SAB 80286

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

Preliminary

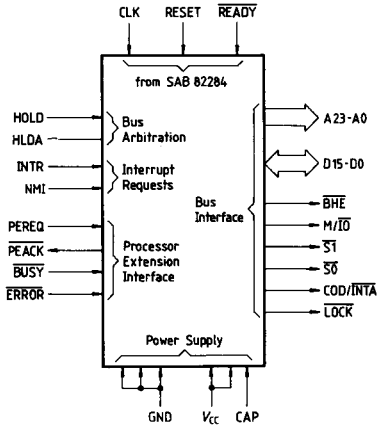
- High-performance processor (up to six times the SAB 8086)
- Large address space:
  - 16 megabytes physical
  - 1 gigabyte virtual per task
- High bandwidth bus interface (10 megabyte/s at 20 MHz system clock)
- Full hardware and software support
- Two SAB 8086 upward-compatible operating modes:
  - 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

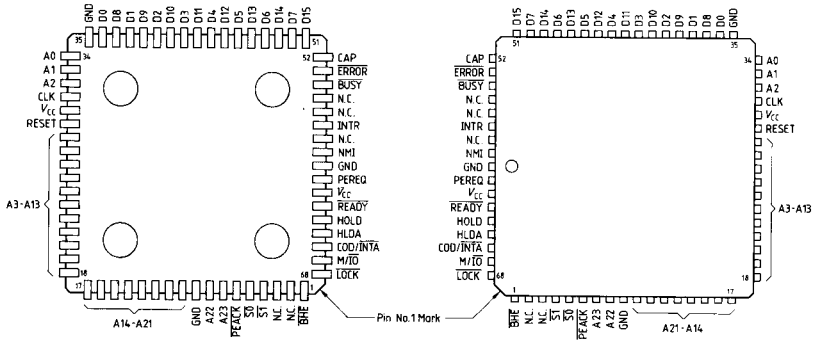
A23-A0	Address Bus
D15-D0	Data Bus
S1, S0	Bus Cycle Status
$\overline{BHE}$	Bus High Enable
M/$\overline{IO}$	Memory/IO Select
COD/$\overline{INTA}$	Code /Interrupt Acknowledge
$\overline{LOCK}$	Bus Lock
CLK	System Clock
RESET	System Reset
$\overline{READY}$	Bus Ready
CAP	Substrate Filter Capacitor
HOLD	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

<table>
<tr><th>Symbol</th><th>Pin</th><th>Input (I)
Output (O)</th><th>Function</th></tr>
<tr><td>$\overline{BHE}$</td><td>1</td><td>O</td><td>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.
<table>
<tr><th colspan="3">$\overline{BHE}$ and A0 encodings</th></tr>
<tr><th>$\overline{BHE}$ value</th><th>A0 value</th><th>Function</th></tr>
<tr><td>0</td><td>0</td><td>Word transfer</td></tr>
<tr><td>0</td><td>1</td><td>Byte transfer on upper half of data bus (D15-8)</td></tr>
<tr><td>1</td><td>0</td><td>Byte transfer on lower half of data bus (D7-0)</td></tr>
<tr><td>1</td><td>1</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>$\overline{S1}$, $\overline{S0}$</td><td>4, 5</td><td>O</td><td>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.
<table>
<tr><th colspan="5">Bus cycle status definition</th></tr>
<tr><th>COD/$\overline{INTA}$</th><th>M/$\overline{IO}$</th><th>$\overline{S1}$</th><th>$\overline{S0}$</th><th>Bus cycle initiated</th></tr>
<tr><td>0 (low)</td><td>0</td><td>0</td><td>0</td><td>Interrupt acknowledge</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>None; not a status cycle</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>IF A1 = 1 then halt; else shutdown</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>Memory data read</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>Memory data write</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>None; not a status cycle</td></tr>
<tr><td>1 (high)</td><td>0</td><td>0</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>I/O read</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>I/O write</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>None; not a status cycle</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>Memory instruction read</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>None; not a status cycle</td></tr>
</table></td></tr>
</table>

Pin Definitions and Functions (cont'd)

Symbol	Pin	Input (I) Output (O)	Function
A23–A0	7 – 34	O	**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.
RESET	29	I	**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: Pin state during reset Pin value: 1 (high) — Pin names: $\overline{S0}$, $\overline{S1}$, $\overline{PEACK}$, A23–A0, $\overline{BHE}$, $\overline{LOCK}$ Pin value: 0 (low) — Pin names: M/$\overline{IO}$, COD/$\overline{INTA}$, HLDA Pin value: Tristate off — Pin names: D15 – D0 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.
CLK	31	I	**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.
D15 – D0	36 – 51	I O	**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.
$\overline{BUSY}$ $\overline{ERROR}$	53, 54	I I	**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.

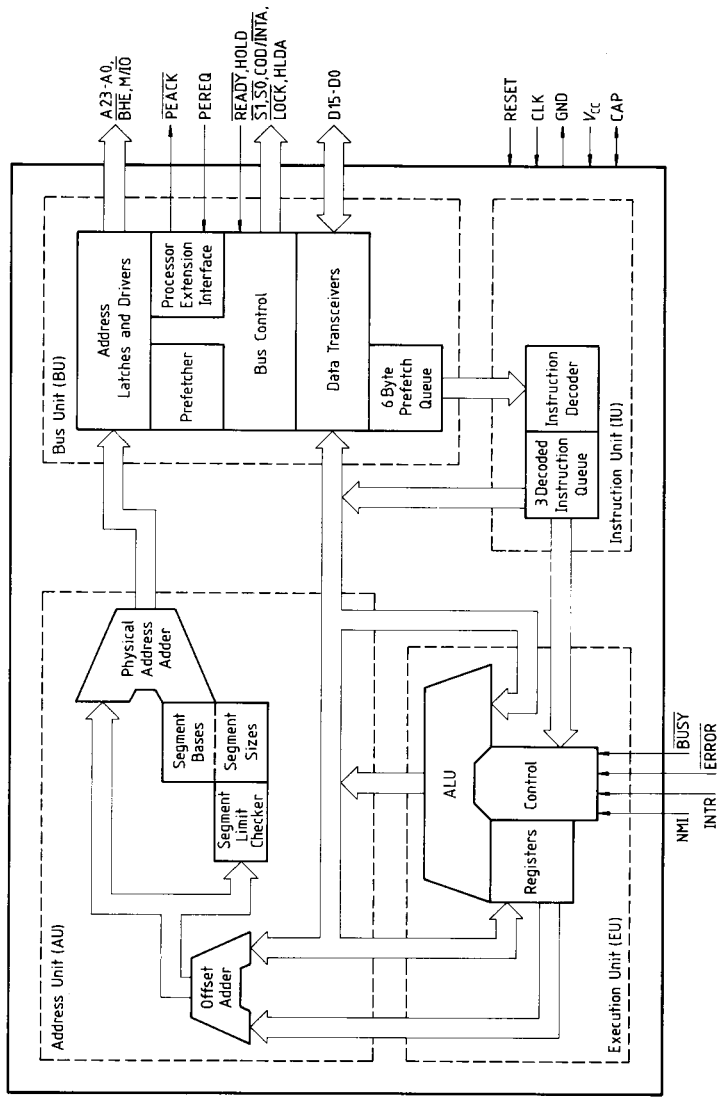
Pin Definitions and Functions (cont'd)

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/$\overline{INTA}$	66	O	**CODE/INTERRUPT ACKNOWLEDGE** distinguishes instruction fetch cycles from memory data read cycles. Also distinguishes interrupt acknowledge cycles from I/O cycles. COD/$\overline{INTA}$ floats to tristate off during bus hold acknowledge.
M/$\overline{IO}$	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/$\overline{IO}$ floats to tristate off during bus hold acknowledge.
$\overline{LOCK}$	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 $\overline{LOCK}$ 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. $\overline{LOCK}$ 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

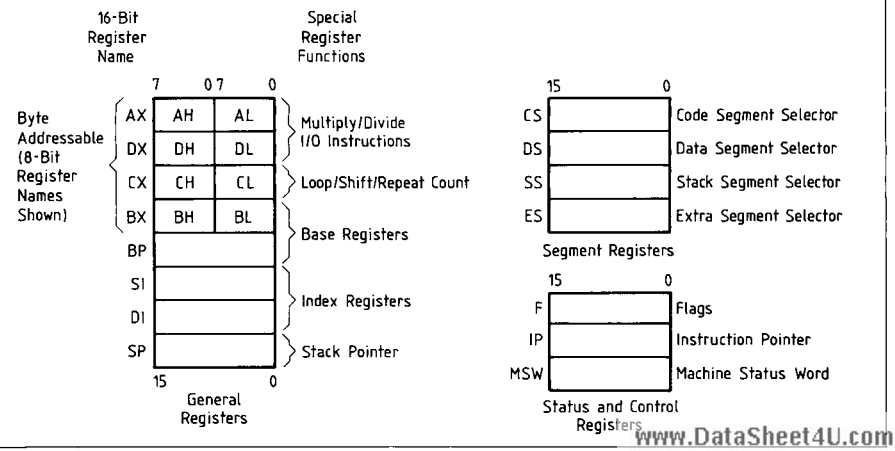
The processors of the Intel/Siemens 16-bit CPU family all contain the same basic set of registers, instructions, and addressing modes. Therefore, the SAB 80286 processor is upward-compatible with the SAB 8086, 8088 and 80186 CPUs.

Register Set

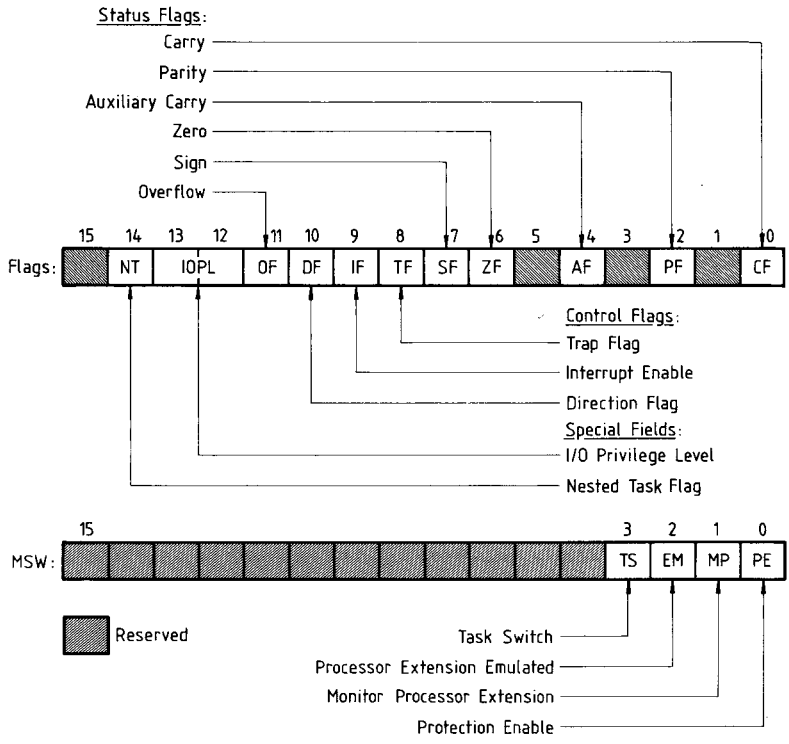
The SAB 80286 basic architecture has fifteen registers as shown below. These registers are grouped into the following four categories:

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

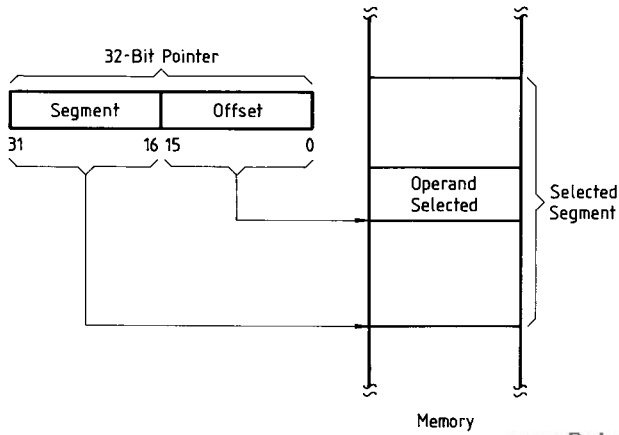


Table 2
Segment Register Selection Rules

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 conform to IEEE Std 754) 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

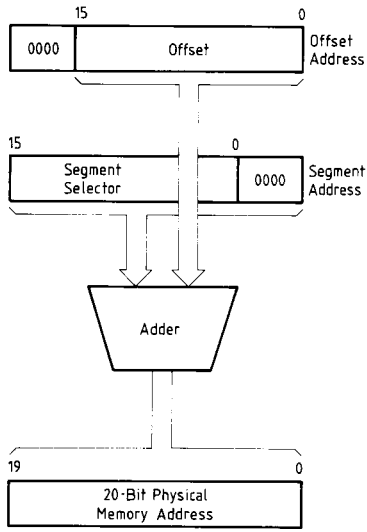


Table 8
Real Address Mode, Addressing Interrupts

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.

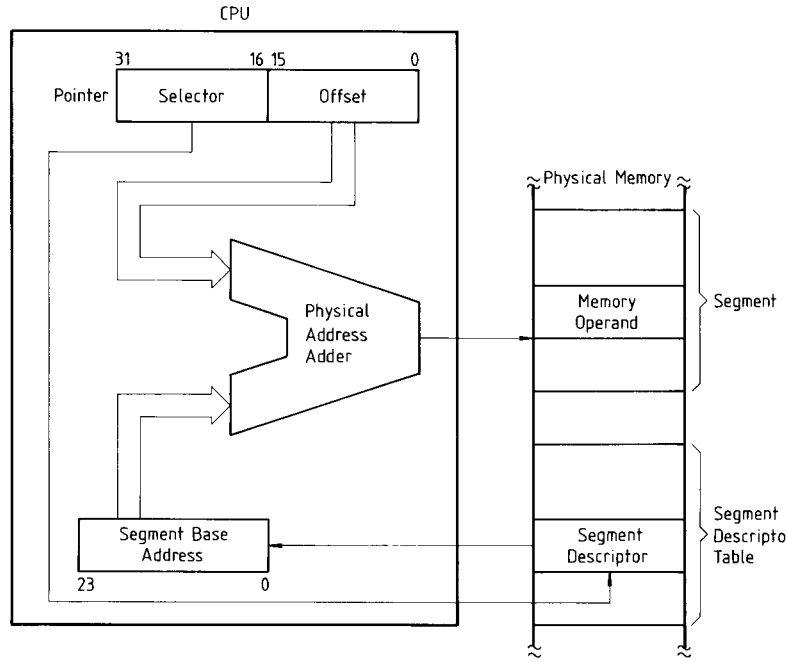
Memory Addressing

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

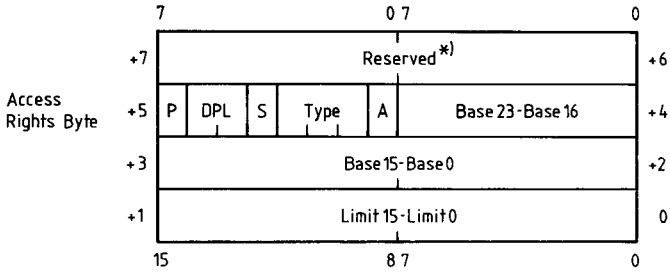


Code and data segment descriptors (S = 1)

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



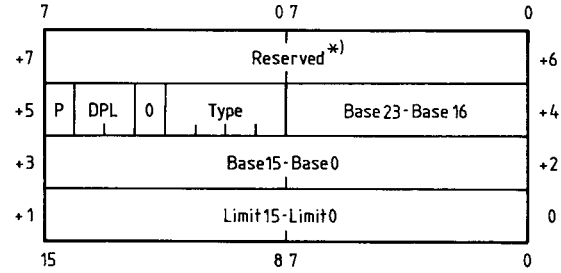
Access Rights Byte Definition

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

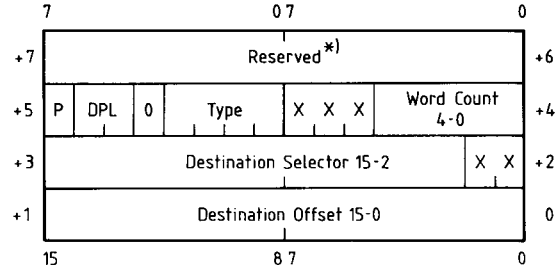
Name	Value	Description
TYPE	1 2 3	Available task state segment Local descriptor table descriptor Busy task state segment
P	0 1	Descriptor contents are not valid Descriptor contents are valid
DPL	0–3	Descriptor privilege level
BASE	24-bit number	Base address of special system data segment in real memory
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

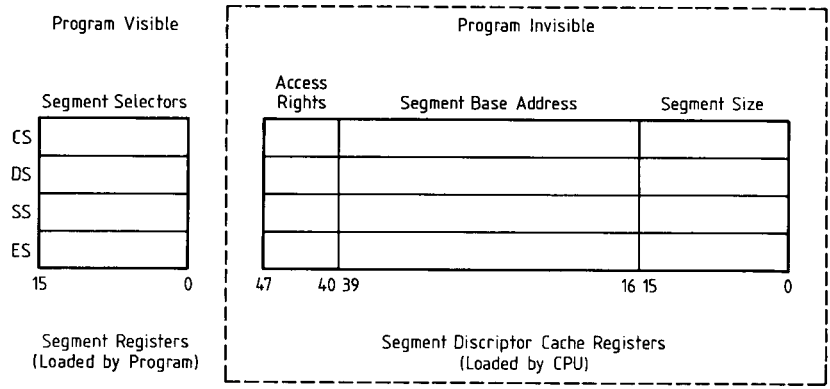
Segment descriptor cache registers

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.

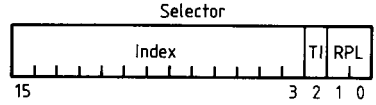
Selector fields

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

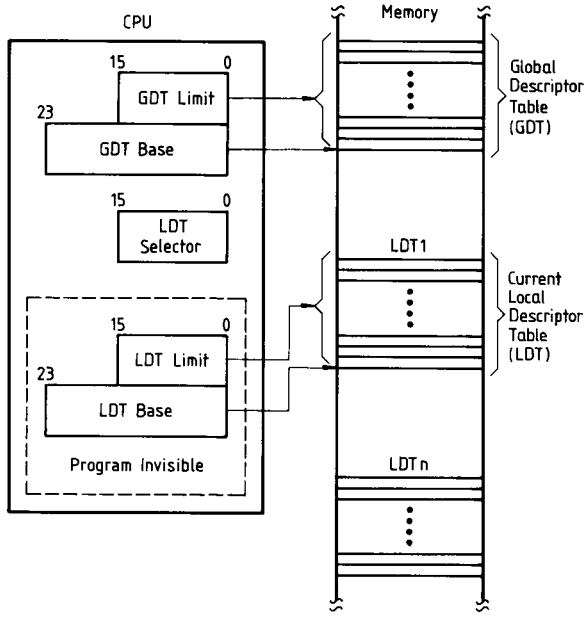


Bits	Name	Function
1-0	Requested Privilege Level (RPL)	Indicates Selector Privilege Level Desired
2	Table Indicator (TI)	TI = 0 Use Global Descriptor Table (GDT) TI = 1 Use Local Descriptor Table (LDT)
15-3	Index	Select Descriptor Entry in Table

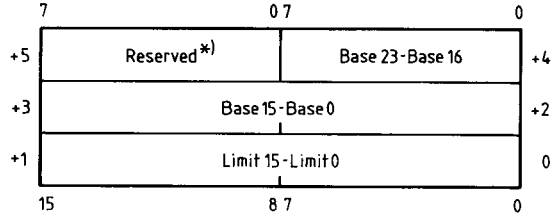
Local and global descriptor tables (LDT, GDT)

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

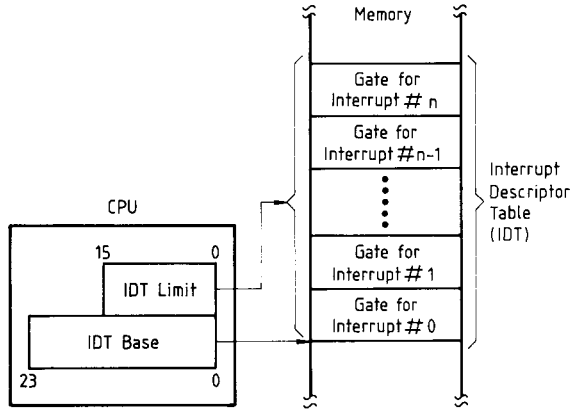


*) Must be set to 0 for compatibility with SAB 80386

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.

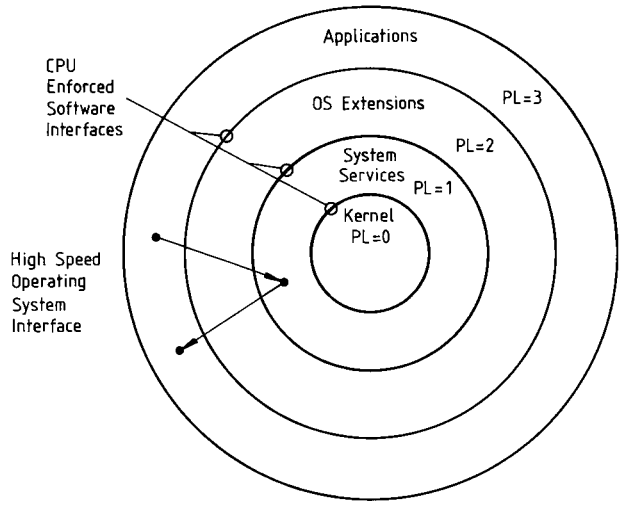
Interrupt Descriptor Table Definition



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

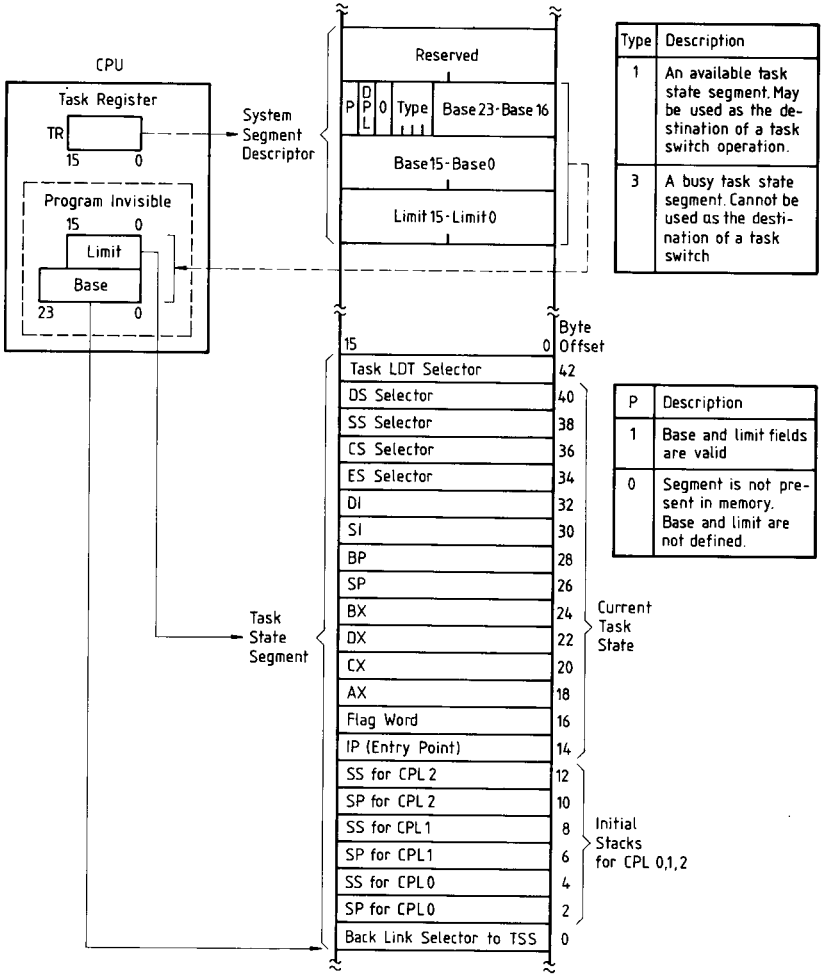
The context of a processor extension (such as the SAB 80287 numerics processor) is not changed by the task switch operation. A processor extension context need only be changed when a different task attempts to use the processor extension (which still contains the context of a previous task).

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.

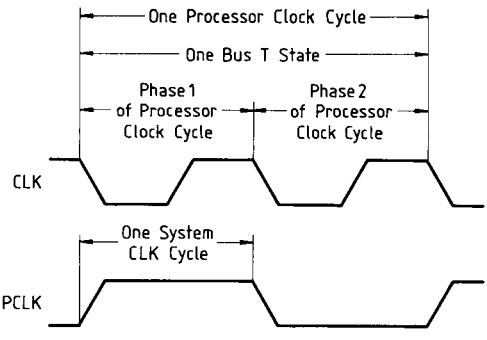
The SAB 82289 bus arbiter generates Multibus bus arbitration signals. These components can provide the timing and electrical power drive levels required for most system bus interfaces including the Multibus.

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

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.

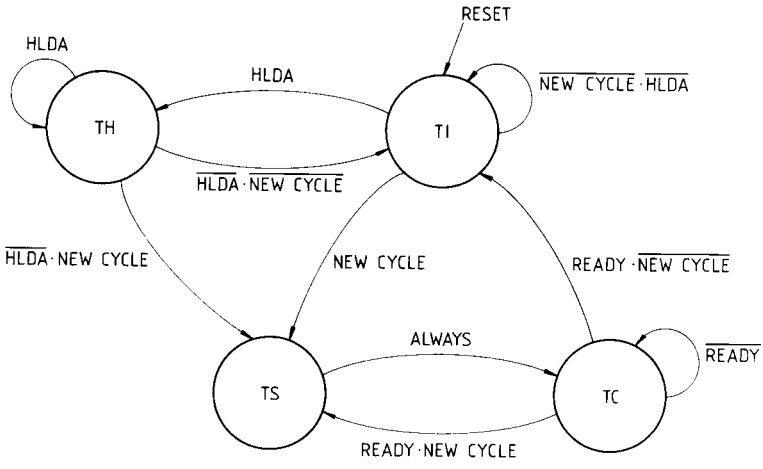
Each bus state is one processor clock long. The figure below shows the four SAB 80286 local bus states and allowed transitions.

Pipelined Addressing

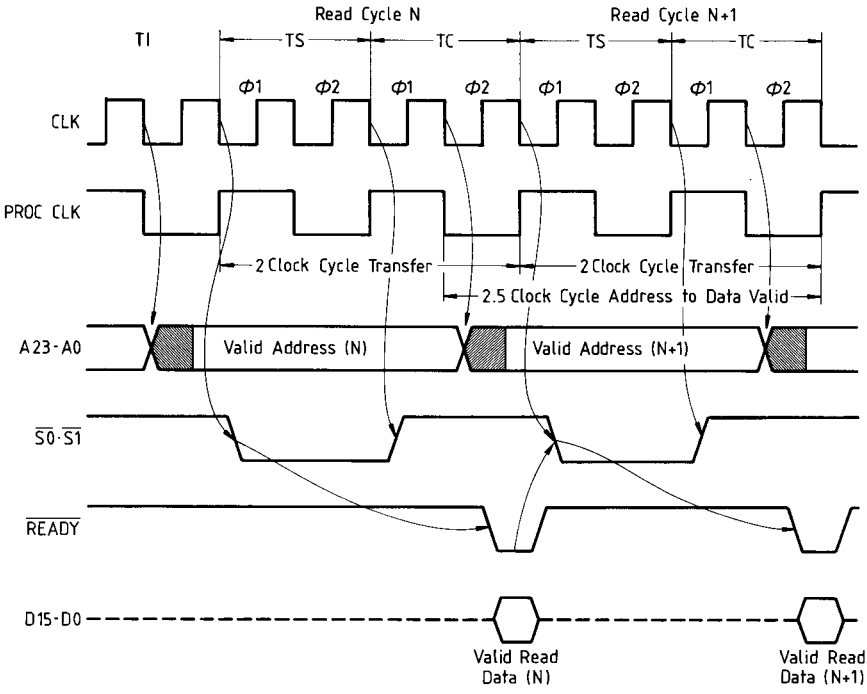
The SAB 80286 uses a local bus interface with pipelined timing to allow as much time as possible for data access. Pipelined timing allows bus operations to be performed in two processor cycles, while allowing each individual bus operation to last for three processor cycles. The timing of the address 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 $\overline{\text{READY}}$.

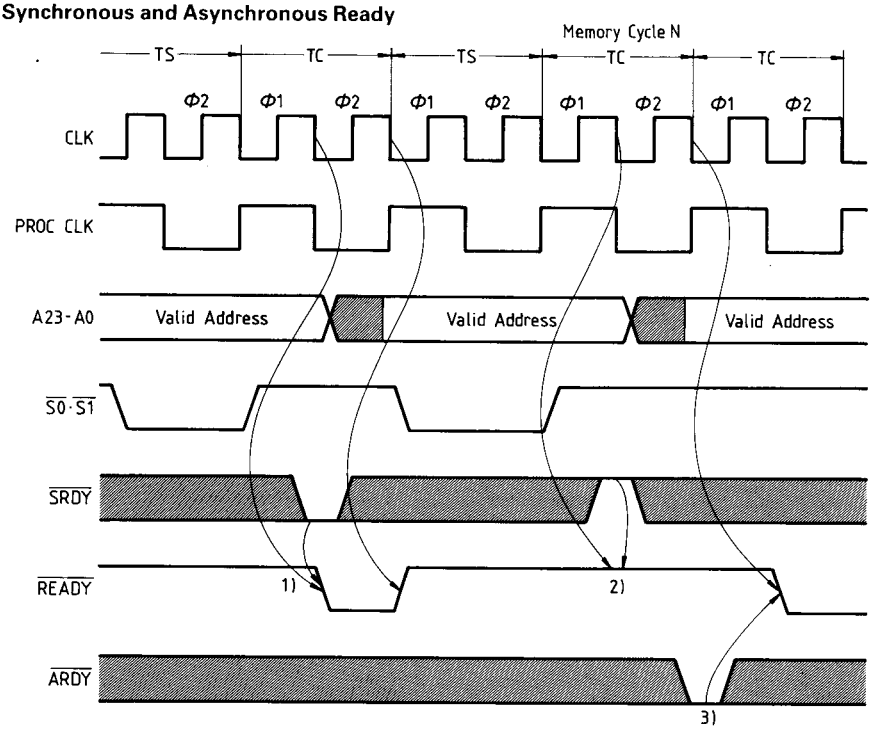
$\overline{\text{READY}}$ Operation

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

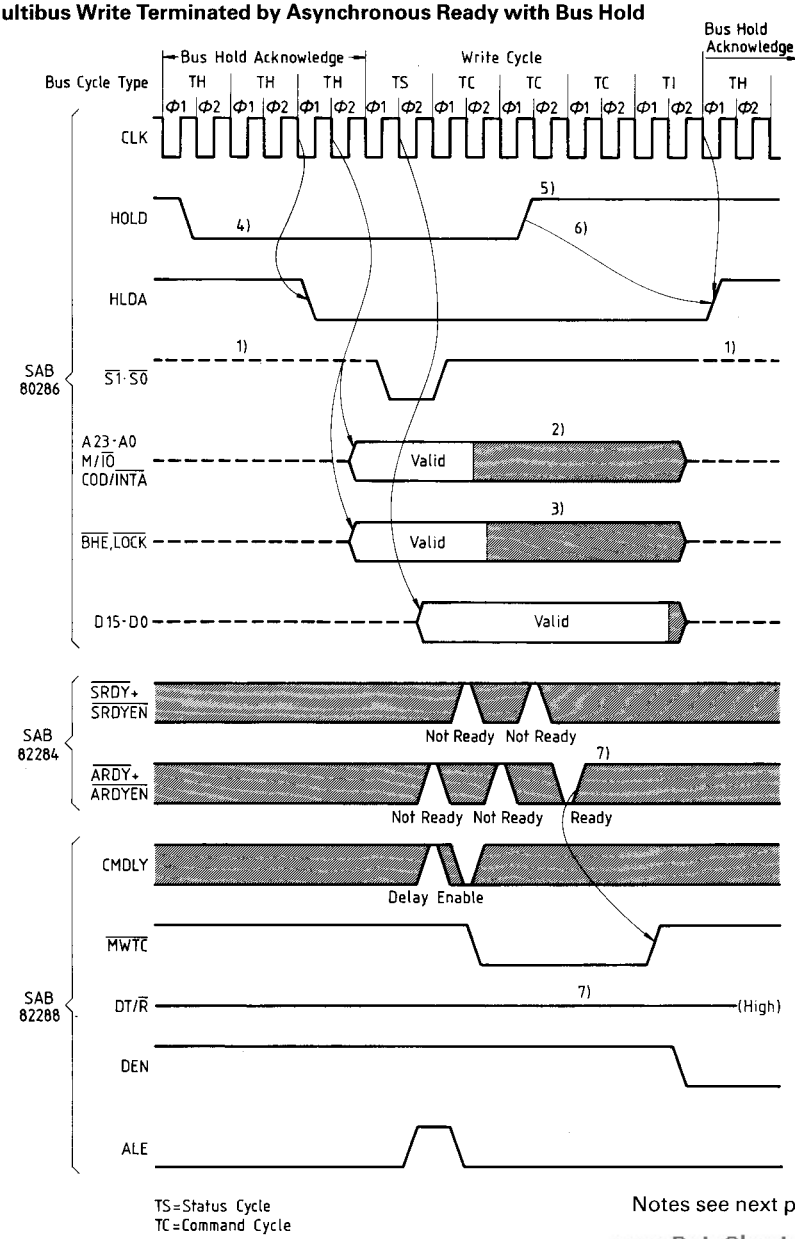
Asynchronous Ready

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

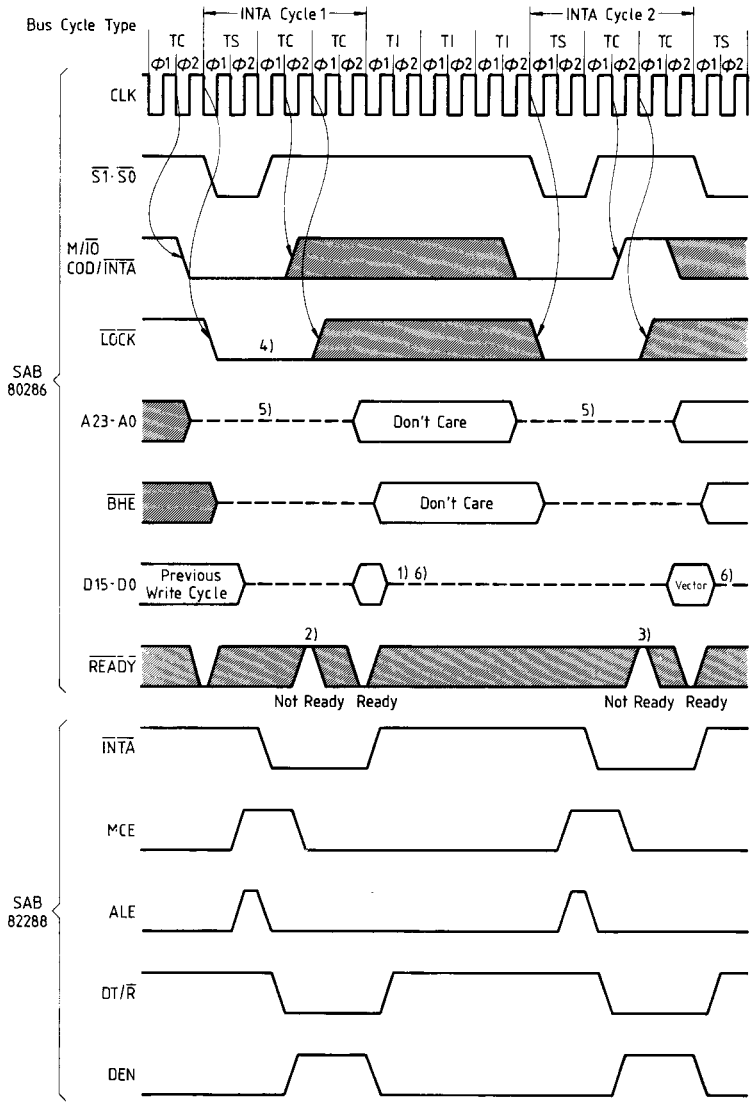
The processor extension interface uses I/O port addresses 00F8(H), 00FA(H), and 00FC(H) which are part of the I/O port address range reserved. An ESC instruction with EM = 0 and TS = 0 will perform I/O bus operations to one or more of these I/O port addresses independent of the value of IOPL and CPL.

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.

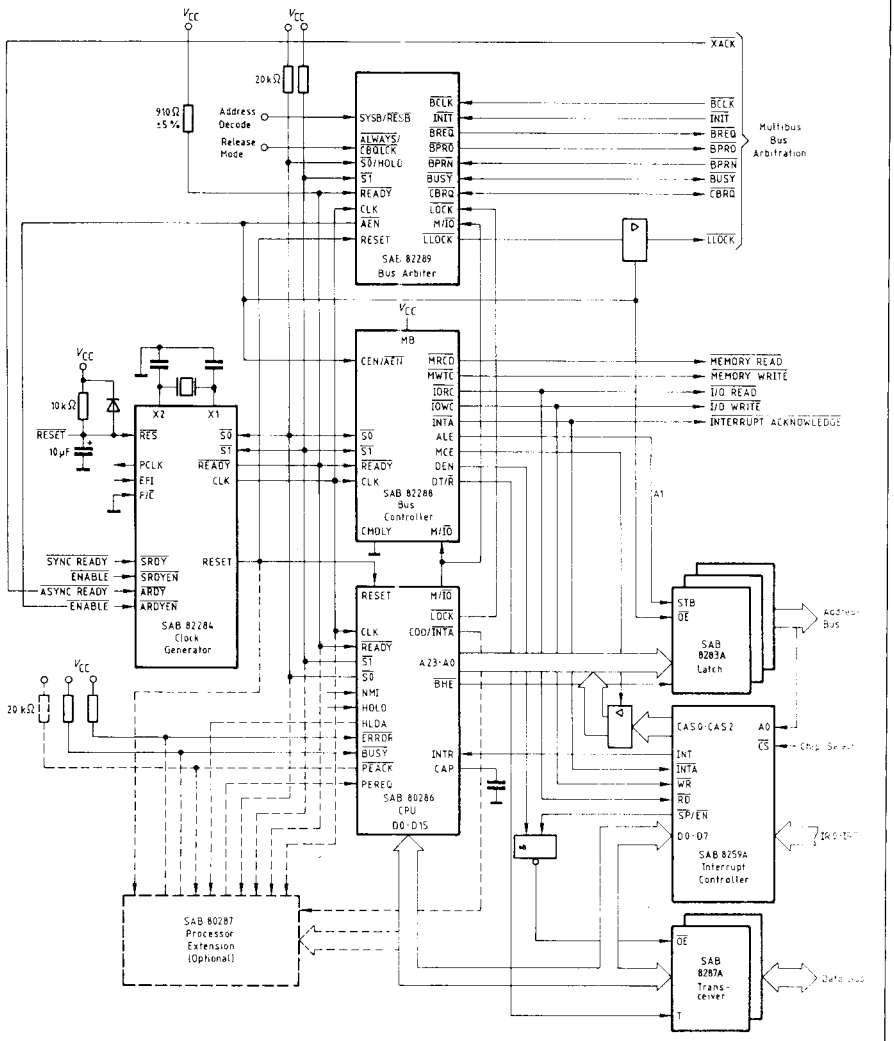
As indicated by the dashed lines in the figure above, the ability to add processor extensions is an integral feature of SAB 80286 microsystems. The processor extension interface allows external hardware to perform special functions and transfer data concurrently with CPU execution of other instructions. 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.

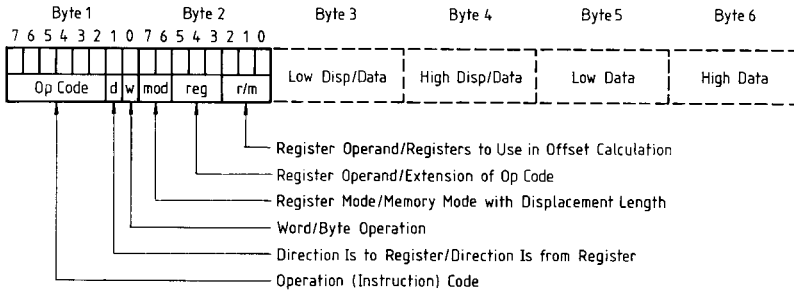
The SAB 80286 can overlap chip select decoding and address propagation during the data transfer for the previous bus operation. This information is latched into the SAB 8282A/8283A's by ALE in the middle of a TS cycle. The latched chip select and address information remains stable during the bus operation 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.

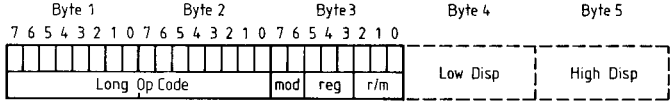
Basic SAB 80286 System Configuration



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 $\overline{LOCK}$ 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		
SBB = 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
Memory-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)	011010s1 | mod reg r/m | data | data if s = 0	21, 24 *	21, 24 *	2	9
DIV = Divide unsigned):	1111011w | mod 110 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):	1111011w | mod 111 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	11010100 | 00001010	16	16		
AAD = ASCII adjust for divide	11010101 | 00001010	14	14		
CBW = Convert byte to word	10011000	2	2		
CWD = Convert word to double word	10011001	2	2		
Logic					
Shift/rotate instructions:					
Register/memory by 1	1101000w | mod TTT r/m	2, 7 *	2, 7 *	2	9
Register/memory by CL	1101001w | mod TTT r/m	5+n, 8+n*	5+n, 8+n*	2	9
Register/memory by count	1100000w | mod TTT r/m | count	5+n, 8+n*	5+n, 8+n*	2	9

TTT	Instruction
000	ROL
001	ROR
010	RCL
011	RCR
100	SHL/SAL
101	SHR
111	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	001000dw | mod reg r/m	2, 7 *	2, 7 *	2	9
Immediate to register/memory	1000000w | mod 100 r/m | data | data if w = 1	3, 7 *	3, 7 *	2	9
Immediate to accumulator	0010010w | data | data f w = 1	3	3		
TEST = **And function to flags, no result:**					
Register/memory and register	1000010w | mod reg r/m	2, 6 *	2, 6 *	2	9
Immediate data and register/memory	1111011w | mod 000 r/m | data | data if w = 1	3, 6 *	3, 6 *	2	9
Immediate data and accumulator	1010100w | data | data if w = 1	3	3		
OR = **Or:**					
Reg/memory and register to either	000010dw | mod reg r/m	2, 7 *	2, 7 *	2	9
Immediate to register/memory	1000000w | mod 001 r/m | data | data if w = 1	3, 7 *	3, 7 *	2	9
Immediate to accumulator	0000110w | data | data if w = 1	3	3		
XOR = **Exclusive or:**					
Reg/memory and register to either	001100dw | mod reg r/m	2, 7 *	2, 7 *	2	9
Immediate to register/memory	1000000w | mod 110 r/m | data | data if w = 1	3, 7 *	3, 7 *	2	9
Immediate to accumulator	0011010w | data | data if w = 1	3	3		
NOT = Invert register/memory	1111011w | mod 010 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 010 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 011 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

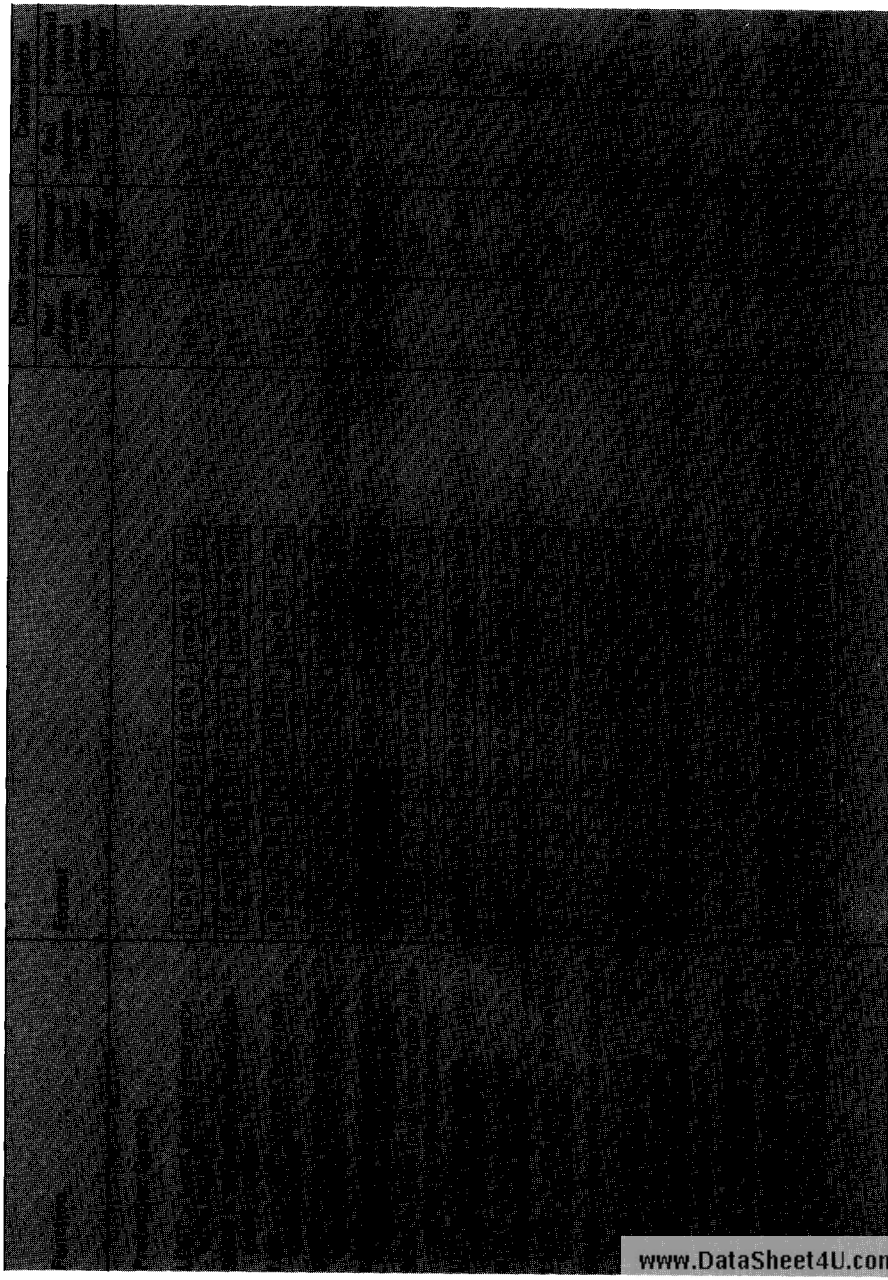
Shaded areas indicate instructions not available in SAB 8086, 8088 microsystems.

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	8	8	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
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

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
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	5	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.



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} = +5V ± 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

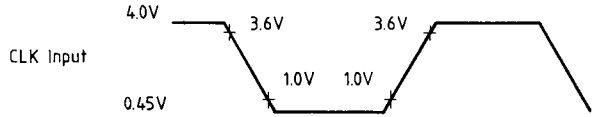
4) Float condition occurs when output current is less than I_{LO} in magnitude.

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.

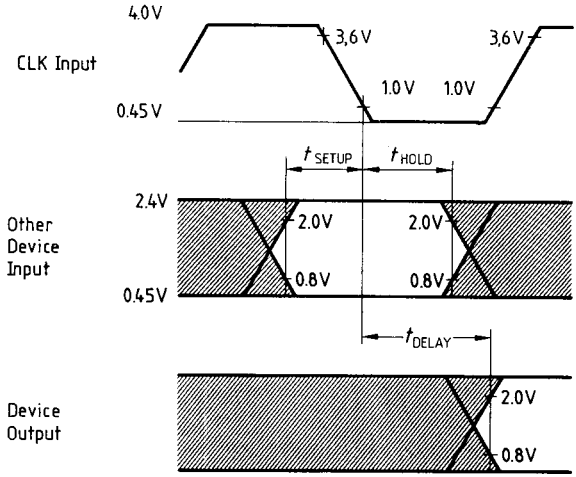
AC Testing Waveforms

Test Loading on Outputs

Drive and Measurement Points – CLK Input

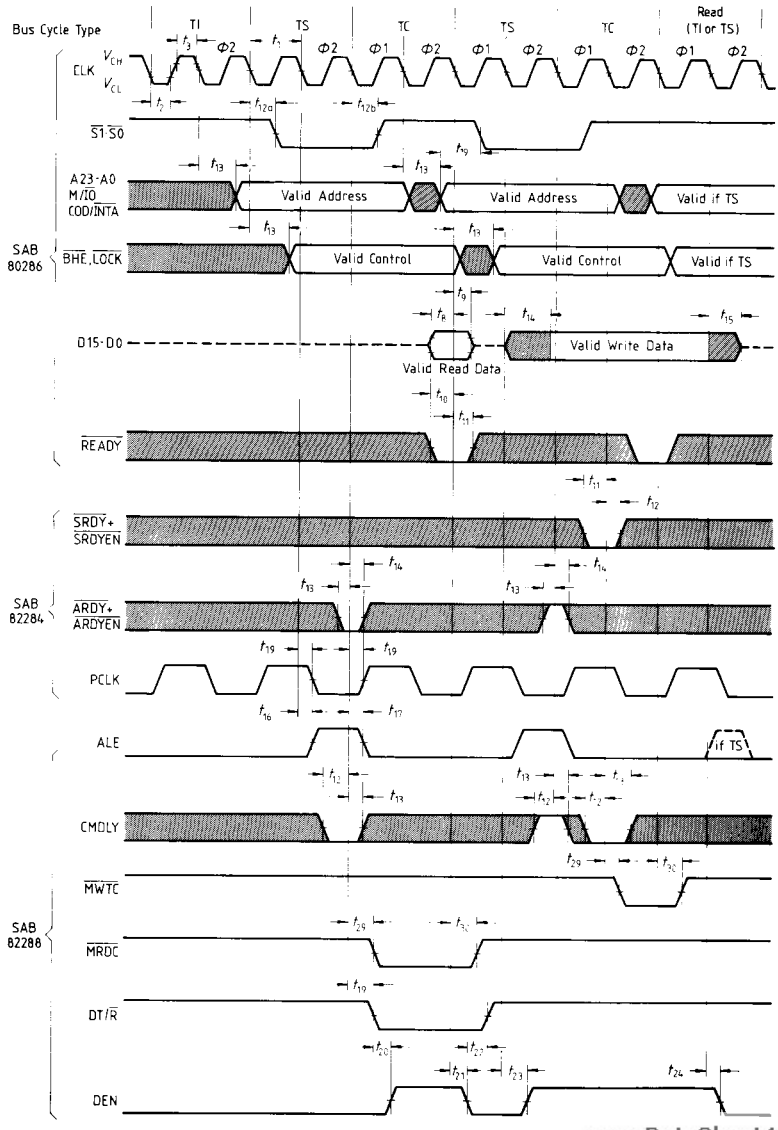


Setup, Hold and Delay Time Measurement – General

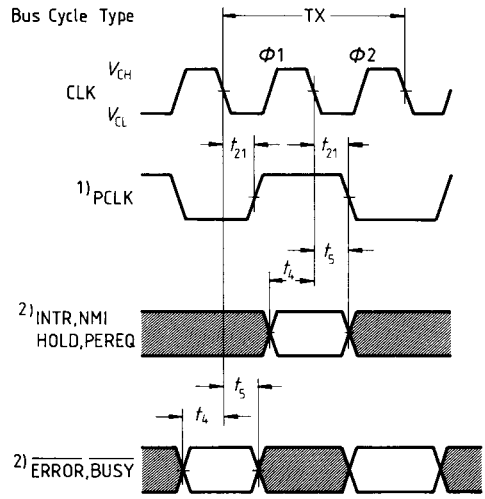


Waveforms

Major Cycle Timing



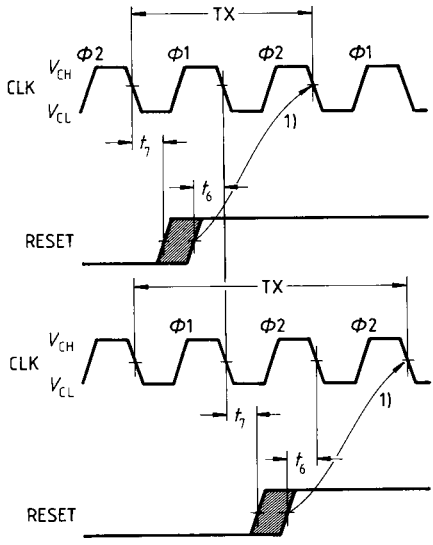
Asynchronous Input Signal Timing



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.

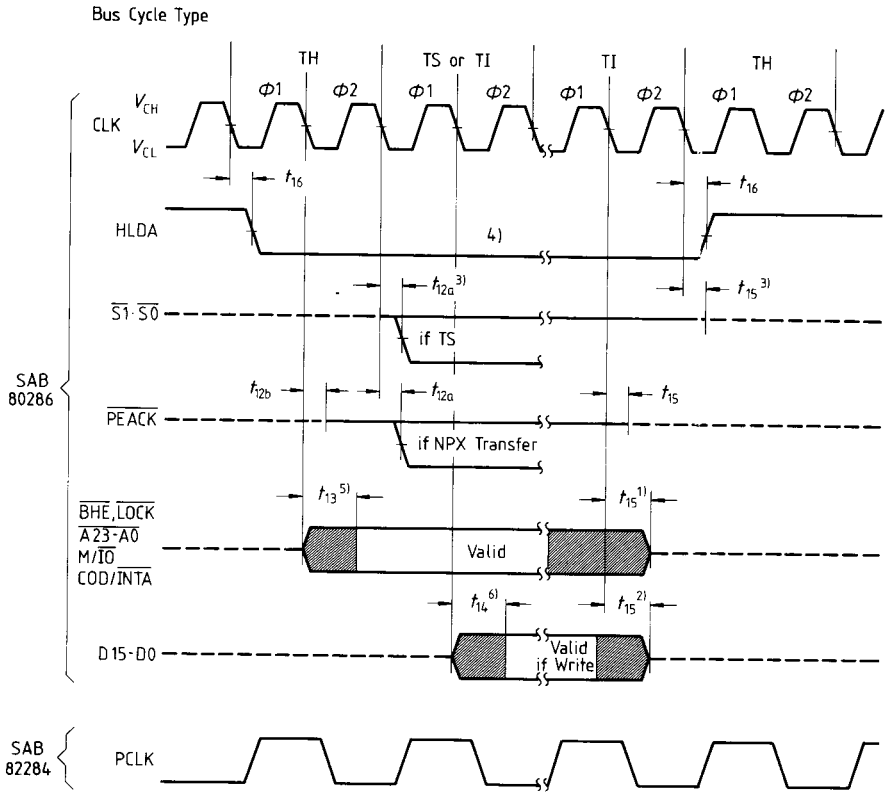
2) These inputs are asynchronous. The setup and hold times shown assure recognition for testing purposes.

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.

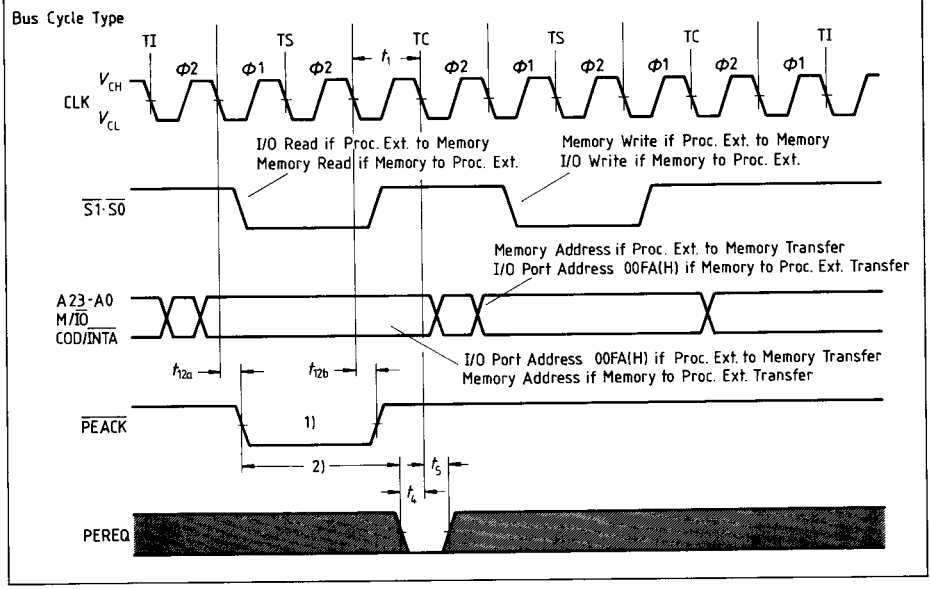
3) The SAB 80286 floats its status pins during TH. Pullup resistors in SAB 82288 keep these signals high.

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.

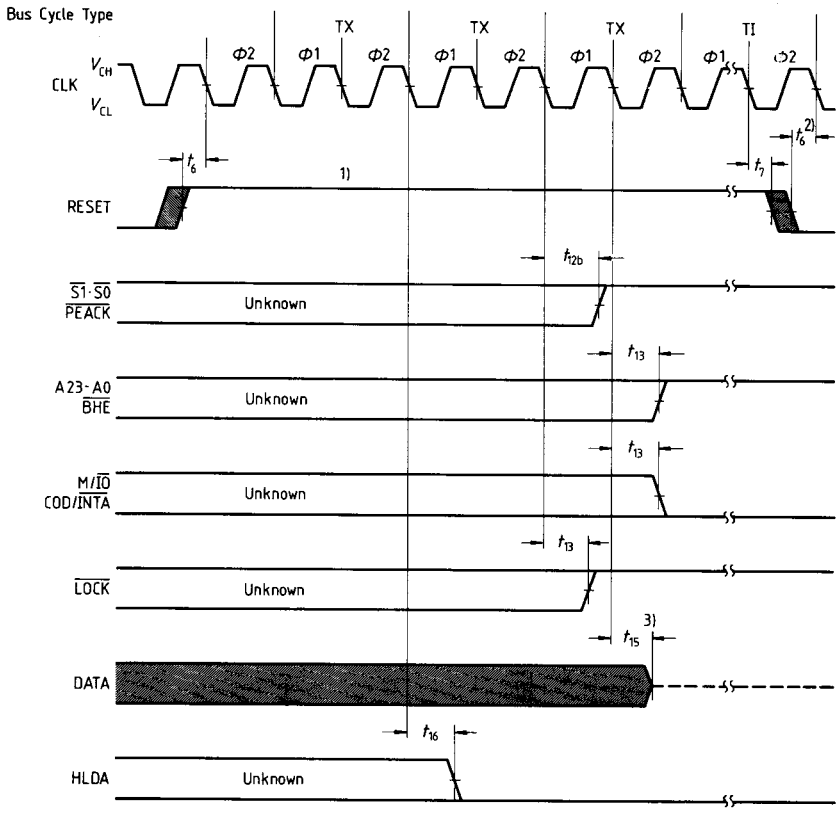
PEREQ/PEACK Timing for One Transfer Only



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



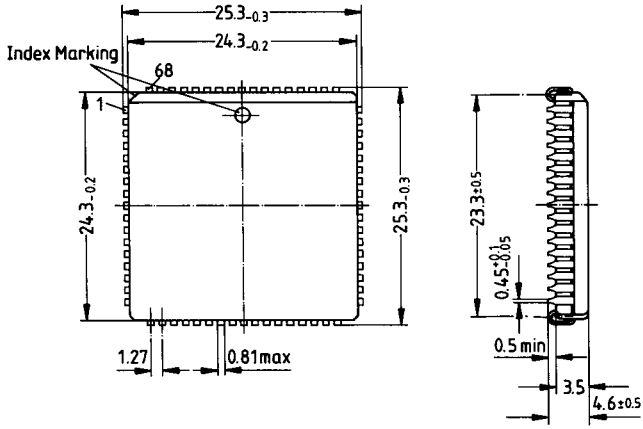
1) Setup time for RESET↑ may be violated with the consideration that Φ 1 of the processor clock may begin one system CLK period later.

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.

Package Outlines

Plastic Package, PL-CC-68 (SMD)
(Plastic leaded chip carrier)



Dimensions in mm