
SAB 8086-2-P	Q67120-C142	16-bit microprocessor – 8 MHz (plastic)
SAB 8086-1-P	Q67120-C141	16-bit microprocessor – 10 MHz (plastic)

SAB 8088
8-Bit Microprocessor

Preliminary

SAB 8088 5 MHz
SAB 8088-2 8 MHz

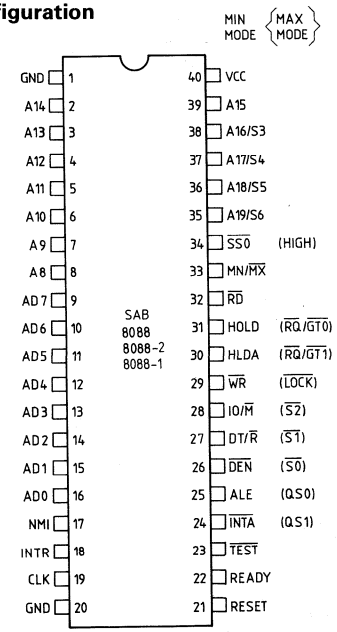
- 8-bit data bus interface
- 16-bit internal architecture
- Direct addressing capability to 1 Mbyte of memory
- Software compatible with SAB 8086
- 14-word by 16-bit register set with symetrical operations
- Byte, word and block operations
- 24 operand addressing modes

SAB 8088-1 10 MHz

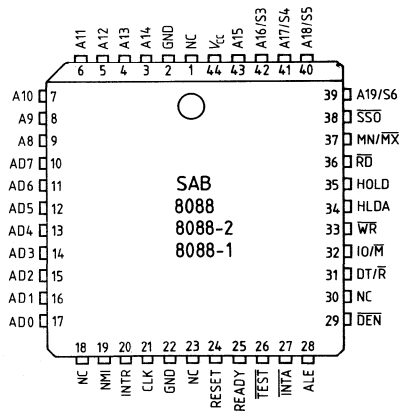
- 8-bit and 16-bit signed and unsigned arithmetic in binary or decimal, including multiply and divide
- Clock rates:
  5 MHz for SAB 8088
  8 MHz for SAB 8088-2
  10 MHz for SAB 8088-1
- Compatible with industry standard 8088
- Available in a 40-pin plastic dual-in-line package (P-DIP-40) or in a plastic leaded chip carrier package (PL-CC-44)

Pin Configuration

P-DIP-40



PL-CC-44



SAB 8088 is a high-performance 8-bit microprocessor implemented in +5V advanced Siemens MYMOS technology, packaged in a 40-pin plastic dual-in-line package (P-DIP-40) or in a 44-pin plastic leaded chip carrier package (PL-CC-44). It is 100 percent compatible with the industry standard 8088. With features like string handling, 16-bit arithmetic with multiply and divide it significantly increases system performance. It is highly suited for multi-processor applications in various configurations.

Pin Definitions and Functions

The following pin definitions are for SAB 8088 systems in **either minimum or maximum mode.** The "local bus" in these descriptions is the direct multiplexed bus interface connection to the SAB 8088 (without regard to additional bus buffers).

<table>
<tr><th>Symbol</th><th>Pin</th><th>Input (I)
Output (O)</th><th>Function</th></tr>
<tr><td>AD7–AD0</td><td>9–16</td><td>I/O</td><td>ADDRESS DATA BUS
These lines constitute the time multiplexed memory I/O address (T1) and data (T2, T3, Tw, and T4) bus. These lines are active high and float to tristate off during interrupt acknowledge and local bus "hold acknowledge".</td></tr>
<tr><td>A15–A8</td><td>39, 2–8</td><td>O</td><td>ADDRESS BUS
These lines provide address bits 8 through 15 for the entire bus cycle (T1–T4). These lines do not have to be latched by ALE to remain valid. A15–A8 are active high and float to tristate off during interrupt acknowledge and local bus "hold acknowledge".</td></tr>
<tr><td>A19/S6, A18/S5,
A17/S4, A16/S3</td><td>34–38</td><td>O</td><td>ADDRESS/STATUS
During T1, these are the four most significant address lines for memory operations. During I/O operations, these lines are low. During memory and I/O operations, status information is available on these lines during T2, T3, TW and T4. S6 is always low. The status of the interrupt enable flag bit (S5) is updated at the beginning of each clock cycle. S4 and S3 are encoded as shown.
<table>
<tr><th>S4</th><th>S3</th><th>Characteristics</th></tr>
<tr><td>0</td><td>0</td><td>Alternate Data</td></tr>
<tr><td>0</td><td>1</td><td>Stack</td></tr>
<tr><td>1</td><td>0</td><td>Code or None</td></tr>
<tr><td>1</td><td>1</td><td>Data</td></tr>
</table>
This information indicates which segment register is presently being used for data accessing.
These lines float to tristate off during local bus "hold acknowledge".</td></tr>
<tr><td>$\overline{RD}$</td><td>32</td><td>O</td><td>$\overline{READ}$
Read strobe indicates that the processor is performing a memory or I/O read cycle, depending on the state of the IO/$\overline{M}$ pin or S2. This signal is used to read devices which reside on the SAB 8088 local bus. $\overline{RD}$ is active low during T2, T3 and TW of any read cycle, and is guaranteed to remain high in T2 until the SAB 8088 local bus has floated.
This signal floats to tristate off in "hold acknowledge".</td></tr>
<tr><td>READY</td><td>22</td><td>I</td><td>READY
This is the acknowledgement from the addressed memory or I/O device that it will complete the data transfer. The RDY signal from memory or I/O is synchronized by the SAB 8284A/8284B clock generator to form READY. This signal is active high. The SAB 8088 READY input is not synchronized. Correct operation is not guaranteed if the setup and hold times are not met.</td></tr>
</table>

Pin Definitions and Functions (cont'd)

Symbol	Pin	Input (I) Output (O)	Function
INTR	18	I	**INTERRUPT REQUEST** This is a level triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. A subroutine is vectored to via an interrupt vector lookup table located in system memory. It can be internally masked by software resetting the interrupt enable bit. INTR is internally synchronized. This signal is active high.
$\overline{\text{TEST}}$	23	I	**$\overline{\text{TEST}}$** This input is examined by the "wait for test" instruction. If the $\overline{\text{TEST}}$ input is low, execution continues, otherwise the processor waits in an "idle" state. This input is synchronized internally during each clock cycle on the leading edge of CLK.
NMI	17	I	**NON-MASKABLE INTERRUPT** This is an edge triggered input which causes a type 2 interrupt. A subroutine is vectored to via interrupt vector lookup table located in system memory. NMI is not maskable internally by software. A transition from low to high initiates the interrupt at the end of the current instruction. This input is internally synchronized.
RESET	21	I	**RESET** Causes the processor to immediately terminate its present activity. The signal must be active high for at least four clock cycles. It restarts execution, as described in the instruction set description, when RESET returns low. RESET is internally synchronized.
CLK	19	I	**CLOCK** Provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing.
V_{CC}	40	–	**POWER SUPPLY** (+5V)
GND	1, 20	–	**GROUND** (0 V)
MN/$\overline{\text{MX}}$	33	I	**MINIMUM/$\overline{\text{MAXIMUM}}$** Indicates what mode the processor is to operate in. The two modes are discussed in the following sections.

Pin Definitions and Functions (cont'd)

The following pin descriptions are for the SAB 8088 **minimum mode** (i.e. MN/$\overline{MX}$ = V_{CC}). Only the pin functions which are unique to minimum mode are described; all other pin functions are as already described.

Symbol	Pin	Input (I) Output (O)	Function
IO/$\overline{M}$	28	O	**STATUS LINE** Is an inverted maximum mode $\overline{S2}$. It is used to distinguish a memory access from an I/O access. IO/$\overline{M}$ becomes valid in the T4 preceding a bus cycle and remains valid until the final T4 of the cycle (I/$\overline{O}$ = high, M = low). IO/$\overline{M}$ floats to tristate off in local bus "hold acknowledge".
$\overline{WR}$	29	O	**$\overline{WRITE}$** The write strobe indicates that the processor is performing a write memory or write I/O cycle depending on the state of the IO/$\overline{M}$ signal. $\overline{WR}$ is active for T2, T3, and TW of any write cycle. It is active low, and floats to tristate off in local bus "hold acknowledge".
$\overline{INTA}$	24	O	**$\overline{INTERRUPT\ ACKNOWLEDGE}$** Is used as a read strobe for interrupt acknowledge cycles. It is active low during T2, T3, and TW of each interrupt acknowledge cycle.
ALE	25	O	**ADDRESS LATCH ENABLE** Is provided by the processor to latch the address into the SAB 8282 /8282A/8283/8283A address latch. It is a high pulse active during clock low of T1 of any bus cycle. Note that ALE is never floated.
DT/$\overline{R}$	27	O	**DATA TRANSMIT/$\overline{RECEIVE}$** Is needed in a minimum system that desires to use an SAB 8286/8286A/8287/8287A data bus transceiver. It is used to control the direction of data flow through the transceiver. Logically, DT/$\overline{R}$ is equivalent to $\overline{S1}$ in the maximum mode, and its timing is the same as for IO/$\overline{M}$ (T = high, R = low). This signal floats to tristate off in local "hold acknowledge".
$\overline{DEN}$	26	O	**$\overline{DATA\ ENABLE}$** Is provided as an output enable for the SAB 8286/8286A/ 8287/8287A in a minimum system which uses the transceiver. $\overline{DEN}$ is active low during each memory and I/O access, and for $\overline{INTA}$ cycles. For a read or $\overline{INTA}$ cycle, it is active from the middle of T2 until the middle of T4, while for a write cycle, it is active from the beginning of T2 until the middle of T4. $\overline{DEN}$ floats to tristate off during local bus "hold acknowledge".
HOLD, HLDA	31, 30	I/O	**HOLD** Indicates that another master is requesting a local bus "hold". To be acknowledged, HOLD must be active high. The processor receiving the "hold" request will issue HLDA (high) as an acknowledgement, in the middle of a T4 or T1 clock cycle. Simultaneous with the issuance of HLDA the processor will float the local bus and control lines. After HOLD is detected as being low, the processor lowers HLDA, and when the processor needs to run another cycle, it will again drive the local bus and control lines. HOLD is not an asynchronous input. External synchronization should be provided if the system cannot otherwise guarantee the setup time.

Pin Definitions and Functions (cont'd)

<table>
<tr><th>Symbol</th><th>Pin</th><th>Input (I)
Output (O)</th><th>Function</th></tr>
<tr><td>$\overline{SS0}$</td><td>34</td><td>O</td><td>$\overline{STATUS\ LINE}$
It is logically equivalent to $\overline{S0}$ in the maximum mode. The combination of $\overline{SS0}$, IO/$\overline{M}$ and DT/$\overline{R}$ allows the system to completely decode the current bus cycle status.
<table>
<tr><th>IO/$\overline{M}$</th><th>DT/$\overline{R}$</th><th>$\overline{SS0}$</th><th>Characteristics</th></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Interrupt Acknowledge</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Read I/O Port</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Write I/O Port</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Halt</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Code Access</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Read Memory</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Write Memory</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Passive</td></tr>
</table></td></tr>
</table>

Pin Definitions and Functions (cont'd)

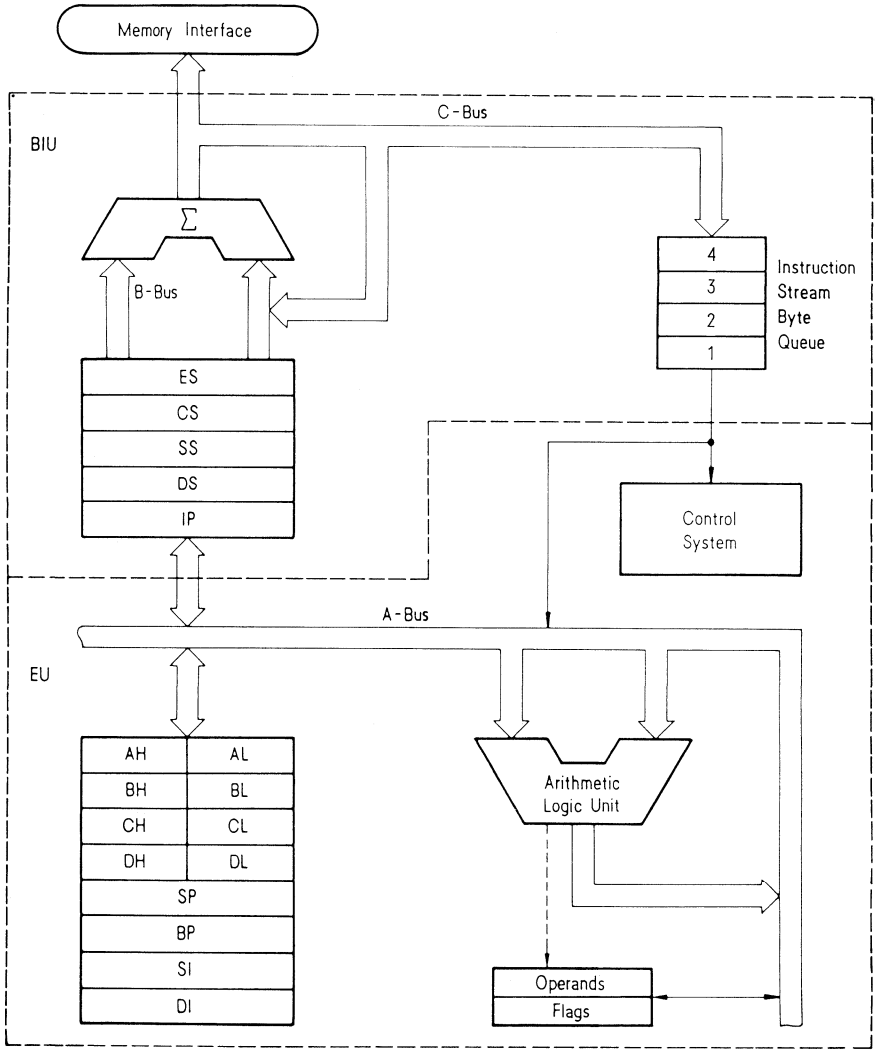
The following pin descriptions are for the SAB 8088/8288 system in **maximum mode** (i.e. MN/$\overline{MX}$ = GND). Only the pin functions which are unique to maximum mode are described. All other pin functions are as already described.

<table>
<tr><th>Symbol</th><th>Pin</th><th>Input (I)
Output (O)</th><th>Function</th></tr>
<tr><td>$\overline{S2}$, $\overline{S1}$, $\overline{S0}$</td><td>28–26</td><td>O</td><td>$\overline{STATUS}$
Is active during clock high of T4, T1, and T2, and is returned to the passive state (1,1,1) during T3 or during TW when READY is high. This status is used by the SAB 8288/8288A bus controller to generate all memory and I/O access control signals. Any change by $\overline{S2}$, $\overline{S1}$, or $\overline{S0}$ during T4 is used to indicate the beginning of a bus cycle, and the return to the passive state in T3 or TW is used to indicate the end of a bus cycle.
These signals float to tristate off during "hold acknowledge". During the first clock cycle after RESET becomes active, these signals are active high. After this first clock, they float to tristate off.
<table>
<tr><th>$\overline{S2}$</th><th>$\overline{S1}$</th><th>$\overline{S0}$</th><th>Characteristics</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Interrupt Acknowledge</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Read I/O Port</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Write I/O Port</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Halt</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Code Access</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Read Memory</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Write Memory</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Passive</td></tr>
</table>
</td></tr>
<tr><td>$\overline{RQ}$/$\overline{GT0}$
RQ/$\overline{GT1}$</td><td>31
30</td><td>I/O
I/O</td><td>REQUEST/$\overline{GRANT}$
Pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bidirectional with $\overline{RQ}$/$\overline{GT0}$ having higher priority than $\overline{RQ}$/$\overline{GT1}$. RQ/$\overline{GT}$ has an internal pullup resistor so may be left unconnected. The request/grant sequence is as follows (see page 38):
1. A pulse of one CLK wide from another local bus master indicates a local bus request ("hold") to the SAB 8088 (pulse 1).
2. During a T4 or T1 clock cycle, a pulse one clock wide from the SAB 8088 to the requesting master (pulse 2), indicates that the SAB 8088 has allowed the local bus to float and that it will enter the "hold acknowledge" state at the next CLK. The CPU's bus interface unit is disconnected logically from the local bus during "hold acknowledge". The same rules as for HOLD/HOLDA apply as for when the bus is released.
3. A pulse one CLK wide from the requesting master indicates to the SAB 8088 (pulse 3) that the "hold" request is about to end and that the SAB 8088 can reclaim the local bus at the next CLK. The CPU then enters T4.
Each master-master exchange of the local bus is a sequence of three pulses. There must be one idle CLK cycle after each bus exchange. Pulses are active low.</td></tr>
</table>

Pin Definitions and Functions (cont'd)

<table>
<tr><th>Symbol</th><th>Pin</th><th>Input (I)
Output (O)</th><th>Function</th></tr>
<tr><td></td><td></td><td></td><td>If the request is made while the CPU is performing a memory cycle, it will release the local bus during T4 of the cycle when all the following conditions are met:
1. Request occurs on or before T2.
2. Current cycle is not the low byte of a word.
3. Current cycle is not the first acknowledge of an interrupt acknowledge sequence.
4. A locked instruction is not currently executing.
If the local bus is idle when the request is made the two possible events will follow:
1. Local bus will be released during the next clock.
2. A memory cycle will start within 3 clocks. Now the four rules for a currently active memory cycle apply with condition number 1 already satisfied.</td></tr>
<tr><td>$\overline{\text{LOCK}}$</td><td>29</td><td>O</td><td>$\overline{\text{LOCK}}$
Indicates that other system bus masters are not to gain control of the system bus while $\overline{\text{LOCK}}$ is active (low). The $\overline{\text{LOCK}}$ signal is activated by the "LOCK" prefix instruction and remains active until the completion of the next instruction. This signal is active low, and floats to tristate off in "hold acknowledge".</td></tr>
<tr><td>QS1, QS0</td><td>24, 25</td><td>O</td><td>QUEUE STATUS
Provide status to allow external tracking of the internal SAB 8088 instruction queue.
The queue status is valid during the CLK cycle after which the queue operation is performed.
<table>
<tr><th>QS1</th><th>QS0</th><th>Characteristics</th></tr>
<tr><td>0</td><td>0</td><td>No Operation</td></tr>
<tr><td>0</td><td>1</td><td>First Byte of Op Code from Queue</td></tr>
<tr><td>1</td><td>0</td><td>Empty the Queue</td></tr>
<tr><td>1</td><td>1</td><td>Subsequent Byte from Queue</td></tr>
</table></td></tr>
<tr><td>–</td><td>34</td><td>O</td><td>Pin 34 is always high in the maximum mode.</td></tr>
</table>

Block Diagram



Functional Description

Memory Organization

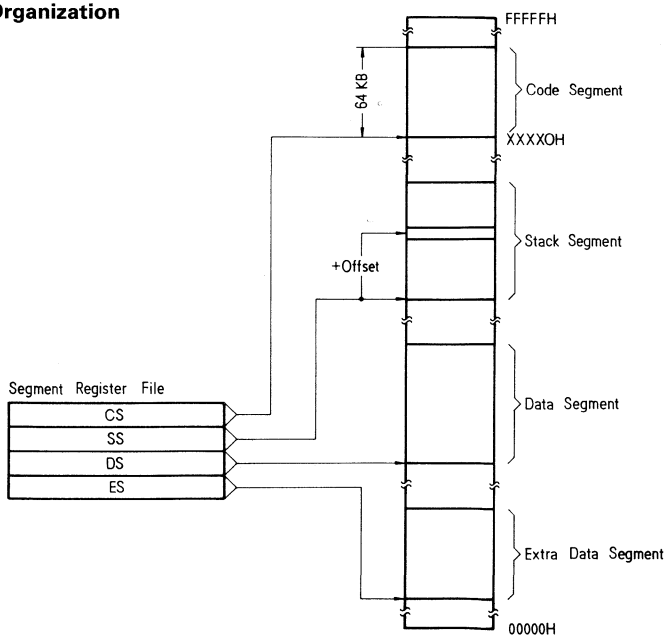
The processor provides a 20-bit address to memory which locates the byte being referenced. The memory is organized as a linear array of up to 1 million bytes, addressed as 00000(H) to FFFFF(H). The memory is logically divided into code, data, extra data, and stack segments of up to 64 Kbytes each, with each segment falling on 16-byte boundaries.

All memory references are made relative to base addresses contained in high-speed segment registers. The segment types were chosen based on the addressing needs of programs. The segment register to be selected is automatically chosen according to the rules of the following table. All information in one segment type shares the same logical attributes (e.g. code or data). By structuring memory into relocatable areas of similar characteristics and by automatically selecting segment registers, programs are shorter, faster, and more structured.

Word (16-bit) operands can be located on even or odd address boundaries. For address and data operands, the least significant byte of the word is stored in the lower valued address location and the most significant byte in the next higher address location. The BIU will automatically execute two fetch or write cycles for 16-bit operands.

Certain locations in memory are reserved for specific CPU operations. Locations from addresses FFFF0H trough FFFFFH are reserved for operations including a jump to the initial system initialization routine. Following RESET, the CPU will always begin execution at location FFFF0H where the jump must be located. Locations 00000H through 003FFH are reserved for interrupt operations. Four-byte pointers consisting of a 16-bit segment address and a 16-bit offset address direct program flow to one of the 256 possible interrupt service routines. The pointer elements are assumed to have been stored at their respective places in reserved memory prior to the occurrence of interrupts.

Memory Organization



Minimum and Maximum Modes

The requirements for supporting minimum and maximum mode in SAB 8088 systems are sufficiently different that they cannot be met efficiently with 40 uniquely defined pins. Consequently, the SAB 8088 is equipped with a strap pin (MN/$\overline{MX}$) which defines the system configuration. The definition of a certain subset of the pins changes, dependent on the condition of the strap pin. When the MN/$\overline{MX}$ pin is strapped to GND, the SAB 8088 defines pins 24 through 31 and 34 in maximum mode. When the MN/$\overline{MX}$ pin is strapped to VCC, the SAB 8088 generates bus control signals itself on pins 24 through 31 and 34.

The minimum mode SAB 8088 can be used with either a multiplexed or demultiplexed bus. The multiplexed bus configuration is compatible with the SAB 8085A multiplexed bus peripherals (e.g. SAB 8155) and provides the user with a minimum chip count system. This architecture provides the SAB 8088 processing power in a highly integrated form.

The demultiplexed mode requires one latch (for 64K addressability) or two latches (for a full megabyte of addressing). A third latch can be used for buffering if the address bus loading requires it. An SAB 8286/8286A or SAB 8287/8287A transceiver can also be used if data bus buffering is required. The SAB 8088 provides $\overline{DEN}$ and DT/$\overline{R}$ to control the transceiver, and ALE to latch the addresses. This configuration of the minimum mode provides the standard demultiplexed bus structure with heavy bus buffering and relaxed bus timing requirements.

The maximum mode employs the SAB 8288/8288A bus controller. The SAB 8288/8288A decodes status lines $\overline{S0}$, $\overline{S1}$, and $\overline{S2}$, and provides the system with all bus control signals. Moving the bus control to the SAB 8288/8288A provides better source and sink current capability to the control lines, and frees the SAB 8088 pins for extended large system features. Hardware lock, queue status, and two request/grant interfaces are provided by the SAB 8088 in maximum mode. These features allow coprocessors in local bus and remote bus configurations.

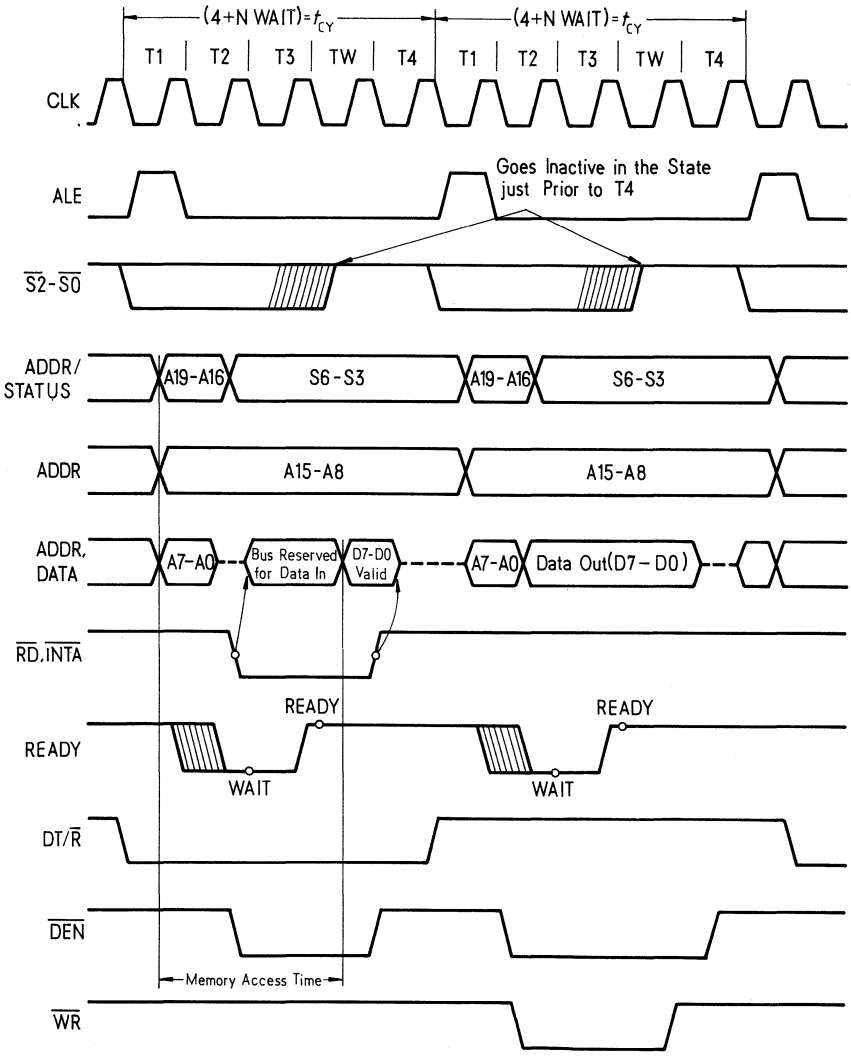
Bus Operation

The SAB 8088 address/data bus is broken into three parts – the lower eight address/data bits (AD0–AD7), the midle eight address bits (A8–A15), and the upper four address bits (A16–A19). The address/data bits and the highest four address bits are time multiplexed. This technique provides the most efficient use of pins on the processor, permitting the use of a standard 40 lead package. The middle eight address bits are not multiplexed, i.e. they remain valid throughout each bus cycle. In addition, the bus can be demultiplexed at the processor with a single address latch if a standard, non-multiplexed bus is desired for the system.

Each processor bus cycle consists of at least four CLK cycles. These are referred to as T1, T2, T3 and T4. The address is emitted from the processor during T1 and data transfer occurs on the bus during T3 and T4. T2 is used primarily for changing the direction of the bus during read operations. In the event that a "NOT READY" indication is given by the addressed device, wait states (TW) are inserted between T3 and T4. Each inserted wait state is of the same duration as a CLK cycle. Periods can occur between SAB 8088 driven bus cycles. These are referred to as "idle" states (Ti), or inactive CLK cycles. The processor uses these cycles for internal housekeeping.

During T1 of any bus cycle, the ALE (address latch enable) signal is emitted (by either the processor or the SAB 8288/8288A bus controller, depending on the MN/$\overline{MX}$ strap). At the trailing edge of this pulse, a valid address and certain status information for the cycle may be latched.

Basic System Timing



Status bits $\overline{S0}$, $\overline{S1}$, and $\overline{S2}$ are used, in maximum mode, by the bus controller to identify the type of bus transaction according to the following table:

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	Characteristics
0 (Low)	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt
1 (High)	0	0	Instruction Fetch
1	0	1	Read Data from Memory
1	1	0	Write Data to Memory
1	1	1	Passive (no bus cycle)

Status bits S3 through S6 are multiplexed with highorder address bits and are therefore valid during T2 through T4. S3 and S4 indicate which segment register was used for this bus cycle in forming the address according to the following table:

S4	S3	Characteristics
0 (Low)	0	Alternate Data (extra segment)
0	1	Stack
1 (High)	0	Code or None
1	1	Data

S5 is a reflection of the PSW interrupt enable bit. S6 is always equal to 0.

I/O Addressing

In the SAB 8088, I/O operations can address up to a maximum of 64K I/O registers. The I/O address appears in the same format as the memory address on bus lines A15 to A0. The address lines A19 to A16 are zero in I/O operations. The variable I/O instructions, which use register DX as a pointer have full address capability, while the direct I/O instructions directly address one or two of the 256 I/O byte locations in page 0 of the I/O address space. I/O ports are addressed in the same manner as memory locations.

Design engineers familiar with the SAB 8085 or upgrading an SAB 8085 design should observe that the SAB 8085 addresses I/O with an 8-bit address on both halves of the 16-bit address bus. The SAB 8088 uses a full 16-bit address on its lower 16 address lines.

System Components

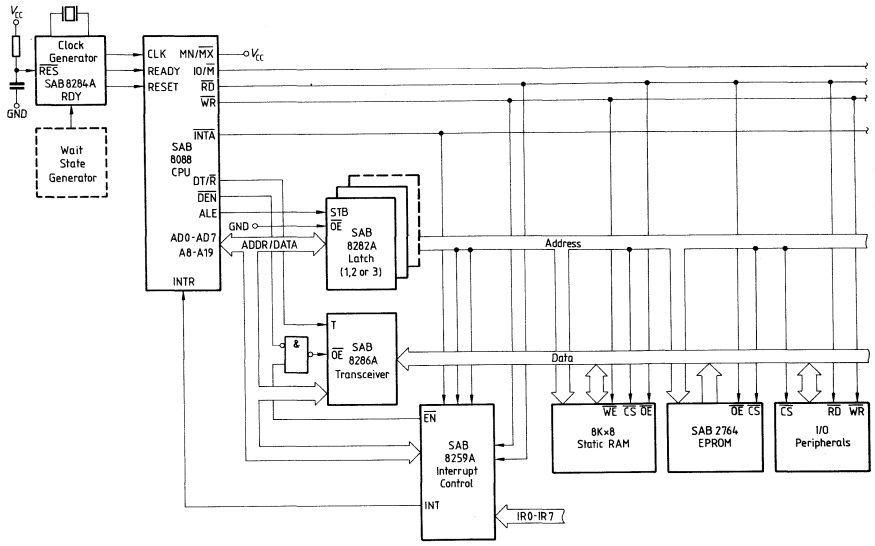
Support Circuits

SAB 8282/8282A	Octal Latch
SAB 8283/8283A	Octal Latch (inverting)
SAB 8284A/8284B	Clock Generator and Driver
SAB 8286/8286A	Octal Bus Transceiver
SAB 8287/8287A	Octal Bus Transceiver (inverting)
SAB 8288/8288A	Bus Controller
SAB 8289	Bus Arbiter
SAB 8259A	Programmable Interrupt Controller

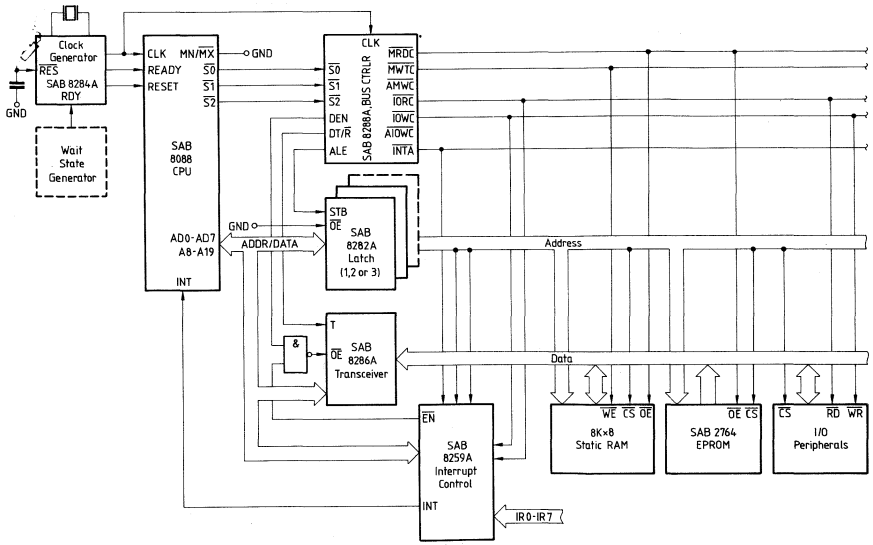
Typical Applications

The SAB 8088 is a general-purpose 8-bit micro processor which can be used for applications ranging from process control to data processing. The next page shows typical system configurations for SAB 8088 familiy components.

Minimum Mode SAB 8088 Typical System Configuration



Maximum Mode SAB 8088 Typical System Configuration



Instruction Set Summary

Data Transfer
MOV = Move:

	76543210	76543210	76543210	76543210
Register / memory to / from register	100010dw	mod reg r/m		
Immediate to register/memory	1100011w	mod 000 r/m	data	data if w=1
Immediate to register	1011w reg	data	data if w=1	
Memory to accumulator	1010000w	addr-low	addr-high	
Accumulator to memory	1010001w	addr-low	addr-high	
Register/memory to segment register	10001110	mod 0 reg r/m		
Segment register to register/memory	10001100	mod 0 reg r/m		

PUSH = Push:

Register/memory	11111111	mod 110 r/m
Register	01010 reg	
Segment register	000 reg 110	

POP = Pop:

Register/memory	10001111	mod 000 r/m
Register	01011 reg	
Segment register	000 reg 111	

XCHG = Exchange:

Register/memory with register	1000011w	mod reg r/m
Register with accumulator	10010 reg	

IN = Input from:

Fixed port	1110010w	port
Variable port	1110110w	

OUT = Output to:	76543210	76543210	76543210	76543210
Fixed port	1110011w	port		
Variable port	1110111w			
XLAT = Translate byte to AL	11010111			
LEA = Load EA to register	10001101	mod reg r/m		
LDS = Load pointer to DS	11000101	mod reg r/m		
LES = Load pointer to ES	11000100	mod reg r/m		
LAHF = Load AH with flags	10011111			
SAHF = Store AH into flags	10011110			
PUSHF = Push flags	10011100			
POPF =Pop flags	10011101			

Arithmetic

ADD = Add:				
Reg./memory with register to either	000000dw	mod reg r/m		
Immediate to register/memory	100000sw	mod 000 r/m	data	data if s: w=01
Immediate to accumulator	0000010w	data	data if w=1	

ADC = Add with carry:				
Reg./memory with register to either	000100dw	mod reg r/m		
Immediate to register/memory	100000sw	mod 010 r/m	data	data if s:w=01
Immediate to accumulator	0001010w	data	data if w=1	

INC = Increment:		
Register/memory	1111111w	mod 000 r/m
Register	01000reg	
AAA = ASCII adjust for add	00110111	
DAA = Decimal adjust for add	00100111	

SUB = Subtract:	76543210	76543210	76543210	76543210
Reg./memory and register to either	0 0 1 0 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 1 0 1 r/m	data	data if s:w=01
Immediate from accumulator	0 0 1 0 1 1 0 w	data	data if w=1	

SBB = Subtract with borrow:				
Reg./memory and register to either	0 0 0 1 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 0 1 1 r/m	data	data if s:w=01
Immediate from accumulator	0 0 0 1 1 1 0 w	data	data if w=1	

DEC = Decrement:	76543210	76543210	76543210	76543210
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 1 r/m		
Register	0 1 0 0 1 reg			
NEG = Change sign	1 1 1 1 0 1 1 w	mod 0 1 1 r/m		

CMP = Compare:				
Register/memory and register	0 0 1 1 1 0 d w	mod reg r/m		
Immediate with register/memory	1 0 0 0 0 0 s w	mod 1 1 1 r/m	data	data if s:w=01
Immediate with accumulator	0 0 1 1 1 1 0 w	data	data if w=1	
AAS = ASCII adjust for subtract	0 0 1 1 1 1 1 1			
DAS = Decimal adjust for subtract	0 0 1 0 1 1 1 1			
MUL = Multiply (unsigned)	1 1 1 1 0 1 1 w	mod 1 0 0 r/m		
IMUL = Integer multiply (signed)	1 1 1 1 0 1 1 w	mod 1 0 1 r/m		
AAM = ASCII adjust for multiply	1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0		
DIV = Divide (unsigned)	1 1 1 1 0 1 1 w	mod 1 1 0 r/m		
IDIV = Integer divide (signed)	1 1 1 1 0 1 1 w	mod 1 1 1 r/m		
AAD = ASCII adjust for divide	1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0		
CBW = Convert byte to word	1 0 0 1 1 0 0 0			
CWD = Convert word to double word	1 0 0 1 1 0 0 1			

Logic	76543210	76543210	76543210	76543210
NOT = Invert	1111011w	mod 010 r/m		
SHL/SAL = Shift logical/arithmetic left	110100vw	mod 100 r/m		
SHR = Shift logical right	110100vw	mod 101 r/m		
SAR = Shift arithmetic right	110100vw	mod 111 r/m		
ROL = Rotate left	110100vw	mod 000 r/m		
ROR = Rotate right	110100vw	mod 001 r/m		
RCL = Rotate through carry flag left	110100vw	mod 010 r/m		
RCR = Rotate through carry flag right	110100vw	mod 011 r/m		
AND = And:				
Reg./memory and register to either	001000dw	mod reg r/m		
Immediate to register/memory	1000000w	mod 100 r/m	data	data if w=1
Immediate to accumulator	0010010w	data	data if w=1	
TEST = And function to flags, no result:				
Register/memory and register	1000010w	mod reg r/m		
Immediate data and register/memory	1111011w	mod 000 r/m	data	data if w=1
Immediate data and accumulator	1010100w	data	data if w=1	
OR = Or:				
Reg./memory and register to either	000010dw	mod reg r/m		
Immediate to register/memory	1000000w	mod 001 r/m	data	data if w=1
Immediate to accumulator	0000110w	data	data if w=1	
XOR = Exclusive Or:				
Reg./memory and register to either	001100dw	mod reg r/m		
Immediate to register/memory	1000000w	mod 110 r/m	data	data if w=1
Immediate to accumulator	0011010w	data	data if w=1	

String Manipulation	**76543210**	**76543210**	**76543210**
REP = Repeat	1111001z		
MOVS = Move byte/word	1010010w		
CMPS = Compare byte/word	1010011w		
SCAS = Scan byte/word	1010111w		
LODS = Load byte/word to AL/AX	1010110w		
STOS = Store byte/word from AL/A	1010101w		

Control Transfer

CALL = Call:			
Direct within segment	11101000	disp-low	disp-high
Indirect within segment	11111111	mod 010 r/m	
Direct intersegment	10011010	offset-low	offset-high
		seg-low	seg-high
Indirect intersegment	11111111	mod 011 r/m	

JMP = Unconditional jump:			
Direct within segment	11101001	disp-low	disp-high
Direct within segment short	11101011	disp	
Indirect within segment	11111111	mod 100 r/m	
Direct intersegment	11101010	offset-low	offset-high
		seg-low	seg-high
Indirect intersegment	11111111	mod 101 r/m	

RET = Return from CALL:	**76543210**	**76543210**	**76543210**
Within segment	11000011		
Within seg. adding immediate to SP	11000010	data-low	data-high
Intersegment	11001011		
Intersegment adding immediate to SP	11001010	data-low	data-high
JE/JZ = Jump on equal/zero	01110100	disp	
JL/JNGE = Jump on less/not greater or equal	01111100	disp	
JLE/JNG = Jump on less or equal/not greater	01111110	disp	
JB/JNAE = Jump on below/not above or equal	01110010	disp	
JBE/JNA = Jump on below or equal/ not above	01110110	disp	
JP/JPE = Jump on parity/parity even	01111010	disp	
JO = Jump on overflow	01110000	disp	
JS = Jump on sign	01111000	disp	
JNE/JNZ = Jump on not equal/not zero	01110101	disp	
JNL/JGE = Jump on not less/greater or equal	01111101	disp	
JNLE/JG = Jump on not less or equal/ greater	01111111	disp	
JNB/JAE = Jump on not below/above or equal	01110011	disp	
JNBE/JA = Jump on not below or equal/above	01110111	disp	
JNP/JPO = Jump on not parity/parity odd	01111011	disp	
JNO = Jump on not overflow	01110001	disp	
JNS = Jump on not sign	01111001	disp	
LOOP = Loop CX times	11100010	disp	
LOOPZ/LOOPE = Loop while zero/equal	11100001	disp	
LOOPNZ/LOOPNE = Loop while not zero/equal	11100000	disp	
JCXZ = Jump on CX zero	11100011	disp	

INT = Interrupt	**76543210**	**76543210**
Type specified	11001101	type
Type 3	11001100	
INTO = Interrupt on overflow	11001110	
IRET = Interrupt return	11001111	

Processor Control

CLC = Clear carry	11111000	
CMC = Complement carry	11110101	
STC = Set carry	11111001	
CLD = Clear direction	11111100	
STD = Set direction	11111101	
CLI = Clear interrupt	11111010	
STI = Set interrupt	11111011	
HLT = Halt	11110100	
WAIT = Wait	10011011	
ESC = Escape (to external device)	11011xxx	mod x x x r/m
LOCK = Bus lock prefix	11110000	

Notes:

AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value.
Greater = more positive;
Less = less positive (more negative) signed values
if d = 1 then "to" reg; if d = 0 then "from" reg
if w = 1 then word instruction; if w = 0 then byte instruction
if s:w = 01 then 16-bits of immediate data from the operand
it s:w = 11 then an immediate data byte is sign extended to form the 16-bit operand
if v = 0 then "count" = 1; if v = 1 then "count" in (CL)
x = don't care
z is used for string primitives for comparsion with ZF FLAG

Segment Override Prefix

0 0 1 reg 1 1 0

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp high is absent
if mod = 10 then DISP = disp-high: disp low

if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) +DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)

* except if mod = 00 and r/m = 110 then EA = disp-high:disp-low.

REG is assigned according to the following table

16-bit (w=1)	8-bit (w=0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instruction which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:

FLAGS = X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

Absolute Maximum Ratings

Ambient temperature under bias	0 to 70°C
Storage temperature	−65 to +150°C
Voltage on any pin with respect to ground	−1.0 to +7V
Power dissipation	2.5 W

Note:
Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

SAB 8088: T_A = 0 to 70°C, V_{CC} = 5V ±10%
SAB 8088-2: T_A = 0 to 70°C, V_{CC} = 5V ± 5%
SAB 8088-1: T_A = 0 to 70°C, V_{CC} = 5V ± 5%

Parameter	Symbol	Limit values		Unit	Test conditions
		min.	max.		
Input low voltage	V_{IL}	−0.5	+0.8	V	1)
Input high voltage	V_{IH}	2.0	V_{CC}+0.5	V	1) 2)
Output low voltage	V_{OL}	–	0.45	V	I_{OL} = 2.0 mA
Output high voltage	V_{OH}	2.4	–	V	I_{OH} = −400 µA
Power supply current	I_{CC}	–	340	mA	All outputs open T_A = 25°C
Input leakage current	I_{LI}	–	±10	µA	0 V ≤ V_{IN} ≤ V_{CC}
Output leakage current	I_{LO}	–	±10	µA	0.45 V ≤ V_{OUT} ≤ V_{CC}
Clock input low voltage	V_{CL}	−0.5	+0.6	V	–
Clock input high voltage	V_{CH}	3.9	V_{CC}+1.0	V	–
Capacitance of input buffer (all inputs except AD0 to AD7, RQ/GT)	C_{IN}	–	15	pF	f_c = 1 MHz
Capacitance of I/O buffer (AD0 to AD7, RQ/GT)	C_{IO}	–	15	pF	f_c = 1 MHz

1) V_{IL} tested with MN/$\overline{MX}$ in = 0 V
V_{IH} tested with MN/$\overline{MX}$ in = 5 V
MN/$\overline{MX}$ pin is a strap pin.

2) Not applicable to $\overline{RG}/\overline{GT0}$ and $\overline{RG}/\overline{GT1}$ pins (pins 30 and 31)

AC Characteristics for SAB 8088/8088-2

SAB 8088: T_A = 0 to 70°C, V_{CC} = 5V ± 10%
SAB 8088-2: T_A = 0 to 70°C, V_{CC} = 5V ± 5%

Minimum Complexity System Timing Requirements

Parameter	Symbol	Limit values				Unit	Test conditions
		SAB 8088		SAB 8088-2			
		min.	max.	min.	max.		
CLK cycle period	t_{CLCL}	200	500	125	500	ns	–
CLK low time	t_{CLCH}	118	–	68	–	ns	–
CLK high time	t_{CHCL}	69	–	44	–	ns	–
CLK rise time	t_{CH1CH2}	–	10	–	10	ns	from 1.0 to 3.5V
CLK fall time	t_{CL2CL1}	–	10	–	10	ns	from 3.5 to 1.0V
Data in setup time	t_{DVCL}	30	–	20	–	ns	–
Data in hold time	t_{CLDX}	10	–	10	–	ns	–
RDY setup time into SAB 8284A/8284B[1) 2)]	t_{R1VCL}	35	–	35	–	ns	–
RDY hold time into SAB 8284A/8284B [1) 2)]	t_{CLR1X}	0	–	0	–	ns	–
READY setup time into SAB 8088	t_{RYHCH}	118	–	68	–	ns	–
READY hold time into SAB 8088	t_{CHRYX}	30	–	20	–	ns	–
READY inactive to CLK [3)]	t_{RYLCL}	−8	–	−8	–	ns	–
HOLD setup time	t_{HVCH}	35	–	20	–	ns	–
INTR, NMI, $\overline{\text{TEST}}$ setup time [2)]	t_{INVCH}	30	–	15	–	ns	–
Input rise time (except CLK)	t_{ILIH}	–	20	–	20	ns	from 0.8 to 2.0V
Input fall time (except CLK)	t_{IHIL}	–	12	–	12	ns	from 2.0 to 0.8V

[1)] Signal at SAB 8284A/8284B shown for reference only.
[2)] Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
[3)] Applies only to T2 state (8 ns into T3).

Timing Responses

Parameter	Symbol	Limit values				Unit	Test conditions
		SAB 8088		SAB 8088-2			
		min.	max.	min.	max.		
Address valid delay	t_{CLAV}	10	110	10	60	ns	[1]
Address hold time	t_{CLAX}	10	–	10	–	ns	[1]
Address float delay	t_{CLAZ}	t_{CLAX}	80	t_{CLAX}	50	ns	[1]
ALE width	t_{LHLL}	$t_{CLCH}-20$	–	$t_{CLCH}-10$	–	ns	[1]
ALE active delay	t_{CLLH}	–	80	–	50	ns	[1]
ALE inactive delay	t_{CHLL}	–	85	–	55	ns	[1]
Address hold time to ALE inactive	t_{LLAX}	$t_{CHCL}-10$	–	$t_{CHCL}-10$	–	ns	[1]
Data valid delay	t_{CLDV}	10	110	10	60	ns	[1]
Data hold time	t_{CHDX}	10	–	10	–	ns	[1]
Data hold time after $\overline{WR}$	t_{WHDX}	$t_{CLCH}-30$	–	$t_{CLCH}-30$	–	ns	[1]
Control active delay 1	t_{CVCTV}	10	110	10	70	ns	[1]
Control active delay 2	t_{CHCTV}	10	110	10	60	ns	[1]
Control inactive delay	t_{CVCTX}	10	110	10	70	ns	[1]
Address float to READ active	t_{AZRL}	0	–	0	–	ns	[1]
$\overline{RD}$ active delay	t_{CLRL}	10	165	10	100	ns	[1]
$\overline{RD}$ inactive delay	t_{CLRH}	10	150	10	80	ns	[1]
$\overline{RD}$ inactive to next address active	t_{RHAV}	$t_{CLCL}-45$	–	$t_{CLCL}-40$	–	ns	[1]
HLDA valid delay	t_{CLHAV}	10	160	10	100	ns	[1]
$\overline{RD}$ width	t_{RLRH}	$2\,t_{CLCL}-75$	–	$2\,t_{CLCL}-50$	–	ns	[1]
$\overline{WR}$ width	t_{WLWH}	$2\,t_{CLCL}-60$	–	$2\,t_{CLCL}-40$	–	ns	[1]
Address valid to ALE low	t_{AVAL}	$t_{CLCH}-60$	–	$t_{CLCH}-40$	–	ns	[1]
Output rise time	t_{OLOH}	–	20	–	20	ns	from 0.8 to 2.0V
Output fall time	t_{OHOL}	–	12	–	12	ns	from 2.0 to 0.8V

[1] C_L = 20–100 pF for all SAB 8088 outputs in addition to the internal loads

Maximum Mode System (using SAB 8288/8288A bus controller) Timing Requirements

Parameter	Symbol	Limit values				Unit	Test conditions
		SAB 8088		SAB 8088-2			
		min.	max.	min.	max.		
CLK cycle period	t_{CLCL}	200	500	125	500	ns	–
CLK low time	t_{CLCH}	118	–	68	–	ns	–
CLK high time	t_{CHCL}	69	–	44	–	ns	–
CLK rise time	t_{CH1CH2}	–	10	–	10	ns	from 1.0 to 3.5V
CLK fall time	t_{CL2CL1}	–	10	–	10	ns	from 3.5 to 1.0V
Data in setup time	t_{DVCL}	30	–	20	–	ns	–
Data in hold time	t_{CLDX}	10	–	10	–	ns	–
RDY setup time into SAB 8284A/8284B [1) 2)]	t_{R1VCL}	35	–	35	–	ns	–
RDY hold time into SAB 8284A/8284B [1) 2)]	t_{CLR1X}	0	–	0	–	ns	–
READY setup time into SAB 8088	t_{RYHCH}	118	–	68	–	ns	–
READY hold time into SAB 8088	t_{CHRYX}	30	–	20	–	ns	–
READY inactive to CLK [3)]	t_{RYLCL}	−8	–	−8	–	ns	–
Setup time for recognition (INTR, NMI, $\overline{TEST}$) [2)]	t_{INVCH}	30	–	15	–	ns	–
$\overline{RQ}/\overline{GT}$ setup time	t_{GVCH}	30	–	15	–	ns	–
$\overline{RQ}$ hold time into SAB 8088	t_{CHGX}	40	–	30	–	ns	–
Input rise time (except CLK)	t_{ILIH}	–	20	–	20	ns	from 0.8 to 2.0V
Input fall time (except CLK)	t_{IHIL}	–	12	–	12	ns	from 2.0 to 0.8V

1) Signal at SAB 8284A/8284B or SAB 8288/8288A shown for reference only.
2) Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3) Applies only to T2 state (8 ns into T3).

Timing Responses

Parameter	Symbol	Limit values				Unit	Test conditions
		SAB 8088		SAB 8088-2			
		min.	max.	min.	max.		
Command Active Delay [1]	t_{CLML}	10	35	10	35	ns	[3]
Command Inactive Delay [1]	t_{CLMH}	10	35	10	35	ns	[3]
READY Active to status passive [2]	t_{RYHSH}	–	110	–	65	ns	[3]
Status active delay	t_{CHSV}	10	110	10	60	ns	[3]
Status inactive delay	t_{CLSH}	10	130	10	70	ns	[3]
Address valid delay	t_{CLAV}	10	110	10	60	ns	[3]
Address hold time	t_{CLAX}	10	–	10	–	ns	[3]
Address float delay	t_{CLAZ}	t_{CLAX}	80	t_{CLAX}	50	ns	[3]
Status valid to ALE high [1]	t_{SVLH}	–	20	–	20	ns	[3]
Status valid to MCE high [1]	t_{SVMCH}	–	20	–	20	ns	[3]
CLK low to ALE valid [1]	t_{CLLH}	–	20	–	20	ns	[3]
CLK low to MCE high [1]	t_{CLMCH}	–	20	–	20	ns	[3]
ALE inactive delay [1]	t_{CHLL}	4	15	4	15	ns	[3]
Data valid delay	t_{CLDV}	10	110	10	60	ns	[3]
Data hold time	t_{CHDX}	10	–	10	–	ns	[3]
Control active delay [1]	t_{CVNV}	5	45	5	45	ns	[3]
Control inactive delay [1]	t_{CVNX}	10	45	10	45	ns	[3]

[1] Signal at SAB 8284A/8284B or SAB 8288/8288A shown for reference only.
[2] Applies only to T2 state (8 ns into T3).
[3] C_L = 20–100 pF for all SAB 8088 outputs in addition to the internal loads.

Timing Responses (cont'd)

Parameter	Symbol	Limit values				Unit	Test conditions
		SAB 8088		SAB 8088-2			
		min.	max.	min.	max.		
Address float to READ active	t_{AZRL}	0	–	0	–	ns	[2]
$\overline{RD}$ active delay	t_{CLRL}	10	165	10	100	ns	[2]
$\overline{RD}$ inactive delay	t_{CLRH}	10	150	10	80	ns	[2]
$\overline{RD}$ inactive to next address active	t_{RHAV}	$t_{CLCL}-45$	–	$t_{CLCL}-40$	–	ns	[2]
Direction control active delay [1]	t_{CHDTL}	–	50	–	50	ns	[2]
Direction control inactive delay [1]	t_{CHDTH}	–	30	–	30	ns	[2]
$\overline{GT}$ active delay	t_{CLGL}	–	85	–	50	ns	[2]
$\overline{GT}$ inactive delay	t_{CLGH}	–	85	–	50	ns	[2]
$\overline{RD}$ width	t_{RLRH}	$2\,t_{CLCL}-75$	–	$2\,t_{CLCL}-50$	–	ns	[2]
Output rise time	t_{OLOH}	–	20	–	20	ns	from 0.8 to 2.0V
Output fall time	t_{OHOL}	–	12	–	12	ns	from 2.0 to 0.8V

[1] Signal at SAB 8284A/8284B or SAB 8288/8288A shown for reference only.
[2] C_L = 20–100 pF for all SAB 8088 outputs in addition to the internal loads.

AC Characteristics for SAB 8088-1

SAB 8088-1: T_A = 0 to 70°C, V_{CC} = 5V ± 5%

Minimum Complexity System Timing Requirements

Parameter	Symbol	Limit values		Unit	Test conditions
		min.	max.		
CLK cycle period	t_{CLCL}	100	500	ns	–
CLK low time	t_{CLCH}	53	–	ns	–
CLK high time	t_{CHCL}	39	–	ns	–
CLK rise time	t_{CH1CH2}	–	10	ns	from 1.0 to 3.5V
CLK fall time	t_{CL2CL1}	–	10	ns	from 3.5 to 1.0V
Data in setup time	t_{DVCL}	5	–	ns	–
Data in hold time	t_{CLDX}	10	–	ns	–
RDY setup time into SAB 8284A/8284B [1] [2]	t_{R1VCL}	35	–	ns	–
RDY hold time into SAB 8284A/8284B [1] [2]	t_{CLR1X}	0	–	ns	–
READY setup time into SAB 8088	t_{RYHCH}	53	–	ns	–
READY hold time into SAB 8088	t_{CHRYX}	20	–	ns	–
READY inactive to CLK [3]	t_{RYLCL}	−10	–	ns	–
HOLD setup time	t_{HVCH}	20	–	ns	–
INTR, NMI, $\overline{\text{TEST}}$ setup time [2]	t_{INVCH}	15	–	ns	–
Input rise time (except CLK)	t_{ILIH}	–	20	ns	from 0.8 to 2.0V
Input fall time (except CLK)	t_{IHIL}	–	12	ns	from 2.0 to 0.8V

[1] Signal at SAB 8284A/8284B shown for reference only.
[2] Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
[3] Applies only to T2 state (8 ns into T3).

Timing Responses SAB 8088-1

Parameter	Symbol	Limit values		Unit	Test conditions
		min.	max.		
Address valid delay	t_{CLAV}	10	50	ns	[1]
Address hold time	t_{CLAX}	10	–	ns	[1]
Address float delay	t_{CLAZ}	10	40	ns	[1]
ALE width	t_{LHLL}	$t_{CLCH}-10$	–	ns	[1]
ALE active delay	t_{CLLH}	–	40	ns	[1]
ALE inactive delay	t_{CHLL}	–	45	ns	[1]
Address hold time to ALE inactive	t_{LLAX}	$t_{CHCL}-10$	–	ns	[1]
Data valid delay	t_{CLDV}	10	50	ns	[1]
Data hold time	t_{CHDX}	10	–	ns	[1]
Data hold time after $\overline{WR}$	t_{WHDX}	$t_{CLCH}-25$	–	ns	[1]
Control active delay 1	t_{CVCTV}	10	50	ns	[1]
Control active delay 2	t_{CHCTV}	10	45	ns	[1]
Control inactive delay	t_{CVCTX}	10	50	ns	[1]
Address float to READ active	t_{AZRL}	0	–	ns	[1]
$\overline{RD}$ active delay	t_{CLRL}	10	70	ns	[1]
$\overline{RD}$ inactive delay	t_{CLRH}	10	60	ns	[1]
$\overline{RD}$ inactive to next address active	t_{RHAV}	$t_{CLCL}-35$	–	ns	[1]
HLDA valid delay	t_{CLHAV}	10	60	ns	[1]
$\overline{RD}$ width	t_{RLRH}	$2\,t_{CLCL}-40$	–	ns	[1]
$\overline{WR}$ width	t_{WLWH}	$2\,t_{CLCL}-35$	–	ns	[1]
Address valid to ALE low	t_{AVAL}	$t_{CLCH}-35$	–	ns	[1]
Output rise time	t_{OLOH}	–	20	ns	from 0.8 to 2.0V
Output fall time	t_{OHOL}	–	12	ns	from 2.0 to 0.8V

[1] C_L = 20–100 pF for all SAB 8088 outputs in addition to the internal loads.

Maximum Mode System (using SAB 8288/8288A bus controller) Timing Requirements SAB 8088-1

Parameter	Symbol	Limit values		Unit	Test conditions
		min.	max.		
CLK cycle period	t_{CLCL}	100	500	ns	–
CLK low time	t_{CLCH}	53	–	ns	–
CLK high time	t_{CHCL}	39	–	ns	–
CLK rise time	t_{CH1CH2}	–	10	ns	from 1.0 to 3.5V
CLK fall time	t_{CL2CL1}	–	10	ns	from 3.5 to 1.0V
Data in setup time	t_{DVCL}	5	–	ns	–
Data in hold time	t_{CLDX}	10	–	ns	–
RDY setup time into SAB 8284A/8284B [1] [2]	t_{R1VCL}	35	–	ns	–
RDY hold time into SAB 8284A/8284B [1] [2]	t_{CLR1X}	0	–	ns	–
READY setup time into SAB 8088	t_{RYHCH}	53	–	ns	–
READY hold time into SAB 8088	t_{CHRYX}	20	–	ns	–
READY inactive to CLK [3]	t_{RYLCL}	−10	–	ns	–
Setup time for recognition ($\overline{INTR}$, NMI, $\overline{TEST}$) [2]	t_{INVCH}	15	–	ns	–
$\overline{RQ}/\overline{GT}$ setup time	t_{GVCH}	12	–	ns	–
$\overline{RQ}$ hold time into SAB 8088	t_{CHGX}	20	–	ns	–
Input rise time (except CLK)	t_{ILIH}	–	20	ns	from 0.8 to 2.0V
Input fall time (except CLK)	t_{IHIL}	–	12	ns	from 2.0 to 0.8V

[1] Signal at SAB 8284A/8284B or SAB 8288/8288A shown for reference only.
[2] Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
[3] Applies only to T2 state (8 ns into T3).

Timing Responses SAB 8088-1

Parameter	Symbol	Limit values		Unit	Test conditions
		min.	max.		
Command active delay [1])	t_{CLML}	10	35	ns	[3]
Command inactive delay [1])	t_{CLMH}	10	35	ns	[3]
READY active to status passive [2])	t_{RYHSH}	–	45	ns	[3]
Status active delay	t_{CHSV}	10	45	ns	[3]
Status inactive delay	t_{CLSH}	10	55	ns	[3]
Address valid delay	t_{CLAV}	10	50	ns	[3]
Address hold time	t_{CLAX}	10	–	ns	[3]
Address float delay	t_{CLAZ}	10	40	ns	[3]
Status valid to ALE high [1])	t_{SVLH}	–	20	ns	[3]
Status valid to MCE high [1])	t_{SVMCH}	–	20	ns	[3]
CLK low to ALE valid [1])	t_{CLLH}	–	20	ns	[3]
CLK low to MCE high [1])	t_{CLMCH}	–	20	ns	[3]
ALE inactive delay [1])	t_{CHLL}	4	15	ns	[3]
Data valid delay	t_{CLDV}	10	50	ns	[3]
Data hold time	t_{CHDX}	10	–	ns	[3]
Control active delay [1])	t_{CVNV}	5	45	ns	[3]
Control inactive delay [1])	t_{CVNX}	10	45	ns	[3]

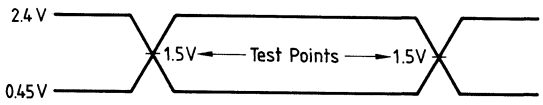
[1]) Signal at SAB 8284A/8284B or SAB 8288/8288A shown for reference only.
[2]) Applies only to T2 state (8 ns into T3).
[3]) C_L = 20–100 pF for all SAB 8088 outputs in addition to the internal loads.

Timing Responses SAB 8088-1 (cont'd)

Parameter	Symbol	Limit values		Unit	Test conditions
		min.	max.		
Address float to READ active	t_{AZRL}	0	–	ns	2)
$\overline{RD}$ active delay	t_{CLRL}	10	70	ns	2)
$\overline{RD}$ inactive delay	t_{CLRH}	10	60	ns	2)
$\overline{RD}$ inactive to next address active	t_{RHAV}	$t_{CLCL}-35$	–	ns	2)
Direction control active delay 1)	t_{CHDTL}	–	50	ns	2)
Direction control inactive delay 1)	t_{CHDTH}	–	30	ns	2)
$\overline{GT}$ active delay	t_{CLGL}	0	45	ns	2)
$\overline{GT}$ inactive delay	t_{CLGH}	0	45	ns	2)
$\overline{RD}$ width	t_{RLRH}	$2\,t_{CLCL}-40$	–	ns	2)
Output rise time	t_{OLOH}	–	20	ns	from 0.8 to 2.0V
Output fall time	t_{OHOL}	–	12	ns	from 2.0 to 0.8V

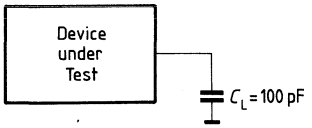
1) Signal at SAB 8284A/8284B or SAB 8288/8288A shown for reference only.
2) C_L = 20–100 pF for all SAB 8088 outputs in addition to the internal loads.

Input/Output Waveforms for AC Tests



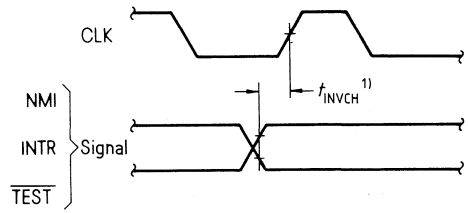
AC Testing: Inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0". The clock is driven at 4.3V and 0.25V. Timing measurements are made at 1.5V for both a logic "1" and "0".

Load Circuit for AC Tests



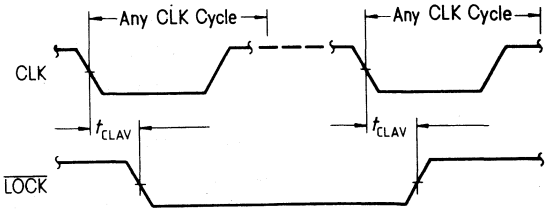
C_L includes Jig Capacitance

Asynchronous Signal Recognition

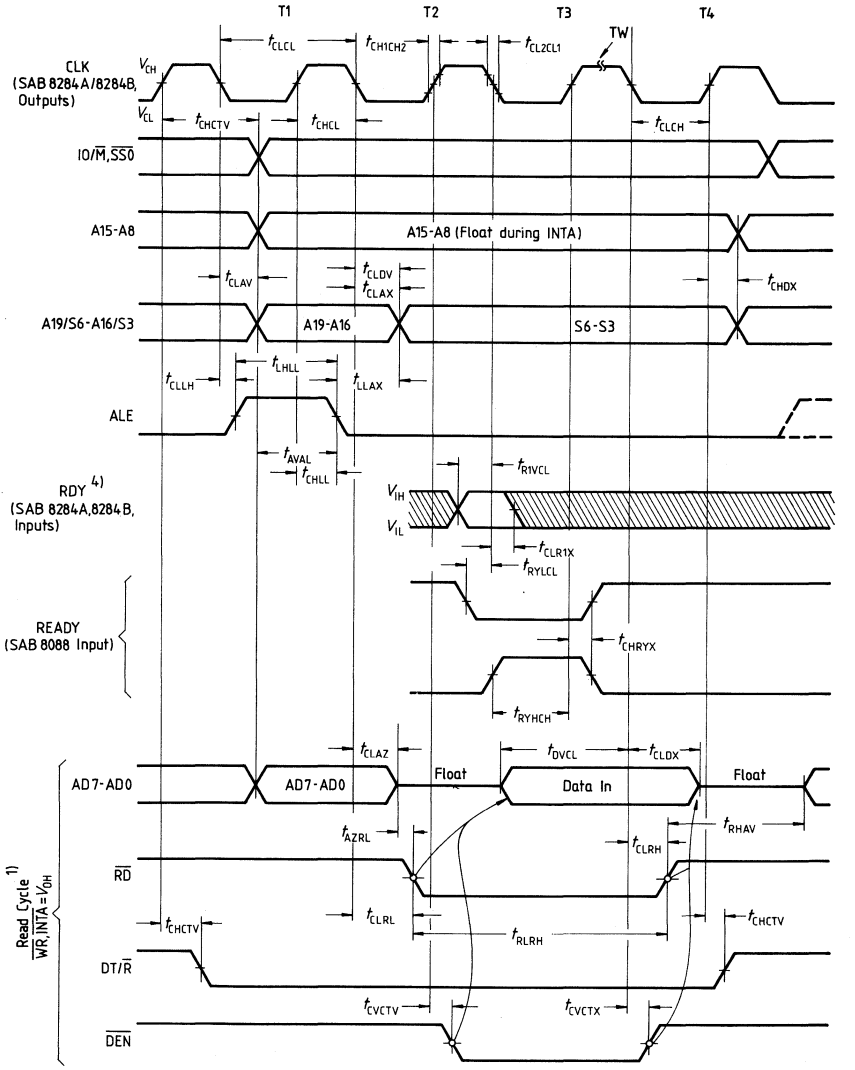


1) Setup requirements for asynchronous signals only to guarantee recognition at next CLK

Bus Lock Signal Timing (Maximum Mode only)

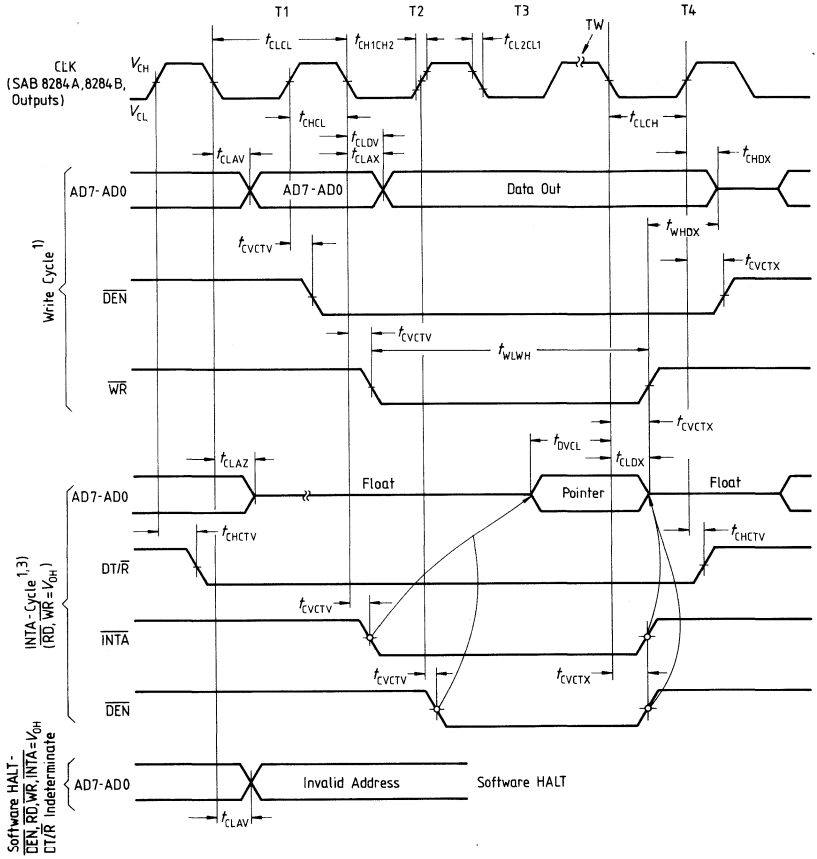


Bus Timing – Minimum Mode System



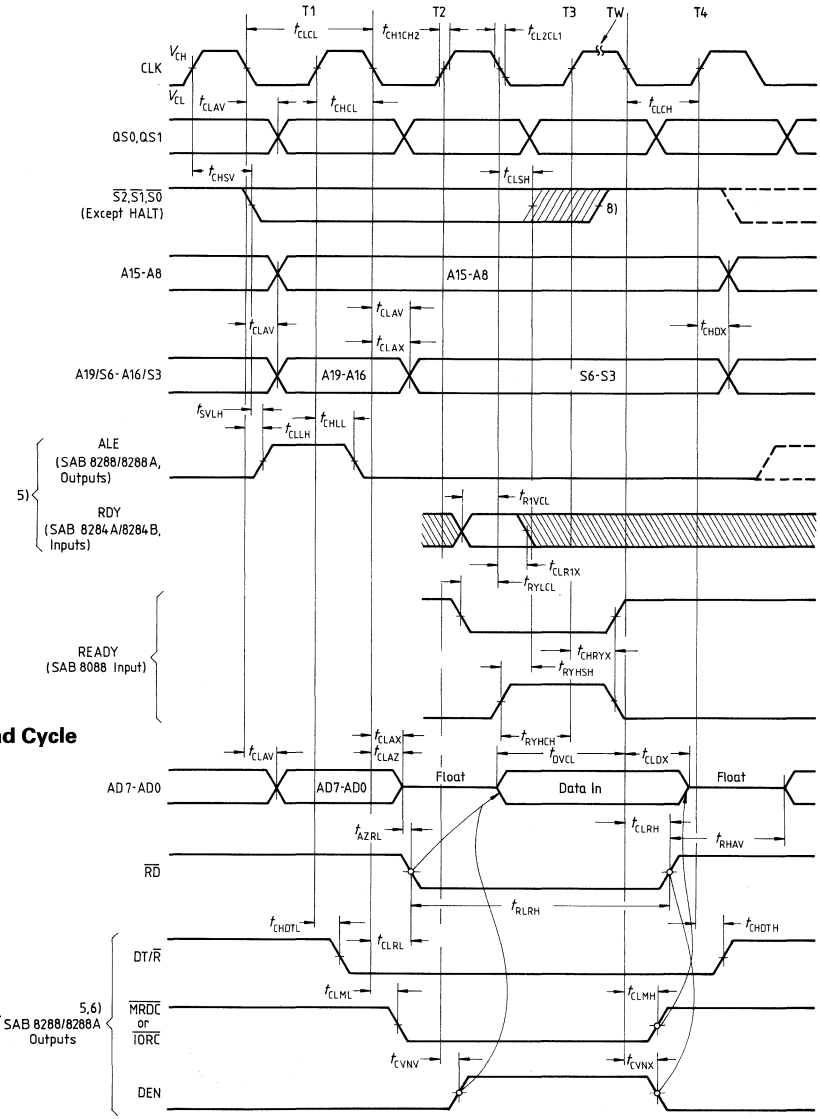
Notes see next page

Bus Timing – Minimum Mode System (cont'd)



1) All signals switch between V_{OH} and V_{OL} unless otherwise specified.
2) RDY is sampled near the end of T2, T3, TW to determine if TW machine states are to be inserted.
3) Two INTA cycles run back to back. The SAB 8088 local ADDR/DATA bus is floating during both INTA cycles. Control signals shown for second INTA cycle.
4) Signals at SAB 8284A/8284B are shown for reference only.
5) All timing measurements are made at 1.5V unless otherwise noted.

Bus Timing – Maximum Mode System (using SAB 8288/8288A)

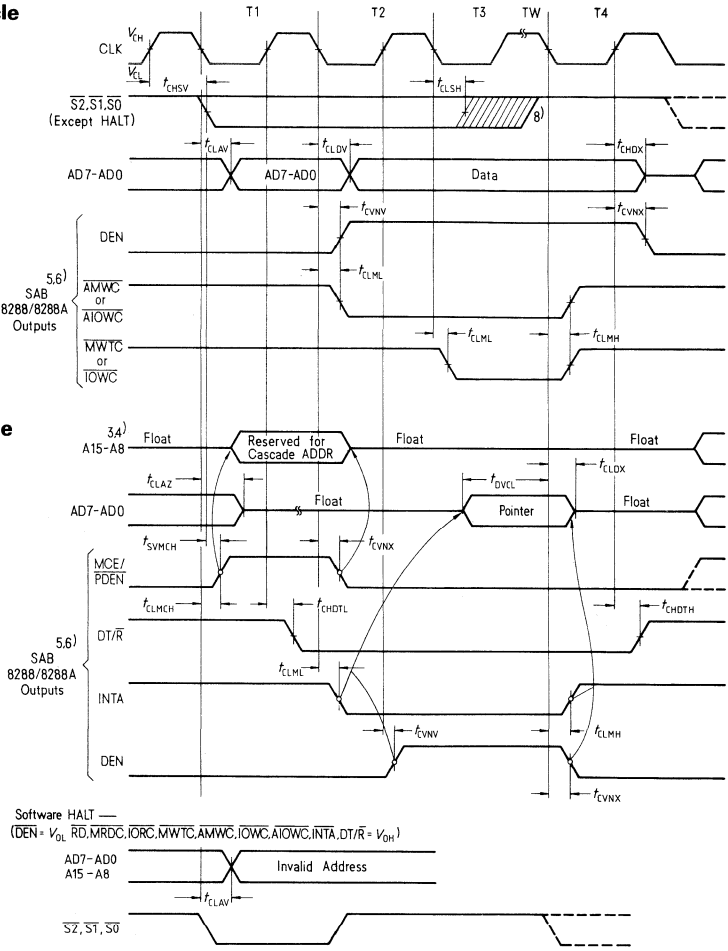


Read Cycle

Notes see next page.

Bus Timing – Maximum Mode System (using SAB 8288/8288A)

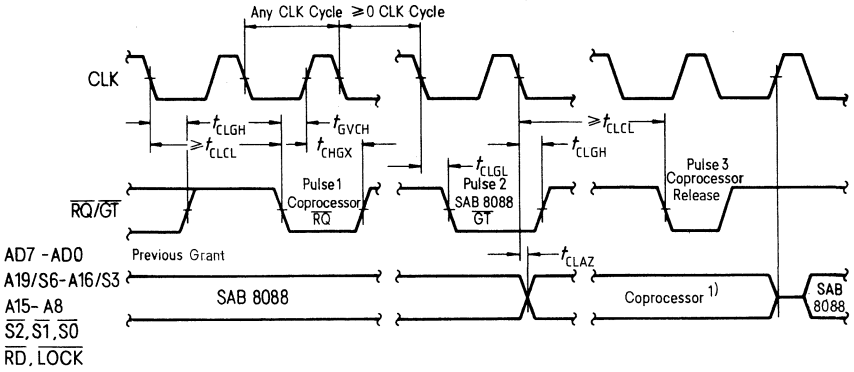
Write Cycle



INTA Cycle

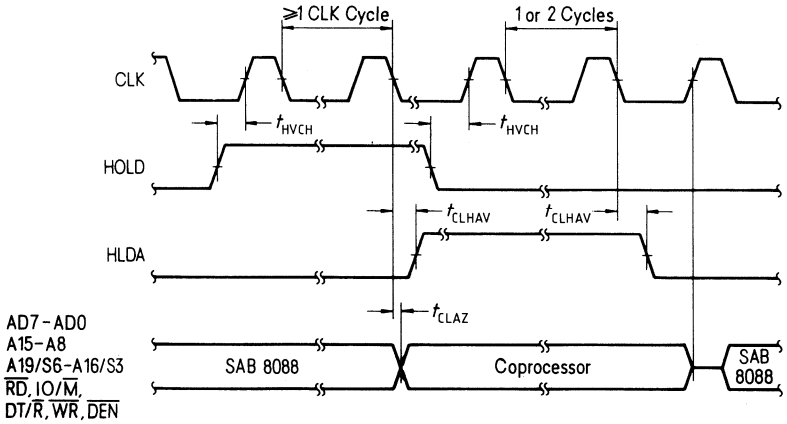
1) All signals switch between V_{OH} and V_{OL} unless otherwise specified.
2) RDY is sampled near the end of T2, T3, TW to determine if TW machine states are to be inserted.
3) Cascade address is valid between first and second INTA cycle.
4) Two INTA cycles run back-to-back. The SAB 8088 local ADDR/DATA bus is floating during both INTA cycles. Control for pointer address is shown for second INTA cycle.
5) Signals at SAB 8284A/8284B or SAB 8288/8288A are shown for reference only.
6) The issuance of the SAB 8288/8288A command and control signals ($\overline{MRDC}$, $\overline{MWTC}$, $\overline{AMWC}$, $\overline{IORC}$, $\overline{IOWC}$, $\overline{AIOWC}$, $\overline{INTA}$ and DEN) lags the active high SAB 8288/8288A DEN.
7) All timing measurements are made at 1.5 V unless otherwise noted.
8) Status inactive in state just prior to T4.

Request/Grant Sequence Timing (Maximum Mode only)



1) The coprocessor may not drive the buses outside the region shown without risking contention

Hold/Hold Acknowledge Timing (Minimum Mode only)



Ordering Information

Type	Ordering code	Function
SAB 8088-P	Q67120-C106	8-bit microprocessor – 5 MHz (P-DIP-40)
SAB 8088-2-P	Q67120-C213	8-bit microprocessor – 8 MHz (P-DIP-40)
SAB 8088-1-P	Q67120-C249	8-bit microprocessor – 10 MHz (P-DIP-40)
SAB 8088-N	Q67120-C301	8-bit microprocessor – 5 MHz (PL-CC-44)
SAB 8088-2-N	Q67120-C302	8-bit microprocessor – 8 MHz (PL-CC-44)
SAB 8088-1-N	Q67120-C321	8-bit microprocessor – 10 MHz (PL-CC-44)

SIEMENS

High-Integration 16-Bit Microprocessor

SAB 80186

Preliminary

SAB 80186 8 MHz

- Integrated feature set
  - enhanced SAB 8086-2 CPU
  - clock generator
  - 2 independent high-speed DMA channels
  - programmable interrupt controller
  - 3 programmable 16-bit timers
  - programmable memory and peripheral chip-select logic
  - programmable wait state generator
  - local bus controller
- High-performance processor
  - twice the performance of the standard SAB 8086 at 8 MHz
  - 4 Mbyte/s bus bandwidth interface at 8 MHz

SAB 80186-1 10 MHz

- Direct addressing capability to 1 Mbyte of memory
- Completely object-code compatible with all existing SAB 8086/8088 software
  - 10 new instruction types
- Optional numerical processor extension
- Compatible with the bus support components SAB 8282A/8283A/8286A/8287A and SAB 8288A/8289
- Compatible with industry standard 80186 processor
- Available in 10 MHz (SAB 80186-1) and 8 MHz (SAB 80186) versions

The SAB 80186 is a highly integrated 16-bit microprocessor implemented in +5V advanced Siemens MYMOS technology. It effectively combines 15 to 20 of the most common SAB 8086 system components onto one chip. The 8 MHz SAB 80186 provides twice the throughput of the standard 5 MHz SAB 8086. The SAB 80186 is upward-compatible with SAB 8086 and SAB 8088 software, and adds 10 new instruction types to the existing set. The SAB 80186 comes in a 68-pin plastic leaded chip carrier (PL-CC-68) package and requires single +5V power supply.

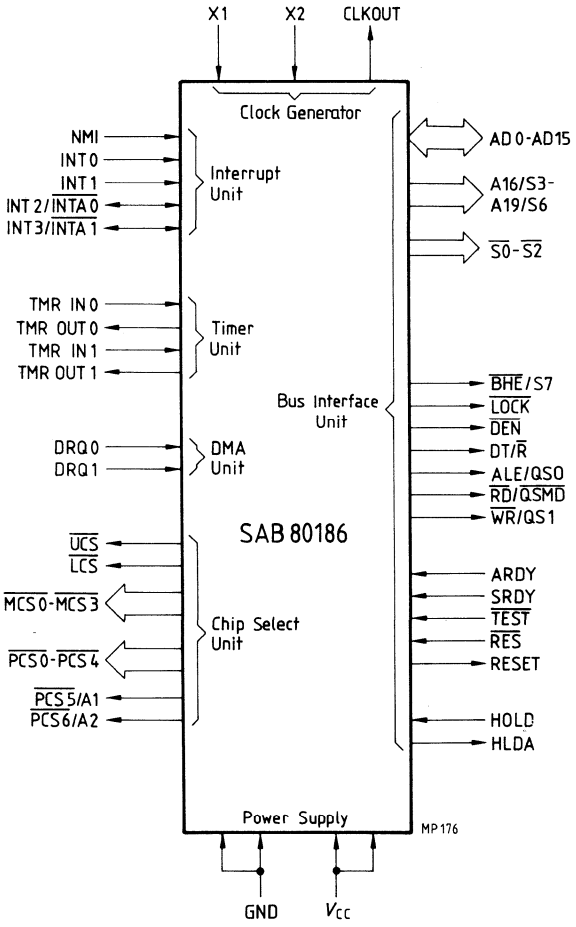
Ordering Information

Type	Ordering code	Function
SAB 80186-N	Q67120-C250	16-bit microprocessor, 8 MHz (PL-CC)
SAB 80186-1-N	Q67120-C306	16-bit microprocessor, 10 MHz (PL-CC)

Pin Names

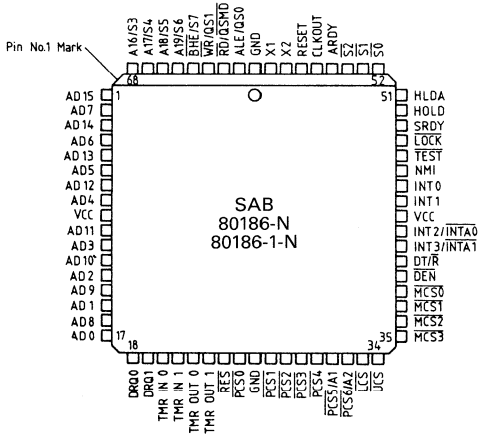
AD0–AD15	Address/Data Bus
A16/S3–A19/S6	Address Bus/Status Lines
S0–S2	Status Lines
$\overline{BHE}$/S7	Bus High Enable/Status Line
$\overline{LOCK}$	Bus Lock
$\overline{DEN}$	Data Enable
DT/$\overline{R}$	Data Transmit/Receive
ALE/QS0	Address Latch Enable/Queue Status
$\overline{RD}$/QSMD	Read Strobe/Queue Status Mode
$\overline{WR}$/QS1	Write Strobe/Queue Status
ARDY	Asynchronous Ready
SRDY	Synchronous Ready
$\overline{TEST}$	Test Line
$\overline{RES}$	Reset Input
RESET	Reset Output
HOLD	Hold Request
HLDA	Hold Acknowledge
NMI	Non-Maskable Interrupt
INT0, INT1	Interrupt Request
INT2/INTA0	Interrupt Request/Acknowledge
INT3/INTA1	Interrupt Request/Acknowledge
TMR IN0/TMR OUT0	Timer 0 Input/Output
TMR IN1/TMR OUT1	Timer 1 Input/Output
DRQ0, DRQ1	Data Request
$\overline{UCS}$, $\overline{LCS}$	Upper/Lower Chip Select
$\overline{MCS0}$–$\overline{MCS3}$	Memory Chip Select
$\overline{PCS0}$–$\overline{PCS4}$	Peripheral Chip Select
$\overline{PCS5}$/A1, $\overline{PCS6}$/A2	Peripheral Chip Select/Address
VCC	Power Supply (+5 V)
GND	Ground (0 V)

Logic Symbol



Pin Configuration

PL-CC Package (top view)



Pin Definitions and Functions

Symbol	Pin	Input (I) Output (O)	Function
AD15–AD0	10–17, 1–8	I/O	**ADDRESS/DATA BUS (0 TO 15)** These signals constitute the time-multiplexed memory or I/O address (T1) and data (T2, T3, TW, and T4) bus. The bus is active high. A0 is analogous to $\overline{BHE}$ for the lower byte of the data bus, pins D7 through D0. It is low during T1 when a byte is to be transferred onto the lower portion of the bus in memory or I/O operations.
DRQ0 DRQ1	18 19	I I	**DMA REQUEST** Is driven high by an external device when it desires that a DMA channel (channel 0 or 1) performs a transfer. These signals are active high, level-triggered, and internally synchronized.
TMR IN 0, TMR IN 1	20 21	I I	**TIMER INPUTS** Are used either as clock or control signals, depending upon the programmed timer mode. These inputs are active high (or low-to-high transitions are counted) and internally synchronized.
TMR OUT 0, TMR OUT 1	22 23	O O	**TIMER OUTPUTS** Are used to provide single pulse or continuous waveform generation, depending upon the timer mode selected.
$\overline{RES}$	24	I	**$\overline{SYSTEM\ RESET}$** Causes the SAB 80186 to immediately terminate its present activity, clear the internal logic, and enter a dormant state. This signal may be asynchronous to the SAB 80186 clock. The SAB 80186 begins fetching instructions approximately 7 clock cycles after $\overline{RES}$ is returned high. $\overline{RES}$ is required to be low for greater than 4 clock cycles and is internally synchronized. For proper initialization, the low-to-high transition of $\overline{RES}$ must occur no sooner than 50 microseconds after power up. This input is provided with a Schmitt trigger to facilitate power-on $\overline{RES}$ generation via an RC network. When $\overline{RES}$ occurs, the SAB 80186 will drive the status lines to an inactive level for one clock, and then tristate them.
$\overline{PCS0}$ $\overline{PCS1-4}$	25 27, 28, 29, 30	O O	**$\overline{PERIPHERAL\ CHIP\ SELECT\ 0\ TO\ 4}$** These signals are active low when a reference is made to the defined peripheral area (64 Kbyte I/O space). These lines are not tristated during bus hold. The address ranges activating $\overline{PCS0-4}$ are software-programmable.
$\overline{PCS5}$/A1	31	O	**$\overline{PERIPHERAL\ CHIP\ SELECT\ 5}$/LATCHED A1** May be programmed to provide a sixth peripheral chip select, or to provide an internally latched A1 signal. The address range activating $\overline{PCS5}$ is software-programmable. When programmed to provide latched A1, rather than $\overline{PCS5}$, this pin will retain the previously latched value of A1 during a bus HOLD. A1 is active high.
$\overline{PCS6}$/A2	32	O	**$\overline{PERIPHERAL\ CHIP\ SELECT\ 6}$/LATCHED A2** May be programmed to provide a seventh peripheral chip select, or to provide an internally latched A2 signal. The address range activating $\overline{PCS6}$ is software-programmable. When programmed to provide latched A2, rather than $\overline{PCS6}$, this pin will retain the previously latched value of A2 during a bus HOLD. A2 is active high.

Pin Definitions and Functions (cont'd)

Symbol	Pin	Input (I) Output (O)	Function
$\overline{LCS}$	33	O	**LOWER MEMORY CHIP SELECT** Is active low whenever a memory reference is made to the defined lower portion (1 K to 256 K) of memory. This line is not tristated during bus HOLD. The address range activating $\overline{LCS}$ is software-programmable.
$\overline{UCS}$	34	O	**UPPER MEMORY CHIP SELECT** Is an active low output whenever a memory reference is made to the defined upper portion (1 K to 256 K block) of memory. This line is not tristated during bus HOLD. The address range activating $\overline{UCS}$ is software-programmable.
$\overline{MCS0-3}$	38, 37, 36, 35	O	**MIDRANGE MEMORY CHIP SELECT 0 TO 3** These signals are active low when a memory reference is made to the defined midrange portion of memory (8 K to 512 K). These lines are not tristated during bus HOLD. The address ranges activating $\overline{MCS0-3}$ are software-programmable.
$\overline{DEN}$	39	O	**DATA ENABLE** Is provided as an SAB 8286A/8287A data bus transceiver output enable. $\overline{DEN}$ is active low during each memory and I/O access. $\overline{DEN}$ is high whenever DT/$\overline{R}$ changes its state.
DT/$\overline{R}$	40	O	**DATA TRANSMIT/$\overline{RECEIVE}$** Controls the direction of data flow through the external SAB 8286A/8287A data bus transceiver. When low data is transferred to the SAB 80186. When high the SAB 80186 places write data on the data bus.
INT0, INT1, INT2/$\overline{INTA0}$ INT3/$\overline{INTA1}$	45, 44 42 41	I I/O I/O	**MASKABLE INTERRUPT REQUEST** Can be requested by strobing one of these pins. When configured as inputs, these pins are active high. Interrupt requests are synchronized internally. INT2 and INT3 may be configured via software to provide active low interrupt-acknowledge output signals. All interrupt inputs may be configured via software to be either edge or level-triggered. To ensure recognition, all interrupt requests must remain active until the interrupt is acknowledged. When iRMX mode is selected, the function of these pins changes (see Interrupt Controller section of this data sheet).
NMI	46	I	**NON-MASKABLE INTERRUPT** Is an edge-triggered input which causes a type 2 interrupt. NMI is not maskable internally. A transition from low to high initiates the interrupt at the next instruction boundary. NMI is latched internally. An NMI duration of one clock or more will guarantee service. This input is internally synchronized.
$\overline{TEST}$	47	I	**$\overline{TEST}$** Is examined by the WAIT instruction. If the $\overline{TEST}$ input is high when "WAIT" execution begins, instruction execution will suspend. $\overline{TEST}$ will be resampled until it goes low, at which time execution will be resumed. If interrupts are enabled while the SAB 80186 is waiting for $\overline{TEST}$, interrupts will be serviced. This input is synchronized internally.

Pin Definitions and Functions (cont'd)

Symbol	Pin	Input (I) Output (O)	Function
$\overline{LOCK}$	48	O	**$\overline{LOCK}$** This output indicates that other system bus masters are not to gain control of the system bus while $\overline{LOCK}$ is active low. The $\overline{LOCK}$ signal is requested by the LOCK prefix instruction and is activated at the beginning of the first data cycle associated with the instruction following the $\overline{LOCK}$ prefix. It remains active until the completion of the instruction following the $\overline{LOCK}$ prefix. No pre-fetches will occur while $\overline{LOCK}$ is asserted. $\overline{LOCK}$ is active low, is driven high for one clock during reset, and then tristated.
SRDY	49	I	**SYNCHRONOUS READY** Must be synchronized externally to the SAB 80186. The use of SRDY provides a relaxed system-timing specification on the ready input. This is accomplished by eliminating the one-half clock cycle which is required for internally resolving the signal level when using the ARDY input. This line is active high. If this line is connected to VCC no wait states are inserted. Asynchronous ready (ARDY) or synchronous ready (SRDY) must be active before a bus cycle is terminated. If unused, this line should be tied low.
HOLD HLDA	50 51	I O	**HOLD/HOLD ACKNOWLEDGE** Indicates that another bus master is requesting the local bus. The HOLD input is active high. HOLD may be asynchronous with respect to the SAB 80186 clock. The SAB 80186 will issue a HLDA (high) in response to a HOLD request at the end of T4 or T1. Simultaneous with issuing HLDA, the SAB 80186 will tristate the local bus and control lines. After HOLD is detected as being low, the SAB 80186 will lower HLDA. When the SAB 80186 needs to run another bus cycle, it will again drive the local bus and control lines.
$\overline{S0}$, $\overline{S1}$, $\overline{S2}$	52–54	O	**BUS CYCLE STATUS** $\overline{S0}$ to $\overline{S2}$ are encoded to provide bus transaction information: SAB 80186 Bus Cycle Status Information

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	Bus Cycle Initiated
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt
1	0	0	Instruction Fetch
1	0	1	Read Data from Memory
1	1	0	Write Data to Memory
1	1	1	Passive (no bus cycle)

The status pins are tristated during "HOLD."
$\overline{S2}$ may be used as a logical M/$\overline{IO}$ indicator, and $\overline{S1}$ as a DT/$\overline{R}$ indicator.
The status lines are driven high for one clock during reset, and then tristated until a bus cycle begins.

Pin Definitions and Functions (cont'd)

Symbol	Pin	Input (I) Output (O)	Function
ARDY	55	I	**ASYNCHRONOUS READY** Informs the SAB 80186 that the addressed memory space or I/O device will complete a data transfer. The ARDY input pin will accept an asynchronous input and is active high. Only the rising edge is internally synchronized by the SAB 80186. This means that the falling edge of ARDY must be synchronized to the SAB 80186 clock. If connected to VCC no wait states are inserted. Asynchronous ready (ARDY) or synchronous ready (SRDY) must be active to terminate a bus cycle. If unused, this line should be tied low.
CLKOUT	56	O	**CLOCK OUTPUT** Provides the system with a 50% duty cycle waveform. All device pin timings are specified relative in CLKOUT.
RESET	57	O	**RESET** This output indicates that the SAB 80186 CPU is being reset, and can be used as a system reset. It is active high, synchronized with the processor clock, and lasts an integer number of clock periods corresponding to the length of the $\overline{RES}$ signal.
X1, X2	59, 58	I	**CRYSTAL INPUTS** X1 and X2 provide an external connection for a fundamental-mode parallel resonant crystal for the internal crystal oscillator. X1 can interface to an external clock instead of a crystal. In this case, minimize the capacitance on X2 or drive X2 with complemented X1. The input or oscillator frequency is internally divided by two to generate the clock signal (CLKOUT).
ALE/QS0	61	O	**ADDRESS LATCH ENABLE/QUEUE STATUS 0** Is provided by the SAB 80186 to latch the address into the SAB 8282A/8283A address latches. ALE is active high. Addresses are guaranteed to be valid on the trailing edge of ALE. The ALE rising edge is generated off the rising edge of the CLKOUT immediately preceding T1 of the associated bus cycle, effectively half a clock cycle earlier than in the standard SAB 8086. The trailling edge is generated off the CLKOUT rising edge in T1 like in the SAB 8086. Note that ALE is never tristated.
$\overline{RD}$/QSMD	62	O	**$\overline{READY\ STROBE}$** Indicates that the SAB 80186 is performing a memory or I/O read cycle. $\overline{RD}$ is active low for T2, T3 and TW of any read cycle. It is guaranteed not to go low in T2 until after the address bus is tristated. $\overline{RD}$ is active low and tristated during "HOLD". $\overline{RD}$ is driven high for one clock during reset, and then the output driver is tristated. A weak internal pullup mechanism on the $\overline{RD}$ line holds it high when the line is not driven. During reset, the pin is sampled to determine whether the SAB 80186 should provide ALE, $\overline{WR}$ and $\overline{RD}$, or if the queue status should be provided. $\overline{RD}$ should be connected to GND to provide queue status data.

Pin Definitions and Functions (cont'd)

$\overline{WR}$/QS1	63	O	**$\overline{WRITE\ STROBE}$/QUEUE STATUS 1** Indicates that the data on the bus is to be written into a memory or an I/O device. $\overline{WR}$ is active for T2, T3, and TW of any write cycle. It is active low and tristated during "HOLD." It is driven high for one clock during reset, and then tristated. When the SAB 80186 is in queue status mode, the ALE/QS0 and $\overline{WR}$/QS1 pins provide information about processor/instruction queue interaction.

QS1	QS0	Queue Operation
0	0	No Queue Operation
0	1	First Op Code Byte Fetched from the Queue
1	1	Subsequent Byte Fetched from the Queue
1	0	Empty the Queue

$\overline{BHE}$/S7	64	O	**$\overline{BUS\ HIGH\ ENABLE}$** During T1 this signal should be used to determine if data is to be enabled onto the most significant half of the data bus pins D15 to D8. $\overline{BHE}$ is low during T1 for read, write, and interrupt acknowledge cycles when a byte is to be transferred on the higher half of the bus. The S7 status information is available during T2, T3, and T4. S7 is logically equivalent to $\overline{BHE}$. The signal is active low, and is tristated off during bus HOLD.

$\overline{BHE}$ and A0 Encodings

$\overline{BHE}$ Value	A0 Value	Function
0	0	Word Transfer
0	1	Byte Transfer on Upper Half of Data Bus (D15 to D8)
1	0	Byte Transfer on Lower Half of Data Bus (D7 to D0)
1	1	Reserved

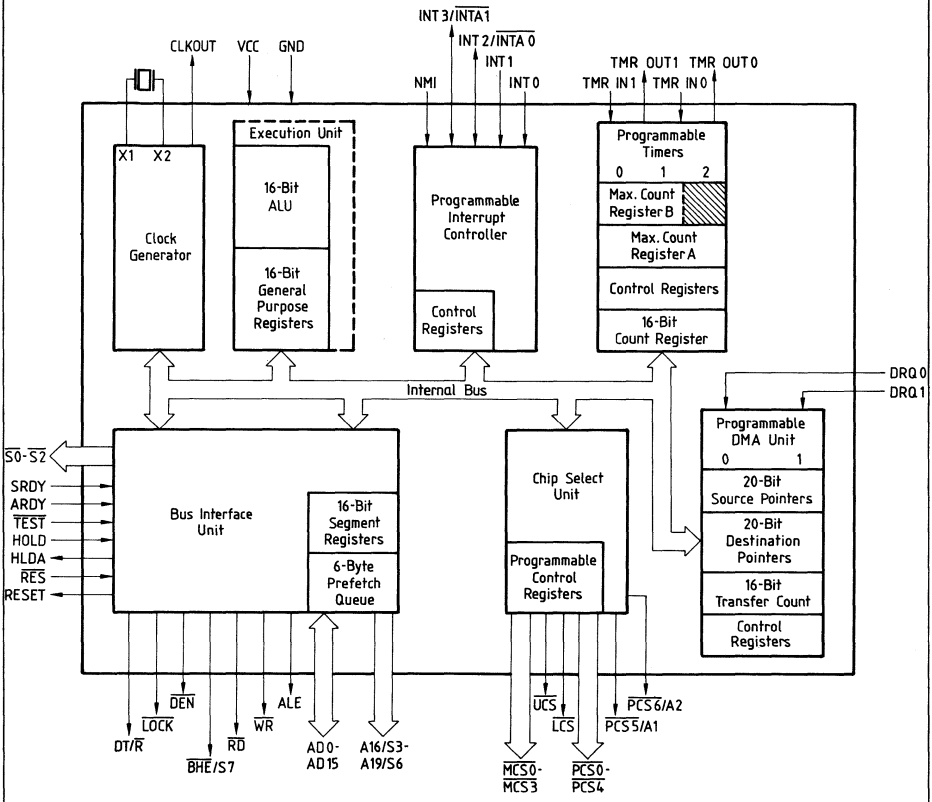
A19/S6, A18/S5, A17/S4, A16/S3	65 66 67 68	O O O O	**ADDRESS BUS OUTPUTS** (16 to 19) and **BUS CYCLE STATUS** (3 to 6) They reflect the four most significant address bits during T1. These signals are active high. During T2, T3, TW, and T4, status information is available on these lines as encoded below:

	Low	High
S6	Processor Cycle	DMA Cycle

S3, S4, and S5 are defined as being low during T2 to T4.

V_{CC}	9, 43	I	**POWER SUPPLY** (+5 V)
GND	26, 60	I	**GROUND** (0 V)

Block Diagram



Functional Description

The SAB 8086,8088, 80186, 80188 and 80286 families all contain the same basic set of registers, instructions, and addressing modes. The SAB 80186 processor is upward-compatible with the SAB 8086 and SAB 8088 CPUs.

Register Set

The SAB 80186 base architecture has fourteen registers. These registers are grouped into the following categories.

General registers

Eight 16-bit general-purpose registers may be used to contain arithmetic and logical operands. Four of these (AX, BX, CX, and DX) can be used as 16-bit registers or split into pairs of separate 8-bit registers.

Segment registers

Four 16-bit special-purpose registers select, at any time given, the segments of memory that are immediately addressable for code, stack, and data.

Base and index registers

Four of the general-purpose registers may also be used to determine offset addresses of operands in memory. These registers may contain base addresses or indexes to particular locations within a segment. The addressing mode selects the specific registers for operand and address calculations.

Status and control registers

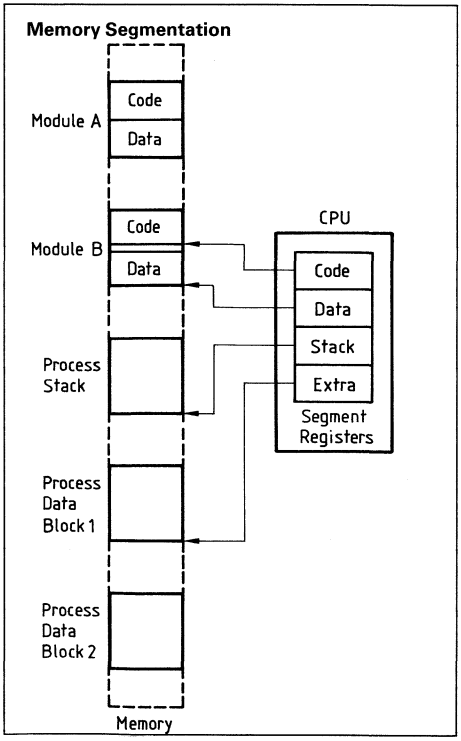
Two 16-bit special-purpose registers record or alter certain aspects of the SAB 80186 processor state. These are the instruction pointer register, which contains the offset address of the next sequential instruction to be executed, and the status word register, which contains status and control flag bits.

Memory Organization

Memory is organized in sets of segments. Each segment is a linear contiguous sequence of up to 64 K (2^{16}) 8-bit bytes. Memory is addressed using a two-component address (a pointer) that consists of a 16-bit base segment and a 16-bit offset. The 16-bit base values are contained in one of four internal segment registers (code, data, stack, extra). The physical address is calculated by shifting the base value left by four bits and adding the 16-bit offset value to yield a 20-bit physical address. This allows for a 1 Mbyte physical address size.

All instructions that address operands in memory must specify the base segment and the 16-bit offset value. For speed and compact instruction encoding, the segment register used for physical address generation is implied by the addressing mode used. These rules follow the way programs are written as independent modules that require areas for code and data, a stack, and access to external data areas.

Special segment override instruction prefixes allow the implicit segment register selection rules to be overridden for special cases. The stack, data, and extra segments may coincide for simple programs.



I/O Space

The I/O space consists of 64 K 8-bit or 32 K 16-bit ports. Separate instructions address the I/O space with either an 8-bit port address, specified in the instruction, or a 16-bit port address in the DX register. 8-bit port addresses are zero-extended such that A15 to A8 are low I/O port addresses 00F8(H) through 00FF(H) are reserved.

Interrupts

An interrupt transfers execution to a new program location. The old program address (CS:IP) and machine state (status word) are saved on the stack to allow resumption of the interrupted program. Interrupts fall into three classes: hardware-initiated, INT instructions, and instruction exceptions. Hardware-initiated interrupts occur in response to an external input and are classified as non-maskable or maskable.

Interrupt Sources

The SAB 80186 can service interrupts generated by software or hardware. The software interrupts are generated by specific instructions (INT, ESC, unused OP, etc.) or the results of conditions specified by instructions (array bounds check, INT0, DIV, IDIV, etc.). All interrupt sources are serviced by an indirect call through an element of a vector table. This vector table is indexed by using the interrupt vector type, multiplied by four. All hardware-generated interrupts are sampled at the end of each instruction. Thus, the software interrupts will begin service first. Once the service routine is entered and interrupts are enabled, any hardware source of sufficient priority can interrupt the service routine in progress.

Interrupt Vector Table

Interrupt Name	Vector Type	Default Priority [5]	Related Instructions
Divide Error Exception	0	1 [1]	DIV, IDIV
Single-Step Interrupt	1	12 [2] 2	All
NMI	2	1	All
Breakpoint Interrupt	3	1 [1]	INT
INT0 Detected Overflow Exception	4	1 [1]	INT0
Array Bounds Exception	5	1 [1]	BOUND
Unused Op Code Exception	6	1 [1]	Undefined Op Codes
ESC Op Code Exception	7	1 [1] [3]	ESC Op Codes
Timer 0 Interrupt	8	2A [4]	
Timer 1 Interrupt	18	2B [4]	
Timer 2 Interrupt	19	2C [4]	
Reserved	9	3	
DMA 0 Interrupt	10	4	
DMA 1 Interrupt	11	5	
INT0 Interrupt	12	6	
INT1 Interrupt	13	7	
INT2 Interrupt	14	8	
INT3 Interrupt	15	9	

1) These are generated as the result of an instruction execution.
2) This is handled as in the SAB 8086.
3) An escape op code will cause a trap only if the proper bit is set in the peripheral control block relocation register.
4) All three timers constitute one source of request to the interrupt controller. The timer interrupts all have the same default priority level with respect to all other interrupt sources. However, they have a defined priority ordering amongst themselves. (Priority 2A is higher priority than 2B.) Each timer interrupt has a separate vector type number.
5) Default priorities for the interrupt sources are used only if the user does not program each source into a unique priority level.

Initialization and Processor Reset

Processor initialization or startup is accomplished by driving the $\overline{RES}$ input pin low. $\overline{RES}$ forces the SAB 80186 to terminate all execution and local bus activity. No instruction or bus activity will occur as long as $\overline{RES}$ is active. After $\overline{RES}$ becomes inactive and an internal processing interval elapses, the SAB 80186 begins execution with the instruction at physical location FFFF0(H). $\overline{RES}$ also sets some registers to predefined values.

Initial Register State after RESET

Status Word	F002(H)
Instruction Pointer	0000(H)
Code Segment	FFFF(H)
Data Segment	0000(H)
Extra Segment	0000(H)
Stack Segment	0000(H)
Relocation Register	20FF(H)
UMCS	FFFB(H)

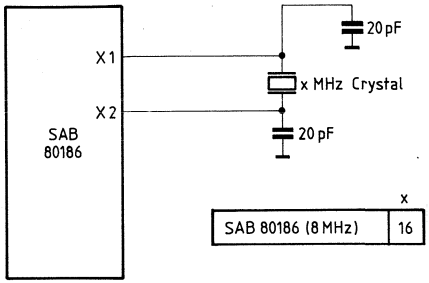
Clock Generator

The SAB 80186 provides an on-chip clock generator for both internal and external clock generation. The clock generator features a crystal oscillator, a divide-by-two counter, synchronous and asynchronous ready inputs and reset circuitry.

Oscillator

The oscillator circuit of the SAB 80186 is designed to be used with a parallel resonant fundamental mode crystal. This is used as the time base for the SAB 80186. The crystal frequency selected will be twice the CPU clock frequency. Use of an LC or RC circuit is not recommended with this oscillator. If an external oscillator is used, it can be connected directly to input pin X1 in lieu of a crystal. The output of the oscillator is not directly available outside the SAB 80186.

Recommended SAB 80186 Crystal Configuration



The following parameters may be used for choosing a crystal:

Temperature range	0 to 70 °C
ESR (equivalent series resistance)	30 Ω max.
C0 (shunt capacitance of crystal)	7.0 pF max.
C1 (load capacitance)	20 pF ± 2 pF
Drive level	1 mW max.

Ready Synchronization

The SAB 80186 provides both synchronous and asynchronous ready inputs. Asynchronous ready synchronization is accomplished by circuitry which samples ARDY in the middle of T2, T3 and again in the middle of each TW until ARDY is sampled high.

A second ready input (SRDY) is provided to interface with externally synchronized ready signals. This input is sampled at the end of T2,T3 and again at the end of each TW until it is sampled high.

Reset Logic

The SAB 80186 provides both a $\overline{RES}$ input pin and a synchronized reset pin for use with other system components. The $\overline{RES}$ input pin on the SAB 80186 is provided with hysteresis in order to facilitate power-on reset generation via an RC network.

Local Bus Controller

The SAB 80186 provides a local bus controller to generate the local bus control signals. In addition, it employs a HOLD/HLDA protocol for relinquishing the local bus to other bus masters. It also provides control lines that can be used to enable external buffers and to direct the flow of data on and off the local bus.

When the SAB 80186 relinquishes control of the local bus, it tristates $\overline{DEN}$, $\overline{RD}$, $\overline{WR}$, $\overline{S0}$ to $\overline{S2}$, $\overline{LOCK}$, AD0 to AD15, A16 to A19, $\overline{BHE}$, and DT/$\overline{R}$ to allow another master to drive these lines directly.

Local bus controller and reset

Upon receipt of a reset pulse from the $\overline{RES}$ input, the local bus controller will perform the following actions:

- Drive $\overline{DEN}$, $\overline{RD}$, and $\overline{WR}$ high for one clock cycle, then float.

  Note: $\overline{RD}$ is also provided with an internal pullup device to prevent the processor from inadvertently entering queue status mode during reset.
- Drive $\overline{S0}$ to $\overline{S2}$ to the passive state (all high) and then float.
- Drive $\overline{LOCK}$ high and then float.
- Tristate AD0 to AD15, A16 to A19, $\overline{BHE}$, DT/$\overline{R}$.
- Drive ALE low (ALE is never tristated).
- Drive HLDA low.

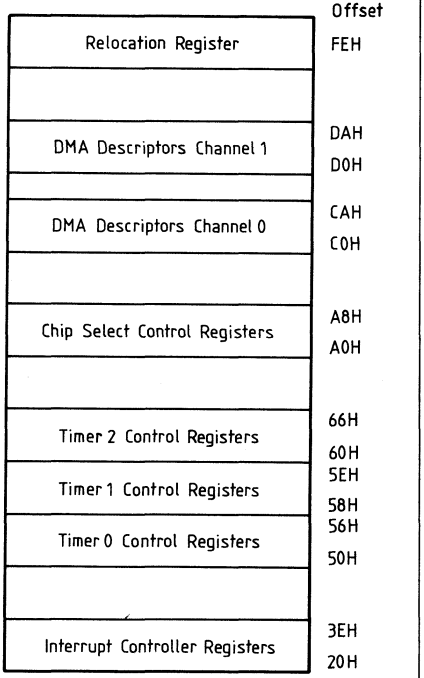
Internal Peripheral Interface

All the SAB 80186 integrated peripherals are controlled via 16-bit registers contained within an internal 256 byte control block. This control block may be mapped into either memory or I/O space. Internal logic will recognize the address and respond to the bus cycle. During bus cycles to internal registers, the bus controller will signal the operation externally (i.e., the $\overline{RD}$, $\overline{WR}$, status, address, data, etc., lines will be driven as in a normal bus cycle), but D15 to D0, SRDY, and ARDY will be ignored.

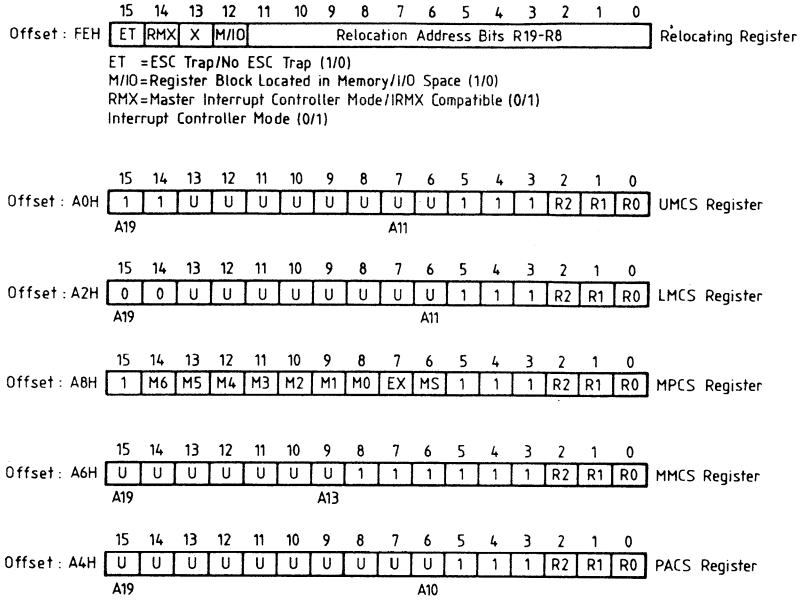
The control block base address is programmed via a 16-bit relocation register contained within the control block at offset FEH from the base address of the control block. It provides the upper 12 bits of the base address of the control block.

The integrated SAB 80186 peripherals operate semi-autonomously from the CPU. Access to them for the most part is via software read/write of the control and data locations in the control block. Most of these registers can be both read and written. A few dedicated lines, such as interrupts and DMA request provide real-time communication between the CPU and peripherals as in a more conventional system utilizing discrete peripheral blocks.

Internal Register Map



Chip Select Registers



UMCS Programming Values

Starting Address (Base Address)	Memory Block Size	UMCS Value (Assuming R0=R1=R2=0)
FFC00	1 K	FFF8H
FF800	2 K	FFB8H
FF000	4 K	FF38H
FE000	8 K	FE38H
FC000	16 K	FC38H
F8000	32 K	F838H
F0000	64 K	F038H
E0000	128 K	E038H
C0000	256 K	C038H

LMCS Programming Values

Upper Address	Memory Block Size	LMCS Value (Assuming R0=R1=R2=0)
003FFH	1 K	0038H
007FFH	2 K	0078H
00FFFH	4 K	00F8H
01FFFH	8 K	01F8H
03FFFH	16 K	03F8H
07FFFH	32 K	07F8H
0FFFFH	64 K	0FF8H
1FFFFH	128 K	1FF8H
3FFFFH	256 K	3FF8H

MPCS Programming Values

Total Block Sice	Individual Select Size	MPCS Bits 14 to 8
8 K	2 K	0000001B
16 K	4 K	0000010B
32 K	8 K	0000100B
64 K	16 K	0001000B
128 K	32 K	0010000B
256 K	64 K	0100000B
512 K	128 K	1000000B

MS, EX Programming Values

Bit	Description
MS	1=Peripherals Mapped into Memory Space. 0=Peripherals Mapped into I/O Space.
EX	0=5 $\overline{\text{PCS}}$ Lines. A1, A2 Provided. 1=7 $\overline{\text{PCS}}$ Lines. A1, A2 Are Not Provided.

Ready Bits Programming

R2	R1	R0	Number of Wait States Generated
0	0	0	0 Wait States, External RDY Also Used
0	0	1	1 Wait State Inserted, External RDY Also Used.
0	1	0	2 Wait States Inserted, External RDY Also Used.
0	1	1	3 Wait States Inserted, External RDY Also Used
1	0	0	0 Wait States, External RDY Ignored.
1	0	1	1 Wait State Inserted, External RDY Ignored.
1	1	0	2 Wait States Inserted, External RDY ignored.
1	1	1	3 Wait States Inserted, External RDY Ignored.

Chip Select Logic

The SAB 80186 contains logic which provides programmable chip select generation for both, memories and peripherals. In addition, it can be programmed to provide ready (or wait state) generation. It can also provide latched address bits A1 and A2. The chip select lines are active for all memory and I/O cycles in their programmed areas, whether they be generated by the CPU or by the integrated DMA unit.

Memory Chip Selects

The SAB 80186 provides 6 memory chip select outputs for 3 address areas: upper memory, lower memory, and midrange memory. One each is provided for upper memory and lower memory, while four are provided for midrange memory.

Upper Memory $\overline{CS}$

The SAB 80186 provides a chip select, called $\overline{UCS}$, for the top of memory. The top of memory is usually used as the system memory because after reset the SAB 80186 begins executing at memory location FFFF0H. After reset, the UMCS register is programmed for a 1 K area. It must be reprogrammed if a larger upper memory area is desired.

Lower Memory $\overline{CS}$

The SAB 80186 provides a chip select for low memory called $\overline{LCS}$. The bottom of memory contains the interrupt vector table, starting at location 00000H. After reset, the LMCS register value is undefined. However, the $\overline{LCS}$ chip select line will not become active until the LMCS register is accessed.

Midrange Memory $\overline{CS}$

The SAB 80186 provides four $\overline{MCS}$ lines which are active within a user-locatable memory block. This block can be located anywhere within the 1 Mbyte memory address space exclusive of the areas defined by $\overline{UCS}$ and $\overline{LCS}$. Both the base address and size of this memory block are programmable.

Each of the four chip select lines is active for one of the four equal contiguous divisions of the midrange block. Thus, if the total block size is 32 K, each chip select is active for 8 K of memory with $\overline{MCS0}$ being active for the first range and $\overline{MCS3}$ being active for the last range. After reset, the contents of both of these registers is undefined.

However, none of the $\overline{MCS}$ lines will be active until both the MMCS and MPCS registers are accessed.

Peripheral Chip Selects

The SAB 80186 can generate chip selects for up to seven peripheral devices. These chip selects are active for seven contiguous blocks of 128 bytes above a programmable base address. This base address may be located in either memory or I/O space.

Seven $\overline{CS}$ lines called $\overline{PCS0-6}$ are generated by the SAB 80186. The base address is user-programmable; however it can only be a multiple of 1 Kbytes, i.e. the least significant 10 bits of the starting address are always 0.

$\overline{PCS5}$ and $\overline{PCS6}$ can also be programmed to provide latched address bits A1, A2. If so programmed, they cannot be used as peripheral selects.

The starting address of the peripheral chip select block is defined by the PACS register. This register is located at offset A4H in the internal control block. Bits 15 to 6 of this register correspond to bits 19 to 10 of the 20-bit Programmable Base Address (PBA) of the peripheral chip-select block.

PCS Address Ranges

PCS Line	Active between Locations
PCS0	PBA – PBA+127
PCS1	PBA+128 – PBA+255
PCS2	PBA+256 – PBA+383
PCS3	PBA+384 – PBA+511
PCS4	PBA+512 – PBA+639
PCS5	PBA+640 – PBA+767
PCS6	PBA+768 – PBA+895

Ready Generation Logic

The SAB 80186 can generate a ready signal internally for each of the memory or peripheral $\overline{CS}$ lines. The number of wait states to be inserted for each peripheral or memory is programmable to provide 0 to 3 wait states for all accesses to the area for which the chip select is active. In addition, the SAB 80186 may be programmed to either ignore external ready for each chip select range individually or to factor external ready with the integrated ready generator.

Ready control consists of 3 bits for each $\overline{CS}$ line or group of lines generated by the SAB 80186.

Chip Select/Ready Logic and Reset

Upon reset, the chip select/ready logic will perform the following actions:

- All chip select outputs will be driven high.
- Upon leaving reset, the $\overline{UCS}$ line will be programmed to provide chip selects to a 1 K block with the accompanying ready control bits set at 011 to allow the maximum number of internal wait states in conjunction with external ready consideration (i.e. UMCS resets to FFFBH).
- No other chip select or ready control registers have any predefined values after reset. They will not become active until the CPU accesses their control registers. Both the PACS and MPCS registers must be accessed before the $\overline{PCS}$ lines will become active.

DMA Channels

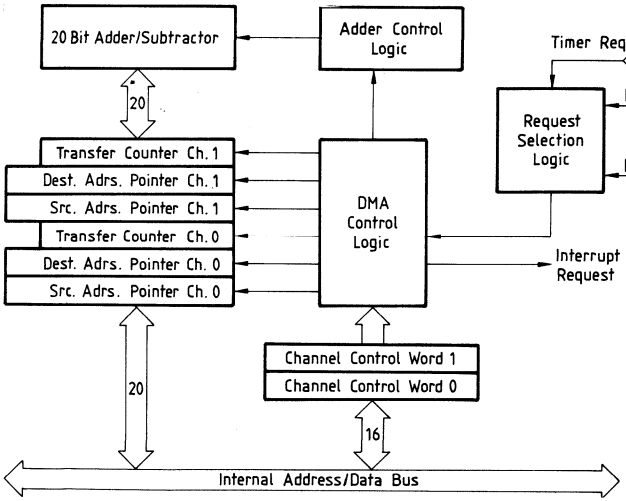
The SAB 80186 DMA controller provides two independent high-speed DMA channels. Data transfers can occur between memory and I/O spaces (e.g., Memory-to-I/O) or within the same space (e.g., Memory-to-Memory or I/O-to-I/O). Data can be transferred either in bytes (8 bits) or in words (16 bits) to or from even or odd addresses. Each DMA channel maintains both a 20-bit source and destination pointer which can be optionally incremented or decremented after each data transfer (by one or two depending on byte or word transfers). Each data transfer consumes 2 bus cycles (a minimum of 8 clocks), one cycle to fetch data and the other to store data. This provides a maximum data transfer rate of one Mword/s or 2 Mbytes/s.

DMA Channel Control Word Register

Each DMA channel control word determines the mode of operation for the particular SAB 80186 DMA channel.

The DMA channel control registers may be changed while the channel is operating. However, any changes made during operation will affect the current DMA transfer.

DMA Unit Block Diagram



DMA Control Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M/$\overline{IO}$	Destination DEC	INC	M/$\overline{IO}$	Source DEC	INC	TC	INT	SYN		P	TDRQ	X	CHG/$\overline{NOCHG}$	ST/$\overline{STOP}$	$\overline{B}$/W

X=Don't Care

DMA Memory Pointer Register Format

Higher Register Address	XXX	XXX	XXX	A19-A16
Lower Register Address	A15-A12	A11-A8	A7-A4	A3-A0

15 0

XXX=Don't Care

DMA Control Block Format

Register Name	Register Address	
	Chan. 0	Chan. 1
Control Word	CAH	DAH
Transfer Count	C8H	D8H
Destination Pointer (upper 4 bits)	C6H	D6H
Destination Pointer	C4H	D4H
Source Pointer (upper 4 bits)	C2H	D2H
Source Pointer	C0H	D0H

DMA Control Word Bit Description

$\overline{B}$/W — Byte/word (0/1) transfers.

ST/$\overline{STOP}$ — Start/stop (1/0) channel.

CHG/$\overline{NOCHG}$ — Change/do not change (1/0) ST/$\overline{STOP}$ bit. If this bit is set when writing to the control word, the ST/$\overline{STOP}$ bit will be programmed by the write to the control word. If this bit is cleared when writing the control word, the ST/$\overline{STOP}$ bit will not be altered. This bit is not stored; it will always be a 0 on read.

INT — Enable interrupts to CPU on tranfer count termination.

TC — If set, DMA will terminate when the contents of the transfer count register reach zero. The ST/$\overline{STOP}$ bit will also be reset at this point if TC is set. If this bit is cleared, the DMA unit will decrement the transfer count register for each DMA cycle, but the DMA transfer will not stop when the contents of the TC register reach zero.

SYN (2 bits) — 00 No synchronization.
Note: The ST bit will be cleared automatically when the contents of the TC register reach zero regardless of the state of the TC bit.
01 Source synchronization.
10 Destination synchronization.
11 Unused.

TDRQ — 0: Disable DMA requests from timer 2.
1: Enable DMA requests from timer 2.

Bit 3 — Bit 3 is not used.

Source: INC — Increment source pointer by 1 or 2 (depends on $\overline{B}$/W) after each transfer.

M/$\overline{IO}$ — Source pointer is in M/IO space (1/0).

DEC — Decrement source pointer by 1 or 2 (depends on $\overline{B}$/W) after each transfer.

Destin.: INC — Increment destination pointer by 1 or 2 ($\overline{B}$/W) after each transfer.

M/$\overline{IO}$ — Destination pointer is in M/IO space (1/0).

DEC — Decrement destination pointer by 1 or 2 (depending on $\overline{B}$/W) after each transfer.

P — Channel priority – relative to other channel.
0 low priority.
1 high priority.
Channels will alternate cycles if both set at same priority level.

DMA Destination and Source Pointer Register

Each DMA channel maintains a 20-bit source and a 20-bit destination pointer. Each of these pointers takes up two full 16-bit registers in the peripheral control block. The lower four bits of the upper register contain the upper four bits of the 20-bit physical address.

DMA Transfer Count Register

Each DMA channel maintains a 16-bit transfer count register (TC). This register is decremented after every DMA cycle, regardless of the state of the TC bit in the DMA control register.

DMA Requests

Data transfers may be either source or destination synchronized, that is either the source of the data or the destination of the data may request the data transfer. In addition, DMA transfers may be unsynchronized; that is, the transfer will take place continually until the correct number of transfers has occurred.

DMA Priority

The DMA channels may be programmed such that one channel is always given priority over the other, or they may be programmed such as to alternate cycles when both have DMA requests pending. DMA cycles always have priority over internal CPU cycles except between locked memory accesses or word accesses to odd memory locations; however, an external bus hold takes priority over an internal DMA cycle.

DMA Channels and Reset

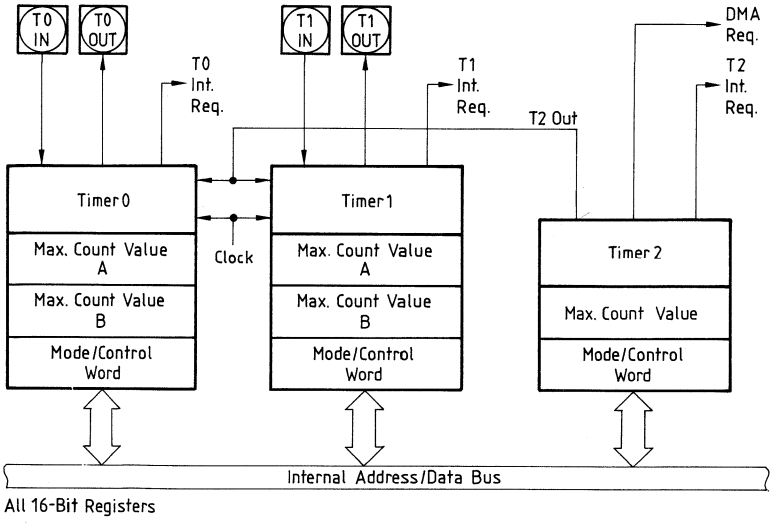
Upon reset, the DMA channels will perform the following actions:

- The start/stop bit for each channel will be reset to STOP.
- Any transfer in progress is aborted.

Timers

The SAB 80186 provides three internal 16-bit programmable timers. Two of these are highly flexible and are connected to four external pins (2 per timer). They can be used to count external events, time external events, generate nonrepetitive waveforms, etc. The third timer is not connected to any external pins, and is useful for real-time coding and time delay applications. In addition, this third timer can be used as a prescaler to the other two, or as a DMA request source.

Timer Block Diagram



Timer Mode/Control Register

15	14	13	12	11	.....	5	4	3	2	1	0
EN	$\overline{INH}$	INT	RIU	0	.....	MC	RTG	P	EXT	ALT	CONT

Timer Control Block Format

Register Name	Register Offset		
	TMR 0	TMR 1	TMR 2
Mode/Control Word	56H	5EH	66H
Max. Count B	54H	5CH	not present
Max. Count A	52H	5AH	62H
Count Register	50H	58H	60H

Timer Operation

The timers are controlled by eleven 16-bit registers in the internal peripheral control block. The count register contains the current value of the timer. It can be read or written at any time independent of whether the timer is running or not. The value of this register will be incremented for each timer event. Each of the timers is equipped with a max. count register, which defines the maximum count the timer will reach. After reaching the max. count register value, the timer count value will be reset to zero during that same clock.

Each timer gets serviced every fourth CPU clock cycle, and thus can operate at speeds up to one-quarter the internal clock frequency (one-eighth the crystal rate). External clocking of the timers may be done at up to a rate of one-quarter of the internal CPU clock rate (2 MHz for an 8 MHz CPU clock).

The timers have several programmable options.

- All three timers can be set to halt or continue on a terminal count.
- Timers 0 and 1 can select between internal and external clocks, alternate between max. count registers and be set to retrigger on external events.
- The timers may be programmed to cause an interrupt on terminal count.

Count Registers

Each of the three timers has a 16-bit count register. The current contents of this register may be read or written by the processor at any time. If the register is written into while the timer is counting, the new value will take effect in the current count cycle.

Max. Count Registers

Timers 0 and 1 have two max.count registers, while timer 2 has a single max.count register. These contain the number of events the timer will count. In timers 0 and 1, the max.count register used can alternate between the two max.count values whenever the current maximum count is reached.

Timer Mode/Control Register

The mode/control register allows the user to program the specific mode of operation or check the current programmed status for any of the three integrated timers.

ALT:
The ALT bit determines which of two max. count registers is used for count comparison. If ALT = 0, register A for that timer is always used, while if ALT = 1, the comparison will alternate between register A and register B when each maximum count is reached. This alternation allows the user to change one max.count register while the other is being used, and thus provides a method of generating nonrepetitive waveforms. Square waves and pulse outputs of any duty cycle are a subset of available signals obtained by not changing the final count registers. The ALT bit also determines the function of the timer output pin. If ALT is zero, the output pin will go low for one clock, the clock after the maximum count is reached. If ALT is one, the output pin will reflect the current max.count register being used (0/1 for B/A).

CONT:
Setting the CONT bit causes the associated timer to run continuously, while resetting it causes the timer to halt upon maximum count. If CONT = 0 and ALT = 1, the timer will count to the max. count register A value, be reset, count to the register B value, be reset, and halt.

EXT:
The external bit selects between internal and external clocking for the timer. The external signal may be asynchronous with respect to the SAB 80186 clock. If EXT is set, the timer will count low-to-high transitions on the input pin. If cleared, it will count an internal clock while using the input pin for control. In this mode, the function of the external pin is defined by the RTG bit. The maximum input-to-output transition latency time may be as much as 6 clocks. However, clock inputs may be pipelined as closely together as every 4 clocks without losing clock pulses.

P:
The prescaler bit is ignored unless internal clocking has been selected (EXT = 0). If the P bit is a zero, the timer will count at one-fourth the internal CPU clock rate. If the P bit is a one, the output of timer 2 will be used as a clock for the timer. Note that the user must initialize and start timer 2 to obtain the prescaled clock.

RTG:
Retrigger bit is only active for internal clocking (EXT = 0). In this case it determines the control function provided by the input pin.

If RTG = 0, the input level gates the internal clock on and off. If the input pin is high, the timer will count; if the input pin is low, the timer will hold its value. As indicated previously, the input signal may be asynchronous with respect to the SAB 80186 clock.

When RTG = 1, the input pin detects low-to-high transitions. The first transition starts the timer running, clearing the timer value to zero on the first clock, and then incrementing thereafter. Further transitions on the input pin will again reset the timer to zero, from which it will start counting up again. If CONT = 0, when the timer has reached maximum count, the EN bit will be cleared, inhibiting further timer activity.

EN:
The enable bit provides programmer control over the timer's RUN/HALT status. When set, the timer is enabled to increment subject to the input pin constraints in the internal clock mode (discussed previously). When cleared, the timer will be inhibited from counting. All input pin transitions during which the time EN is zero will be ignored. If CONT is zero, the EN bit is automatically cleared upon maximum count.

INH:
The inhibit bit allows for selective updating of the enable (EN) bit. If INH is a one during the write to the mode/control word, then the state of the EN bit will be modified by the write. If INH is a zero during the write, the EN bit will be unaffected by the operation. This bit is not stored; it will always be a 0 on a read.

INT:
When set, the INT bit enables interrupts from the timer, which will be generated on every terminal count. If the timer is configured in dual max.count register mode, an interrupt will be generated each time the value in max.count register A is reached, and each time the value in max.count register B is reached. If this enable bit is cleared after the interrupt request has been generated, but before a pending interrupt is serviced, the interrupt request will still be in force. (The request is latched in the interrupt controller.)

MC:
The maximum count bit is set whenever the timer reaches its final maximum count value. If the timer is configured in dual max.count register mode, this bit will be set each time the value in max.count register A is reached, and each time the value in max.count register B is reached. This bit is set regardless of the timer's interrupt enable bit. The MC bit gives the user the ability to monitor timer status through software instead of through interrupts.
Programmer intervention is required to clear this bit.

RIU:
The register in use bit indicates which max.count register is currently being used for comparison to the timer count value. A zero indicates register A. The RIU bit cannot be written, i.e. its value is not affected when the control register is written. It is always cleared when the ALT bit is zero.
Not all mode bits are provided for timer 2. Certain bits are hardwired as indicated below:

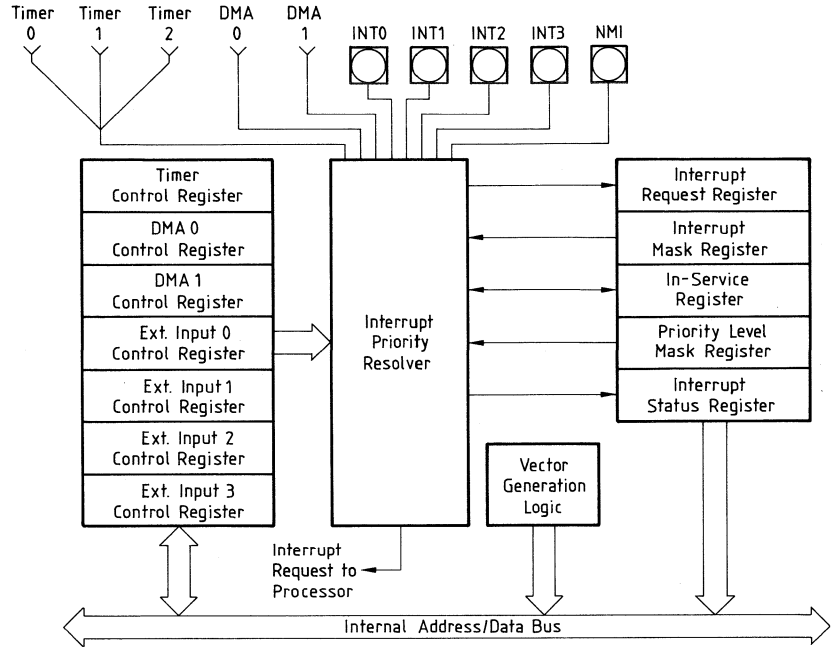
ALT = 0, EXT = 0, P = 0, RTG = 0, RIU = 0

Timers and Reset

Upon reset, the timers will perform the following actions:

- All EN (enable) bits are reset preventing timer counting.
- All SEL (select) bits are reset to zero. This selects max. count register A, resulting in the timer-out pins going high upon reset.

Interrupt Controller Block Diagram



Interrupt Controller

The SAB 80186 can receive interrupts from a number of sources, both internal and external. The internal interrupt controller serves to merge these requests on a priority basis, for individual service by the CPU. Internal interrupt sources (timers and DMA channels) can be disabled by their own control registers or by mask bits within the interrupt controller. The SAB 80186 interrupt controller has its own control registers that set the mode of operation for the controller.

The interrupt controller will resolve priority among requests that are pending simultaneously. Nesting is provided so interrupt service routines for lower priority interrupts may themselves be interrupted by higher priority interrupts.
The interrupt controller has a special iRMX 86 compatibility mode that allows the use of the SAB 80186 within the iRMX 86 operating system interrupt structure.

Master Mode Operation

Interrupt Controller External Interface

For external interrupt sources, five dedicated pins are provided. One of these pins is dedicated to NMI, non-maskable interrupt. This is typically used for power-fail interrupts, etc. The other four pins may function either as four interrupt input lines with internally generated interrupt vectors, as an interrupt line and an interrupt acknowledge line (called the "cascade mode") along with two other input lines with internally generated interrupt vectors, or as two interrupt input lines and two dedicated interrupt acknowledge output lines. When the interrupt lines are configured in cascade mode, the SAB 80186 interrupt controller will not generate internal interrupt vectors.

Interrupt Controller Modes of Operation

The basic modes of operation of the interrupt controller in master mode are similar to the SAB 8259A. The interrupt controller responds identical to internal interrupts in all three modes: the difference is only in the interpretation of function of the four external interrupt pins.

Fully nested mode

When in fully nested mode, four pins are used as direct interrupt requests. The vectors for these four inputs are generated internally. An in-service bit (IS) is provided for every interrupt source. If a lower priority device requests an interrupt while the in-service bit is set, no interrupt will be generated by the interrupt controller. In addition, if another interrupt request occurs from the same interrupt source while the in-service bit is set, no interrupt will be generated by the interrupt controller.
When a service routine is completed, the proper IS bit must be reset by writing the proper pattern to the EOI register.

Cascade mode

The SAB 80186 has four interrupt pins and two of them have dual functions. In fully nested mode, the four pins are used as direct interrupt inputs and the corresponding vectors are generated internally. In cascade mode, the four pins are configured into interrupt input-dedicated acknowledge signal pairs. INT0 is an interrupt input interfaced to an SAB 8259A, while INT2/$\overline{\text{INTA0}}$ serves as the dedicated interrupt acknowledge signal to the peripheral. The same is true for INT1 and INT3/$\overline{\text{INTA1}}$. Each pair can selectively be placed in the cascade or non-cascade mode by programming the proper value INT0 and INT1 control registers.
The primary cascade mode allows the capability to serve up to 128 external interrupt sources through the use of external master and slave SAB 8259As.

Special fully nested mode

This mode is entered by setting the SFNM bit in INT0 or INT1 control register. It enables complete nestability with external SAB 8259A masters. In special fully nested mode, the SAB 80186 interrupt controller will allow interrupts from an external pin regardless of the state of the in-service bit for an interrupt source in order to allow multiple interupts from a single pin.

Polling operation

The controller may be used in a polled mode if interrupts are undesirable. When polling, the processor disables interrupts and then polls the interrupt controller whenever it is convenient. Polling the interrupt controller is accomplished by reading the poll word. Bit 15 in the poll word indicates to the processor that an interrupt of high enough priority is requesting service.

Interrupt Controller Registers (non-iRMX 86 mode)

	Offset
INT3 Control Register	3EH
INT2 Control Register	3CH
INT1 Control Register	3AH
INT0 Control Register	38H
DMA1 Control Register	36H
DMA0 Control Register	34H
Timer Control Register	32H
Interrupt Status Register	30H
Interrupt Request Register	2EH
In-Service Register	2CH
Priority Mask Register	2AH
Mask Register	28H
Poll Status Register	26H
Poll Register	24H
EOI Register	22H

Timer/DMA Control Register Formats

15	14	...	4	3	2	1	0
0	0	...	0	MSK	PR2	PR1	PR0

INT0/INT1 Control Register Formats

15	14	...	7	6	5	4	3	2	1	0
0	0	...	0	SFNM	C	LTM	MSK	PR2	PR1	PR0

INT2/INT3 Control Register Formats

15	14	...	5	4	3	2	1	0
0	0	...	0	LTM	MSK	PR2	PR1	PR0

In-Service, Interrupt Request, and Mask Register Formats

15	14	...	10	9	8	7	6	5	4	3	2	1	0
0	0	...	0	0	0	I3	I2	I1	I0	D1	D0	0	TMR

Priority Mask Register Format

15	14	...	3	2	1	0
0	0	...	0	PRM2	PRM1	PRM0

Interrupt Status Register Format

15	14	...	7	6	5	4	3	2	1	0
DHLT	0	...	0	0	0	0	0	IRT2	IRT1	IRT0

EOI Register Format

15	14	13	...	5	4	3	2	1	0
SPEC/NSPEC	0	0	...	0	S4	S3	S2	S1	S0

Poll Register Format

15	14	13	...	5	4	3	2	1	0
INT REQ	0	0	...	0	S4	S3	S2	S1	S0

Master Mode Features

Programmable priority
The user can program the interrupt sources into any of eight different priority levels. The programming is done by placing a 3-bit priority level (0 to 7) in the control register of each interrupt source.

End-of-interrupt command
The end-of-interrupt (EOI) command is used by the programmer to reset the in-service (IS) bit when an interrupt service routine is completed. The EOI command is issued by writing the proper pattern to the EOI register. There are two types of EOI commands, specific and nonspecific. The non-specific command does not specify which IS bit is reset.

Trigger mode
The four external interrupt pins can be programmed in either edge or level-trigger mode. The control register for each external source has a level-trigger mode (LTM) bit. All interrupt inputs are active high. In the edge-sense mode or the level-trigger mode, the interrupt request must remain active (high) until the interrupt request is acknowledged by the SAB 80186 CPU. In the edge-sense mode, if the level remains high after the interrupt is acknowledged, the input is disabled and no further requests will be generated. The input level must go low for at least one clock cycle to reenable the input.

Interrupt vectoring
The SAB 80186 interrupt controller will generate interrupt vectors for the integrated DMA channels and the integrated timers. In addition, the interrupt controller will generate interrupt vectors for the external interrupt lines if they are not configured in cascade or special fully nested mode.

Interrupt Controller Registers

In-service register
This register contains the in-service bit for each of the interrupt sources. The in-service bit is set to indicate that a source's service routine is in progress. When an in-service bit is set, the interrupt controller will not generate interrupts to the CPU when it receives interrupt requests from devices with a lower programmed priority level. The TMR bit is the in-service bit for all three timers; the D0 and D1 bits are the in-service bits for the two DMA channels; the I0 – I3 bits are the in-service bits for the external interrupt pins.

Interrupt request register
The internal interrupt sources have interrupt request bits inside the interrupt controller. A read from this register yields the status of these bits. The TMR bit is the logical OR of all timer interrupt requests. D0 and D1 are the interrupt request bits for the DMA channels.

Mask register
This is a 16-bit register that contains a mask bit for each interrupt source. A one in a bit position corresponding to a particular source serves to mask the source from generating interrupts. These mask bits are exactly the same bits which are used in the individual control registers.

Priority mask register
This register is used to mask all interrupts below particular interrupt priority levels. The code in the lower three bits of this register inhibits interrupts of priority lower (a higher priority number) than the code specified.

Interrupt status register
This register contains general interrupt controller status information. The bits in the status register have the following functions:

DHLT: DMA halt transfer; setting this bit halts all DMA transfers. It is automatically set whenever a non-maskable interrupt occurs, and it is reset when an IRET instruction is executed.

IRTx: These three bits represent the individual timer interrupt request bits. These bits are used to differentiate the timer interrupts, since the timer IR bit in the interrupt request register is the "OR" function of all timer interrupt requests.

Timer, DMA 0, 1 control registers
These registers are the control words for all the internal interrupt sources. The three bit positions PR0, PR1, and PR2 represent the programmable priority level of the interrupt source. The MSK bit inhibits interrupt requests from the interrupt source. The MSK bits in the individual control registers are exactly the same bits as are in the mask register; modifying them in the individual control registers will also modify them in the mask register, and vice versa.

INT0 to INT3 control registers
These registers are the control words for the four external input pins. In cascade mode or special fully nested mode, the control words for INT2 and INT3 are not used.

PR0-2: Priority programming information. Highest priority = 000, lowest priority = 111

LTM: Level-trigger mode bit. 1 = level-triggered; 0 = edge-triggered. Interrupt input levels are active high. In level-triggered mode, an interrupt is generated whenever the external line is high. In edge-triggered mode, an interrupt will be generated only when this level is preceded by an inactive-to-active transition on the line. In both cases, the level must remain active until the interrupt is acknowledged.

MSK: Mask bit, 1 = mask; 0 = nonmask.

C: Cascade mode bit, 1 = cascade; 0 = direct

SFNM: Special fully nested mode bit, 1 = SFNM

EOI register
The end-of-interrupt register is a command register which can only be written into. It initiates an EOI command when written to by the SAB 80186 CPU.

S_x: Encoded information that specifies an interrupt source vector type

NSPEC/SPEC: A bit that determines the type of EOI command. Nonspecific = 1, specific = 0.

Poll and poll status registers
These registers contain polling information. They can only be read. Reading the poll register constitutes a software poll. This will set the IS bit of the highest priority pending interrupt.

S_x: Encoded information that indicates the vector type of the highest priority interrupting source. Valid only when INTREQ = 1.

INTREQ: This bit determines if an interrupt request is present. Interrupt request = 1; no interrupt request = 0.

iRMX 86 Compatibility Mode

This mode allows iRMX 86-SAB 80186 compatibility. The interrupt model of iRMX 86 requires one master and multiple slave SAB 8259As in cascaded mode. When iRMX mode is used, the internal SAB 80186 interrupt controller will be used as a slave controller to an external master interrupt controller.
Upon reset, the SAB 80186 interrupt controller will be in the non-iRMX 86 mode of operation. To set the controller in the iRMX 86 mode, bit 14 of the relocation register should be set.

The iRMX 86 operating system requires peripherals to be assigned fixed priority levels. This is incompatible with the normal operation of the SAB 80186 interrupt controller. Therefore, the initialization software must program the proper priority levels for each source.

Required iRMX Internal Source Priority Levels

Priority Level	Interrupt Source
0	Timer 0
1	(reserved)
2	DMA 0
3	DMA 1
4	Timer 1
5	Timer 2

iRMX 86 Mode External Interface

In the iRMX 86 configuration of the SAB 80186 the INT0 input is used as the SAB 80186 CPU interrupt input. INT3 functions as an output to send the SAB 80186 slave-interrupt-request to one of the 8 master PIC inputs. Correct master-slave interface requires decoding of the slave addresses (CAS0–2). Slave SAB 8259As do this internally. Because of pin limitations, the SAB 80186 slave address will have to be decoded externally. $\overline{INT1}$ is used as a slave-select input.

$\overline{INT2}$ is used as an acknowledge output, suitable to drive the $\overline{INTA}$ input of an SAB 8259A.

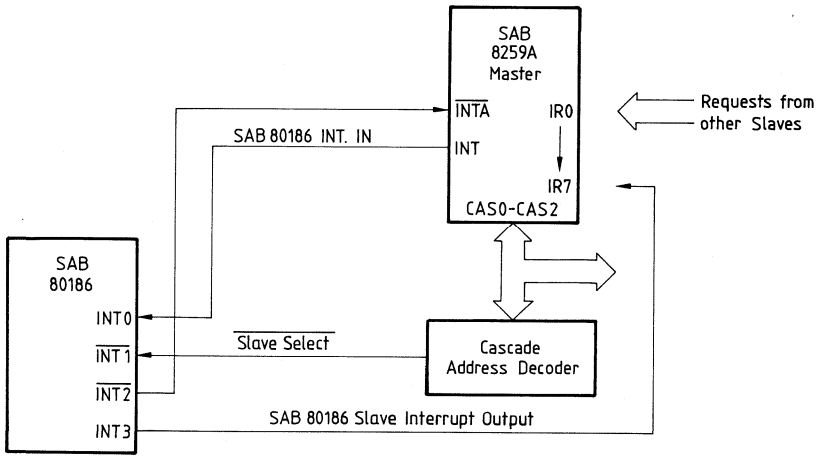
Vector Generation in the iRMX 86 Mode

Vector generation in iRMX mode is exactly like that of an SAB 8259A slave. The interrupt controller generates an 8-bit vector which the CPU multiplies by four and uses as an address into a vector table. The significant five bits of the vector are user-programmable while the lower three bits are generated by the priority logic.

Specific End-of-Interrupt

In iRMX mode the specific EOI command operates to reset an in-service bit of a specific priority. The user supplies a 3-bit priority-level value that points to an in-service bit to be reset.

iRMX 86 Interrupt Controller Interconnection



Interrupt Controller Registers (iRMX 86 mode)

	Offset
Level 5 Control Register (Timer 2)	3AH
Level 4 Control Register (Timer1)	38H
Level 3 Control Register (DMA1)	36H
Level 2 Control Register (DMA 0)	34H
Level 0 Control Register (Timer0)	32H
Interrupt Status Register	30H
Interrupt-Request Register	2EH
In-Service Register	2CH
Priority-Level Mask Register	2AH
Mask Register	28H
Specific EOI Register	22H
Interrupt Vector Register	20H

Specific EOI Register Format

15	14	13	· · · · · · · ·	8	7	6	5	4	3	2	1	0
0	0	0		0	0	0	0	0	0	L2	L1	L0

In-Service, Interrupt Request, and Mask Register Format

15	14	13	· · · · · · · ·	8	7	6	5	4	3	2	1	0
0	0	0		0	0	0	TMR 2	TMR 1	D1	D0	0	TMR 0

Control Word Format

15	14	13	· · · · · · · ·	8	7	6	5	4	3	2	1	0
0	0	0		0	0	0	0	0	MSK	PR2	PR1	PR0

Interrupt Vector Register Format

15	14	13	· · · · · · · ·	8	7	6	5	4	3	2	1	0
0	0	0		0	t4	t3	t2	t1	t0	0	0	0

Priority Level Mask Register

15	14	13	· · · · · · · ·	8	7	6	5	4	3	2	1	0
0	0	0		0	0	0	0	0	0	m2	m1	m0

Interrupt Controller Registers in the iRMX 86 Mode

End-of-interrupt register
The end-of-interrupt register is a command register which can only be written to. It initiates an EOI command when written to by the SAB 80186 CPU.

Lx: Encoded value indicating the priority of the IS bit to be reset.

In-service register
This register contains the in-service bit for each of the internal interrupt sources. Bit positions 2 and 3 correspond to the DMA channels; positions 0, 4, and 5 correspond to the integral timers.

Interrupt request register
This register indicates which internal peripherals have interrupt requests pending. The interrupt request bits are set when a request arrives from an internal source, and are reset when the processor acknowledges the request.

Mask register
This register contains a mask bit for each interrupt source. If the bit in this register corresponding to a particular interrupt source is set, any interrupts from that source will be masked.

Control registers
These registers are the control words for all the internal interrupt sources.

PRx: 3-bit encoded field indicating a priority level for the source; note that each source must be programmed at specified levels.

MSK: Mask bit for the priority level indicated by PRx bits.

Interrupt vector register
This register provides the upper five bits of the interrupt vector address. The interrupt controller itself provides the lower three bits of the interrupt vector as determined by the priority level of the interrupt request.
The format of the bits in this register is:
tx: 5-bit field indicating the upper five bits of the vector address.

Priority-level mask register
This register indicates the lowest priority-level interrupt which will be serviced.

The encoding of the bits in this register is:
mx: 3-bit encoded field indication priority-level value. All levels of lower priority will be masked.

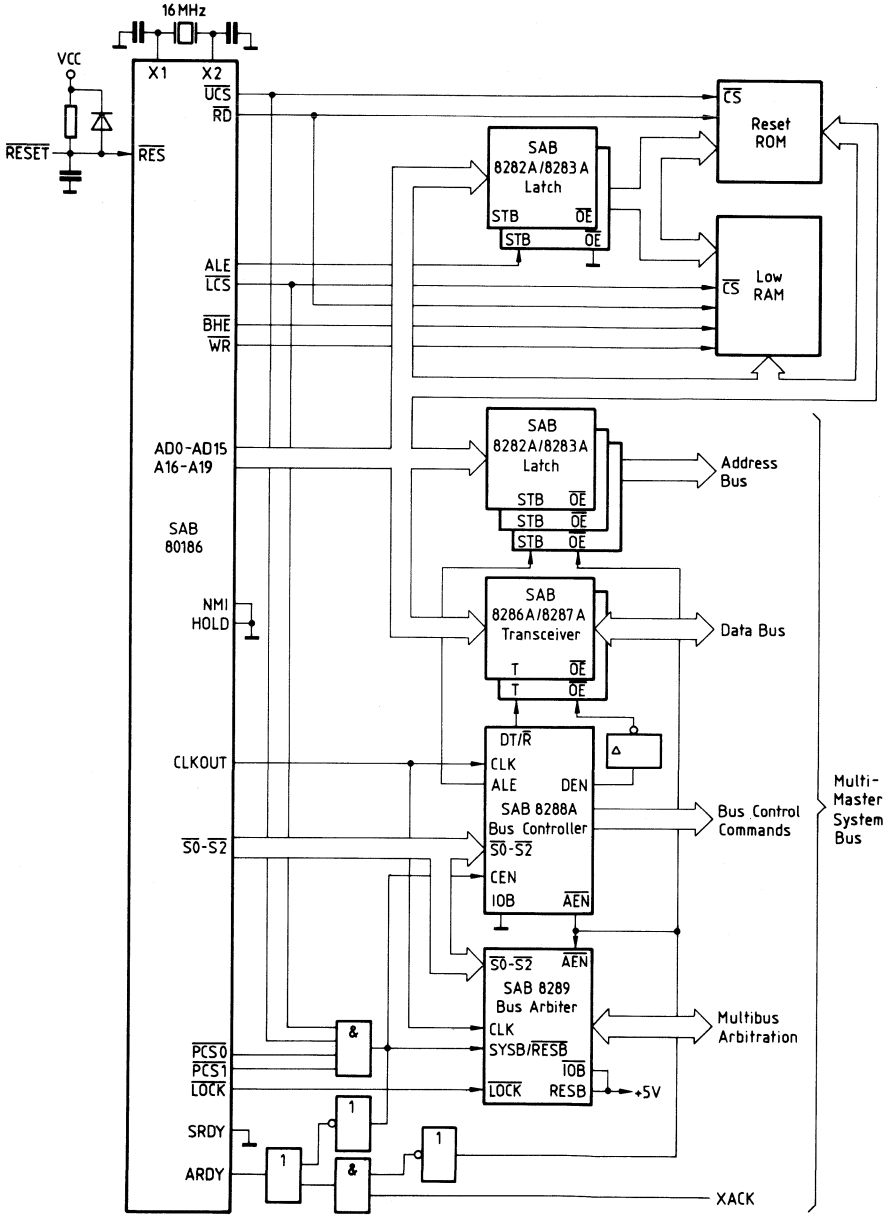
Interrupt status register
This register is defined exactly as in non-iRMX mode.

Interrupt Controller and Reset

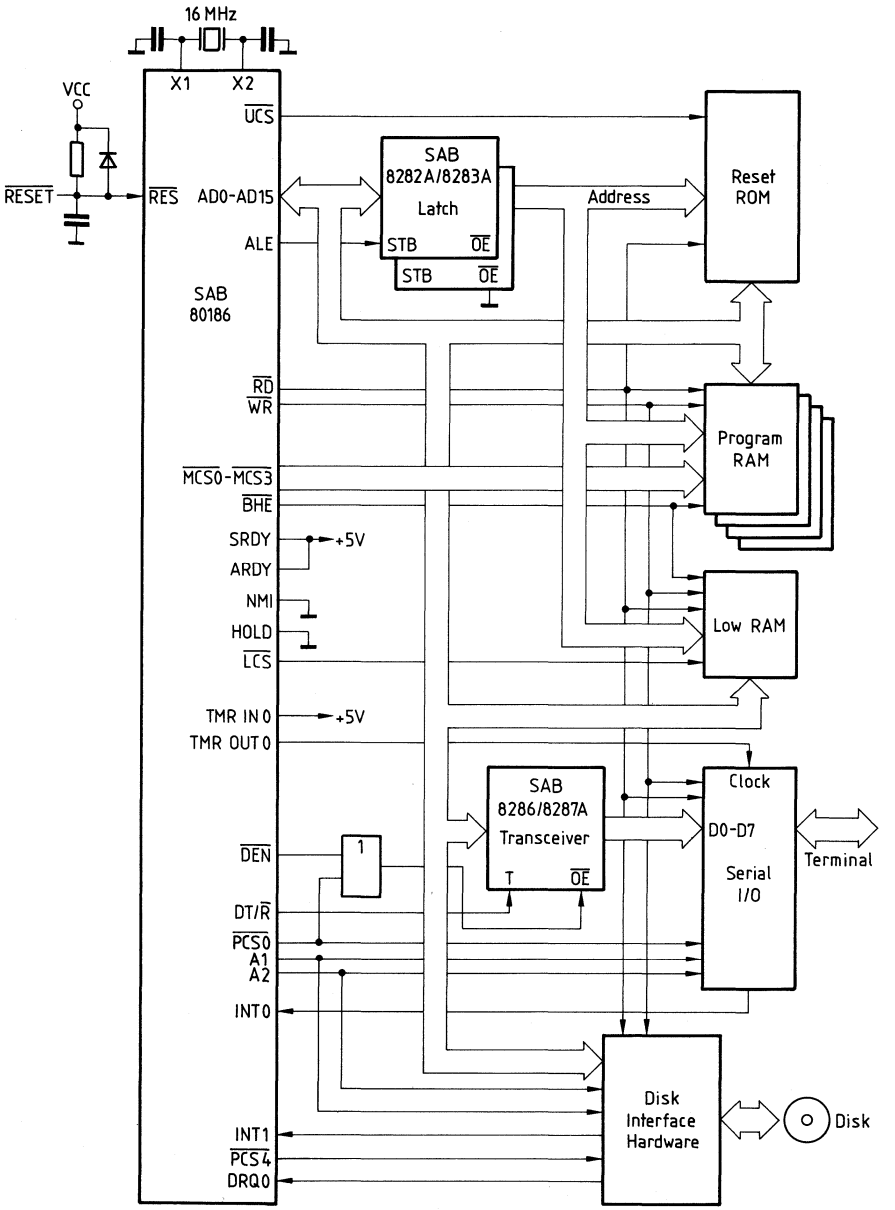
Upon reset, the interrupt controller will perform the following actions:

- All SFNM bits reset to 0, implying fully nested mode.
- All PR bits in the various control registers set to 1. This places all sources at lowest priority (level 111).
- All LTM bits reset to 0, resulting in edge-sense mode.
- All interrupt service bits reset to 0.
- All interrupt request bits reset to 0.
- All MSK (interrupt mask) bits set to 1 (mask).
- All C (cascade) bits reset to 0 (non-cascade).
- All PRM (Priority mask) bits set to 1, implying no levels masked.
- Initialized to non-iRMX 86 mode.

Typical SAB 80186 Multimaster Bus Interface



Typical SAB 80186 System Configuration



Instruction Timings

The following instruction timings represent the minimum execution time in clock cycles for each instruction. The timings given are based on the following assumptions:

- The op code, along with any data or displacement required for execution of a particular instruction, has been prefetched and resides in the queue at the time it is needed.
- No wait states or bus HOLDs occur.
- All word data is located on even-address boundaries.

All jumps and calls include the time required to fetch the op code of the next instruction at the destination address.

All instructions which involve memory reference can require one (and in some cases, two) additional clocks above the minimum timings shown. This is due to the asynchronous nature of the handshake between the BIU and the execution unit.

Notes:

The effective address (EA) of the memory operand is computed according to the mod and r/m fields:

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low

If r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP

DISP follows 2nd byte of instruction (before data if required)

* except if mod = 00 and r/m = 110 then EA = disp-high: disp-low

Note: EA calculation time is 4 clock cycles for all modes, and is included in the execution times given whenever appropriate.

Segment override prefix

0	0	1	reg	1	1	0

reg is assigned according to the following:

reg	Segment register
00	ES
01	CS
10	SS
11	DS

REG is assigned according to the following table:

16-bit (w = 1)	8-bit (w = 0)
000 AX	000 AL
001 CX	001 CL
010 DX	010 DL
011 BX	011 BL
100 SP	100 AH
101 BP	101 CH
110 SI	110 DH
111 DI	111 BH

The physical addresses of all operands addressed by the BP register are computed using the SS segment register. The physical addresses of the destination operands of the string primitive operations (those addressed by the DI register) are computed using the ES segment, which may not be overridden.

Instruction Set Summary

Function	Format	Clock Cycles	Comments
Data Transfer			
MOV = Move:			
Register to register/memory	1 0 0 0 1 0 0 w | mod reg r/m	2/12	
Register/memory to register	1 0 0 0 1 0 1 w | mod reg r/m	2/9	
Immediate to register/memory	1 1 0 0 0 1 1 w | mod 0 0 0 r/m | data | data if w = 1	12–13	8/16-bit
Immediate to register	1 0 1 1 w reg | data | data if w = 1	3–4	8/16-bit
Memory to accumulator	1 0 1 0 0 0 0 w | addr-low | addr-high	9	
Accumulator to memory	1 0 1 0 0 0 1 w | addr-low | addr-high	8	
Register/memory to segment register	1 0 0 0 1 1 1 0 | mod 0 reg r/m	2/9	
Segment register to register/memory	1 0 0 0 1 1 0 0 | mod 0 reg r/m	2/11	
PUSH = Push:			
Memory	1 1 1 1 1 1 1 1 | mod 1 1 0 r/m	16	
Register	0 1 0 1 0 reg	10	
Segment register	0 0 0 reg 1 1 0	9	
Immediate	0 1 1 0 1 0 s 0 | data | data if s = 0	10	
PUSHA = Push All	0 1 1 0 0 0 0 0	36	
POP = Pop:			
Memory	1 0 0 0 1 1 1 1 | mod 0 0 0 r/m	20	
Register	0 1 0 1 1 reg	10	
Segment register	0 0 0 reg 1 1 1 (reg ≠ 01)	8	

Shaded areas indicate instructions not available on SAB 8086/8088 processors.

Function	Format			Clock Cycles	Comments
Data Transfer (cont'd)					
POPA = Pop All	01100001			51	
XCHG = **Exchange:**					
Register/memory with register	1000011w	mod reg r/m		4/17	
Register with accumulator	10010 reg			3	
IN = **Input from:**					
Fixed port	1110010w	port		10	
Variable port	1110110w			8	
OUT = **Output to:**					
Fixed port	1110011w	port		9	
Variable port	1110111w			7	
XLAT = Translate byte to AL	11010111			11	
LEA =Load EA to register	10001101	mod reg r/m		6	
LDS = Load pointer to DS	11000101	mod reg r/m	(mod ≠ 11)	18	
LES = Load pointer to ES	11000100	mod reg r/m	(mod ≠ 11)	18	
LAHF = Load AH with flags	10011111			2	
SAHF = Store AH into flags	10011110			3	
PUSHF = Push flags	10011100			9	
POPF = Pop flags	10011101			8	
SEGMENT = **Segment override:**					
CS	00101110			2	
SS	00110110			2	
DS	00111110			2	
ES	00100110			2	

Shaded areas indicate instructions not available on SAB 8086/8088 processors.

Function	Format	Clock Cycles	Comments
Arithmetic			
ADD = Add:			
Reg./memory with register to either	0 0 0 0 0 d w | mod reg. r/m	3/10	
Immediate to register/memory	1 0 0 0 0 0 s w | mod 0 0 0 r/m | data | data if s w = 01	4/16	
Immediate to accumulator	0 0 0 0 0 1 0 w | data | data if w = 1	3/4	8/16-bit
ADC = Add with carry:			
Reg./memory with register to either	0 0 0 1 0 0 d w | mod reg r/m	3/10	
Immediate to register/memory	1 0 0 0 0 0 s w | mod 0 1 0 r/m | data | data if s w = 01	4/16	
Immediate to accumulator	0 0 0 1 0 1 0 w | data | data if w = 1	3/4	8/16-bit
INC = Increment			
Register/memory	1 1 1 1 1 1 1 w | mod 0 0 0 r/m	3/15	
Register	0 1 0 0 0 reg	3	
SUB = Subtract			
Reg./memory and register to either	0 0 1 0 1 0 d w | mod reg r/m	3/10	
Immediate from register/memory	1 0 0 0 0 0 s w | mod 1 0 1 r/m | data | data if s w = 01	4/16	
Immediate from accumulator	0 0 1 0 1 1 0 w | data | data if w = 1	3/4	8/16-bit
SBB = Subtract with borrow:			
Reg./memory and register to either	0 0 0 1 1 0 d w | mod reg r/m	3/10	
Immediate from register/memory	1 0 0 0 0 0 s w | mod 0 1 1 r/m | data | data if s w = 01	4/16	
Immediate from accumulator	0 0 0 1 1 1 0 w | data | data if w = 1	3/4	8/16-bit

Function	Format	Clock Cycles	Comments
Arithmetic (cont'd):			
DEC = **Decrement:**			
Register/memory	1 1 1 1 1 1 1 w | mod 0 0 1 r/m	3/15	
Register	0 1 0 0 1 reg	3	
CMP = **Compare:**			
Register/memory with register	0 0 1 1 1 0 1 w | mod reg r/m	3/10	
Register with register/memory	0 0 1 1 1 0 0 w | mod reg r/m	3/10	
Immediate with register/memory	1 0 0 0 0 0 s w | mod 1 1 1 r/m | data | data if s w=01	3/10	
Immediate with accumulator	0 0 1 1 1 1 0 w | data | data if w = 1	3/4	8/16-bit
NEG = Change sign	1 1 1 1 0 1 1 w | mod 0 1 1 r/m	3	
AAA = ASCII adjust for add	0 0 1 1 0 1 1 1	8	
DAA = Decimal adjust for add	0 0 1 0 0 1 1 1	4	
AAS = ASCII adjust for subtract	0 0 1 1 1 1 1 1	7	
DAS = Decimal adjust for subtract	0 0 1 0 1 1 1 1	4	
MUL = Multiply (unsigned):	1 1 1 1 0 1 1 w | mod 1 0 0 r/m		
register–byte		26–28	
register–word		35–37	
memory–byte		32–34	
memory–word		41–43	
IMUL = Integer multiply (signed):	1 1 1 1 0 1 1 w | mod 1 0 1 r/m		
register–byte		25–28	
register–word		34–37	
memory–byte		31–34	
memory–word		40–43	

Function	Format	Clock Cycles	Comments
Arithmetic (cont'd):			
IMUL = Integer immediate multiply (signed)	0 1 1 0 1 0 s 1 | mod reg r/m | data | data if s = 0	22–25/29–32	
DIV = Divide (unsigned):	1 1 1 1 0 1 1 w | mod 1 1 0 r/m		
register–byte		29	
register–word		38	
memory–byte		35	
memory–word		44	
IDIV = Integer divide (signed):	1 1 1 1 0 1 1 w | mod 1 1 1 r/m		
register–byte		44–52	
register–word		53–61	
memory–byte		50–58	
memory–word		59–67	
AAM = ASCII adjust for multiply	1 1 0 1 0 1 0 0 | 0 0 0 0 1 0 1 0	19	
AAD = ASCII adjust for divide	1 1 0 1 0 1 0 1 | 0 0 0 0 1 0 1 0	15	
CBW = Convert byte to word	1 0 0 1 1 0 0 0	2	
CWD = Convert word to double word	1 0 0 1 1 0 0 1	4	
Logic			
Shift/rotate instructions:			
Register/memory by 1	1 1 0 1 0 0 0 w | mod TTT r/m	2/15	
Register/memory by CL	1 1 0 1 0 0 1 w | mod TTT r/m	5+n/17+n	
Register/memory by count	1 1 0 0 0 0 0 w | mod TTT r/m | count	5+n/17+n	

TTT	Instruction
0 0 0	ROL
0 0 1	ROR
0 1 0	RCL
0 1 1	RCR
1 0 0	SHL/SAL
1 0 1	SHR
1 1 1	SAR

Shaded areas indicate instructions not available on SAB 8086/8088 processors.

Function	Format				Clock Cycles	Comments
Logic (cont'd):						
AND = **And:**						
Reg./memory and register to either	0 0 1 0 0 0 d w	mod reg r/m			3/10	
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 0 0 r/m	data	data if w = 1	4/16	
Immediate to accumulator	0 0 1 0 0 1 0 w	data	data f w = 1		3/4	8/16-bit
TEST = **And function to flags, no result:**						
Register/memory and register	1 0 0 0 0 1 0 w	mod reg r/m			3/10	
Immediate data and register/memory	1 1 1 1 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1	4/10	
Immediate data and accumulator	1 0 1 0 1 0 0 w	data	data if w = 1		3/4	8/16-bit
OR = **Or:**						
Reg./memory and register to either	0 0 0 0 1 0 d w	mod reg r/m			3/10	
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 0 0 1 r/m	data	data if w = 1	4/16	
Immediate to accumulator	0 0 0 0 1 1 0 w	data	data if w = 1		3/4	8/16-bit
XOR = **Exclusive Or:**						
Reg./memory and register to either	0 0 1 1 0 0 d w	mod reg r/m			3/10	
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 1 0 r/m	data	data if w = 1	4/16	
Immediate to accumulator	0 0 1 1 0 1 0 w	data	data if w = 1		3/4	8/16-bit
NOT = Invert register/memory	1 1 1 1 0 1 1 w	mod 0 1 0 r/m			3	

Function	Format	Clock Cycles	Comments
String Manipulation:			
MOVS = Move byte/word	1 0 1 0 0 1 0 w	14	
CMPS = Compare byte/word	1 0 1 0 0 1 1 w	22	
SCAS = Scan byte/word	1 0 1 0 1 1 1 w	15	
LODS = Load byte/word to AL/AX	1 0 1 0 1 1 0 w	12	
STOS = Store byte/word from AL/AX	1 0 1 0 1 0 1 w	10	
INS = Input byte/word from DX port	0 1 1 0 1 1 0 w	14	
OUTS = Output byte/word to DX port	0 1 1 0 1 1 1 w	14	
Repeated by count in CX			
MOVS = Move string	1 1 1 1 0 0 1 0 | 1 0 1 0 0 1 0 w	8 + 8n	
CMPS = Compare string	1 1 1 1 0 0 1 z | 1 0 1 0 0 1 1 w	5 + 22n	
SCAS = Scan string	1 1 1 1 0 0 1 z | 1 0 1 0 1 1 1 w	5 + 15n	
LODS = Load string	1 1 1 1 0 0 1 0 | 1 0 1 0 1 1 0 w	6 + 11n	
STOS = Store string	1 1 1 1 0 0 1 0 | 1 0 1 0 1 0 1 w	6 + 9n	
INS = Input string	1 1 1 1 0 0 1 0 | 0 1 1 0 1 1 0 w	8 + 8n	
OUTS = Output string	1 1 1 1 0 0 1 0 | 0 1 1 0 1 1 1 w	8 + 8n	

Shaded areas indicate instructions not available on SAB 8086/8088 processors.

Function	Format	Clock Cycles	Comments
Control Transfer:			
CALL = Call:			
Direct within segment	11101000 | disp-low | disp-high	15	
Register/memory indirect within segment	11111111 | mod 0 1 0 r/m	13/19	
Direct intersegment	10011010 | segment offset segment selector	23	
Indirect intersegment	11111111 | mod 0 1 1 r/m (mod ≠ 11)	38	

Function	Format	Clock Cycles	Comments
Control Transfer (cont'd):			
JMP = Unconditional jump:			
Short/long	11101011 | disp-low	14	
Direct within segment	11101001 | disp-low | disp-high	14	
Register/memory indirect within segment	11111111 | mod 100 r/m	11/17	
Direct intersegment	11101010 | segment offset segment selector	14	
Indirect intersegment	11111111 | mod 101 r/m (mod ≠ 11)	26	
RET = Return from CALL:			
Within segment	11000011	16	
Within seg. adding immediate to SP	11000010 | data-low | data-high	18	
Intersegment	11001011	22	
Intersegment adding immediate to SP	11001010 | data-low | data-high	25	

Function	Format		Clock Cycles	Comments
Control Transfer (cont'd):				
JE/JZ = Jump on equal/zero	01110100	disp	4/13	JMP not taken/JMP taken
JL/JNGE = Jump on less/not greater or equal	01111100	disp	4/13	
JLE/JNG = Jump on less or equal/not greater	01111110	disp	4/13	
JB/JNAE = Jump on below/not above or equal	01110010	disp	4/13	
JBE/JNA = Jump on below or equal/not above	01110110	disp	4/13	
JP/JPE = Jump on parity/parity even	01111010	disp	4/13	
JO = Jump on overflow	01110000	disp	4/13	
JS = Jump on sign	01111000	disp	4/13	
JNE/JNZ = Jump on not equal/not zero	01110101	disp	4/13	
JNL/JGE = Jump on not less/greater or equal	01111101	disp	4/13	
JNLE/JG = Jump on not less or equal/greater	01111111	disp	4/13	
JNB/JAE = Jump on not below/above or equal	01110011	disp	4/13	
JNBE/JA = Jump on not below or equal/above	01110111	disp	4/13	
JNP/JPO = Jump on not parity/parity odd	01111011	disp	4/13	
JNO = Jump on not overflow	01110001	disp	4/13	
JNS = Jump on not sign	01111001	disp	4/13	
JCXZ = Jump on CX zero	11100011	disp	5/15	
LOOP = Loop CX times	11100010	disp	6/16	LOOP not taken/LOOP taken
LOOPZ/LOOPE = Loop while zero/equal	11100001	disp	6/16	
LOOPNZ/LOOPNE = Loop while not zero/equal	11100000	disp	6/16	

Function	Format	Clock Cycles	Comments
Control Transfer (cont'd):			
ENTER = Enter procedure	11001000 | data-low | data-high | L		
L = 0		15	
L = 1		25	
L > 1		22 + 16(n−1)	
LEAVE = Leave procedure	11001001	8	
INT = **Interrupt:**			
Type specified	11001101 | type	47	
Type 3	11001100	45	
INTO = Interrupt on overflow	11001110	48/4	if INT taken/ if INT not taken
IRET = Interrupt return	11001111	28	
BOUND = Detect value out of range	01100010 | mod reg r/m	33–35	

Shaded areas indicate instructions not available on SAB 8086/8088 processors.

Function	Format	Clock Cycles	Comments
Processor Control			
CLC = Clear carry	11111000	2	
CMC = Complement carry	11110101	2	
STC = Set carry	11111001	2	
CLD = Clear direction	11111100	2	
STD = Set direction	11111101	2	
CLI = Clear interrupt	11111010	2	
STI = Set interrupt	11111011	2	
HLT = Halt	11110100	2	
WAIT = Wait	10011011	6	if $\overline{TEST}$ = 0
LOCK = Bus lock prefix	11110000	2	
ESC = Processor extension escape	11011TTT | mod LLL r/m (TTT LLL are op codes to processor extension)	6	

Absolute Maximum Ratings

Ambient temperature under bias	0 to 70°C
Storage temperature	−65 to +150°C
Voltage on any pin with respect to ground	−0.5 to +7V
Power dissipation	3 W

Note: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

T_A = 0 to 70°C; V_{CC} = 5 V ±10%

Parameter	Symbol	Limit values		Unit	Test conditions
		min.	max.		
Input low voltage	V_{IL}	−0.5	+0.8	V	–
Input high voltage (all except X1 and $\overline{RES}$)	V_{IH}	2.0	V_{CC} +0.5	V	–
Input high voltage ($\overline{RES}$)	V_{IH1}	3.0	V_{CC} +0.5	V	–
Output low voltage	V_{OL}	–	0.45	V	Ia = 2.5 mA for $\overline{S0}$-$\overline{S2}$ Ia = 2.0 mA for all other outputs
Output high voltage	V_{OH}	2.4	–	V	I_{OA} = −400 µA
Power supply current	I_{CC}	–	550	mA	T_A = 0°C; All outputs disconnected
		–	450	mA	T_A = 70°C; All outputs disconnected
Input leakage current	I_{LI}	–	±10	µA	0 V < V_{IN} < V_{CC}
Output leakage current	I_{LO}	–	±10	µA	0.45 V < V_{OUT} < V_{CC}
Clock output low	V_{CLO}	–	0.6	V	I_A = 4.0 mA
Clock output high	V_{CHO}	4.0	–	V	I_{OA} = −200 µA
Clock input low voltage	V_{CLI}	−0.5	0.6	V	–
Clock input high voltage	V_{CHI}	3.9	V_{CC} +1.0	V	–
Input capacitance	C_{IN}	–	10	pF	–
I/O capacitance	C_{IO}	–	20	pF	–

AC Characteristics

T_A = 0 to 70°C, V_{CC} = 5 V ±10%

Timing Requirements: all timings measured at 1.5 V unless otherwise specified

Parameter	Symbol	SAB 80186		SAB 80186-1		Unit	Test conditions
		min.	max.	min.	max.		
Data in setup (A/D)	t_{DVCL}	20	–	15	–	ns	–
Data in hold (A/D)	t_{CLDX}	10	–	8	–	ns	–
Asynchronous ready (ARDY) active setup time [1]	t_{ARYHCH}	20	–	15	–	ns	–
ARDY inactive setup time	t_{ARYLCL}	35	–	25	–	ns	–
ARDY hold time	t_{CHARYX}	15	–	15	–	ns	–
Asynchronous ready inactive hold time	t_{ARYCHL}	15	–	15	–	ns	–
Synchronous ready (SRDY) transition setup time	t_{SRYCL}	20	–	20	–	ns	–
SRDY transition hold time	t_{CLSRY}	15	–	15	–	ns	–
HOLD setup [1]	t_{HVCL}	25	–	20	–	ns	–
INTR, NMI, $\overline{\text{TEST}}$, TIMERIN, setup [1]	t_{INVCH}	25	–	25	–	ns	–
DRQ0, DRQ1, setup [1]	t_{INVCL}	25	–	20	–	ns	–

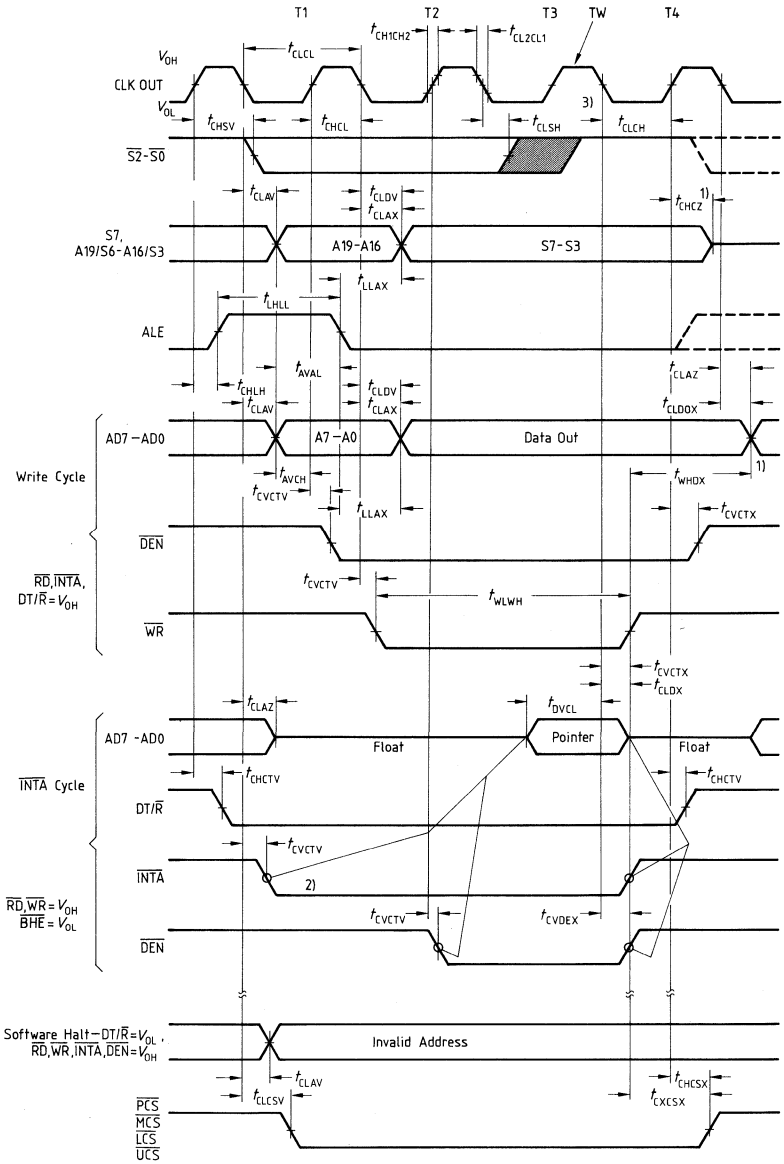
[1] To guarantee recognition at next clock.

Parameter	Symbol	SAB 80186		SAB 80186-1		Unit	Test conditions
		min.	max.	min.	max.		
Master Interfaces Timing Responses							
Address valid delay	t_{CLAV}	5	55	5	44	ns	C_L = 20 to 200 pF all outputs
Address hold	t_{CLAX}	10	–	10	–	ns	–
Address float delay	t_{CLAZ}	t_{CLAX}	35	t_{CLAX}	30	ns	–
Command lines float delay	t_{CHCZ}	–	45	–	40	ns	–
Command lines valid delay (after float)	t_{CHCV}	–	55	–	45	ns	–
ALE width	t_{LHLL}	t_{CLCL} -35	–	t_{CLCL} -30	–	ns	–
ALE active delay	t_{CHLH}	–	35	–	30	ns	–
ALE inactive delay	t_{CHLL}	–	35	–	30	ns	–
Address hold to ALE inactive	t_{LLAX}	t_{CHCL} -25	–	t_{CHCL} -20	–	ns	–
Data valid delay	t_{CLDV}	10	44	10	40	ns	–
Data hold time	t_{CLDOX}	10	–	10	–	ns	–
Data hold after WR	t_{WHDX}	t_{CLCL} -40	–	t_{CLCL} -34	–	ns	–
Control active delay 1	t_{CVCTV}	10	70	5	40	ns	–
Control active delay 2	t_{CHCTV}	10	55	10	44	ns	–
Control inactive delay	t_{CVCTX}	5	55	5	44	ns	–
$\overline{DEN}$ inactive delay (non-write cycle)	t_{CVDEX}	10	70	10	56	ns	–
Address float to $\overline{RD}$ active	t_{AZRL}	0	–	0	–	ns	–
$\overline{RD}$ active delay	t_{CLRL}	10	70	10	56	ns	–
$\overline{RD}$ inactive delay	t_{CLRH}	10	55	10	44	ns	–
$\overline{RD}$ inactive to address active	t_{RHAV}	t_{CLCL} -40	–	t_{CLCL} -40	–	ns	–
HLDA valid delay	t_{CLHAV}	5	50	5	40	ns	–
$\overline{RD}$ width	t_{RLRH}	2 t_{CLCL} -50	–	2 t_{CLCL} -46	–	ns	–
$\overline{WR}$ width	t_{WLWH}	2 t_{CLCL} -40	–	2 t_{CLCL} -34	–	ns	–
Address valid to ALE low	t_{AVAL}	t_{CLCH} -25	–	t_{CLCH} -19	–	ns	–
Status active delay	t_{CHSV}	10	55	10	45	ns	–
Status inactive delay	t_{CLSH}	10	65	10	50	ns	–
Timer output delay	t_{CLTMV}	–	60	–	48	ns	100 pF max.
Reset delay	t_{CLRO}	–	60	–	48	ns	–
Queue status delay	t_{CHQSV}	–	35	–	28	ns	–
Status hold time	t_{CHDX}	10	–	10	–	ns	–
Address valid to clock high	t_{AVCH}	10	–	10	–	ns	–
$\overline{LOCK}$ valid/invalid delay	t_{CLLV}	5	65	5	60	ns	–

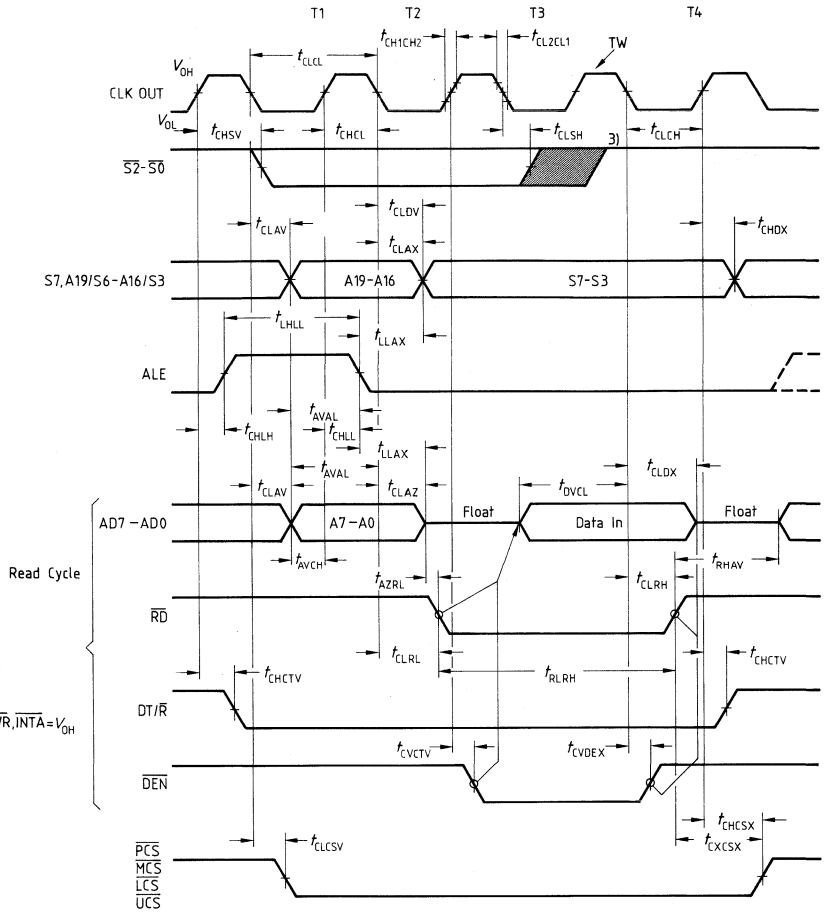
Parameter	Symbol	SAB 80186		SAB 80186-1		Unit	Test conditions
		min.	max.	min.	max.		
Chip Select Timing Responses							
Chip select active delay	t_{CLCSV}	–	66	–	45	ns	–
Chip select hold from command inactive	t_{CXCSX}	35	–	35	–	ns	–
Chip select inactive delay	t_{CHCSX}	5	35	5	32	ns	–
CLKIN Requirements							
CLKIN period	t_{CKIN}	62.5	250	50	250	ns	–
CLKIN fall time	t_{CKHL}	–	10	–	10	ns	3.5 to 1.0 V
CLKIN rise time	t_{CKLH}	–	10	–	10	ns	1.0 to 3.5 V
CLKIN low time	t_{CLCK}	25	–	20	–	ns	1.5 V
CLKIN high time	t_{CHCK}	25	–	20	–	ns	1.5 V
CLKOUT Timing (200 pF load)							
CLKIN to CLKOUT skew	t_{CICO}	–	50	–	25	ns	–
CLKOUT period	t_{CLCL}	125	500	100	500	ns	–
CLKOUT low time	t_{CLCH}	$1/2\ t_{CLCL} - 7.5$	–	$1/2\ t_{CLCL} - 6.0$	–	ns	1.5 V
CLKOUT high time	t_{CHCL}	$1/2\ t_{CLCL} - 7.5$	–	$1/2\ t_{CLCL} - 6.0$	–	ns	1.5 V
CLKOUT rise time	t_{CH1CH2}	–	15	–	12	ns	1.0 to 3.5 V
CLKOUT fall time	t_{CL2CL1}	–	15	–	12	ns	3.5 to 1.0 V

Waveforms

Major Cycle Timing



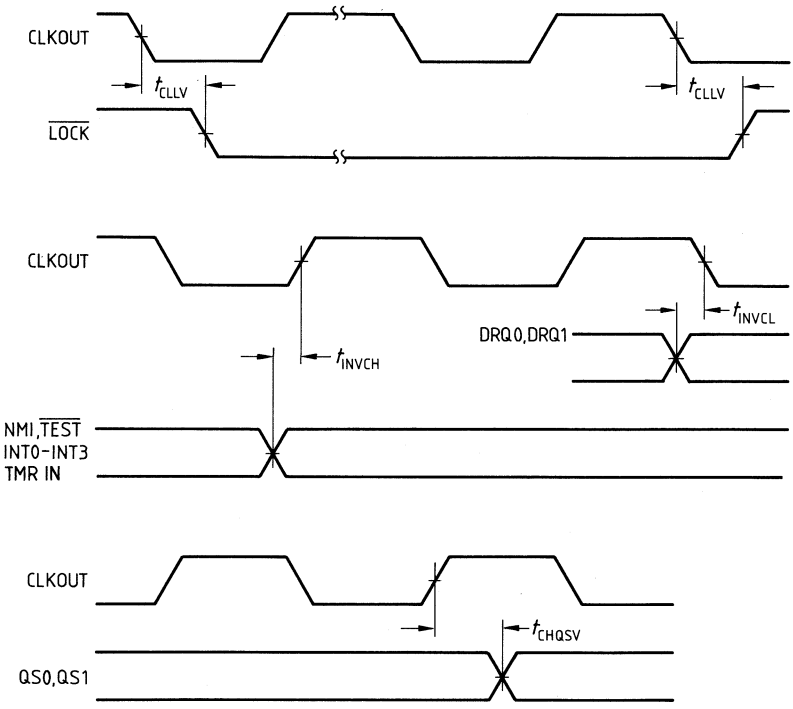
Major Cycle Timing (cont'd)



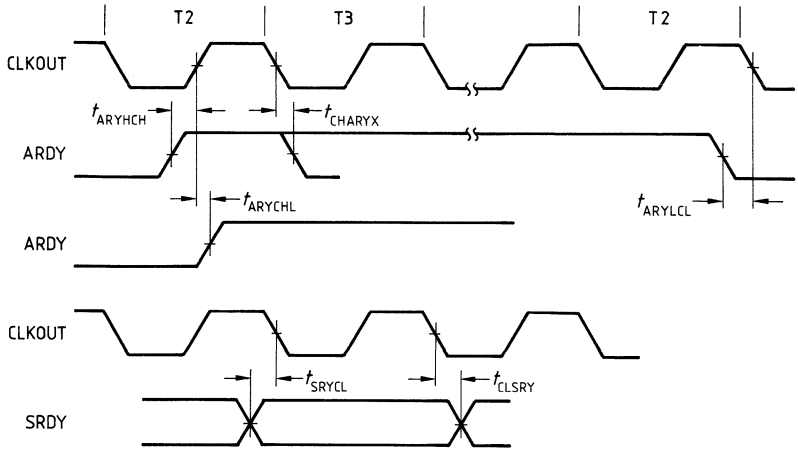
Notes

1) Following a write cycle, the local bus is tristated by the SAB 80186 only when the SAB 80186 enters a "hold acknowledge" state.
2) INTA occurs one clock later in iRMX mode.
3) Status inactive just prior to T4.

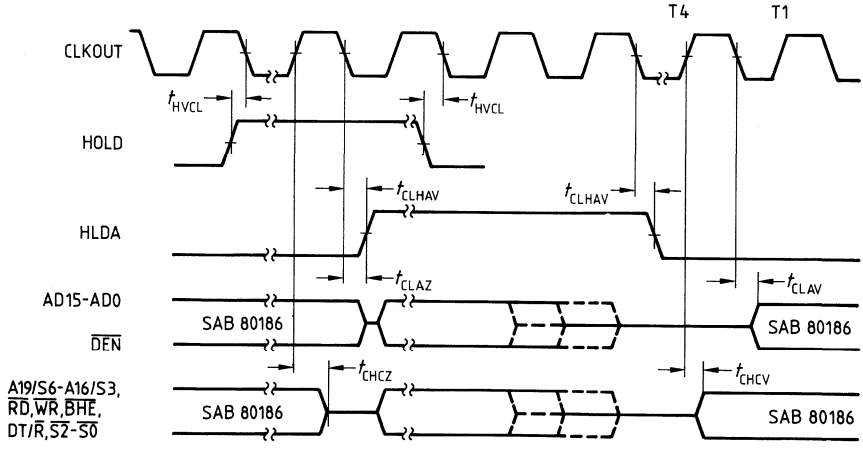
CLKOUT Timing Relationships



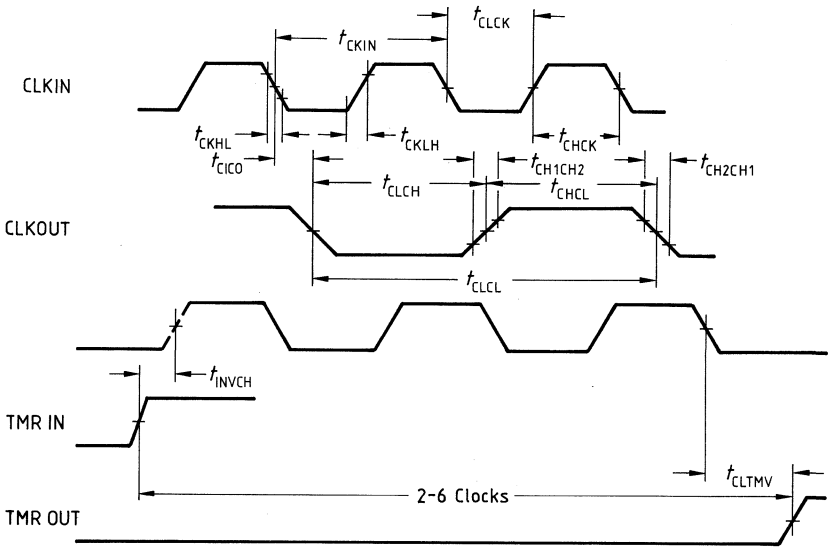
Ready Timing



HOLD-HLDA Timing



Timer Timing



SIEMENS

High-Integration SAB 80188/80188-1 8-Bit Microprocessor

Preliminary

SAB 80188 8 MHz

- Integrated feature set
  - enhanced SAB 8088-2 CPU
  - clock generator
  - 2 independent high-speed DMA channels
  - programmable interrupt controller
  - 3 programmable 16-bit timers
  - programmable memory and peripheral chip-select logic
  - programmable wait state generator
  - local bus controller
- High-performance processor
  - twice the performance of the standard SAB 8088 at 8 MHz
  - 2 Mbyte/s bus bandwidth interface at 8 MHz

SAB 80188-1 10 MHz

- Direct addressing capability to 1 Mbyte of memory
- Completely object-code compatible with all existing SAB 8086/8088 software
  - 10 new instruction types
- Optional numerical processor extension
- Compatible with the bus support components SAB 8282A/8283A/8286A/8287A and SAB 8288A/8289
- Compatible with industry standard 80188 processor
- Available in 10 MHz (SAB 80188-1) and 8 MHz (SAB 80188) versions

The SAB 80188 is a highly integrated 8-bit microprocessor implemented in +5V advanced Siemens MYMOS technology. It effectively combines 15 to 20 of the most common SAB 8088 system components onto one chip. The 8 MHz SAB 80188 provides twice the throughput of the standard 5 MHz SAB 8088. The SAB 80188 is upward-compatible with SAB 8086 and SAB 8088 software, and adds 10 new instruction types to the existing set. The SAB 80188 comes in a 68-pin packages and requires single +5V power supply.

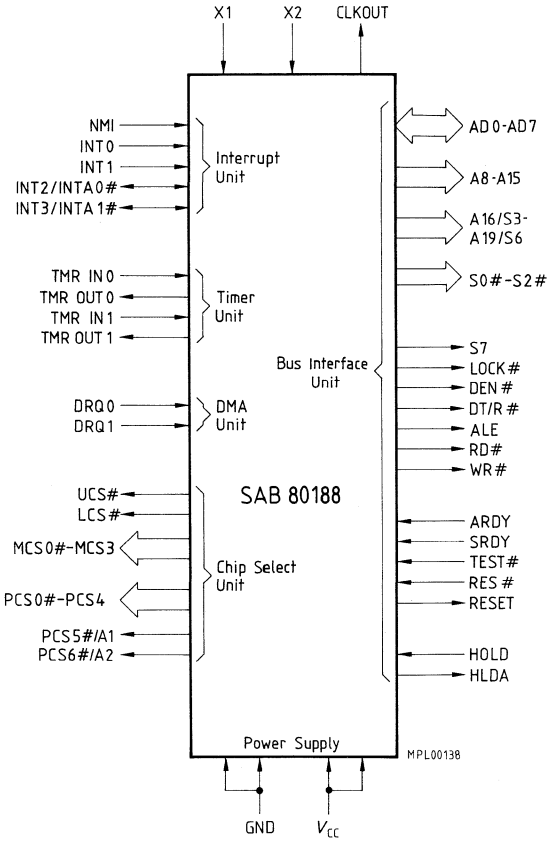
Ordering Information

Type	Ordering code	Package	Description
SAB 80188-N	Q67120-C252	PL-CC-68	8-bit microprocessor, 8 MHz
SAB 80188-1-N	Q67120-C299	PL-CC-68	8-bit microprocessor, 10 MHz

Pin Names

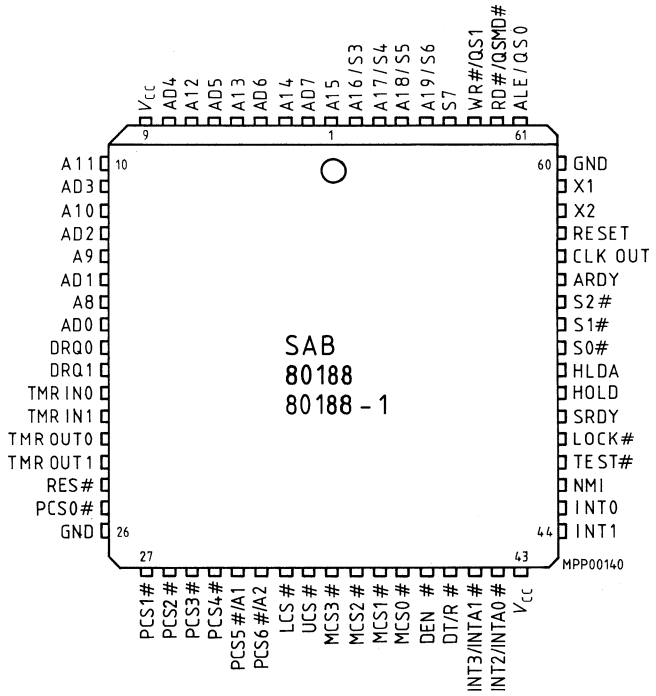
AD0–AD7	Address/Data Bus
A8–A15	Address Bus
A16/S3–A19/S6	Address Bus/Status Lines
S0–S2	Status Lines
S7	Status Line
$\overline{\text{LOCK}}$	Bus Lock
$\overline{\text{DEN}}$	Data Enable
DT/$\overline{\text{R}}$	Data Transmit/Receive
ALE/QS0	Address Latch Enable/Queue Status
$\overline{\text{RD}}$/$\overline{\text{QSMD}}$	Read Strobe/Queue Status Mode
$\overline{\text{WR}}$/QS1	Write Strobe/Queue Status
ARDY	Asynchronous Ready
SRDY	Synchronous Ready
$\overline{\text{TEST}}$	Test Line
$\overline{\text{RES}}$	Reset Input
RESET	Reset Output
HOLD	Hold Request
HLDA	Hold Acknowledge
NMI	Non-Maskable Interrupt
INT0, INT1	Interrupt Request
INT2/INTA0	Interrupt Request/Acknowledge
INT3/INTA1	Interrupt Request/Acknowledge
TMR IN0/TMR OUT0	Timer 0 Input/Output
TMR IN1/TMR OUT1	Timer 1 Input/Output
DRQ0, DRQ1	Data Request
$\overline{\text{UCS}}$, $\overline{\text{LCS}}$	Upper/Lower Chip Select
$\overline{\text{MCS0}}$–$\overline{\text{MCS3}}$	Memory Chip Select
$\overline{\text{PCS0}}$–$\overline{\text{PCS4}}$	Peripheral Chip Select
$\overline{\text{PCS5}}$/A1, $\overline{\text{PCS6}}$/A2	Peripheral Chip Select/Address
V_{CC}	Power Supply (+5 V)
GND	Ground (0 V)

Logic Symbol



Pin Configuration

Leaded Chip Carrier Package (top view)



Pin Definitions and Functions

Symbol	Pin	Input (I) Output (O)	Function
AD7–AD0	2, 4, 6, 8, 11, 13, 15, 17	I/O	**ADDRESS/DATA BUS (0 TO 7)** These signals constitute the time-multiplexed memory or I/O address (T1) and data (T2, T3, TW, and T4) bus. The bus is active high.
A15–A8	1, 3, 5, 7 10, 12, 14, 16	O	**ADDRESS BUS (8 TO 15)** Contains valid address from T1 to T4. The bus is active high.
DRQ0 DRQ1	18 19	I I	**DMA REQUEST** Is driven high by an external device when it desires that a DMA channel (channel 0 or 1) performs a transfer. These signals are active high, level-triggered, and internally synchronized.
TMR IN 0, TMR IN 1	20 21	I I	**TIMER INPUTS** Are used either as clock or control signals, depending upon the programmed timer mode. These inputs are active high (or low-to-high transitions are counted) and internally synchronized.
TMR OUT 0, TMR OUT 1	22 23	O O	**TIMER OUTPUTS** Are used to provide single pulse or continuous waveform generation, depending upon the timer mode selected.
$\overline{RES}$	24	I	**$\overline{SYSTEM\ RESET}$** Causes the SAB 80188 to immediately terminate its present activity, clear the internal logic, and enter a dormant state. This signal may be asynchronous to the SAB 80188 clock. The SAB 80188 begins fetching instructions approximately 7 clock cycles after $\overline{RES}$ is returned high. $\overline{RES}$ is required to be low for greater than 4 clock cycles and is internally synchronized. For proper initialization, the low-to-high transition of $\overline{RES}$ must occur no sooner than 50 microseconds after power up. This input is provided with a Schmitt trigger to facilitate power-on $\overline{RES}$ generation via an RC network. When $\overline{RES}$ occurs, the SAB 80188 will drive the status lines to an inactive level for one clock, and then tristate them.
$\overline{PCS0}$ $\overline{PCS1–4}$	25 27, 28, 29, 30	O O	**$\overline{PERIPHERAL\ CHIP\ SELECT\ 0\ TO\ 4}$** These signals are active low when a reference is made to the defined peripheral area (64 Kbyte I/O space). These lines are not tristated during bus hold. The address ranges activating $\overline{PCS0–4}$ are software-programmable.
$\overline{PCS5}$/A1	31	O	**$\overline{PERIPHERAL\ CHIP\ SELECT\ 5}$ LATCHED A1** May be programmed to provide a sixth peripheral chip select, or to provide an internally latched A1 signal. The address range activating $\overline{PCS5}$ is software-programmable. When programmed to provide latched A1, rather than $\overline{PCS5}$, this pin will retain the previously latched value of A1 during a bus HOLD. A1 is active high.
$\overline{PCS6}$/A2	32	O	**$\overline{PERIPHERAL\ CHIP\ SELECT\ 6}$ LATCHED A2** May be programmed to provide a seventh peripheral chip select, or to provide an internally latched A2 signal. The address range activating $\overline{PCS6}$ is software-programmable. When programmed to provide latched A2, rather than $\overline{PCS6}$, this pin will retain the previously latched value of A2 during a bus HOLD. A2 is active high.

Pin Definitions and Functions (cont'd)

Symbol	Pin	Input (I) Output (O)	Function
$\overline{LCS}$	33	O	**$\overline{LOWER\ MEMORY\ CHIP\ SELECT}$** Is active low whenever a memory reference is made to the defined lower portion (1 K to 256 K) of memory. This line is not tristated during bus HOLD. The address range activating $\overline{LCS}$ is software-programmable.
$\overline{UCS}$	34	O	**$\overline{UPPER\ MEMORY\ CHIP\ SELECT}$** Is an active low output whenever a memory reference is made to the defined upper portion (1 K to 256 K block) of memory. This line is not tristated during bus HOLD. The address range activating $\overline{UCS}$ is software-programmable.
$\overline{MCS0-3}$	38, 37, 36, 35	O	**$\overline{MIDRANGE\ MEMORY\ CHIP\ SELECT\ 0\ TO\ 3}$** These signals are active low when a memory reference is made to the defined midrange portion of memory (8 K to 512 K). These lines are not tristated during bus HOLD. The address ranges activating $\overline{MCS0-3}$ are software-programmable.
$\overline{DEN}$	39	O	**$\overline{DATA\ ENABLE}$** Is provided as an SAB 8286A/8287A data bus transceiver output enable. $\overline{DEN}$ is active low during each memory and I/O access. $\overline{DEN}$ is high whenever DT/$\overline{R}$ changes its state.
DT/$\overline{R}$	40	O	**DATA TRANSMIT/$\overline{RECEIVE}$** Controls the direction of data flow through the external SAB 8286A/8287A data bus transceiver. When low data is transferred to the SAB 80188. When high the SAB 80188 places write data on the data bus.
INT0, INT1, INT2/$\overline{INTA0}$ INT3/$\overline{INTA1}$	45, 44 42 41	I I/O I/O	**MASKABLE INTERRUPT REQUEST** Can be requested by strobing one of these pins. When configured as inputs, these pins are active high. Interrupt requests are synchronized internally. INT2 and INT3 may be configured via software to provide active low interrupt-acknowledge output signals. All interrupt inputs may be configured via software to be either edge or level-triggered. To ensure recognition, all interrupt requests must remain active until the interrupt is acknowledged. When iRMX mode is selected, the function of these pins changes (see Interrupt Controller section of this data sheet).
NMI	46	I	**NON-MASKABLE INTERRUPT** Is an edge-triggered input which causes a type 2 interrupt. NMI is not maskable internally. A transition from low to high initiates the interrupt at the next instruction boundary. NMI is latched internally. An NMI duration of one clock or more will guarantee service. This input is internally synchronized.
$\overline{TEST}$	47	I	**$\overline{TEST}$** Is examined by the WAIT instruction. If the $\overline{TEST}$ input is high when "WAIT" execution begins, instruction execution will suspend. $\overline{TEST}$ will be resampled until it goes low, at which time execution will be resumed. If interrupts are enabled while the SAB 80188 is waiting for $\overline{TEST}$, interrupts will be serviced. This input is synchronized internally.

Pin Definitions and Functions (cont'd)

Symbol	Pin	Input (I) Output (O)	Function
$\overline{\text{LOCK}}$	48	O	**$\overline{\text{LOCK}}$** This output indicates that other system bus masters are not to gain control of the system bus while $\overline{\text{LOCK}}$ is active low. The $\overline{\text{LOCK}}$ signal is requested by the LOCK prefix instruction and is activated at the beginning of the first data cycle associated with the instruction following the $\overline{\text{LOCK}}$ prefix. It remains active until the completion of the instruction following the $\overline{\text{LOCK}}$ prefix. No prefetches will occur while $\overline{\text{LOCK}}$ is asserted. $\overline{\text{LOCK}}$ is active low, is driven high for one clock during reset, and then tristated.
SRDY	49	I	**SYNCHRONOUS READY** Must be synchronized externally to the SAB 80188. The use of SRDY provides a relaxed system-timing specification on the ready input. This is accomplished by eliminating the one-half clock cycle which is required for internally resolving the signal level when using the ARDY input. This line is active high. If this line is connected to V_{CC} no wait states are inserted. Asynchronous ready (ARDY) or synchronous ready (SRDY) must be active before a bus cycle is terminated. If unused, this line should be tied low.
HOLD HLDA	50 51	I O	**HOLD/HOLD ACKNOWLEDGE** Indicates that another bus master is requesting the local bus. The HOLD input is active high. HOLD may be asynchronous with respect to the SAB 80188 clock. The SAB 80188 will issue a HLDA (high) in response to a HOLD request at the end of T4 or T1. Simultaneous with issuing HLDA, the SAB 80188 will tristate the local bus and control lines. After HOLD is detected as being low, the SAB 80188 will lower HLDA. When the SAB 80188 needs to run another bus cycle, it will again drive the local bus and control lines.
$\overline{\text{S0}}$, $\overline{\text{S1}}$, $\overline{\text{S2}}$	52–54	O	**$\overline{\text{BUS CYCLE STATUS}}$** $\overline{\text{S0}}$ to $\overline{\text{S2}}$ are encoded to provide bus transaction information: SAB 80188 Bus Cycle Status Information $\overline{\text{S2}}$ $\overline{\text{S1}}$ $\overline{\text{S0}}$ — Bus Cycle Initiated 0 0 0 — Interrupt Acknowledge 0 0 1 — Read I/O 0 1 0 — Write I/O 0 1 1 — Halt 1 0 0 — Instruction Fetch 1 0 1 — Read Data from Memory 1 1 0 — Write Data to Memory 1 1 1 — Passive (no bus cycle) The status pins are tristated during "HOLD." $\overline{\text{S2}}$ may be used as a logical M/$\overline{\text{IO}}$ indicator, and $\overline{\text{S1}}$ as a DT/$\overline{\text{R}}$ indicator. The status lines are driven high for one clock during reset, and then tristated until a bus cycle begins.

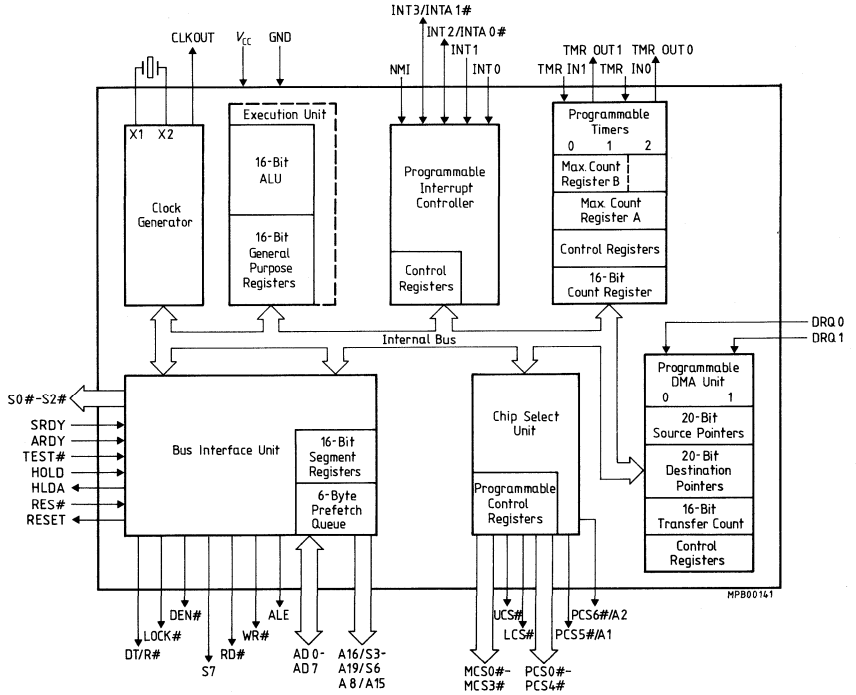
Pin Definitions and Functions (cont'd)

Symbol	Pin	Input (I) Output (O)	Function
ARDY	55	I	**ASYNCHRONOUS READY** Informs the SAB 80188 that the addressed memory space or I/O device will complete a data transfer. The ARDY input pin will accept an asynchronous input and is active high. Only the rising edge is internally synchronized by the SAB 80188. This means that the falling edge of ARDY must be synchronized to the SAB 80188 clock. If connected to V_{CC} no wait states are inserted. Asynchronous ready (ARDY) or synchronous ready (SRDY) must be active to terminate a bus cycle. If unused, this line should be tied low.
CLKOUT	56	O	**CLOCK OUTPUT** Provides the system with a 50% duty cycle waveform. All device pin timings are specified relative in CLKOUT.
RESET	57	O	**RESET** This output indicates that the SAB 80188 CPU is being reset, and can be used as a system reset. It is active high, synchronized with the processor clock, and lasts an integer number of clock periods corresponding to the length of the $\overline{RES}$ signal.
X1, X2	59, 58	I	**CRYSTAL INPUTS** X1 and X2 provide an external connection for a fundamental-mode parallel resonant crystal for the internal crystal oscillator. X1 can interface to an external clock instead of a crystal. In this case, minimize the capacitance on X2 or drive X2 with complemented X1. The input or oscillator frequency is internally divided by two to generate the clock signal (CLKOUT).
ALE/QS0	61	O	**ADDRESS LATCH ENABLE/QUEUE STATUS 0** Is provided by the SAB 80188 to latch the address into the SAB 8282A/8283A address latches. ALE is active high. Addresses are guaranteed to be valid on the trailing edge of ALE. The ALE rising edge is generated off the rising edge of the CLKOUT immediately preceding T1 of the associated bus cycle, effectively half a clock cycle earlier than in the standard SAB 8088. The trailling edge is generated off the CLKOUT rising edge in T1 like in the SAB 8088. Note that ALE is never tristated.
$\overline{RD}$/QSMD	62	O	**$\overline{READY\ STROBE}$** Indicates that the SAB 80188 is performing a memory or I/O read cycle. $\overline{RD}$ is active low for T2, T3 and TW of any read cycle. It is guaranteed not to go low in T2 until after the address bus is tristated. $\overline{RD}$ is active low and tristated during "HOLD". $\overline{RD}$ is driven high for one clock during reset, and then the output driver is tristated. A weak internal pullup mechanism on the $\overline{RD}$ line holds it high when the line is not driven. During reset, the pin is sampled to determine whether the SAB 80188 should provide ALE, $\overline{WR}$ and $\overline{RD}$, or if the queue status should be provided. $\overline{RD}$ should be connected to GND to provide queue status data.

Pin Definitions and Functions (cont'd)

<table>
<tr><th>Symbol</th><th>Pin</th><th>Input (I)
Output (O)</th><th>Function</th></tr>
<tr><td>$\overline{WR}$/QS1</td><td>63</td><td>O</td><td>$\overline{WRITE\ STROBE}$/QUEUE STATUS 1
Indicates that the data on the bus is to be written into a memory or an I/O device. $\overline{WR}$ is active for T2, T3, and TW of any write cycle. It is active low and tristated during "HOLD." It is driven high for one clock during reset, and then tristated. When the SAB 80188 is in queue status mode, the ALE/QS0 and $\overline{WR}$/QS1 pins provide information about processor/instruction queue interaction.
<table>
<tr><th>QS1</th><th>QS0</th><th>Queue Operation</th></tr>
<tr><td>0</td><td>0</td><td>No Queue Operation</td></tr>
<tr><td>0</td><td>1</td><td>First Op Code Byte Fetched from the Queue</td></tr>
<tr><td>1</td><td>1</td><td>Subsequent Byte Fetched from the Queue</td></tr>
<tr><td>1</td><td>0</td><td>Empty the Queue</td></tr>
</table></td></tr>
<tr><td>S7</td><td>64</td><td>O</td><td>STATUS LINE 7
This signal is always high to indicate that the SAB 80188 has an 8-bit data bus, and is tristated off during bus hold.</td></tr>
<tr><td>A19/S6,
A18/S5,
A17/S4,
A16/S3</td><td>65
66
67
68</td><td>O
O
O
O</td><td>ADDRESS BUS OUTPUTS (16 TO 19) and
BUS CYCLE STATUS (3 TO 6)
They reflect the four most significant address bits during T1. These signals are active high. During T2, T3, TW, and T4, status information is available on these lines as encoded below:
<table>
<tr><th></th><th>Low</th><th>High</th></tr>
<tr><td>S6</td><td>Processor Cycle</td><td>DMA Cycle</td></tr>
</table>
S3, S4, and S5 are defined as being low during T2 to T4.</td></tr>
<tr><td>V_{CC}</td><td>9, 43</td><td>I</td><td>POWER SUPPLY (+5V)</td></tr>
<tr><td>GND</td><td>26, 60</td><td>I</td><td>GROUND (0V)</td></tr>
</table>

Block Diagram



Functional Description

The SAB 8086, 8088, 80186, 80188 and 80286 families all contain the same basic set of registers, instructions, and addressing modes. The SAB 80188 processor is upward-compatible with the SAB 8086 and SAB 8088 CPUs.

Register Set

The SAB 80188 base architecture has fourteen registers. These registers are grouped into the following categories.

General registers

Eight 16-bit general-purpose registers may be used to contain arithmetic and logical operands. Four of these (AX, BX, CX, and DX) can be used as 16-bit registers or split into pairs of separate 8-bit registers.

Segment registers

Four 16-bit special-purpose registers select, at any time given, the segments of memory that are immediately addressable for code, stack, and data.

Base and index registers

Four of the general-purpose registers may also be used to determine offset addresses of operands in memory. These registers may contain base addresses or indexes to particular locations within a segment. The addressing mode selects the specific registers for operand and address calculations.

Status and control registers

Two 16-bit special-purpose registers record or alter certain aspects of the SAB 80188 processor state. These are the instruction pointer register, which contains the offset address of the next sequential instruction to be executed, and the status word register, which contains status and control flag bits.

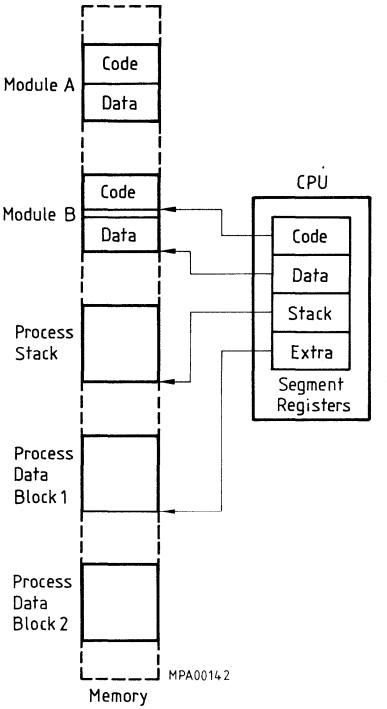
Memory Organization

Memory is organized in sets of segments. Each segment is a linear contiguous sequence of up to 64 K (2^{16}) 8-bit bytes. Memory is addressed using a two-component address (a pointer) that consists of a 16-bit base segment and a 16-bit offset. The 16-bit base values are contained in one of four internal segment registers (code, data, stack, extra). The physical address is calculated by shifting the base value left by four bits and adding the 16-bit offset value to yield a 20-bit physical address. This allows for a 1 Mbyte physical address size.

All instructions that address operands in memory must specify the base segment and the 16-bit offset value. For speed and compact instruction encoding, the segment register used for physical address generation is implied by the addressing mode used. These rules follow the way programs are written as independent modules that require areas for code and data, a stack, and access to external data areas.

Special segment override instruction prefixes allow the implicit segment register selection rules to be overridden for special cases. The stack, data, and extra segments may coincide for simple programs.

Memory Segmentation



I/O Space

The I/O space consists of 64 K 8-bit or 32 K 16-bit ports. Separate instructions address the I/O space with either an 8-bit port address, specified in the instruction, or a 16-bit port address in the DX register. 8-bit port addresses are zero-extended such that A15 to A8 are low I/O port addresses 00F8(H) through 00FF(H) are reserved.

Interrupts

An interrupt transfers execution to a new program location. The old program address (CS:IP) and machine state (status word) are saved on the stack to allow resumption of the interrupted program. Interrupts fall into three classes: hardware-initiated, INT instructions, and instruction exceptions. Hardware-initiated interrupts occur in response to an external input and are classified as non-maskable or maskable.

Interrupt Sources

The SAB 80188 can service interrupts generated by software or hardware. The software interrupts are generated by specific instructions (INT, ESC, unused OP, etc.) or the results of conditions specified by instructions (array bounds check, INT0, DIV, IDIV, etc.). All interrupt sources are serviced by an indirect call through an element of a vector table. This vector table is indexed by using the interrupt vector type, multiplied by four. All hardware-generated interrupts are sampled at the end of each instruction. Thus, the software interrupts will begin service first. Once the service routine is entered and interrupts are enabled, any hardware source of sufficient priority can interrupt the service routine in progress.

Interrupt Vector Table

Interrupt Name	Vector Type	Default Priority [5]	Related Instructions
Divide Error Exception	0	1 [1]	DIV, IDIV
Single-Step Interrupt	1	12 [2] 2	All
NMI	2	1	All
Breakpoint Interrupt	3	1 [1]	INT
INT0 Detected Overflow Exception	4	1 [1]	INT0
Array Bounds Exception	5	1 [1]	BOUND
Unused Op Code Exception	6	1 [1]	Undefined Op Codes
ESC Op Code Exception	7	1 [1] [3]	ESC Op Codes
Timer 0 Interrupt	8	2A [4]	
Timer 1 Interrupt	18	2B [4]	
Timer 2 Interrupt	19	2C [4]	
Reserved	9	3	
DMA 0 Interrupt	10	4	
DMA 1 Interrupt	11	5	
INT0 Interrupt	12	6	
INT1 Interrupt	13	7	
INT2 Interrupt	14	8	
INT3 Interrupt	15	9	

1) These are generated as the result of an instruction execution.
2) This is handled as in the SAB 8088.
3) An escape op code will cause a trap only if the proper bit is set in the peripheral control block relocation register.
4) All three timers constitute one source of request to the interrupt controller. The timer interrupts all have the same default priority level with respect to all other interrupt sources. However, they have a defined priority ordering amongst themselves. (Priority 2A is higher priority than 2B.) Each timer interrupt has a separate vector type number.
5) Default priorities for the interrupt sources are used only if the user does not program each source into a unique priority level.

Initialization and Processor Reset

Processor initialization or startup is accomplished by driving the $\overline{RES}$ input pin low. $\overline{RES}$ forces the SAB 80188 to terminate all execution and local bus activity. No instruction or bus activity will occur as long as $\overline{RES}$ is active. After $\overline{RES}$ becomes inactive and an internal processing interval elapses, the SAB 80188 begins execution with the instruction at physical location FFFF0(H). $\overline{RES}$ also sets some registers to predefined values.

Initial Register State after RESET

Status Word	F002(H)
Instruction Pointer	0000(H)
Code Segment	FFFF(H)
Data Segment	0000(H)
Extra Segment	0000(H)
Stack Segment	0000(H)
Relocation Register	20FF(H)
UMCS	FFFB(H)

Clock Generator

The SAB 80188 provides an on-chip clock generator for both internal and external clock generation. The clock generator features a crystal oscillator, a divide-by-two counter, synchronous and asynchronous ready inputs and reset circuitry.

Oscillator

The oscillator circuit of the SAB 80188 is designed to be used with a parallel resonant fundamental mode crystal. This is used as the time base for the SAB 80188. The crystal frequency selected will be twice the CPU clock frequency. Use of an LC or RC circuit is not recommended with this oscillator. If an external oscillator is used, it can be connected directly to input pin X1 in lieu of a crystal. The output of the oscillator is not directly available outside the SAB 80188.

Ready Synchronization

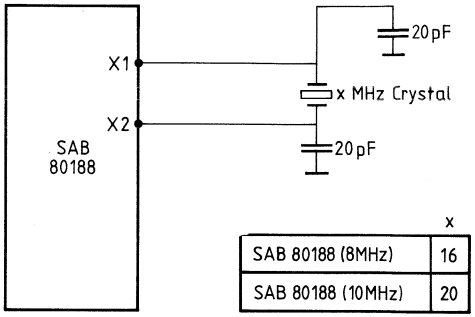
The SAB 80188 provides both synchronous and asynchronous ready inputs. Asynchronous ready synchronization is accomplished by circuitry which samples ARDY in the middle of T2, T3 and again in the middle of each TW until ARDY is sampled high.

A second ready input (SRDY) is provided to interface with externally synchronized ready signals. This input is sampled at the end of T2, T3 and again at the end of each TW until it is sampled high.

Reset Logic

The SAB 80188 provides both a $\overline{RES}$ input pin and a synchronized reset pin for use with other system components. The $\overline{RES}$ input pin on the SAB 80188 is provided with hysteresis in order to facilitate power-on reset generation via an RC network.

Recommended SAB 80188 Crystal Configuration



	x
SAB 80188 (8MHz)	16
SAB 80188 (10MHz)	20

MPA00143

The following parameters may be used for choosing a crystal:

Temperature range	0 to 70 °C
ESR (equivalent series resistance)	30 Ω max.
C0 (shunt capacitance of crystal)	7.0 pF max.
C1 (load capacitance)	20 pF ± 2 pF
Drive level	1 mW max.

Local Bus Controller

The SAB 80188 provides a local bus controller to generate the local bus control signals. In addition, it employs a HOLD/HLDA protocol for relinquishing the local bus to other bus masters. It also provides control lines that can be used to enable external buffers and to direct the flow of data on and off the local bus.

When the SAB 80188 relinquishes control of the local bus, it tristates $\overline{DEN}$, $\overline{RD}$, $\overline{WR}$, $\overline{S0}$ to $\overline{S2}$, $\overline{LOCK}$, AD0 to AD7, A8 to A19, S7 and DT/$\overline{R}$ to allow another master to drive these lines directly.

Local bus controller and reset

Upon receipt of a reset pulse from the $\overline{RES}$ input, the local bus controller will perform the following actions:

- Drive $\overline{DEN}$, $\overline{RD}$, and $\overline{WR}$ high for one clock cycle, then float.
  Note: $\overline{RD}$ is also provided with an internal pullup device to prevent the processor from inadvertently entering queue status mode during reset.
- Drive $\overline{S0}$ to $\overline{S2}$ to the passive state (all high) and then float.
- Drive $\overline{LOCK}$ high and then float.
- Tristate AD0 to AD7, A8 to A19, S7, DT/$\overline{R}$.
- Drive ALE low (ALE is never tristated).
- Drive HLDA low.

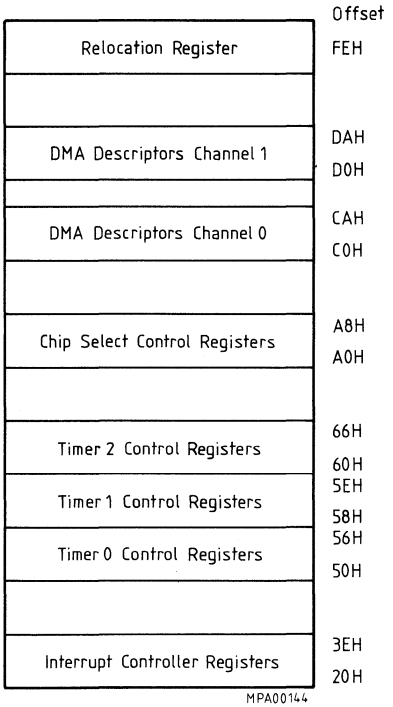
Internal Peripheral Interface

All the SAB 80188 integrated peripherals are controlled via 16-bit registers contained within an internal 256 byte control block. This control block may be mapped into either memory or I/O space. Internal logic will recognize the address and respond to the bus cycle. During bus cycles to internal registers, the bus controller will signal the operation externally (i.e., the $\overline{RD}$, $\overline{WR}$, status, address, data, etc., lines will be driven as in a normal bus cycle), but D7 to D0, SRDY, and ARDY will be ignored.

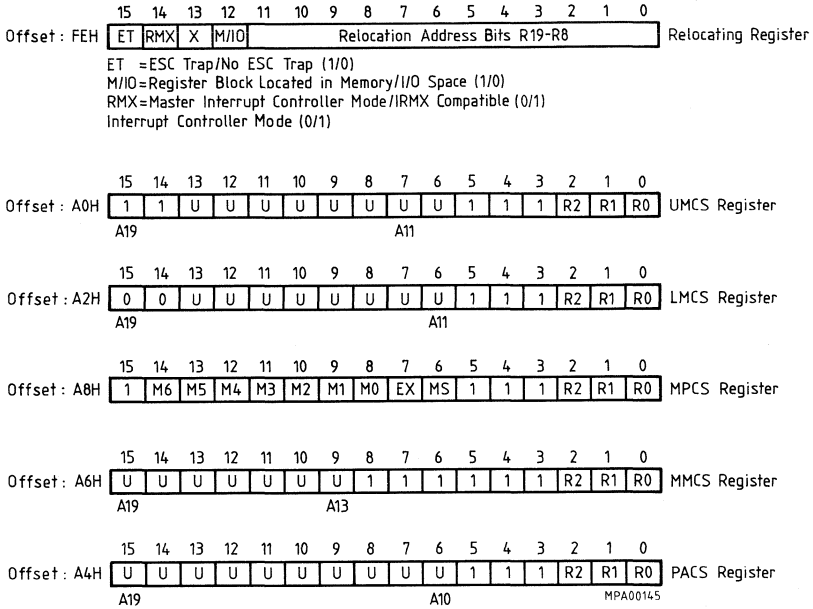
The control block base address is programmed via a 16-bit relocation register contained within the control block at offset FEH from the base address of the control block. It provides the upper 12 bits of the base address of the control block.

The integrated SAB 80188 peripherals operate semi-autonomously from the CPU. Access to them for the most part is via software read/write of the control and data locations in the control block. Most of these registers can be both read and written. A few dedicated lines, such as interrupts and DMA request provide real-time communication between the CPU and peripherals as in a more conventional system utilizing discrete peripheral blocks.

Internal Register Map



Chip Select Registers



UMCS Programming Values

Starting Address (Base Address)	Memory Block Size	UMCS Value (Assuming R0=R1=R2=0)
FFC00	1 K	FFF8H
FF800	2 K	FFB8H
FF000	4 K	FF38H
FE000	8 K	FE38H
FC000	16 K	FC38H
F8000	32 K	F838H
F0000	64 K	F038H
E0000	128 K	E038H
C0000	256 K	C038H

LMCS Programming Values

Upper Address	Memory Block Size	LMCS Value (Assuming R0=R1=R2=0)
003FFH	1 K	0038H
007FFH	2 K	0078H
00FFFH	4 K	00F8H
01FFFH	8 K	01F8H
03FFFH	16 K	03F8H
07FFFH	32 K	07F8H
0FFFFH	64 K	0FF8H
1FFFFH	128 K	1FF8H
3FFFFH	256 K	3FF8H

MPCS Programming Values

Total Block Size	Individual Select Size	MPCS Bits 14 to 8
8 K	2 K	0000001B
16 K	4 K	0000010B
32 K	8 K	0000100B
64 K	16 K	0001000B
128 K	32 K	0010000B
256 K	64 K	0100000B
512 K	128 K	1000000B

MS, EX Programming Values

Bit	Description
MS	1=Peripherals Mapped into Memory Space. 0=Peripherals Mapped into I/O Space.
EX	0=5 $\overline{\text{PCS}}$ Lines. A1, A2 Provided. 1=7 $\overline{\text{PCS}}$ Lines. A1, A2 Are Not Provided.

Ready Bits Programming

R2	R1	R0	Number of Wait States Generated
0	0	0	0 Wait States, External RDY Also Used
0	0	1	1 Wait State Inserted, External RDY Also Used
0	1	0	2 Wait States Inserted, External RDY Also Used
0	1	1	3 Wait States Inserted, External RDY Also Used
1	0	0	0 Wait States, External RDY Ignored
1	0	1	1 Wait State Inserted, External RDY Ignored
1	1	0	2 Wait States Inserted, External RDY Ignored
1	1	1	3 Wait States Inserted, External RDY Ignored

Chip Select Logic

The SAB 80188 contains logic which provides programmable chip select generation for both, memories and peripherals. In addition, it can be programmed to provide ready (or wait state) generation. It can also provide latched address bits A1 and A2. The chip select lines are active for all memory and I/O cycles in their programmed areas, whether they be generated by the CPU or by the integrated DMA unit.

Memory Chip Selects

The SAB 80188 provides 6 memory chip select outputs for 3 address areas: upper memory, lower memory, and midrange memory. One each is provided for upper memory and lower memory, while four are provided for midrange memory.

Upper Memory $\overline{CS}$

The SAB 80188 provides a chip select, called $\overline{UCS}$, for the top of memory. The top of memory is usually used as the system memory because after reset the SAB 80188 begins executing at memory location FFFF0H. After reset, the UMCS register is programmed for a 1 K area. It must be reprogrammed if a larger upper memory area is desired.

Lower Memory $\overline{CS}$

The SAB 80188 provides a chip select for low memory called $\overline{LCS}$. The bottom of memory contains the interrupt vector table, starting at location 00000H. After reset, the LMCS register value is undefined. However, the $\overline{LCS}$ chip select line will not become active until the LMCS register is accessed.

Midrange Memory $\overline{CS}$

The SAB 80188 provides four $\overline{MCS}$ lines which are active within a user-locatable memory block. This block can be located anywhere within the 1 Mbyte memory address space exclusive of the areas defined by $\overline{UCS}$ and $\overline{LCS}$. Both the base address and size of this memory block are programmable.

Each of the four chip select lines is active for one of the four equal contiguous divisions of the midrange block. Thus, if the total block size is 32 K, each chip select is active for 8 K of memory with $\overline{MCS0}$ being active for the first range and $\overline{MCS3}$ being active for the last range. After reset, the contents of both of these registers is undefined.

However, none of the $\overline{MCS}$ lines will be active until both the MMCS and MPCS registers are accessed.

Peripheral Chip Selects

The SAB 80188 can generate chip selects for up to seven peripheral devices. These chip selects are active for seven contiguous blocks of 128 bytes above a programmable base address. This base address may be located in either memory or I/O space.

Seven $\overline{CS}$ lines called $\overline{PCS0-6}$ are generated by the SAB 80188. The base address is user-programmable; however it can only be a multiple of 1 Kbytes, i.e. the least significant 10 bits of the starting address are always 0.

$\overline{PCS5}$ and $\overline{PCS6}$ can also be programmed to provide latched address bits A1, A2. If so programmed, they cannot be used as peripheral selects.

The starting address of the peripheral chip select block is defined by the PACS register. This register is located at offset A4H in the internal control block. Bits 15 to 6 of this register correspond to bits 19 to 10 of the 20-bit Programmable Base Address (PBA) of the peripheral chip-select block.

PCS Address Ranges

PCS Line	Active between Locations
PCS0	PBA – PBA+127
PCS1	PBA+128 – PBA+255
PCS2	PBA+256 – PBA+383
PCS3	PBA+384 – PBA+511
PCS4	PBA+512 – PBA+639
PCS5	PBA+640 – PBA+767
PCS6	PBA+768 – PBA+895

Ready Generation Logic

The SAB 80188 can generate a ready signal internally for each of the memory or peripheral $\overline{CS}$ lines. The number of wait states to be inserted for each peripheral or memory is programmable to provide 0 to 3 wait states for all accesses to the area for which the chip select is active. In addition, the SAB 80188 may be programmed to either ignore external ready for each chip select range individually or to factor external ready with the integrated ready generator.

Ready control consists of 3 bits for each $\overline{CS}$ line or group of lines generated by the SAB 80188.

Chip Select/Ready Logic and Reset

Upon reset, the chip select/ready logic will perform the following actions:

- All chip select outputs will be driven high.
- Upon leaving reset, the $\overline{UCS}$ line will be programmed to provide chip selects to a 1 K block with the accompanying ready control bits set at 011 to allow the maximum number of internal wait states in conjunction with external ready consideration (i.e. UMCS resets to FFFBH).
- No other chip select or ready control registers have any predefined values after reset. They will not become active until the CPU accesses their control registers. Both the PACS and MPCS registers must be accessed before the $\overline{PCS}$ lines will become active.

DMA Channels

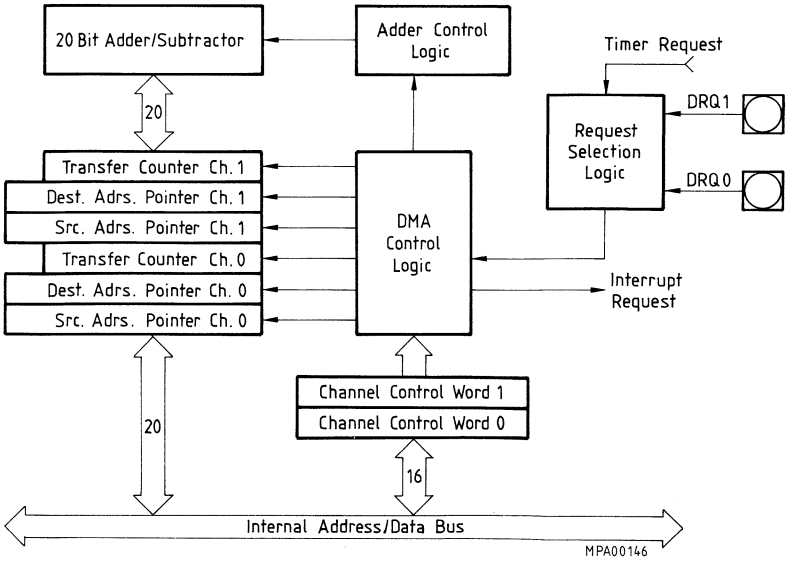
The SAB 80188 DMA controller provides two independent high-speed DMA channels. Data transfers can occur between memory and I/O spaces (e.g., memory-to-I/O) or within the same space (e.g., memory-to-memory or I/O-to-I/O). Each DMA channel maintains both a 20-bit source and destination pointer which can be optionally incremented or decremented after each data transfer. Each data transfer consumes 2 bus cycles (a minimum of 8 clocks), one cycle to fetch data and the other to store data. This provides a maximum data transfer rate of 1 Mbyte/s.

DMA Channel Control Word Register

Each DMA channel control word determines the mode of operation for the particular SAB 80188 DMA channel.

The DMA channel control registers may be changed while the channel is operating. However, any changes made during operation will affect the current DMA transfer.

DMA Unit Block Diagram

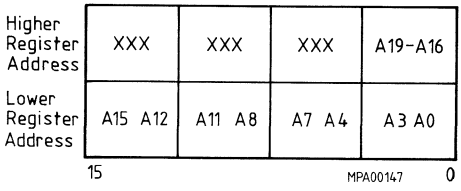


DMA Control Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M/IO#	Destination DEC	INC	M/IO#	Source DEC	INC	TC	INT	SYN		P	TDRQ	X	CHG/NOCHG#	ST/STOP#	X

X=Don't Care

DMA Memory Pointer Register Format



DMA Control Block Format

Register Name	Register Address	
	Chan. 0	Chan. 1
Control Word	CAH	DAH
Transfer Count	C8H	D8H
Destination Pointer (upper 4 bits)	C6H	D6H
Destination Pointer	C4H	D4H
Source Pointer (upper 4 bits)	C2H	D2H
Source Pointer	C0H	D0H

DMA Control Word Bit Description

ST/$\overline{\text{STOP}}$ Start/stop (1/0) channel.

CHG/$\overline{\text{NOCHG}}$ Change/do not change (1/0)ST/$\overline{\text{STOP}}$ bit. If this bit is set when writing to the control word, the ST/$\overline{\text{STOP}}$ bit will be programmed by the write to the control word. If this bit is cleared when writing the control word, the ST/$\overline{\text{STOP}}$ bit will not be altered. This bit is not stored; it will always be a 0 on read.

INT Enable interrupts to CPU on byte count termination.

TC If set, DMA will terminate when the contents of the transfer count register reach zero. The ST/$\overline{\text{STOP}}$ bit will also be reset at this point if TC is set. If this bit is cleared, the DMA unit will decrement the transfer count register for each DMA cycle, but the DMA transfer will not stop when the contents of the TC register reach zero.

SYN (2 bits) 00 No synchronization.
Note: The ST bit will be cleared automatically when the contents of the TC register reach zero regardless of the state of the TC bit.
01 Source synchronization. 10 Destination synchronization. 11 Unused.

TDRQ 0: Disable DMA requests from timer 2.
1: Enable DMA requests from timer 2.

Source: INC Increment source pointer by 1 after each transfer.
M/$\overline{\text{IO}}$ Source pointer is in M/IO space (1/0).
DEC Decrement source pointer by 1 after each transfer.

Destin.: INC Increment destination pointer by 1 after each transfer.
M/$\overline{\text{IO}}$ Destination pointer is in M/IO space (1/0).
DEC Decrement destination pointer by 1 after each transfer.

P Channel priority – relative to other channel.
0 low priority. 1 high priority.
Channels will alternate cycles if both set at same priority level.

Bit 0 Bit 0 is not used.

Bit 3 Bit 3 is not used.

DMA Destination and Source Pointer Register

Each DMA channel maintains a 20-bit source and a 20-bit destination pointer. Each of these pointers takes up two full 16-bit registers in the peripheral control block. The lower four bits of the upper register contain the upper four bits of the 20-bit physical address.

DMA Transfer Count Register

Each DMA channel maintains a 16-bit transfer count register (TC). This register is decremented after every DMA cycle, regardless of the state of the TC bit in the DMA control register.

DMA Requests

Data transfers may be either source or destination synchronized, that is either the source of the data or the destination of the data may request the data transfer. In addition, DMA transfers may be unsynchronized; that is, the transfer will take place continually until the correct number of transfers has occurred.

DMA Priority

The DMA channels may be programmed such that one channel is always given priority over the other, or they may be programmed such as to alternate cycles when both have DMA requests pending. DMA cycles always have priority over internal CPU cycles except between locked memory accesses or word accesses to odd memory locations; however, an external bus hold takes priority over an internal DMA cycle.

DMA Channels and Reset

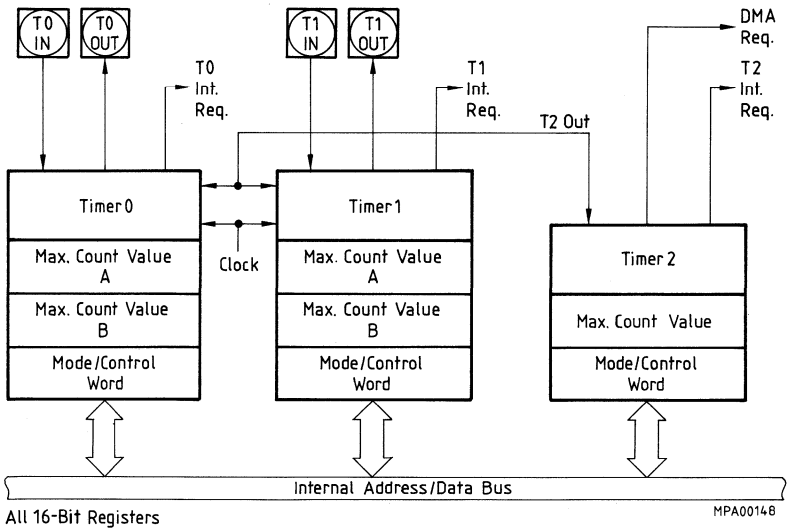
Upon reset, the DMA channels will perform the following actions:

- The start/stop bit for each channel will be reset to STOP.
- Any transfer in progress is aborted.

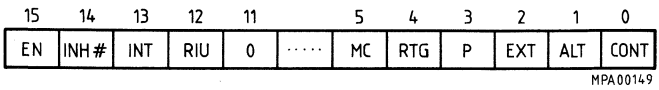
Timers

The SAB 80188 provides three internal 16-bit programmable timers. Two of these are highly flexible and are connected to four external pins (2 per timer). They can be used to count external events, time external events, generate nonrepetitive waveforms, etc. The third timer is not connected to any external pins, and is useful for real-time coding and time delay applications. In addition, this third timer can be used as a prescaler to the other two, or as a DMA request source.

Timer Block Diagram



Timer Mode/Control Register



Timer Control Block Format

Register Name	Register Offset		
	TMR 0	TMR 1	TMR 2
Mode/Control Word	56H	5EH	66H
Max. Count B	54H	5CH	not present
Max. Count A	52H	5AH	62H
Count Register	50H	58H	60H

Timer Operation

The timers are controlled by eleven 16-bit registers in the internal peripheral control block. The count register contains the current value of the timer. It can be read or written at any time independent of whether the timer is running or not. The value of this register will be incremented for each timer event. Each of the timers is equipped with a max. count register, which defines the maximum count the timer will reach. After reaching the max. count register value, the timer count value will be reset to zero during that same clock.

Each timer gets serviced every fourth CPU clock cycle, and thus can operate at speeds up to one-quarter the internal clock frequency (one-eighth the crystal rate). External clocking of the timers may be done at up to a rate of one-quarter of the internal CPU clock rate (2 MHz for an 8 MHz CPU clock).

The timers have several programmable options.

- All three timers can be set to halt or continue on a terminal count.
- Timers 0 and 1 can select between internal and external clocks, alternate between max. count registers and be set to retrigger on external events.
- The timers may be programmed to cause an interrupt on terminal count.

Count Registers

Each of the three timers has a 16-bit count register. The current contents of this register may be read or written by the processor at any time. If the register is written into while the timer is counting, the new value will take effect in the current count cycle.

Max. Count Registers

Timers 0 and 1 have two max. count registers, while timer 2 has a single max.count register. These contain the number of events the timer will count. In timers 0 and 1, the max.count register used can alternate between the two max.count values whenever the current maximum count is reached.

Timer Mode/Control Register

The mode/control register allows the user to program the specific mode of operation or check the current programmed status for any of the three integrated timers.

ALT:
The ALT bit determines which of two max. count registers is used for count comparison. If ALT = 0, register A for that timer is always used, while if ALT = 1, the comparison will alternate between register A and register B when each maximum count is reached. This alternation allows the user to change one max.count register while the other is being used, and thus provides a method of generating nonrepetitive waveforms. Square waves and pulse outputs of any duty cycle are a subset of available signals obtained by not changing the final count registers. The ALT bit also determines the function of the timer output pin. If ALT is zero, the output pin will go low for one clock, the clock after the maximum count is reached. If ALT is one, the output pin will reflect the current max.count register being used (0/1 for B/A).

CONT:
Setting the CONT bit causes the associated timer to run continuously, while resetting it causes the timer to halt upon maximum count. If CONT = 0 and ALT = 1, the timer will count to the max. count register A value, be reset, count to the register B value, be reset, and halt.

EXT:
The external bit selects between internal and external clocking for the timer. The external signal may be asynchronous with respect to the SAB 80188 clock. If EXT is set, the timer will count low-to-high transitions on the input pin. If cleared, it will count an internal clock while using the input pin for control. In this mode, the function of the external pin is defined by the RTG bit. The maximum input-to-output transition latency time may be as much as 6 clocks. However, clock inputs may be pipelined as closely together as every 4 clocks without losing clock pulses.

P:
The prescaler bit is ignored unless internal clocking has been selected (EXT = 0). If the P bit is a zero, the timer will count at one-fourth the internal CPU clock rate. If the P bit is a one, the output of timer 2 will be used as a clock for the timer. Note that the user must initialize and start timer 2 to obtain the prescaled clock.

RTG:
Retrigger bit is only active for internal clocking (EXT = 0). In this case it determines the control function provided by the input pin.

If RTG = 0, the input level gates the internal clock on and off. If the input pin is high, the timer will count; if the input pin is low, the timer will hold its value. As indicated previously, the input signal may be asynchronous with respect to the SAB 80188 clock.

When RTG = 1, the input pin detects low-to-high transitions. The first transition starts the timer running, clearing the timer value to zero on the first clock, and then incrementing thereafter. Further transitions on the input pin will again reset the timer to zero, from which it will start counting up again. If CONT = 0, when the timer has reached maximum count, the EN bit will be cleared, inhibiting further timer activity.

EN:
The enable bit provides programmer control over the timer's RUN/HALT status. When set, the timer is enabled to increment subject to the input pin constraints in the internal clock mode (discussed previously). When cleared, the timer will be inhibited from counting. All input pin transitions during which the time EN is zero will be ignored. If CONT is zero, the EN bit is automatically cleared upon maximum count.

INH:
The inhibit bit allows for selective updating of the enable (EN) bit. If INH is a one during the write to the mode/control word, then the state of the EN bit will be modified by the write. If INH is a zero during the write, the EN bit will be unaffected by the operation. This bit is not stored; it will always be a 0 on a read.

INT:
When set, the INT bit enables interrupts from the timer, which will be generated on every terminal count. If the timer is configured in dual max.count register mode, an interrupt will be generated each time the value in max.count register A is reached, and each time the value in max.count register B is reached. If this enable bit is cleared after the interrupt request has been generated, but before a pending interrupt is serviced, the interrupt request will still be in force. (The request is latched in the interrupt controller.)

MC:
The maximum count bit is set whenever the timer reaches its final maximum count value. If the timer is configured in dual max.count register mode, this bit will be set each time the value in max.count register A is reached, and each time the value in max.count register B is reached. This bit is set regardless of the timer's interrupt enable bit. The MC bit gives the user the ability to monitor timer status through software instead of through interrupts.
Programmer intervention is required to clear this bit.

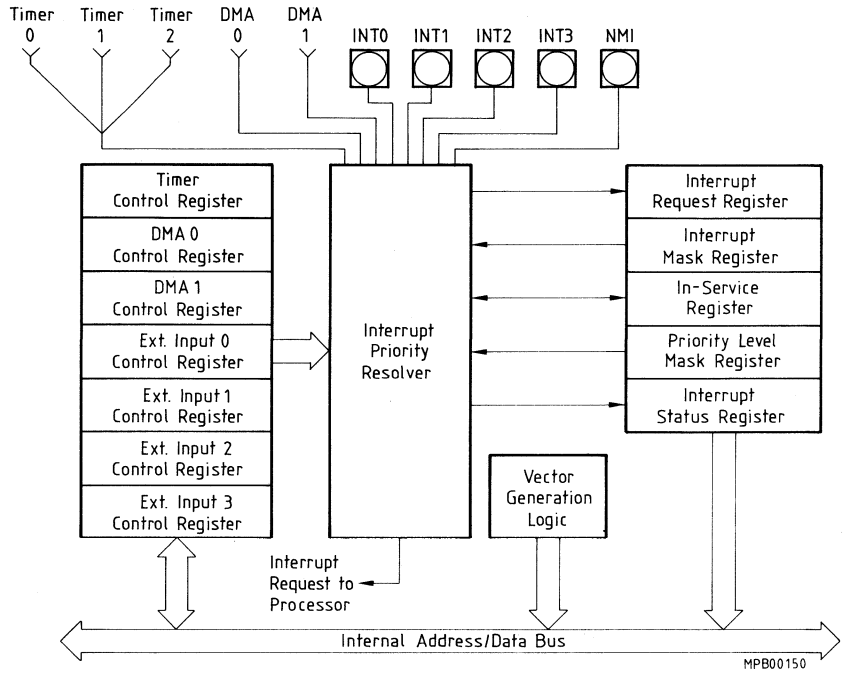
RIU:
The register in use bit indicates which max.count register is currently being used for comparison to the timer count value. A zero indicates register A. The RIU bit cannot be written, i.e. its value is not affected when the control register is written. It is always cleared when the ALT bit is zero.
Not all mode bits are provided for timer 2. Certain bits are hardwired as indicated below:
ALT = 0, EXT = 0, P = 0, RTG = 0, RIU = 0

Timers and Reset

Upon reset, the timers will perform the following actions:

- All EN (enable) bits are reset preventing timer counting.
- All SEL (select) bits are reset to zero. This selects max. count register A, resulting in the timer-out pins going high upon reset.

Interrupt Controller Block Diagram



Interrupt Controller

The SAB 80188 can receive interrupts from a number of sources, both internal and external. The internal interrupt controller serves to merge these requests on a priority basis, for individual service by the CPU. Internal interrupt sources (timers and DMA channels) can be disabled by their own control registers or by mask bits within the interrupt controller. The SAB 80188 interrupt controller has its own control registers that set the mode of operation for the controller.

The interrupt controller will resolve priority among requests that are pending simultaneously. Nesting is provided so interrupt service routines for lower priority interrupts may themselves be interrupted by higher priority interrupts.
The interrupt controller has a special iRMX 86 compatibility mode that allows the use of the SAB 80188 within the iRMX 86 operating system interrupt structure.

Master Mode Operation

Interrupt Controller External Interface

For external interrupt sources, five dedicated pins are provided. One of these pins is dedicated to NMI, non-maskable interrupt. This is typically used for power-fail interrupts, etc. The other four pins may function either as four interrupt input lines with internally generated interrupt vectors, as an interrupt line and an interrupt acknowledge line (called the "cascade mode") along with two other input lines with internally generated interrupt vectors, or as two interrupt input lines and two dedicated interrupt acknowledge output lines. When the interrupt lines are configured in cascade mode, the SAB 80188 interrupt controller will not generate internal interrupt vectors.

Interrupt Controller Modes of Operation

The basic modes of operation of the interrupt controller in master mode are similar to the SAB 8259A. The interrupt controller responds identical to internal interrupts in all three modes: the difference is only in the interpretation of function of the four external interrupt pins.

Fully nested mode

When in fully nested mode, four pins are used as direct interrupt requests. The vectors for these four inputs are generated internally. An in-service bit (IS) is provided for every interrupt source. If a lower priority device requests an interrupt while the in-service bit is set, no interrupt will be generated by the interrupt controller. In addition, if another interrupt request occurs from the same interrupt source while the in-service bit is set, no interrupt will be generated by the interrupt controller.
When a service routine is completed, the proper IS bit must be reset by writing the proper pattern to the EOI register.

Cascade mode

The SAB 80188 has four interrupt pins and two of them have dual functions. In fully nested mode, the four pins are used as direct interrupt inputs and the corresponding vectors are generated internally. In cascade mode, the four pins are configured into interrupt input-dedicated acknowledge signal pairs. INT0 is an interrupt input interfaced to an SAB 8259A, while INT2/$\overline{\text{INTA0}}$ serves as the dedicated interrupt acknowledge signal to the peripheral. The same is true for INT1 and INT3/$\overline{\text{INTA1}}$. Each pair can selectively be placed in the cascade or non-cascade mode by programming the proper value INT0 and INT1 control registers.
The primary cascade mode allows the capability to serve up to 128 external interrupt sources through the use of external master and slave SAB 8259As.

Special fully nested mode

This mode is entered by setting the SFNM bit in INT0 or INT1 control register. It enables complete nestability with external SAB 8259A masters. In special fully nested mode, the SAB 80188 interrupt controller will allow interrupts from an external pin regardless of the state of the in-service bit for an interrupt source in order to allow multiple interrupts from a single pin.

Polling operation

The controller may be used in a polled mode if interrupts are undesirable. When polling, the processor disables interrupts and then polls the interrupt controller whenever it is convenient. Polling the interrupt controller is accomplished by reading the poll word. Bit 15 in the poll word indicates to the processor that an interrupt of high enough priority is requesting service.

Interrupt Controller Registers (non-iRMX 86 mode)

	Offset
INT3 Control Register	3EH
INT2 Control Register	3CH
INT1 Control Register	3AH
INT0 Control Register	38H
DMA1 Control Register	36H
DMA0 Control Register	34H
Timer Control Register	32H
Interrupt Status Register	30H
Interrupt Request Register	2EH
In-Service Register	2CH
Priority Mask Register	2AH
Mask Register	28H
Poll Status Register	26H
Poll Register	24H
EOI Register	22H

Timer/DMA Control Register Formats

15	14	...	4	3	2	1	0
0	0	...	0	MSK	PR2	PR1	PR0

INT0/INT1 Control Register Formats

15	14	...	7	6	5	4	3	2	1	0
0	0	...	0	SFNM	C	LTM	MSK	PR2	PR1	PR0

INT2/INT3 Control Register Formats

15	14	...	5	4	3	2	1	0
0	0	...	0	LTM	MSK	PR2	PR1	PR0

In-Service, Interrupt Request, and Mask Register Formats

15	14	...	10	9	8	7	6	5	4	3	2	1	0
0	0	...	0	0	0	I3	I2	I1	I0	D1	D0	0	TMR

Priority Mask Register Format

15	14	...	3	2	1	0
0	0	...	0	PRM2	PRM1	PRM0

Interrupt Status Register Format

15	14	...	7	6	5	4	3	2	1	0
DHLT	0	...	0	0	0	0	0	IRT2	IRT1	IRT0

EOI Register Format

15	14	13	...	5	4	3	2	1	0
SPEC NSPEC	0	0	...	0	S4	S3	S2	S1	S0

Poll Register Format

15	14	13	...	5	4	3	2	1	0
INT REQ	0	0	...	0	S4	S3	S2	S1	S0

MPA00151

Master Mode Features

Programmable priority

The user can program the interrupt sources into any of eight different priority levels. The programming is done by placing a 3-bit priority level (0 to 7) in the control register of each interrupt source.

End-of-interrupt command

The end-of-interrupt (EOI) command is used by the programmer to reset the in-service (IS) bit when an interrupt service routine is completed. The EOI command is issued by writing the proper pattern to the EOI register. There are two types of EOI commands, specific and nonspecific. The non-specific command does not specify which IS bit is reset.

Trigger mode

The four external interrupt pins can be programmed in either edge or level-trigger mode. The control register for each external source has a level-trigger mode (LTM) bit. All interrupt inputs are active high. In the edge-sense mode or the level-trigger mode, the interrupt request must remain active (high) until the interrupt request is acknowledged by the SAB 80188 CPU. In the edge-sense mode, if the level remains high after the interrupt is acknowledged, the input is disabled and no further requests will be generated. The input level must go low for at least one clock cycle to reenable the input.

Interrupt vectoring

The SAB 80188 interrupt controller will generate interrupt vectors for the integrated DMA channels and the integrated timers. In addition, the interrupt controller will generate interrupt vectors for the external interrupt lines if they are not configured in cascade or special fully nested mode.

Interrupt Controller Registers

In-service register

This register contains the in-service bit for each of the interrupt sources. The in-service bit is set to indicate that a source's service routine is in progress. When an in-service bit is set, the interrupt controller will not generate interrupts to the CPU when it receives interrupt requests from devices with a lower programmed priority level. The TMR bit is the in-service bit for all three timers; the D0 and D1 bits are the in-service bits for the two DMA channels; the I0 – I3 bits are the in-service bits for the external interrupt pins.

Interrupt request register

The internal interrupt sources have interrupt request bits inside the interrupt controller. A read from this register yields the status of these bits. The TMR bit is the logical OR of all timer interrupt requests. D0 and D1 are the interrupt request bits for the DMA channels.

Mask register

This is a 16-bit register that contains a mask bit for each interrupt source. A one in a bit position corresponding to a particular source serves to mask the source from generating interrupts. These mask bits are exactly the same bits which are used in the individual control registers.

Priority mask register

This register is used to mask all interrupts below particular interrupt priority levels. The code in the lower three bits of this register inhibits interrupts of priority lower (a higher priority number) than the code specified.

Interrupt status register

This register contains general interrupt controller status information. The bits in the status register have the following functions:

DHLT: DMA halt transfer; setting this bit halts all DMA transfers. It is automatically set whenever a non-maskable interrupt occurs, and it is reset when an IRET instruction is executed.

IRTx: These three bits represent the individual timer interrupt request bits. These bits are used to differentiate the timer interrupts, since the timer IR bit in the interrupt request register is the "OR" function of all timer interrupt requests.

Timer, DMA 0, 1 control registers

These registers are the control words for all the internal interrupt sources. The three bit positions PR0, PR1, and PR2 represent the programmable priority level of the interrupt source. The MSK bit inhibits interrupt requests from the interrupt source. The MSK bits in the individual control registers are exactly the same bits as are in the mask register; modifying them in the individual control registers will also modify them in the mask register, and vice versa.

INT0 to INT3 control registers

These registers are the control words for the four external input pins. In cascade mode or special fully nested mode, the control words for INT2 and INT3 are not used.

PR0-2: Priority programming information. Highest priority = 000, lowest priority = 111

LTM: Level-trigger mode bit. 1 = level-triggered; 0 = edge-triggered. Interrupt input levels are active high. In level-triggered mode, an interrupt is generated whenever the external line is high. In edge-triggered mode, an interrupt will be generated only when this level is preceded by an inactive-to-active transition on the line. In both cases, the level must remain active until the interrupt is acknowledged.

MSK: Mask bit, 1 = mask; 0 = nonmask.

C: Cascade mode bit, 1 = cascade; 0 = direct

SFNM: Special fully nested mode bit, 1 = SFNM

EOI register

The end-of-interrupt register is a command register which can only be written into. It initiates an EOI command when written to by the SAB 80188 CPU.

S_x: Encoded information that specifies an interrupt source vector type

NSPEC/SPEC: A bit that determines the type of EOI command. Nonspecific = 1, specific = 0.

Poll and poll status registers

These registers contain polling information. They can only be read. Reading the poll register constitutes a software poll. This will set the IS bit of the highest priority pending interrupt.

S_x: Encoded information that indicates the vector type of the highest priority interrupting source. Valid only when INTREQ = 1.

INTREQ: This bit determines if an interrupt request is present. Interrupt request = 1; no interrupt request = 0.

iRMX 86 Compatibility Mode

This mode allows iRMX 86-SAB 80188 compatibility. The interrupt model of iRMX 86 requires one master and multiple slave SAB 8259As in cascaded mode. When iRMX mode is used, the internal SAB 80188 interrupt controller will be used as a slave controller to an external master interrupt controller.
Upon reset, the SAB 80188 interrupt controller will be in the non-iRMX 86 mode of operation. To set the controller in the iRMX 86 mode, bit 14 of the relocation register should be set.

The iRMX 86 operating system requires peripherals to be assigned fixed priority levels. This is incompatible with the normal operation of the SAB 80188 interrupt controller. Therefore, the initialization software must program the proper priority levels for each source.

Required iRMX Internal Source Priority Levels

Priority Level	Interrupt Source
0	Timer 0
1	(reserved)
2	DMA 0
3	DMA 1
4	Timer 1
5	Timer 2

iRMX 86 Mode External Interface

In the iRMX 86 configuration of the SAB 80188 the INT0 input is used as the SAB 80188 CPU interrupt input. INT3 functions as an output to send the SAB 80188 slave-interrupt-request to one of the 8 master PIC inputs. Correct master-slave interface requires decoding of the slave addresses (CAS0–2). Slave SAB 8259As do this internally. Because of pin limitations, the SAB 80188 slave address will have to be decoded externally. $\overline{\text{INT1}}$ is used as a slaveselect input. $\overline{\text{INT2}}$ is used as an acknowledge output, suitable to drive the $\overline{\text{INTA}}$ input of an SAB 8259A.

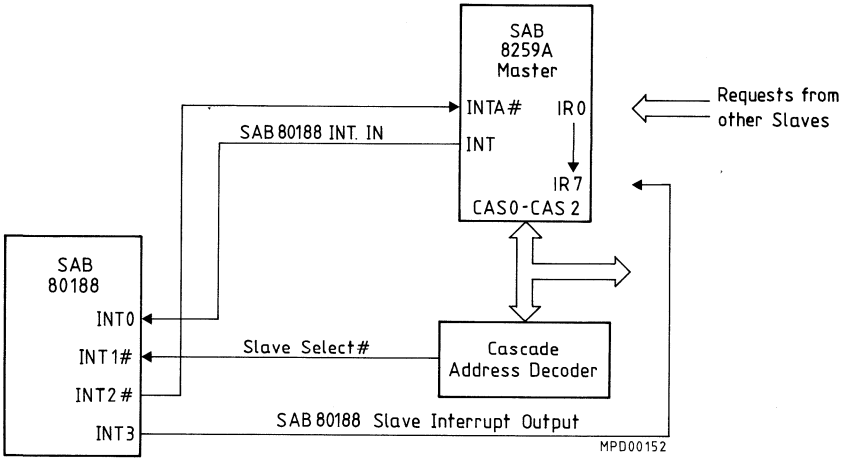
Vector Generation in the iRMX 86 Mode

Vector generation in iRMX mode is exactly like that of an SAB 8259A slave. The interrupt controller generates an 8-bit vector which the CPU multiplies by four and uses as an address into a vector table. The significant five bits of the vector are user-programmable while the lower three bits are generated by the priority logic.

Specific End-of-Interrupt

In iRMX mode the specific EOI command operates to reset an in-service bit of a specific priority. The user supplies a 3-bit priority-level value that points to an in-service bit to be reset.

iRMX 86 Interrupt Controller Interconnection



Interrupt Controller Registers (iRMX 86 mode)

	Offset
Level 5 Control Register (Timer 2)	3AH
Level 4 Control Register (Timer1)	38H
Level 3 Control Register (DMA1)	36H
Level 2 Control Register (DMA 0)	34H
Level 0 Control Register (Timer0)	32H
Interrupt Status Register	30H
Interrupt-Request Register	2EH
In-Service Register	2CH
Priority-Level Mask Register	2AH
Mask Register	28H
Specific EOI Register	22H
Interrupt Vector Register	20H

Specific EOI Register Format

15	14	13	· · · · · · · ·	8	7	6	5	4	3	2	1	0
0	0	0		0	0	0	0	0	0	L2	L1	L0

In-Service, Interrupt Request, and Mask Register Format

15	14	13	· · · · · · · ·	8	7	6	5	4	3	2	1	0
0	0	0		0	0	0	TMR 2	TMR 1	D1	D0	0	TMR 0

Control Word Format

15	14	13	· · · · · · · ·	8	7	6	5	4	3	2	1	0
0	0	0		0	0	0	0	0	MSK	PR2	PR1	PR0

Interrupt Vector Register Format

15	14	13	· · · · · · · ·	8	7	6	5	4	3	2	1	0
0	0	0		0	t4	t3	t2	t1	t0	0	0	0

Priority Level Mask Register

15	14	13	· · · · · · · ·	8	7	6	5	4	3	2	1	0
0	0	0		0	0	0	0	0	0	m2	m1	m0

MPA00153

Interrupt Controller Registers in the iRMX 86 Mode

End-of-interrupt register

The end-of-interrupt register is a command register which can only be written to. It initiates an EOI command when written to by the SAB 80188 CPU.

Lx: Encoded value indicating the priority of the IS bit to be reset.

In-service register

This register contains the in-service bit for each of the internal interrupt sources. Bit positions 2 and 3 correspond to the DMA channels; positions 0, 4, and 5 correspond to the integral timers.

Interrupt request register

This register indicates which internal peripherals have interrupt requests pending. The interrupt request bits are set when a request arrives from an internal source, and are reset when the processor acknowledges the request.

Mask register

This register contains a mask bit for each interrupt source. If the bit in this register corresponding to a particular interrupt source is set, any interrupts from that source will be masked.

Control registers

These registers are the control words for all the internal interrupt sources.

PRx: 3-bit encoded field indicating a priority level for the source; note that each source must be programmed at specified levels.

MSK: Mask bit for the priority level indicated by PRx bits.

Interrupt vector register

This register provides the upper five bits of the interrupt vector address. The interrupt controller itself provides the lower three bits of the interrupt vector as determined by the priority level of the interrupt request.

The format of the bits in this register is:

tx: 5-bit field indicating the upper five bits of the vector address.

Priority-level mask register

This register indicates the lowest priority-level interrupt which will be serviced.

The encoding of the bits in this register is:

mx: 3-bit encoded field indication priority-level value. All levels of lower priority will be masked.

Interrupt status register

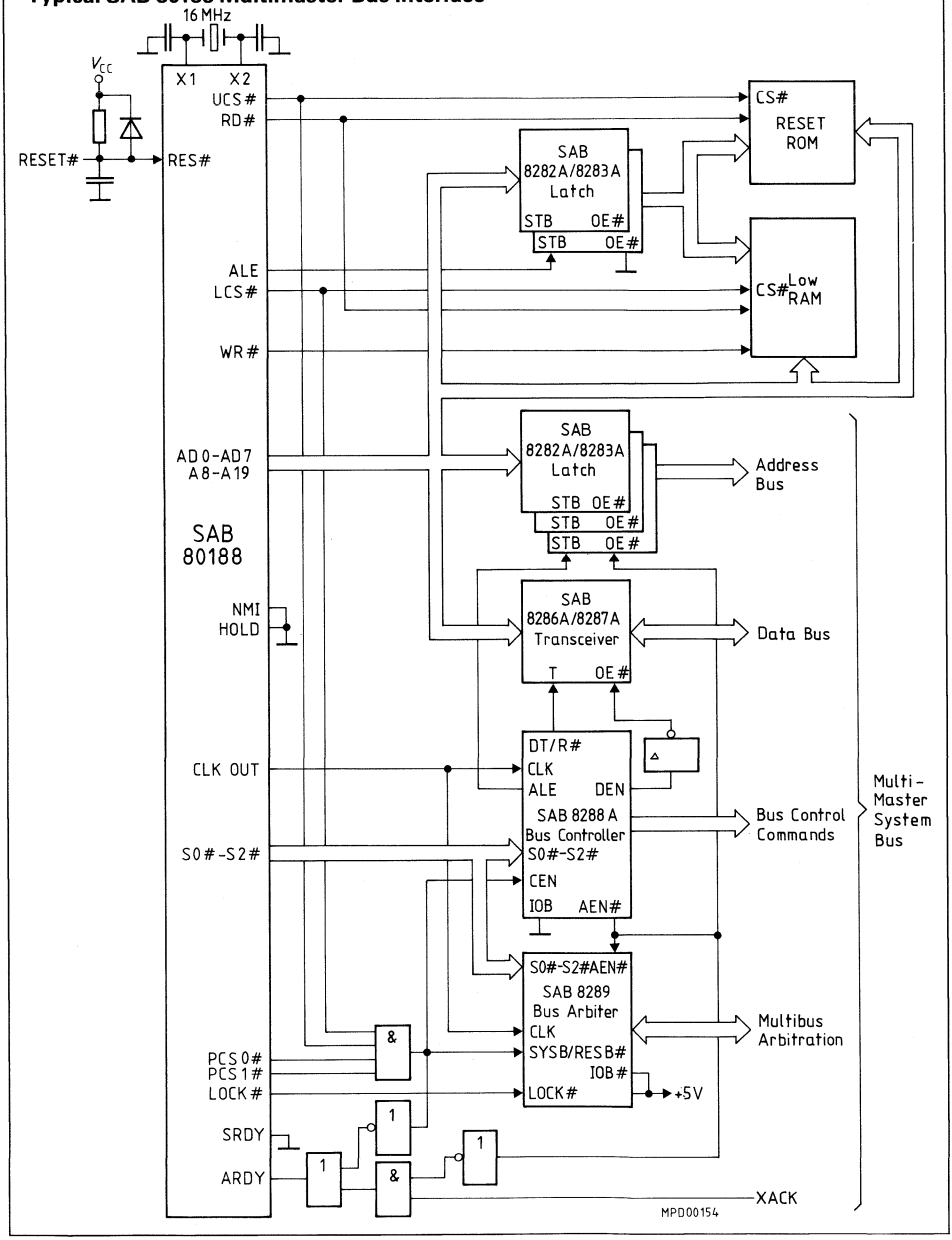
This register is defined exactly as in non-iRMX mode.

Interrupt Controller and Reset

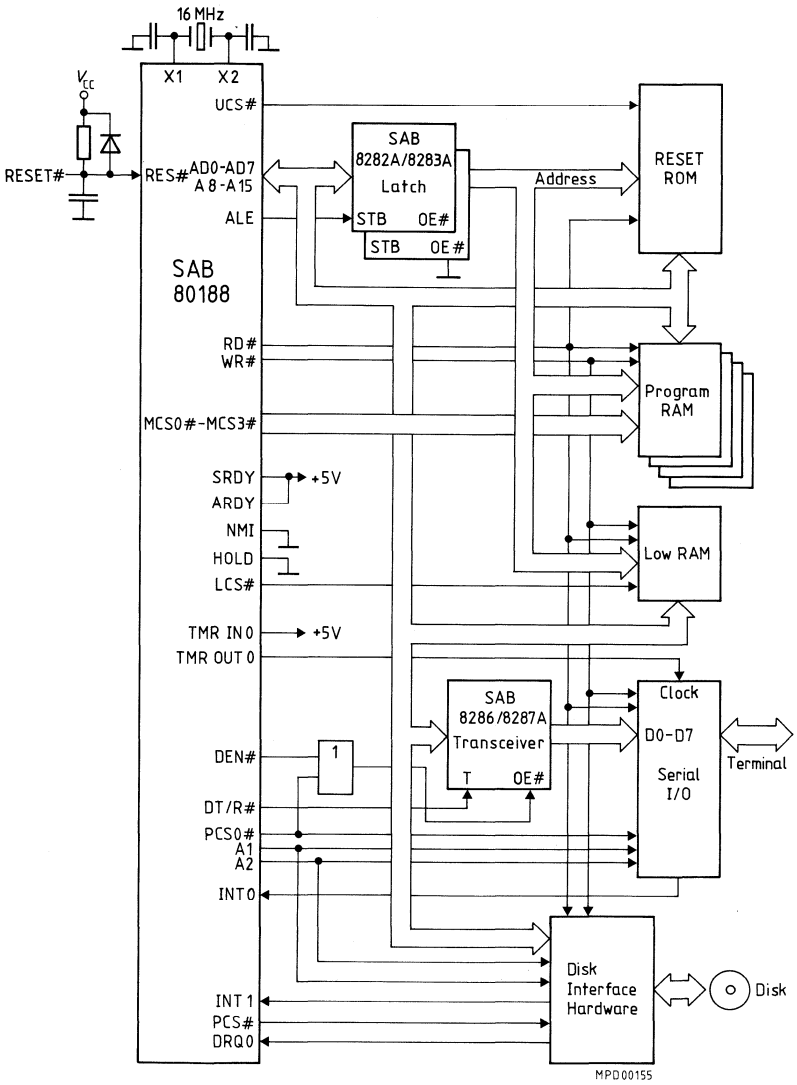
Upon reset, the interrupt controller will perform the following actions:

- All SFNM bits reset to 0, implying fully nested mode.
- All PR bits in the various control registers set to 1. This places all sources at lowest priority (level 111).
- All LTM bits reset to 0, resulting in edge-sense mode.
- All interrupt service bits reset to 0.
- All interrupt request bits reset to 0.
- All MSK (interrupt mask) bits set to 1 (mask).
- All C (cascade) bits reset to 0 (non-cascade).
- All PRM (Priority mask) bits set to 1, implying no levels masked.
- Initialized to non-iRMX 86 mode.

Typical SAB 80188 Multimaster Bus Interface



Typical SAB 80188 System Configuration



Instruction Timings

The following instruction timings represent the minimum execution time in clock cycles for each instruction. The timings given are based on the following assumptions:

- The op code, along with any data or displacement required for execution of a particular instruction, has been prefetched and resides in the queue at the time it is needed.
- No wait states or bus HOLDs occur.
- All word data is located on even-address boundaries.

All jumps and calls include the time required to fetch the op code of the next instruction at the destination address.

All instructions which involve memory reference can require one (and in some cases, two) additional clocks above the minimum timings shown. This is due to the asynchronous nature of the handshake between the BIU and the execution unit.

Notes:

The effective address (EA) of the memory operand is computed according to the mod and r/m fields:

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low

If r/m = 000 then EA = (BX) + (SI) + DISP	if r/m = 100 then EA = (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP	if r/m = 101 then EA = (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP	if r/m = 110 then EA = (BP) + DISP*
if r/m = 011 then EA = (BP) + (DI) + DISP	if r/m = 111 then EA = (BX) + DISP

DISP follows 2nd byte of instruction (before data if required)

REG is assigned according to the following table:

16-bit (w = 1)	8-bit (w = 0)	16-bit (w = 1)	8-bit (w = 0)
000 AX	000 AL	100 SP	100 AH
001 CX	001 CL	101 BP	101 CH
010 DX	010 DL	110 SI	110 DH
011 BX	011 BL	111 DI	111 BH

The physical addresses of all operands addressed by the BP register are computed using the SS segment register. The physical addresses of the destination operands of the string primitive operations (those addressed by the DI register) are computed using the ES segment, which may not be overridden.

* except if mod = 00 and r/m = 110 then EA = disp-high: disp-low

Note: EA calculation time is 4 clock cycles for all modes, and is included in the execution times given whenever appropriate.

Segment override prefix

0	0	1	reg	1	1	0

reg is assigned according to the following:

reg	Segment register
00	ES
01	CS
10	SS
11	DS

Instruction Set Summary

Function	Format	Clock Cycles	Comments
Data Transfer			
MOV = Move:			
Register to register/memory	1000100w | mod reg r/m	2/12	
Register/memory to register	1000101w | mod reg r/m	2/9	
Immediate to register/memory	1100011w | mod 000 r/m | data | data if w = 1	12–13	8/16-bit
Immediate to register	1011w reg | data | data if w = 1	3–4	8/16-bit
Memory to accumulator	1010000w | addr-low | addr-high	9	
Accumulator to memory	1010001w | addr-low | addr-high	8	
Register/memory to segment register	10001110 | mod 0 reg r/m	2/9	
Segment register to register/memory	10001100 | mod 0 reg r/m	2/11	
PUSH = Push:			
Memory	11111111 | mod 110 r/m	16	
Register	01010 reg	10	
Segment register	000reg110	9	
Immediate	011010s0 | data | data if s = 0	10	
PUSHA = Push All	01100000	36	
POP = Pop:			
Memory	10001111 | mod 000 r/m	20	
Register	01011 reg	10	
Segment register	000reg111 (reg ≠ 01)	8	

Shaded areas indicate instructions not available on SAB 8086/8088 processors.

Function	Format			Clock Cycles	Comments
Data Transfer (cont'd)					
POPA = Pop All	01100001			51	
XCHG = **Exchange:**					
Register/memory with register	1000011w	mod reg r/m		4/17	
Register with accumulator	10010 reg			3	
IN = **Input from:**					
Fixed port	1110010w	port		10	
Variable port	1110110w			8	
OUT = **Output to:**					
Fixed port	1110011w	port		9	
Variable port	1110111w			7	
XLAT = Translate byte to AL	11010111			11	
LEA = Load EA to register	10001101	mod reg r/m		6	
LDS = Load pointer to DS	11000101	mod reg r/m	(mod ≠ 11)	18	
LES = Load pointer to ES	11000100	mod reg r/m	(mod ≠ 11)	18	
LAHF = Load AH with flags	10011111			2	
SAHF = Store AH into flags	10011110			3	
PUSHF = Push flags	10011100			9	
POPF = Pop flags	10011101			8	
SEGMENT = **Segment override:**					
CS	00101110			2	
SS	00110110			2	
DS	00111110			2	
ES	00100110			2	

Shaded areas indicate instructions not available on SAB 8086/8088 processors.

Function	Format	Clock Cycles	Comments
Arithmetic			
ADD = Add:			
Reg./memory with register to either	0 0 0 0 0 d w | mod reg. r/m	3/10	
Immediate to register/memory	1 0 0 0 0 0 s w | mod 0 0 0 r/m | data | data if s w = 01	4/16	
Immediate to accumulator	0 0 0 0 0 0 1 0 w | data | data if w = 1	3/4	8/16-bit
ADC = Add with carry:			
Reg./memory with register to either	0 0 0 1 0 0 d w | mod reg r/m	3/10	
Immediate to register/memory	1 0 0 0 0 0 s w | mod 0 1 0 r/m | data | data if s w = 01	4/16	
Immediate to accumulator	0 0 0 1 0 1 0 w | data | data if w = 1	3/4	8/16-bit
INC = Increment			
Register/memory	1 1 1 1 1 1 1 w | mod 0 0 0 r/m	3/15	
Register	0 1 0 0 0 reg	3	
SUB = Subtract			
Reg./memory and register to either	0 0 1 0 1 0 d w | mod reg r/m	3/10	
Immediate from register/memory	1 0 0 0 0 0 s w | mod 1 0 1 r/m | data | data if s w = 01	4/16	
Immediate from accumulator	0 0 1 0 1 1 0 w | data | data if w = 1	3/4	8/16-bit
SBB = Subtract with borrow:			
Reg./memory and register to either	0 0 0 1 1 0 d w | mod reg r/m	3/10	
Immediate from register/memory	1 0 0 0 0 0 s w | mod 0 1 1 r/m | data | data if s w = 01	4/16	
Immediate from accumulator	0 0 0 1 1 1 0 w | data | data if w = 1	3/4	8/16-bit

Function	Format	Clock Cycles	Comments
Arithmetic (cont'd):			
DEC = **Decrement:**			
Register/memory	1 1 1 1 1 1 1 w | mod 0 0 1 r/m	3/15	
Register	0 1 0 0 1 reg	3	
CMP = **Compare:**			
Register/memory with register	0 0 1 1 1 0 1 w | mod reg r/m	3/10	
Register with register/memory	0 0 1 1 1 0 0 w | mod reg r/m	3/10	
Immediate with register/memory	1 0 0 0 0 0 s w | mod 1 1 1 r/m | data | data if s w=01	3/10	
Immediate with accumulator	0 0 1 1 1 1 0 w | data | data if w = 1	3/4	8/16-bit
NEG = Change sign	1 1 1 1 0 1 1 w | mod 0 1 1 r/m	3	
AAA = ASCII adjust for add	0 0 1 1 0 1 1 1	8	
DAA = Decimal adjust for add	0 0 1 0 0 1 1 1	4	
AAS = ASCII adjust for subtract	0 0 1 1 1 1 1 1	7	
DAS = Decimal adjust for subtract	0 0 1 0 1 1 1 1	4	
MUL = Multiply (unsigned):	1 1 1 1 0 1 1 w | mod 1 0 0 r/m		
register–byte		26–28	
register–word		35–37	
memory–byte		32–34	
memory–word		41–43	
IMUL = Integer multiply (signed):	1 1 1 1 0 1 1 w | mod 1 0 1 r/m		
register–byte		25–28	
register–word		34–37	
memory–byte		31–34	
memory–word		40–43	

Function	Format	Clock Cycles	Comments
Arithmetic (cont'd):			
IMUL = Integer immediate multiply (signed)	011010s1 | mod reg r/m | data | data if s = 0	22–25/29–32	
DIV = Divide (unsigned):	1111011w | mod 110 r/m		
register–byte		29	
register–word		38	
memory–byte		35	
memory–word		44	
IDIV = Integer divide (signed):	1111011w | mod 111 r/m		
register–byte		44–52	
register–word		53–61	
memory–byte		50–58	
memory–word		59–67	
AAM = ASCII adjust for multiply	11010100 | 00001010	19	
AAD = ASCII adjust for divide	11010101 | 00001010	15	
CBW = Convert byte to word	10011000	2	
CWD = Convert word to double word	10011001	4	
Logic **Shift/rotate instructions:**			
Register/memory by 1	1101000w | mod TTT r/m	2/15	
Register/memory by CL	1101001w | mod TTT r/m	5+n/17+n	
Register/memory by count	1100000w | mod TTT r/m | count	5+n/17+n	
	TTT Instruction 000 ROL 001 ROR 010 RCL 011 RCR 100 SHL/SAL 101 SHR 111 SAR		

Shaded areas indicate instructions not available on SAB 8086/8088 processors.

Function	Format				Clock Cycles	Comments
Logic (cont'd):						
AND = **And:**						
Reg./memory and register to either	0 0 1 0 0 0 d w	mod reg r/m			3/10	
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 0 0 r/m	data	data if w = 1	4/16	
Immediate to accumulator	0 0 1 0 0 1 0 w	data	data f w = 1		3/4	8/16-bit
TEST = **And function to flags, no result:**						
Register/memory and register	1 0 0 0 0 1 0 w	mod reg r/m			3/10	
Immediate data and register/memory	1 1 1 1 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1	4/10	
Immediate data and accumulator	1 0 1 0 1 0 0 w	data	data if w = 1		3/4	8/16-bit
OR = **Or:**						
Reg./memory and register to either	0 0 0 0 1 0 d w	mod reg r/m			3/10	
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 0 0 1 r/m	data	data if w = 1	4/16	
Immediate to accumulator	0 0 0 0 1 1 0 w	data	data if w = 1		3/4	8/16-bit
XOR = **Exclusive Or:**						
Reg./memory and register to either	0 0 1 1 0 0 d w	mod reg r/m			3/10	
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 1 0 r/m	data	data if w = 1	4/16	
Immediate to accumulator	0 0 1 1 0 1 0 w	data	data if w = 1		3/4	8/16-bit
NOT = Invert register/memory	1 1 1 1 0 1 1 w	mod 0 1 0 r/m			3	

Function	Format	Clock Cycles	Comments
String Manipulation:			
MOVS = Move byte/word	1010010w	14	
CMPS = Compare byte/word	1010011w	22	
SCAS = Scan byte/word	1010111w	15	
LODS = Load byte/word to AL/AX	1010110w	12	
STOS = Store byte/word from AL/AX	1010101w	10	
INS = Input byte/word from DX port	0110110w	14	
OUTS = Output byte/word to DX port	0110111w	14	
Repeated by count in CX			
MOVS = Move string	11110010 1010010w	8 + 8n	
CMPS = Compare string	1111001z 1010011w	5 + 22n	
SCAS = Scan string	1111001z 1010111w	5 + 15n	
LODS = Load string	11110010 1010110w	6 + 11n	
STOS = Store string	11110010 1010101w	6 + 9n	
INS = Input string	11110010 0110110w	8 + 8n	
OUTS = Output string	11110010 0110111w	8 + 8n	

Shaded areas indicate instructions not available on SAB 8086/8088 processors.

Function	Format	Clock Cycles	Comments
Control Transfer:			
CALL = Call:			
Direct within segment	11101000 | disp-low | disp-high	15	
Register/memory indirect within segment	11111111 | mod 010 r/m	13/19	
Direct intersegment	10011010 | segment offset segment selector	23	
Indirect intersegment	11111111 | mod 011 r/m (mod ≠ 11)	38	

Function	Format	Clock Cycles	Comments
Control Transfer (cont'd):			
JMP = Unconditional jump:			
Short/long	11101011 | disp-low	14	
Direct within segment	11101001 | disp-low | disp-high	14	
Register/memory indirect within segment	11111111 | mod 100 r/m	11/17	
Direct intersegment	11101010 | segment offset segment selector	14	
Indirect intersegment	11111111 | mod 101 r/m (mod ≠ 11)	26	
RET = Return from CALL:			
Within segment	11000011	16	
Within seg. adding immediate to SP	11000010 | data-low | data-high	18	
Intersegment	11001011	22	
Intersegment adding immediate to SP	11001010 | data-low | data-high	25	

Function	Format		Clock Cycles	Comments
Control Transfer (cont'd):				
JE/JZ = Jump on equal/zero	01110100	disp	4/13	JMP not taken/JMP taken
JL/JNGE = Jump on less/not greater or equal	01111100	disp	4/13	
JLE/JNG = Jump on less or equal/not greater	01111110	disp	4/13	
JB/JNAE = Jump on below/not above or equal	01110010	disp	4/13	
JBE/JNA = Jump on below or equal/not above	01110110	disp	4/13	
JP/JPE = Jump on parity/parity even	01111010	disp	4/13	
JO = Jump on overflow	01110000	disp	4/13	
JS = Jump on sign	01111000	disp	4/13	
JNE/JNZ = Jump on not equal/not zero	01110101	disp	4/13	
JNL/JGE = Jump on not less/greater or equal	01111101	disp	4/13	
JNLE/JG = Jump on not less or equal/greater	01111111	disp	4/13	
JNB/JAE = Jump on not below/above or equal	01110011	disp	4/13	
JNBE/JA = Jump on not below or equal/above	01110111	disp	4/13	
JNP/JPO = Jump on not parity/parity odd	01111011	disp	4/13	
JNO = Jump on not overflow	01110001	disp	4/13	
JNS = Jump on not sign	01111001	disp	4/13	
JCXZ = Jump on CX zero	11100011	disp	5/15	
LOOP = Loop CX times	11100010	disp	6/16	LOOP not taken/LOOP taken
LOOPZ/LOOPE = Loop while zero/equal	11100001	disp	6/16	
LOOPNZ/LOOPNE = Loop while not zero/equal	11100000	disp	6/16	

Function	Format	Clock Cycles	Comments
Control Transfer (cont'd):			
ENTER = Enter procedure	11001000 | data-low | data-high | L		
L = 0		15	
L = 1		25	
L > 1		22 + 16(n − 1)	
LEAVE = Leave procedure	11001001	8	
INT = **Interrupt**:			
Type specified	11001101 | type	47	
Type 3	11001100	45	
INTO = Interrupt on overflow	11001110	48/4	if INT taken/ if INT not taken
IRET = Interrupt return	11001111	28	
BOUND = Detect value out of range	01100010 | mod reg r/m	33 - 35	

Shaded areas indicate instructions not available on SAB 8086/8088 processors.

Function	Format	Clock Cycles	Comments
Processor Control			
CLC = Clear carry	11111000	2	
CMC = Complement carry	11110101	2	
STC = Set carry	11111001	2	
CLD = Clear direction	11111100	2	
STD = Set direction	11111101	2	
CLI = Clear interrupt	11111010	2	
STI = Set interrupt	11111011	2	
HLT = Halt	11110100	2	
WAIT = Wait	10011011	6	if $\overline{\text{TEST}}$ = 0
LOCK = Bus lock prefix	11110000	2	
ESC = Processor extension escape	11011TTT mod LLL r/m (TTT LLL are op codes to processor extension)	6	

Absolute Maximum Ratings

Ambient temperature under bias . . . . . 0 to 70°C
Storage temperature . . . . . −65 to +150°C
Voltage on any pin with respect to ground . . . . . −0.5 to +7 V
Power dissipation . . . . . 3 W

Note: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

T_A = 0 to 70°C; V_{CC} = 5V ±10%

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
Input low voltage	V_{IL}	−0.5	+0.8	V	–
Input high voltage (all except X1 and $\overline{RES}$)	V_{IH}	2.0	V_{CC} +0.5	V	–
Input high voltage ($\overline{RES}$)	V_{IH1}	3.0	V_{CC} +0.5	V	–
Output low voltage	V_{OL}	–	0.45	V	I_a = 2.5 mA for $\overline{S0}$-$\overline{S2}$ I_a = 2.0 mA for all other outputs
Output high voltage	V_{OH}	2.4	–	V	I_{oa} = −400 µA
Power supply current	I_{CC}	–	550	mA	T_A = 0°C; All outputs disconnected
		–	450	mA	T_A = 70°C; All outputs disconnected
Input leakage current	I_{LI}	–	±10	µA	$0V < V_{IN} < V_{CC}$
Output leakage current	I_{LO}	–	±10	µA	$0.45V < V_{OUT} < V_{CC}$
Clock output low	V_{CLO}	–	0.6	V	I_a = 4.0 mA
Clock output high	V_{CHO}	4.0	–	V	I_{oa} = −200 µA
Clock input low voltage	V_{CLI}	−0.5	0.6	V	–
Clock input high voltage	V_{CHI}	3.9	V_{CC} +1.0	V	–
Input capacitance	C_{IN}	–	10	pF	–
I/O capacitance	C_{IO}	–	20	pF	–

AC Characteristics

T_A = 0 to 70°C, V_{CC} = 5V ±10%

Timing requirements: all timings measured at 1.5V unless otherwise specified

Parameter	Symbol	Limit values				Unit	Test condition
		SAB 80188		SAB 80188-1			
		min.	max.	min.	max.		
Data in setup (A/D)	t_{DVCL}	20	–	15	–	ns	–
Data in hold (A/D)	t_{CLDX}	10	–	8	–	ns	–
Asynchronous ready (ARDY) active setup time [1]	t_{ARYHCH}	20	–	15	–	ns	–
ARDY inactive setup time	t_{ARYLCL}	35	–	25	–	ns	–
ARDY hold time	t_{CHARYX}	15	–	15	–	ns	–
Asynchronous ready inactive hold time	t_{ARYCHL}	15	–	15	–	ns	–
Synchronous ready (SRDY) transition setup time	t_{SRYCL}	20	–	20	–	ns	–
SRDY transition hold time	t_{CLSRY}	15	–	15	–	ns	–
HOLD setup time [1]	t_{HVCL}	25	–	20	–	ns	–
INTR, NMI, $\overline{\text{TEST}}$, TIMERIN, setup time [1]	t_{INVCH}	25	–	25	–	ns	–
DRQ0, DRQ1 setup time [1]	t_{INVCL}	25	–	20	–	ns	–

[1] To guarantee recognition at next clock.

Master Interfaces Timing Responses

Parameter	Symbol	Limit values				Unit	Test condition
		SAB 80188		SAB 80188-1			
		min.	max.	min.	max.		
Address valid delay	t_{CLAV}	5	55	5	44	ns	C_L = 20 to 200 pF all outp.
Address hold time	t_{CLAX}	10	–	10	–	ns	–
Address float delay	t_{CLAZ}	t_{CLAX}	35	t_{CLAX}	30	ns	–
Command lines float delay	t_{CHCZ}	–	45	–	40	ns	–
Command lines valid delay (after float)	t_{CHCV}	–	55	–	45	ns	–
ALE width	t_{LHLL}	t_{CLCL}-35	–	t_{CLCL}-30	–	ns	–
ALE active delay	t_{CHLH}	–	35	–	30	ns	–
ALE inactive delay	t_{CHLL}	–	35	–	30	ns	–
Address hold to ALE inactive	t_{LLAX}	t_{CHCL}-25	–	t_{CHCL}-20	–	ns	–
Data valid delay	t_{CLDV}	10	44	10	40	ns	–
Data hold time	t_{CLDOX}	10	–	10	–	ns	–
Data hold after $\overline{WR}$	t_{WHDX}	t_{CLCL}-40	–	t_{CLCL}-34	–	ns	–
Control active delay 1	t_{CVCTV}	10	70	5	40	ns	–
Control active delay 2	t_{CHCTV}	10	55	10	44	ns	–
Control inactive delay	t_{CVDEX}	5	55	5	44	ns	–
$\overline{DEN}$ inactive delay (non-write cycle)	t_{CVDEX}	10	70	10	56	ns	–
Address float to $\overline{RD}$ active	t_{AZRL}	0	–	0	–	ns	–
$\overline{RD}$ active delay	t_{CLRL}	10	70	10	56	ns	–
$\overline{RD}$ inactive delay	t_{CLRH}	10	55	10	44	ns	–
$\overline{RD}$ inactive to address active	t_{RHAV}	t_{CLCL}-40	–	t_{CLCL}-40	–	ns	–
HLDA valid delay	t_{CLHAV}	5	50	5	40	ns	–
$\overline{RD}$ width	t_{RLRH}	$2t_{CLCL}$-50	–	$2t_{CLCL}$-46	–	ns	–
$\overline{WR}$ width	t_{WLWH}	$2t_{CLCL}$-40	–	$2t_{CLCL}$-34	–	ns	–
Address valid to ALE low	t_{AVAL}	t_{CLCH}-25	–	t_{CLCH}-19	–	ns	–
Status active delay	t_{CHSV}	10	55	10	45	ns	–
Status inactive delay	t_{CLSH}	10	65	10	50	ns	–
Timer output delay	t_{CLTMV}	–	60	–	48	ns	100 pF max.
Reset delay	t_{CLRO}	–	60	–	48	ns	–
Queue status delay	t_{CHQSV}	–	35	–	28	ns	–
Status hold time	t_{CHDX}	10	–	10	–	ns	–
Address valid to clock high	t_{AVCH}	10	–	10	–	ns	–
$\overline{LOCK}$ valid/invalid delay	t_{CLLV}	5	65	5	60	ns	–

Chip Select Timing Responses

Parameter	Symbol	Limit values				Unit	Test condition
		SAB 80188		SAB 80188-1			
		min.	max.	min.	max.		
Chip select active delay	t_{CLCSV}	–	66	–	45	ns	–
Chip select hold from command inactive	t_{CXCSX}	35	–	35	–	ns	–
Chip select inactive delay	t_{CHCSX}	5	35	5	32	ns	–

CLKIN Requirements

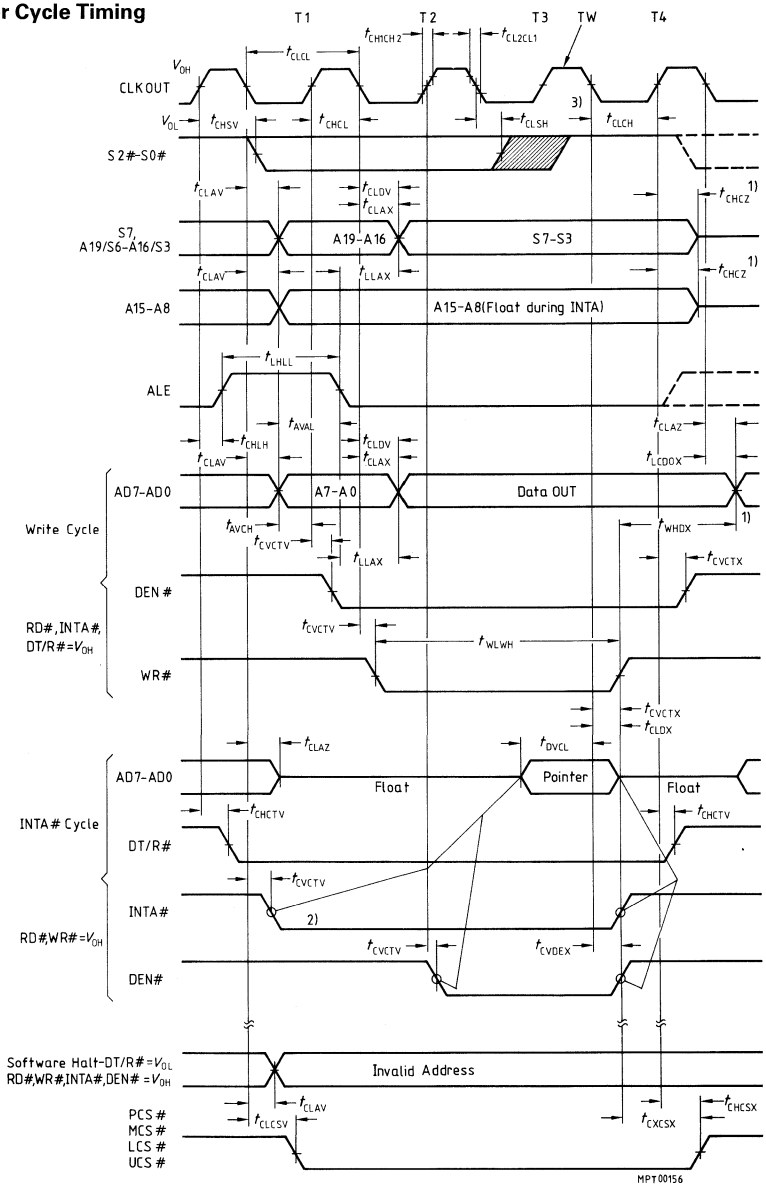
Parameter	Symbol	Limit values				Unit	Test condition
		SAB 80188		SAB 80188-1			
		min.	max.	min.	max.		
CLKIN period	t_{CKIN}	62.5	250	50	250	ns	–
CLKIN fall time	t_{CKHL}	–	10	–	10	ns	3.5 to 1.0 V
CLKIN rise time	t_{CKLH}	–	10	–	10	ns	1.0 to 3.5 V
CLKIN low time	t_{CLCK}	25	–	20	–	ns	1.5 V
CLKIN high time	t_{CHCK}	25	–	20	–	ns	1.5 V

CLKOUT Timing (200 pF load)

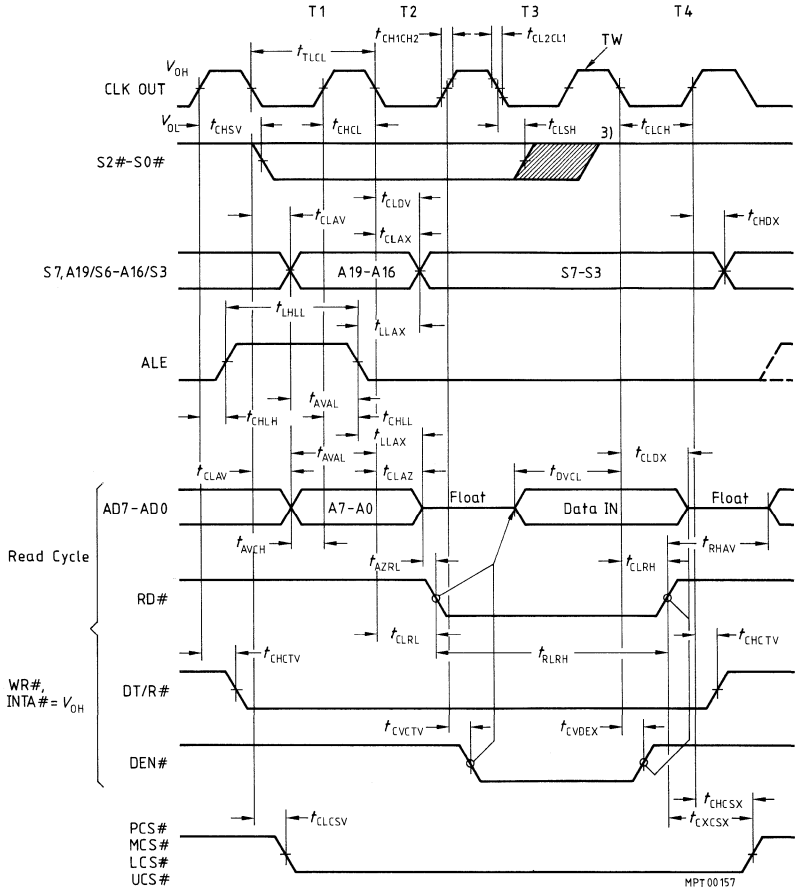
Parameter	Symbol	Limit values				Unit	Test condition
		SAB 80188		SAB 80188-1			
		min.	max.	min.	max.		
CLKIN to CLKOUT skew	t_{CICO}	–	50	–	25	ns	–
CLKOUT period	t_{CLCL}	125	500	100	500	ns	–
CLKOUT low time	t_{CLCH}	$1/2t_{CLCL}$-7.5	–	$1/2t_{CLCL}$-6.0	–	ns	1.5 V
CLKOUT high time	t_{CHCL}	$1/2t_{CLCL}$-7.5	–	$1/2t_{CLCL}$-6.0	–	ns	1.5 V
CLKOUT rise time	t_{CH1CH2}	–	15	–	12	ns	1.0 to 3.5 V
CLKOUT fall time	t_{CL2CL1}	–	15	–	12	ns	3.5 to 1.0 V

Waveforms

Major Cycle Timing



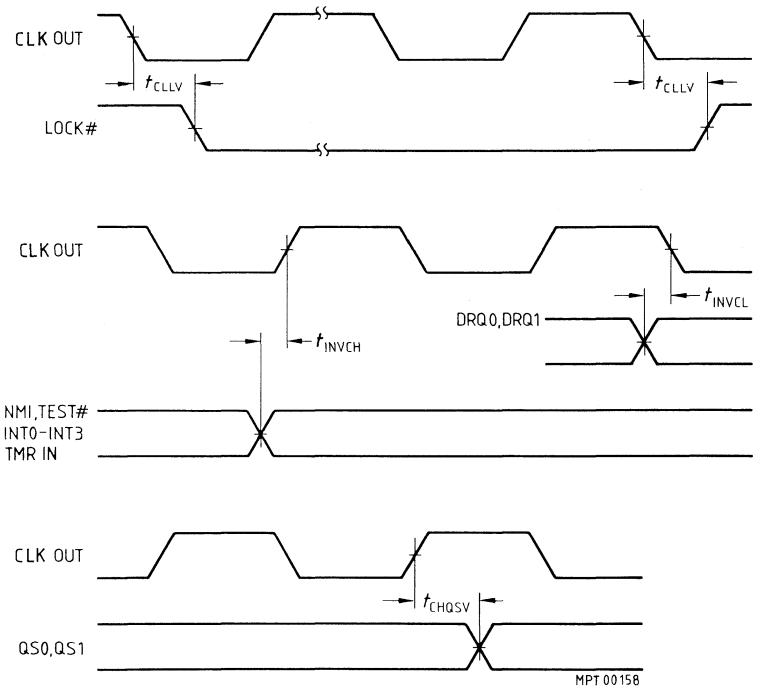
Major Cycle Timing (cont'd)



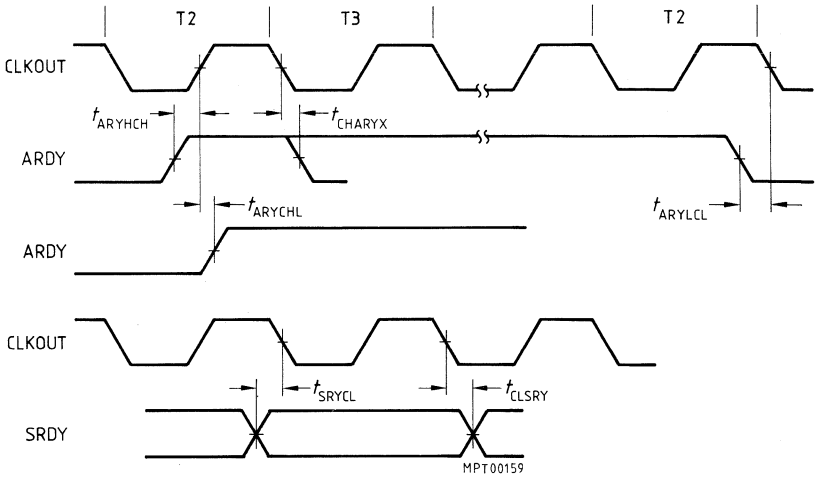
Notes

1. Following a write cycle, the local bus is tristated by the SAB 80188 only when the SAB 80188 enters a "hold acknowledge" state.
2. $\overline{\text{INTA}}$ occurs one clock later in iRMX mode.
3. Status inactive just prior to T4.

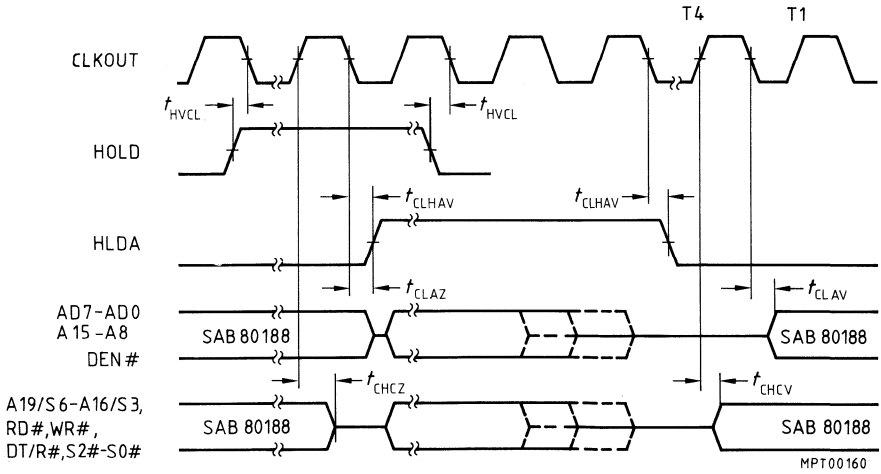
CLKOUT Timing Relationships



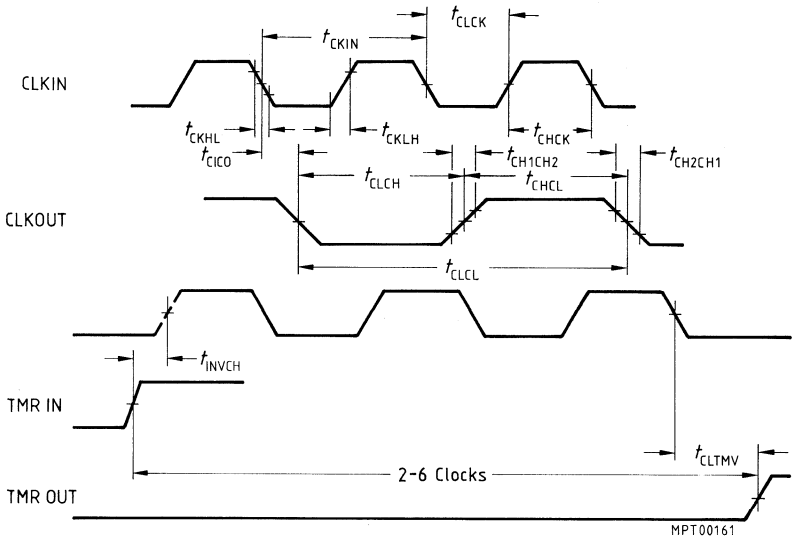
Ready Timing



HOLD-HLDA Timing



Timer Timing



SIEMENS

SAB 80286

High-Performance Microprocessor with Memory Management and Protection for Clock Rates up to 8 MHz, 10 MHz or 12.5 MHz

Preliminary

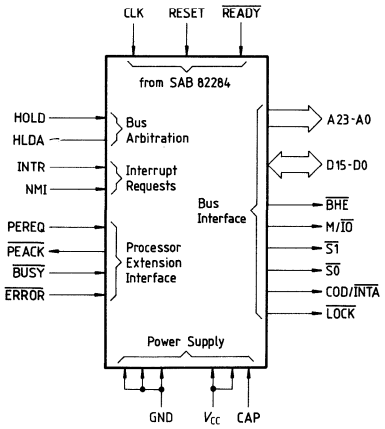
- High-performance processor (up to six times the SAB 8086)
- Large address space:
  - 16 megabytes physical
  - 1 gigabyte virtual per task
- High bandwidth bus interface (10 megabyte/s at 20 MHz system clock)
- Full hardware and software support
- Two SAB 8086 upward-compatible operating modes:
  - SAB 8086 real address mode
  - protected virtual address mode
- Integrated memory management, four-level memory protection and support for virtual memory and multitasking operating systems
- Plastic Package: PL-CC-68

The SAB 80286 is an advanced, high-performance microprocessor with specially optimized capabilities for multiple user and multitasking systems. The SAB 80286 has built-in memory protection that supports operating system and task isolation as well as program and data privacy within tasks. An 8 MHz SAB 80286 provides up to six times greater throughput than the standard 5 MHz SAB 8086. The SAB 80286 includes memory management capabilities that map up to 2^{30} (one gigabyte) of virtual address space per task into 2^{24} bytes (16 megabytes) of physical memory.
The SAB 80286 is upward-compatible with SAB 8086/8088 software. Using SAB 8086 real address mode, the SAB 80286 is object-code compatible with existing SAB 8086/8088 software. In protected virtual address mode, the SAB 80286 is source code compatible with SAB 8086/8088 software and may require upgrading to use virtual addresses supported by SAB 80286's integrated memory management and protection mechanism. Both modes operate at full SAB 80286 performance and execute a superset of the SAB 8086/8088 instructions.
The SAB 80286 provides special operations to support the efficient implementation and execution of operating systems. For example, one instruction can end execution of one task, save its state, switch to a new task, load its state, and start execution of the new task. The SAB 80286 also supports virtual memory systems by providing a segment-not-present exception and restartable instructions.

Ordering Information

Type	Ordering code	Package	Description
SAB 80286-N	Q67120-C330	PL-CC-68	16-bit microprocessor, 8 MHz
SAB 80286-1-N	Q67120-C269	PL-CC-68	16-bit microprocessor, 10 MHz
SAB 80286-12-N	Q67120-C381	PL-CC-68	16-bit microprocessor, 12.5 MHz

Logic Symbol



Pin Names

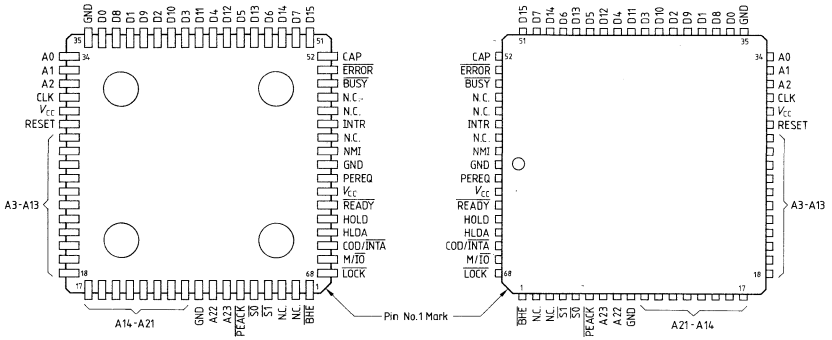
A23-A0	Address Bus
D15-D0	Data Bus
S1, S0	Bus Cycle Status
$\overline{BHE}$	Bus High Enable
M/$\overline{IO}$	Memory/IO Select
COD/$\overline{INTA}$	Code /Interrupt Acknowledge
$\overline{LOCK}$	Bus Lock
CLK	System Clock
RESET	System Reset
$\overline{READY}$	Bus Ready
CAP	Substrate Filter Capacitor
HOLD	Bus Hold Request
HLDA	Bus Hold Acknowledge
INTR	Interrupt Request
NMI	Non-Maskable Interrupt Request
PEREQ	Processor Extension Operand Request
$\overline{PEACK}$	Processor Extension Operand Acknowledge
$\overline{BUSY}$	Processor Extension Busy
$\overline{ERROR}$	Processor Extension Error
V_{CC}	Power Supply (+5V)
GND	Ground (0V)

Pin Configuration

PL-CC-68

Compact pad view – as viewed from underside of component when mounted on the board.

PC board view – as viewed from the component side of the pc board.



Note: N.C. pads must not be connected.

Pin Definitions and Functions

Symbol	Pin	Input (I) Output (O)	Function
$\overline{BHE}$	1	O	**BUS HIGH ENABLE** indicates transfer of data on the upper byte of the data bus D15-8. Eight-bit oriented devices assigned to the upper byte of the data bus would normally use $\overline{BHE}$ to condition chip select functions. $\overline{BHE}$ is active low and floats to tristate off during bus hold acknowledge.

$\overline{BHE}$ and A0 encodings

$\overline{BHE}$ value	A0 value	Function
0	0	Word transfer
0	1	Byte transfer on upper half of data bus (D15-8)
1	0	Byte transfer on lower half of data bus (D7-0)
1	1	Reserved

Symbol	Pin	Input (I) Output (O)	Function
$\overline{S1}$, $\overline{S0}$	4, 5	O	**BUS CYCLE STATUS** indicates initiation of a bus cycle and, along with M/$\overline{IO}$ and COD/$\overline{INTA}$, defines the type of bus cycle. The bus is in a TS state whenever one or both are low, $\overline{S1}$ and $\overline{S0}$ are active low and float to tristate off during bus hold acknowledge.

Bus cycle status definition

COD/$\overline{INTA}$	M/$\overline{IO}$	$\overline{S1}$	$\overline{S0}$	Bus cycle initiated
0 (low)	0	0	0	Interrupt acknowledge
0	0	0	1	Reserved
0	0	1	0	Reserved
0	0	1	1	None; not a status cycle
0	1	0	0	IF A1 = 1 then halt; else shutdown
0	1	0	1	Memory data read
0	1	1	0	Memory data write
0	1	1	1	None; not a status cycle
1 (high)	0	0	0	Reserved
1	0	0	1	I/O read
1	0	1	0	I/O write
1	0	1	1	None; not a status cycle
1	1	0	0	Reserved
1	1	0	1	Memory instruction read
1	1	1	0	Reserved
1	1	1	1	None; not a status cycle

Pin Definitions and Functions (cont'd)

<table>
<tr><th>Symbol</th><th>Pin</th><th>Input (I)
Output (O)</th><th>Function</th></tr>
<tr><td>A23–A0</td><td>7 – 34</td><td>O</td><td>ADDRESS BUS outputs physical memory and I/O port addresses. A0 is low when data is to be transferred on pins D7–0. A23–A10 are low during I/O transfers. The address bus is active high and floats to tristate off during bus hold acknowledge.</td></tr>
<tr><td>RESET</td><td>29</td><td>I</td><td>SYSTEM RESET clears the internal logic of the SAB 80286 and is active high. The SAB 80286 may be reinitialized at any time with a low-to-high transition on RESET which remains active for more than 16 system clock cycles. During RESET active, the output pins of the SAB 80286 enter the state shown below:
<table><tr><th colspan="2">Pin state during reset</th></tr><tr><th>Pin value</th><th>Pin names</th></tr><tr><td>1 (high)</td><td>$\overline{S0}$, $\overline{S1}$, $\overline{PEACK}$, A23–A0, $\overline{BHE}$, $\overline{LOCK}$</td></tr><tr><td>0 (low)</td><td>M/$\overline{IO}$, COD/$\overline{INTA}$, HLDA</td></tr><tr><td>Tristate off</td><td>D15 – D0</td></tr></table>Operation of the SAB 80286 begins after a high-to-low transition on RESET. The high-to-low transition of RESET must be synchronous to the system clock. Approximately 50 system clock cycles from the trailing edge of RESET are required by the SAB 80286 for internal initialization before performing the first bus cycle to fetch code from the power-on execution address.
A low-to-high transition of RESET synchronous to the system clock will end a processor cycle at the second high-to-low transition of the system clock. The low-to-high transition of RESET may be asynchronous to the system clock; however, in this case it cannot be predetermined which phase of the processor clock will occur during the next system clock period. Synchronous low-to-high transitions of RESET are required only for systems where the processor clock must be phase-synchronous to another clock.</td></tr>
<tr><td>CLK</td><td>31</td><td>I</td><td>SYSTEM CLOCK provides the fundamental timing for SAB 80286 systems. It is divided by two (inside the SAB 80286) to generate the processor clock. The internal divide-by-two circuitry can be synchronized to an external clock generator by a low-to-high transition on the RESET input.</td></tr>
<tr><td>D15 – D0</td><td>36 – 51</td><td>I
O</td><td>DATA BUS inputs data during memory, I/O, and interrupt acknowledge read cycles: outputs data during memory and I/O write cycles. The data bus is active high and floats to tristate off during bus hold acknowledge.</td></tr>
<tr><td>$\overline{BUSY}$
$\overline{ERROR}$</td><td>53, 54</td><td>I
I</td><td>PROCESSOR EXTENSION BUSY AND ERROR indicate the operating condition of a processor extension to the SAB 80286. An active $\overline{BUSY}$ input stops the SAB 80286 program execution on WAIT and some ESC instructions until $\overline{BUSY}$ becomes inactive (high). The SAB 80286 may be interrupted while waiting for $\overline{BUSY}$ to become inactive. An active $\overline{ERROR}$ input causes the SAB 80286 to perform a processor extension interrupt when executing WAIT or some ESC instructions. These inputs are active low and may be asynchronous to the system clock.</td></tr>
</table>

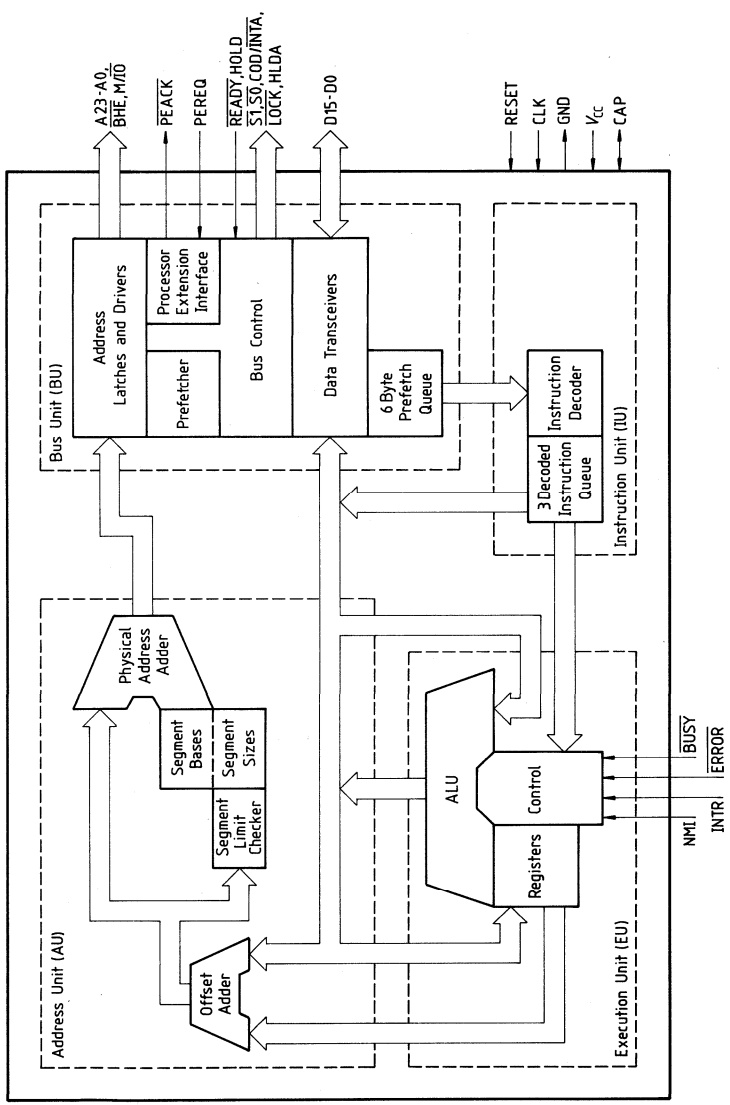
Pin Definitions and Functions (cont'd)

Symbol	Pin	Input (I) Output (O)	Function
INTR	57	I	**INTERRUPT REQUEST** requests the SAB 80286 to suspend its current program execution and service a pending external request. Interrupt requests are masked whenever the interrupt enable bit in the flag word is cleared. When the SAB 80286 responds to an interrupt request, it performs two interrupt acknowledge bus cycles to read an 8-bit interrupt vector that identifies the source of the interrupt. To assure program interruption, INTR must remain active until the first interrupt acknowledge cycle is completed. INTR is sampled at the beginning of each processor cycle and must be active high at least two processor cycles before the current instruction ends in order to interrupt before the next instruction. INTR is level sensitive, active high, and may be asynchronous to the system clock.
NMI	59	I	**NON-MASKABLE INTERRUPT REQUEST** interrupts the SAB 80286 with an internally supplied vector value of 2. No interrupt acknowledge cycles are performed. The interrupt enable bit in the SAB 80286 flag word does not affect this input. The NMI input is active high, may be asynchronous to the system clock, and is edge-triggered after internal synchronization. For proper recognition, the input must have been previously low for at least four system clock cycles and remain high for at least four system clock cycles.
PEREQ $\overline{\text{PEACK}}$	1 6	I O	**PROCESSOR EXTENSION OPERAND REQUEST AND ACKNOWLEDGE** extend the memory management and protection capabilities of the SAB 80286 to processor extensions. The PEREQ input requests the SAB 80286 to perform a data operand transfer for a processor extension. The $\overline{\text{PEACK}}$ output signals the processor extension when the requested operand is being transferred. PEREQ is active high. $\overline{\text{PEACK}}$ is active low and floats to tristate off during bus hold acknowledge. $\overline{\text{PEACK}}$ may be asynchronous to the system clock.
$\overline{\text{READY}}$	63	I	**BUS READY** terminates a bus cycle. Bus cycles are extended without limit until terminated by $\overline{\text{READY}}$ low. $\overline{\text{READY}}$ is an active low synchronous input requiring setup and hold times relative to the system clock be met for correct operation. $\overline{\text{READY}}$ is ignored during bus hold acknowledge.
HOLD HLDA	64 65	I O	**BUS HOLD REQUEST AND HOLD ACKNOWLEDGE** control ownership of the SAB 80286 local bus. The HOLD input allows another local bus master to request control of the local bus. When control is granted, the SAB 80286 will float its bus drivers to tristate off and then activate HLDA, thus entering the bus hold acknowledge condition. The local bus will remain granted to the requesting master until HOLD becomes inactive which results in the SAB 80286 deactivating HLDA and regaining control of the local bus. This terminates the bus hold acknowledge condition, HOLD may be asynchronous to the system clock. These signals are active high.

Pin Definitions and Functions (cont'd)

Symbol	Pin	Input (I) Output (O)	Function
COD/$\overline{\text{INTA}}$	66	O	**CODE/INTERRUPT ACKNOWLEDGE** distinguishes instruction fetch cycles from memory data read cycles. Also distinguishes interrupt acknowledge cycles from I/O cycles. COD/$\overline{\text{INTA}}$ floats to tristate off during bus hold acknowledge.
M/$\overline{\text{IO}}$	67	O	**MEMORY / I/O SELECT** distinguishes memory access from I/O access. If high during TS, a memory cycle or a halt/shutdown cycle is in progress. If low, an I/O cycle or an interrupt acknowledge cycle is in progress M/$\overline{\text{IO}}$ floats to tristate off during bus hold acknowledge.
$\overline{\text{LOCK}}$	68	O	**BUS LOCK** indicates that other system bus masters are not to gain control of the system bus following the current bus cycle. The $\overline{\text{LOCK}}$ signal may be activated explicitly by the "LOCK" instruction prefix or automatically by SAB 80286 hardware during memory XCHG instructions, interrupt acknowledge, or descriptor table access. $\overline{\text{LOCK}}$ is active low and floats to tristate off during bus hold acknowledge.
V_{CC}	30, 62	–	**POWER SUPPLY** (+5 V)
GND	9, 35, 60	–	**GROUND** (0 V)
CAP	52	I	**SUBSTRATE FILTER CAPACITOR:** a 0.047 µF ±20% 12 V capacitor must be connected between this pin and ground. This capacitor filters the output of the internal substrate bias generator. A maximum DC leakage current of 1 µA is allowed through the capacitor. For correct operation of the SAB 80286 the substrate bias generator must charge this capacitor to its operating voltage. The capacitor's charging time is 5 milliseconds (max.) after V_{CC} and CLK reach their specified AC and DC parameters. RESET may be applied to prevent spurious activity by the CPU during this time. After this time, the SAB 80286 processor clock can be phase-synchronized to another clock by pulsing RESET low synchronous to the system clock.

Internal Block Diagram



Functional Description

Introduction

The SAB 80286 is an advanced, high-performance microprocessor with specially optimized capabilities for multiple user and multitasking systems. Depending on the application, the SAB 80286's performance is up to six times faster than that of the standard 5 MHz SAB 8086, while providing complete upward software compatibility with the Siemens 16-bit CPU family (SAB 8086/88, SAB 80186/88).

The SAB 80286 operates in two modes: real address mode (8086 mode) and protected virtual address mode. Both modes execute a superset of the SAB 8086/88 instruction set. In real address mode programs use real addresses with up to one megabyte of address space. Programs use virtual addresses in protected virtual address mode, also called protected mode. In protected mode, the SAB 80286 CPU automatically maps 1 gigabyte of virtual addresses per task into a 16 megabyte real address space. This mode also provides memory protection to isolate the operating system and ensure privacy of each task's programs and data. Both modes provide the same basic instruction set, registers, and addressing modes.
The following functional description describes first the basic SAB 80286 architecture common to both modes, second the real address mode, and third the protected mode.

Basic Architecture

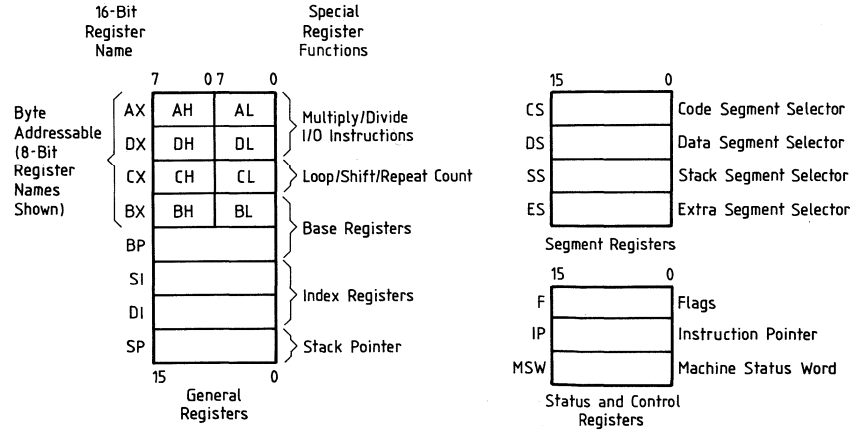
The processors of the Intel/Siemens 16-bit CPU family all contain the same basic set of registers, instructions, and addressing modes. Therefore, the SAB 80286 processor is upward-compatible with the SAB 8086, 8088 and 80186 CPUs.

Register Set

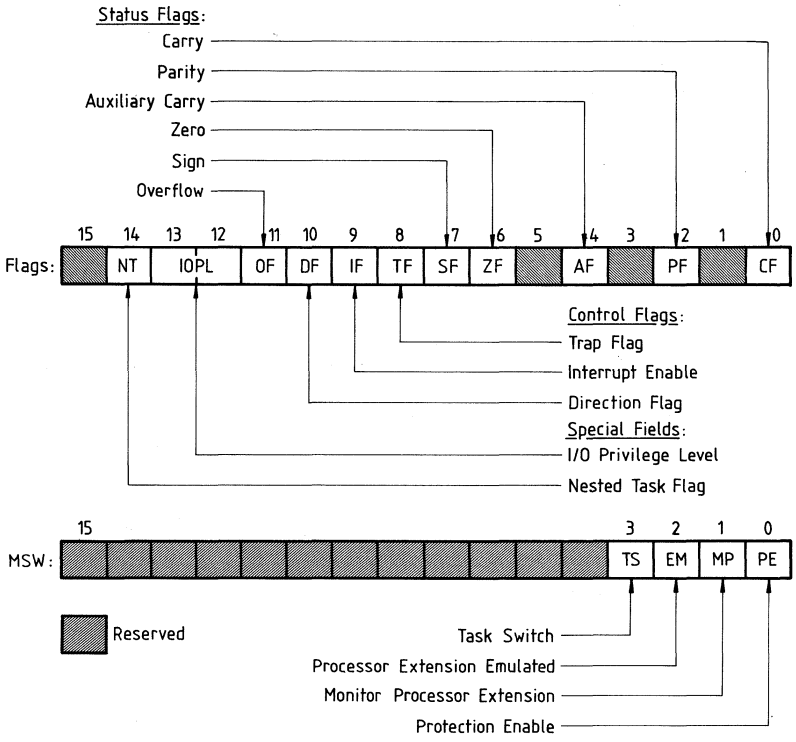
The SAB 80286 basic architecture has fifteen registers as shown below. These registers are grouped into the following four categories:

General registers: Eight 16-bit general purpose registers used to contain arithmetic and logical operands. Four of these (AX, BX, CX, and DX) can be used either in their entirety as 16-bit words or split into pairs of separate 8-bit registers.

Register Set



Status and Control Register Bit Functions



Segment registers: Four 16-bit special purpose registers select, at any given time, the segments of memory that are immediately addressable for code, stack, and data (For usage, refer to Memory Organization).

Base and index registers: Four of the general purpose registers may also be used to determine offset addresses of operands in memory. These registers may contain base addresses or indexes to particular locations within a segment. The addressing mode determines the specific registers used for operand address calculations.

Status and control registers: The three 16-bit special purpose registers in the figure below record or control certain aspects of the SAB 80286 processor state including the instruction pointer which contains the offset address of the next sequential instruction to be executed.

Flags Word Description

The flags word (flags) records specific characteristics of the result of logical and arithmetic instructions (bits 0, 2, 4, 7, and 11) and controls the operation of the SAB 80286 within a given operating mode (bits 8 and 9). Flags is a 16-bit register. The function of the flag bits is given in table 1.

Instruction Set

The instruction set is divided into seven categories: data transfer, arithmetic, shift/rotate/logical, string manipulation, control transfer, high level instructions, and processor control.

An SAB 80286 instruction can reference zero, one, or two operands; where an operand resides in a register, in the instruction itself, or in memory. Zero-operand instructions (e.g. HLT) are usually one byte long. One-operand instructions (e.g. INC and DEC) are usually two bytes long but some are encoded in only one byte. One-operand instructions may reference a register or memory location.
Two-operand instructions permit the following six types of instruction operations:
- register to register
- memory to register
- immediate data to register
- memory to memory
- register to memory
- immediate data to memory

Two-operand instructions (e.g. MOV and ADD) are usually three to six bytes long. Memory-to-memory operations are provided by a special class of string instructions requiring one to three bytes. For detailed instruction formats and encodings refer to the instruction set summary.

Table 1
Flags Word Bit Functions

Bit position	Name	Functions
0	CF	Carry Flag – Set on high-order bit carry or borrow; cleared otherwise
2	PF	Parity Flag – Set if low-order 8 bits of result contain an even number of 1-bits; cleared otherwise
4	AF	Set on carry from or borrow to the low-order 4 bits of AL; cleared otherwise
6	ZF	Zero Flag – Set if result is zero; cleared otherwise
7	SF	Sign Flag – Set equal to high-order bit of result (0 if positive, 1 if negative)
11	OF	Overflow Flag – Set if result is a positive number too large or a negative number too small (excluding sign bit) to fit in destination operand; cleared otherwise
8	TF	Single Step Flag – Once set, a single step interrupt occurs after the next instruction has been executed. TF is cleared by the single step interrupt
9	IF	Interrupt Enable Flag – When set, maskable interrupts will cause the CPU to transfer control to an interrupt vector specified location
10	DF	Direction Flag – Causes string instructions to autodecrement the appropriate index registers when set. Clearing DF causes autoincrement

Memory Organization

Memory is organized as sets of variable length segments. Each segment is a linear contiguous sequence of up to 64 K (2^{16}) 8-bit bytes. Memory is addressed using a two-component address (a pointer) that consists of a 16-bit segment selector, and a 16-bit offset. The segment selector indicates the desired segment in memory. The offset component indicates the desired byte address within the segment.

All instructions that address operands in memory must specify the segment and the offset. For speed and compact instruction encoding, segment selectors are usually stored in the high-speed segment registers. An instruction needs to specify only the desired segment register and an offset in order to address a memory operand.

Most instructions need not explicity specify which segment register is used. The correct segment register is automatically chosen according to the rules of table 2.

These rules follow the way programs are written as independent modules that require areas for code and data, a stack, and access to external data areas. Special segment override instruction prefixes allow the implicit segment register selection rules to be overridden for special cases. The stack, data, and extra segments may coincide for simple programs. To access operands not residing in one of the four immediately available segments, a full 32-bit pointer or a new segment selector must be loaded.

I/O Space

The I/O space consists of 64 K 8-bit or 32 K 16-bit ports. I/O instructions address the I/O space with either an 8-bit port address, specified in the instruction, or a 16-bit port address in the DX register, 8-bit port addresses are zero extended such that A15-A8 are low. I/O port addresses 00F8(H) through 00FF(H) are reserved.

Addressing Modes

The SAB 80286 provides a total of eight adressing modes for instructions to specify operands. Two addressing modes are provided for instructions that operate on register or immediate operands:

Register operand mode: The operand is located in one of the 8 or 16-bit general registers.

Immediate operand mode: The operand is included in the instruction.

Direct mode: The operand's offset is contained in the instruction as an 8 or 16-bit displacement element.

Two Component Address

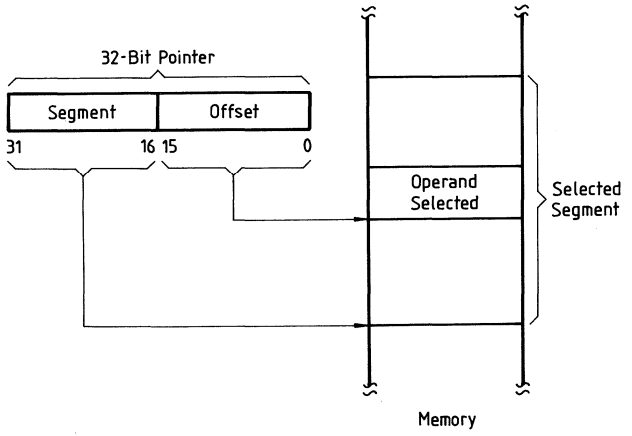


Table 2
Segment Register Selection Rules

Memory reference needed	Segment register used	Implicit segment selection rule
Instructions	Code (CS)	Automatic with instruction prefetch
Stack	Stack (SS)	All stack pushes and pops. Any memory reference which uses BP as a base register
Local data	Data (DS)	All data references except when relative to stack or string destination
External (global) data	Extra (ES)	Alternate data segment and destination of string operation

Register indirect mode: The operand's offset is in one of the registers SI, DI, BX, or BP.

Based mode: The operand's offset is the sum of an 8 or 16-bit displacement and the contents of a base register (BX or BP).

Indexed mode: The operand's offset is the sum of an 8 or 16-bit displacement and the contents of an index register (SI or DI).

Based indexed mode: The operand's offset is the sum of the contents of a base register and an index register.

Based indexed mode with displacement: The operand's offset is the sum of a base register's contents, an index register's contents, and an 8 or 16-bit displacement.

Data Types

The SAB 80286 directly supports the following data types:

Integer:
A signed binary numeric value contained in an 8-bit byte or a 16-bit word. All operations assume a 2's complement representation. Signed 32 and 64-bit integers are supported using the numeric data processor extension.

Ordinal:
An unsigned binary numeric value contained in an 8-bit byte or 16-bit word.

Pointer:
A 32-bit quantity, composed of a segment selector component and an offset component. Each component is a 16-bit word.

String:
A contiguous sequence of bytes or words. A string may contain between 1 byte and 64 Kbytes.

ASCII:
A byte representation of alphanumeric and control characters using the ASCII standard of character representation.

BCD:
A byte (unpacked) representation of the decimal digits 0 to 9.

Packed BCD:
A byte (packed) representation of two decimal digits 0 to 9 storing one digit in each nibble of the byte.

Floating Point:
A signed 32, 64, or 80-bit real number representation (Floating point operands are supported using the extended processor configuration with SAB 80287).

Interrupts

An interrupt transfers execution to a new program location. The old program address (CS:IP) and machine state (flags) are saved on the stack to allow resumption of the interrupted program. Interrupts fall into three classes: hardware initiated, INT instructions, and instruction exceptions. Hardware-initiated interrupts occur in response to an external input and are classifield as non-maskable or maskable. Programs may cause an interrupt with an INT instruction. Instruction exceptions occur when an unusual condition, which prevents further instruction processing, is detected while attempting to execute an instruction. The return address from an exception will always point at the instruction causing the exception and include any leading instruction prefixes.

A table containing up to 256 pointers defines the proper interrupt service routine for each interrupt. Interrupts 0 to 31, some of which are used for instruction exceptions, are reserved. For each interrupt, an 8-bit vector must be supplied to the SAB 80286 which identifies the appropriate table entry. Exceptions supply the interrupt vector internally. INT instructions contain or imply the vector and allow access to all 256 interrupts. Maskable hardware initiated interrupts supply the 8-bit vector to the CPU during an interrupt acknowledge bus sequence. Non-maskable hardware interrupts use a predefined internally supplied vector.

Single Step Interrupt

The SAB 80286 has an internal interrupt that allows programs to execute one instruction at a time. It is called the single step interrupt and is controlled by the single step flag bit (TF) in the flag word. Once this bit is set, an internal single step interrupt will occur after the next instruction has been executed. The interrupt clears the TF bit and uses an internally supplied vector of 1. The IRET instruction is used to set the TF bit and transfer control to the next instruction to be single-stepped.

Interrupt Priorities

When simultaneous interrupt requests occur, they are processed in a fixed order as shown in table 4. Interrupt processing involves saving the flags, return address, and setting CS:IP to point at the first instruction of the interrupt handler. If other interrupts remain enabled they are processed before the first instruction of the current interrupt handler is executed. The last interrupt processed is therefore the first one serviced.

Table 3
Interrupt Vector Assignments

Function	Interrupt Number	Related instructions	Return address before instruction causing exception?
Divide error exception	0	DIV, IDIV	Yes
Single step interrupt	1	All	–
NMI interrupt	2	All	–
Breakpoint interrupt	3	INT	–
INT0 detected overflow exception	4	INT0	No
BOUND range exceeded exception	5	BOUND	Yes
Invalid op code exception	6	any undefined op code	Yes
Processor extension not available exception	7	ESC or WAIT	Yes
Reserved	8–15		–
Processor extension error interrupt	16	ESC or WAIT	–
Reserved	17–31		–
User defined	32–255		–

Table 4
Interrupt Processing Order

Order	Interrupt
1	Instruction exception
2	Single step
3	NMI
4	Processor extension segment overrun
5	INTR
6	INT instruction

Initialization and Processor Reset

Processor initialization or start up is accomplished by driving the RESET input pin high. RESET forces the SAB 80286 to terminate all execution and local bus activity. No instruction or bus activity will occur as long as RESET is active. After RESET became inactive and an internal processing interval has elapsed, the SAB 80286 begins execution in real address mode with the instruction at physical location FFFFF0(H). RESET also sets some registers to predefined values as shown in table 5. A23 to A20 will be high when the SAB 80286 performs memory references relative to the CS register until CS is changed. A23 to A20 will be zero for references to the DS, ES, or SS segments.
Changing CS in real address mode will force A23 to A20 low whenever CS is used again. The initial CS:IP value of F000:FFF0 provides 64 Kbytes of code space for initialization code without changing CS.

Table 5
SAB 80286 Initial Register State after RESET

Flag word	0002(H)
Machine status word	FFF0(H)
Instruction pointer	FFF0(H)
Code segment	F000(H)
Data segment	0000(H)
Extra segment	0000(H)
Stack segment	0000(H)

Machine Status Word Description

The machine status word (MSW) records when a task switch takes place and controls the operating mode of the SAB 80286. It is a 16-bit register of which the lower four bits are used. One bit places the CPU into protected mode, while the other three bits, as shown in table 6, control the processor extension interface. After RESET, this register contains FFF0(H) which place the SAB 80286 in real address mode.

The LMSW ans SMSW instructions can load and store the MSW in real address mode. The recommended use ot TS, EM, and MP is shown in table 7.

Table 6
MSW Bit Functions

Bit position	Name	Function
0	PE	**P**rotected mode **e**nable places the SAB 80286 into protected mode and cannot be cleared except by RESET
1	MP	**M**onitor **p**rocessor extension allows WAIT instructions to cause a processor extension not present exception (number 7)
2	EM	**Em**ulate processor extension causes a processor extension not present exception (number 7) on ESC instructions to allow emulating a processor extension
3	TS	**T**ask **s**witched indicates that the next instruction using a processor extension will cause exception 7, allowing software to test whether the current processor extension context belongs to the current task

Table 7
Recommended MSW Encodings For Processor Extension Control

TS	MP	EM	Recommended use	Instructions causing exception 7
0	0	0	Initial encoding after RESET. SAB 20286 operation is identical with SAB 8086/88 operation	none
0	0	1	No processor extension is available. Software will emulate its function	ESC
1	0	1	No processor extension is available. Software will emulate its function. The current processor extension context may belong to another task	ESC
0	1	0	A processor extension exists	none
1	1	0	A processor extension exists. The current processor extension context may belong to another task. The exception on WAIT allows software to test for an error pending from a previous processor extension operation	ESC or WAIT

Halt

The HLT instruction stops program execution and prevents the CPU from using the local bus until restarted. Either NMI, INTR with IF = 1, or RESET will force the SAB 80286 out of halt. If interrupted the saved CS:IP will point to the next instruction after the HLT.

Real Address Mode

The SAB 80286 executes a fully upward-compatible superset of the SAB 8086's instruction set in real address mode. In real address mode, the SAB 80286 is object code compatible with SAB 8086 and SAB 8088 software. The real address mode architecture (registers and addressing modes) is exactly as described in the SAB 80286 basic architecture section of this functional description.

Memory Size

Physical memory is a contiguous array of up to 1,048,576 bytes (one megabyte) addressed by pins A0 through A19 and $\overline{BHE}$. A20 through A23 may be ignored.

Memory Addressing

In real address mode the processor generates 20-bit physical addresses directly from a 20-bit-segment base address and a 16-bit offset.

The selector portion of a pointer is interpreted as the upper 16 bits of a 20-bit segment address. The lower four bits of the 20-bit segment addresses are always zero. Segment addresses, therefore, begin on multiples of 16 bytes. See figure on address calculation for a graphic representation of address formation.

All segments in real address mode are 64 Kbytes in size and may be read, written, or executed. An exception or interrupt can occur if data operands or instructions attempt to wrap around the end of a segment (e.g. a word with its low-order byte at offset FFFF(H) and its high-order byte at offset 0000(H). If, in real address mode, the information contained in a segment does not use the full 64 Kbytes, the unused end of the segment may be overlayed by another segment to reduce physical memory requirements.

Real Address Mode, Address Calculation

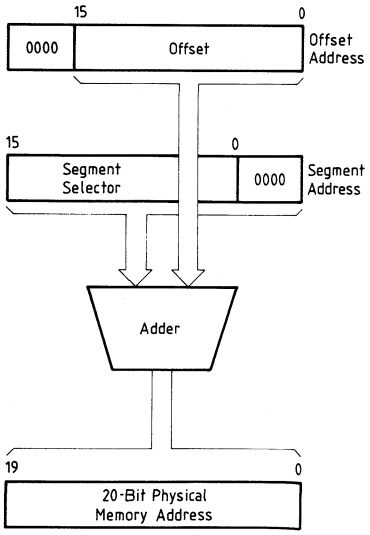


Table 8
Real Address Mode, Addressing Interrupts

Function	Interrupt number	Related instructions	Return address before instruction?
Interrupt table limit too small exception	8	INT vector is not within table limit	Yes
Processor extension segment overrun interrupt	9	ESC with memory operand extending beyond offset FFFF(H)	No
Segment overrun exception	13	Word memory reference with offset = FFFF(H) or an attempt to execute past the end of a segment	Yes

Interrupts

Table 8 shows the interrupt vectors reserved for exceptions and interrupts which indicate an addressing error. The exceptions leave the CPU in the state existing before attempting to execute the failing instruction (except for PUSH, POP, PUSHA, or POPA).

Shutdown

Shutdown occurs when a severe error is detected that prevents further instruction processing by the CPU. Shutdown and halt are externally signalled via a halt bus operation. They can be distinguished by A1 high for halt and A1 low for shutdown.
In real address mode, shutdown can occur under two conditions:

- Exceptions 8 or 13 happen and the IDT limit does not include the interrupt vector.
- A CALL, INT or POP instruction attempts to wrap around the stack segment when SP is not even.

An NMI input can bring the CPU out of shutdown if the IDT limit is at least 000F(H) and SP is greater than 0005(H), otherwise shutdown can only be exited via the RESET input.

Protected Virtual Address Mode

The SAB 80286 executes a fully upward-compatible superset of the SAB 8086 instruction set in protected virtual address mode (protected mode). Protected mode also provides memory management and protection mechanisms and associated instructions.

The SAB 80286 enters protected virtual address mode from real address mode by setting the PE (Protection Enable) bit of the machine status word with the Load Machine Status Word (LMSW) instruction. Protected mode offers extended physical and virtual memory address space, memory protection mechanisms, and new operations to support operating system and virtual memory.

All registers, instructions and addressing modes described in the SAB 80286 basic architecture section of the functional description remain the same. Programs for the SAB 8086, SAB 8088, SAB 80186 and real address mode SAB 80286 can be run in protected mode: however, embedded constants for segment selectors are different.

Memory Size

The protected mode SAB 80286 provides a 1 gigabyte virtual address space per task mapped into a 16 megabyte physical address space defined by the address pins A23–A0 and $\overline{BHE}$. The virtual address space may be larger than the physical address space since any use of an address that does not map to a physical memory location will cause a restartable exception.

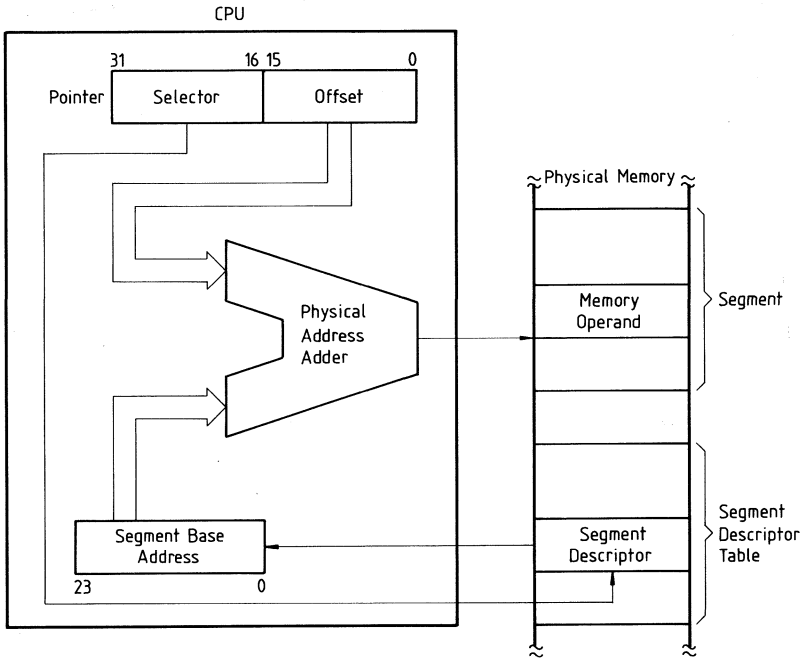
Memory Addressing

As in real address mode, protected mode uses 32-bit pointers, consisting of 16-bit selector and offset components. The selector, however, specifies an index into a memory resident table rather than the upper 16-bits of a real memory address. The 24-bit base address of the desired segment is obtained from the tables in memory. The 16-bit offset is added to the segment base address to form the physical address as shown in the figure below. The tables are automatically referenced by the CPU whenever a segment register is loaded with a selector. All SAB 80286 instructions which load a segment register will reference the memory-based tables without additional software. The memory-based tables contain 8 byte values called descriptors.

Descriptors

Descriptors define the use of memory. Special types of descriptors also define new functions for transfer of control and task switching. The SAB 80286 has segment descriptors for code, stack and data segments, as well as system control descriptors for special system data segments and control transfer operations. Descriptor accesses are performed as locked bus operations to assure descriptor integrity in multiprocessor systems.

Protected Mode, Memory Addressing

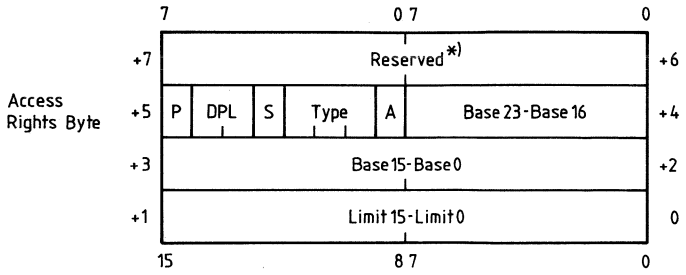


Code and data segment descriptors (S = 1)

Besides segment base addresses, code and data descriptors contain other segment attributes including segment size (1 to 64 Kbytes), access rights (read only, read/write, execute only, and execute/read), and presence in memory (for virtual memory systems; figure and table next page). Any segment usage violating a segment attribute indicated by the segment descriptor will prevent the memory cycle and cause an exception or interrupt.

Code and data (including stack data) are stored in two types of segments: code segments and data segments. Both types are indentified and defined by segment descriptors (S = 1). Code segments are identified by the executable (E) bit se to 1 in the descriptor access rights byte, whereas the data segments have the E bit set to Ø.

Code or Data Segment Descriptor



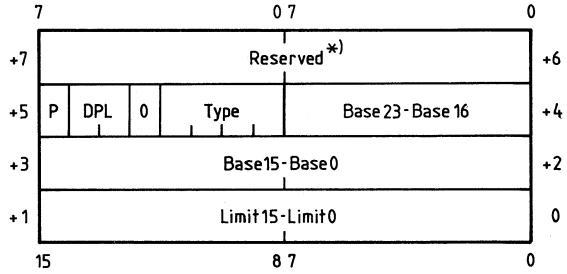
Access Rights Byte Definition

	Bit Position	Name	Function		
	7	Present (P)	P = 1	Segment is mapped into physical memory	
			P = 0	No mapping to physical memory exists, base and limit are not used	
	6-5	Descriptor privilege level (DPL)		Segment privilege attribute used in privilege tests	
	4	Segment descriptor (S)	S = 1	Code or data (includes stacks) segment descriptor	
			S = 0	System segment descriptor or gate descriptor	
Type field definition	3	Executable (E)	E = 0	Data segment descriptor type is:	If data segment (S = 1, E = 0)
	2	Expansion direction (ED)	ED = 0	Expand up segment, offsets must be ≤ limit	
			ED = 1	Expand down segment, offsets must be > limit	
	1	Writeable (W)	W = 0	Data segment may not be written into	
			W = 1	Data segment may be written into	
	3	Executable (E)	E = 1	Code segment descriptor type is:	If code segment (S = 1, E = 1)
	2	Conforming (C)	C = 1	Code segment may only be executed when CPL ≥ DPL and CPL remains unchanged	
	1	Readable (R)	R = 0	Code segment may not be read	
			R = 1	Code segment may be read	
	0	Accessed (A)	A = 0	Segment has not be accessed	
			A = 1	Segment selector has been loaded into segment register or used by selector test instructions	

System segment descriptors (S = 0, type = 1-3)

In addition to code and data segment descriptors, the protected mode SAB 80286 defines system segment descriptors. These descriptors define special system data segments which contain a table of descriptors (local descriptor table descriptor) or segments which contain the execution state of a task (task state segment descriptor).
The figure and table on next page show the formats for the special system data segment descriptors.

System Segment Descriptor



System Segment Descriptor Fields

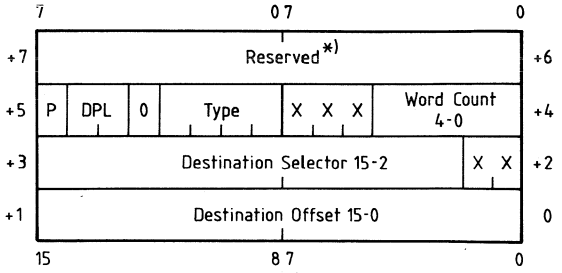
Name	Value	Description
TYPE	1 2 3	Available task state segment Local descriptor table descriptor Busy task state segment
P	0 1	Descriptor contents are not valid Descriptor contents are valid
DPL	0–3	Descriptor privilege level
BASE	24-bit number	Base address of special system data segment in real memory
LIMIT	16-bit number	Offset of last byte in segment

Gate descriptors (S = 0, Type = 4-7)

Gates are used to control access to entry points within the target code segment. The gate descriptors are **call** gates, **task** gates, **interrupt** gates and **trap** gates. Gates provide a level of indirection between the source and destination of the control transfer. This indirection allows the CPU to automatically perform protection checks and control entry point of the destination. Call gates are used to change privilege levels (see privilege), task gates are used to perform a task switch, and interrupt and trap gates are used to specify interrupt service routines. The interrupt gate disables interrupts (resets IF) while the trap gate does not.

The figure and table on the next page show the format of the gate descriptors. The descriptor contains a destination pointer that points to the descriptor of the target segment and the entry point offset. The destination selector in an interrupt gate, trap gate, and call gate must refer to a code segment descriptor. Exception 13 is generated when the gate is used if a destination selector does not refer to the correct descriptor type.

Gate Descriptor



Gate Descriptor Fields

Name	Value	Description
TYPE	4 5 6 7	– Call gate – Task gate – Interrupt gate – Trap gate
P	0 1	– Descriptor contents are not valid – Descriptor contents are valid
DPL	0–3	Descriptor privilege level
WORD COUNT	0–31	Number of words to copy from callers stack to called procedures stack. Only used with call gate
DESTINATION SELECTOR	16-bit selector	Selector to the target code segment (call, interrupt or trap gate)
DESTINATION OFFSET	16-bit offset	Entry point within the target code segment

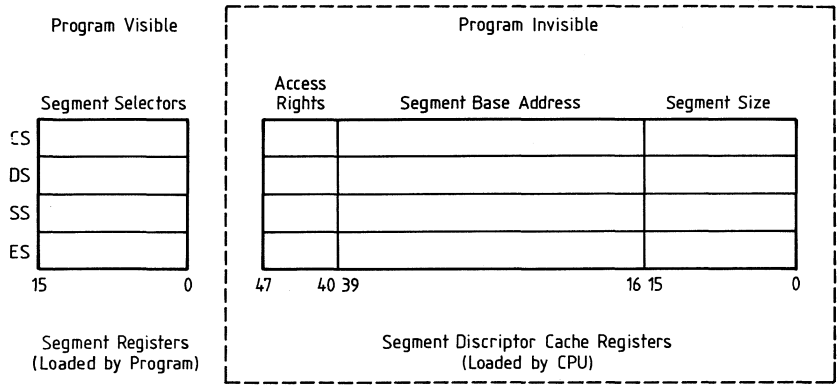
Segment descriptor cache registers

A segment descriptor cache register is assigned to each of the four segment registers (CS, SS, DS, ES). Segment descriptors are automatically loaded (cached) into a segment descriptor cache register (see figure) whenever the associated segment register is loaded with a selector. Only segment descriptors may be loaded into segment descriptor cache registers. Once loaded, all references to that segment of memory use the cached descriptor information instead of reaccessing memory. The descriptor cache registers are not visible to programs. No instructions exist to store their contents. They only change when a segment register is loaded.

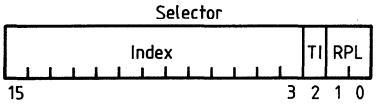
Selector fields

A protected mode selector has three fields: descriptor entry index, local or global descriptor table indicator (TI), and selector privilege (RPL) as shown in the figure on selector fields. These fields select one of two memory-based tables of descriptors, select the appropriate table entry and allow highspeed testing of the selector's privilege attribute (refer to privilege discussion below).

Descriptor Cache Registers



Selector Fields

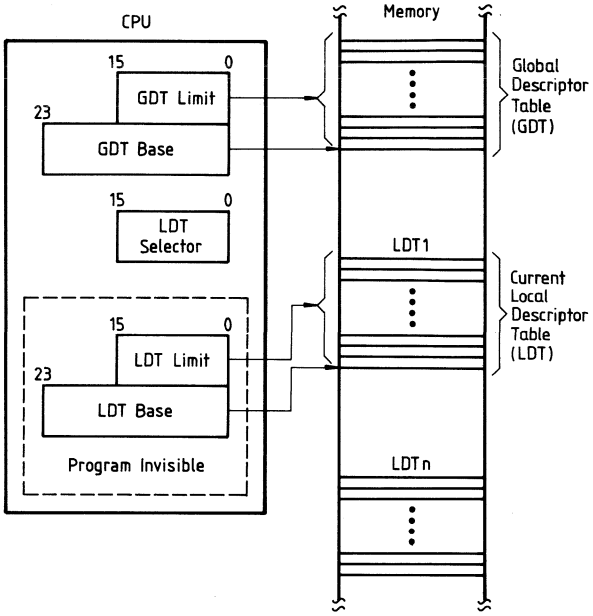


Bits	Name	Function
1-0	Requested Privilege Level (RPL)	Indicates Selector Privilege Level Desired
2	Table Indicator (TI)	TI = 0 Use Global Descriptor Table (GDT) TI = 1 Use Local Descriptor Table (LDT)
15-3	Index	Select Descriptor Entry in Table

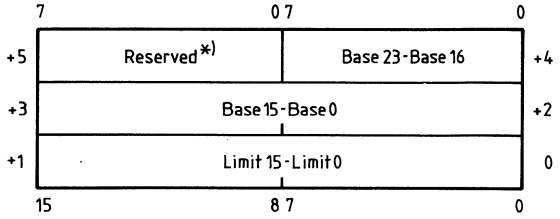
Local and global descriptor tables (LDT, GDT)

Two tables of descriptors, called descriptor tables, contain all descriptors accessible by a task at any given time. A descriptor table is a linear array of up to 8192 descriptors. The upper 13 bits of the selector value are an index into a descriptor table. Each table has a 24-bit base register to locate the descriptor table in physical memory and a 16-bit limit register that confine descriptor access to the defined limits of the table as shown in the figure below. A restartable exception (13) will occur if an attempt is made to reference a descriptor outside the table limits.

Local and Global Descriptor Table Definition



Global Descriptor Table and Interrupt Descriptor Table Data Type

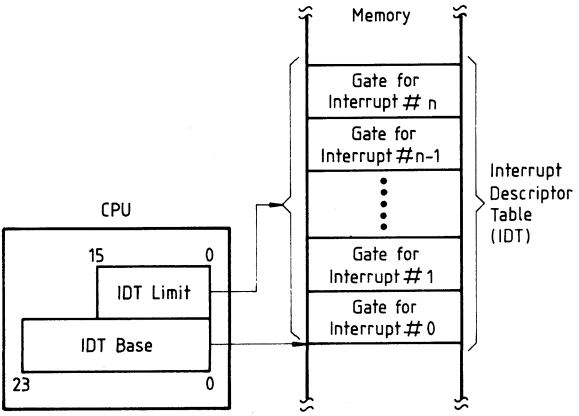


*) Must be set to 0 for compatibility with SAB 80386

Interrupt descriptor table

The protected mode SAB 80286 has a third descriptor table, called the interrupt descriptor table (DT) (see top figure on next page), used to define up to 256 interrupts. It may contain only task gates, interrupt gates and trap gates. The IDT (interrupt descriptor table) has a 24-bit base and 16-bit limit register in the CPU. References to IDT entries are made via INT instructions, external interrupt vectors, or exceptions. The IDT must be at least 256 bytes in size to allocate space for all reserved interrupts.

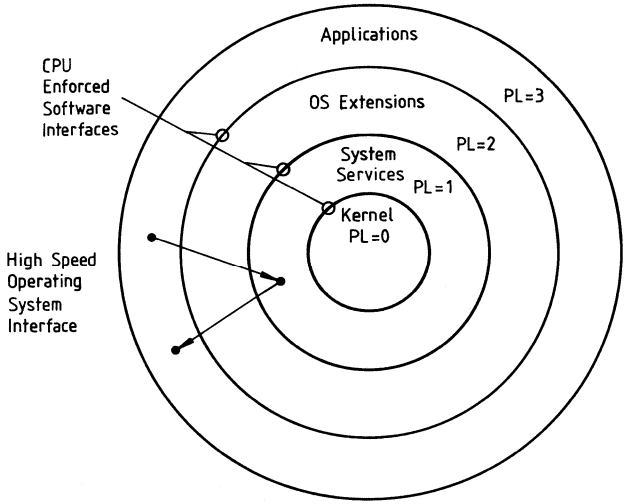
Interrupt Descriptor Table Definition



Privilege

The SAB 80286 has a four-level hierarchical privilege system which controls the use of privileged instructions and access to descriptors (and their associated segments) within a task (figure below). The privilege levels are numbered 0 through 3. Level 0 is the most privileged level.

Hierarchical Privilege Levels



Protection

The SAB 80286 includes mechanisms to protect critical instructions that affect the CPU execution state (e.g. HLT) and code or data segments from improper usage. These mechanisms are grouped under the term "protection" and have three forms:

Restricted usage of segments (e.g. no write allowed to read-only data segments). The only segments available for use are defined by descriptors in the local descriptor table (LDT) and global descriptor table (GDT).

Restricted access to segments via the rules of privilege and descriptor usage.

Privileged instructions or operations that may only be executed at certain privilege levels as determined by the CPL and I/O privilege level (IOPL). The IOPL is defined by bits 14 and 13 of the flag word.

These checks are performed for all instructions and can be split into three categories: segment load checks (table 9), operand reference checks (table 10), and privileged instruction checks (table 11). Any violation of the rules shown will result in an exception. A not-present exception related to the stack segment causes exception 12.

The IRET and POPF instructions do not perform some of their defined functions if CPL is not of sufficient privilege (numerically small enough). No exceptions or other indication are given when these conditions occur.

The IF bit is not changed if CPL > IOPL.
The IOPL field of the flag word is not changed if CPL > 0.

Table 9
Segment Register Load Checks

Error description	Exception number
Descriptor table limit exceeded	13
Segment descriptor not present	11 or 12
Privilege rules violated	13
Invalid descriptor/segment type segment register load: – read only data segment load to SS – special control descriptor load to DS, ES, SS – execute only segment load to DS, ES, SS – data segment load to CS – read/execute code segment load to SS	13

Table 10
Operand Reference Checks

Error description	Exception number
Write into code segment	13
Read from execute-only code segment	13
Write to read-only data segment	13
Segment limit exceeded [1]	12 or 13

[1] Carry out in offset calculations is ignored.

Table 11
Privileged Instruction Checks

Error description	Exception number
CPL > 0 when executing the following instructions LIDT, LLDT, LGDT, LTR, LMSW, CTS, HLT	13
CPL > IOPL when executing the following instructions INS, IN, OUTS, OUT, STI, CLI, LOCK	13

Exceptions

The SAB 80286 detects several types of exceptions and interrupts in protected mode (see table 12). Most of them are restartable after the exceptional condition is removed. Interrupt handlers for most exceptions receive an error code, pushed on the stack after the return address, that identifies the selector involved (0 if none). The return address normally points to the failing instruction, including all leading prefixes. For a processor extension segment overrun exception, the return address will not point at the ESC instruction that caused the exception; however, the processor extension registers may contain the address of the failing instruction.

Table 12
Protected Mode Exceptions

Interrupt vector	Function	Return address at failing instruction?	Always restart-able?	Error code on stack?
8	Double exception detected	yes	no	yes
9	Processor extension segment overrun	no	no	no
10	Invalid task state segment	yes	yes	yes
11	Segment not present	yes	yes	yes
12	Stack segment overrun or segment not present	yes	yes[1)]	yes
13	General protection	yes	no	yes

[1)] When a PUSHA instruction attempts to wrap around the stack segment, the machine state after the exception will not be restartable. This condition is identified by the value of the saved SP being either 0000(H), 0001(H), FFFE(H), or FFFF(H).

Special Operations

Task switch operation

The SAB 80286 provides a built-in task switch operation which saves the entire SAB 80286 execution state (registers, address space, and a link to the previous task), loads a new execution state, and commences execution in the new task. Like gates, the task switch operation is invoked by executing an inter-segment JMP or CALL instruction which refers to a task state segment (TSS) or task gate descriptor in the GDT or LDT. An INT instruction, exception, or external interrupt may also invoke the task switch operation by selecting a task gate descriptor in the associated IDT descriptor entry.

The TSS descriptor points at a segment (see figure on next page) containing the entire SAB 80286 execution state while a task gate descriptor contains a TSS selector. The limit field must be > 002B(H).

The task state segment is marked busy by changing the descriptor type field from type 1 to type 3. Use of a selector that references a busy task state segment causes exception 13.

Processor extension context switching

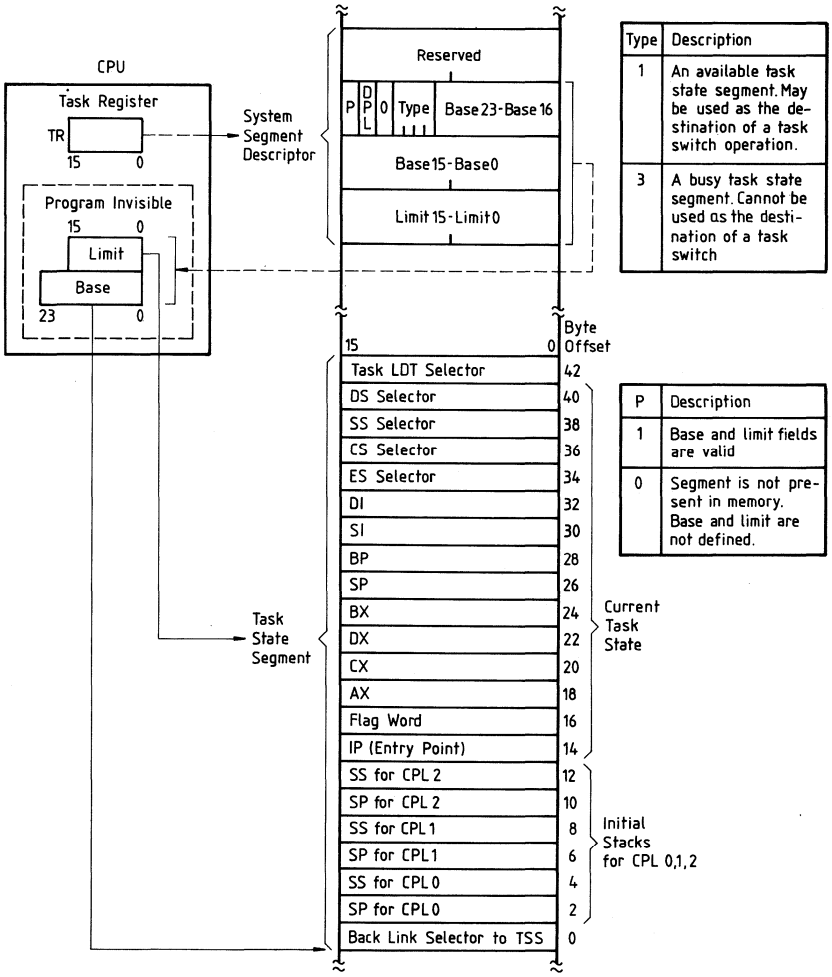
The context of a processor extension (such as the SAB 80287 numerics processor) is not changed by the task switch operation. A processor extension context need only be changed when a different task attempts to use the processor extension (which still contains the context of a previous task).

Whenever the SAB 80286 switches tasks, it sets the task switched (TS) bit of the MSW. TS indicates that a processor extension context may belong to a different task than the current one. The processor extension not present exception (7) will occur when attempting to execute an ESC or WAIT instruction if TS = 1 and a processor extension is present (MP = 1 in MSW).

Double fault and shutdown

If two seperate exceptions are detected during a single instruction execution, the SAB 80286 performs the double fault exception (8). If an exception occurs during processing of the double fault exception, the SAB 80286 will enter shutdown. During shutdown no further instructions or exceptions are processed. Either NMI (CPU remains in protected mode) or RESET (CPU exits protected mode) can force the SAB 80286 out of shutdown. Shutdown is externally signalled via a HALT bus operation with A1 low.

Task State Segment and TSS Registers



Type	Description
1	An available task state segment. May be used as the destination of a task switch operation.
3	A busy task state segment. Cannot be used as the destination of a task switch

P	Description
1	Base and limit fields are valid
0	Segment is not present in memory. Base and limit are not defined.

System Interface

The SAB 80286 system interface appears in two forms: a local bus and a system bus. The local bus consists of address, data, status, and control signals at the pins of the CPU. A system bus is any buffered version of the local bus. A system bus may also differ from the local bus in terms of coding of status and control lines and/or timing and loading of signals. The SAB 80286 family includes several devices to generate standard system buses such as the IEEE 796 standard Multibus™ and the IEEE 796 AMS-M Bus.

Bus Interface Signals and Timing

The SAB 80286 local bus interfaces the SAB 80286 to local memory and I/O components. The interface has 24 address lines, 16 data lines, and 8 status and control signals.

The SAB 80286 CPU, SAB 82284 clock generator, SAB 82288 bus controller, SAB 82289 bus arbiter, SAB 8286A/8287A transceivers, and SAB 8282A/8283A latches provide a buffered and decoded system bus interface. The SAB 82284 generates the system clock and synchronizes $\overline{\text{READY}}$ and RESET. The SAB 82288 converts bus operation status encoded by the SAB 80286 into command and bus control signals.

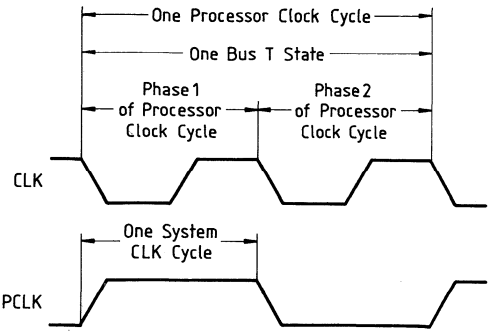
The SAB 82289 bus arbiter generates Multibus bus arbitration signals. These components can provide the timing and electrical power drive levels required for most system bus interfaces including the Multibus.

Physical Memory and I/O Interface

A maximum of 16 megabytes of physical memory can be addressed in protected mode. One megabyte can be addressed in real address mode. Memory is accessible as bytes or words. Words consist of any two consecutive bytes addressed with the least significant byte stored in the lowest address.

The I/O address space contains 64 K addresses in both modes. The I/O space is accessible as either bytes or words, as is memory. Byte-wide peripheral devices may be attached to either the upper or lower byte of the data bus. An interrupt controller such as the SAB 8259A must be connected to the lower byte of the data bus (D7-D0) for proper return of the interrupt vector.

System and Processor Clock Relationships



Bus Operation

The SAB 80286 uses a double frequency system clock (CLK input) to control bus timing. All signals on the local bus are measured relative to the system CLK input. The CPU divides the

system clock by 2 to produce the internal processor clock, which determines bus state. Each processor clock is composed of two system clock cycles named phase 1 and phase 2. The SAB 82284 clock generator output (PCLK) identifies the next phase of the processor clock (see figure on system and processor clock relationship).

Six types of bus operations are supported: memory read, memory write, I/O read, I/O write, interrupt acknowledge, and halt/shutdown. Data can be transferred at a maximum rate of one word per two processor clock cycles.

The SAB 80286 bus has three basic states: idle (TI), send status (TS), and perform command (TC). The SAB 80286 CPU also has a fourth local bus state called hold (TH). TH indicates that the SAB 80286 has surrendered control of the local bus to another bus master in response to a HOLD request.

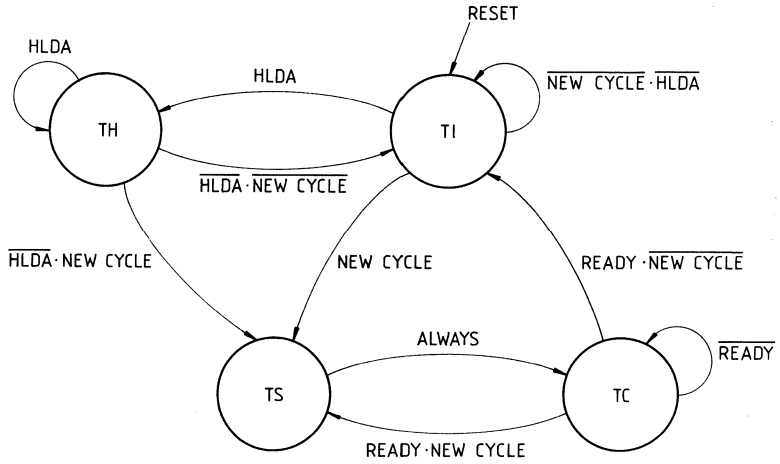
Each bus state is one processor clock long. The figure below shows the four SAB 80286 local bus states and allowed transitions.

Pipelined Addressing

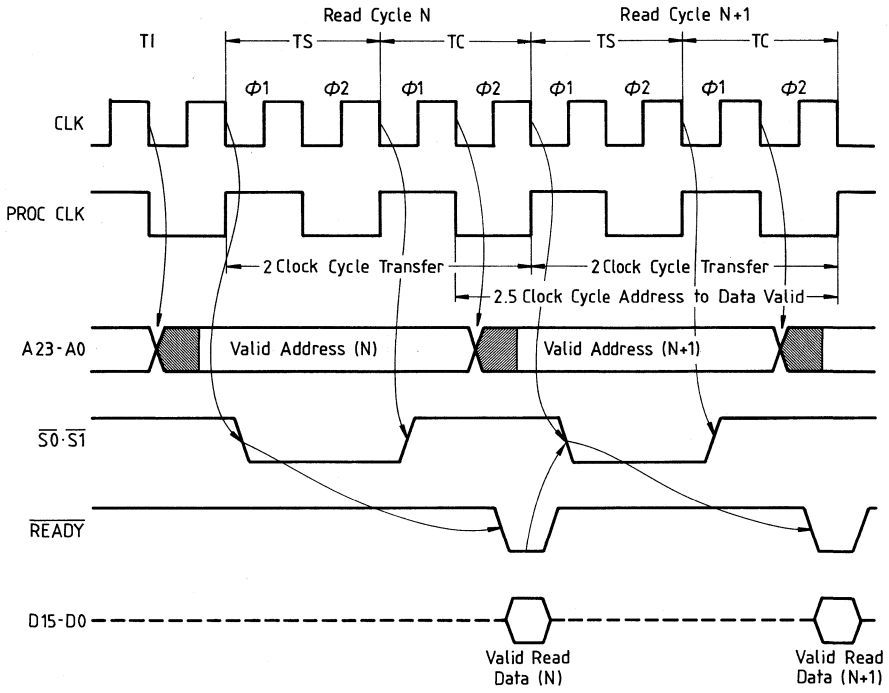
The SAB 80286 uses a local bus interface with pipelined timing to allow as much time as possible for data access. Pipelined timing allows bus operations to be performed in two processor cycles, while allowing each individual bus operation to last for three processor cycles. The timing of the address outputs is pipelined such that the address of the next bus operation becomes available during the current bus operation. Or in other words, the first clock of the next bus operation is overlapped with the last clock of the current bus operation. Therefore, address decoder and routing logic can operate in advance of the next bus operation. External address latches may hold the address stable for the entire bus operation, and provide additional ac and dc buffering.

The SAB 80286 does not maintain the address of the current bus operation during all TC states. Instead, the address for the next bus operation may be emitted during phase 2 of any TC. The address remains valid during phase 1 of the first TC to guarantee hold time, relative to ALE, for the address latch inputs.

SAB 80286 Bus States



Basic Bus Cycle



Bus Cycle Termination

At maximum transfer rates, the SAB 80286 bus alternates between the status and command states. The bus status signals become inactive after TS so that they may correctly signal the start of the next bus operation after the completion of the current cycle. No external indication of TC exists on the SAB 80286 local bus. The bus master and bus controller enter TC directly after TS, and continue executing TC cycles until terminated by $\overline{\text{READY}}$.

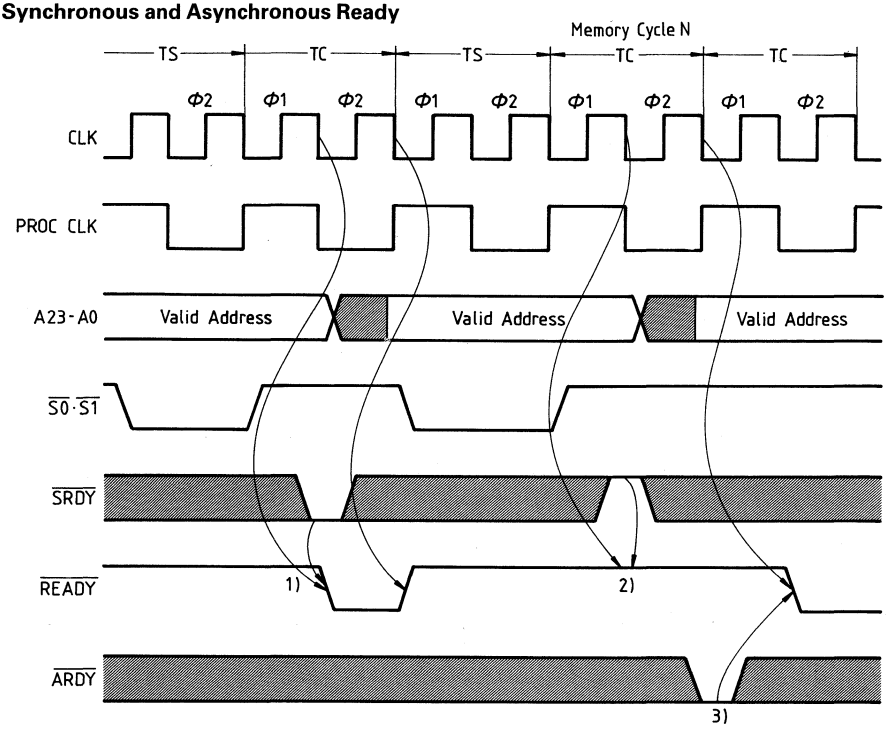
$\overline{\text{READY}}$ Operation

The current bus master and SAB 82288 bus controller terminate each bus operation simultaneously to achieve maximum bus bandwidth. Both are informed in advance by $\overline{\text{READY}}$ active which identifies the last TC cycle of the current bus operation. The bus master and bus controller must see the same sense of the $\overline{\text{READY}}$ signal, there by requiring $\overline{\text{READY}}$ be synchronous to the system clock.

Synchronous Ready

The SAB 82284 clock generator provides $\overline{\text{READY}}$ synchronization from both synchronous and asynchronous sources (see figure on next page). The synchronous ready input ($\overline{\text{SRDY}}$) of the clock generator is sampled with the falling edge of CLK at the end of phase 1 of each TC. The state of $\overline{\text{SRDY}}$ is then transferred to the bus master and bus controller via the $\overline{\text{READY}}$ output line.

Synchronous and Asynchronous Ready



1) $\overline{\text{SRDYEN}}$ is active low
2) If $\overline{\text{SRDYEN}}$ is high, the state of $\overline{\text{SRDY}}$ will not effect $\overline{\text{READY}}$
3) $\overline{\text{ARDYEN}}$ is active low

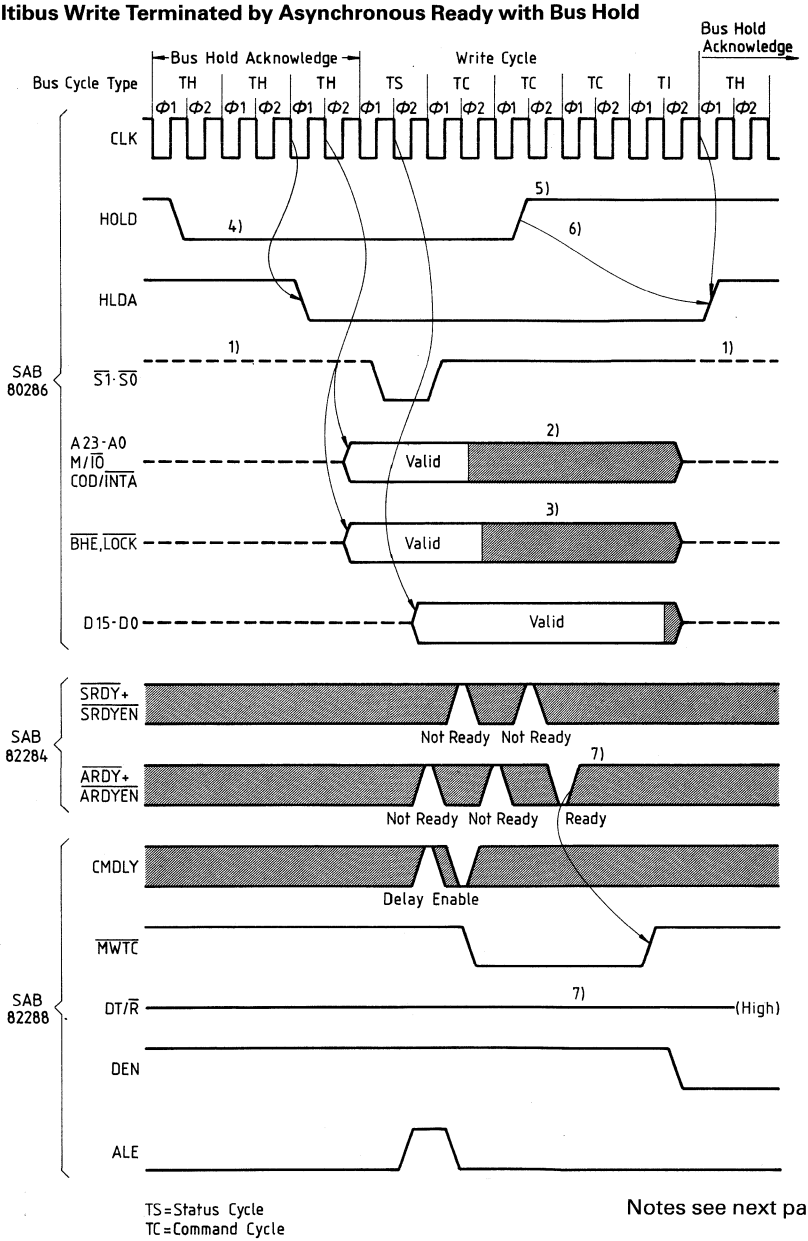
Asynchronous Ready

Many systems have devices of subsystems that are asynchronous to the system clock. As a result, their ready outputs cannot be guaranteed to meet the SAB 82284 $\overline{\text{SRDY}}$ setup and hold time requirements. But the SAB 82284 asynchronous ready input ($\overline{\text{ARDY}}$) is designed to accept such signals. The $\overline{\text{ARDY}}$ input is sampled at the beginning of each TC cycle by SAB 82284 synchronization logic. This provides one system CLK cycle time to resolve its value before transferring it to the bus master and bus controller.

$\overline{\text{ARDY}}$ or $\overline{\text{ARDYEN}}$ must be high at the end of TS. $\overline{\text{ARDY}}$ cannot be used to terminate a bus cycle with no wait states.

Each ready input of the SAB 82284 has an enable pin ($\overline{\text{SRDYEN}}$ and $\overline{\text{ARDYEN}}$) to select whether the current bus operation will be terminated by the synchronous or asynchronous ready. Either of the ready inputs may terminate a bus operation. These enable inputs are active low and have the same timing as their respective ready inputs. An address decode logic usually selects whether the current bus operation should be terminated by $\overline{\text{ARDY}}$ or $\overline{\text{SRDY}}$.

Multibus Write Terminated by Asynchronous Ready with Bus Hold



Notes see next page.

Notes for the figure on page 35:

1) Status lines are not driven by SAB 80286 yet remain high due to pullup resistors in SAB 80288 and SAB 82289 during HOLD state.
2) Address, M/$\overline{IO}$ and COD/$\overline{INTA}$ may start floating during any TC depending on when internal SAB 80286 bus arbiter decides to release bus to external HOLD. The float starts in ∅ 2 of TC.
3) $\overline{BHE}$ and $\overline{LOCK}$ may start floating after the end of any TC depending on when internal SAB 80286 bus arbiter decides to release bus to external HOLD. The float starts in ∅ 1 of TC.
4) The minimum HOLD to HLDA time is shown. Maximum is one TH longer.
5) The earliest HOLD time is shown. It will always allow a subsequent memory cycle if pending as shown.
6) The minimum HOLD in HLDA time is shown. Maximum is a function of the instruction, type of bus cycle and other machine status (i.e., interrupts, waits, lock, etc.)
7) Asynchronous ready allows termination of the cycle. Synchronous ready does not signal ready in this example. Synchronous ready state in ignored after ready is signalled via the asynchronous input.

HOLD and HLDA

HOLD and HLDA allow another bus master to gain control of the local bus by placing the SAB 80286 bus into the TH state. The sequence off events required to pass control between the SAB 80286 and another local bus master is shoen in the next figure.

HOLD must not be active during the time from the leading edge of RESET until 34 CLKs following the trailing edge of RESET unless the SAB 80286 is in the Halt condition until the processor Reset operation is complete, no interrupts should occur after the execution of HLT until 34 CLKs after the trailing edge of the RESET pulse.

Processor Extension Transfers

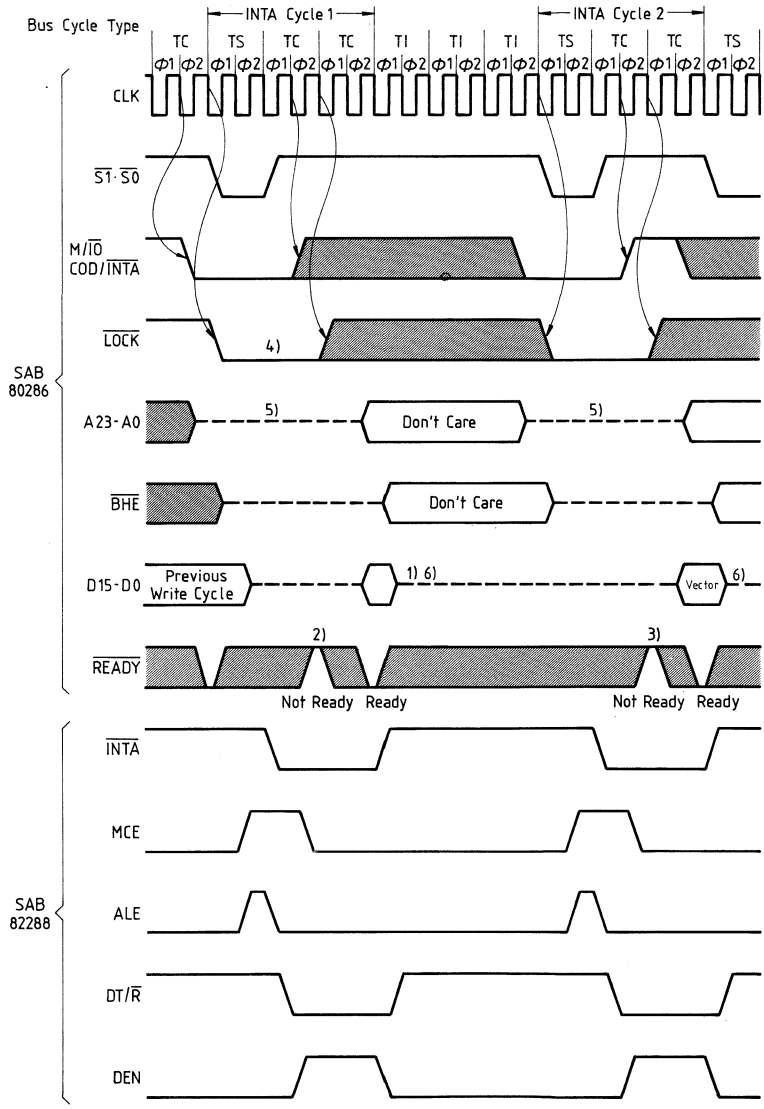
The processor extension interface uses I/O port addresses 00F8(H), 00FA(H), and 00FC(H) which are part of the I/O port address range reserved. An ESC instruction with EM = 0 and TS = 0 will perform I/O bus operations to one or more of these I/O port addresses independent of the value of IOPL and CPL.

ESC instructions with memory references enable the CPU to accept PEREQ inputs for processor extension operand transfers. The CPU will determine the operand starting address and read/write status of the instruction. For each operand transfer, two or three bus operations are performed, one word transfer with I/O port address 00FA(H) and one or two bus operations with memory. Three bus operations are required for each word operand aligned on an odd byte address.

Notes for the figure on page 37

1) Data is ignored.
2) First INTA cycles should have at least one wait state inserted to meet SAB 8259A minimum INTA pulse width.
3) Second INTA cycle must have at least one wait state inserted since the CPU will not drive A23–A0. $\overline{BHE}$, and $\overline{LOCK}$ until after the first TC state.
   The CPU-imposed one clock delay prevents bus contention between cascade address buffer being disabled by MCE↓ and address outputs. Without the wait state, the SAB 80286 address will not be valid for a memory cycle started immediately after the second INTA cycle.
   The SAB 8259A also requires one wait state for minimum INTA pulse width.
4) $\overline{LOCK}$ is activated during the INTA cycles to prevent the SAB 82289 from releasing the bus between INTA cycles in a multimaster system.
5) A23–A0 exit Tri-state OFF during ∅ 2 of the second TC in the INTA cycle.
6) D15–D8 should not change state during this time.

Interrupt Acknowledge Sequence



Interrupt Acknowledge Sequence

The figure Interrupt Acknowledge Sequence illustrates a sequence performed by the SAB 80286 in response to an INTR input. An interrupt acknowledge sequence consists of two INTA bus operations. The first allows a master SAB 8259A programmable interrupt controller (PIC) to determine which if any of its slaves should return the interrupt vector. An eight-bit vector is read by the SAB 80286 during the second INTA bus operation to select an interrupt handler routine from the interrupt table.

The master cascade enable (MCE) signal of the SAB 82288 is used to enable the cascade address drivers, during INTA bus operations (see interrupt acknowledge sequence figure), onto the local address bus for distribution to slave interrupt controllers via the system address bus. The SAB 80286 emits the $\overline{\text{LOCK}}$ signal (active low) during TS of both INTA bus operations. A local bus "hold" request will not be honored until the end of the second INTA bus operation.

Three idle processor clocks are provided by the SAB 80286 between INTA bus operations to allow for the minimum INTA to INTA time and CAS (cascade address) out delay of the SAB 8259A. The second INTA bus operation must always have at least one extra TC state added via logic controlling $\overline{\text{READY}}$. A23–A0 are in tristate off until the first TC state of the second INTA bus operation. This prevents bus contention between the cascade address drivers and CPU address drivers. The extra TC state provides time for the SAB 80286 to resume driving the address lines for subsequent bus operations.

Local Bus Usage Priorities

The SAB 80286 local bus is shared among several internal units and external HOLD requests. In case of simultaneous requests, their relative priorities are:

(Highest)
Any transfers which assert $\overline{\text{LOCK}}$ either explicitly (via the LOCK instruction prefix) or implicitly (i.e. segment descriptor access, interrupt acknowledge sequence, or an XCHG with memory).

The second of the two byte bus operations required for an odd-aligned word operand.

The second or third cycle of a processor extension data transfer.

Local bus request via HOLD input.

Processor extension data operand transfer via PEREQ input.

Data transfer performed by EU as part of an instruction.

(Lowest)
An instruction prefetch request from BU. The EU wil inhibit prefetching two processor clocks in advance of any data transfer to minimize waiting by EU for a prefetch to finish.

Halt or Shutdown Cycles

The SAB 80286 externally indicates halt or shutdown conditions as a bus operation. These conditions occur due to an HLT instruction or multiple protection exceptions while attempting to execute one instruction. A halt or shutdown bus operation is signalled when $\overline{\text{S1}}$, $\overline{\text{S0}}$ and COD/$\overline{\text{INTA}}$ are low and M/$\overline{\text{IO}}$ is high. A1 high indicates halt, and A1 low indicates shutdown. The SAB 82288 bus controller does not issue ALE, nor is $\overline{\text{READY}}$ required to terminate a halt or shutdown bus operation.

During halt or shutdown, the SAB 80286 may service PEREQ or HOLD requests. A processor extension segment overrun exception during shutdown will inhibit further service of PEREQ. Either NMI or RESET will force the SAB 80286 out of halt or shutdown. An INTR, if interrupts are enabled, or a processor extension segment overrun exception will also force the SAB 80286 out of halt.

System Configuration

The versatile bus structure of the SAB 80286 microsystem, with a full complement of support chips, allows flexible configuration of a wide range of systems. The basic configuration shown below is similar to an SAB 8086 maximum mode system. It includes the CPU plus an SAB 8259A interrupt controller, SAB 82284 clock generator and the SAB 82288 bus controller. The latches (SAB 8282A and SAB 8283A) and transceivers (SAB 8286A and SAB 8287A) used in an SAB 8086 system may be used in a SAB 80286 microsystem.

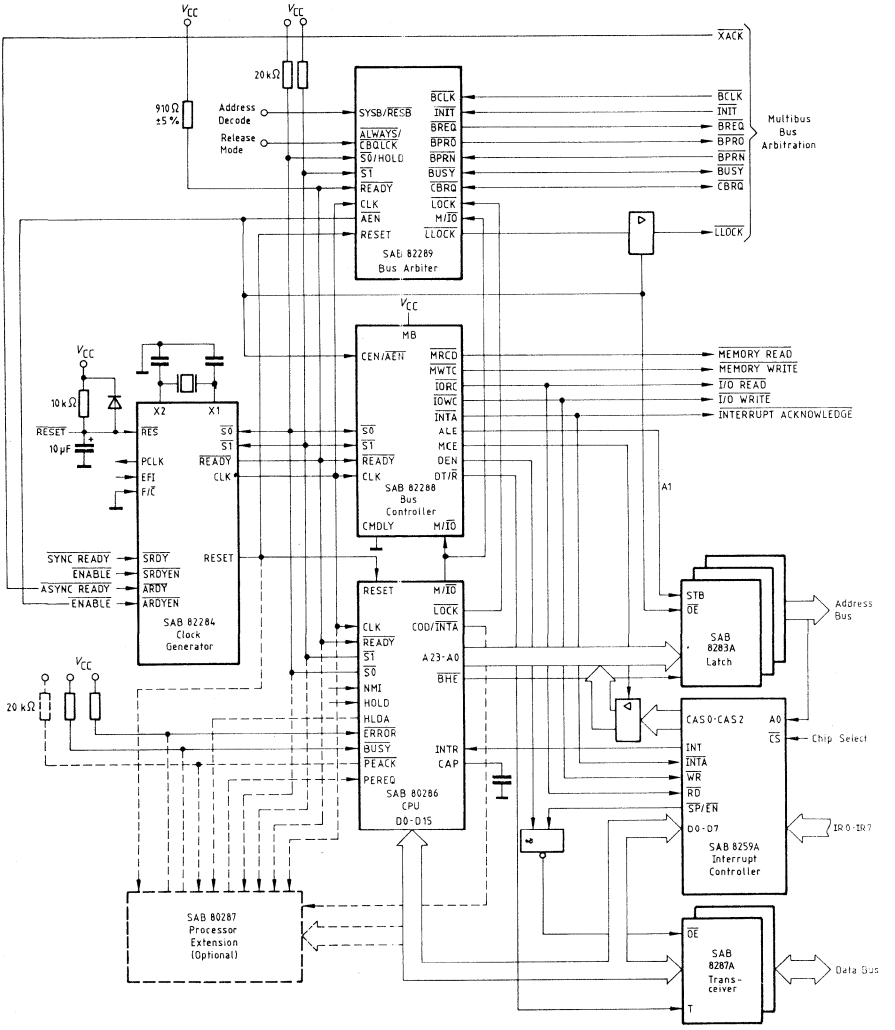
As indicated by the dashed lines in the figure above, the ability to add processor extensions is an integral feature of SAB 80286 microsystems. The processor extension interface allows external hardware to perform special functions and transfer data concurrently with CPU execution of other instructions. Full system integrity is maintained because the SAB 80826 supervises all data transfers and instruction execution for the processor extension.

The SAB 80287 numeric processor extension (NPX), for example, uses this interface. The SAB 80287 has all the instructions and data types of an SAB 8087. The SAB 80287 NPX can perform numeric calculations and data transfers concurrently with CPU program execution. Numerics code and data have the same integrity as all other information protected by the SAB 80286 protection mechanism.

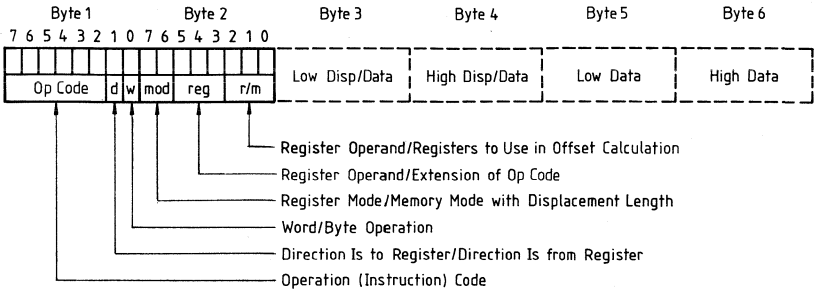
The SAB 80286 can overlap chip select decoding and address propagation during the data transfer for the previous bus operation. This information is latched into the SAB 8282A/8283A's by ALE in the middle of a TS cycle. The latched chip select and address information remains stable during the bus operation while the next cycles address is being decoded and propagated into the system. Decode logic can be implemented with a high-speed bipolar PROM.

The optional decode logic shown in the figure on basic system configuration takes advantage of the overlap between address and data of the SAB 80286 bus cycle to generate advance memory and IO-select signals. This minimizes system performance degradation caused by address propagation and decode delays. In addition to selection of memory and I/O, the advance selects may be used with configurations supporting local and system buses to enable the appropriate bus interface for each bus cycle. The COD/$\overline{\text{INTA}}$ and M/$\overline{\text{IO}}$ signals are applied to the decode logic to distinguish between interrupt, I/O, code, and data bus cycles. By adding the SAB 82289 bus arbiter chip, the SAB 80286 provides a Multibus system bus interface. A second SAB 82288 bus controller and additional latches and transceivers could be added to the local bus. This configuration allows the SAB 80286 to support an on-board bus for local memory and peripherals, and the Multibus for system bus interfacing.

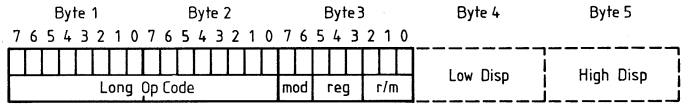
Basic SAB 80286 System Configuration



Instruction Format Examples



A. Short Op Code Format Example



B. Long Op Code Format Example

SAB 80286 Instruction Set Summary

Instruction Timing Notes

The instruction clock counts listed below establish the maximum execution rate of the SAB 80286. With no delays in bus cycles, the actual clock count of an SAB 80286 program will average 5% more than the calculated clock count, due to instruction sequences which execute faster than they can be fetched from memory.

To calculate elapsed times for instruction sequences multiply the sum of all instruction clock counts, as listed in the table below, by the processor clock period. An 8 MHz processor clock has a clock period of 125 nanoseconds and requires an SAB 80286 system clock (CLK input) of 16 MHz.

Instruction Clock Count Assumptions

1. The instruction has been prefetched, decoded, and is ready for execution. Control transfer instruction clock counts include all time required to fetch, decode, and prepare the next instruction for execution.
2. Bus cycles do not require wait states.
3. There are no processor extension data transfer or local bus HOLD requests.
4. No exceptions occur during instruction execution.

Instruction Set Summary Notes

Addressing displacements selected by the MOD field are not shown. If necessary they appear after the instruction fields shown.

Above/below refers to unsigned value

Greater refers to positive signed value

Less refers to less positive (more negative) signed values

if d = 1 then to register; if d = 0 then from register
if w = 1 then word instruction; if w = 0 then byte instruction
if s = 0 then 16-bit immediate data form the operand
if s = 1 then an immediate data byte is sign-extended to form the 16-bit operand
x don't care
Z used for string primitives for comparison with ZF FLAG

If two clock counts are given, the smaller refers to a register operand and the larger refers to a memory operand

* = add one clock if offset calculation requires summing 3 elements
n = number of times repeated
m = number of bytes of code in next instruction
Level(L) – Lexical nesting level of the procedure

The following comments describe possible exceptions, side effects, and allowed usage for instructions in both operating modes of the SAB 80286.

Real address mode only

1. This is a protected mode instruction. Attempted execution in real address mode will result in an undefined opcode exception (6).
2. A segment overrun exception (13) will occur if a word operand reference at offset FFFF(H) is attempted.
3. This instruction may be executed in real address mode to initialize the CPU for protected mode.
4. The IOPL and NT fields will remain 0.
5. Processor extension segment overrun interrupt (9) will occur if the operand exceeds the segment limit.

Either mode

6. An exception may occur, depending on the value of the operand.
7. $\overline{\text{LOCK}}$ is automatically asserted regardless of the presence or absence of the LOCK instruction prefix.
8. $\overline{\text{LOCK}}$ does not remain active between all operand transfers.

Protected virtual address mode only

9. A general protection exception (13) will occur if the memory operand can not be used due to either a segment limit or access rights violation. If a stack segment limit is violated, a stack segment overrun exception (12) occurs.
10. For segment load operations, the CPL, RPL and DPL must agree with privilege rules to avoid an exception. The segment must be present to avoid a not-present exception (11). If the SS register is the destination and a segment not-present violation occurs, a stack exception (12) occurs.
11. All segment descriptor accesses in the GDT or LDT made by this instruction will automatically assert $\overline{\text{LOCK}}$ to maintain descriptor integrity in multiprocessor systems.
12. JMP, CALL, INT, RET, IRET instructions referring to another code segment will cause a general protection exception (13) if any privilege rule is violated.
13. A general protection exception (13) occurs if CPL $\neq$ 0.
14. A general protection exception (13) occurs if CPL>IOPL.
15. The IF field of the flag word is not updated if CPL>IOPL. The IOPL field is updated only if CPL = 0.
16. Any violation of privilege rules as applied to the selector operand does not cause a protection exception; rather, the instruction does not return a result and the zero flag is cleared.
17. If the starting address of the memory operand violates a segment limit, or an invalid access is attempted, a general protection exception (13) will occur before the ESC instruction is executed. A stack segment overrun exception (12) will occur if the stack limit is violated by the operand's starting address. If a segment limit is violated during an attempted data transfer then a processor extension segment overrun exception (9) occurs.
18. The destination of an INT, JMP, CALL, RET or IRET instruction must be in the defined limit of a code segment or a general protection exception (13) will occur.

Instruction Set Summary

Function	Format	Clock count: Real address mode	Clock count: Protected virtual address mode	Comments: Real address mode	Comments: Protected virtual address mode
Data Transfer					
MOV = Move:					
Register to register/memory	1000100w | mod reg r/m	2, 3 *	2, 3 *	2	9
Register/memory to register	1000101w | mod reg r/m	2, 5 *	2, 5 *	2	9
Immediate to register/memory	1100011w | mod 000 r/m | data | data if w = 1	2, 3 *	2, 3 *	2	9
Immediate to register	1011w reg | data | data if w = 1	2	2		
Memory to accumulator	1010000w | addr-low | addr-high	5	5	2	9
Accumulator to memory	1010001w | addr-low | addr-high	3	3	2	9
Register/memory to segment register	10001110 | mod 0 reg r/m	2, 5 *	17, 19 *	2	9, 10, 11
Segment register to register/memory	10001100 | mod 0 reg r/m	2, 3 *	2, 3 *	2	9
PUSH = Push:					
Memory	11111111 | mod 110 r/m	5 *	5 *	2	9
Register	01010 reg	3	3	2	9
Segment register	000 reg 110	3	3	2	9
Immediate	011010s0 | data | data if s = 0	3	3	2	9
PUSHA = Push All	01100000	17	17	2	9
POP = Pop:					
Memory	10001111 | mod 000 r/m	5 *	5 *	2	9
Register	01011 reg	5	5	2	9
Segment register	000 reg 111 (reg ≠ 01)	5	20	2	9, 10, 11

Shaded areas indicate instructions not available in SAB 8086, 8088 microsystems.

Function	Format	Clock count: Real address mode	Clock count: Protected virtual address mode	Comments: Real address mode	Comments: Protected virtual address mode
Data Transfer (cont'd)					
POPA = Pop All	01100001	19	19	2	9
XCHG = **Exchange:**					
Register/memory with register	1000011w mod reg r/m	3, 5 *	3, 5 *	2, 7	7, 9
Register with accumulator	10010 reg	3	3		
IN = **Input from:**					
Fixed port	1110010w port	5	5		14
Variable port	1110110w	5	5		14
OUT = **Output to:**					
Fixed port	1110011w port	3	3		14
Variable port	1110111w	3	3		14
XLAT = Translate byte to AL	11010111	5	5		9
LEA = Load EA to register	10001101 mod reg r/m	3 *	3 *		
LDS = Load pointer to DS	11000101 mod reg r/m (mod ≠ 11)	7 *	21 *	2	9, 10, 11
LES = Load pointer to ES	11000100 mod reg r/m (mod ≠ 11)	7 *	21 *	2	9, 10, 11
LAHF = Load AH with flags	10011111	2	2		
SAHF = Store AH into flags	10011110	2	2		
PUSHF = Push flags	10011100	3	3	2	9
POPF = Pop flags	10011101	5	5	2, 4	2, 4

Shaded areas indicate instructions not available in SAB 8086, 8088 microsystems.

Function	Format	Clock count: Real address mode	Clock count: Protected virtual address mode	Comments: Real address mode	Comments: Protected virtual address mode
Arithmetic					
ADD = Add:					
Reg/memory with register to either	000000dw | mod reg. r/m	2, 7 *	2, 7*	2	9
Immediate to register memory	100000sw | mod 000 r/m | data | data if s w = 01	3, 7 *	3, 7 *	2	9
Immediate to accumulator	0000010w | data | data if w = 1	3	3		
ADC = Add with carry:					
Reg memory with register to either	000100dw | mod reg r/m	2, 7 *	2, 7 *	2	9
Immediate to register/memory	100000sw | mod 010 r/m | data | data if s w = 01	3, 7 *	3, 7 *	2	9
Immediate to accumulator	0001010w | data | data if w = 1	3	3		
INC = Increment					
Register memory	1111111w | mod 000 r/m	2, 7 *	2, 7 *	2	9
Register	01000 reg	2	2		
SUB = Subtract					
Reg memory and register to either	001010dw | mod reg r/m	2, 7 *	2, 7 *	2	9
Immediate from register memory	100000sw | mod 101 r/m | data | data if s w = 01	3, 7 *	3, 7 *	2	9
Immediate from accumulator	0010110w | data | data if w = 1	3	3		
SSB = Subtract with borrow:					
Reg/memory and register to either	000110dw | mod reg r/m	2, 7 *	2, 7 *	2	9
Immediate from register/memory	100000sw | mod 011 r/m | data | data if s w = 01	3, 7 *	3, 7 *	2	9
Immediate from accumulator	0001110w | data | data if w = 1	3	3		

Shaded areas indicate instructions not available in SAB 8086, 8088 microsystems.

Function	Format	Clock count: Real address mode	Clock count: Protected virtual address mode	Comments: Real address mode	Comments: Protected virtual address mode
Arithmetic (cont'd):					
DEC = **Decrement:**					
Register memory	1 1 1 1 1 1 1 w | mod 0 0 1 r/m	2, 7 *	2, 7 *	2	9
Register	0 1 0 0 1 reg	2	2		
CMP = **Compare:**					
Register memory with register	0 0 1 1 1 0 1 w | mod reg r/m	2, 6 *	2, 6 *	2	9
Register with register/memory	0 0 1 1 1 0 0 w | mod reg r/m	2, 7 *	2, 7 *	2	9
Immediate with register memory	1 0 0 0 0 0 s w | mod 1 1 1 r/m | data | data if s w=01	3, 6 *	3, 6 *	2	9
Immediate with accumulator	0 0 1 1 1 1 0 w | data | data if w = 1	3	3		
NEG = Change sign	1 1 1 1 0 1 1 w | mod 0 1 1 r/m	2	7 *	2	7
AAA = ASCII adjust for add	0 0 1 1 0 1 1 1	3	3		
DAA = Decimal adjust for add	0 0 1 0 0 1 1 1	3	3		
AAS = ASCII adjust for subtract	0 0 1 1 1 1 1 1	3	3		
DAS = Decimal adjust for substract	0 0 1 0 1 1 1 1	3	3		
MUL = Multiply (unsigned):	1 1 1 1 0 1 1 w | mod 1 0 0 r/m				
register-byte		13	13		
register-word		21	21		
memory-byte		16 *	16 *	2	9
menory-word		24 *	24 *	2	9
IMUL = Integer multiply (signed):	1 1 1 1 0 1 1 w | mod 1 0 1 r/m				
register-byte		13	13		
register-word		21	21		
memory-byte		16 *	16 *	2	9
memory-word		24 *	24 *	2	9

Shaded areas indicate instructions not available in SAB 8086, 8088 microsystems.

Function	Format	Clock count: Real address mode	Clock count: Protected virtual address mode	Comments: Real address mode	Comments: Protected virtual address mode
Arithmetic (cont'd):					
IMUL = Integer immediate multiply (signed)	0 1 1 0 1 0 s 1 | mod reg r/m | data | data if s = 0	21, 24 *	21, 24 *	2	9
DIV = Divide unsigned):	1 1 1 1 0 1 1 w | mod 1 1 0 r/m				
register-byte		14	14	6	6
register-word		22	22	6	6
memory-byte		17 *	17 *	2, 6	6, 9
memory-word		25 *	25 *	2, 6	6, 9
IDIV = Integer divide (signed):	1 1 1 1 0 1 1 w | mod 1 1 1 r/m				
register-byte		17	17	6	6
register-word		25	25	6	6
memory-byte		20 *	20 *	2, 6	6, 9
memory-word		28 *	28 *	2, 6	6, 9
AMM = ASCII adjust for multiply	1 1 0 1 0 1 0 0 | 0 0 0 0 1 0 1 0	16	16		
AAD = ASCII adjust for divide	1 1 0 1 0 1 0 1 | 0 0 0 0 1 0 1 0	14	14		
CBW = Convert byte to word	1 0 0 1 1 0 0 0	2	2		
CWD = Convert word to double word	1 0 0 1 1 0 0 1	2	2		
Logic Shift/rotate instructions:					
Register/memory by 1	1 1 0 1 0 0 0 w | mod TTT r/m	2, 7 *	2, 7 *	2	9
Register/memory by CL	1 1 0 1 0 0 1 w | mod TTT r/m	5+n, 8+n*	5+n, 8+n*	2	9
Register/memory by count	1 1 0 0 0 0 0 w | mod TTT r/m | count	5+n, 8+n*	5+n, 8+n*	2	9

TTT	Instruction
0 0 0	ROL
0 0 1	ROR
0 1 0	RCL
0 1 1	RCR
1 0 0	SHL/SAL
1 0 1	SHR
1 1 1	SAR

Shaded areas indicate instructions not available in SAB 8086, 8088 microsystems.

Function	Format	Clock count: Real address mode	Clock count: Protected virtual address mode	Comments: Real address mode	Comments: Protected virtual address mode
Logic (cont'd):					
AND = **And:**					
Reg/memory and register to either	0 0 1 0 0 0 d w | mod reg r/m	2, 7 *	2, 7 *	2	9
Immediate to register/memory	1 0 0 0 0 0 0 w | mod 1 0 0 r/m | data | data if w = 1	3, 7 *	3, 7 *	2	9
Immediate to accumulator	0 0 1 0 0 1 0 w | data | data f w = 1	3	3		
TEST = **And function to flags, no result:**					
Register/memory and register	1 0 0 0 0 1 0 w | mod reg r/m	2, 6 *	2, 6 *	2	9
Immediate data and register/memory	1 1 1 1 0 1 1 w | mod 0 0 0 r/m | data | data if w = 1	3, 6 *	3, 6 *	2	9
Immediate data and accumulator	1 0 1 0 1 0 0 w | data | data if w = 1	3	3		
OR = **Or:**					
Reg/memory and register to either	0 0 0 0 1 0 d w | mod reg r/m	2, 7 *	2, 7 *	2	9
Immediate to register/memory	1 0 0 0 0 0 0 w | mod 0 0 1 r/m | data | data if w = 1	3, 7 *	3, 7 *	2	9
Immediate to accumulator	0 0 0 0 1 1 0 w | data | data if w = 1	3	3		
XOR = **Exclusive or:**					
Reg/memory and register to either	0 0 1 1 0 0 d w | mod reg r/m	2, 7 *	2, 7 *	2	9
Immediate to register/memory	1 0 0 0 0 0 0 w | mod 1 1 0 r/m | data | data if w = 1	3, 7 *	3, 7 *	2	9
Immediate to accumulator	0 0 1 1 0 1 0 w | data | data if w = 1	3	3		
NOT = Invert register/memory	1 1 1 1 0 1 1 w | mod 0 1 0 r/m	2, 7 *	2, 7 *	2	9

Shaded areas indicate instructions not available in SAB 8086, 8088 microsystems.

Function	Format	Clock count: Real address mode	Clock count: Protected virtual address mode	Comments: Real address mode	Comments: Protected virtual address mode
String Manipulation:					
MOVS = Move byte word	1010010w	5	5	2	9
CMPS = Compare byte/word	1010011w	8	8	2	9
SCAS = Scan byte/word	1010111w	7	7	2	9
LODS = Load byte/wd to AL/AX	1010110w	5	5	2	9
STOS = Store byte/wd from AL/A	1010101w	3	3	2	9
INS = Input byte/wd from DX port	0110110w	5	5	2	9, 14
OUTS = Output byte/wd to DX port	0110111w	5	5	2	9, 14
Repeated by count in CX					
MOVS = Move string	11110011 1010010w	5 + 4n	5 + 4n	2	9
CMPS = Compare string	1111001z 1010011w	5 + 9n	5 + 9n	2, 8	8, 9
SCAS = Scan string	1111001z 1010111w	5 + 8n	5 + 8n	2, 8	8, 9
LODS = Load string	11110011 1010110w	5 + 4n	5 + 4n	2, 8	8, 9
STOS = Store string	11110011 1010101w	4 + 3n	4 + 3n	2, 8	8, 9
INS = Input string	11110011 0110110w	5 + 4n	5 + 4n	2	9, 14
OUTS = Output string	11110011 0110111w	5 + 4n	5 + 4n	2	9, 14

Shaded areas indicate instructions not available in SAB 8086, 8088 microsystems.

Function	Format	Clock count: Real address mode	Clock count: Protected virtual address mode	Comments: Real address mode	Comments: Protected virtual address mode
Control Transfer:					
CALL = Call:					
Direct within segment	11101000 | disp-low | disp-high	7 + m	7 + m	2	18
Register/memory indirect within segment	11111111 | mod 0 1 0 r/m	7+m,11+m*	7+m,11+m*	2, 8	8, 9, 18
Direct intersegment	10011010 | segment offset | segment selector	13 + m	26 + m	2	11,12,18
Protected mode only (direct intersegment):					
Via call gate to same privilege level			41 + m		8,11,12,18
Via call gate to different privilege level, no parameters			82+m		8,11,12,18
Via call gate to different privilege level, x parameters			86+4x+m		8,11,12,18
Via TSS			177+m		8,11,12,18
Via task gate			182+m		8,11,12,18
Indirect intersegment	11111111 | mod 0 1 1 r/m (mod ≠ 11)	16+n	29+n*	2	8,9,11,12,18
Protected mode only (indirect intersegment):					
Via call gate to same privileg level			44+m*		8,9,11,12,18
Via call gate different privilege level, no parameters			83+m*		8,9,11,12,13
Via call gate to different privilege level, x parameters			90+4x+m*		8,9,11,12,18
Via TSS			180+m*		8,9,11,12,18
Via task gate			185+m*		8,9,11,12,18

Shaded areas indicate instructions not available in SAB 8086, 8088 microsystems.

Function	Format	Clock count: Real address mode	Clock count: Protected virtual address mode	Comments: Real address mode	Comments: Protected virtual address mode
Control Transfer (cont'd):					
JMP = Unconditional jump:					
Short/long	11101011 | disp-low	7+m	7+m		18
Direct within segment	11101001 | disp-low | disp-high	7+m	7+m		18
Register/memory indirect within segment	11111111 | mod 100 r/m	7+m,11+m*	7+m,11+m*	2	9, 18
Direct intersegment	11101010 | segment offset segment selector	11+m	23+m		11,12,18
Protected mode only (direct intersegment):					
Via call gate to same privilege level			38+m		8,11,12,18
Via TSS			175+m		8,11,12,18
Via task gate			180+m		8,11,12,18
Indirect intersegment	11111111 | mod 101 r/m (mod ≠ 11)	15+m*	26+m*	2	8,9,11,12,18
Protected mode only (indirect intersegment):					
Via call gate to same privilege level			41+m*		8,9,11,12,18
Via TSS			178+m*		8,9,11,12,18
Via task gate			183+m*		8,9,11,12,18
RET = Return from CALL:					
Within segment	11000011	11+m	11+m	2	8,9,18
Within segment adding immediate to SP	11000010 | data-low | data-high	11+m	11+m	2	8,9,18
Intersegment	11001011	15+m	25+m	2	8,9,11,12,18
Intersegment adding immediate to SP	11001010 | data-low | data-high	15+m		2	8,9,11,12,18
Protected mode only (RET):					
To different privilege level			55+m		9,11,12,18

Shaded areas indicate instructions not available in SAB 8086, 8088 microsystems.

Function	Format	Clock count: Real address mode	Clock count: Protected virtual address mode	Comments: Real address mode	Comments: Protected virtual address mode
Control Transfer (cont'd):					
JE/JZ = Jump on equal/zero	01110100 disp	7+m or 3	7+m or 3		18
JL/JNGE = Jump on less/not greater equal	01111100 disp	7+m or 3	7+m or 3		18
JLE/JNG = Jump on less or equal/not greater	01111110 disp	7+m or 3	7+m or 3		18
JB/JNAE = Jump on below/not above or equal	01110010 disp	7+m or 3	7+m or 3		18
JBE/JNA = Jump on below or equal/not above	01110110 disp	7+m or 3	7+m or 3		18
JP/JPE = Jump on parity/parity even	01111010 disp	7+m or 3	7+m or 3		18
JO = Jump on overflow	01110000 disp	7+m or 3	7+m or 3		18
JS = Jump on sign	01111000 disp	7+m or 3	7+m or 3		18
JNE/JNZ = Jump on not equal/not zero	01110101 disp	7+m or 3	7+m or 3		18
JNL/JGE = Jump on not less/greater or equal	01111101 disp	7+m or 3	7+m or 3		18
JNLE/JG = Jump on not less or equal/greater	01111111 disp	7+m or 3	7+m or 3		18
JNB/JAE = Jump on not below/above or equal	01110011 disp	7+m or 3	7+m or 3		18
JNBE/JA = Jump on not below or equal/above	01110111 disp	7+m or 3	7+m or 3		18
JNP/JPO = Jump on not par/par odd	01111011 disp	7+m or 3	7+m or 3		18
JNO = Jump on not overflow	01110001 disp	7+m or 3	7+m or 3		18
JNS = Jump on not sign	01111001 disp	7+m or 3	7+m or 3		18
LOOP = Loop CX times	11100010 disp	8+m or 4	8+m or 4		18
LOOPZ/LOOPE = Loop while zero/equal	11100001 disp	8+m or 4	8+m or 4		18
LOOPNZ/LOOPNE = Loop while not zero/equal	11100000 disp	8+m or 4	8+m or 4		18
JCXZ = Jump on CX zero	11100011 disp	8+m or 4	8+m or 4		18

Shaded areas indicate instructions not available in SAB 8086, 8088 microsystems.

Function	Format	Clock count: Real address mode	Clock count: Protected virtual address mode	Comments: Real address mode	Comments: Protected virtual address mode
Control Transfer (cont'd):					
ENTER = Enter procedure	11001000 | data-low | data-high | L			2, 8	8, 9
L = 0		11	11	2, 8	8, 9
L = 1		15	15	2, 8	8, 9
L > 1		16+4(L-1)	16+4(L-1)	2, 8	8, 9
LEAVE = Leave procedure	11001001	5	5	2, 8	8, 9
INT = **Interrupt:**					
Type specified	11001101 | type	23+m		2, 7, 8	
Type 3	11001100	23+m		2, 7, 8	
INTO = Interrupt on overflow	11001110	24+m or 3 (3 if no interrupt)	(3 if no interrupt)	2, 6, 8	
Protected mode only:					
Via interrupt or trap gate to same privilege level			40+m		7,8,11,12,18
Via interrupt or trap gate to fit different privilege level			78+m		7,8,11,12,18
Via Task Gate			167+m		7,8,11,12,18
IRET = Interrupt return	11001111	17+m	31+m	2, 4	8,9,11,12,15,18
Protected mode only:					
To different privilege level			55+m		8,9,11,12,15,18
To different task (NT = 1)			169+m		8,9,11,12,18
BOUND = Detect value out of range	01100010 | mod reg r/m	13 *	13 * (Use INT clock count if exception 5)	2, 6	6,8,9,11,12,18

Shaded areas indicate instructions not available in SAB 8086, 8088 microsystems.

Function	Format	Clock count: Real address mode	Clock count: Protected virtual address mode	Comments: Real address mode	Comments: Protected virtual address mode
Processor Control					
CLC = Clear carry	11111000	2	2		
CMC = Complement carry	11110101	2	2		
STC = Set carry	11111001	2	2		
CLD = Clear direction	11111100	2	2		
STD = Set direction	11111101	2	2		
CLI = Clear interrupt	11111010	3	3		14
STI = Set interrupt	11111011	2	2		14
HLT = Halt	11110100	2	2		13
WAIT = Wait	10011011	3	3		
LOCK = Bus lock prefix	11110000	0	0		14
CTS = Clear task switched flag	00001111 00000110	2	2	3	13
ESC = Processor extension escape	11011TTT mod LLL r/m (TTT LLL are opcode to processor extension)	9-20 *	9-20 *	5, 8	8, 17
SEG = Segment override prefix	001 reg 110	0	0		

Shaded areas indicate instructions not available in SAB 8086, 8088 microsystems.

Function	Format	Clock count		Comments	
		Real address mode	Protected virtual address mode	Real address mode	Protected virtual address mode
Protection Control					
LGDT = Load global descriptor table register	00001111 | 00000001 | mod 010 r/m	11 *	11 *	2, 3	9, 13
SGDT = Store global descriptor table register	00001111 | 00000001 | mod 000 r/m	11 *	11 *	2, 3	9
LIDT = Load interrupt descriptor table register	00001111 | 00000001 | mod 011 r/m	12 *	12 *	2, 3	9, 13
SIDT = Store interrupt descriptor table register	00001111 | 00000001 | mod 001 r/m	12 *	12*	2, 3	9
LLDT = Load local descriptor table register from register memory	00001111 | 00000000 | mod 010 r/m		17, 19 *	1	9, 11, 13
SLDT = Store local descriptor table register to register/memory	00001111 | 00000000 | mod 000 r/m		2, 3 *	1	9
LTR = Load task register from register/memory	00001111 | 00000000 | mod 011 r/m		17, 19 *	1	9, 11, 13
STR = Store task register to register memory	00001111 | 00000000 | mod 001 r/m		2, 3*	1	9
LMSW = Load machine status word from register/memory	00001111 | 00000001 | mod 110 r/m	3, 6 *	3, 6 *	2,3	9, 13
SMSW = Store machine status word	00001111 | 00000001 | mod 100 r/m	2, 3 *	2, 3 *	2, 3	9
LAR = Load access rights from register/memory	00001111 | 00000010 | mod reg r/m		14, 16 *	1	9, 11, 16
LSL = Load segment limit from register/memory	00001111 | 00000011 | mod reg r/m		14, 16 *	1	9, 11, 16
ARPL = Adjust requested privilege level: from register/memory	01100011 | mod reg r/m		10*, 11*	2	8, 9
VERR = Verify read access: register/memory	00001111 | 00000000 | mod 100 r/m		14, 16 *	1	9, 11, 16
VERW = Verify write access:	00001111 | 00000000 | mod 101 r/m		14, 16*	1	9, 11, 16

Shaded areas indicate instructions not available in SAB 8086, 8088 microsystems.

Notes:

The effective address (EA) of the memory operand is computed according to the mod and r/m fields:

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low

if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP

DISP follows 2nd byte of instruction (before data if required)

REG is assigned according to the following table:

16-bit (w = 1)	8-bit (w = 0)	16-bit (w = 1)	8-bit (w = 0)
000 AX	000 AL	100 SP	100 AH
001 CX	001 CL	101 BP	101 CH
010 DX	010 DL	110 SI	110 DH
011 BX	011 BL	111 DI	111 DI

The physical addresses of all operands addressed by the BP register are computed using the SS segment register. The physical addresses of the destination operands of the string primitive operations (those addressed by the DI register) are computed using the ES segment, which may not be overridden.

* except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.

segment override prefix

0	0	1	reg	1	1	0

reg is assigned according to the following:

reg	Segment register
00	ES
01	CS
10	SS
11	DS

Table 13
SAB 80286 Systems, Recommended Pullup Resistor Values

SAB 80286 Pin and name	Pullup value	Purpose
4-$\overline{S1}$	20 kΩ ± 10%	Pull $\overline{S0}$, $\overline{S1}$, and $\overline{PEACK}$ inactive during SAB 80286 hold periods
5-$\overline{S0}$		
6-$\overline{PEACK}$		
53-$\overline{ERROR}$	20 kΩ ± 10%	Pull $\overline{ERROR}$ and $\overline{BUSY}$ inactive when SAB 80287 not present (or temporarily removed from socket)
54-$\overline{BUSY}$		
63-$\overline{READY}$	910 Ω ± 5%	Pull $\overline{READY}$ inactive within required minimum time CL = 150 pF)

Absolute Maximum Ratings

Temperature under bias . . . . . (T_C) 0 to 100°C
Storage temperature . . . . . −65 to +150°C
Voltage on any pin with respect to ground . . . . . −0.5 to +7V
Power dissipation 3.3W

Note: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

T_C = 0 to 85°C; V_{CC} = +5V ±5%

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
Input low voltage	V_{IL}	−0.5	+0.8	V	–
Input high voltage	V_{IH}	2.0	V_{CC}+0.5	V	–
Clock input low voltage	V_{CL}	−0.5	+0.6	V	–
Clock input high voltage	V_{CH}	3.8	V_{CC}+0.5	V	–
Output low voltage	V_{OL}	–	0.45	V	I_{OL} = 2.0 mA
Output high voltage	V_{OH}	2.4	–	V	I_{OH} = −400 µA
Power supply current	I_{CC}	–	600	mA	T_C = 0°C (turn on) [1]
Input leakage current	I_{LI}	–	±10	µA	0 V ≤ V_{IN} ≤ V_{CC}
CLK input leakage current	I_{LCR}	–	±10	µA	0.45 V ≤ V_{IN} ≤ V_{CC}
CLK input leakage current	I_{LCR}	–	±1	mA	0 V ≤ V_{IN} < 0.45 V
Input sustaining current on $\overline{\text{BUSY}}$ and $\overline{\text{ERROR}}$	I_{IL}	30	500	µA	V_{IN} = 0 V
Output leakage current	I_{LO}	–	±10	µA	0.45 V ≤ V_{OUT} ≤ V_{CC}
Output leakage current	I_{LO}	–	±1	mA	0 V ≤ V_{OUT} < 0.45 V
Capacitance of inputs (all input exept CLK)	C_{IN}	–	10	pF	f_C = 1 MHz [2]
Capacitance of I/O or outputs	C_{IO}	–	20	pF	f_C = 1 MHz [2]
Capacitance of CLK, $\overline{\text{READY}}$, $\overline{\text{BUSY}}$, $\overline{\text{ERROR}}$, and RESET inputs	C_{CLK}	–	20	pF	f_C = 1 MHz [2]

[1] Low temperature is worst case.

[2] Not 100% tested, guaranteed by design characterization.

AC Characteristics SAB 80286

AC timings are referenced to 0.8 V and 2.0 V points of signals as illustrated in data sheet waveforms, unless otherwise noted.

T_C = 0 to 85°C; V_{CC} = +5V ±5%

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
System clock (CLK) period	t_1	62	250	ns	–
System clock (CLK) low time	t_2	15	225	ns	at 1.0 V
System clock (CLK) high time	t_3	25	235	ns	at 3.6 V
System clock (CLK) rise time	t_{17}	–	10	ns	1.0 V to 3.6 V
System clock (CLK) fall time	t_{18}	–	10	ns	3.6 V to 1.0 V
Async inputs, setup time	t_4	20	–	ns	1)
Async inputs, hold time	t_5	20	–	ns	1)
RESET setup time	t_6	28	–	ns	–
RESET hold time	t_7	5	–	ns	–
Read data setup time	t_8	10	–	ns	–
Read data hold time	t_9	8	–	ns	–
$\overline{\text{READY}}$ setup time	t_{10}	38	–	ns	–
$\overline{\text{READY}}$ hold time	t_{11}	25	–	ns	–
Status/$\overline{\text{PEACK}}$ active delay	t_{12a}	1	40	ns	2) 3)
Status/$\overline{\text{PEACK}}$ inactive delay	t_{12b}	1	40	ns	2) 3)
Address valid delay	t_{13}	1	60	ns	2) 3)
Write data valid delay	t_{14}	0	50	ns	2) 3)
Address/status/data float delay	t_{15}	0	50	ns	2) 4)
HLDA valid delay	t_{16}	0	50	ns	2) 3)
Address valid to status valid setup time	t_{19}	38	–	ns	3) 5) 6)

1) Asynchronous inputs are INTR, NMI, HOLD, PEREQ, $\overline{\text{ERROR}}$, and $\overline{\text{BUSY}}$. The specification is given only for testing purposes, to assure recognition at a specific CLK edge.

2) Delay from 1.0 V on the CLK to 0.8 V or 2.0 V or float on the output as appropriate for valid or floating condition.

3) Output load C_L = 100 pF

4) Float condition occurs when output current is less than I_{LO} in magnitude.

5) Delay measured from address either reaching 0.8 V or 2.0 V (valid) to status going active reaching 2.0 V or status going inactive reaching 0.8 V.

6) For a load capacitance of 10 pF on status/$\overline{\text{PEACK}}$ lines, subtract typically 7 ns.

SAB 82284 Timing Requirements

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
$\overline{\text{SRDY}}/\overline{\text{SRDYEN}}$ setup time	t_{11}	15	–	ns	–
$\overline{\text{SRDY}}/\overline{\text{SRDYEN}}$ hold time	t_{12}	0	–	ns	–
$\overline{\text{ARDY}}/\overline{\text{ARDYEN}}$ setup time	t_{13}	0	–	ns	1)
$\overline{\text{ARDY}}/\overline{\text{ARDYEN}}$ hold time	t_{14}	30	–	ns	1)
PCLK delay	t_{19}	0	45	ns	C_L = 75 pF I_{OL} = 5 mA I_{OH} = −1 mA

SAB 82288 Timing Requirements

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
CMDLY setup time	t_{12}	20	–	ns	–
CMDLY hold time	t_{13}	0	–	ns	–
Command inactive delay	t_{30}	3	25	ns	2)
Command active delay	t_{29}	3	25	ns	2)
ALE active delay	t_{16}	3	20	ns	3)
ALE inactive delay	t_{17}	–	20	ns	3)
DT/$\overline{\text{R}}$ read active delay	t_{19}	–	25	ns	3)
DT/$\overline{\text{R}}$ read inactive delay	t_{22}	5	35	ns	3)
DEN read active delay	t_{20}	5	35	ns	3)
DEN read inactive delay	t_{21}	3	35	ns	3)
DEN write active delay	t_{23}	–	30	ns	3)
DEN write inactive delay	t_{24}	3	30	ns	3)

1) These times are given for testing purposes to assure a predetermined action.

2) C_L = 300 pF max.
I_{OL} = 32 mA max.
I_{OH} = −5 mA max.

3) C_L = 150 pF
I_{OL} = 16 mA max.
I_{OH} = −1 mA max.

AC Characteristics SAB 80286-1

AC timings are referenced to 0.8 V and 2.0 V points of signals as illustrated in data sheet waveforms, unless otherwise noted.

T_C = 0 to 85°C; V_{CC} = +5 V ± 5%

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
System clock (CLK) period	t_1	50	250	ns	–
System clock (CLK) low time	t_2	12	234	ns	at 1.0 V
System clock (CLK) high time	t_3	16	238	ns	at 3.6 V
System clock (CLK) rise time	t_{17}	–	8	ns	1.0 V to 3.6 V
System clock (CLK) fall time	t_{18}	–	8	ns	3.6 V to 1.0 V
Async inputs, setup time	t_4	20	–	ns	1)
Async inputs, hold time	t_5	20	–	ns	1)
RESET setup time	t_6	23	–	ns	–
RESET hold time	t_7	5	–	ns	–
Read data setup time	t_8	8	–	ns	–
Read data hold time	t_9	8	–	ns	–
$\overline{\text{READY}}$ setup time	t_{10}	26	–	ns	–
$\overline{\text{READY}}$ hold time	t_{11}	25	–	ns	–
Status/$\overline{\text{PEACK}}$ active delay	t_{12a}	1	22	ns	2) 3)
Status/$\overline{\text{PEACK}}$ inactive delay	t_{12b}	1	30	ns	2) 3)
Address valid delay	t_{13}	1	35	ns	2) 3)
Write data valid delay	t_{14}	0	30	ns	2) 3)
Address/status/data float delay	t_{15}	0	47	ns	2) 4)
HLDA valid delay	t_{16}	0	47	ns	2) 3)
Address valid to status valid setup time	t_{19}	27	–	ns	3) 5) 6)

1) Asynchronous inputs are INTR, NMI, HOLD, PEREQ, $\overline{\text{ERROR}}$, and $\overline{\text{BUSY}}$. The specification is given only for testing purposes, to assure recognition at a specific CLK edge.

2) Delay from 1.0 V on the CLK to 0.8 V or 2.0 V or float on the output as appropriate for valid or floating condition.

3) Output load C_L = 100 pF

4) Float condition occurs when output current is less than I_{LO} in magnitude.

5) Delay measured from address either reaching 0.8 V or 2.0 V (valid) to status going active reaching 2.0 V or status going inactive reaching 0.8 V.

6) For a load capacitance of 10 pF on status/$\overline{\text{PEACK}}$ lines, subtract maximum 7 ns.

SAB 82284-1 Timing Requirements

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
$\overline{\text{SRDY}}/\overline{\text{SRDYEN}}$ setup time	t_{11}	15	–	ns	–
$\overline{\text{SRDY}}/\overline{\text{SRDYEN}}$ hold time	t_{12}	0	–	ns	–
$\overline{\text{ARDY}}/\overline{\text{ARDYEN}}$ setup time	t_{13}	0	–	ns	1)
$\overline{\text{ARDY}}/\overline{\text{ARDYEN}}$ hold time	t_{14}	30	–	ns	1)
PCLK delay	t_{19}	0	35	ns	C_L = 75 pF I_{OL} = 5 mA I_{OH} = –1 mA

SAB 82288-1 Timing Requirements

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
CMDLY setup time	t_{12}	15	–	ns	–
CMDLY hold time	t_{13}	0	–	ns	–
Command inactive delay	t_{30}	5	20	ns	2)
Command active delay	t_{29}	3	21	ns	2)
ALE active delay	t_{16}	3	16	ns	3)
ALE inactive delay	t_{17}	–	19	ns	3)
DT/$\overline{\text{R}}$ read active delay	t_{19}	–	23	ns	3)
DT/$\overline{\text{R}}$ read inactive delay	t_{22}	5	20	ns	3)
DEN read active delay	t_{20}	5	21	ns	3)
DEN read inactive delay	t_{21}	3	21	ns	3)
DEN write active delay	t_{23}	–	23	ns	3)
DEN write inactive delay	t_{24}	3	19	ns	3)

1) These times are given for testing purpose to assure a predetermined action

2) C_L = 300 pF max.
I_{OL} = 32 mA max.
I_{OH} = –5 mA max.

3) C_L = 150 pF
I_{OL} = 16 mA max.
I_{OH} = –1 mA max.

AC Characteristics SAB 80286-12

AC timings are referenced to 0.8 V and 2.0 V points of signals as illustrated in data sheet waveforms, unless otherwise noted.

T_C = 0 to 85°C; V_{CC} = +5 V ±5%

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
System clock (CLK) period	t_1	40	250	ns	–
System clock (CLK) low time	t_2	11	237	ns	at 1.0 V
System clock (CLK) high time	t_3	13	239	ns	at 3.6 V
System clock (CLK) rise time	t_{17}	–	8	ns	1.0 V to 3.6 V
System clock (CLK) fall time	t_{18}	–	8	ns	3.6 V to 1.0 V
Async inputs, setup time	t_4	15	–	ns	1)
Async inputs, hold time	t_5	15	–	ns	1)
RESET setup time	t_6	18	–	ns	–
RESET hold time	t_7	5	–	ns	–
Read data setup time	t_8	5	–	ns	–
Read data hold time	t_9	6	–	ns	–
$\overline{\text{READY}}$ setup time	t_{10}	22	–	ns	–
$\overline{\text{READY}}$ hold time	t_{11}	20	–	ns	–
Status active delay	t_{12a1}	3	18	ns	2) 3)
$\overline{\text{PEACK}}$ active delay	t_{12a2}	3	20	ns	2) 3)
Status/$\overline{\text{PEACK}}$ inactive delay	t_{12b}	3	22	ns	2) 3)
Address valid delay	t_{13}	1	32	ns	2) 3)
Write data valid delay	t_{14}	0	30	ns	2) 3)
Address/status/data float delay	t_{15}	0	32	ns	2) 4)
HLDA valid delay	t_{16}	0	27	ns	2) 3)
Address valid to status valid setup time	t_{19}	22	–	ns	3) 5)

1) Asynchronous inputs are INTR, NMI, HOLD, PEREQ, $\overline{\text{ERROR}}$, and $\overline{\text{BUSY}}$. The specification is given only for testing purposes, to assure recognition at a specific CLK edge.

2) Delay from 1.0 V on the CLK to 0.8 V or 2.0 V or float on the output as appropriate for valid or floating condition.

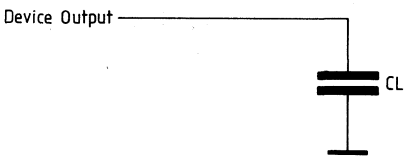
3) Output load C_L = 100 pF

4) Float condition occurs when output current is less than I_{LO} in magnitude.

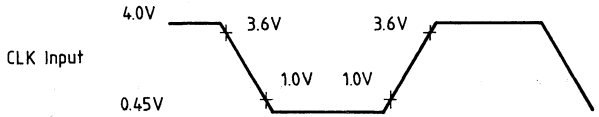
5) Delay measured from address either reaching 0.8 V or 2.0 V (valid) to status going active reaching 2.0 V or status going inactive reaching 0.8 V.

AC Testing Waveforms

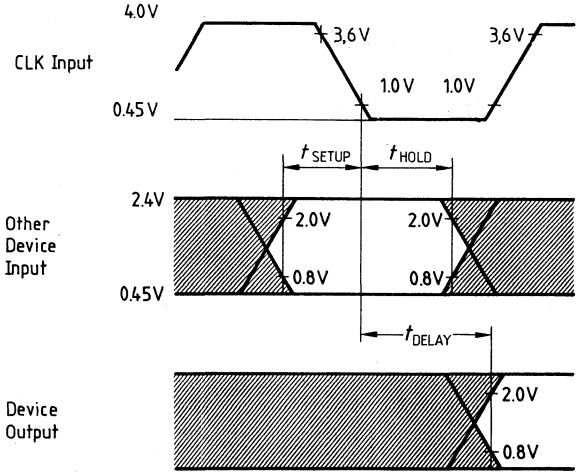
Test Loading on Outputs



Drive and Measurement Points – CLK Input

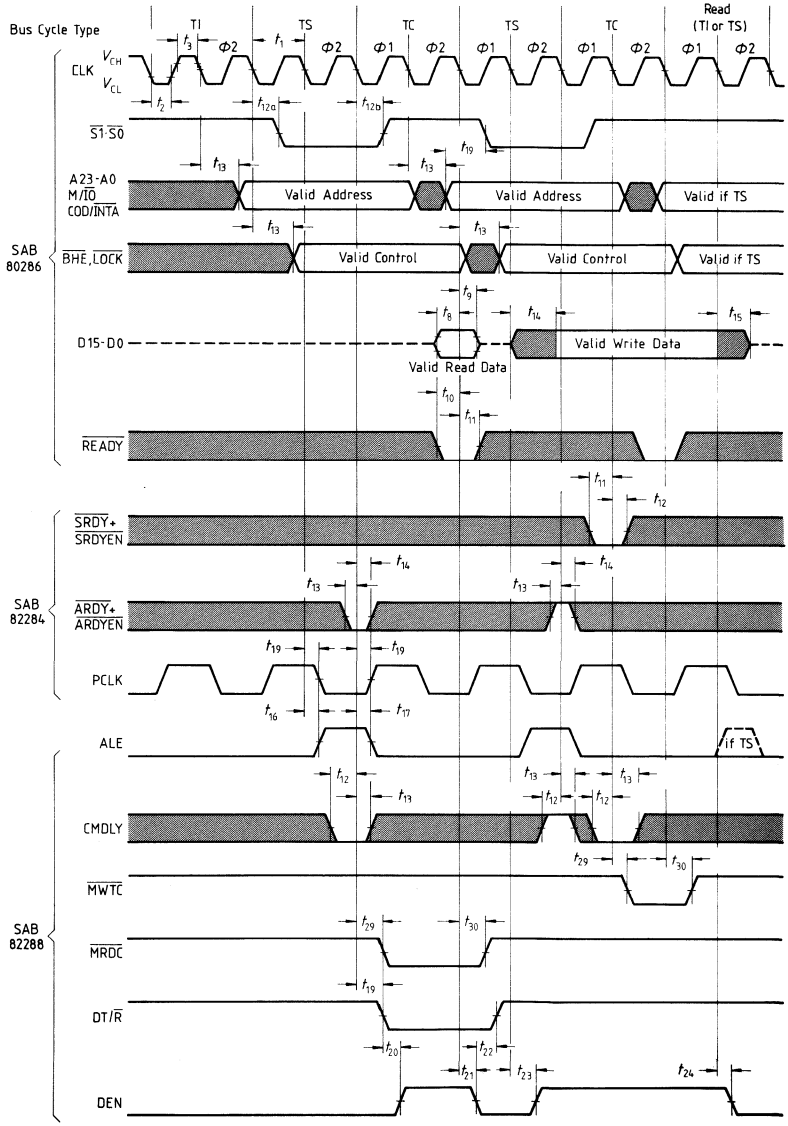


Setup, Hold and Delay Time Measurement – General

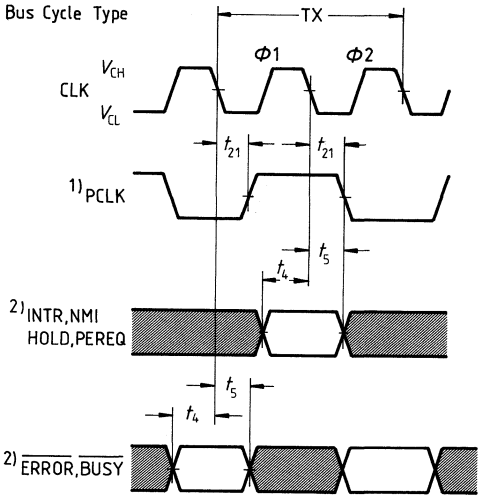


Waveforms

Major Cycle Timing



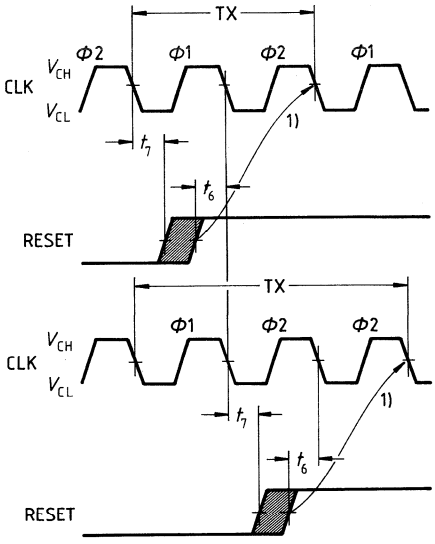
Asynchronous Input Signal Timing



1) PCLK indicates which processor cycle phase will occur on the next CLK, PCLK may not indicate the correct phase until the first bus cycle is performed.

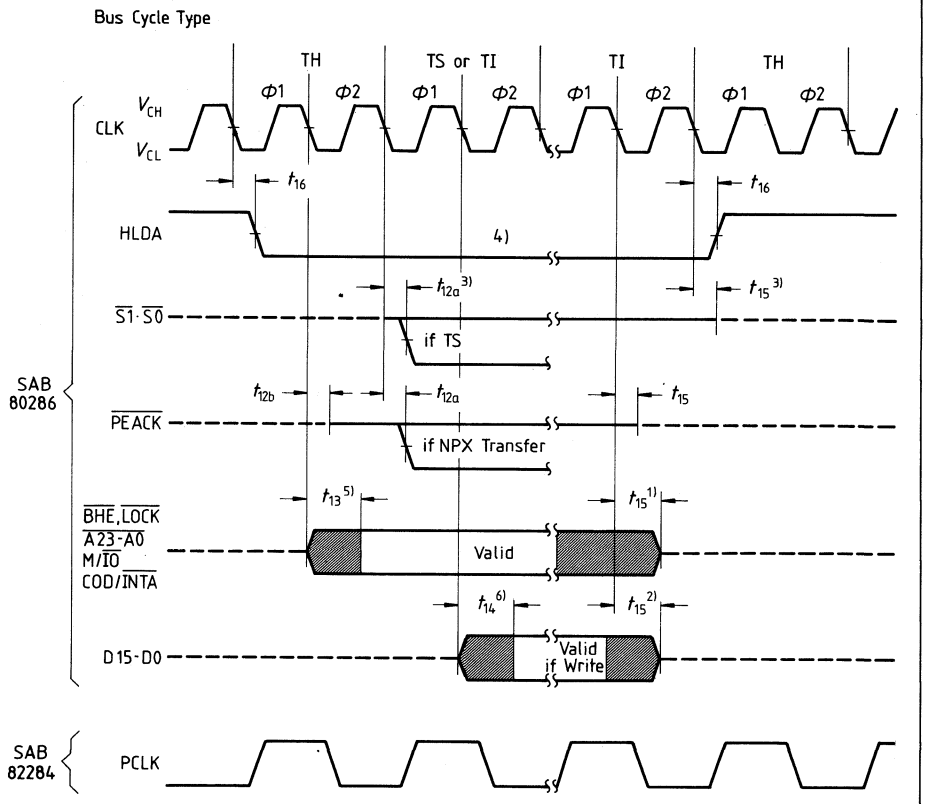
2) These inputs are asynchronous. The setup and hold times shown assure recognition for testing purposes.

Reset Input Timing and Subsequent Processor Cycle Phase



[1] When RESET meets the setup time shown, the next CLK will start or repeat Φ 2 of a processor cycle.

Exiting and Entering HOLD



1) These signals may not be driven by the SAB 80286 during the time shown. The worst case in terms of latest float time is shown.

2) The data bus will be driven as shown if the last cycle before TI in the diagram was a write TC.

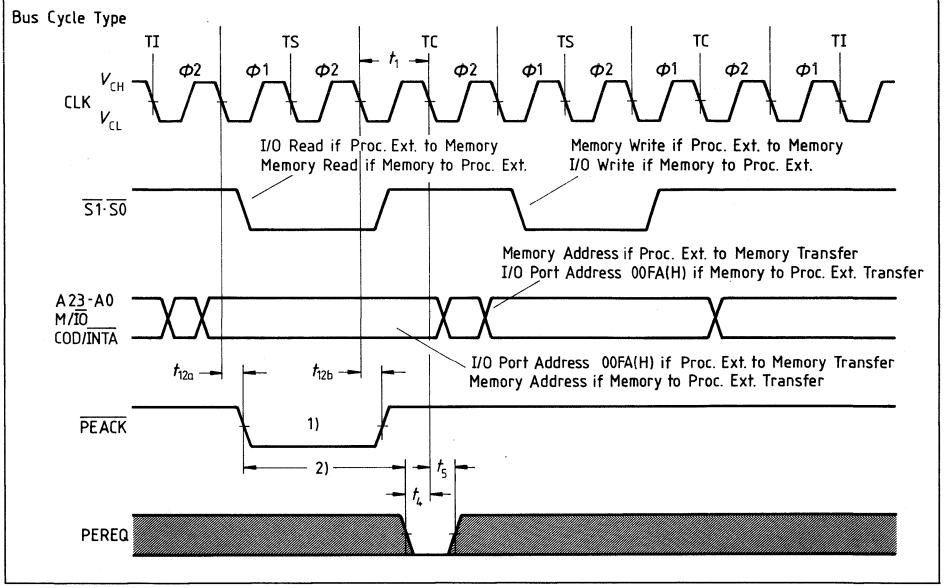
3) The SAB 80286 floats its status pins during TH. Pullup resistors in SAB 82288 keep these signals high.

4) For HOLD request set up to HLDA (refer to figure on Multibus write terminated by async ready).

5) $\overline{BHE}$ and $\overline{LOCK}$ are driven at this time but will not become valid until TS.

6) The data bus will remain in tristate off, if a read cycle is performed.

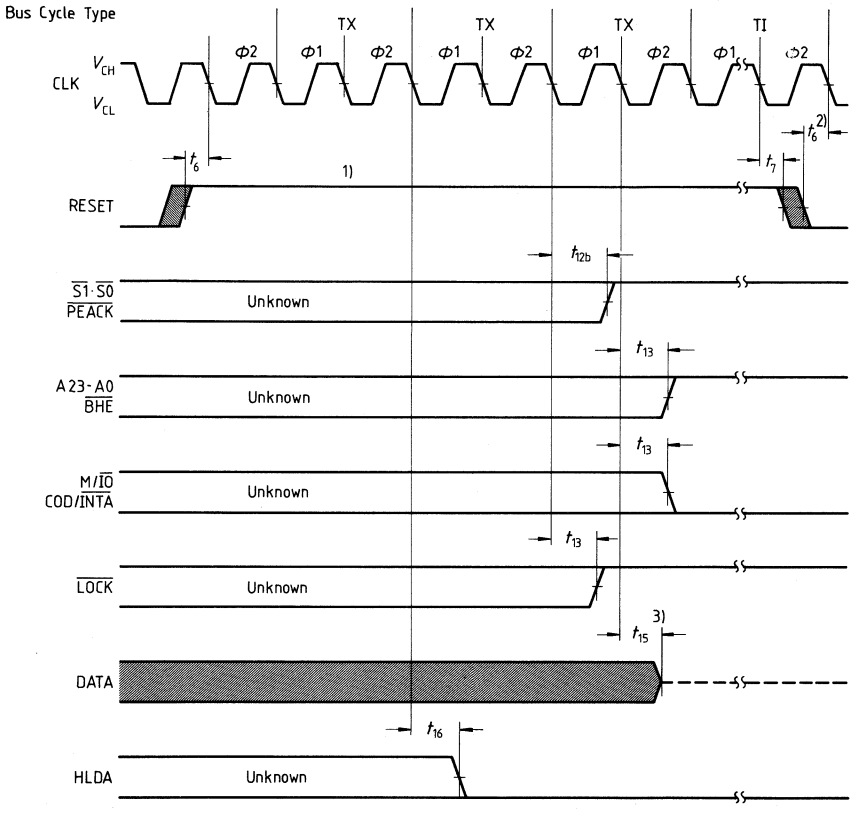
PEREQ/PEACK Timing for One Transfer Only



1) PEACK always goes active during the first bus operation of a processor extension data operand transfer sequence. The first bus operation will be either a memory read at operand address or I/O read at port address 00FA(H).

2) To prevent a second processor extension data operand transfer, the worst case maximum time (shown above) is: $3\times t_1 - t_{12a\,max} - t_{4\,min}$. The actual, configuration-dependent, maximum time is: $3\times t_1 - t_{12a\,max} - t_{4\,min} + A\times 2\times t_1$.
A is the number of extra TC states added to either the first or second bus operation of the processor extension data operand transfer sequence.

Initial SAB 80286 Pin State during RESET



1) Setup time for RESET↑ may be violated with the consideration that Φ 1 of the processor clock may begin one system CLK period later.

2) Setup and hold times for RESET↓ must be met for proper operation.

3) The data bus is only guaranteed to be in tristate off at the time shown.

32-Bit Microprocessors

SIEMENS

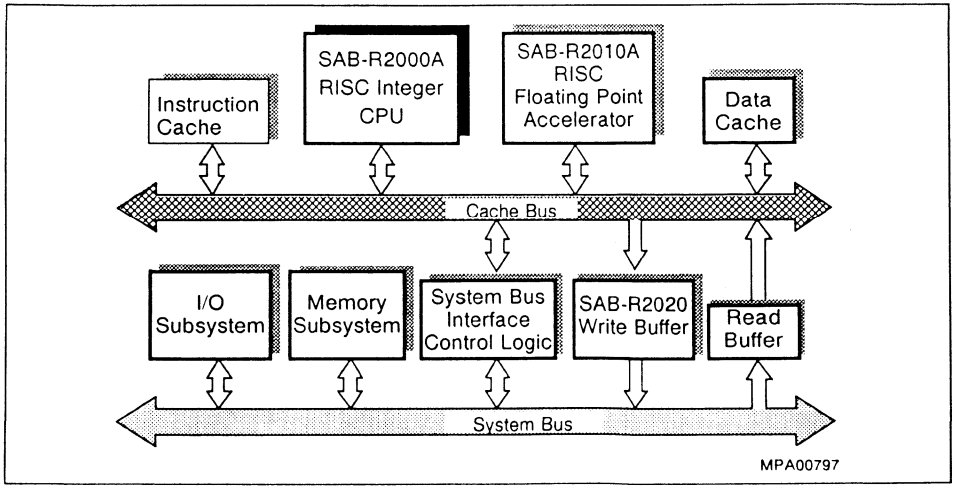
High Performance 32-Bit RISC Microprocessor

SAB-R2000A

Including on-chip memory management and cache control with support for up to three external coprocessors including the SAB-R2010A floating point accelerator.

Advance Information

- Two tightly-coupled 16 MHz units on a single chip
  - full 32-bit RISC CPU
  - system control processor (CP0)
- Load / store architecture
  - support for loading misaligned data
  - configurable endianness
- Full 32-bit operation
  - thirty two general purpose 32-bit registers
  - all instructions and addresses are 32-bits
- On-chip cache control
- On-chip memory management unit
- High Performance
  - 12 VAX 11/780 mips average at 16 MHz
- Extensive software and development support
- Instruction set compatible to R3000 processors
- Fully pin and functionally compatible to all R2000A processors of other manufacturers
- Ceramic package: C-PGA-145



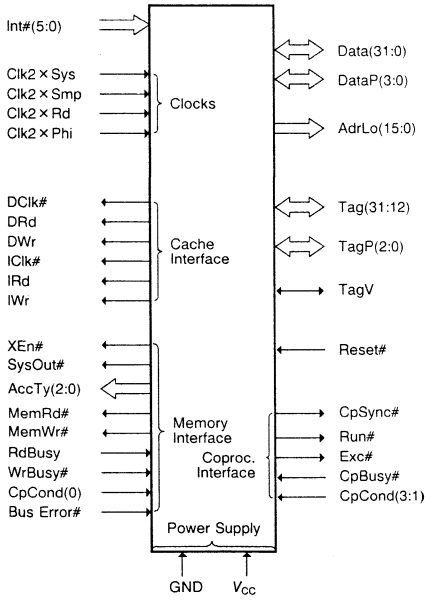
Ordering Information

Type	Ordering code	Package	Description
SAB-R2000A-12-A	Q67120-C552	C-PGA-145	32-bit RISC Processor, 12.5 MHz
SAB-R2000A-16-A	Q67120-C494	C-PGA-145	32-bit RISC Processor, 16.67 MHz
SAB-R2000A-20-A	Q67120-C517	C-PGA-145	32-bit RISC Processor, 20 MHz

Introduction

The SAB-R2000A is a high-performance microprocessor architecture implemented as a full-custom CMOS VLSI chip which achieves 20 VAX 11/780 mips average at 16 MHz. It is a single chip microprocessor that consists of two tightly-coupled 16 MHz units. The first is a full 32-bit RISC CPU. The second unit is a System Control Processor (CP0) that integrates the functions needed to keep the CPU from idling for memory access (Memory Management) and/or for Interrupt and Exception handling. The System Control Processor contains a Translation Lookaside Buffer (TLB) and control registers to efficiently support a virtual memory system, as well as all the control logic to realize separate caches for instructions and data. This architecture permits a dual-cache bandwidth of up to 200 Mbytes/second at 16 MHz using standard SRAM devices. It is possible for up to three external coprocessors (including the SAB-R2010A floating point accelerator) to be coupled with the SAB-R2000A. The synchronous coprocessor interface generates all the addresses and manages the memory interface control.

Figure 1
Logic Symbol



MPL00798

Note: "#" signifies an active low signal.

Pin Names

Data(31:0)	Data Bus
DataP(3:0)	Even Parity for Data Bus
AdrLo(15:0)	Low Adress Bus
Tag(31:12)	Cache Tag and High Adress Bus
TagP(2:0)	Even Parity over TagV and Tag Bus
TagV	Tag Validity Indicator
Reset#	Synchronous Initialization
CpSync#	Coprocessor Synchronization
Run#	Processor in Run or Stall State
Exc#	Exception
CpBusy#	Coprocessor Busy
CpCond(3:0)	Coprocessor Condition
BusError#	Bus Error
WrBusy#	Write Busy for Main Memory
RdBusy#	Read Busy for Main Memory
MemWr#	Main Memory Write
MemRd#	Main Memory Read
AccTy(2:0)	Access Type
SysOut#	System Clock Out
XEn#	Read Enable (Read Buffer)
DWr	Data Cache Write Enable
IWr	Instruction Cache Write Enable
DRd	Data Cache Read Enable
IRd	Instruction Cache Read Enable
DClk#	Data Cache Latch Enable
IClk#	Instruction Cache Latch Enable
Int#(5:0)	Interrupt Bus

Pin Configurations

Figure 2
C-PGA-145 (Top View)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	V_{CC14}	AdrLo (6)	AdrLo (10)	AdrLo (11)	V_{CC12}	AdrLo (14)	AdrLo (15)	CpCon d (0)	CpCon d (2)	CpCond (3)	Intr# (2)	Intr# (5)	Wr Busy#	Reset#	V_{CC10}
B	AdrLo (3)	Resvd 2	AdrLo (7)	AdrLo (9)	AdrLo (12)	Resvd 3	AdrLo (13)	CpCon d (1)	Intr# (1)	Intr# (3)	Cp Busy#	Bus Error#	Resvd 4	Tag(12)	Tag(15)
C	AdrLo (0)	AdrLo (4)	V_{CC13}	AdrLo (5)	AdrLo (8)	Gnd13	Gnd12	V_{CC11}	Intr# (0)	Intr# (4)	Rd Busy	Gnd11	Tag(13)	TagP(0)	Tag(18)
D	Data (1)	AdrLo (2)	Gnd0										Tag(14)	Tag(17)	Tag(19)
E	DataP (0)	Data (0)	AdrLo (1)										Tag(16)	Tag(20)	V_{CC9}
F	V_{CC0}	Data (7)	Data (2)										Gnd10	Tag(21)	Tag(23)
G	Data (4)	Data (3)	Gnd1										Gnd9	Tag(22)	TagP(1)
H	Data (6)	Data (5)	Data (8)										V_{CC8}	Tag(25)	Tag(24)
J	Data (10)	DataP (1)	Data (9)										Tag(28)	Tag(29)	Tag(26)
K	Data (15)	Data (11)	Gnd2										Gnd8	TagP(2)	Tag(27)
L	V_{CC1}	Data (12)	Data (17)										Acc Ty(2)	Tag(31)	Tag(30)
M	Data (13)	Data (16)	DataP (2)										Gnd7	Acc Ty(1)	V_{CC7}
N	Data (14)	Data (18)	Data (19)	Gnd3	Data (24)	Data P(3)	V_{CC3}	V_{CC4}	Gnd5	Gnd6	DRd	MemWr #	MemRd #	Run#	TagV
P	Data (23)	Data (20)	Resvd 0	Data (22)	Data (26)	Data (27)	Resvd 1	Data (30)	Clk2× Sys	Clk2× Rd	DClk #	IRd	IWr	Cp Sync#	Acc Ty(0)
Q	V_{CC2}	Data (21)	Data (25)	Data (31)	Data (28)	Gnd4	Data (29)	Excep tion#	Clk2× Phi	Clk2× Smp	SysOut #	V_{CC5}	IClk#	DWr	V_{CC6}

MPP00799

Pin Definitions and Functions

Symbol	Pin Number	Input (I) Output (O)	Function
Data(31:0)	Q4,P8,Q7,Q5, P6,P5,Q3,N5, P1,P4,Q2,P2, N3,N2,L3,M2, K1,N1,M1,L2, K2,J1,J3,H3, F2,H1,H2,G1, G2,F3,D1,E2	I/O	A 32-bit bus used for all instruction and data transmission among the processor, caches, memory interface, and coprocessors.
DataP(3:0)	N6,M3,J2,E1	I/O	A 4-bit bus containing even parity over the data bus.
Tag(31:12)	L14,L15,J14, J13,K15,J15, H14,H15,F15, G14,F14,E14, D15,C15,D14, E13,B15,D13, C13,B14	I/O	A 20-bit bus used for transferring cache tags and high addresses between the processor, caches, and memory interface.
TagV	N15	I/O	The tag validity indicator
TagP(2:0)	K14,G15,C14	I/O	A 3-bit bus containing even parity over the catenation of TagV and Tag31:12.
AdrLo (15:0)	A7,A6,B7,B5, A4,A3,B4,C5, B3,A2,C4,C2, B1,D2,E3,C1	O	A 16-bit bus containing byte addresses used for transferring low addresses from the processor to the caches and memory interface.
IRd	P12	O	Read enable for the instruction cache.
IWr	P13	O	Write enable for the instruction cache.
IClk#	Q13	O	The instruction cache address latch clock. This clock runs continuously.
DRd	N11	O	The read enable for the data cache.
DWr	Q14	O	The write enable for the data cache.
DClk#	P11	O	The data cache address latch clock. This clock runs continuously.

Pin Definitions and Functions (cont'd)

<table>
<tr><th>Symbol</th><th>Pin Number</th><th>Input (I)
Output (O)</th><th>Function</th></tr>
<tr><td>AccTy(2:0)</td><td>L13,M14,P15</td><td>O</td><td>A 3-bit bus used to indicate the size of data being transferred on the data bus, whether or not a data transfer is occurring, and the purpose of the transfer. The run encoding of the Access Type is illustrated in the table below.
<table>
<tr><th>AccTy(2)</th><th>AccTy(1:0)</th><th>size</th></tr>
<tr><td>1</td><td>XX</td><td>no transaction</td></tr>
<tr><td>0</td><td>00</td><td>byte</td></tr>
<tr><td>0</td><td>01</td><td>half word</td></tr>
<tr><td>0</td><td>10</td><td>tribyte</td></tr>
<tr><td>0</td><td>11</td><td>word</td></tr>
</table></td></tr>
<tr><td>MemWr#</td><td>N12</td><td>O</td><td>Signals the occurrence of a main memory write.</td></tr>
<tr><td>MemRd#</td><td>N13</td><td>O</td><td>Signals the occurrence of a main memory read.</td></tr>
<tr><td>BusError#</td><td>B12</td><td>I</td><td>Signals the occurrence of a bus error during a main memory read or write.</td></tr>
<tr><td>Run#</td><td>N14</td><td>O</td><td>Indicates whether the processor is in the run or stall state.</td></tr>
<tr><td>Exc#</td><td>Q8</td><td>O</td><td>Indicates that the instruction about to commit state should be aborted.</td></tr>
<tr><td>SysOut#</td><td>Q11</td><td>O</td><td>A reflection of the internal processor clock used to generate the system clock.</td></tr>
<tr><td>CpSync#</td><td>P14</td><td>O</td><td>A clock which is identical to SysOut# and used by coprocessors for timing synchronization with the CPU.</td></tr>
<tr><td>RdBusy</td><td>C11</td><td>I</td><td>The main memory read stall termination signal. In most system designs RdBusy is normally asserted and is deasserted only to indicate the successful completion of a memory read. RdBusy is sampled by the processor only during memory read stalls.</td></tr>
<tr><td>WrBusy#</td><td>A13</td><td>I</td><td>The main memory write stall initiation/termination signal.</td></tr>
</table>

Pin Definitions and Functions (cont'd)

<table>
<tr><th>Symbol</th><th>Pin Number 144-Pin</th><th>Input (I) Output (O)</th><th>Function</th></tr>
<tr><td>CpBusy#</td><td>B11</td><td>I</td><td>The coprocessor busy stall initiation/termination signal.</td></tr>
<tr><td>CpCond (3:0)</td><td>A10,A9,B8,A8</td><td>I</td><td>A 4-bit bus used to transfer conditional branch status from the coprocessors to the main processor.</td></tr>
<tr><td>Int#(5:0)</td><td>A12,C10,B10, A11,B9,C9</td><td>I</td><td>A 6-bit bus used by the memory interface and coprocessor to signal maskable interrupts to the processor. It is also used to specify the processor's mode during Reset. The Table below summarizes the mode selectable features.
<table>
<tr><th rowspan="2">Inter-rupt#</th><th colspan="2">Y Cycle Modes</th><th colspan="2">Z Cycle Modes</th></tr>
<tr><th>Phase 1</th><th>Phase 2</th><th>Phase 1</th><th>Phase 2</th></tr>
<tr><td>Int#(0)</td><td>Reserved</td><td>Reserved</td><td>Reserved</td><td>BigEndian#</td></tr>
<tr><td>Int#(1)</td><td>Reserved</td><td>Reserved</td><td>Reserved</td><td>Tristate#</td></tr>
<tr><td>Int#(2)</td><td>Reserved</td><td>Reserved</td><td>Reserved</td><td>NoCache#</td></tr>
<tr><td>Int#(3)</td><td>Reserved</td><td>Reserved</td><td>Reserved</td><td>Bus DriveOn</td></tr>
<tr><td>Int#(4)</td><td>Phase DelayOn#</td><td>Phase DelayOn#</td><td>Asserted#</td><td>Phase DelayOn#</td></tr>
<tr><td>Int#(5)</td><td>Reserved</td><td>Reserved</td><td>Asserted#</td><td>R2000Md</td></tr>
</table></td></tr>
<tr><td>Clk2 × Sys</td><td>P9</td><td>I</td><td>The master double frequency input clock used for generating SysOut#.</td></tr>
<tr><td>Clk2 × Smp</td><td>Q10</td><td>I</td><td>A double frequency clock input used to determine the sample point for data coming into the processor and coprocessors.</td></tr>
<tr><td>Clk2 × Rd</td><td>P10</td><td>I</td><td>A double frequency clock input used to determine the enable time of the cache RAMs.</td></tr>
<tr><td>Clk2 × Phi</td><td>Q9</td><td>I</td><td>A double frequency clock input used to determine the position of the internal phases, phase1 and phase2.</td></tr>
<tr><td>Reset#</td><td>A14</td><td>I</td><td>Synchronous initialization input used to force execution starting from the reset memory address. Reset# must be deasserted synchronously but asserted asynchronously. The deassertion of Reset# must be synchronized by the leading edge of SysOut.</td></tr>
</table>

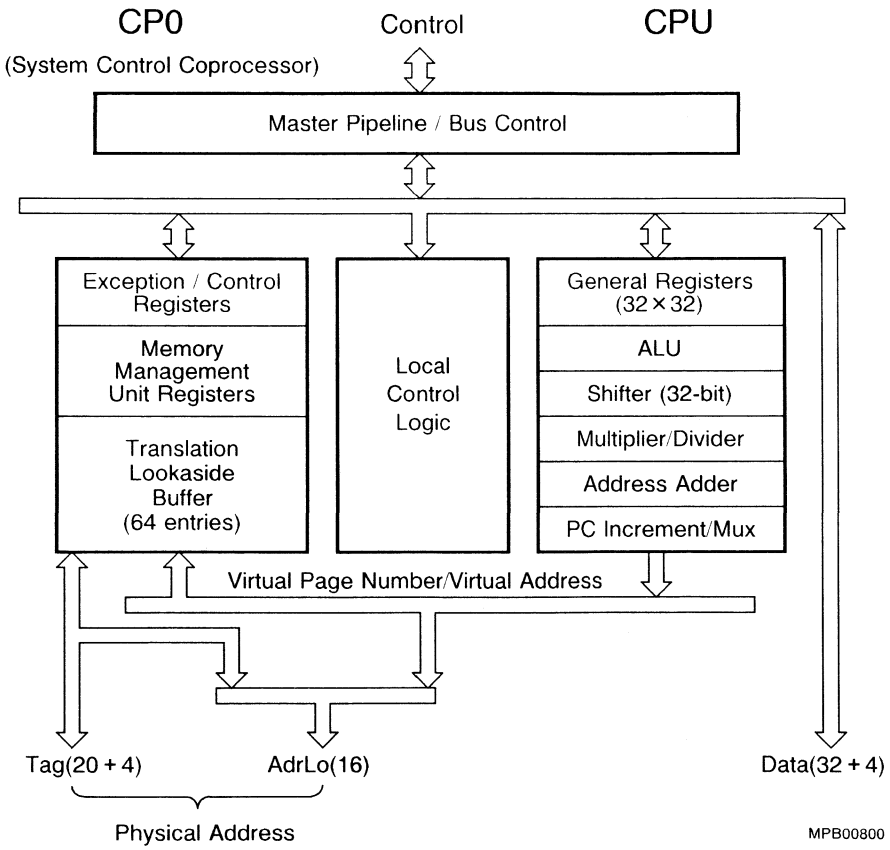
Pin Definitions and Functions (cont'd)

Symbol	Pin Number 144-Pin	Input (I) Output (O)	Function
GND13-0	C6,C7,C12, F13,G13, K13,M13, N10,N9, Q6,N4,K3, G3,D3		Ground
V_{CC}14-0	A1,C3,A5,C8, A15,E15,H13, M15,Q15, Q12,N8,N7, Q1,L1,F1		Power Supply (+ 5 V)
Resvd4-0	B13,B6,B2, P7,P3		Reserved

Functional Description

The SAB-R2000A consists of two integrated processors – a RISC CPU and a System Control Processor (CP0). Figure 3 is a block diagram of the SAB-R2000A which shows the functions incorporated within it.

Figure 3
Functional Block Diagram



Basic Architecture

On the right hand side of figure 3 is the CPU datapath that implements the 5-stage pipeline, which will be explained shortly. The datapath consists of a stack of functional units including 32 General Registers, ALU, 32-bit Shifter and an autonomous Multiply Divide unit. An Address Adder and Increment/MUX for the PC generate instruction and data addresses alternatively at double the basic clock rate. This is necessary so that the SAB-R2000A can access both the Instruction and Data caches in a single CPU cycle, due to the multiplexed Data bus.

On the left of figure 3 is the System Control Processor (CP0). It's major element is a fully associative 64-entry Translation Lookaside Buffer (TLB), which translates a 20-bit virtual page number into a physical page frame number in a clock phase. As well as address translation the System Control Coprocessor also manages the exception handling and error recovery, the external cache interface, the memory control interface and the external coprocessor interface. It also incorporates on-chip tag comparators, parity generators and checkers.

Integer CPU

Pipeline Architecture

The execution of a single SAB-R2000A instruction consists of five primary steps or pipe stages.

(1) IF — Instruction Fetch:
Access the TLB and translate the instructions virtual address into its physical address to read an instruction from the I-cache. Note that the instruction is not actually read into the processor until the beginning (phase 1) of the RD pipe stage. Refer to figure 4.

(2) RD — Register Fetch/Instruction Decode:
Read any required operands from the CPU registers while decoding the instruction.

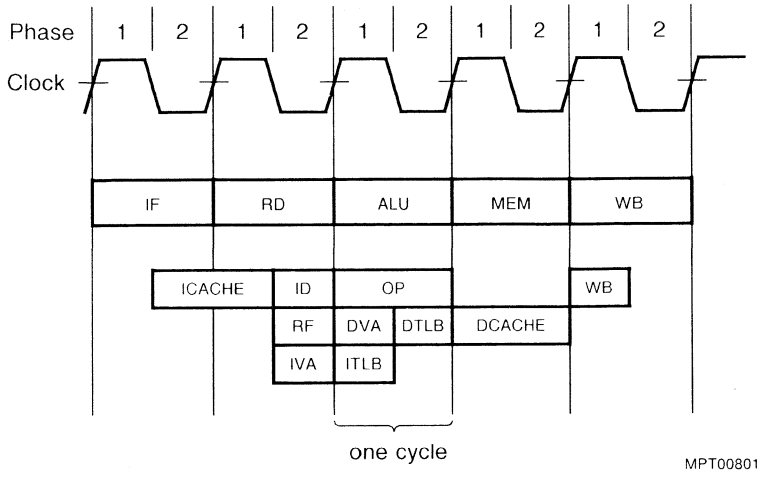
(3) ALU — ALU Operation:
Perform the required operation on instruction operands..

(4) MEM — Memory Access.
Access memory (D-cache) if required (for a Load or Store instruction).

(5) WB — Writeback:
Write back ALU results or value loaded from D-cache to the register file.

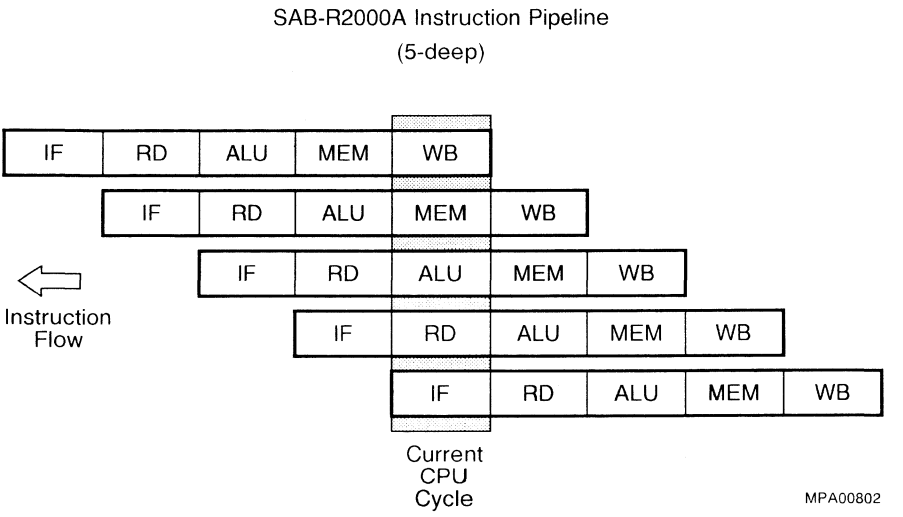
Each of these steps requires approximately one CPU cycle as shown in figure 4 (parts of some operations lap over into another cycle while other operations require only 1 2 cycle).

Figure 5
Execution Sequence of an Instruction



ICACHE	:	Instruction Cache Access
ID	:	Instruction Decode
RF	:	Register Operand Fetch
OP	:	Operation (ALU/Shift)
IVA	:	Instruction Virtual Address Calculation
ITLB	:	TLB Access for Instruction
DVA	:	Data Virtual Address Calculation
DTLB	:	TLB Access for Data
DCACHE	:	Data Cache Access
WB	:	Write Back to Register File

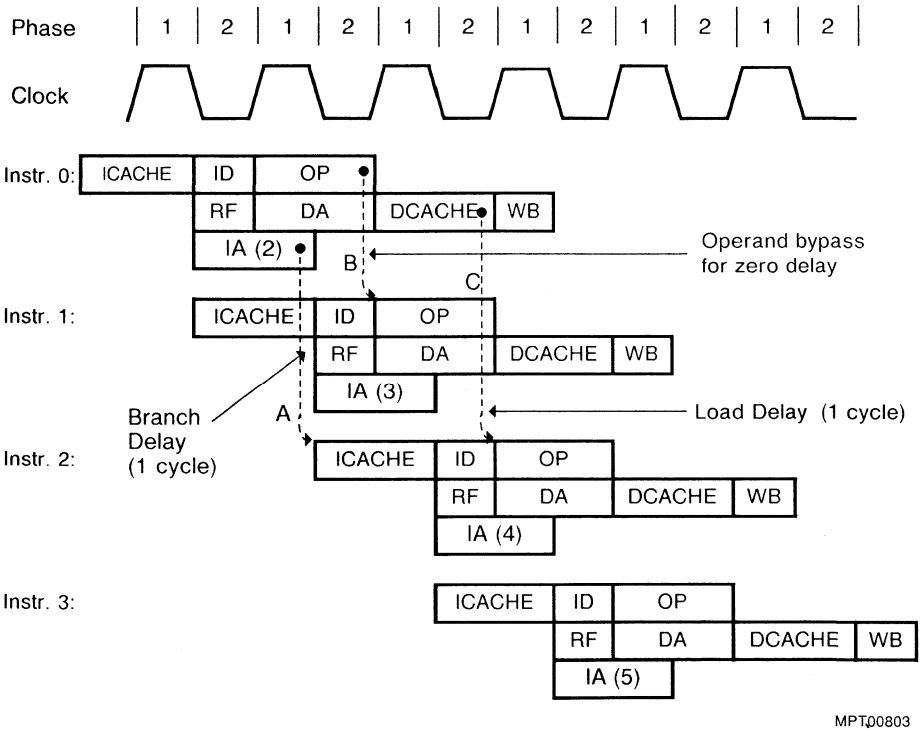
Figure 5
Overlapped execution of 5 instructions



The SAB-R2000A uses a 5-stage pipeline to achieve an instruction execution rate approaching one instruction per CPU cycle. Thus the executions of five instructions at a time are overlapped as shown in figure 5. This pipeline operates efficiently because different CPU resources (address and data bus accesses, ALU operations, register accesses and so on) are utilized on a non-interfering basis. As figure 6 illustrates there is not only parallelism due to pipelining but also parallelism within the execution of a single instruction.

The clock cycle is divided into two phases. To access the external instruction and data caches (ICACHE, DCACHE) requires 1 cycle, as do major internal operations (OP, DA, IA). Instruction Decode (ID) is simple enough to occur within one phase, overlapped with Register Fetch (RF). Instruction address calculation and translation (IA) also overlaps Instruction Decode and Register Fetch, it consists of instruction virtual address calculation (IVA see figure 4) and TLB access for instruction address translation (ITLB). The Instruction address calculation and translation that is performed in the present instruction is for the second instruction following the present instruction, as shown in figure 6 (i.e. IA in Instr. 0 is for Instr. 2).

Figure 6
Instruction Pipeline Dependencies



IA : Instruction address calculation and translation
DA : Data address calculation and translation

IA is performed here especially so that a branch, for example, at Instr. 0 in figure 6 can address the ICACHE access of Instr. 2 (see dotted line A), i.e. one cycle latency. What happens is that after the condition has been evaluated (i.e. for a conditional branch), either PC + 2 or the Branch Target address is MUXed to the IF stage of the second succeeding instruction (again dotted line A). This means that if the branch is taken the branch target address is the address of Instr. 2. On the other hand, if the branch is not taken, PC + 2 (i.e. two sequential instructions after the current PC) is the address of Instr. 2. Data address calculation and translation (DA) is similar to IA. Similarly, a load at Instr. 0 fetches data that are immediately used by the OP of Instr. 2 (dotted line C), while an ALU/Shift result gets passed directly into Instr. 1 with no delay (dotted line B). This tight coupling between instructions makes for a highly efficient pipeline. Note that the IA-ICACHE and DA-DCACHE cycles are displaced by one phase, so that the corresponding TLB and cache accesses can be interleaved on a single set of buses (see also figure 4).

As shown the SAB-R2000A uses a number of techniques internally to enable execution of all instructions in a single cycle, such as bypassing between instructions in the pipeline to keep the latency of branches and memory references to 1 cycle and to allow ALU results to be used in the succeeding instruction. However, there are two categories of instructions whose special requirements could disturb the smooth flow of instructions through the pipeline:

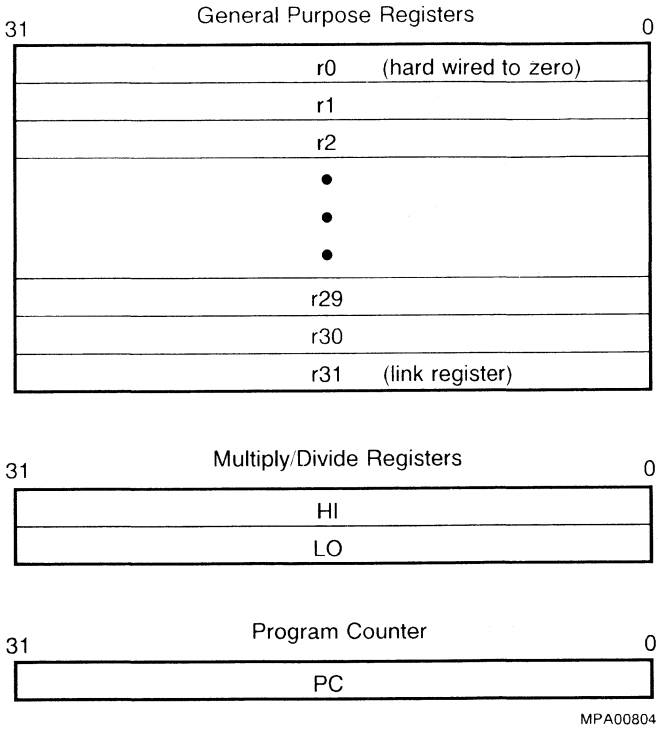
- **Load** instructions have a delay, or latency, of one cycle before the data being loaded is available to another instruction.
- **Jump and Branch** instructions also have a delay of one cycle while they fetch the instruction and target address if the branch is taken.

The SAB-R2000A continues execution despite the inherent 1-cycle delay. Loads, Jumps and Branches do not interrupt the normal flow of instructions through the pipeline, i.e. the pipeline is not stalled. The processor always executes the instruction immediately following one of these "delayed instructions". Instead of having the processor manage these pipeline delays, the SAB-R2000A turns over the responsibility for dealing with "delayed instructions" to software. Thus the assembler must insert an appropriate instruction immediately following a "delayed instruction". It also has the responsibility of ensuring that the inserted instruction has no dependencies relating to the "delayed instruction". In the SAB-R2000A architecture only multi-cycle delays (e.g. waiting for SAB-R2010A results, MUL/DIV results, etc.) are handled by hardware interlocks – 1-cycle delays, as described, are handled much more efficiently by software for all implementations (thus guaranteeing scalability).

CPU Registers

There are 32 general purpose 32-bit registers, two 32-bit registers that hold the results of integer multiply and divide operations, and a 32-bit program counter as shown in figure 7.

Figure 7
CPU Registers



The 32 General Purpose Registers are treated symmetrically, with two exceptions – r0 is hardwired to a zero value, and r31 is the link register for Jump And Link instructions. Register r0 may be specified as a target register for any instruction when the result of the operation is discarded. The register maintains a value of zero under all conditions when used as a source register.

The two Multiply/Divide registers (HI, LO) store the double-word, 64-bit result, of multiply operations or the quotient and remainder of divide operations.

The Program Counter contains the current virtual address of the next instruction to be executed.

There is no condition code register. If an instruction generates a condition, the corresponding flags are stored in a general purpose register. This means that the pipeline is freed of any special mechanisms to by-pass condition codes, interlock on them, or abort writing them on exceptions. Instead the methods already implemented to deal with register-value dependencies are employed. Further, conditions mapped onto the register file are subject to the same compile-time optimizations in allocation and reuse as other register variables.

There is also no Program Status Word (PSW) register - the functions traditionally provided by a PSW are instead provided in the Status and Cause registers incorporated within the CP0. CP0 has a number of special purpose registers that are used in conjunction with the memory management system and during exception processing. These will be explained in the CP0 section.

Data Formats

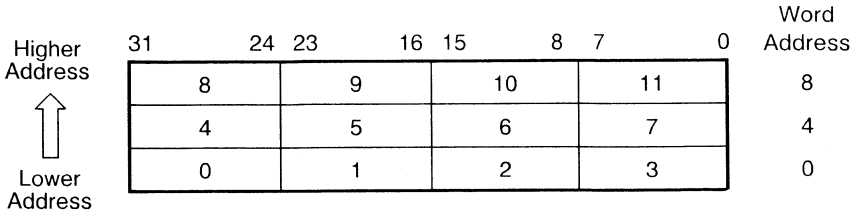
The SAB-R2000A defines signed/unsigned 32-bit words, 16-bit half-words and 8-bit bytes. The byte ordering is configurable either little-endian (iAPX x86®, NS32000®, DEC VAX®) or big-endian (MC680x0®, IBM 370®) byte ordering. Hence it is compatible with existing databases generated by machines that access bytes in either order.

Bit 0 is always the least significant (rightmost) bit, thus bit designations are always little-endian (although no instructions explicitly designate bit positions within words). Figures 8 and 9 show the ordering of bytes within words, and the ordering of words within multiple-word structures for the big-endian and little-endian conventions.

Special instructions are provided for addressing words that are not aligned on 4-byte (word) boundaries (Load/Store-Word-Left/Right; LWL, LWR, SWL, SWR). These instructions are used in pairs to provide addressing of misaligned words with one additional instruction cycle over that required for aligned words.

iAPX x86® is a trademark of INTEL
NS32000® is a trademark of National Semiconductor
DECVAX® is a trademark of Digital Equipment Cooperation
MC680x0® is a trademark of Motorola Inc.
IBM 370® is a trademark of IBM Cooperation

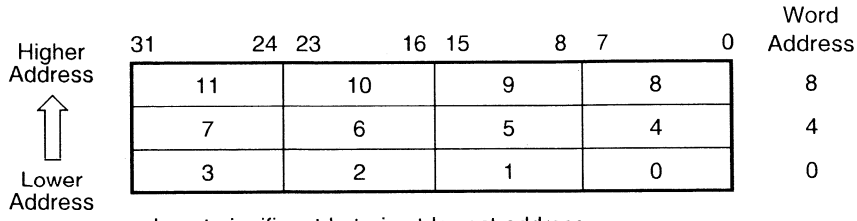
Figure 8
Addresses of Bytes within Words: Big Endian



- Most significant byte is at lowest address.
- Word is addressed by byte address of most significant byte.

MPA00805

Figure 9
Addresses of Bytes within Words: Little Endian



- Least significant byte is at lowest address.
- Word is addressed by byte address of least significant byte.

MPA00806

Addressing

The SAB-R2000A uses byte addressing, with alignment constraints for half-word and word accesses; half-word accesses must be aligned on an even byte boundary and word accesses must be aligned on a byte boundary divisible by four. Any attempt to address a data item that does not have the proper alignment causes an alignment exception. As said earlier special instructions are provided for addressing words that are not aligned.

The SAB-R2000A supports only one addressing mode – base register plus a signed 16-bit offset, which covers the most common case in High Level Languages and is very fast. The assembler, however, synthesizes some additional addressing modes to present more traditional addressing capabilities to the assembly language programmer.

Instruction Set Overview

All SAB-R2000A instructions consist of one 32-bit word. There are only three instruction formats (immediate, jump and register) as shown in figure 10.

The single instruction length simplifies instruction fetch and decode and eliminates the overhead for instructions crossing word and page boundaries within the memory hierarchy, thereby simplifying the interaction of instruction fetch with the virtual memory management unit. The three instruction formats ensure that opcodes and register descriptors are always found in the same bit locations. This enables register fetch to proceed in parallel with operation decode on all instructions, which is exactly what happens in the RD pipestage. More complicated (and less frequently used) operations can be synthesized by the compiler using sequences of simple instructions.

The SAB-R2000A instruction set can be divided into the following groups:

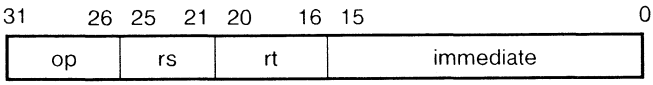
- **Load/Store** instructions move data between memory and general registers. They are all I-type instructions, since the only addressing mode supported is base register plus 16-bit, signed immediate offset. These are the only instructions which access memory.
- **Computational** instructions perform arithmetic, logical and shift operations on values in registers. They occur in both R-type (both operands and the result are registers) and I-type (one operand is a 16-bit immediate value) formats.
- **Jump and branch** instructions change the control flow of a program. Jumps are always to a paged absolute address formed by combining a 26-bit target with four bits of the Program Counter (J-type format, for subroutine calls) or 32-bit register addresses (R-type, for returns and dispatches). Branches have 16-bit offsets relative to the Program Counter (I-type). Jump and Link Instructions save a return address in Register 31.
- **Coprocessor** instructions perform operations in the coprocessors. Coprocessor Loads and Stores are I-type. Coprocessor computational instructions have co-processor-dependent formats (see SAB-R2010A data sheet).

- **Coprocessor 0** instructions perform operations on the System Control Coprocessor (CP0) registers to manipulate the memory management and exception handling facilities of the processor.
- **Special** instructions perform a variety of tasks, including movement of data between special and general registers, system calls and breakpoint. They are always R-type.

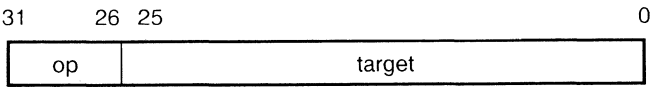
A detailed summary of the instruction set is provided in the Instruction Set Summary section.

Figure 10
Instruction Formats

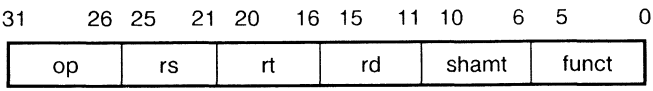
I-Type (Immediate)



J-Type (Jump)



R-Type (Register)



MPA00807

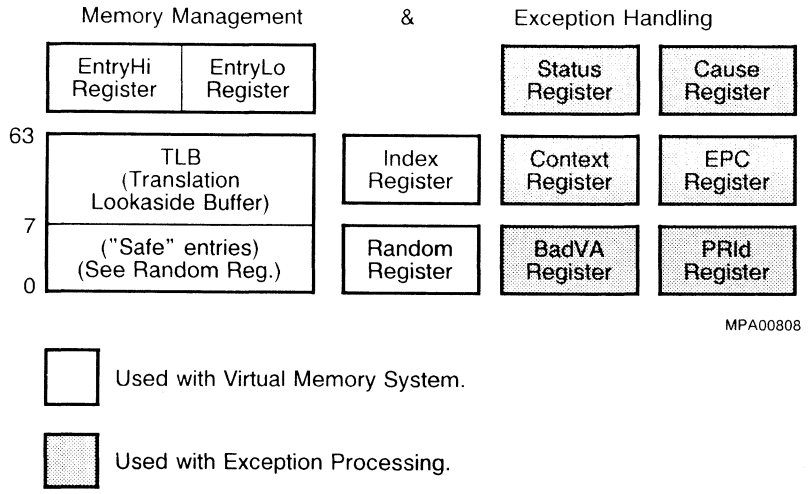
where:

op	:	is a 6-bit operation code
rs	:	is a 5-bit source register specifier
rt	:	is a 5-bit target (source/destination) register or branch condition
immediate	:	is a 16-bit immediate, branch displacement or address displacement
target	:	is a 26-bit jump target address
rd	:	is a 5-bit destination register specifier
shamt	:	is a 5-bit shift amount
funct	:	is a 6-bit function field

System Control Coprocessor

The SAB-R2000A CPU supports up to 4 coprocessors (designated CP0 through CP3), which are tightly coupled co-execution units that share a single instruction stream with the CPU. The System Control Coprocessor (CP0), is incorporated on the SAB-R2000A chip and supports the virtual memory system and exception handling functions (traps, interrupts, memory and internal operation faults) of the SAB-R2000A. CP0 incorporates a 64-entry TLB plus the registers shown in figure 11.

Figure 11
The CP0 Registers



The virtual memory system is implemented using a TLB and a group of programmable registers as shown in figure 11. The other registers shown are used to perform the exception handling capabilities. Table 1 provides a brief description of each register. In this table the number of each register is given. Logically the registers are numbered from 0 to 31. The numbers not contained in the table are unused.

Table 1
System Control Coprocessor (CP0) Registers

Register	Description	Number
EntryHi	High half of a TLB entry	10
EntryLo	Low half of a TLB entry	2
Index	Programmable pointer into TLB array	0
Random	Pseudo-random pointer into TLB array	1
Status	Mode, Interrupt enables, and diagnostic status info	12
Cause	Indicates nature of last exception	13
EPC	Exception Program Counter	14
Context	Pointer into kernel's virtual Page Table Entry array	4
BadVA	Most recent bad virtual address	8
PRId	Processor revision identification	15

Access to these registers and CP0 specific instructions (e.g. MTC0) may be restricted by setting CP0 to an "unusable" state (see status register). When the processor is executing in Kernel mode the usability of CP0 is ignored, i.e. it is always considered usable. This means that requests to manipulate CP0 from Kernel mode are always granted. However, it is possible for the Kernel to grant unrestricted access to CP0 registers by setting it to a usable state. Kernel and User modes are described in the next section.

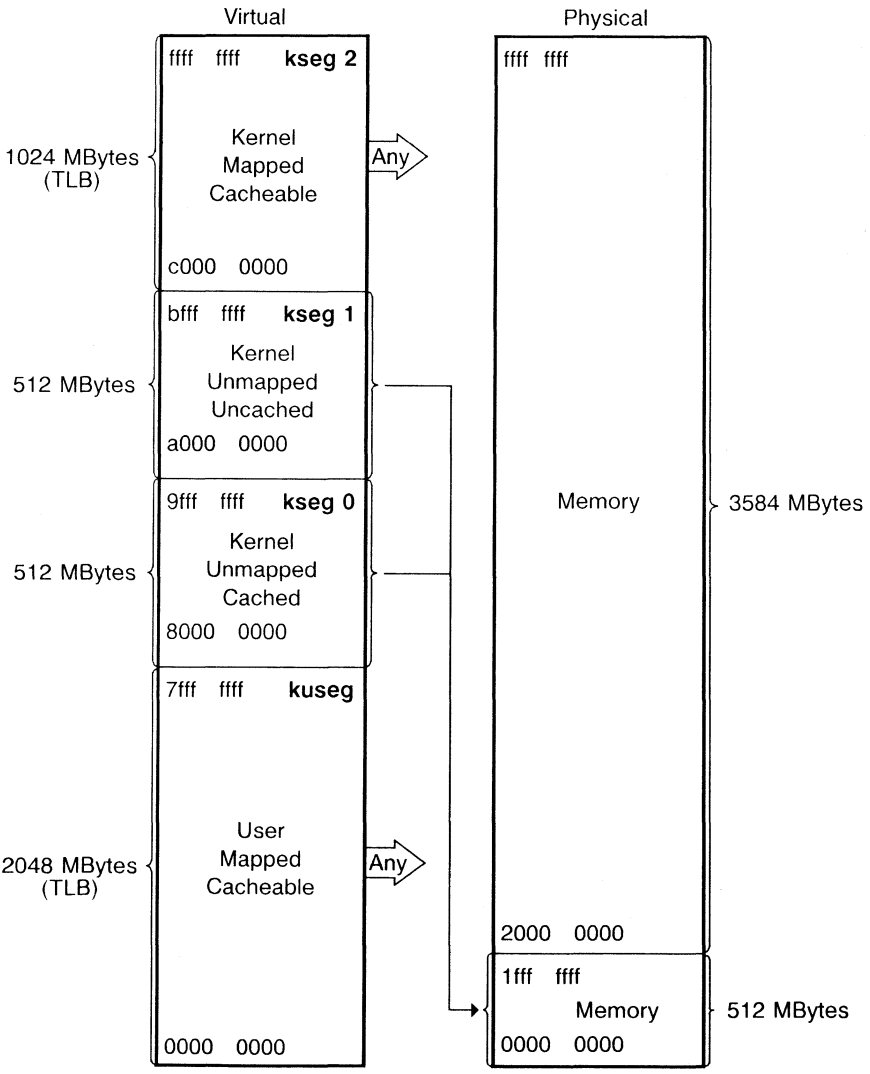
Memory Management System

The SAB-R2000A has an addressing range of 4 Gbytes. Since most systems implement physical memory sizes under 4 Gbytes, the SAB-R2000A's virtual memory system logically expands the available physical memory space by translating addresses composed in a large virtual address space into the physical memory space. All mapping is performed by the TLB in CP0. The TLB is fully associative and maps 64 4-Kbyte pages, sharable among user processes with minimal context-switch overhead, due to the PID (Process Identifier) number associated with each TLB entry.

Operating Modes

There are two operating modes in the SAB-R2000A, the Kernel mode and the User mode. The processor normally operates in User mode until an exception is detected forcing it into Kernel mode. It remains there until a Restore From Exception (rfe) instruction is executed.

Figure 12
Virtual Memory Map



Address mapping is different for Kernel and User modes. The User mode address space is a subset of the Kernel mode address space – i.e. the current User process owns a linear 2 Gbyte address space that is included in a 4 Gbyte Kernel address space. Figure 12 shows the virtual-to-physical memory map for both User mode and Kernel mode segments.

User Mode Virtual Addressing

When the processor is operating in User mode, a single, uniform virtual address space (kuseg) of 2 Gbytes is available to the current user process. All valid User mode virtual addresses have the most significant bit cleared to 0 – i.e. address references above 0x7FFFFFFF (0x means hex in a "C" type notation) cause an Address Error exception. All references to kuseg are mapped through the TLB, which also controls the cacheability of an access (the N-bit in a TLB entry). In Kernel mode, references to kuseg are treated just like User mode references, streamlining Kernel access to User data. Kuseg is typically used to hold all User code and data, and the current User process typically resides here, plus shared libraries in systems that have them.

Kernel Mode Virtual Addressing

In Kernel mode, three distinct virtual address spaces (in addition to kuseg) are available.

Kernel Cached, Unmapped – kseg0:

This segment is 512 Mbytes long starting at 0x80000000. References within kseg0 are direct-mapped onto the first 512 Mbytes of physical address space, use cache memory, but do not use TLB entries. That is to say that the physical address selected is defined by subtracting 0x80000000 from the virtual address (in order to directly map it into the first 512 Mbytes of physical memory). Typically some Kernel data and some of its executable code are kept here.

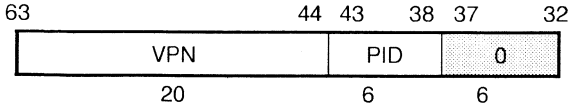
Kernel Uncached, Unmapped – kseg1:

Kernel segment kseg1 begins at 0xa0000000 and is also 512 Mbytes long. Like kseg0 it is direct-mapped onto the first 512 Mbytes of physical address space using no TLB entries. That is to say that the physical address selected is defined by subtracting 0xa0000000 from the virtual address. Unlike kseg0, it uses uncached references. An operating system typically uses kseg1 for I/O registers, ROM code and disk buffers.

Kernel Mapped – kseg2:
This segment is 1 Gbyte long, beginning at 0xc0000000. Like kuseg, it uses TLB entries to map virtual addresses to arbitrary physical ones, with or without caching (N-bit in a TLB entry). Operating systems typically use kseg2 for Kernel stacks andpre-process data that it must remap on context switches, for User page tables (memory maps), and some dynamically allocated data areas.

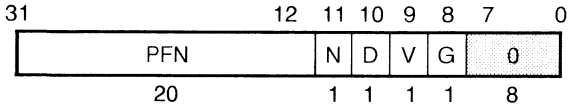
Figure 13
TLB EntryLo & EntryHi Registers

TLB EntryHi Register



VPN : Virtual Page Number. Bits 31. .12 of virtual address.
PID : Process ID field. A 6-bit field which lets multiple processes share the TLB while each process has a distinct mapping of otherwise identical virtual page numbers.
0 : Reserved. Currently ignores writes, returns zero when read.

TLB EntryLo Register



MPA00810

PFN : Page Frame Number. Bits 31. .12 of the physical address. The SAB-R2000A maps a virtual page to the PFN.
N : Non-cacheable. If this bit is set, the page is marked as non-cacheable and the SAB-R2000A directly accesses main memory instead of first accessing the cache.
D : Dirty. If this is set, the page is marked as "dirty" and therefore writable. This bit is actually a "write-protect" bit that software can use to prevent alteration of data. If an entry is accessed for a write operation when the D bit is cleared, the SAB-R2000A causes a TLB Mod trap. The TLB entry is not modified on such a trap.
V : Valid. If this bit is set, it indicates that the TLB entry is valid; otherwise, a TLBL or TLBS Miss occurs.
G : Global. If this bit it set, the SAB-R2000A ignores the PID match requirement for valid translation. In kseg2, the Global bit lets the kernel access all mapped data without requiring it to save or restore PID (Process ID) values.
0 : Reserved. Currently ignores writes, returns zero when read.

Virtual Memory Control

A high level description of the address space has been given – now the low-level details of the virtual memory system shall be discussed. The following registers which will be described are used for virtual memory control.

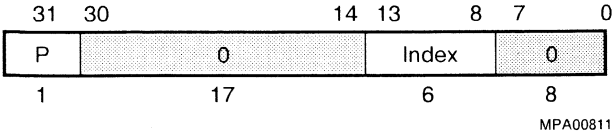
EntryHi and EntryLo:

This register pair and a single TLB entry have the same format, and so are described together. These two registers provide the data pathway (i.e are a buffer) through which the TLB is read, written or probed. When address translation exceptions occur, these registers are loaded with the relevant information about the address that caused the exception. EntryLo is the natural form of a Page Table Entry (PTE), however, since PTE's are always loaded by system software, not by the SAB-R2000A hardware, an operating system can use another format for memory resident PTE's. The register pair is illustrated in figure 12, a TLB entry is equivalent to the concatenation of these two registers.

Index Register:

The Index register is a 32-bit, read/write register, which has a 6-bit Index field. This field runs from 0 to 63, indexes an entry in the TLB and is used as a subscript to read or write any TLB entry. The high-order bit shows the success or failure of a TLB Probe (tlbp) Instruction. Figure 14 shows the format of the Index register.

Figure 14
Index Register



P : Probe failure. Set to 1 when the last TLBProbe (tlbp) instruction was unsuccessful.

Index : Index to the TLB entry that will be affected by the TLBRead and TLBWrite instructions.

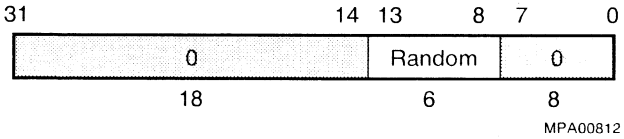
0 : Reserved. Currently ignores writes, returns zero when read.

Random Register:

The Random register is a 32-bit register, which has a 6-bit Random field which indexes a random entry in the TLB. The Random field is decremented every machine cycle, but is constrained to the range from 8 to 63. It is used to speed up pseudo-random TLB refill under Operating System control. The TLB Write Random (tlbwr) instruction is used to write the TLB entry that this register indexes. The first 8 entries (0 to 7) in the TLB are "safe" entries (see figure 11) because a "tlbwr" instruction can never replace the contents of these entries. Typically these 8 entries are reserved by the Operating System.

The contents of this register can be read to verify proper operation of the process (not normally required). To further simplify testing, the Random field is set to a value of 63 when the SAB-R2000A is reset. Figure 15 illustrates the Random register.

Figure 15
The Random Register



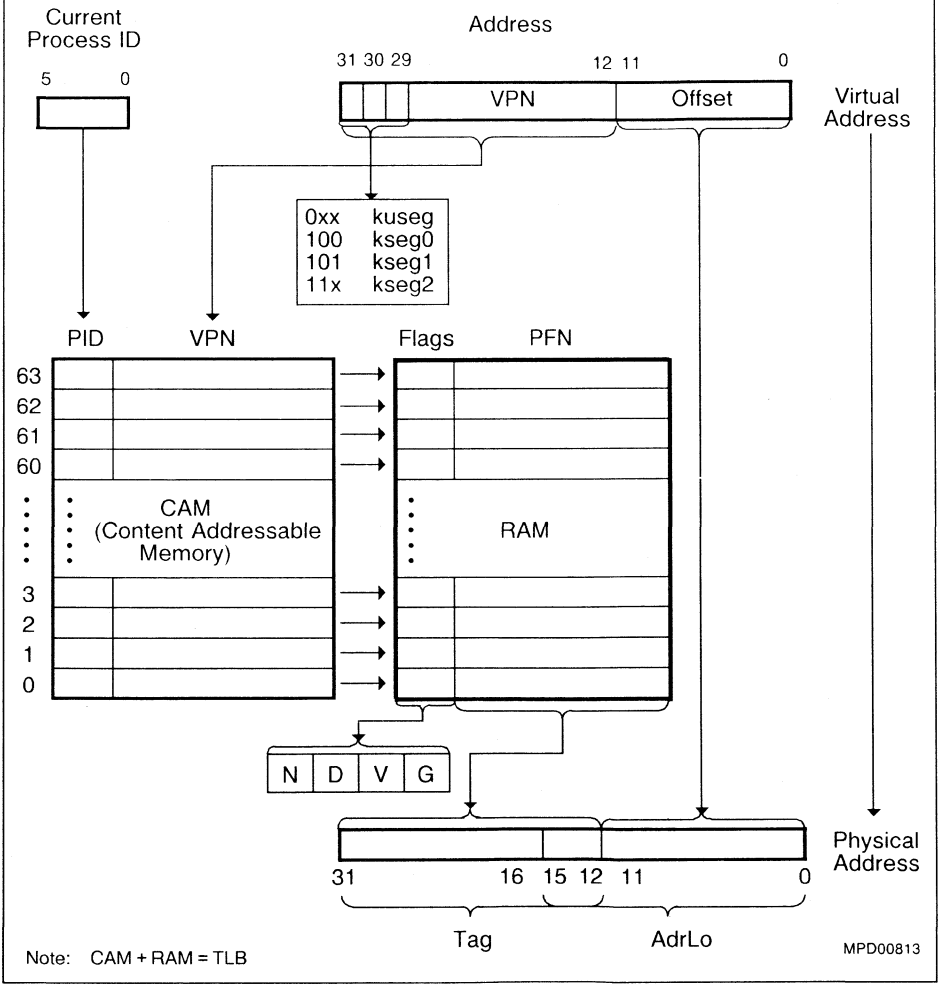
Random: A random index (with a value ranging from 8 to 63) to a TLB entry.

0 : Reserved. Currently ignores writes, returns zero when read.

Virtual Address Translation

During virtual-to-physical address translation, the processor compares the PID (Process Identification) number and the highest 20 bits (the VPN) of the virtual address to the contents of the TLB. Figure 16 illustrates the TLB address translation process.

Figure 16
TLB Operation



Note: CAM + RAM = TLB

A virtual address matches a TLB entry when:

- the Virtual Page Number (VPN) field of the virtual page address matches the VPN field of the TLB entry and,
- either the Global (G) bit of the TLB entry is set, or the PID field of the virtual address (as held in the EntryHi register) matches the PID field of the TLB entry.

While the Valid (V) bit of the TLB entry must be set for a valid translation to take place, it is not involved in the determination of a matching TLB entry.

If a TLB entry matches, the physical address and access control bits (N, D and V) are retrieved from the matching TLB entry. Otherwise a TLB miss (reference to kseg2), or a UTLB miss (reference to kuseg) exception occurs. If the V bit is not set, a TLB miss exception is taken. Finally, if the access is a Write and the Dirty (D) bit is not set, a TLB modification exception occurs. If the Non-cacheable (N) bit is set, the physical address that is retrieved is used to access main memory, bypassing the cache.

Exception Handling System

The term exception is used for any infrequent or exceptional event that causes the processor to make a temporary transfer of control from its current process to another process that services the event. There are two main classes of exceptions in the SAB-R2000A:

- machine exceptions, such as program traps, overflow and address translation exceptions.
- external asynchronous exceptions, which include six maskable external hardware interrupts, bus error and reset.

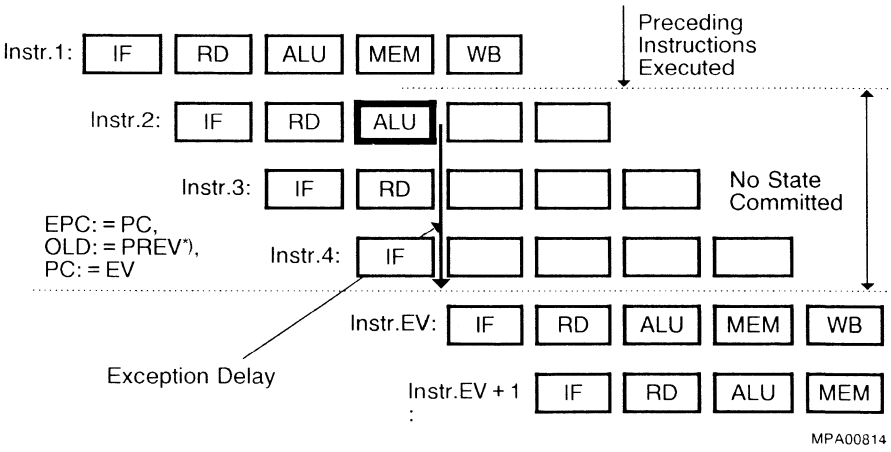
There are eight interrupts available, the hardware generates six and software two.

The exception handling system is responsible for efficiently handling relatively infrequent events, such as TLB misses, arithmetic overflow, I/O interrupts and system calls. On detection of an Exception the SAB-R2000A suspends the normal sequence of instruction execution; the processor exits User mode and is forced into Kernel mode where it can respond to the exceptional event. These events interrupt the normal execution flow by aborting the instruction which causes the exception and all those following in the pipeline which have already begun execution. The Exception Program Counter (EPC) is loaded with the appropriate restart location where execution should resume after the exception has been serviced. The restart location in the EPC is the address of the instruction which caused the exception or, if the instruction was executing in a branch delay slot, the address of the branch or jump instruction immediately preceding the delay slot. The SAB-R2000A then performs a direct jump into a designated handler routine.

A minimal amount of state is saved in the CP0 registers in order to facilitate the analysis of the cause of the exception, the servicing of the event which caused it, and the resumption of the original flow of execution.

The pipelined nature of the SAB-R2000A complicates the exception handling system. Exceptions which occur late in the pipeline effectively necessitate aborting instructions which have already begun execution but which logically should execute after the exception handling routine. Even taking this into account all SAB-R2000A exceptions are precise. That is, each exception is handled in a way that reflects serial completion of all instructions prior to the exception; the instruction which causes it and all those that follow are aborted and can be re-executed after servicing the exception. What this means (referring to figure 17) is that for example when an exception occurs in the ALU stage of Instr. 2, that Instr. 3 and 4, which have already started execution in the pipeline, do not alter the state of the machine so that execution may always properly resume after servicing the exception. Precise exception handling is shown in figure 17.

Figure 17
Precise Exception Handling

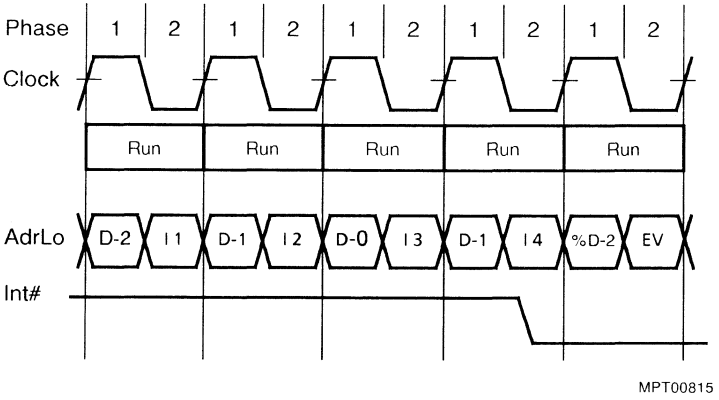


*) Refer to KU and IE bits of Status Register
EV: Exception Vector
EPC: Exception PC

Even though the machine is relatively deeply pipelined, exceptions are reported synchronously, so that all exceptions for a particular instruction are reported prior to exceptions for its succeeding instructions. Said another way, all exceptions are reported as if the processor was not pipelined.

There is only a one-cycle delay from when an exception occurs to when the exception handling routine is started (i.e. reaction time). This can be seen in figure 17, i.e. an exception occurs in the ALU cycle of Instr.2 (e.g. Overflow) and one cycle later the first instruction of the servicing routine is started (i.e. Instr.EV). This can be seen on the AdrLo bus for an Interrupt exception as is shown in figure 18. In the cycle after Int# (indicating an external interrupt exception) has been asserted the Exception Vector (EV) address is on the address bus.

Figure 18
Interrupt Exception Latency



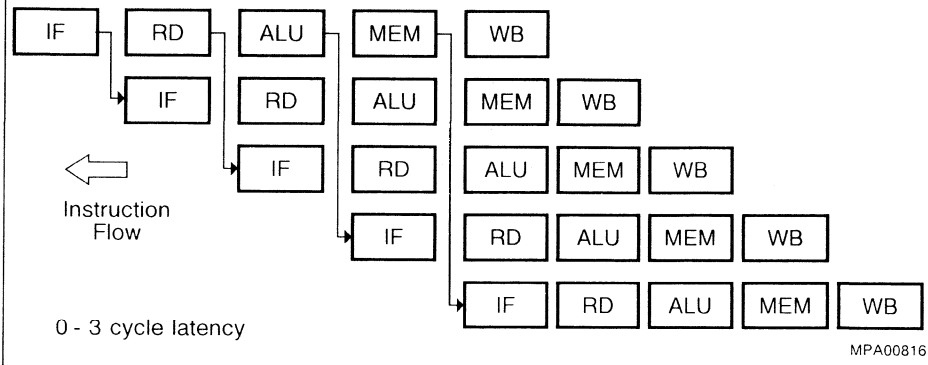
MPT00815

Where:

D-n	is the Data Transaction for Instr. n
In	is the Instruction Fetch for Instr. n
%	means an incorrect datum.
EV	Exception Vector

There is also an associated exception latency. This depends on how late the exception occurs in the pipeline. The exception latency can be 0-3 cycles in duration. It is the number of instructions which were aborted, that had already begun execution in the pipeline but which logically should execute after the exception handling routine – as discussed earlier. The concept of Exception latency is illustrated in figure 19.

Figure 19
Exception Latency



Exception Types

Table 2 lists each of the exception types which are handled by the processor, giving a short description of each.

There are only 3 exception vectors provided, one for Reset and one for UTLB miss exceptions (0xbfc00000 and 0x80000000 respectively), each of the remaining exceptions causes execution to resume at the General exception vector (0x80000080). When the BEV bit of the Status Register is set, the UTLB miss exception vector address is changed to 0xbfc00100 and the General exception vector address is changed to 0xbfc00180.

Table 2
SAB-R2000A Exceptions

Maskable	Exception	Mnemonic	Cause	Exception Vector
NO	Reset	Reset	Assertion of SAB-2000A's Reset signal causes an exception that transfers control to the special vector at virtual address 0xbfc00000.	0xbfc00000
	UTLB miss	UTLB	User TLB miss. A reference is made (in either User mode or Kernel mode) to a page in kuseg that has no matching TLB entry.	0x80000000 or 0xbfc00100
	TLB miss	TLBL (load) TLBS (store)	A referenced TLB entry's Valid bit isn't set or there is a reference to a *kseg2* page that has no matching TLB entry.	0x80000080 or 0xbfc00180
	TLB modified	Mod	During a store instruction, the Valid bit is set but the Dirty bit is not set.	
	Bus Error	IBE (Instruction) DBE (data)	Assertion of the SAB-R2000's BERR# signal due to such external events as bus timeout, backplane bus parity errors, invalid physical addresses or invalid access types.	
	Address Error	AdEL (load) AdES (store)	Attempt to load, fetch, or store an unaligned word; that is, a word or halfword at an address not evenly divisible by 4 or 2, respectively. Also caused by reference to a virtual address with most significant bit set while in User mode	
	Overflow	Ovf	Two's complement overflow during add or subtract.	
	System call	Sys	Execution of the syscall instruction.	
	Breakpoint	Bp	Execution of the break instruction.	
	Reserved Instruction	RI	Execution of an instruction with an undefined or reserved major operation code (bits 31..26), or a special instruction whose minor opcode (bits 5..0) is undefined.	
	Coprocessor Unusable	CpU	Execution of a coprocessor instruction when the CU (Coprocessor Usable) bit is not set for the target coprocessor.	
YES	Interrupt	Int	Assertion of one of the SAB-R2000A's six hardware interrupt inputs or setting of one of the two software interrupt bits in the Cause register.	

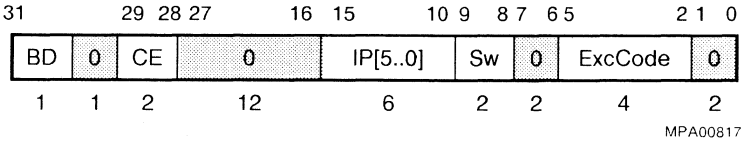
The Exception Handling Registers

The CP0 registers shown on the right of figure 11 contain information that is related to exception processing. Software can examine these registers during exception processing to determine such things as the cause of an exception, and the state of the CPU at the time of an exception. Two other registers, the Index register and the Random register, described in the Memory Management System section, may also contain useful information when handling exceptions related to virtual memory errors.

Cause Register:

The contents of this 32-bit register describe the last exception as shown in figure 20. The ExcCode field indicates the reason for the exception as listed in table 3. The remaining fields indicate pending external interrupts (IP), pending software interrupts (Sw), which, if any, coprocessor was found unusable (CE), and the occurrence of an exception in a branch delay slot (BD). All bits in the register, excluding the Sw bits, are read only. The Sw bits can be written into, to set or reset software interrupts.

Figure 20
Cause Register



BD : Branch Delay. Set to 1 if last exception was taken while executing in a branch delay slot.
CE : Coprocessor Error. Indicates the unit number referenced when a Coprocessor Unusable Exception is taken.
IP : Interrupts Pending. Indicates the external interrupts that are pending. IP[5..0] = Interrupt [5..0]
Sw : Software Interrupts. Indicates which of the two software interupts are pending. This field may be written into to set or reset software interrupts.
ExcCode : Exception Code field. Described in table 3.
0 : Reserved. Currently ignores writes, returns zero when read.

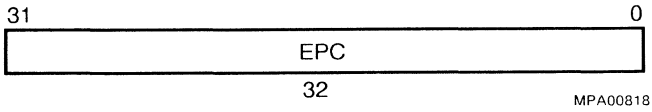
Table 3
Cause Register ExcCode Field

Number	Mnemonic	Description
0	Int	External interrupt
1	MOD	TLB modification exception
2	TLBL	TLB miss exception (Load or instruction fetch)
3	TLBS	TLB miss exception (Store)
4	AdEL	Address error exception (Load or instruction fetch)
5	AdES	Address error exception (Store)
6	IBE	Bus error exception (for an instruction fetch)
7	DBE	Bus error exception (for a data load or store)
8	Sys	Syscall exception
9	Bp	Breakpoint exception
10	RI	Reserved Instruction exception
11	CpU	Coprocessor Unusable exception
12	Ovf	Arithmetic overflow exception
13 – 15	-	reserved

Exception Program Counter Register (EPC):

On exception, this register records the address to where processing should be resumed after an exception has been serviced. The EPC register contains the virtual address of the instruction which was the cause of the exception; when that instruction is in a branch delay slot, the EPC contains the virtual address of the immediately preceding branch or jump instruction. The EPC register format is shown in figure 21.

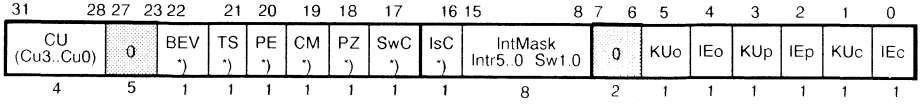
Figure 21
EPC Register



Status Register:

The Status register is a read/write register that contains the Kernel/User mode, interrupt enable and diagnostic state of the processor, i.e. it contains all major machine status bits. All bits in the Status register, excluding TS which is read only, are readable and writable. Figure 22 shows the format of the Status Register.

Figure 22
Status Register



MPA00819

*) Indicates primary use is for diagnostics and testing.

CU : Coprocessor Usability. These bits control usability of the four possible coprocessors: Cu3, Cu2, Cu1, and Cu0. If a CU bit is set (= 1), that coprocessor is usable.

*) BEV : Bootstrap Exception Vector. If set to 1, causes the SAB-R2000A to use the alternate, bootstrap vectors for UTLB Miss and general exceptions.

*) TS : TLB Shutdown. Set to 1 if SAB-R2000A has disabled TLB due to catastrophic error. Cleared only by Reset.

*) PE : Parity Error. Set to 1 if cache parity error occurs. Reset by writing a 1 to this bit.

*) CM : Cache Miss. Set to 1 if most recent D-Cache load resulted in a miss (only when the D-Cache is isolated).

*) PZ : Parity Zero. When set to 1, causes zero to replace normal outgoing parity bits.

*) SwC : Swap Caches. Controls switching of control signals for I-Cache and D-Cache.

*) IsC : Isolate Cache. When set to 1, isolates D-Cache from main memory system.

IntMask : Interrupt Mask. When a bit is set to 1, the corresponding hardware interrupt [Intr# 5..0] or software interrupt [Sw1..0] is enabled.

KUo : Kernel/User mode, old. Set to 0 if Kernel, 1 if User.

IEo : Interrupt Enable, old. Set to 1 to enable, 0 to disable.

KUp : Kernel/User mode, previous. Set to 0 if Kernel, 1 if User.

IEp : Interrupt Enable, previous. Set to 1 to enable, 0 to disable.

KUc : Kernel/User mode, current. Set to 0 if Kernel, 1 if User.

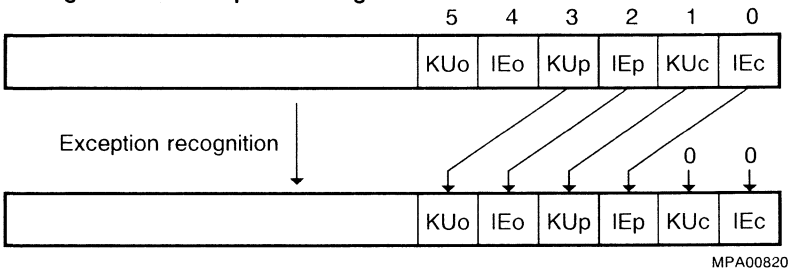
IEc : Interrupt Enable, current. Set to 1 to enable, 0 to disable.

0 : Reserved. Currently ignores writes, returns zero when read.

The Status register contains a three level stack (current, previous, and old) of the Kernel/ User mode bit (KU) and the Interrupt Enable (IE) bit.

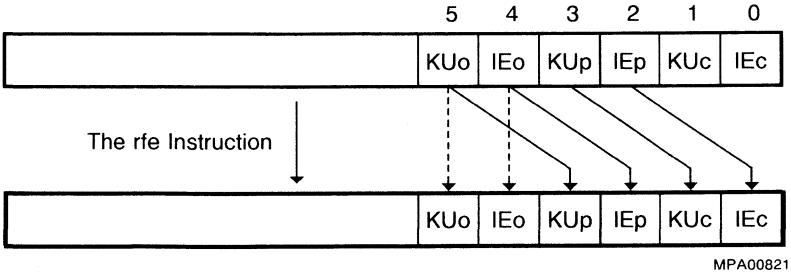
When an exception is taken the stack is pushed, i.e. the current Kernel/User mode (KUc) and current interrupt enable mode (IEc) bits are saved into the previous mode bits. The previous mode bits (KUp and IEp) are saved into the old mode bits (KUo and IEo). The current mode bits are cleared to cause the processor to enter Kernel mode and turn off interrupts. This three level set of mode bits allows the SAB-R2000A to respond to two levels of exceptions before software must save the contents of the Status register. figure 23 shows how the mode bits are pushed when an exception is taken.

Figure 23
Status Register and Exception Recognition



When an exception handler has completed execution the processor must return to the system context that existed prior to the exception. This is achieved by the Restore From Exception (rfe) instruction. The rfe instruction, when executed, pops the three level stack, i.e. the previous mode bits (KUp and IEp) are restored into the current mode bits (KUc and IEc). Likewise, the old mode bits (KUo and IEo) are restored into the previous mode bits. The old mode bits themselves remain unchanged. The actions of the rfe instruction are illustrated in figure 24.

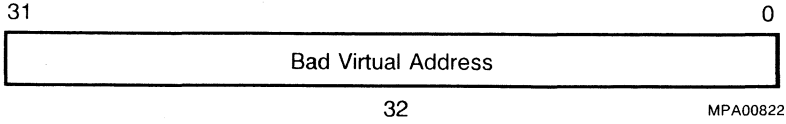
Figure 24
Restoring from Exceptions



Bad Virtual Address Register:

The BadVAddr register saves the entire bad virtual address for any addressing exception; AdeL or AdeS. Figure 25 shows this register format.

Figure 25
BadVAddr Register



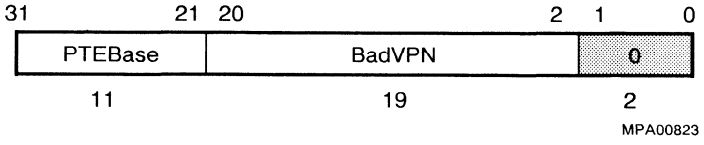
Context Register:

The Context register contains a pointer to the current user process's page map, located in kseg2 (kernel-mapped). It is designed for use in the UTLB miss handler, which loads TLB entries for normal user mode references.

The BadVPN field is not writable, it holds the VPN from the most recent virtual address for which the translation was invalid (i.e. an address exception). The 19-bit BadVPN field contains bits 30...12 (user segment Virtual Page Number) of the BadVAddr register. Bit 31 is excluded, because the UTLB miss handler is only invoked on user segment references whose highest virtual address is 0x7FFFFFFF.

The PTEBase field is writable as well as readable and indicates the base address of the page map of the current user address space. This register is implemented for the convenience of the Operating System. Figure 26 shows the format of the Context register.

Figure 26
Context Register



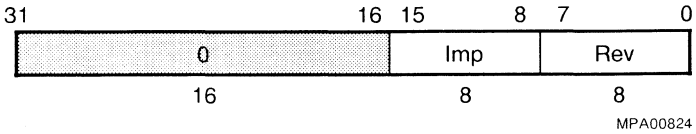
PTEBase : Holds the base for the Page Table Entry (set by software).
BadVPN : Holds the failing Virtual Page Number (set by hardware). This field is read-only and contains bits 30..12 (user-segment VPN) of the BadVAddr register.
0 : Unused; ignored on writes, zero when read.

Processor Revision Identifier Register:

This 32-bit read only register contains information that identifies the implementation and revision level of the Processor and System Control Coprocessor. The format of the register is shown in figure 27.

Figure 27
Processor Revision Identifier Register

PRId Register



Imp : Implementation identifier.
Rev : Revision identifier.
0 : Reserved. Currently ignores writes, returns zero when read.

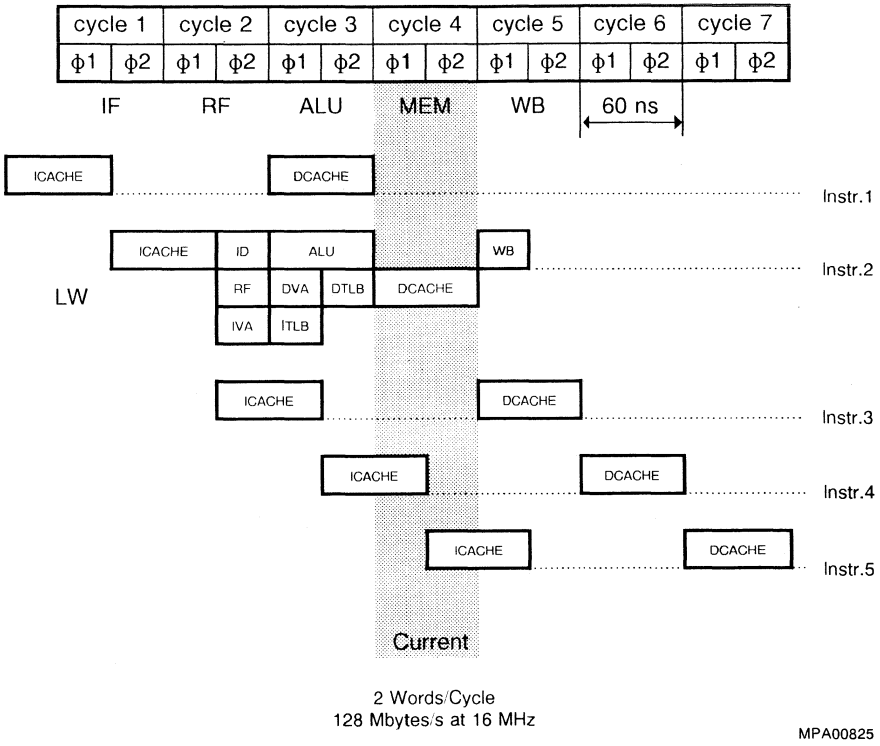
Memory System Hierarchy

The high performance capabilities of the SAB-R2000A processor places stringent demands on the memory system configuration. In order to achieve the goal of an instruction execution rate of one instruction per CPU cycle, the SAB-R2000A demands a memory bandwidth of 128 MBytes/second at 16 MHz from the memory system configuration. The memory system requirements can be seen in figure 28.

This high memory bandwidth is realized by a high performance memory hierarchy which centers on the use of external caches. Separate data and instruction caches are implemented, and the processor alternates accesses of the two caches during each CPU cycle - thus 2 words/CPU cycle, as shown in figure 28, are accessed. Both caches are physical as opposed to virtual, and ,may vary in size from 4 to 64 Kbytes each depending on the performance required, and are implemented using standard SRAM devices.

The update policy employed is a write-through policy, which simplifies the data consistency problem between cache and main memory. All data that is written to the data cache is also written out to main memory. Write buffers capture this data from the SAB-R2000A at CPU clock rates and then update main memory at its slower clock rate – therefore not stalling the processor. A simplified diagram of the high performance memory system is shown in figure 29.

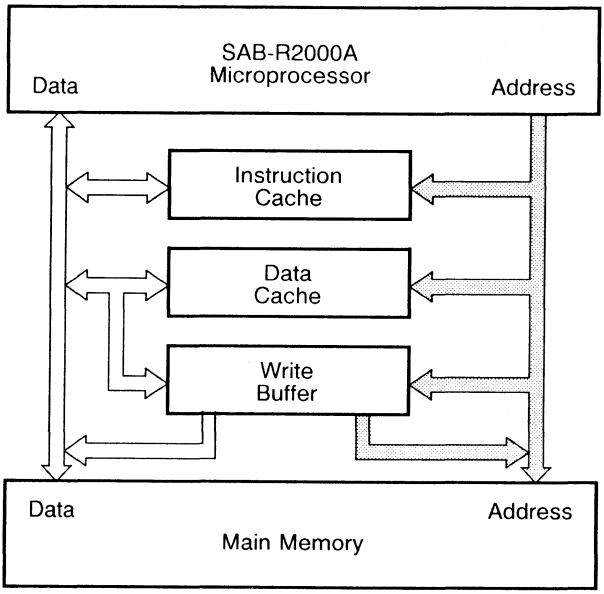
Figure 28
Memory Bandwidth



2 Words/Cycle
128 Mbytes/s at 16 MHz

MPA00825

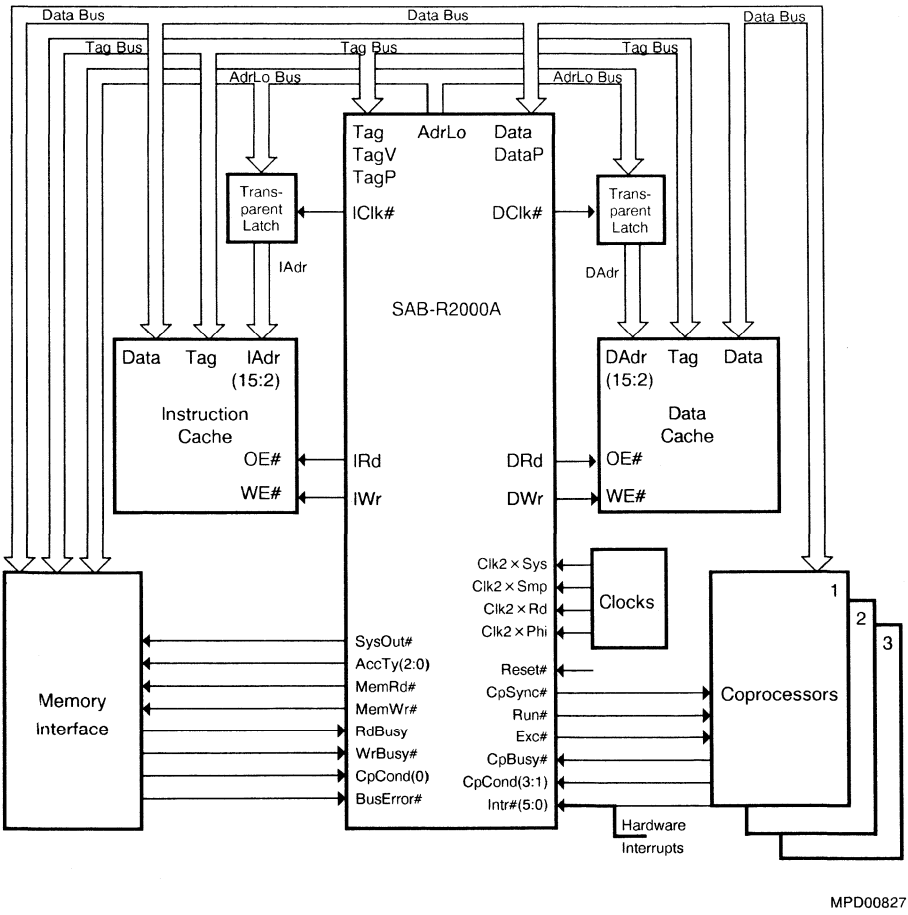
Figure 29
An SAB-R2000A System with a High-Performance Memory System



Processor Interface

The SAB-R2000A supports interfaces to external caches, main memory and coprocessors. Figure 30 illustrates the external interfaces of the SAB-R2000A processor.

Figure 30
Processor Interface

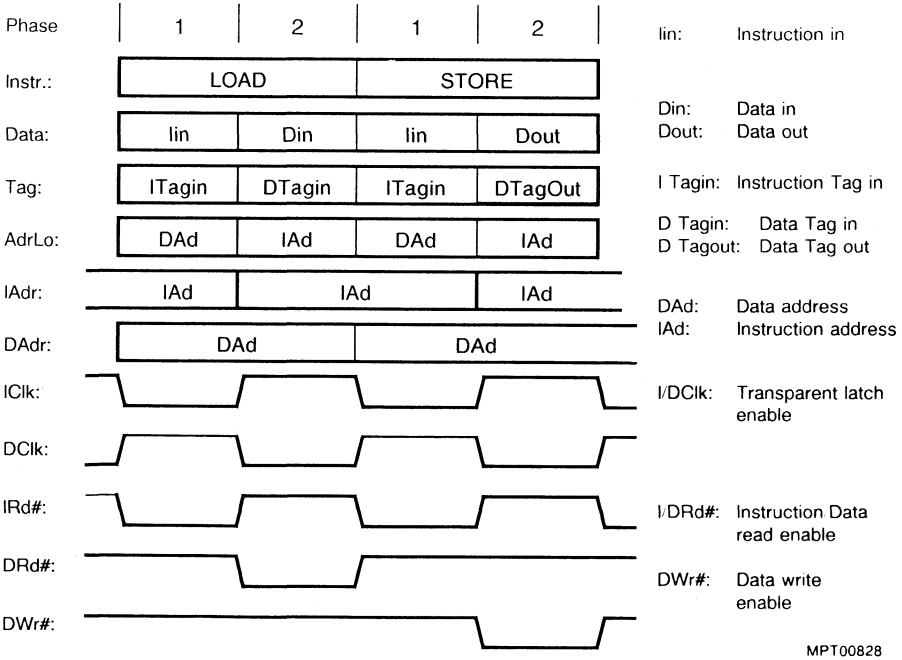


External Cache Interface

As described earlier the SAB-R2000A supports separate caches for instructions and data. As was seen in figure 3 the physical address coming out of the TLB is split across the external buses; AdrLo (16 bits) and TAG (20 bits plus valid and 3 parity bits). The caches are addressed by the 16 bit address bus AdrLo. Since AdrLo presents byte addresses and the caches are organised as words, its least significant 2 bits are not used by the caches. The most significant four bits of AdrLo bus are identical to the least significant four bits of the TAG bus but are output with AdrLo timing. This overlap allows cache size to vary with implementation (i.e. from 4 to 64 Kbytes).

The processor interleaves accesses to the two caches on the AdrLo, Tag and Data buses. Instruction fetch begins with AdrLo (IAd) clocked through a transparent latch by IClk# during phase 2 of a machine cycle, and continues until Data (Iin) and Tag (ITin) are latched on the chip at the end of the next phase 1. This is shown in figure 31. In the diagram the IAdr and DAdr buses are latched versions of AdrLo – refer also to figure 30.

Figure 31
Cache Interface Operation



Similarly, data fetch begins with AdrLo (DAd) clocked by DClk# during phase 1, and completes with Data (Din) and Tag (DTin) latched on the chip at the end of phase 2. During data stores, all three buses are outputs from the chip to the cache and memory interface. The AdrLo (DAd) is transmitted during phase 1 and the Data (Dout) and Tag (DTagout) during phase 2. The memory interface combines AdrLo and Tag to generate the full 32-bit real address for main memory access. Refer to figure 31 for details.

The cache interface integrates all circuitry that would normally be required between a processor and raw cache RAMS, such as the control lines for cache write and tristate output enables which are all generated on chip.

Note: Partial word stores such as Store byte and Store halfword are written directly to memory and the associated word in the cache is invalidated.

Memory Interface

The memory interface which is shown on the bottom left of figure 30 contains several signals which synchronize memory access events. MemRd# and MemWr# are asserted on cache miss (i.e. main memory read access initiated) and store respectively. The access type, i.e. byte, half word, tribyte, and word transfers are determined by the AccTy2:0 bits.

The principal supporting mechanism for main memory operations is the processor stall cycle. Main memory stalls occur when loads miss in the cache or when stores are blocked by the write buffer. RdBusy and WrBusy# control the termination and initiation of the stalls when the cache misses or the write buffer is full. BusError# warns of memory access errors such as parity error or bus timeout. The memory interface can also support system configurations where one or both caches are missing.

External Coprocessor Interface

The external coprocessor interface is illustrated on the bottom right of figure 30. It is designed to support the SAB-R2010A floating point accelerator, in what is called a tightly coupled interface, and up to two additional coprocessors. External coprocessors are connected to the Data bus only. During each cycle in which a valid Instruction-Data pair is on the bus, the coprocessors accept an Instruction. The coprocessors decode the Instruction in parallel with the main processor and, if it is a coprocessor Instruction, one of the coprocessors will proceed to execute the Instruction. A coprocessor condition (CpCond) signal, one for each coprocessor type, allows the main processor to branch on a coprocessor condition set up by a previous operation. Any coprocessor can assert CpBusy to stall the main CPU when a coprocessor instruction is issued while the coprocessor in question still has the required functional unit busy with an earlier operation. The SAB-R2000A asserts Run# to advance operations in the coprocessors. When Run# is deasserted in the n th cycle, coprocessors disregard the Instruction-Data pair presented in the n-1th cycle. The assertion of Exc# (Exception) indicates that the SAB-R2000A is taking an exception. CpSync# is used for timing synchronization between the SAB-R2000A and the coprocessor.

System Configuration

Due to the flexible interfaces of the SAB-R2000A it can be used in a variety of system configurations, ranging from high-end Workstations and parallel processors to low-end embedded control applications.

A high performance system configuration, which is suitable for high-end computing applications, is shown in figure 32.

The main components of this system, along with the SAB-R2000A, are separate Instruction and Data caches (64Kbyte each) and the SAB-R2010A Floating Point Accelerator. Main memory consists of DRAM. When a 16 MHz SAB-R2000A and SAB-R2010A are used in conjunction with 30ns SRAMs this system can deliver 12 VAX mips. Such system configurations are employed in high-end UNIX Workstations.

The SAB-R2000A is not only suited to high-end computing applications, it is also well suited to provide cost effective solutions for embedded control applications. The SAB-R2000A may be used to design systems with different degrees of performance. This can be achieved by varying the cache size (4 to 64 Kbyte each), the number of primary caches (0, 1 or 2), the I/O system configuration and the frequency the system will run at. There is a 512 Mbyte unmapped uncached region in the Address Space organisation (see the Memory Management System section) which can be used for a slow main memory interface when a cache is not implemented. There are also two other possibilities to implement a system without a cache

(a) cause a cache miss all the time; this can be achieved by having an external register, which upon cache access requests by the processor returns a deasserted Valid bit (i.e. invalid cache entry).

(b) Have the Operating System mark every page as uncacheable in the page table/TLB by setting the "N" (Non-cacheable) bit in the TLB entry. This should be done for both Kernel and User.

An example of a cost effective system configuration for a deterministic real time embedded application is illustrated in figure 33.

Figure 32
Solution for High-End-Computing Application

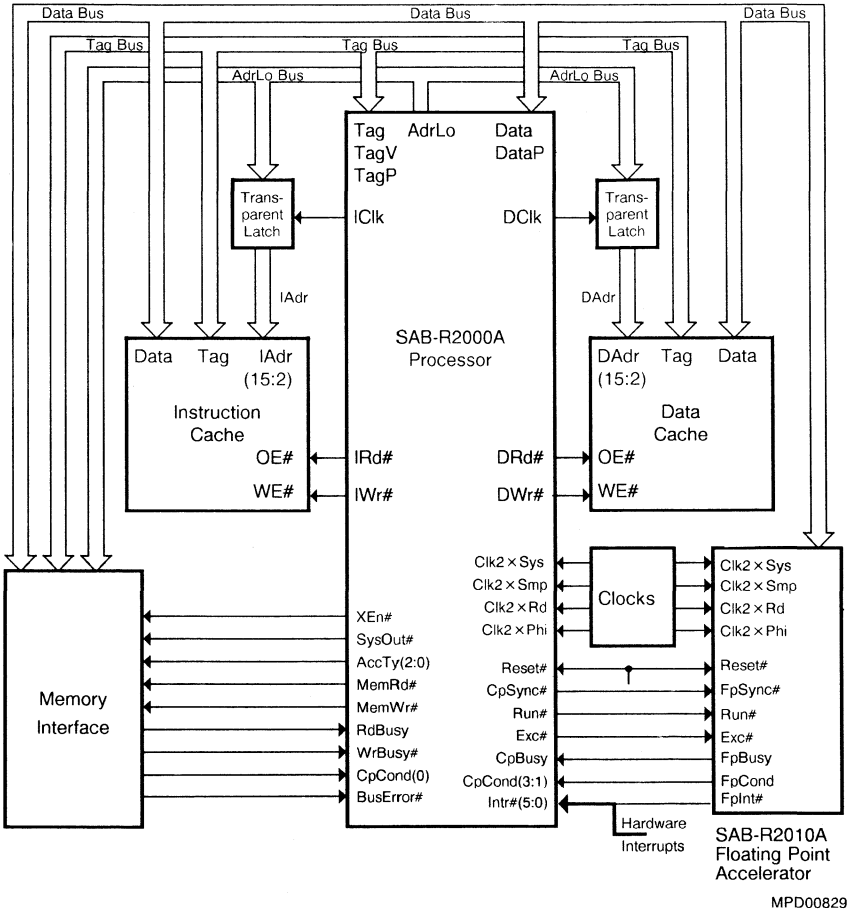
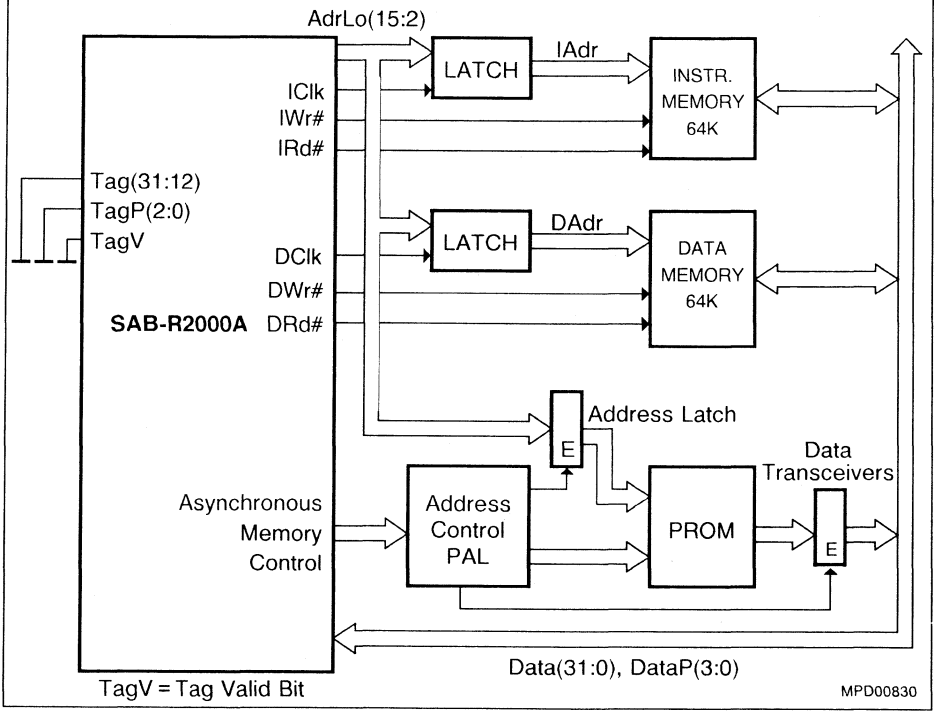


Figure 33
Solution for a Predictable Real Time Controller System



It is extremely difficult to apply the normal cache solution here due to the deterministic requirement. The SAB-R2000A can be configured so that a deterministic behaviour can be guaranteed. The technique employed is to use the synchronous bus (cache interface) to drive SRAMs where they preform the function of local main memory. The system configuration illustrated here is an example of a real time system with predictable responses of external and internal events, minimum context switch overhead and with a deterministic behaviour.

The system consists of a memory system with 64 Kbytes for Instructions and 64 Kbytes for Data (SRAMs). The asynchronous memory control (main memory interface) is used to address the PROM which contains the program to be loaded at initialization. With a 16 MHz SAB-R2000A and 20 ns SRAMs 10-12 VAX mips can be achieved. The worst case reaction time to an interrupt is 7 to 9 instructions, which consists of the system overhead until the user interrupt routine takes control. In this case, due to the fact that everything is in the SRAM's it takes only one cycle per instruction so the response time to an interrupt is 7 to 9 machine cycles (420-540 ns).

Instruction Set Summary

The following section is a table of the instructions available in the SAB-R2000A. The instructions are listed in alphabetical order. For a more detailed description of the operation of each instruction refer to the "SAB-R2000A Users Manual". A chart at the end of this section lists the bit encoding for the constant fields of each instruction.

Instruction Notation Convention

The table that follows is split up into three columns: Instruction, Format and Operation. The Instruction column contains the mnemonic name of the instruction and its meaning. The instruction format (refer to figure 10) and Assembly language notation, for each instruction, is listed in the Format column. The Operation column describes the operation performed by each instruction using a high level language notation. Special symbols used in the notation are described in table 4.

Table 4
SAB-R2000A Instruction Operation Notations

Symbol	Meaning
←	Assignment
‖	Bit string concatenation
x^y	Replication of bit value *x* into a *y*-bit string. Note that *x* is always a single-bit value.
$x_{y..z}$	Selection of bits *y* through *z* of bit string *x*. Little-endian bit notation is always used. If y is less than *z*, this expression is an empty (zero length) bit string.
+	Two's complement addition
–	Two's complement subtraction
*	Two's complement multiplication
div	Two's complement integer division
mod	Two's complement modulo
<	Two's complement less than comparison
and	Bitwise logic AND
or	Bitwise logic OR
xor	Bitwise logic XOR
nor	Bitwise logic NOR
GPR[x]	General Register *x*. The content of GPR[0] is always zero. Attempts to alter the content of GPR[0] have no effect
CPR[z,x]	Coprocessor unit *z*, general register *x*
CCR[z,x]	Coprocessor unit *z*, control register *x*
T + i	Indicates the time steps (CPU cycles) between operations. Thus, operations identified as occurring at T + 1 are performed during the cycle following the one where the instruction was initiated. This type of operation occurs with loads, stores, jumps, branches and coprocessor instructions.
vAddress	Virtual address
pAddress	Physical address

In the Load/Store operation descriptions, the functions listed in table 5 are used to summarize the handling of virtual addresses and physical memory.

Table 5
Load/Store Common Functions

Function	Desription
Addr Translation	Uses the TLB to find the physical address given the virtual address. The function fails and an exception is taken if the entry for the page containing the virtual address is not present in the TLB (Translation Lookaside Buffer).
Load Memory	Uses the cache and main memory to find the contents of the word containing the specified physical address. The low-order two bits of the address and the access type field indicate which of each of the four bytes within the data word need to be returned. If the cache is enabled for this access. The entire word is returned and loaded into the cache.
Store Memory	Uses the cache, write buffer, and main memory to store the word or part of word specified as data into the word containing the specified physical address. The low-order two bits of the address and the access type field indicate which of the four bytes within the data word should be stored.

Instruction	Format	Operation
ADD: Add	R-type; ADD rd, rs, rt	T: GPR[rd] ← GPR[rs] + GPR[rt]
ADDI: Add Immediate	I-type; ADDI rt, rs, immediate	T: GPR[rt] ← GPR[rs] + (immediate$_{15}$)^{16} ‖immediate$_{15..0}$
ADDIU: Add Immediate Unsigned	I-type; ADDIU rt, rs, immediate	T: GPR[rt] ← GPR[rs] + (immediate$_{15}$)^{16} ‖immediate$_{15..0}$
ADDU: Add Unsigned	R-type; ADDU rd, rs, rt	T: GPR[rd] ← GPR[rs] + GPR[rt]
AND: And	R-type; AND rd,rs,rt	T: GPR[rd] ← GPR[rs] and GPR[rt]
ANDI: And Immediate	I-type; ANDI rt, rs, immediate	T: GPR[rt] ← 0^{16} ‖(immediate and GPR[rs]$_{15..0}$)
BCzF: Branch On Coprocessor z False	I-type; BCzF offset	T: target ← (offset$_{15}$)^{14} ‖offset ‖0^2 condition ← not CpCond[z] T + 1: if condition then PC ← PC + target endif
BCzT: Branch On Coprocessor z True	I-type; BCzT offset	T: target ← (offset$_{15}$)^{14} ‖offset ‖0^2 condition ← CpCond[z] T + 1: if condition then PC ← PC + target endif
BEQ: Branch on Equal	I-type; BEQ rs, rt, offset	T: target ← (offset$_{15}$)^{14} ‖ offset ‖ 0^2 condition ← (GPR[rs] = GPR[rt]) T + 1: if condition then PC ← PC + target endif
BGEZ: Branch on Greater than or Equal to Zero	I-type; BGEZ rs, offset	T: target ← (offset$_{15}$)^{14} ‖ offset ‖ 0^2 condition ← (GPR[rs]$_{31}$ = 0) T + 1: if condition then PC ← PC + target endif

Instruction	Format	Operation
BGEZAL: Branch on Greater than or Equal to Zero And Link	I-type; BGEZAL rs, offset	T: target ← $(offset_{15})^{14} \| offset \| 0^2$ condition ← (GPR[rs] $_{31}$ = 0) GPR[31] ← PC + 8 T + 1: if condition then PC ← PC + target endif
BGTZ: Branch on Greater Than Zero	I-type; BGTZ rs, offset	T: target ← $(offset_{15})^{14} \| offset \| 0^2$ condition ← (GPR[rs] $_{31}$ = 0) and (GPR[rs] ≠ GPR[r0]) T + 1: if condition then PC ← PC + target endif
BLEZ: Branch on Less than or Equal to Zero	I-type; BLEZ rs, offset	T: target ← $(offset_{15})^{14} \| offset \| 0^2$ condition ← (GPR[rs] $_{31}$ = 1) or (GPR[rs] = GPR[r0]) T + 1: if condition then PC ← PC + target endif
BLTZ: Branch on Less Than Zero	I-type; BLTZ rs, offset	T: target ← $(offset_{15})^{14} \| offset \| 0^2$ condition ← (GPR[rs] $_{31}$ = 1) T + 1: if condition then PC ← PC + target endif
BLTZAL: Branch on Less Than Zero And Link	I-type; BLTZAL rs, offset	T: target ← $(offset_{15})^{14} \| offset \| 0^2$ condition ← (GPR[rs] $_{31}$ = 1) GPR[31] ← PC + 8 T + 1: if condition then PC ← PC + target endif
BNE: Branch on Not Equal	I-type; BNE rs, rt, offset	T: target ← $(offset_{15})^{14} \| offset \| 0^2$ condition ← (GPR[rs] ≠ GPR[rt]) T + 1: if condition then PC ← PC + target endif
BREAK: Break	R-type; BREAK	PC← Exception Handler

Instruction	Format	Operation
CFCz: Move Control From Co-processor z	R-type; CFCz rt, rd	T: data ← CCR[z,rd] T + 1: GPR[rt] ← data
COPz: Coprocessor Operation z	Coprocessor type: COPz cofun	T: CoprocessorOperation (z. cofun)
CTCz: Move Control To Coprocessor z	R-type; CTCz	T: data ← GPR[rt] T + 1: CCR[z,rd] ← data
DIV: Divide	R-type; DIV rs, rt	T-2: LO ← undefined HI ← undefined T–1: LO ← undefined HI ← undefined T: LO ← GPR[rs] *div* GPR[rt] HI ← GPR[rs] *mod* GPR[rt]
DIVU: Divide Unsigned	R-type; DIVU rs, rt	T-2: LO ← undefined HI ← undefined T–1: LO ← undefined HI ← undefined T: LO ← (0‖GPR[rs]) *div* (0‖GPR[rt]) HI ← (0‖GPR[rs]) *mod* (0‖GPR[rt])
J: Jump	J-type; J target	T: temp ← $PC_{31..28}$ ‖ target ‖ 0^2 T + 1: PC ← temp
JAL: Jump And Link	J-type; JAL target	T: temp ← $PC_{31..28}$ ‖ target ‖ 0^2 GPR[31] ← PC + 8 T + 1: PC ← temp
JALR: Jump And Link Register	R-type; JALR rs JALR rd, rs	T: temp ← GPR[rs] GPR[rd] ← PC + 8 T + 1: PC ← temp
JR: Jump Register	R-type; JR rs	T: temp ← GPR[rs] T + 1: PC ← temp

Instruction	Format	Operation

LB:
Load Byte
	I-type;
	LB rt, offset (base)
		T: vAddress←$(offset_{15})^{16} \| offset_{15..0}$ + GPR[base]
		(pAddress, nonCacheable)← AddrTranslation (vAddress)
		mem←LoadMemory (nonCacheable, BYTE, pAddress)
		byte←$vAddress_{1..0}$
		T + 1: if BigEndian then
		GPR[rt]←$(mem_{31-8*byte})^{24} \| mem_{31-8*byte..24-8*byte}$
		else
		GPR[rt]←$(mem_{7+8*byte})^{24} \| mem_{7+8*byte..8*byte}$
		endif

LBU:
Load Byte
Unsigned
	I-type;
	LBU rt, offset (base)
		T: vAddress←$(offset_{15})^{16} \| offset_{15..0}$ + GPR[base]
		(pAddress, nonCacheable)← AddrTranslation (vAddress)
		mem←LoadMemory (nonCacheable, BYTE, pAddress)
		byte←$vAddress_{1..0}$
		T + 1: if BigEndian then
		GPR[rt]←$0^{24} \| mem_{31-8*byte..24-8*byte}$
		else
		GPR[rt]←$0^{24} \| mem_{7+8*byte..8*byte}$
		endif

LH:
Load Halfword
	I-type;
	LH rt, offset (base)
		T: vAddress←$(offset_{15})^{16} \| offset_{15..0}$ + GPR[base]
		(pAddress, nonCacheable)← AddrTranslation (vAddress)
		mem←LoadMemory(nonCacheable,HALFWORD,pAddress)
		byte←$vAddress_{1..0}$
		T + 1: if BigEndian then
		GPR[rt]←$(mem_{31-8*byte})^{16} \| mem_{31-8*byte..16-8*byte}$
		else
		GPR[rt]← $(mem_{15+8*byte})^{16} \| mem_{15+8*byte..8*byte}$
		endif

Instruction	Format	Operation

LHU:
Load Halfword Unsigned
	I-type;
	LHU rt, offset (base)
		T: vAddress←(offset$_{15}$)^{16} ‖ offset$_{15..0}$ + GPR[base]
		(pAddress, nonCacheable)← AddrTranslation (vAddress)
		mem←LoadMemory(nonCacheable, HALFWORD,pAddress)
		byte←vAddress$_{1..0}$
		T + 1: if BigEndian then
		GPR[rt]←0^{16} ‖ mem$_{31-8*byte..16-8*byte}$
		else
		GPR[rt]←0^{16} ‖ mem$_{15+8*byte..8*byte}$
		endif

LUI: Load Upper Immediate	I-type; LUI rt, immediate	T: GPR[rt]←immediate ‖ 0^{16}
LW: Load Word	I-type; LW rt, offset (base)	T: vAddress←(offset$_{15}$)^{16} ‖ offset$_{15..0}$ + GPR[base] (pAddress, nonCacheable)← AddrTranslation (vAddress) mem←LoadMemory (nonCacheable,WORD, pAddress) byte←vAddress$_{1..0}$ T + 1: GPR[rt]←mem;
LWCz: Load Word to Coprocessor z	I-type; LWCz rt, offset (base)	T: vAddress←(offset$_{15}$)^{16} ‖offset$_{15..0}$ + GPR[base] (pAddress, nonCacheable)← AddrTranslation (vAddress) mem←LoadMemory(nonCacheable,WORD,pAddress$_{31..2}$ ‖0^{2}) byte←vAddress$_{1..0}$ T + 1: CPR[z,rt]←mem;

Instruction	Format	Operation

LWL:
Load Word Left
	I-type;
	LWL rt, offset (base)
		T: vAddress←$(offset_{15})^{16} \| offset_{15..0}$ + GPR[base]
		(pAddress, nonCacheable)←AddrTranslation (vAddress)
		byte←$vAddress_{1..0}$
		if BigEndian then
		mem←LoadMemory(nonCacheable,WORD-byte,pAddress)
		else
		mem←LoadMemory(nonCacheable,byte,$pAddress_{31..2} \| 0^2$)
		endif
		T + 1: if BigEndian then
		GPR[rt]←$mem_{31-8*byte..0} \| GPR[rt]_{8*byte-1..0}$
		else
		GPR[rt]←$mem_{7+8*byte..0} \| GPR[rt]_{23-8*byte..0}$
		endif

LWR:
Load Word Right
	I-type;
	LWR rt, offset (base)
		T: vAddress←$(offset_{15})^{16} \| offset_{15..0}$ + GPR[base]
		(pAddress, nonCacheable)←AddrTranslation (vAddress)
		byte←$vAddress_{1..0}$
		if BigEndian then
		mem←LoadMemory(nonCacheable,byte,$pAddress_{31..2} \| 0^2$)
		else
		mem←LoadMemory(nonCacheable,byte,WORD-byte,pAddress)
		endif
		T + 1: if BigEndian then
		GPR[rt]←$GPR[rt]_{31..8+8*byte} \| mem_{31..24-8*byte}$
		else
		GPR[rt]←$GPR[rt]_{31..24-8*byte} \| mem_{31..8+8*byte}$
		endif

MFC0: Move From System Control Coprocessor z	R-type; MFC0 rt, rd	T: data CPR[0,rd] T + 1: GPR[rt] ← data
MFCz: Move From Coprocessor z	R-type; MFCz rt, rd	T: data ← CPR[z,rd] T + 1: GPR[rt] ← data

Instruction	Format	Operation
MFHI: Move From HI	R-type; MFHI rd	T: GPR[rd] ← HI
MFLO: Move From LO	R-type; MFLO, rd	T: GPR[rd] ← LO
MTC0: Move To System Control Coprocessor	R-type; MTC0 rt, rd	T: data ← GPR[rt] T + 1: CPR[0,rd] ← data
MTCz: Move To Coprocessor z	R-type; MTCz rt, rd	T: data ← GPR[rt] T + 1: CPR[z,rd] ← data
MTHI: Move To HI	R-type; MTHI rs	T–2: HI ← undefined T–1: HI ← undefined T: HI ← GPR[rs]
MTLO: Move To LO	R-type; MTLO rs	T–2: LO ← undefined T–1: LO ← undefined T: LO ← GPR[rs]
MULT: Multiply	R-type; MULT rs,rt	T–2: LO ← undefined HI ← undefined T–1: LO ← undefined HI ← undefined T: t ← GPR[rs]*GPR[rt] LO ← $t_{31..0}$ HI ← $t_{63..32}$
MULTU: Multiply Unsigned	R-type; MULTU rs,rt	T–2: LO ← undefined HI ← undefined T–1: LO ← undefined HI ← undefined T: t ← (0 ‖GPR[rs])*(0 ‖GPR[rt]) LO ← $t_{31..0}$ HI ← $t_{63..32}$
NOR: Nor	R-type; NOR rd, rs, rt	T: GPR[rd] ← GPR[rs] *nor* GPR[rt]

Instruction	Format	Operation
OR: Or	R-type; OR rd, rs, rt	T: GPR[rd] ← GPR[rs] *or* GPR[rt]
ORI: Or Immediate	I-type; ORI rt, rs, immediate	T: GPR[rt] ← GPR[rs]$_{31..16}$ ‖(immediate *or* GPR[rs]$_{15..0}$)
RFE: Restore From Exception	R-type; RFE	T: SR ← SR$_{31..4}$ ‖ SR$_{5...2}$

SB:
Store Byte
	I-type;
	SB rt, offset (base)
		T: vAddress←(offset$_{15}$)^{16} ‖ offset$_{15..0}$ + GPR[base]
		(pAddress, nonCacheable)← AddrTranslation (vAddress)
		byte←vAddress$_{1..0}$
		if BigEndian then
		data←GPR[rt]$_{7+8*byte..0}$ ‖ $0^{24-8*byte}$
		else
		data←GPR[rt]$_{31-8*byte..0}$ ‖ 0^{8*byte}
		endif
		T + 1: StoreMemory (nonCacheable, BYTE, data, pAddress)

SH:
Store Halfword
	I-type;
	SH rt, offset (base)
		T: vAddress←(offset$_{15}$)^{16} ‖ offset$_{15..0}$ + GPR[base]
		(pAddress, nonCacheable)← AddrTranslation (vAddress)
		byte←vAddress$_{1..0}$
		IF BigEndian then
		data←GPR[rt]$_{15+8*byte..0}$ ‖ $0^{15-8*byte}$
		else
		data←GPR[rt]$_{31-8*byte..0}$ ‖ 0^{8*byte}
		endif
		T + 1: StoreMemory (nonCacheable, HALFWORD, data, pAddress)

SLL: Shift Left Logical	I-type; SLL rd, rt shamt	T: GPR[rd] ← GPR[rt]$_{31-shamt..0}$ ‖ 0^{shamt}
SLLV: Shift Left Logical Variable	R-type; SLLV rd, rt, rs	T: GPR[rd] ← GPR[rt]$_{(31-GPR[rs]_{4..0})..0}$ ‖ $0^{GPR[rs]_{4..0}}$

Instruction	Format	Operation

SLT:
Set on Less Than
	R-type;
	SLT rd, rs, rt
		T: if GPR[rs] < GPR[rt] then
		GPR[rd] ← $0^{31} \| 1$
		else
		GPR[rd] ← 0^{32}
		endif

SLTI:
Set on Less Than Immediate
	I-type;
	SLTI rt, rs, immediate
		T: if GPR[rs] < $((immediate_{15})^{16} \| immediate_{15..0})$ then
		GPR[rt] ← $0^{31} \| 1$
		else
		GPR[rt] ← 0^{32}
		endif

SLTIU:
Set on Less Than Immediate Unsigned
	I-type;
	SLTIU rt, rs, immediate
		T: if $(0 \| GPR[rs]) < (0 \| (immediate_{15})^{16} \| immediate_{15..0})$ then
		GPR[rt] ← $0^{31} \| 1$
		else
		GPR[rt] ← 0^{32}
		endif

SLTU:
Set on Less Than Unsigned
	R-type;
	SLTU rd, rs, rt
		T: if $(0 \| GPR[rs]) < (0 \| GPR[rt])$ then
		GPR[rd] ← $0^{31} \| 1$
		else
		GPR[rd] ← 0^{32}
		endif

SRA: Shift Right Arithmetic	R-type; SRA rd, rt, shamt	T: $GPR[rd] \leftarrow (GPR[rt]_{31})^{shamt} \| GPR[rt]_{31..shamt}$
SRAV: Shift Right Arithmetic Variable	R-type; SRAV rd, rt, rs	T: $GPR[rd] \leftarrow (GPR[rt]_{31})^{GPR[rs]4..0} \| GPR[rt]_{31..(GPR[rs]_{4..0})}$
SRL: Shift Right Logical	R-type; SRL rd, rt, shamt	T: $GPR[rd] \leftarrow 0^{shamt} \| GPR[rt]_{31..shamt}$
SRLV: Shift Right Logical Variable	R-type; SRLV rd, rt, rs	T: $GPR[rd] \leftarrow 0^{GPR[rs]4..0} \| GPR[rt]_{31..(GPR[rs]_{4..0})}$

Instruction	Format	Operation
SUBU: Subtract Unsigned	R-type; SUBU rd, rs, rt	T: GPR[rd] ← GPR[rs] –GPR[rt]
SUB: Subtract	R-type; SUB rd, rs, rt	T: GPR[rd] ← GPR[rs] –GPR[rt]
SW: Store Word	I-type; SW rt, offset (base)	T: vAddress←$(offset_{15})^{16} \| offset_{15..0}$ + GPR[base] (pAddress, nonCacheable)← AddrTranslation (vAddress) data←GPR[rt] T + 1: StoreMemory (nonCacheable, WORD, data, pAddress)
SWCz: Store Word from Coprocessor z	I-type; SWCz rt, offset (base)	T: vAddress←$(offset_{15})^{16} \| offset_{15..0}$ + GPR[base] (pAddress, nonCacheable)← AddrTranslation (vAddress) data←CPR[z,t] T + 1: StoreMemory (nonCacheable, 15, data, $pAddress_{31..2} \| 0^2$)

SWL:
Store Word Left
	I-type;
	SWL rt, offset (base)
		T: vAddress←$(offset_{15})^{16} \| offset_{15..0}$ + GPR[base]
		(pAddress,nonCacheable)← AddrTranslation (vAddress)
		byte←$vAddress_{1..0}$
		if BigEndian then
		data←$0^{8*byte} \| GPR[rt]_{31..8*byte}$
		else
		data← $0^{24-8*byte} \| GPR[rt]_{31..24-8*byte}$
		endif
		T + 1: if BigEndian then
		StoreMemory (nonCacheable,WORD-byte,data,pAddress)
		else
		StoreMemory (nonCacheable,byte,data,$pAddress_{31..2} \| 0^2$)
		endif

Instruction	Format	Operation

SWR:
Store Word Right
	I-type;
	SWR rt, offset (base)
		T: vAddress←(offset$_{15}$)^{16} ‖ offset$_{15..0}$ + GPR[base]
		(pAddress, nonCacheable)← AddrTranslation (vAddress)
		byte←vAddress$_{1..0}$
		if BigEndian then
		data←GPR[rt]$_{7+8*byte..0}$ ‖ 0$^{24-8*byte}$
		else
		data← GPR[rt]$_{31-8*byte..0}$ ‖ 0^{8*byte}
		endif
		T + 1: if BigEndian then
		StoreMemory (nonCacheable,byte,data, pAddress$_{31..2}$ ‖ 0^2)
		else
		StoreMemory (nonCacheable,WORD-byte,data,pAddress)
		endif

SYSCALL: System Call	R-type; SYSCALL	PC ← ExceptionHandler

TLBP:
Probe TLB for matching entry
	R-type;
	TLBP
		T: Index← 1 ‖ 0^{31}
		for i in 0..TLBEntries-1
		if ((TLB$_{63..44}$[i] = EntryHi$_{31..12}$) and
		(TLB$_8$ or (TLB$_{43..38}$ = EntryHi$_{11..6}$))) then
		Index← 0^{18} ‖ i$_{5..0}$ ‖ 0^8
		endif
		endfor

TLBR: Read indexed TLB entry	R-type; TLBR	T: EntryHi← TLB [Index$_{13..8}$]$_{63..32}$ EntryLo← TLB [Index$_{13..8}$]$_{31..0}$
TLBWI: Write Indexed TLB entry	R-type; TLBWI	T: TLB [Index$_{13..8}$]$_{63..32}$← EntryHi TLB [Index$_{13..8}$]$_{31..0}$← EntryLo
TLBWR: Write Random TLB entry	R-type; TLBWR	T: TLB[Random$_{13..8}$]$_{63..32}$← EntryHi TLB[Random$_{13..8}$]$_{31..0}$← EntryLo
XOR: Exclusive Or	R-type; XOR rd, rs, rt	T: GPR[rd]← GPR[rs] *xor* GPR[rt]
XORI: Exclusive Or Immediate	I-type; XORI rt, rs, immediate	T: GPR[rt]← GPR[rs]$_{31..16}$ ‖ (immediate *xor* GPR[rs]$_{15..0}$)

Instruction Encoding

op

31..29 \ 28..26	0	1	2	3	4	5	6	7
0	SPECIAL	REGIMM	J	JAL	BEQ	BNE	BLEZ	BGTZ
1	ADDI	ADDIU	SLTI	SLTIU	ANDI	ORI	XORI	LUI
2	COP0	COP1	COP2	COP3	⊗	⊗	⊗	⊗
3	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗
4	LB	LH	LWL	LW	LBU	LHU	LWR	⊗
5	SB	SH	SWL	SW	⊗	⊗	SWR	⊗
6	⊗	LWC1	LWC2	LWC3	⊗	⊗	⊗	⊗
7	⊗	SWC1	SWC2	SWC3	⊗	⊗	⊗	⊗

SPECIAL function

5..3 \ 2..0	0	1	2	3	4	5	6	7
0	SLL	⊗	SRL	SRA	SLLV	⊗	SRLV	SRAV
1	JR	JARL	⊗	⊗	SYSCALL	BREAK	⊗	⊗
2	MFHI	MTHI	MFLO	MTLO	⊗	⊗	⊗	⊗
3	MULT	MULTU	DIV	DIVU	⊗	⊗	⊗	⊗
4	ADD	ADDU	SUB	SUBU	AND	OR	XOR	NOR
5	⊗	⊗	SLT	SLTU	⊗	⊗	⊗	⊗
6	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗
7	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗

REGIMM rt

20..19 \ 18..16	0	1	2	3	4	5	6	7
0	BLTZ	BGEZ	~	~	~	~	~	~
1	~	~	~	~	~	~	~	~
2	BLTZAL	BGEZAL	~	~	~	~	~	~
3	~	~	~	~	~	~	~	~

⊗ Codes marked with a '⊗' cause unimplemented instruction exceptions and are reserved for future versions of the architecture.

~ Codes marked with a '~' are not valid and are reserved for future versions of the architecture. The results of such an encoding are undefined.

COPz rs

25..24 \ 23..21	0	1	2	3	4	5	6	7
0	MF	~	CF	~	MT	~	CT	~
1	~	⊗	⊗	⊗	⊗	⊗	⊗	⊗
2	CO							
3								

COPz rt

20..19 \ 18..16	0	1	2	3	4	5	6	7
0	BCF	BCT	~	~	~	~	~	~
1	~	~	~	~	~	~	~	~
2	~	~	~	~	~	~	~	~
3	~	~	~	~	~	~	~	~

COP0 function

4..3 \ 2..0	0	1	2	3	4	5	6	7
0	~	TLBR	TLBWI	~	~	~	TLBWR	~
1	TLBP	~	~	~	~	~	~	~
2	RFE	~	~	~	~	~	~	~
3	~	~	~	~	~	~	~	~
4	~	~	~	~	~	~	~	~
5	~	~	~	~	~	~	~	~
6	~	~	~	~	~	~	~	~
7	~	~	~	~	~	~	~	~

⊗ Codes marked with a '⊗' cause unimplemented instruction exceptions and are reserved for future versions of the architecture.

~ Codes marked with a '~' are not valid and are reserved for future versions of the architecture. The results of such an encoding are undefined.

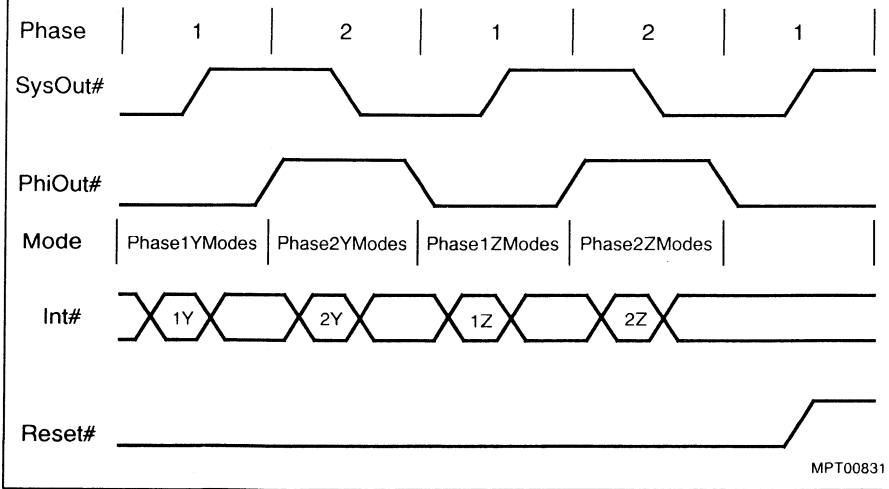
Resetting the SAB-2000A

The Reset# input signal is used to force processor execution to start at the reset vector (reset exception servicing routine) and to initialize the processor state. The Reset# signal must be asserted for a minimum of 6 cycles to guarantee processor initialization. After a reset has occured (i.e. before the exception handling routine has been executed) the following processor state is guaranteed:

- KUc, the current Kernel/User bit, is zero corresponding to Kernel mode.
- IEc, the current interrupt enable bit, is zero corresponding to interrupts disabled.
- TS, the TLB shutdown bit, is zero corresponding to TLB enabled.
- SwC, the Swap Cache bit, is zero corresponding to caches not swapped.
- BEV, the Boot Exception Vector bit, is one corresponding to selection of the bootstrap exception vector.
- The Random register is set to 63.

When the Reset# signal is deasserted in the n th cycle, the logic levels on the 6 interrupt pins during the n-2 and n-1 cycles are sampled by the processor to determine various processor operating modes such as Endianness, Cachelessness, Test etc. The last two cycles before the cycle in which the Reset# signal is deasserted are called the Y and Z cycles, respectively. Figure 34 illustrates these two cycles. The Reset# timings are described in the Timing Parameters section.

Figure 34
Y and Z Cycles on Reset# Deassertion



The Mode Select Summary Table, given in the Pin Definitions and Functions section, summarizes the processors mode selectable features and is reproduced here.

Table 6
Summary of Mode Select

Interrupt#	Y Cycle Modes		Z Cycle Modes	
	Phase 1	Phase 2	Phase 1	Phase 2
Int#(0)	Reserved	Reserved	Reserved	Big Endian#
Int#(1)	Reserved	Reserved	Reserved	Tristate#
Int#(2)	Reserved	Reserved	Reserved	NoCache#
Int#(3)	Reserved	Reserved	Reserved	BusDriveOn
Int#(4)	PhaseDelayOn#	PhaseDelayOn#	Asserted#	PhaseDelayOn#
Int#(5)	Reserved	Res5rved	Asserted#	R2000Md

Note that all reserved modes in this table must be driven asserted to guarantee compatibility with future processor revisions.

Asserting **PhaseDelayOn#** causes the processor to insert additional phase delay into its input clock paths. This additional phase delay allows coprocessors to minimize their skew, i.e. phase lock to the SAB-R2000A.

Byte order or Endianness is determined by the value of **BigEndian#**. Assertion will result in Big Endian ordering, while deassertion will result in Little Endian ordering.

Assertion of **Tristate#** causes the processor to tristate all of its outputs. In this condition the processor outputs can be driven by an external medium.

When **NoCache#** is asserted all memory references are forced to occur at the processor cycle rate, i.e. no cache miss stalls occur.

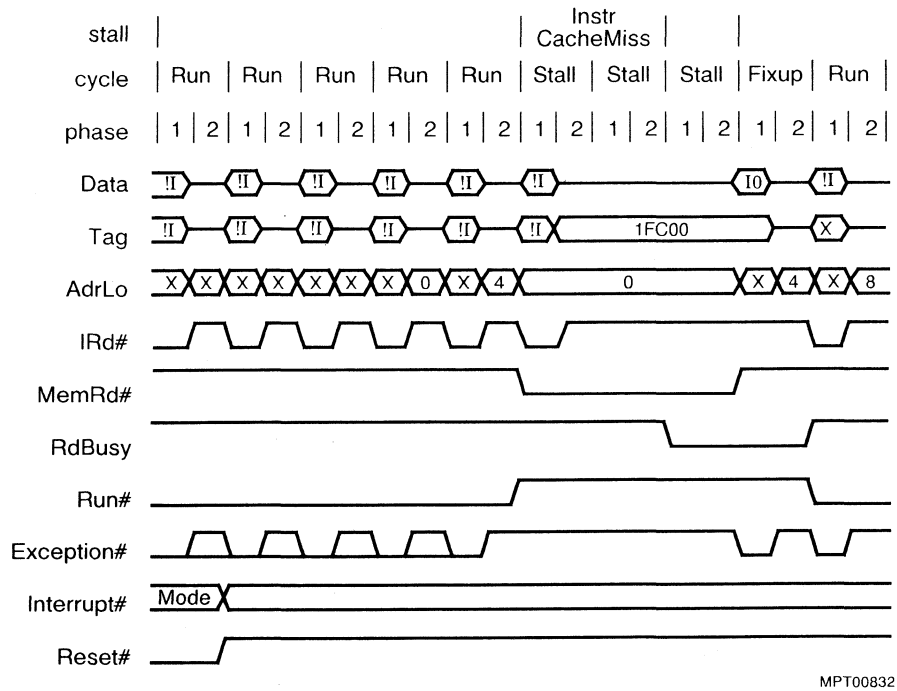
Data and Tag buses are driven during phase 2 of write busy and coprocessor busy stalls when **BusDriveOn** is asserted. If the Data and Tag buses are not being driven externally during these stalls and fast TTL inputs are connected to the bus, the BusDriveOn should be asserted to prevent bus oscillation due to the bus being pulled to the trip point of Fast TTL.

In the Mode Select Summary Table "#" means active low. This means, for example, in phase 2 of the Z cycle, if Int#0 has logic value low, Big Endian mode will be selected. On the other hand, if Int#0 has logic value high, Little Endian mode will be selected (i.e. the opposite).

Note that the Reset# signal must be asserted asynchronously and deasserted synchronously with the output Clock SysOut#. The reason for asserting Reset# asynchronously is to avoid a possible dead-lock if the Sysout# is used to clock it. More detailed information can be obtained in the "MIPS R3000 Processor Interface" specification and in a MIPS application note entitled "Resetting the R3000 and R3010".

The Reset Exception itself occurs when the Reset# signal is asserted and then deasserted. The Reset exception vector is selected to appear within the uncached, unmapped memory space (kseg1) of the machine so that instructions can be fetched and executed while the cache and memory system are still in an undefined state. The Reset exceptions special interrupt vector is 0xbfc00000. This is a virtual address and resides in kseg1 as explained in the Memory Management section. Kseg1 is direct mapped into the first 512 Mbytes of physical memory and the physical address is defined by subtracting 0xa0000000 from the virtual address of the reset exceptions interrupt vector, which yields 0x1fc00000. This can be seen in figure 35, which illustrates the sequence of events when Reset# is deasserted, i.e. when the processor comes out of reset. As can be seen the address (physical) on the Tag and AdrLo Buses is 0x1fc00000, as expected. Refer to the Timing Specification section for the notation.

Figure 35
Reset Behavior



Timing Specifications

Absolute Maximum Ratings

Ambient temperature under bias	0 to +70 °C
Storage temperature	– 65 to +150 °C
Supply Voltage (V_{CC})	– 0.5 to +7.0 V
Input voltage (V_{IN})	– 0.5 to +7.0 V
Load Capacitance on any Pin (C_{Ld})	100 pF

Note: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.
Not more than one output should be shorted at a time. Duration of the short should not exceed 30 seconds.

DC Characteristics

T_A = 0 to +70 °C; V_{CC} = 5 V ±5%

Parameter	Symbol	Limit values						Unit	Test condition
		12.5 MHz		16.67 MHz		20 MHz			
		min.	max.	min.	max.	min.	max.		
Operating Parameters									
Output HIGH voltage	V_{OH}	3.5	–	3.5	–	3.5	–	V	V_{CC} = min. I_{OH} = –4mA
Output LOW voltage	V_{OL}	–	0.4	–	0.4	–	0.4	V	V_{CC} = min. I_{OL} = 4mA
Input HIGH voltage	V_{IH}	2	V_{CC} +0.25	2	V_{CC} +0.25	2.0	V_{CC} +0.25	V	
Input LOW voltage	V_{IL}	– 0.5[1]	0.8	– 0.5[1]	0.8	–	0.8	V	
Input HIGH voltage	V_{IHS} [2]	2.5	V_{CC} +0.25	3.0	V_{CC} +0.25	3.0	V_{CC} +0.25	V	
Input LOW voltage	V_{ILS} [2]	– 0.5[1]	0.4	– 0.5[1]	0.4	–	0.4	V	
Input capacitance	C_{In}	–	10	–	10	–	10	pF	
Output capacitance	C_{Out}	–	10	–	10	–	10	pF	
Operating current	I_{CC}	–	550	–	600	–	630	mA	V_{CC} = 5.25V

1) V_{IL} min. = – 3.0 V for pulse width less than 15 ns
2) V_{IHS} and V_{ILS} apply to Clk2 × Sys, Clk2 × Smp, Clk2 × Rd, Clk2 × Phi, CpBusy, and Reset#.

AC Characteristics

T_A = 0 to 70 °C; V_{CC} = 5 V ±5%

Note: *All output timings are given assuming 25 pf of capacitive load. Output timings should be derated where appropriate as per the table below.*

All timings referenced to 1.5 V.

Parameter	Symbol	Limit values						Unit	Test condition
		12.5 MHz		16.67 MHz		20 MHz			
		min.	max.	min.	max.	min.	max.		
Clock Parameters 3)									
Input clock high	$t_{ClkHigh}$	16	–	8	–	10	–	ns	Transition ≤ 5ns
Input clock low	t_{ClkLow}	16	–	8	–	10	–	ns	Transition ≤ 5ns
Input clock period	t_{ClkP}	40	1000	30	1000	25	1000	ns	
Clk2 × Sys to Clk2 × Smp		0	$t_{\frac{Cyc}{4}}$	0	$t_{\frac{Cyc}{4}}$	0	$t_{\frac{Cyc}{4}}$	ns	
Clk2 × Smp to Clk2 × Rd		0	$t_{\frac{Cyc}{4}}$	0	$t_{\frac{Cyc}{4}}$	0	$t_{\frac{Cyc}{4}}$	ns	
Clk2 × Smp to Clk2 × Phi		11	$t_{\frac{Cyc}{4}}$	5	$t_{\frac{Cyc}{4}}$	7	$t_{\frac{Cyc}{4}}$	ns	
Run Operation Parameters									
Data enable	t_{DEn}	– 1	– 2.5	– 1	– 2	–	– 2	ns	–
Data disable	t_{DDis}	0	– 1	0	– 1	0	– 1	ns	–
Data valid	t_{DVal}	–	3.5	–	3	–	3	ns	25 pF Load
Write delay	t_{WrDly}	0	7.5	0	5	0	4	ns	25 pF Load
Data setup	t_{DS}	11.5	–	9	–	8	–	ns	–
Data hold	t_{DH}	– 2.5	–	– 2.5	–	– 2.5	–	ns	–
CpBusy setup	t_{CBS}	15	–	13	–	11	–	ns	–
CpBusy hold	t_{CBH}	– 2.5	–	– 2.5	–	– 2.5	–	ns	–
Access type(1-0)	t_{AcTy}	1	10	1	7	–	6	ns	25 pF Load
Access type(2)	t_{AT2}	1	20	1	17	–	14	ns	25 pF Load
Memory write	t_{MWr}	1	10	1	7	–	23	ns	25 pF Load
Exception	t_{Exc}	1	10	1	7	–	7	ns	25 pF Load

3) The clock parameters apply to all four 2xClocks: Clk2 × Sys, Clk2 × Smp, CLK2 × Rd, and Clk2 × Phi.

AC Characteristics (cont'd)

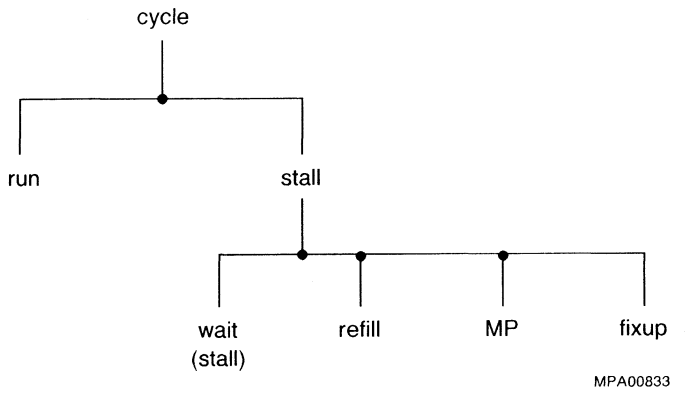
Parameter	Symbol	Limit values						Unit	Test condition
		12.5 MHz		16.67 MHz		20 MHz			
		min.	max.	min.	max.	min.	max.		
Stall Operation Parameters									
Address valid	t_{SAVal}	–	38	–	30	–	23	ns	25 pF Load
Access type	t_{SAcTy}	–	35	–	27	–	23	ns	25 pF Load
Memory read initiate	t_{MRdI}	1	35	1	27	–	23	ns	25 pF Load
Memory read terminate	t_{MRdT}	1	10	1	7	–	7	ns	25 pF Load
Run terminate	t_{StI}	5	25	5	17	–	15	ns	25 pF Load
Run initiate	t_{Run}	5	10	5	7	–	6	ns	25 pF Load
Memory write	t_{SMWr}	5	35	5	27	–	23	ns	25 pF Load
Exception valid	t_{SExc}	5	28	5	20	–	18	ns	25 pF Load
Capacitive Load Deration									
Load derate	C_{LD}	0.5	2.5	0.5	2	0.5	1	ns/ 25pF	

As described earlier the SAB-2000A supports interfaces to external cache, main memory and coprocessors. This section describes the timing parameters and operation of the important cases for each of these interfaces along with Interrupt, Reset and Multi-processing examples.

Operation Fundamentals

A "cycle" is the basic instruction processing unit of the SAB-2000A processor. Cycles in which forward progress is made, i.e. an instruction is retired, are called "run" cycles. An instruction is retired either by its completion or in the presence of an exception its abortion. Cycles in which no forward progress is made are called "stall" cycles. Stall cycles are used for resolving urgent situations such as cache misses on loads, write system busy during stores, and coprocessor interlocks. All cycles can be classified as either run cycles or stall cycles. There are four types of stall cycles: "wait" stall cycles - simply known as stall cycles; "refill" stall cycles - which occur only during main memory reads; "multi-processor" (MP) stall cycles – allow the memory system to read or invalidate specific data cache entries; and "fixup" stall cycles – occur during the final cycle of the stall and are used in general to fix up the conditions which caused the stall. Processor transactions which occur during the first half of the cycle are called phase 1 transactions while those which occur during the second half of the cycle are called phase 2 transactions. Figure 36 summarizes the cycles in the SAB-2000A.

Figure 36
SAB-R2000A Cycles



As described earlier coprocessors maintain synchronization with the SAB-2000A by monitoring the signals Run# and Exception#. Run# is asserted by the SAB-2000A during run cycles and deasserted during stall cycles. When Run# is deasserted during the n th cycle, the coprocessor(s) disregard(s) the instruction-data pair presented during the n-1 th cycle. When Run# is reasserted during the m th cycle, the coprocessor(s) take(s), as a replacement for the instruction-data pair which was disregarded, the instruction-data pair presented during the m-1 th cycle – which was the final fixup cycle for whatever stall sequence was occurring.

Exception# is used by the coprocessor(s) to track exception related information during run cycles and stall related information during stall cycles.

- During phase 1 of run cycles Exception# indicates whether an exception has occurred for the instruction which is currently in its "writeback" pipestage. Unless the exception is occurring as a result of an interrupt request by the coprocessor, the assertion of Exception# prevents any state from being committed in the coprocessor.
- During phase 2 of run cycles Exception# indicates whether an interrupt request is being granted for the instruction which is currently in its "memory access" pipestage.
- During phase 1 of stall cycles Exception# indicates whether the current stall cycle is a fixup cycle. When a fixup cycle is occurring, it is guaranteed that the data present on the Data bus is valid. The coprocessor uses the fixup indication to qualify the use of data sampled from the bus during the stall.
- During phase 2 of stall cycles Exception# indicates whether the current stall is a Coprocessor Busy stall.

The use of the Exception# signal is summarized below.

Table 7

	phase 1	phase 2
Run	Exc1W#	IntGr2M#
Stall	Fixup1#	CpBusy2#

Processor Input Clocks

The SAB-R2000A has four separate double frequency (i.e. in a 16 MHz system these clocks are 32 MHz) input clocks. They can be adjusted to obtain optimum positioning of cache interface signals. The absolute timing of these clocks with respect to the processor outputs is undefined, only the differences are important. A short description of these four clocks follows.

- Clk2×Sys:
  is the master clock and must lead all others. It determines the position of SysOut# (the processors output clock) with respect to Data, Tag and Address buses.
- Clk2xSmp:
  determines the sample point for data coming into the processor on all processor inputs except those coming directly from coprocessors.
- Clk2xRd:
  controls output enable time and provides sufficient address access to sample address hold from end of write, and data hold from end of write.
- Clk2xPhi:
  determines the position of the internal phases, phase 1 and phase 2.

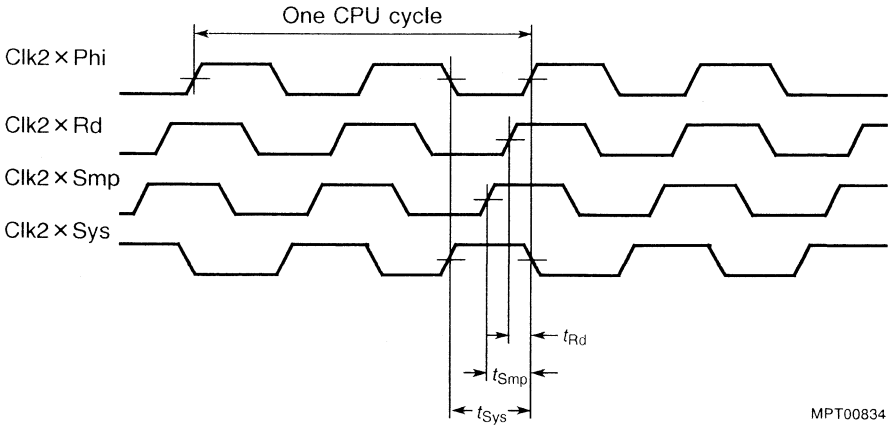
Table 8 illustrates the 2×Clock dependency of the processors timing controlled outputs. Outputs are referenced only to "rising edges" of the 2×Clocks. The assertion dependency is indicated by ↑ and deassertion by ↓.

Table 8

	Clk2×Sys	Clk2×Smp	Clk2×Rd	Clk2×Phi
IClk#, DClk#		↓	↑	
IRd, DRd	↓		↑	
IWr, DWr		↑ ↓		
SysOut#	↑ ↓			
Data, Tag			↓	↑
Address				↑ ↓
All Others				↑ ↓

Figure 37 shows the four 2x input clocks.

Figure 37
2x Input Clocks



In the timing diagrams which follow, timing specifications are given relative to a shifted version of the processor output clock SysOut#. The clock is called PhiOut# and is a virtual clock, i.e. the processor does not actually produce this output. It is shown in the timing diagrams for reasons of clarity, because its period is synchronous with a machine cycle. The shift amount is equal to the difference between Clk2 × Sys to Clk2 × Phi and, as is shown in figure 37, is t_{Sys}. Also, in the timing diagrams Clk2 × Sys and SysOut# are shown to clarify the relationship between these signals.

In reality SysOut# is produced rather than PhiOut# since this provides a signal with timing appropriate for synchronizing system transactions to the processor. Timings are given relative to PhiOut# since this makes determining the position of the input clocks the most straightforward. The timing of any output with respect to SysOut# can be determined from its timing with respect to PhiOut# by adding t_{Sys}.

Timing Diagram Notation

The following timing diagrams describe various transactions of the processor. Table 9 illustrates the notational conventions used in these diagrams.

Table 9

Character	Meaning
I	Instruction
D	Data
#	Active low
%	An incorrect datum
!	An unused datum
Z	The high impendance state
Ad	Address
in	into processor
out	out of processor
//////	not valid or Don't Care

Cache Timing

Cache operation was explained in the Interface section. Figure 38 illustrates cache operation and timing. During run cycles the Access Type bus, AccTy2:0, indicates whether or not a phase 2 transaction is scheduled for that cycle and the size of the datum being transferred. Table 10, below, summarizes AccTy encoding during run cycles.

Table 10

AccTy(2)	AccTy(1:0)	size
1	XX	no transaction
0	00	byte
0	01	half word
0	10	tribyte
0	11	word

Main Memory Reads

When a LOAD misses in the cache, a main memory read is initiated. Main memory reads are supported by read busy stalls and the MemRd#, RdBusy. Table 11 summarizes the meaning of the AccTy2:0 bus during main memory reads.

Table 11

AccTy(2)	AccTy(1:0)	size	type
0	00	byte	uncached/unkown
0	01	half word	uncached/unkown
0	10	tribyte	uncached/unkown
0	11	word	uncached/unkown
1	X0	word	cached/data
1	X1	word	cached/instruction

Figure 38 illustrates a single word transfer. Entry into the stall is indicated by the assertion of MemRd# which occurs in the cycle following the one in which the LOAD missed. During the stall the SAB-R2000A presents the read address on the AdrLo and Tag buses and tristates the Data bus. This state is maintained until RdBusy is deasserted. RdBusy is deasserted during phase 1 of the cycle in which the memory system will provide Data and Data parity on the Data bus. This is the termination of a read busy stall cycle.

The cycle following that in which RdBusy is deasserted is the fixup cycle. During this cycle the appropriate cache is written (in this case the Data cache) with the data returned by main memory. Simultaneously, the generated Data parity, Tag and Tag parity are also written to the cache.

Note: The cache write does not occur if the stall was due to an uncached reference. The processor resumes run operation on the cycle following the fixup cycle if no other stall is pending.

Figure 38
Cache Timing

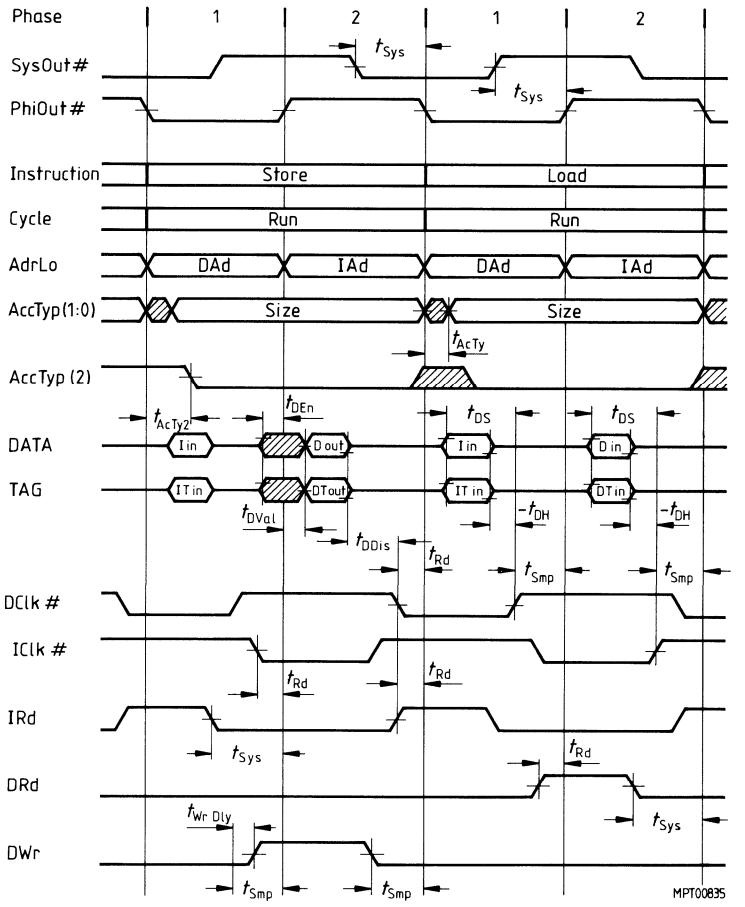
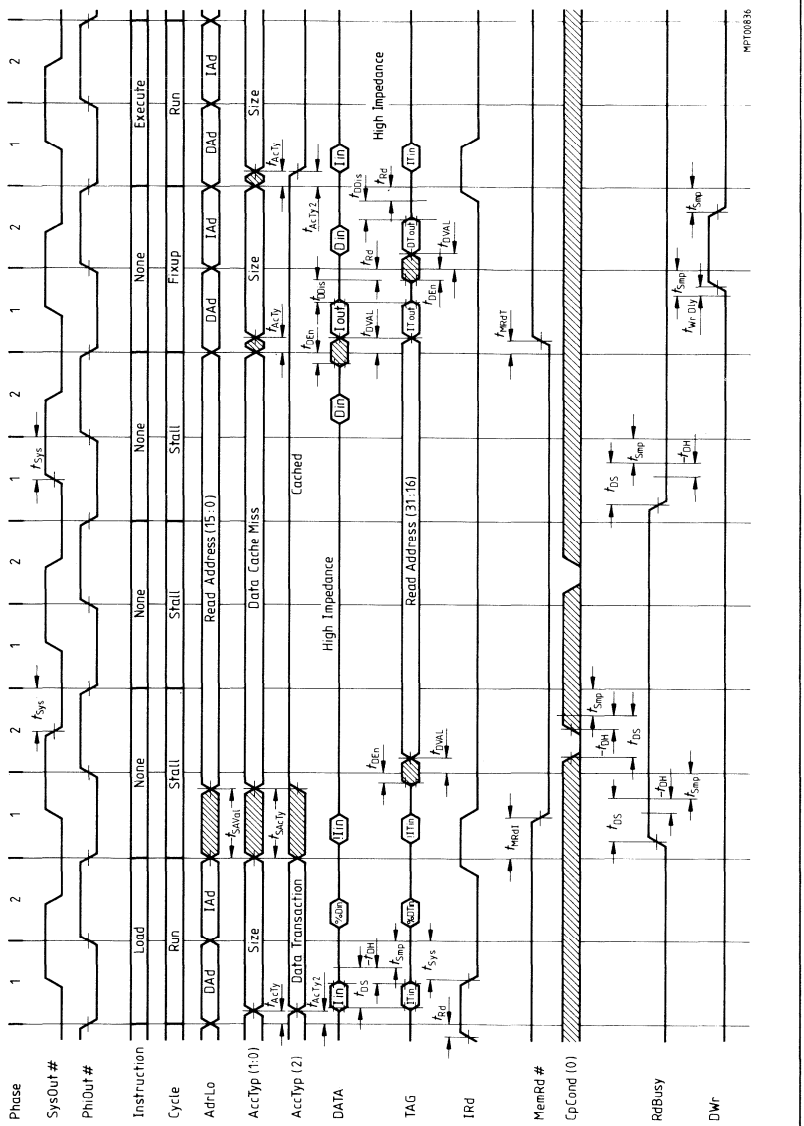


Figure 39
Single Word Main Memory Read (Data Cache Miss - Cached)



Main Memory Writes

Main memory writes are accomplished through a write buffer which accepts writes at cache speeds. A write is indicated to the Write Buffer by the assertion of the MemWr# signal. If the write buffer becomes full and a further write is attempted it causes a write busy stall, this is illustrated in figure 39. The first write (Store) shown fills the write buffer causing it to assert WrBusy#. The write which is attempted in the next cycle is not accepted by the write buffer and is redone by the processor during the fixup cycle. The write busy stall is terminated when the write buffer deasserts WrBusy# to indicate it can accept another write. The cycle following its deassertion will be the fixup cycle, where the write to the Write Buffer (Main Memory) is redone.

Interrupts

The SAB-R2000A has 6 general purpose interrupt inputs which are sampled during phase 2 of all run and fixup cycles. After causing an interrupt exception to occur, the interrupts continue to be sampled during each phase 2 to provide a level sensitive indication of the active interrupt(s). Figure 40 shows the Interrupt Timing.

Reset Timing

The Reset# input is used to force processor execution starting at the reset exception vector and to initialize processor state. Its operation is explained in the Resetting the SAB-R2000A section. Figure 41 illustrates its timing parameters.

Coprocessor Timing

During run cycles, the operation and timing of coprocessor LOADS and STORES is identical to that of the main processor. This can be seen in figure 42. To provide synchronization when required, the SAB-R2000A supports coprocessor busy stalls. The operation of such a stall is also illustrated in figure 42. The coprocessor must assert CpBusy during phase 2 of the "ALU" cycle of the coprocessor instruction to initiate such a stall. To terminate the stall CpBusy must be deasserted during phase 1. The cycle following this deassertion is the fixup cycle.

Figure 40
Main memory Write – With Write Busy Stall

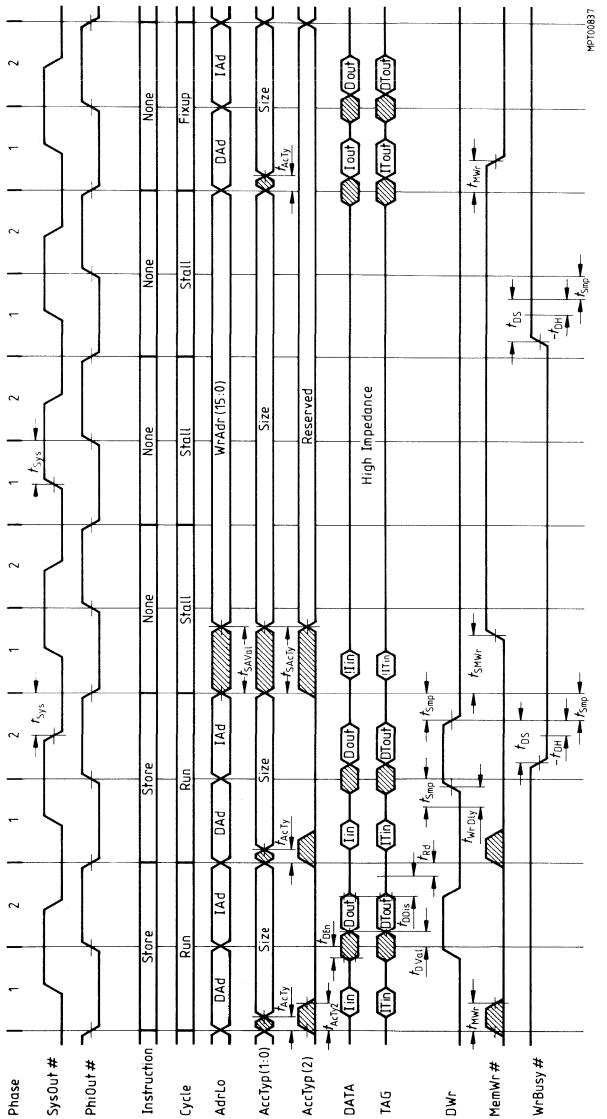


Figure 41
Interrupt Timing

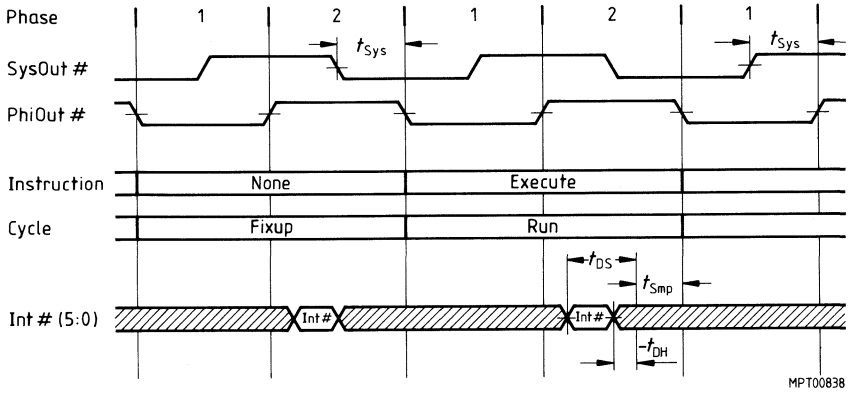


Figure 42
Mode Select Timing

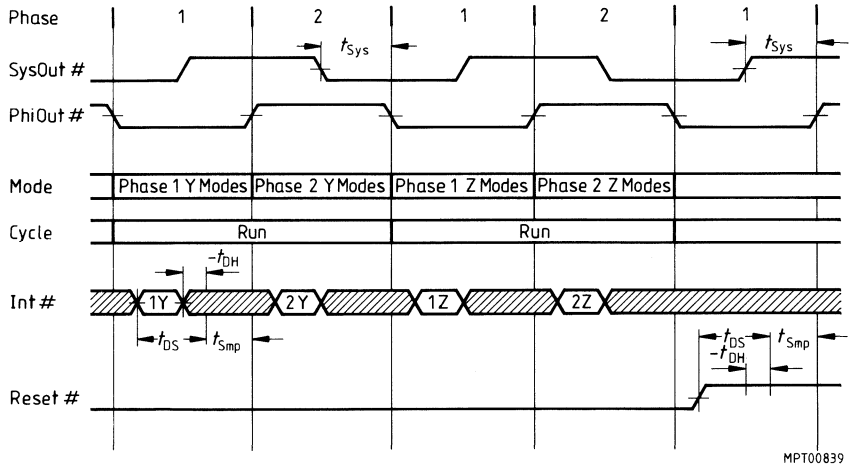
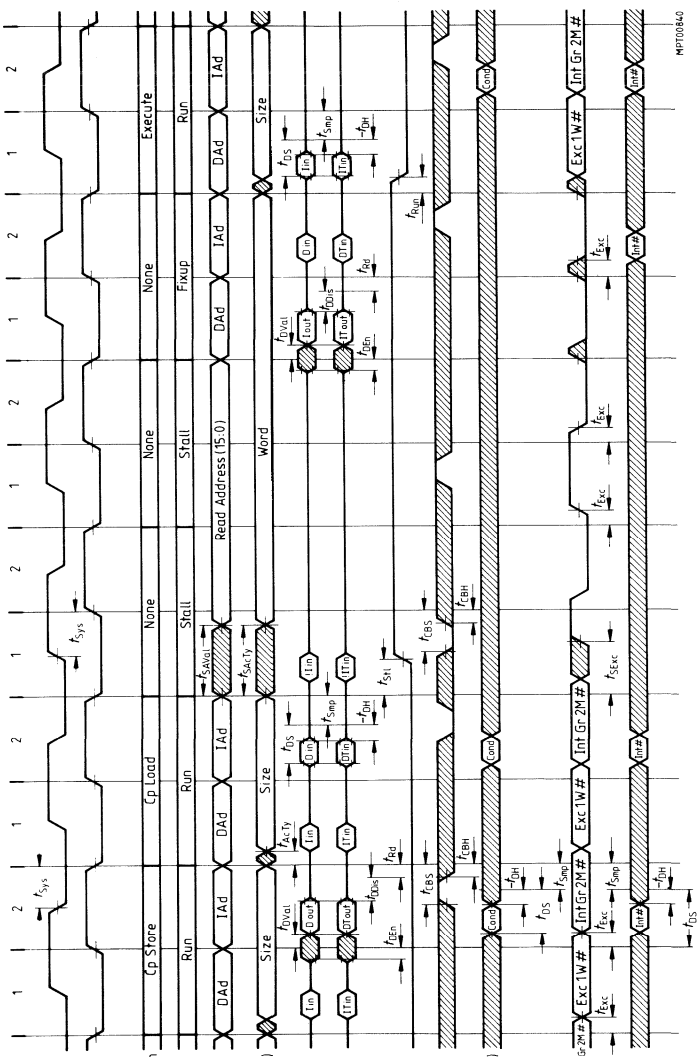


Figure 43
Coprocessor Interface Timing – With CoprocessorBusy Stall



SIEMENS

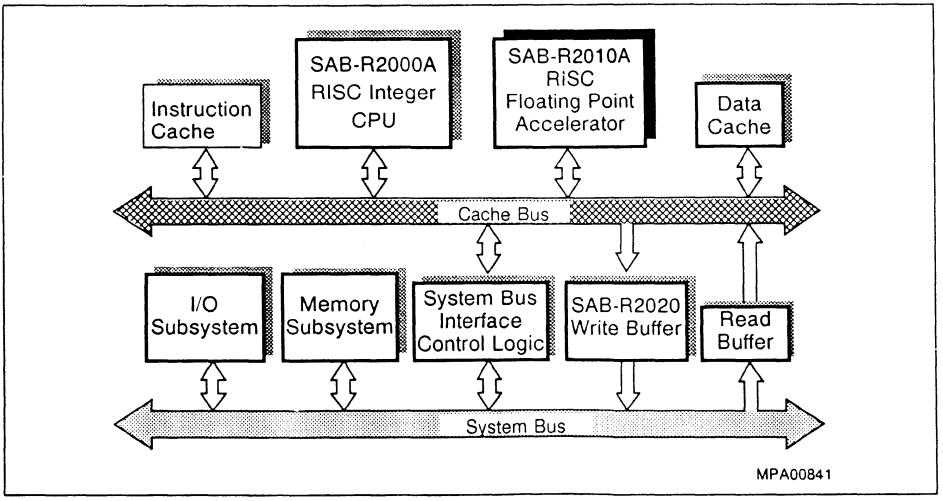
High Performance Floating-Point Coprocessor

SAB-R2010A

Based on advanced RISC architecture with four independent arithmetic functional units

Advance Information

- Fully conforms to ANSI/IEEE standard 754-1985 for binary floating-point arithmetic
- Load/store instruction set
  - single cycle loads and stores
- Four independent functional units
  - Register, Add, Divide and Multiply units
  - allows up to four floating-point instructions to be executed in parallel
- Full 64-bit operation
  - sixteen 64-bit floating-point registers
- Seamless coprocessor interface to SAB-R2000A
- Transparent addition of floating-point extensions to the SAB-R2000A's instruction set
- Fully compatible to all R2010A processors of other manufacturers
- Ceramic package: CL-CC-84



Ordering Information

Type	Ordering code	Package	Description
SAB-R2010A-12-QJ	Q67120-C553	CL-CC-84	32/64-bit Floating-Point Coprocessor, 12.5 MHz
SAB-R2010A-16-QJ	Q67120-C495	CL-CC-84	32/64-bit Floating-Point Coprocessor, 16.67 MHz

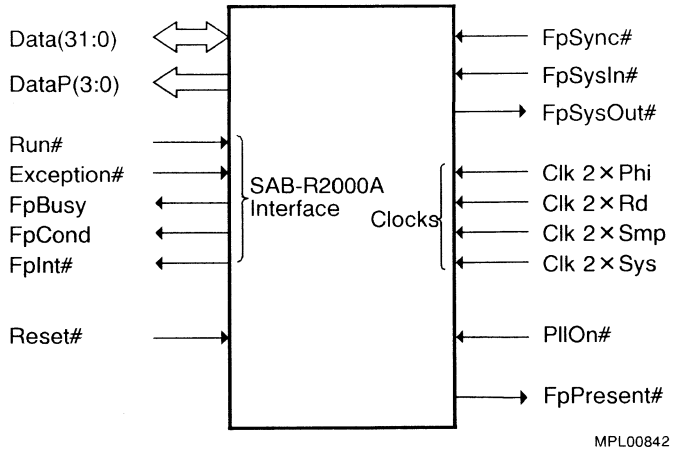
Introduction

The SAB-R2010A is a high performance Floating-Point Accelerator (FPA) which is implemented as a full-custom VLSI CMOS chip. It serves as a coprocessor to the SAB-R2000A RISC microprocessor. It transparently extends the SAB-R2000A's instruction set to perform floating-point operations, by cointerpreting the common instruction stream. The FPA, with associated system software, fully conforms to the requirements of ANSI/IEEE standard 754-1985 "IEEE Standard for Binary Floating-Point Arithmetic". In addition the SAB-R2010A fully supports the standard recommendations. The SAB-R2010A's architecture is organised as follows – hardware directly implements the essential floating-point operations of addition, subtraction, division, multiplication, comparison, conversion between formats, absolute value and negation. These operations are highly optimized. System software supplies the more complex functions and while doing so benefits from the underlying fast arithmetic hardware. Figure 1 illustrates the SAB-R2010A Logic symbol.

Pin Names

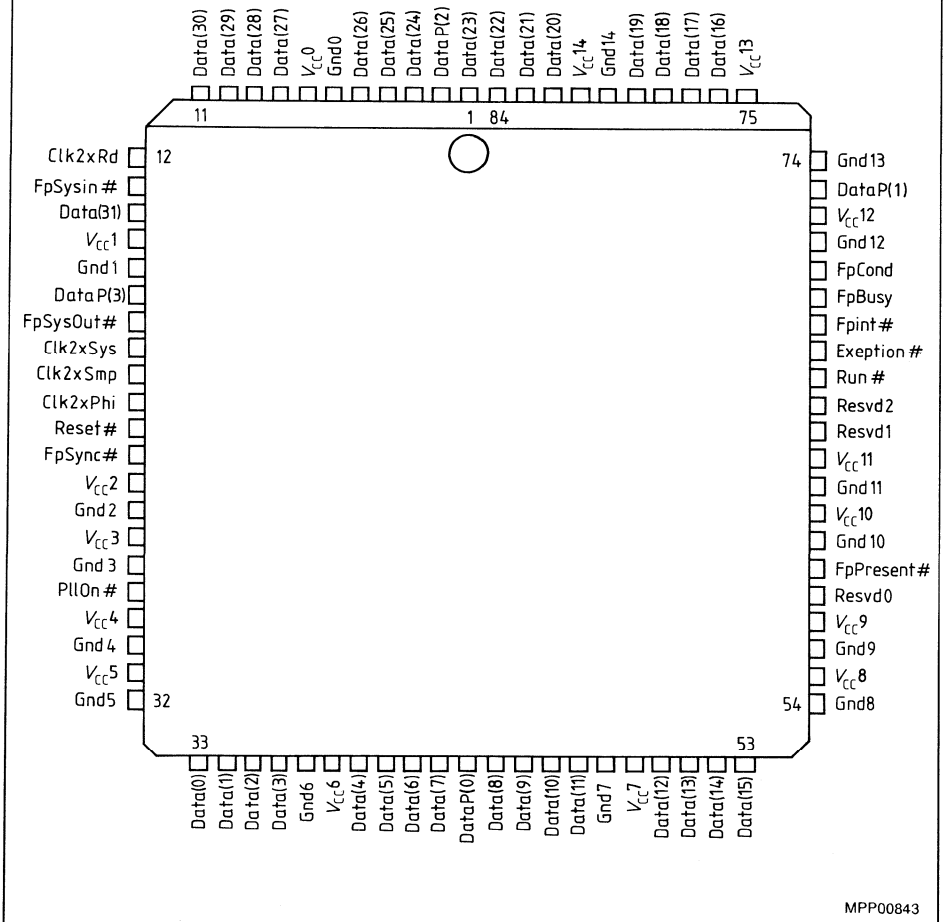
Data(31:0)	Data Bus
DataP(3:0)	Even parity for Data Bus
Run#	System in Run or Stall state
Exception#	Exception related information
FpBusy	Floating-point busy stall
FpCond	Floating-point condition
FpInt#	Floating-point Interrupt
Reset#	Synchronous Initialization
FpSync#	Floating-point Synchronize
FpSysIn#	Floating-point System clock in
FpSysOut#	Floating-point System clock out
PllOn#	Phase Lock Loop On
FpPresent#	Floating-point present

Figure 1
Logic Symbol



Pin Configurations

Figure 2
CL-CC-84 (Top View)



Pin Definitions and Functions

Symbol	Pin Number	Input (I) Output (O)	Function
Data(31:0)	14,11,10,9,8,5,4, 3,1,84,83,82,79, 78,77,76,53,52, 51,50,47,46,45, 44,42,41,40,39, 36,35,34,33	I/O	A multiplexed 32-bit bus used for instruction and data transfers on phases 1 and 2, respectively.
DataP(3:0)	17,2,73,43	O	A 4-bit bus containing even parity over the data bus. Parity is generated by the FPC on stores.
Run#	66	I	Input to the FPC which indicates whether the processor-coprocessor system is in the run or stall state.
Exception#	67	I	Input to the FPC which indicates exception related status information.
FpBusy	69	O	Signal to the CPU indicating a request for a coprocessor busy stall.
FpCond	70	O	Signal to the CPU indicating the result of the last comparison operation.
FpInt#	68	O	Signal to the CPU indicating that a floating-point exception has occurred for the current FPC instruction.
Reset#	22	I	Synchronous initialization input used to distinguish the processor-FPC synchronization period from the execution period. Reset# must be synchronized by the leading edge of SysOut from the CPU.
PllOn#	28	I	Input which during the reset period determines whether the phase lock mechanism is enabled, and during the execution period determines the output timing model.

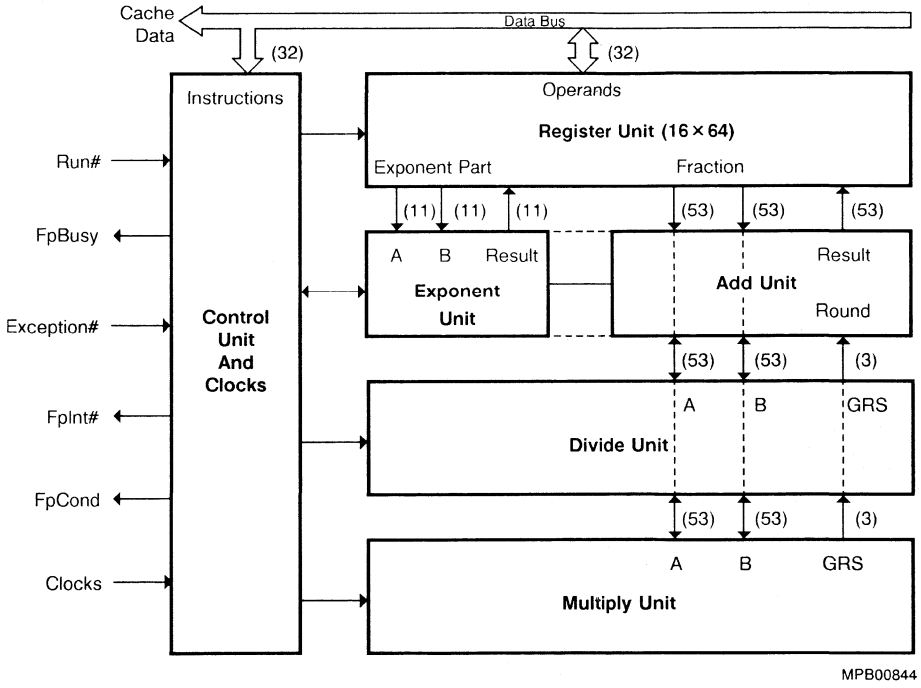
Pin Definitions and Functions (cont'd)

Symbol	Pin Number	Input (I) Output (O)	Function
FpPresent#	59	O	Output which is pulled to ground through an impedance of approximately 0.5 kΩ. By providing an external pull-up on this line an indication of the presence or absence of the FPC can be obtained.
Clk2 × Sys	19	I	A double frequency clock input used for generating FpSysOut#.
Clk2 × Smp	20	I	A double frequency clock input used to determine the sample point for data coming into the FPC.
Clk2 × Rd	12	I	A double frequency clock input used to determine the disable point for the data drivers.
Clk2 × Phi	21	I	A double frequency clock input used to determine the position of the internal phases 1 and 2.
FpSysOut#	18	O	Synchronization clock from the FPC.
FpSysIn#	13	I	Input used to receive the synchronization clock from the FPC.
FpSync#	23	I	Input used to receive the synchronization clock from the CPU.
GND14-1	80,74,71,62,60, 56,54,48,37,32, 30,27,25,16,6		Ground
V_{CC}14-1	81,75,72,63,61, 57,55,49,38,31, 29,26,24,15,7		Power Supply (+ 5 V)
Resvd2-0	65,64,58		Reserved

Functional Description

The SAB-R2010A contains four independent arithmetic functional units (Register, Add, Divide and Multiply) which interact with a scheduling and managing control unit. Figure 3 shows the block diagram of the SAB-R2010A.

Figure 3
Functional Block Diagram



Note: A hash sign # indicates an Active-LOW Signal

Basic Architecture

As figure 3 shows, the SAB-R2010A consists of five main units.

The **Control Unit** continually monitors the transactions between the SAB-R2000A (with which the SAB-R2010A shares the data bus) and the instruction cache (i.e. the instruction stream). If an instruction does not apply to the SAB-R2010A, it ignores it. When an instruction does apply, it interprets it. Synchronization between the coprocessor and the main processor is also managed by the control unit. The control unit monitors the signals Run# and Exception# to see what state the SAB-R2000A is in. Run# is used to track pipeline disruptions due to non-exceptional events (i.e. CPU stalls) such as cache misses, write busy etc. Exception# is used to track pipeline disruptions due to exceptional events such as virtual to physical address translation misses, interrupts etc. When either of these cases occurs, the SAB-R2010A's pipeline is shut down (stalled), in such away that unfinished instructions can be restarted later without numerical inconsistencies. Whenever the SAB-R2000A requires the result of a floating-point instruction which is not yet completed, the control unit signals the CPU to wait through the assertion of the FpBusy signal. The control unit also schedules the execution of each instruction with the four arithmetic units.

The **Register Unit's** register file can perform two 64-bit operand reads, one 64-bit write result and one memory load/data write in one cycle. This implies that four ports exist. Physically, a two-port design, which is accessed twice per cycle, implements the register file.

The **Add Unit** executes add, subtract, convert, compare, negate and absolute value instructions as well as the final IEEE rounding step of multiply and divide operations. The exponent data path (exponent unit) is included in this unit and it computes the 8-bits (single-precision) or 11-bits (double-precision) of exponents for all arithmetic operations.

The **Divide Unit** uses a radix-4, SRT-division algorithm to produce four quotient bits per cycle. This method uses a redundant encoding of the quotient as a sum of digits with values 2, 1, 0, -1 and -2. A double-precision divide requires a total of 19 cycles (12 cycles for a single-precision divide).

The **Multiply Unit** computes the product of the mantissa portions of its operands (refer to the Data Formats section). In the case of double-precision, the multiplier computes the product of two 53-bit operands in less than four cycles. It retains the most significant 56-bits of the 106-bit product.

Results of both the multiplier and divider are returned to the add unit over the two operand buses (A and B) for final carry propagation and rounding. A separate path exists for the guard, round and sticky bits (GRS) required for IEEE rounding. The interaction of the five units and the width of the major data buses can be seen in figure 3.

The autonomy of the four arithmetic units enables them to run in parallel. Concurrently executing instructions generally do not conflict for resources – except at the beginning and end cycles of each operation, mainly due to simultaneous requests for the add unit (see the Pipeline Architecture section). Based on the latency of each operation, the control unit schedules instructions to ensure that no two will need, for example, the exponent unit or rounding function of the add unit at the same time. The floating-point architecture requires that instructions must appear to complete in the order they were issued. However the control unit recognizes the special cases of operations with exceptional results, conflicts for one arithmetic unit and data dependencies between operations – therefore it will reschedule instructions for maximum pipeline efficiency. In such instances it adjusts the issue schedule to maintain the illusion of in-order instruction completion. Refer to the Pipeline Architecture section for more details.

Pipeline Architecture

The SAB-R2010A has an instruction pipeline which mirrors that of the SAB-R2000A processor. However, there is a difference: the FPA (SAB-R2010A) has a 6-stage pipeline in contrast to the 5-stage pipeline of the SAB-R2000A. It uses an extra pipestage to provide efficient coordination of exception responses between the FPA and the CPU (SAB-R2000A).The six stages of the SAB-R2010A pipeline are:

(1) IF — Instruction Fetch:
The CPU calculates the instruction address required to read an instruction from the instruction cache. The instruction address is generated and output during phase 2 of this pipestage. No action is required by the SAB-R2010A during this pipestage since the main processor is responsible for address generation. Note that the instruction is not actually read into the processor until the beginning of the RD pipestage. Refer to figure 4.

(2) RD — Register Fetch/Instruction Decode:
The instruction is present on the Data bus during phase 1 of this pipestage and the FPA decodes the data on the bus to determine whether it is an instruction for the FPA. The FPA reads any required operands from its registers (RF in figure 4) while decoding the instruction.

(3) ALU — ALU Operation:
If an instruction is one for the FPA, execution commences during this pipestage. If the instruction causes an exception, the FPA notifies the SAB-R2000A of the exception during this pipestage by asserting the FpInt# signal. If the SAB-R2010A determines that it requires additional time (i.e. more than 1 cycle) to complete this instruction, it initiates a stall during this pipestage.

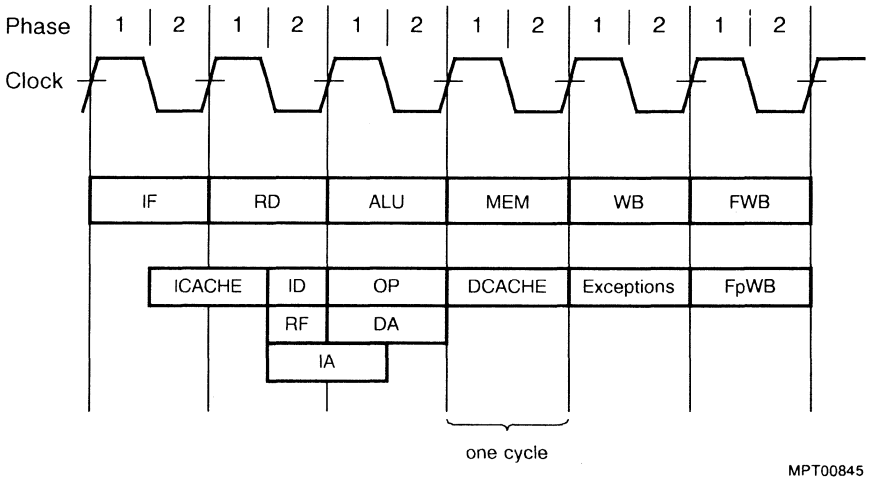
(4) MEM — Memory Access:
If it is a coprocessor Load or Store instruction, the FPA presents or captures the data during phase 2 of this pipestage. If an interrupt is taken by the main processor, it notifies the SAB-R2010A during phase 2 of this pipestage (via the Exception# signal).

(5) WB Write back:
If the instruction that is currently in the write back (WB) stage caused an exception, the main processor notifies the FPA by asserting the Exception# signal during this pipestage. Thus, the FPA uses this pipestage solely to deal with exceptions.

(6) FWB Floating-Point Write back:
The SAB-R2010A uses this pipestage to write back ALU results to its register file. This stage is the equivalent of the WB stage in the SAB-R2000A pipeline.

Figure 4 illustrates the 6 stages of the SAB-R2010A pipeline. Each step requires approximately one machine cycle.

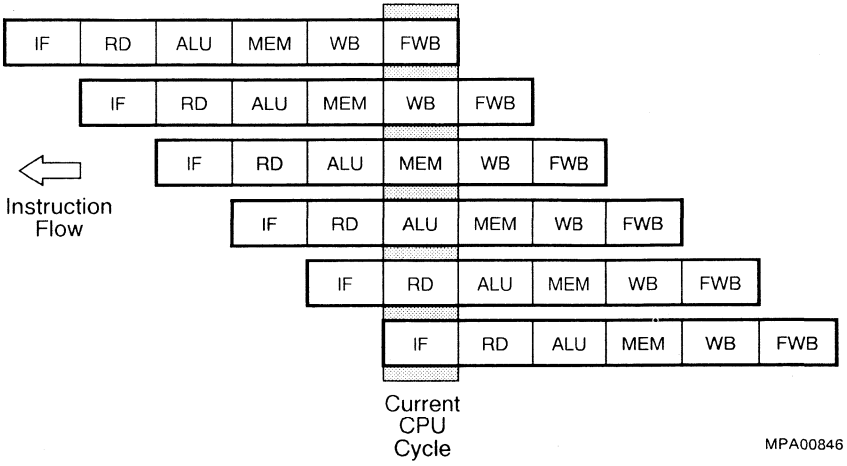
Figure 4
Instruction Execution Sequence



ICACHE	:	Instruction cache access
ID	:	Instruction decode
RF	:	Register operand fetch
IA	:	Instruction address calculation and translation
OP	:	Operation
DA	:	Data address calculation and translation
DCACHE	:	Data cache access
FpWB	:	Floating-point write back to register file
FpWB	:	Floating-point WB

The executions of six instructions are overlapping as shown in figure 5.

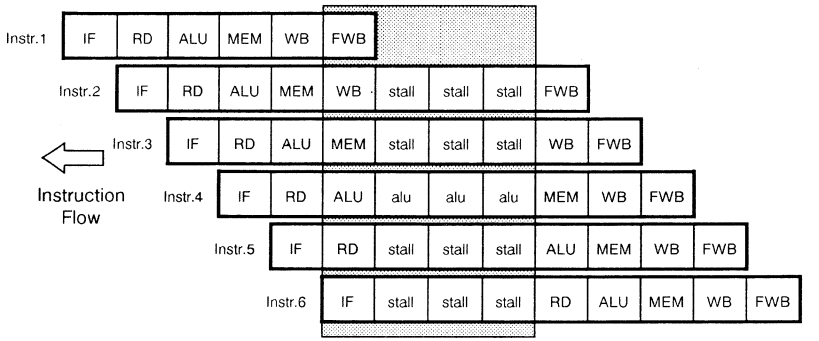
Figure 5
FPA Instruction Pipeline



This is a simplified view of the overlapped instruction execution of the SAB-R2010A because the figure assumes that each instruction can be completed in a single cycle. Most FPA instructions, however, require more than one cycle to be completed. Therefore, the pipeline must be stalled whenever register or resource conflicts occur. Figure 6 illustrates the effect of a three-cycle stall on the SAB-R2010A pipeline.

To alleviate the performance impact that would result from frequently stalling the pipeline, the SAB-R2010A overlaps instructions so that instruction execution can proceed so long as there are no resource conflicts, data dependencies or exceptional conditions.

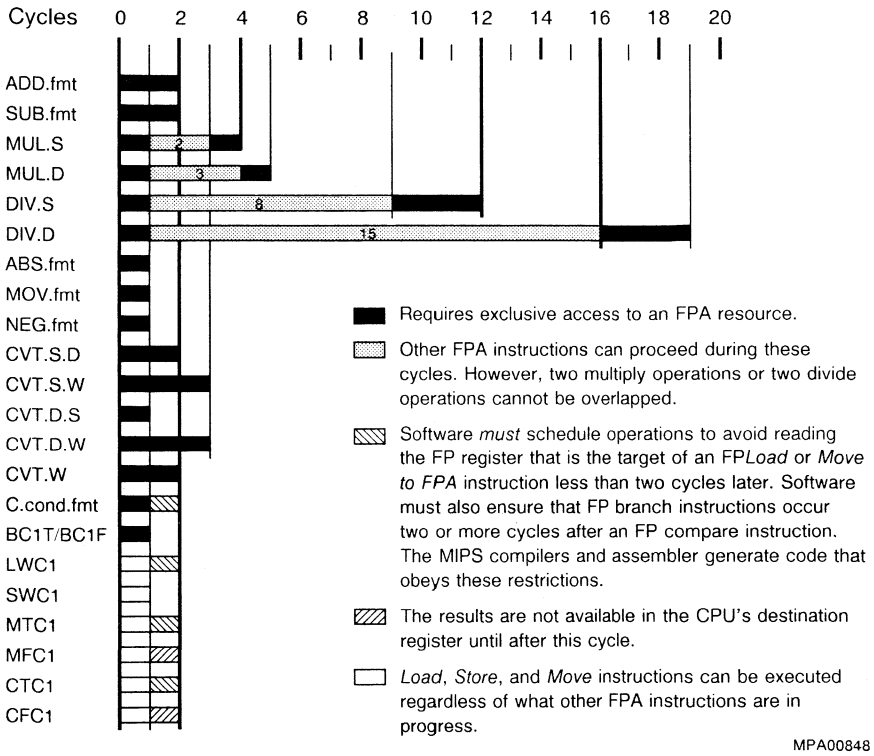
Figure 6
An FPA Pipeline Stall



As mentioned earlier the majority of SAB-R2010A instructions require more than one cycle to be completed. Figure 7 shows the number of cycles required to execute each of the FPA instructions, which varies from 1 to 19 cycles.

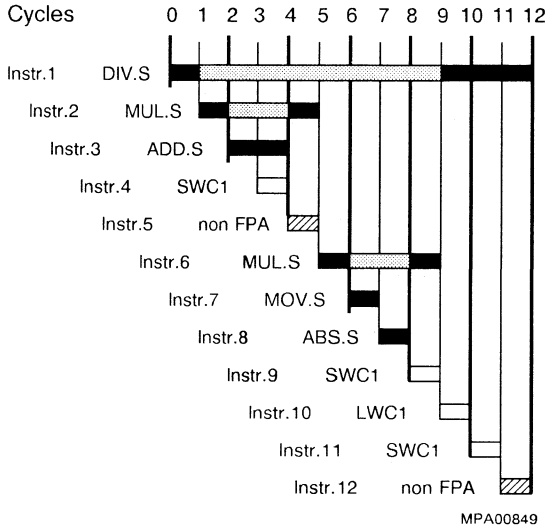
In figure 7 the cycles of an instruction's execution time which are shaded darkest (i.e. at the beginning and at the end of instruction execution time) require exclusive access to an FPA resource (such as the Add unit) that precludes the concurrent use by another instruction and therefore prohibits overlapping execution of another FPA instruction. However, Load and Store operations can be overlapped with these cycles because the SAB-R2010A's register unit can execute memory operations when the other arithmetic units are busy.

Figure 7
FPA Instruction Execution Times



Those cycles that are lightly shaded (i.e. in the middle of the Multiply and Divide instructions execution time) place minimal demands on SAB-R2010A resources (i.e. for a Multiply-instruction only the Multiply unit is being used) and other instructions can be overlapped to obtain simultaneous execution of instructions without stalling the pipeline. However, two Multiply or two Divide operations cannot be overlapped. An example of overlapped FPA and non-FPA instructions is shown in figure 8.

Figure 8
Overlapping FPA Instructions



In this figure the first operation (DIV.S) requires a total of 12 cycles for execution. Only the first and last 3 cycles of this operation preclude the simultaneous execution of another FPA operation. Similarly, in the second operation (MUL.S) there are two cycles in the middle where an FPA operation can be overlapped. In this case the overlapping operation is ADD.S. Although the execution of an instruction requires 6 pipestages, the SAB-R2010A does not require that each instruction complete execution within 6 cycles to avoid stalling the instruction pipeline. If a subsequent instruction does not require the FPA resources being used by a preceding instruction and has no data dependencies with preceding uncompleted instructions, then execution continues. This can be seen clearly in figure 8.

This figure assumes that there are no data dependencies between the instructions that would stall the pipeline. For example, if any instruction before Instr.13 (not shown in figure 8) required the results of Instr.1 (DIV.S), then the pipeline would be stalled until the results are available.

Note: For a detailed discussion of the individual pipestages refer to the SAB-R2000A data sheet.

Coprocessor Registers

Floating-Point Registers

The SAB-R2010A provides thirty-two 32-bit Floating-Point General Registers (FGR's). These are accessed through coprocessor Load/Store instructions and Move to/from coprocessor register instructions. There are two views of the thirty-two coprocessor FGR's. One is from the standpoint of the SAB-R2000A, which has no intrinsic representation of coprocessor registers. It regards these registers as simply thirty-two 32-bit registers. From the standpoint of the SAB-R2010A, pairs of these single word registers form Floating-Point Registers (FPR's), on which floating-point operations are performed. The SAB-R2010A contains 16 FPR's. Figure 9 shows the FGR's and the corresponding FPR's.

The FPR's provide a sufficient amount of registers to support the allocation of floating-point values in registers and to permit overlapping and scheduling of floating-point operations. Each FPR can hold one value of either a single- or double-precision format floating-point number. Only even numbers are used to address FPR's, odd FPR register numbers are invalid. During single-precision floating-point operations, only the even numbered (least) FGR's are used, and during double-precision operations, the FGR's are accessed in pairs. Thus, in double-precision operation, selecting FPR0 addresses FGR0 and FGR1. Table 1 shows the register addresses.

Figure 9
FPA Registers

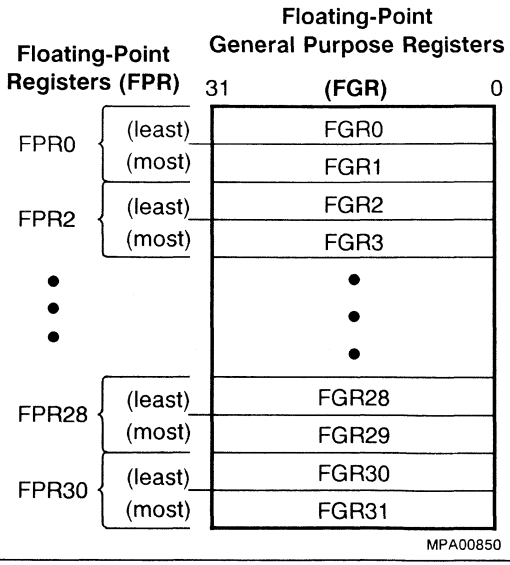


Table 1
Floating-Point General Registers

FGR Number	Usage
0	FPR 0 (least)
1	FPR 0 (most)
2	FPR 2 (least)
3	FPR 2 (most)
•	•
•	•
•	•
28	FPR 28 (least)
29	FPR 28 (most)
30	FPR 30 (least)
31	FPR 30 (most)

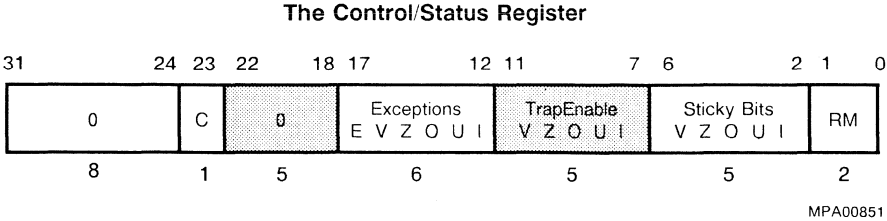
Floating-Point Control Registers

Coprocessors for the SAB-R2000A can have up to thirty-two 32-bit control registers. The SAB-R2010A implements two Floating-Point Control Registers (FCR's). These registers are the Control/Status register (FCR31) and the Implementation/Revision register (FCR0). These registers can only be accessed through Move to/from coprocessor register instructions which address floating-point control registers.

Control/Status Register:
contains control and status data and can be accessed by instructions running in either Kernel or User mode. It controls the arithmetic rounding mode and the enabling of exceptions. It also indicates the exceptions that occurred in the most recently executed instruction, and all exceptions that have occurred since the register was cleared.

Reading this register (using a Move Control From Coprocessor 1 instruction, CFC1), causes all unfinished instructions in the SAB-R2010A's pipeline to be completed before the contents of the register are transferred to the SAB-R2000A. If an exception occurs as the pipeline empties, the exception is taken and the Move instruction can be re-executed after the exception is serviced. Figure 10 illustrates the Control/Status register.

Figure 10
Control/Status Register Bit Assignments



C : Condition bit. Set/cleared to reflect result of Compare instruction; drives the FPA's CpCond output signal.

Exceptions : These bits are set to indicate any exceptions that occurred during the most recent instruction.

TrapEnable : Trap Enables. These bits enable assertion of the CpInt# signal if the corresponding *Exception* bit is set during a floating-point operation.

Sticky bits : These bits are set if an exception occurs and are reset only by explicitly loading new settings into this register (with a Move instruction).

RM : Rounding Mode. These two bits specify which of the four rounding modes is to be used by the FPA.

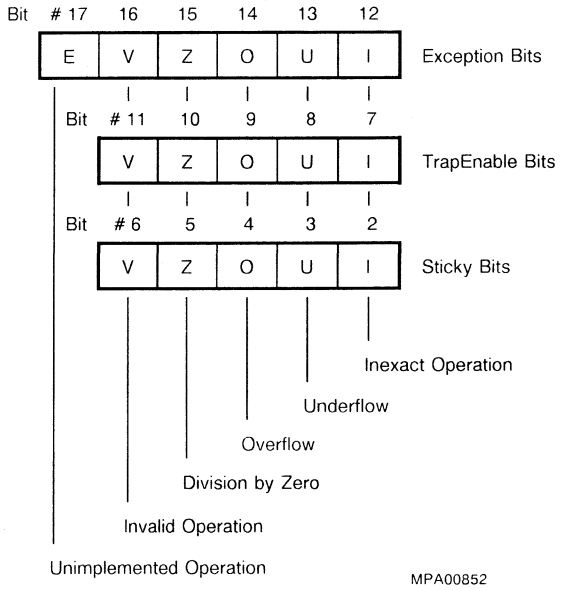
0 : Reserved. Currently ignores writes, undefined when read.

The bits in the Control/Status register can be set or cleared by writing to the register using a Move Control To Coprocessor 1 (CTC1) instruction. This register must only be written to when the FPA is not actually executing floating-point operations. This can be assured by first reading the contents of this register to empty the pipeline.

Condition bit:
When a floating-point Compare instruction takes place, the detected condition is placed at bit 23, the "C" (condition) bit, so that the state of the condition line may be saved or restored. If the condition is true it is set (1) and cleared (0) if it is false. This bit is only affected by Compare and Move To Control Register instructions.

Exception bits:
These are bits 17 through 12 in the Control/Status register, which are shown in figure 11, which indicates the meaning of each bit.

Figure 11
Control/Status Register Exception/Sticky/TrapEnable Bits



These bits are appropriately set or cleared after each floating-point instruction. This is a side effect of each floating-point operation (excluding Loads, Stores and unformatted Moves). The exceptions which were caused by the immediately previous floating-point operation can be determined by reading the exception field.

If two exceptions occur together in one instruction, both appropriate bits in the exception bit field will be set. When an exception occurs, both the corresponding exception and sticky bits are set. The exception bits cover the five IEEE standard exceptions and an extra unimplemented operation exception (E bit). The unimplemented operation exception is not one of the standard IEEE exceptions. It is provided to permit software implementation of IEEE standard operations and exceptions that are not fully supported by the FPA hardware. Trapping on this exception cannot be disabled – there is no TrapEnable bit for E.

Sticky bits:

Hold the accumulated or accured exception bits required by the IEEE standard for trap disabled operation. These bits are set whenever an FPA operation result causes one of the corresponding Exception bits to be set. However, unlike the Exception bits, the Sticky bits are never cleared as a side effect of floating-point operations; they can be cleared only by writing a new value into the Control/Status register.

TrapEnable bits:

Are used to enable a user trap when an exception occurs during a floating-point operation. If the TrapEnable bit corresponding to the exception is set (1) it causes the assertion of the FPA's FpInt# signal. The SAB-R2000A responds to the FpInt# signal by taking an interrupt exception which can be used to implement trap handling of the FPA exception.

Rounding Mode Control bits:

These bits specify the rounding mode the FPA will use for all floating-point operations as shown in table 2.

Table 2
Rounding Mode Bit Decoding

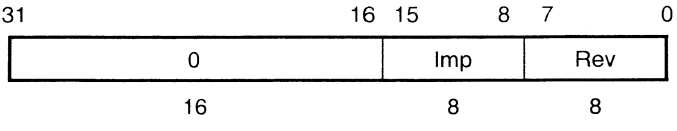
RM Bits	Mnemonic	Rounding Mode Description
00	RN	Rounds result to nearest representable value; rounds to value with least significant bit zero when the two nearest representable values are equally near.
01	RZ	Rounds result toward zero; rounds to value closest to and not greater in magnitude than the infinitely precise result.
10	RP	Rounds toward + ∞; rounds to value closest to and not less than the infinitely precise result.
11	RM	Rounds toward – ∞; rounds to value closest to and not greater than the infinitely precise result.

Implementation and Revision Register:

This read only register, FCR0, contains values that define the implementation and revision number of the SAB-R2010A. This information can be used to determine the co-processor revision and performance level and can also be used by diagnostic software. However, due to the variety of levels at which design changes may be implemented to the silicon, the revision information cannot be guaranteed with every revision of the device nor assured to follow a completely predictable numerical sequence. Siemens has complete discretion over defining these characteristics of the FPA.

Only the low-order bits of the implementation and revision register are defined. Bit 15 through 8 identify the implementation and bits 7 through 0 identify the revision number as shown in figure 12.

Figure 12
Implementation/Revision Register

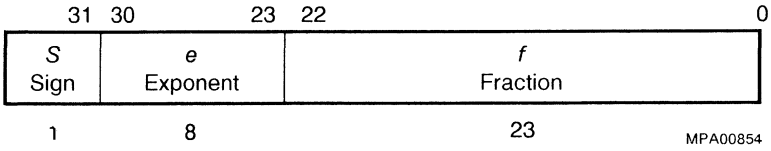


Imp : Implementation: 0 × 10 = R2010
Rev : Revision of FPA
0 : Unused; ignored on writes, zero when read.

Floating-Point Formats

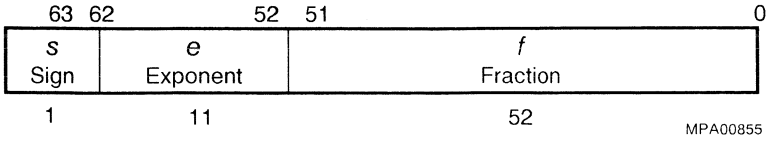
The SAB-R2000A performs both 32-bit (single-precision) and 64-bit (double-precision) IEEE standard floating-point operations. The 32-bit format ist divided into 3 fields: a single-bit sign, an 8-bit biased exponent and a 23-bit fraction, as shown in figure 13.

Figure 13
Single-Precision Floating-Point Format



The 64-bit format has a 1-bit sign, an 11-bit biased exponent and a 52-bit fraction field, as shown in figure 14.

Figure 14
Double-Precision Floating-Point Format



Numbers in the single- and double-precision floating-point formats are composed of three fields;

- A 1-bit sign: s
- A biased exponent: $e = E + bias$
- A fraction: $f = .b_1b_2...b_{p-1}$

The range of the unbiased exponent "E" includes every integer between and including two values "E_{min}" and "E_{max}", and also two other reserved values: "E_{min} - 1" to encode +/- 0 and denormalized numbers, and "E_{max} + 1" to encode +/- ∞ and NaNs (Not a Number). For single- and double-precision each representable non-zero numerical value has just one encoding.

For single- and double-precision formats, the value of a number, "v", is determined by the equations shown in table 3.

Table 3
Equations for Calculating Values in Floating-Point Format

(1)	if $E = E_{max} + 1$ and $f \neq 0$, then v is NAN, regardless of s.
(2)	if $E = E_{max} + 1$ and $f = 0$, then $v = (-1)^S \infty$.
(3)	if $E_{min} \leq E \leq E_{max}$, then $v = (-1)^S 2^E (1.f)$.
(4)	if $E = E_{min} - 1$ and $f \neq 0$, then $v = (-1)^S 2^{E_{min}} (0.f)$.
(5)	if $E = E_{min} - 1$ and $f = 0$, then $v = (-1)^S 0$.

For all floating-point formats, if "v" is NaN, the most significant bit of "f" determines whether the value is a signaling or quiet NaN. "v" is a signaling NaN if the most significant bit of "f" is set; otherwise, "v" is a quiet NaN. Signaling NaNs indicate uninitialized variables or variables for implementing user-designed extensions to the operations provided by the IEEE standard. Quiet NaNs are generated for invalid operations. Table 4 defines the values for the format parameters in the preceding description.

Table 4
Floating-Point Format Parameter Values

Parameter	Single	Double
P	24	53
E_{max}	+ 127	+ 1023
E_{min}	– 126	– 1022
exponent *bias*	+ 127	+ 1023
exponent width in bits	8	11
integer bit	hidden	hidden
fraction width in bits	23	52
format width in bits	32	64

Number Definitions

This subsection contains a definition of the following number types specified in the IEEE 754 standard:

- Normalized Numbers
- Denormalized Numbers
- Infinity
- Zero

Normalized Numbers:

The majority of floating-point calculations are performed on normalized numbers. Normalized numbers have a biased exponent "e" and a normalized fraction field "f" – which means that the leftmost (i.e. the one to the immediate left of the binary point), or hidden, bit is one.

Denormalized Numbers:

Have a zero exponent and a denormalized (hidden bit equal to zero) non-zero fraction field.

Infinity:

Has an exponent of all ones and a fraction field equal to zero. Both positive and negative infinity are supported.

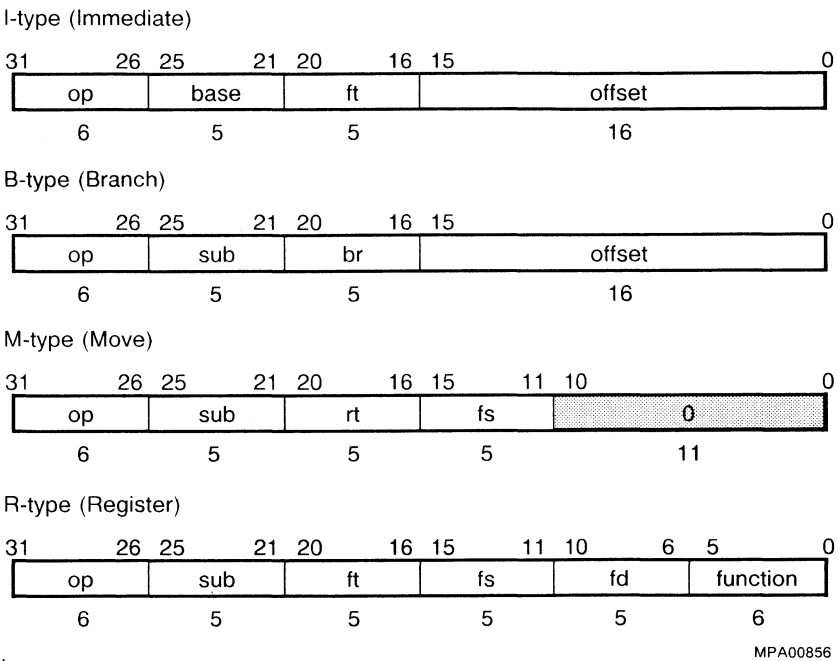
Zero:

Has an exponent of zero, a hidden bit equal to zero and a value of zero in the fraction field. Both positive and negative zero are supported.

Instruction Set Overview

All SAB-R2010A instructions are 32-bits long. There are four basic instruction format types as shown in figure 15.

Figure 15
Instruction Formats



where

op	:	is a 6-bit operation code
sub	:	is a 5-bit sub-operation code
br	:	is a 5-bit branch code
rt	:	is a 5-bit source/destination general register specifier
ft	:	is a 5-bit source/destination float register specifier
fs	:	is a 5-bit source register specifier
fd	:	is a 5-bit destination register specifier
offset	:	is a 16-bit address/branch displacement
function	:	is a 6-bit function code
0	:	result of operation undefined if non-zero

The single instruction length simplifies instruction fetch and decode and eliminates the overhead for instructions crossing word and page boundaries within the memory hierarchy, thereby simplifying the interaction of instruction fetch with the virtual memory management unit. The four instruction formats ensure that opcodes and register descriptors are always found in the same bit locations. This enables register fetch to proceed in parallel with instruction decode on all instructions.

The SAB-R2010A instruction set can be divided into the following groups:

- **Load/Store and Move** instructions move data between memory, the main processor and the FPA general registers.
- **Computational** instructions perform arithmetic operations on floating-point values in the FPA registers.
- **Conversion** instructions perform conversion operations between the various data formats, e.g. floating-point to fixed-point format.
- **Compare** instructions perform comparisons of the contents of registers and set the condition bit based on the results.

Table 5 lists the instruction set of the SAB-R2010A FPA. A more detailed summary is contained in the Instruction Set Summary section.

Table 5
Instruction Set Summary

OP	Description
Load/Store/Move Instructions	
LWC1	Load word to FPA
SWC1	Store word from FPA
MTC1	Move word to FPA
MFC1	Move word from FPA
CTC1	Move control word to FPA
CFC1	Move control word from FPA
Conversion Instructions	
CVT.S.fmt	Floating-point convert to single FP
CVT.D.fmt	Floating-point convert to double FP
CVT.W.fmt	Floating-point convert to fixed-point

OP	Description
Computational Instructions	
ADD.fmt	Floating-point add
SUB.fmt	Floating-point subtract
MUL.fmt	Floating-point multiply
DIV.fmt	Floating-point divide
ABS.fmt	Floating-point absolute value
MOV.fmt	Floating-point move
NEG.fmt	Floating-point negate
Compare Instructions	
C.cond.fmt	Floating-point compare

Exception Handling

This section describes how the SAB-R2010A FPA handles floating-point exceptions. The term exception is used for any infrequent or exceptional event that causes the SAB-R2010A to make a temporary transfer of control from its current process to another process that services the event. A floating-point exception occurs whenever the FPA cannot handle the operands or results of a floating-point operation in the normal way. On the occurrence of an exception the FPA either generates an interrupt (by asserting the signal FpInt#) to initiate a software trap, or sets a flag. If the trap is taken, the FPA remains in the state found at the beginning of the operation (i.e. execution is suspended) and a software exception handling routine is executed. If no trap is taken (i.e. a flag is set), an appropriate value is written into the SAB-R2010A destination register (of the exceptional instruction) and execution continues (see table 6).

The five IEEE standard exceptions are supported with exception bits, trap enables and sticky bits (status flags). Refer to the Control/Status register in the Coprocessor's Registers section. The SAB-R2010A has an additional exception type, unimplemented operation exception (E). This is used in cases where the FPA itself cannot implement the floating-point architecture specification, including cases where the FPA cannot determine the correct exception behaviour. The unimplemented operation exception has no trap enable or sticky bits; whenever this exception occurs, an unimplemented exception trap is taken (if the FPA's interrupt input to the SAB-R2000A is enabled). It is impossible to disable this exception, there is no trap enable bit.

Each of the five IEEE exceptions (Invalid Operation, Division by Zero, Overflow Exception, Underflow Exception and Inexact Operation) is associated with a trap under user control which is enabled by setting one of the five TrapEnable bits. When an exception occurs, both the corresponding Exception and Sticky bits are set. If the corresponding TrapEnable bit is set, the FPA generates an interrupt to the SAB-R2000A and the subsequent exception processing allows a trap to be taken.

Exception Processing

When a floating-point exception trap is taken, the SAB-R2000A processor's Cause register (refer to the SAB-R2000A data sheet) indicates that an external interrupt from the FPA is the cause of the exception and the SAB-R2000A's EPC (Exception Program Counter) contains the address of the instruction that caused the exception trap.

When no exception trap is signalled, a default action is taken, which provides a substitute value for the original, exceptional result of the floating-point operation. The default action taken depends on the type of exception and, in the case of the Overflow exception, the current rounding mode. Table 6 lists the default action taken by the FPA for each of the IEEE exceptions.

Table 6
FPA Exception Default Actions

Exception		Rounding Mode	Default Action (no exception trap signaled)
V	Invalid operation	– –	Supply a quiet NaN.
Z	Division by zero	– –	Supply a properly signed ∞.
O	Overflow	RN	Modify overflow values to ∞ with the sign of the intermediate result.
		RZ	Modify overflow values to the format's largest finite number with the sign of the intermediate result.
		RP	Modify negative overflows to the format's most negative finite number. Modify positive overflows to + ∞.
		RM	Modify positive overflows to the format's largest finite number. Modify negative overflow to – ∞.
U	Underflow	– –	Generate an unimplemented exception.
I	Inexact	– –	Supply a rounded result.

Internally the SAB-R2010A detects eight different conditions that can cause exceptions. When it encounters one of these unusual situations, it will cause either an IEEE exception or an Unimplemented Operation exception. Table 7 lists the exception-causing situations and contrasts the behaviour of the SAB-R2010A with the IEEE standard's requirements.

Table 7
FPA Exception-causing Conditions

FPA internal result	IEEE Stndrd	Trap Enab.	Trap Disab.	Note
Inexact result	I	I	I	loss of accuracy
Exponent overflow	O I')	O I	O I	normalized exponent > E_{max}
Divide by zero	Z	Z	Z	zero is (exponent = E_{min}-1, mantissa = 0)
Overflow on convert	V	V	E	source out of integer range
Signaling NaN source	V	V	E	quiet NaN source produces quiet NaN result
Invalid operation	V	V	E	0/0 etc.
Exponent underflow	U	E	E	normalized exponent < E_{min}
Denormalized source	none	E	E	exponent = E_{min}-1 and mantissa < > 0

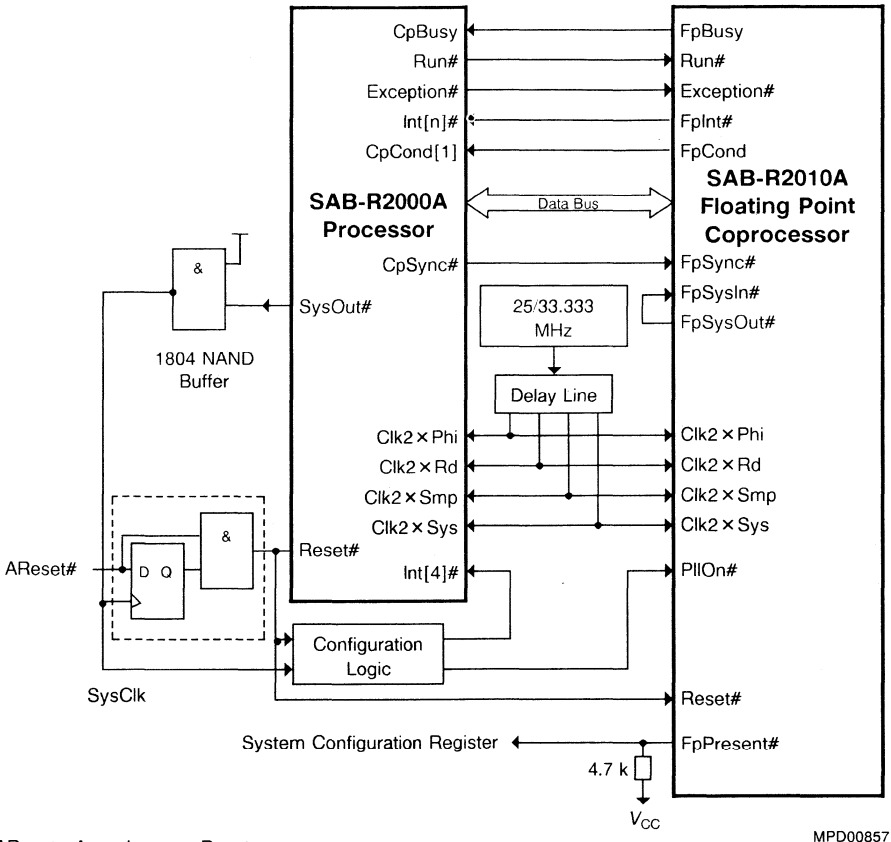
') Standard specifies inexact exception on overflow only if overflow trap is disabled.

Note: *A detailed description of the "exception handling" system for a SAB-R2000A is contained in the SAB-R2000A data sheet.*

Processor Interface

Figure 16 illustrates the tightly coupled coprocessor interface between the SAB-R2000A and the SAB-R2010A.

Figure 16
SAB-R2000 and SAB-R2010 Coprocessor Interface



AReset = Asynchronous Reset

This external coprocessor interface of the SAB-R2000A is designed to support the SAB-R2010A floating point accelerator, in what is called a tightly coupled interface, and up to two additional coprocessors. The SAB-R2010A is connected to the DATA bus only. During each cycle in which a valid Instruction-Data pair is on the bus, the FPA accepts an Instruction. The coprocessor decodes the Instruction in parallel with the main processor and if it is a floating-point Instruction it will proceed to execute the Instruction. The coprocessor condition (CpCond(1) – FpCond) signal allows the main processor to branch on a coprocessor condition set up by a previous operation. The SAB-R2010A can assert FpBusy to stall the main CPU when a floating-point instruction is issued while the FPA still has the required functional unit busy with an earlier operation. The SAB-R2000A asserts Run# to advance operations in the SAB-R2010A. When Run# is deasserted in the n th cycle the FPA disregards the instruction-data pair presented in the n-1 th cycle. The assertion of Exception# indicates that the SAB-R2000A is taking an exception. FpSync# is used for timing synchronization between the SAB-R2000A and the coprocessor.

Instruction Set Summary

The following section is a table of the instructions available in the SAB-R2010A. The instructions are listed in alphabetical order. For a more detailed description of the operation of each instruction refer to the "SAB-R2010A Users Manual". A chart at the end of this section lists the bit encoding for the constant fields of each instruction.

Instruction Notation Convention

The table that follows is split up into three columns: Instruction, format and operation. The instruction column contains the mnemonic name of the instruction and its meaning. The instruction format (refer to figure 15) and assembly language notation for each instruction are listed in the format column. The operation column describes the operation performed by each instruction using a high level language notation. Special symbols used in the notation are described in table 8.

Table 8
FPA Instruction Operation Notations

Symbol	Meaning
←	Assignment
||	Bit string concatenation
x^y	Replication of bit value *x* into a *y*-bit string. Note that *x* is always a single-bit value.
$x_{y..z}$	Selection of bits *y* through *z* of bit string *x*. Little-endian bit notation is always used. If *y* is less than *z*, this expression is an empty (zero length) bit string.
+	Two's complement or floating-point addition
–	Two's complement or floating-point subtraction
*	Two's complement or floating-point multiplication
div	Two's complement integer division
mod	Two's complement modulo
<	Two's complement less than comparison
and	Bitwise logic AND
or	Bitwise logic OR
xor	Bitwise logic XOR
nor	Bitwise logic NOR
GPR[*x*]	SAB-R2000A General Register *x*. Note that the contents of GPR[0] are always zero: attempts to alter GPR[0] contents have no effect.
FGR[*x*]	FPA General Register *x*. As viewed by the R2000A processor.
FPR[*x*]	FPA Floating-Point register *x*. Each FPR is assembled from two FGRs.
FCR[*x*]	FPA Control Register *x*.
T + *i*	Indicates the time steps (CPU cycles) between operations. Thus, operations identified as occurring at T + 1 are performed during the cycle following the one where the instruction was initiated. This type of operation occurs with loads, stores, jumps, branches and coprocessor instructions.
virtualAddress	Virtual address
physicalAddress	Physical address

In the Load/Store operation descriptions, the functions listed in table 9 are used to summarize the handling of virtual addresses and physical memory.

Table 9
Load/Store Common Functions

Function	Description
Addr Translation	Uses the TLB to find the physical address given the virtual address. The function fails and an exception is taken if the entry for the page containing the virtual address is not present in the TLB (Translation Lookaside Buffer).
Load Memory	Uses the cache and main memory to find the contents of the word containing the specified physical address. The low-order two bits of the address and the access type field indicate which of each of the four bytes within the data word need to be returned. If the cache is enabled for this access, the entire word is returned and loaded into the cache.
Store Memory	Uses the cache, write buffer, and main memory to store the word or part of word specified as data into the word containing the specified physical address. The low-order two bits of the address and the access type field indicate which of the four bytes within the data word should be stored.

The mnemonics of the floating-point instructions contain a ".fmt" field. This means "format" and table 10 shows the three formats available.

Table 10
".fmt" field encoding

Mnemonic	Size	Format
S	single	binary floating-point
D	double	binary floating-point
W	single	binary fixed-point

Instruction	Format	Operation
ABS.fmt: Floating-Point Absolute Value	R-Type; ABS.fmt fd, fs	T: StoreFPR (fd, fmt, AbsoluteValue(Value FPR (fs.fmt)));
ADD.fmt: Floating-Point Add	R-Type; ADD.fmt fd, fs, ft	T: StoreFPR (fd, fmt, ValueFPR(fs.fmt) + ValueFPR (ft.fmt));
BC1F: Branch On FPA False (Coprocessor 1)	B-Type; BC1F offset	T: target ← (offset $_{15}$) ^{14} ‖ offset ‖ 0^2 condition ← not CpCond(1) T + 1: If condition then PC ← PC + target endif
BC1T: Branch On FPA True (Coprocessor 1)	B-Type; BC1T offset	T: target ← (offset $_{15}$) ^{14} ‖ offset ‖ 0^2 condition ← CpCond(1) T + 1: If condition then PC ← PC + target endif

C.cond.fmt:
Floating-Point Compare
	R-Type;
	C.cond.fmt fs, ft
		T: if NaN(ValueFPR(fs, fmt)) or NaN(ValueFPR(ft, fmt)) then
		less ← false
		equal ← false
		unordered ← true
		if cond$_3$ then
		signal InvalidOperationException
		endif
		else
		less ← ValueFPR(fs.rmt) < ValueFPR(ft.rmt)
		equal ← ValueFPR(fs.rmt) = ValueFPR(ft.rmt)
		unordered ← false
		endif
		T + 1: condition ← (cond$_2$ and less) or
		(cond$_1$ and equal) or
		(cond$_0$ and unordered)

Instruction	Format	Operation
CFC1: Move Control word from FPA (Coprocessor1)	M-Type; CFC1 rt, ts	T: temp ← FCR[fs]; T + 1: GPR[rt] ← temp;
CTC1: Move Control word to FPA (Coprocessor1)	M-Type; CTC1 rt, fs	T: temp ← GPR[rt]; T + 1: FCR[fs] ← temp
CVT.D.fmt: Floating-Point Convert to Double FloatingPoint Format	R-Type; CVT.D.fmt fd, fs	T: StoreFPR (fd, D, ConvertFmt(ValueFPR(fs, fmt), fmt, D))
CVT.S.fmt: FloatingPoint Convert to Single FloatingPoint Format	R-Type; CVT.S.fmt fd, fs	T: StoreFPR (fd, S, ConvertFmt(ValueFPR(fs, fmt), fmt, S))
CVT.W.fmt: FloatingPoint Convert to FixedPoint Format	R-Type; CVT.W.fmt fd, fs	T: StoreFPR (fd, W, ConvertFmt(ValueFPR(fs, fmt), fmt, W))
DIV.fmt: Floating-Point Divide	R-Type; DIV.fmt fd, fs, ft	T: StoreFPR (fd, fmt, ValueFPR(fs, fmt) / ValueFPR (ft, fmt));
LWC1: Load Word to FPA (Coprocessor 1)	I-Type; LWC1 ft, offset (base)	T: virtualAddress ← (offset $_{15}$) ^{16} ‖ offset $_{15..0}$ + GPR[base]; physicalAddress ← AddressTranslation (virtualAddress); mem ← LoadMemory (WORD, physicalAddress); byte ← virtualAddress $_{1..0}$; T + 1: FGR[ft] ← mem
MFC1: Move from FPA (Coprocessor 1)	M-Type; MFC1 rt, fs	T: temp ← FGR[fs]; T + 1: GPR[rt] ← temp
MOV.fmt: Floating-Point Move	R-Type; MOV.fmt fd, fs	T: StoreFPR (fd, fmt, ValueFPR (fs, fmt));

Instruction	Format	Operation
MTC1: Move to FPA (Coprocessor1)	M-Type; MTC1 rt, fs	T: temp ← GPR[rt]; T + 1: FGR[fs] ← data;
MUL.fmt: Floating-Point Multiply	R-Type; MUL.fmt fd, fs, ft	T: StoreFPR (fd, fmt, ValueFPR(fs, fmt) * ValueFPR (ft, fmt));
NEG.fmt: Floating-Point Negate	R-Type; NEG.fmt fd, fs	T: StoreFPR (fd, fmt, Negate(ValueFPR(fs, fmt)));
SUB.fmt: Floating-Point Subtract	R-Type; SUB.fmt fd, fs, ft	T: StoreFPR (fd, fmt, ValueFPR(fs, fmt) – ValueFPR (ft, fmt));
SWC1: Store Word from FPA (Coprocessor 1)	I-Type; SWC1 ft, offset (base)	T: virtualAddress ← (offset $_{15}$) ^{16} ‖ offset $_{15..0}$ + GPR[base] physicalAddress ← AddressTranslation (virtualAddress); data ← FGR[ft] T + 1: StoreMemory (WORD, data, physicalAddress)

Instruction Encoding

31..29 \ 28..26	0	1	2	3	4	5	6	7
	Opcode							
0	~	~	~	~	~	~	~	~
1	~	~	~	~	~	~	~	~
2	~	COP1	~	~	~	~	~	~
3	~	~	~	~	~	~	~	~
4	~	~	~	~	~	~	~	~
5	~	~	~	~	~	~	~	~
6	~	LWC1	~	~	~	~	~	~
7	~	SWC1	~	~	~	~	~	~

25..24 \ 23..21	0	1	2	3	4	5	6	7
	sub							
0	MF	~	CF	~	MT	~	CF	~
1	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗
2	Single	Double	⊗	⊗	⊗	⊗	⊗	⊗
3	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗

20..19 \ 18..16	0	1	2	3	4	5	6	7
	br							
0	BCF	BCT	~	~	~	~	~	~
1	~	~	~	~	~	~	~	~
2	~	~	~	~	~	~	~	~
3	~	~	~	~	~	~	~	~

5..3 \ 2..0	0	1	2	3	4	5	6	7
	function							
0	ADD.fmt	SUB.fmt	MUL.fmt	DIV.fmt	⊗	ABS.fmt	MOV.fmt	NEG.fmt
1	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗
2	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗
3	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗
4	CVT.S	CVT.D	⊗	⊗	CVT.W	⊗	⊗	⊗
5	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗
6	C.F	C.UN	C.EQ	C.UEQ	C.OLT	C.ULT	C.OLE	C.ULE
7	C.SF	C.NGLE	C.SEQ	C.NGL	C.LT	C.NGE	C.LE	C.NGT

⊗ Codes marked with a '⊗' cause unimplemented operation exceptions and are reserved for future versions of the architecture.

~ Codes marked with a '~' are not valid and are reserved for future versions of the architecture. The results of such an encoding are undefined

Timing Specifications

Absolute Maximum Ratings

Ambient temperature under bias (T_A)	0 to +70 °C
Storage temperature (T_{ST})	– 65 to +150 °C
Supply Voltage (V_{CC})	– 0.5 to +7.0 V
Input voltage (V_{IN})	– 0.5 to +7.0 V

Note: *Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*
Not more than one output should be shorted at a time. Duration of the short should not exceed 30 seconds.

DC Characteristics

$T_A = 0$ to +70 °C; $V_{CC} = 5$ V ±5%

Parameter	Symbol	Limit values				Unit	Test condition
		12.5 MHz		16.67 MHz			
		min.	max.	min.	max.		
Operating Parameters							
Output HIGH voltage	V_{OH}	3.5	–	3.5	–	V	V_{CC} = min. I_{OH} = – 4mA
Output LOW voltage	V_{OL}	–	0.4	–	0.4	V	V_{CC} = min. I_{OL} = 4mA
Input HIGH voltage	V_{IH}	2	V_{CC} + 0.25	2	V_{CC} + 0.25	V	
Input LOW voltage	V_{IL}	– 0.5 [1]	0.8	– 0.5 [1]	0.8	V	
Input HIGH voltage	V_{IHS} [2]	2.5	V_{CC} + 0.25	3.0	V_{CC} + 0.25	V	
Input LOW voltage	V_{ILS} [2]	– 0.5 [1]	0.4	– 0.5 [1]	0.4	V	
Input HIGH voltage	V_{IHC} [3]	4.0	V_{CC} + 0.25	4.0	V_{CC} + 0.25	V	
Input LOW voltage	V_{IL} [3]	– 0.5 [1]	0.4	– 0.5 [1]	0.4	V	
Input capacitance	C_{In}	–	10	–	10	pF	
Output capacitance	C_{Out}	–	10	–	10	pF	
Operating current	I_{CC}	–	550	–	650	mA	V_{CC} = 5.25 V

1) V_{IL} min. = –3.0 V for pulse width less than 15 ns
2) V_{IHS} and V_{ILS} apply to Clk2 × Sys, Clk2 × Smp, Clk2 × Rd, Clk2 × Phi, FpSysIn#, FpSync# and Reset#.
3) V_{IHC} and V_{ILC} apply to Run# and Exception#.

AC Characteristics

T_A = 0 to 70 °C; V_{CC} = 5 V ±5%

Notes: *All output timings are given assuming 25 pf of capacitive load. Output timings should be derated where appropriate as per the table below. All timings referenced to 1.5 V.*

Parameter	Symbol	Limit values				Unit	Test condition
		12.5 MHz		16.67 MHz			
		min.	max.	min.	max.		
Clock Parameters 4)							
Input clock high	$t_{ClkHigh}$	16	–	12	–	ns	Transition ≤ 5 ns
Input clock low	t_{ClkLow}	16	–	12	–	ns	Transition ≤ 5 ns
Input clock period	t_{ClkP}	40	1000	30	1000	ns	
Clk2 × Sys to Clk2 × Smp		0	$t_{\frac{Cyc}{4}}$	0	$t_{\frac{Cyc}{4}}$	ns	
Clk2 × Smp to Clk2 × Rd		0	$t_{\frac{Cyc}{4}}$	0	$t_{\frac{Cyc}{4}}$	ns	
Clk2 × Smp to Clk2 × Phi		11	$t_{\frac{Cyc}{4}}$	9	$t_{\frac{Cyc}{4}}$	ns	
Run Operation Parameters							
Data enable	t_{DEn}	– 1	– 2.5	– 1	– 2	ns	–
Data disable	t_{DDis}	0	– 1	0	– 1	ns	–
Data valid	t_{DVal}	–	3.5	–	3	ns	25 pF load
Data setup	t_{DS}	11.5	–	9	–	ns	–
Data hold	t_{DH}	– 2.5	–	– 2.5	–	ns	–
FpCondition	t_{FpCond}	0	45	0	35	ns	–
FpBusy	t_{FpBusy}	0	20	0	15	ns	–
FpInterrupt	t_{FpInt}	0	55	0	40	ns	–
FpMove To	t_{FpMov}	0	45	0	35	ns	–
Exception setup	t_{ExS}	15	–	10	–	ns	–
Exception hold	t_{ExH}	0	–	0	–	ns	–
Run setup	t_{RunS}	15	–	10	–	ns	–
Run hold	t_{RunH}	– 2	–	– 2	–	ns	–
Capacitive Load Deration							
Load derate	C_{LD}	0.5	2.5	0.5	2	ns/ 25pF	

4) The clock parameters apply to all four 2xClocks: Clk2 × Sys, Clk2 × Smp, Clk2 × Rd, and Clk2 × Phi.

Operation Fundamentals

A "cycle" is the basic instruction processing unit of the SAB-R2010A processor. Cycles in which forward progress is made, i.e. an instruction is retired, are called "run" cycles. An instruction is retired either by its completion or, in the presence of an exception, its abortion. Cycles in which no forward progress is made are called "stall" cycles. Stall cycles are used for resolving urgent situations such as cache misses on loads, write system busy during stores, and coprocessor interlocks. All cycles can be classified as either run cycles or stall cycles. "Fixup" stall cycles occur during the final cycle of the stall and are used in general to fix up the conditions which caused the stall. Processor transactions which occur during the first half of the cycle are called phase 1 transactions while those which occur during the second half of the cycle are called phase 2 transactions.

As described earlier Run# is asserted by the SAB-R2000A during run cycles and deasserted during stall cycles. When Run# is deasserted during the n th cycle, the SAB-R2000A disregards the instruction-data pair presented during the n-1 th cycle. When Run# is reasserted during the m th cycle, the SAB-R2010A takes, as are placement for the instruction-data pair which was disregarded, the instruction-data pair presented during the m-1 th cycle – which was the final fixup cycle for whatever stalls equence was occurring.

Exception# is used by the FPA to track exception related information during run cycles and stall related information during stall cycles.

- During phase 1 of run cycles Exception# indicates whether an exception has occurred for the instruction which is currently in its "write back" pipestage. Unless the exception is occurring as a result of an interrupt request by the SAB-R2010A, the assertion of Exception# prevents any state from being committed in the FPA.
- During phase 2 of run cycles Exception# indicates whether an interrupt request is being granted for the instruction which is currently in its "memory access" pipestage.
- During phase 1 of stall cycles Exception# indicates whether the current stall cycle is a fixup cycle. When a fixup cycle is occurring, it is guaranteed that the data present on the data bus is valid. The FPA uses the fixup indication to qualify the use of data sampled from the bus during the stall.
- During phase 2 of stall cycles Exception# indicates whether the current stall is a Coprocessor Busy stall. The FPA does not use this information.

The use of the Exception# signal is summarized below.

Table 11
Exception#

Cycle	Phase 1	Phase 2
Run	Exc1W#	IntGr2M#
Stall	Fixup1#	CPBusy2#

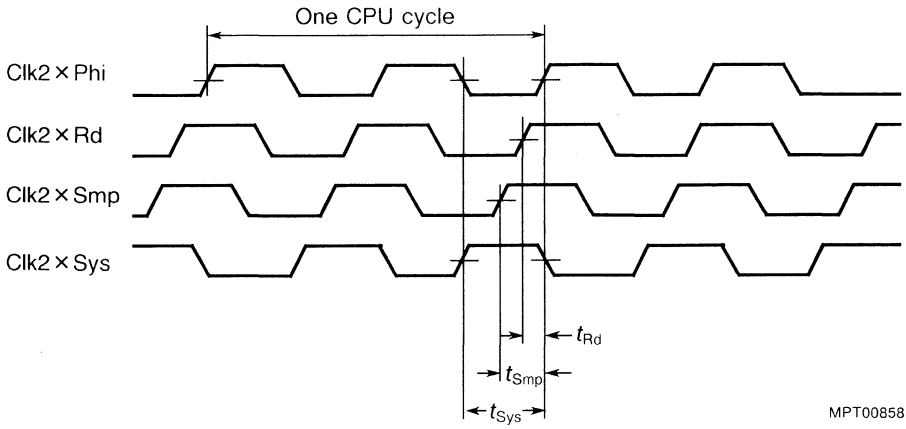
Processor Input Clocks

The SAB-R2010A has the same four separate double-frequency (i.e. in a 16.67 MHz system these clocks are 33.33 MHz) input clocks as the SAB-R2000A. They can be adjusted to obtain optimum positioning of cache interface signals. The absolute timing of these clocks with respect to the processor outputs is undefined, only the differences are important. A short description of these four clocks follows. Refer to figure 16 for the various signal names.

- Clk2 × Sys:
  is the master clock and must lead all others. It determines the position of SysOut# (the processors output clock) with respect to data, Tag and Address buses.
- Clk2 × Smp:
  determines the sample point for data coming into the SAB-R2000A on all its inputs except those coming directly from coprocessors.
- Clk2 × Rd:
  controls output enable time and provides sufficient address access to sample address hold from end of write, and data hold from end of write.
- Clk2 × Phi:
  determines the position of the internal phases 1 and 2.

Figure 17 shows the four 2 × input clocks.

Figure 17
2 × Input Clocks



In the timing diagrams which follow, timing specifications are given relative to a shifted version of the FPA output clock, FpSysOut#. The clock is called FpPhiOut# and is a virtual clock, i.e. the processor does not actually produce this output (FpSysOut# and FpPhiOut# are equivalent to SysOut# and PhiOut# for the SAB-R2000A). It is shown in the timing diagrams for reasons of clarity, because its period is synchronous with a machine cycle. The shift amount is equal to the difference between Clk2 × Sys to Clk2 × Phi and, as is shown in figure 17, is t_{Sys}. Also in the timing diagrams Clk2 × Sys and FpSysOut# are shown to clarify the relationship between these signals.

In reality FpSysOut# is produced rather than FpPhiOut# since this provides a signal with timing appropriate for synchronizing system transactions to the processor. Timings are given relative to FpPhiOut# since this makes determining the position of the input clocks the most straightforward. The timing of any output with respect to FpSysOut# can be determined from its timing with respect to FpPhiOut# by adding t_{Sys}.

Timing Diagram Notation

The following timing diagrams describe various transactions of the processor. Table 12 illustrates the notational conventions used in these diagrams.

Table 12
Notational Conventions for Timing Diagramms

Character	Meaning
I	Instruction
D	Data
#	Active low
%	An incorrect datum
!	An unused datum
Z	The high impedance state
Ad	Address
in	into coprocessor
out	out of coprocessor
//////	not valid or Don't Care

Load/Store and Processor Transfer Timings

During run cycles, the operation and timing of FPA loads and stores are identical to that of the SAB-R2000A. In the case of a load the SAB-R2010A accepts data from the data bus and on stores it drives data on the data bus. Both of these data bus transactions occur during the MEM pipestage of each instruction (refer to figure 4 in the pipeline architecture section). On FPA loads, the SAB-R2000A reads in the data and Tag buses for purposes of miss detection. The Tag bus is the cache Tag bus and is only connected to the CPU – for more information refer to the SAB-R2000A data sheet. For miss detection the SAB-R2000A checks the valid bit, does the Tag comparison and checks parity on the Tag and data buses. On Stores the SAB-R2010A generates data parity. All address generation, cache and memory control functions are provided by the SAB-R2000A.

During all stall and fixup cycles, the FPA is passive; if an FPA Store is blocked by a write busy stall or if the cycle in which the FPA Store occurs is redone due to any other stall, the SAB-R2000A will re-present the FPA data during the stall's fixup cycle. Timing of FPA Loads/Stores is illustrated in figure 19.

Transfers between the SAB-R2010A and the SAB-R2000A have identical input and output characteristics as Loads and Stores. That is, for a Move to FPA transfer (MTC1), the SAB-R2000A drives the data bus and the SAB-R2010A reads it, as for an FPA Load. For a Move from FPA instruction (MFC1) the roles are reversed, as for an FPA Store. Parity is not checked for either direction of transfer. The timings for these transfers are also illustrated in figure 19.

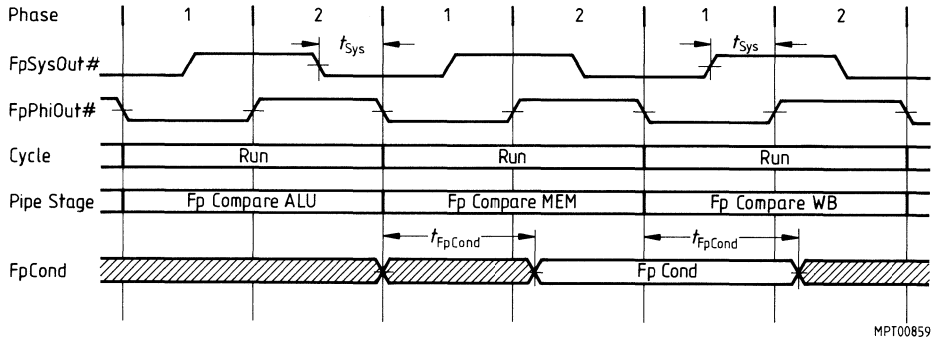
Floating-Point Condition Timing

Floating-point operations occur within the SAB-R2010A and only affect the interface when they change the Floating-Point Condition output or cause stalls or exceptions. Floating-point conditions are described here, stalls and exceptions are described in subsequent sections.

The SAB-R2010A has a Floating-Point Condition output called FpCond. The FpCond output is connected directly to the CpCond(1) input of the SAB-R2000A, refer to figure 16. The FpCond signal is sampled by the SAB-R2000A during phase 2 of every run cycle. If the SAB-R2000A executes an FPA branch instruction, the state of the FpCond signal determines the direction of the branch.

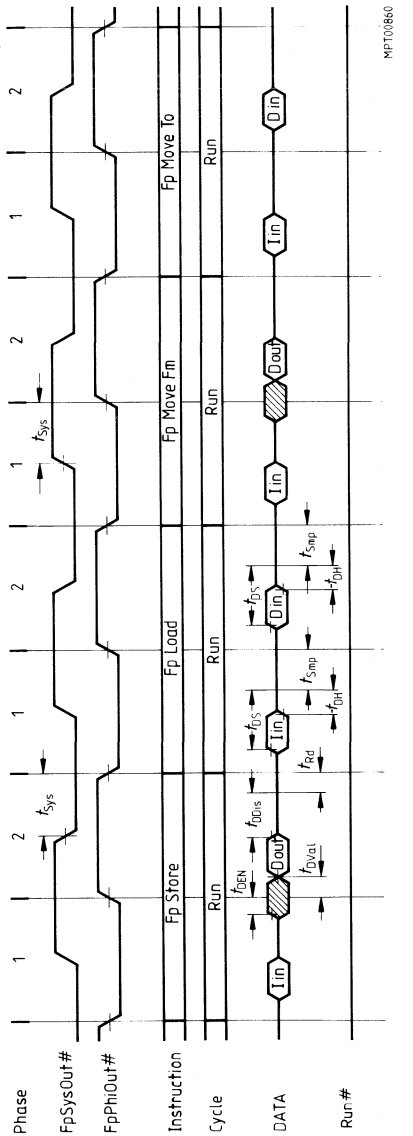
Floating-point instructions which affect the FpCond signal are two-cycle operations (e.g. Floating-Point Compare). This can be seen externally by the invalidity of the FpCond output during the entire ALU pipestage of Instruction Execution, refer to figure 18 which illustrates the FpCond timing. The FpCond output becomes valid during the MEM pipestage of instruction execution, which is too late to be used by the succeeding instruction, therefore the operation requires two cycles. Refer to the Pipeline Architecture section and the SAB-R2000A data sheet for more details on two cycle instructions.

Figure 18
FPCond Timing



Note: The Instruction used here to illustrate the FpCond timing is a Floating-Point Compare Instruction, where FpCompare ALU/MEM/WB are the Floating-Point Compare ALU/MEMORY/Write Back pipestages respectively.

Figure 19
Load/Store and Transfer Timing



FpMoveFm (FpMoveTo) has the same timing as FpStore (FpLoad).

Floating-Point Coprocessor Stall Timing

As described earlier, to maintain synchronization with the SAB-R2000A the FPA requests "Coprocessor Busy" stalls as required. To initiate such a stall the SAB-R2010A asserts FpBusy during phase 2 of the ALU pipestage of the stalling FPA instruction. To terminate the stall FpBusy is deasserted during phase 1 of the stall cycle in which it will complete the operation whose incompletion required the stall. In the absence of other stall requests, the cycle following that in which FpBusy is deasserted will be the fixup cycle. Figure 20 illustrates the FPA busy timing.

For all stalls, whether FPA-initiated or not, the indication of the stall condition is signalled by the SAB-R2000A via the deassertion of the Run# signal. If Run# is not deasserted following the assertion of FpBusy, then the FPA stall request has been ignored by the SAB-R2000A due to the occurrence of some exceptional event.

Exception/Interrupt Timing

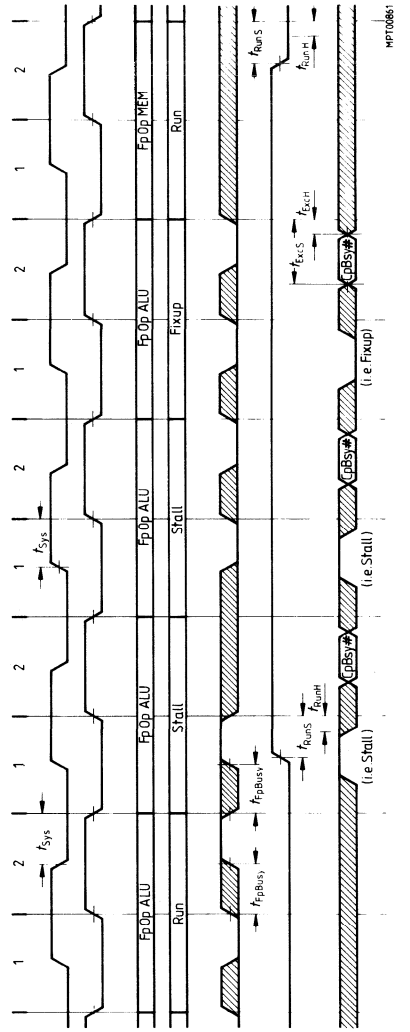
The SAB-R2010A signals exceptions to the SAB-R2000A through one of its interrupt inputs using the FpInt# output, refer to figure 16. The SAB-R2000A samples the interrupt inputs during phase 2 of every run cycle and final fixup cycle of a stall sequence. The FPA signals exceptions by asserting FpInt# during the ALU pipestage of the instruction causing the exception. If the SAB-R2000A takes the interrupt during that cycle, it signals interrupt grant (IntGr2M#) back to the FPA (via the Exception# signal) during the MEM pipestage of the exceptional instruction. Interrupt grant is signalled to the FPA on its Exception# input during phase 2 of the MEM pipestage.

The occurrence of any exception, including those caused by the FPA, is signalled to the FPA during the WB pipestage of the exception-causing instruction. The occurrence of an exception prevents any non-exception-related state from being committed within the FPA. This means that when an exception occurs which is not FPA related, execution in the FPA is suspended until the exception is resolved. The occurrence of an exception is signalled to the FPA on its Exception# input during phase 1 of the WB pipestage of the exceptional instruction. Figure 21 illustrates an Interrupt timing sequence.

Special Case

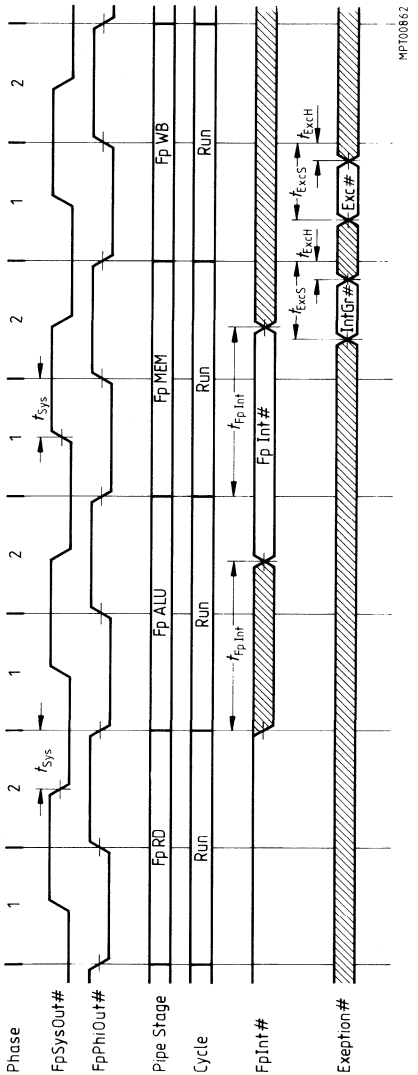
A special case of FPA – SAB-R2000A transfers is the MoveTo FPA Status register. When moves to this register occur the interface can be further affected via a change in the FpInt# or FpCond outputs. Figure 22 illustrates the timing of the FpCond and FpInt# outputs in conjunction with an FPA MoveTo instruction.

Figure 20
Busy Timing



Note: CpBusy# is the same as CpBusy2# in table 12.

Figure 21
Interrupt and Exception Timing



Where: IntGr# = IntGr2M# and Exc# = Exc1W# in table 12.

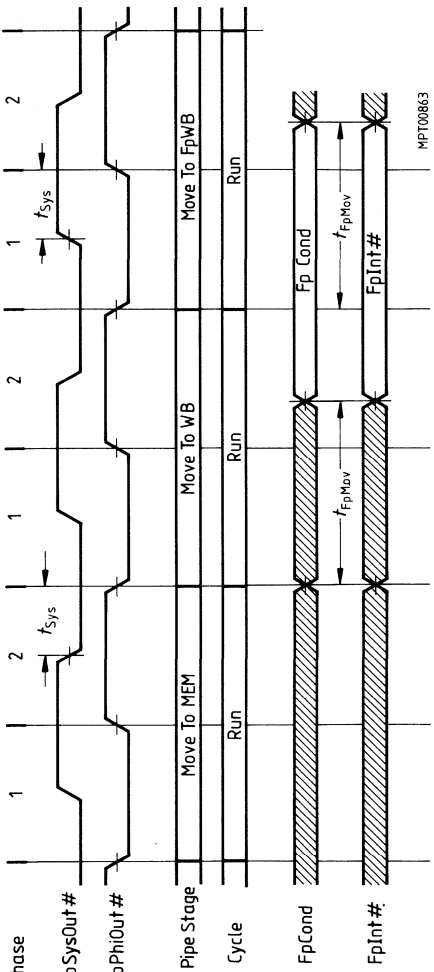


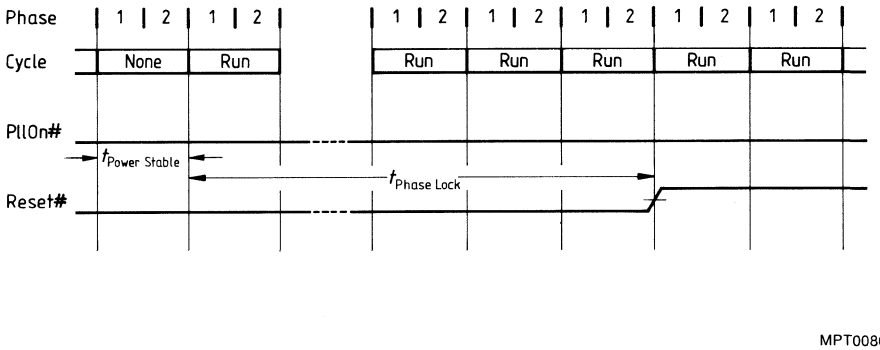
Figure 22
Move to FPA Status Timing

SAB-R2000A – FPA Timing Synchronization

The processor – coprocessor (SAB-R2000A – SAB-R2010A) system requires that there be minimum timing skew between the SAB-R2000A and FPA to operate at maximum speed. To facilitate deskewing, a phase lock loop is provided on the FPA.

This synchronization must be achieved during the reset period to ensure that clock skew is acceptably small when the first instruction is fetched. During the reset period, the SAB-R2010A's phase lock circuitry acquires and locks to the SAB-R2000A's output clock. For correct operation, the CPU – FPA system must remain in reset for 3000 clocks or 200 microseconds after power is stable, whichever is longer. If the phase lock mechanism is not enabled, the reset period can be shortened to 128 clock cycles after power is stable. Figure 23 illustrates the required reset sequence for the case where the phase lock mechanism is enabled.

Figure 23
Reset Sequence



Note: PllOn# must be asserted continuously after Reset# is deasserted for correct operation of the SAB-R2010A – SAB-R2000A system.

SIEMENS

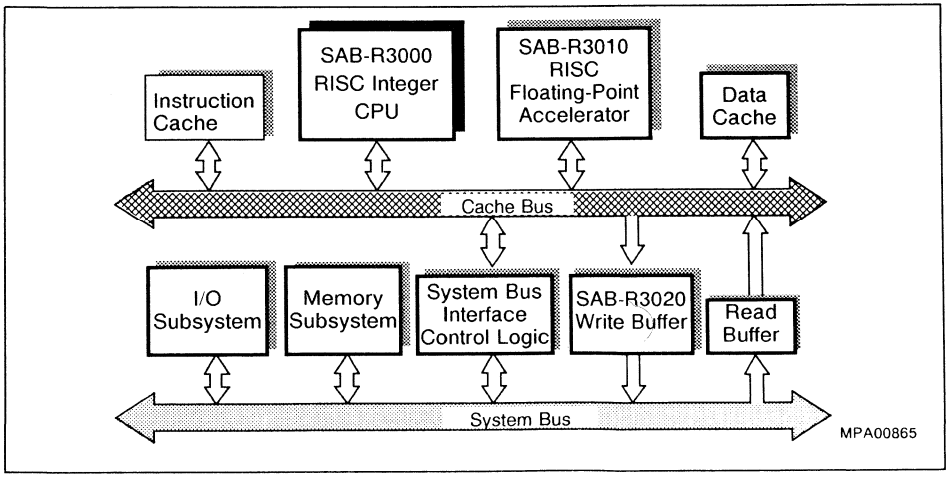
High- Performance 32-Bit RISC Microprocessor

SAB-R3000

including on-chip Memory Management and Cache Control with support for up to three external coprocessors including the SAB-R3010 Floating-Point Accelerator

Advance Information

- Two tightly-coupled 25 MHz units on a single chip
  - full 32-bit RISC CPU
  - system control processor (CP0)
- Load/Store architecture
  - support for loading misaligned data
  - configurable endianness
- Full 32-bit operation
  - thirty two general purpose 32-bit registers
  - all instructions and addresses are 32-bits
- On-chip cache control
- High-Performance
  - 20-22 VAX 11/780 mips average at 25 MHz
- On-chip Memory Management Unit
- Extensive Software and Development Support
- Instruction set compatible to R2000 processors
- Fully pin- and functionally compatible to all R3000 processors from other manufacturers
- Ceramic packages: C-QFP-172, C-PGA-145



7.90

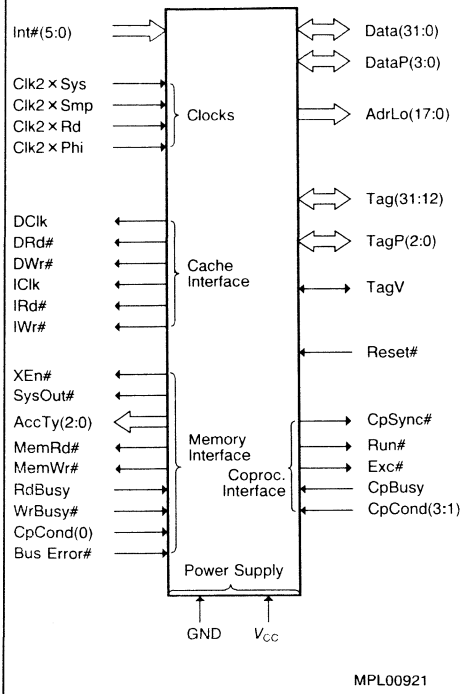
Ordering Information

Type	Ordering code	Package	Function
SAB-R3000-16-A	Q67120-C554	C-PGA-145	32-bit RISC Processor, 16.67 MHz
SAB-R3000-20-A	Q67120-C515	C-PGA-145	32-bit RISC Processor, 20 MHz
SAB-R3000-20-QF	Q67120-C519	C-QFP-172	32-bit RISC Processor, 20 MHz
SAB-R3000-25-A	Q67120-C514	C-PGA-145	32-bit RISC Processor, 25 MHz
SAB-R3000-25-QF	Q67120-C496	C-QFP-172	32-bit RISC Processor, 25 MHz

Introduction

The SAB-R3000 is a high-performance microprocessor architecture implemented as a full-custom CMOS VLSI chip which achieves 22 VAX 11/780 mips average at 25 MHz. It is a single chip microprocessor that consists of two tightly-coupled 25 MHz units. The first is a full 32-bit RISC CPU. The second unit is a System Control Processor (CP0) that integrates the functions needed to keep the CPU from idling for memory access (Memory Management) and/or for Interrupt and Exception handling. The System Control Processor contains a Translation Lookaside Buffer (TLB) and control registers to efficiently support a virtual memory system, as well as all the control logic to realize separate caches for instructions and data. This architecture permits a dual-cache bandwidth of up to 200 Mbytes/second at 25 MHz using standard SRAM devices. It is possible for up to three external coprocessors (including the SAB-R3010 floating point accelerator) to be coupled with the SAB-R3000. The synchronous coprocessor interface generates all the addresses and manages the memory interface control, which minimises the ammount of glue logic required to build an SAB-R3000 based system.

Figure 1
Logic Symbol



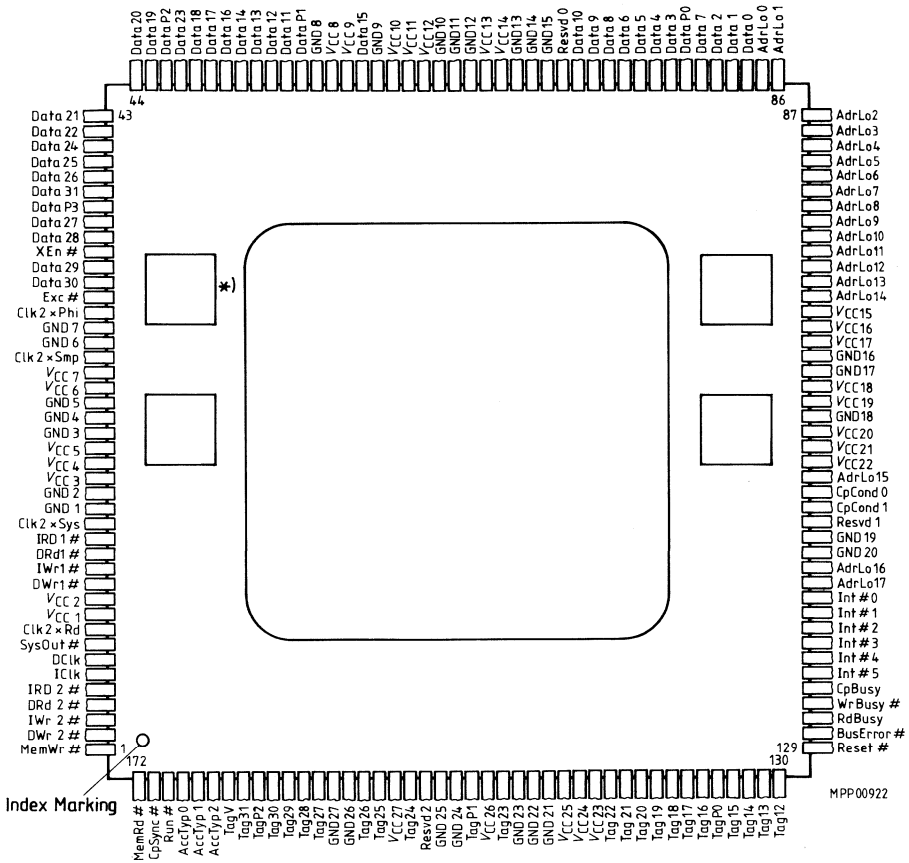
Pin Names

Data(31:0)	Data Bus
DataP(3:0)	Even Parity for Data Bus
AdrLo(17:0)	Low Adress Bus
Tag(31:12)	Cache Tag and High Adress Bus
TagP(2:0)	Even Parity over TagV and Tag Bus
TagV	Tag Validity Indicator
Reset#	Synchronous Initialization
CpSync#	Coprocessor Synchronization
Run#	Processor in Run or Stall State
Exc#	Exception
CpBusy#	Coprocessor Busy
CpCond(3:0)	Coprocessor Condition
BusError#	Bus Error
WrBusy#	Write Busy for Main Memory
RdBusy#	Read Busy for Main Memory
MemWr#	Main Memory Write
MemRd#	Main Memory Read
AccTy(2:0)	Access Type
SysOut#	System Clock Out
XEn#	Read Enable (Read Buffer)
DWr	Data Cache Write Enable
IWr	Instruction Cache Write Enable
DRd	Data Cache Read Enable
IRd	Instruction Cache Read Enable
DClk#	Data Cache Latch Enable
IClk#	Instruction Cache Latch Enable
Int#(5:0)	Interrupt Bus

Note: "#" signifies an active low signal.

Pin Configurations

Figure 2
C-QFP-172 Cavity Up (Top View)



*) V_{CC} and GND connections for capacitors

Pin Configuration (cont'd)

Figure 3
C-PGA-145 Cavity Up (Top View)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	V_{CC14}	AdrLo 6	AdrLo 10	AdrLo 11	V_{CC12}	AdrLo 14	AdrLo 15	CpCond 0	AdrLo*) 16	AdrLo*) 17	Intr# 2	Intr# 5	Wr Busy#	Reset#	V_{CC10}
B	AdrLo 3	DRd2#	AdrLo 7	AdrLo 9	AdrLo 12	IRd2#	AdrLo 13	CpCond 1	Intr# 1	Intr# 3	Cp Busy	Bus Error#	DWr2#	Tag12	Tag15
C	AdrLo 0	AdrLo 4	V_{CC13}	AdrLo 5	AdrLo 8	Gnd13	Gnd12	V_{CC11}	Intr# 0	Intr# 4	Rd Busy	Gnd11	Tag13	TagP0	Tag18
D	Data 1	AdrLo 2	Gnd0	Index Pin									Tag14	Tag17	Tag19
E	DataP 0	Data 0	AdrLo 1										Tag16	Tag20	V_{CC9}
F	V_{CC0}	Data 7	Data 2										Gnd10	Tag21	Tag23
G	Data 4	Data 3	Gnd1										Gnd9	Tag22	TagP1
H	Data 6	Data 5	Data 8										V_{CC8}	Tag25	Tag24
J	Data 10	DataP 1	Data 9										Tag28	Tag29	Tag26
K	Data 15	Data 11	Gnd2										Gnd8	TagP2	Tag27
L	V_{CC1}	Data 12	Data 17										Acc Ty2	Tag31	Tag30
M	Data 13	Data 16	DataP 2										Gnd7	Acc Ty1	V_{CC7}
N	Data 14	Data 18	Data 19	Gnd3	Data 24	Data P3	V_{CC3}	V_{CC4}	Gnd5	Gnd6	DRd1#	MemWr #	MemRd #	Run#	TagV
P	Data 23	Data 20	IWr2#	Data 22	Data 26	Data 27	XEn#	Data 30	Clk2× Sys	Clk2× Rd	DClk	IRd1#	IWr1#	Cp Sync#	Acc Ty0
Q	V_{CC2}	Data 21	Data 25	Data 31	Data 28	Gnd4	Data 29	Excep tion#	Clk2× Phi	Clk2× Smp	SysOut #	V_{CC5}	IClk	DWr1#	V_{CC6}

MPP00868

*) AdrLo 16 & 17: are multi-function pins which are controlled by mode select programming on the interrupt pins at reset time.

AdrLo 16: MP_Invalidate, CpCond (2).
AdrLo 17: MP_Stall, CpCond (3).

Pin Definitions and Functions

Symbol	Pin Number		I/O *)	Function
	144-Pin	172-Pin		
Data(31:0)	Q4,P8,Q7, Q5,P6,P5, Q3,N5,P1, P4,Q2,P2, N3,N2,L3, M2,K1,N1, M1,L2,K2,J1, J3,H3,F2,H1, H2,G1,G2, F3,D1, E2	38,32,33,35, 36,39,40,41, 47,42,43,44, 45,48,49,50, 59,51,52,53, 54,73,74,75, 81,76,77,78, 79,82,83,84	I/O	A 32-bit bus used for all instruction and data transmission among the processor, caches, memory interface, and coprocessors.
DataP(3:0)	N6,M3,J2,E1	37,46,55,80	I/O	A 4-bit bus containing even parity over the data bus.
Tag(31:12)	L14,L15,J14, J13,K15,J15, H14,H15, F15,G14, F14,E14, D15,C15, D14,E13, B15,D13, C13,B14	165,163,162,1 61,160,157,15 6,154,148,141 ,140,139,138, 137,136,135,1 33,132,131,13 0	I/O	A 20-bit bus used for transferring cache tags and high addresses between the processor, caches, and memory interface.
TagV	N15	166	I/O	The tag validity indicator
TagP(2:0)	K14,G15, C14	164,150,134	I/O	A 3-bit bus containing even parity over the catenation of TagV and Tag(31:12).
AdrLo(17:0)	A10,A9,A7, A6,B7,B5, A4,A3,B4, C5,B3,A2, C4,C2,B1, D2,E3,C1	118,117,111, 99,98,97,96, 95,94,93,92, 91,90,89,88, 87,86,85	O	An 18-bit bus containing byte addresses used for transferring low addresses from the processor to the caches and memory interface. AdrLo 16 and 17 are Multi-function pins, AdrLo(16): MP_INVALIDATE, CpCond2 AdrLo(17): MP_Stall, CpCond3

*) I = Input
O = Output

Pin Definitions and Functions (cont'd)

Symbol	Pin Number		I/O *)	Function
	144-Pin	172-Pin		
IRd1#	P12	15	O	Read enable for the instruction cache.
IWr1#	P13	13	O	Write enable for the instruction cache.
IRd2#	B6	5	O	An identical copy of IRd1# used to split the load.
IWr2#	P3	3	O	An identical copy of IWr1# used to split the load.
IClk	Q13	6	O	The instruction cache address latch clock. This clock runs continuously.
DRd1#	N11	14	O	The read enable for the data cache.
DWr1#	Q14	12	O	The write enable for the data cache.
DRd2#	B2	4	O	An identical copy of DRd1# used to split the load.
DWr2#	B13	2	O	An identical copy of DWr1# used to split the load.
DClk	P11	7	O	The data cache address latch clock. This clock runs continuously.
XEn#	P7	34	O	The read enable for the Read Buffer.
AccTy(2:0)	L13,M14,P15	167,168,169	O	A 3-bit bus used to indicate the size of data being transferred on the data bus, whether or not a data transfer is occurring, and the purpose of the transfer. The run encoding of the Access Type is illustrated in the table below.

AccTy(2)	AccTy(1:0)	size
1	XX	no transaction
0	00	byte
0	01	half word
0	10	tribyte
0	11	word

*) I = Input
O = Output

Pin Definitions and Functions (cont'd)

Symbol	Pin Number		I/O *)	Function
	144-Pin	172-Pin		
MemWr#	N12	1	O	Signals the occurrence of a main memory write.
MemRd#	N13	172	O	Signals the occurrence of a main memory read.
BusError#	B12	128	I	Signals the occurrence of a bus error during a main memory read or write.
Run#	N14	170	O	Indicates whether the processor is in the run or stall state.
Exc#	Q8	31	O	Indicates that the instruction about to commit state should be aborted and other exception related information.
SysOut#	Q11	8	O	A reflection of the internal processor clock used to generate the system clock.
CpSync#	P14	171	O	A clock which is identical to SysOut# and used by coprocessors for timing synchronization with the CPU.
RdBusy	C11	127	I	The main memory read stall termination signal. In most system designs RdBusy is normally asserted and is deasserted only to indicate the successful completion of a memory read. RdBusy is sampled by the processor only during memory read stalls.
WrBusy#	A13	126	I	The main memory write stall initiation/termination signal.
CpBusy	B11	125	I	The coprocessor busy stall initiation/termination signal.
CpCond(1:0)	B8,A8	113,112	I	A 2-bit bus used to transfer conditional branch status from the coprocessors to the main processor.

*) I = Input
O = Output

Pin Definitions and Functions (cont'd)

<table>
<tr><th rowspan="2">Symbol</th><th colspan="2">Pin Number</th><th rowspan="2">I/O *)</th><th rowspan="2">Function</th></tr>
<tr><th>144-Pin</th><th>172-Pin</th></tr>
<tr><td>Int#(5:0)</td><td>A12,C10,
B10,A11,
B9,C9</td><td>124,123,122,
121,120,119</td><td>I</td><td>A 6-bit bus used by the memory interface and coprocessor to signal maskable interrupts to the processor. It is also used to specify the processor's mode during Reset. The Table below summarizes the mode selectable features
<table>
<tr><th>Int#</th><th>WCycle</th><th>XCycle</th><th>YCycle</th><th>ZCycle</th></tr>
<tr><td>Int#(0)</td><td>DBlkSize0#</td><td>DBlkSize1#</td><td>ExCache</td><td>BigEndian#</td></tr>
<tr><td>Int#(1)</td><td>IBlkSize0#</td><td>IBlkSize1#</td><td>Reserved</td><td>Tristate#</td></tr>
<tr><td>Int#(2)</td><td>Reserved</td><td>IStream</td><td>Reserved</td><td>NoCache#</td></tr>
<tr><td>Int#(3)</td><td>Reserved</td><td>StorePartial</td><td>MPS</td><td>BusDriveOn</td></tr>
<tr><td>Int#(4)</td><td>Phase DelayOn#</td><td>Phase DelayOn#</td><td>Phase DelayOn#</td><td>Phase DelayOn#</td></tr>
<tr><td>Int#(5)</td><td>R3K#</td><td>R3K#</td><td>R3K#</td><td>R3K#</td></tr>
</table></td></tr>
<tr><td>Clk2×Sys</td><td>P9</td><td>16</td><td>I</td><td>The master double frequency input clock used for generating SysOut#.</td></tr>
<tr><td>Clk2×Smp</td><td>Q10</td><td>27</td><td>I</td><td>A double frequency clock input used to determine the sample point for data coming into the processor and coprocessors.</td></tr>
<tr><td>Clk2×Rd</td><td>P10</td><td>9</td><td>I</td><td>A double frequency clock input used to determine the enable time of the cache RAMs.</td></tr>
<tr><td>Clk2×Phi</td><td>Q9</td><td>30</td><td>I</td><td>A double frequency clock input used to determine the position of the internal phases 1 and 2.</td></tr>
<tr><td>Reset#</td><td>A14</td><td>129</td><td>I</td><td>Synchronous initialization input used to force execution starting from the reset memory address. Reset# must be deasserted synchronously but asserted asynchronously. The deassertion of Reset# must be synchronized by the leading edge of SysOut#.</td></tr>
</table>

*) I = Input
O = Output

Pin Definitions and Functions (cont'd)

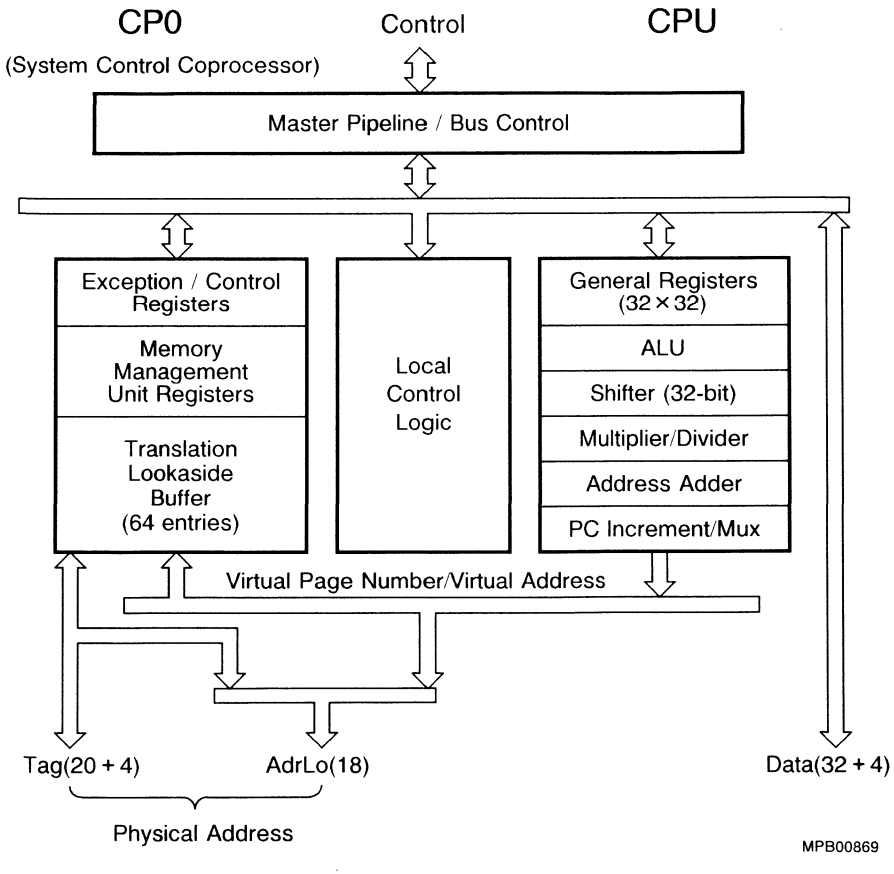
Symbol	Pin Number		I/O *)	Function
	144-Pin	172-Pin		
GND0	D3	–		Ground
GND13:1	C6,C7,C12, F13,G13,K13, M13,N10,N9,Q 6,N4,K3,G3,	69,66,65,64, 60,56,29,28, 24,23,22,18, 17		Ground
GND27:14	–	159,158,152,1 51,147,146,14 5,116,115,107, 104,103,71,70		Ground
$V_{CC}0$	F1	–		Power Supply (+ 5 V)
$V_{CC}14:1$	A1,C3,A5,C8,A 15,E15,H13,M 15,Q15, Q12,N8,N7, Q1,L1	68,67,63,62, 61,58,57,26, 25,21,20,19, 11,10		Power Supply (+ 5 V)
$V_{CC}27:15$	–	155,149,144,1 43,142,110,10 9,108,106,105, 102,101,100		Power Supply (+ 5 V)
Resvd2:0	–	153,114,72		Reserved

*) I = Input
O = Output

Functional Description

The SAB-R3000 consists of two integrated processors – a RISC CPU and a System Control Processor (CP0). Figure 4 is a block diagram of the SAB-R3000 which shows the functions incorporated within it.

Figure 4
Functional Block Diagram



Basic Architecture

On the right hand side of figure 4 is the CPU datapath that implements the 5-stage pipeline, which will be explained shortly. The datapath consists of a stack of functional units including 32 General Registers, ALU, 32-bit Shifter and an autonomous Multiply/Divide unit. An Address Adder and Increment/MUX for the PC generate instruction and data addresses alternatively at double the basic clock rate. This is necessary so that the SAB-R3000 can access both the Instruction and Data caches in a single CPU cycle, due to the multiplexed Data bus.

On the left of figure 4 is the System Control Processor (CP0). It's major element is a fully associative 64-entry Translation Lookaside Buffer (TLB), which translates a 20-bit virtual page number into a physical page frame number in a clock phase. As well as address translation the System Control Coprocessor also manages the exception handling and error recovery, the external cache interface, the memory control interface and the external coprocessor interface. It also incorporates on-chip tag comparators, parity generators and checkers.

Integer CPU

Pipeline Architecture

The execution of a single SAB-R3000 instruction consists of five primary steps or pipe stages.

(1) IF — Instruction Fetch:
Access the TLB and translate the instructions virtual address into its physical address to read an instruction from the I-cache. Note that the instruction is not actually read into the processor until the beginning (phase 1) of the RD pipe stage. Refer to figure 5.

(2) RD — Register Fetch / Instruction:
Read any required operands from the CPU registers while decoding the instruction.

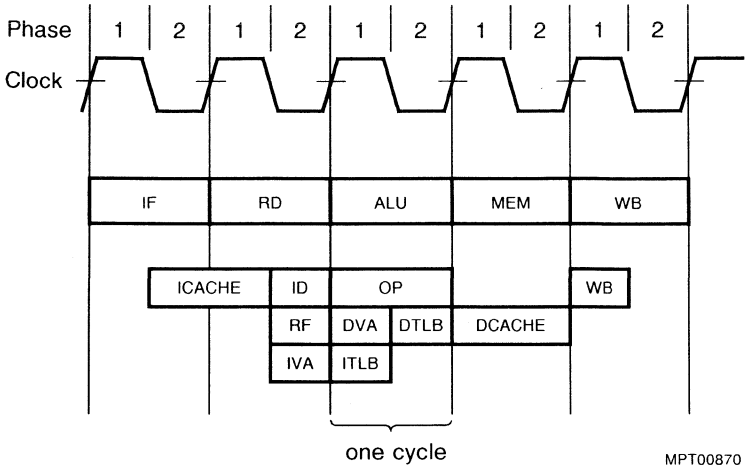
(3) ALU — ALU Operation:
Perform the required operation on instruction operands.

(4) MEM — Memory Access:
Access memory (D-cache) if required (for a Load or Store instruction).

(5) WB — Writeback:
Write back ALU results or value loaded from D-cache to the register file.

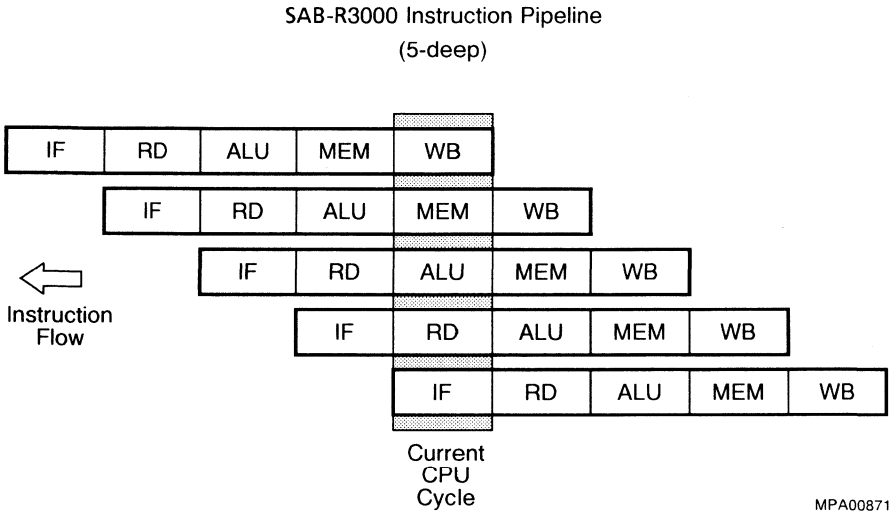
Each of these steps requires approximately one CPU cycle as shown in figure 5 (parts of some operations lap over into another cycle while other operations require only 1/2 cycle).

Figure 5
Execution Sequence of an Instruction



ICACHE	:	Instruction Cache Access
ID	:	Instruction Decode
RF	:	Register Operand Fetch
OP	:	Operation (ALU/Shift)
IVA	:	Instruction Virtual Address Calculation
ITLB	:	TLB Access for Instruction
DVA	:	Data Virtual Address Calculation
DTLB	:	TLB Access for Data
DCACHE	:	Data Cache Access
WB	:	Write Back to Register File

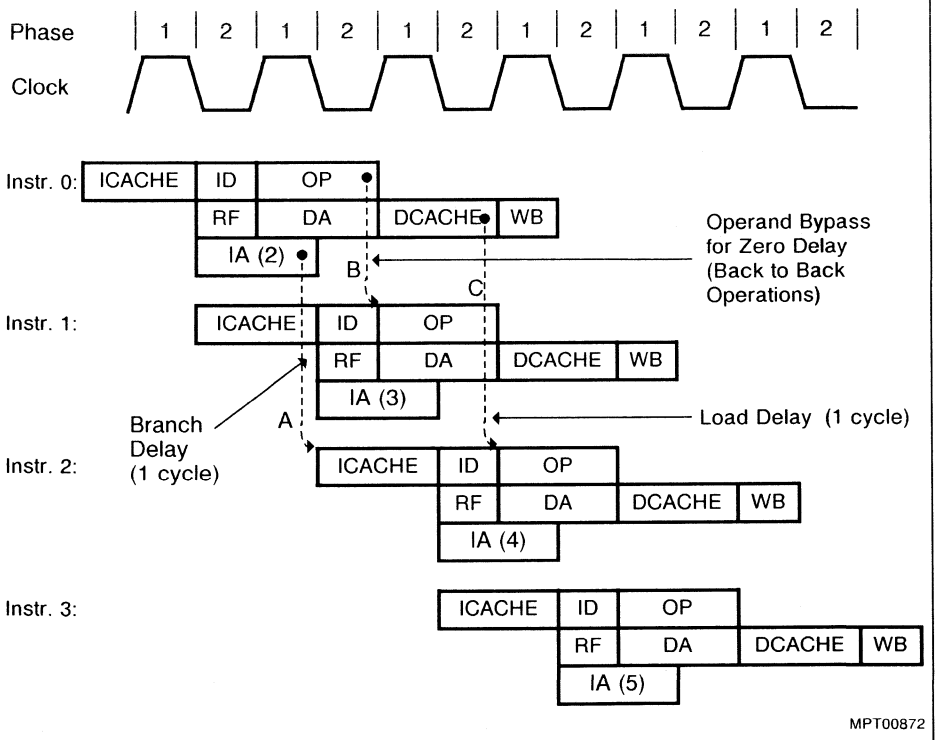
Figure 6
Overlapped execution of 5 instructions



The SAB-R3000 uses a 5-stage pipeline to achieve an instruction execution rate approaching one instruction per CPU cycle. Thus the execution of five instructions at a time are overlapped as shown in figure 6. This pipeline operates efficiently because different CPU resources (address and data bus accesses, ALU operations, register accesses and so on) are utilized on a non-interfering basis. As figure 7 illustrates there is not only parallelism due to pipelining but also parallelism within the execution of a single instruction.

The clock cycle is divided into two phases. To access the external Instruction and data caches (ICACHE, DCACHE) requires 1 cycle, as do major internal operations (OP, DA, IA). Instruction Decode (ID) is simple enough to occur within one phase, overlapped with Register Fetch (RF). Instruction address calculation and translation (IA) also overlaps Instruction Decode and Register Fetch, it consists of instruction virtual address calculation (IVA see figure 5) and TLB access for instruction address translation (ITLB). The Instruction address calculation and translation that is performed in the present instruction is for the second instruction following the present instruction, as shown in figure 7 (i.e. IA in Instr. 0 is for Instr. 2).

Figure 7
Instruction Pipeline Dependecies



IA (n) : Instruction Address Calculation and Translation for Instr."n"
DA : Data Address Calculation and Translation
(For the other abbreviations see figure 6)

IA is performed here especially so that a branch, for example, at Instr. 0 in figure 7 can address the ICACHE access of Instr. 2 (see dotted line A), i.e. one cycle latency. What happens is that after the condition has been evaluated (i.e. for a conditional branch), either PC + 2 or the Branch Target address is MUXed to the IF stage of the second succeeding instruction (again dotted line A). This means that if the branch is taken the branch target address is the address of Instr. 2. On the other hand, if the branch is not taken, PC + 2 (i.e. two sequential instructions after the current PC) is the address of Instr. 2. Data address calculation and translation (DA) is similar to IA.

Similarly, a load at Instr. 0 fetches data that are immediately used by the OP of Instr. 2 (dotted line C), while an ALU/Shift result gets passed directly into Instr. 1 with no delay (dotted line B). This tight coupling between instructions makes for a highly efficient pipeline. Note that the IA-ICACHE and DA-DCACHE cycles are displaced by one phase, so that the corresponding TLB and cache accesses can be interleaved on a single set of buses (see also figure 5).

As shown the SAB-R3000 uses a number of techniques internally to enable execution of all instructions in a single cycle, such as bypassing between instructions in the pipeline to keep the latency of branches and memory references to 1 cycle and to allow ALU results to be used in the succeeding instruction. However, there are two categories of instructions whose special requirements could disturb the smooth flow of instructions through the pipeline:

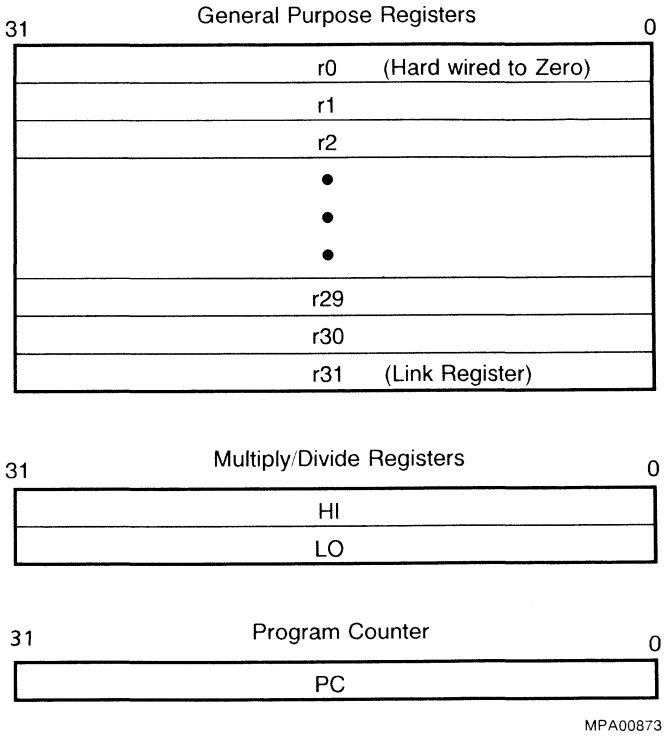
- **Load** instructions have a delay, or latency, of one cycle before the data being loaded is available to another instruction.
- **Jump and Branch** instructions also have a delay of one cycle while they fetch the instruction and target address if the branch is taken.

The SAB-R3000 continues execution despite the inherent 1-cycle delay. Loads, Jumps and Branches do not interrupt the normal flow of instructions through the pipeline, i.e. the pipeline is not stalled. The processor always executes the instruction immediately following one of these " delayed instructions ". Instead of having the processor manage these pipeline delays, the SAB-R3000 turns over the responsibility for dealing with " delayed instructions " to software. Thus the assembler must insert an appropriate instruction immediately following a " delayed instruction ". It also has the responsibility of ensuring that the inserted instruction has no dependencies relating to the " delayed instruction ". In the SAB-R3000 architecture only multi-cycle delays (e.g. waiting for SAB-R3010 results, MUL/DIV results, etc.) are handled by hardware interlocks – 1-cycle delays, as described, are handled much more efficiently by software for all implementations (thus guaranteeing scalability).

CPU Registers

There are 32 general purpose 32-bit registers, two 32-bit registers that hold the results of integer multiply and divide operations, and a 32-bit program counter as shown in figure 8.

Figure 8
CPU Registers



The 32 General Purpose Registers are treated symmetrically, with two exceptions – r0 is hardwired to a zero value, and r31 is the link register for Jump And Link instructions. Register r0 may be specified as a target register for any instruction when the result of the operation is discarded. The register maintains a value of zero under all conditions when used as a source register.

The two Multiply/Divide registers (HI, LO) store the double-word, 64-bit result, of multiply operations or the quotient and remainder of divide operations.

The Program Counter contains the current virtual address of the next instruction to be executed.

There is no condition code register. If an instruction generates a condition, the corresponding flags are stored in a general purpose register. This means that the pipeline is freed of any special mechanisms to by-pass condition codes, interlock on them, or abort writing them on exceptions. Instead the methods already implemented to deal with register-value dependencies are employed. Further, conditions mapped onto the register file are subject to the same compile-time optimizations in allocation and reuse as other register variables.

There is also no Program Status Word (PSW) register – the functions traditionally provided by a PSW are instead provided in the Status and Cause registers incorporated within the CP0. CP0 has a number of special purpose registers that are used in conjunction with the memory management system and during exception processing. These will be explained in the CP0 section

Data Formats

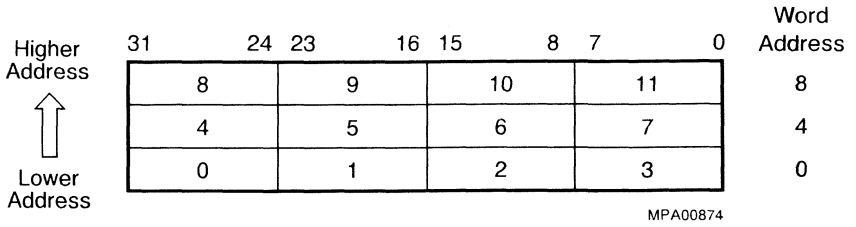
The SAB-R3000 defines signed/unsigned 32-bit words, 16-bit half-words and 8-bit bytes. The byte ordering is configurable either little-endian (iAPX x86®, NS32000®, DEC VAX®) or big-endian (MC680x0®, IBM 370®) byte ordering. Hence it is compatible with existing databases generated by machines that access bytes in either order.

Bit 0 is always the least significant (rightmost) bit, thus bit designations are always little-endian (although no instructions explicitly designate bit positions within words). Figures 9 and 10 show the ordering of bytes within words, and the ordering of words within multiple-word structures for the big-endian and little-endian conventions.

Special instructions are provided for addressing words that are not aligned on 4-byte (word) boundaries (Load/Store-Word-Left/Right; LWL, LWR, SWL, SWR). These instructions are used in pairs to provide addressing of misaligned words with one additional instruction cycle over that required for aligned words.

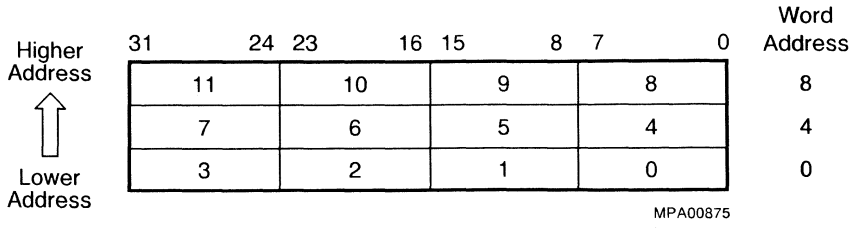
iAPX x86® is a trademark of INTEL
NS32000® is a trademark of National Semiconductor
DECVAX® is a trademark of Digital Equipment Cooperation
MC680x0® is a trademark of Motorola Inc.
IBM 370® is a trademark of IBM Cooperation

Figure 9
Addresses of Bytes within Words: Big Endian



- Most significant byte is at lowest address.
- Word is addressed by byte address of most significant byte.

Figure 10
Addresses of Bytes within Words: Little Endian



- Least significant byte is at lowest address.
- Word is addressed by byte address of least significant byte.

Addressing

The SAB-R3000 uses byte addressing, with alignment constraints for half-word and word accesses; half-word accesses must be aligned on an even byte boundary and word accesses must be aligned on a byte boundary divisible by four. Any attempt to address a data item that does not have the proper alignment causes an alignment exception. As said earlier special instructions are provided for addressing words that are not aligned.

The SAB-R3000 supports only one addressing mode - base register plus a signed 16-bit offset, which covers the most common case in High Level Languages and is very fast. The assembler, however, synthesizes some additional addressing modes to present more traditional addressing capabilities to the assembly language programmer.

Instruction Set Overview

All SAB-R3000 instructions consist of one 32-bit word. There are only three instruction formats (immediate, jump and register) as shown in figure 11.

The single instruction length simplifies instruction fetch and decode and eliminates the overhead for instructions crossing word and page boundaries within the memory hierarchy, thereby simplifying the interaction of instruction fetch with the virtual memory management unit. The three instruction formats ensure that opcodes and register descriptors are always found in the same bit locations. This enables register fetch to proceed in parallel with operation decode on all instructions, which is exactly what happens in the RD pipestage. More complicated (and less frequently used) operations can be synthesized by the compiler using sequences of simple instructions.

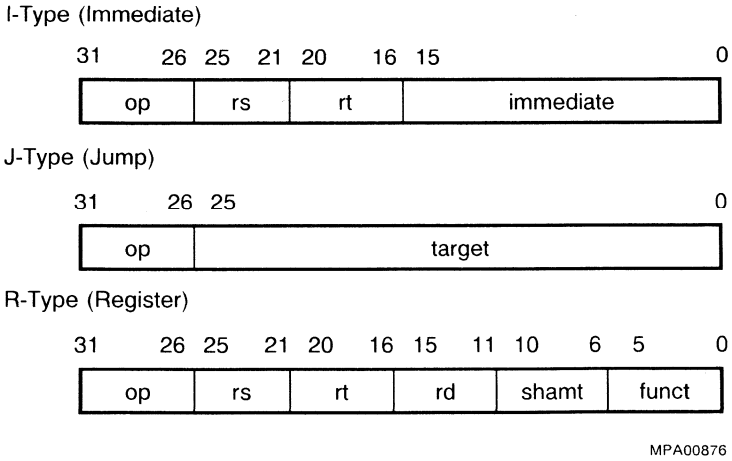
The SAB-R3000 instruction set can be divided into the following groups:

- **Load/Store** instructions move data between memory and general registers. They are all I-type instructions, since the only addressing mode supported is base register plus 16-bit, signed immediate offset. These are the only instructions which access memory.
- **Computational** instructions perform arithmetic, logical and shift operations on values in registers. They occur in both R-type (both operands and the result are registers) and I-type (one operand is a 16-bit immediate value) formats.
- **Jump and branch** instructions change the control flow of a program. Jumps are always to a paged absolute address formed by combining a 26-bit target with four bits of the Program Counter (J-type format, for subroutine calls) or 32-bit register addresses (R-type, for returns and dispatches). Branches have 16-bit offsets relative to the Program Counter (I-type). Jump and Link Instructions save a return address in Register 31.
- **Coprocessor** instructions perform operations in the coprocessors. Coprocessor Loads and Stores are I-type. Coprocessor computational instructions have coprocessor-dependent formats (see SAB-R3010 datasheet).

- **Coprocessor 0** instructions perform operations on the System Control Coprocessor (CP0) registers to manipulate the memory management and exception handling facilities of the processor.
- **Special** instructions perform a variety of tasks, including movement of data between special and general registers, system calls and breakpoint. They are always R-type.

A detailed summary of the instruction set is provided in the Instruction Set Summary section.

Figure 11
Instruction Formats



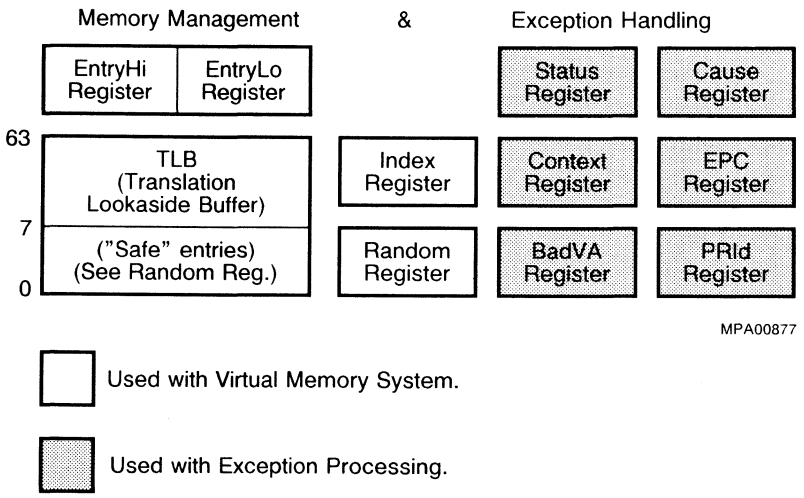
where:

op	:	is a 6-bit operation code
rs	:	is a 5-bit source register specifier
rt	:	is a 5-bit target (source/destination) register or branch condition
immediate	:	is a 16-bit immediate, branch displacement or address displacement
target	:	is a 26-bit jump target address
rd	:	is a 5-bit destination register specifier
shamt	:	is a 5-bit shift amount
funct	:	is a 6-bit function field

System Control Coprocessor

The SAB-R3000 CPU supports up to 4 coprocessors (designated CP0 through CP3), which are tightly coupled co-execution units that share a single instruction stream with the CPU. The System Control Coprocessor (CP0) is incorporated on the SAB-R3000 chip and supports the virtual memory system and exception handling functions (traps, interrupts, memory and internal operation faults) of the SAB-R3000. CP0 incorporates a 64-entry TLB plus the registers shown in figure 12.

Figure 12
The CP0 Registers



The virtual memory system is implemented using a TLB and a group of programmable registers as shown in figure 12. The other registers shown are used to perform the exception handling capabilities. Table 1 provides a brief description of each register. In this table the number of each register is given. Logically the registers are numbered from 0 to 31. The numbers not contained in the table are unused.

Table 1
System Control Coprocessor (CP0) Registers

Register	Description	Number
EntryHi	High half of a TLB entry	10
EntryLo	Low half of a TLB entry	2
Index	Programmable pointer into TLB array	0
Random	Pseudo-random pointer into TLB array	1
Status	Mode, Interrupt enables, and diagnostic status info	12
Cause	Indicates nature of last exception	13
EPC	Exception Program Counter	14
Context	Pointer into kernel's virtual Page Table Entry array	4
BadVA	Most recent bad virtual address	8
PRId	Processor revision identification	15

Access to these registers and CP0 specific instructions (e.g. MTC0) may be restricted by setting CP0 to an "unusable" state (see status register). When the processor is executing in kernel mode the usability of CP0 is ignored, i.e. it is always considered usable.This means that requests to manipulate CP0 from kernel mode are always granted. However, it is possible for the Kernel to grant unrestricted access to CP0 registers by setting it to a usable state. Kernel and User modes are described in the next section.

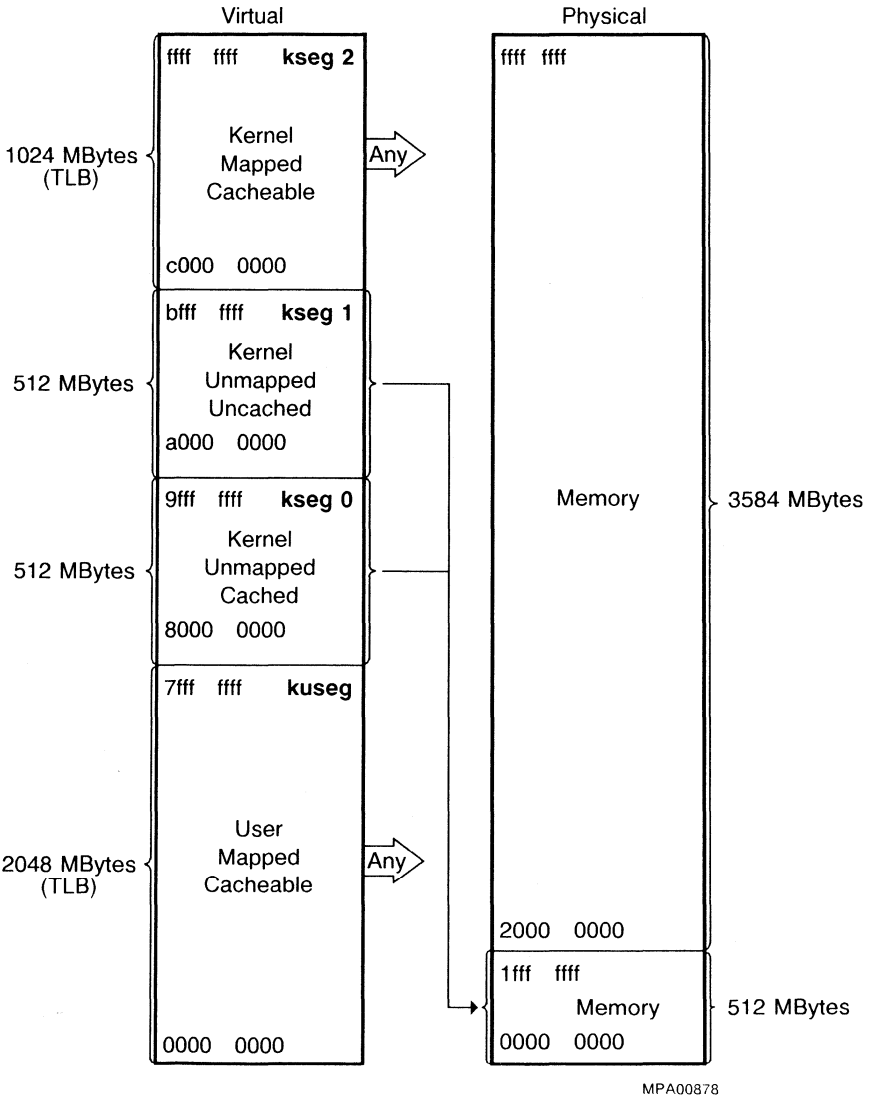
Memory Management System

The SAB-R3000 has an addressing range of 4 Gbytes. Since most systems implement physical memory sizes under 4 Gbytes, the SAB-R3000's virtual memory system logically expands the available physical memory space by translating addresses composed in a large virtual address space into the physical memory space. All mapping is performed by the TLB in CP0. The TLB is fully associative and maps 64 4-Kbyte pages, sharable among user processes with minimal context-switch overhead, due to the PID (Process Identifier) number associated with each TLB entry.

Operating Modes

There are two operating modes in the SAB-R3000, the Kernel mode and the User mode. The processor normally operates in User mode until an exception is detected forcing it into Kernel mode. It remains there until a Restore From Exception (rfe) instruction is executed.

Figure 13
Virtual Memory Map



Address mapping is different for Kernel and User modes. The User mode address space is a subset of the Kernel mode address space – i.e. the current User process owns a linear 2 Gbyte address space that is included in a 4 Gbyte Kernel address space. Figure 13 shows the virtual-to-physical memory map for both User mode and Kernel mode segments.

User Mode Virtual Addressing

When the processor is operating in User mode, a single, uniform virtual address space (kuseg) of 2 Gbytes is available to the current user process. All valid User mode virtual addresses have the most significant bit cleared to 0 – i.e. address references above 0x7FFFFFFF (0x means hex in a "C" type notation) cause an Address Error exception. All references to kuseg are mapped through the TLB, which also controls the cacheability of an access (the N-bit in a TLB entry). In Kernel mode, references to kuseg are treated just like User mode references, streamlining Kernel access to User data. Kuseg is typically used to hold all User code and data, and the current User process typically resides here, plus shared libraries in systems that have them.

Kernel Mode Virtual Addressing

In Kernel mode, three distinct virtual address spaces (in addition to kuseg) are available.

Kernel Cached, Unmapped – kseg0:

This segment is 512 Mbytes long starting at 0x80000000. References within kseg0 are direct-mapped onto the first 512 Mbytes of physical address space, use cache memory, but do not use TLB entries. That is to say that the physical address selected is defined by subtracting 0x80000000 from the virtual address (in order to directly map it into the first 512 Mbytes of physical memory). Typically some Kernel data and some of its executable code are kept here.

Kernel Uncached, Unmapped – kseg1:

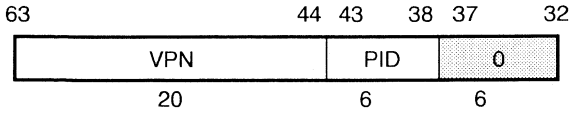
Kernel segment kseg1 begins at 0xa0000000 and is also 512 Mbytes long. Like kseg0 it is direct-mapped onto the first 512 Mbytes of physical address space using no TLB entries. That is to say that the physical address selected is defined by subtracting 0xa0000000 from the virtual address. Unlike kseg0, it uses uncached references. An operating system typically uses kseg1 for I/O registers, ROM code and disk buffers.

Kernel Mapped – kseg2:

This segment is 1 Gbyte long, beginning at 0xc0000000. Like kuseg, it uses TLB entries to map virtual addresses to arbitrary physical ones, with or without caching (N-bit in a TLB entry). Operating systems typically use kseg2 for Kernel stacks and pre-process data that it must remap on context switches, for User page tables (memory maps), and some dynamically allocated data areas.

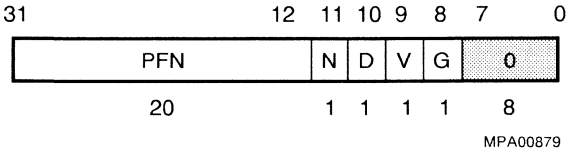
Figure 14
TLB EntryLo & EntryHi Registers

TLB EntryHi Register



VPN : Virtual Page Number. Bits 31. .12 of virtual address.
PID : Process ID field. A 6-bit field which lets multiple processes share the TLB while each process has a distinct mapping of otherwise identical virtual page numbers.
0 : Reserved. Currently ignores writes, returns zero when read.

TLB EntryLo Register



PFN : Page Frame Number. Bits 31. .12 of the physical address. The SAB-R3000 maps a virtual page to the PFN.
N : Non-cacheable. If this bit is set, the page is marked as non-cacheable and the SAB-R3000 directly accesses main memory instead of first accessing the cache.
D : Dirty. If this is set, the page is marked as "dirty" and therefore writable. This bit is actually a "write-protect" bit that software can use to prevent alteration of data. If an entry is accessed for a write operation when the D bit is cleared, the SAB-R3000 causes a TLB Mod trap. The TLB entry is not modified on such a trap.
V : Valid. If this bit is set, it indicates that the TLB entry is valid; otherwise, a TLBL or TLBS Miss occurs.
G : Global. If this bit it set, the SAB-R3000 ignores the PID match requirement for valid translation. In kseg2, the Global bit lets the kernel access all mapped data without requiring it to save or restore PID (Process ID) values.
0 : Reserved. Currently ignores writes, returns zero when read.

Virtual Memory Control

A high level description of the address space has been given - now the low-level details of the virtual memory system shall be discussed. The following registers which will be described are used for virtual memory control.

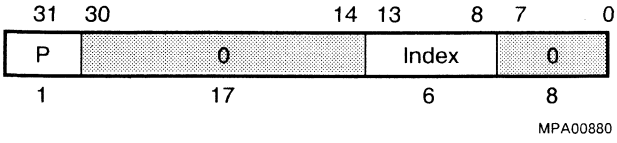
EntryHi and EntryLo:

This register pair and a single TLB entry have the same format, and so are described together. These two registers provide the data pathway (i.e are a buffer) through which the TLB is read, written or probed. When address translation exceptions occur, these registers are loaded with the relevant information about the address that caused the exception. EntryLo is the natural form of a Page Table Entry (PTE), however, since PTE's are always loaded by system software, not by the SAB-R3000 hardware, an operating system can use another format for memory resident PTE's. The register pair is illustrated in figure 14, a TLB entry is equivalent to the concatenation of these two registers.

Index Register:

The Index register is a 32-bit, read/write register, which has a 6-bit Index field. This field runs from 0 to 63, indexes an entry in the TLB and is used as a subscript to read or write any TLB entry. The high-order bit shows the success or failure of a TLB Probe (tlbp) Instruction. Figure 15 shows the format of the Index register.

Figure 15
Index Register



P : Probe failure. Set to 1 when the last TLBProbe (tlbp) instruction was unsuccessful.

Index : Index to the TLB entry that will be affected by the TLBRead and TLBWrite instructions.

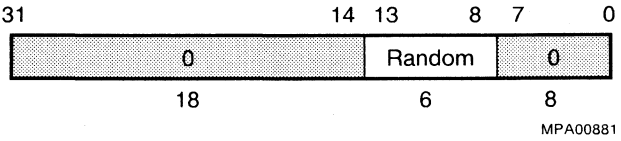
0 : Reserved. Currently ignores writes, returns zero when read.

Random Register:

The Random register is a 32-bit register, which has a 6-bit Random field which indexes a random entry in the TLB. The Random field is decremented every machine cycle, but is constrained to the range from 8 to 63. It is used to speed up pseudo-random TLB refill under Operating System control. The TLB Write Random (tlbwr) instruction is used to write the TLB entry that this register indexes. The first 8 entries (0 to 7) in the TLB are "safe" entries (see figure 12) because a tlbwr instruction can never replace the contents of these entries. Typically these 8 entries are reserved by the Operating System.

The contents of this register can be read to verify proper operation of the process (not normally required). To further simplify testing, the Random field is set to a value of 63 when the SAB-R3000 is reset. Figure 16 illustrates the Random register.

Figure 16
Random Register



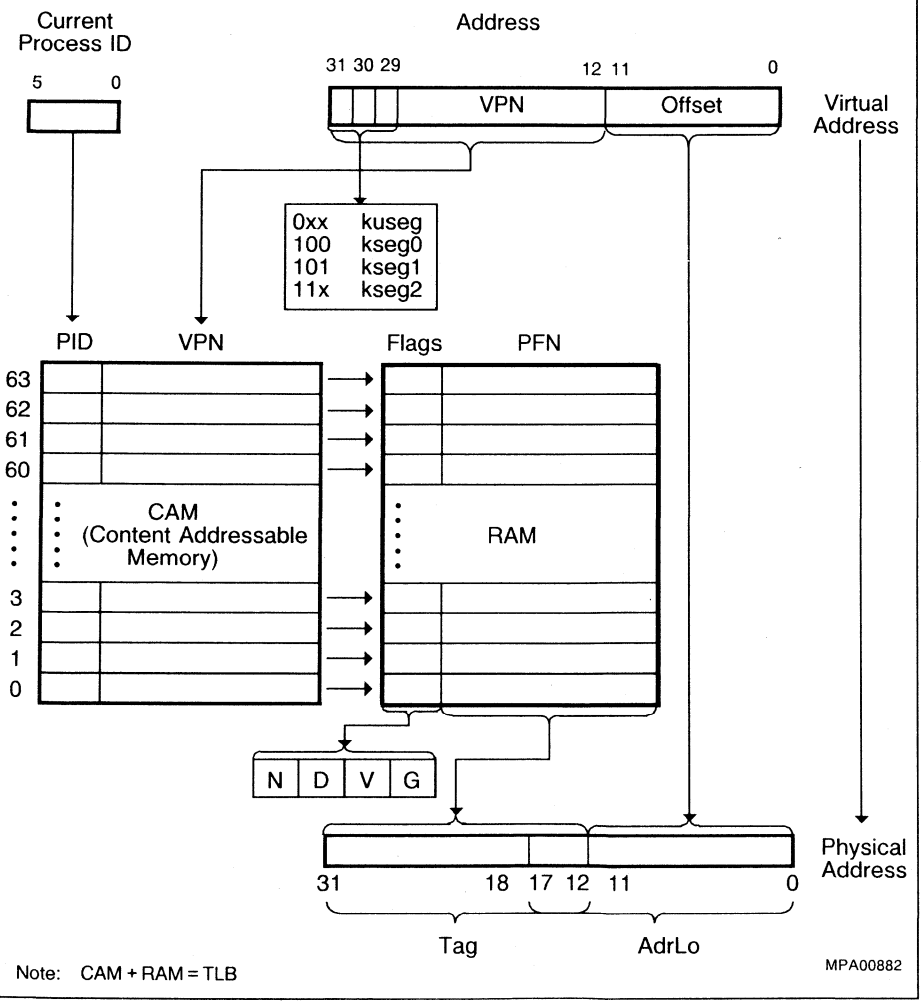
Random: A random index (with a value ranging from 8 to 63) to a TLB entry.

0 : Reserved. Currently ignores writes, returns zero when read.

Virtual Address Translation

During virtual-to-physical address translation, the processor compares the PID number and the highest 20 bits (the VPN) of the virtual address to the contents of the TLB. Figure 17 illustrates the TLB address translation process.

Figure 17
TLB Operation



A virtual address matches a TLB entry when:

- the Virtual Page Number (VPN) field of the virtual page address matches the VPN field of the TLB entry and,
- either the Global (G) bit of the TLB entry is set, or the PID field of the virtual address (as held in the EntryHi register) matches the PID field of the TLB entry.

While the Valid (V) bit of the TLB entry must be set for a valid translation to take place, it is not involved in the determination of a matching TLB entry.

If a TLB entry matches, the physical address and access control bits (N, D and V) are retrieved from the matching TLB entry. Otherwise a TLB miss (reference to kseg2), or a UTLB miss (reference to kuseg) exception occurs. If the V bit is not set, a TLB miss exception is taken. Finally, if the access is a Write and the Dirty (D) bit is not set, a TLB modification exception occurs. If the Noncacheable (N) bit is set, the physical address that is retrieved is used to access main memory, bypassing the cache.

Exception Handling System

The term exception is used for any infrequent or exceptional event that causes the processor to make a temporary transfer of control from its current process to another process that services the event. There are two main classes of exceptions in the SAB-R3000:

- machine exceptions, such as program traps, overflow and address translation exceptions.
- external asynchronous exceptions, which include six maskable external hardware interrupts, bus error and reset.

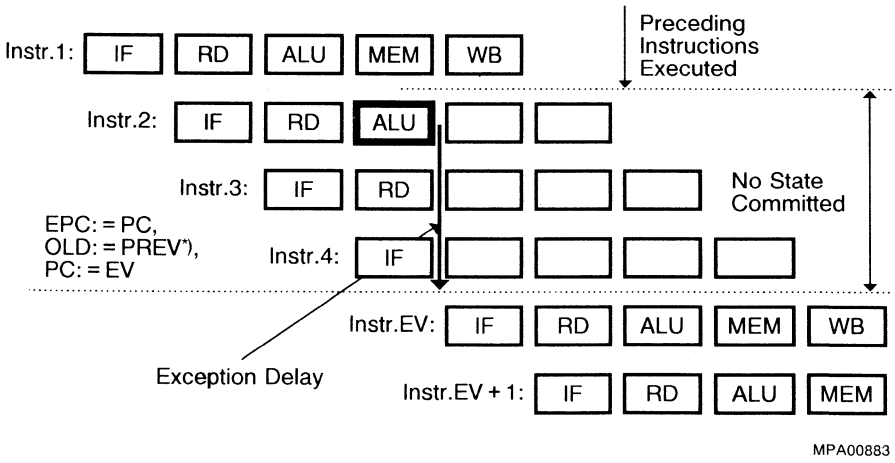
There are eight interrupts available, the hardware generates six and software two.

The exception handling system is responsible for efficiently handling relatively infrequent events, such as TLB misses, arithmetic overflow, I/O interrupts and system calls. On detection of an Exception the SAB-R3000 suspends the normal sequence of instruction execution; the processor exits User mode and is forced into Kernel mode where it can respond to the exceptional event. These events interrupt the normal execution flow by aborting the instruction which causes the exception and all those following in the pipeline which have already begun execution. The Exception Program Counter (EPC) is loaded with the appropriate restart location where execution should resume after the exception has been serviced. The restart location in the EPC is the address of the instruction which caused the exception or, if the instruction was executing in a branch delay slot, the address of the branch or jump instruction immediately preceding the delay slot. The SAB-R3000 then performs a direct jump into a designated handler routine.

A minimal amount of state is saved in the CP0 registers in order to facilitate the analysis of the cause of the exception, the servicing of the event which caused it, and the resumption of the original flow of execution.

The pipelined nature of the SAB-R3000 complicates the exception handling system. Exceptions which occur late in the pipeline effectively necessitate aborting instructions which have already begun execution but which logically should execute after the exception handling routine. Even taking this into account all SAB-R3000 exceptions are precise. That is, each exception is handled in a way that reflects serial completion of all instructions prior to the exception; the instruction which causes it and all those that follow are aborted and can be re-executed after servicing the exception. What this means (referring to figure 18) is that, for example, when an exception occurs in the ALU stage of Instr. 2, that Instr. 3 and 4, which have already started execution in the pipeline, do not alter the state of the machine so that execution may always properly resume after servicing the exception. Precise exception handling is shown in figure 18.

Figure 18
Precise Exception Handling

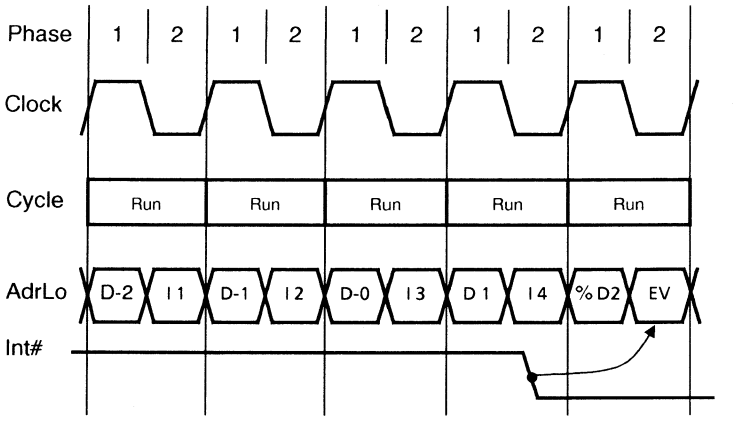


*) Refer to KU and IE bits of Status Register
EV: Exception Vector
EPC: Exception PC

Even though the machine is relatively deeply pipelined, exceptions are reported synchronously, so that all exceptions for a particular instruction are reported prior to exceptions for its succeeding instructions. Said another way, all exceptions are reported as if the processor was not pipelined.

There is only a one-cycle delay from when an exception occurs to when the exception handling routine is started (i.e. reaction time). This can be seen in figure 18, i.e. an exception occurs in the ALU cycle of Instr.2 (e.g. Overflow) and one cycle later the first instruction of the servicing routine is started (i.e. Instr.EV). This can be seen on the AdrLo bus for an interrupt exception as is shown in figure 19. In the cycle after Int# (indicating an external interrupt exception) has been asserted the Exception Vector (EV) address is on the address bus.

Figure 19
Interrupt Exception Latency

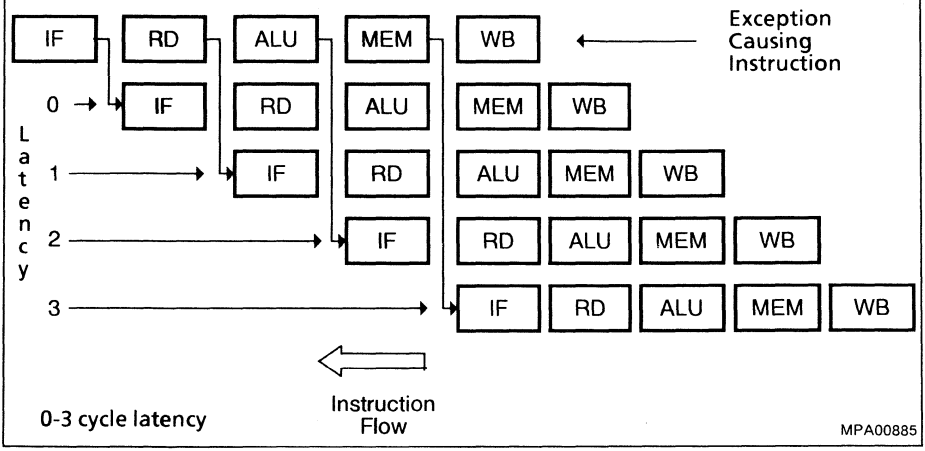


Where:

Dn : is the Data Transaction for Instr. n
In : is the Instruction Fetch for Instr. n
% : means an incorrect datum.
EV : Exception Vector

There is also an associated exception latency. This depends on how late the exception occurs in the pipeline. The exception latency can be 0-3 cycles in duration. It is the number of instructions which were aborted, that had already begun execution in the pipeline but which logically should execute after the exception handling routine – as discussed earlier. The concept of Exception latency is illustrated in figure 20.

Figure 20
Exception Latency



Exception Types

Table 2 lists each of the exception types which are handled by the processor, giving a short description of each.

There are only 3 exception vectors provided, one for Reset and one for UTLB miss exceptions (0xbfc00000 and 0x80000000 respectively), each of the remaining exceptions causes execution to resume at the General exception vector (0x80000080). When the BEV bit of the Status Register is set, the UTLB miss exception vector address is changed to 0xbfc00100 and the General exception vector address is changed to 0xbfc00180.

Table 2
SAB-R3000 Exceptions

Maskable	Exception	Mnemonic	Cause	Exception Vector
NO	Reset	Reset	Assertion of SAB-R3000's Reset signal causes an exception that transfers control to the special vector at virtual address 0xbfc00000.	0xbfc00000
	UTLB miss	UTLB	User TLB miss. A reference is made (in either User mode or Kernel mode) to a page in kuseg that has no matching TLB entry.	0x80000000 or 0xbfc00100
	TLB miss	TLBL (load) TLBS (store)	A referenced TLB entry's Valid bit isn't set or there is a reference to a *kseg2* page that has no matching TLB entry.	0x80000080 or 0xbfc00180
	TLB modified	Mod	During a store instruction, the Valid bit is set but the Dirty bit is not set.	
	Bus Error	IBE (Instruction) DBE (data)	Assertion of the SAB-R3000's BERR# signal due to such external events as bus timeout, backplane bus parity errors, invalid physical addresses or invalid access types.	
	Address Error	AdEL (load) AdES (store)	Attempt to load, fetch, or store an unaligned word; that is, a word or halfword at an address not evenly divisible by 4 or 2, respectively. Also caused by reference to a virtual address with most significant bit set while in User mode	
	Overflow	Ovf	Two's complement overflow during add or subtract.	
	System call	Sys	Execution of the syscall instruction.	
	Breakpoint	Bp	Execution of the break instruction.	
	Reserved Instruction	RI	Execution of an instruction with an undefined or reserved major operation code (bits 31..26), or a special instruction whose minor opcode (bits 5..0) is undefined.	
	Coprocessor Unusable	CpU	Execution of a coprocessor instruction when the CU (Coprocessor Usable) bit is not set for the target coprocessor.	
YES	Interrupt	Int	Assertion of one of the SAB-R3000's six hardware interrupt inputs or setting of one of the two software interrupt bits in the Cause register.	

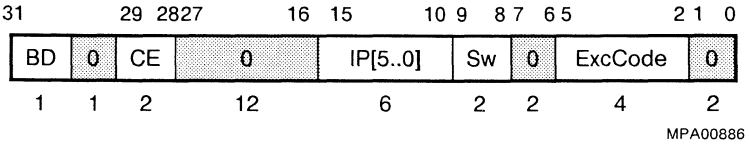
The Exception Handling Registers

The CP0 registers shown on the right of figure 12 contain information that is related to exception processing. Software can examine these registers during exception processing to determine such things as the cause of an exception, and the state of the CPU at the time of an exception. Two other registers, the Index register and the Random register, described in the Memory Management System section, may also contain useful information when handling exceptions related to virtual memory errors.

Cause Register:

The contents of this 32-bit register describe the last exception as shown in figure 21. The ExcCode field indicates the reason for the exception as listed in table 3. The remaining fields indicate pending external interrupts (IP), pending software interrupts (Sw), which, if any, coprocessor was found unusable (CE), and the occurrence of an exception in a branch delay slot (BD). All bits in the register, excluding the Sw bits, are read only. The Sw bits can be written into, to set or reset software interrupts.

Figure 21
Cause Register



BD : Branch Delay. Set to 1 if last exception was taken while executing in a branch delay slot.
CE : Coprocessor Error. Indicates the unit number referenced when a Coprocessor Unusable Exception is taken.
IP : Interrupts Pending. Indicates the external interrupts that are pending. IP[5..0] = Interrupt [5..0]
Sw : Software Interrupts. Indicates which of the two software interupts are pending. This field may be written into to set or reset software interrupts.
ExcCode : Exception Code field. Described in table 3.
0 : Reserved. Currently ignores writes, returns zero when read.

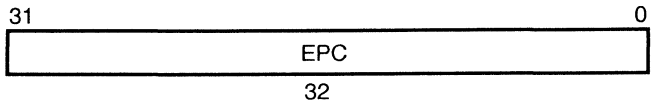
Table 3
Cause Register ExcCode Field

Number	Mnemonic	Description
0	Int	External interrupt
1	MOD	TLB modification exception
2	TLBL	TLB miss exception (Load or instruction fetch)
3	TLBS	TLB miss exception (Store)
4	AdEL	Address error exception (Load or instruction fetch)
5	AdES	Address error exception (Store)
6	IBE	Bus error exception (for an instruction fetch)
7	DBE	Bus error exception (for a data load or store)
8	Sys	Syscall exception
9	Bp	Breakpoint exception
10	RI	Reserved Instruction exception
11	CpU	Coprocessor Unusable exception
12	Ovf	Arithmetic overflow exception
13-15	–	reserved

Exception Program Counter Register (EPC):

On exception, this register records the address to where processing should be resumed after an exception has been serviced. The EPC register contains the virtual address of the instruction which was the cause of the exception; when that instruction is in a branch delay slot, the EPC contains the virtual address of the immediately preceding branch or jump instruction. The EPC register format is shown in figure 22.

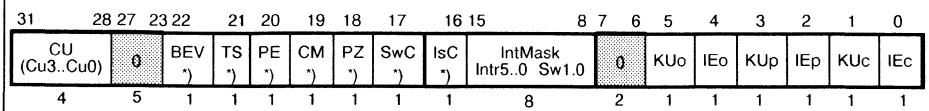
Figure 22
EPC Register



Status Register:

The Status register is a read/write register that contains the Kernel/User mode, interrupt enable and diagnostic state of the processor, i.e. it contains all major machine status bits. All bits in the Status register, excluding TS which is read only, are readable and writable. Figure 23 shows the format of the Status Register.

Figure 23
Status Register



MPA00888

*) Indicates primary use is for diagnostics and testing.

CU : Coprocessor Usability. These bits control usability of the four possible coprocessors: Cu3, Cu2, Cu1, and Cu0. If a CU bit is set (= 1), that coprocessor is usable.

*) BEV : Bootstrap Exception Vector. If set to 1, causes the SAB-R3000 to use the alternate, bootstrap vectors for UTLB Miss and general exceptions.

*) TS : TLB Shutdown. Set to 1 if SAB-R3000 has disabled TLB due to catastrophic error. Cleared only by Reset.

*) PE : Parity Error. Set to 1 if cache parity error occurs. Reset by writing a 1 to this bit.

*) CM : Cache Miss. Set to 1 if most recent D-Cache load resulted in a miss (only when the D-Cache is isolated).

*) PZ : Parity Zero. When set to 1, causes zero to replace normal outgoing parity bits.

*) SwC : Swap Caches. Controls switching of control signals for I-Cache and D-Cache.

*) IsC : Isolate Cache. When set to 1, isolates D-Cache from main memory system.

IntMask : Interrupt Mask. When a bit is set to 1, the corresponding hardware interrupt [Intr# 5..0] or software interrupt [Sw1..0] is enabled.

KUo : Kernel/User mode, old. Set to 0 if Kernel, 1 if User.

IEo : Interrupt Enable, old. Set to 1 to enable, 0 to disable.

KUp : Kernel/User mode, previous. Set to 0 if Kernel, 1 if User.

IEp : Interrupt Enable, previous. Set to 1 to enable, 0 to disable.

KUc : Kernel/User mode, current. Set to 0 if Kernel, 1 if User.

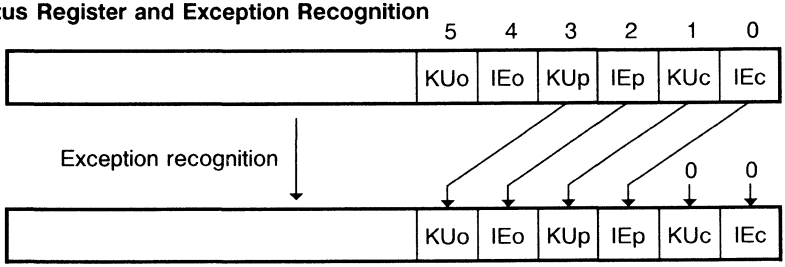
IEc : Interrupt Enable, current. Set to 1 to enable, 0 to disable.

0 : Reserved. Currently ignores writes, returns zero when read.

The Status register contains a three level stack (current, previous, and old) of the Kernel/User mode bit (KU) and the interrupt enable (IE) bit.

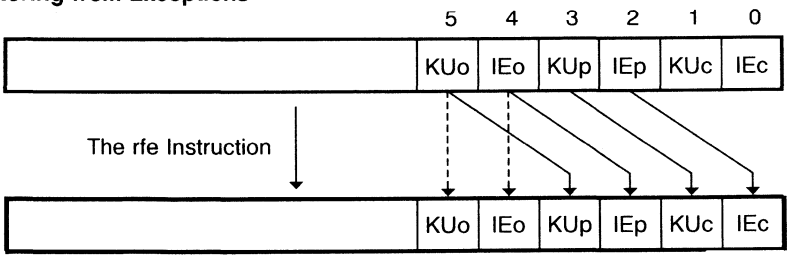
When an exception is taken the stack is pushed, i.e. the current Kernel/User mode (KUc) and current interrupt enable mode (IEc) bits are saved into the previous mode bits. The previous mode bits (KUp and IEp) are saved into the old mode bits (KUo and IEo). The current mode bits are cleared to cause the processor to enter Kernel mode and turn off interrupts. This three level set of mode bits allows the SAB-R3000 to respond to two levels of exceptions before software must save the contents of the Status register. Figure 24 shows how the mode bits are pushed when an exception is taken.

Figure 24
Status Register and Exception Recognition



When an exception handler has completed execution the processor must return to the system context that existed prior to the exception. This is achieved by the Restore From Exception (rfe) instruction. The rfe instruction, when executed, pops the three level stack, i.e. the previous mode bits (KUp and IEp) are restored into the current mode bits (KUc and IEc). Likewise, the old mode bits (KUo and IEo) are restored into the previous mode bits. The old mode bits themselves remain unchanged. The actions of the rfe instruction are illustrated in figure 25.

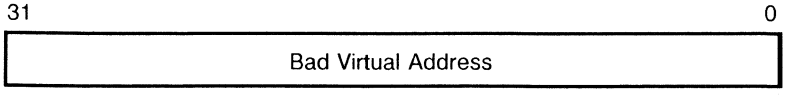
Figure 25
Restoring from Exceptions



Bad Virtual Address Register:

The BadVAddr register saves the entire bad virtual address for any addressing exception; AdeL or AdeS. Figure 26 shows this register format.

Figure 26
BadVAddr Register



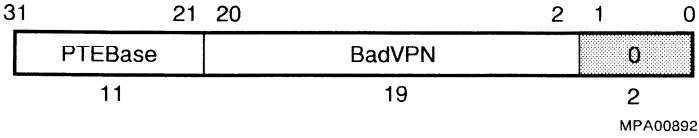
Context Register:

The Context register contains a pointer to the current user process's page map, located in kseg2 (kernel-mapped). It is designed for use in the UTLB miss handler, which loads TLB entries for normal user mode references.

The BadVPN field is not writable, it holds the VPN from the most recent virtual address for which the translation was invalid (i.e. an address exception). The 19-bit BadVPN field contains bits 30...12 (user segment Virtual Page Number) of the BadVAddr register. Bit 31 is excluded, because the UTLB miss handler is only invoked on user segment references whose highest virtual address is 0x7FFFFFFF.

The PTEBase field is writable as well as readable and indicates the base address of the page map of the current user address space. This register is implemented for the convenience of the Operating System. Figure 27 shows the format of the Context register.

Figure 27
Context Register



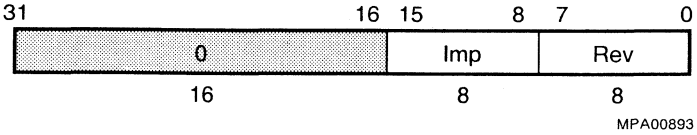
PTEBase : Holds the base for the Page Table Entry (set by software).
BadVPN : Holds the failing Virtual Page Number (set by hardware). This field is read-only and contains bits 30..12 (user-segment VPN) of the BadVAddr register.
0 : Unused; ignored on writes, zero when read.

Processor Revision Identifier Register:

This 32-bit read only register contains information that identifies the implementation and revision level of the Processor and System Control Coprocessor. The format of the register is shown in figure 28.

Figure 28
Processor Revision Identifier Register

PRId Register



Imp : Implementation identifier.
Rev : Revision identifier.
0 : Reserved. Currently ignores writes, returns zero when read.

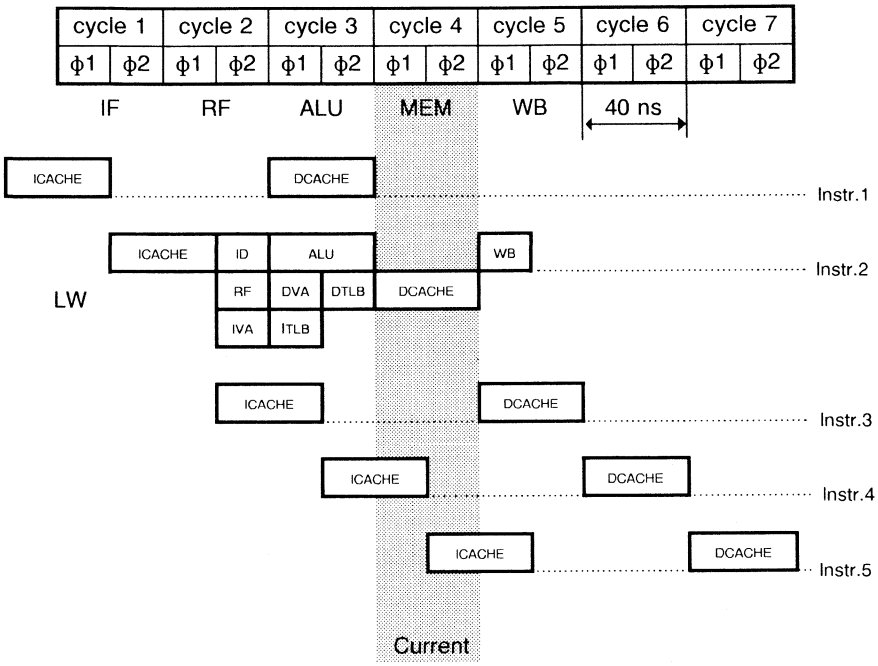
Memory System Hierarchy

The high performance capabilities of the SAB-R3000 processor places stringent demands on the memory system configuration. In order to achieve the goal of an instruction execution rate of one instruction per CPU cycle, the SAB-R3000 demands a memory bandwidth of 200 MBytes/second at 25 MHz from the memory system configuration. The memory system requirements can be seen in figure 29.

This high memory bandwidth is realized by a high performance memory hierarchy which centers on the use of external caches. Separate data and instruction caches are implemented, and the processor alternates accesses of the two caches during each CPU cycle – thus 2 words/CPU cycle, as shown in figure 29, are accessed. Both caches are physical as opposed to virtual, may vary in size from 4 to 256 Kbytes each depending on the performance required, and are implemented using standard SRAM devices.

The update policy employed is a write-through policy, which simplifies the data consistency problem between cache and main memory. All data that is written to the data cache is also written out to main memory. Write buffers capture this data from the SAB-R3000 at CPU clock rates and then update main memory at its slower clock rate – therefore not stalling the processor. A simplified diagram of the high performance memory system is shown in figure 30.

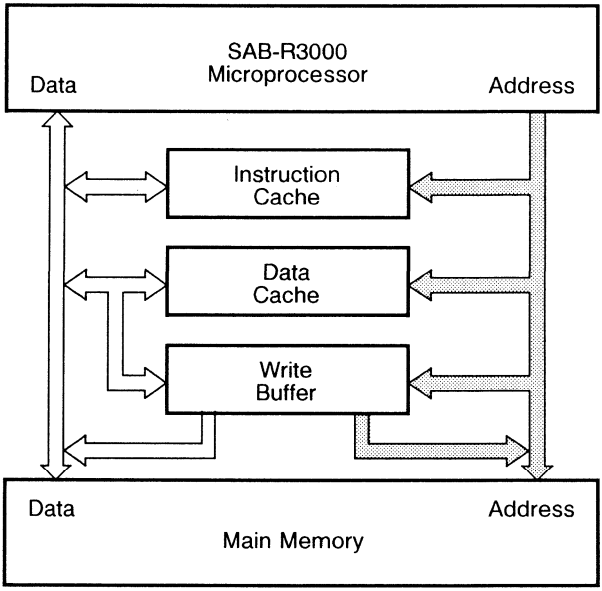
Figure 29
Memory Bandwidth



2 Words/Cycle
200 Mbytes/s at 25 MHz

MPA00894

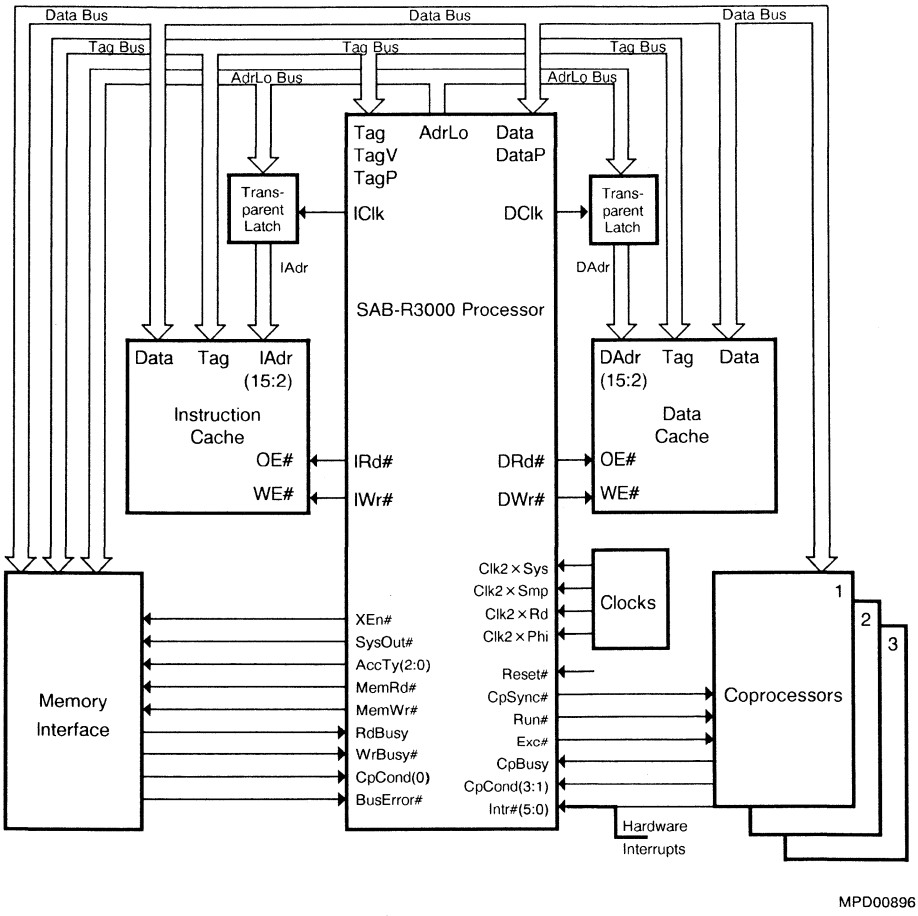
Figure 30
An SAB-R3000 System with a High-Performance Memory System



Processor Interface

The SAB-R3000 supports interfaces to external caches, main memory and coprocessors. Figure 31 illustrates the external interfaces of the SAB-R3000 processor.

Figure 31
Processor Interface

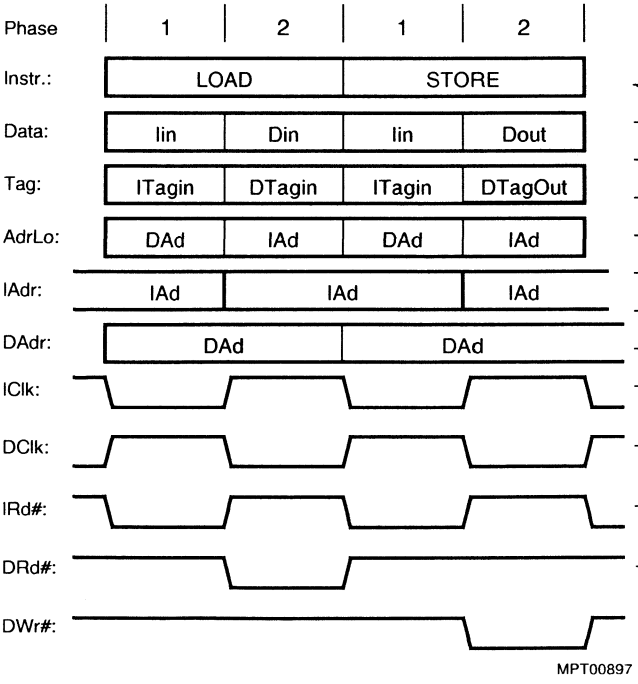


External Cache Interface

As described earlier the SAB-R3000 supports separate caches for instructions and data. As was seen in figure 4 the physical address coming out of the TLB is split across the external buses; AdrLo (18 bits) and TAG (20 bits plus valid and 3 parity bits). The caches are addressed by the 18 bit address bus AdrLo. Since AdrLo presents byte addresses and the caches are organised as words, its least significant 2 bits are not used by the caches. The most significant six bits of AdrLo bus are identical to the least significant six bits of the TAG bus but are output with AdrLo timing. This overlap allows cache size to vary with implementation (i.e. from 4 to 256 Kbytes).

The processor interleaves accesses to the two caches on the AdrLo, TAG and DATA buses. Instruction fetch begins with AdrLo (IAd) clocked through a transparent latch by IClk during phase 2 of a machine cycle, and continues until DATA (Iin) and TAG (ITagin) are latched on the chip at the end of the next phase 1. This is shown in figure 32. In the diagram IAdr and DAdr buses are latched versions of AdrLo. – Refer also to figure 31.

Figure 32
Cache Interface Operation



Iin	Instruction in
Din	Data in
Dout	Data out
I Tagin	Instruction Tag in
D Tagin	Data Tag in
D Tagout	Data Tag out
DAd	Data address
IAd	Instruction address
I/DClk	Transparent latch enable
I/DRd#	Instruction/Data read enable
DWr#	Data write enable

Similarly, data fetch begins with AdrLo (DAd) clocked by DClk during phase 1, and completes with Data (Din) and Tag (DTagin) latched on the chip at the end of phase 2. During data stores, all three buses are outputs from the chip to the cache and memory interface. The AdrLo (DAd) is transmitted during phase 1 and the Data (Dout) and Tag (DTagout) during phase 2. The memory interface combines AdrLo and Tag to generate the full 32-bit real address for main memory access. Refer to figure 32 for details.

The cache interface integrates all circuitry that would normally be required between a processor and raw cache RAMS, such as the control lines for cache write and tristate output enables which are all generated on chip.

Note: *Partial word stores such as Store byte and Store halfword are handled as read-modify-write operations by the SAB-R3000. During the initial memory access pipestage of a partial word store, the processor does a Load (a read) from the cache at the store address and forces a single cycle stall. If the load hits, then the data to be stored is merged with the cache data – this is done in the SAB-R3000. Then the result is written into the cache during the stall cycle. It is done this way because the cache is word addressable only.*

Memory Interface

The memory interface which is shown on the bottom left of figure 31 contains several signals which synchronize memory access events. MemRd# and MemWr# are asserted on cache miss (i.e. main memory read access initiated) and store respectively, while CpCond0 determines if it is a single or multiple word transfer. The access type, i.e. byte, half word, tribyte, and word transfers are determined by the AccTy(2:0) bits.

The principal supporting mechanism for main memory operations is the processor stall cycle. Main memory stalls occur when loads miss in the cache or when stores are blocked by the write buffer. RdBusy and WrBusy# control the termination and initiation of the stalls when the cache misses or the write buffer is full. BusError# warns of memory access errors such as parity error or bus timeout. XEn# is the read enable signal for the read buffer. The memory interface can also support system configurations where one or both caches are missing.

External Coprocessor Interface

The external coprocessor interface is illustrated on the bottom right of figure 31. It is designed to support the SAB-R3010 floating point accelerator, in what is called a tightly coupled interface, and up to two additional coprocessors. External coprocessors are connected to the DATA bus only. During each cycle in which a valid Instruction-Data pair is on the bus, the coprocessors accept an Instruction. The coprocesors decode the Instruction in parallel with the main processor and, if it is a coprocessor Instruction, one of the coprocessors will proceed to execute the Instruction. A coprocessor condition (CpCond) signal, one for each coprocessor type, allows the main processor to branch on a coprocessor condition set up by a previous operation. Any coprocessor can assert

CpBusy to stall the main CPU when a coprocessor instruction is issued while the coprocessor in question still has the required functional unit busy with an earlier operation. The SAB-R3000 asserts Run# to advance operations in the coprocessors. When Run# is deasserted in the n th cycle, coprocessors disregard the Instruction-Data pair presented in the n-1th cycle. The assertion of Exc# (Exception) indicates that the SAB-R3000 is taking an exception. CpSync# is used for timing synchronization between the SAB-R3000 and the coprocessor.

System Configuration

Due to the flexible interfaces of the SAB-R3000 it can be used in a variety of system configurations, ranging from high-end Workstations and parallel processors to low-end embedded control applications.

A high performance system configuration, which is suitable for high-end computing applications, is shown in figure 33.

The main components of this system, along with the SAB-R3000, are separate Instruction and Data caches (64KByte each) and the SAB-R3010 Floating Point Accelerator. Main memory consists of DRAM. When a 25 MHz SAB-R3000 and SAB-R3010 are used in conjunction with 20ns SRAMs this system can deliver 22 VAX mips. Such system configurations are employed in high-end UNIX Workstations.

The SAB-R3000 is not only suited to high-end computing applications, it is also well suited to provide cost effective solutions for embedded control applications. The SAB-R3000 may be used to design systems with different degrees of performance. This can be achieved by varying the cache size (4 to 256 KByte each), the number of primary caches (0, 1 or 2), the I/O system configuration and the frequency the system will run at.

There is a 512 Mbyte unmapped uncached region in tbe Address Space organisation (see Memory Management System) which can be used for a slow main memory interface when a cache is not implemented. There are also two other possibilities to implement a system without cache

(a) cause a cache miss all the time; this can be achieved by having an external register, which upon cache access requests by the processor returns a deasserted Valid bit (i.e. invalid cache entry).

(b) Have the operating system mark every page as uncacheable in the page table/TLB by setting the "N" (Non-cacheable) bit in the TLB entry. This should be done for both Kernel and User.

An example of a cost effective system configuration for a deterministic real time embedded application is illustrated in figure 34.

Figure 33
Solution for High-End-Computing Application

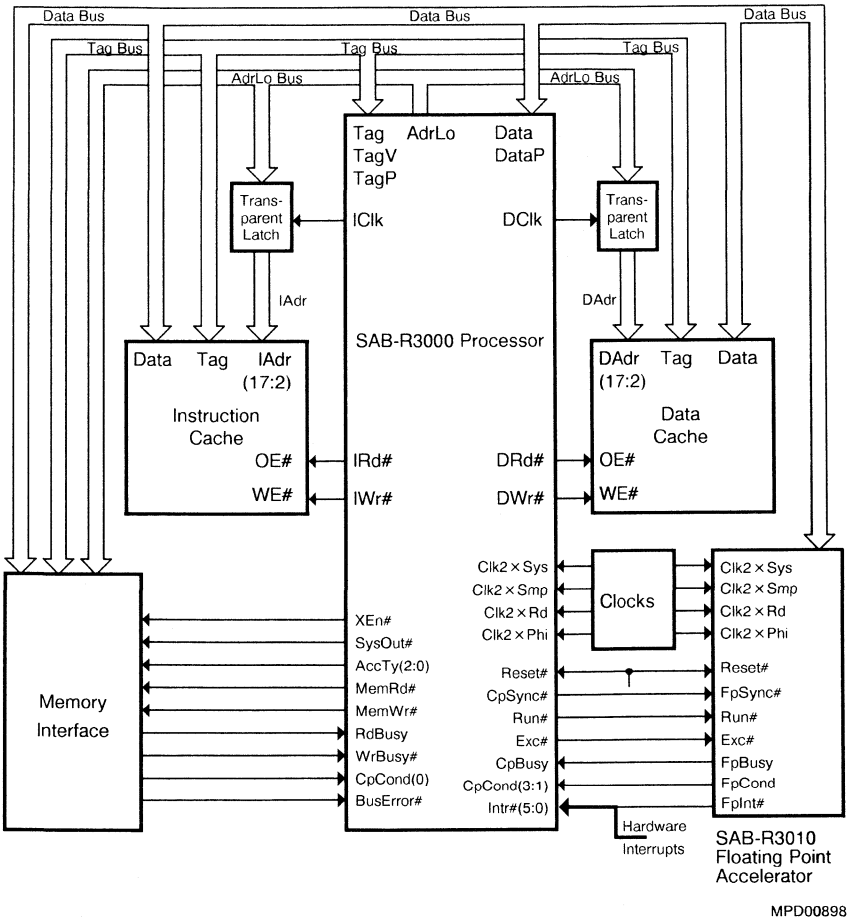
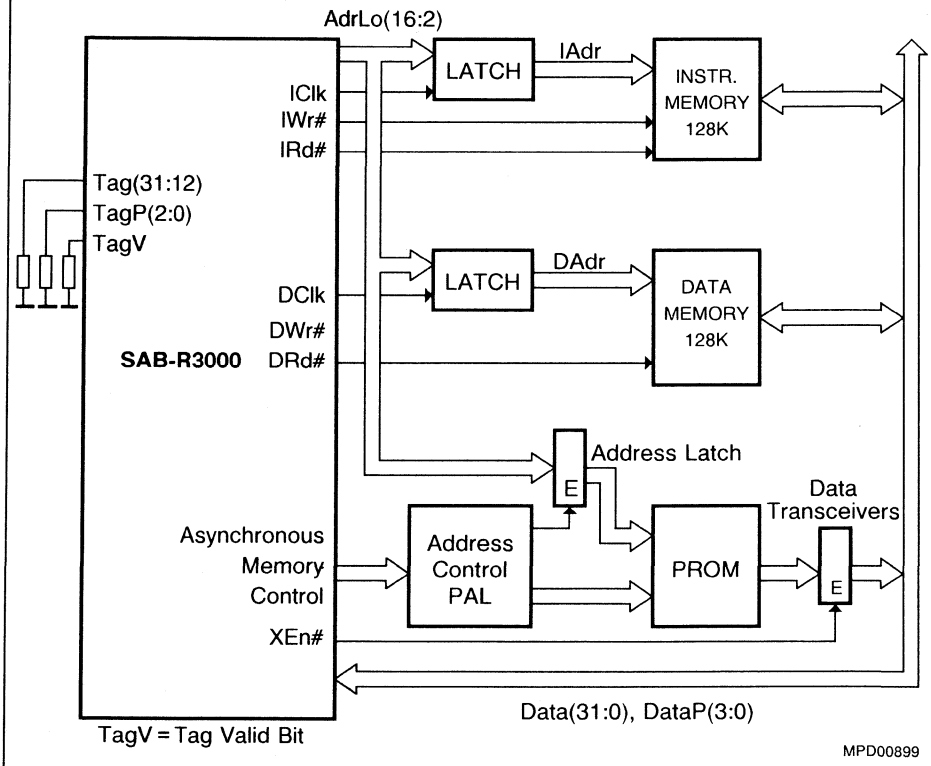


Figure 34
Solution for a Predictable Real Time Controller System



It is extremely difficult to apply the normal cache solution here due to the deterministic requirement. The SAB-R3000 can be configured so that a deterministic behaviour can be gauranteed. The technique employed is to use the synchronous bus (cache interface) to drive SRAMs where they perform the function of local main memory. The system configuration illustrated here is an example of a real time system with predictable responses of external and internal events, minimum context switch overhead and with a deterministic behaviour.

The system consists of a memory system with 128 Kbytes for Instructions and 128 Kbytes for Data (SRAMs). The asynchronous memory control (main memory interface) is used to address the PROM which contains the program to be loaded at initialization. With a 25 MHz SAB-R3000 and 20 ns SRAMs over 20 VAX mips can be achieved. The worst case reaction time to an interrupt is 7 to 9 instructions, which consists of the system overhead until the user interrupt routine takes control. In this case, due to the fact that everything is in the SRAMs, it takes only one cycle per instruction so the reaction time to an interrupt is 7 to 9 machine cycles (280 - 360 ns).

Instruction Set Summary

The following section is a table of the instructions available in the SAB-R3000. The instructions are listed in alphabetical order. For a more detailed description of the operation of each instruction refer to the "MIPS RISC ARCHITECTURE" book by Gerry Kane. A chart at the end of this section lists the bit encoding for the constant fields of each instruction.

Instruction Notation Convention

The table that follows is split up into three columns:- Instruction, Format and Operation. The Instruction column contains the mnemonic name of the instruction and its meaning. The instruction format (refer to figure 11) and Assembly language notation, for each instruction, is listed in the Format column. The Operation column describes the operation performed by each instruction using a high level language notation. Special symbols used in the notation are described in table 4.

Table 4
SAB-R3000 Instruction Operation Notations

Symbol	Meaning
←	Assignment
‖	Bit string concatenation
x^y	Replication of bit value *x* into a *y*-bit string. Note that *x* is always a single-bit value.
$x_{y..z}$	Selection of bits *y* through *z* of bit string *x*. Little-endian bit notation is always used. If y is less than *z*, this expression is an empty (zero length) bit string.
+	Two's complement addition
–	Two's complement subtraction
*	Two's complement multiplication
div	Two's complement integer division
mod	Two's complement modulo
<	Two's complement less than comparison
and	Bitwise logic AND
or	Bitwise logic OR
xor	Bitwise logic XOR
nor	Bitwise logic NOR
GPR[x]	General Register *x*. The content of GPR[0] is always zero. Attempts to alter the content of GPR[0] have no effect
CPR[z,x]	Coprocessor unit *z*, general register *x*
CCR[z,x]	Coprocessor unit *z*, control register *x*
T + i	Indicates the time steps (CPU cycles) between operations. Thus, operations identified as occurring at T + 1 are performed during the cycle following the one where the instruction was initiated. This type of operation occurs with loads, stores, jumps, branches and coprocessor instructions.
vAddress	Virtual address
pAddress	Physical address

In the Load/Store operation descriptions, the functions listed in table 5 are used to summarise the handling of virtual addresses and physical memory.

Table 5
Load/Store Common Functions

Function	Description
Addr Translation	Uses the TLB to find the physical address given the virtual address. The function fails and an exception is taken if the entry for the page containing the virtual address is not present in the TLB (Translation Lookaside Buffer).
Load Memory	Uses the cache and main memory to find the contents of the word containing the specified physical address. The low-order two bits of the address and the access type field indicate which of each of the four bytes within the data word need to be returned. If the cache is enabled for this access. The entire word is returned and loaded into the cache.
Store Memory	Uses the cache, write buffer, and main memory to store the word or part of word specified as data into the word containing the specified physical address. The low-order two bits of the address and the access type field indicate which of the four bytes within the data word should be stored.

Instruction	Format	Operation
ADD: Add	R-type; ADD rd, rs, rt	T: GPR[rd] ← GPR[rs] + GPR[rt]
ADDI: Add Immediate	I-type; ADDI rt, rs, immediate	T: GPR[rt] ← GPR[rs] + $(immediate_{15})^{16} \| immediate_{15..0}$
ADDIU: Add Immediate Unsigned	I-type; ADDIU rt, rs, immediate	T: GPR[rt] ← GPR[rs] + $(immediate_{15})^{16} \| immediate_{15..0}$
ADDU: Add Unsigned	R-type; ADDU rd, rs, rt	T: GPR[rd] ← GPR[rs] + GPR[rt]
AND: And	R-type; AND rd,rs,rt	T: GPR[rd] ← GPR[rs] and GPR[rt]
ANDI: And Immediate	I-type; ANDI rt, rs, immediate	T: GPR[rt] ← $0^{16} \|$(immediate and $GPR[rs]_{15..0}$)
BCzF: Branch On Coprocessor z False	I-type; BCzF offset	T: target ← $(offset_{15})^{14} \|$offset $\|0^2$ condition ← not CpCond[z] T + 1: if condition then PC ← PC + target endif
BCzT: Branch On Coprocessor z True	I-type; BCzT offset	T: target ← $(offset_{15})^{14} \|$offset $\|0^2$ condition ← CpCond[z] T + 1: if condition then PC ← PC + target endif
BEQ: Branch on Equal	I-type; BEQ rs, rt, offset	T: target ← $(offset_{15})^{14} \|$ offset $\| 0^2$ condition ← (GPR[rs] = GPR[rt]) T + 1: if condition then PC ← PC + target endif
BGEZ: Branch on Greater than or Equal to Zero	I-type; BGEZ rs, offset	T: target ← $(offset_{15})^{14} \|$ offset $\| 0^2$ condition ← $(GPR[rs]_{31} = 0)$ T + 1: if condition then PC ← PC + target endif

Instruction	Format	Operation
BGEZAL: Branch on Greater than or Equal to Zero And Link	I-type; BGEZAL rs, offset	T: target ← $(offset_{15})^{14}$ ‖ offset ‖ 0^2 condition ← (GPR[rs] $_{31}$ = 0) GPR[31] ← PC + 8 T + 1: if condition then PC ← PC + target endif
BGTZ: Branch on Greater Than Zero	I-type; BGTZ rs, offset	T: target ← $(offset_{15})^{14}$ ‖ offset ‖ 0^2 condition ← (GPR[rs] $_{31}$ = 0) and (GPR[rs] ≠ GPR[r0]) T + 1: if condition then PC ← PC + target endif
BLEZ: Branch on Less than or Equal to Zero	I-type; BLEZ rs, offset	T: target ← $(offset_{15})^{14}$ ‖ offset ‖ 0^2 condition ← (GPR[rs] $_{31}$ = 1) or (GPR[rs] = GPR[r0]) T + 1: if condition then PC ← PC + target endif
BLTZ: Branch on Less Than Zero	I-type; BLTZ rs, offset	T: target ← $(offset_{15})^{14}$ ‖ offset ‖ 0^2 condition ← (GPR[rs] $_{31}$ = 1) T + 1: if condition then PC ← PC + target endif
BLTZAL: Branch on Less Than Zero And Link	I-type; BLTZAL rs, offset	T: target ← $(offset_{15})^{14}$ ‖ offset ‖ 0^2 condition ← (GPR[rs] $_{31}$ = 1) GPR[31] ← PC + 8 T + 1: if condition then PC ← PC + target endif
BNE: Branch on Not Equal	I-type; BNE rs, rt, offset	T: target ← $(offset_{15})^{14}$ ‖ offset ‖ 0^2 condition ← (GPR[rs] ≠ GPR[rt]) T + 1: if condition then PC ← PC + target endif
BREAK: Break	R-type; BREAK	PC← Exception Handler

Instruction	Format	Operation
CFCz: Move Control From Co-processor z	R-type; CFCz rt, rd	T: data ← CCR[z,rd] T + 1: GPR[rt] ← data
COPz: Coprocessor Operation z	Coprocessor type: COPz cofun	T: CoprocessorOperation (z. cofun)
CTCz: Move Control To Coprocessor z	R-type; CTCz	T: data ← GPR[rt] T + 1: CCR[z,rd] ← data
DIV: Divide	R-type; DIV rs, rt	T-2: LO ← undefined HI ← undefined T–1: LO ← undefined HI ← undefined T: LO ← GPR[rs] *div* GPR[rt] HI ← GPR[rs] *mod* GPR[rt]
DIVU: Divide Unsigned	R-type; DIVU rs, rt	T-2: LO ← undefined HI ← undefined T–1: LO ← undefined HI ← undefined T: LO ← (0‖GPR[rs]) *div* (0‖GPR[rt]) HI ← (0‖GPR[rs]) *mod* (0‖GPR[rt])
J: Jump	J-type; J target	T: temp ← $PC_{31..28}$ ‖ target ‖ 0^2 T + 1: PC ← temp
JAL: Jump And Link	J-type; JAL target	T: temp ← $PC_{31..28}$ ‖ target ‖ 0^2 GPR[31] ← PC + 8 T + 1: PC ← temp
JALR: Jump And Link Register	R-type; JALR rs JALR rd, rs	T: temp ← GPR[rs] GPR[rd] ← PC + 8 T + 1: PC ← temp
JR: Jump Register	R-type; JR rs	T: temp ← GPR[rs] T + 1: PC ← temp

Instruction	Format	Operation

LB:
Load Byte
	I-type;
	LB rt, offset (base)
		T: vAddress←(offset$_{15}$)^{16} ‖ offset$_{15..0}$ + GPR[base]
		(pAddress, nonCacheable)←AddrTranslation (vAddress)
		mem←LoadMemory (nonCacheable, BYTE, pAddress)
		byte←vAddress$_{1..0}$
		T + 1: if BigEndian then
		GPR[rt]←(mem$_{31-8*byte}$)^{24} ‖ mem$_{31-8*byte..24-8*byte}$
		else
		GPR[rt]←(mem$_{7+8*byte}$)^{24} ‖ mem$_{7+8*byte..8*byte}$
		endif

LBU:
Load Byte
Unsigned
	I-type;
	LBU rt, offset (base)
		T: vAddress←(offset$_{15}$)^{16} ‖ offset$_{15..0}$ + GPR[base]
		(pAddress, nonCacheable)←AddrTranslation (vAddress)
		mem←LoadMemory (nonCacheable, BYTE, pAddress)
		byte←vAddress$_{1..0}$
		T + 1: if BigEndian then
		GPR[rt]←0^{24} ‖ mem$_{31-8*byte..24-8*byte}$
		else
		GPR[rt]←0^{24} ‖ mem$_{7+8*byte..8*byte}$
		endif

LH:
Load Halfword
	I-type;
	LH rt, offset (base)
		T: vAddress←(offset$_{15}$)^{16} ‖ offset$_{15..0}$ + GPR[base]
		(pAddress, nonCacheable)←AddrTranslation (vAddress)
		mem←LoadMemory(nonCacheable,HALFWORD,pAddress)
		byte←vAddress$_{1..0}$
		T + 1: if BigEndian then
		GPR[rt]←(mem$_{31-8*byte}$)^{16} ‖ mem$_{31-8*byte..16-8*byte}$
		else
		GPR[rt]← (mem$_{15+8*byte}$)^{16} ‖ mem$_{15+8*byte..8*byte}$
		endif

Instruction	Format	Operation

LHU:
Load Halfword Unsigned
	I-type;
	LHU rt, offset (base)
		T: vAddress←$(offset_{15})^{16}$ ‖ $offset_{15..0}$ + GPR[base]
		(pAddress, nonCacheable)← AddrTranslation (vAddress)
		mem←LoadMemory(nonCacheable, HALFWORD,pAddress)
		byte←$vAddress_{1..0}$
		T + 1: if BigEndian then
		GPR[rt]←0^{16} ‖ $mem_{31-8*byte..16-8*byte}$
		else
		GPR[rt]←0^{16} ‖ $mem_{15+8*byte..8*byte}$
		endif

LUI: Load Upper Immediate	I-type; LUI rt, immediate	T: GPR[rt]←immediate ‖ 0^{16}
LW: Load Word	I-type; LW rt, offset (base)	T: vAddress←$(offset_{15})^{16}$ ‖ $offset_{15..0}$ + GPR[base] (pAddress, nonCacheable)← AddrTranslation (vAddress) mem←LoadMemory (nonCacheable,WORD, pAddress) byte←$vAddress_{1..0}$ T + 1: GPR[rt]←mem;
LWCz: Load Word to Coprocessor z	I-type; LWCz rt, offset (base)	T: vAddress←$(offset_{15})^{16}$ ‖$offset_{15..0}$ + GPR[base] (pAddress, nonCacheable)← AddrTranslation (vAddress) mem←LoadMemory(nonCacheable,WORD,$pAddress_{31..2}$ ‖0^2) byte←$vAddress_{1..0}$ T + 1: CPR[z,rt]←mem;

Instruction	Format	Operation

LWL:
Load Word Left
	I-type;
	LWL rt, offset (base)
		T: vAddress←(offset$_{15}$)^{16} ‖ offset$_{15..0}$ + GPR[base]
		(pAddress, nonCacheable)← AddrTranslation (vAddress)
		byte←vAddress$_{1..0}$
		if BigEndian then
		mem←LoadMemory(nonCacheable,WORD-byte,pAddress)
		else
		mem←LoadMemory(nonCacheable,byte,pAddress$_{31..2}$ ‖0^2)
		endif
		T + 1: if BigEndian then
		GPR[rt]←mem$_{31-8*byte..0}$ ‖ GPR[rt]$_{8*byte-1..0}$
		else
		GPR[rt]←mem$_{7+8*byte..0}$ ‖ GPR[rt]$_{23-8*byte..0}$
		endif

LWR:
Load Word Right
	I-type;
	LWR rt, offset (base)
		T: vAddress←(offset$_{15}$)^{16} ‖ offset$_{15..0}$ + GPR[base]
		(pAddress, nonCacheable)← AddrTranslation (vAddress)
		byte←vAddress$_{1..0}$
		if BigEndian then
		mem←LoadMemory(nonCacheable,byte,pAddress$_{31..2}$ ‖0^2)
		else
		mem←LoadMemory(nonCacheable,byte,WORD-byte,pAddress)
		endif
		T + 1: if BigEndian then
		GPR[rt]←GPR[rt]$_{31..8+8*byte}$ ‖ mem$_{31..24-8*byte}$
		else
		GPR[rt]←GPR[rt]$_{31..24-8*byte}$ ‖ mem$_{31..8+8*byte}$
		endif

MFC0: Move From System Control Coprocessor z	R-type; MFC0 rt, rd	T: data CPR[0,rd] T + 1: GPR[rt] ← data
MFCz: Move From Coprocessor z	R-type; MFCz rt, rd	T: data ← CPR[z,rd] T + 1: GPR[rt] ← data

Instruction	Format	Operation
MFHI: Move From HI	R-type; MFHI rd	T: GPR[rd] ← HI
MFLO: Move From LO	R-type; MFLO, rd	T: GPR[rd] ← LO
MTC0: Move To System Control Coprocessor	R-type; MTC0 rt, rd	T: data ← GPR[rt] T + 1: CPR[0,rd] ← data
MTCz: Move To Coprocessor z	R-type; MTCz rt, rd	T: data ← GPR[rt] T + 1: CPR[z,rd] ← data
MTHI: Move To HI	R-type; MTHI rs	T–2: HI ← undefined T–1: HI ← undefined T: HI ← GPR[rs]
MTLO: Move To LO	R-type; MTLO rs	T–2: LO ← undefined T–1: LO ← undefined T: LO ← GPR[rs]
MULT: Multiply	R-type; MULT rs,rt	T–2: LO ← undefined HI ← undefined T–1: LO ← undefined HI ← undefined T: t ← GPR[rs]*GPR[rt] LO ← $t_{31..0}$ HI ← $t_{63..32}$
MULTU: Multiply Unsigned	R-type; MULTU rs,rt	T–2: LO ← undefined HI ← undefined T–1: LO ← undefined HI ← undefined T: t ← (0 ‖GPR[rs])*(0 ‖GPR[rt]) LO ← $t_{31..0}$ HI ← $t_{63..32}$
NOR: Nor	R-type; NOR rd, rs, rt	T: GPR[rd] ← GPR[rs] *nor* GPR[rt]

Instruction	Format	Operation
OR: Or	R-type; OR rd, rs, rt	T: GPR[rd] ← GPR[rs] *or* GPR[rt]
ORI: Or Immediate	I-type; ORI rt, rs, immediate	T: GPR[rt] ← $GPR[rs]_{31..16}$ ‖(immediate *or* $GPR[rs]_{15..0}$)
RFE: Restore From Exception	R-type; RFE	T: SR ← $SR_{31..4}$ ‖ $SR_{5...2}$

SB:
Store Byte
	I-type;
	SB rt, offset (base)
		T: vAddress←$(offset_{15})^{16}$ ‖ $offset_{15..0}$ + GPR[base]
		(pAddress, nonCacheable)← AddrTranslation (vAddress)
		byte←$vAddress_{1..0}$
		if BigEndian then
		data←$GPR[rt]_{7+8*byte..0}$ ‖ $0^{24-8*byte}$
		else
		data←$GPR[rt]_{31-8*byte..0}$ ‖ 0^{8*byte}
		endif
		T + 1: StoreMemory (nonCacheable, BYTE, data, pAddress)

SH:
Store Halfword
	I-type;
	SH rt, offset (base)
		T: vAddress←$(offset_{15})^{16}$ ‖ $offset_{15..0}$ + GPR[base]
		(pAddress, nonCacheable)← AddrTranslation (vAddress)
		byte←$vAddress_{1..0}$
		IF BigEndian then
		data←$GPR[rt]_{15+8*byte..0}$ ‖ $0^{15-8*byte}$
		else
		data←$GPR[rt]_{31-8*byte..0}$ ‖ 0^{8*byte}
		endif
		T + 1: StoreMemory (nonCacheable, HALFWORD, data, pAddress)

SLL: Shift Left Logical	I-type; SLL rd, rt shamt	T: GPR[rd] ← $GPR[rt]_{31-shamt..0}$ ‖ 0^{shamt}
SLLV: Shift Left Logical Variable	R-type; SLLV rd, rt, rs	T: GPR[rd] ← $GPR[rt]_{(31-GPR[rs]_{4..0})..0}$ ‖ $0^{GPR[rs]_{4..0}}$

Instruction	Format	Operation

SLT:
Set on Less Than
	R-type;
	SLT rd, rs, rt
		T: if GPR[rs] < GPR[rt] then
		GPR[rd] ← $0^{31} \| 1$
		else
		GPR[rd] ← 0^{32}
		endif

SLTI:
Set on Less Than Immediate
	I-type;
	SLTI rt, rs, immediate
		T: if GPR[rs] < $((immediate_{15})^{16} \| immediate_{15..0})$ then
		GPR[rt] ← $0^{31} \| 1$
		else
		GPR[rt] ← 0^{32}
		endif

SLTIU:
Set on Less Than Immediate Unsigned
	I-type;
	SLTIU rt, rs, immediate
		T: if $(0 \| GPR[rs]) < (0 \| (immediate_{15})^{16} \| immediate_{15..0})$ then
		GPR[rt] ← $0^{31} \| 1$
		else
		GPR[rt] ← 0^{32}
		endif

SLTU:
Set on Less Than Unsigned
	R-type;
	SLTU rd, rs, rt
		T: if $(0 \| GPR[rs]) < (0 \| GPR[rt])$ then
		GPR[rd] ← $0^{31} \| 1$
		else
		GPR[rd] ← 0^{32}
		endif

SRA: Shift Right Arithmetic	R-type; SRA rd, rt, shamt	T: $GPR[rd] \leftarrow (GPR[rt]_{31})^{shamt} \| GPR[rt]_{31..shamt}$
SRAV: Shift Right Arithmetic Variable	R-type; SRAV rd, rt, rs	T: $GPR[rd] \leftarrow (GPR[rt]_{31})^{GPR[rs]4..0} \| GPR[rt]_{31..(GPR[rs]_{4..0})}$
SRL: Shift Right Logical	R-type; SRL rd, rt, shamt	T: $GPR[rd] \leftarrow 0^{shamt} \| GPR[rt]_{31..shamt}$
SRLV: Shift Right Logical Variable	R-type; SRLV rd, rt, rs	T: $GPR[rd] \leftarrow 0^{GPR[rs]4..0} \| GPR[rt]_{31..(GPR[rs]_{4..0})}$

Instruction	Format	Operation
SUBU: Subtract Unsigned	R-type; SUBU rd, rs, rt	T: GPR[rd] ← GPR[rs] –GPR[rt]
SUB: Subtract	R-type; SUB rd, rs, rt	T: GPR[rd] ← GPR[rs] –GPR[rt]
SW: Store Word	I-type; SW rt, offset (base)	T: vAddress←$(offset_{15})^{16} \| offset_{15..0}$ + GPR[base] (pAddress, nonCacheable)←AddrTranslation (vAddress) data←GPR[rt] T + 1: StoreMemory (nonCacheable, WORD, data, pAddress)
SWCz: Store Word from Coprocessor z	I-type; SWCz rt, offset (base)	T: vAddress←$(offset_{15})^{16} \| offset_{15..0}$ + GPR[base] (pAddress, nonCacheable)←AddrTranslation (vAddress) data←CPR[z,t] T + 1: StoreMemory (nonCacheable, 15, data, $pAddress_{31..2} \| 0^2$)

SWL:
Store Word Left
	I-type;
	SWL rt, offset (base)
		T: vAddress←$(offset_{15})^{16} \| offset_{15..0}$ + GPR[base]
		(pAddress,nonCacheable)←AddrTranslation (vAddress)
		byte←$vAddress_{1..0}$
		if BigEndian then
		data←$0^{8*byte} \| GPR[rt]_{31..8*byte}$
		else
		data← $0^{24-8*byte} \| GPR[rt]_{31..24-8*byte}$
		endif
		T + 1: if BigEndian then
		StoreMemory (nonCacheable,WORD-byte,data,pAddress)
		else
		StoreMemory (nonCacheable,byte,data,$pAddress_{31..2} \| 0^2$)
		endif

Instruction	Format	Operation

SWR:
Store Word Right
	I-type;
	SWR rt, offset (base)
		T: vAddress←(offset$_{15}$)^{16} ‖ offset$_{15..0}$ + GPR[base]
		(pAddress, nonCacheable)← AddrTranslation (vAddress)
		byte←vAddress$_{1..0}$
		if BigEndian then
		data←GPR[rt]$_{7+8*byte..0}$ ‖ 0$^{24-8*byte}$
		else
		data← GPR[rt]$_{31-8*byte..0}$ ‖ 0^{8*byte}
		endif
		T + 1: if BigEndian then
		StoreMemory (nonCacheable,byte,data, pAddress$_{31..2}$ ‖ 0^{2})
		else
		StoreMemory (nonCacheable,WORD-byte,data,pAddress)
		endif

SYSCALL: System Call	R-type; SYSCALL	PC ← ExceptionHandler

TLBP:
Probe TLB for matching entry
	R-type;
	TLBP
		T: Index← 1 ‖ 0^{31}
		for i in 0..TLBEntries-1
		if ((TLB$_{63..44}$[i] = EntryHi$_{31..12}$) and
		(TLB$_{8}$ or (TLB$_{43..38}$ = EntryHi$_{11..6}$))) then
		Index← 0^{18} ‖ i$_{5..0}$ ‖ 0^{8}
		endif
		endfor

TLBR: Read indexed TLB entry	R-type; TLBR	T: EntryHi← TLB [Index$_{13..8}$]$_{63..32}$ EntryLo← TLB [Index$_{13..8}$]$_{31..0}$
TLBWI: Write Indexed TLB entry	R-type; TLBWI	T: TLB [Index$_{13..8}$]$_{63..32}$← EntryHi TLB [Index$_{13..8}$]$_{31..0}$← EntryLo
TLBWR: Write Random TLB entry	R-type; TLBWR	T: TLB[Random$_{13..8}$]$_{63..32}$← EntryHi TLB[Random$_{13..8}$]$_{31..0}$← EntryLo
XOR: Exclusive Or	R-type; XOR rd, rs, rt	T: GPR[rd]← GPR[rs] *xor* GPR[rt]
XORI: Exclusive Or Immediate	I-type; XORI rt, rs, immediate	T: GPR[rt]← GPR[rs]$_{31..16}$ ‖ (immediate *xor* GPR[rs]$_{15..0}$)

Instruction Encoding

op

31..29 \ 28..26	0	1	2	3	4	5	6	7
0	SPECIAL	REGIMM	J	JAL	BEQ	BNE	BLEZ	BGTZ
1	ADDI	ADDIU	SLTI	SLTIU	ANDI	ORI	XORI	LUI
2	COP0	COP1	COP2	COP3	⊗	⊗	⊗	⊗
3	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗
4	LB	LH	LWL	LW	LBU	LHU	LWR	⊗
5	SB	SH	SWL	SW	⊗	⊗	SWR	⊗
6	⊗	LWC1	LWC2	LWC3	⊗	⊗	⊗	⊗
7	⊗	SWC1	SWC2	SWC3	⊗	⊗	⊗	⊗

SPECIAL function

5..3 \ 2..0	0	1	2	3	4	5	6	7
0	SLL	⊗	SRL	SRA	SLLV	⊗	SRLV	SRAV
1	JR	JALR	⊗	⊗	SYSCALL	BREAK	⊗	⊗
2	MFHI	MTHI	MFLO	MTLO	⊗	⊗	⊗	⊗
3	MULT	MULTU	DIV	DIVU	⊗	⊗	⊗	⊗
4	ADD	ADDU	SUB	SUBU	AND	OR	XOR	NOR
5	⊗	⊗	SLT	SLTU	⊗	⊗	⊗	⊗
6	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗
7	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗

REGIMM rt

20..19 \ 18..16	0	1	2	3	4	5	6	7
0	BLTZ	BGEZ	~	~	~	~	~	~
1	~	~	~	~	~	~	~	~
2	BLTZAL	BGEZAL	~	~	~	~	~	~
3	~	~	~	~	~	~	~	~

⊗ Codes marked with a '⊗' cause unimplemented instruction exceptions and are reserved for future versions of the architecture.

~ Codes marked with a '~' are not valid and are reserved for future versions of the architecture. The results of such an encoding are undefined.

COPz rs

25..24 \ 23..21	0	1	2	3	4	5	6	7
0	MF	~	CF	~	MT	~	CT	~
1	~	⊗	⊗	⊗	⊗	⊗	⊗	⊗
2	CO							
3								

COPz rt

20..19 \ 18..16	0	1	2	3	4	5	6	7
0	BCF	BCT	~	~	~	~	~	~
1	~	~	~	~	~	~	~	~
2	~	~	~	~	~	~	~	~
3	~	~	~	~	~	~	~	~

COP0 function

4..3 \ 2..0	0	1	2	3	4	5	6	7
0	~	TLBR	TLBWI	~	~	~	TLBWR	~
1	TLBP	~	~	~	~	~	~	~
2	RFE	~	~	~	~	~	~	~
3	~	~	~	~	~	~	~	~
4	~	~	~	~	~	~	~	~
5	~	~	~	~	~	~	~	~
6	~	~	~	~	~	~	~	~
7	~	~	~	~	~	~	~	~

⊗ Codes marked with a '⊗' cause unimplemented instruction exceptions and are reserved for future versions of the architecture.

~ Codes marked with a '~' are not valid and are reserved for future versions of the architecture. The results of such an encoding are undefined.

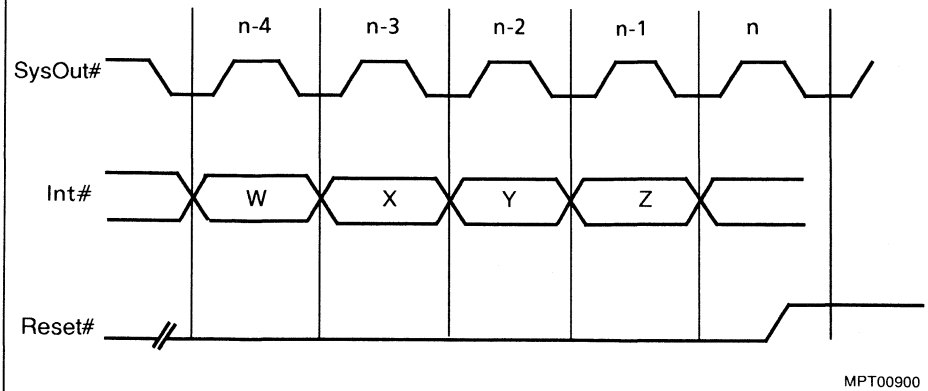
Resetting the SAB-R3000

The Reset# input signal is used to force processor execution to start at the reset vector (reset exception servicing routine) and to initialize the processor state. The Reset# signal must be asserted for a minimum of 6 cycles to guarantee processor initialization. After a reset has occurred (i.e. before the exception handling routine has been executed) the following processor state is guaranteed:

- KUc, the current Kernel/User bit, is zero corresponding to Kernel mode.
- IEc, the current interrupt enable bit, is zero corresponding to interrupts disabled.
- TS, the TLB shutdown bit, is zero corresponding to TLB enabled.
- SwC, the Swap Cache bit, is zero corresponding to caches not swapped.
- BEV, the Boot Exception Vector bit, is one corresponding to selection of the bootstrap exception vector.
- The Random register is set to 63.

When the Reset# signal is deasserted in the n th cycle, the logic levels on the 6 interrupt pins during the n-4, n-3, n-2 and n-1 cycles are sampled by the processor to determine various processor operating modes such as Endianness, Cachelessness, Test etc. The last four cycles before the cycle in which the Reset# signal is deasserted are called the W, X, Y and Z cycles, respectively. Figure 35 illustrates these four cycles. The Reset# timings are described in the Timing Parameters section.

Figure 35
W, X, Y and Z Cycles on Reset# Deassertion



The Mode Select Summary Table, given in the Pin Definitions and Functions section, summarizes the processors mode selectable features and is reproduced here.

Table 6
Summary of Mode Select

Int#	W Cycle	X Cycle	Y Cycle	Z Cycle
Int#(0)	DBlkSize0#	DBlkSize1#	ExCache	BigEndian#
Int#(1)	IBlkSize0#	IBlkSize1#	Reserved	Tristate#
Int#(2)	Reserved	IStream	Reserved	NoCache#
Int#(3)	Reserved	StorePartial	MPS	BusDriveOn
Int#(4)	PhaseDelayOn#	PhaseDelayOn#	PhaseDelayOn#	PhaseDelayOn#
Int#(5)	R3K#	R3K#	R3K#	R3K#

Note that all reserved modes in this table must be driven asserted to guarantee compatibility with future processor revisions.

Data Block Size **(DBlkSize#(1-0))** and Instruction Block Size **(IBlkSize#(1-0))** are two bit encodings of the block length used for data and instructions respectively, when block refill is enabled. Block refill is enabled by the CpCond(0) signal; refer to the main memory reads section on page 80. Table 7 illustrates this encoding.

Table 7

DBlkSize#(1-0)	IBlkSize#(1-0)	size	
00	00	32	words
01	01	16	words
10	10	8	words
11	11	4	words

Asserting **PhaseDelayOn#** causes the processor to insert additional phase delay into its input clock paths. This additional phase delay allows coprocessors to minimize their skew, i.e. phase lock, to the SAB-R3000.

When **R3K#** is asserted the processor is enabled to operate as per the SAB-R3000 specification. Deassertion of R3K# enables the processor to function as an SAB-R2000.

The instruction streaming capability is enabled when **IStream** is asserted. When IStream is deasserted, instruction streaming is disabled. Note, instruction streaming is only available when R3K# mode is selected.

Asserting **StorePartial** causes the processor to handle partial word store operations according to the SAB-R3000 specification. When StorePartial is deasserted the processor handles stores like the SAB-R2000. Note, StorePartial is only available when R3K# mode is selected.

The assertion of **ExCache** (Extended Cache size) causes the processor to provide two additional cache address outputs, AdrLo17-16. Deassertion causes these two pins to function as CpCond3-2.

Asserting **MPS** (Multi-Processing Stall) enables the multiprocessing support features of the SAB-R3000. Deasserting disables these features. Note, multiprocessing features are only available when R3K# mode is selected.

Byte order or Endianness is determined by the value of **BigEndian#**. Assertion will result in Big Endian ordering, while deassertion will result in Little Endian ordering.

Assertion of **Tristate#** causes the processor to tristate all of its outputs. In this condition the processor outputs can be driven by an external medium.

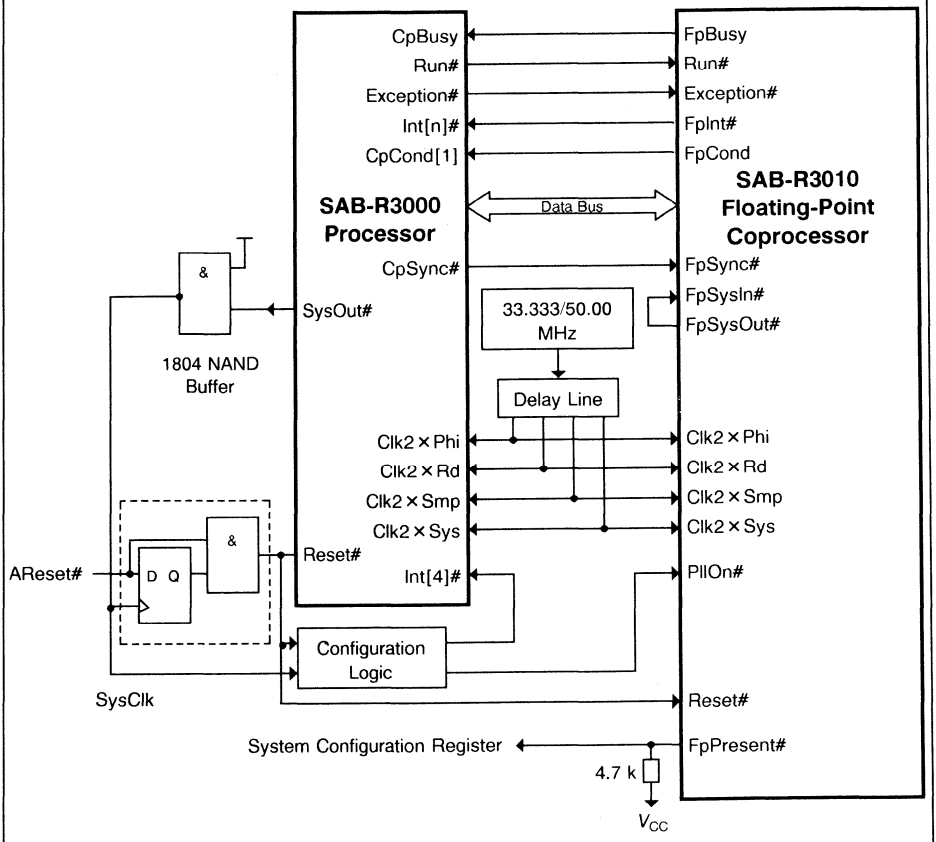
When **NoCache#** is asserted all memory references are forced to occur at the processor cycle rate, i.e. no cache miss stalls occur.

Data and Tag buses are driven during phase 2 of write busy and coprocessor busy stalls when **BusDriveOn** is asserted. If the Data and Tag buses are not being driven externally during these stalls and fast TTL inputs are connected to the bus, the BusDriveOn should be asserted to prevent bus oscillation due to the bus being pulled to the trip point of Fast TTL.

In the Mode Select Summary Table "#" means active low. This means, for example, in the Z cycle, if Int#(0) has logic value low, Big Endian mode will be selected. On the other hand, if Int#(0) has logic value high, Little Endian mode will be selected (i.e. the opposite).

Note that the Reset# signal must be asserted asynchronously and deasserted synchronously with the output Clock Sysout#. The reason for asserting Reset# asynchronously is to avoid a possible dead-lock if the Sysout# is used to clock it. More detailed information can be obtained in the "MIPS R3000 Processor Interface" specification and in a MIPS application note entitled "Resetting the R3000 and R3010". Figure 36 illustrates the SAB-R3000 and SAB-R3010 Coprocessor Interface with the reset logic.

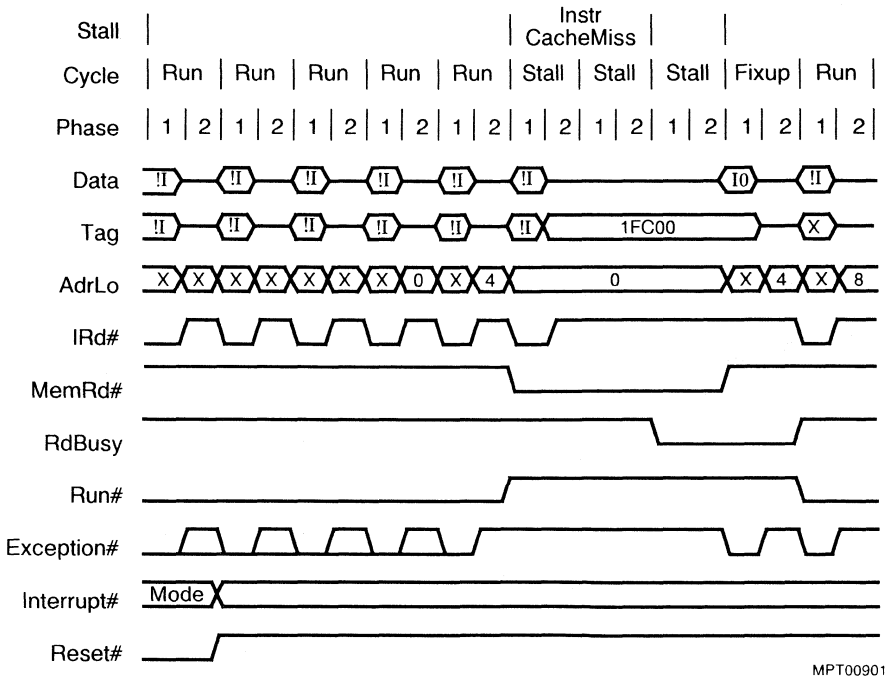
Figure 36
SAB-R3000 and SAB-R3010 Coprocessor Interface



AReset = Asynchronous Reset

The Reset Exception itself occurs when the Reset# signal is asserted and then deasserted. The Reset exception vector is selected to appear within the uncached, unmapped memory space (kseg1) of the machine so that instructions can be fetched and executed while the cache and memory system are still in an undefined state.The Reset exceptions special interrupt vector is 0xbfc00000. This is a virtual address and resides in kseg1 as explained in the Memory Management section. Kseg1 is direct mapped into the first 512 MBytes of physical memory and the physical address is defined by subtracting 0xa0000000 from the virtual address of the reset exceptions interrupt vector, which yields 0x1fc00000. This can be seen in figure 37, which illustrates the sequence of events when Reset# is deasserted, i.e. when the processor comes out of reset. As can be seen the address (physical) on the Tag and AdrLo Buses is 0x1fc00000, as expected. Refer to table 10 in the Timing Specification section for the notation.

Figure 37
Reset Behavior



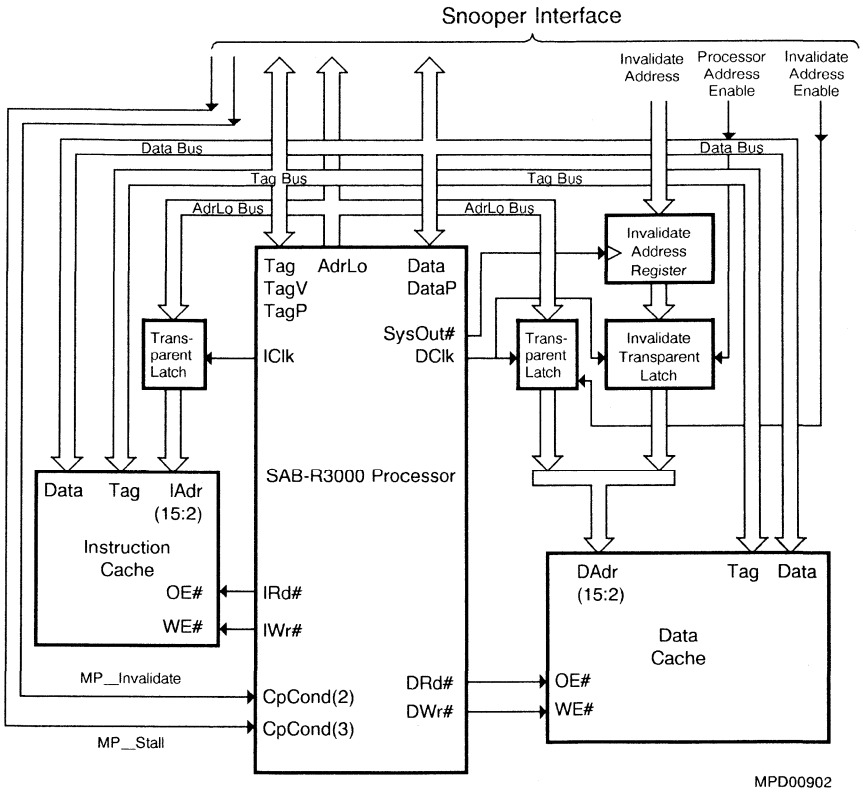
Multi-Processing Support

Architecturally the SAB-R3000 is suitable for Multi-processing. The physical cache (tag and index being physical) eases the implementation of a multi-processing system, due to the fact that there are no synonym problems caused by mapping different virtual addresses to the same physical address. The way in which the architecture is partitioned, i.e. on-chip cache control and MMU and off-chip Floating Point Accelerator, allows the designer to choose a compact and flexible processing node as required.

As described in the previous section it is possible, during Reset, to select various modes in which the SAB-R3000 should operate, one of these is multi-processing mode. When this mode is selected the signals CpCond2 and CpCond3 which normally double as address pins now switch roles to MP_Invalidate and MP_Stall respectively. This has the consequence that now the maximum primary cache addressable is 128 Kbytes (64 Kbytes data and 64 Kbytes instruction). This comprises the hardware support provided by the SAB-R3000 for multi-processing.

The functionality of these two signals shall be described using a multi-processing system where a Snooper is employed to watch the main memory bus. The Snooper compares the addresses on the main memory bus to those in a secondary cache (high performance systems) or to a duplicate set of Tags (low cost implementation). If there is a match the Snooper can assert the MP_Stall signal, which forces a multi-processing stall (MP stall) cycle. Figure 38 shows the multi-processing interface with additional latches and the Snooper outputs.

Figure 38
MP Interface Block Diagram



During the first cycle of an MP stall no cache activity occurs. This allows the system to turn off the output enable of the SAB-R3000's transparent latch, that holds the data cache address and to turn on the output enable of a transparent latch that will be used to hold the system's read and invalidate addresses. If the MP_Invalidate signal is not asserted by the Snooper, the CPU will issue DRd# pulses every cycle and read data from the data cache until the end of the stall or until MP_Invalidate is asserted.

When MP_Invalidate is asserted it causes the processor to assert DWr# and a "0" is written to the Tag Valid bit of the cache entry, thereby invalidating that entry. At the same time an arbitrary Tag and Data, essentially the contents of the internal latches left over from earlier bus transactions, are driven out on their respective buses. The cycle after MP_Stall is deasserted also contains no cache activity. This allows the system to re-enable the transparent latch and to disable the invalidate transparent latch.

The Snooper can always invalidate a cache entry without ever reading out the Tags, by simply asserting MP_Stall and MP_Invalidate simultaneously. It also has the possibility to first read the Tags to determine if indeed the invalidation is necessary and then decide to assert the MP_Invalidate signal - this can be used in a two level cache design to check if the entry in the primary cache must be invalidated.

The only accesses made to the processor's data cache by the Snooper occur when Tags match. This minimises the frequency of interference. Instead of updating entries they are simply invalidated to minimise cost and data traffic.

An MP stall sequence is described in the Timing Parameters section.

Absolute Maximum Ratings

Case temperature under bias (T_C) . . . . . 0 to + 115 °C
Storage temperature (T_{ST}) . . . . . – 65 to + 150 °C
Supply Voltage (V_{CC}) . . . . . – 0.5 to + 7.0 V
Input voltage (V_{IN} min. = – 3.0 V for pulse width less than 15 ns) . . . . – 0.5 to + 7.0 V

Note: *Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*
Not more than one output should be shorted at a time. Duration of the short should not exceed 30 seconds.

DC Characteristics

T_C = 0 to + 90 °C; V_{CC} = 5 V ± 5%

Parameter	Symbol	Limit values						Unit	Test condition
		16.67 MHz		20 MHz		25 MHz			
		min.	max.	min.	max.	min.	max.		
Operating Parameters									
Output HIGH voltage	V_{OH}	3.5	–	3.5	–	3.5	–	V	V_{CC} = min. I_{OH} = – 4mA
Output HIGH voltage	V_{OHC} 3)	4.0	–	4.0	–	4.0	–	V	V_{CC} = min. I_{OH} = – 4mA
Output LOW voltage	V_{OL}	–	0.4	–	0.4	–	0.4	V	V_{CC} = min. I_{OL} = 4mA
Input HIGH voltage	V_{IH}	2	V_{CC} + 0.25	2	V_{CC} + 0.25	2	V_{CC} + 0.25	V	
Input LOW voltage	V_{IL} 1)	– 0.5	0.8	– 0.5	0.8	– 0.5	0.8	V	
Input HIGH voltage	V_{IHS} 2)	3.0	V_{CC} + 0.25	3.0	V_{CC} + 0.25	3.0	V_{CC} + 0.25	V	
Input LOW voltage	V_{ILS} 2)	– 0.5	0.4	– 0.5	0.4	– 0.5	0.4	V	
Input capacitance	C_{In}	–	10	–	10	–	10	pF	
Output capacitance	C_{Out}	–	10	–	10	–	10	pF	
Operating current	I_{CC}	–	600	–	630	–	650	mA	V_{CC} = 5.25V
Load capacitance	C_{Ld} 4)	–	75	–	50	–	50	pF	

1) V_{IL} min. = – 3.0 V for pulse width less than 15 ns
2) V_{IHS} and V_{ILS} apply to Clk2 × Sys, Clk2 × Smp, Clk2 × Rd, Clk2 × Phi, CpBusy, and Reset#.
3) V_{OHC} applies to Run# and Exception#.
4) Operation above the C_{Ld} maximum may impair the useful life of the device.

AC Characteristics

T_C = 0 to 90 °C; V_{CC} = 5 V ±5%

Notes: *All output timings are given assuming 25 pF of capacitive load. Output timings should be derated where appropriate as per the table below.*

All timings referenced to 1.5 V.

Parameter	Symbol	Limit values						Unit	Test condition
		16.67 MHz		20 MHz		25 MHz			
		min.	max.	min.	max.	min.	max.		
Clock Parameters[5]									
Input clock high	$t_{ClkHigh}$	12	–	10	–	8	–	ns	Trans. ≤ 5 ns
Input clock low	t_{ClkLow}	12	–	10	–	8	–	ns	Trans. ≤ 5 ns
Input clock period	t_{ClkP}	30	1000	25	1000	20	1000	ns	–
Clk2×Sys to Clk2×Smp		0	$t_{\frac{Cyc}{4}}$	0	$t_{\frac{Cyc}{4}}$	0	$t_{\frac{Cyc}{4}}$	ns	–
Clk2×Smp to Clk2×Rd		0	$t_{\frac{Cyc}{4}}$	0	$t_{\frac{Cyc}{4}}$	0	$t_{\frac{Cyc}{4}}$	ns	–
Clk2×Smp to Clk2×Phi		9	$t_{\frac{Cyc}{4}}$	7	$t_{\frac{Cyc}{4}}$	5	$t_{\frac{Cyc}{4}}$	ns	–
Run Operation Parameters									
Data enable	t_{DEn}	–1	–2	–1	–2	– 0.5	–1.5	ns	–
Data disable	t_{DDis}	0	–1	0	–1	0	–0.5	ns	–
Data valid	t_{DVal}	–	3	–	3	–	3	ns	25 pF Load
Write delay	t_{WrDly}	0	5	0	4	0	3	ns	25 pF Load
Data setup	t_{DS}	9	–	8	–	7	–	ns	–
Data hold	t_{DH}	–2.5	–	–2.5	–	–2.5	–	ns	–
CpBusy setup	t_{CBS}	13	–	11	–	9	–	ns	–
CpBusy hold	t_{CBH}	–2.5	–	–2.5	–	–2.5	–	ns	–
Access type(1:0)	t_{AcTy}	1	7	–	6	1	5	ns	25 pF Load
Access type(2)	t_{AT2}	1	17	–	14	1	12	ns	25 pF Load
Memory write	t_{MWr}	1	27	–	23	1	18	ns	25 pF Load
Exception	t_{Exc}	1	7	–	7	1	5	ns	25 pF Load
Int Setup	t_{IntS}	9	–	8	–	7	–	ns	25 pF Load
Int Hold	t_{IntH}	–2.5	–	–2.5	–	–2.5	–	ns	25 pF Load

5) The clock parameters apply to all four 2xClocks: Clk2×Sys, Clk2×Smp, CLK2×Rd, and Clk2×Phi.

AC Characteristics (cont'd)

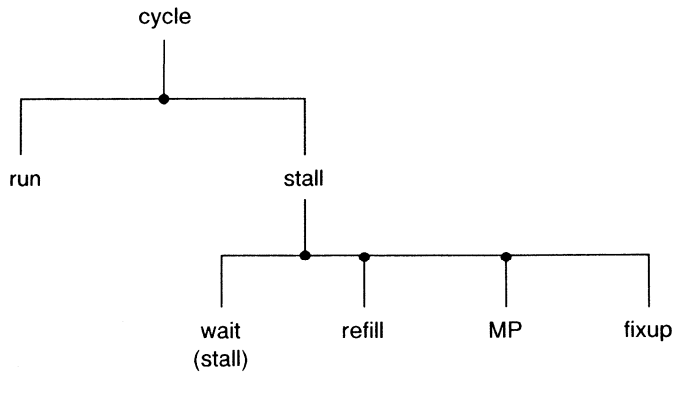
Parameter	Symbol	Limit values						Unit	Test condition
		16.67 MHz		20 MHz		25 MHz			
		min.	max.	min.	max.	min.	max.		
Capacitive Load Deration									
Load derate	C_{LD}	0.3	2	0.5	1	0.5	1	ns/ 25pF	–
Stall Operation Parameters									
Address valid	t_{SAVal}	–	30	–	23	–	20	ns	25 pF Load
Access type	t_{SAcTy}	–	27	–	23	–	18	ns	25 pF Load
Memory read initiate	t_{MRdI}	0	27	–	23	0	18	ns	25 pF Load
Memory read terminate	t_{MRdT}	1	7	–	7	1	5	ns	25 pF Load
Run terminate	t_{StI}	3	17	–	15	3	10	ns	25 pF Load
Run initiate	t_{Run}	0	7	–	6	0	4	ns	25 pF Load
Memory write	t_{SMWr}	3	27	–	23	3	18	ns	25 pF Load
Exception valid	t_{SExc}	3	20	–	18	3	15	ns	25 pF Load

As described earlier the SAB-R3000 supports interfaces to external cache, main memory and coprocessors. This section describes the timing parameters and operation of the important cases for each of these interfaces along with Interrupt, Reset and Multi-processing examples.

Operation Fundamentals

A "cycle" is the basic instruction processing unit of the SAB-R3000 processor. Cycles in which forward progress is made, i.e. an instruction is retired, are called "run" cycles. An instruction is retired either by its completion or in the presence of an exception its abortion. Cycles in which no forward progress is made are called "stall" cycles. Stall cycles are used for resolving urgent situations such as cache misses on loads, write system busy during stores, and coprocessor interlocks. All cycles can be classified as either run cycles or stall cycles. There are four types of stall cycles: "wait" stall cycles - simply known as stall cycles; "refill" stall cycles - which occur only during main memory reads; "multi-processor" (MP) stall cycles - allow the memory system to read or invalidate specific data cache entries; and "fixup" stall cycles - occur during the final cycle of the stall and are used in general to fix up the conditions which caused the stall. Processor transactions which occur during the first half of the cycle are called phase 1 transactions while those which occur during the second half of the cycle are called phase 2 transactions. Figure 39 summarizes the cycles in the SAB-R3000.

Figure 39
SAB-R3000 Cycles



As described earlier coprocessors maintain synchronization with the SAB-R3000 by monitoring the signals Run# and Exception#. Run# is asserted by the SAB-R3000 during run cycles and deasserted during stall cycles. When Run# is deasserted during the n th cycle, the coprocessor(s) disregard(s) the instruction-data pair presented during the n-1 th cycle. When Run# is reasserted during the m th cycle, the coprocessor(s) take(s), as a replacement for the instruction-data pair which was disregarded, the instruction-data pair presented during the m-1 th cycle – which was the final fixup cycle for whatever stall sequence was occurring.

Exception# is used by the coprocessor(s) to track exception related information during run cycles and stall related information during stall cycles.

- During phase 1 of run cycles Exception# indicates whether an exception has occurred for the instruction which is currently in its "writeback" pipestage. Unless the exception is occurring as a result of an interrupt request by the coprocessor, the assertion of Exception# prevents any state from being committed in the coprocessor.
- During phase 2 of run cycles Exception# indicates whether an interrupt request is being granted for the instruction which is currently in its "memory access" pipestage.
- During phase 1 of stall cycles Exception# indicates whether the current stall cycle is a fixup cycle. When a fixup cycle is occurring, it is guaranteed that the data present on the Data bus is valid. The coprocessor uses the fixup indication to qualify the use of data sampled from the bus during the stall.
- During phase 2 of stall cycles Exception# indicates whether the current stall is a Coprocessor Busy Stall.

The use of the Exception# signal is summarised below.

Table 8

	phase 1	phase 2
Run	Exc1W#	IntGr2M#
Stall	Fixup1#	CpBusy2#

Processor Input Clocks

The SAB-R3000 has four separate double frequency (i.e. in a 25 MHz system these clocks are 50 MHz) input clocks. They can be adjusted to obtain optimum positioning of cache interface signals. The absolute timing of these input clocks with respect to the processor outputs is undefined, only the differences are important. A short description of these four clocks follows.

- Clk2×Sys:
  is the master clock and must lead all others. It determines the position of SysOut* (the processors output clock) with respect to Data, Tag and Address buses.
- Clk2×Smp:
  determines the sample point for data coming into the processor on all processor inputs except those coming directly from coprocessors.
- Clk2×Rd:
  controls output enable time and provides sufficient address access to sample address hold from end of write, and data hold from end of write.
- Clk2×Phi:
  determines the position of the internal phases, phase 1 and phase 2.

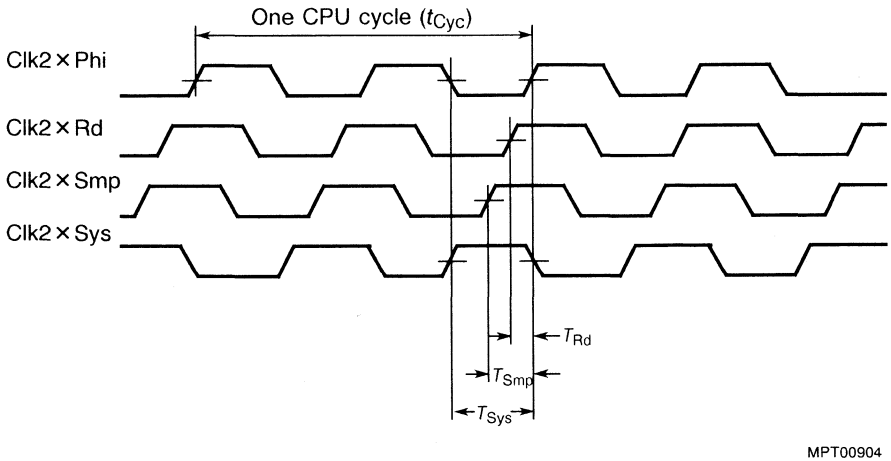
Table 9 illustrates the 2×Clock dependency of the processors timing controlled outputs. Outputs are referenced only to "rising edges" of the 2×Clocks. The assertion dependency is indicated by ↑ and deassertion by ↓.

Table 9

	Clk2×Sys	Clk2×Smp	Clk2×Rd	Clk2×Phi
IClk,DClk		↓	↑	
IRd#, DRd#, XEn#	↓		↑	
IWr#, DWr#		↑ ↓		
SysOut#	↑ ↓			
Data, Tag			↓	↑
Address				↑ ↓
All Others				↑ ↓

Figure 40 shows the four 2× input clocks.

Figure 40
2x Input Clocks



In the timing diagrams which follow, timing specifications are given relative to a shifted version of the processor output clock SysOut#. The clock is called PhiOut# and is a virtual clock, i.e. the processor does not actually produce this output. It is shown in the timing diagrams for reasons of clarity, because its period is synchronous with a machine cycle. The shift amount is equal to the difference between Clk2×Sys to Clk2×Phi and, as is shown in figure 40, is T_{Sys}. Also, in the timing diagrams Clk2×Sys and SysOut# are shown to clarify the relationship between these signals.

In reality SysOut# is produced rather than PhiOut# since this provides a signal with timing appropriate for synchronizing system transactions to the processor. Timings are given relative to PhiOut# since this makes determining the position of the input clocks the most straightforward. The timing of any output with respect to SysOut# can be determined from its timing with respect to PhiOut# by adding T_{Sys}.

Timing Diagram Notation

The following timing diagrams describe various transactions of the processor. Table 10 illustrates the notational conventions used in these diagrams.

Table 10

Character	Meaning
I	Instruction
D	Data
#	Active low
%	An incorrect datum
!	An unused datum
Z	The high impendance state
Ad	Address
in	into processor
out	out of processor
//////	not valid or Don't Care
c D	cache Data
p D	processor Data

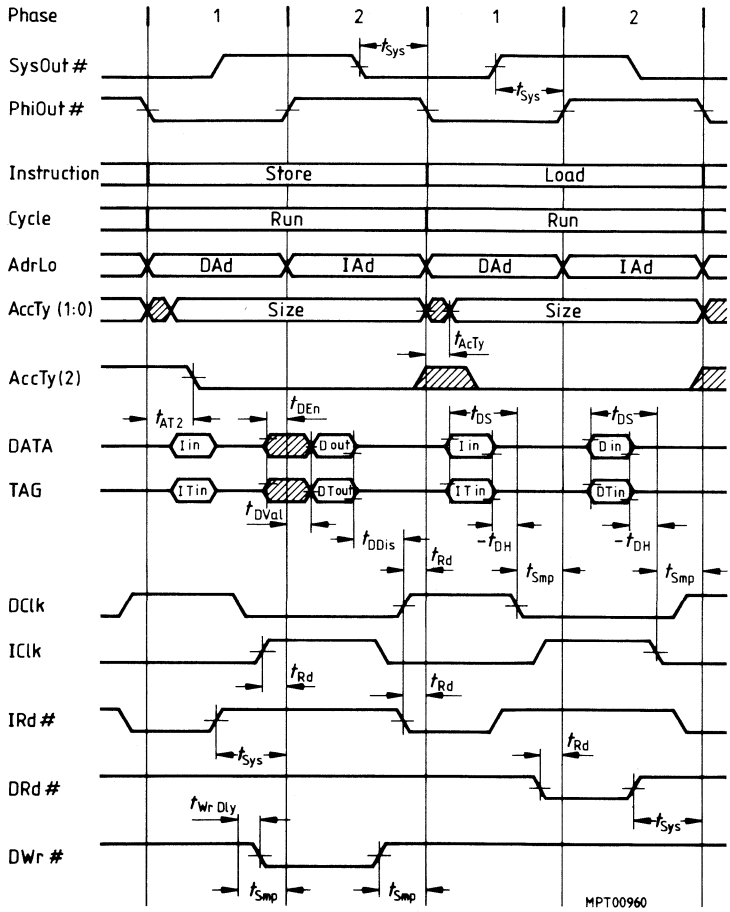
Cache Timing

Cache operation was explained in the Interface section. Figure 41 illustrates cache operation and timing. During run cycles the Access Type bus, AccTy(2:0), indicates whether or not a phase 2 transaction is scheduled for that cycle and the size of the datum being transferred. Table 11, below, summarizes AccTy encoding during run cycles.

Table 11

AccTy(2)	AccTy(1:0)	size
1	XX	no transaction
0	00	byte
0	01	half word
0	10	tribyte
0	11	word

Figure 41
Cache Timing



Main Memory Reads

When a LOAD misses in the cache, a main memory read is initiated. Main memory reads are supported by read busy stalls and the MemRd#, RdBusy and CpCond0 signals. Two types of main memory reads are supported, single word transfers and multiple word transfers. Multiple word transfer or block refill is further divided into two cases, Refill and Streaming. Block size for block refill can be varied from 4, 8, 16 or 32 words. Refill occurs while the processor is in the non run state, i.e. instructions are not being executed. Refill is supported for both instructions and data. Streaming is only for instructions and occurs while the processor is in the run state, i.e. the processor executes instructions directly from the instruction stream coming from memory.

Selection between single and multiple word transfers is determined by the CpCond0 input (i.e. asserted – multiple word transfer, deasserted – single word transfer). When an uncached reference occurs CpCond0 is ignored and the reference is forced to be a single word transfer.

Note : CpCond0 is used for transfer size selection only during read busy stalls.

Table 12 summarizes the meaning of the AccTy(2:0) bus during main memory reads.

Table 12

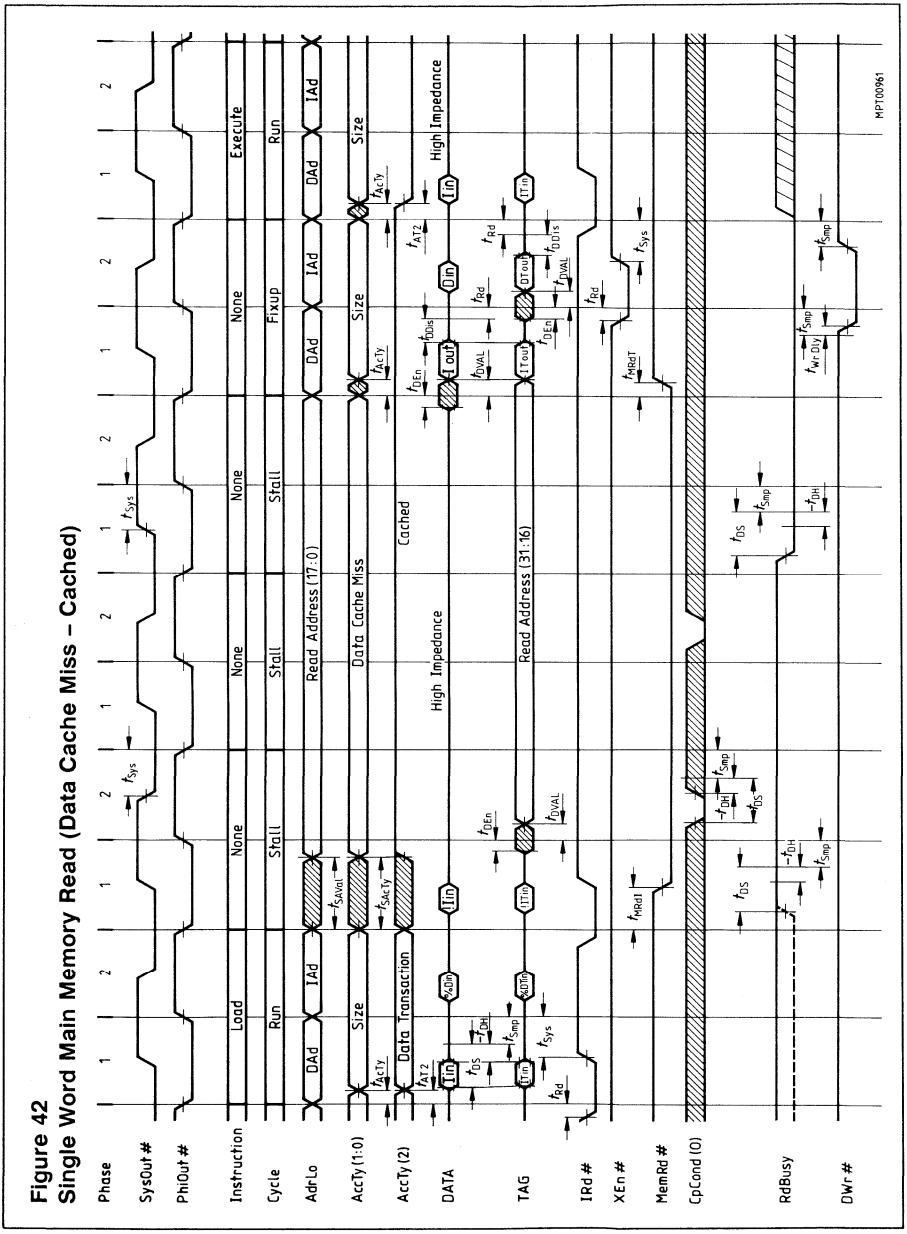
AccTy(2)	AccTy(1:0)	size	type
0	00	byte	uncached/unkown
0	01	half word	uncached/unkown
0	10	tribyte	uncached/unkown
0	11	word	uncached/unkown
1	X0	word	cached/data
1	X1	word	cached/instruction

Figure 42 illustrates a single word transfer. Entry into the stall is indicated by the assertion of MemRd# which occurs in the cycle following the one in which the LOAD missed. During the stall the SAB-R3000 presents the read address on the AdrLo and Tag buses and tristates the Data bus. This state is maintained until RdBusy is deasserted. RdBusy is deasserted during phase 1 of the cycle in which the memory system will provide Data and Data parity to the processors read buffer. This is the termination of a read busy stall cycle. The read buffer is the register which couples the main memory bus to the cache bus during main memory reads.

The cycle following that in which RdBusy is deasserted is the fixup cycle. During this cycle the appropriate cache is written (in this case the Data cache) with the data returned by main memory. This is performed by reading the Read Buffer (XEn#) and writing to the Data cache at the same time (DWr#). Simultaneously, Tag and Tag parity are also written to the cache.

Note : The cache write does not occur if the stall was due to an uncached reference. The processor resumes run operation on the cycle following the fixup cycle if no other stall is pending.

Figure 42
Single Word Main Memory Read (Data Cache Miss – Cached)



Main Memory Writes

Main memory writes are accomplished through a write buffer which accepts writes at cache speeds. A write is indicated to the Write Buffer by the assertion of the MemWr# signal. If the write buffer becomes full and a further write is attempted it causes a write busy stall, this is illustrated in figure 43. The first write shown fills the write buffer causing it to assert WrBusy#. The write which is attempted in the next cycle is not accepted by the write buffer and is redone by the processor during the fixup cycle. The write busy stall is terminated when the write buffer deasserts WrBusy# to indicate it can accept another write. The cycle following its deassertion will be the fixup cycle.

Interrupts

The SAB-R3000 has 6 general purpose interrupt inputs which are sampled during phase 2 of all run and fixup cycles. After causing an interrupt exception to occur, the interrupts continue to be sampled during each phase 2 to provide a level sensitive indication of the active interrupt(s). Figure 44 shows the Interrupt Timing.

Reset Timing

The Reset# input is used to force processor execution starting at the reset exception vector and to initialize processor state. Its operation is explained in the Resetting the SAB-R3000 section. Figure 45 illustrates its timing parameters.

Coprocessor Timing

During run cycles, the operation and timing of coprocessor LOADS and STORES is identical to that of the main processor. This can be seen in figure 46. To provide synchronization when required, the SAB-R3000 supports coprocessor busy stalls. The operation of such a stall is also illustrated in figure 46. The coprocessor must assert CpBusy during the "ALU" cycle of the coprocessor instruction to initiate such a stall. To terminate the stall CpBusy must be deasserted during phase 1. The cycle following this deassertion is the fixup cycle.

Multi-Processor Timing

The multi-processing interface and operation was described in the Multi-processing Support section. A MP stall cycle is illustrated in figures 47 and 48. In this example CpCond2 (MP_Invalidate) is asserted for two consecutive cycles causing two DWr#'s to occur. Further cases are covered in detail in the Processor Interface Specification (MIPS Computer Systems).

Figure 43
Main Memory Write –With Write Busy Stall

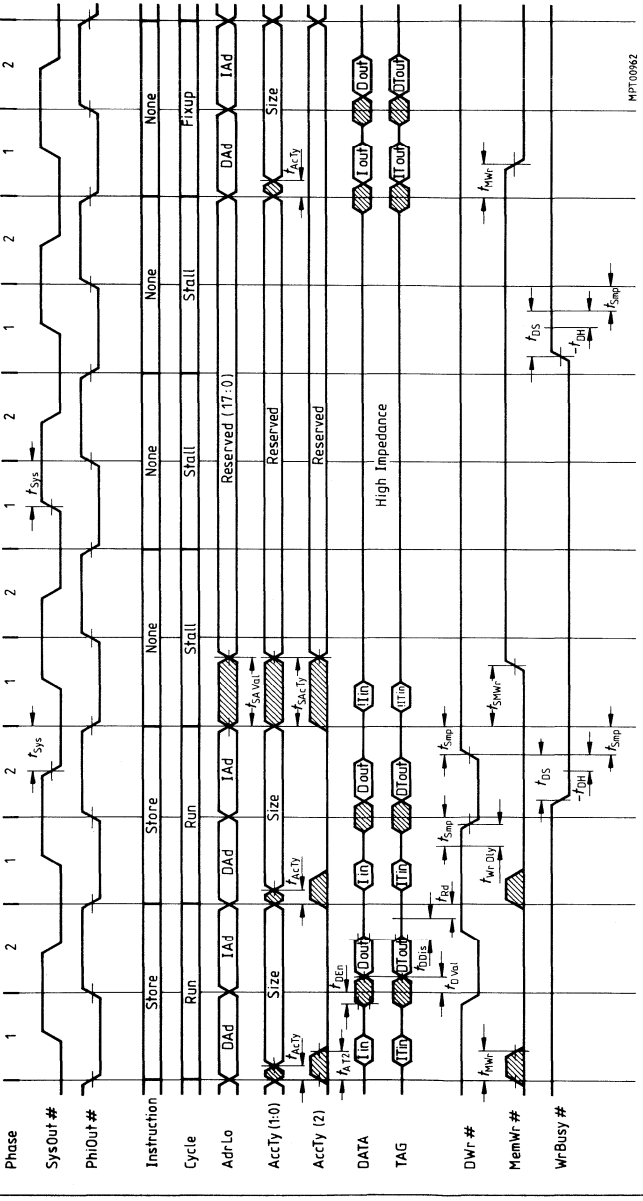


Figure 44
Interrupt Timing

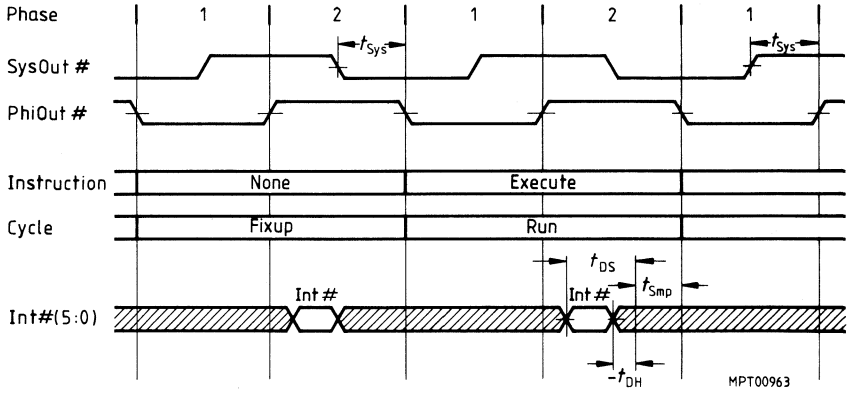


Figure 45
Reset, W, X, Y, and Z Mode Select Timing

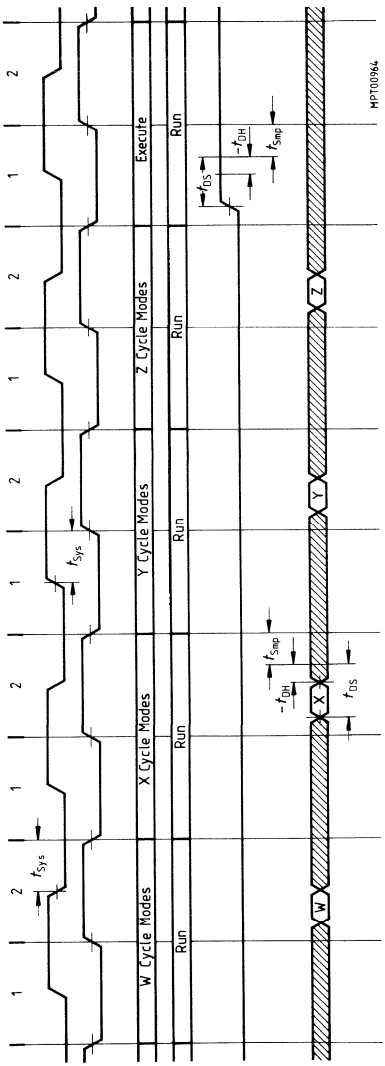
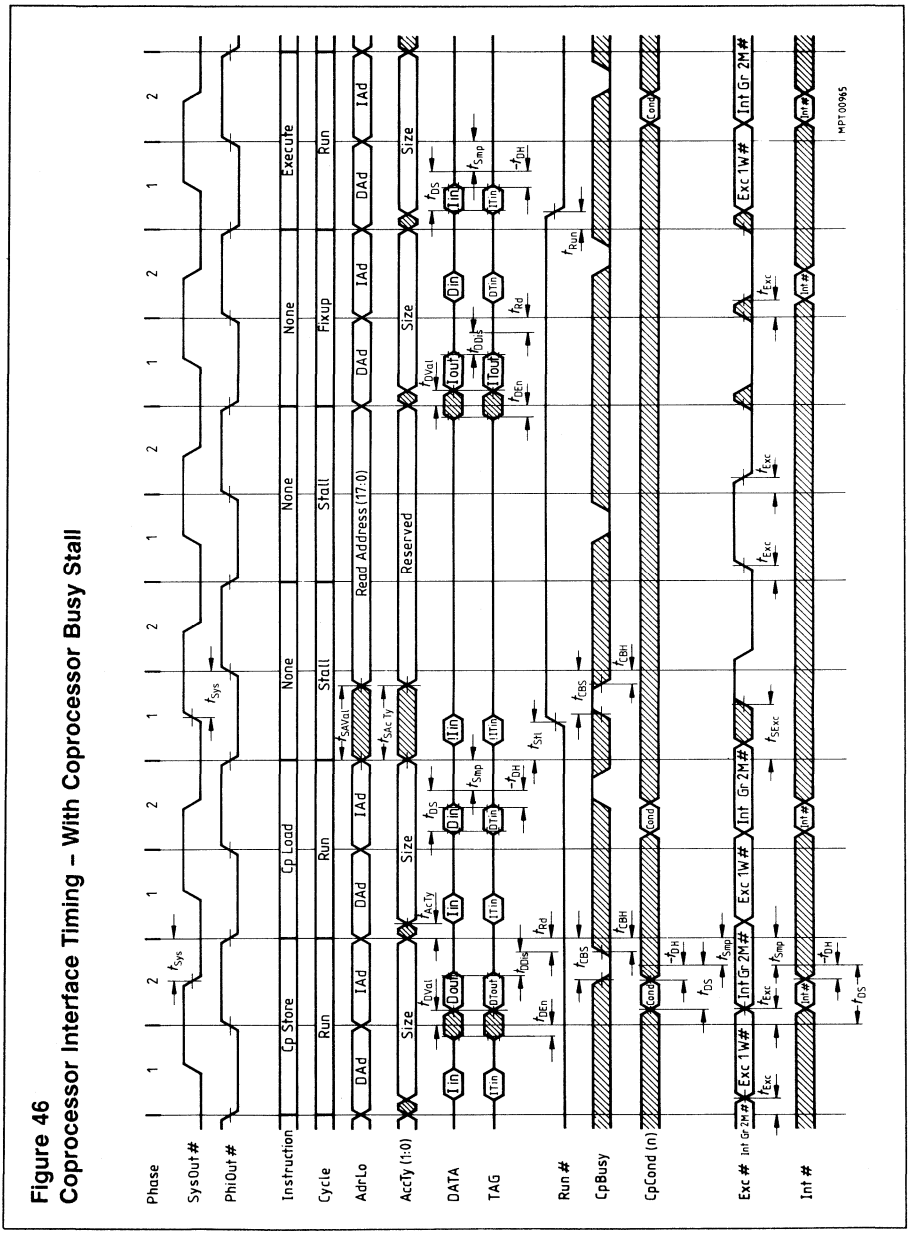


Figure 46
Coprocessor Interface Timing – With Coprocessor Busy Stall



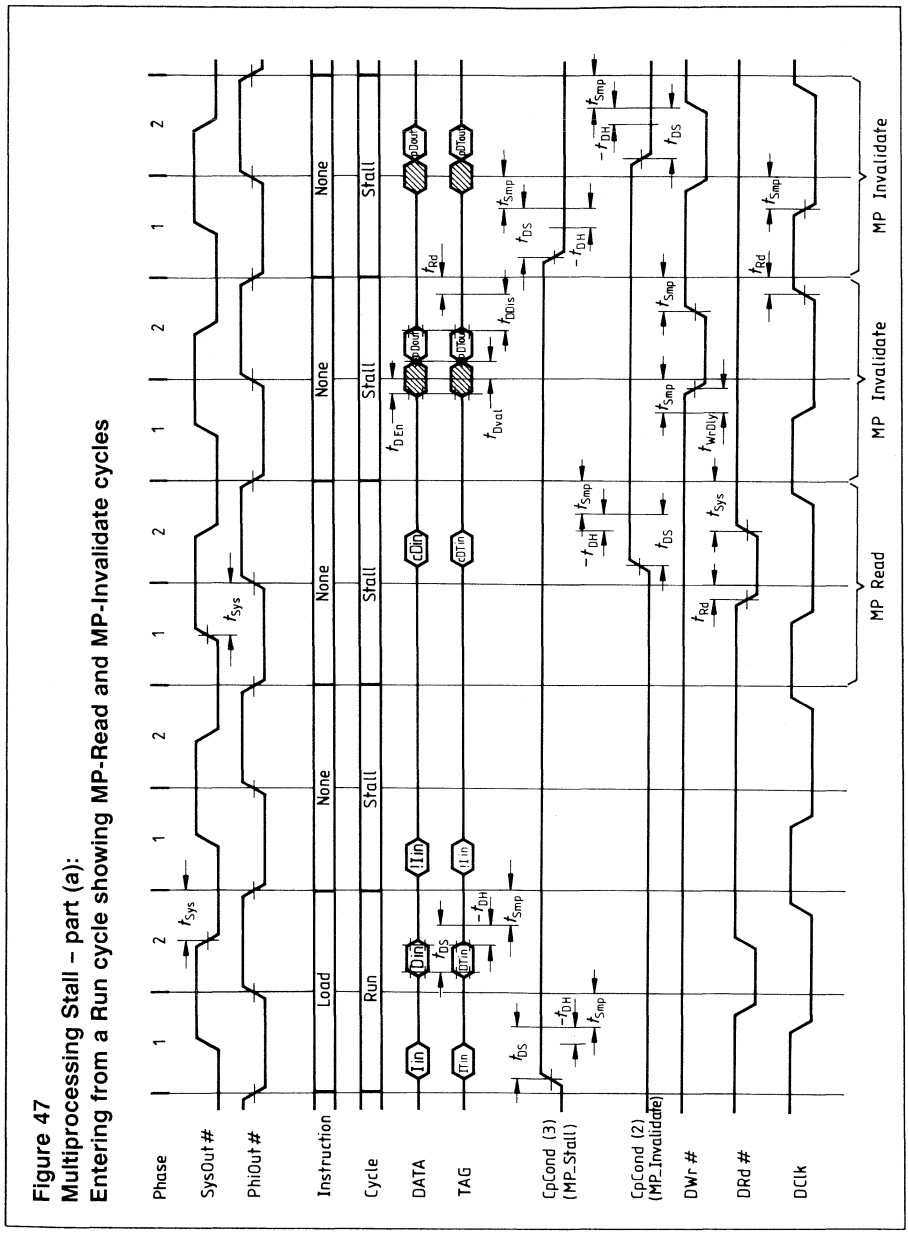
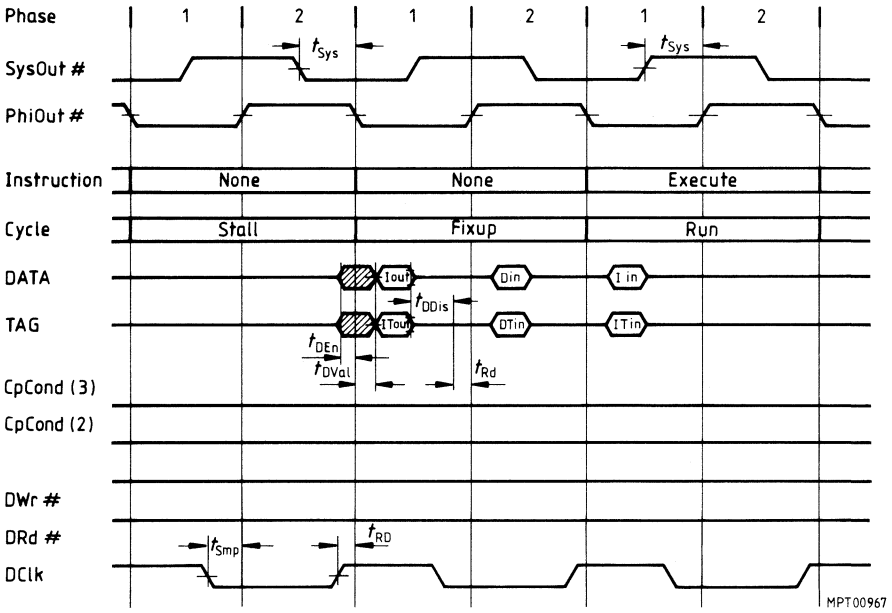


Figure 47
Multiprocessing Stall – part (a):
Entering from a Run cycle showing MP-Read and MP-Invalidate cycles

Figure 48
Multiprocessor Stall – part (b):Leaving to a Run cycle via a Fixup cycle



SIEMENS

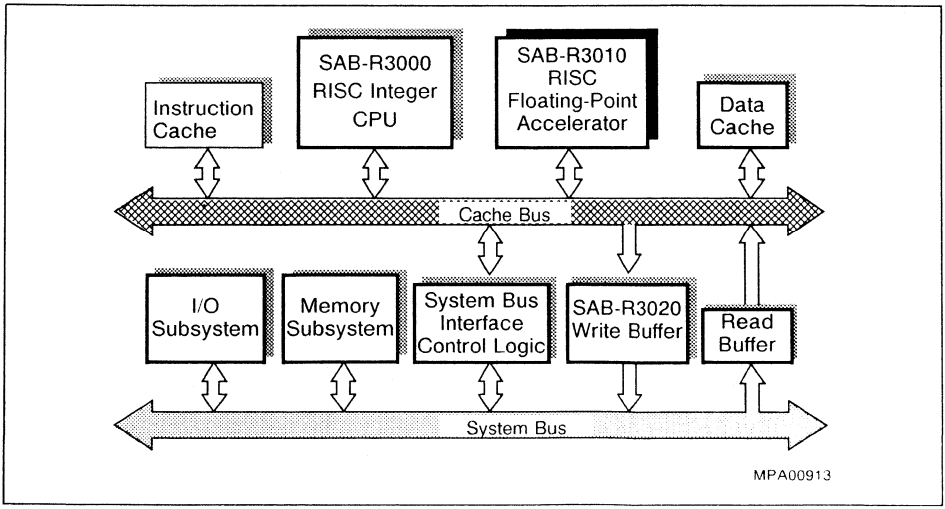
High-Performance Floating-Point Coprocessor

SAB-R3010

based on Advanced RISC Architecture with four Independent Arithmetic Functional Units

Advance Information

- Fully conforms to ANSI/IEEE standard 754-1985 for binary floating-point arithmetic
- Load/Store instruction set
- Full 64-bit operation
  - sixteen 64-bit floating-point registers
- Seamless coprocessor interface to SAB-R3000
- Transparent addition of floating-point extensions to the SAB-R3000's instruction set
- Four independent functional units
  - Register, Add, Divide and Multiply units
  - allows up to four floating-point instructions to be executed in parallel
- Software-, pin- and functionally compatible with all R2010A coprocessors
- Fully compatible to all R3010 processors of other manufacturers
- Ceramic package: CL-CC-84



Ordering Information

Type	Ordering code	Package	Function
SAB-R3010-16-QJ	Q67120-C555	CL-CC-84	32/64-bit Floating-Point Coprocessor, 16.67 MHz
SAB-R3010-20-QJ	Q67120-C565	CL-CC-84	32/64-bit Floating-Point Coprocessor, 20 MHz
SAB-R3010-25-QJ	Q67120-C497	CL-CC-84	32/64-bit Floating-Point Coprocessor, 25 MHz

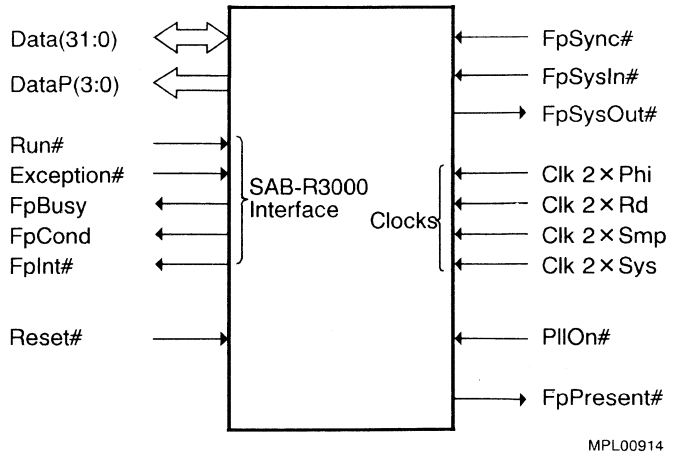
Introduction

The SAB-R3010 is a high performance Floating-Point Accelerator (FPA) which is implemented as a full-custom VLSI CMOS chip. It serves as a coprocessor to the SAB-R3000 RISC microprocessor. It transparently extends the SAB-R3000's instruction set to perform floating-point operations, by cointerpreting the common instruction stream. The FPA, with associated system software, fully conforms to the requirements of ANSI/IEEE standard 754-1985 "IEEE Standard for Binary Floating-Point Arithmetic". In addition the SAB-R3010 fully supports the standard recommendations. The SAB-R3010's architecture is organised as follows – hardware directly implements the essential floating-point operations of addition, subtraction, division, multiplication, comparison, conversion between formats, absolute value and negation. These operations are highly optimized. System software supplies the more complex functions and while doing so benefits from the underlying fast arithmetic hardware. Figure 1 illustrates the SAB-R3010 Logic symbol.

Pin Names

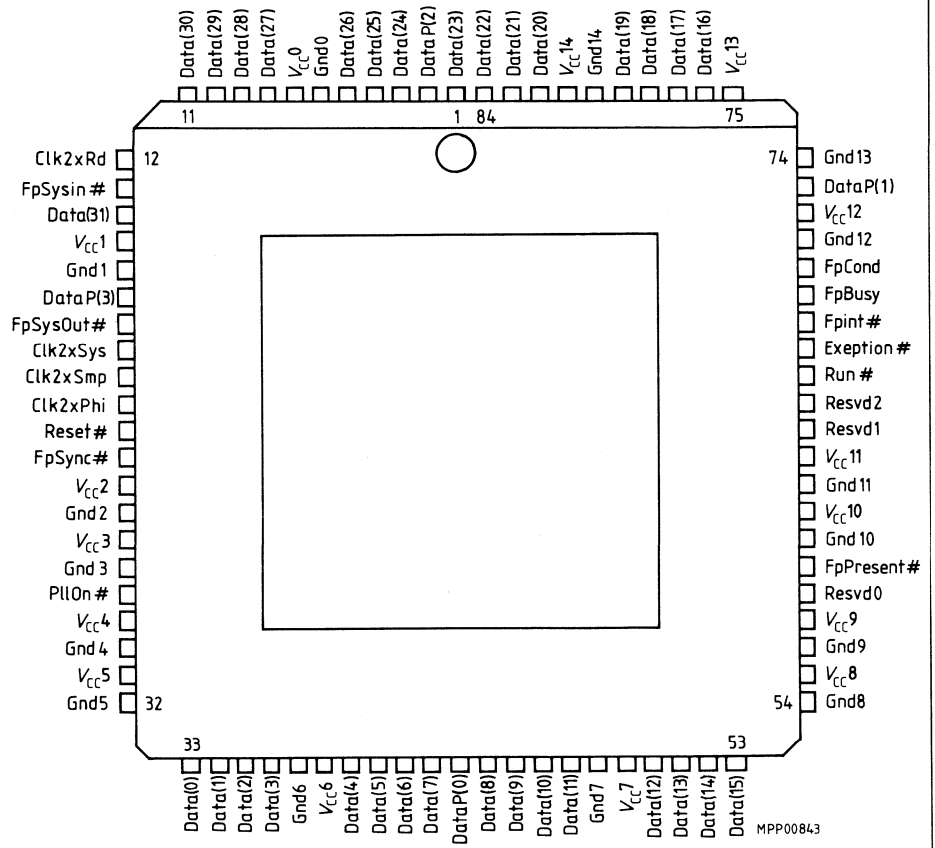
Data(31:0)	Data Bus
DataP(3:0)	Even parity for Data Bus
Run#	System in Run or Stall state
Exception#	Exception related information
FpBusy	Floating-point busy stall
FpCond	Floating-point condition
FpInt#	Floating-point Interrupt
Reset#	Synchronous Initialization
FpSync#	Floating-point Synchronize
FpSysIn#	Floating-point System clock in
FpSysOut#	Floating-point System clock out
PllOn#	Phase Lock Loop On
FpPresent#	Floating-point present

Figure 1
Logic Symbol



Pin Configurations

Figure 2
CL-CC-84 Cavity Up (Top View)



Pin Definitions and Functions

Symbol	Pin Number	Input (I) Output (O)	Function
Data(31:0)	14,11,10,9,8,5,4, 3,1,84,83,82,79, 78,77,76,53,52, 51,50,47,46,45, 44,42,41,40,39, 36,35,34,33	I/O	A multiplexed 32-bit bus used for instruction and data transfers on phases 1 and 2, respectively.
DataP(3:0)	17,2,73,43	O	A 4-bit bus containing even parity over the data bus. Parity is generated by the FPC on stores.
Run#	66	I	Input to the FPC which indicates whether the processor-coprocessor system is in the run or stall state.
Exception#	67	I	Input to the FPC which indicates exception related status information.
FpBusy	69	O	Signal to the CPU indicating a request for a coprocessor busy stall.
FpCond	70	O	Signal to the CPU indicating the result of the last comparison operation.
FpInt#	68	O	Signal to the CPU indicating that a floating-point exception has occurred for the current FPC instruction.
Reset#	22	I	Synchronous initialization input used to distinguish the processor-FPC synchronization period from the execution period. Reset# must be synchronized by the leading edge of SysOut from the CPU.
PllOn#	28	I	Input which during the reset period determines whether the phase lock mechanism is enabled, and during the execution period determines the output timing model.

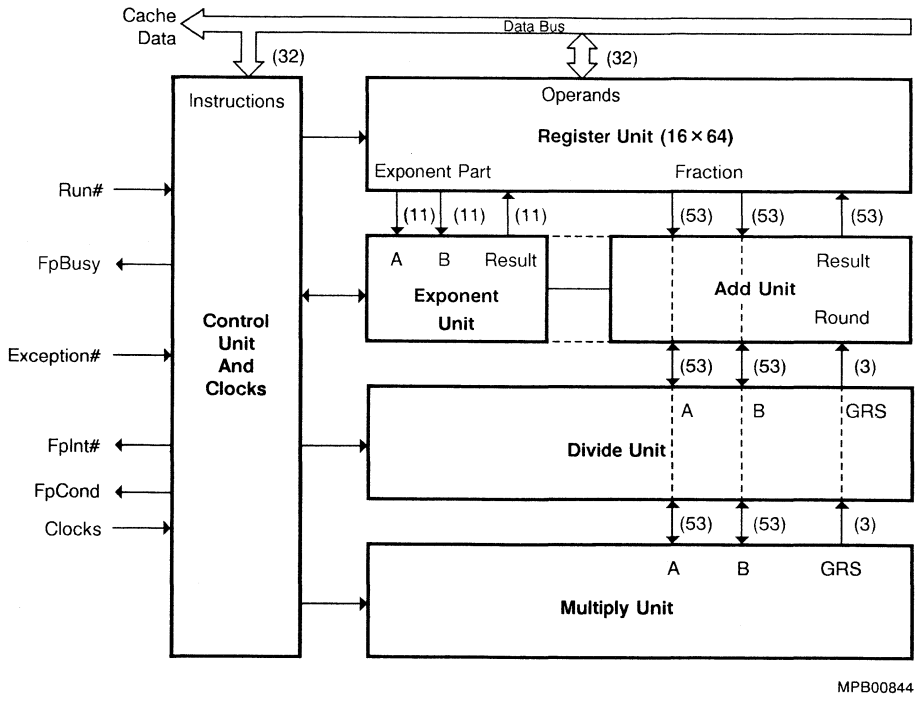
Pin Definitions and Functions (cont'd)

Symbol	Pin Number	Input (I) Output (O)	Function
FpPresent#	59	O	Output which is pulled to ground through an impedance of approximately 0.5 kΩ. By providing an external pull-up on this line an indication of the presence or absence of the FPC can be obtained.
Clk2 × Sys	19	I	A double frequency clock input used for generating FpSysOut#.
Clk2 × Smp	20	I	A double frequency clock input used to determine the sample point for data coming into the FPC.
Clk2 × Rd	12	I	A double frequency clock input used to determine the disable point for the data drivers.
Clk2 × Phi	21	I	A double frequency clock input used to determine the position of the internal phases 1 and 2.
FpSysOut#	18	O	Synchronization clock from the FPC.
FpSysIn#	13	I	Input used to receive the synchronization clock from the FPC.
FpSync#	23	I	Input used to receive the synchronization clock from the CPU.
GND14-1	80,74,71,62,60, 56,54,48,37,32, 30,27,25,16,6		Ground
V_{CC}14-1	81,75,72,63,61, 57,55,49,38,31, 29,26,24,15,7		Power Supply (+ 5 V)
Resvd2-0	65,64,58		Reserved

Functional Description

The SAB-R3010 contains four independent arithmetic functional units (Register, Add, Divide and Multiply) which interact with a scheduling and managing control unit. Figure 3 shows the block diagram of the SAB-R3010.

Figure 3
Functional Block Diagram



Note: A hash sign # indicates an Active-LOW Signal

Basic Architecture

As figure 3 shows, the SAB-R3010 consists of five main units.

The **Control Unit** continually monitors the transactions between the SAB-R3000 (with which the SAB-R3010 shares the data bus) and the instruction cache (i.e. the instruction stream). If an instruction does not apply to the SAB-R3010, it ignores it. When an instruction does apply, it interprets it. Synchronization between the coprocessor and the main processor is also managed by the control unit. The control unit monitors the signals Run# and Exception# to see what state the SAB-R3000 is in. Run# is used to track pipeline disruptions due to non-exceptional events (i.e. CPU stalls) such as cache misses, write busy etc. Exception# is used to track pipeline disruptions due to exceptional events such as virtual to physical address translation misses, interrupts etc. When either of these cases occurs, the SAB-R3010's pipeline is shut down (stalled), in such away that unfinished instructions can be restarted later without numerical inconsistencies. Whenever the SAB-R3000 requires the result of a floating-point instruction which is not yet completed, the control unit signals the CPU to wait through the assertion of the FpBusy signal. The control unit also schedules the execution of each instruction with the four arithmetic units.

The **Register Unit's** register file can perform two 64-bit operand reads, one 64-bit write result and one memory load/data write in one cycle. This implies that four ports exist. Physically, a two-port design, which is accessed twice per cycle, implements the register file.

The **Add Unit** executes add, subtract, convert, compare, negate and absolute value instructions as well as the final IEEE rounding step of multiply and divide operations. The exponent data path (exponent unit) is included in this unit and it computes the 8-bits (single-precision) or 11-bits (double-precision) of exponents for all arithmetic operations.

The **Divide Unit** uses a radix-4, SRT-division algorithm to produce four quotient bits per cycle. This method uses a redundant encoding of the quotient as a sum of digits with values 2, 1, 0, -1 and -2. A double-precision divide requires a total of 19 cycles (12 cycles for a single-precision divide).

The **Multiply Unit** computes the product of the mantissa portions of its operands (refer to the Data Formats section). In the case of double-precision, the multiplier computes the product of two 53-bit operands in less than four cycles. It retains the most significant 56-bits of the 106-bit product.

Results of both the multiplier and divider are returned to the add unit over the two operand buses (A and B) for final carry propagation and rounding. A separate path exists for the guard, round and sticky bits (GRS) required for IEEE rounding. The interaction of the five units and the width of the major data buses can be seen in figure 3.

The autonomy of the four arithmetic units enables them to run in parallel. Concurrently executing instructions generally do not conflict for resources – except at the beginning and end cycles of each operation, mainly due to simultaneous requests for the add unit (see the Pipeline Architecture section). Based on the latency of each operation, the control unit schedules instructions to ensure that no two will need, for example, the exponent unit or rounding function of the add unit at the same time. The floating-point architecture requires that instructions must appear to complete in the order they were issued. However the control unit recognizes the special cases of operations with exceptional results, conflicts for one arithmetic unit and data dependencies between operations – therefore it will reschedule instructions for maximum pipeline efficiency. In such instances it adjusts the issue schedule to maintain the illusion of in-order instruction completion. Refer to the Pipeline Architecture section for more details.

Pipeline Architecture

The SAB-R3010 has an instruction pipeline which mirrors that of the SAB-R3000 processor. However, there is a difference: the FPA (SAB-R3010) has a 6-stage pipeline in contrast to the 5-stage pipeline of the SAB-R3000. It uses an extra pipestage to provide efficient coordination of exception responses between the FPA and the CPU (SAB-R3000).The six stages of the SAB-R3010 pipeline are:

(1) IF Instruction Fetch:
The CPU calculates the instruction address required to read an instruction from the instruction cache. The instruction address is generated and output during phase 2 of this pipestage. No action is required by the SAB-R3010 during this pipestage since the main processor is responsible for address generation. Note that the instruction is not actually read into the processor until the beginning of the RD pipestage. Refer to figure 4.

(2) RD Register Fetch/Instruction Decode:
The instruction is present on the Data bus during phase 1 of this pipestage and the FPA decodes the data on the bus to determine whether it is an instruction for the FPA. The FPA reads any required operands from its registers (RF in figure 4) while decoding the instruction.

(3) ALU ALU Operation:
If an instruction is one for the FPA, execution commences during this pipestage. If the instruction causes an exception, the FPA notifies the SAB-R3000 of the exception during this pipestage by asserting the FpInt# signal. If the SAB-R3010 determines that it requires additional time (i.e. more than 1 cycle) to complete this instruction, it initiates a stall during this pipestage.

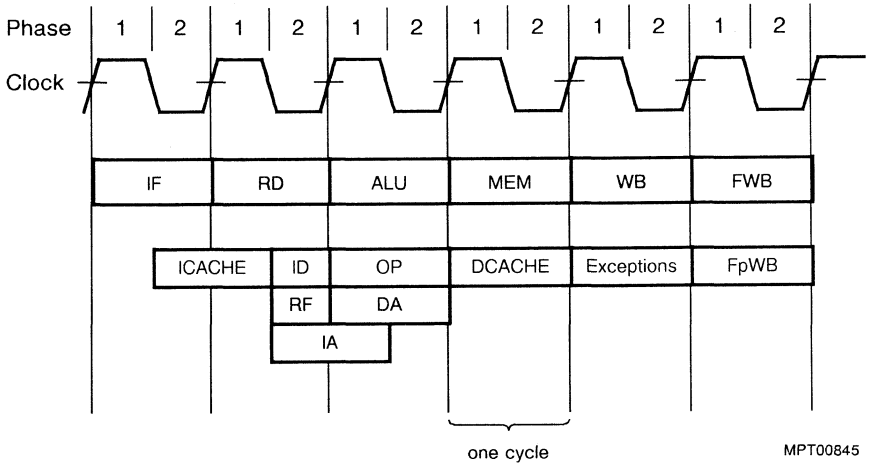
(4) MEM Memory Access:
If it is a coprocessor Load or Store instruction, the FPA presents or captures the data during phase 2 of this pipestage. If an interrupt is taken by the main processor, it notifies the SAB-R3010 during phase 2 of this pipestage (via the Exception# signal).

(5) WB Write back:
If the instruction that is currently in the write back (WB) stage caused an exception, the main processor notifies the FPA by asserting the Exception# signal during this pipestage. Thus, the FPA uses this pipestage solely to deal with exceptions.

(6) FWB Floating-Point Write back:
The SAB-R3010 uses this pipestage to write back ALU results to its register file. This stage is the equivalent of the WB stage in the SAB-R3000 pipeline.

Figure 4 illustrates the 6 stages of the SAB-R3010 pipeline. Each step requires approximately one machine cycle.

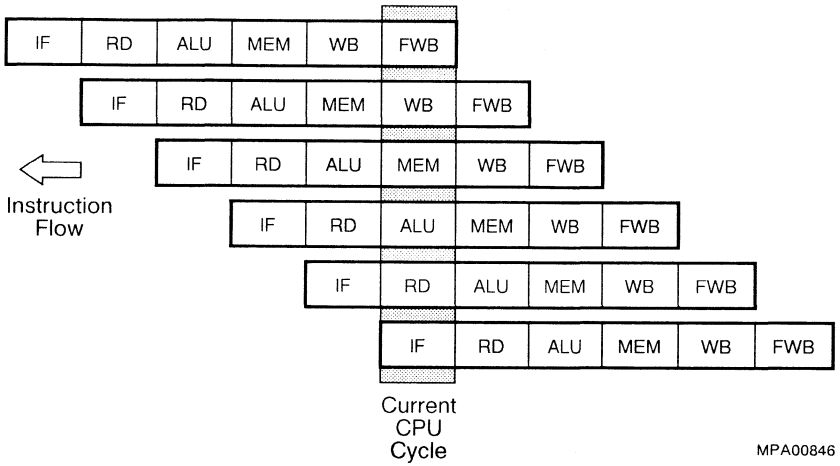
Figure 4
SAB-R3010 Instruction Execution Sequence



ICACHE	:	Instruction cache access
ID	:	Instruction decode
RF	:	Register operand fetch
IA	:	Instruction address calculation and translation
OP	:	Operation
DA	:	Data address calculation and translation
DCACHE	:	Data cache access
FpWB	:	Floating-point write back to register file
FpWB	:	Floating-point WB

The executions of six instructions are overlapping as shown in figure 5.

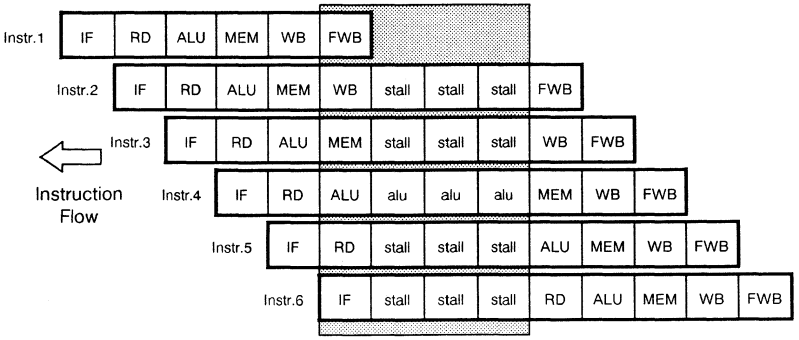
Figure 5
R3010 FPA Instruction Pipeline



This is a simplified view of the overlapped instruction execution of the SAB-R3010 because the figure assumes that each instruction can be completed in a single cycle. Most FPA instructions, however, require more than one cycle to be completed. Therefore, the pipeline must be stalled whenever register or resource conflicts occur. Figure 6 illustrates the effect of a three-cycle stall on the SAB-R3010 pipeline.

To alleviate the performance impact that would result from frequently stalling the pipeline, the SAB-R3010 overlaps instructions so that instruction execution can proceed so long as there are no resource conflicts, data dependencies or exceptional conditions.

Figure 6
An FPA Pipeline Stall



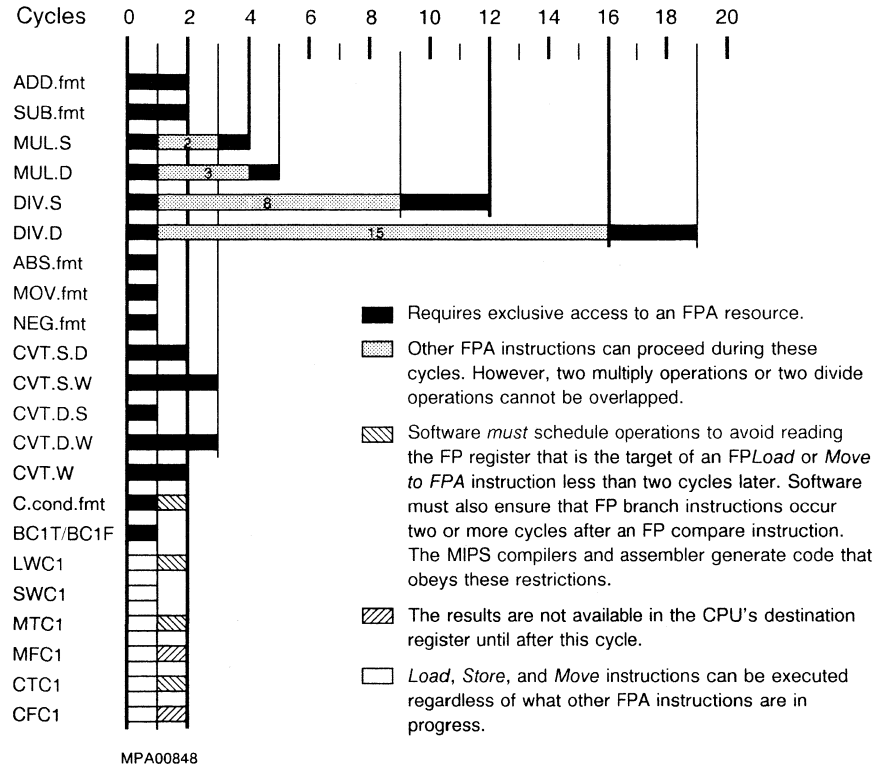
Stall initiated by Instr.4
during its ALU pipe stage.

MPA00847

As mentioned earlier the majority of SAB-R3010 instructions require more than one cycle to be completed. Figure 7 shows the number of cycles required to execute each of the FPA instructions, which varies from 1 to 19 cycles.

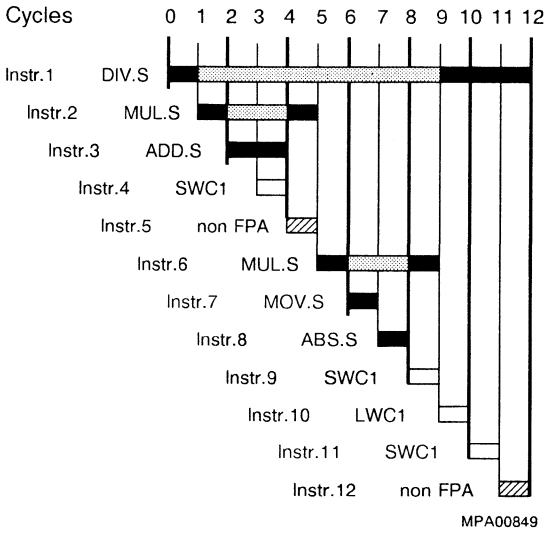
In figure 7 the cycles of an instruction's execution time which are shaded darkest (i.e. at the beginning and at the end of instruction execution time) require exclusive access to an FPA resource (such as the Add unit) that precludes the concurrent use by another instruction and therefore prohibits overlapping execution of another FPA instruction. However, Load and Store operations can be overlapped with these cycles because the SAB-R3010's register unit can execute memory operations when the other arithmetic units are busy.

Figure 7
FPA Instruction Execution Times



Those cycles that are lightly shaded (i.e. in the middle of the Multiply and Divide instructions execution time) place minimal demands on SAB-R3010 resources (i.e. for a Multiply instruction only the Multiply unit is being used) and other instructions can be overlapped to obtain simultaneous execution of instructions without stalling the pipeline. However, two Multiply or two Divide operations cannot be overlapped. An example of overlapped FPA and non-FPA instructions is shown in figure 8.

Figure 8
Overlapping FPA Instructions



In this figure the first operation (DIV.S) requires a total of 12 cycles for execution. Only the first and last 3 cycles of this operation preclude the simultaneous execution of another FPA operation. Similarly, in the second operation (MUL.S) there are two cycles in the middle where an FPA operation can be overlapped. In this case the overlapping operation is ADD.S. Although the execution of an instruction requires 6 pipestages, the SAB-R3010 does not require that each instruction complete execution within 6 cycles to avoid stalling the instruction pipeline. If a subsequent instruction does not require the FPA resources being used by a preceding instruction and has no data dependencies with preceding uncompleted instructions, then execution continues. This can be seen clearly in figure 8.

This figure assumes that there are no data dependencies between the instructions that would stall the pipeline. For example, if any instruction before Instr.13 (not shown in figure 8) required the results of Instr.1 (DIV.S), then the pipeline would be stalled until the results are available.

Note: For a detailed discussion of the individual pipestages refer to the SAB-R3000 data sheet.

Coprocessor Registers

Floating-Point Registers

The SAB-R3010 provides thirty-two 32-bit Floating-Point General Registers (FGR's). These are accessed through coprocessor Load/Store instructions and Move to/from coprocessor register instructions. There are two views of the thirty-two coprocessor FGR's. One is from the standpoint of the SAB-R3000, which has no intrinsic representation of coprocessor registers. It regards these registers as simply thirty-two 32-bit registers. From the standpoint of the SAB-R3010, pairs of these single word registers form Floating-Point Registers (FPR's), on which floating-point operations are performed. The SAB-R3010 contains 16 FPR's. Figure 9 shows the FGR's and the corresponding FPR's.

The FPR's provide a sufficient amount of registers to support the allocation of floating-point values in registers and to permit overlapping and scheduling of floating-point operations. Each FPR can hold one value of either a single- or double-precision format floating-point number. Only even numbers are used to address FPR's, odd FPR register numbers are invalid. During single-precision floating-point operations, only the even numbered (least) FGR's are used, and during double-precision operations, the FGR's are accessed in pairs. Thus, in double-precision operation, selecting FPR0 addresses FGR0 and FGR1. Table 1 shows the register addresses.

Figure 9
FPA Registers

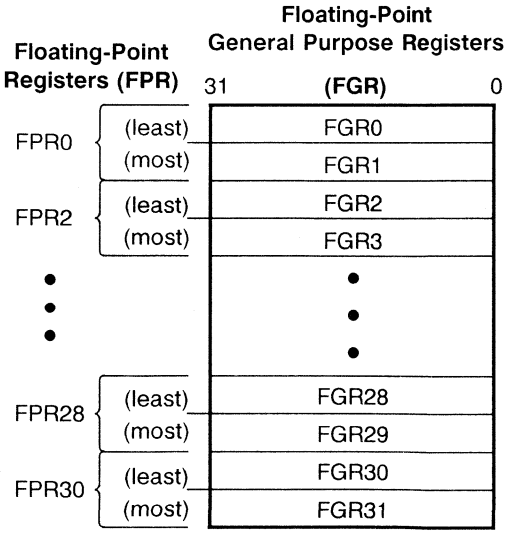


Table 1
Floating-Point General Registers

FGR Number	Usage
0	FPR 0 (least)
1	FPR 0 (most)
2	FPR 2 (least)
3	FPR 2 (most)
•	•
•	•
•	•
28	FPR 28 (least)
29	FPR 28 (most)
30	FPR 30 (least)
31	FPR 30 (most)

Floating-Point Control Registers

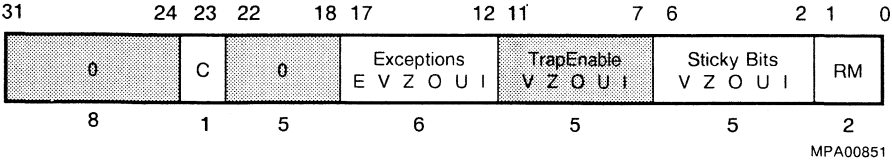
Coprocessors for the SAB-R3000 can have up to thirty-two 32-bit control registers. The SAB-R3010 implements two Floating-Point Control Registers (FCR's). These registers are the Control/Status register (FCR31) and the Implementation/Revision register (FCR0). These registers can only be accessed through Move to/from coprocessor register instructions which address floating-point control registers.

Control/Status Register:
contains control and status data and can be accessed by instructions running in either Kernel or User mode. It controls the arithmetic rounding mode and the enabling of exceptions. It also indicates the exceptions that occurred in the most recently executed instruction, and all exceptions that have occurred since the register was cleared.

Reading this register (using a Move Control From Coprocessor 1 instruction, CFC1), causes all unfinished instructions in the SAB-R3010's pipeline to be completed before the contents of the register are transferred to the SAB-R3000. If an exception occurs as the pipeline empties, the exception is taken and the Move instruction can be re-executed after the exception is serviced. Figure 10 illustrates the Control/Status register.

Figure 10
Control/Status Register Bit Assignments

The Control/Status Register



C : Condition bit. Set/cleared to reflect result of Compare instruction; drives the FPA's CpCond output signal.

Exceptions : These bits are set to indicate any exceptions that occurred during the most recent instruction.

TrapEnable : Trap Enables. These bits enable assertion of the CpInt# signal if the corresponding *Exception* bit is set during a floating-point operation.

Sticky bits : These bits are set if an exception occurs and are reset only by explicitly loading new settings into this register (with a Move instruction).

RM : Rounding Mode. These two bits specify which of the four rounding modes is to be used by the FPA.

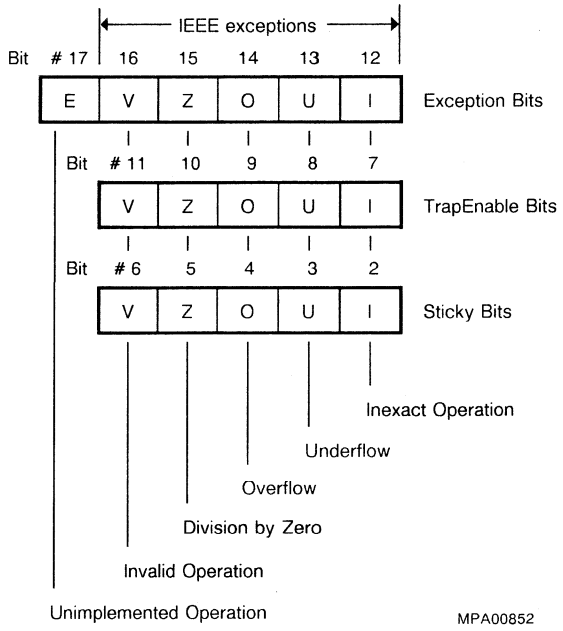
0 : Reserved. Currently ignores writes, undefined when read.

The bits in the Control/Status register can be set or cleared by writing to the register using a Move Control To Coprocessor 1 (CTC1) instruction. This register must only be written to when the FPA is not actually executing floating-point operations. This can be assured by first reading the contents of this register to empty the pipeline.

Condition bit:
When a floating-point Compare instruction takes place, the detected condition is placed at bit 23, the "C" (condition) bit, so that the state of the condition line may be saved or restored. If the condition is true it is set (1) and cleared (0) if it is false. This bit is only affected by Compare and Move To Control Register instructions.

Exception bits:
These are bits 17 through 12 in the Control/Status register, which are shown in figure 11, which indicates the meaning of each bit.

Figure 11
Control/Status Register Exception/Sticky/TrapEnable Bits



These bits are appropriately set or cleared after each floating-point instruction. This is a side effect of each floating-point operation (excluding Loads, Stores and unformatted Moves). The exceptions which were caused by the immediately previous floating-point operation can be determined by reading the exception field.

If two exceptions occur together in one instruction, both appropriate bits in the exception bit field will be set. When an exception occurs, both the corresponding exception and sticky bits are set. The exception bits cover the five IEEE standard exceptions and an extra unimplemented operation exception (E bit). The unimplemented operation exception is not one of the standard IEEE exceptions. It is provided to permit software implementation of IEEE standard operations and exceptions that are not fully supported by the FPA hardware. Trapping on this exception cannot be disabled – there is no TrapEnable bit for E.

Sticky bits:

Hold the accumulated or accured exception bits required by the IEEE standard for trap disabled operation. These bits are set whenever an FPA operation result causes one of the corresponding Exception bits to be set. However, unlike the Exception bits, the Sticky bits are never cleared as a side effect of floating-point operations; they can be cleared only by writing a new value into the Control/Status register.

TrapEnable bits:

Are used to enable a user trap when an exception occurs during a floating-point operation. If the TrapEnable bit corresponding to the exception is set (1) it causes the assertion of the FPA's FpInt# signal. The SAB-R3000 responds to the FpInt# signal by taking an interrupt exception which can be used to implement trap handling of the FPA exception.

Rounding Mode Control bits:

These bits specify the rounding mode the FPA will use for all floating-point operations as shown in table 2.

Table 2
Rounding Mode Bit Decoding

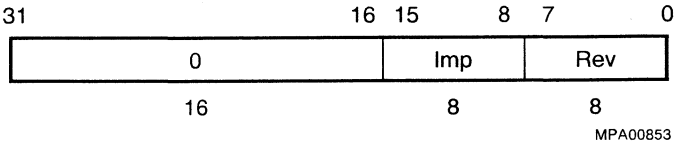
RM Bits	Mnemonic	Rounding Mode Description
00	RN	Rounds result to nearest representable value; rounds to value with least significant bit zero when the two nearest representable values are equally near.
01	RZ	Rounds result toward zero; rounds to value closest to and not greater in magnitude than the infinitely precise result.
10	RP	Rounds toward $+\infty$; rounds to value closest to and not less than the infinitely precise result.
11	RM	Rounds toward $-\infty$; rounds to value closest to and not greater than the infinitely precise result.

Implementation and Revision Register:

This read only register, FCR0, contains values that define the implementation and revision number of the SAB-R3010. This information can be used to determine the coprocessor revision and performance level and can also be used by diagnostic software. However, due to the variety of levels at which design changes may be implemented to the silicon, the revision information cannot be guaranteed with every revision of the device nor assured to follow a completely predictable numerical sequence. Siemens has complete discretion over defining these characteristics of the FPA.

Only the low-order bits of the implementation and revision register are defined. Bit 15 through 8 identify the implementation and bits 7 through 0 identify the revision number as shown in figure 12.

Figure 12
Implementation/Revision Register

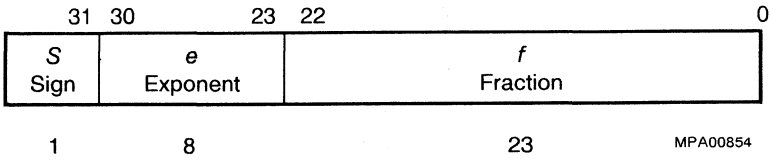


Imp : Implementation: 0 × 10 = R3010
Rev : Revision of FPA
0 : Unused; ignored on writes, zero when read.

Floating-Point Formats

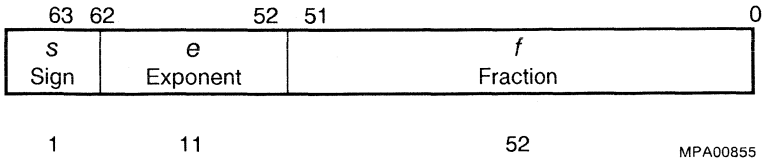
The SAB-R3010 performs both 32-bit (single-precision) and 64-bit (double-precision) IEEE standard floating-point operations. The 32-bit format ist divided into 3 fields: a single-bit sign, an 8-bit biased exponent and a 23-bit fraction, as shown in figure 13.

Figure 13
Single-Precision Floating-Point Format



The 64-bit format has a 1-bit sign, an 11-bit biased exponent and a 52-bit fraction field, as shown in figure 14.

Figure 14
Double-Precision Floating-Point Format



Numbers in the single- and double-precision floating-point formats are composed of three fields;

- A 1-bit sign: s
- A biased exponent: $e = E + \text{bias}$
- A fraction: $f = .b_1b_2...b_{p-1}$

The range of the unbiased exponent "E" includes every integer between and including two values "E_{min}" and "E_{max}", and also two other reserved values: "E_{min} - 1" to encode +/- 0 and denormalized numbers, and "E_{max} + 1" to encode +/- ∞ and NaNs (Not a Number). For single- and double-precision each representable non-zero numerical value has just one encoding.

For single- and double-precision formats, the value of a number, "v", is determined by the equations shown in table 3.

Table 3
Equations for Calculating Values in Floating-Point Format

(1)	if $E = E_{max} + 1$ and $f \neq 0$, then v is NAN, regardless of s.
(2)	if $E = E_{max} + 1$ and $f = 0$, then $v = (-1)^S \infty$.
(3)	if $E_{min} \leq E \leq E_{max}$, then $v = (-1)^S 2^E (1.f)$.
(4)	if $E = E_{min} - 1$ and $f \neq 0$, then $v = (-1)^S 2^{E_{min}} (0.f)$.
(5)	if $E = E_{min} - 1$ and $f = 0$, then $v = (-1)^S 0$.

For all floating-point formats, if "v" is NaN, the most significant bit of "f" determines whether the value is a signaling or quiet NaN. "v" is a signaling NaN if the most significant bit of "f" is set; otherwise, "v" is a quiet NaN. Signaling NaNs indicate uninitialized variables or variables for implementing user-designed extensions to the operations provided by the IEEE standard. Quiet NaNs are generated for invalid operations. Table 4 defines the values for the format parameters in the preceding description.

Table 4
Floating-Point Format Parameter Values

Parameter	Single	Double
P	24	53
E_{max}	+ 127	+ 1023
E_{min}	- 126	- 1022
exponent *bias*	+ 127	+ 1023
exponent width in bits	8	11
integer bit	hidden	hidden
fraction width in bits	23	52
format width in bits	32	64

Number Definitions

This subsection contains a definition of the following number types specified in the IEEE 754 standard:

- Normalized Numbers
- Denormalized Numbers
- Infinity
- Zero

Normalized Numbers:

The majority of floating-point calculations are performed on normalized numbers. Normalized numbers have a biased exponent "e" and a normalized fraction field "f" - which means that the leftmost (i.e. the one to the immediate left of the binary point), or hidden, bit is one.

Denormalized Numbers:

Have a zero exponent and a denormalized (hidden bit equal to zero) non-zero fraction field.

Infinity:

Has an exponent of all ones and a fraction field equal to zero. Both positive and negative infinity are supported.

Zero:

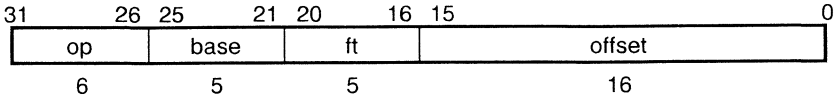
Has an exponent of zero, a hidden bit equal to zero and a value of zero in the fraction field. Both positive and negative zero are supported.

Instruction Set Overview

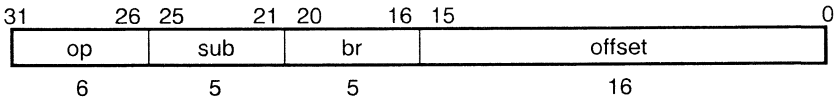
All SAB-R3010 instructions are 32-bits long. There are four basic instruction format types as shown in figure 15.

Figure 15
Instruction Formats

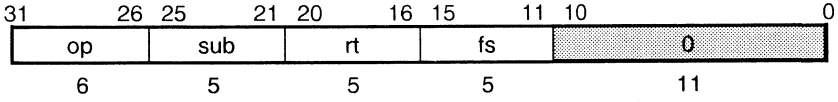
I-type (Immediate)



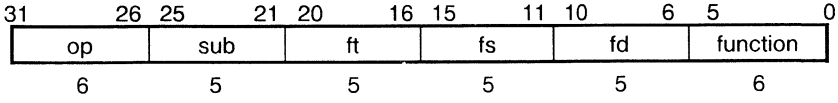
B-type (Branch)



M-type (Move)



R-type (Register)



MPA00856

where

op	:	is a 6-bit operation code
sub	:	is a 5-bit sub-operation code
br	:	is a 5-bit branch code
rt	:	is a 5-bit source/destination general register specifier
ft	:	is a 5-bit source/destination float register specifier
fs	:	is a 5-bit source register specifier
fd	:	is a 5-bit destination register specifier
offset	:	is a 16-bit address/branch displacement
function	:	is a 6-bit function code
0	:	result of operation undefined if non-zero

The single instruction length simplifies instruction fetch and decode and eliminates the overhead for instructions crossing word and page boundaries within the memory hierarchy, thereby simplifying the interaction of instruction fetch with the virtual memory management unit. The four instruction formats ensure that opcodes and register descriptors are always found in the same bit locations. This enables register fetch to proceed in parallel with instruction decode on all instructions.

The SAB-R3010 instruction set can be divided into the following groups:

- **Load/Store and Move** instructions move data between memory, the main processor and the FPA general registers.
- **Computational** instructions perform arithmetic operations on floating-point values in the FPA registers.
- **Conversion** instructions perform conversion operations between the various data formats, e.g. floating-point to fixed-point format.
- **Compare** instructions perform comparisons of the contents of registers and set the condition bit based on the results.

Table 5 lists the instruction set of the SAB-R3010 FPA. A more detailed summary is contained in the Instruction Set Summary section.

Table 5
Instruction Set Summary

OP	Description
Load/Store/Move Instructions	
LWC1	Load word to FPA
SWC1	Store word from FPA
MTC1	Move word to FPA
MFC1	Move word from FPA
CTC1	Move control word to FPA
CFC1	Move control word from FPA
Conversion Instructions	
CVT.S.fmt	Floating-point convert to single FP
CVT.D.fmt	Floating-point convert to double FP
CVT.W.fmt	Floating-point convert to fixed-point

OP	Description
Computational Instructions	
ADD.fmt	Floating-point add
SUB.fmt	Floating-point subtract
MUL.fmt	Floating-point multiply
DIV.fmt	Floating-point divide
ABS.fmt	Floating-point absolute value
MOV.fmt	Floating-point move
NEG.fmt	Floating-point negate
Compare Instructions	
C.cond.fmt	Floating-point compare

Exception Handling

This section describes how the SAB-R3010 FPA handles floating-point exceptions. The term exception is used for any infrequent or exceptional event that causes the SAB-R3010 to make a temporary transfer of control from its current process to another process that services the event. A floating-point exception occurs whenever the FPA cannot handle the operands or results of a floating-point operation in the normal way. On the occurrence of an exception the FPA either generates an interrupt (by asserting the signal FpInt#) to initiate a software trap, or sets a flag. If the trap is taken, the FPA remains in the state found at the beginning of the operation (i.e. execution is suspended) and a software exception handling routine is executed. If no trap is taken (i.e. a flag is set), an appropriate value is written into the SAB-R3010 destination register (of the exceptional instruction) and execution continues (see table 6).

The five IEEE standard exceptions are supported with exception bits, trap enables and sticky bits (status flags). Refer to the Control/Status register in the Coprocessor's Registers section. The SAB-R3010 has an additional exception type, unimplemented operation exception (E). This is used in cases where the FPA itself cannot implement the floating-point architecture specification, including cases where the FPA cannot determine the correct exception behaviour. The unimplemented operation exception has no trap enable or sticky bits; whenever this exception occurs, an unimplemented exception trap is taken (if the FPA's interrupt input to the SAB-R3000 is enabled). It is impossible to disable this exception, there is no trap enable bit.

Each of the five IEEE exceptions (Invalid Operation, Division by Zero, Overflow Exception, Underflow Exception and Inexact Operation) is associated with a trap under user control which is enabled by setting one of the five TrapEnable bits. When an exception occurs, both the corresponding Exception and Sticky bits are set. If the corresponding TrapEnable bit is set, the FPA generates an interrupt to the SAB-R3000 and the subsequent exception processing allows a trap to be taken.

Exception Processing

When a floating-point exception trap is taken, the SAB-R3000 processor's Cause register (refer to the SAB-R3000 data sheet) indicates that an external interrupt from the FPA is the cause of the exception and the SAB-R3000's EPC (Exception Program Counter) contains the address of the instruction that caused the exception trap.

When no exception trap is signalled, a default action is taken, which provides a substitute value for the original, exceptional result of the floating-point operation. The default action taken depends on the type of exception and, in the case of the Overflow exception, the current rounding mode. Table 6 lists the default action taken by the FPA for each of the IEEE exceptions.

Table 6
FPA Exception Default Actions

Exception		Rounding Mode	Default Action (no exception trap signaled)
V	Invalid operation	– –	Supply a quiet NaN.
Z	Division by zero	– –	Supply a properly signed ∞.
O	Overflow	RN	Modify overflow values to ∞ with the sign of the intermediate result.
		RZ	Modify overflow values to the format's largest finite number with the sign of the intermediate result.
		RP	Modify negative overflows to the format's most negative finite number. Modify positive overflows to + ∞.
		RM	Modify positive overflows to the format's largest finite number. Modify negative overflow to – ∞.
U	Underflow	– –	Generate an unimplemented exception.
I	Inexact	– –	Supply a rounded result.

Internally the SAB-R3010 detects eight different conditions that can cause exceptions. When it encounters one of these unusual situations, it will cause either an IEEE exception or an Unimplemented Operation exception. Table 7 lists the exception-causing situations and contrasts the behaviour of the SAB-R3010 with the IEEE standard's requirements.

Table 7
FPA Exception-causing Conditions

FPA internal result	IEEE Stndrd	Trap Enab.	Trap Disab.	Note
Inexact result	I	I	I	loss of accuracy
Exponent overflow	O I*)	O I	O I	normalized exponent > E_{max}
Divide by zero	Z	Z	Z	zero is (exponent = E_{min}-1, mantissa = 0)
Overflow on convert	V	V	E	source out of integer range
Signaling NaN source	V	V	E	quiet NaN source produces quiet NaN result
Invalid operation	V	V	E	0/0 etc.
Exponent underflow	U	E	E	normalized exponent < E_{min}
Denormalized source	none	E	E	exponent = E_{min}-1 and mantissa < > 0

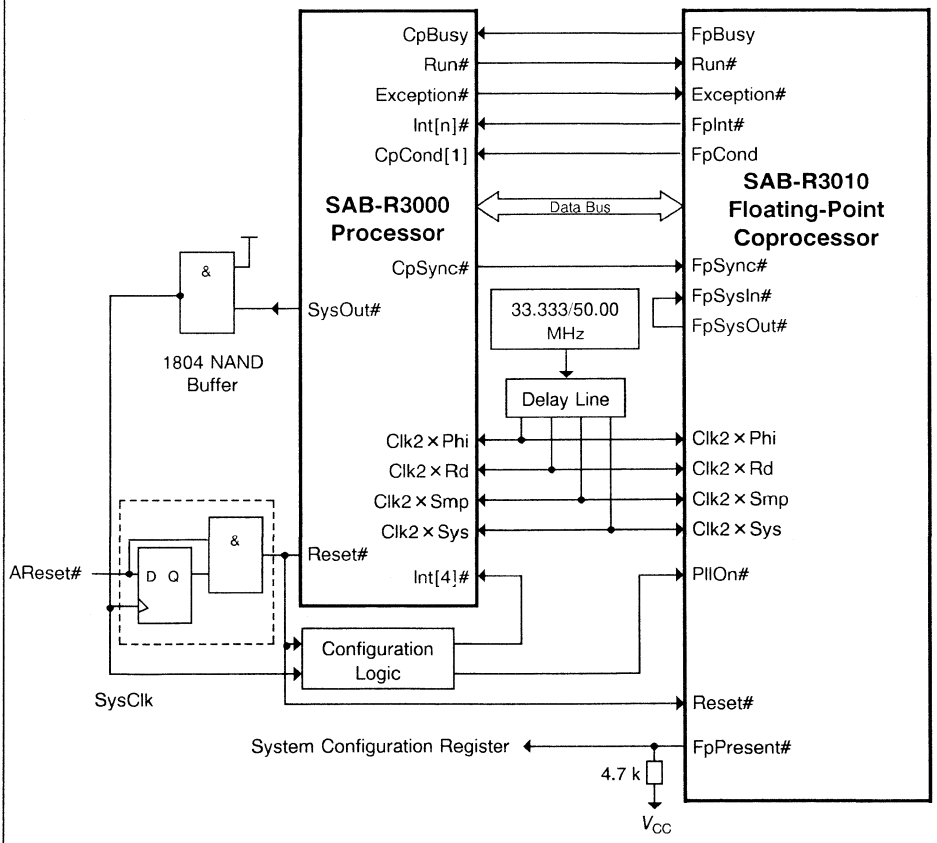
*) Standard specifies inexact exception on overflow only if overflow trap is disabled.

Note: *A detailed description of the "exception handling" system for a SAB-R3000 is contained in the SAB-R3000 data sheet.*

Processor Interface

Figure 16 illustrates the tightly coupled coprocessor interface between the SAB-R3000 and the SAB-R3010.

Figure 16
SAB-R3000 and SAB-R3010 Coprocessor Interface



AReset = Asynchronous Reset

This external coprocessor interface of the SAB-R3000 is designed to support the SAB-R3010 floating point accelerator, in what is called a tightly coupled interface, and up to two additional coprocessors. The SAB-R3010 is connected to the DATA bus only. During each cycle in which a valid Instruction-Data pair is on the bus, the FPA accepts an Instruction. The coprocessor decodes the Instruction in parallel with the main processor and if it is a floating-point Instruction it will proceed to execute the Instruction. The coprocessor condition (CpCond(1) – FpCond) signal allows the main processor to branch on a coprocessor condition set up by a previous operation. The SAB-R3010 can assert FpBusy to stall the main CPU when a floating-point instruction is issued while the FPA still has the required functional unit busy with an earlier operation. The SAB-R3000 asserts Run# to advance operations in the SAB-R3010. When Run# is deasserted (i.e. a stall) in the n th cycle the FPA disregards the instruction-data pair presented in the n-1 th cycle. The assertion of Exception# indicates that the SAB-R3000 is taking an exception. FpSync# is used for timing synchronization between the SAB-R3000 and the coprocessor.

Instruction Set Summary

The following section is a table of the instructions available in the SAB-R3010. The instructions are listed in alphabetical order. For a more detailed description of the operation of each instruction refer to the "SAB-R3010 Users Manual". A chart at the end of this section lists the bit encoding for the constant fields of each instruction.

Instruction Notation Convention

The table that follows is split up into three columns: Instruction, format and operation. The instruction column contains the mnemonic name of the instruction and its meaning. The instruction format (refer to figure 15) and assembly language notation for each instruction are listed in the format column. The operation column describes the operation performed by each instruction using a high level language notation. Special symbols used in the notation are described in table 8.

Table 8
FPA Instruction Operation Notations

Symbol	Meaning
←	Assignment
||	Bit string concatenation
x^y	Replication of bit value *x* into a *y*-bit string. Note that *x* is always a single-bit value.
$x_{y..z}$	Selection of bits *y* through *z* of bit string *x*. Little-endian bit notation is always used. If *y* is less than *z*, this expression is an empty (zero length) bit string.
+	Two's complement or floating-point addition
–	Two's complement or floating-point subtraction
*	Two's complement or floating-point multiplication
div	Two's complement integer division
mod	Two's complement modulo
<	Two's complement less than comparison
and	Bitwise logic AND
or	Bitwise logic OR
xor	Bitwise logic XOR
nor	Bitwise logic NOR
GPR[*x*]	SAB-R3000 General Register *x*. Note that the contents of GPR[0] are always zero: attempts to alter GPR[0] contents have no effect.
FGR[*x*]	FPA General Register *x*. As viewed by the R3000 processor.
FPR[*x*]	FPA Floating-Point register *x*. Each FPR is assembled from two FGRs.
FCR[*x*]	FPA Control Register *x*.
T + *i*	Indicates the time steps (CPU cycles) between operations. Thus, operations identified as occurring at T + 1 are performed during the cycle following the one where the instruction was initiated. This type of operation occurs with loads, stores, jumps, branches and coprocessor instructions.
virtualAddress	Virtual address
physicalAddress	Physical address

In the Load/Store operation descriptions, the functions listed in table 9 are used to summarize the handling of virtual addresses and physical memory.

Table 9
Load/Store Common Functions

Function	Description
Addr Translation	Uses the TLB to find the physical address given the virtual address. The function fails and an exception is taken if the entry for the page containing the virtual address is not present in the TLB (Translation Lookaside Buffer).
Load Memory	Uses the cache and main memory to find the contents of the word containing the specified physical address. The low-order two bits of the address and the access type field indicate which of each of the four bytes within the data word need to be returned. If the cache is enabled for this access, the entire word is returned and loaded into the cache.
Store Memory	Uses the cache, write buffer, and main memory to store the word or part of word specified as data into the word containing the specified physical address. The low-order two bits of the address and the access type field indicate which of the four bytes within the data word should be stored.

The mnemonics of the floating-point instructions contain a ".fmt" field. This means "format" and table 10 shows the three formats available.

Table 10
".fmt" field encoding

Mnemonic	Size	Format
S	single	binary floating-point
D	double	binary floating-point
W	single	binary fixed-point

Instruction	Format	Operation
ABS.fmt: Floating-Point Absolute Value	R-Type; ABS.fmt fd, fs	T: StoreFPR (fd, fmt, AbsoluteValue(Value FPR (fs.fmt)));
ADD.fmt: Floating-Point Add	R-Type; ADD.fmt fd, fs, ft	T: StoreFPR (fd, fmt, ValueFPR(fs.fmt) + ValueFPR (ft.fmt));
BC1F: Branch On FPA False (Coprocessor 1)	B-Type; BC1F offset	T: target ← (offset $_{15}$) ^{14} ‖ offset ‖ 0^2 condition ← not CpCond(1) T + 1: If condition then PC ← PC + target endif
BC1T: Branch On FPA True (Coprocessor 1)	B-Type; BC1T offset	T: target ← (offset $_{15}$) ^{14} ‖ offset ‖ 0^2 condition ← CpCond(1) T + 1: If condition then PC ← PC + target endif

C.cond.fmt:
Floating-Point Compare
	R-Type;
	C.cond.fmt fs, ft
		T: if NaN(ValueFPR(fs, fmt)) or NaN(ValueFPR(ft, fmt)) then
		less ← false
		equal ← false
		unordered ← true
		if cond$_3$ then
		signal InvalidOperationException
		endif
		else
		less ← ValueFPR(fs.rmt) < ValueFPR(ft.rmt)
		equal ← ValueFPR(fs.rmt) = ValueFPR(ft.rmt)
		unordered ← false
		endif
		T + 1: condition ← (cond$_2$ and less) or
		(cond$_1$ and equal) or
		(cond$_0$ and unordered)

Instruction	Format	Operation
CFC1: Move Control word from FPA (Coprocessor 1)	M-Type; CFC1 rt, ts	T: temp ← FCR[fs]; T + 1: GPR[rt] ← temp;
CTC1: Move Control word to FPA (Coprocessor 1)	M-Type; CTC1 rt, fs	T: temp ← GPR[rt]; T + 1: FCR[fs] ← temp
CVT.D.fmt: Floating-Point Convert to Double Floating-Point Format	R-Type; CVT.D.fmt fd, fs	T: StoreFPR (fd, D, ConvertFmt(ValueFPR(fs, fmt), fmt, D))
CVT.S.fmt: Floating-Point Convert to Single Floating-Point Format	R-Type; CVT.S.fmt fd, fs	T: StoreFPR (fd, S, ConvertFmt(ValueFPR(fs, fmt), fmt, S))
CVT.W.fmt: Floating-Point Convert to Fixed-Point Format	R-Type; CVT.W.fmt fd, fs	T: StoreFPR (fd, W, ConvertFmt(ValueFPR(fs, fmt), fmt, W))
DIV.fmt: Floating-Point Divide	R-Type; DIV.fmt fd, fs, ft	T: StoreFPR (fd, fmt, ValueFPR(fs, fmt) / ValueFPR (ft, fmt));
LWC1: Load Word to FPA (Coprocessor 1)	I-Type; LWC1 ft, offset (base)	T: virtualAddress ← (offset $_{15}$) ^{16} ‖ offset $_{15..0}$ + GPR[base]; physicalAddress ← AddressTranslation (virtualAddress); mem ← LoadMemory (WORD, physicalAddress); byte ← virtualAddress $_{1..0}$; T + 1: FGR[ft] ← mem
MFC1: Move from FPA (Coprocessor 1)	M-Type; MFC1 rt, fs	T: temp ← FGR[fs]; T + 1: GPR[rt] ← temp
MOV.fmt: Floating-Point Move	R-Type; MOV.fmt fd, fs	T: StoreFPR (fd, fmt, ValueFPR (fs, fmt));

Instruction	Format	Operation
MTC1: Move to FPA (Coprocessor 1)	M-Type; MTC1 rt, fs	T: temp ← GPR[rt]; T + 1: FGR[fs] ← data;
MUL.fmt: Floating-Point Multiply	R-Type; MUL.fmt fd, fs, ft	T: StoreFPR (fd, fmt, ValueFPR(fs, fmt) * ValueFPR (ft, fmt));
NEG.fmt: Floating-Point Negate	R-Type; NEG.fmt fd, fs	T: StoreFPR (fd, fmt, Negate(ValueFPR(fs, fmt)));
SUB.fmt: Floating-Point Subtract	R-Type; SUB.fmt fd, fs, ft	T: StoreFPR (fd, fmt, ValueFPR(fs, fmt) – ValueFPR (ft, fmt));
SWC1: Store Word from FPA (Coprocessor 1)	I-Type; SWC1 ft, offset (base)	T: virtualAddress ← (offset $_{15}$) ^{16} ‖ offset $_{15..0}$ + GPR[base] physicalAddress ← AddressTranslation (virtualAddress); data ← FGR[ft] T + 1: StoreMemory (WORD, data, physicalAddress)

Instruction Encoding

31..29 \ 28..26 (Opcode)	0	1	2	3	4	5	6	7
0	~	~	~	~	~	~	~	~
1	~	~	~	~	~	~	~	~
2	~	COP1	~	~	~	~	~	~
3	~	~	~	~	~	~	~	~
4	~	~	~	~	~	~	~	~
5	~	~	~	~	~	~	~	~
6	~	LWC1	~	~	~	~	~	~
7	~	SWC1	~	~	~	~	~	~

25..24 \ 23..21 (sub)	0	1	2	3	4	5	6	7
0	MF	~	CF	~	MT	~	CF	~
1	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗
2	S	D	⊗	⊗	W	⊗	⊗	⊗
3	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗

20..19 \ 18..16 (br)	0	1	2	3	4	5	6	7
0	BCF	BCT	~	~	~	~	~	~
1	~	~	~	~	~	~	~	~
2	~	~	~	~	~	~	~	~
3	~	~	~	~	~	~	~	~

5..3 \ 2..0 (function)	0	1	2	3	4	5	6	7
0	ADD.fmt	SUB.fmt	MUL.fmt	DIV.fmt	⊗	ABS.fmt	MOV.fmt	NEG.fmt
1	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗
2	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗
3	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗
4	CVT.S	CVT.D	⊗	⊗	CVT.W	⊗	⊗	⊗
5	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗
6	C.F	C.UN	C.EQ	C.UEQ	C.OLT	C.ULT	C.OLE	C.ULE
7	C.SF	C.NGLE	C.SEQ	C.NGL	C.LT	C.NGE	C.LE	C.NGT

⊗ Codes marked with a '⊗' cause unimplemented instruction exceptions and are reserved for future versions of the architecture.

~ Codes marked with a '~' are not valid and are reserved for future versions of the architecture. The results of such an encoding are undefined.

Absolute Maximum Ratings

Case temperature under bias (T_C) . . . . . 0 to + 115 °C
Storage temperature (T_{ST}) . . . . . – 65 to + 150 °C
Supply Voltage (V_{CC}) . . . . . – 0.5 to + 7.0 V
Input voltage (V_{IN} min. = – 3.0 V for pulse width less than 15 ns) . . . – 0.5 to + 7.0 V

Note: *Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*
Not more than one output should be shorted at a time. Duration of the short should not exceed 30 seconds.

DC Characteristics

T_C = 0 to + 90 °C; V_{CC} = 5 V ± 5%

Parameter	Symbol	Limit values						Unit	Test condition
		16.67 MHz		20 MHz		25 MHz			
		min.	max.	min.	max.	min.	max.		
Operating Parameters									
Output HIGH voltage	V_{OH}	3.5	–	3.5	–	3.5	–	V	V_{CC} = min. I_{OH} = – 4mA
Output LOW voltage	V_{OL}	–	0.4	–	0.4	–	0.4	V	V_{CC} = min. I_{OL} = 4mA
Input HIGH voltage	V_{IH}	2	V_{CC} + 0.25	2	V_{CC} + 0.25	2	V_{CC} + 0.25	V	–
Input LOW voltage	V_{IL}	– 0.5 [1]	0.8	– 0.5 [1]	0.8	– 0.5 [1]	0.8	V	–
Input HIGH voltage	V_{IHS} [2]	3.0	V_{CC} + 0.25	3.0	V_{CC} + 0.25	3.0	V_{CC} + 0.25	V	–
Input LOW voltage	V_{ILS} [2]	– 0.5 [1]	0.4	– 0.5 [1]	0.4	– 0.5 [1]	0.4	V	–
Input HIGH voltage	V_{IHC} [3]	4.0	V_{CC} + 0.25	4.0	V_{CC} + 0.25	4.0	V_{CC} + 0.25	V	–
Input LOW voltage	V_{ILC} [3]	– 0.5 [1]	0.4	– 0.5 [1]	0.4	– 0.5 [1]	0.4	V	–
Input capacitance	C_{In}	–	10	–	10	–	10	pF	–
Output capacitance	C_{Out}	–	10	–	10	–	10	pF	–
Operating current	I_{CC}	–	650	–	700	–	750	mA	V_{CC} = 5.25 V
Load capacitance	C_{Ld} [4]	–	75	–	50	–	50	pF	–

1) V_{IL} min. = –3.0 V for pulse width less than 15 ns
2) V_{IHS} and V_{ILS} apply to Clk2 × Sys, Clk2 × Smp, Clk2 × Rd, Clk2 × Phi, FpSysIn#, FpSync#, and Reset#.
3) V_{IHC} and V_{ILC} apply to Run# and Exception#.
4) Operation above the C_{Ld} maximum may impair the useful life of the device.

AC Characteristics

T_C = 0 to 90 °C; V_{CC} = 5 V ±5%

Notes: All output timings are given assuming 25 pf of capacitive load. Output timings should be derated where appropriate as per the table below. All timings referenced to 1.5 V.

Parameter	Symbol	Limit values						Unit	Test condition
		16.67 MHz		20 MHz		25 MHz			
		min.	max.	min.	max.	min.	max.		
Clock Parameters 5)									
Input clock high	$t_{ClkHigh}$	12	–	10	–	8	–	ns	Trans. ≤ 5 ns
Input clock low	t_{ClkLow}	12	–	10	–	8	–	ns	Trans. ≤ 5 ns
Input clock period	t_{ClkP}	30	1000	25	1000	20	1000	ns	–
Clk2 × Sys to Clk2 × Smp		0	$t_{\frac{Cyc}{4}}$	0	$t_{\frac{Cyc}{4}}$	0	$t_{\frac{Cyc}{4}}$	ns	–
Clk2 × Smp to Clk2 × Rd		0	$t_{\frac{Cyc}{4}}$	0	$t_{\frac{Cyc}{4}}$	0	$t_{\frac{Cyc}{4}}$	ns	–
Clk2 × Smp to Clk2 × Phi		9	$t_{\frac{Cyc}{4}}$	7	$t_{\frac{Cyc}{4}}$	5	$t_{\frac{Cyc}{4}}$	ns	–
Run Operation Parameters									
Data enable	t_{DEn}	– 1	– 2	– 1	– 2	– 0.5	– 1.5	ns	–
Data disable	t_{DDis}	0	– 1	0	– 1	0	– 0.5	ns	–
Data valid	t_{DVal}	–	3	–	3	–	3	ns	25 pF load
Data setup	t_{DS}	9	–	8	–	7	–	ns	–
Data hold	t_{DH}	– 2.5	–	– 2.5	–	– 2.5	–	ns	–
FpCondition	t_{FpCond}	0	35	–	30	0	25	ns	–
FpBusy	t_{FpBusy}	0	15	–	13	0	10	ns	–
FpInterrupt	t_{FpInt}	0	40	–	35	0	27	ns	–
FpMove To	t_{FpMov}	0	35	–	30	0	25	ns	–
Exception setup	t_{ExS}	10	–	9	–	7	–	ns	–
Exception hold	t_{ExH}	0	–	0	–	0	–	ns	–
Run setup	t_{RunS}	17	–	16	–	15	–	ns	–
Run hold	t_{RunH}	– 2	–	–2	–	– 2	–	ns	–
Stall setup	t_{StallS}	10	–	9	–	9	–	ns	–
Stall hold	t_{StallH}	– 2	–	– 2	–	– 2	–	ns	–
Capacitive Load Deration									
Load derate	C_{LD}	0.5	2	0.5	1	0.5	1	ns/ 25pF	–

5) The clock parameters apply to all four 2xClocks: Clk2 × Sys, Clk2 × Smp, Clk2 × Rd, and Clk2 × Phi.

Operation Fundamentals

A "cycle" is the basic instruction processing unit of the SAB-R3010 processor. Cycles in which forward progress is made, i.e. an instruction is retired, are called "run" cycles. An instruction is retired either by its completion or, in the presence of an exception, its abortion. Cycles in which no forward progress is made are called "stall" cycles. Stall cycles are used for resolving urgent situations such as cache misses on loads, write system busy during stores, and coprocessor interlocks. All cycles can be classified as either run cycles or stall cycles. "Fixup" stall cycles occur during the final cycle of the stall and are used in general to fix up the conditions which caused the stall. Processor transactions which occur during the first half of the cycle are called phase 1 transactions while those which occur during the second half of the cycle are called phase 2 transactions.

As described earlier Run# is asserted by the SAB-R3000 during run cycles and deasserted during stall cycles. When Run# is deasserted during the n th cycle, the SAB-R3000 disregards the instruction-data pair presented during the n-1 th cycle. When Run# is reasserted during the m th cycle, the SAB-R3010 takes, as are placement for the instruction-data pair which was disregarded, the instruction-data pair presented during the m-1 th cycle – which was the final fixup cycle for whatever stalls equence was occurring.

Exception# is used by the FPA to track exception related information during run cycles and stall related information during stall cycles.

- During phase 1 of run cycles Exception# indicates whether an exception has occurred for the instruction which is currently in its "write back" pipestage. Unless the exception is occurring as a result of an interrupt request by the SAB-R3010, the assertion of Exception# prevents any state from being committed in the FPA.
- During phase 2 of run cycles Exception# indicates whether an interrupt request is being granted for the instruction which is currently in its "memory access" pipestage.
- During phase 1 of stall cycles Exception# indicates whether the current stall cycle is a fixup cycle. When a fixup cycle is occurring, it is guaranteed that the data present on the data bus is valid. The FPA uses the fixup indication to qualify the use of data sampled from the bus during the stall.
- During phase 2 of stall cycles Exception# indicates whether the current stall is a Coprocessor Busy stall. The FPA does not use this information.

The use of the Exception# signal is summarized below.

Table 11
Exception#

Cycle	Phase 1	Phase 2
Run	Exc1W#	IntGr2M#
Stall	Fixup1#	CPBusy2#

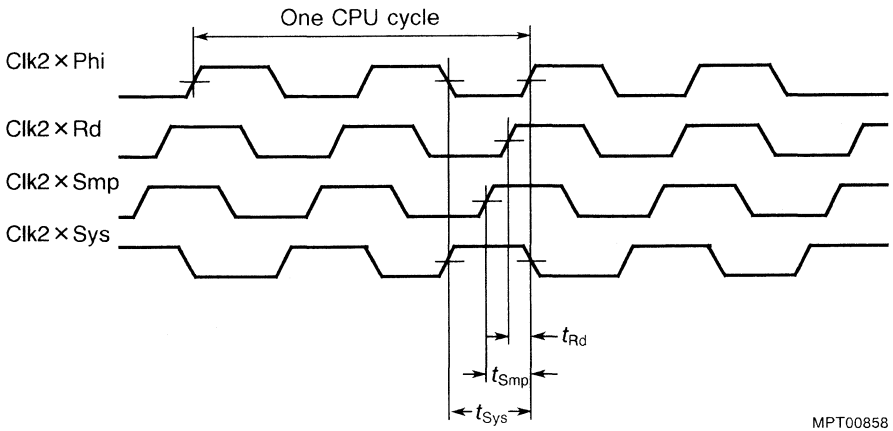
Processor Input Clocks

The SAB-R3010 has the same four separate double-frequency (i.e. in a 25 MHz system these clocks are 50 MHz) input clocks as the SAB-R3000. They can be adjusted to obtain optimum positioning of cache interface signals. The absolute timing of these clocks with respect to the processor outputs is undefined, only the differences are important. A short description of these four clocks follows. Refer to figure 16 for the various signal names.

- Clk2 × Sys:
  is the master clock and must lead all others. It determines the position of SysOut# (the processors output clock) with respect to data, Tag and Address buses.
- Clk2 × Smp:
  determines the sample point for data coming into the SAB-R3000 on all its inputs except those coming directly from coprocessors.
- Clk2 × Rd:
  controls output enable time and provides sufficient address access to sample address hold from end of write, and data hold from end of write.
- Clk2 × Phi:
  determines the position of the internal phases 1 and 2.

Figure 17 shows the four 2×input clocks.

Figure 17
2×Input Clocks



In the timing diagrams which follow, timing specifications are given relative to a shifted version of the FPA output clock, FpSysOut#. The clock is called FpPhiOut# and is a virtual clock, i.e. the processor does not actually produce this output (FpSysOut# and FpPhiOut# are equivalent to SysOut# and PhiOut# for the SAB-R3000). It is shown in the timing diagrams for reasons of clarity, because its period is synchronous with a machine cycle. The shift amount is equal to the difference between Clk2×Sys to Clk2×Phi and, as is shown in figure 17, is t_{Sys}. Also in the timing diagrams Clk2×Sys and FpSysOut# are shown to clarify the relationship between these signals.

In reality FpSysOut# is produced rather than FpPhiOut# since this provides a signal with timing appropriate for synchronizing system transactions to the processor. Timings are given relative to FpPhiOut# since this makes determining the position of the input clocks the most straightforward. The timing of any output with respect to FpSysOut# can be determined from its timing with respect to FpPhiOut# by adding t_{Sys}.

Timing Diagram Notation

The following timing diagrams describe various transactions of the processor. Table 12 illustrates the notational conventions used in these diagrams.

Table 12
Notational Conventions for Timing Diagramms

Character	Meaning
I	Instruction
D	Data
#	Active low
%	An incorrect datum
!	An unused datum
Z	The high impedance state
Ad	Address
in	into coprocessor
out	out of coprocessor
//////	not valid or Don't Care

Load/Store and Processor Transfer Timings

During run cycles, the operation and timing of FPA loads and stores are identical to that of the SAB-R3000. In the case of a load the SAB-R3010 accepts data from the data bus and on stores it drives data on the data bus. Both of these data bus transactions occur during the MEM pipestage of each instruction (refer to figure 4 in the pipeline architecture section). On FPA loads, the SAB-R3000 reads in the data and Tag buses for purposes of miss detection. The Tag bus is the cache Tag bus and is only connected to the CPU – for more information refer to the SAB-R3000 data sheet. For miss detection the SAB-R3000 checks the valid bit, does the Tag comparison and checks parity on the Tag and data buses. On Stores the SAB-R3010 generates data parity. All address generation, cache and memory control functions are provided by the SAB-R3000.

During all stall and fixup cycles, the FPA is passive; if an FPA Store is blocked by a write busy stall or if the cycle in which the FPA Store occurs is redone due to any other stall, the SAB-R3000 will re-present the FPA data during the stall's fixup cycle. Timing of FPA Loads/Stores is illustrated in figure 19.

Transfers between the SAB-R3010 and the SAB-R3000 have identical input and output characteristics as Loads and Stores. That is, for a Move to FPA transfer (MTC1), the SAB-R3000 drives the data bus and the SAB-R3010 reads it, as for an FPA Load. For a Move from FPA instruction (MFC1) the roles are reversed, as for an FPA Store. Parity is not checked for either direction of transfer. The timings for these transfers are also illustrated in figure 19.

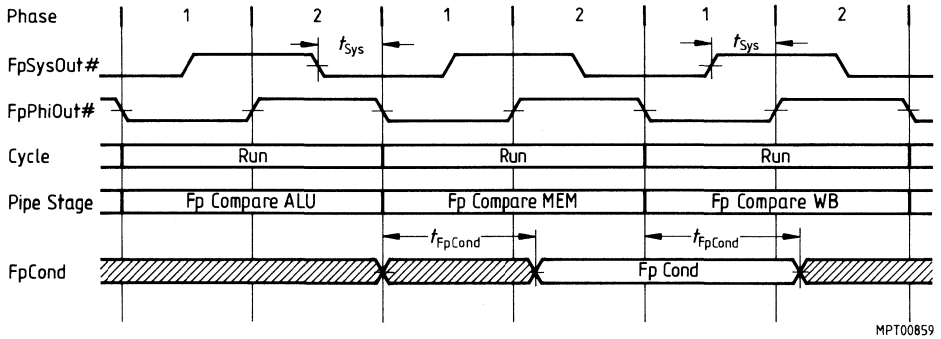
Floating-Point Condition Timing

Floating-point operations occur within the SAB-R3010 and only affect the interface when they change the Floating-Point Condition output or cause stalls or exceptions. Floating-point conditions are described here, stalls and exceptions are described in subsequent sections.

The SAB-R3010 has a Floating-Point Condition output called FpCond. The FpCond output is connected directly to the CpCond(1) input of the SAB-R3000, refer to figure 16. The FpCond signal is sampled by the SAB-R3000 during phase 2 of every run cycle. If the SAB-R3000 executes an FPA branch instruction, the state of the FpCond signal determines the direction of the branch.

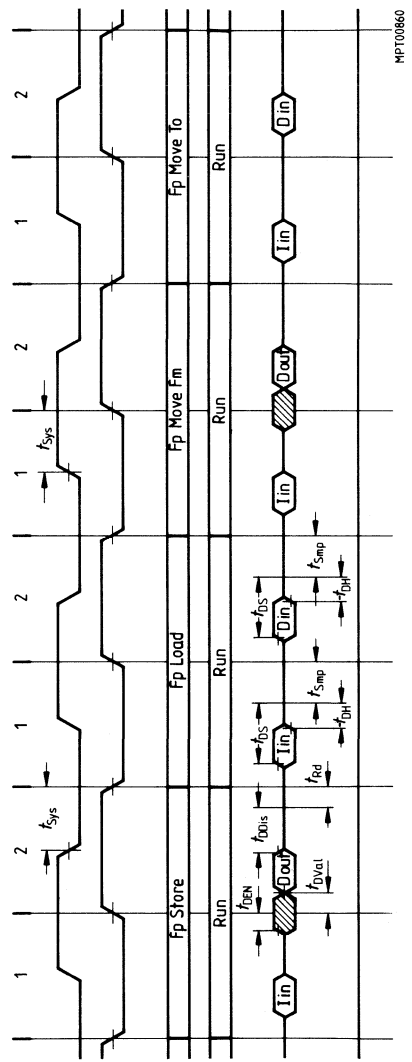
Floating-point instructions which affect the FpCond signal are two-cycle operations (e.g. Floating-Point Compare). This can be seen externally by the invalidity of the FpCond output during the entire ALU pipestage of Instruction Execution, refer to figure 18 which illustrates the FpCond timing. The FpCond output becomes valid during the MEM pipestage of instruction execution, which is too late to be used by the succeeding instruction, therefore the operation requires two cycles. Refer to the Pipeline Architecture section and the SAB-R3000 data sheet for more details on two cycle instructions.

Figure 18
FPCond Timing



Note: The Instruction used here to illustrate the FpCond timing is a Floating-Point Compare Instruction, where FpCompare ALU/MEM/WB are the Floating-Point Compare ALU/MEMORY/Write Back pipestages respectively.

Figure 19
Load/Store and Transfer Timing



FpMoveFm (FpMoveTo) has the same timing as FpStore (FpLoad).

Floating-Point Coprocessor Stall Timing

As described earlier, to maintain synchronization with the SAB-R3000 the FPA requests "Coprocessor Busy" stalls as required. To initiate such a stall the SAB-R3010 asserts FpBusy during phase 2 of the ALU pipestage of the stalling FPA instruction. To terminate the stall FpBusy is deasserted during phase 1 of the stall cycle in which it will complete the operation whose incompletion required the stall. In the absence of other stall requests, the cycle following that in which FpBusy is deasserted will be the fixup cycle. Figure 20 illustrates the FPA busy timing.

For all stalls, whether FPA-initiated or not, the indication of the stall condition is signalled by the SAB-R3000 via the deassertion of the Run# signal. If Run# is not deasserted following the assertion of FpBusy, then the FPA stall request has been ignored by the SAB-R3000 due to the occurrence of some exceptional event.

Exception/Interrupt Timing

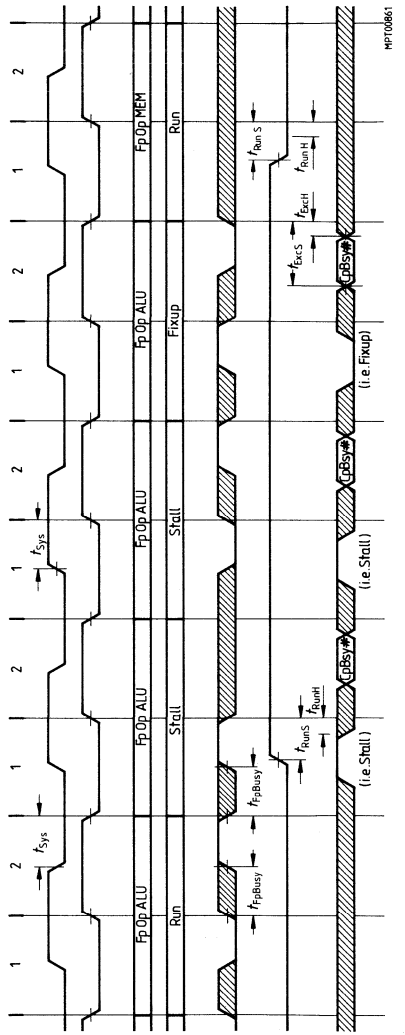
The SAB-R3010 signals exceptions to the SAB-R3000 through one of its interrupt inputs using the FpInt# output, refer to figure 16. The SAB-R3000 samples the interrupt inputs during phase 2 of every run cycle and final fixup cycle of a stall sequence. The FPA signals exceptions by asserting FpInt# during the ALU pipestage of the instruction causing the exception. If the SAB-R3000 takes the interrupt during that cycle, it signals interrupt grant (IntGr2M#) back to the FPA (via the Exception# signal) during the MEM pipestage of the exceptional instruction. Interrupt grant is signalled to the FPA on its Exception# input during phase 2 of the MEM pipestage.

The occurrence of any exception, including those caused by the FPA, is signalled to the FPA during the WB pipestage of the exception-causing instruction. The occurrence of an exception prevents any non-exception-related state from being committed within the FPA. This means that when an exception occurs which is not FPA related, execution in the FPA is suspended until the exception is resolved. The occurrence of an exception is signalled to the FPA on its Exception# input during phase 1 of the WB pipestage of the exceptional instruction. Figure 21 illustrates an Interrupt timing sequence.

Special Case

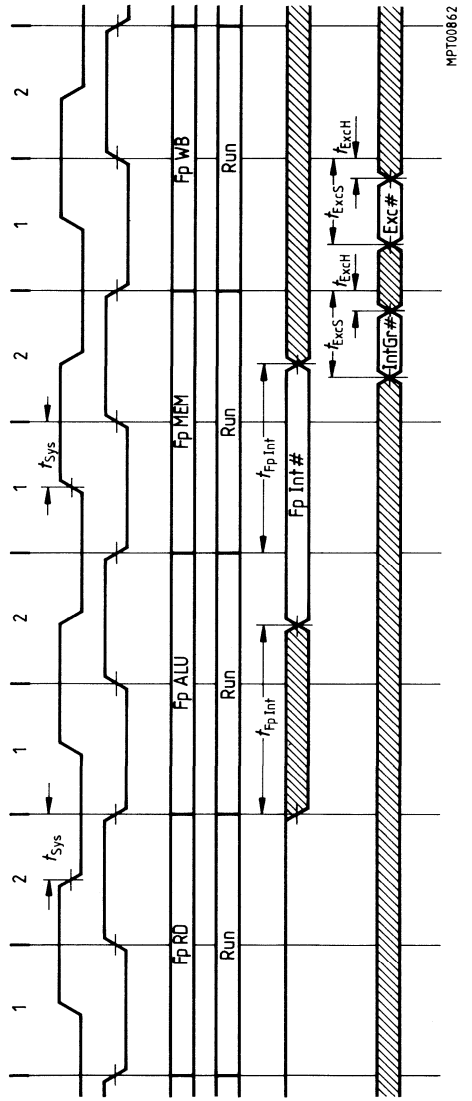
A special case of FPA – SAB-R3000 transfers is the MoveTo FPA Status register. When moves to this register occur the interface can be further affected via a change in the FpInt# or FpCond outputs. Figure 22 illustrates the timing of the FpCond and FpInt# outputs in conjunction with an FPA MoveTo instruction.

Figure 20
SAB-R3010 Busy Timing



Note: CpBusy# is the same as CpBusy2# in table 12.

Figure 21
Interrupt and Exception Timing



Where: IntGr# = IntGr2M# and Exc# = Exc1W# in table 12.

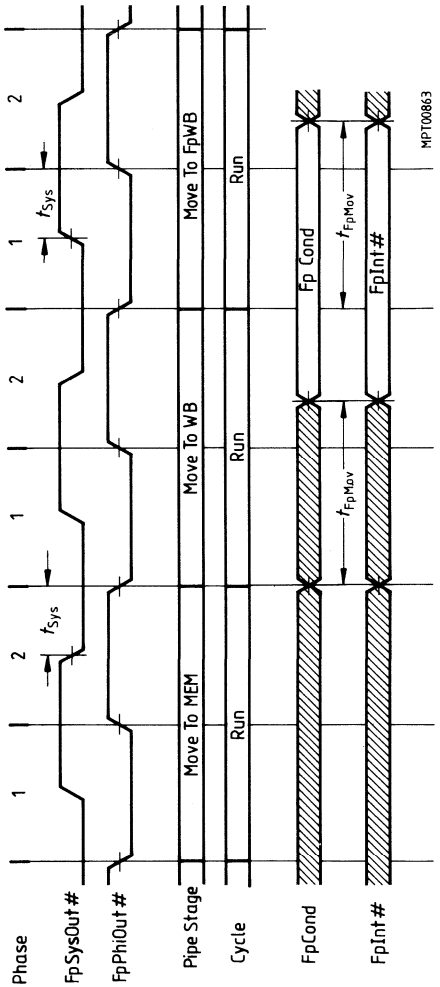


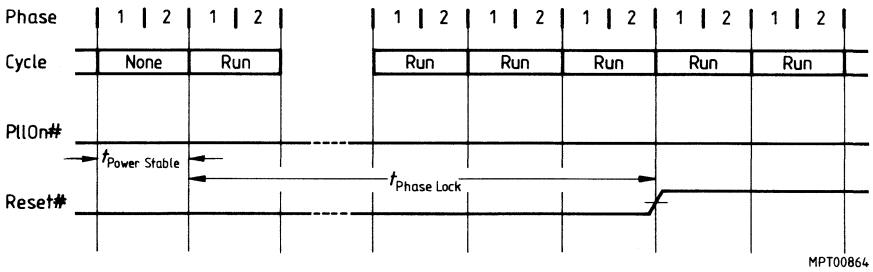
Figure 22
Move to FPA Status Timing

SAB-R3000 – FPA Timing Synchronization

The processor – coprocessor (SAB-R3000 – SAB-R3010) system requires that there be minimum timing skew between the SAB-R3000 and FPA to operate at maximum speed. To facilitate deskewing, a phase lock loop is provided on the FPA.

This synchronization must be achieved during the reset period to ensure that clock skew is acceptably small when the first instruction is fetched. During the reset period, the SAB-R3010's phase lock circuitry acquires and locks to the SAB-R3000's output clock. For correct operation, the CPU – FPA system must remain in reset for 3000 clocks or 200 microseconds after power is stable, whichever is longer. If the phase lock mechanism is not enabled, the reset period can be shortened to 128 clock cycles after power is stable. Figure 23 illustrates the required reset sequence for the case where the phase lock mechanism is enabled.

Figure 23
Reset Sequence



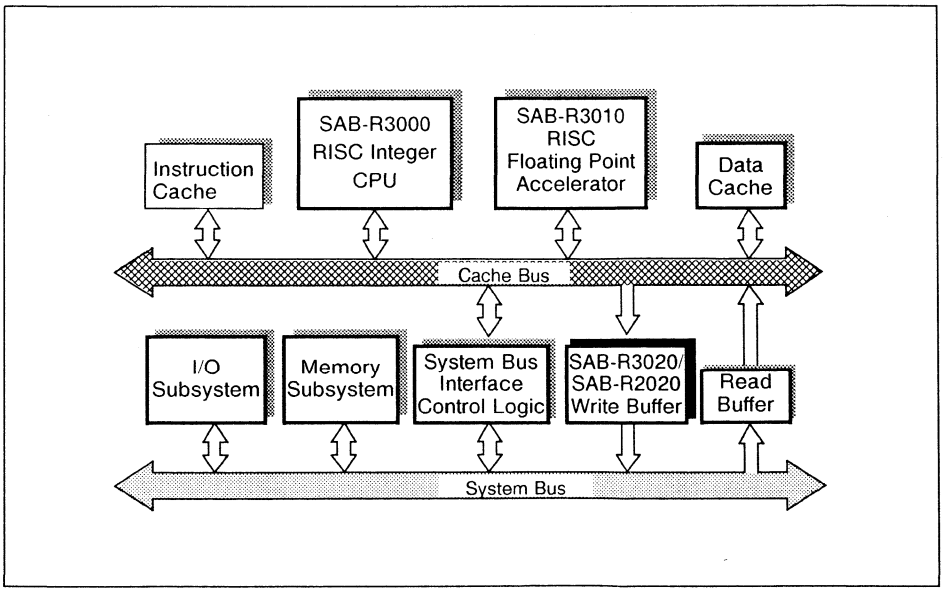
Note: PllOn# must be asserted continuously after Reset# is deasserted for correct operation of the SAB-R3010 – SAB-R3000 system.

SIEMENS

SAB-R3020/SAB-R2020
Write Buffer

Advance Information

- Temporary storage buffers to enhance the performance of the SAB-R3000 (SAB-R2000A) RISC microprocessor.
- Decouples the SAB-R3000 (SAB-R2000A) from slow main memory.
- Enables write operations from the SAB-R3000 (SAB-R2000A) to occur at processor speeds.
- Supports big endian and little endian byte-order addressing.
- Each Write Buffer has four locations to handle an 8-bit address slice and a 9-bit data slice (including a parity bit)
- Fully pin- and functionally compatible to all R3020/R2020 write buffers from other manufacturers.
- Plastic Package PL-CC-68



Ordering Information

Type	Ordering code	Package	Function
SAB-R2020-16-N	Q67120-C556	PL-CC-68	Write Buffer, 16.67 MHz
SAB-R3020-25-N	Q67120-C557	PL-CC-68	Write Buffer, 25 MHz

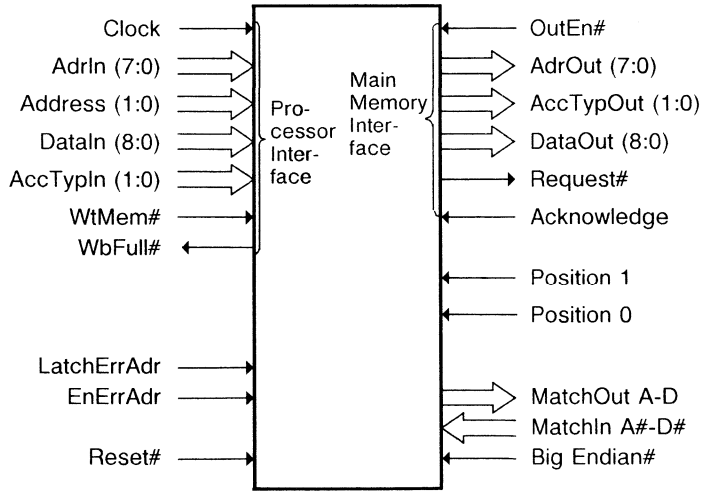
Introduction

The SAB-R3020/SAB-R2020 write buffer enhances performance of MIPS architecture based systems by allowing the processor to perform write operations during run cycles instead of stalling the pipeline. Each device handles an 8-bit slice of address and a 9-bit slice of data (one parity bit per byte). Four write buffers are used per system to provide four-deep buffering of 32 bits of address and 36 bits of data and parity.

Pin Names

Clock	System Clock
AdrIn (7:0)	Address Bus In
Address (1:0)	Byte Address
DataIn (8:0)	Data Bus In (includes one parity bit)
AccTypIn (1:0)	Access Type In
WtMem#	Main Memory Write
WbFull#	Write Buffer Full
LatchErrAdr	Latch Error Address
EnErrAdr	Enable Error Address
Reset#	Initializiation
OutEn#	Enable Tristate Outputs
AdrOut (7:0)	Address Bus Out
AccTypOut (1:0)	Access Type Out
DataOut (8:0)	Data Bus Out (including one parity bit)
Request#	Request access to Main Memory
Acknowledge	Acknowledge capture of Data
Position 1, Position 0	Position of Bytes
MatchOut A-D	Match Out to avoid Read/Write Conflicts
MatchIn A#-D#	Match In for Byte Gathering
Big Endian#	Big/Little-Endian Selection

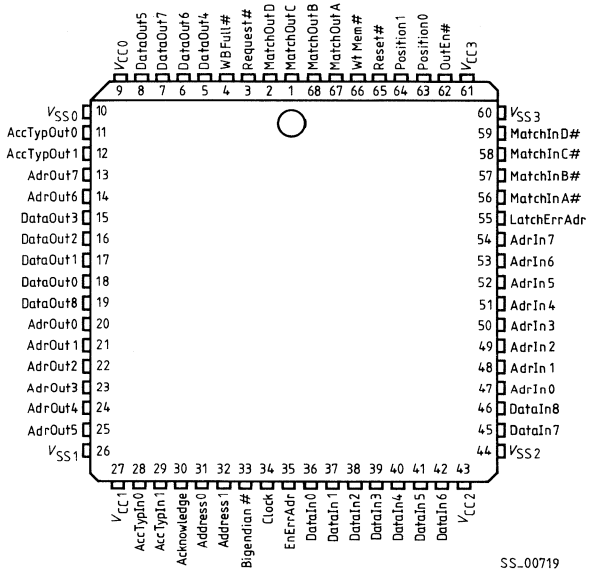
Figure 1
Logic Symbol



Note: "#" signifies an active low signal.

Pin Configuration

Figure 2
PL-CC-68 (Top View)



Pin Definitions and Functions

Symbol	Pin Number	Input (I) Output (O)	Function
Clock	34	I	An inverted version of the SysOut# signal from the CPU that synchronizes data transfers. The write buffer uses the trailing edge of Clock to latch the contents of the AdrLo bus, and the leading edge to latch the contents of the Data and Tag buses.
DataIn (8:0)	46, 45, 42, 41, 40, 39, 38, 37, 36	I	Nine input data lines from the processor's data Bus (eight bits of data and one bit of parity).
AdrIn (7:0)	54, 53, 52, 51, 50, 49, 48, 47	I	Eight input address lines from the CPU. The address lines are taken from the AdrLo and Tag buses.
Address (1:0)	32, 31	I	The least two significant address bits from the CPU. These two address bits must be connected to all four write buffers and are used in conjunction with the Access Type signals, the Position signals, and the BigEndian signal to determine which byte(s) in a word are being written into a particular write buffer.
AccTypIn (1:0)	29, 28	I	The access type signals from the CPU specifying the size of data transfer: word, tri-byte, half-word, or byte.
WtMem#	66	I	This input is connected to the MemWr# signal of the processor, which is asserted whenever the CPU performs a store operation.
Request#	3	O	This signal is used to request access to main memory. Request# may also be tied to the CpCond0 input of the CPU. Since Request# is asserted only when the write buffer contains data, software can determine if a previous write operation (for example, to an I/O device) has been completed before initiating a read to that device.

Pin Definitions and Functions (cont'd)

Symbol	Pin Number	Input (I) Output (O)	Function
WbFull#	4	O	The write buffer asserts this signal to the CPU WrBusy# input when it cannot accept more data. The processor performs a write busy stall if data must be stored while WbFull# is asserted.
AdrOut (7:0)	13, 14, 25, 24, 23, 22, 21, 20	O*)	Eight address lines output from each write buffer.
DataOut (8:0)	19, 7, 6, 8, 5, 15, 16, 17, 18	O*)	Nine output data lines from each write buffer (one bit parity).
AccTypOut (1:0)	12, 11	O*)	The access type signals from the write buffer specifying the size of a data access: word, tri-byte, half-word, or byte.
OutEn#	62	I	The memory controller asserts this write buffer input to enable the tri-state outputs of the write buffer address and data signals.
Acknowledge	30	I	The main memory system asserts this signal when it has captured the data presented by the write buffer on the DataOut lines.
Position1, Position0	64, 63	I	These signals (in conjunction with Big Endian#) determine how each write buffer decodes Address I/O and AccTyp I/O to ensure proper storage of the data inputs.
Big Endian#	33	I	When asserted, byte 0 is the leftmost, most significant byte (big-endian). When deasserted, byte 0 is the rightmost, least significant byte (little-endian).
MatchOutA-D	67, 68, 1, 2	O	Outputs to main memory controller logic used to resolve conflicts when the processor reads or writes to memory.

*) = Tri-state Output
= Active Low

Pin Definitions and Functions (cont'd)

Symbol	Pin Number	Input (I) Output (O)	Function
MatchInA#-D#	56, 57, 58, 59	I	These four active low inputs are used by internal byte gathering logic to combine bytes common to a word, thus increasing memory bandwidth.
EnErrAdr	35	I	When asserted, the contents of an internal error latch are output to the DataOut bus.
LatchErrAdr	55	I	When asserted, the value currently available to the address outputs of the write buffer is latched into an internal error latch.
Reset#	65	I	Used to initialize the write buffer to a known state and clear the contents of its registers.
V_{CC} (3:0)	61, 43, 27, 9	I	Power Supply
V_{SS} (3:0)	60, 44, 26, 10	I	Ground

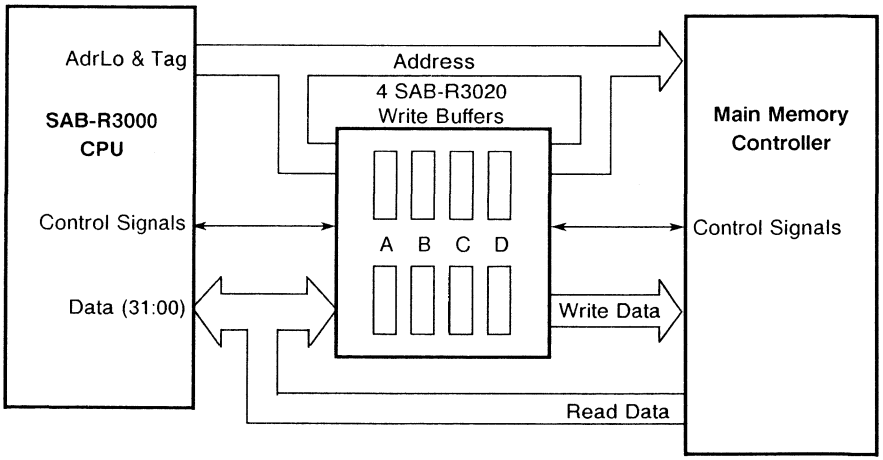
Operation

When the SAB-R3000 (SAB-R2000A) performs a write operation, the write buffer captures the output data and its address (including the access type bits). The write buffer can hold up to four data-address sets while it waits to pass the data on to main memory. Transfers from the processor to the write buffers occur synchronously at the cycle rate of the processor and the write buffer signals the processor if it is unable to accept data. The write buffer also provides a set of handshake signals to communicate with a main memory controller to coordinate the transfer of write data to main memory.

System Interface

Figure 3 shows the functional position of the write buffer in a SAB-R3000 (SAB-R2000A) based system.

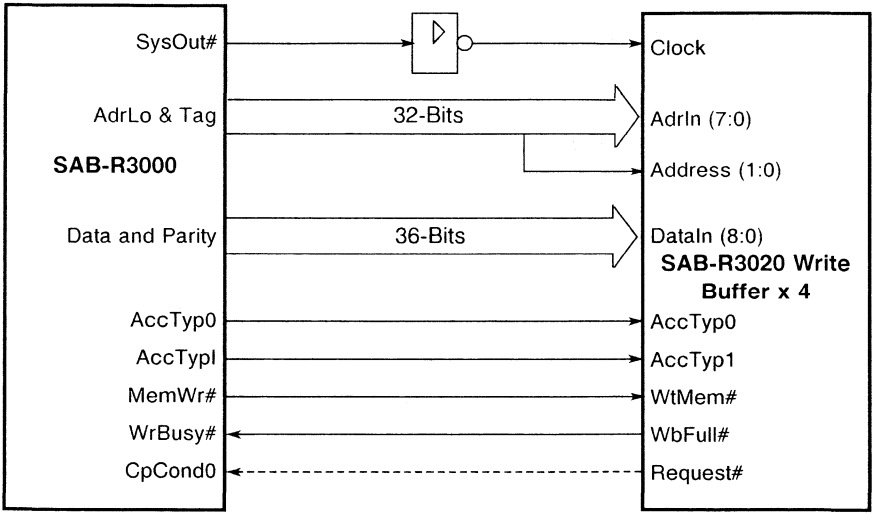
Figure 3
The SAB-R3020/SAB-R2020 in an SAB-R3000 (SAB-R2000A) System



Write Buffer-CPU Interface

Figure 4 details the interface between the CPU and the write buffer. The description assumes that four write buffers will be used to implement a 32-bit, buffered interface. The AdrLo bus and Tag bus bits from the CPU are both connected to form the 32-bit physical address that is captured by the write buffer. Thirty-two bits of data, four bits of parity, and two access type bits are also captured by the write buffer.

Figure 4
SAB-R3020/SAB-R3000 Interface (SAB-R2020/SAB-R2000A)



Data and Address Connections

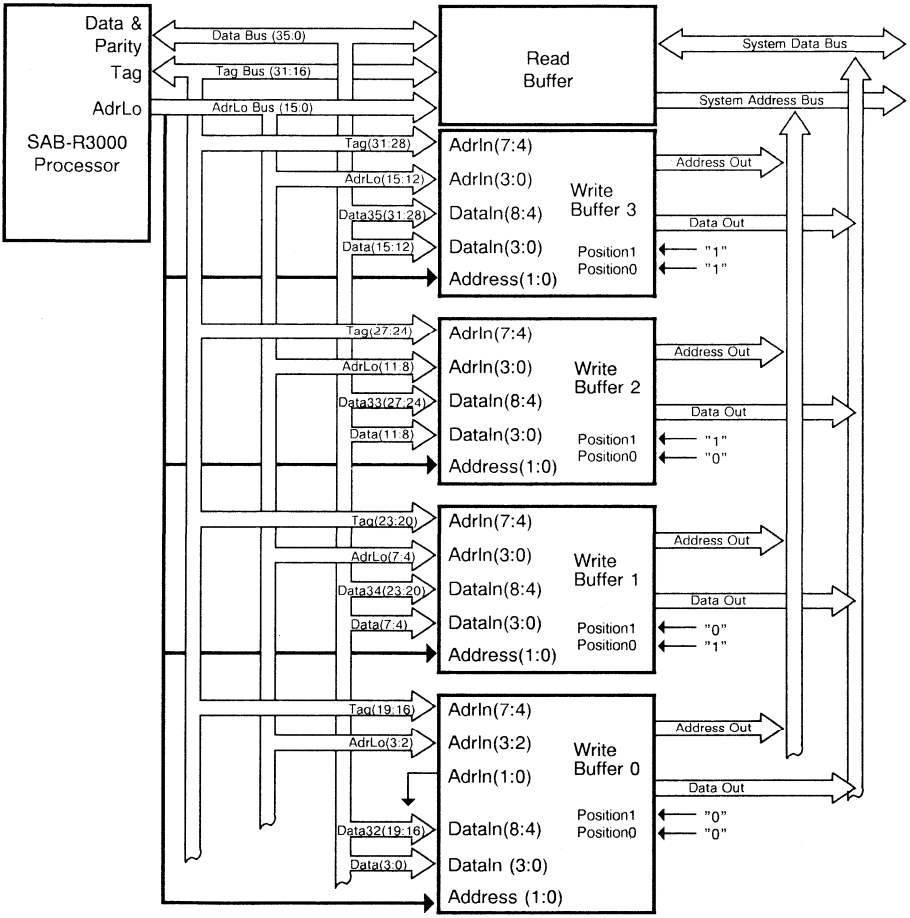
Figure 5 illustrates the address and data connection between four write buffers and the RISC CPU.

Each write buffer has eight address inputs (AdrIn7:0). The four low order bits (AdrIn3:0) are clocked into the device on the trailing edge of the Clock signal and taken from the SAB-R3000 (SAB-R2000A) AdrLo bus. The four high order bits (AdrIn7:4) are clocked into the device on the rising edge of the clock signal and taken from the SAB-R3000 (SAB-R2000A) Tag bus.

Each device also has separate inputs (Address1, Address0) for the two low order bits from the AdrLo bus. These bits act as a byte pointer within the write buffer. Note in figure 5 that the two low order AdrIn inputs (AdrIn1:0) to the write buffer device 0 are connected to ground since the Address1, Address0 inputs already supply these bits to the device.

The write buffer has nine data inputs that are clocked into the device on the leading edge of the clock signal and taken from the SAB-R3000 (SAB-R2000A) data bus. Note that the data bits assigned to each device correspond to the address bits. This arrangement is required since data selection is dependent on a combination of the AccTyp signals and the two low order address bits.

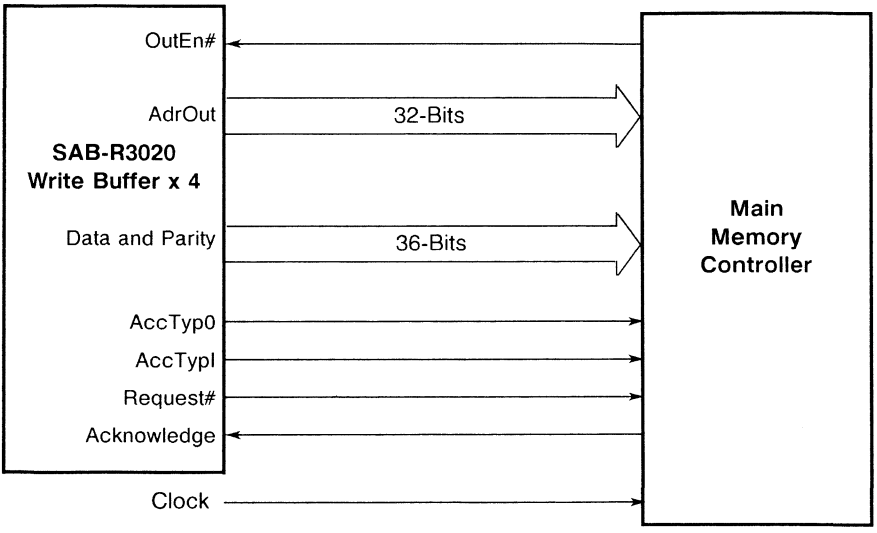
Figure 5
Write Buffer Data and Address Line Connections



Write Buffer-Main Memory Interface

Figure 6 shows the signals comprising the write buffer interface to main memory. The interface is essentially decoupled from the write buffer-processor interface, although some synchronization of the memory interface signals and the clock signal is required. The handshaking signals with main memory have no direct connection with the write buffer-processor interface.

Figure 6
Write Buffer/Main Memory Interface



Byte Gathering

The write buffers perform byte (half-word, tri-byte, and word) gathering to decrease the number of write transfers to the same word location. Sequential writes to the same word address have their data combined into the same address-data pair buffer.

Byte gathering is prohibited in the address-data pair that is currently available to the memory controller. Thus the first write into an empty write buffer will not have subsequent writes gathered into it. Writes to the same location (byte) will be overwritten in the write buffer if gathering is not prohibited by the preceding rule.

The write buffers present address-data pairs to the main memory in the sequence in which they were received from the processor except in the case of gathered data, where bytes or half-words can be collected and written to main memory in a single write operation.

If the address-data pair is scheduled to be output, then gathering is inhibited and the buffer contents are presented to the main memory controller. Subsequent writes are then placed in other buffers. No reliance should be placed on byte gathering as it is not readily deterministic. Non-sequential writes to the same address are not gathered.

In some cases gathering may require that two memory controller references be used to empty a single write buffer entry. For example, this may occur if bytes 0 and 3 of a word are sequentially written. Where order in writing is important, such as with I/O controllers, software should avoid sequential access to the same word. In cases where write-read access ordering is important but reading of the write locations is not desired (such as I/O), a write followed by a write to a dummy location will ensure that the first write has occurred before continuing. Alternatively, the REQUEST# signal can be tested to determine that the write buffer is empty.

Configuration Logic

Because of their byte gathering ability, each write buffer internally maintains a record of each valid byte in an address/data pair. To do this, each device must have a way of determining which data bits within a word it is handling. AccType0/1, Address0/1, BigEndian, and Position0/1 determine how the write buffers handle data when it is received. The table below shows the position of bytes within a word based on these signals.

Table 1
Position of Bytes within words

Access Type		Address		Bytes Accessed							
1	0	1	0	31	Big-Endian		0	31	Little-Endian		0
1	1	0	0	0	1	2	3	3	2	1	0
(word)											
1	0	0	0	0	1	2			2	1	0
		0	1		1	2	3	3	2	1	
(triple-byte)											
0	1	0	0	0	1					1	0
		1	0			2	3	3	2		
(halfword)											
0	0	0	0	0							0
		0	1		1					1	
		1	0			2			2		
(byte)											
		1	1				3	3			

Note also that the lower two address bits of the device in position zero (as determined by the two Position inputs) are inhibited. Instead on output, the lower two address bits are generated from the positions of the valid data bytes as determined by the above table.

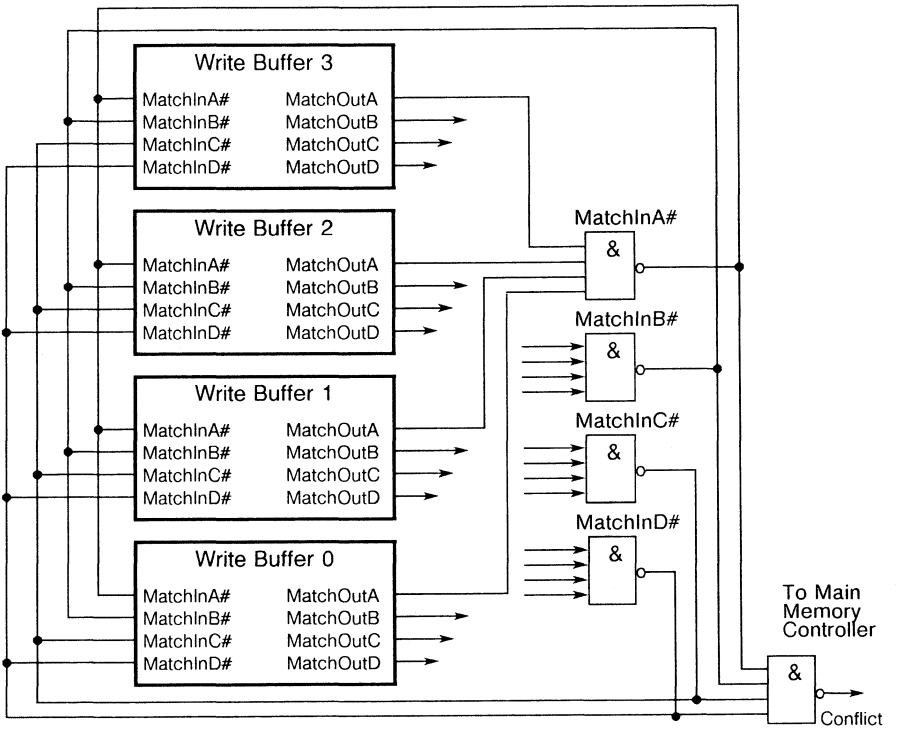
Conflict Resolution

When the SAB-R3000 (SAB-R2000A) references main memory (either a write or a read reference), the write buffers compare the word address from the CPU with the word address stored in the buffers. If any word address matches, the write buffers assert signals that can be used by the main memory controller to ensure that the write buffer is emptied before the read access with the conflicting address has been performed.

Figure 7 illustrates the write buffer signals involved in the address comparison logic. Each write buffer provides four output signals (MatchOutA, B, C, D) which correspond to the four buffer ranks in each device. These MatchOut signals can be externally NANDed to determine if the address being input matches those in any rank of the write buffer.

The outputs of the NAND gates are fed back into the write buffers via MatchA#, B#, C# and D# and are used within each device as part of the byte gathering logic. The NAND gate outputs can be NANDed together as shown with the resultant signal (in conjunction with the CPU's MEMRD signal) to alert the main memory controller logic that there is a pending buffered write that conflicts with a just issued read. The main memory controller can then delay the read access until the Request is deasserted indicating that the write buffer has been emptied.

Figure 7
Write Buffer MatchIn/MatchOut Logic

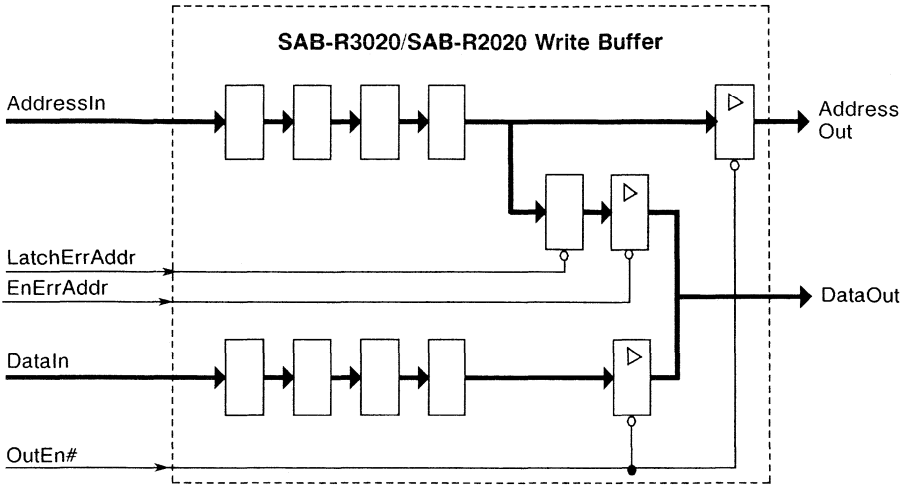


Error Address Latch

The write buffer incorporates an internal latch that can be loaded with one of the buffered addresses and subsequently enabled out onto the data lines. This feature can be used by error handling to read an address back from the write buffer and analyze, or recover from, certain bus errors. Figure 8 shows the signals involved in the operation of this address latch.

When the LatchErrAdr signal is asserted, the address currently available to the address outputs of the write buffer is latched into the internal latch. This address can then be output on the DataOut lines by asserting the EnErrAdr signal so that the processor can read the address in as data.

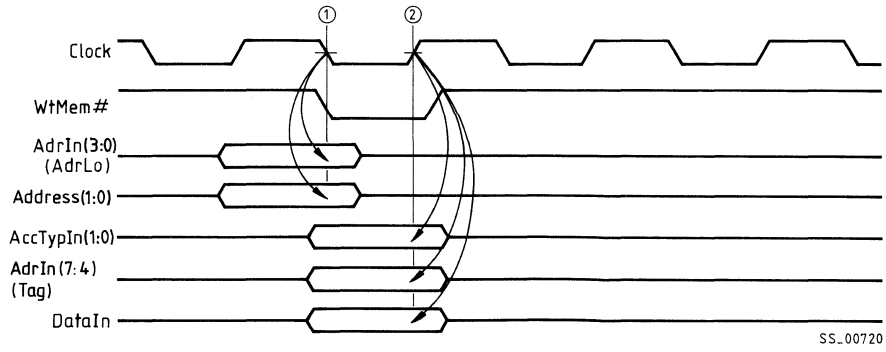
Figure 8
Write Buffer Error Address Latch



AC Timing

Transfers between the processor and the write buffers occur synchronously: the clock signal from the processor is input to the write buffers and used to clock the address and data information into the write buffer's latches. The relative timing is shown in figure 9.

Figure 9
AC Timing Diagram



When the WtMem# signal is asserted, the low-order address bits, the Address (1:0) inputs, and the access type inputs are latched on the trailing edge of the clock signal①. The rising edge of Clock② is used to latch the high-order address bits and the contents of the data bus.

Figure 10 illustrates the timing for transfer of data from the write buffer to the main memory system. The sequence is as follows:

① When the write buffer has a data-address pair for transfer to the memory system, it asserts the Request# signal.

② When the memory system is ready to handle the write buffer data, it asserts the OutEn# signal to enable the write buffers' address and data outputs onto the system bus.

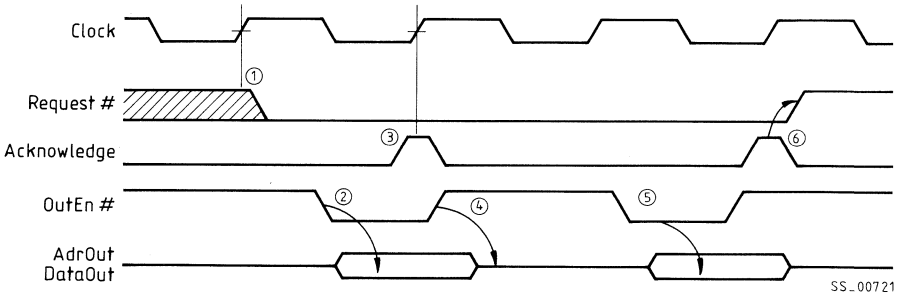
③ When the memory system no longer requires the write buffer address and data outputs, it asserts the Acknowledge signal. The write buffer responds to this signal by discarding the address-data pair that was just output.

④ The memory system can deassert the OutEn# signal to return the write buffers' address and data outputs to their tri-state condition.

⑤ Since the Request# signal remains asserted, the memory system asserts the OutEn# signal again to enable the next address-data pair onto the system buses.

⑥ When the memory system has accepted the second address-pair it again asserts the Acknowledge signal. If the write buffer is now empty, it responds to this signal by deasserting the Request# signal.

Figure 10
Write Buffer/Main Memory Interface Timing



Note that the write buffer's interface to main memory is not completely asynchronous; assertion of the Request# signal by the write buffer is synchronized with the rising edge of Clock, and the Acknowledge signal input by main memory has a minimum set up and hold time in relation to the Clock signal.

Absolute Maximum Ratings

Ambient temperature under bias (T_A) . . . . . 0 to +70 °C
Storage temperature (T_{ST}) . . . . . – 40 to +125 °C
Lead temperature (T_L) . . . . . 300 °C
Supply voltage (V_{CC}) . . . . . – 0.5 to +7.0 V
Input voltage (V_{IN}) . . . . . – 0.5 to +7.0 V

Note: *Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*
Not more than one output should be shorted at a time. Duration of the short should not exceed 30 seconds.

DC Characteristics

T_A = 0 to +70 °C; V_{CC} = 5 V ± 5 %

Parameter	Symbol	Limit values				Unit	Test condition
		SAB-R2020 16.67 MHz		SAB-R3020 25 MHz			
		min.	max.	min.	max.		
Operating Parameters							
Output HIGH voltage	V_{OH}	2.4	–	2.4	–	V	V_{CC} = min. I_{OH} = – 4 mA
Output LOW voltage	V_{OL}	–	0.4	–	0.4	V	V_{CC} = min. I_{OL} = 4 mA
Input HIGH voltage	V_{IH}	2	V_{CC} +0.25	2	V_{CC} +0.25	V	
Input LOW voltage	V_{IL}	– 0.5[1]	0.8	– 0.5[1]	0.8	V	
Input Leakage	I_{IN}	– 80	+80	– 80	+80	µA	V_{IN} = V_{DD} or GND
Output Leakage	I_{OZ}	– 40	+40	– 40	+40	µA	V_{OUT} = V_{DD} or GND
Operating current	I_{CC}	–	70	–	70	mA	V_{CC} = 5.25 V
Load capacitance	C_{Ld} [2]	–	50	–	50	pF	

1) V_{IL} min. = – 3.0 V for pulse width less than 15 ns
2) Operation above the C_{Ld} maximum may impair the useful life of the device.

AC Characteristics

$T_A = 0$ to $+70$ °C; $V_{CC} = 5$ V ± 5 %

Parameter	Symbol	Limit values				Unit
		SAB-R2020 16.67 MHz		SAB-R3020 25 MHz		
		min.	max.	min.	max.	
AdrIn (3:0) to clock falling setup	t_{01}	2	–	6	–	ns
AdrIn (3:0) from clock falling hold	t_{02}	5	–	4	–	ns
Address (1:0) to clock falling setup	t_{03}	5	–	6	–	ns
Address (1:0) from clock falling hold	t_{04}	4	–	6	–	ns
AccessType (1:0) to clock rising setup	t_{05}	4	–	6	–	ns
AccessType (1:0) from clock rising hold	t_{06}	3	–	4	–	ns
AdrIn (7:4) to clock rising setup	t_{07}	5	–	4	–	ns
AdrIn (7:4) from clock rising hold	t_{08}	3	–	3	–	ns
DataIn (8:0) to clock rising setup	t_{09}	4	–	4	–	ns
DataIn (8:0) from clock rising hold	t_{10}	3	–	3	–	ns
WtMem# to clock rising setup	t_{11}	10	–	7	–	ns
WtMem# from clock rising hold	t_{12}	5	–	4	–	ns
Request from clock rising	t_{13}	–	25	–	22	ns
Acknowledge to clock rising setup	t_{14}	10	–	6	–	ns
Acknowledge from clock rising hold	t_{15}	6	–	6	–	ns
LatchErrAdr to clock rising setup	t_{16}	2	–	5	–	ns
WbFull# active from clock rising	t_{17}	–	21	–	17.3	ns
WbFull# inactive from clock rising	t_{18}	–	21	–	11	ns
OutEn to AdrOut (7:0), DataOut(8:0) valid	t_{19}	–	16	–	16	ns
OutEn to AdrOut (7:0), DataOut(8:0) Tri-State	t_{20}	–	15	–	15	ns
MatchOut (A:D) from clock rising	t_{21}	–	24	–	22	ns
MatchIn (A#:D#) to clock rising setup	t_{22}	10	–	10	–	ns
MatchIn (A#:D#) from clock rising hold	t_{23}	0	–	3	–	ns
EnErrAdr to data (error latch) valid	t_{24}	–	16	–	23	ns

AC Characteristics (cont'd)

Parameter	Symbol	Limit values				Unit
		SAB-R2020 16.67 MHz		SAB-R3020 25 MHz		
		min.	max.	min.	max.	
EnErrAdr to data (error latch) Tri-state	t_{25}	–	11	–	15	ns
Address/DataOut from clock rising	t_{26}	–	30	–	34	ns
Reset# to clock rising setup	t_{27}	10	–	10	–	ns
Reset# from clock rising hold	t_{28}	1	–	1	–	ns
Reset# low pulse width	t_{29}	8	–	8	–	ns
WbFull# high from Clk Rsg (after Reset#)	t_{30}	–	14	–	20	ns
Request# high from Reset# Low	t_{31}	–	17	–	18	ns
AccTypIn (1:0) low from Reset# Low	t_{32}	–	21	–	25	ns
MatchOut (A:D) low from Reset# Low	t_{33}	–	19	–	20	ns
Address/DataOut from Reset# Low	t_{34}	–	32	–	27	ns
AccTypeOut (0:1) from clock rising	t_{35}	–	32	–	27	ns

Figure 11
Write Buffer Timing Specifications

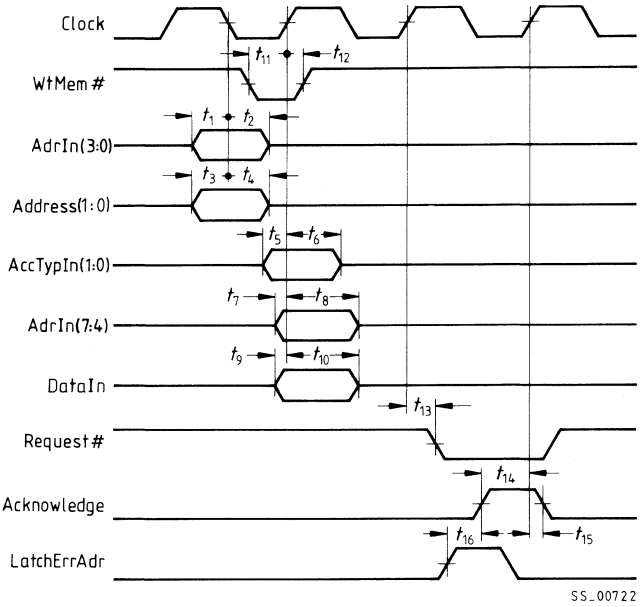


Figure 12
WbFull# Timing Specification

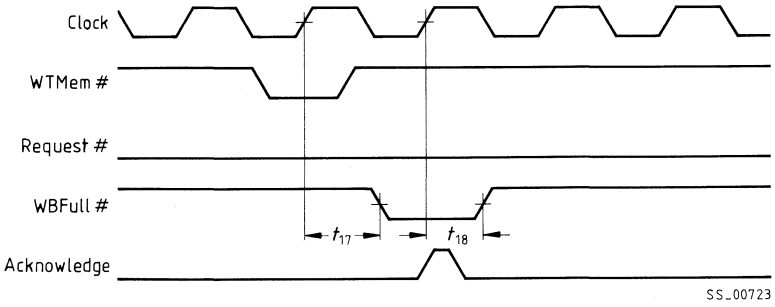


Figure 13
OutEn# Timing Specification

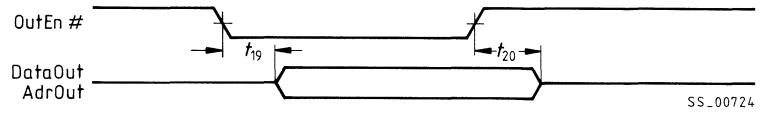


Figure 14
Match and Error Latch Timing Specifications

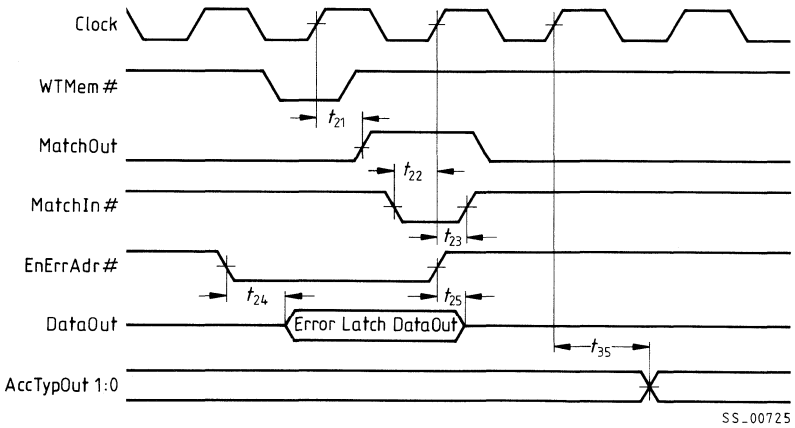


Figure 15
Address/DataOut Timing Specification

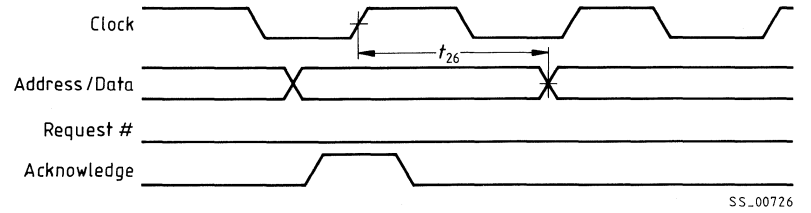
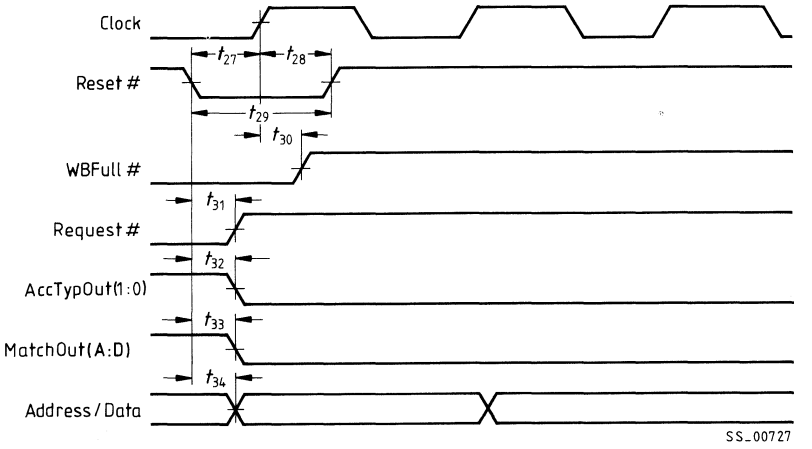


Figure 16
Timing Specifications



Support Components

SIEMENS

Preliminary

SAB 82284 Clock Generator and Ready Interface for SAB 80286 Processors

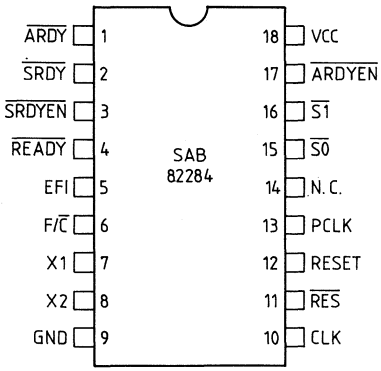
SAB 82284 up to 16 MHz

- Generates system clock for SAB 80286 processors
- Uses crystal or TTL signal for frequency source
- Provides local $\overline{\text{READY}}$ and multimaster system bus $\overline{\text{READY}}$ synchronization

SAB 82284-1 up to 20 MHz

- 18-pin package
- Single +5 V power supply
- Generates system reset output from Schmitt-trigger input

Pin Configuration



Pin Names

CLK	System Clock
F/$\overline{\text{C}}$	Frequency/Crystal Select
X1, X2	Crystal In
EFI	External Frequency In
PCLK	Peripheral Clock
$\overline{\text{ARDYEN}}$	Asynchronous Ready Enable
$\overline{\text{ARDY}}$	Asynchronous Ready
$\overline{\text{SRDYEN}}$	Synchronous Ready Enable
$\overline{\text{SRDY}}$	Synchronous Ready
$\overline{\text{READY}}$	Bus Cycle Termination
$\overline{\text{S0}}$, $\overline{\text{S1}}$	Status
RESET	Reset
$\overline{\text{RES}}$	Reset In
VCC	Power supply (+5V)
GND	Ground (0V)

The SAB 82284 is a bipolar clock generator/driver which provides clock signals for SAB 80286 processors and support components. It also contains logic to supply $\overline{\text{READY}}$ to the CPU from either asynchronous or synchronous sources and synchronous RESET from an asynchronous input with hysteresis.

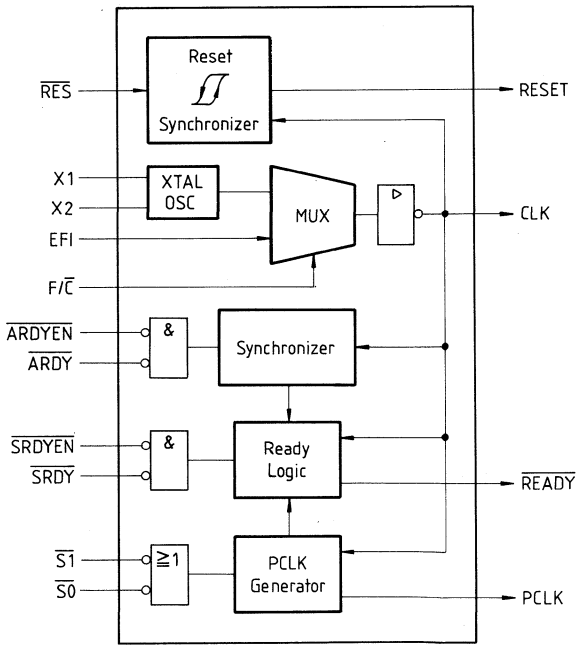
Pin Definitions and Functions

Symbol	Pin	Input (I) Output (O)	Function
$\overline{ARDY}$	1	I	ASYNCHRONOUS READY is an active low input used to terminate the current bus cycle. The $\overline{ARDY}$ input is qualified by $\overline{ARDYEN}$. Inputs to ARDY may be applied asynchronously to CLK. Setup and hold times are given to assure a guaranteed response to synchronous inputs.
$\overline{SRDY}$	2	I	SYNCHRONOUS READY is an active low input used to terminate the current bus cycle. The $\overline{SRDY}$ input is qualified by the $\overline{SRDYEN}$ input. Setup and hold times must be satisfied for proper operation.
$\overline{SRDYEN}$	3	I	SYNCHRONOUS READY ENABLE is an active low input which qualifies $\overline{SRDY}$. $\overline{SRDYEN}$ selects $\overline{SRDY}$ as the source for $\overline{READY}$ to the CPU for the current bus cycle. Setup and hold times must be satisfied for proper operation.
$\overline{READY}$	4	O	READY is an active low output which signals the current bus cycle is to be completed. The $\overline{SRDY}$, $\overline{SRDYEN}$, $\overline{ARDY}$, $\overline{ARDYEN}$, $\overline{S1}$, $\overline{S0}$ and $\overline{RES}$ inputs control $\overline{READY}$ as explained later in the $\overline{READY}$ generator section. $\overline{READY}$ is an open collector output requiring an external pullup resistor.
EFI	5	I	EXTERNAL FREQUENCY IN drives CLK when the F/$\overline{C}$ input is strapped high. The EFI input frequency must be twice the desired internal processor clock frequency.
F/$\overline{C}$	6	I	FREQUENCY/CRYSTAL SELECT is a strapping option to select the source for the CLK output. When F/$\overline{C}$ is strapped low, the internal crystal oscillator drives CLK. When F/$\overline{C}$ is strapped high, the EFI input drives the CLK output.
X1, X2	7, 8	I	CRYSTAL IN are the pins to which a parallel resonant fundamental mode crystal is attached for the internal oscillator. When F/$\overline{C}$ is low, the internal oscillator will drive the CLK output at the crystal frequency. The crystal frequency must be twice the desired internal processor clock frequency.
CLK	10	O	SYSTEM CLOCK is the signal used by the processor and support devices which must be synchronous with the processor. The frequency of the CLK output has twice the desired internal processor clock frequency. CLK can drive both TTL and MOS level inputs.
$\overline{RES}$	11	I	RESET IN is an active low input which generates the system reset signal RESET. Signals to $\overline{RES}$ may be applied asynchronously to CLK. A Schmitt-trigger input is provided on $\overline{RES}$, so that an RC circuit can be used to provide a time delay. Setup and hold times are given to assure a guaranteed response to synchronous inputs.
RESET	12	O	RESET is an active high output which is derived from the $\overline{RES}$ input. RESET is used to force the system into an initial state. When RESET is active, $\overline{READY}$ will be active (low).

Pin Definitions and Functions (cont'd)

Symbol	Pin	Input (I) Output (O)	Function
PCLK	13	O	PERIPHERAL CLOCK is an output which provides a 50% duty cycle clock with 1/2 the frequency of CLK. PCLK will be in phase with the internal processor clock following the first bus cycle after the processor has been reset.
$\overline{S0}$, $\overline{S1}$	15, 16	I	STATUS inputs prepare the SAB 82284 for a subsequent bus cycle. $\overline{S0}$ and $\overline{S1}$ synchronize PCLK to the internal processor clock and control $\overline{READY}$. These inputs have pullup resistors to keep them high if nothing is driving them. Setup and hold times must be satisfied for proper operation.
$\overline{ARDYEN}$	17	I	ASYNCHRONOUS READY ENABLE is an active low input which qualifies the $\overline{ARDY}$ input. $\overline{ARDYEN}$ selects $\overline{ARDY}$ as the source of ready for the current bus cycle. Inputs to $\overline{ARDYEN}$ may be applied asynchronously to CLK. Setup and hold times are given to assure a guaranteed response to synchronous inputs.
VCC	18	–	POWER SUPPLY (+5 V)
GND	9	–	GROUND (0 V)

Block Diagram



Functional Description

Introduction

The SAB 82284 generates the clock, ready, and reset signals required for SAB 80286 processors and support components. The SAB 82284 is packaged in an 18-pin DIP package and contains a crystal-controlled oscillator, MOS clock generator, peripheral clock generator, Multibus ready synchronization logic, and system reset generation logic.

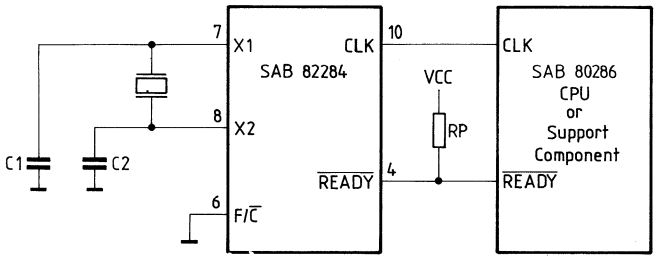
Clock generator

The CLK output provides the basic timing control for an SAB 80286 system. CLK has output characteristics sufficient to drive MOS devices. CLK is generated by either an internal crystal oscillator or an external source as selected by the F/$\overline{C}$ strapping option. When F/$\overline{C}$ is low, the crystal oscillator drives the CLK output. When F/$\overline{C}$ is high, the EFI input drives the CLK output. The SAB 82284 provides a second clock output (PCLK) for peripheral devices. PCLK is CLK divided by two. PCLK has a duty cycle of 50% and TTL output drive characteristics. PCLK is normally synchronized to the internal processor clock. After reset, the PCLK signal may be out of phase with the internal processor clock. The $\overline{S1}$ and $\overline{S0}$ signals of the first bus cycle are used to synchronize PCLK to the internal processor clock. The phase of the PCLK output changes by extending its high time beyond one system clock (see waveforms). PCLK is forced high whenever either $\overline{S0}$ or $\overline{S1}$ were active (low) for the two previous CLK cycles. PCLK continues to oscillate when both $\overline{S0}$ and $\overline{S1}$ are high.
Since the phase of the internal processor clock will not change except during reset, the phase of PCLK will not change except during the first bus cycle after reset.

Oscillator

The oscillator circuit of the SAB 82284 is a linear Pierce oscillator which requires an external, parallel, resonant, fundamental-mode crystal. The output of the oscillator is internally buffered. The crystal frequency chosen should be twice the required internal processor clock frequency. The crystal should have a typical load capacitance of 32 pF.
X1 and X2 are the oscillator crystal connections. For stable operation of the oscillator, two loading capacitors are recommended, as shown in the figure below. The sum of the board capacitance and loading capacitance should equal the values shown. It is advisable to limit stray board capacitances (not including the effect of the loading capacitors or crystal capacitance) to less than 10pF between the X1 and X2 pins.
Decouple VCC and GND as close to the SAB 82284 as possible.

Recommended Crystal and $\overline{\text{READY}}$ Connections
(for RP see note 6 of ac characteristics)



Crystal Loading Table

Crystal Frequency	C1 Capacitance	C2 Capacitance
2 to 8MHz	60pF	40pF
8 to 20MHz	25pF	15pF

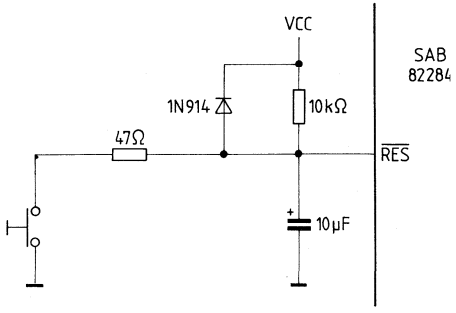
Reset Operation

The reset logic provides the RESET output to force the system into a known initial state. When the $\overline{RES}$ input is active (low), the RESET output becomes active (high). $\overline{RES}$ is synchronized internally at the falling edge of CLK before generating the RESET output (see waveforms). Synchronization of the $\overline{RES}$ input introduces a one or two CLK delay before affecting the RESET output.
At power up, a system does not have a stable VCC and CLK. To prevent spurious activity, $\overline{RES}$ should be asserted until VCC and CLK stabilize at their operating values. SAB 80286 processors and support components also require their RESET inputs be high a minimum number of CLK cycles. An RC network, as shown below, will keep $\overline{RES}$ low long enough to satisfy both needs.
A Schmitt-trigger input with hysteresis on $\overline{RES}$ assures a single transition of RESET with an RC circuit on $\overline{RES}$. The hysteresis separates the input voltage level at which the circuit output switches from high to low from the input voltage level at which the circuit output switches from low to high. The $\overline{RES}$ high to low input transition voltage is lower than the $\overline{RES}$ low to high input transition voltage. As long as the slope of the $\overline{RES}$ input voltage remains in the same direction (increasing or decreasing) around the $\overline{RES}$ input transition voltage, the RESET output will make a single transition.

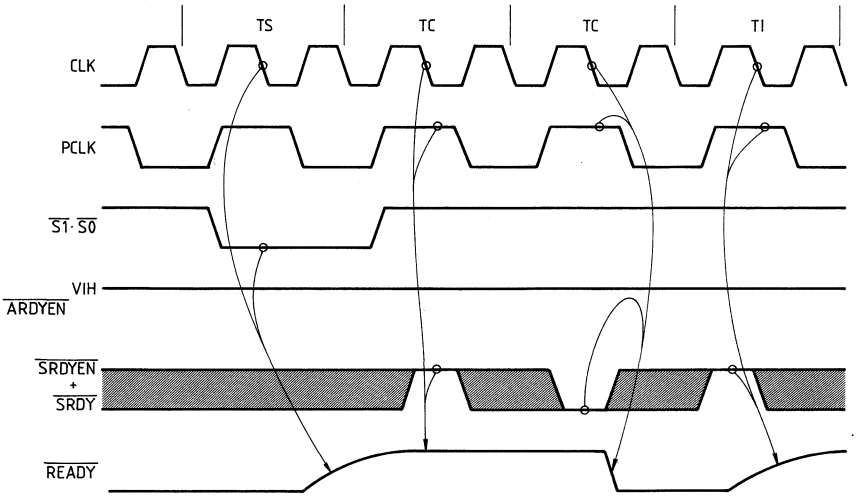
Ready Operation

The SAB 82284 accepts two ready sources for the system ready signal which terminates the current bus cycle. Either a synchronous ($\overline{SRDY}$) or asynchronous ready ($\overline{ARDY}$) source may be used. Each ready input has an enable ($\overline{SRDYEN}$ and $\overline{ARDYEN}$) for selecting the type of ready source required to terminate the current bus cycle. An address decoder would normally select one of the enable inputs.
The figure on synchronous ready mode illustrates the operation of $\overline{SRDY}$ and $\overline{SRDYEN}$. These inputs are sampled on the falling edge of CLK when $\overline{S1}$ and $\overline{S0}$ are inactive and PCLK is high. $\overline{READY}$ is forced active when both $\overline{SRDY}$ and $\overline{SRDYEN}$ are sampled as low.
The figure on asynchronous ready mode shows the operation of $\overline{ARDY}$ and $\overline{ARDYEN}$. These inputs are sampled by an internal synchronizer at each falling edge of CLK. The output of the synchronizer is then sampled when PCLK is high. If the synchronizer resolved both the $\overline{ARDY}$ and $\overline{ARDYEN}$ inputs to have been active (low), $\overline{READY}$ becomes active (low) and the $\overline{SRDY}$ and $\overline{SRDYEN}$ inputs are ignored.
$\overline{READY}$ remains active until either $\overline{S1}$ or $\overline{S0}$ is sampled low, or the ready inputs are sampled as inactive.
$\overline{READY}$ is enabled (low), if either $\overline{SRDY}$ + $\overline{SRDYEN}$ = 0 or $\overline{ARDY}$ + $\overline{ARDYEN}$ = 0 when sampled by the SAB 82284 $\overline{READY}$ generation logic. $\overline{READY}$ will remain active for at least two CLK cycles.
The $\overline{READY}$ output has an open-collector driver allowing other ready circuits to be wire-ORed with it. The $\overline{READY}$ signal of an SAB 80286 system requires an external pullup resistor (see Note 6 of AC Characteristics). To force the $\overline{READY}$ signal inactive (high) at the start of a bus cycle, the $\overline{READY}$ output floats when either $\overline{S1}$ or $\overline{S0}$ are sampled low at the falling edge of CLK. Two system clock periods are allowed for the pullup resistor to pull the $\overline{READY}$ signal to VIH. When RESET is active, $\overline{READY}$ is forced active one CLK later (see waveforms).

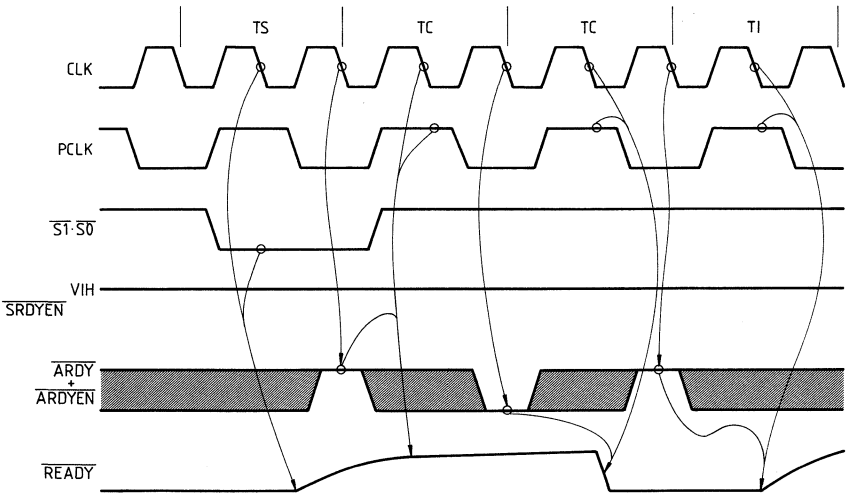
Typical RC $\overline{RES}$ Timing Circuit



Synchronous Ready Operation



Asynchronous Ready Operation



Absolute Maximum Ratings [1)]

Temperature under bias	0 to 70°C
Storage temperature	−65 to +150°C
All output and supply voltages	−0.5 to +7V
All input voltages	−1.0 to +5.5V
Power dissipation	1 W

DC Characteristics

TA = 0 to 70°C, VCC = 5 V ±10%

Symbol	Parameter	Limit values		Unit	Test condition
		min.	max.		
IF	Forward input current	–	−0.5	mA	VF = 0.45V
IR	Reverse input current	–	50	µA	VR = VCC
VC	Input forward clamp voltage	–	−1.0	V	IC = −5mA
ICC	Power supply current	–	145	mA	–
VIL	Input low voltage	–	0.8	V	–
VIH	Input high voltage	2.0	–	V	–
VOL, VCL	Output low voltage	–	0.45	V	IOL = 5mA (8.5 mA at $\overline{\text{READY}}$)
VCH	CLK output high voltage	4.0	–	V	IOH = −1mA
VOH	Output high voltage	2.4	–	V	IOH = −1mA
VIHR	$\overline{\text{RES}}$ input high voltage	2.6	–	V	–
VIHR – VILR	$\overline{\text{RES}}$ input hysteresis	0.25	–	V	–
CI	Input capacitance	–	10	pF	fC = 1 MHz

[1)] Stresses above those listed under "absolute maximum ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

AC Characteristics SAB 82284

TA = 0 to 70°C, VCC = 5V ±10%
AC timings are referenced to 0.8 and 2.0V points of signals as illustrated in data sheet waveforms, unless otherwise noted.

Symbol	Parameter	Limit values		Unit	Test condition
		min.	max.		
T1	EFI to CLK delay	–	30	ns	at 1.5V 1)
T2	EFI low time	22	–	ns	at 1.5 V 1) 2)
T3	EFI high time	30	–	ns	at 1.5V 1) 2)
T4	CLK period	62	500	ns	–
T5	CLK low time	15	–	ns	at 1.0 V 1) 2) 3) 4)
T6	CLK high time	25	–	ns	at 3.6 V 1) 2) 3) 4)
T7	CLK rise time	–	10	ns	from 1.0 V to 3.6 V 1)
T8	CLK fall time	–	10	ns	from 3.6 V to 1.0 V 1)
T9	Status setup time	22.5	–	ns	1)
T10	Status hold time	1	–	ns	1)
T11	$\overline{\text{SRDY}}$ + $\overline{\text{SRDYEN}}$ setup time	15	–	ns	1)
T12	$\overline{\text{SRDY}}$ + $\overline{\text{SRDYEN}}$ hold time	0	–	ns	1)
T13	$\overline{\text{ARDY}}$ + $\overline{\text{ARDYEN}}$ setup time	0	–	ns	1) 5)
T14	$\overline{\text{ARDY}}$ + $\overline{\text{ARDYEN}}$ hold time	30	–	ns	1) 5)
T15	$\overline{\text{RES}}$ setup time	20	–	ns	1) 5)
T16	$\overline{\text{RES}}$ hold time	10	–	ns	1) 5)
T17	$\overline{\text{READY}}$ inactive delay	5	–	ns	at 0.8 V 6)
T18	$\overline{\text{READY}}$ active delay	0	24	ns	at 0.8 V 6)
T19	PCLK delay	0	45	ns	7)
T20	RESET delay	5	34	ns	7)
T21	PCLK low time	T4–20	–	ns	at 0.6 V 7) 8)
T22	PCLK high time	T4–20	–	ns	at 2.0 V 7) 8)

For notes refer to page 10.

AC Characteristics SAB 82284-1

TA = 0 to 70°C, VCC = 5V ±10%
AC timings are referenced to 0.8 and 2.0V points of signals as illustrated in data sheet waveforms, unless otherwise noted.

Symbol	Parameter	Limit values		Unit	Test condition
		min.	max.		
T1	EFI to CLK delay	–	30	ns	at 1.5V [1]
T2	EFI low time	25	–	ns	at 1.5V [1] [2]
T3	EFI high time	25	–	ns	at 1.5V [1] [2]
T4	CLK period	50	500	ns	–
T5	CLK low time	12	–	ns	at 1.0V [1] [2] [3] [4]
T6	CLK high time	16	–	ns	at 3.6V [1] [2] [3] [4]
T7	CLK rise time	–	8	ns	from 1.0V to 3.6V [1]
T8	CLK fall time	–	8	ns	from 3.6V to 1.0V [1]
T9	Status setup time	20	–	ns	[1]
T10	Status hold time	1	–	ns	[1]
T11	$\overline{\text{SRDY}}$ + $\overline{\text{SRDYEN}}$ setup time	15	–	ns	[1]
T12	$\overline{\text{SRDY}}$ + $\overline{\text{SRDYEN}}$ hold time	0	–	ns	[1]
T13	$\overline{\text{ARDY}}$ + $\overline{\text{ARDYEN}}$ setup time	0	–	ns	[1] [5]
T14	$\overline{\text{ARDY}}$ + $\overline{\text{ARDYEN}}$ hold time	30	–	ns	[1] [5]
T15	$\overline{\text{RES}}$ setup time	20	–	ns	[1] [5]
T16	$\overline{\text{RES}}$ hold time	10	–	ns	[1] [5]
T17	$\overline{\text{READY}}$ inactive delay	5	–	ns	at 0.8V [6]
T18	$\overline{\text{READY}}$ active delay	0	24	ns	at 0.8V [6]
T19	PCLK delay	0	35	ns	[7]
T20	RESET delay	5	27	ns	[7]
T21	PCLK low time	T4–20	–	ns	at 0.6V [7] [8]
T22	PCLK high time	T4–20	–	ns	at 2.0V [7] [8]

For notes refer to page 10.

Notes referring to AC Characteristics:

1) CLK loading: CL = 150 pF.
The SAB 82284's X1 and X2 inputs are designed primarily for parallel resonant crystals. Serial resonant crystals may oscillate up to 0.01% faster than their rated frequencies, when used with the SAB 82284. For either type capacitive loading should be according to the crystal loading table.

2) At CLK frequencies above 12 MHz, CLK high and low times are guaranteed only when using a crystal with recommended capacitive loading (see table), not when driving the component from EFI input.

3) With either the internal oscillator and recommended crystal and load or with the EFI input meeting specifications T2 and T3. The values from the crystal loading table are ± 5 pF and include all stray capacitances. Decouple VCC and GND as close to the SAB 82284 as possible.

4) When using a crystal (with recommended load) appropriate for speed of the SAB 80286, CLK output low and high times are guaranteed to meet the SAB 80286 requirements.

5) This is an asynchronous input. The specification is given for testing purposes only, to assure recognition at a specific clock edge.

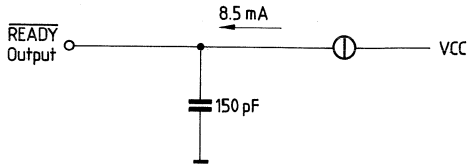
6) $\overline{\text{READY}}$ loading: CL = 150 pF, pullup resistor RP, with RP = 910 Ω.

7) PCLK and RESET loading: CL = 75 pF. PCLK output with 750 Ω pullup resistor.

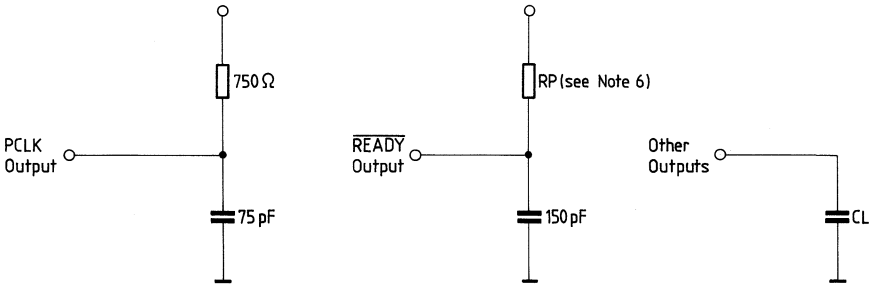
8) T4 refers to any allowable CLK period.

Testing Waveforms

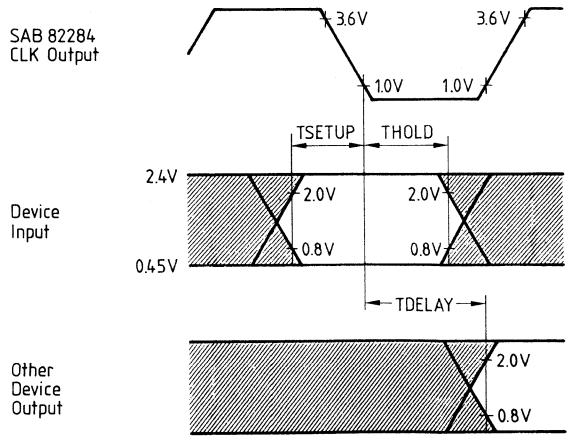
DC Test Loadings



AC Test Loadings

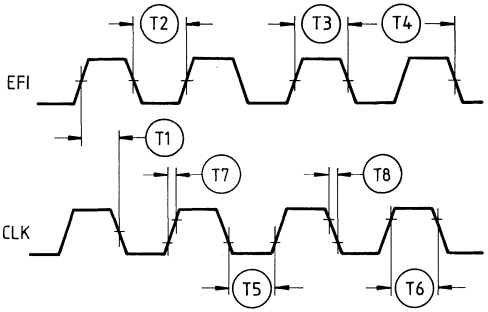


Setup, Hold and Delay Time Measurement – General



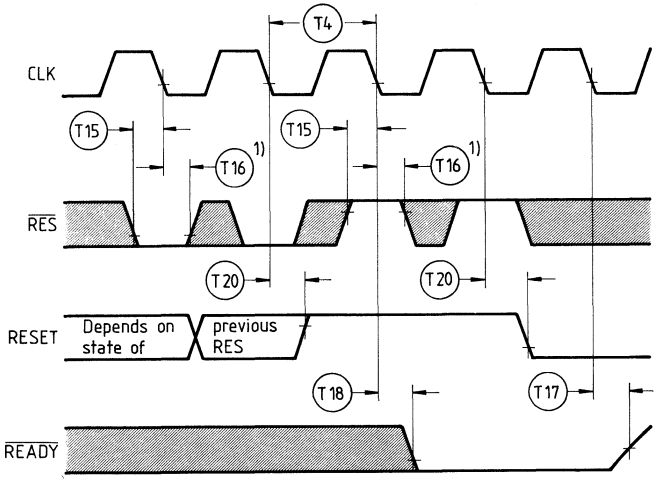
Waveforms

CLK versus EFI



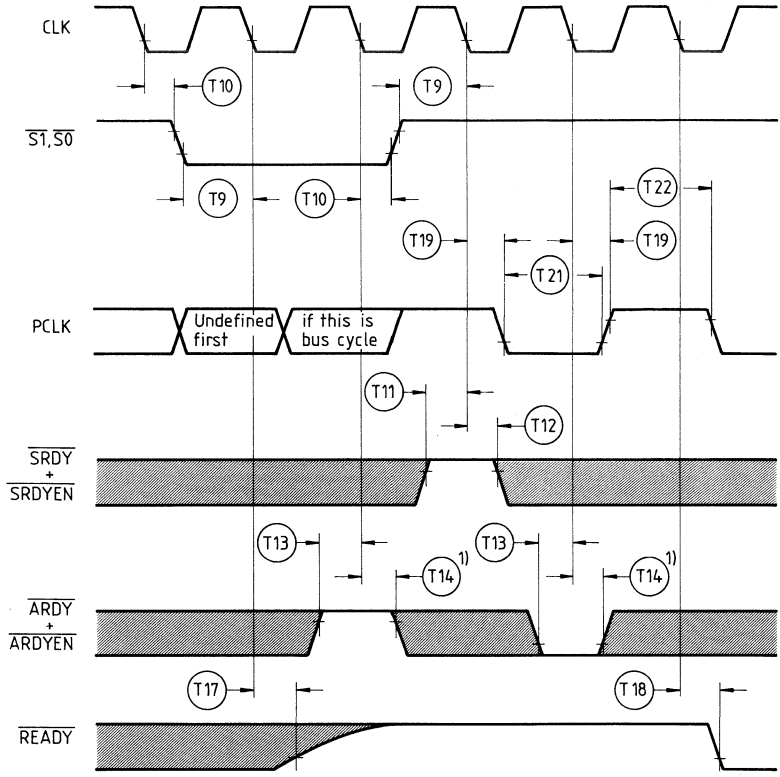
The EFI input low and high times as shown are required to guarantee the CLK low and high times shown.

RESET and $\overline{\text{READY}}$ Timing versus $\overline{\text{RES}}$ with $\overline{\text{S1}}$ and $\overline{\text{S0}}$ High



[1] This is an asynchronous input. The setup and hold times shown are required to guarantee the response shown.
[2] One or two CLK cycles depending on PCLK phase.

$\overline{\text{READY}}$ and PCLK Timing with $\overline{\text{RES}}$ High



1) This is an asynchronous input. The setup and hold times shown are required to guarantee the response shown.

Ordering Information

Type	Description	Ordering code
SAB 82284-P	Clock generator (plastic package) up to 16 MHz	Q67020-Y162
SAB 82284-1-P	Clock generator (plastic package) up to 20 MHz	Q67020-Y167

SIEMENS

Bus Controller for SAB 80286 Processors

SAB 82288

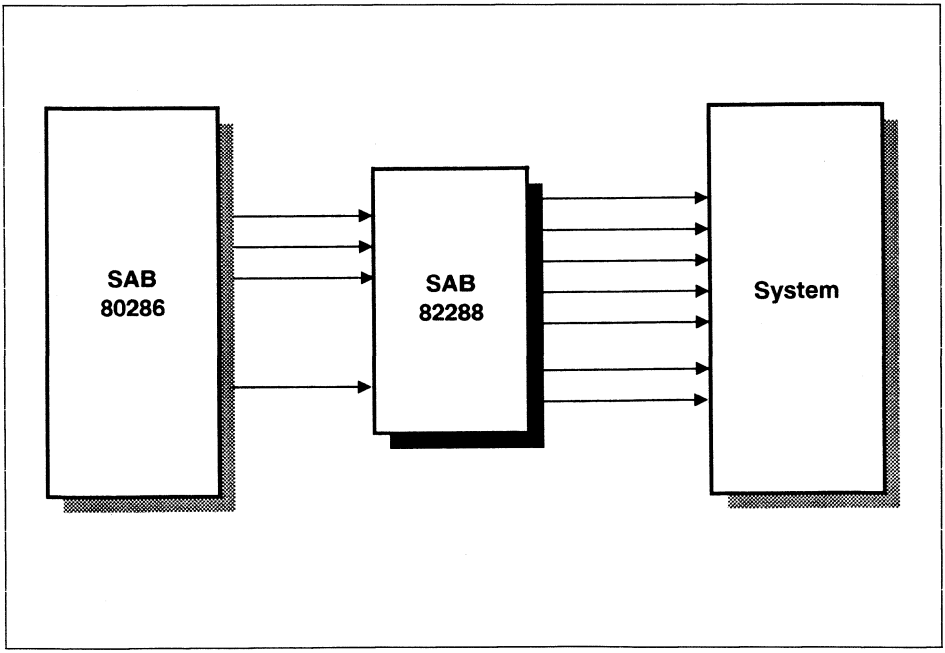
Preliminary

SAB 82288-6 up to 12 MHz
SAB 82288 up to 16 MHz

- Provides commands and control for local and system bus
- Offers wide flexibility in system configurations
- Flexible command timing

SAB 82288-1 up to 20 MHz

- Optimal Multibus®-compatible timing
- Control drivers with 16 mA I_{OL} and tristate command drivers with 32 mA I_{OL}
- Single +5V supply
- Plastic Package: P-DIP-20

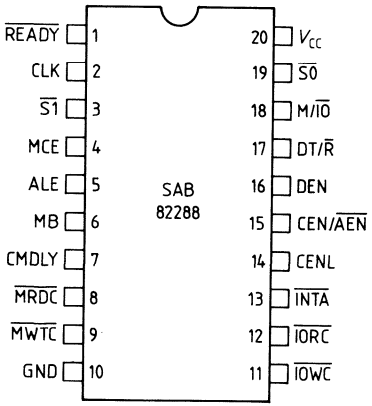


The SAB 82288 bus controller is a 20-pin MYMOS component for use in SAB 80286 microsystems. The bus controller provides command and control outputs with flexible timing options. Separate command outputs are used for memory and I/O devices. The data bus is controlled with separate data enable and direction control signals.
Two modes of operation are possible: Multibus-compatible bus cycles, and high-speed bus cycles.

Ordering Information

Type	Ordering code	Package	Description
SAB 82288-P	Q67120-Y75	P-DIP-20	Bus controller up to 16 MHz
SAB 82288-6-P	Q67120-Y110	P-DIP-20	Bus controller up to 12 MHz
SAB 82288-1-P	Q67120-Y69	P-DIP-20	Bus controller up to 20 MHz

Pin Configuration
(P-DIP-20)



Pin Names

CLK	System Clock
$\overline{S0}$, $\overline{S1}$	Bus Cycle Status
M/$\overline{IO}$	Memory or I/O Select
MB	Multibus Mode Select
CENL	Command Enable Latched
CMDLY	Command Delay
$\overline{READY}$	Bus Cycle Termination
CEN/$\overline{AEN}$	Command Enable/Address Enable
ALE	Address Latch Enable
MCE	Master Cascade Enable
DEN	Data Enable
DT/$\overline{R}$	Data Transmit/Receive
$\overline{IOWC}$	I/O Write Command
$\overline{IORC}$	I/O Read Command
$\overline{MWTC}$	Memory Write Command
$\overline{MRDC}$	Memory Read Command
$\overline{INTA}$	Interrupt Acknowledge
V_{CC}	Power supply (+5V)
GND	Ground (0V)

Pin Definitions and Functions

Symbol	Pin	Input (I) Output (O)	Function
$\overline{\text{READY}}$	1	I	**READY** indicates the end of the current bus cycle. $\overline{\text{READY}}$ is an active low input. Multibus mode requires at least one wait state to allow the command outputs to become active. $\overline{\text{READY}}$ must be low during reset, to force the SAB 82288 into the idle state. Setup and hold times must be met for proper operation.
CLK	2	I	**SYSTEM CLOCK** provides the basic timing control for the SAB 82288 in an SAB 80286 microsystem. Its frequency is twice the internal processor clock frequency. The falling edge of this input signal establishes when inputs are sampled and control outputs change.
$\overline{\text{S0}}$, $\overline{\text{S1}}$	3, 19	I	**BUS CYCLE STATUS** starts a bus cycle and, along with M/$\overline{\text{IO}}$, defines the type of bus cycle. These inputs are active low. A bus cycle is started when either $\overline{\text{S1}}$ or $\overline{\text{S0}}$ is sampled low at the falling edge of CLK. These inputs have pullup resistors sufficient to hold them high when nothing drives them. Setup and hold times must be met for proper operation.

SAB 80286 bus cycle status definition

M/$\overline{\text{IO}}$	$\overline{\text{S1}}$	$\overline{\text{S0}}$	Type of bus cycle
0	0	0	Interrupt acknowledge
0	0	1	I/O read
0	1	0	I/O write
0	1	1	None; idle
1	0	0	Halt or shutdown
1	0	1	Memory read
1	1	0	Memory write
1	1	1	None; idle

MCE	4	O	**MASTER CASCADE ENABLE** signals that a cascade address from a master SAB 8259A interrupt controller may be placed onto the CPU address bus for latching by the address latches under ALE control. The CPU's address bus may then be used to broadcast the cascade address to slave interrupt controllers so only one of them will respond to the interrupt acknowledge cycle. This control output is active high. MCE is only active during interrupt acknowledge cycles and is not affected by any control input. Using MCE to enable cascade address drivers requires latches which save the cascade address on the falling edge of ALE.
ALE	5	O	**ADDRESS LATCH ENABLE** controls the address latches used to hold an address stable during a bus cycle. This control output is active high. ALE will not be issued for the halt bus cycle and is not affected by any of the control inputs.
MB	6	I	**MULTIBUS MODE SELECT** determines timing of the command and control outputs. When high, the bus controller operates in Multibus mode. When low, the bus controller optimizes the command and control output timing for short bus cycles. The function of the CEN/$\overline{\text{AEN}}$ input pin is selected by this signal. Typically, this input is a strapping option and not dynamically changed. This input may be connected to V_{CC} or GND.

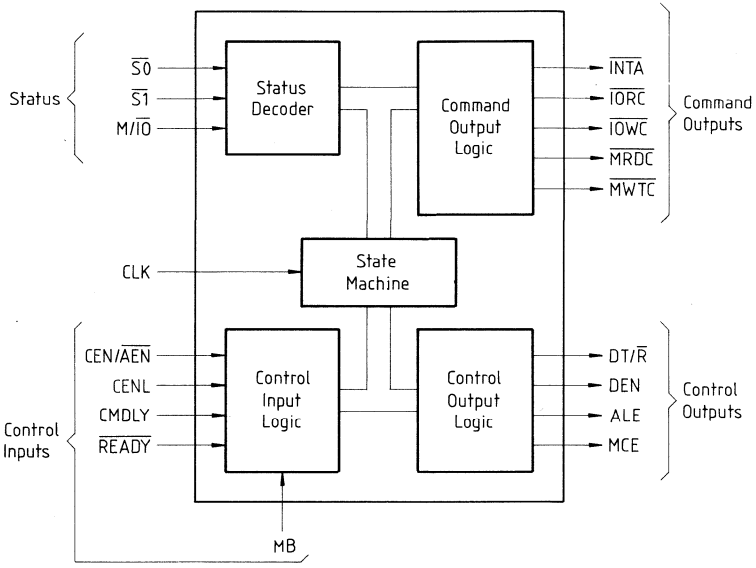
Pin Definitions and Function (cont'd)

Symbol	Pin	Input (I) Output (O)	Function
CMDLY	7	I	**COMMAND DELAY** allows delaying the start of a command. CMDLY is an active high input. If sampled high, the command output is not activated and CMDLY is again sampled at the next CLK cycle. When sampled low the selected command is enabled. If $\overline{\text{READY}}$ is detected low before the command output is activated, the SAB 82288 will terminate the bus cycle, even if no command was issued. Setup and hold times must be satisfied for proper operation. This input may be connected to GND if no delays are required before starting a command.
$\overline{\text{MRDC}}$	8	O	**$\overline{\text{MEMORY READ COMMAND}}$** instructs the memory device to place data onto the data bus. This command output is active low. The MB and CMDLY inputs control when this output becomes active. $\overline{\text{READY}}$ controls when it becomes inactive.
$\overline{\text{MWTC}}$	9	O	**$\overline{\text{MEMORY WRITE COMMAND}}$** instructs a memory device to read the data on the data bus. This command output is active low. The MB and CMDLY inputs control when this output becomes active. $\overline{\text{READY}}$ controls when it becomes inactive.
$\overline{\text{IOWC}}$	11	O	**$\overline{\text{I/O WRITE COMMAND}}$** instructs an I/O device to read the data on the data bus. This command output is active low. The MB and CMDLY inputs control when this output becomes active. $\overline{\text{READY}}$ controls when it becomes inactive.
$\overline{\text{IORC}}$	12	O	**$\overline{\text{I/O READ COMMAND}}$** instructs an I/O device to place data onto the data bus. This command output is active low. The MB and CMDLY inputs control when this output becomes active. $\overline{\text{READY}}$ controls when it becomes inactive.
$\overline{\text{INTA}}$	13	O	**$\overline{\text{INTERRUPT ACKNOWLEDGE}}$** tells an interrupting device that its interrupt request is being acknowledged. This command output is active low. The MB and CMDLY inputs control when this output becomes active. $\overline{\text{READY}}$ controls when it becomes inactive.
CENL	14	I	**COMMAND ENABLE LATCHED** is a bus controller select signal which enables the bus controller to respond to the current bus cycle being initiated. CENL is an active high input latched internally at the start of each bus cycle. CENL is used to select the appropriate bus controller for each bus cycle in a system where the CPU has more than one bus it can use. This input may be connected to V_{CC} to select this SAB 82288 for all transfers. No control inputs affect CENL. Setup and hold times must be met for proper operation.

Pin Definitions and Functions (cont'd)

Symbol	Pin	Input (I) Output (O)	Function
CEN/$\overline{\text{AEN}}$	15	I	**COMMAND ENABLE/$\overline{\text{ADDRESS ENABLE}}$** controls the command and DEN outputs of the bus controller. CEN/$\overline{\text{AEN}}$ inputs may be asynchronous to CLK. Setup and hold times are given to assure a guaranteed response to synchronous inputs. This input may be connected to V_{CC} or GND. When MB is high this pin has the $\overline{\text{AEN}}$ function. $\overline{\text{AEN}}$ is an active low input which indicates that the CPU has been granted use of a shared bus and the bus controller command outputs may exit tristate off and become inactive (high). $\overline{\text{AEN}}$ high indicates that the CPU does not have control of the shared bus and forces the command outputs into tristate off and DEN inactive (low). $\overline{\text{AEN}}$ would normally be controlled by an SAB 82289 bus arbiter which activates $\overline{\text{AEN}}$ when that arbiter owns the bus to which the bus controller is attached. When MB is low this pin has the CEN function. CEN is an unlatched active high input which allows the bus controller activate its command and DEN outputs. With MB low, CEN low forces the command and DEN outputs inactive but does not tristate them.
DEN	16	O	**DATA ENABLE** controls when data transceivers connected to the local data bus should be enabled. DEN is an active high control output. DEN is delayed for write cycles in the Multibus mode.
DT/$\overline{\text{R}}$	17	O	**DATA TRANSMIT/$\overline{\text{RECEIVE}}$** establishes the direction of data flow to or from the local data bus. When high, this control output indicates that a write bus cycle is being performed. A low indicates a read bus cycle. DEN is always inactive when DT/$\overline{\text{R}}$ changes states. This output is high when no bus cycle is active. DT/$\overline{\text{R}}$ is not affected by any of the control inputs.
M/$\overline{\text{IO}}$	18	I	**MEMORY or $\overline{\text{I/O}}$** SELECT determines whether the current bus cycle is in the memory space or I/O space. When low, the current bus cycle is in the I/O space. This input has a pullup resistor sufficient to hold it high when nothing drives it. Setup and hold times must be met for proper operation.
V_{CC}	20	–	**POWER SUPPLY** (+5 V)
GND	10	–	**GROUND** (0 V)

Block Diagram



Functional Description

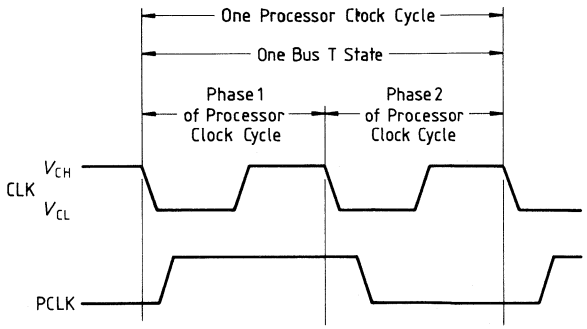
Introduction

The SAB 82288 bus controller is used in SAB 80286 systems to provide address latch control, data transceiver control, and standard level-type command outputs. The command outputs are timed and have sufficient drive capabilities for large TTL buses and meet all IEEE-796 requirements for Multibus. A special Multibus mode is provided to statisfy all address/data setup and hold time requirements. Command timing may be tailored to special needs via a CMDLY input to determine the start of a command and $\overline{\text{READY}}$ to determine the end of a command.
Connection to multiple buses is supported with a latched enable input (CENL). An address decoder can determine which, if any, bus controller should be enabled for the bus cycle. This input is latched to allow an address decoder to take full advantage of the pipelined timing on the SAB 80286 local bus.
Buses shared by several bus controllers are supported. An $\overline{\text{AEN}}$ input prevents the bus controller from driving the shared bus command and data signals except when enabled by an external bus arbiter such as the SAB 82289.
Separate DEN and DT/$\overline{\text{R}}$ outputs control the data transceivers for all buses. Bus contention is eliminated by disabling DEN before changing DT/$\overline{\text{R}}$. The DEN timing allows sufficient time for tristate bus drivers to enter tristate off before enabling other drivers onto the same bus.
The term CPU refers to any SAB 80286 processor or SAB 80286 support component which may become an SAB 80286 local bus master and thereby drive the SAB 82288 status inputs.

Processor Cycle Definition

Any CPU which drives the local bus uses an internal clock which is one half the frequency of the system clock (CLK) (see figure below). Knowledge of the phase of the local bus master's internal clock is required for proper operation of the SAB 80286 local bus. The local bus master informs the bus controller of its internal clock phase when it asserts the status signals. Status signals are always asserted in phase 1 of the local bus master's internal clock.

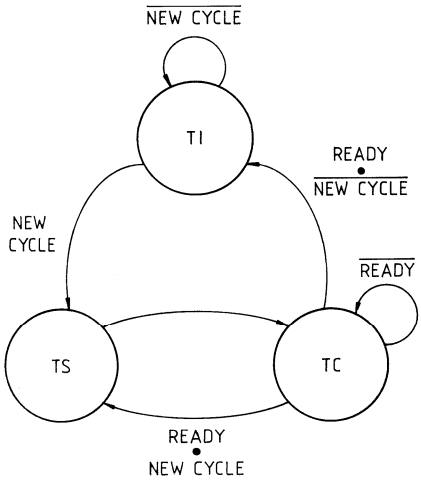
CLK Relationship to the Processor Clock and Bus T-States



Bus State Definition

The SAB 82288 bus controller has three bus states (see figure below): Idle (TI), Status (TS), and Command (TC). Each bus state is two CLK cycles long. Bus state phases correspond to the internal CPU processor clock phases.
The TI bus state occurs when no bus cycle is currently active on the SAB 80286 local bus. This state may be repeated indefinitely. When control of the local bus is being passed between masters, the bus remains in the TI state.

Bus States



Bus Cycle Definition

The $\overline{S1}$ and $\overline{S0}$ inputs signal the start of a bus cycle. When either input becomes low, a bus cycle is started. The TS bus state is defined to be the two CLK cycles during which either $\overline{S1}$ or $\overline{S0}$ is active (see figure on bus cycle definition). These inputs are sampled by the SAB 82288 at every falling edge of CLK. When either $\overline{S1}$ or $\overline{S0}$ is sampled low, the next CLK cycle is considered the second phase of the internal CPU clock cycle.
The local bus enters the TC bus state after the TS state. The shortest bus cycle may have one TS state and one TC state. Longer bus cycles are formed by repeating TC states. A repeated TC bus state is called a wait state.
The $\overline{READY}$ input determines whether the current TC bus state is to be repeated. The $\overline{READY}$ input has the same timing and effect for all bus cycles. $\overline{READY}$ is sampled at the end of each TC bus state to see if it is active. If sampled high, the TC bus state is repeated. This is called inserting a wait state. The control and command outputs do not change during wait states. When $\overline{READY}$ is sampled low, the current bus cycle is terminated. Note that the bus controller may enter the TS bus state directly from TC if the status lines are sampled active at the next falling edge of CLK.

Bus Cycle Definition

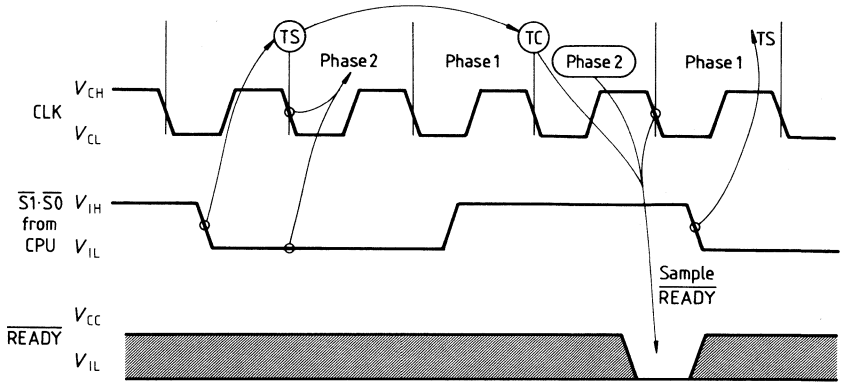


Table 2
Command and Control Output for Each Type of Bus Cycle

Type of bus cycle	M/$\overline{IO}$	$\overline{S1}$	$\overline{S0}$	Command activated	DT/$\overline{R}$ state	ALE, DEN issued?	MCE issued?
Interrupt acknowledge	0	0	0	$\overline{INTA}$	low	yes	yes
I/O read	0	0	1	$\overline{IORC}$	low	yes	no
I/O write	0	1	0	$\overline{IOWC}$	high	yes	no
None; idle	0	1	1	none	high	no	no
Halt/shutdown	1	0	0	none	high	no	no
Memory read	1	0	1	$\overline{MRDC}$	low	yes	no
Memory write	1	1	0	$\overline{MWTC}$	high	yes	no
None; idle	1	1	1	none	high	no	no

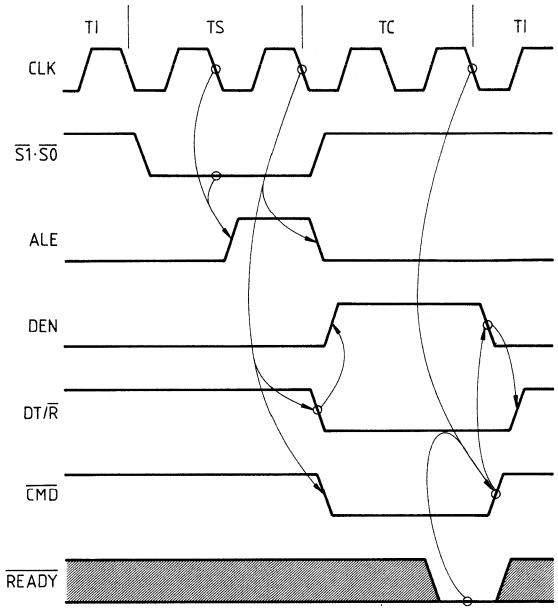
Operating Modes

Two types of buses are supported by the SAB 82288: Multibus and non-Multibus. When the MB input is high, Multibus timing is used. In Multibus mode, the SAB 82288 delays command and data activation to meet IEEE-796 requirements on address to command active and write data to command active setup timing. Multibus mode requires at least one wait state in the bus cycle since the command outputs are delayed. The non-Multibus mode does not delay any outputs and does not require wait states. The MB input affects the timing of the command and DEN outputs.

Command and Control Outputs

The type of bus cycle performed by the local bus master is encoded in the M/$\overline{IO}$, $\overline{S1}$, and $\overline{S0}$ inputs. Different command and control outputs are activated depending on the type of bus cycle. Table 2 indicates the cycle decoding done by the SAB 82288 and the effect on command, DT/$\overline{R}$, ALE, DEN, and MCE outputs.
Bus cycles come in three forms: read, write, and halt. Read bus cycles include memory read, I/O read, and interrupt acknowledge. The timing of the associated read command outputs ($\overline{MRDC}$, $\overline{IORC}$, and $\overline{INTA}$), control outputs (ALE, DEN, DT/$\overline{R}$) and control inputs (CEN/$\overline{AEN}$, CENL, CMDLY, MB, and $\overline{READY}$) are identical for all read bus cycles. Read cycles differ only in which command output is activated. The MCE control output is only asserted during interrupt acknowledge cycles.
Write bus cycles activate different control and command outputs with different timing than read bus cycles. Memory write and I/O write are write bus cycles whose timing for command outputs ($\overline{MWTC}$ and $\overline{IOWC}$), control outputs (ALE, DEN, DT/$\overline{R}$) and control inputs (CEN/$\overline{AEN}$, CENL, CMDLY, MB, and $\overline{READY}$) are identical. They differ only in which command output is activated.
Halt bus cycles are different because no command or control output is activated. All control inputs are ignored until the next bus cycle is started via $\overline{S1}$ and $\overline{S0}$.
The basic command and control output timing for read and write bus cycles is shown in the next five figures. Halt bus cycles are not shown since they activate no outputs. The basic idle-read-idle and idle-write-idle bus cycles are shown. The signal label $\overline{CMD}$ represents the appropriate command output for the bus cycle. For those five figures, the CMDLY input is connected to GND and CENL to V_{CC}. The effects of CENL and CMDLY are described later in the section on control inputs.
The next two figures show non-Multibus cycles. MB is connected to GND while CEN is connected to V_{CC}. The figure on page 11 shows a read cycle with no wait states while the figure on page 12 shows a write cycle with one wait state. The $\overline{READY}$ input is shown to illustrate how wait states are added.

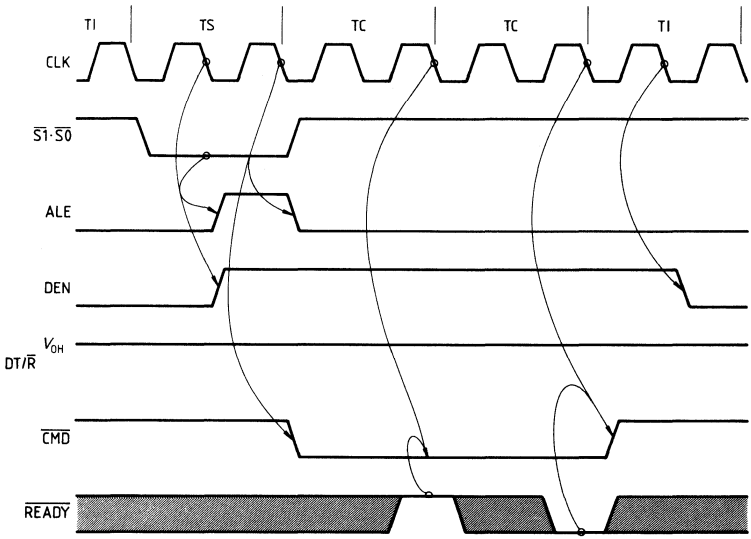
Idle-Read-Idle Bus Cycles with MB = 0



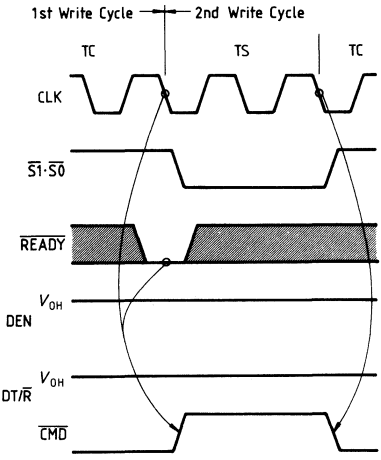
Bus cycles can occur back-to-back with no TI bus states between TC and TS. Back-to-back cycles do not affect the timing of the command and control outputs. Command and control outputs always reach the states shown for the same clock edge (within TS, TC, or following bus state) of a bus cycle. A special case in control timing occurs for back-to-back write cycles with MB = 0. In this case, DT/$\overline{R}$ and DEN remain high between the bus cycles (see respective write-write cycle diagram). The command and ALE output timing does not change.

The figures on pages 13 and 14 show a Multibus cycle with MB = 1. $\overline{AEN}$ and CMDLY are connected to GND. The effects of CMDLY and $\overline{AEN}$ are described later in the section on control inputs. The top figure shows a read cycle with one wait state and the figure below shows a write cycle with two wait states. The second wait state of the write cycle is shown only for example purposes and is not required. The $\overline{READY}$ input is shown to illustrate how wait states are added.

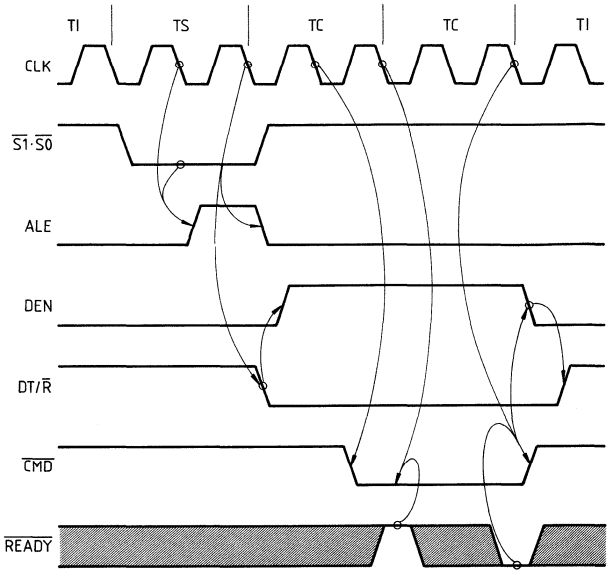
Idle-Write-Idle Bus Cycles with MB = 0



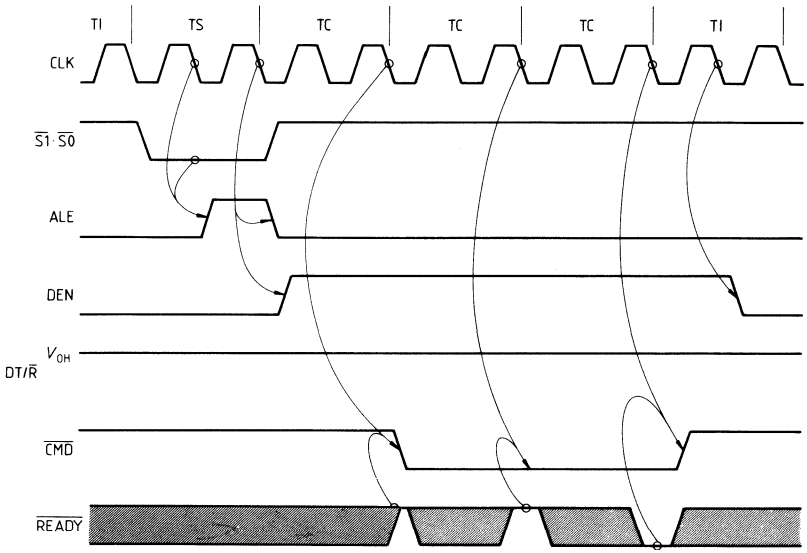
Write-Write Bus Cycles with MB = 0



Idle-Read-Idle Bus Cycles with MB = 1



Idle-Write-Idle Bus Cycles with MB = 1



The MB control input affects the timing of the command and DEN outputs. These outputs are automatically delayed in Multibus mode to satisfy three requirements:

1) 50 ns minimum setup time for valid address before any command output becomes active.
2) 50 ns minimum setup time for valid write data before any write command output becomes active.
3) 65 ns maximum time from when any read command becomes inactive until the slave's read data drivers reach tristate off.

Three signal transitions are delayed by MB = 1 as compared to MB = 0:

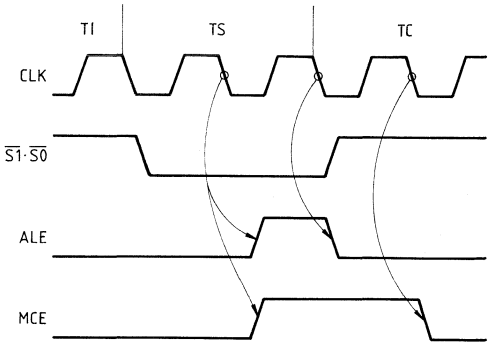
1) The high to low transition of the read command outputs ($\overline{IORC}$, $\overline{MRDC}$, and $\overline{INTA}$) is delayed one CLK cycle.
2) The high to low transition of the write command outputs ($\overline{IOWC}$ and $\overline{MWTC}$) is delayed two CLK cycles.
3) The low to high transition of DEN for write cycles is delayed one CLK cycle.

Back-to-back bus cycles with MB = 1 do not change the timing of any of the command or control outputs. DEN always becomes inactive between bus cycles with MB = 1.

Except for a halt or shutdown bus cycle, ALE will be issued during the second half of TS for any bus cycle. ALE becomes inactive at the end of the TS to allow latching the address to keep it stable during the entire bus cycle. The address outputs may change during phase 2 of any TC bus state. ALE is not affected by any control input.

The following figure shows how MCE is timed during interrupt acknowledge (INTA) bus cycles. MCE is one CLK cycle longer than ALE to hold the cascade address from a master SAB 8259A valid after the falling edge of ALE. With the exception of the MCE control output, an INTA bus cycle is identical in timing with a read bus cycle. MCE is not affected by any control input.

MCE Operation for an INTA Bus Cycle



Control Inputs

The control inputs can alter the basic timing of command outputs, allow interfacing to multiple buses, and share a bus between different masters. For many SAB 80286 systems, each CPU will have more than one bus which may be used to perform a bus cycle. Normally, a CPU will only have one bus controller active for each bus cycle. Some buses may be shared by more than one CPU (i.e. Multibus) requiring only one of them use the bus at a time.

Systems with multiple and shared buses use two control input signals of the SAB 82288 bus controller, CENL and $\overline{AEN}$ (see figure on system use of those signals). CENL enables the bus controller to control the current bus cycle. The $\overline{AEN}$ input prevents a bus controller from driving its command outputs. $\overline{AEN}$ high means that another bus controller may be driving the shared bus.

In the figure on the $\overline{AEN}$ and CENL signal, two buses are shown: a local bus and a Multibus. Only one bus is used for each CPU bus cycle. The CENL inputs of the bus controllers select which bus controller is to perform the bus cycle. An address decoder determines which bus to use for each bus cycle. The SAB 82288 connected to the shared Multibus must be selected by CENL and be given access to the Multibus by $\overline{AEN}$ before it will begin a Multibus operation.

CENL must be sampled high at the end of the TS bus state (see waveforms) to enable the bus controller to activate its command and control outputs. If sampled low the commands and DEN will not go active and DT/$\overline{R}$ will remain high. The bus controller will ignore the CMDLY, CEN, and $\overline{READY}$ inputs until another bus cycle is started via $\overline{S1}$ and $\overline{S0}$. Since an address decoder is commonly used to identify which bus is required for each bus cycle, CENL is latched to avoid the need for latching its input.

The CENL input can effect the DEN control output.
When MB = 0, DEN normally becomes active during phase 2 of TS in write bus cycles. This transition occurs before CENL is sampled. If CENL is sampled low, the DEN output will be forced low during TC as shown in the timing waveforms.

When MB = 1, CEN/$\overline{AEN}$ becomes $\overline{AEN}$, $\overline{AEN}$ controls when the bus controller command outputs enter and exit tristate off. $\overline{AEN}$ is intended to be driven by a bus arbiter, like the SAB 82289, which assures only one bus controller is driving the shared bus at any time. When $\overline{AEN}$ makes a low to high transition, the command outputs immediately enter tristate off and DEN is forced inactive. An inactive DEN should force the local data transceivers connected to the shared data bus into tristate off (see next figure). The low to high transition of $\overline{AEN}$ should only occur during TI or TS bus states.

The high-to-low transition of $\overline{AEN}$ signals that the bus controller may now drive the shared bus command signals. Since a bus cycle may be active or be in the process of starting, $\overline{AEN}$ can become active during any T-state. $\overline{AEN}$ low immediately allows DEN to go to the appropriate state. Three CLK edges later, the command outputs will go active (see timing waveforms). The Multibus requires this delay for the address and data to be valid on the bus before the commands become active.

When MB = 0, CEN/$\overline{AEN}$ becomes CEN. CEN is an asynchronous input which immediately affects the command and DEN outputs. When CEN makes a high-to-low transition, the commands and DEN are immediately forced inactive. When CEN makes a low-to-high transition, the commands and DEN outputs immediately go to the appropriate state (see timing waveforms). $\overline{READY}$ must still become active to terminate a bus cycle if CEN remains low for a selected bus controller (CENL was latched high).

Some memory or I/O systems may require more address or write data setup time to command active than provided by the basic command output timing. To provide flexible command timing, the CMDLY input can delay the activation of command outputs. The CMDLY input must be sampled low to activate the command outputs. CMDLY does not affect the control outputs ALE, MCE, DEN, and DT/$\overline{R}$.

CMDLY is first sampled on the falling edge of the CLK ending TS. If sampled high, the command output is not activated, and CMDLY is again sampled on the next falling edge of CLK. Once sampled low, the proper command output becomes active immediately if MB = 0. If MB = 1, the proper command goes active no earlier than shown in the figures on pages 12 and 13.

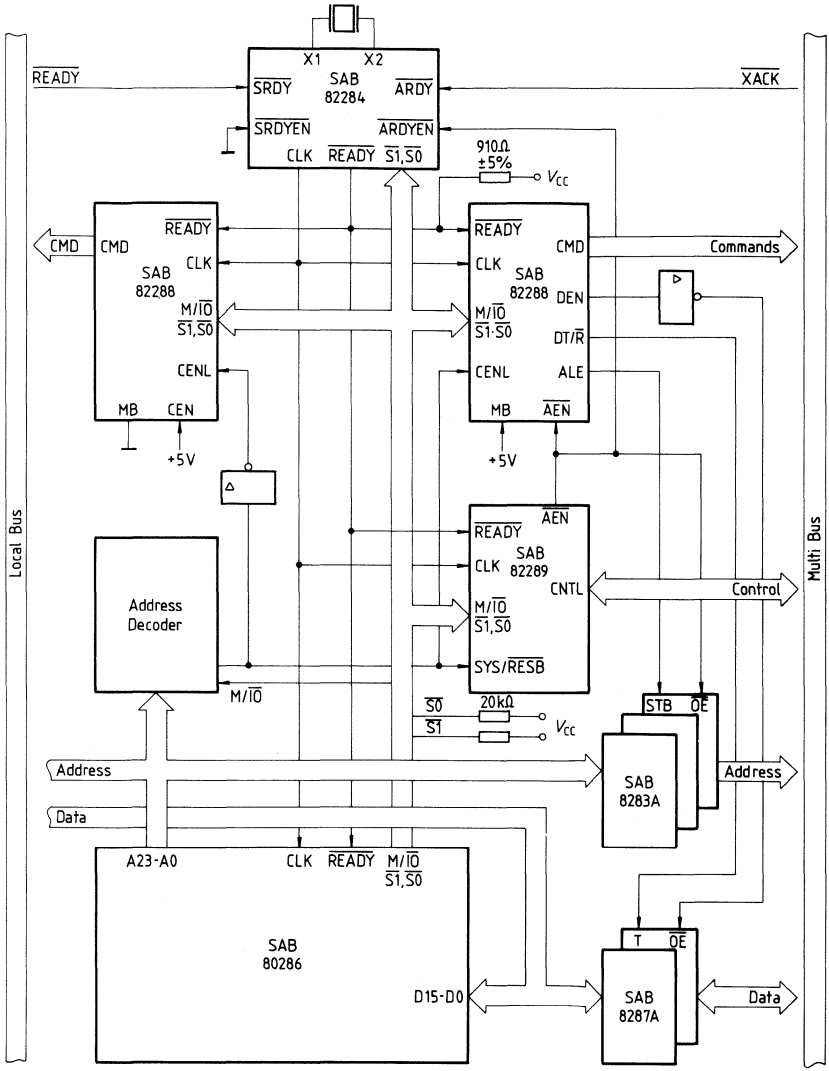
$\overline{READY}$ can terminate a bus cycle before CMDLY allows a command to be issued. In this case no commands are issued and the bus controller will deactivate DEN and DT/$\overline{R}$ in the same manner as if a command had been issued.

Waveforms

The waveforms show the timing relationships of inputs and outputs and do not show all possible transitions of all signals in all modes. Instead, all signal timing relationships are shown via the general cases. Special cases are shown when needed. The waveforms provide some functional descriptions of the SAB 82288; however, most functional descriptions are provided in the figures of section Functional Description.

To find the timing specification for a signal transition in a particular mode, first look for a special case in the waveforms. If no special case applies, then use a timing specification for the same or related function in another mode.

System Use of $\overline{\text{AEN}}$ and CENL



Absolute Maximum Ratings

Ambient temperature under bias . . . . . 0 to 70°C
Storage temperature . . . . . −65 to +150°C
Voltage on any pin with respect to GND . . . . . −0.5 to +7V
Power dissipation . . . . . 1 W

Note: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

T_A = 0 to 70°C, V_{CC} = 5 V ±10%

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
Power supply current	I_{CC}	–	100	mA	–
Forward input current ($\overline{S0}$, $\overline{S1}$, M/$\overline{IO}$)	I_F	–	−0.5	mA	V_F = 0.45 V
Input leakage current (all other)	I_{LI}	–	±10	µA	0 V ≦ V_{IN} ≦ V_{CC}
Output leakage current	I_{LO}	–	±10	µA	0.45 V ≦ V_{OUT} ≦ V_{CC}
Low output voltage Command outputs Control outputs	V_{OL}	–	0.45	V	 I_{OL} = 32 mA I_{OL} = 16 mA
High output voltage Command outputs Control outputs	V_{OH}	2.4	–	V	 I_{OH} = −5 mA I_{OH} = −1 mA
Low input voltage	V_{IL}	−0.5	0.8	V	–
CLK low input voltage	V_{CL}	−0.5	0.6	V	–
High input voltage	V_{IH}	2.0	V_{CC} +0.5	V	–
CLK high input voltage	V_{CH}	3.8		V	–

Capacitance

T_A = 25°C, V_{CC} = GND = 0 V, f_C = 1 MHz

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
CLK input capacitance	C_{CLK}	–	12	pF	Unmeasured pins returned to GND
Input capacitance	C_I	–	10	pF	
Input/output capacitance	C_{IO}	–	20	pF	

Note: Not 100%, guaranteed by design characterization.

AC Characteristics SAB 82288

Unless otherwise specified, the AC timings are referred to signal points of 0.8 V and 2.0 V as illustrated in the waveforms.

T_A = 0 to 70°C, V_{CC} = 5 V ±10%

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
CLK period	t_1	62	250	ns	–
CLK high time	t_2	20	235	ns	at 3.6 V
CLK low time	t_3	15	230	ns	at 1.0 V
CLK rise time	t_4	–	10	ns	1.0 to 3.6 V
CLK fall time	t_5	–	10	ns	3.6 to 1.0 V
M/$\overline{IO}$ and status setup time	t_6	22	–	ns	–
M/$\overline{IO}$ and status hold time	t_7	0	–	ns	–
CENL setup time	t_8	20	–	ns	–
CENL hold time	t_9	0	–	ns	–
$\overline{READY}$ setup time	t_{10}	38	–	ns	–
$\overline{READY}$ hold time	t_{11}	25	–	ns	–
CMDLY setup time	t_{12}	20	–	ns	–
CMDLY hold time	t_{13}	0	–	ns	–
$\overline{AEN}$ setup time	t_{14}	20	–	ns	1)
$\overline{AEN}$ hold time	t_{15}	0	–	ns	1)
ALE, MCE active delay	t_{16}	3	20	ns	2)
ALE, MCE inactive delay	t_{17}	–	20	ns	2)
DEN (write) inactive from CENL	t_{18}	–	35	ns	2)
DT/$\overline{R}$ low from CLK	t_{19}	–	25	ns	2)
DEN (read) active from DT/$\overline{R}$	t_{20}	5	35	ns	2)
DEN (read) inactive delay	t_{21}	3	35	ns	2)
DT/$\overline{R}$ high from DEN inactive	t_{22}	5	35	ns	2)
DEN (write) active delay	t_{23}	–	30	ns	2)
DEN (write) inactive delay	t_{24}	3	30	ns	2)
DEN inactive from CEN	t_{25}	–	25	ns	2)
DEN active from CEN	t_{26}	–	30	ns	2)
DT/$\overline{R}$ high from CLK and CEN	t_{27}	–	35	ns	2) 3)

Notes see next page

AC Characteristics SAB 82288 (cont'd)

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
DEN active from $\overline{AEN}$	t_{28}	–	30	ns	2)
Command active delay	t_{29}	3	25	ns	4)
Command inactive delay	t_{30}	3	25	ns	4)
Command inactive from CEN	t_{31}	–	25	ns	4)
Command active from CEN	t_{32}	–	25	ns	4)
Command valid delay from $\overline{AEN}$	t_{33}	–	40	ns	4)
Command float time	t_{34}	–	40	ns	4)
MB setup time	t_{35}	20	–	ns	–
MB hold time	t_{36}	0	–	ns	–
Command inactive enable from MB↓	t_{37}	–	40	ns	4)
Command float time from MB↑	t_{38}	–	40	ns	–
DEN inactive from MB↑	t_{39}	–	30	ns	2)
DEN active from MB↓	t_{40}	–	35	ns	2)

1) $\overline{AEN}$ is an asynchronous input. $\overline{AEN}$ setup and hold times are specified to guarantee the response shown in the waveforms.

2) Control output load: $C_L = 150$ pF, $I_{OL} = 16$ mA, $I_{OH} = -1$ mA.

3) t_{27} only applies to bus cycles where MB = 0, the SAB 82288 was selected, and DEN = 0 when the cycle is terminated (because CEN = 0).

4) Command output load: $C_L = 300$ pF, $I_{OL} = 32$ mA, $I_{OH} = -5$ mA.

AC Characteristics SAB 82288-6

Unless otherwise specified, the AC timings are referred to signal points of 0.8 V and 2.0 V as illustrated in the waveforms.

T_A = 0 to 70°C, V_{CC} = 5 V ±10%

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
CLK period	t_1	83	250	ns	–
CLK high time	t_2	25	235	ns	at 3.6 V
CLK low time	t_3	20	225	ns	at 1.0 V
CLK rise time	t_4	–	10	ns	1.0 to 3.6 V
CLK fall time	t_5	–	10	ns	3.6 to 1.0 V
M/$\overline{IO}$ and status setup time	t_6	28	–	ns	–
M/$\overline{IO}$ and status hold time	t_7	0	–	ns	–
CENL setup time	t_8	30	–	ns	–
CENL hold time	t_9	0	–	ns	–
$\overline{READY}$ setup time	t_{10}	50	–	ns	–
$\overline{READY}$ hold time	t_{11}	35	–	ns	–
CMDLY setup time	t_{12}	25	–	ns	–
CMDLY hold time	t_{13}	0	–	ns	–
$\overline{AEN}$ setup time	t_{14}	25	–	ns	1)
$\overline{AEN}$ hold time	t_{15}	0	–	ns	1)
ALE, MCE active delay	t_{16}	3	25	ns	2)
ALE, MCE inactive delay	t_{17}	–	35	ns	2)
DEN (write) inactive from CENL	t_{18}	–	35	ns	2)
DT/$\overline{R}$ low from CLK	t_{19}	–	40	ns	2)
DEN (read) active from DT/$\overline{R}$	t_{20}	5	50	ns	2)
DEN (read) inactive delay	t_{21}	3	40	ns	2)
DT/$\overline{R}$ high from DEN inactive	t_{22}	5	45	ns	2)
DEN (write) active delay	t_{23}	–	35	ns	2)
DEN (write) inactive delay	t_{24}	3	35	ns	2)
DEN inactive from CEN	t_{25}	–	40	ns	2)
DEN active from CEN	t_{26}	–	35	ns	2)
DT/$\overline{R}$ high from CLK and CEN	t_{27}	–	50	ns	2) 3)

Notes see next page

AC Characteristics SAB 82288-6 (cont'd)

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
DEN active from $\overline{AEN}$	t_{28}	–	35	ns	2)
Command active delay	t_{29}	3	40	ns	4)
Command inactive delay	t_{30}	3	30	ns	4)
Command inactive from CEN	t_{31}	–	35	ns	4)
Command active from CEN	t_{32}	–	45	ns	4)
Command valid delay from $\overline{AEN}$	t_{33}	–	40	ns	4)
Command float time	t_{34}	–	40	ns	4)
MB setup time	t_{35}	25	–	ns	–
MB hold time	t_{36}	0	–	ns	–
Command inactive enable from MB↓	t_{37}	–	40	ns	4)
Command float time from MB↑	t_{38}	–	40	ns	–
DEN inactive from MB↑	t_{39}	–	40	ns	2)
DEN active from MB↓	t_{40}	–	35	ns	2)

1) $\overline{AEN}$ is an asynchronous input. $\overline{AEN}$ setup and hold times are specified to guarantee the response shown in the waveforms.

2) Control output load: $C_L = 150$ pF, $I_{OL} = 16$ mA, $I_{OH} = -1$ mA.

3) t_{27} only applies to bus cycles where MB = 0, the SAB 82288 was selected, and DEN = 0 when the cycle is terminated (because CEN = 0).

4) Command output load: $C_L = 300$ pF, $I_{OL} = 32$ mA, $I_{OH} = -5$ mA.

AC Characteristics SAB 82288-1

Unless otherwise specified, the AC timings are referred to signal points of 0.8 V and 2.0 V as illustrated in the waveforms.

T_A = 0 to 70°C, V_{CC} = 5 V ±10%

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
CLK period	t_1	50	250	ns	–
CLK high time	t_2	16	235	ns	at 3.6 V
CLK low time	t_3	12	230	ns	at 1.0 V
CLK rise time	t_4	–	8	ns	1.0 to 3.6 V
CLK fall time	t_5	–	8	ns	3.6 to 1.0 V
M/$\overline{IO}$ and status setup time	t_6	18	–	ns	–
M/$\overline{IO}$ and status hold time	t_7	0	–	ns	–
CENL setup time	t_8	15	–	ns	–
CENL hold time	t_9	0	–	ns	–
$\overline{READY}$ setup time	t_{10}	26	–	ns	–
$\overline{READY}$ hold time	t_{11}	25	–	ns	–
CMDLY setup time	t_{12}	15	–	ns	–
CMDLY hold time	t_{13}	0	–	ns	–
$\overline{AEN}$ setup time	t_{14}	15	–	ns	1)
$\overline{AEN}$ hold time	t_{15}	0	–	ns	1)
ALE, MCE active delay	t_{16}	3	16	ns	2)
ALE, MCE inactive delay	t_{17}	–	19	ns	2)
DEN (write) inactive from CENL	t_{18}	–	23	ns	2)
DT/$\overline{R}$ low from CLK	t_{19}	–	23	ns	2)
DEN (read) active from DT/$\overline{R}$	t_{20}	5	21	ns	2)
DEN (read) inactive delay	t_{21}	3	21	ns	2)
DT/$\overline{R}$ high from DEN inactive	t_{22}	5	20	ns	2)
DEN (write) active delay	t_{23}	–	23	ns	2)
DEN (write) inactive delay	t_{24}	3	19	ns	2)
DEN inactive from CEN	t_{25}	–	25	ns	2)
DEN active from CEN	t_{26}	–	24	ns	2)
DT/$\overline{R}$ high from CLK and CEN	t_{27}	–	25	ns	2) 3)

Notes see next page

AC Characteristics SAB 82288-1 (cont'd)

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
DEN active from $\overline{AEN}$	t_{28}	–	26	ns	2)
Command active delay	t_{29}	3	21	ns	4)
Command inactive delay	t_{30}	3	20	ns	4)
Command inactive from CEN	t_{31}	–	25	ns	4)
Command active from CEN	t_{32}	–	25	ns	4)
Command valid delay from $\overline{AEN}$	t_{33}	–	40	ns	4)
Command float time	t_{34}	–	40	ns	4)
MB setup time	t_{35}	20	–	ns	–
MB hold time	t_{36}	0	–	ns	–
Command inactive enable from MB↓	t_{37}	–	40	ns	4)
Command float time from MB↑	t_{38}	–	40	ns	–
DEN inactive from MB↑	t_{39}	–	26	ns	2)
DEN active from MB↓	t_{40}	–	35	ns	2)

1) $\overline{AEN}$ is an asynchronous input. $\overline{AEN}$ setup and hold times are specified to guarantee the response shown in the waveforms.

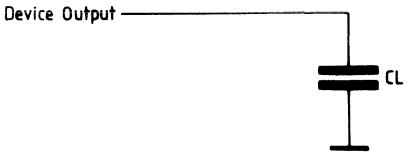
2) Control output load: $C_L = 150$ pF, $I_{OL} = 16$ mA, $I_{OH} = -1$ mA.

3) t_{27} only applies to bus cycles where MB = 0, the SAB 82288 was selected, and DEN = 0 when the cycle is terminated (because CEN = 0).

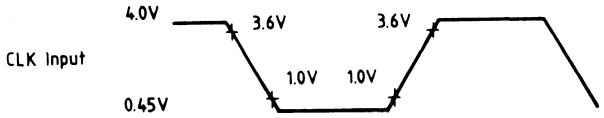
4) Command output load: $C_L = 300$ pF, $I_{OL} = 32$ mA, $I_{OH} = -5$ mA.

AC Testing Waveforms

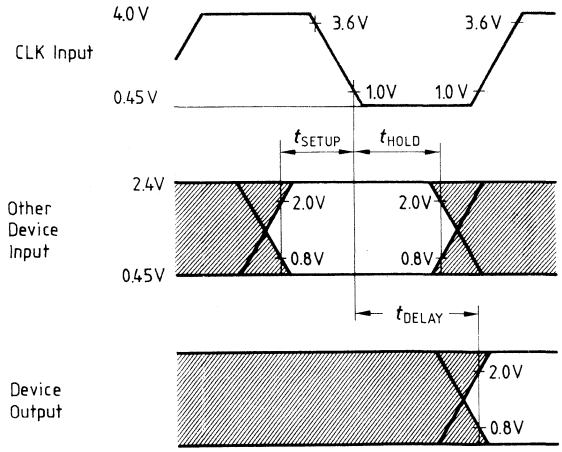
Test Loading on Outputs



Drive and Measurement Points – CLK Input

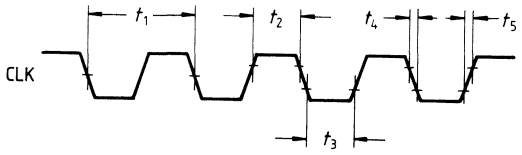


Setup, Hold and Delay Time Measurement – General

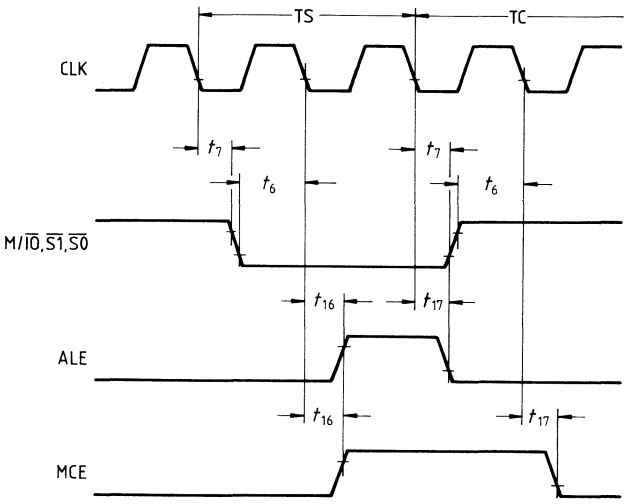


Waveforms

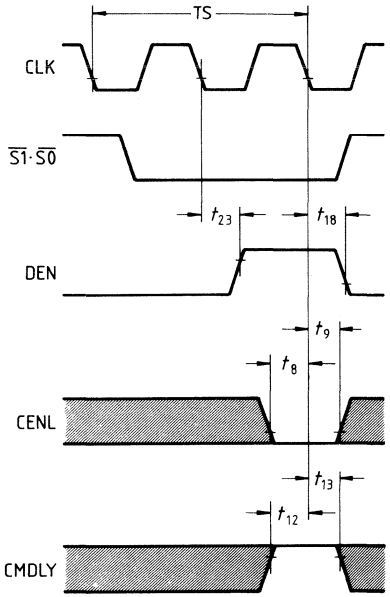
CLK Characteristics



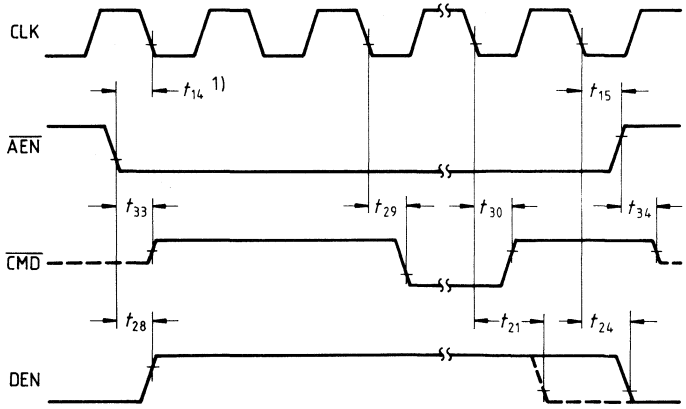
Status, ALE, MCE Characteristics



CENL, CMDLY, DEN Characteristics with MB = 0 and CEN = 1 during Write Cycle

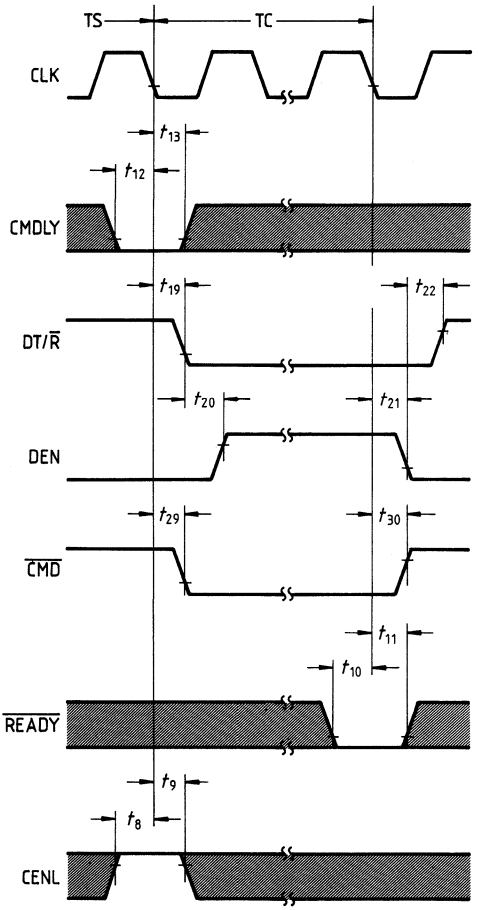


$\overline{AEN}$ Characteristics with MB = 1

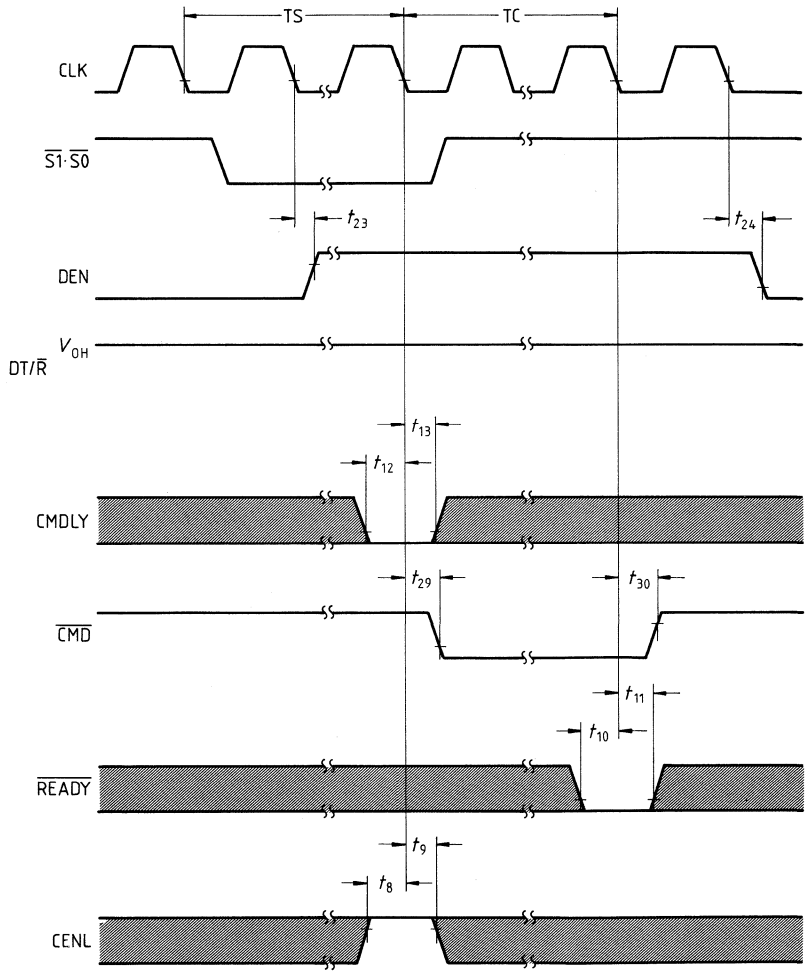


1) $\overline{AEN}$ is an asynchronous input. $\overline{AEN}$ setup and hold time is specified to guarantee the response shown in the waveforms.

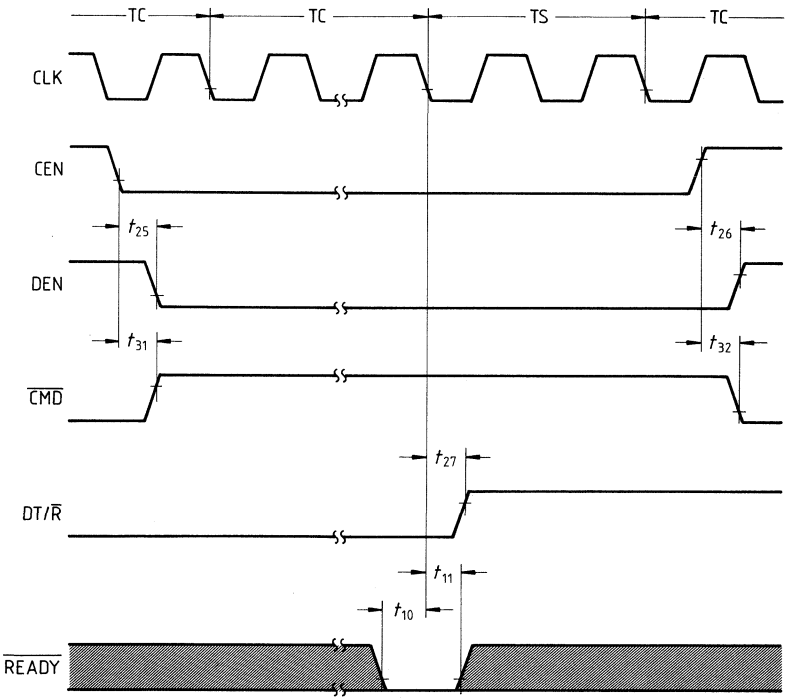
Read Cycle Characteristics with MB = 0 and CEN = 1



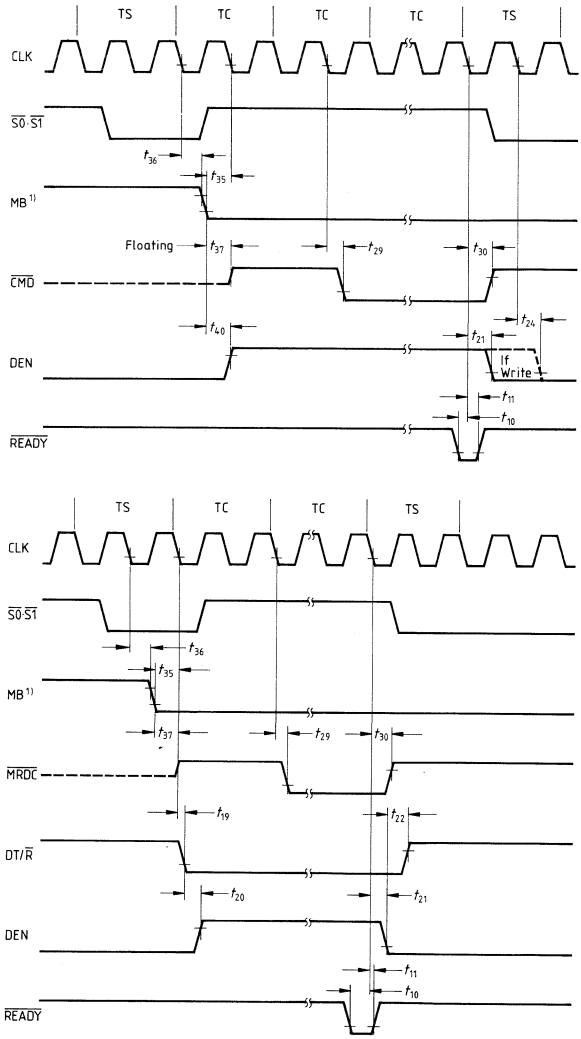
Write Cycle Characteristics with MB = 0 and CEN = 1



CEN Characteristics with MB = 0



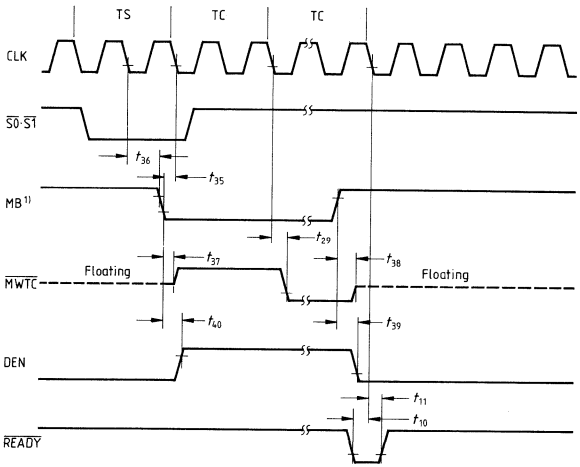
MB Characteristics with $\overline{AEN}$/CEN = High



1) MB is an asynchronous input. MB setup and hold times specified to guarantee the response shown in the waveforms.

If the setup time t_{35} is met, two clock cycles will occur before $\overline{CMD}$ becomes active after the falling edge of MB.

MB Characteristics with $\overline{AEN}$/CEN = High (cont'd)



1) MB is an asynchronous input. MB setup and hold times specified to guarantee the response shown in the waveforms.

If the setup time t_{35} is met, two clock cycles will occur before $\overline{CMD}$ becomes active after the falling edge of MB.

Preliminary

SAB 82289 Bus Arbiter for SAB 80286 Processors

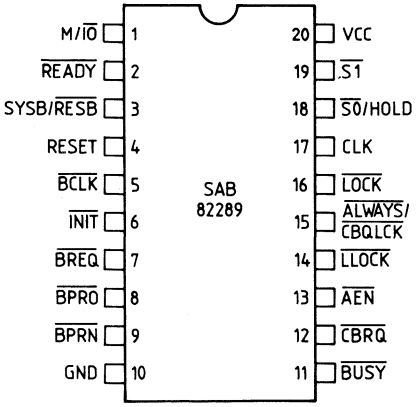
SAB 82289-6 up to 12 MHz

- Supports multimaster system bus arbitration protocol
- Synchronizes SAB 80286 processor with multimaster bus
- Compatible with IEEE 796 standard bus (Multibus®)

SAB 82289 up to 16 MHz

- Three modes of bus release operation for flexible system configuration
- Supports parallel, serial and rotating priority resolving schemes

Pin Configuration



Pin Names

S0/HOLD	Status S0/Hold Input
S1, M/IO	Status Inputs
BUSY	Busy Input/Output
READY	Ready Input
SYSB/RESB	System/Resident Bus
RESET	Processor Reset
BCLK	Bus Clock
INIT	Initialize
BREQ	Bus Request
BPRO	Bus Priority Out
BPRN	Bus Priority In
CLK	System Clock
AEN	Address Enable
ALWAYS/ CBQLCK	Always Release/ Common Bus Request Lock
LLOCK	Level Lock
CBRQ	Common Bus Request
LOCK	Bus Lock
VCC	+5V
GND	Ground (0V)

The SAB 82289 Bus Arbiter is a 5 V, 20-pin MYMOS component for use in multiple bus master SAB 80286 systems. The SAB 82289 provides a compact solution to system bus arbitration for the SAB 80286 CPU.

The complete IEEE 796 Standard bus arbitration protocol is supported. Three modes of bus release operation support a number of bus usage models.

Multibus® is a registered trademark of Intel Corporation.

Pin Definitions and Functions

<table>
<tr><th>Symbol</th><th>Pin</th><th>Input (I)
Output (O)</th><th>Function</th></tr>
<tr><td>M/$\overline{IO}$, $\overline{S1}$</td><td>1, 19</td><td>I</td><td>STATUS INPUTS are the status input signal pins from the SAB 80286 processor. The arbiter decodes these inputs together with the $\overline{S0}$/HOLD input to initiate bus request and surrender actions. A bus cycle is started when either $\overline{S1}$ or $\overline{S0}$ is sampled low at the falling edge of CLK. The SAB 80286's $\overline{S1}$ and M/$\overline{IO}$ pins meet the setup and hold time requirements of these pins.
SAB 80286 Bus Cycle Status Encoding
<table><tr><th>M/$\overline{IO}$</th><th>$\overline{S1}$</th><th>$\overline{S0}$/HOLD</th><th>Type of Bus Cycle</th></tr><tr><td>0</td><td>0</td><td>0</td><td>Interrupt acknowledge</td></tr><tr><td>0</td><td>0</td><td>1</td><td>I/O read</td></tr><tr><td>0</td><td>1</td><td>0</td><td>I/O write</td></tr><tr><td>0</td><td>1</td><td>1</td><td>None, bus idle</td></tr><tr><td>1</td><td>0</td><td>0</td><td>Halt or shutdown</td></tr><tr><td>1</td><td>0</td><td>1</td><td>Memory read</td></tr><tr><td>1</td><td>1</td><td>0</td><td>Memory write</td></tr><tr><td>1</td><td>1</td><td>1</td><td>None, bus idle</td></tr></table>When supporting the HOLD output of another bus master, the $\overline{S1}$ and M/$\overline{IO}$ pins must be held high during TS, the Status Cycle, for proper operation.</td></tr>
<tr><td>$\overline{READY}$</td><td>2</td><td>I</td><td>$\overline{READY}$ is an active-low signal which indicates the end of the bus cycle. The SAB 80286 halt or shutdown cycle does not require $\overline{READY}$ to terminate the bus cycle. Setup and hold times for this pin must be met for proper operation.</td></tr>
<tr><td>SYSB/$\overline{RESB}$</td><td>3</td><td>I</td><td>SYSTEM BUS/$\overline{RESIDENT\ BUS}$ is an input signal which determines whether the multimaster system bus is required for the current bus cycle. The signal can originate from address mapping circuitry such as a decoder or PROM attached to the processor address and status pins. The arbiter will request or retain control of the multimaster system bus when the SYSB/$\overline{RESB}$ pin is sampled high at the end of the TS bus state.
During an interrupt acknowledge cycle, this input is sampled on every falling edge of CLK starting at the end of the TS state until either SYSB/$\overline{RESB}$ is sampled high or the bus cycle is terminated by the $\overline{READY}$ signal. Setup and hold times for this pin must be met for proper operation.</td></tr>
<tr><td>RESET</td><td>4</td><td>I</td><td>PROCESSOR RESET is an active-high input synchronous to the system,clock (CLK). RESET is the processsor initialization and an indication to the arbiter to release the multimaster bus and clear any pending request.</td></tr>
<tr><td>$\overline{BCLK}$</td><td>5</td><td>I</td><td>$\overline{BUS\ CLOCK}$ is the multimaster system bus clock to which the multimaster bus interface signals are synchronized. $\overline{BCLK}$ can be asynchronous to CLK.</td></tr>
<tr><td>$\overline{INIT}$</td><td>6</td><td>I</td><td>$\overline{INITIALIZE}$ is an active-low Multibus signal used to reset all arbiters on the Multibus system. It will cause the release of the multimaster bus, but will not clear the pending bus master request so that the arbiter can again request the multimaster bus. No arbiters have the use of the multimaster bus immediately after initialization. $\overline{INIT}$ is an asynchronous signal to CLK.</td></tr>
</table>

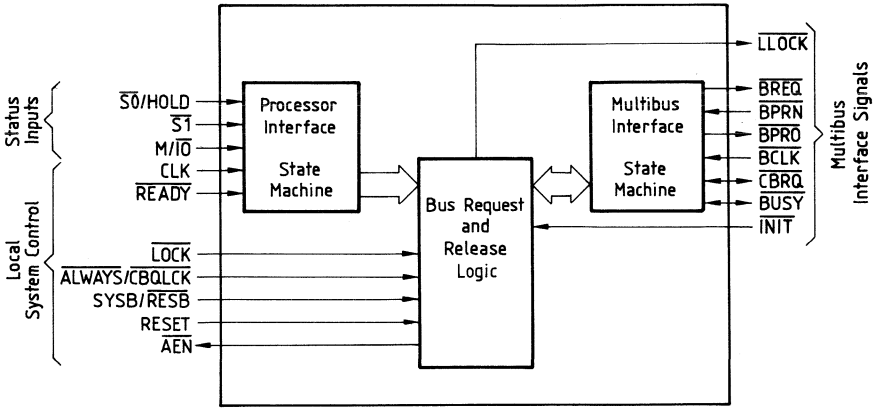
Pin Definitions and Functions (continued)

Symbol	Pin	Input (I) Output (O)	Function
$\overline{BREQ}$	7	O	$\overline{BUS\ REQUEST}$ is an active-low output signal used in the parallel and rotating priority resolving schemes. The arbiter activates $\overline{BREQ}$ to request the use of the multimaster system bus. The arbiter holds $\overline{BREQ}$ active as long as it is requesting or has possession of the multimaster system bus.
$\overline{BPRO}$	8	O	$\overline{BUS\ PRIORITY\ OUT}$ is an active-low output signal used in serial priority resolving scheme. $\overline{BPRO}$ is connected to $\overline{BPRN}$ of the next lower priority arbiter to grant or revoke priority from that arbiter.
$\overline{BPRN}$	9	I	$\overline{BUS\ PRIORITY\ IN}$ is an active-low input indicating that this arbiter has the highest priority of any arbiter requesting the system bus. $\overline{BPRN}$ high signals the arbiter that a higher priority arbiter is requesting or has possession of the system bus. Setup and hold times for this pin must be met for proper operation.
$\overline{BUSY}$	11	I/O (open-drain)	$\overline{BUSY}$ is a Multibus signal which is asserted when the system bus is in use. $\overline{BUSY}$ is an open drain input/output requiring an external pullup resistor. As an input $\overline{BUSY}$ asserted indicates when the Multibus is in use. Setup and hold times must be met for proper operation. As an output $\overline{BUSY}$ is asserted to signal when this arbiter has taken control of the Multibus.
$\overline{CBRQ}$	12	I/O (open-drain)	$\overline{COMMON\ BUS\ REQUEST}$ is a Multibus signal that indicates when an arbiter is requesting the Multibus. This pin is an open-drain input/output requiring an external pullup resistor. As an input $\overline{CBRQ}$ indicates that another arbiter is requesting the multimaster system bus. The input function of this pin is enabled by the $\overline{CBQLCK}$ signal. Setup and hold times for this pin must be met for proper operation. As an output $\overline{CBRQ}$ is asserted to indicate that this arbiter is requesting the Multibus. The arbiter pulls $\overline{CBRQ}$ low when it issues $\overline{BREQ}$. The arbiter releases $\overline{CBRQ}$ when it obtains the Multibus.
$\overline{AEN}$	13	O	$\overline{ADDRESS\ ENABLE}$ is the output of the arbiter which goes directly to the processor's address latches, the SAB 82288 bus controller and the SAB 82284 clock generator. $\overline{AEN}$ asserted causes the bus controller and address latches to enable their output drivers. $\overline{AEN}$ also drives the clock generator's $\overline{ARDYEN}$ input to enable its asynchronous ready input ($\overline{ARDY}$). $\overline{AEN}$ can also be used as an active-low hold acknowledge to a bus master other than the SAB 80286. It signals to the bus master that control of the system bus has been relinquished when $\overline{AEN}$ is inactive (high). Note that $\overline{AEN}$ goes active relative to $\overline{BCLK}$ and goes inactive relative to CLK.

Pin Definitions and Functions (continued)

Symbol	Pin	Input (I) Output (O)	Function
$\overline{\text{LLOCK}}$	14	O	$\overline{\text{LEVEL LOCK}}$ is an active-low output signal decoded from the processor $\overline{\text{LOCK}}$ signal. $\overline{\text{LLOCK}}$ can be used as Multibus $\overline{\text{LOCK}}$ when buffered with a tri-state buffer enabled by the $\overline{\text{AEN}}$ signal. $\overline{\text{LLOCK}}$ will be cleared by RESET but not by $\overline{\text{INIT}}$.
$\overline{\text{ALWAYS}}$/ $\overline{\text{CBQLCK}}$	15	I	$\overline{\text{ALWAYS RELEASE}}$ or $\overline{\text{COMMON BUS REQUEST LOCK}}$ can be programmed at processor reset to be either the ALWAYS RELEASE ($\overline{\text{ALWAYS}}$) strapping option or the COMMON BUS REQUEST LOCK ($\overline{\text{CBQLCK}}$) control input. Setup and hold times for this pin must be met for proper programming. When this pin is low during the falling edge of processor reset ($\overline{\text{ALWAYS}}$ option) the arbiter is programmed to surrender the multimaster system bus after each bus transfer cycle. The SAB 82289 will remain in the ALWAYS RELEASE mode until it is reprogrammed during the next processor reset. The bus arbiter is programmed to support the COMMON BUS REQUEST LOCK function by forcing this input pin high during the falling edge of the processor reset. $\overline{\text{CBQLCK}}$ itself is an active-low signal which when active prevents the arbiter from surrendering the multimaster system bus to a common bus request through the $\overline{\text{CBRQ}}$ input pin.
$\overline{\text{LOCK}}$	16	I	$\overline{\text{LOCK}}$ is a processor-generated signal which when asserted (low) prevents the arbiter from surrendering the multi-master system bus to any other bus arbiter, regardless of its priority. $\overline{\text{LOCK}}$ is sampled by the arbiter at the end of the TS (status) bus state. Setup and hold times for this pin must be met for proper operation.
CLK	17	I	SYSTEM CLOCK accepts the CLK signal from the SAB 82284 clock generator chip as the timing reference for the bus arbiter and processor interface signals.
$\overline{\text{S0}}$/HOLD	18	I	STATUS INPUT $\overline{\text{S0}}$ or HOLD is either the $\overline{\text{S0}}$ status signal from the SAB 80286 or the HOLD signal from some other bus master. The function of this input is established during the processor reset of the SAB 82289 bus arbiter. The SAB 80286 $\overline{\text{S0}}$ pin meets the setup and hold time requirements of this pin. The $\overline{\text{S0}}$ pin function is selected by forcing this input high during the falling edge of processor reset if the SAB 82289 is used to support an SAB 80286 processor, the $\overline{\text{S0}}$ output of the processor will be high during reset. In supporting the SAB 80286 processor, the SAB 82289 decodes the $\overline{\text{S0}}$ pin together with the other status input $\overline{\text{S1}}$ and M/$\overline{\text{IO}}$ to determine the beginning of a processor bus cycle and initiate bus request and surrender actions. The HOLD function of the $\overline{\text{S0}}$/HOLD pin is selected by holding this input low during the falling edge of processor reset. When supporting a bus master other than SAB 80286 the SAB 82289 monitors the HOLD signal to initiate bus request and surrender actions.
VCC	20	–	POWER SUPPLY (+5 V)
GND	10	–	GROUND (0 V)

Figure 1
Block Diagram



Functional Description

The SAB 82289 bus arbiter in conjunction with the SAB 82288 bus controller and the SAB 82284 clock generator interfaces the SAB 80286 processor or some other bus master to a multimaster system bus. The arbiter multiplexes a processor to a multimaster system bus. It avoids contention with other bus masters.

The SAB 82289 has two separate state machines which communicate through bus request and release logic. The processor interface state machine is synchronous with the local system clock (CLK) and the multimaster system bus interface state machine is synchronous with the bus clock ($\overline{BCLK}$).

The SAB 82289 performs all signaling to request, obtain, and release the system bus. External logic is used to determine which bus cycles require the system bus and to resolve priorities of simultaneous requests for control of the system bus.

SAB 82289 with SAB 80286

In a SAB 80286 system using a SAB 82289 bus arbiter, the SAB 80286 processor is unaware of the arbiter's existence and issues commands as though it had exclusive use of a multimaster system bus, such as Multibus. If the processor cycle requires Multibus access, the arbiter requests control of the Multibus. Until the request is granted the SAB 82289 keeps $\overline{AEN}$ disabled to prevent the SAB 82288 bus controller and the address latches from accessing the Multibus. $\overline{AEN}$ inactive also disasserts the asynchronous ready enable ($\overline{ARDYEN}$) input of the SAB 82284 clock chip so that the system bus will appear as "NOT READY" to the SAB 80286 processor.

Once the SAB 82289 bus arbiter has acquired the bus, it will assert $\overline{AEN}$ allowing the SAB 82288 bus controller and the address latches to access the system bus and asserting the $\overline{ARDYEN}$ input of the SAB 82284 clock chip.

Typically, once the data transfer command has been issued by the SAB 82288 and the data transfer has taken place, a transfer acknowledge ($\overline{XACK}$) signal is returned to the processor on the multimaster system bus to indicate "READY" from the accessed slave device. The processor remains in a series of "wait states" (repeated TC states) until the addressed device responds with $\overline{XACK}$ asserted signal to the SAB 82284's $\overline{ARDY}$ input and the SAB 82284 asserts $\overline{READY}$ to the processor. The processor then completes its bus cycle.

SAB 82289 with Other Bus Masters

When supporting other bus masters, the $\overline{S0}$/HOLD and $\overline{READY}$ pins of the bus arbiter can be connected to the "hold" pin of that master. The inverted $\overline{AEN}$ signal from the SAB 82289 can be used as the hold acknowledge (HLDA) input for the other bus master.

The bus master sends a HOLD signal to the bus arbiter when it needs the system bus for a memory access. If the arbiter currently controls the system bus, $\overline{AEN}$ will be active. Otherwise $\overline{AEN}$ will be inactive and the arbiter will request control of the system bus. The bus master will have to wait until the SAB 82289 has asserted $\overline{AEN}$ (low) before it starts its bus cycle.

When the bus master no longer requires the Multibus it will have to inactivate the HOLD signal. The arbiter interprets the Multibus access as a single bus cycle which is terminated by HOLD going inactive (low). Thus the arbiter will not release the Multibus to any other bus master during a bus access cycle.

Processor Cycle Definition

Any SAB 80286 system which gains access to the Multibus through the SAB 82289 bus arbiter uses an internal clock which is one half the frequency of the system clock (CLK) (see figure 2). Knowledge of the phase of the local bus master internal clock is required for proper SAB 82289 control of the SAB 80286 interface to Multibus. The local bus master informs the bus arbiter of its internal clock phase when it asserts the status signals. The SAB 80286's $\overline{S0}$ and $\overline{S1}$ status signals are always first asserted in phase 1 of the local bus master's internal clock.

Bus State Definition

The SAB 82289 bus arbiter has three processor bus states (see figure 3): idle (TI), status (TS), command (TC). Each bus state is two CLK cycles long. Bus state phases correspond to the internal CPU clock phases.

Figure 2
CLK Relationship to Internal Processor Phase, and Bus T-States

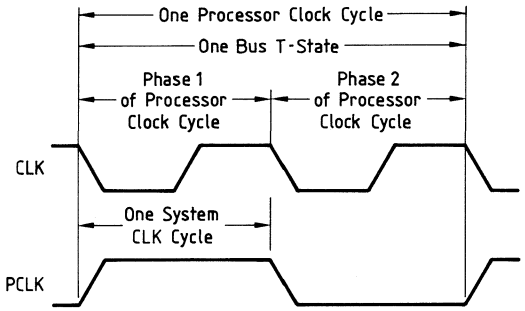
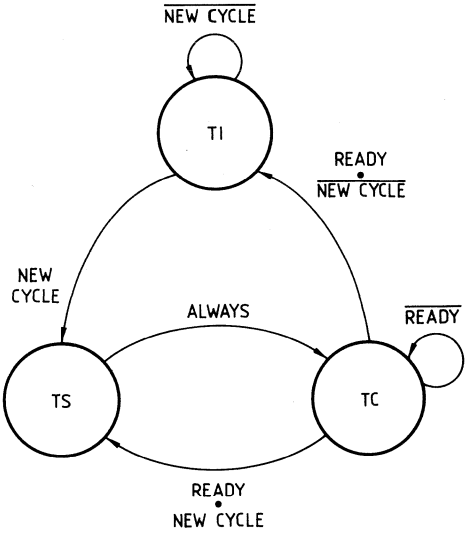


Figure 3
SAB 82289 Processor Bus States



Bus Cycle Definition

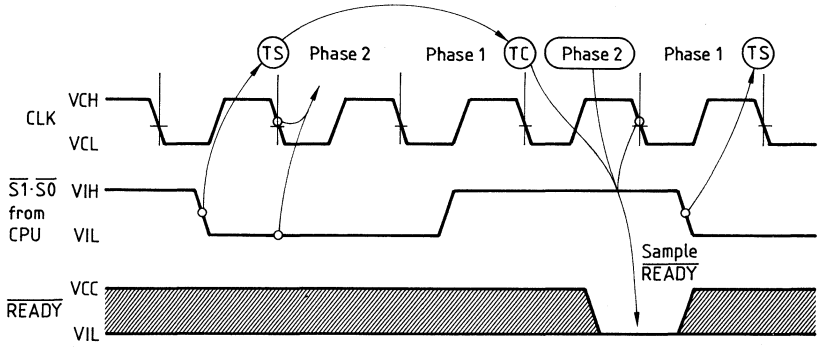
The $\overline{S1}$ and $\overline{S0}$ status inputs are sampled by the SAB 82289 on the falling edge of CLK and signal the start of a bus cycle by going active (low). The TS bus state is defined to be the two CLK cycles during which either $\overline{S1}$ or $\overline{S0}$ is active (see figure 4). When either $\overline{S1}$ or $\overline{S0}$ is sampled low, the next CLK cycle is considered the second phase of the associated processor clock cycle.

The arbiter enters the TC bus state after the TS state. The shortest bus cycle may have one TS state and one TC state. Longer bus cycles are formed by repeating TC states. A repeated TC bus state is called a wait state.

The $\overline{READY}$ input determines whether the current TC bus state is to be repeated. The $\overline{READY}$ input has the same timing and effect for all bus cycles. $\overline{READY}$ is sampled at the end of each TC bus state to see if it is active. If sampled high, the TC bus state is repeated. This is called inserting a wait state.

When $\overline{READY}$ is sampled low, the current bus cycle is terminated. Note that the bus arbiter may enter the TS bus state directly from TC if the status lines are sampled active (low) at the next falling edge of CLK (see figure 4). If neither of the status lines is sampled active at that time the SAB 82289 will enter the TI bus state. The TI bus state will be repeated until the status inputs are sampled active.

Figure 4
SAB 80286 Bus Cycle Definition (without Wait States)



Arbitration between Bus Masters

The Multibus protocol allows multiple processing elements to compete with each other to access common system resources. Since the local SAB 80286 processor does not have exclusive use of the system bus, if the Multibus is "BUSY" the SAB 80286 processor will have to wait before it can access the system bus.

The SAB 82289 bus arbiter provides an integrated solution for controlling access to a multimaster system bus. The bus arbiter allows both higher and lower priority bus masters to acquire the system bus depending on which release mode is used. In general higher priority masters obtain the bus immediately after any lower priority master completes its present transfer cycle. Lower priority bus masters obtain the bus when a higher priority master is not accessing the system bus or the proper surrender conditions exist. The SAB 82289 handles this arbitration in a manner completely transparent to the bus master (e.g. SAB 80286 processor).

At the end of each transfer, the arbiter may retain or release the system bus. This decision is controlled by the processor state, bus arbitration inputs and arbiter strapping options (see releasing the Multibus, ahead).

Priority Resolving Techniques

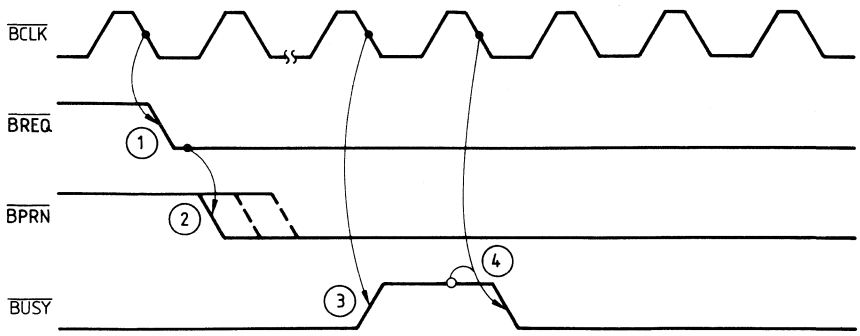
Some means of resolving priority between bus masters requesting the multimaster bus simultaneously must be provided. The SAB 82289 bus arbiter supports parallel, serial and rotating system bus priority resolving techniques. All of these techniques are based on the concept that, at a given time, one bus master will have priority above all the others.

An individual arbiter is the highest priority arbiter requesting the Multibus when its $\overline{\text{BPRN}}$ input is asserted (low). The highest priority requesting arbiter cannot immediately seize the system bus. It must wait until the present bus transaction is completed. Upon completing its current transaction the present bus owner surrenders the bus by releasing $\overline{\text{BUSY}}$.

$\overline{\text{BUSY}}$ is an active-low "wired OR" Multibus signal connecting all bus arbiters on the system bus. When $\overline{\text{BUSY}}$ goes inactive, the arbiter which has requested the system bus and presently has bus priority ($\overline{\text{BPRN}}$ low), seizes the bus by pulling $\overline{\text{BUSY}}$ low (see waveform in figure 5).

The generation of a multimaster bus request ($\overline{\text{BREQ}}$) is controlled by the type of bus cycle and the SYSB/$\overline{\text{RESB}}$ input. Whenever the processor signals the status for memory read, memory write, I/O read,

Figure 5
Bus Exchange Timing for the Multibus



(1) Higher Priority Bus Arbiter Requests the Multimaster System Bus
(2) Attains Priority. (Does not Yet Own Bus).
(3) Lower Priority Bus Arbiter Releases $\overline{\text{BUSY}}$.
(4) Higher Priority Bus Arbiter then Acquires the Bus and Pulls $\overline{\text{BUSY}}$ Low.

I/O write or interrupt acknowledge cycle, and SYSB/$\overline{\text{RESB}}$ is high at the end of TS, a bus request is generated.

When the status inputs indicate an interrupt acknowledge bus cycle, the arbiter allows external logic to decide (through the SYSB/$\overline{\text{RESB}}$ input) whether the interrupt acknowledge cycle should use the Multibus.

Figure 6 shows how SYSB/$\overline{\text{RESB}}$ is repeatedly sampled until it is sampled high or the bus cycle is terminated. If the bus cycle is completed ($\overline{\text{READY}}$ is sampled low) before SYSB/$\overline{\text{RESB}}$ is sampled high, the arbiter will not request the Multibus.

The SAB 82289 bus arbiter does not generate a separate $\overline{\text{BREQ}}$ for each bus cycle. Instead the SAB 82289 generates $\overline{\text{BREQ}}$ when it requests the bus and holds $\overline{\text{BREQ}}$ active during the time that it has possession of the bus. Note that all multimaster system bus requests (via $\overline{\text{BREQ}}$) are synchronized to the system bus clock ($\overline{\text{BCLK}}$).

Figure 6
Bus Request Timing During an Interrupt Acknowledge Cycle

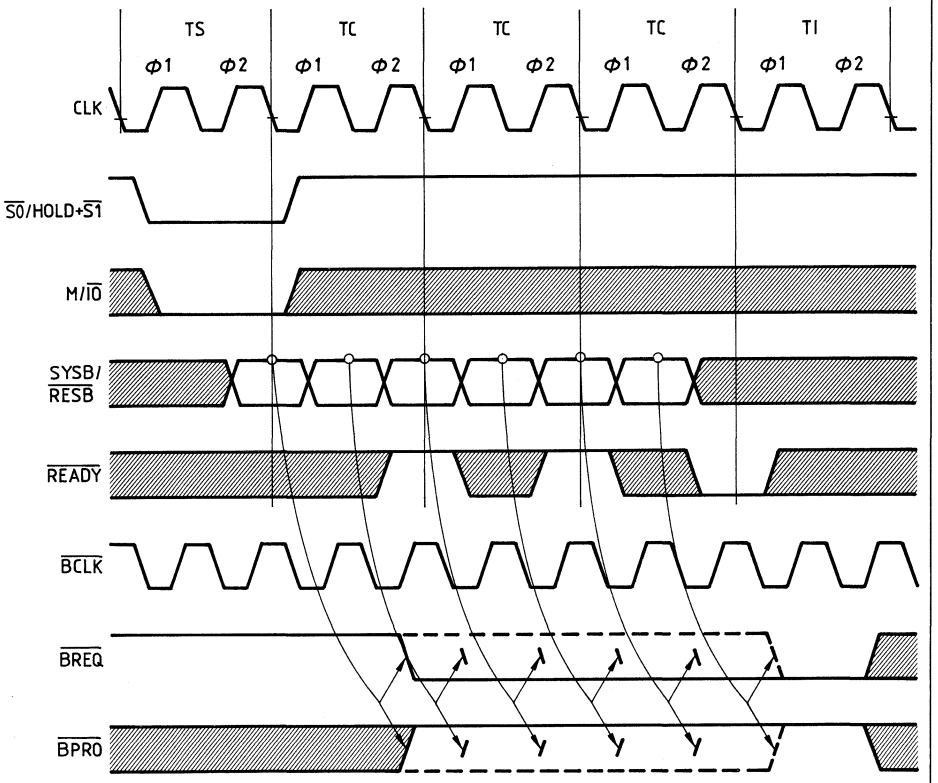


Figure 7
Parallel Priority Resolving Technique

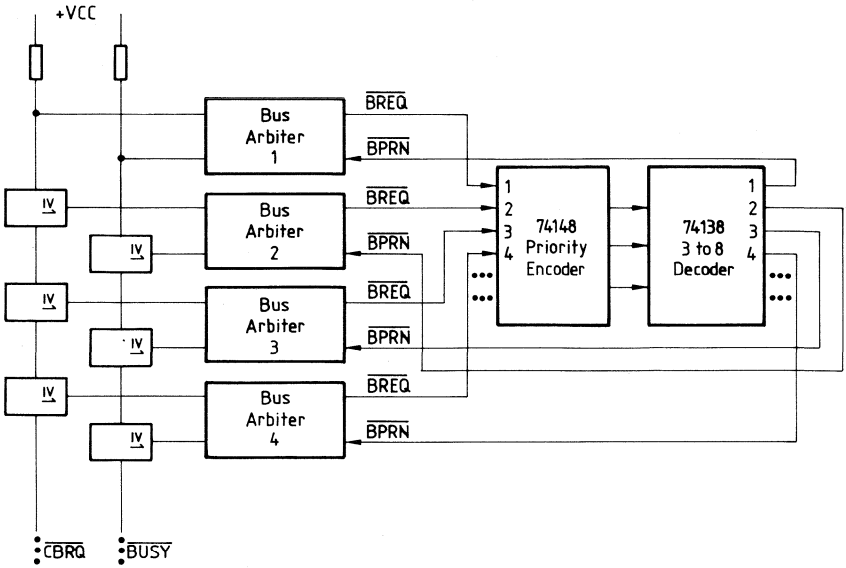
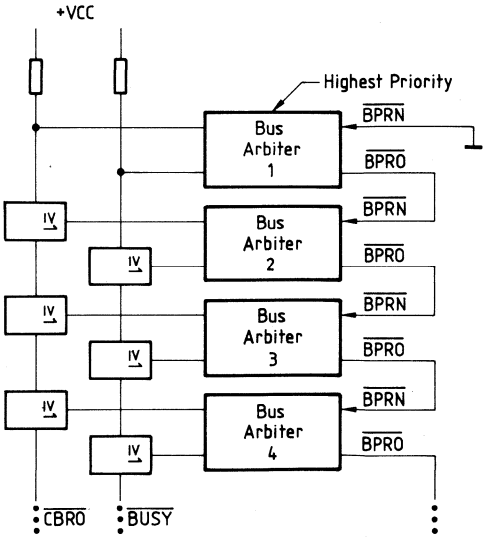


Figure 8
Connections for Serial Priority Resolving Technique



Parallel Priority Resolving Technique

The parallel priority resolving technique requires a separate bus request line ($\overline{BREQ}$) for each arbiter on the multimaster system bus (see figure 7). Each $\overline{BREQ}$ line enters a priority encoder which generates the binary address of the highest priority $\overline{BREQ}$ line currently active. The binary address is decoded to select the $\overline{BPRN}$ line corresponding to the highest priority arbiter requesting the bus. In a parallel scheme, the $\overline{BPRO}$ output is not used.

The arbiter receiving priority ($\overline{BPRN}$ low) then allows its associated bus master onto the multimaster system bus as soon as the bus becomes available (i.e. the bus is no longer busy). Any number of bus masters may be accommodated in this way, limited only by the complexity of the external priority resolving circuitry. Such circuitry must resolve the priority within one $\overline{BCLK}$ period.

Serial Priority Resolving Technique

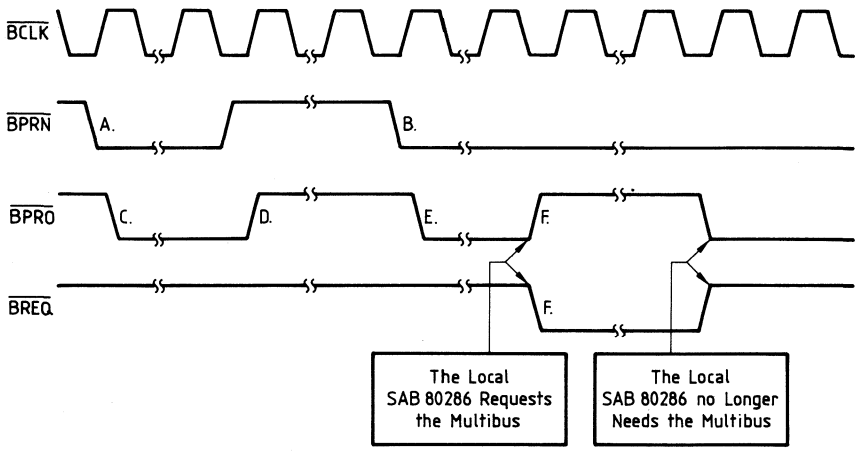
The serial priority resolving technique eliminates the need for the priority circuitry of the parallel technique by daisy-chaining the bus arbiters together, that is, connecting the higher priority arbiter's $\overline{BPRO}$ output to the $\overline{BPRN}$ of the next lower priority arbiter (see figure 8). The highest priority bus arbiter would have its $\overline{BPRN}$ tied low in this configuration, signifying to the arbiter that it always has the highest priority when requesting the system bus. In a serial scheme, the $\overline{BREQ}$ output is not used.

Since arbitration must be resolved within one $\overline{BCLK}$ period the number of arbiters connected together in the serial priority is limited by the arbiter's $\overline{BPRN}$ to $\overline{BPRO}$ propagation delay (18ns). For a 10 MHz Multibus $\overline{BCLK}$, five SAB 82289 bus arbiters may be connected in serial configuration.

Maximum number of chained-priority devices =

$$\frac{\overline{BCLK} \text{ period}}{\overline{BPRN} \text{ to } \overline{BPRO} \text{ delay}}$$

Figure 9
Serial Priority Bus Behavior



Note: Events A through F described in the following

When using the serial priority resolving scheme, a higher priority arbiter (for example, arbiter 2, figure 8) passes priority to the next lower priority arbiter (arbiter 3) by asserting its $\overline{\text{BPRO}}$ signal (low). This asserts $\overline{\text{BPRN}}$ of next arbiter (arbiter 3) as shown in figure 9 event A and event B. An arbiter's $\overline{\text{BPRO}}$ is asserted if the arbiter has priority ($\overline{\text{BPRN}}$ is asserted) but is not accessing or requesting the system bus (as indicated by $\overline{\text{BREQ}}$ inactive as shown in figure 9 event C and event E for arbiter 3). Whenever a higher priority arbiter (arbiter 3) issues a bus request its $\overline{\text{BPRO}}$ goes inactive causing the next lower priority arbiter (arbiter 4) to lose its bus priority (figure 9 event F). Any arbiter (arbiter 3) will also bring its $\overline{\text{BPRO}}$ inactive if its $\overline{\text{BPRN}}$ goes inactive (from arbiter 2), thereby passing the loss of bus priority on to the lower priority arbiters (e.g. arbiter 4) as shown in figure 9 event D.

Rotating Priority Resolving Technique

The rotating priority resolving technique is similar to the parallel priority resolving technique except that priority is dynamically re-assigned. The priority encoder is replaced by a more complex circuitry which rotates priority between requesting arbiters, thus allowing each arbiter an equal chance to use the multimaster system bus over a given period of time.

Selecting the Appropriate Priority Resolving Technique

The choice of a priority resolving technique involves a trade-off between external logic complexity and ease of Multibus access for the different bus masters in the system. The rotating priority resolving technique requires a substantial amount of external logic, but guarantees all the bus masters an equal opportunity to access the system bus. The serial priority resolving technique uses no external logic but has fixed bus master priority levels and can accomodate only a limited number of bus arbiters.

The parallel priority resolving technique is in general a compromise between the other two techniques (for example parallel priority configuration in figure 7 allows up to eight arbiters to be present on the Multibus, with fixed priority levels, while not requiring a large amount of complex external logic to implement).

Releasing the Multibus

Following a data transfer cycle on the Multibus, the SAB 82289 bus arbiter can either retain control of the system bus or release the bus for use by some other bus master. The SAB 82289 can operate in one of three modes, defining different conditions under which the arbiter relinquishes control of the multimaster system bus. These release modes are described in the table below.

If the arbiter was programmed to operate in Always Release Mode (mode 1) during the previous reset, it will surrender the Multibus after each complete transfer cycle. If the arbiter is not in Always Release Mode, it will not surrender the bus until the local SAB 80286 processor enters a halt state, the arbiter is forced off the bus by the loss of $\overline{\text{BPRN}}$ (mode 2 or 3), or by a common bus request when the $\overline{\text{CBRQ}}$ input is enabled by the $\overline{\text{CBQLCK}}$ input (mode 2).

$\overline{\text{CBRQ}}$ can save the bus exchange overhead in many cases. If $\overline{\text{CBRQ}}$ is high, it indicates to the bus master that no other master is requesting the bus and therefore the present bus master can retain the bus. Without $\overline{\text{CBRQ}}$, only $\overline{\text{BPRN}}$ indicates whether or not another master is requesting the bus and that only if the other master is of higher priority. Between its bus transfer cycles the master must give up the bus in order to allow lower priority masters to take the bus if they need it. At the start of the master's next transfer cycle, the bus must be regained. If no other master has the bus, this can take approximately two $\overline{\text{BCLK}}$ periods. To avoid this overhead of unnecessarily giving up and regaining the bus when no other masters needs it, $\overline{\text{CBRQ}}$ is

SAB 82289 Release Modes

Release Mode	Conditions under which the bus arbiter releases the system bus (unless cycles are LOCKed)
Mode 1	The bus arbiter always releases the bus at the end of each transfer cycle
Mode 2	The bus arbiter ratains the bus until: • a higher-priority bus master requests the bus, driving $\overline{\text{BPRN}}$ high • a lower-priority bus master requests the bus by pulling $\overline{\text{CBRQ}}$ low
Mode 3	The bus arbiter retains the bus until: • a higher-priority bus master requests the bus, driving $\overline{\text{BPRN}}$ high ($\overline{\text{CBRQ}}$ low ignored)

extremely useful. Any master that wants but does not have the bus, must assert $\overline{CBRQ}$ (low). If $\overline{CBRQ}$ line is not asserted the bus does not have to be released, thereby eliminating the delay of regaining the bus at the start of the next cycle.

The $\overline{LOCK}$ input to the arbiter can be used to override any of the conditions shown in the table before. While $\overline{LOCK}$ is asserted, the arbiter will not surrender control of the Multibus to any other requesting arbiter. Note that the arbiter will surrender the Multibus (synchronous to $\overline{BCLK}$) either in response to RESET or $\overline{INIT}$ signals independent of the current release mode or the state of the arbiter inputs.

The three bus release modes have the same operation when supporting either the SAB 80286 processor or some other bus master.

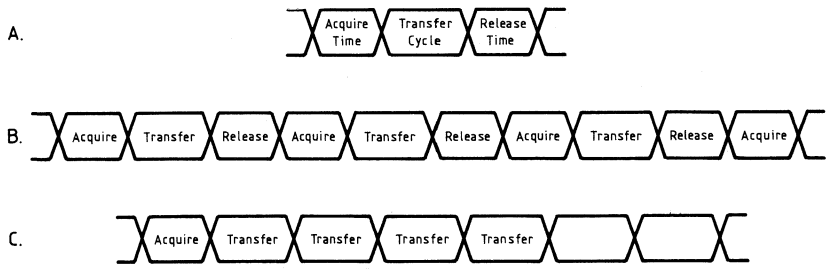
Selecting the Appropriate Release Mode

The choice of which release mode to use may affect the bus utilization of the individual subsystems, and the system as a whole. Mode-dependent performance variations are due to the bus acquisition/release overhead. The effect of these acquire and release times on system bus efficiency is illustrated in figure 10.

An isolated transfer on the multimaster system bus is depicted in figure 10-A. Figure 10-B shows utilization for the bus arbiter operation in mode 1. The arbiter must request and release the system bus for each transfer cycle. Lower priority arbiters have easy access to the system bus, but overall bus efficiency is low. Bus utilization for a bus arbiter operating in mode 2 or 3 is shown in the figure 10-C. In this situation the arbiter acquires the bus once for a sequence of transfers. The arbiter retains the bus until forced off by another bus master's request as defined in table before.

The three release modes of the SAB 82289 allow the designer to optimize the system use of the Multibus.

Figure 10
Effects of Bus Contention on Bus Efficiency



Configuring the SAB 82289 Release Mode

The SAB 82289 bus arbiter can be configured in any of its three bus release modes without additional hardware. It can also be configured to switch between mode 2 and mode 3 under software control of the SAB 80286 processor, requiring that a parallel port or addressable latch be used to drive the $\overline{\text{ALWAYS}}/\overline{\text{CBQLCK}}$ input pin of the SAB 82289 (see figure below).

Figure 11
SAB 82289 Release Mode Configurations

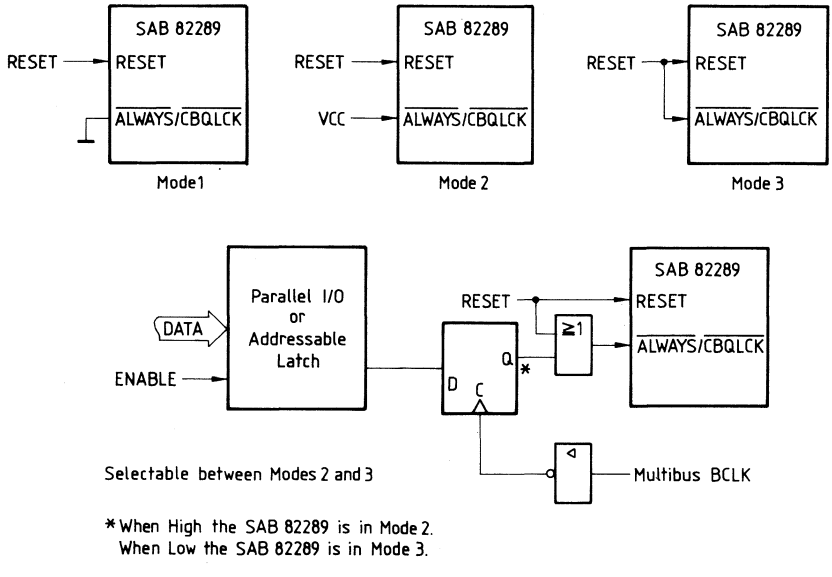
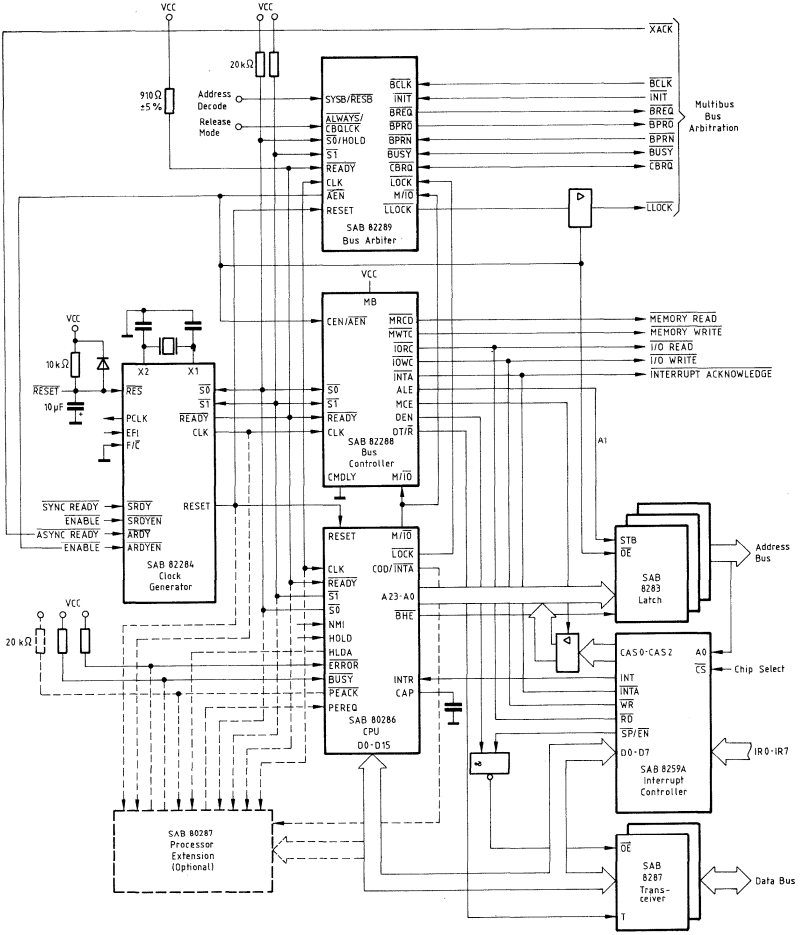


Figure 12
Typical SAB 80286 Subsystem Multibus Interface



Asserting the $\overline{\text{LOCK}}$ Signal

Independent of the particular release mode of the SAB 82289 bus arbiter, the SAB 80286 processor can assert a $\overline{\text{LOCK}}$ signal synchronously to CLK to prevent the arbiter from releasing the Multibus. This software-controlled $\overline{\text{LOCK}}$ signal prevents the SAB 82289 from surrendering the system bus to any other bus master, whether that bus master is of higher or lower priority. The $\overline{\text{LOCK}}$ signal is typically used for implementing software semaphores for shared resources or for critical processes that must run in real-time.

The SAB 82289 $\overline{\text{LLOCK}}$ output is the Multibus timing-compatible signal asserted during all bus cycles which are locked together. The $\overline{\text{LLOCK}}$ is set or reset depending on processor $\overline{\text{LOCK}}$ at the end of the TS cycle. $\overline{\text{LLOCK}}$ will delay going inactive until the termination of the current transfer cycle.

The SAB 82289 will continue to assert the $\overline{\text{LLOCK}}$ signal retaining control of the Multibus until the end of the first "unLOCKed" SAB 80286 bus cycle (SAB 80286 disables its $\overline{\text{LOCK}}$ output on the last bus cycle indicating that no future locked cycles are needed). While the $\overline{\text{LOCK}}$ signal will force the arbiter presently in control to hold the system bus, it cannot force another arbiter to surrender the bus any earlier than it normally would.

The $\overline{\text{LLOCK}}$ signal from the SAB 82289 must be connected to a tri-state buffer in order to drive the Multibus $\overline{\text{LOCK}}$ signal. This tri-state buffer should be enabled by the $\overline{\text{AEN}}$ signal from the arbiter going active.

SAB 82289 Reset and Initialization

The SAB 82289 bus arbiter provides the RESET and $\overline{\text{INIT}}$ pins for initialization. RESET is a CLK synchronous signal from the SAB 82284 clock generator and $\overline{\text{INIT}}$ is an asynchronous signal on the multimaster system bus. By having RESET pin high or $\overline{\text{INIT}}$ pin low the $\overline{\text{BREQ}}$, $\overline{\text{BUSY}}$ and $\overline{\text{AEN}}$ output pins will all become inactive. RESET will also deactivate the $\overline{\text{LLOCK}}$ signal. Unlike RESET, $\overline{\text{INIT}}$ will not clear any pending bus request, the bus request would be asserted after the $\overline{\text{INIT}}$ signal goes inactive.

Note that when the SAB 82289 is initialized by the RESET input it does not wait until the end of the current bus cycle to reset. Any bus cycle in process when RESET goes active will be aborted by the arbiter. Although the $\overline{\text{INIT}}$ signal will also interrupt an active bus cycle, the arbiter can request the Multibus and complete the bus cycle when $\overline{\text{INIT}}$ goes inactive.

As mentioned in the pin description and figure 11 the functions of the $\overline{\text{S0}}$/HOLD pin and the release mode ($\overline{\text{ALWAYS}}$/$\overline{\text{CBQLCK}}$ pin) are programmed at the falling edge of RESET.

Absolute Maximum Ratings [1)]

Ambient Temperature Under Bias	0 to 70°C
Storage Temperature	−65 to +150°C
Voltage on Any Pin With Respect to GND	−0.5 to +7V
Power Dissipation	1 W

DC Characteristics

TA = 0 to 70°C, VCC = 5V ±5%

Symbol	Parameter	Limit values		Unit	Test condition
		min.	max.		
VIL	Input low Voltage	−0.5	0.8	V	–
VIH	Input High Voltage	2.0	VCC+0.5	V	–
VILC	CLK Input Low Voltage	−0.5	0.6	V	–
VIHC	CLK Input High Voltage	3.8	VCC+1.0	V	–
VOL	Output Low Voltage: $\overline{BUSY}$, $\overline{CBRQ}$ $\overline{BPRO}$, $\overline{BREQ}$, $\overline{AEN}$ $\overline{LLOCK}$	–	0.45	V	 IOL = 32mA IOL = 16 mA IOL = 5mA
VOH	Output High Voltage	2.4	–	V	IOH = 400 µA
ILI	Input Leakage Current	–	±10 ±1	µA mA	0.45 V ≤VIN ≤VCC 0 V ≤VIN <0.45 V
ILO	Output Leakage Current	–	±10	µA	0.45 V ≤VOUT ≤VCC
ICC	Power Supply Current	–	120	mA	–
CCLK	CLK, $\overline{BCLK}$ Input Capacitance	–	12	pF	fC = 1 MHz
CIN	Input Capacitance	–	10	pF	fC = 1 MHz
CIO	Input/Output Capacitance	–	20	pF	fC = 1 MHz

1) Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

AC Characteristics SAB 82289

TA = 0 to 70°C, VCC = 5V ±5%
AC timings are referenced to 0.8 and 2.0 V points of signals as illustrated in data sheet waveforms, unless otherwise noted.

Symbol	Parameter	Limit values * 8 MHz		Unit	Test condition	Shown in Figure
		min.	max.			
1	CLK Cycle Period	62	t5+50	ns	–	13
2	CLK Low Time	15	230	ns	at 1.0 V	13
3	CLK High Time	20	235	ns	at 3.6V	13
4	CLK Rise/Fall Time	–	10	ns	1.0 to 3.6V	13
5	$\overline{\text{BCLK}}$ Cycle Time	100	∞	ns	–	13
6	$\overline{\text{BCLK}}$ High/Low Time	30	–	ns	–	13
7	$\overline{\text{S0}}$/HOLD, $\overline{\text{S1}}$, M/$\overline{\text{IO}}$ Setup	22	–	ns	–	13, 22
8	$\overline{\text{S0}}$/HOLD, $\overline{\text{S1}}$, M/$\overline{\text{IO}}$ Hold	1	–	ns	–	13, 22
9	$\overline{\text{READY}}$ Setup	38	–	ns	–	13
10	$\overline{\text{READY}}$ Hold Time	25	–	ns	–	13
11	$\overline{\text{LOCK}}$, SYSB/$\overline{\text{RESB}}$ Setup Time	20	–	ns	–	13, 18
12	$\overline{\text{LOCK}}$, SYSB/$\overline{\text{RESB}}$ Hold Time	1	–	ns	–	13, 18
13	RESET Setup Time	20	–	ns	–	19
14	RESET Hold Time	1	–	ns	–	19
15	RESET Active Pulse Width	16	–	CLKs	–	19
16	$\overline{\text{INIT}}$ Setup Time	45	–	ns	9)	20
17	$\overline{\text{INIT}}$ Hold Time	1	–	ns	9)	20
18	$\overline{\text{INIT}}$ Active Pulse Width	3 (t1) +3(t14)	–	ns	–	20
19	$\overline{\text{BUSY}}$, $\overline{\text{BPRN}}$, $\overline{\text{CBRQ}}$, $\overline{\text{CBQLCK}}$/$\overline{\text{ALWAYS}}$ Setup to $\overline{\text{BCLK}}$ (or to RESET)	20	–	ns	–	13, 15, 21, 22
20	$\overline{\text{BUSY}}$, $\overline{\text{BPRN}}$, $\overline{\text{CBRQ}}$, $\overline{\text{CBQLCK}}$/$\overline{\text{ALWAYS}}$ Hold to $\overline{\text{BCLK}}$ (or to RESET)	1	–	ns	–	13, 15, 21, 22
21	$\overline{\text{BCLK}}$ to $\overline{\text{BREQ}}$ Delay	–	30	ns	1)	13,14,22
22	$\overline{\text{BCLK}}$ to $\overline{\text{BPRO}}$ Delay	–	35	ns	2)	17
23	$\overline{\text{BPRN}}$ to $\overline{\text{BPRO}}$ Delay	–	25	ns	2)	17
24	$\overline{\text{BCLK}}$ to $\overline{\text{BUSY}}$ Active Delay	1	60	ns	3)	13, 22

For notes see next page.

AC Characteristics SAB 82289 (continued)

Symbol	Parameter	Limit values * 8 MHz		Unit	Test condition	Shown in Figure
		min.	max.			
25	$\overline{BCLK}$ to $\overline{BUSY}$ Float Delay	–	35	ns	4)	13, 14
26	$\overline{BCLK}$ to $\overline{CBRQ}$ Active Delay	–	55	ns	5)	13, 22
27	$\overline{BCLK}$ to $\overline{CBRQ}$ Float Delay	–	35	ns	4)	13,20,22
28	$\overline{BCLK}$ to $\overline{AEN}$ Active Delay	1	25	ns	6)	13
29	CLK to $\overline{AEN}$ Inactive Delay	3	25	ns	6)	13, 14
30	CLK to $\overline{LLOCK}$ Delay	–	20	ns	7)	18
31	RESET to $\overline{LLOCK}$ Delay	–	35	ns	7)	19
32	CLK to $\overline{BCLK}$ Setup Time	38	–	ns	8)	13,16,20
33	$\overline{BCLK}$ to $\overline{AEN}$ Output Delay	1	30	ns	6)	22

*) Preliminary

1) $\overline{BREQ}$ load CL = 60 pF

2) $\overline{BPRO}$ load CL = 60 pF

3) $\overline{BUSY}$ load CL = 300 pF

4) Float condition occurs when output current is less than ILO in magnitude

5) $\overline{CBRQ}$ load CL = 300 pF

6) $\overline{AEN}$ load CL = 150 pF

7) $\overline{LLOCK}$ load CL = 60 pF

8) In actual use, CLK and $\overline{BCLK}$ are usually asynchronous to each other. However, for component testing purposes this specification is required to assure signal recognition at specific CLK and $\overline{BCLK}$ edges.

9) $\overline{INIT}$ is asynchronous to CLK and to $\overline{BCLK}$. However, for component testing purposes, this specification is required to assure signal recognition at specific CLK and $\overline{BCLK}$ edges.

AC Characteristics SAB 82289-6

TA = 0 to 70°C, VCC = 5V ±5%
AC timings are referenced to 0.8 and 2.0 V points of signals as illustrated in data sheet waveforms, unless otherwise noted.

Symbol	Parameter	Limit values * 6 MHz		Unit	Test condition	Shown in Figure
		min.	max.			
1	CLK Cycle Period	83	t5+50	ns	–	13
2	CLK Low Time	20	225	ns	at 1.0 V	13
3	CLK High Time	25	230	ns	at 3.6 V	13
4	CLK Rise/Fall Time	–	10	ns	1.0 to 3.6 V	13
5	$\overline{\text{BCLK}}$ Cycle Time	100	∞	ns	–	13
6	$\overline{\text{BCLK}}$ High/Low Time	30	–	ns	–	13
7	$\overline{\text{S0}}$/HOLD, $\overline{\text{S1}}$, M/$\overline{\text{IO}}$ Setup	28	–	ns	–	13, 22
8	$\overline{\text{S0}}$/HOLD, $\overline{\text{S1}}$, M/$\overline{\text{IO}}$ Hold	1	–	ns	–	13, 22
9	$\overline{\text{READY}}$ Setup	50	–	ns	–	13
10	$\overline{\text{READY}}$ Hold Time	35	–	ns	–	13
11	$\overline{\text{LOCK}}$, SYSB/$\overline{\text{RESB}}$ Setup Time	28	–	ns	–	13, 18
12	$\overline{\text{LOCK}}$, SYSB/$\overline{\text{RESB}}$ Hold Time	1	–	ns	–	13, 18
13	RESET Setup Time	28	–	ns	–	19
14	RESET Hold Time	1	–	ns	–	19
15	RESET Active Pulse Width	16	–	CLKs	–	19
16	$\overline{\text{INIT}}$ Setup Time	45	–	ns	9)	20
17	$\overline{\text{INIT}}$ Hold Time	1	–	ns	9)	20
18	$\overline{\text{INIT}}$ Active Pulse Width	3 (t1) +3(t14)	–	ns	–	20
19	$\overline{\text{BUSY}}$, $\overline{\text{BPRN}}$, $\overline{\text{CBRQ}}$, $\overline{\text{CBQLCK}}$/ALWAYS Setup to $\overline{\text{BCLK}}$ (or to RESET)	20	–	ns	–	13, 15, 21, 22
20	$\overline{\text{BUSY}}$, $\overline{\text{BPRN}}$ $\overline{\text{CBRQ}}$, $\overline{\text{CBQLCK}}$/ALWAYS Hold to $\overline{\text{BCLK}}$ (or to RESET)	1	–	ns	–	13, 15, 21, 22
21	$\overline{\text{BCLK}}$ to $\overline{\text{BREQ}}$ Delay	–	30	ns	1)	13,14,22
22	$\overline{\text{BCLK}}$ to $\overline{\text{BPRO}}$ Delay	–	35	ns	2)	17
23	$\overline{\text{BPRN}}$ to $\overline{\text{BPRO}}$ Delay	–	25	ns	2)	17
24	$\overline{\text{BCLK}}$ to $\overline{\text{BUSY}}$ Active Delay	1	60	ns	3)	13, 22

For notes see next page.

AC Characteristics SAB 82289-6 (continued)

Symbol	Parameter	Limit values * 6 MHz		Unit	Test condition	Shown in Figure
		min.	max.			
25	$\overline{\text{BCLK}}$ to $\overline{\text{BUSY}}$ Float Delay	–	35	ns	4)	13, 14
26	$\overline{\text{BCLK}}$ to $\overline{\text{CBRQ}}$ Active Delay	–	55	ns	5)	13, 22
27	$\overline{\text{BCLK}}$ to $\overline{\text{CBRQ}}$ Float Delay	–	35	ns	4)	13,20,22
28	$\overline{\text{BCLK}}$ to $\overline{\text{AEN}}$ Active Delay	1	25	ns	6)	13
29	CLK to $\overline{\text{AEN}}$ Inactive Delay	3	25	ns	6)	13, 14
30	CLK to $\overline{\text{LLOCK}}$ Delay	–	20	ns	7)	18
31	RESET to $\overline{\text{LLOCK}}$ Delay	–	35	ns	7)	19
32	CLK to $\overline{\text{BCLK}}$ Setup Time	38	–	ns	8)	13,16,20
33	$\overline{\text{BCLK}}$ to $\overline{\text{AEN}}$ Output Delay	1	30	ns	6)	22

*) Preliminary

1) $\overline{\text{BREQ}}$ load CL = 60 pF

2) $\overline{\text{BPRO}}$ load CL = 60 pF

3) $\overline{\text{BUSY}}$ load CL = 300 pF

4) Float condition occurs when output current is less than ILO in magnitude

5) $\overline{\text{CBRQ}}$ load CL = 300 pF

6) $\overline{\text{AEN}}$ load CL = 150 pF

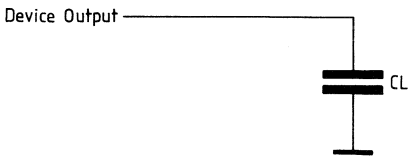
7) $\overline{\text{LLOCK}}$ load CL = 60 pF

8) In actual use, CLK and $\overline{\text{BCLK}}$ are usually asynchronous to each other. However, for component testing purposes this specification is required to assure signal recognition at specific CLK and $\overline{\text{BCLK}}$ edges.

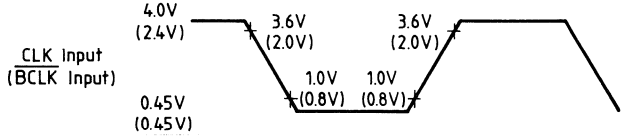
9) $\overline{\text{INIT}}$ is asynchronous to CLK and to $\overline{\text{BCLK}}$. However, for component testing purposes, this specification is required to assure signal recognition at specific CLK and $\overline{\text{BCLK}}$ edges.

AC Testing Waveforms

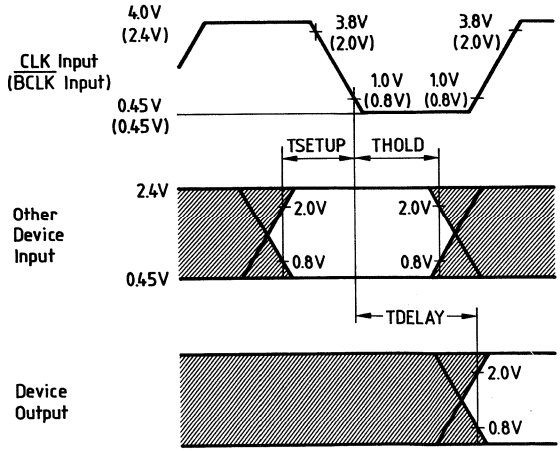
Test Loading on Outputs



Drive and Measurement Points – CLK Input ($\overline{BCLK}$ Input)



Setup, Hold and Delay Time Measurement – General



Waveforms

The waveforms (figures 13 to 21) show the timing relationships of the inputs and the outputs and do not show all possible transitions of all signals in all modes. Instead, all signal timing relationships are shown via the general cases. Special cases are shown when needed.

To find the timing specification for a signal transition in a particular mode, first look for a special case in the waveforms. If no special case applies, then use a timing specification for the same or related function in another mode.

The SAB 82289 bus arbiter serves as an interface between the SAB 80286 subsystem which operates synchronously to the CLK signal and Multibus which operates synchronously to $\overline{\text{BCLK}}$ signal. CLK and $\overline{\text{BCLK}}$ generally operate asynchronously to each other and at different frequencies. Thus, the exact clock period in which an input synchronous to one clock will cause a response synchronous to the other clock depends on the relative phase and frequency of CLK and $\overline{\text{BCLK}}$ at the time the input is sensed.

One strict relation between CLK and $\overline{\text{BCLK}}$ must be maintained for proper Multibus arbitration. If the CLK period is too long relative to $\overline{\text{BCLK}}$ period (t1 greater than t5 +50 ns), another arbiter could gain control of the system bus before this arbiter has released $\overline{\text{AEN}}$ synchronously to its CLK. This situation arises since the release of $\overline{\text{AEN}}$ is synchronous to the next falling CLK edge after the processor cycle ends but the release of $\overline{\text{BREQ}}$ and $\overline{\text{BUSY}}$ is synchronous to the next falling $\overline{\text{BCLK}}$ edge after the processor cycle ends. In practice, any CLK frequency greater than 6.66 MHz (i.e. SAB 80286 processor speeds greater than 3.33 MHz) will avoid conflict with a 10 MHz $\overline{\text{BCLK}}$. Therefore all SAB 80286 speed selections are Multibus compatible.

Figure 13
Multibus Acquisition and Always-Release Operation

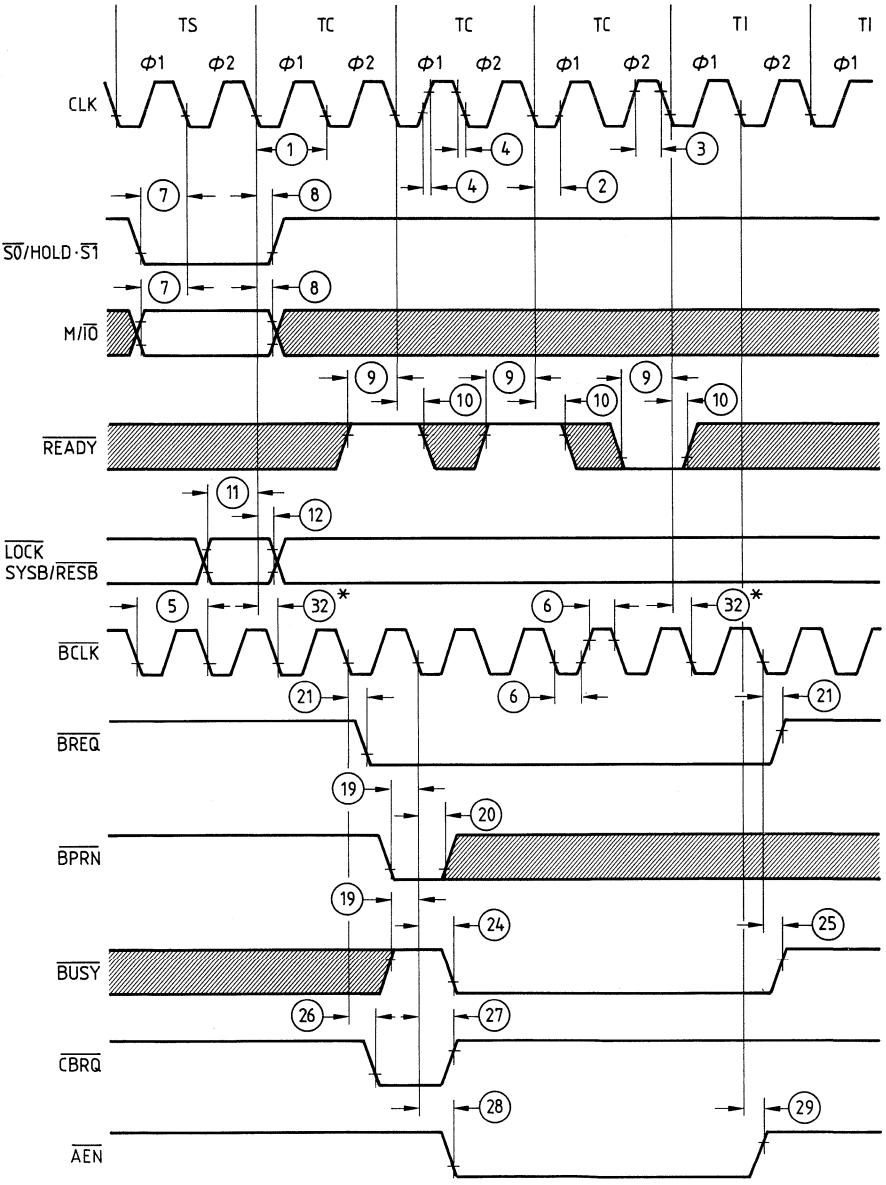


Figure 14
Multibus Release due to $\overline{\text{BPRN}}$ Inactive

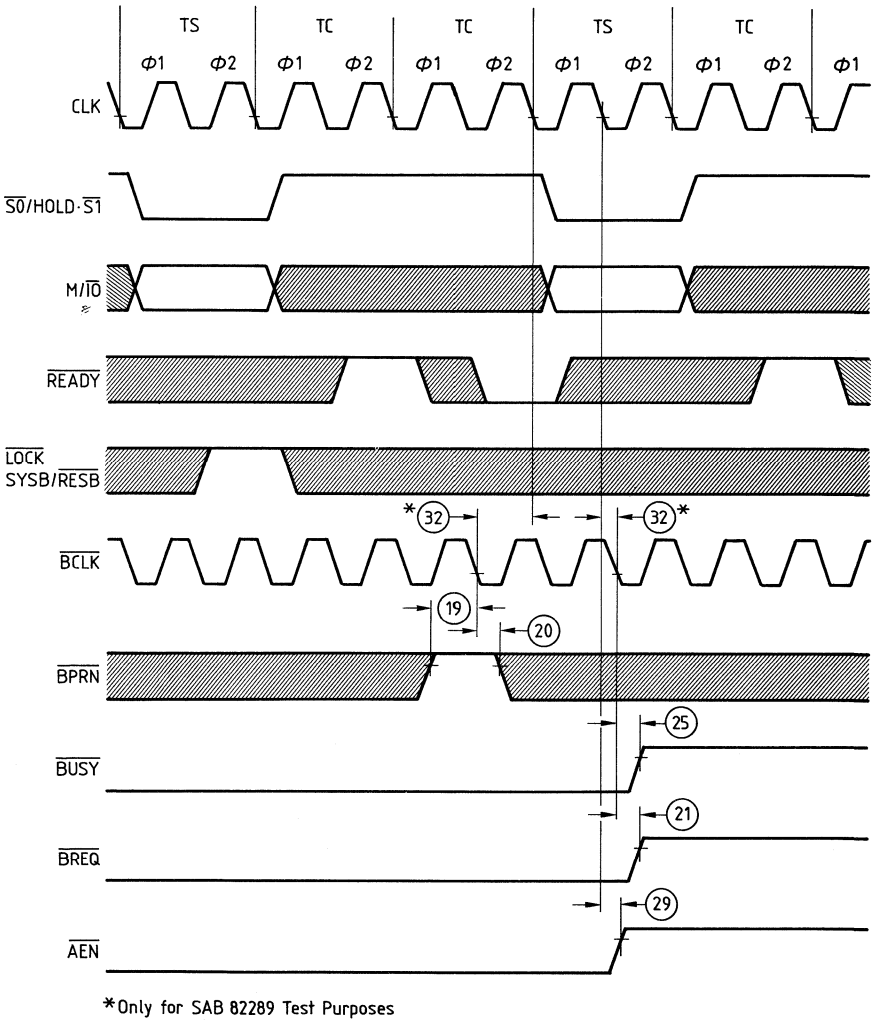


Figure 15
Multibus Release due to $\overline{\text{CBRQ}}$ Active

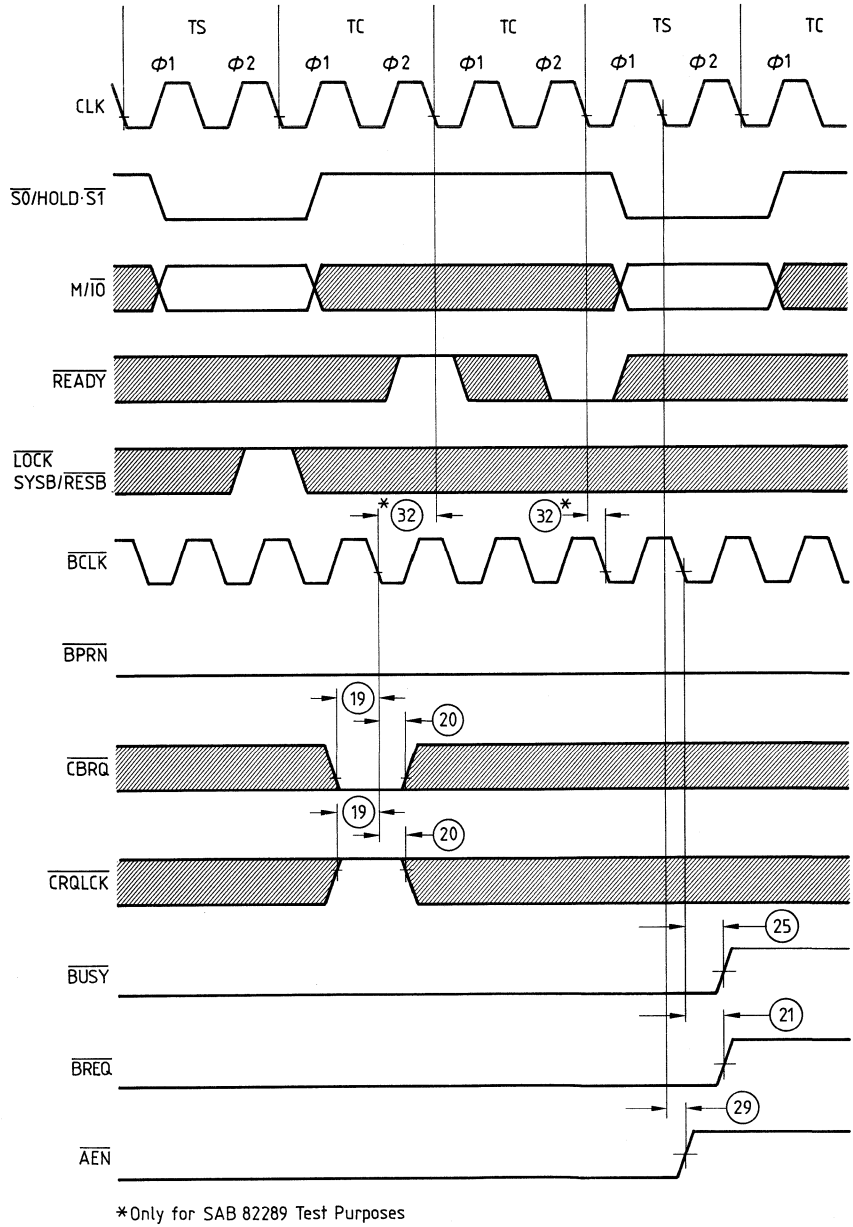
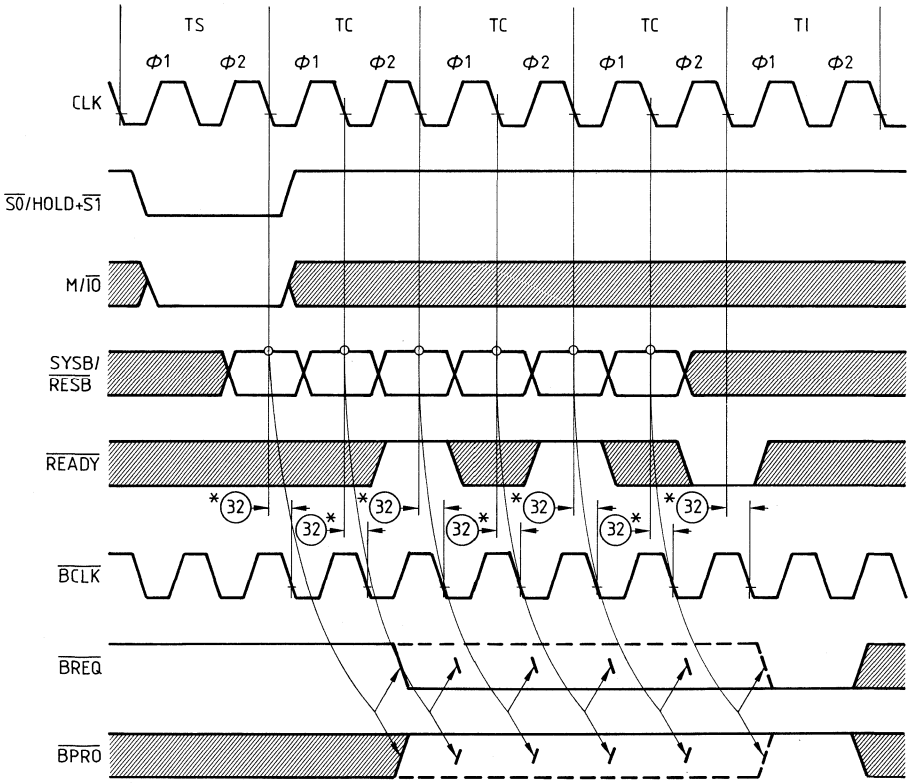


Figure 16
Multibus Acquisition during SAB 80286 INTA Cycles



*Only for SAB 82289 test purposes

Figure 17
$\overline{\text{BPRN}}$ to $\overline{\text{BPRO}}$ Timing Relationship

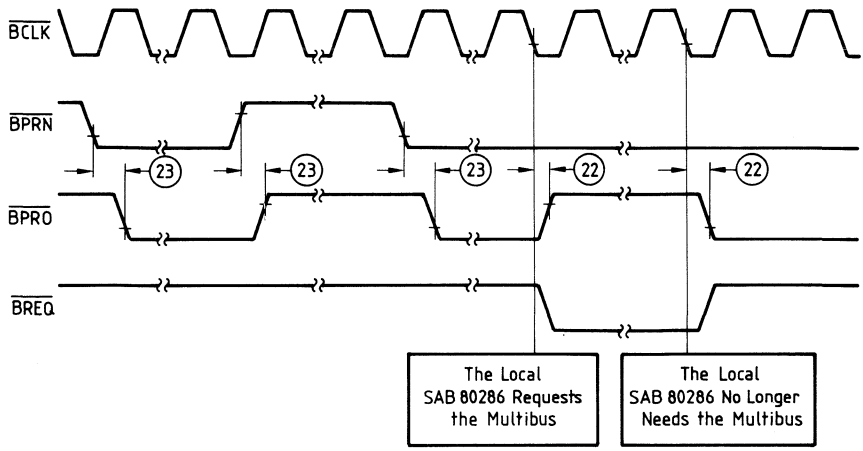


Figure 18
SAB 80286 $\overline{\text{LOCK}}$ and SAB 82289 $\overline{\text{LLOCK}}$ Relationship

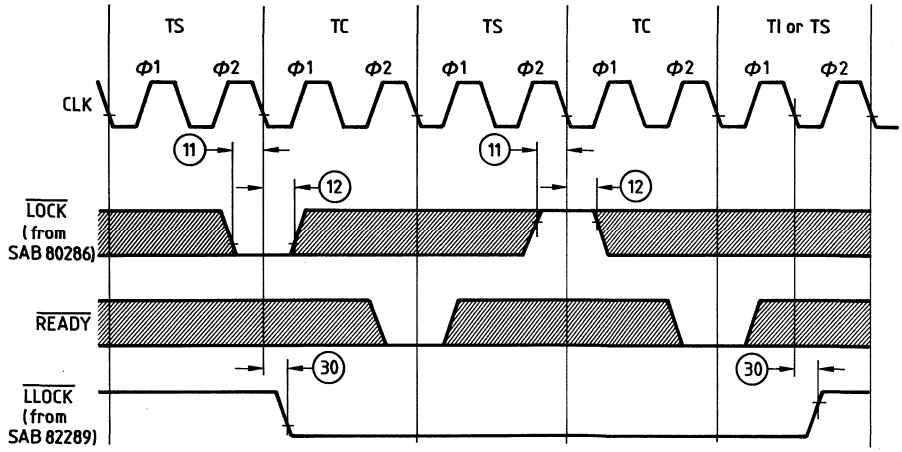
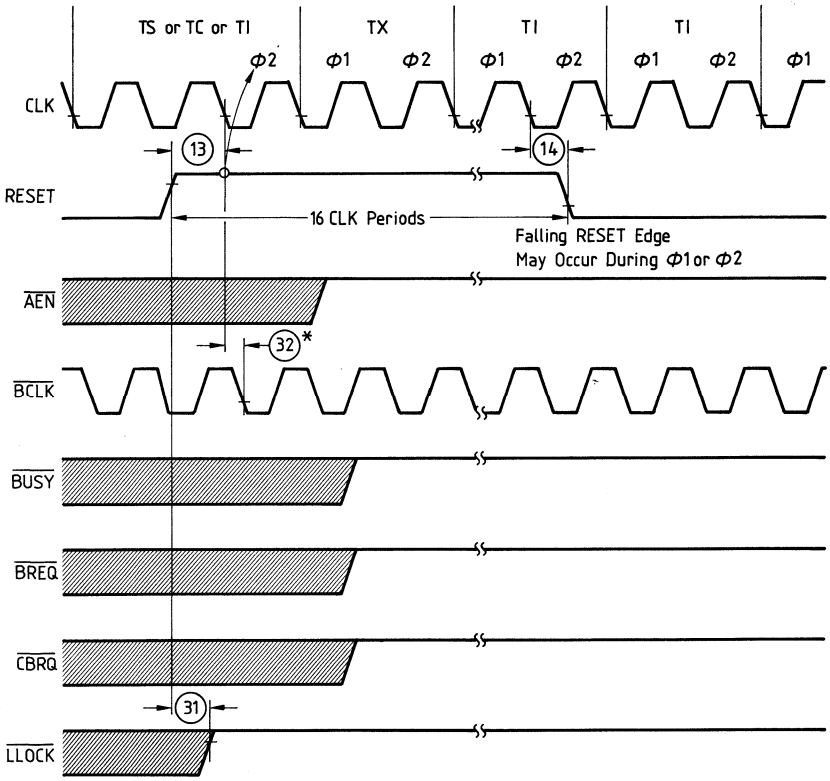
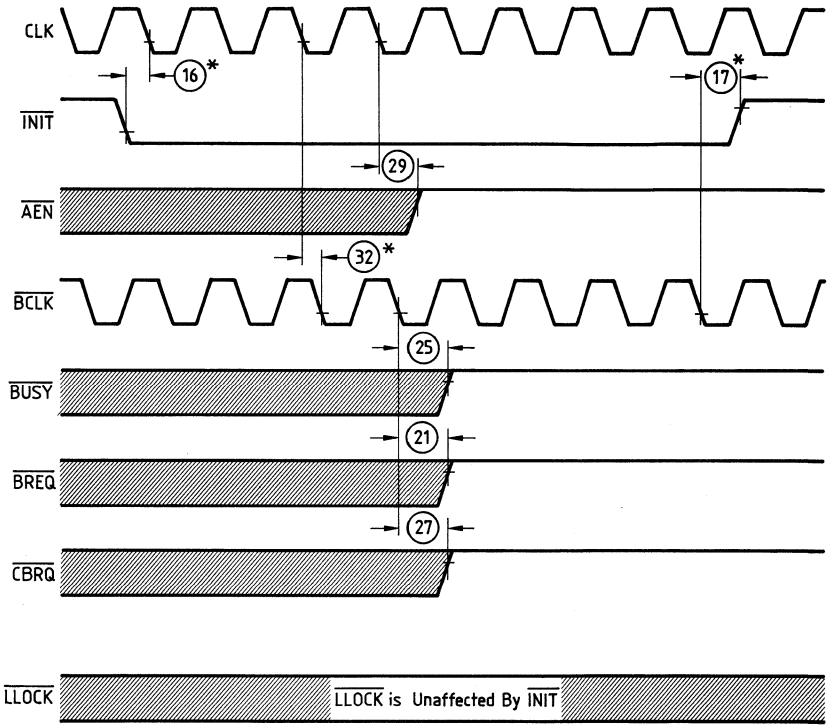


Figure 19
RESET Active Pulse



*For SAB 82289 Test Purposes Only.

Figure 20
$\overline{\text{INIT}}$ Active Pulse



*For SAB 82289 Test Purposes Only.

Figure 21
Programming the Always-Release/Common-Bus-Request-Release Option

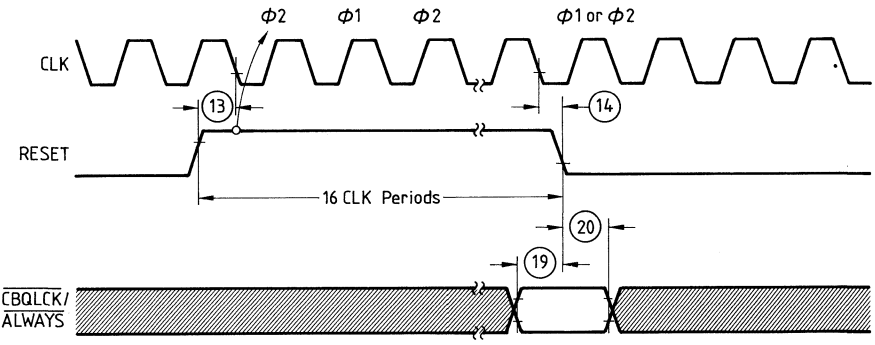
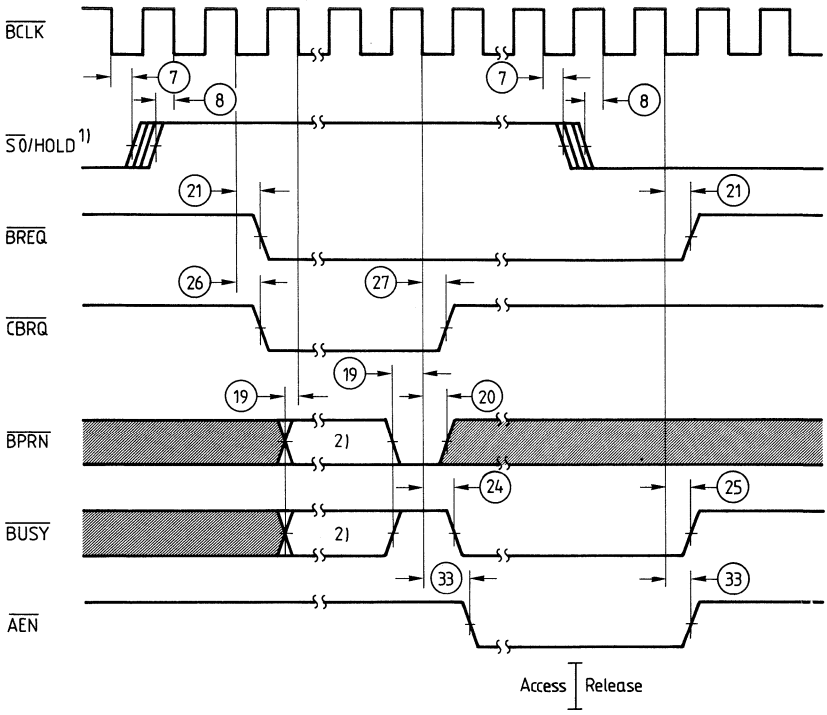


Figure 22
Multibus Arbitration Activated by HOLD Input



1) Used as HOLD input

2) Valid conditions are $\overline{\text{BPRN}}$ high or $\overline{\text{BUSY}}$ low. If one of the two conditions is fulfilled, the level of the other signal ($\overline{\text{BUSY}}$ or $\overline{\text{BPRN}}$) may be arbitrary

Ordering Information

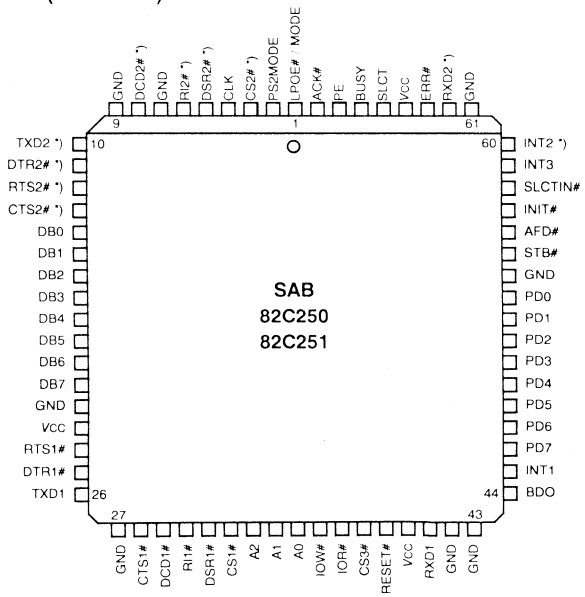
Type	Ordering code	Description
SAB 82289-P	Q67020-Y77	Bus Arbiter 16 MHz (plastic)
SAB 82289-6-P	Q67120-Y111	Bus Arbiter 12 MHz (plastic)

SAB 82C250/SAB 82C251
Advanced Peripheral Interface Controller

Advance Information

- SAB 82C250: Dual channel asynchronous serial controller
- SAB 82C251: One channel asynchronous serial controller
- Implementation of the standard PC-XT™/AT™/PS/2™ application
- SAB 16C450 compatible serial controller with improved 16550 AC characteristics
- 8-bit bi-directional parallel port
- Supports fully Centronics printer interface with bi-directional function (16C452/16C451-compatible mode)
- PS/2 register compatible mode for bi-directional parallel port
- Fully programmable serial interface characteristics for each serial channel
  - 5/6/7/8-bit character length
  - Parity generation and detection
  - Error reporting capabilities
  - Common external clock input for the baud rate generators
  - Baud rate up to 512 KBaud
- Mode selection by pin strapping
- I_{OL}/I_{OH} for printer data pins: 24 mA / -15 mA
- CMOS implementation for high speed and low power requirements
- 68-pin plastic leaded chip carrier package (PL-CC-68)

Figure 1
Pin Configuration (PL-CC-68)



*) These pins are not connected (N.C.) at the SAB 82C251

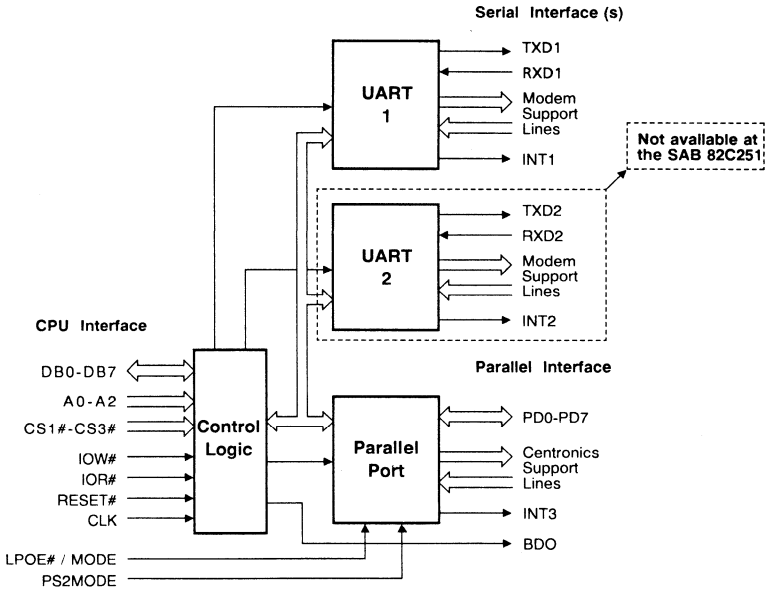
General Information

The SAB 82C250/82C251 are advanced serial/parallel interface controllers for standard and IBM PC-XT™/PC-AT™/PS/2™ compatible interface applications. Their serial channels provide the functionality of the SAB 16C450 universal asynchronous receiver/transmitter. Only one serial channel is available in the SAB 82C251.

The SAB 82C250/82C251 also provide a parallel port, which fully supports the standard Centronics printer interface, as well as the bi-directional feature of the PS/2 parallel port implementation. This feature allows full PS/2 register compatible bi-directional input/output operations in the parallel port.

The SAB 82C250/82C251 are fabricated in Siemens AC-MOS technology and are available in the 68-pin plastic-leaded-chip-carrier (PL-CC-68) package. The SAB 82C250/82C251 are compatible to the standard 16C452/16C451 serial-parallel controllers.

Figure 2
Block Diagram



Pin Definitions and Functions

Symbol	Pin	Input (I) Output (O)	Function
Bus Interface			
CLK	4	I	**Serial Clock:** External clock input to the baud rate generator(s) of the serial channel(s).
DB0-DB7	14-21	I/O	**Data Bus 0-7:** The data bus provides 8 bi-directional IO-lines for the transfer of data, status and control information between the SAB 82C250/82C251 and the CPU. These lines are inputs except during read operations. DB0 is the least significant bit (LSB) and is the first serial data bit to be received or transmitted.
A0-A2	35-33	I	**Address lines 0-2:** The address lines are used together with the chip select inputs to select the internal registers of the SAB 82C250/82C251.
IOW#	36	I	**IO Write Strobe#:** This is an active low input which causes the data at the data bus to be written into the selected registers of the SAB 82C250/82C251. The destination of the data depends on the state of CS1#, CS2#, CS3# and A0–A2.
IOR#	37	I	**IO Read Strobe#:** This is an active low input which causes the selected IO-device to output data to the data bus DB0-DB7. The source of the data depends on the state of CS1#, CS2#, CS3# and A0-A2.
RESET#	39	I	**RESET#:** With RESET# = low the two serial ports and the parallel port are put into an initial state.
BDO	44	O	**Bus Buffer Output:** This line goes high whenever either serial channel registers or the parallel port registers are read. This output can be used to control system bus drivers.
V_{CC}	23,40, 64	–	**Power Supply (+ 5 V)**
GND	7, 9, 22, 27, 42, 43, 54, 61	–	**Ground (0 V)**

Pin Definitions and Functions (continued)

Symbol	Pin [1]	Input (I) Output (O)	Function
Serial Interface			
RTS1# RTS2#	24 12	O	**Request To Send#:** When active (low), the UART indicates that data is ready to be transmitted. The RTS# pins are set low by writing a 1 to the appropriate bits of the modem control registers.
DTR1# DTR2#	25 11	O	**Data Terminal Ready#:** When active (low), the UART indicates that it is ready to receive data. The RTS# pins are set low by writing a 1 to the appropriate bits of the modem control registers.
TXD1 TXD2	26 10	O	**Serial Data Outputs:** These lines are the serial data outputs from the UARTs transmitter circuitry.
CTS1# CTS2#	28 13	I	**Clear To Send#:** The states of these pins are reflected in bits of the modem control registers. When CTS# = low, data can be transmitted at the TXD lines.
DCD1# DCD2#	29 8	I	**Data Carrier Detect#:** The states of these pins are reflected in bits of the modem control registers. DCD# = low indicates that the data carrier has been detected by the modem.
RI1# RI2#	30 6	I	**Ring Indicator#:** The states of these pins are reflected in bits of the modem control registers. When low, RI# = low indicates that a telephone ringing signal has been received by the modem.
DSR1# DSR2#	31 5	I	**Data Set Ready#:** The states of these pins are reflected in bits of the modem control registers. When DSR# = low, a modem is indicating that it is ready to exchange data with the associated UART.
CS1# CS2#	32 3	I	**Serial Channel Chip Select#:** CS1# and CS2# are the enable lines for the read and write control signals of the SAB 82C250/82C251 serial channels. Only one of the 3 chip select inputs may be active at a time!

1) All signals with extension "2" are not available at the SAB 82C251. The corresponding pins are not connected (N.C.) at the SAB 82C251.

Pin Definitions and Functions (continued)

Symbol	Pin 1)	Input (I) Output (O)	Function
Serial Interface			
RXD1 RXD2	41 62	I	**Serial Data Input:** These lines are the serial data inputs to the SAB 82C250/82C251 serial channels receiver circuitry.
INT1 INT2	45 60	O	**Serial Channel Interrupts:** These tristate outputs are enabled by the modem control register bits 2. An occurring interrupt condition of the serial channels will activate (set high) these interrupt lines. The interrupts are reset (low) upon an appropriate service or a reset.
Parallel Interface			
LPOE# /MODE	1	I	**Printer Output Enable#/Mode Select:** In normal mode (PS2MODE = low) this input signal enables the PD0–PD7 outputs. With PS2MODE = high, this input selects between the uni- and bi-directional printer port operation.
PS2MODE	2	I	**Enable Printer PS/2 Mode:** With high at this input, the PS/2 mode of the printer interface is selected. A low level selects normal Centronics compatible mode operation of the parallel interface.
CS3#	38	I	**Parallel Port Chip Select#:** CS3# = low enables the read and write control lines of the SAB 82C250/ 82C251 parallel port section. Only one of the 3 chips select inputs may be active at a time!
PD0–PD7	53–46	I/O	**Parallel Data Bits (0–7):** These lines provide the byte-wide input or output port of the SAB 82C250/ 82C251.
STB#	55	O	**Line Printer Strobe#:** When low, this line provides the line printer with a signal to latch data which is currently available at the parallel port.

1) All signals with extension "2" are not available at the SAB 82C251. The corresponding pins are not connected (N.C.) at the SAB 82C251.

Pin Definitions and Functions (continued)

Symbol	Pin	Input (I) Output (O)	Function
Parallel Interface			
AFD#	56	O	**Line Printer Autofeed#:** With AFD# = low the printer does a continous form feed of the paper.
INIT#	57	O	**Line Printer Initialize:** This signal (low) causes the printer to start its initialization routine.
SLCTIN#	58	O	**Line Printer Select#:** When active (low), this line selects the line printer.
INT3	59	O	**Interrupt Printer Port:** This interrupt output is activated (high) by a low ACK# signal. The function of INT3 is controlled by bit 4 of the parallel port control register.
ERR#	63	I	**Line Printer Error#:** If this input goes low, the line printer reports an error condition.
SLCT	65	I	**Line Printer Select:** This is an input line from the line printer that goes high when the printer has been selected.
BUSY	66	I	**Line Printer Busy:** This is an input line from the line printer that goes high when the printer has a local operation in progress.
PE	67	I	**Line Printer Paper Empty:** This is an input line from the line printer that goes high when the printer runs out of paper.
ACK#	68	I	**Line Printer Acknowledge:** This input goes low if a successful data transfer has occurred.

Serial Channel Description

The SAB 82C250 contains two serial channels. In the SAB 82C251, the second serial channel of the SAB 82C250 is not available. In the following text, no number extensions are used when referring to the signal names. In the SAB 82C250/82C251 the number extension "1" refers to serial channel 1. In the SAB 82C250 the number extension "2" refers to serial channel 2.

Serial Port Register Addressing

When CS1# is low, registers for serial channel 0 can be accessed, and when CS2# is low, registers for serial channel 1 can be accessed. No more than one chip select line should ever be low at a time (invalid condition). Address lines A0, A1 and A2 are used to select the appropriate register of the serial channels (Table 1). The Divisor Latch Access Bit (DLAB) in table 1 is the MSB of the Line Control Register. DLAB must be set by software to access the baud rate generator divisor latches.

Table 1
Serial Channel Register Addressing

DLAB	A_2	A_1	A_0	Register
0	0	0	0	Receiver Buffer Register (RBR, read) Transmitter Holding Register (THR, read)
0	0	0	1	Interrupt Enable Register (IER)
X	0	1	0	Interrupt Identification Register (IIR, read only)
X	0	1	1	Line Control Register (LCR)
X	1	0	0	Modem Control Register (MCR)
X	1	0	1	Line Status Register (LSR)
X	1	1	0	Modem Status Register (MSR)
X	1	1	1	Scratch Pad Register
1	0	0	0	Divisor Latch (least significant byte)
1	0	0	1	Divisor Latch (most significant byte)

Register Description

Three types of internal registers are used in the serial channels of the SAB 82C250/82C251: control, status and data registers. The control registers are two divisor latches for the baud rate generator, the Line Control Register (LCR), the Interrupt Enable Register (IER), and the Modem Control Register (MCR). Two status registers are available: Line Status Register (LSR) and Modem Status Register (MSR).
The data registers are the Receiver Buffer Register (RBR), for read operations, and the Transmitter Holding Register (THR), for write operations. Table 2 shows the contents of the SAB 82C250/82C251 serial channel registers in detail.

Receiver/Transmitter Buffer Register

The Receiver Buffer Register and the Transmitter Buffer Register are data registers, which hold between five and eight bits of data. If less than eight data bits are transmitted, then the data will be right justified to the LSB. Bit 0 of a data byte is always the first serial data bit received and transmitted. The SAB 82C250/82C251 serial channel data registers are double-buffered so that read and write operations can be performed at the same time the UART is performing the serial-to-parallel and parallel-to-serial conversion.

Table 2
Summary of Registers

	Register Address				
	0 (DLAB = 0)	**0 (DLAB = 0)**	**1 (DLAB = 0)**	**2**	**3**
Bit no.	Receiver Buffer Register (read only)	Transmitter Holding Register (write only)	Interrupt Enable Register	Interrupt Identification Register (read only)	Line Control Register
	RBR	THR	IER	IIR	LCR
0	Data bit 0 [1]	Data bit 0 [1]	Received data available (RDI)	"0" if interrupt is pending (IP)	Word length select bit 0 (WSL0)
1	Data bit 1	Data bit 1	Transmitter holding register empty (THI)	Interrupt ID bit 0 (IIB0)	Word length select bit 1 (WLS1)
2	Data bit 2	Data bit 2	Receiver line status (RSI)	Interrupt ID bit 1 (IIB1)	Number of stop bits (STB)
3	Data bit 3	Data bit 3	Modem status (MSI)	0	Parity enable (PEN)
4	Data bit 4	Data bit 4	0	0	Even parity select (EPS)
5	Data bit 5	Data bit 5	0	0	Stick parity
6	Data bit 6	Data bit 6	0	0	Set break
7	Data bit 7	Data bit 7	0	0	Divisor latch access bit (DLAB)

[1] Bit 0 is the least significant bit. It is the first bit serially transmitted or received.

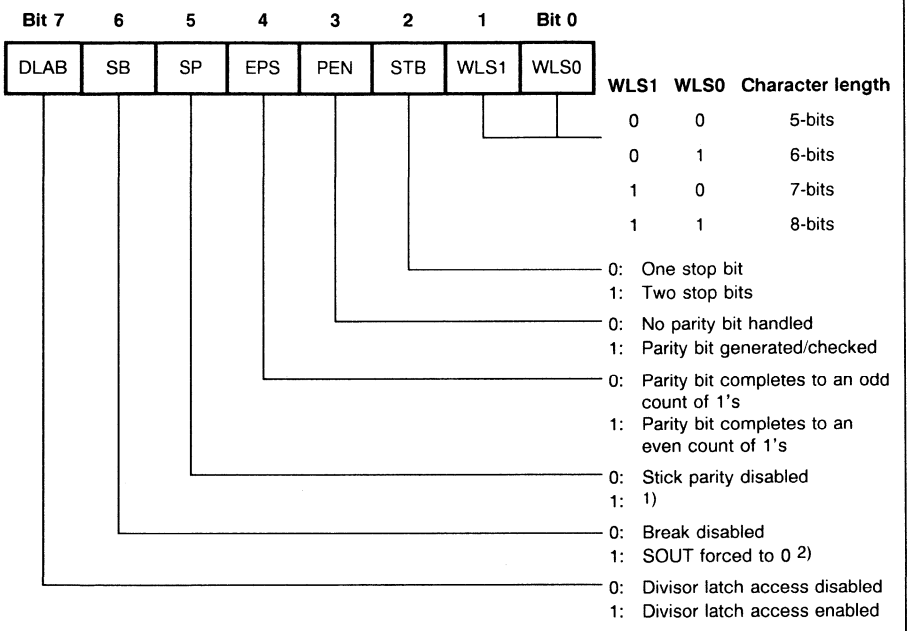
Table 2
Summary of Registers (continued)

	Register Address					
	4	**5**	**6**	**7**	**0 (DLAB = 1)**	**1 (DLAB = 1)**
Bit no.	Modem Control Register	Line Status Register	Modem Status Register	Scratch Register	Divisor Latch (LS)	Divisor Latch (MS)
	MCR	LSR	MSR	SCR	DLL	DLM
0	Data terminal ready (DTR)	Data ready (DR)	Delta clear to send (DCTS)	Bit 0	Bit 0	Bit 8
1	Request to send (RTS)	Overrun error (OE)	Delta data set ready (DDSR)	Bit 1	Bit 1	Bit 9
2	Not Connected (NC)	Parity error (PE)	Trailing edge ring indicator (TERI)	Bit 2	Bit 2	Bit 10
3	Interrupt Output Enable (IOE)	Framing error (FE)	Delta data carrier detect (DDCD)	Bit 3	Bit 3	Bit 11
4	Loop (LOOP)	Break interrupt (BI)	Clear to send (CTS)	Bit 4	Bit 4	Bit 12
5	0	Transmitter holding register (THRE)	Data set ready (DSR)	Bit 5	Bit 5	Bit 13
6	0	Transmitter empty (TEMT)	Ring indicator (RI)	Bit 6	Bit 6	Bit 14
7	0	0	Data carrier detect (DCD)	Bit 7	Bit 7	Bit 15

Line Control Register (LCR)

The format of the asynchronous data communication exchange and the access to the divisor latch is controlled via the line control register.

Figure 3
Line Control Register



1) This bit is the stick parity bit. When bits 3, 4 and 5 are logic 1, the parity bit is transmitted and checked as a logic 0. If bits 3 and 5 are 1 and bit 4 is logic 0 then the parity bit is transmitted and checked as a logic 1.

2) This feature enables the CPU to alert a terminal in a computer communications system. If the following sequence is used, no erroneous or extraneous characters will be transmitted because of the break.

   1. Load an all O's pad character in response to THRE.
   2. Set break after the next THRE.
   3. Wait for the transmitter to be idle, (TEMT = 1), and clear break when normal transmission is to be restored. During the break, the transmitter can be used as a character timer to accurately establish the break duration.

Programmable Baud Rate Generator

The serial channels of the SAB 82C250/82C251 contain a programmable baud rate generator, which is capable of taking any clock input from DC to 8.0 MHz and dividing it by any divisor from 2 to 2^{16}-1. The output frequency of the baud rate generator is 16 x the baud rate.

$$\text{Divisor} = \frac{\text{Input frequency at CLK}}{\text{Desired baud rate} \times 16}$$

Two 8-bit latches store the divisor in a 16-bit binary format. These divisor latches must be loaded during initialization to ensure proper operation of the baud rate generator. By loading one of the divisor latches, the 16-bit baud rate counter is immediately loaded. Tables 3, 4 and 5 provide decimal divisors to use with CLK clock frequencies of 1.8432 MHz, 3.072 MHz and 8 MHz, respectively. The maximum operating frequency of the baud rate generator is 8.0 MHz. In this case, the data rate can be 512 K baud maximum.

Table 3
Baud Rates Using 1.8432 MHz Clock

Desired baud rate	Decimal divisor used to generate 16 x clock	Percent error difference between desired and actual
50	2304	-
75	1536	-
110	1047	0.026
134.5	857	0.058
150	768	-
300	384	-
600	192	-
1200	96	-
1800	64	-
2000	58	0.69
2400	48	-
3600	32	-
4800	24	-
7200	16	-
9600	12	-
19200	6	-
38400	3	-
56000	2 [1]	2.86

[1] Smallest allowable divisor when using corresponding clock.

Table 4
Baud Rates Using 3.072 MHz Clock

Desired baud rate	Decimal divisor used to generate 16 × clock	Percent error difference between desired and actual
50	3840	–
75	2560	–
110	1745	0.026
134.5	1428	0.034
150	1280	–
300	640	–
600	320	–
1200	160	–
1800	107	0.312
2000	96	–
2400	80	–
3600	53	0.628
4800	40	–
7200	27	1.23
9600	20	–
19200	10	–
38400	5	–
56000	3 [1]	14.285

Table 5
Baud Rates Using 8 MHz Clock

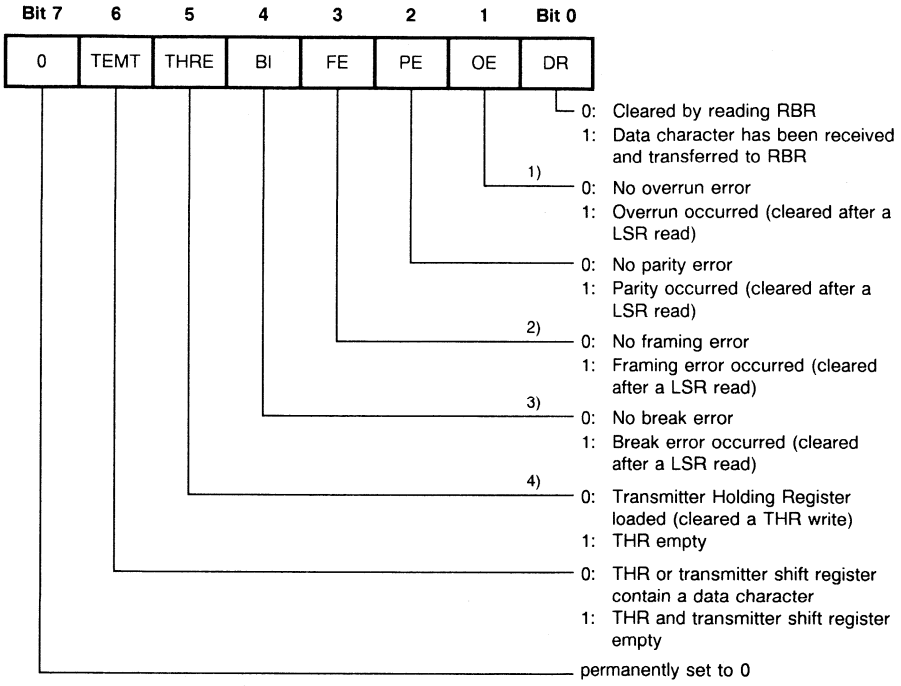
Desired baud rate	Decimal divisor used to generate 16 × clock	Percent error difference between desired and actual
50	10000	–
75	6667	0.005
110	4545	0.010
134.5	3717	0.013
150	3333	0.010
300	1667	0.020
600	833	0.040
1200	417	0.080
1800	277	0.080
2000	250	–
2400	208	0.160
3600	139	0.080
4800	104	0.160
7200	69	0.644
9600	52	0.160
19200	26	0.160
38400	13	0.160
56000	9	0.790
128000	4	2.344
256000	2	2.344
512000	1 [1]	2.344

[1] Smallest allowable divisor when using corresponding clock.

Line Status Register (LSR)

This 8-bit register provides the CPU with status information concerning the data transfer.

Figure 4
Line Status Register



Bits 1 through 4 of LSR are the error conditions that produce a receiver line status interrupt whenever any of the corresponding conditions are detected and the interrupt is enabled.

1) This bit is the overrun error (OE) indicator. Bit 1 indicates that data in the receiver buffer register was not read by the CPU before the transfer of the next character into the receiver buffer register, thus destroying the previous character.

2) The UART will try to resynchronize after a framing error. Therefore, it assumes that the framing error was due to the next start bit, samples this "start" bit twice and then takes in the "data".

3) This bit is the break interrupt (BI) indicator. BI is set to a logic 1 whenever the received data input is held in the spacing (logic 0) state for longer than a full word transmission time (i.e., the total time of start bit + data bits + parity + stop bits).
Restarting after a break is received requires the RXD pin to be logic 1 for at least $\frac{1}{2}$-bit time.

4) Three indicates that the UART is ready to accept a new character for transmission. In addition, this bit causes the UART to issue an interrupt to the CPU when the transmitter holding register empty interrupt enable is set high.

Interrupt Identification Register (IIR)

The UARTs of the SAB 82C250/82C251 have interrupt capabilities that allow interfacing to all popular microprocessors presently available.

In order to provide minimum software overhead during data character transfers, the UART prioritizes interrupts into four levels and records these in the interrupt identification register. The four levels of interrupt conditions in order of priority are: receiver line status, received data ready, transmitter holding register empty and modem status. When the CPU accesses the IIR, the UART freezes all interrupts and indicates the highest priority pending interrupt to the CPU. While this CPU access is occurring, the UART records new interrupts, but does not change its current indication until the access is complete. Figure 5 shows the contents of the IIR.

Figure 5
Interrupt Identification Register

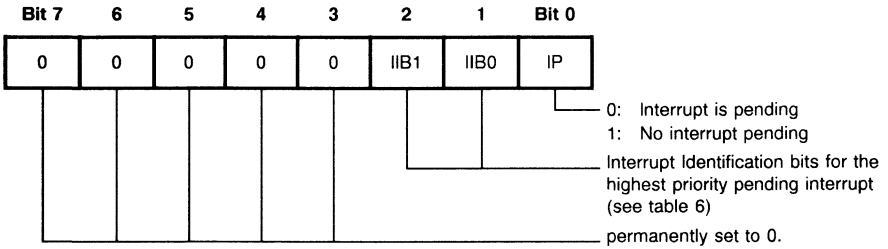


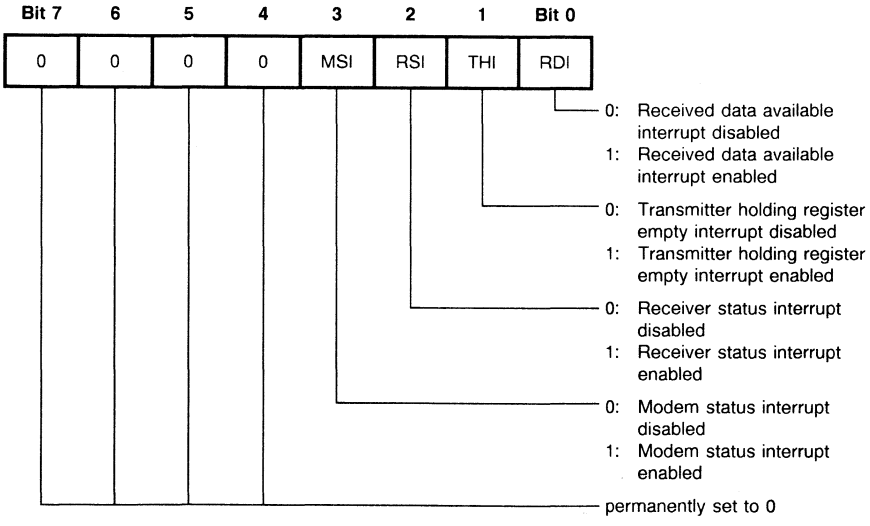
Table 6
Interrupt Control Functions

Interrupt identification register				Interrupt set and reset functions		
IIB1	IIB0	IP	Priority level	Interrupt type	Interrupt source	Interrupt reset control
0	0	1	–	None	None	–
1	1	0	Highest	Receiver line status	Overrun error or parity error or framing error or break interrupt	Reading the line status register
1	0	0	Second	Received data available	Receive data available	Reading the receiver buffer register
0	1	0	Third	Transmitter holding register empty	Transmitter holding register empty	Reading the IIR register (if source of interrupt) or Writing into the transmitter holding register
0	0	0	Fourth	Modem status	Clear to send or data set ready or ring indicator or data carrier detect	Reading the modem status register

Interrupt Enable Register (IER)

This register enables the four types of UART interrupts. Each interrupt can individually activate the interrupt (INT1/INT2) output signals. It is possible to totally disable the interrupt system by resetting bits 0 through 3 of the interrupt enable register (IER). Similarly, setting bits of this register to a logic 1 enables the selected interrupt(s). Disabling an interrupt prevents it from being indicated as active in the IIR and from activating the INT1/INT2 output signals. All other system functions operate in their normal manner, including the setting of the line status and modem status registers. Figure 6 shows the contents of the IER.

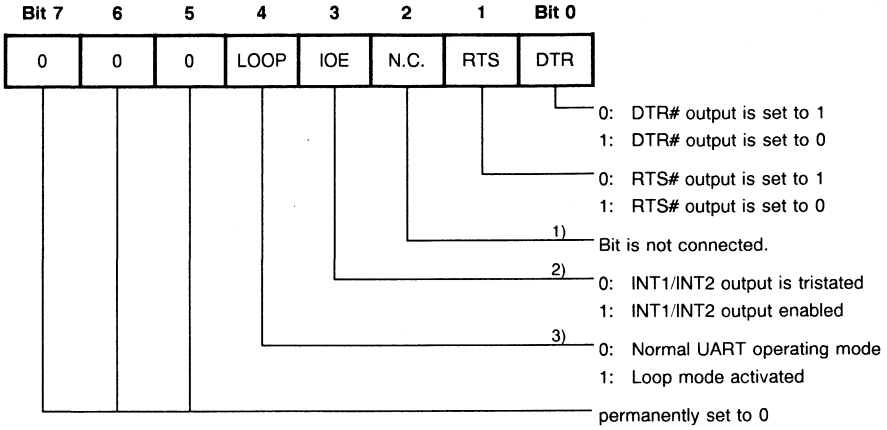
Figure 6
Interrupt Enable Register



Modem Control Register (MCR)

This register controls the interface to the Modem or data set (or a peripheral device emulating a Modem). The contents of the Modem control register are described in figure 7.

Figure 7
Modem Control Register



1) In loop mode this bit is equivalent to the MSR bit 6 (RI).

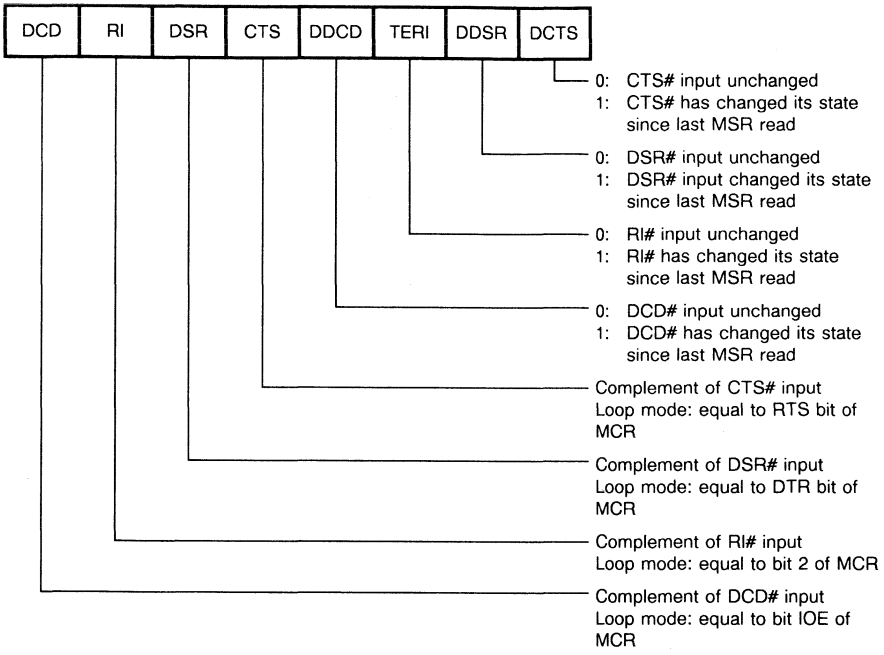
2) In loop mode this bit is equivalent to the MSR bit 7 (DCD).

3) When LOOP is set to logic 1, the following occurs: the transmitter serial output (TXD) is set the the marking (logic 1) state; the receiver serial input (RXD), is disconnected; the output of the transmitter shift register is "looped back" into the receiver shift register input; the four Modem control inputs (CTS#, DSR#, RI# and DCD#) are disconnected; and the four Modem control bits (0-3) are internally connected to the four Modem control inputs. The Modem control output pins are forced to their inactive state (high). In the diagnostic mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit and received data paths of the UART. In the loop mode, the receiver and transmitter interrupts are fully operational. The Modem control interrupts are also operational, but the interrupt's sources are now the lower four bits of the Modem control register instead of the four Modem control inputs. The interrupts are still controlled by the interrupt enable register.

Modem Status Register (MSR)

This register provides the current state of the control lines from the Modem (or peripheral device) to the CPU. In addition to this current-state information, four bits of the Modem status register provide change information. These bits are set to a logic 1 whenever a control input from the Modem changes state. They are reset logic 0 whenever the CPU reads the Modem status register. Figure 8 shows the contents of the MSR.

Figure 8
Modem Status Register



Note: Whenever bit 0, 1, 2 or 3 is set to logic 1, a modem status interrupt is generated.

Scratchpad Register

This 8-bit read/write register does not control the UART in any way. It is intended as a scratchpad register to be used by the programmer to hold data temporarily.

Reset Operation

A low level at the RESET# input pin causes the UART to reset to the condition listed in table 7.

Table 7
Register Reset Functions

Register/signal	Reset Control	Reset state
Interrupt Enable Register	Master Reset	All bits low (0-3 forced and 4-7 permanent)
Interrupt Identification Register	Master Reset	Bit 0 is high, bits 1 and 2 low, bits 3-7 are permanently low
Line Control Register	Master Reset	All bits low
Modem Control Register	Master Reset	All bits low
Line Status Register	Master Reset	All bits low, except bits 5 and 6 are high
Modem Status Register	Master Reset	Bits 0-3 low, bits 4-7 input signal
SOUT	Master Reset	High
BDO	Master Reset	High
Interrupt (RCVR ERRS)	Master Reset/Read LSR	Low
Interrupt (RCVR Data Ready)	Master Reset/Read RBR	Low
Interrupt (THRE)	Master Reset/Read IIR/ Write THR	Low
Interrupt (Modem status changes)	Master Reset/Read MSR	Low
RTS#	Master Reset	High
DTR#	Master Reset	High
Interrupt output line	Master Reset	tristated

Parallel Port Description

Register Selection

The internal data, status and control registers of the parallel port are selected by two address lines, the chip select line CS3# and the read/write control inputs.

Table 8
Parallel Port Register Addresses

CS3#	A1	A0	IOR#	IOW#	Operation
1	X	X	X	X	No Operation
0	0	0	1	0	Write Data Register
0	0	0	0	1	Read Data Register
0	0	1	0	1	Read Status Register
0	1	0	1	0	Write Control Register
0	1	0	0	1	Read Control Register

While CS3# is activated, CS1# and CS2# must be inactive. Other combinations of A0, A1, IOR#, and IOW# together with CS3# = 0 as shown in table 8 are illegal combinations.

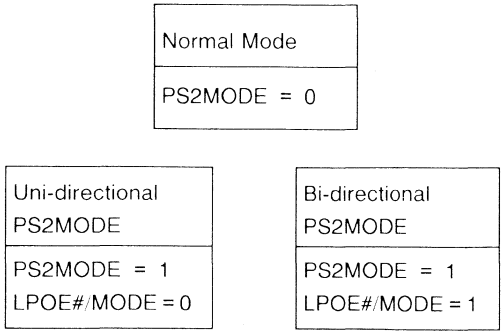
Table 9
Parallel Port Register Definition

	Register Address		
	0	1	2
Bit No.	Data Register	Status Register	Control Register
0	Data Bit 0	1	STB
1	Data Bit 1	1	AFD
2	Data Bit 2	PS2MODE = 0:1 PS2MODE = 1:IRQSTAT	INIT
3	Data Bit 3	ERROR	SLCTIN
4	Data Bit 4	SLCT	IRQEN
5	Data Bit 5	PE	PS2MODE = 0:1 PS2MODE = 1:DIR
6	Data Bit 6	ACK	1
7	Data Bit 7	BUSY	1

Parallel Port Mode Selection

The parallel port of the SAB 82C250/82C251 can work in two basic operating modes: the normal mode and the PS/2 mode. In PS/2 mode a uni-directional or bi-directional operation is possible. Two strapping pins are used to select the operating mode. With PS2MODE = 0 the normal mode is selected. In PS/2 mode, the LPOE#/MODE pin selects further, mode operations (figure 9).

Figure 9
Parallel Port Mode Selection



In the normal mode, the SAB 82C250/82C251 parallel port operates in a XT/AT and 16C452/16C451 compatible mode. The input line LPOE#/MODE acts as an output enable input for PD0-PD7. With LPOE#/MODE = 1, PD0-PD7 may be used as an 8-bit input port.

In the PS/2 mode, the parallel port is fully register compatible to the parallel port implementation of the PS/2 PC's.

Data Register

The data register is a read/write register for the 8-bit data of the parallel port. The data transfer to or from the PD0-PD7 pins is handled via this register. The source of data during a read operation of the data register depends on the selected mode of the parallel port. Either the data register itself or the data at the PD0-7 pins may be read back (figure 10).

Figure 10
Parallel Port Internal Data Path

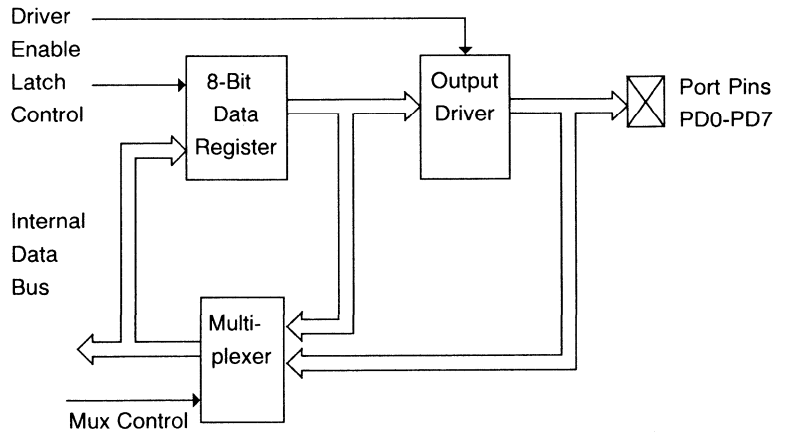


Table 10 shows all mode dependent functions of a read/write access of the parallel port data register.

The PS/2 mode is divided into an uni-directional and a bi-directional operation mode.
In the uni-directional PS/2 mode output drivers are always enabled. In this mode a read access to the data register directly returns the data register content.
In the bi-directional PS/2 mode, bit 5 of the control register (DIR) controls the direction of the data paths. If DIR is set, data at PD0-7 can be read (output driver disabled). With DIR = 0, the data register information is available at PD0-7 (output driver enabled).

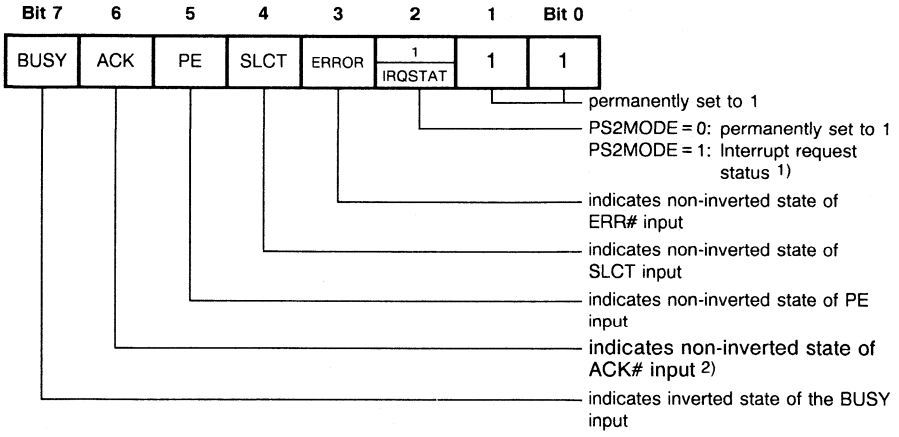
Table 10
Parallel Port Data Register Access Combinations

Mode		Pin PS2MODE	Pin LPOE#/ MODE	Bit DIR	Read Operation	Write Operation
Normal Mode		0	0	X	Read data register via the enabled output driver	Latch data in data register with output driver enabled
		0	1	X	Output driver disabled; Read data from PD0-7	Latch data in data register with output driver disabled
PS/2 Mode	Uni-directional	1	0	X	Read data of data register with output driver enabled	Latch data in data register with output driver enabled
	Bi-directional	1	1	0	Read data of data register with output driver enabled	Latch data in data register with output driver enabled
		1	1	1	Output driver disabled; Read data from PD0-7	Latch data in data register with output driver disabled

Status Register

The status register is a read-only register. A read operation of this register presents the real-time status of the parallel port interface input lines to the CPU.

Figure 11
Parallel Port Status Register



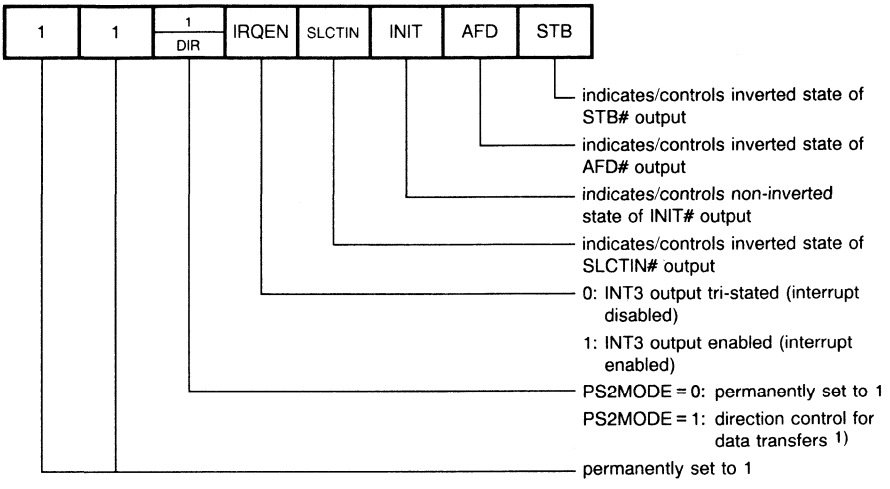
1) If PS2MODE = 1, this bit indicates an interrupt request status. IRQSTAT is cleared with every transition of the ACK# input signal, if the interrupt requests are enabled by the IRQEN bit of the control register (IRQEN = 1). If IRQSTAT = low, a read operation of the status register sets IRQSTAT back to high level. IRQSTAT is also set high by writing the control port with IRQEN = 0 (interrupt disable).

2) If parallel port interrupt is enabled (IRQEN = 1), the ACK# input is directly connected to the INT3 output.

Control Register

The control register of the parallel port may be read or written. A write operation to the control register latches the five (PS/2 mode : six) least significant bits of the write data.

Figure 12
Parallel Port Control Register



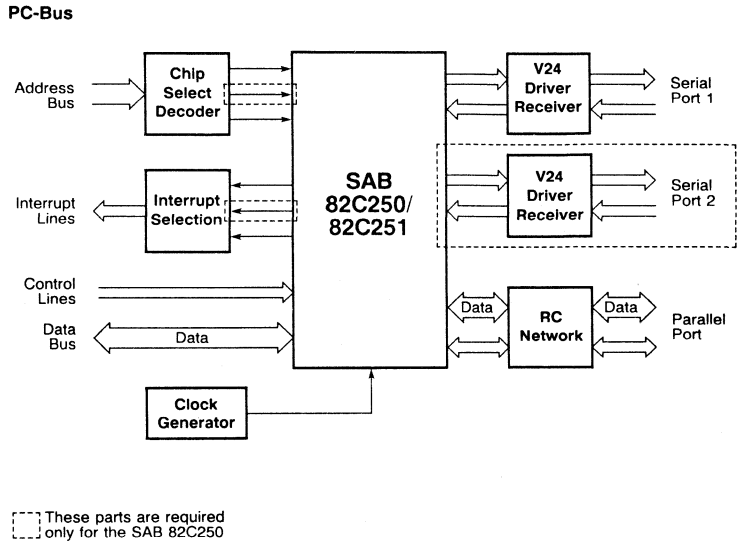
1) Details about DIR are described in table 10.

After a reset operation (RESET# = 0) the active bits of the control register are set in the following way:

STB, AFD, SLCTIN = 1

INIT, IRQEN, DIR = 0

Figure 13
SAB 82C250/82C251 Typical Application



Absolute Maximum Ratings

Ambient temperature under bias	0 to 70° C
Storage temperature	– 65 to + 150° C
Supply voltage	– 0.5 to + 70 V
Voltage on any pin with respect to ground	– 0.5 to V_{CC} + 0.5 V
Power dissipation	500 mW

Note: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum ratings for extended periods may affect device reliability.

DC Characteristics

T_A = 0 to 70 °C; V_{CC} = 5 V ± 5 %; GND = 0 V

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
Clock input low voltage	V_{IL}	-0.5	+ 0.8	V	–
Clock input high voltage	V_{IH}	2.0	V_{CC} + 0.5	V	–
Input low voltage	V_{IL}	-0.5	+ 0.8	V	–
Input high voltage	V_{IH}	2.0	V_{CC} + 0.5	V	–
Output low voltage	V_{OL}	–	0.4	V	I_{OL} = 24 mA on PD0-PD7 I_{OL} = 10 mA on INIT#, AFD#, STB#, SLCTIN# I_{OL} = 4.0 mA on other inputs
Output high voltage	V_{OH}	2.4	–	V	I_{OH} = -15 mA on PD0-PD7 I_{OH} = -1.5 mA on INIT#, AFD#, STB#, SLCTIN# I_{OH} = -2.5 mA on other inputs
Input leakage current	I_{IL}	–	± 10	µA	V_{CC} = 5.25 V; GND = 0 V; 0 V ≤ V_{IN} ≤ V_{CC} All other pins floating
Clock leakage current	I_{CL}	–	± 10	µA	V_{IN} = 0 V, 5.25 V

DC Characteristics (Continued)

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
Tristate leakage current	I_{OZ}	–	±20	µA	V_{CC} = 5.25 V; GND = 0 V; V_{OUT} = 0 V, 5.25 V 1. Chip selected 2. Chip and write mode selected
Power supply current	I_{CC}	–	20	mA	V_{CC} = 5.25 V; No loads on RXD0,1 , DSR0,1#, DCD0,1#, CTS0,1#; RI0,1# = 2.0 V; Other inputs = 0.8 V; Baud rate = 256 K; Baud rate gen. = 8 MHz

Capacitance

T_A = 25 °C; f = 1 MHz; V_{CC} = GND = 0 V

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
Clock input capacitance	C_{XIN}	–	10	pF	f_C = 1 MHz
Input capacitance	C_{IN}	–	10	pF	Unmeasured pins are returned to GND
Output capacitance	C_{OUT}	–	10	pF	Unmeasured pins are returned to GND

AC Characteristics

T_A = 0 to 70 °C; V_{CC} = 5 V ±5 %; GND = 0 V

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
Bus Interface Timing					
IOR# delay from address	t_{AR}	30	–	ns	–
IOW# delay from address	t_{AW}	30	–	ns	–
IOR# delay from chip select	t_{CSR}	30	–	ns	–
IOW# delay from chip select	t_{CSW}	30	–	ns	–
Data hold time	t_{DH}	30	–	ns	–
Data setup time	t_{DS}	30	–	ns	–
BDO delay from IOR#	t_{DD}	60	–	ns	1 TTL load
IOR# to floating data delay	t_{HZ}	0	100	ns	150 pF loading [1]
Parallel port data hold time	t_{PDS}	20	–	ns	–
Parallel port data setup time	t_{PDH}	20	–	ns	–
Address hold time from IOR#	t_{RA}	20	–	ns	–
Chip select hold time from IOR#	t_{RCS}	20	–	ns	–
IOR# strobe time	t_{RD}	125	–	ns	–
Delay from IOR# to data	t_{RWD}	–	125	ns	150 pF loading [1]
Address hold time from IOW#	t_{WA}	20	–	ns	–
Chip select hold time from IOW#	t_{WCS}	20	–	ns	–
Parallel port data valid after IOW#	t_{WOL}	–	90	ns	–
IOW# strobe width	t_{WR}	100	–	ns	–
Clock cycle time	t_{XC}	125	–	ns	–
Clock high time	t_{XH}	55	–	ns	–
Clock low time	t_{XL}	55	–	ns	–
Read cycle	R_C	280	–	ns	–
Write cycle	W_C	280	–	ns	–

[1] Charge and discharge time is determined by V_{OL}, V_{OH} and the external loading.

AC Characteristics (Continued)

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
Receiver/Transmitter Timing					
Delay from IOR# to reset interrupt (read RBR or read LSR)	t_{RINT}	–	175	ns	150 pF loading
Delay from stop to set interrupt	t_{SINT}	–	1	t_{XC}	150 pF loading
Delay from IOW# to reset interrupt (write THR)	t_{HR}	–	175	ns	150 pF loading
Delay from IOR# to reset interrupt (write THR)	t_{IR}	–	175	ns	150 pF loading
Delay from Initial interrupt reset to transmit start	t_{IRS}	24	40	1)	–
Delay from initial write to interrupt	t_{SI}	32	48	1)	–
Delay from stop to interrupt (THRE)	t_{SINT}	8	8	1)	–
Modem Control Timing					
Delay from IOW# to output (write MCR)	t_{MDO}	–	200	ns	150 pF loading
Delay from IOW# to interrupt active/ tri-state	t_{WI}	–	200	ns	150 pF loading
Delay to reset interrupt from MODEM input	t_{SIM}	–	200	ns	150 pF loading
Delay to reset interrupt from IOR# (read MSR)	t_{RIM}	–	200	ns	150 pF loading
Reset Timing					
Output float from reset	t_{RSF}	–	100	ns	150 pF loading 2)
Output low from reset	t_{RSL}	–	100	ns	–
Output high from reset	t_{RSH}	–	100	ns	–
Reset pulse width	t_{RW}	500	–	ns	–

1) The unit is the clock frequency (baud rate) which is defined by dividing the CLK input frequency by the specified divisor in the baud generator divisor latches.

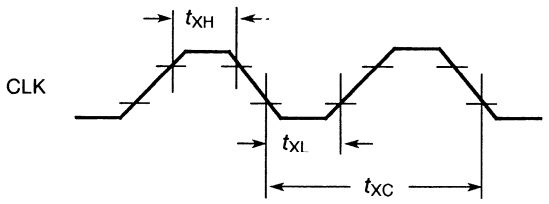
2) Charge and discharge time is determined by V_{OL}, V_{OH} and the external loading.

AC Characteristics (Continued)

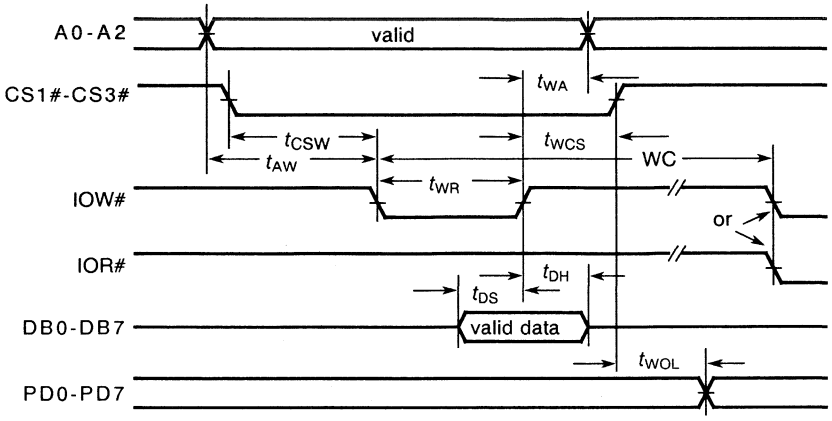
Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
Parallel Port Timing					
Data to STB# setup/hold time	t_{DT}	1	–	µs	–
STB# width	t_{SB}	1	500	µs	–
BUSY# start to ACK#	t_{BSA}	–	–	µs	Printer dependent
BUSY# delay from STB#	t_{BSD}	–	–	µs	Printer dependent
BUSY# width	t_{BSY}	–	–	µs	Printer dependent
ACK# width	t_{AK}	–	–	µs	Printer dependent
ACK# delay	t_{AKD}	–	–	µs	Printer dependent
INT3 delay to ACK# transition	t_{ID}	–	50	ns	–
INT3 disable time	t_{IDI}	–	50	ns	150 pF loading [1]
INT3 enable time	t_{IEN}	–	50	ns	150 pF loading [1]

[1] Charge and discharge time is determined by V_{OL}, V_{OH} and the external loading.

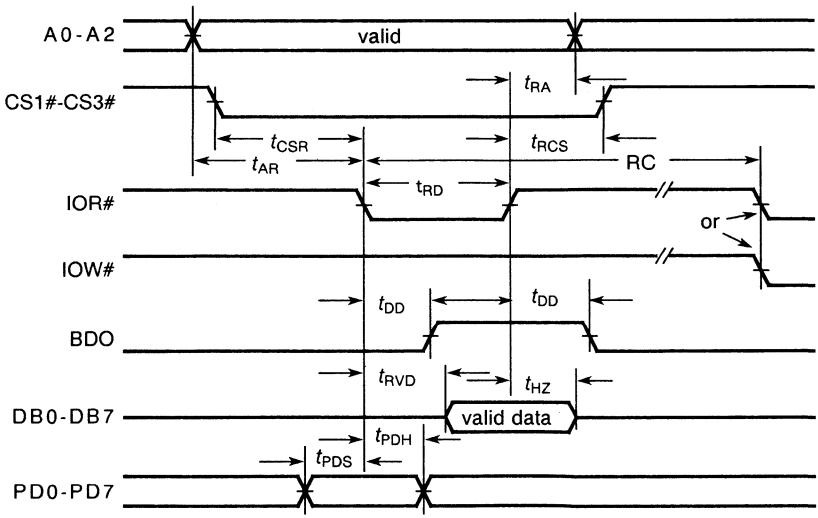
CLK Input



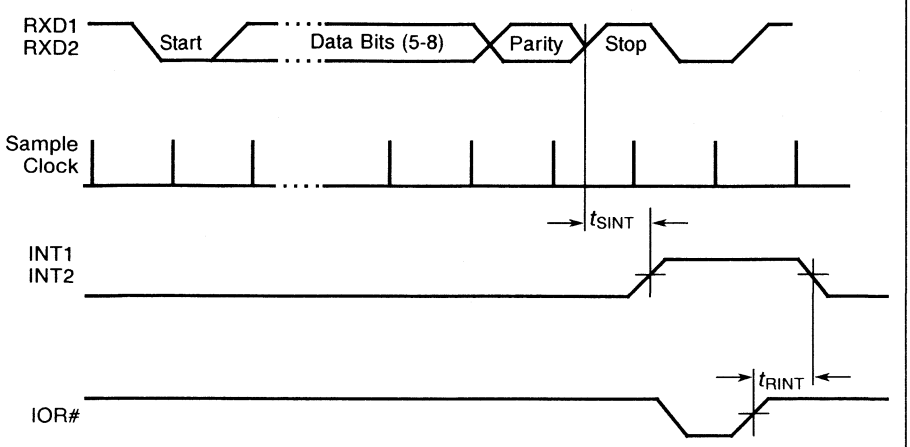
Write Cycle Timing



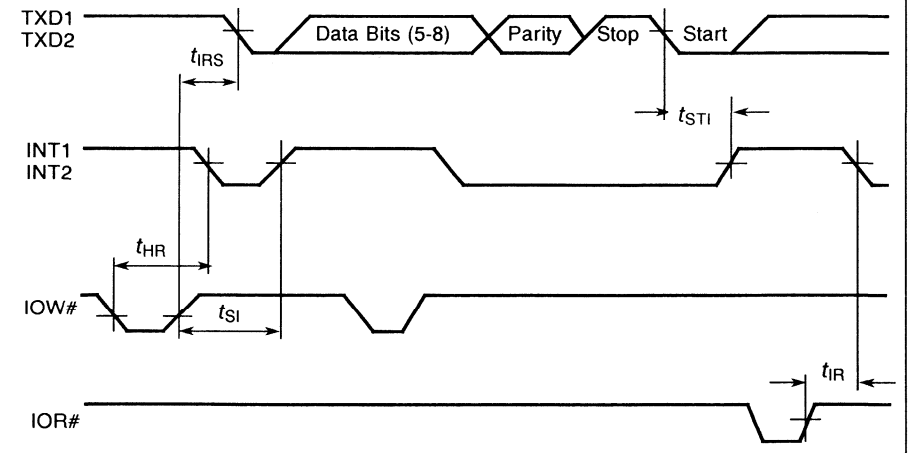
Read Cycle Timing



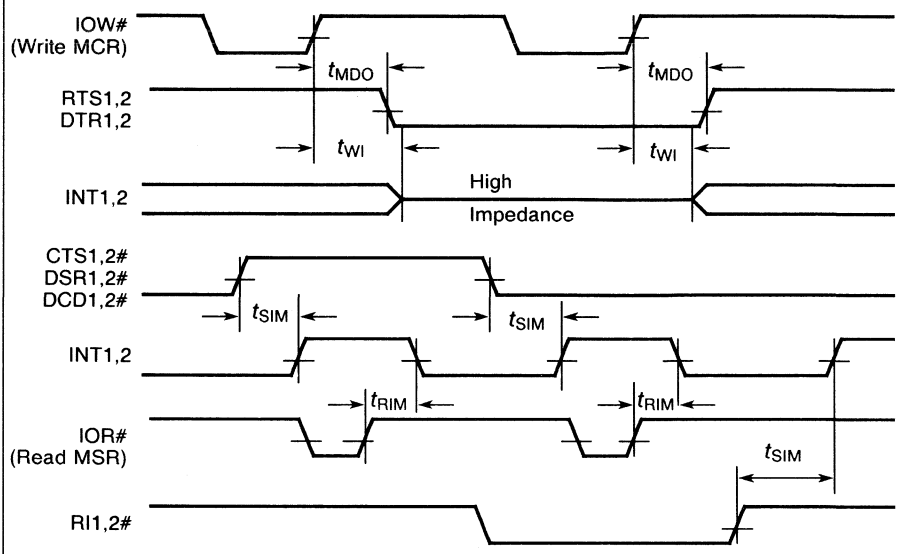
Receiver Timing



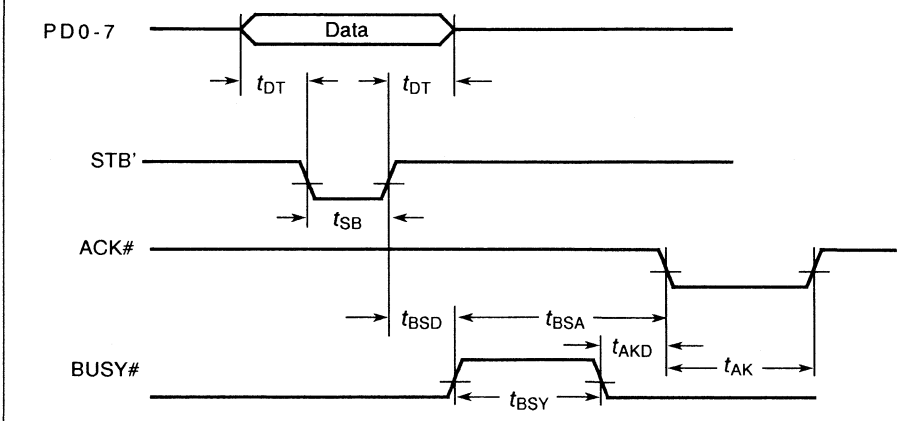
Transmitter Timing



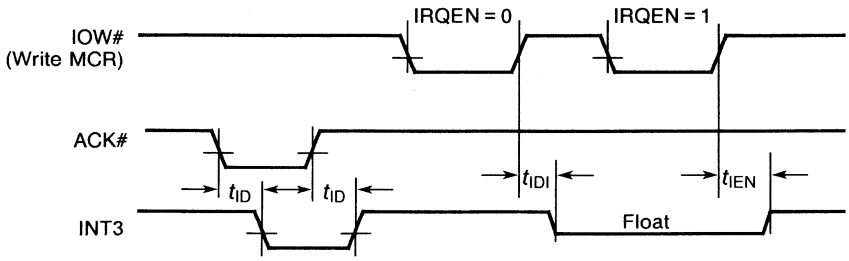
Modem Timing



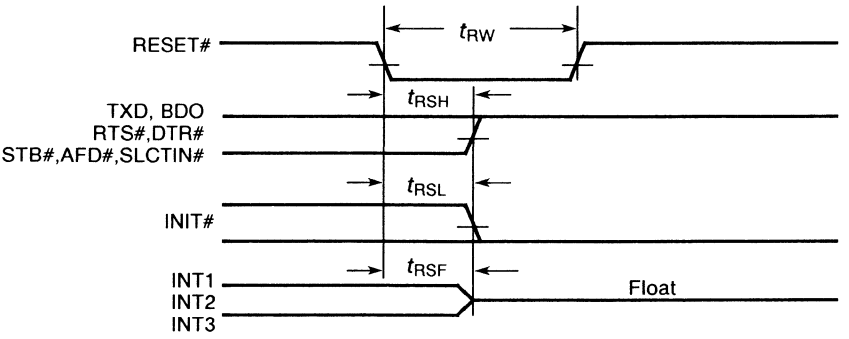
Typical Centronics Parallel Port Timing



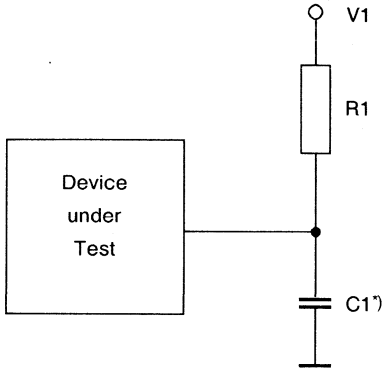
Parallel Port Interrupt Timing



Reset Timing



AC Test Circuits



Related Device Pins	V1	R1	C1
PD0 - PD7	1.63 V	51 Ω	150 pF
INIT#, AFD#, STB#, SLCTIN#	2.54 V	174 Ω	150 pF
All other Pins Pins	1.63 V	308 Ω	150 pF

*) Includes stray and jig capacitance

Ordering Information

Type	Ordering code	Package	Function
SAB 82C250-N	Q67120-P302	PL-CC-68 (SMD)	Advanced peripheral interface controller (2 serial port, 1 parallel port)
SAB 82C251-N	Q67120-P304	PL-CC-68 (SMD)	Advanced peripheral interface controller (1 serial port, 1 parallel port)

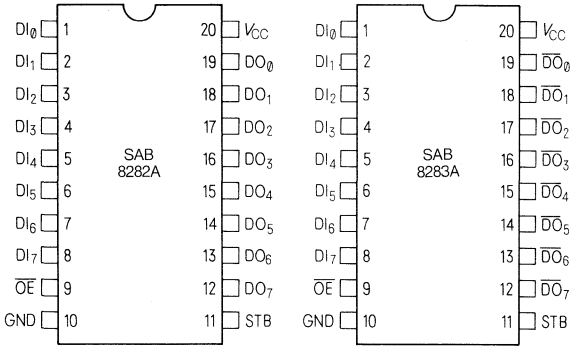
SIEMENS

Preliminary

SAB 8282A/SAB 8283A Octal Latch

- Fully compatible with SAB 8282/SAB 8283
- 40% Less Power Supply Current than Standard SAB 8282/SAB 8283
- Address Latch for SAB 80286, SAB 80186, SAB 8086, SAB 8085, SAB 8048 and SAB 8051 Families
- High Output Drive Capability for Driving System Data Bus
- Fully Parallel 8-Bit Data Register and Buffer
- No Output Low Noise when Entering or Leaving High Impedance State
- 3-State Outputs
- Transparent during Active Strobe
- 20-Pin Package

Pin Configuration



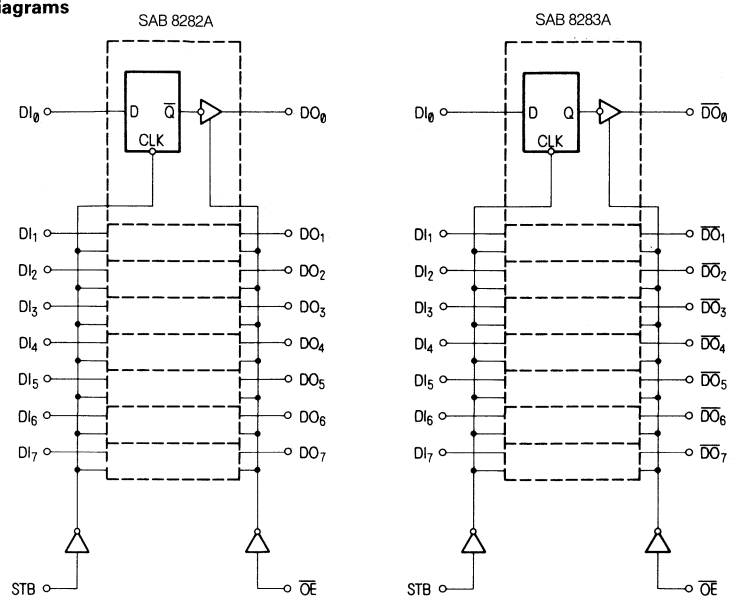
Pin Names

DI_0–DI_7	Data In
DO_0–DO_7	Data Out
$\overline{OE}$	Output Enable
STB	Strobe
V_{CC}	Power Supply (+5V)
GND	Ground (0V)

The SAB 8282A and SAB 8283A are 8-bit bipolar latches with 3-state output buffers. They can be used to implement latches, buffers, or multiplexers. The SAB 8283A inverts the input data at its outputs while the SAB 8282A does not.

Thus, all of the principal peripheral and input/output functions of a microcomputer system can be implemented with these devices. This device is fabricated in a fast bipolar ASBC (Advanced Standard Buried Collector) process of Siemens.

Logic Diagrams



Pin Definitions and Functions

Symbol	Number	Input (I) Output (O)	Function
STB	11	I	STROBE – STB is an input control pulse used to strobe data at the data input pins (A_0–A_7) into the data latches. This signal is active HIGH to admit input data. The data is latched at the HIGH to LOW transition of STB.
$\overline{OE}$	9	I	OUTPUT ENABLE – $\overline{OE}$ is an input control signal which when active LOW enables the contents of the data latches onto the data output pin (DO_0–DO_7 or $\overline{DO_0}$–$\overline{DO_7}$). $\overline{OE}$ being inactive HIGH forces the output buffers to their high impedance state.
DI_0–DI_7	1–8	I	DATA INPUT PINS – Data presented at these pins satisfying setup time requirements when STB is strobed and latched into the data input latches.
DO_0–DO_7 (SAB 8282A) $\overline{DO_0}$–$\overline{DO_7}$ (SAB 8283A)	12–19	O	DATA OUTPUT PINS – When $\overline{OE}$ is true, the data in the data latches is presented as inverted (SAB 8283A) or non-inverted (SAB 8282A) data onto the data output pins.
V_{CC}	20	–	Power Supply (+5V)
GND	10	–	Ground (0V)

Functional Description

The SAB 8282A and SAB 8283A octal latches are 8-bit latches with 3-state output buffers.
Data having satisfied the setup time requirements is latched into the data latches by strobing the STB line HIGH to LOW. Holding the STB line in its active HIGH state makes the latches appear transparent.

Data is presented to the data output pins by activating the $\overline{OE}$ input line. When $\overline{OE}$ is inactive HIGH the output buffers are in their high impedance state. Enabling or disabling the output buffers will not cause negative-going transients to appear on the data output bus.

Absolute Maximum Ratings[1)]

Temperature Under Bias	0 to +70°C
Storage Temperature	−65 to +150°C
All Output and Supply Voltages	−0.5 to +7V
All Input Voltages	−1.0 to +5.5V
Power Dissipation	1W

D. C. Characteristics

T_A = 0 to 70°C; V_{CC} = +5V ±10%

Symbol	Parameter	Limit Values		Units	Test Conditions
		Min.	Max.		
V_C	Input Clamp Voltage	–	−1	V	I_C = −5 mA
I_{CC}	Power Supply Current SAB 8282A SAB 8283A	–	100 90	mA	all outputs open
I_F	Forward Input Current	–	−0.2		V_F = 0.45V
I_R	Reverse Input Current		50	µA	V_R = 5.25V
V_{OL}	Output LOW Voltage		0.45	V	I_{OL} = 32 mA
V_{OH}	Output HIGH Voltage	2.4	–		I_{OH} = −5 mA
I_{OFF}	Output Off Current	–	±50	µA	V_{OFF} = 0.45 to 5.25V
V_{IL}	Input LOW Voltage		0.8	V	V_{CC} = 5.0V [2)]
V_{IH}	Input HIGH Voltage	2.0	–		
C_{IN}	Input Capacitance	–	12	pF	F = 1 MHz V_{BIAS} = 2.5V, V_{CC} = 5V T_A = 25°C

1) Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

2) Output Loading: I_{OL} = 32 mA; I_{OH} = −5 mA; C_L = 300 pF

A.C. Characteristics

T_A = 0 to +70°C; V_{CC} = +5 V ± 10%

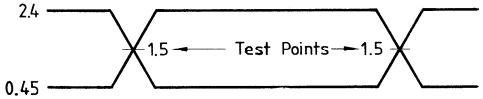
Loading

Outputs: I_{OL} = 32 mA; I_{OH} = −5 mA; C_L = 300 pF

Symbol	Parameter	Limit Values		Units	Test Conditions
		Min.	Max.		
t_{IVOV}	Input to Output Delay –Inverting –Non-Inverting	 5 5	 22 30	ns	2)
t_{SHOV}	STB to Output Delay –Inverting –Non-Inverting	 10 10	 40 45		
t_{EHQZ}	Output Disable Time	5	18		
t_{ELOV}	Output Enable Time	10	30		
t_{IVSL}	Input to STB Setup Time	0	–		
t_{SLIX}	Input to STB Hold Time	25			
t_{SHSL}	STB HIGH Time	15			
t_{ILIH}, t_{OLOH}	Input, Output Rise Time	–	20		From 0.8 to 2.0 V
t_{IHIL}, t_{OHOL}	Input, Output Fall Time		12		From 2.0 to 0.8 V

A.C. Testing Input, Output Waveform

Input/Output

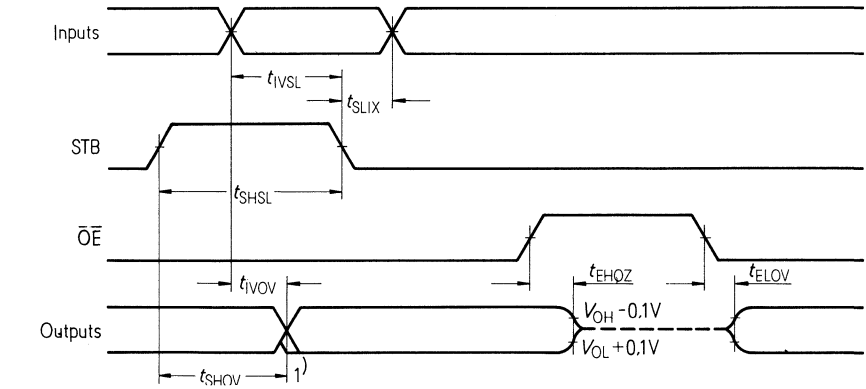


A.C. Testing: Inputs are driven at 2.4 V for a logic "1" and 0.45 V for a logic "0". Timing measurements are made at 1.5 V for both a logic "1" and "0".

Waveforms

All timing measurements are made at 1.5 V unless otherwise noted.

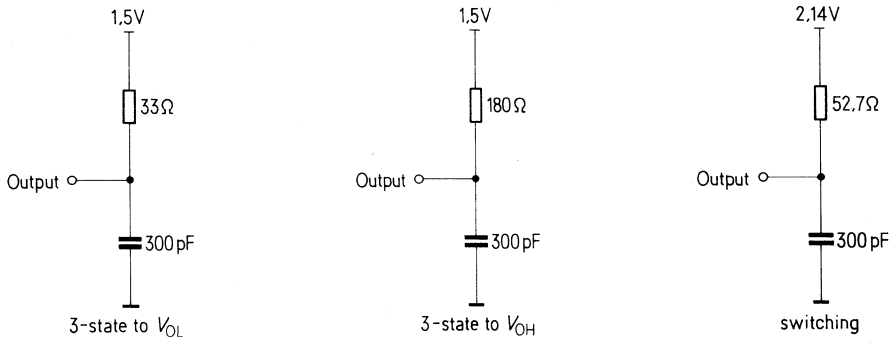
Timing



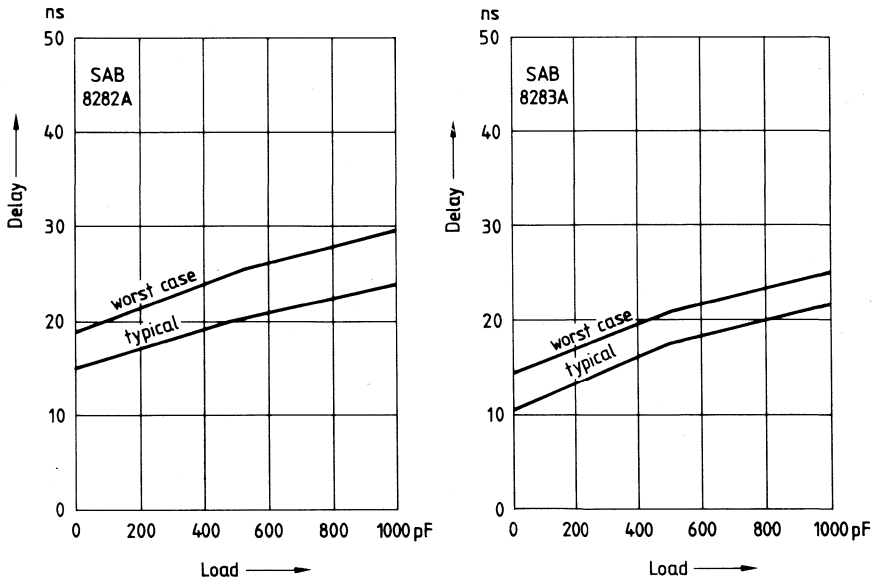
1) SAB 8283A Only – Output may be momentarily invalid following the high going STB transition.

2) See waveforms and test load circuit.

Output Test Load Circuits



Output Delay vs. Capacitance



Ordering Information

Type	Description	Ordering code
SAB 8282A–P	Octal Latch, non inverting (plastic)	Q 67020–Y149
SAB 8283A–P	Octal Latch, inverting (plastic)	Q 67020–Y150

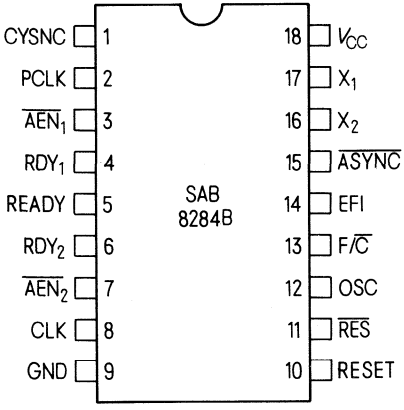
SIEMENS

Preliminary

SAB 8284B, SAB 8284B-1 Clock Generator and Driver for SAB 8086 Family Processors

- Fully compatible with SAB 8284A, SAB 8284A-1
- 30% Less Power Supply Current than Standard SAB 8284A, SAB 8284A-1
- Generates the System clock for SAB 8086 and SAB 8088 Processors: upto 8 MHz with SAB 8284B upto 10 MHz with SAB 8284B-1
- Uses a Crystal or a TTL Signal for Frequency Source upto 30 MHz
- Provides Synchronization for Synchronous and Asynchronous READY Signals
- 18-Pin Package
- Single +5V Power Supply
- Generates System Reset Output from Schmitt Trigger Input
- Capable of Clock Synchronization with Other SAB 8284Bs

Figure 1
Pin Configuration



Pin Names

X_1 X_2	Connections for crystal
F/$\overline{C}$	Clock source select
EFI	External clock input
CSYNC	Clock synchronization input
$\overline{ASYNC}$	Ready synchronization select
RDY_1 RDY_2	Ready signal
$\overline{AEN_1}$ $\overline{AEN_2}$	Address enabled qualifiers for $RDY_{1,2}$
$\overline{RES}$	Reset input
RESET	Synchronized reset output
OSC	Oscillator output
CLK	MOS Clock for the processor
PCLK	TTL Clock for peripherals
READY	Synchronized ready output
V_{CC}	Power Supply (+5V)
GND	Ground (OV)

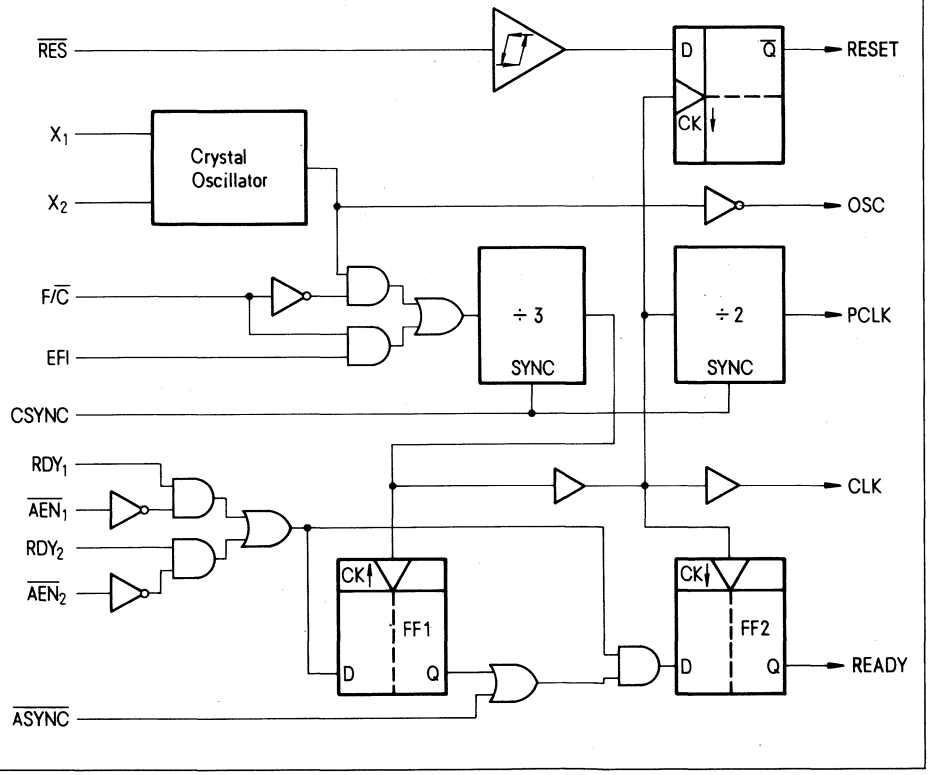
SAB 8284B is a bipolar clock generator/driver designed to provide clock signals for SAB 8086 and SAB 8088 processors and peripherals. It also contains READY logic for operation with two bus systems and provides the processors required READY synchronization and timing. Reset logic with hysteresis and synchronization is also provided. This device is fabricated in a fast bipolar ASBC (Advanced Standard Buried Collector) process of Siemens.

Pin Definitions and Functions

Symbol	Number	Input (I) Output (O)	Function
$\overline{AEN_1}$ $\overline{AEN_2}$	3, 7	I	ADDRESS ENABLE. $\overline{AEN}$ is an active LOW signal. $\overline{AEN}$ serves to qualify its respective Bus Ready Signal (RDY_1 or RDY_2). $\overline{AEN_1}$ validates RDY_1 while $\overline{AEN_2}$ validates RDY_2. Two AEN signal inputs are useful in system configurations which permit the processor to access two Multi-Master System Busses. In non Multi-Master configurations the $\overline{AEN}$ signal inputs are tied true (LOW).
RDY_1, RDY_2	4, 6	I	BUS READY (Transfer Complete). RDY is an active HIGH signal which is an indication from a device located on the system data bus that data has been received, or is available. RDY1 is qualified by $\overline{AEN_1}$ while RDY2 is qualified by $\overline{AEN_2}$.
$\overline{ASYNC}$	15	I	READY SYNCHRONIZATION SELECT. $\overline{ASYNC}$ is an input which defines the synchronization mode of the READY logic. When $\overline{ASYNC}$ is low, two stages of READY synchronization are provided. When $\overline{ASYNC}$ is left open or HIGH a single stage of READY synchronization is provided.
READY	5	O	READY. READY is an active HIGH signal which is the synchronized RDY signal input. READY is cleared after the guaranteed hold time to the processor has been met.
X_1, X_2	16, 17	I	CRYSTAL IN. X_1 and X_2 are the pins to which a crystal is attached. The crystal frequency is 3 times the desired processor clock frequency.
F/$\overline{C}$	13	I	FREQUENCY/CRYSTAL SELECT. F/$\overline{C}$ is a strapping option. When strapped LOW, F/$\overline{C}$ permits the processors clock to be generated by the crystal. When F/$\overline{C}$ is strapped HIGH, CLK is generated from the EFI input.
EFI	14	I	EXTERNAL FREQUENCY IN. When F/$\overline{C}$ is strapped HIGH, CLK is generated from the input frequency appearing on this pin. The input signal is a square wave 3 times the frequency of the desired CLK output.
CLK	8	O	PROCESSOR CLOCK. CLK is the clock output used by the processor and all devices which directly connect to the processor's local bus (i.e., the bipolar support chips and other MOS devices). CLK has an output frequency which is 1/3 of the crystal or EFI input frequency and a 1/3 duty cycle. An output HIGH of 4.5 volts (V_{CC} = 5V) is provided on this pin to drive MOS devices.
PCLK	2	O	PERIPHERAL CLOCK. PCLK is a TTL level peripheral clock signal whose output frequency is 1/2 that of CLK and has 50% duty cycle.
OSC	12	O	OSCILLATOR OUTPUT. OSC is the TTL level output of the internal oscillator circuitry. Its frequency is equal to that of the crystal.
$\overline{RES}$	11	I	RESET IN. $\overline{RES}$ is an active LOW signal which is used to generate RESET. The SAB 8284B provides a Schmitt trigger input so that an RC connection can be used to establish the power-up reset of proper duration.

Symbol	Number	Input (I) Output (O)	Function
RESET	10	O	RESET. RESET is an active HIGH signal which is used to reset the SAB 8086 family processors. Its timing characteristics are determined by $\overline{RES}$.
CSYNC	1	I	CLOCK SYNCHRONIZATION. CSYNC is an active HIGH signal which allows multiple SAB 8284B to be synchronized to provide clocks that are in phase. When CSYNC is HIGH the internal counters are reset. When CSYNC goes LOW the internal counters are allowed to resume counting. CSYNC needs to be externally synchronized to EFI. When using the internal oscillator CSYNC should be hard-wired to ground.
V_{CC}	18	–	Power Supply (+5V)
GND	9	–	Ground (OV)

Figure 2
Block Diagram



Functional Description

General

The SAB 8284B is a single chip clock generator/driver for SAB 8086 and SAB 8088 processors. The chip contains a crystal-controlled oscillator, a divide-by-three counter, "Ready" synchronization and reset logic. Refer to Figure 2 for "Block Diagram" and Figure 1 for "Pin Configuration".

Oscillator

The oscillator circuit of the SAB 8284B is designed primarily for use with an external series resonant fundamental mode crystal from which the basic operating frequency is derived.

The crystal frequency should be selected at three times the required CPU clock. X1 and X2 are the two crystal input crystal connections. For the most stable operation of the oscillator (OSC) output circuit, two series resistors ($R_1 = R_2 = 510\ \Omega$) as shown in figure 7 are recommended. The output of the oscillator is buffered and brought out on OSC so that other system timing signals can be derived from this stable, crystal-controlled source.

It is advisable to limit stray capacitances to less than 10pF on X1 and X2 to minimize deviation from operating at the fundamental frequency.

Clock Generator

The clock generator consists of a synchronous divide-by-three counter with a special clear input that inhibits the counting. This clear input (CSYNC) allows the output clock to be synchronized with an external event (such as another SAB 8284B clock). It is necessary to synchronize the CSYNC input to the EFI clock external to the SAB 8284B. This is accomplished with two Schottky flip-flops (see figure 3). The counter output is a 33% duty cycle clock at one-third the input frequency.

The F/$\overline{C}$ input is a strapping pin that selects either the crystal oscillator or the EFI input as the clock for the ÷3 counter. If the EFI input is selected as the clock source, the oscillator section can be used independently for another clock source. Output is taken from OSC.

Clock Outputs

The CLK output is a 33% duty cycle MOS clock driver designed to drive the SAB 8086 and SAB 8088 processors directly. PCLK is a TTL level peripheral clock signal whose output frequency is 1/2 that of CLK. PCLK has 50% duty cycle.

Reset Logic

The reset logic provides a Schmitt trigger input ($\overline{RES}$) and a synchronizing flip-flop to generate the reset timing. The reset signal is synchronized to the falling edge of CLK. A simple RC network can be used to provide power-on reset by utilizing this function of the SAB 8284B. Waveforms for clocks and reset signals are illustrated in Figure 4.

READY Synchronization

Two READY inputs (RDY_1, RDY_2) are provided to accomodate two Multi-Master system busses. Each input has a qualifier ($\overline{AEN_1}$ and $\overline{AEN_2}$, respectively).

The $\overline{\text{AEN}}$ signals validate their respective RDY signals. If a Multi-Master system is not being used the $\overline{\text{AEN}}$ pin should be tied LOW.

Synchronization is required for all asynchronous active going edges of either RDY input to guarantee that the RDY setup and hold times are met. Inactive-going edges of RDY in normally ready systems do not require synchronization but must satisfy RDY setup and hold as a matter of proper system design.

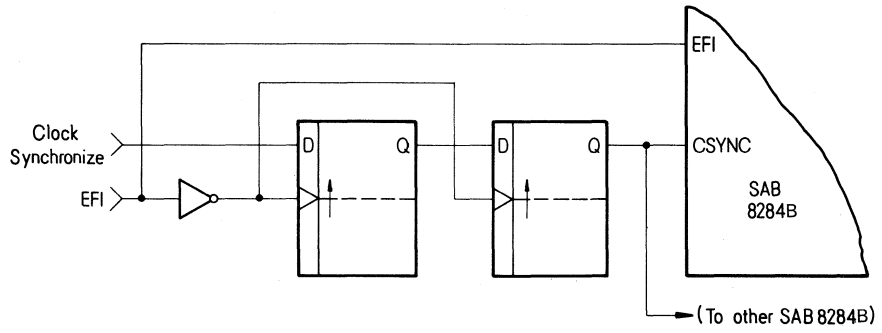
The $\overline{\text{ASYNC}}$ input defines two modes of READY synchronization operation.

When $\overline{\text{ASYNC}}$ is LOW, two stages of synchronization are provided for active READY input signals. Positive-going asynchronous READY inputs will first be synchronized to flip-flop one at the rising edge of CLK (requiring a setup time t_{R1VCH}) and then synchronized to flip-flop two at the next falling edge of CLK, after which time the READY output will go active (HIGH). Negative-going asynchronous READY inputs will be synchronized directly to flip-flop two at the falling edge of CLK, after which time the READY output will go inactive. This mode of operation is intended for use by asynchronous (normally not ready) devices in the system which cannot be guaranteed by design to meet the required RDY setup timing, t_{R1VCL}, on each bus cycle (Refer to Figure 5).

When $\overline{\text{ASYNC}}$ is high or left open, the first READY flip-flop is bypassed in the READY synchronization logic. READY inputs are synchronized by flip-flop two on the falling edge of CLK before they are presented to the processor. This mode is available for synchronous devices **that can be guaranteed to meet the required RDY time** (Refer to Figure 6).

$\overline{\text{ASYNC}}$ can be changed on every bus cycle to select the appropriate mode of synchronization for each device in the system.

Figure 3
CSYNC Synchronization



Absolute maximum ratings [1]

Temperature Under Bias	0 to 70°C
Storage Temperature	− 65 to 150°C
All Output and Supply Voltages	−0.5 to 7 V
All Input Voltages	−1.0 to 5.5 V
Power Dissipation	1W

D.C. Characteristics

T_A = 0 to 70°C; V_{CC} = +5 V ±10%

Symbol	Parameter	Limit Values		Unit	Test Condition
		Min.	Max.		
I_F	Forward Input Current ($\overline{ASYNC}$) Other Inputs	–	−1.3 −0.5	mA	V_F = 0.45 V V_F = 0.45 V
I_R	Reverse Input Current ($\overline{ASYNC}$) Other Inputs		50 50	µA	$V_R = V_{CC}$ V_R = 5.25 V
V_C	Input Forward Clamp Voltage		−1.0	V	I_C = −5 mA
I_{CC}	Power Supply Current		110	mA	All outputs open
V_{IL}	Input LOW Voltage		0.8	V	–
V_{IH}	Input HIGH Voltage	2.0	–		
V_{IHR}	Reset Input HIGH Voltage	2.6			
V_{OL}	Output LOW Voltage	–	0.45		5 mA
V_{OH}	Output HIGH Voltage CLK Other Outputs	4 2.4	–		−1 mA −1 mA
$V_{IHR}-V_{ILR}$	$\overline{RES}$ Input Hysteresis	0.25			–

[1]) Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

A.C. Characteristics

T_A = 0 to +70°C; V_{CC} = +5 V ±10%

Timing Requirements

Symbol	Parameter	Limit Values		Unit	Test Condition
		Min.	Max.		
t_{EHEL}	External Frequency HIGH Time	13	–	ns	90% – 90% V_{IN}
t_{ELEH}	External Frequency LOW Time				10% – 10% V_{IN}
t_{ELEL}	EFI Period	$t_{EHEL}+t_{ELEH}+\delta$			[3]
	XTAL Frequency	12	25 [7]	MHz	–
t_{R1VCL}	RDY_1, RDY_2 Active Setup to CLK	35	–	ns	$\overline{ASYNC}$ = HIGH
t_{R1VCH}	RDY_1, RDY_2 Active Setup to CLK				$\overline{ASYNC}$ = LOW
t_{R1VCL}	RDY_1, RDY_2 Inactive Setup to CLK				
t_{CLR1X}	RDY_1, RDY_2 Hold to CLK	0			
t_{AYVCL}	$\overline{ASYNC}$ Setup to CLK	50			
t_{CLAYX}	$\overline{ASYNC}$ Hold to CLK	0			
t_{A1VR1V}	$\overline{AEN_1}$, $\overline{AEN_2}$ Setup to RDY_1, RDY_2	15			–
t_{CLA1X}	$\overline{AEN_1}$, $\overline{AEN_2}$ Hold to CLK	0			
t_{YHEH}	CSYNC Setup to EFI	20			
t_{EHYL}	CSYNC Hold to EFI	10			
t_{YHYL}	CSYNC Width	$2 \cdot t_{ELEL}$			
t_{I1HCL}	$\overline{RES}$ Setup to CLK	65			[4]
t_{CLI1H}	$\overline{RES}$ Hold to CLK	20			
t_{ILIH}	Input Rise Time	–	20		From 0.8 V to 2.0 V
t_{IHIL}	Input Fall Time		12		From 2.0 V to 0.8 V

Notes see next page.

Timing Responses

Symbol	Parameter	Limit Values		Unit	Test Condition
		Min.	Max.		
t_{CLCL}	CLK Cycle Period	100			–
t_{CHCL}	CLK HIGH Time	[1]	–		Fig. 7 & Fig. 8
t_{CLCH}	CLK LOW Time	[2]			Fig. 7 & Fig. 8
t_{CH1CH2} t_{CL2CL1}	CLK Rise or Fall Time	–	10		1.0 V to 3.5 V
t_{PHPL}	PCLK HIGH Time	t_{CLCL}–20			–
t_{PLPH}	PCLK LOW Time	t_{CLCL}–20	–		
t_{RYLCL}	Ready Inactive to CLK [6]	−8		ns	Fig. 9 & Fig. 10
t_{RYHCH}	Ready Active to CLK [5]	[2]			Fig. 9 & Fig. 10
t_{CLIL}	CLK to Reset Delay		40		
t_{CLPH}	CLK to PCLK HIGH Delay	–			
t_{CLPL}	CLK to PCLK LOW Delay		22		–
t_{OLCH}	OSC to CLK HIGH Delay	−5			
t_{OLCL}	OSC to CLK LOW Delay	2	35		
t_{OLOH}	Output Rise Time (except CLK)	–	20		From 0.8 V to 2.0 V
t_{OHOL}	Output Fall Time (except CLK)		12		From 2.0 V to 0.8 V

[1]) ($^1/_3\ t_{CLCL}$) +2 for CLK Freq. ≤ 8 MHz
($^1/_3\ t_{CLCL}$) +6 for CLK Freq. = 10 MHz

[2]) ($^2/_3\ t_{CLCL}$) −15 for CLK Freq. ≤ 8 MHz
($^2/_3\ t_{CLCL}$) −14 for CLK Freq. = 10 MHZ

[3]) δ = EFI rise (5 ns max) + EFI fall (5 ns max).

[4]) Setup and hold necessary only to guarantee recognition at next clock.

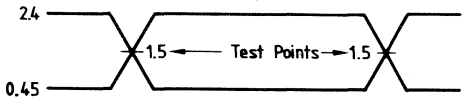
[5]) Applies only to T_3 and T_W states.

[6]) Applies only to T_2 states.

[7]) 30 MHz for SAB 8284B–1

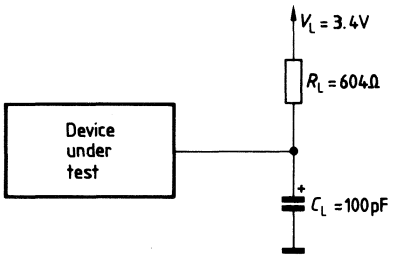
A.C. Testing

Input/Output Waveform

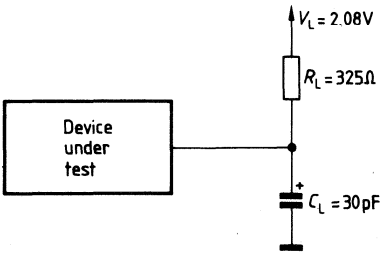


A.C. Testing: Input are driven at 2.4 V for a Logic "1" and 0.45 V for a Logic "0".
Timing Measurements are made at 1.5 V for Both a Logic "1" and "0".

Load Circuit for CLK output

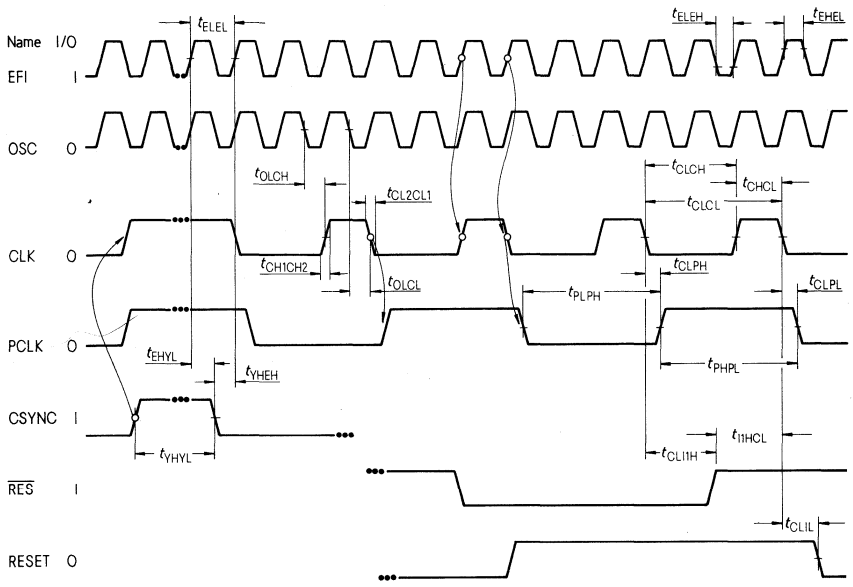


Load Circuit for all other output



Waveforms

Figure 4
Clocks and Reset Signals



Note: All timing measurements are made at 1.5 V, unless otherwise noted.

Figure 5
Ready Signals – Asynchronous Devices

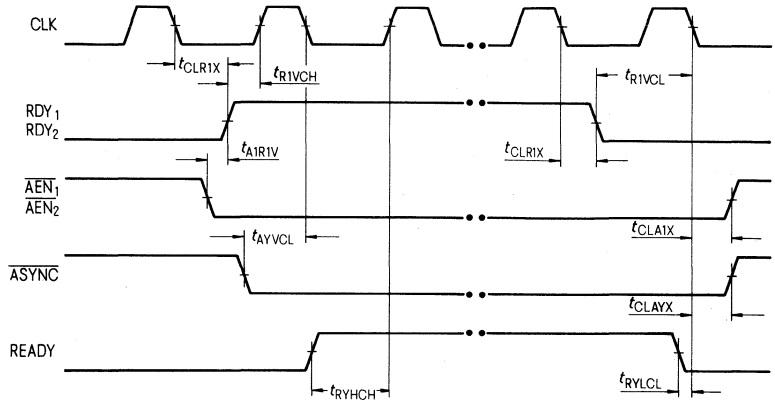
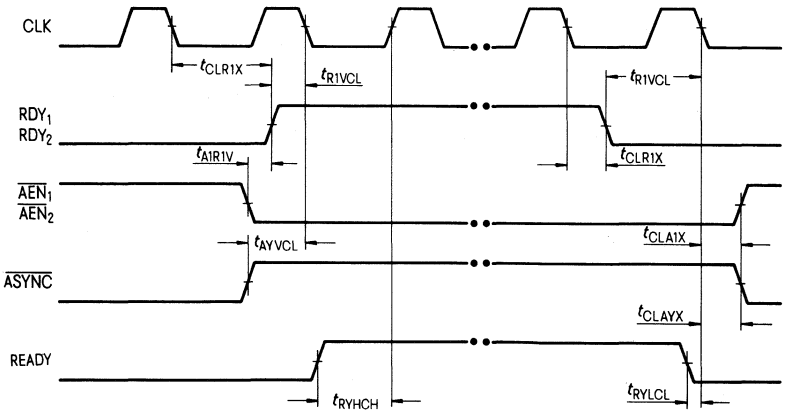


Figure 6
Ready Signals – Synchronous Devices



Testconditions

Figure 7
Clock High- and Low Time; Using X1, X2

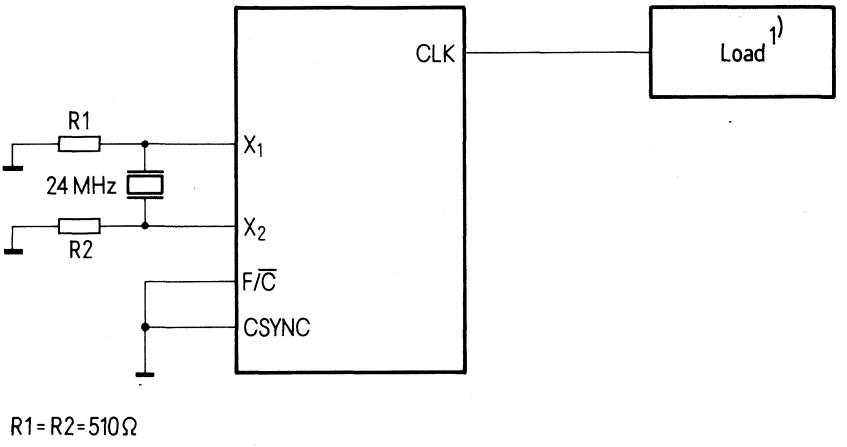
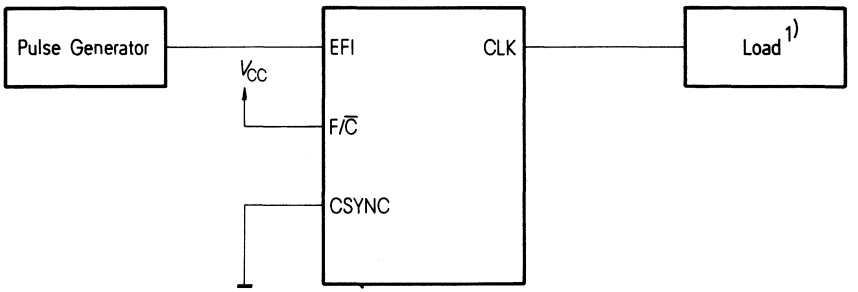
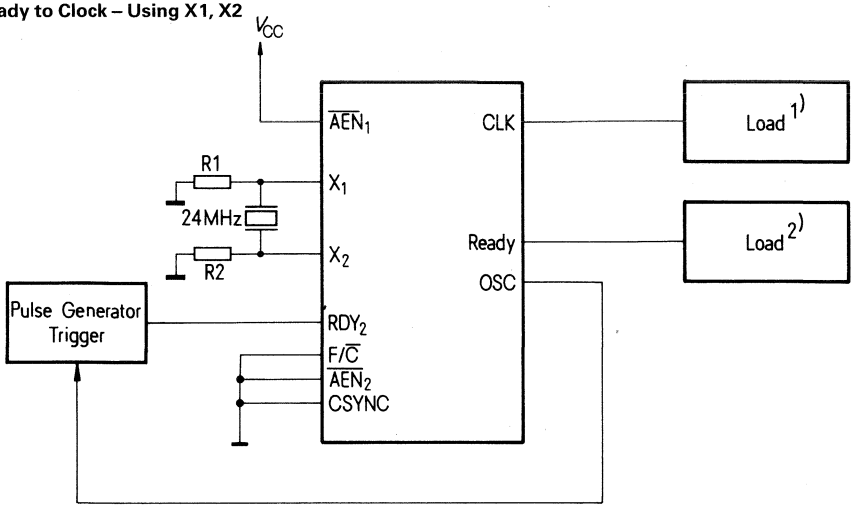


Figure 8
Clock High- and Low Time; Using EFI



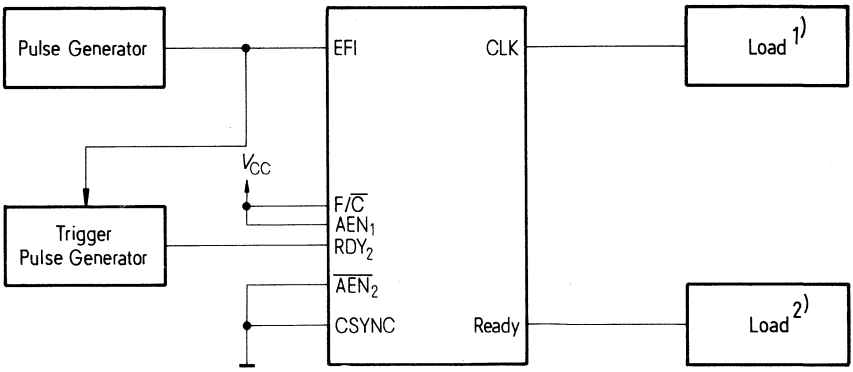
1) C_L = 100 pF

Figure 9
Ready to Clock – Using X1, X2



R1 = R2 = 510 Ω

Figure 10
Ready to Clock – Using EFI



1) C_L = 100 pF
2) C_L = 30 pF

Ordering Information

Component	Description	Ordering Code
	Clock Generator- (Plastic Package)	
SAB 8284B – P	upto 8 MHz	Q67020–Y151
SAB 8284B-1 – P	upto 10 MHz	Q67020–Y152

SIEMENS

Preliminary

SAB 8286A/SAB 8287A Octal Bus Transceiver

- Fully compatible with SAB 8286/SAB 8287
- 40% Less Power Supply Current than Standard SAB 8286/SAB 8287
- Data Bus Buffer Driver for SAB 80286, SAB 80186, SAB 8086, SAB 8085, SAB 8048 and SAB 8051 Families
- High Output Drive Capability for Driving System Data Bus
- Fully Parallel 8-Bit Transceivers
- 3-State Outputs
- 20-Pin Package
- No Output Low Noise when Entering or Leaving High Impedance State

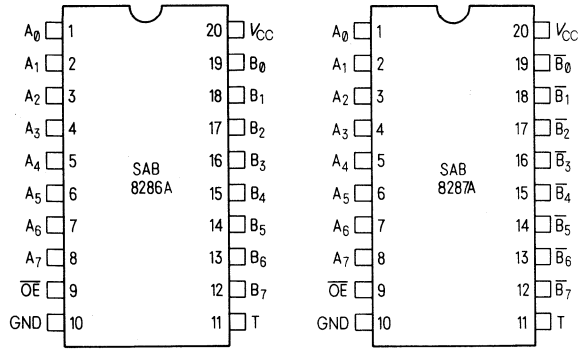
Functional Description

The SAB 8286A and SAB 8287A transceivers are 8-bit transceivers with high impedance outputs. With T active HIGH and $\overline{OE}$ active LOW, data at the A_0-A_7 pins is driven onto the B_0-B_7 pins. With T inactive LOW and $\overline{OE}$ active LOW, data at the B_0-B_7 pins is driven onto the A_0-A_7 pins. No output low glitching will occur whenever the transceivers are entering or leaving the high impedance state.

Absolute Maximum Ratings[1)]

Temperature Under Bias	0 to +70°C
Storage Temperature	−65 to +150°C
All Output and Supply Voltages	−0.5 to +7V
All Input Voltages	−1.0 to +5.5V
Power Dissipation	1W

Pin Configuration

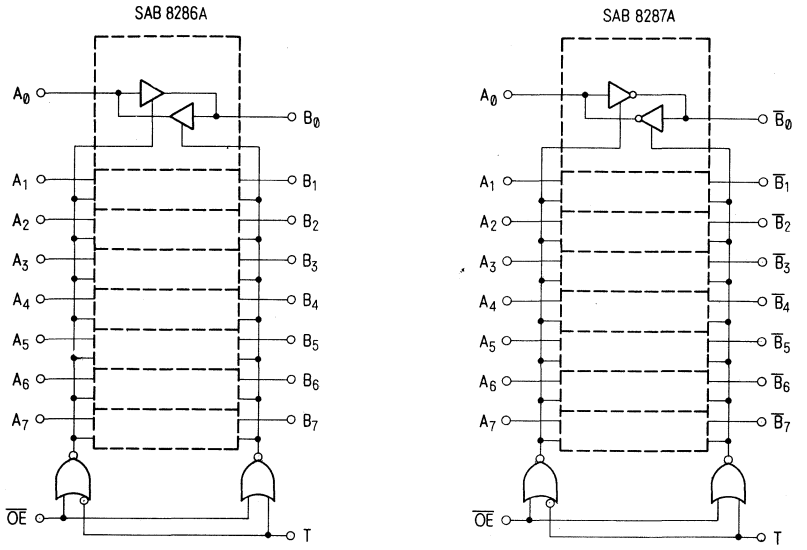


Pin Names

A_0-A_7	Local Bus Data
B_0-B_7	System Bus Data
$\overline{OE}$	Output Enable
T	Transmit
V_{CC}	Power Supply (+5V)
GND	Ground (0V)

The SAB 8286A and SAB 8287A are 8-bit bipolar transceivers with 3-state outputs. The SAB 8287A inverts the input data at its outputs while the SAB 8286A does not. Thus, a wide variety of applications for buffering in microcomputer systems can be met. This device is fabricated in a fast bipolar ASBC (Advanced Standard Buried Collector) process of Siemens.

Logic Diagrams



Pin Definitions and Functions

Symbol	Number	Input (I) Output (O)	Function
T	11	I	TRANSMIT – T is an input control signal used to control the direction of the transceivers. When HIGH, it configures the transceiver's B_0–B_7 as outputs with A_0–A_7 as inputs. T LOW configures A_{0-7} as the outputs with B_0–B_7 serving as the inputs.
$\overline{OE}$	9	I	OUTPUT ENABLE – $\overline{OE}$ is an input control signal used to enable the appropriate output driver (as selected by T) onto its respective bus. This signal is active LOW.
A_0–A_7	1–8	I/O	LOCAL BUS DATA PINS – These pins serve to either present data to or accept data from the processor's local bus depending upon the state of the T pin.
B_0–B_7 (SAB 8286A) $\overline{B_0}$–$\overline{B_7}$ (SAB 8287A)	12–19	I/O	SYSTEM BUS DATA PINS – These pins serve to either present data to or accept data from the system bus depending upon the state of the T pin.
V_{CC}	20	–	Power Supply (+5V)
GND	10	–	Ground (0V)

Functional Description

The SAB 8286A and SAB 8287A transceivers are 8-bit transceivers with high impedance outputs. With T active HIGH and $\overline{OE}$ active LOW, data at the A_0–A_7 pins is driven onto the B_0–B_7 pins.

With T inactive LOW and $\overline{OE}$ active LOW, data at the B_0–B_7 pins is driven onto the A_0–A_7 pins. No output low glitching will occur whenever the transceivers are entering or leaving the high impedance state.

Absolute Maximum Ratings[1)]

Temperature Under Bias	0 to +70°C
Storage Temperature	−65 to +150°C
All Output and Supply Voltages	−0.5 to +7V
All Input Voltages	−1.0 to +5.5V
Power Dissipation	1W

D. C. Characteristics

T_A = 0 to 70°C; V_{CC} = +5V ±10%

Symbol	Parameter	Limit Values		Unit	Test Condition
		Min.	Max.		
V_C	Input Clamp Voltage	–	−1	V	I_C = −5 mA
I_{CC}	Power Supply Current	–	90	mA	All outputs open
I_F	Forward Input Current	–	−0.2	mA	V_F = 0.45V
I_R	Reverse Input Current	–	50	µA	V_R = 5.25V
V_{OL}	Output LOW Voltage – B Outputs – A Outputs	–	0.45 0.45	V	I_{OL} = 32 mA I_{OL} = 16 mA
V_{OH}	Output HIGH Voltage – B Outputs – A Outputs	2.4 2.4	–	V	I_{OH} = −5 mA I_{OH} = −1 mA
I_{OFF} I_{OFF}	Output Off Current Output Off Current	–	I_F I_R	–	V_{OFF} = 0.45V V_{OFF} = 5.25V
V_{IL}	Input LOW Voltage – A Side – B Side	–	0.8 0.9	V	V_{CC} = 5.0V, See Note 2 V_{CC} = 5.0V, See Note 2
V_{IH}	Input HIGH Voltage	2.0			V_{CC} = 5.0V, See Note 2
C_{IN}	Input Capacitance	–	12	pF	F = 1 MHz V_{BIAS} = 2.5V, V_{CC} = 5V T_A = 25°C

1) Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

2) B Outputs: I_{OL} = 32 mA; I_{OH} = −5 mA; C_L = 300 pF
A Outputs: I_{OL} = 16 mA; I_{OH} = −1 mA; C_L = 100 pF

A.C. Characteristics

T_A = 0 to +70°C; V_{CC} = +5 V ± 10%

Loading
B Outputs: I_{OL} = 32 mA; I_{OH} = −5 mA; C_L = 300 pF A Outputs: I_{OL} = 16 mA; I_{OH} = −1 mA; C_L = 100 pF

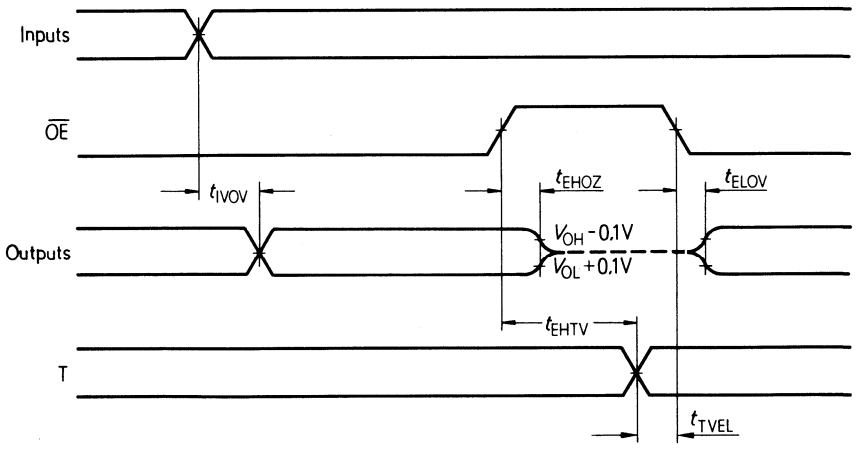
Symbol	Parameter	Limit Values		Unit	Test Condition
		Min.	Max.		
t_{IVOV}	Input to Output Delay Inverting Non-Inverting	 5 5	 22 30	ns	1)
t_{EHTV}	Transmit/Receive Hold Time	5	–		
t_{TVEL}	Transmit/Receive Setup	10			
t_{EHOZ}	Output Disable Time	5	18		
t_{ELOV}	Output Enable Time	10	30		
t_{ILIH}, t_{OLOH}	Input, Output Rise Time	–	20		From 0.8 to 2.0 V
t_{IHIL}, t_{OHOL}	Input, Output Fall Time		12		From 2.0 to 0.8 V

1) See waveforms and test load circuit.

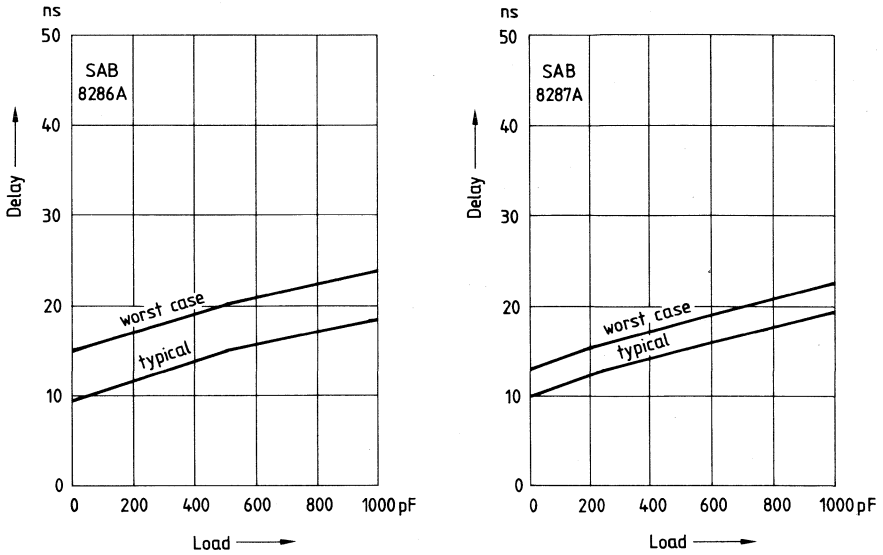
Waveforms

All timing measurements are made at 1.5 V unless otherwise noted.

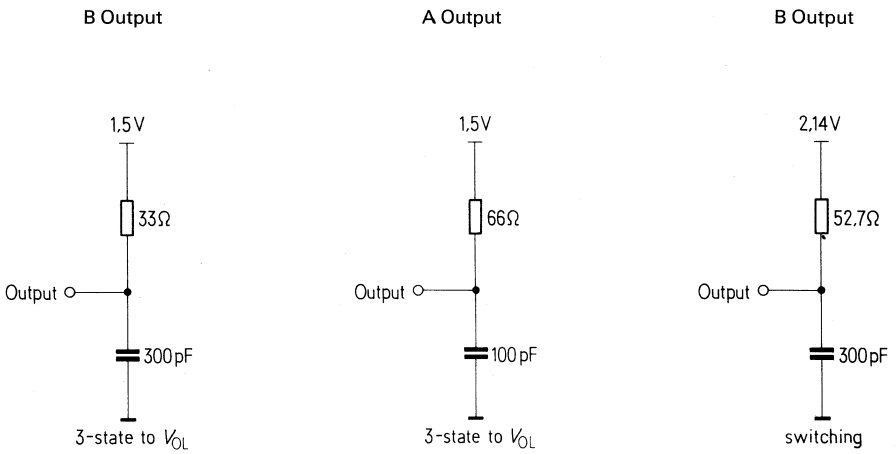
Timing



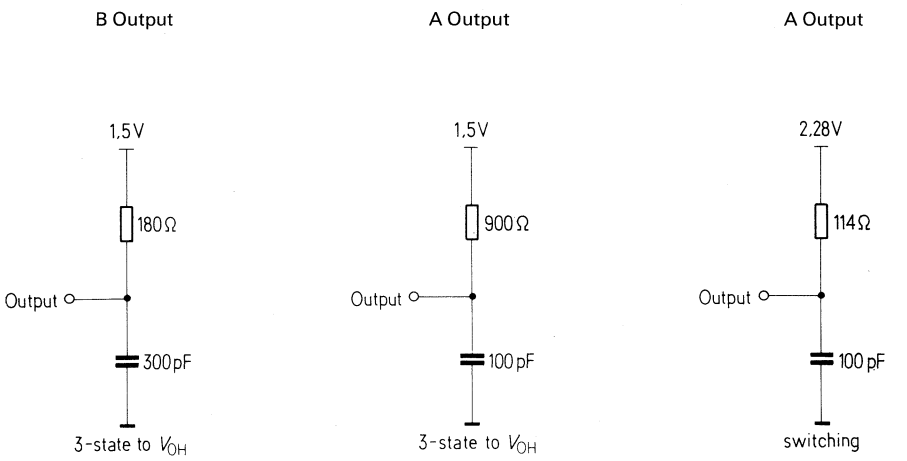
Output Delay vs. Capacitance



Output Test Load Circuit



Output Test Load Circuit



Ordering Information

Type	Description	Ordering code
SAB 8286A–P	Octal Bus Transceiver, non inverting (plastic)	Q 67020–Y 153
SAB 8287A–P	Octal Bus Transceiver, inverting (plastic)	Q 67020–Y 154

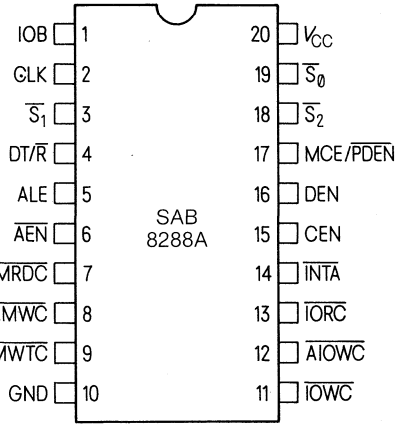
SIEMENS

Preliminary

SAB 8288A Bus Controller for SAB 8086 Family Processors

- Fully compatible with SAB 8288
- 40% Less Power Supply Current than Standard SAB 8288
- Bipolar Drive Capability
- Provides Advanced Commands
- Provides Wide Flexibility in System Configurations
- 3-State Command Output Drivers
- Configurable for Use with an I/O Bus
- Facilitates Interface to One or Two Multi-Master Busses

Pin Configuration



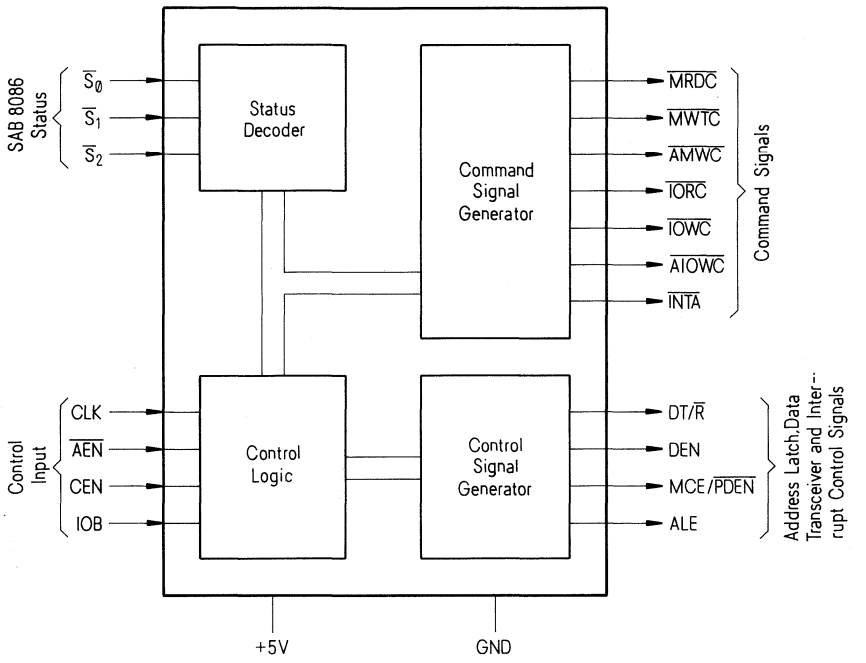
Pin Names

$\overline{S_0} - \overline{S_2}$	Status
CLK	Clock
ALE	Adress Latch Enable
DEN	Data Enable
DT/$\overline{R}$	Data Transmit/Receive
$\overline{AEN}$	Address Enable
CEN	Command Enable
IOB	Input/Output Bus Mode
$\overline{AIOWC}$	Advanced I/O Write
$\overline{IOWC}$	I/O Write
$\overline{IORC}$	I/O Read
$\overline{AMWC}$	Advanced Memory Write
$\overline{MWTC}$	Memory Write
$\overline{MRDC}$	Memory Read
$\overline{INTA}$	Interrupt Acknowledge
MCE/$\overline{PDEN}$	Master Cascade/Peripheral Data
V_{CC}	Power Supply (+5V)
GND	Ground (0V)

SAB 8288A Bus Controller is a 20-pin bipolar component for use with medium-to-large SAB 80186, SAB 80188, SAB 8086 and SAB 8088 processing systems. The bus controller provides command and control timing generation as well as bipolar bus drive capability while optimizing system performance.

A strapping option on the bus controller configures it for use with a multi-master system bus and separate I/O bus.
This device is fabricated in a fast bipolar ASBC (Advanced Standard Buried Collector) process of Siemens.

Block Diagram



Pin Definitions and Functions

Symbol	Number	Input (I) Output (O)	Function
IOB	1	I	INPUT/OUTPUT BUS MODE – When the IOB is strapped HIGH the SAB 8288A functions in the I/O Bus mode. When it is strapped LOW, the SAB 8288A functions in the System Bus mode. (See sections on I/O Bus and Systems Bus modes).
CLK	2	I	CLOCK – This is a clock signal from the SAB 8284A or SAB 8284B clock generator and serves to establish when command and control signals are generated.
$\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$	3, 18, 19	I	STATUS INPUT PINS – These pins are the status input pins from the SAB 80186, SAB 80188, SAB 8086 or SAB 8088 processors. The SAB 8288A decodes these inputs to generate command and control signals at the appropriate time. When these pins are not in use (passive) they are all HIGH. (See chart under Functional Description).

Symbol	Number	Input (I) Output (O)	Function
DT/$\overline{R}$	4	O	DATA TRANSMIT/RECEIVE – This signal establishes the direction of data flow through the transceivers. A HIGH on this line indicates Transmit (write to I/O or memory) and a LOW indicates Receive (Read).
ALE	5	O	ADDRESS LATCH ENABLE – This signal serves to strobe an address into the address latches. This signal is active HIGH and latching occurs on the falling (HIGH to LOW) transition. ALE is intended for use with transparent D type latches.
$\overline{AEN}$	6	I	ADDRESS ENABLE – $\overline{AEN}$ enables command outputs of the SAB 8288A Bus Controller at least 105 ns after it becomes active (LOW). $\overline{AEN}$ going inactive immediately 3-states the command output drivers. $\overline{AEN}$ does not affect the I/O command lines if the SAB 8288A is in the I/O Bus mode (IOB tied HIGH).
$\overline{MRDC}$	7	O	MEMORY READ COMMAND – This command line instructs the memory to drive its data onto the data bus. This signal is active LOW.
$\overline{AMWC}$	8	O	ADVANCED MEMORY WRITE COMMAND – The AMWC issues a memory write command earlier in the machine cycle to give memory devices an early indication of a write instruction. Its timing is the same as a read command signal. $\overline{AMWC}$ is active LOW.
$\overline{MWTC}$	9	O	MEMORY WRITE COMMAND – This command line instructs the memory to record the data present on the data bus. This signal is active LOW.
$\overline{IOWC}$	11	O	I/O WRITE COMMAND – This command line instructs an I/O device to read the data on the data bus. This signal is active LOW.
$\overline{AIOWC}$	12	O	ADVANCED I/O WRITE COMMAND – The $\overline{AIOWC}$ issues an I/O Write Command earlier in the machine cycle to give I/O devices an early indication of a write instruction. Its timing is the same as a read command signal. $\overline{AIOWC}$ is active LOW.
$\overline{IORC}$	13	O	I/O READ COMMAND – This command line instructs an I/O device to drive its data onto the data bus. This signal is active LOW.
$\overline{INTA}$	14	O	INTERRUPT ACKNOWLEDGE – This command line tells an interrupting device that its interrupt has been acknowledged and that it should drive vectoring information onto the data bus. This signal is active LOW.
CEN	15	I	COMMAND ENABLE – When this signal is LOW all SAB 8288A command outputs and the DEN and $\overline{PDEN}$ control outputs are forced to their inactive state. When this signal is HIGH, these same outputs are enabled.
DEN	16	O	DATA ENABLE – This signal serves to enable data transceivers onto either the local or system data bus. This signal is active HIGH.

Pin Definitions and Functions (continued)

Symbol	Number	Input (I) Output (O)	Function
MCE/$\overline{PDEN}$	17	O	This is a dual function pin: MCE (IOB is tied LOW) – **Master Cascade Enable** occurs during an interrupt sequence and serves to read a Cascade. Address from a master PIC (Priority Interrupt Controller) onto the data bus. The MCE signal is active HIGH. $\overline{PDEN}$ (IOB is tied HIGH) – **Peripheral Data Enable** enables the data bus transceiver for the I/O bus during I/O instructions. It performs the same function for the I/O bus that DEN performs for the system bus. $\overline{PDEN}$ is active LOW.
V_{CC}	20	–	Power Supply (+5V)
GND	10	–	Ground (0V)

Functional Description

The command logic decodes the three SAB 80186, SAB 80188, SAB 8086 or SAB 8088 CPU status lines ($\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$) to determine what command is to be issued. This chart shows the meaning of each status "word".

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	Processor State	SAB 8288A Command
0	0	0	Interrupt Acknowledge	$\overline{INTA}$
0	0	1	Read I/O Port	$\overline{IORC}$
0	1	0	Write I/O Port	$\overline{IOWC}$, $\overline{AIOWC}$
0	1	1	Halt	None
1	0	0	Code Access	$\overline{MRDC}$
1	0	1	Read Memory	$\overline{MRDC}$
1	1	0	Write Memory	$\overline{MWTC}$, $\overline{AMWC}$
1	1	1	Passive	None

The command is issued in one of two ways dependent on the mode of the SAB 8288A Bus Controller.

I/O Bus Mode – The SAB 8288A is in the I/O Bus mode if the IOB pin is strapped HIGH. In the I/O Bus mode all I/O command lines ($\overline{IORC}$, $\overline{IOWC}$, $\overline{AIOWC}$, $\overline{INTA}$) are always enabled (i.e., not dependent on $\overline{AEN}$). When an I/O command is initiated by the processor, the SAB 8288A immediately activates the command lines using $\overline{PDEN}$ and DT/$\overline{R}$ to control the I/O bus transceiver. The I/O command lines should not be used to control the system bus in this configuration because no arbitration is present. This mode allows one SAB 8288A Bus Controller to handle two external busses. No waiting is involved when the CPU wants to gain access to the I/O bus. Normal memory access requires a "Bus Ready" signal ($\overline{AEN}$ LOW) before it will proceed. It is advantageous to use the IOB mode if I/O or peripherals dedicated to one processor exist in a multi-processor system.

System Bus Mode – The SAB 8288A is in the System Bus mode if the IOB pin is strapped LOW. In this mode no command is issued until 115 ns after the $\overline{AEN}$ Line is activated (LOW). This mode assumes bus arbitration logic will inform the bus controller (on the $\overline{AEN}$ line) when the bus is free for use. Both memory and I/O commands wait for bus arbitration. This mode is used when only one bus exists. Here, both I/O and memory are shared by more than one processor.

Command Outputs

The advanced write commands are made available to initiate write procedures early in the machine cycle. This signal can be used to prevent the processor from entering an unnecessary wait state.

The command output are:
$\overline{MRDC}$ – Memory Read Command
$\overline{MWTC}$ – Memory Write Command
$\overline{IORC}$ – I/O Read Command
$\overline{IOWC}$ – I/O Write Command
$\overline{AMWC}$ – Advanced Memory Write Command
$\overline{AIOWC}$ – Advanced I/O Write Command
$\overline{INTA}$ – Interrupt Acknowledge

$\overline{INTA}$ (Interrupt Acknowledge) acts as an I/O read during an interrupt cycle. Its purpose is to inform an interrupting device that its interrupt is being acknowledged and that it should place vectoring information onto the data bus.

Control Outputs

The control outputs of the SAB 8288A are Data Enable (DEN), Data Transmit/Receive (DT/$\overline{R}$) and Master Cascade Enable/Peripheral Data Enable (MCE/$\overline{PDEN}$). The DEN signal determines when the external bus should be enable onto the local bus and the DT/$\overline{R}$ determines the direction of data transfer. These two signals usually go to the chip select and direction pins of a transceiver.

The MCE/$\overline{PDEN}$ pin changes function with the two modes of the SAB 8288A. When the SAB 8288A is in the IOB mode (IOB HIGH) the PDEN signal serves as a dedicated data enable signal for the I/O or Peripheral System bus.

Interrupt Acknowledge and MCE

The MCE signal is used during an interrupt acknowledge cycle if the SAB 8288A is in the System Bus mode (IOB LOW). During any interrupt sequence there are two interrupt acknowledge cycle no data or address transfers take place. Logic should be provided to mask off MCE during this cycle. Just before the second cycle begins the MCE signal gates a master Priority Interrupt Controller's (PIC) cascade address onto the processor's local bus where ALE (Address Latch Enable) strobes it into the address latches. On the leading edge of the second interrupt cycle the addressed slave PIC gates an interrupt vector onto the system data bus where it is read by the processor.

If the system contains only one PIC, the MCE signal is not used. In this case the second Interrupt Acknowledge signal gates the interrupt vector onto the processor bus.

Address Latch Enable and Halt

Address Latch Enable (ALE) occurs during each machine cycle and serves to strobe the current address into the address latches. ALE also serves to strobe the status ($\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$) into a latch for halt state decoding.

Command Enable

The Command Enable (CEN) input acts as a command qualifier for the SAB 8288A. If the CEN pin is high the SAB 8288A functions normally. If the CEN pin is pulled LOW, all command lines are held in their inactive state (not 3-state). This feature can be used to implement memory partitioning and to eliminate address conflicts between system bus devices and resident bus devices.

Absolute Maximum Ratings [1]

Temperature Under Bias	0 to + 70 C
Storage Temperature	− 65 to + 150 C
All Output and Supply Voltages	− 0.5 to + 7 V
All Input Voltages	− 1.0 to + 5.5 V
Power Dissipation	1 W

D.C. Characteristics

T_A = 0 to 70 C; V_{CC} = + 5 V + 10%

Symbol	Parameter	Limit Values		Units	Test Conditions
		Min.	Max.		
V_C	Input Clamp Voltage		− 1	V	I_C = − 5 mA
I_{CC}	Power Supply Current		140	mA	All outputs open
I_F	Forward Input Current	−	− 0.7		V_F = 0.45 V
I_R	Reserve Input Current		50	µA	V_R = V_{CC}
V_{OL}	Output Low Voltage Command Outputs Control Outputs		 0.5 0.5	V	 I_{OL} = 32 mA I_{OL} = 16 mA
V_{OH}	Output High Voltage Command Outputs Control Outputs	 2.4 2.4	 −		 I_{OH} = − 5 mA I_{OH} = − 1 mA
V_{IL}	Input Low Voltage	−	0.8		−
V_{IH}	Input High Voltage	2.0	−		
I_{OFF}	Output Off Current	−	100	µA	V_{OFF} = 0.4 to 5.25 V

A.C. Characteristics

T_A = 0 to 70 C; V_{CC} = + 5 V ± 10%

Timing Requirements

Symbol	Parameter	Limit Values		Units	Test Conditions
		Min.	Max.		
t_{CLCL}	CLK Cycle Period	100			
t_{CLCH}	CLK Low Time	50			
t_{CHCL}	CLK High Time	30			
t_{SVCH}	Status Active Setup Time	35	−	ns	−
t_{CHSV}	Status Active Hold Time	10			
t_{SHCL}	Status Inactive Setup Time	35			
t_{CLSH}	Status Inactive Hold Time	10			
t_{ILIH}	Input, Rise Time	−	20		From 0.8V to 2.0V
t_{IHIL}	Input, Fall Time		12		From 2.0V to 0.8V

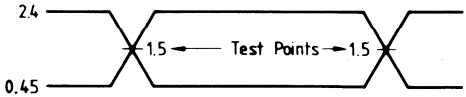
1) Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Timing Responses

Symbol	Parameter	Limit Values		Units	Test Conditions
		Min.	Max.		
t_{CVNV}	Control Active Delay	5	45	ns	
t_{CVNX}	Control Inactive Delay	10			
t_{CLLH}, t_{CLMCH}	ALE MCE Active Delay (from CLK)	–	20		
t_{SVLH}, t_{SVMCH}	ALE MCE Active Delay (from Status)				
t_{CHLL}	AIE Inactive Delay	4	15		$\overline{MRDC}$, $\overline{IORC}$, $\overline{MWTC}$, $\overline{IOWC}$, $\overline{INTA}$, $\overline{AMWC}$, $\overline{AIOWC}$: I_{OL} = 32 mA, I_{OH} = −5 mA, C_L = 300 pF
t_{CLML}	Command Active Delay	10	35		
t_{CLMH}	Command Inactive Delay				
t_{CHDTL}	Direction Control Active Delay	–	50		
t_{CHDTH}	Direction Control Inactive Delay		30		
t_{AELCH}	Command Enable Time		40		Other: I_{OL} = 16 mA, I_{OH} = −1 mA, C_L = 80 pF
t_{AEHCZ}	Command Disable Time				
t_{AELCV}	Enable Delay Time	115	200		
t_{AEVNV}	$\overline{AEN}$ to DEN	–	20		
t_{CEVNV}	CEN to DEN, PDEN		25		
t_{CELRH}	CEN to Command		t_{CLML}		
t_{OLOH}	Output, Rise Time		20		From 0.8V to 2.0V
t_{OHOL}	Output, Fall Time		12		From 2.0V to 0.8V

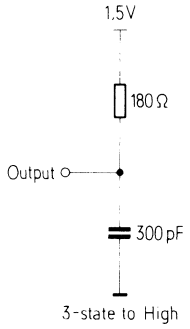
A.C. Testing Input, Output Waveform

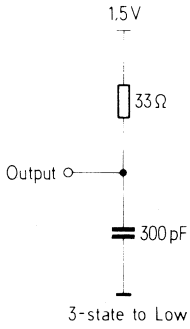
Input/Output



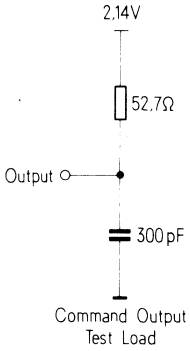
A.C. Testing Inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0". The clock is driven at 4.3V and 0.25V timing measurements are made at 1.5V for both a logic "1" and "0"

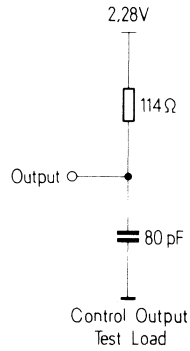
Test Load Circuits – 3-State Command Output Test Load





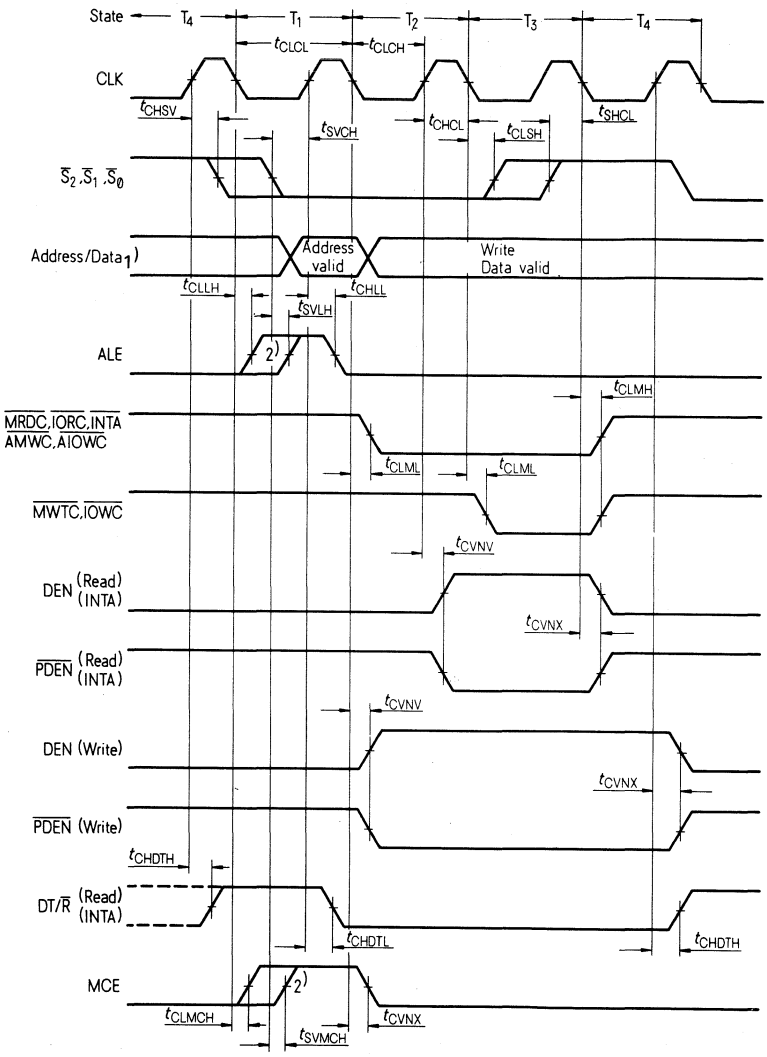
Test Load Circuits – 3-State Command Output Test Load





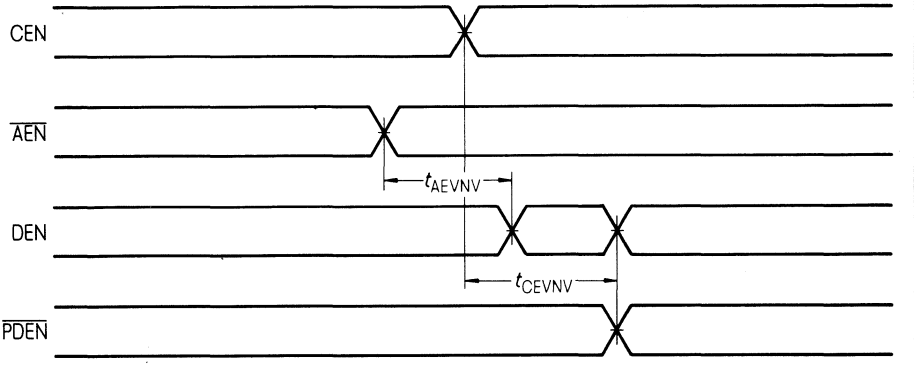
Waveforms

Timing Diagram

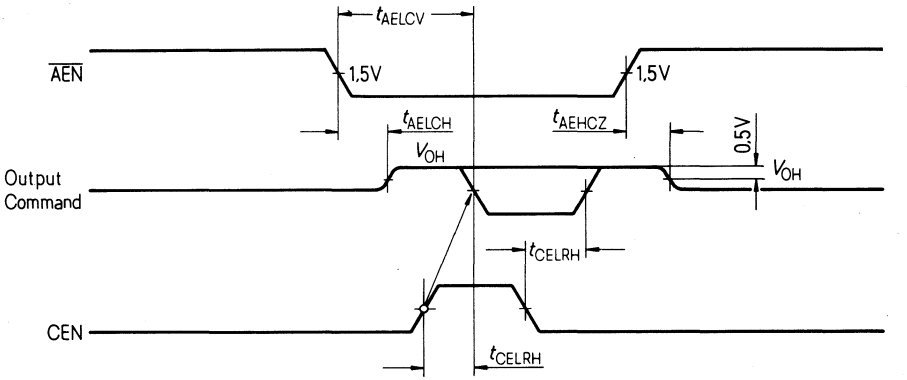


1) Address/Data Bus is shown only for reference purposes
2) Leading edge of ALE and MCE is determined by the falling edge of CLK or status going active, whichever occurs last.
3) All timing measurements are made at 1.5V unless specified otherwise.

DEN, PDEN Qualification Timing



Address Enable (AEN) Timing (3-State Enable/Disable)



CEN must be low or valid prior to T2 to prevent the command from being generated.

Ordering Information

Type	Description	Ordering code
SAB 8288A-P	Bus Controller (plastic)	Q 67020-Y 155

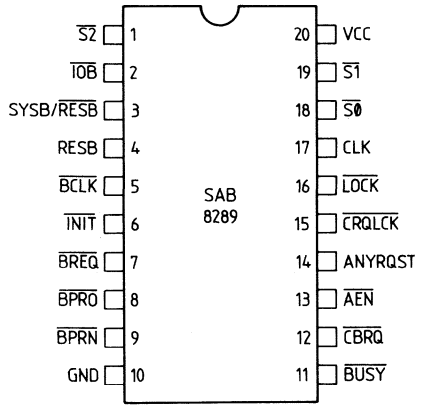
Preliminary

SAB 8289
Bus Arbiter

SAB 8289 8 MHz
SAB 8289-1 10 MHz

- Provides Multi-Master System Bus Protocol
- Synchronizes SAB 8086/SAB 8088 Processors with Multi-Master Bus
- Provides Simple Interface with SAB 8288 Bus Controller
- Four Operating Modes for Flexible System Configuration
- Compatible with Intel Bus Standard MULTIBUS™ (MULTIBUS is a trademark of INTEL Corporation USA)
- Provides System Bus Arbitration for SAB 8089 IOP in Remote Mode

Pin Configuration



Pin Names

$\overline{S0}$-2	Status
$\overline{IOB}$	I/O Bus Present
SYSB/$\overline{RESB}$	System/Resident Bus
RESB	Resident Bus Present
$\overline{BCLK}$	Bus Clock
$\overline{INIT}$	Initialisation
$\overline{BREQ}$	Bus Request
$\overline{BPRO}$	Bus Priority Out
$\overline{BPRN}$	Bus Priority In
CLK	System Clock
$\overline{AEN}$	Address Enable
ANYRQST	Any Request
$\overline{CRQLCK}$	Common Request Lock
$\overline{CBRQ}$	Common Bus Request
$\overline{LOCK}$	Bus Lock
VCC	+5 V
GND	Ground (0V)

The SAB 8289 Bus Arbiter is a 20-pin, 5-volt-only bipolar component for use with medium to large SAB 8086/SAB 8088 multi-master/multiprocessing systems. The SAB 8289 provides system bus arbitration for systems with multiple bus masters, such as an SAB 8086 CPU with SAB 8089 IOP in its REMOTE mode, while providing bipolar buffering and drive capability.

Pin Definitions and Functions

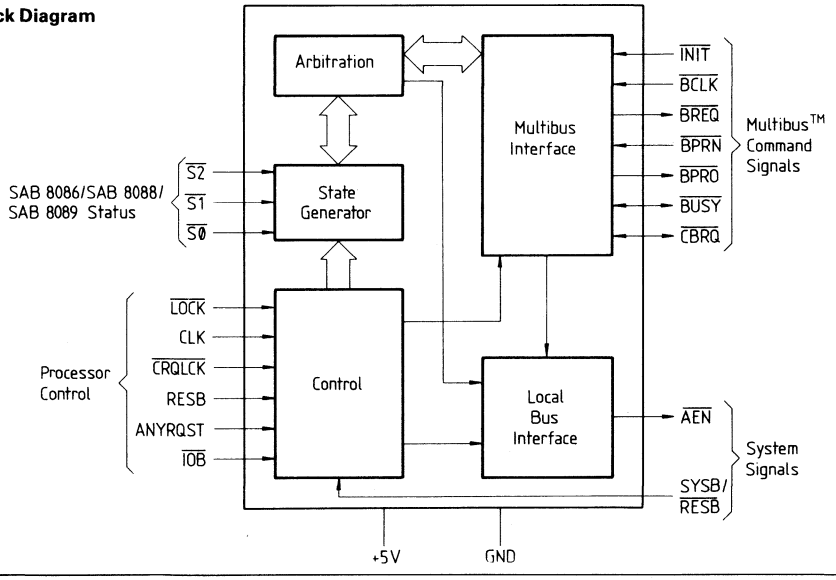
Symbol	Number	Input (I) Output (O)	Function
$\overline{S0}$, $\overline{S1}$, $\overline{S2}$	1, 18, 19	I	**Status Input Pins** These pins are the status input pins from a SAB 8086, SAB 8088 or SAB 8089 processor. The SAB 8289 decodes these pins to initiate bus request and surrender actions.
CLK	17	I	**Clock** This is the clock from the SAB 8284A clock chip and serves to establish when bus arbiter actions are initiated.
$\overline{LOCK}$	16	I	**Lock** $\overline{LOCK}$ is a processor generated signal which when activated (low) serves to prevent the arbiter from surrendering the multi-master system bus to any other bus arbiter, regardless of its priority.
$\overline{CRQLCK}$	15	I	**Common Request Lock** $\overline{CRQLCK}$ is an active low signal which serves to prevent the arbiter from surrendering the multi-master system bus to any other bus arbiter requesting the bus through the $\overline{CBRQ}$ input pin.
RESB	4	I	**Resident Bus** RESB is a strapping option to configure the arbiter to operate in systems having both a multi-master system bus and a Resident Bus. When it is strapped high the multi-master system bus is requested or surrendered as a function of the SYSB/$\overline{RESB}$ input pin. When it is strapped low the SYSB/$\overline{RESB}$ input is ignored.
ANYRQST	14	I	**Any Request** ANYRQST is a strapping option which permits the multimaster system bus to be surrendered to a lower priority arbiter as though it were an arbiter of higher priority (i.e., when a lower priority arbiter requests the use of the multi-master system bus, the bus is surrendered as soon as it is possible). Strapping $\overline{CBRQ}$ low and ANYRQST high forces the SAB 8289 arbiter to surrender the multi-master system bus after each transfer cycle. Note that when surrender occurs $\overline{BREQ}$ is driven false (high).
$\overline{IOB}$	2	I	**IO Bus** $\overline{IOB}$ is a strapping option which configures the SAB 8289 Arbiter to operate in systems having both an IO Bus (Peripheral Bus) and a multimaster system bus. The arbiter requests and surrenders the use of the multimaster system bus as a function of the status line, $\overline{S2}$. The multi-master system bus is permitted to be surrendered while the processor is performing IO commands and is requested whenever the processor performs a memory command. Interrupt cycles are assumed as coming from the peripheral bus and are treated as would be an IO command.
$\overline{AEN}$	13	O	**Address Enable** $\overline{AEN}$ is the output of the SAB 8289 Arbiter to the processor's address latches, to the SAB 8288 Bus Controller and SAB 8284A Clock Generator. $\overline{AEN}$ serves to instruct the Bus Controller and address latches when to tri-state their output drivers.

Symbol	Number	Input (I) Output (O)	Function
SYSB/$\overline{\text{RESB}}$	3	I	**System Bus/Resident Bus** SYSB/$\overline{\text{RESB}}$ is an input signal when the arbiter is configured in the S.R. Mode (RESB is strapped high) which serves to determine when the multimaster system bus is requested and when the multi-master system bus surrendering is permitted. The signal is intended to originate from some form of address mapping circuitry such as a decoder or PROM attached to the resident address bus. Signal transitions and glitches are permitted on this pin from Ø1 of T4 to Ø1 to T2 of the processor cycle. During the period from Ø1 of T2 to Ø1 of T4 only clean transitions are permitted on this pin (no glitches). If a glitch does occur the arbiter may capture or miss it, and the multi-master system bus may be requested or surrendered, depending upon the state of the glitch. The arbiter requests the multi-master system bus in the S.R. Mode when the state of the SYSB/$\overline{\text{RESB}}$ pin is high and permits the bus to be surrendered when this pin is low.
$\overline{\text{CBRQ}}$	12	I/O	**Common Bus Request** $\overline{\text{CBRQ}}$ is an input signal which serves to instruct the arbiter if there are any other arbiters of lower priority requesting the use of the multi-master system bus. The $\overline{\text{CBRQ}}$ pins (open-collector output) of all the SAB 8289 Bus Arbiters which are to surrender the multi-master-system bus upon request are connected together. The Bus Arbiter running the current transfer cycle will not itself pull the $\overline{\text{CBRQ}}$ line low. Any other arbiter connected to the $\overline{\text{CBRQ}}$ line can request the multi-master system bus. The arbiter presently running the current transfer cycle drops its $\overline{\text{BREQ}}$ signal and surrenders the bus whenever the proper surrender conditions exist. Strapping $\overline{\text{CBREQ}}$ low and ANYRQST – high allows the multi-master system bus to be surrendered after each transfer cycle. See the pin definition of ANYRQST.
$\overline{\text{INIT}}$	6	I	**Initialize** $\overline{\text{INIT}}$ is an active low multimaster system bus input signal which is used to reset all the bus arbiters on the multi-master system bus. After initialization, no arbiters have the use of the multi-master system bus.
$\overline{\text{BCLK}}$	5	I	**Bus Clock** $\overline{\text{BCLK}}$ is the multi-master system bus clock to which all multimaster system bus interface signals are synchronized.
$\overline{\text{BREQ}}$	7	0	**Bus Request** $\overline{\text{BREQ}}$ is an active low output signal in the parallel Priority Resolving Scheme which the arbiter activates to request the use of the multi-master system bus.
$\overline{\text{BPRN}}$	9	I	**Bus Priority In** $\overline{\text{BPRN}}$ is the active low signal returned to the arbiter to instruct it that it may acquire the multimaster system bus on the next falling edge of BCLK. $\overline{\text{BPRN}}$ indicates to the arbiter that it is the highest priority requesting arbiter presently on the bus. The loss of $\overline{\text{BPRN}}$ instructs the arbiter that it has loss priority to a higher priority arbiter.

Pin Definitions and Functions (continued)

Symbol	Number	Input (I) Output (O)	Function
$\overline{BPRO}$	8	O	**Bus Priority Out** $\overline{BPRO}$ is an active low output signal which is used in the serial priority resolving scheme where $\overline{BPRO}$ is daisy chained to $\overline{BPRN}$ of the next lower priority arbiter.
$\overline{BUSY}$	11	I/O	**Busy** $\overline{BUSY}$ is an active low open collector multi-master system bus interface signal which is used to instruct all the arbiters on the bus when the multi-master system bus is available. When the multi-master system bus is available the highest requesting arbiter (determined by $\overline{BPRN}$) seizes the bus and pulls $\overline{BUSY}$ low to keep other arbiters off of the bus. When the arbiter is done with the bus it releases the $\overline{BUSY}$ signal permitting it to go high and thereby allowing another arbiter to acquire the multi-master system bus.
VCC	20	I	**Power Supply** (+5 V ± 10%)
GND	10	I	**Ground** (OV)

Block Diagram



Functional Description

The SAB 8289 Bus Arbiter operates in conjunction with the SAB 8288 Bus Controller to interface SAB 8086/SAB 8088/ SAB 8089 processors to a multi-master system bus (both the SAB 8086 and the SAB 8088 are configured in their max mode). The processor is unaware of the arbiter's existence and issues commands as though it has exclusive use of the system bus. If the processor does not have the use of the multi-master system bus, the arbiter prevents the Bus Controller (SAB 8288), the data transceivers and the address latches from accessing the system bus (e.g. all bus driver outputs are forced into the high impedance state). Since the command sequence was not issued by the SAB 8288, the system bus will appear as "Not Ready" and the processor will enter wait states. The processor will remain in Wait until the Bus Arbiter acquires the use of the multi-master system bus whereupon the arbiter will allow the bus controller, the data transceivers, and the address latches to access the system. Typically, once the command has been issued and a data transfer has taken place, a transfer acknowledge (XACK) is returned to the processor to indicate "READY" from the accessed slave device. The processor then completes its transfer cycle. Thus the arbiter serves to multiplex a processor (or bus master) onto a multi-master system bus and avoid contention problems between bus masters.

Arbitration between Bus Masters

In general, higher priority masters obtain the bus when a lower priority master completes its present transfer cycle. Lower priority bus masters obtain the bus when a higher priority master is not accessing the system bus. A strapping option (ANYRQST) is provided to allow the arbiter to surrender the bus to a lower priority master as though it were a master of higher priority. If there are no other bus masters requesting the bus, the arbiter maintains the bus so long as its processor has not entered the HALT State. The arbiter will not voluntarily surrender the system bus and has to be forced off by another master's bus request, the HALT State being the only exception. Additional strapping options permit other modes of operation wherein the multimaster system bus is surrendered or requested under different sets of conditions.

Modes of Operation

There are two types of processors in the SAB 8086 family. An Input/Output processor (the SAB 8089 IOP) and the SAB 8086/SAB 8088 CPUs. Consequently, there are two basic operating modes in the SAB 8289 bus arbiter. One, the $\overline{IOB}$ (I/O Peripheral Bus) mode, permits the processor access to both an I/O Peripheral Bus and a multi-master system bus. The second, the RESB (Resident Bus mode), permits the processor to communicate over both a Resident Bus and a multi-master system bus. An I/O Peripheral Bus is a bus where all devices on that bus, including memory, are treated as I/O devices and are addressed by I/O commands. All memory commands are directed to another bus, the multi-master system bus. A Resident Bus can issue both memory and I/O commands, but it is a distinct and separate bus from the multi-master system bus. The distinction is that the Resident Bus has only one master, providing full availability and being dedicated to that one master.

The $\overline{IOB}$ strapping option configures the SAB 8289 Bus Arbiter into the $\overline{IOB}$ mode and the strapping option RESB configures it into the RESB mode. It might be noted at this point that if both strapping options are strapped false, the arbiter interfaces the processor to a multi-master system bus only. With both options strapped true, the arbiter interfaces the processor to a multi-master system bus, a Resident Bus, and an I/O Bus.

In the $\overline{IOB}$ mode, the processor communicates and controls a host of peripherals over the Peripheral Bus. When the I/O Processor needs to communicate with system memory, it does so over the system memory bus.

The SAB 8086 and SAB 8088 processor can communicate with a Resident Bus and a multi-master system bus. Two bus controllers and only one Bus Arbiter would be needed in such a configuration. In such a system configuration the processor would have access to memory and peripheral of both busses. Memory mapping techniques are applied to select which bus is to be accessed. The SYSB/$\overline{RESB}$ input on the arbiter serves to instruct the arbiter as to whether or not the system bus is to be accessed. The signal connected to SYSB/$\overline{RESB}$ also enables or disables commands from one of the bus controllers.

Summary of SAB 8289 Modes, Requesting and Relinquisting the Multi-master system bus

Status Lines From SAB 8086 / 88 / 89				IOB Mode Only	RESB (Mode) Only IOB=High RESB=High		IOB Mode RESB Mode IOB=Low RESB=High		Single Bus Mode IOB=High RESB=Low
	$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	IOB=Low	SYSB/$\overline{RESB}$ =High	SYSB/$\overline{RESB}$ =Low	SYSB/$\overline{RESB}$ =High	SYSB/$\overline{RESB}$ =Low	
I/O	0	0	0	x		x	x	x	
COMMANDS	0	0	1	x		x	x	x	
	0	1	0	x		x	x	x	
HALT	0	1	1	x	x	x	x	x	x
MEM	1	0	0			x		x	
COMMANDS	1	0	1			x		x	
	1	1	0			x		x	
IDLE	1	1	1	x	x	x	x	x	x

NOTE:
x = Multi-Master System Bus is allowed to be Surrendered.

Mode	Pin Strapping	Multi-Master System Bus	
		Requested**	Surrendered*
Single Bus Multi-Master Mode	$\overline{IOB}$=High RESB=Low	Whenever the processor's status lines go active	HLT+TI•CBRQ+HPBRQ***
RESB Mode Only	$\overline{IOB}$=High RESB=High	SYSB/$\overline{RESB}$=High• ACTIVE STATUS	(SYSB/$\overline{RESB}$=Low+TI)• CRBQ+HLT+HPBRQ
IOB Mode Only	IOB=Low RESB=Low	Memory Commands	(I/O Status+TI)•CBRQ+ HLT+HPBRQ
IOB Mode•RESB Mode	$\overline{IOB}$=Low RESB=High	(Memory Command)• (SYSB/$\overline{RESB}$=High)	((I/O Status Commands)+ SYSB/RESB=LOW))•CBRQ +HPBRQ***+HLT

NOTES:
*$\overline{LOCK}$ prevents surrender of Bus to any other arbiter, $\overline{CRQLCK}$ prevents surrender of Bus to any lower priority arbiter.
**Except for HALT and Passive or IDLE Status.
***HPBRQ, Higher priority Bus request or $\overline{BPRN}$=1.
1. $\overline{IOB}$ Active Low.
2. RESB Active High.
3. +is read as "OR" and•as "AND".
4. TI=Processor Idle Status $\overline{S2}$, $\overline{S1}$, $\overline{S0}$=111
5. HLT=Processor Halt Status $\overline{S2}$, $\overline{S1}$, $\overline{S0}$=011

Absolute Maximum Ratings*

Ambient Temperature Under Bias	0 to 70°C
Storage Temperature	–65 to 150°C
All Output and Supply Voltages	–0.5 to 7 V
All Input Voltages	–1.0 to 5.5 V
Power Dissipation	1.5 Watt

D.C. Characteristics

T_A = 0 to 70°C, V_{CC} = 5 V ± 10%

Symbol	Parameter	Limit Values		Unit	Test Condition
		Min.	Max.		
V_C	Input Clamp Voltage	–	–1.0	V	V_{CC}=4.5 V, I_C=–5 mA
I_F	Input Forward Current		–0.5	mA	V_{CC}=5.5 V, V_F=0.45 V
I_R	Reverse Input Leakage Current		60	µA	V_{CC}=5.5 V, V_R=5.5 V
V_{OL}	Output Low Voltage $\overline{BUSY}$, $\overline{CBRQ}$ $\overline{AEN}$ $\overline{BPRO}$, $\overline{BREQ}$		0.45	V	I_{OL}=20 mA I_{OL}=16 mA I_{OL}=10 mA
V_{OH}	Output High Voltage $\overline{BUSY}$, $\overline{CBRQ}$	Open Collector			–
	All Other Outputs	2.4	–	V	I_{OH}=400 µA
I_{CC}	Power Supply Current	–	165	mA	–
V_{IL}	Input Low Voltage		0.8	V	
V_{IH}	Input High Voltage	2.0	–		
C_{in} Status	Input Capacitance	–	25	pF	
C_{in} (Others)	Input Capacitance		12		

*) Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended perods may affect device reliability.

A.C. Characteristics

T_A=0 to 70°C, V_{CC}=5 V ± 10%

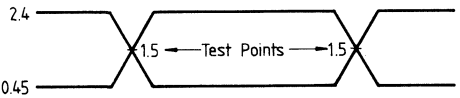
Timing Requirements

Symbol	Parameter	Limit Values				Unit	Test Condition
		SAB 8289		SAB 8289-1			
		Min.	Max.	Min.	Max.		
TCLCL	CLK Cycle Period	125		100			
TCLCH	CLK Low Time	65	–	53	–		
TCHCL	CLK High Time	35		35			
TSVCH	Status Active-Setup	65	TCLCL-10	55	TCLCL-10		
TSHCL	Status Inactive-Setup	50		45			
THVCH	Status Active Hold	10		10			
THVCL	Status Inactive Hold						
TBYSBL	Busy↑↓Setup to BCLK↓	20	–	20	–		–
TCBSBL	CBRQ↑↓Setup to BCLK↓					ns	
TBLBL	BCLK Cycle Time	100		100			
TBHCL	BCLK High Time	30	0.65 [TBLBL]	30	0.65 [TBLBL]		
TCLLL1	LOCK Inactive Hold	10		10			
TCLLL2	LOCK Active Setup	40		40			
TPNBL	BPRN↓↑ to BCLK Setup Time	15	–	15	–		
TCLSR1	SYSB/RESB Setup	0		0			
TCLSR2	SYSB/RESB Hold	20		20			
TNIH	Initialization Pulse Width	3TBLBL+ 3TCLCL		3TBLBL+ 3TCLCL			
TILIH	Input Rise Time	–	20	–	20		From 0.8 to 2.0 V
TIHIL	Input Fall Time		12		12		From 2.0 to 0.8 V

↑↓ Denotes the spec applies to both transitions of the signal.

A.C. Testing Input/Output Waveform

Input/Output



A.C. Testing: Inputs are driven at 2.4 V for a logic "1" and 0.45 V for a logic "0". The clock is driven at 4.3 V and 0.25 V. Timing measurements are made at 1.5 V for both a logic "1" and "0".

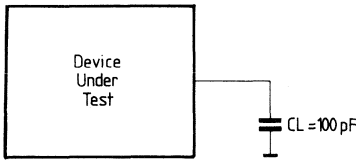
Timing Responses

Symbol	Parameter	Limit Values		Unit	Test Condition
		Min.	Max.		
TBLBRL	BCLK to BREQ Delay↓↑	–	35	–	–
TBLPOH	BCLK to BPRO↓↑[1])		40		
TPNPO	BPRN↓↑ to BPRO↓↑ Delay[1]		25		
TBLBYL	BCLK to BUSY Low		60		
TBLBYH	BCLK to BUSY Float[2]		35		
TCLAEH	CLK to AEN High		65	ns	
TBLAEL	BCLK to AEN Low		40		
TBLCBL	BCLK to CBRQ Low		60		
TBLCRH	BCLK to CBRQ Float[2]		35		
TOLOH	Output Rise Time		20		From 0.8 to 2.0 V
TOHOL	Output Fall Time		12		From 2.0 to 0.8 V

↓↑ Denotes the spec applies to both transitions of the signal.

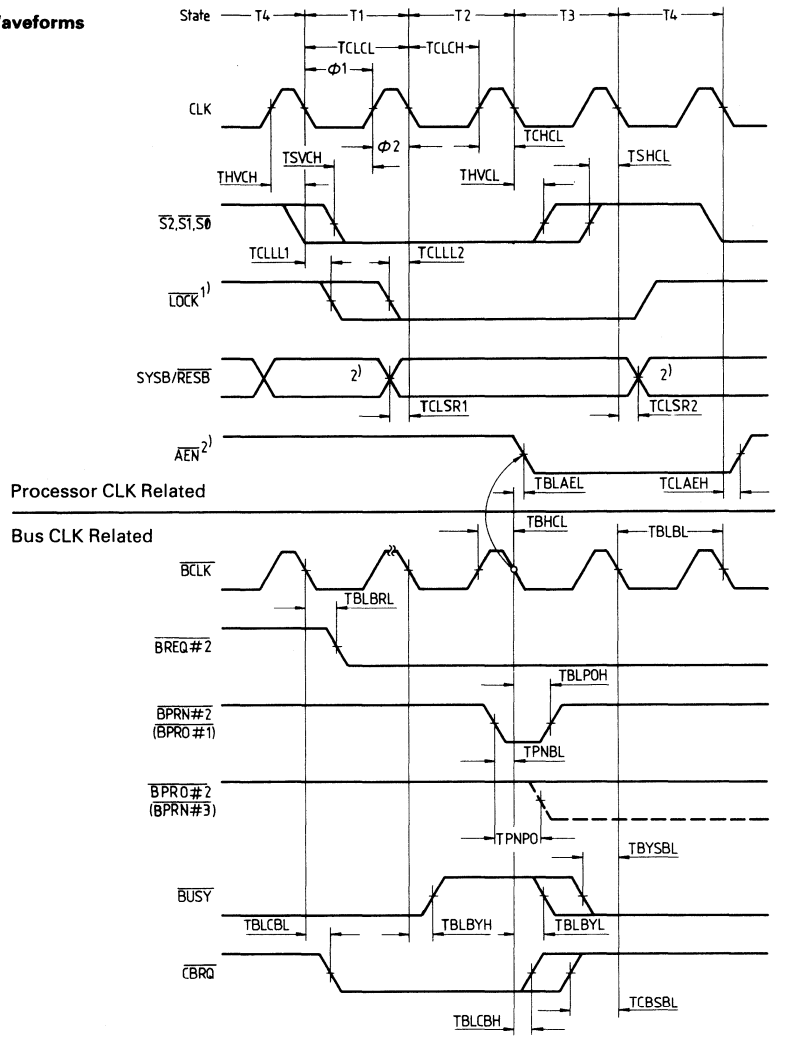
NOTES:
[1] BCLK generates the first BPRO wherein subsequent BPRO changes lower in the chain are generated through BPRON.
[2] Measured at 0.5 V above GND.

A.C. Testing Load Circuit



C_L = 100 pF
C_L Includes jig capacitance

Waveforms



NOTES:

1. LOCK active can occur during any state, as long as the relationships shown above with respect to the CLK are maintained.
   LOCK inactive has not critical time and can be asynchronous.
   CRQLCK has no critical timing and is considered an asynchronous input signal.
2. Glitching of SYSB/RESB pin is permitted during this time. After I 2 of T1, and before I 1 of T4, SYSB/RESB should be stable.
3. AEN leading edge is related to BCLK, trailing edge to CLK. The trailing edge of AEN occurs after bus priority is lost.

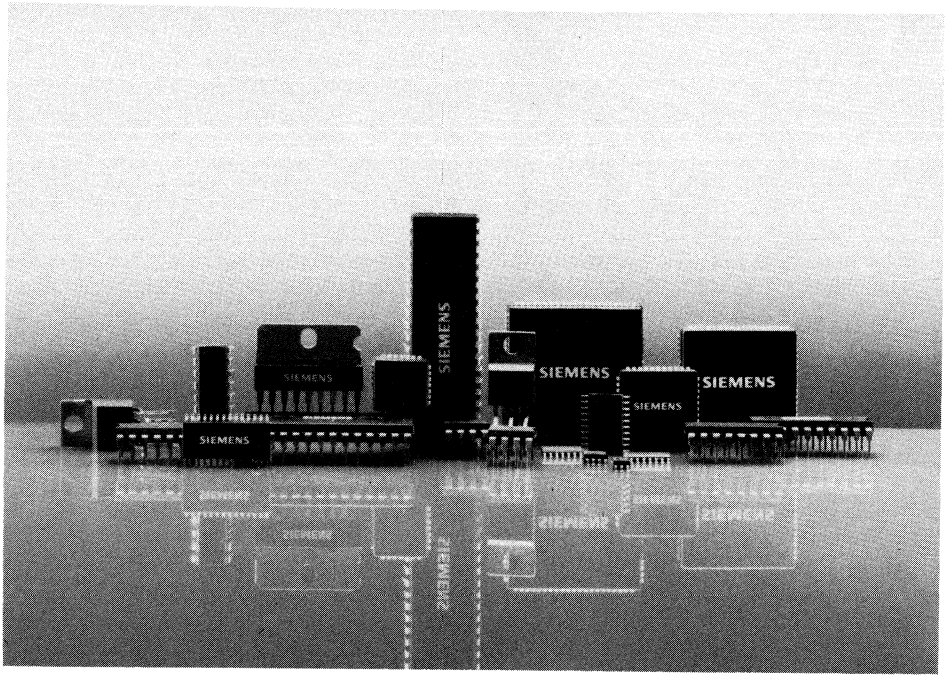
Additional Notes:
The signals related to CLK are typical processor signals, and do not relate to the depicted sequence of events of the signals referenced to $\overline{BCLK}$. The signals shown related to the $\overline{BCLK}$ represent a hypothetical sequence of events for illustration. Assume 3 bus arbiters of priorities 1, 2 and 3 configured in serial priority resolving scheme as shown in Figure 6. Assume arbiter has the bus and is holding busy low. Arbiter #2 detects its processor wants the bus and pulls low $\overline{BREQ\#2}$. If $\overline{BPRN\#2}$ is high (as shown), arbiter #2 will pull low $\overline{CBRQ}$ line. $\overline{CBRQ}$ signals to the higher priority arbiter #1 that a lower priority arbiter wants the bus. [A higher priority arbiter would be granted BPRN when it makes the bus request rather than having to wait for another arbiter to release the bus through $\overline{CBRQ}$].** Arbiter #1 will relinquish the multi-master system bus when it enters a state not requiring it (see Table 1), by lowering its $\overline{BPRO\#1}$ (tied to $\overline{BPRN\#2}$) and releasing BUSY. Arbiter #2 now sees that it has priority from $\overline{BPRN\#2}$ being low and releases CBRQ. As soon as BUSY signifies the bus is available (high), arbiter #2 pulls BUSY low on next falling edge of BCLK. Note that if arbiter #2 didn't want the bus at the time it received priority, it would pass priority to the next lower priority arbiter by lowering its $\overline{BPRO\#2}$ [TPNPO].

**Note that even a higher priority arbiter which is acquiring the bus through $\overline{BPRN}$ will momentarily drop $\overline{CBRQ}$ until it has acquired the bus.

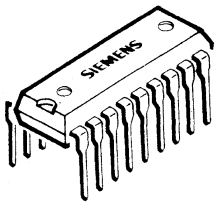
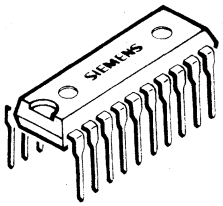
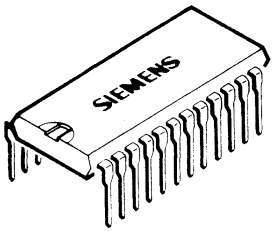
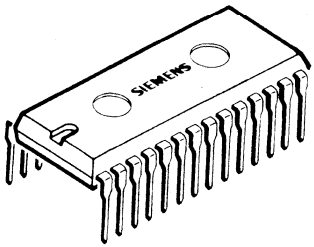
Ordering Information

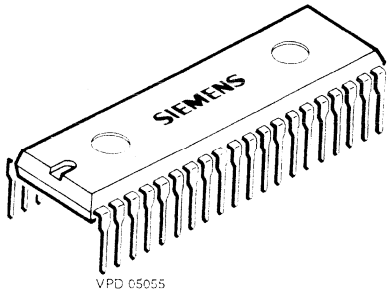
Component	Description	Ordering Code
SAB 8289-P	Bus-Arbiter 8 MHz (plastic)	Q67020-Y74
SAB 8289-1-P	Bus-Arbiter 10 MHz (plastic)	Q67120-Y85

Summary of Package Outlines



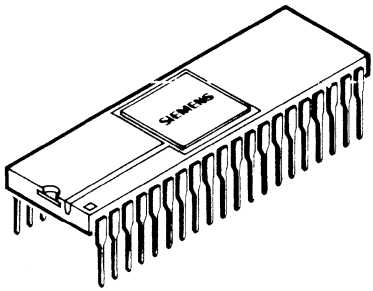
Package Outlines

 VPD 05035	**Plastic Package, P-DIP-18** (dual-in-line package) **20A18 DIN 41870 T9**
 VPD 05031	**Plastic Package, P-DIP-20** (dual-in-line package) **20A20 DIN 41870 T9**
 VRD 05034	**Plastic Package, P-DIP-24** (dual-in-line package) **20A24 DIN 41870 T10**
 VPD 05037	**Plastic Package, P-DIP-28** (dual-in-line package) **20A28 DIN 41870 T10**



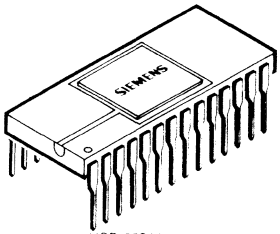
VPD 05055

Plastic Package, P-DIP-40
(dual-in-line package)
20A40 DIN 41870 T10



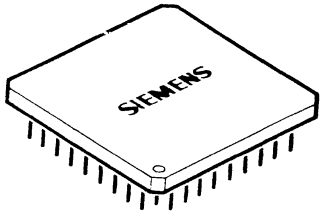
VCD 05087

Ceramic Package, C-DIP-40
(dual-in-line package)



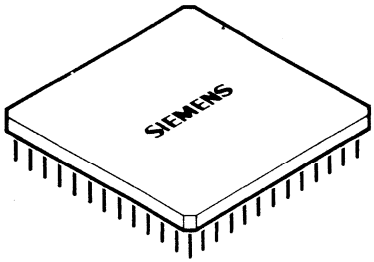
VCD 05044

Ceramic Package,C-DIP-28
(dual-in-line package)



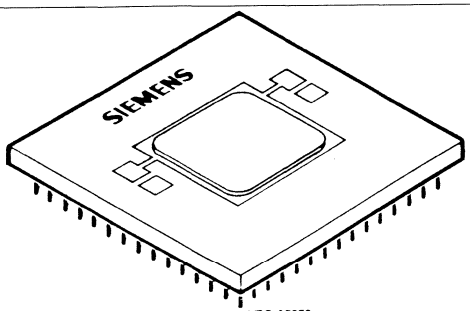
VPG 05003

Ceramic Package,C-PGA-68
(pin-grid-array)



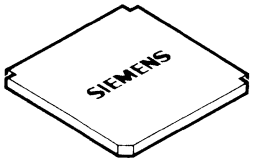
VPL 05002

Ceramic Package,C-PGA-88
(pin-grid-array)



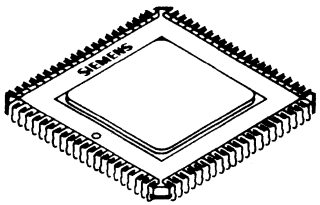
VPG 05058

Ceramic Package,C-PGA-145
(pin-grid-array)



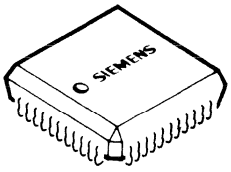
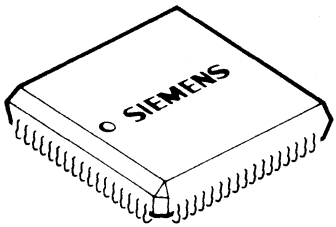
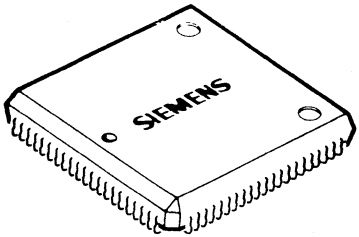
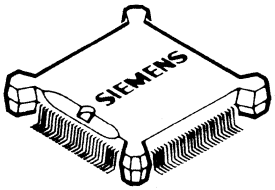
VCC 05027

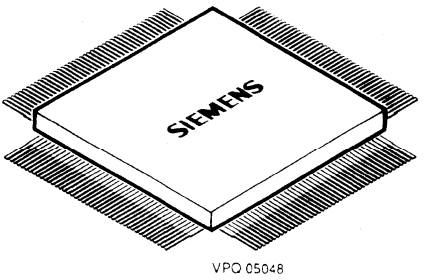
Ceramic Package,C-CC-68
(chip-carrier)



VPL 05126

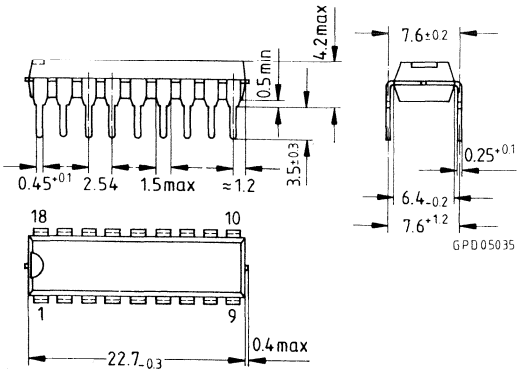
Ceramic Package,CL-CC-84
(chip-carrier)

 VPL 05102	**Plastic Package, PL-CC-44** (chip-carrier) – **SMD**
 VPL 05099	**Plastic Package, PL-CC-68** (chip-carrier) – **SMD**
 VPL 05029	**Plastic Package, PL-CC-84** (chip-carrier) – **SMD**
 VPQ 05032	**Plastic Package, P-QFP-100** (Quad-Flat-Pack) **-SMD**



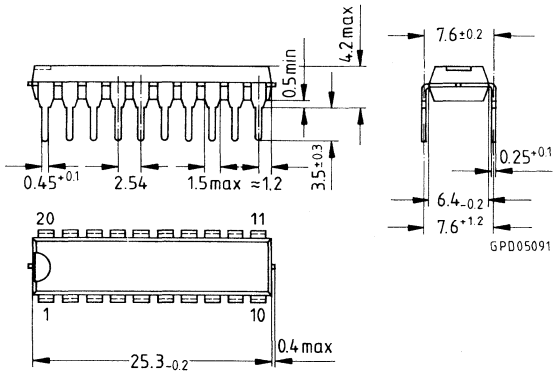
Ceramic Package,C-QFP-172
(Quad-Flat-Pack)

Plastic Package, P-DIP-18
(dual-in-line package)
20A18 DIN 41870 T9



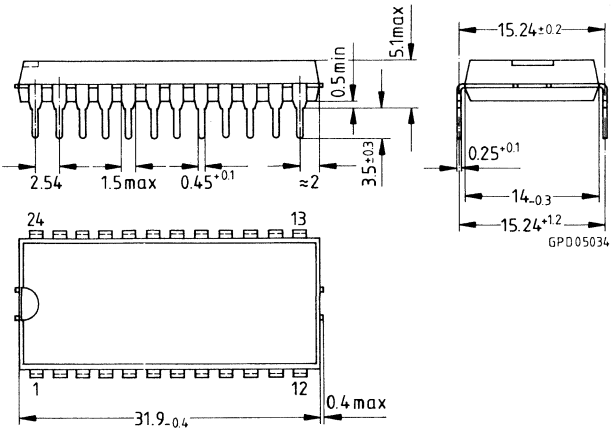
Dimensions in mm

Plastic Package, P-DIP-20
(dual-in-line package)
20A20 DIN 41870 T9



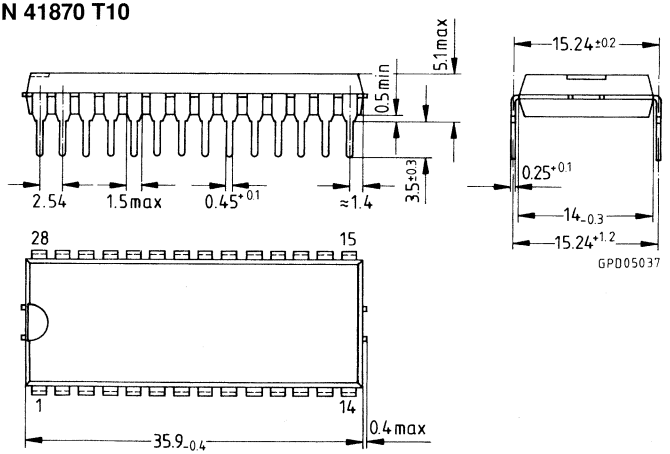
Dimensions in mm

Plastic Package, P-DIP-24
(dual-in-line package)
20A24 DIN 41870 T10



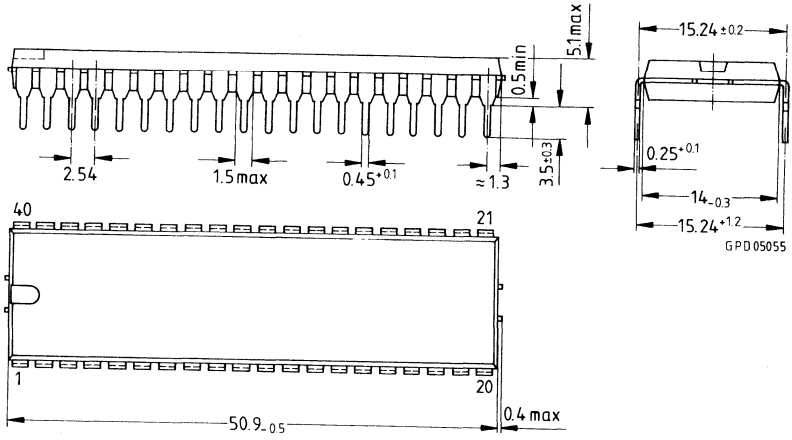
Dimensions in mm

Plastic Package, P-DIP-28
(dual-in-line package)
20A28 DIN 41870 T10



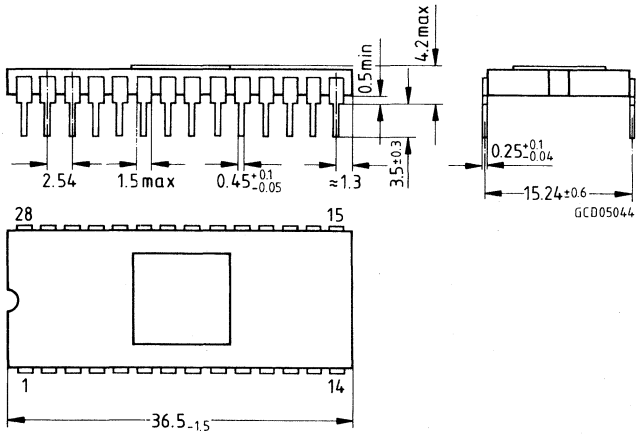
Dimensions in mm

Plastic Package, P-DIP-40
(dual-in-line package)
20A40 DIN 41870 T10



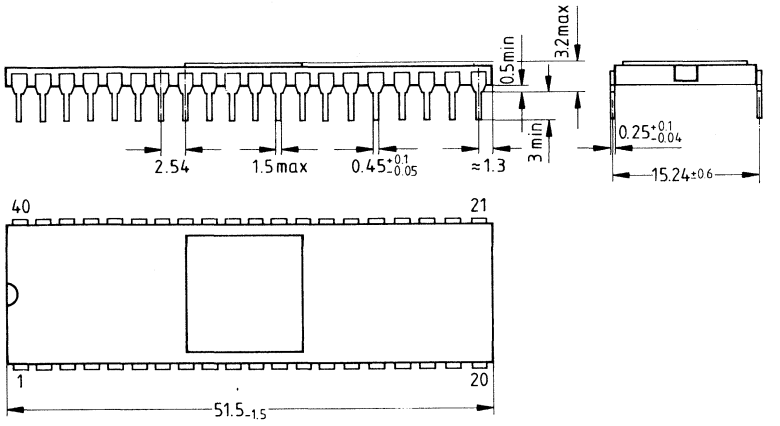
Dimensions in mm

Plastic Package, C-DIP-28
(dual-in-line package)



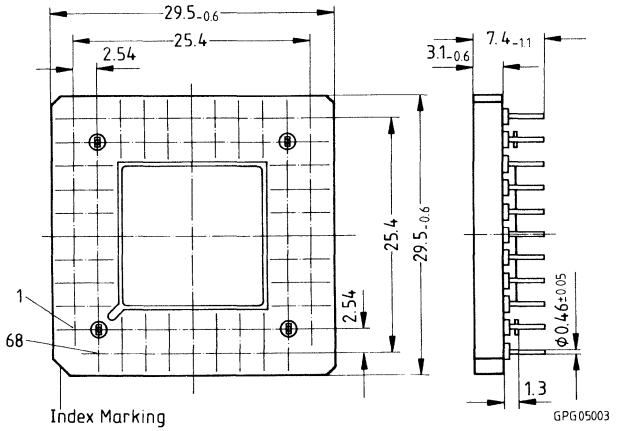
Dimensions in mm

Ceramic Package, C-DIP-40
(dual-in-line package)



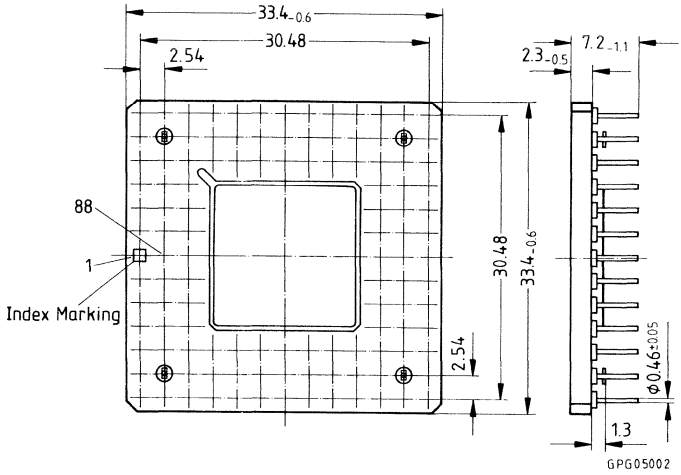
Dimensions in mm

Ceramic Package, C-PGA-68
(pin-grid-array)



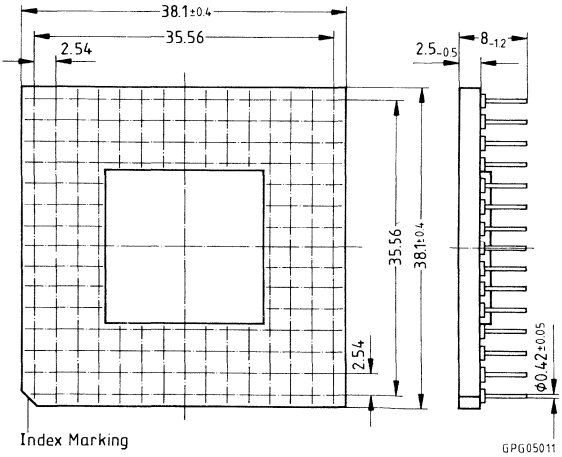
Dimensions in mm

Ceramic Package, C-PGA-88
(pin-grid-array)



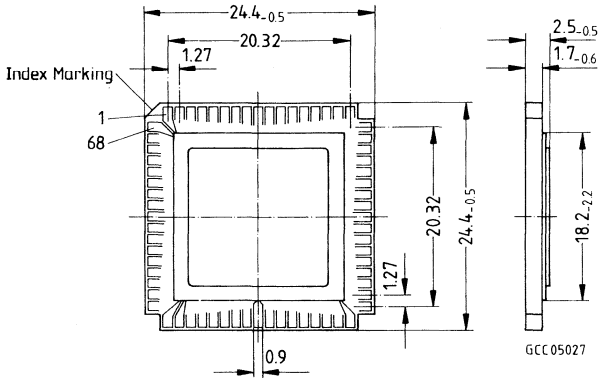
Dimensions in mm

Ceramic Package, C-PGA-145
(pin-grid-array)



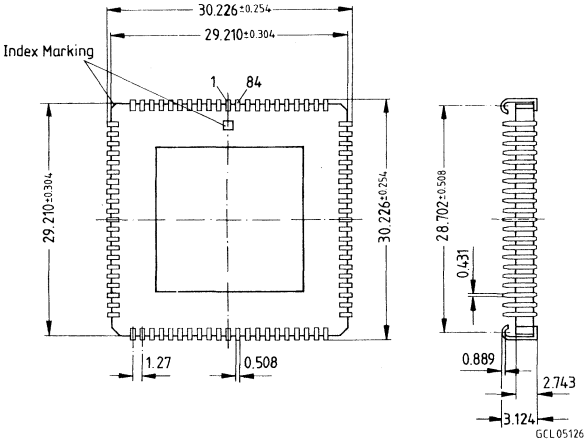
Dimensions in mm

Ceramic Package, C-CC-68
(chip-carrier)



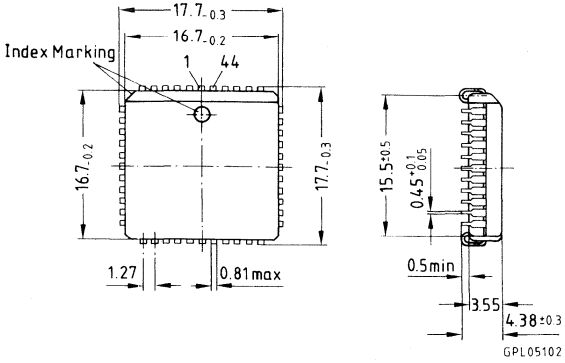
Dimensions in mm

Ceramic Package, CL-CC-84
(chip-carrier)



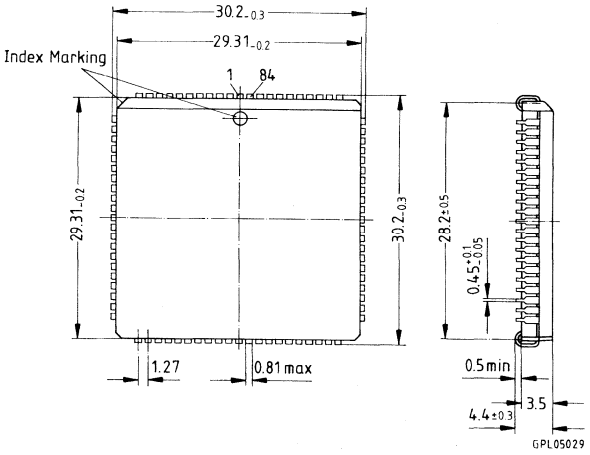
Dimensions in mm

Plastic Package, PL-CC-44
(chip-carrier) – **SMD**



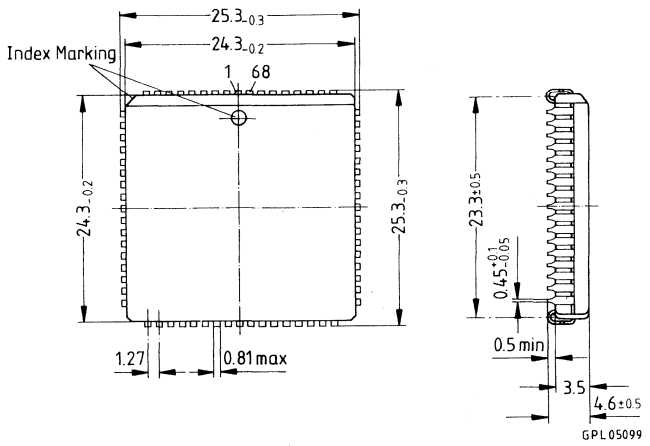
Dimensions in mm

Plastic Package, PL-CC-84
(chip-carrier) – **SMD**



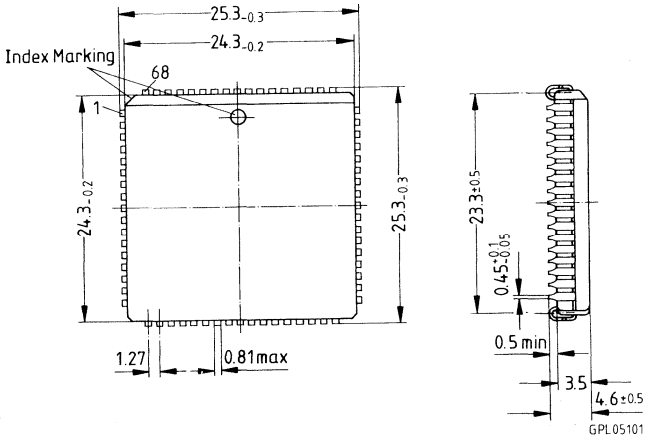
Dimensions in mm

Plastic Package, PL-CC-68
(chip-carrier) – **SMD**



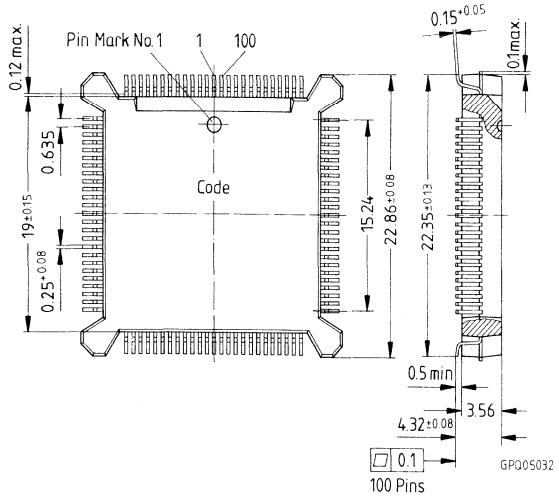
Dimensions in mm

Plastic Package, PL-CC-68
(chip-carrier) – **SMD**



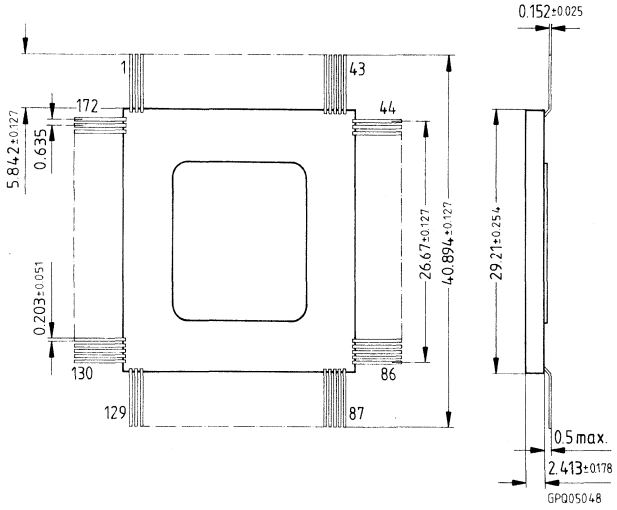
Dimensions in mm

Plastic Package, P-QFP -100
(Quad-Flat-Pack) **- SMD**



Dimensions in mm

Ceramic Package, C-QFP -172
(Quad-Flat-Pack)



Dimensions in mm

Semiconductor Group-Addresses
Information on Literature

Top Tech Semiconductors – Worldwide

(A)
Siemens AG Österreich
Postfach 326
1031 Wien
☎ (0222) 71711-5661
Tx 1372-10
FAX (0222) 71711-6110

(AUS)
Siemens Ltd., Head Office
544 Church Street
Richmond (Melbourne), Vic. 3121
☎ (03) 4207111, Tx 30425
FAX (03) 4207275

(B)
Siemens S.A.
chaussée de Charleroi 116
1060 Bruxelles
☎ (02) 536-2111, Tx 21347
FAX (02) 536-2492

(BR)
ICOTRON S.A.
Indústria de Componentes
Eletrônicos
Avenida Mutinga, 3650
05150 São Paulo-SP
☎ (011) 833-2211
Tx 11-81006
FAX (011) 833-2631

(CDN)
Siemens Electric Limited
Electronic Components Division
1180 Courtney Park Drive
Mississauga, Ontario L5T 1P2
☎ (416) 5641995
Tx (069) 68841
FAX (416) 5645855

(CH)
Siemens-Albis AG
Freilagerstraße 28
8047 Zürich
☎ (01) 495-3111, Tx 823781-23
FAX (01) 495-5050

(D)
Siemens AG
Salzufer 6–8
1000 Berlin 10
☎ (030) 3939-1
Tx 18100-278
FAX (030) 3939-2630
Ttx 308190 = sieznb

Siemens AG
Lahnweg 10
Postfach 1115
4000 Düsseldorf 1
☎ (0211) 399-0
Ttx 21134401
FAX (0211) 399-2928

Siemens AG
Rödelheimer Landstraße 5–9
Postfach 111733
6000 Frankfurt 1
☎ (069) 797-0
Tx 414131-0
FAX (069) 797-2253

Siemens AG
Lindenplatz 2
Postfach 105609
2000 Hamburg 1
☎ (040) 2889-0
Tx 215584-0
FAX (040) 2889-3096

Siemens AG **Hannover**
Hildesheimer Str. 7
Postfach 110551
3014 Laatzen
☎ (0511) 877-0
Tx 922333
FAX (0511) 877-2078

Siemens AG
Richard-Strauss-Straße 76
Postfach 202109
8000 München 80
☎ (089) 9221-4380
Tx 529421-19
FAX (089) 9221-4390, 4692
Ttx 8985084

Siemens AG
Von-der-Tann-Straße 30
Postfach 4844
8500 Nürnberg 1
☎ (0911) 654-0
Tx 622251
FAX (0911) 654-4064

Siemens AG
Geschwister-Scholl-Straße 24
Postfach 106026
7000 Stuttgart 1
☎ (0711) 2076-0
Tx 723941-50
FAX (0711) 2076-2448

(DK)
Siemens A/S
Borupvang 3
2750 Ballerup
☎ (42) 656565, Tx 35313
FAX (42) 656667

(E)
Siemens S.A.
Departamento de Componentes
Orense, 2
Apartado 155
28080 Madrid
☎ (01) 5552500, Tx 44191
FAX (01) 5565408

(F)
Siemens S.A.
39/47, Bd. Ornano
93527 Saint-Denis CEDEX 2
☎ (1) 49223100, Tx 234077
FAX (1) 49223970

(GB)
Siemens plc
Siemens House
Windmill Road
Sunbury on Thames
Middlesex TW16 7HS
☎ (0932) 785691, Tx 8951091
FAX (0932) 752632

(GR)
Siemens AE
Paradissou & Artemidos
P.O.B. 61011
15110 Amaroussio
☎ (01) 6864111, Tx 216292
FAX (01) 6864299

(HK)
Schmidt & Co. (H.K.) Ltd.
18/Fl., Great Eagle Centre
93 Harbour Road
Wonchai
Hong Kong
☎852/8330222
Tx 74766 schmc hx
FAX 8382652

(I)
Siemens S.p.A.
Div. Componenti, Impiante per
la Grafica e il Segnalamento
Via Fabio Filzi, 25/A
Casella Postale 10388
20100 Milano
☎ (02) 6766-1, Tx 330261
FAX (02) 6766-4295

(IND)
Siemens Ltd.
Head Office
134-A, Dr. Annie Besant Road, Worli
P.O.B. 6597
Bombay 400018
☎ 4938786, Tx 1175142
FAX (022) 4940240

(IRL)
Siemens Ltd.
Unit 8-11 Slaney Road
Dublin Industrial Estate
Finglas Road
Dublin 11
☎ (01) 302855, Tx 32547
FAX (01) 303151

(J)
Fuji Electronic Components Ltd.
New Yurakucho Bldg., 8F
12-1 Yurakucho 1-Chome,
Chiyoda-ku
Tokyo 100
☎ (03) 201-2401, Tx 32182
FAX (03) 201-6809

(N)
Siemens A/S
Østre Aker vei 90
Postboks 10, Veitvet
0518 Oslo 5
☎ (02) 633000, Tx 78477
FAX (02) 633805

(NL)
Siemens Nederland N.V.
Postb. 16068
2500 BB Den Haag
☎ (070) 3333333, Tx 31373
FAX (070) 3332782

(P)
Siemens S.A.
Estrada Nacional 117, Km 2,6
Alfragide
2700 Amadora
☎ (01) 4183311, Tx 62955
FAX (01) 4182967

(RA)
Siemens S.A.
Avenida Pte. Julio A. Roca 516
Casilla Correo Central 1232
1067 Buenos Aires
☎ (01) 300411, Tx 21812
FAX (01) 3319997

(RC)
Tai Engineering Co., Ltd.
6th Fl., Central Building
108, Chung Shan North Road, Sec. 2
P.O. Box 68-1882
Taipei
☎ (02) 5234700
Tx 27860 taiengco
FAX (02) 5367070

(ROK)
Siemens Ltd.
P.O.Box 3001
Seoul
☎ (02) 275-6111
Tx 23229
FAX (02) 2752170

(S)
Siemens AB
Hälsingegatan 40
Box 23141
10435 Stockholm
☎ (08) 7281000, Tx 11672
FAX (08) 7281493

(SF)
Siemens Osakeyhtiö
P.O.B 60
02601 ESPOO
☎ (9) 0 51051, Tx 124465
FAX (9) 0 51052398

(SGP)
Siemens Components Pte. Ltd.
Promotion Office
Blk 47 Ayer Rajah Crescent No. 06-12
Singapore 0513
☎ 7760044, Tx RS 21000
FAX 7770813, 7754504

(TR)
SIMKO Ticaret ve Sanayi A.S.
Meclisi Mebusan Cad. No. 125
80040 Findikli
P.K. 1001, 80007 Karaköy
80040 Findikli
☎ (01) 1510900
Tx 24233 sies tr
FAX (01) 1524134

(USA)
Integrated Circuits;
ASIC Products;
Power Semiconductors:
Siemens Components, Inc.
Integrated Circuits Division
2191 Laurelwood Road
Santa Clara, CA 95054-1514
☎ (408) 980-4500
Tx 989791
FAX (408) 980-4596

Optoelectronics:
Siemens Components, Inc.
Optoelectronics Division
19000 Homestead Road
Cupertino, CA 95014
☎ (408) 725-7910
Tx 352084 sie lit opto
FAX (408) 725-3439

Discrete Semiconductors:
Siemens Components, Inc.
Special Products Division
186 Wood Avenue South
Iselin, NJ 08830
☎ (201) 906-4300
Tx 844491 sie isln a
FAX (201) 632-2830

(ZA)
Siemens Limited
Siemens House,
P.O.B. 4583
Johannesburg 2000
☎ (011) 407-4111, Tx 422524
FAX (011) 4075345

2/90

Literaturhinweis
Information on Literature

Titel/Title	Bestell-Nr./Ordering No.	DM
Datenkataloge / Data Catalogs		
Microcontrollers	B158-B6213-X-X-7600	20,-
Microprocessors and Support Components	B158-B6256-X-X-7600	20,-
PC Peripherals and System Components	B158-B6255-X-X-7600	20,-
Memory Components	B158-B6290-X-X-7600	15,-
Benutzer-Handbücher / User's Manuals		
SAB 80512/80532, 8-bit Microcontroller	B2-B3808-X-X-7600	15,-
SAB 80515/80535 – SAB 80C515/80C535 (CMOS), 8-bit Microcontroller	B158-H6367-X-X-7600	15,-
SAB 80C517/80C537, CMOS, 8-bit Microcontroller	B258-B6075-X-X-7600	15,-
SAB 80C166/83C166 – 16-bit CMOS Single-Chip Microcontroller	B158-B6247-X-X-7600	20,-
Addendum to User's Manual SAB 80C166/83C166	B158-H6345-X-X-7600	–
SAB 8256A – MUART, Programmierbarer Multifunktionsbaustein	B2-B2494	10,-
SAB 8256A – UART, Programmable Multifunction Controller	B2-B2494-X-X-7600	10,-
SAB 82257 – High Performance DMA Controller for 16-bit Microcomputer Systems	B2-B3486-X-X-7600	15,-
SAB 82258A/SAB 82C258A – ADMA, Advanced DMA Controller for 16-/32-bit Microcomputer Systems	B158-B6305-X-X-7600	15,-
Produktschriften / Product Information		
SAB 80512 – Ein-Chip Mikrocontroller	B2-B3693	–
SAB 80512 – Single-Chip Microcontroller	B2-B3693-X-X-7600	–
SAB 80515 – Ein-Chip Mikrocontroller	B2-B3340	–
SAB 80515 – Single-Chip Microcontroller	B2-B3340-X-X-7600	–
SAB 8051x – 8-bit Mikrocontroller-Familie mit den neuesten Familienmitgliedern SAB 80C515, 80C517	B258-B6150	–
The SAB 8051x – 8-bit Microcontroller Family with its new Members SAB 80C515, 80C517	B258-B6150-X-X-7600	–
SAB 80C166/83C166 High-Performance16-bit CMOS Single-Chip Microcontroller	B158-B6227-X-X-7600	–
SAB 82258A/SAB 82C258A – ADMA Advanced DMA Controller for 16-/32-bit Microcomputer Systems	B158-B6274-X-X-7600	–
SAB 82511 – TBM, Token Bus Modem	B2-B3796-X-X-7600	–
Multifunktionscontroller (SAB 82556) für serielle Schnittstellen	B258-B6166	–
Themenschriften / Special-Subject Brochures		
Der Mega-Chip	B9-B3971	–
The Mega-Chip	B9-B3971-X-X-7600	–
MEGA Perfektion – Qualität in Siemens 1-Mbit DRAMs	B166-B6286-V1	–
MEGA Perfection – Quality in Siemens 1-Mbit DRAMs	B166-B6286-V1-X-7600	–
RISC-Prozessoren	B158-B6142-V1	–
RISC Processoren	B158-B6142-V1-X-7600	–
Von CISC zu RISC: Leistungsexplosion bei Mikroprozessoren	B158-H6355	–
SAB 8048/8049 – Befehlsliste	B/2516	–
Microcontroller Family SAB 8051 – Pocket Guide	B158-B6229-X-X-7600	2,50

Literaturhinweis
Information on Literature

Titel/Title	Bestell-Nr./Ordering No.	DM
Themenschriften (Forts.) / **Special-Subject Brochures** (cont'd)		
Externer Speicherzugriff mit acht Pointern (mit SAB 80C517 - Sonderdruck)	B258-B6135	–
EPC 535/532 – Experimental Kit for the Microcontroller SAB 80515/80535	B258-B6107-X-X-7600	–
Peripheriebausteine integriert – Detailapplikationen zum SAB 80515	B2-B3677	–
Die 8051 – Mikrocontroller-Familie Hardware- und Softwareeigenschaften; Entwicklungsunterstützung; Applikationsbeispiele und -programme; Spezifikationen	A19100-L531-F186	39,-
Applikationen zur 8051 – Mikrocontroller-Familie Anwendungen der Hardware-Komponenten	A19100-L531-F228	39,-
SAB 80C166/83C166 16-bit CMOS Ein-Chip Mikrocontroller (Kurz Information)	B158-B6186	–
SAB 80C166/83C166 16-bit CMOS Single-Chip Microcontroller (Short Information)	B158-B6186-X-X-7600	–
SAB 80C166 – auf Schnelligkeit getrimmt	B158-B6206	–
Token Bus Carrierband Modem PLL-Clockgenerator for two Marginally spaced Frequencies	B2-B3948-X-X-7600	–
SAB 82556 – USIC, Universal System Interface Controller	B158-B6216-X-X-7600	–
Lieferprogramm / Short-Form Catalog		
Integrierte Schaltungen, Speicher- und Mikrocomputer-Bausteine Integrated Circuits, Memory and Microcomputer Components	B192-B6337-X-X-7400	–

Hinweis für Ihre Druckschriften-Bestellung

Richten Sie bitte Ihre Bestellung an den Ihnen nächstgelegenen Siemens Bauteile-Vertrieb (siehe Anschriften).

Vergessen Sie bitte nicht, Ihre Adresse bzw. Lieferanschrift und die Druckschriften- Bestellnummer deutlich anzugeben.

Die Preise gelten für Bestellungen ab 01.10.1990 in DM ab Lieferort ausschließlich Mehrwertsteuer, Verpackung, Versand und Versicherung. Änderungen der angegebenen Preise behalten wir uns vor. Rechnungsstellung erfolgt nach Lieferung.

Betriebsangehörige bestellen bitte mit dem Bestellzettel H38-S2009 (Inland) bzw. H38-S2021 (Ausland). Bestellungen über DV-Bestellverfahren richten Sie bitte an den BZ-Empfänger G3876 in Fürth.

How to order Literature

Literature is available at your nearest Siemens Office, Semiconductor Group, or Distributor (see addresses).

Make sure that you have clearly stated your address and the ordering number(s) of the literature in question.

The prices are valid for orders as of October 1, 1990. They are quoted in DM ex place of shipment exclusive of VAT, packing, shipment, and insurance. The prices are subject to change without notice. Your literature order will be invoiced after delivery.

Siemens employees are requested to use the ordering form H38-S2021 or to order directly via DP to receiver no. G3876 in Fürth.

Sprachenschlüssel		
e	englisch	-X-X-7600
d/e	deutsch/englisch	-X-X-7400

Language Code		
e	English	-X-X-7600
d/e	German/English	-X-X-7400

Contents

Published by Semiconductor Group

Siemens Aktiengesellschaft

Ordering No. B158-B6256-X-X-7600
Printed in Germany
DB 109010.