

80188

High Integration 8-Bit Microprocessor
iAPX86 Family



80188

Advanced Micro Devices

DISTINCTIVE CHARACTERISTICS

- Integrated feature set
Enhanced 10 MHz 8088-1 CPU
Clock generator
Two independent, high-speed DMA channels
Programmable interrupt controller
Three programmable 16-bit timers
Programmable memory and peripheral chip-select logic
Programmable wait state generator
Local bus controller
- Eight-bit data bus interface, 16-bit internal architecture
- Available in 10 MHz (80188-10), 8 MHz (80188)
- High-performance processor
Two times the performance of the standard 8088
2.25 Mbyte/sec bus bandwidth interface
- Direct addressing capability to 1 Mbyte of memory
- Completely object code compatible with all existing iAPX 86, 88 software
Ten new instruction types
Compatible with 29843/45, 29833/63, 8284, and 8288 bus support components
- Optional numeric processor extension
- Available in 68-pin Plastic Leaded Chip Carrier (PLCC), Ceramic Leadless Chip Carrier (LCC), and Pin Grid Array (PGA) packages.

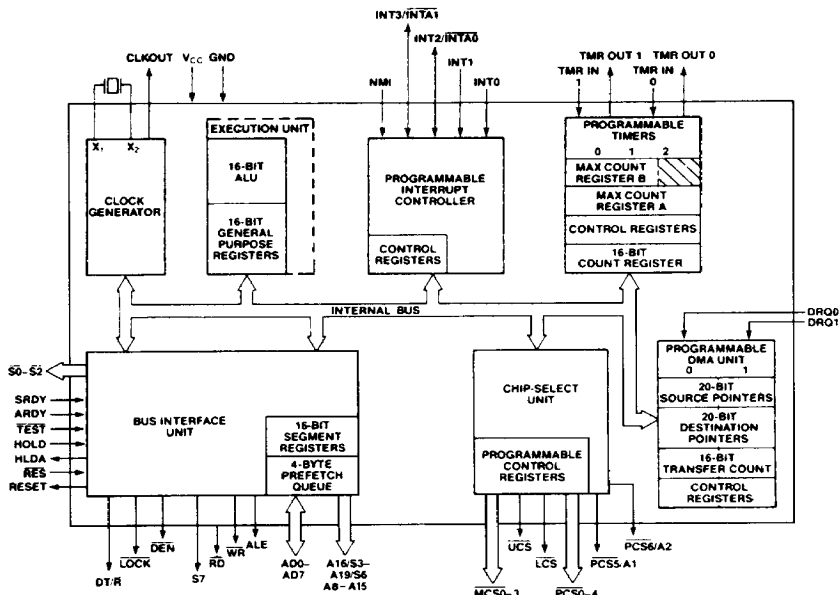
GENERAL DESCRIPTION

The 80188 is a highly-integrated microprocessor with an 8-bit data bus interface and a 16-bit internal architecture for high performance. It effectively combines 15-20 of the most common iAPX 86 system components onto one. The 8 MHz 80188 provides two times greater throughput than

the standard 5 MHz 8088. The 80188 is upward compatible with 8086 and 8088 software and adds 10 new instruction types to the existing set.

The 80188 comes in a 68-pin package and requires a single +5 V power supply.

BLOCK DIAGRAM



BD003561

Reprinted by permission of Intel Corp. copyright 1983.

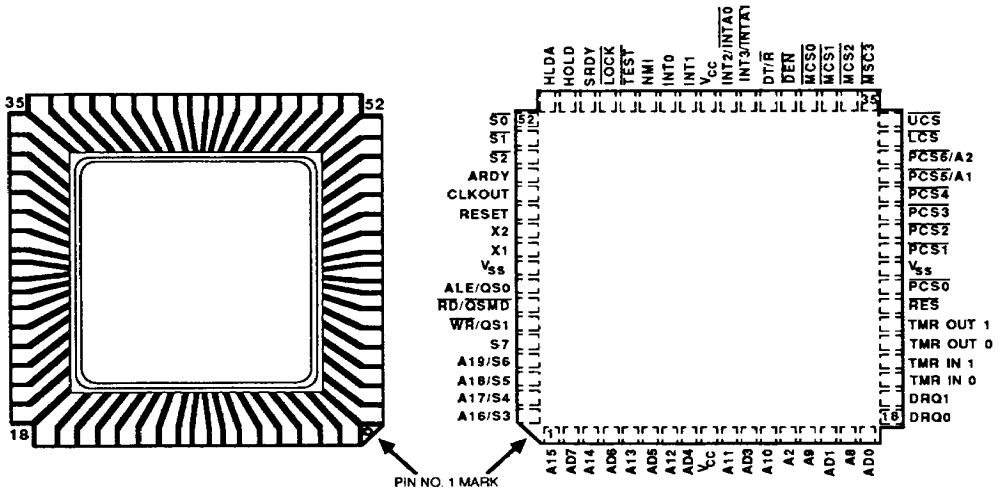
Publication #	Rev.	Amendment
06878	B	/0
Issue Date: January 1989		

CONNECTION DIAGRAMS

Leadless Chip Carrier (LCC*)

Top

Bottom



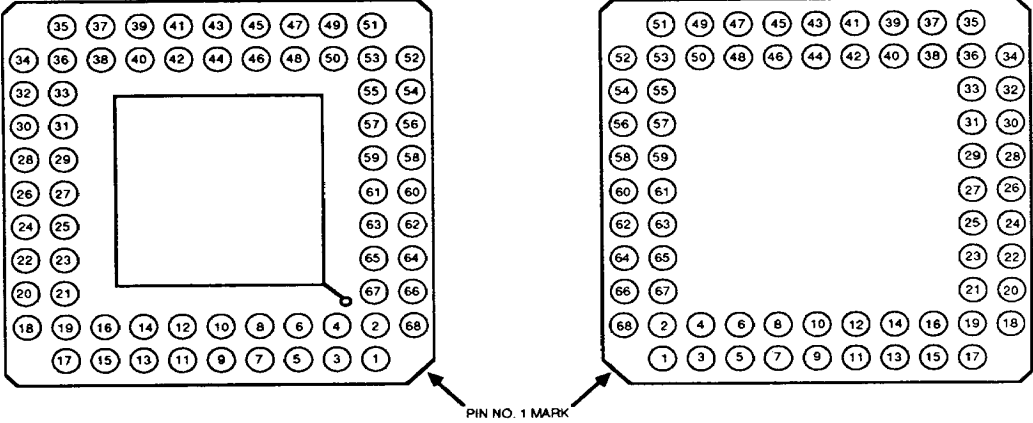
CD010511

*LCC package placed in socket top down.

Pin Grid Array (PGA)

Pins Facing Up

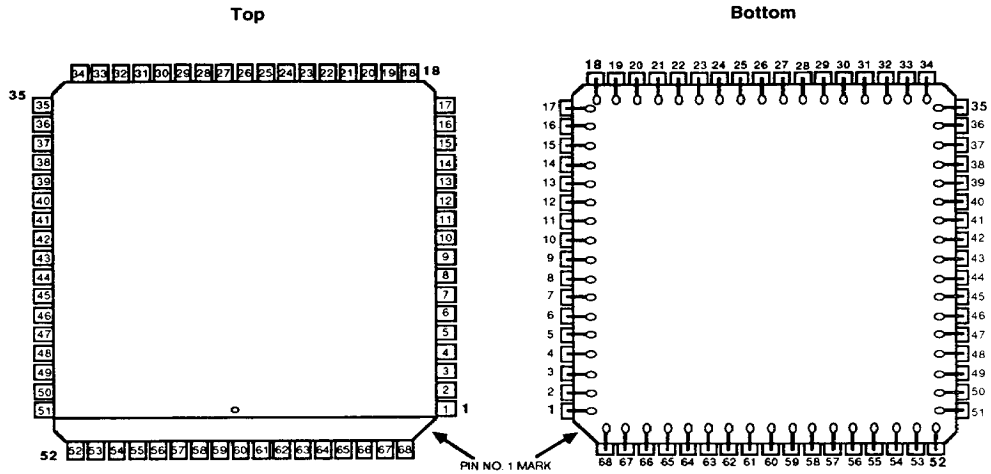
Pins Facing Down



CD010521

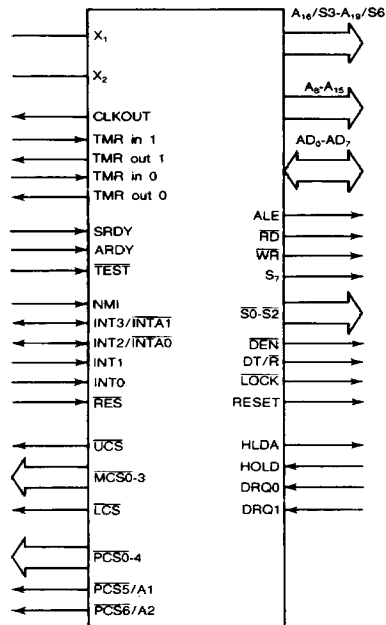
CONNECTION DIAGRAMS (Cont'd.)

Plastic Leaded Chip Carrier (PLCC**)



**PLCC package placed in socket top up.
(PLCC pin-out same as LCC.)

LOGIC SYMBOL

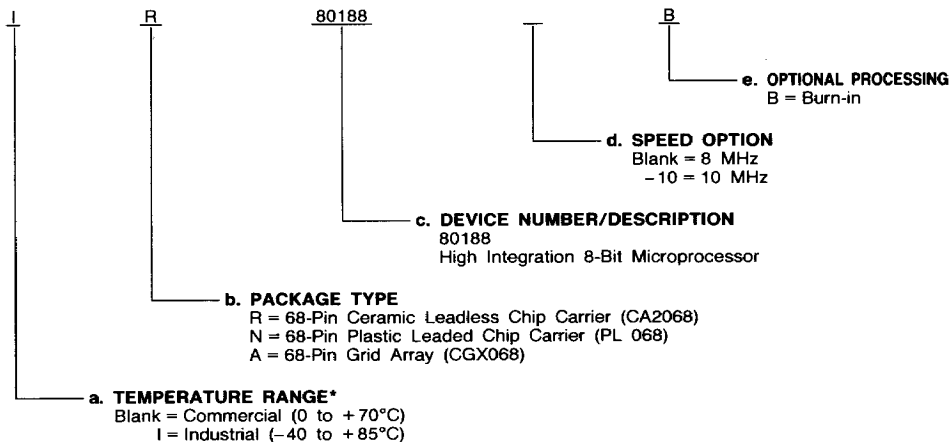


LS001970

ORDERING INFORMATION Commodity Products

AMD commodity products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of:

- a. Temperature Range
- b. Package Type
- c. Device Number
- d. Speed Option
- e. Optional Processing



Valid Combinations	
A, R, N	80188
	80188-10
A, R, IA, IR	80188B

Valid Combinations

Valid combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released valid combinations, and to obtain additional data on AMD's standard military grade products.

*This device is also available in Military temperature range. See MOS Micro-processors and Peripherals Handbook (Order #09275A/0) for electrical performance characteristics.

PIN DESCRIPTION

Active State	Name	I/O	Description						
	V _{CC} , V _{CC}	I	System Power: +5 volt power supply.						
	V _{SS} , V _{SS}	I	System Ground.						
Active HIGH	RESET	O	Reset Output indicates that the 80188 CPU is being reset; and can be used as a system reset. It is active HIGH, synchronized with the processor clock, and lasts an integer number of clock periods corresponding to the length of the RES signal.						
	X1, X2	I	Crystal inputs, X1 and X2, provide an external connection for a fundamental mode parallel resonant crystal for the internal crystal oscillator. X1 can interface to an external clock instead of a crystal. The input or oscillator frequency is internally divided by two to generate the clock signal (CLKOUT).						
	CLKOUT	O	Clock Output provides the system with a 50% duty cycle waveform. All device pin timings are specified relative to CLKOUT. CLKOUT has sufficient MOS drive capabilities for a numeric processor extension.						
Active LOW	RES	I	System Reset causes the 80188 to immediately terminate its present activity, clear the internal logic, and enter a dormant state. This signal may be asynchronous to the 80188 clock. The 80188 begins fetching instructions approximately 7 clock cycles after RES is returned HIGH. RES is required to be LOW for greater than 4 clock cycles and is internally synchronized. For proper initialization, the LOW-to-HIGH transition of RES must occur no sooner than 50 microseconds after power up. This input is provided with a Schmitt-trigger to facilitate power-on RES generation via an RC network. When RES occurs, the 80188 will drive the status lines to an inactive level for one clock, and then tri-state them.						
Active LOW	TEST	I	TEST is examined by the WAIT instruction. If the TEST input is HIGH when "WAIT" execution begins, instruction execution will suspend. TEST will be resampled until it goes LOW, at which time execution will resume. If interrupts are enabled while the 80188 is waiting for TEST, interrupts will be serviced. This input is synchronized internally.						
Active HIGH	TMR in 0, TMR IN1	I I	Timer inputs are used either as clock or control signals, depending upon the programmed timer mode. These inputs are active HIGH (or LOW-to-HIGH transitions are counted) and internally synchronized.						
Active HIGH	TMR OUT 0, TMR OUT 1	O O	Timer outputs are used to provide single pulse or continuous waveform generation, depending upon the timer mode selected.						
Active HIGH	DRQ0 DRQ1	I I	DMA Request is driven HIGH by an external device when it desires that a DMA channel (Channel 0 or 1) perform a transfer. These signals are active HIGH, level-triggered, and internally synchronized.						
Active HIGH	NMI	I	Non-Maskable Interrupt is an edge-triggered input which causes a type 2 interrupt. NMI is not maskable internally. A transition from a LOW to HIGH initiates the interrupt at the next instruction boundary. NMI is latched internally. An NMI duration of one clock or more will guarantee service. This input is internally synchronized.						
Active HIGH or LOW	INT0, INT1 INT2/INTA0 INT3/INTA1	I I/O I/O	Maskable Interrupt Requests can be requested by strobing one of these pins. When configured as inputs, these pins are active HIGH. Interrupt Requests are synchronized internally. INT2 and INT3 may be configured via software to provide active-LOW interrupt-acknowledge output signals. All interrupt inputs may be configured via software to be either edge- or level-triggered. To ensure recognition, all interrupt requests must remain active until the interrupt is acknowledged. When iRMX mode is selected, the function of these pins changes (see Interrupt Controller section of this data sheet).						
Active HIGH	A19/S6, A18/S5, A17/S4, A16/S3	O O O O	Address Bus Outputs (16-19) and Bus Cycle Status (3-6) reflect the four most significant address bits during T ₁ . These signals are active HIGH. During T ₂ , T ₃ , T _W , and T ₄ , status information is available on these lines as encoded below: <table border="1" style="margin: 10px auto; width: 80%;"> <thead> <tr> <th></th> <th>Low</th> <th>High</th> </tr> </thead> <tbody> <tr> <td>S6</td> <td>Processor Cycle</td> <td>DMA Cycle</td> </tr> </tbody> </table> <p>S3, S4, and S5 are defined as LOW during T₂-T₄.</p>		Low	High	S6	Processor Cycle	DMA Cycle
	Low	High							
S6	Processor Cycle	DMA Cycle							
Active HIGH	AD7-AD0	I/O	Address/Data Bus (0-7) signals constitute the time multiplexed memory or I/O address (T ₁) and data (T ₂ , T ₃ , T _W , and T ₄) bus.						
Active HIGH	A15-A8	O	Address-only Bus (8-15), containing valid address from T ₁ -T ₄ .						
Active HIGH	S7	O	This signal is always HIGH to indicate that the 80188 has an 8-bit data bus and is tri-state OFF during bus hold.						
Active HIGH	ALE/QS0	O	Address Latch Enable/Queue Status 0 is provided by the 80188 to latch the address into the 8282/8283 address latches. Addresses are guaranteed to be valid on the trailing edge of ALE. The ALE rising edge is generated off the rising edge of the CLKOUT immediately preceding T ₁ of the associated bus cycle, effectively one-half clock cycle earlier than in the standard 8088. The trailing edge is generated off the CLKOUT rising edge in T ₁ as in the 8088. Note that ALE is never floated.						

PIN DESCRIPTION (Cont.)

Active State	Name	I/O	Description																																								
Active LOW	WR/QS1	O	<p>Write Strobe/Queue Status 1 indicates that the data on the bus is to be written into a memory or an I/O device. WR is active for T₂, T₃, and T_W of any write cycle. Floats during "HOLD." It is driven HIGH for one clock during Reset, and then floated. When the 80188 is in queue status mode, the ALE/QS0 and WR/QS1 pins provide information about processor/instruction queue interaction.</p> <table border="1" style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">QS1</th> <th style="text-align: center;">QS0</th> <th style="text-align: center;">Queue Operation</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>No queue operation</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>First opcode byte fetched from the queue</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>Subsequent byte fetched from the queue</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>Empty the queue</td> </tr> </tbody> </table>	QS1	QS0	Queue Operation	0	0	No queue operation	0	1	First opcode byte fetched from the queue	1	1	Subsequent byte fetched from the queue	1	0	Empty the queue																									
QS1	QS0	Queue Operation																																									
0	0	No queue operation																																									
0	1	First opcode byte fetched from the queue																																									
1	1	Subsequent byte fetched from the queue																																									
1	0	Empty the queue																																									
Active LOW	RD/QSMD	O	<p>Read Strobe indicates that the 80188 is performing a memory or I/O read cycle. RD is active LOW for T₂, T₃, and T_W of any read cycle. It is guaranteed not to go LOW in T₂ until after the Address Bus is floated. RD is active LOW, and floats during "HOLD." RD is driven HIGH for one clock during Reset, and then the output driver is floated. A weak internal pull-up mechanism on the RD line holds it HIGH when the line is not driven. During RESET the pin is sampled to determine whether the 80188 should provide ALE, WR, and RD, or if the Queue-Status should be provided. RD should be connected to GND to provide Queue-Status data.</p>																																								
Active HIGH	ARDY	I	<p>Asynchronous Ready informs the 80188 that the addressed memory space or I/O device will complete a data transfer. The ARDY input pin will accept an asynchronous input, and is active HIGH. Only the rising edge is internally synchronized by the 80188. This means that the falling edge of ARDY must be synchronized to the 80188 clock. If connected to V_{CC}, no WAIT states are inserted. Asynchronous ready (ARDY) or synchronous ready (SRDY) /ust be active to terminate a bus cycle. If line is unused, it may remain connected to V_{CC} or it may be connected to V_{SS} (in which case the programmer must initialize the part to inhibit the external pins).</p>																																								
Active HIGH	SRDY	I	<p>Synchronous Ready must be synchronized externally to the 80188. The use of SRDY provides a relaxed system-timing specification on the Ready input. This is accomplished by eliminating the one-half clock cycle which is required for internally resolving the signal level when using the ARDY input. This line is active HIGH. If this line is connected to V_{CC}, no WAIT states are inserted. Asynchronous ready (ARDY) or synchronous ready (SRDY) must be active before a bus cycle is terminated. If line is unused, it may remain connected to V_{CC} or it may be connected to V_{SS} (in which case the programmer must initialize the part to inhibit the external pins).</p>																																								
Active LOW	LOCK	O	<p>LOCK output indicates that other system bus masters are not to gain control of the system bus while LOCK is active LOW. The LOCK signal is requested by the LOCK prefix instruction and is activated at the beginning of the first data cycle associated with the instruction following the LOCK prefix. It remains active until the completion of the instruction following the LOCK prefix. No prefetches will occur while LOCK is asserted. When executing more than one LOCK instruction, there must be six bytes of code between the end of the first LOCK instruction and the start of the second LOCK instruction. LOCK is active LOW, is driven HIGH for one clock during RESET, and then floated.</p>																																								
	S ₀ , S ₁ , S ₂	O	<p>Bus cycle status S₀-S₂ are encoded to provide bus-transaction information:</p> <table border="1" style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <thead> <tr> <th colspan="4" style="text-align: center;">80188 Bus Cycle Status Information</th> </tr> <tr> <th style="text-align: center;">S₂</th> <th style="text-align: center;">S₁</th> <th style="text-align: center;">S₀</th> <th style="text-align: center;">Bus Cycle Initiated</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>Interrupt Acknowledge</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>Read I/O</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>Write I/O</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>Halt</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>Instruction Fetch</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>Read Data from Memory</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>Write Data to Memory</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>Passive (no bus cycle)</td> </tr> </tbody> </table> <p>The status pins float during "HOLD." S₂ may be used as a logical M/I_O indicator, and S₁ as a DT/\bar{R} indicator. The status lines are driven HIGH for one clock during Reset, and then floated until a bus cycle begins.</p>	80188 Bus Cycle Status Information				S ₂	S ₁	S ₀	Bus Cycle Initiated	0	0	0	Interrupt Acknowledge	0	0	1	Read I/O	0	1	0	Write I/O	0	1	1	Halt	1	0	0	Instruction Fetch	1	0	1	Read Data from Memory	1	1	0	Write Data to Memory	1	1	1	Passive (no bus cycle)
80188 Bus Cycle Status Information																																											
S ₂	S ₁	S ₀	Bus Cycle Initiated																																								
0	0	0	Interrupt Acknowledge																																								
0	0	1	Read I/O																																								
0	1	0	Write I/O																																								
0	1	1	Halt																																								
1	0	0	Instruction Fetch																																								
1	0	1	Read Data from Memory																																								
1	1	0	Write Data to Memory																																								
1	1	1	Passive (no bus cycle)																																								
Active HIGH	HOLD (input) HLDA (output)	I O	<p>HOLD indicates that another bus master is requesting the local bus. The HOLD input is active HIGH. HOLD may be asynchronous with respect to the 80188 clock. The 80188 will issue a HLDA in response to a HOLD request at the end of T₄ or T₁ Simultaneous with the issuance of HLDA, the 80188 will float the local bus and control lines. After HOLD is detected as being LOW, the 80188 will lower HLDA. When the 80188 needs to run another bus cycle, it will again drive the local bus and control lines.</p>																																								
Active LOW	UCS	O	<p>Upper Memory Chip Select is an active LOW output whenever a memory reference is made to the defined upper portion (1K-256K block) of memory. This line is not floated during bus HOLD. The address range activating UCS is software programmable.</p>																																								
Active LOW	LCS	O	<p>Lower Memory Chip Select is active LOW whenever a memory reference is made to the defined lower portion (1K-256K) of memory. This line is not floated during bus HOLD. The address range activating LCS is software programmable.</p>																																								
Active LOW	MCS0-3	O	<p>Mid-Range Memory Chip Select signals are active LOW when a memory reference is made to the defined mid-range portion of memory (8K-512K). These lines are not floated during bus HOLD. The address ranges activating MCS0-3 are software programmable.</p>																																								
Active LOW	PCS0 PCST-4	O O	<p>Peripheral Chip Select signals 0-4 are active LOW when a reference is made to the defined peripheral area (65K byte I/O space). These lines are not floated during bus HOLD. The address ranges activating PCS0-4 are software programmable.</p>																																								

PIN DESCRIPTION (Cont.)

Active State	Name	I/O	Description
Active LOW or HIGH	PCS5/A1	O	Peripheral Chip Select 5 or Latched A1 may be programmed to provide a sixth peripheral chip select, or to provide an internally latched A1 signal. The address range activating PCS5 is software programmable. When programmed to provide latched A1, rather than PCS5, this pin will retain the previously latched value of A1 during a bus HOLD.
Active LOW or HIGH	PCS6/A2	O	Peripheral Chip Select 6 or Latched A2 may be programmed to provide a seventh peripheral chip select, or to provide an internally latched A2 signal. The address range activating PCS6 is software programmable. When programmed to provide latched A2, rather than PCS6, this pin will retain the previously latched value of A2 during a bus HOLD.
	DT/R	O	Data Transmit/Receive controls the direction of data flow through the external 2946/47 data bus transceiver. When LOW, data is transferred to the 80188. When HIGH the 80188 places write data on the data bus.
Active LOW	DEN	O	Data Enable is provided as a 2946/47 data bus transceiver output enable. DEN is active LOW during each memory and I/O access. DEN is HIGH whenever DT/R changes state.

FUNCTIONAL DESCRIPTION

Introduction

The following Functional Description describes the base architecture of the 80188. The architecture is common to the 8086, 8088, and 80286 microprocessor families as well. The 80188 is a very high integration 8-bit microprocessor. It combines 15-20 of the most common microprocessor system components onto one chip while providing twice the performance of the standard 8088. The 80188 is object code compatible with the 8086, 8088 microprocessors and adds 10 new instruction types to the existing 8086, 8088 instruction set.

80188 BASE ARCHITECTURE

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

Register Set

The 80188 base architecture has fourteen registers as shown in Figures 1 and 2. These registers are grouped into the following 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 as 16-bit registers or split into pairs of separate 8-bit registers.

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

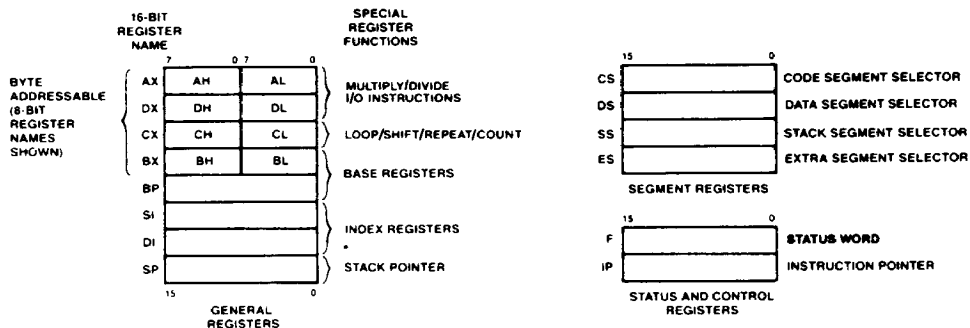
Four of the general purpose registers may also be used to determine offset addresses of operands in memory. These registers may contain base addresses or indexes to particular locations within a segment. The addressing mode selects the specific registers for operand and address calculations.

Status and Control Registers

Two 16-bit special purpose registers record or alter certain aspects of the 80188 processor state. These are the Instruction Pointer Register, which contains the offset address of the next sequential instruction to be executed, and the Status Word Register, which contains status and control flag bits (see Figures 1 and 2).

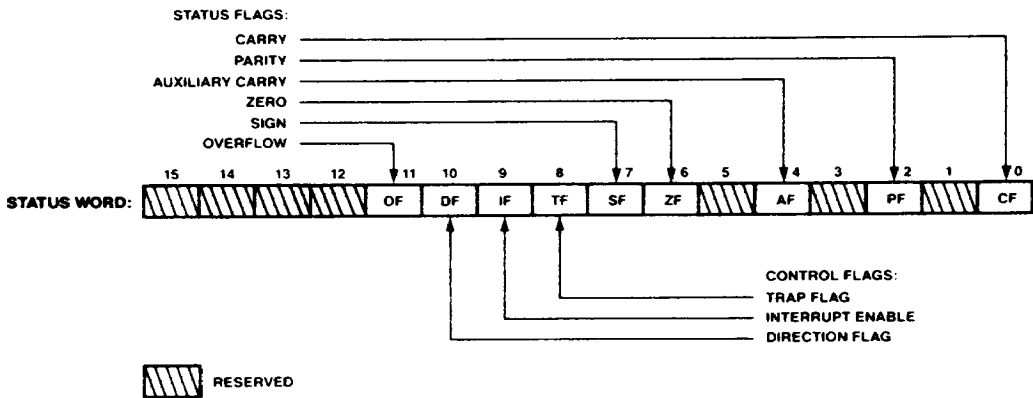
Status Word Description

The Status Word records specific characteristics of the result of logical and arithmetic instructions (bits 0, 2, 4, 6, 7, and 11) and controls the operation of the 80188 within a given operating mode (bits 8, 9, and 10). The Status Word Register is 16-bits wide. The function of the Status Word bits is shown in Table 2.



TB000045

Figure 1. 80188 General Purpose Register Set



DF002910

Figure 2. Status Word Format

Table 2. Status Word Bit Function

Bit Position	Name	Function
0	CF	Carry Flag — Set on high-order bit carry or borrow; cleared otherwise.
2	PF	Parity Flag — Set if low-order 8 bits or result contain an even number of 1-bits; cleared otherwise.
4	AF	Set on carry from or borrow to the low order four 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).
8	TF	Single Step Flag — Once set, a single step interrupt occurs after the next instruction executes. 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 auto decrement the appropriate index register when set. Clearing DF causes auto increment.
11	OF	Overflow Flag — Set if the signed result cannot be expressed within the number of bits in the destination operand; cleared otherwise.

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. These categories are summarized in Figure 3.

An 80188 instruction can reference anywhere from zero to several operands. An operand can reside in a register, in the instruction itself, or in memory. Specific operand addressing modes are discussed later in this data sheet.

Memory Organization

Memory is organized in sets of segments. Each segment is a linear contiguous sequence of up to 64K (2^{16}) 8-bit bytes. Memory is addressed using a two-component address (a pointer) that consists of a 16-bit base segment and a 16-bit offset. The 16-bit base values are contained in one of four internal segment registers (code, data, stack, extra). The physical address is calculated by shifting the base value LEFT by four bits and adding the 16-bit offset value to yield a 20-bit physical address (see Figure 4). This allows for a 1 MByte physical address size.

All instructions that address operands in memory must specify the base segment and the 16-bit offset value. For speed and compact instruction encoding, the segment register used for physical address generation is implied by the addressing mode used (see Table 3). These rules follow the way programs are written (see Figure 5) 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.

GENERAL PURPOSE	
MOV	Move byte or word
PUSH	Push word onto stack
POP	Pop word off stack
PUSHA	Push all registers on stack
POPA	Pop all registers from stack
XCHG	Exchange byte or word
XLAT	Translate byte
INPUT/OUTPUT	
IN	Input byte or word
OUT	Output byte or word
ADDRESS OBJECT	
LEA	Load effective address
LDS	Load pointer using DS
LES	Load pointer using ES
FLAG TRANSFER	
LAHF	Load AH register from flags
SAHF	Store AH register in flags
PUSHF	Push flags onto stack
POPF	Pop flags off stack

ADDITION	
ADD	Add byte or word
ADC	Add byte or word with carry
INC	Increment byte or word by 1
AAA	ASCII adjust for addition
DAA	Decimal adjust for addition
SUBTRACTION	
SUB	Subtract byte or word
SBB	Subtract byte or word with borrow
DEC	Decrement byte or word by 1
NEG	Negate byte or word
CMP	Compare byte or word
AAS	ASCII adjust for subtraction
DAS	Decimal adjust for subtraction
MULTIPLICATION	
MUL	Multiply byte or word unsigned
IMUL	Integer multiply byte or word
AAM	ASCII adjust for multiply
DIVISION	
DIV	Divide byte or word unsigned
IDIV	Integer divide byte or word
AAD	ASCII adjust for division
CBW	Convert byte to word
CWD	Convert word to doubleword
MOVS	Move byte or word string

INS	Input bytes or word string
OUTS	Output bytes or word string
CMPS	Compare byte or word string
SCAS	Scan byte or word string
LODS	Load byte or word string
STOS	Store byte or word string
REP	Repeat
REPE/ REPZ	Repeat while equal/zero
REPNE/ REPNZ	Repeat while not equal/not zero

LOGICALS	
NOT	"Not" byte or word
AND	"And" byte or word
OR	"Inclusive or" byte or word
XOR	"Exclusive or" byte or word
TEST	"Test" byte or word

SHIFTS	
SHL/SAL	Shift logical/arithmetic left byte or word
SHR	Shift logical right byte or word
SAR	Shift arithmetic right byte or word

ROTATES	
ROL	Rotate left byte or word
ROR	Rotate right byte or word
RCL	Rotate through carry left byte or word
RCR	Rotate through carry right byte or word

FLAG OPERATIONS	
STC	Set carry flag
CLC	Clear carry flag
CMC	Complement carry flag
STD	Set direction flag
CLD	Clear direction flag
STI	Set interrupt enable flag
CLI	Clear interrupt enable flag

EXTERNAL SYNCHRONIZATION	
HLT	Halt until interrupt or reset
WAIT	Wait for TEST pin active
ESC	Escape to extension processor
LOCK	Lock bus during next instruction

NO OPERATION	
NOP	No operation

HIGH LEVEL INSTRUCTIONS	
ENTER	Format stack for procedure entry
LEAVE	Restore stack for procedure exit
BOUND	Detects values outside prescribed range

Figure 3. 80188 Instruction Set

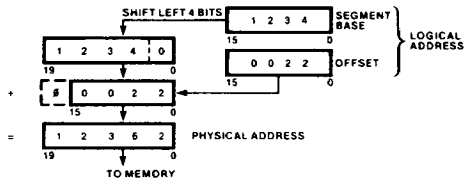
All mnemonics copyright Intel Corp.

CONDITIONAL TRANSFERS		UNCONDITIONAL TRANSFERS	
JA/JNBE	Jump if above/not below nor equal	CALL	Call procedure
JAE/JNB	Jump if above or equal/not below	RET	Return from procedure
JB/JNAE	Jump if below/not above nor equal	JMP	Jump
JBE/JNA	Jump if below or equal/not above	ITERATION CONTROLS	
JC	Jump if carry		
JE/JZ	Jump if equal/zero	LOOP	Loop
JG/JNLE	Jump if greater/not less nor equal	LOOPE/LOOPZ	Loop if equal/zero
JGE/JNL	Jump if greater or equal/not less	LOOPNE/LOOPNZ	Loop if not equal/not zero
JL/JNGE	Jump if less/not greater nor equal	JCXZ	Jump if register CX = 0
JLE/JNG	Jump if less or equal/not greater	INTERRUPTS	
JNC	Jump if not carry		
JNE/JNZ	Jump if not equal/not zero	INT	Interrupt
JNO	Jump if not overflow	INTO	Interrupt if overflow
JNP/JPO	Jump if not parity/parity odd	IRET	Interrupt return
JNS	Jump if not sign		
JO	Jump if overflow		
JP/JPE	Jump if parity/parity even		
JS	Jump if sign		

Figure 3. 80188 Instruction Set (continued)

All mnemonics copyright Intel Corp.

To access operands that do not reside in one of the four immediately available segments, a full 32-bit pointer can be used to reload both the base (segment) and offset values.

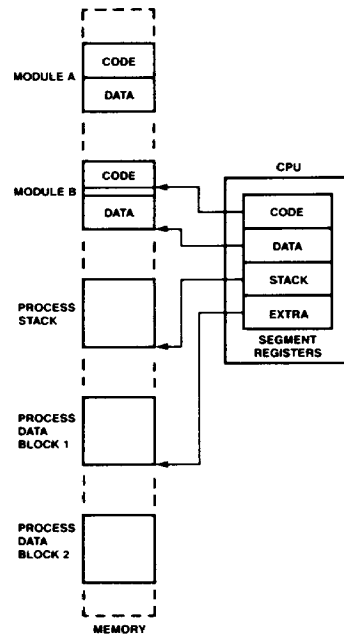


DF002920

Figure 4. Two Component Address

Table 3. Segment Register Selection Rule

Memory Reference Needed	Segment Register Used	Implicit Segment Selection Rule
Instructions	Code (CS)	Instruction prefetch and immediate data.
Stack	Stack (SS)	All stack pushes and pops; any memory references which use BP Register as a base register.
External Data (Global)	Extra (ES)	All string instruction references which use the DI register as an index.
Local Data	Data (DS)	All other data references.



DF002930

Figure 5. Segmented Memory Helps Structure Software

Addressing Modes

The 80188 provides eight categories of addressing modes 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.

Six modes are provided to specify the location of an operand in a memory segment. A memory operand address consists of two 16-bit components: a segment base and an offset. The segment base is supplied by a 16-bit segment register either implicitly chosen by the addressing mode or explicitly chosen by a segment override prefix. The offset, also called the effective address, is calculated by summing any combination of the following three address elements:

- the *displacement* (an 8- or 16-bit immediate value contained in the instruction);
- the *base* (contents of either the BX or BP base registers); and
- the *index* (contents of either the SI or DI index registers)

Any carry out from the 16-bit addition is ignored. Eight-bit displacements are sign extended to 16-bit values.

Combinations of these three address elements define the six memory addressing modes, described below.

- **Direct Mode:** The operand's offset is contained in the instruction as an 8- or 16-bit displacement element.
- **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.

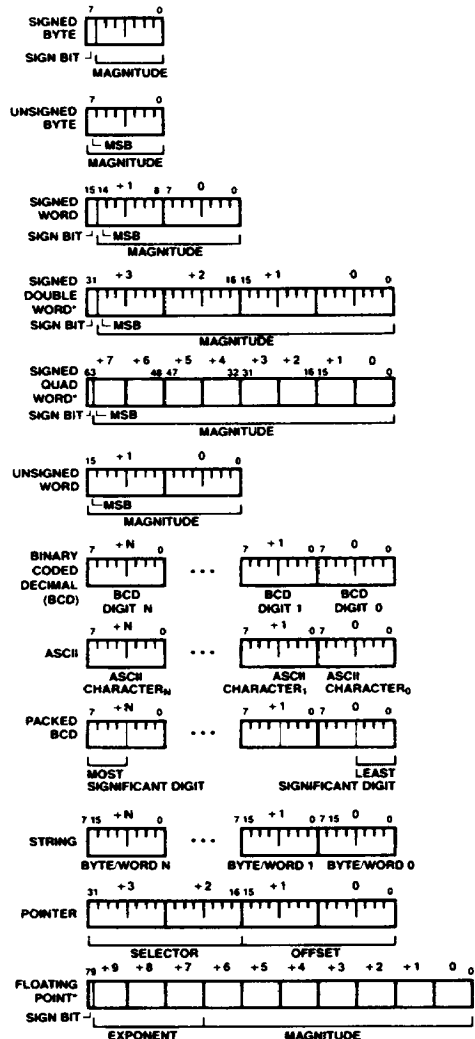
Data Types

The 80188 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 two's complement representation. Signed 32 and 64 bit integers are supported using a numeric data processor.
- **Ordinal:** An unsigned binary numeric value contained in an 8-bit byte or a 16-bit word.
- **Pointer:** A 16- or 32-bit quantity, composed of a 16-bit offset component or a 16-bit segment base component in addition to a 16-bit offset component.
- **String:** A contiguous sequence of bytes or words. A string may contain from 1K to 64K bytes.
- **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 0-9.
- **Packed BCD:** A byte (packed) representation of two decimal digits (0-9). One digit is stored in each nibble (4-bits) of the byte.
- **Floating Point:** A signed 32-, 64-, or 80-bit real number representation. (Floating point operands are supported using a numeric data processor configuration.)

In general, individual data elements must fit within defined segment limits. Figure 6 graphically represents the data types supported by the 80188.



DF002940

NOTE: *SUPPORTED BY 80188 WITH A NUMERIC DATA PROCESSOR

Figure 6. 80188 Supported Data Types

I/O Space

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

Interrupts

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

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. If the exception was caused by executing an ESC instruction with the ESC trap bit set in the relocation register, the return instruction will point to the ESC instruction, or to the segment override prefix immediately preceding the ESC instruction if the prefix was present. In all other cases, the return address from an exception will point at the instruction immediately following the instruction causing the exception.

A table containing up to 256 pointers defines the proper interrupt service routine for each interrupt. Interrupts 0-31, some of which are used for instruction exceptions, are reserved. Table 4 shows the 80188 predefined types and default priority levels. For each interrupt, an 8-bit vector must be supplied to the 80188 which identifies the appropriate table entry. Exceptions supply the interrupt vector internally. In addition, internal peripherals and noncascaded external interrupts will generate their own vectors through the internal interrupt controller. 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.

Interrupt Sources

The 80188 can service interrupts generated by software or hardware. The software interrupts are generated by specific instructions (INT, ESC, unused OP, etc.) or the results of conditions specified by instructions (array bounds check, INTO, DIV, IDIV, etc.) All interrupt sources are serviced by an indirect call through an element of a vector table. This vector table is indexed by using the interrupt vector type (Table 4), multiplied by four. All hardware-generated interrupts are sampled at the end of each instruction. Thus, the software interrupts will begin service first. Once the service routine is entered and interrupts are enabled, any hardware source of sufficient priority can interrupt the service routine in progress.

The software generated 80188 interrupts are described below.

DIVIDE ERROR EXCEPTION (TYPE 0)

Generated when a DIV or IDIV instruction quotient cannot be expressed in the number of bits in the destination.

Table 4. 80188 Interrupt Vectors

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

NOTES:

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

SINGLE-STEP INTERRUPT (TYPE 1)

Generated after most instructions if the TF flag is set. Interrupts will not be generated after prefix instructions (e.g., REP), instructions which modify segment registers (e.g., POP DS), or the WAIT instruction.

NON-MASKABLE INTERRUPT-NMI (TYPE 2)

An external interrupt source which cannot be masked.

BREAKPOINT INTERRUPT (TYPE 3)

A one-byte version of the INT instruction. It uses 12 as an index into the service routine address table (because it is a type 3 interrupt).

INTO DETECTED OVERFLOW EXCEPTION (TYPE 4)

Generated during an INTO instruction if the OF bit is set.

ARRAY BOUNDS EXCEPTION (TYPE 5)

Generated during a BOUND instruction if the array index is outside the array bounds. The array bounds are located in memory at a location indicated by one of the instruction operands. The other operand indicates the value of the index to be checked.

UNUSED OPCODE EXCEPTION (TYPE 6)

Generated if execution is attempted on undefined opcodes.

ESCAPE OPCODE EXCEPTION (TYPE 7)

Generated if execution is attempted of ESC opcodes (D8H-DFH). This exception will only be generated if a bit in the relocation register is set. The return address of this exception will point to the ESC instruction causing the exception. If a segment override prefix preceded the ESC instruction, the return address will point to the segment override prefix.

Hardware-generated interrupts are divided into two groups: maskable interrupts and non-maskable interrupts. The 80188 provides maskable hardware interrupt request pins INTO-INT3. In addition, maskable interrupts may be generated by the 80188 integrated DMA controller and the integrated timer unit. The vector types for these interrupts is shown in Table 4. Software enables these inputs by setting the interrupt flag bit (IF) in the Status Word. The interrupt controller is discussed in the peripheral section of this data sheet.

Further maskable interrupts are disabled while servicing an interrupt because the IF bit is reset as part of the response to an interrupt or exception. The saved Status Word will reflect the enable status of the processor prior to the interrupt. The interrupt flag will remain zero unless specifically set. The interrupt return instruction restores the Status Word, thereby restoring the original status of IF bit. If the interrupt return re-enables interrupts, and another interrupt is pending, the 80188 will immediately service the highest-priority interrupt pending, i.e., no instructions of the main line program will be executed.

Non-Maskable Interrupt Request (NMI)

A non-maskable interrupt (NMI) is also provided. This interrupt is serviced regardless of the state of the IF bit. A typical use of NMI would be to activate a power failure routine. The activation of this input causes an interrupt with an internally supplied vector value of 2. No external interrupt acknowledge sequence is performed. The IF bit is cleared at the beginning of an NMI interrupt to prevent maskable interrupts from being serviced.

Single-Step Interrupt

The 80188 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 Status 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.

Initialization and Processor Reset

Processor initialization or startup is accomplished by driving the RES input pin LOW. RES forces the 80188 to terminate all execution and local bus activity. No instruction or bus activity will occur as long as RES is active. After RES becomes

inactive and an internal processing interval elapses, the 80188 begins execution with the instruction at physical location FFFF0(H). RES also sets some registers to predefined values as shown in Table 5.

Table 5. 80188 Initial Register State after RESET

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

THE 80188 COMPARED TO THE 80186

The 80188 CPU is an 8-bit processor designed around the 80186 internal structure. Most internal functions of the 80188 are identical to the equivalent 80186 functions. The 80188 handles the external bus the same way the 80186 does with the distinction of handling only 8 bits at a time. Sixteen bit operands are fetched or written in two consecutive bus cycles. Both processors will appear identical to the software engineer, with the exception of execution time. The internal register structure is identical and all instructions have the same end result. The differences between the 80188 and 80186 are outlined below. Internally, there are three differences between the 80188 and the 80186. All changes are related to the 8-bit bus interface.

- The queue length is 4 bytes in the 80188, whereas the 80186 queue contains 6 bytes, or three words. The queue was shortened to prevent overuse of the bus by the BIU when prefetching instructions. This was required because of the additional time necessary to fetch instructions 8 bits at a time.
- To further optimize the queue, the prefetching algorithm was changed. The 80188 BIU will fetch a new instruction to load into the queue each time there is a 1-byte hole (space available) in the queue. The 80186 waits until a 2-byte space is available.
- The internal execution time of the instructions is affected by the 8-bit interface. All 16-bit fetches and writes from/to memory take an additional four clock cycles. The CPU may also be limited by the speed of instruction fetches when a series of simple operations occurs. When the more sophisticated instructions of the 80188 are being used, the queue has time to fill and the execution proceeds as fast as the execution unit will allow.

The 80188 and 80186 are completely software compatible by virtue of their identical execution units. Software that is system dependent may not be completely transferable, but software that is not system dependent will operate equally well on an 80188 or an 80186.

The hardware interface of the 80188 contains the major differences between the two CPUs. The pin assignments are nearly identical, however, with the following functional changes.

- A8-A15—These pins are only address outputs on the 80188. These address lines are latched internally and remain valid throughout a bus cycle in a manner similar to the 8085 upper address lines.
- BHE has no meaning on the 80188 and has been eliminated.

80188 CLOCK GENERATOR

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

Oscillator

The oscillator circuit of the 80188 is designed to be used with a parallel resonant fundamental mode crystal. This is used as the time base for the 80188. The crystal frequency selected will be double the CPU clock frequency. Use of an LC or RC circuit is not recommended with this oscillator. If an external oscillator is used, it can be connected directly to input pin X1 in lieu of a crystal. The output of the oscillator is not directly available outside the 80188. The recommended crystal configuration is shown in Figure 7.

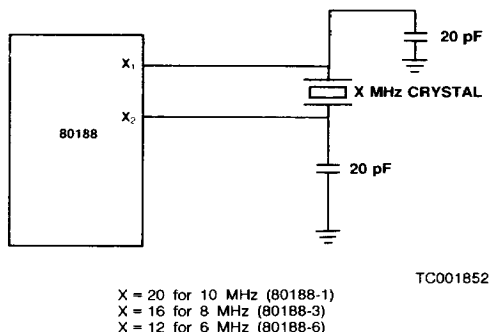


Figure 7. Recommended 80188 Crystal Configuration

Clock Generator

The 80188 clock generator provides the 50% duty cycle processor clock for the 80188. It does this by dividing the oscillator output by 2 forming the symmetrical clock. If an external oscillator is used, the state of the clock generator will change on the falling edge of the oscillator signal. The CLKOUT pin provides the processor clock signal for use outside the 80188. This may be used to drive other system components. All timings are referenced to the output clock.

READY Synchronization

The 80188 provides both synchronous and asynchronous ready inputs. Asynchronous ready synchronization is accomplished by circuitry which samples ARDY in the middle of T_2 , T_3 and again in the middle of each T_W until ARDY is sampled HIGH. One-half CLKOUT cycle of resolution time is used. Full synchronization is performed only on the rising edge of ARDY, i.e., the falling edge of ARDY must be synchronized to the CLKOUT signal if it will occur during T_2 or T_W . HIGH-to-LOW transitions of ARDY must be performed synchronously to the CPU clock.

A second ready input (SRDY) is provided to interface with externally synchronized ready signals. This input is sampled at the end of T_2 and again at the end of each T_W until it is sampled HIGH. By using this input rather than the asynchronous ready input, the half-clock cycle resolution time penalty is eliminated.

This input must satisfy set-up and hold times to guarantee proper operation of the circuit.

In addition, the 80188, as part of the integrated chip-select logic, has the capability to program WAIT states for memory and peripheral blocks. This is discussed in the Chip Select/Ready Logic description.

RESET Logic

The 80188 provides both a \overline{RES} input pin and a synchronized RESET pin for use with other system components. The \overline{RES} input pin on the 80188 is provided with hysteresis in order to facilitate power-on Reset generation via an RC network. RESET is guaranteed to remain active for at least five clocks given a \overline{RES} input of at least six clocks. RESET may be delayed up to two and one-half clocks behind \overline{RES} .

Multiple 80188 processors may be synchronized through the \overline{RES} input pin, since this input resets both the processor and divide-by-two internal counter in the clock generator. In order to insure that the divide-by-two counters all begin counting at the same time, the active going edge of \overline{RES} must satisfy a 25 ns setup time before the falling edge of the 80188 clock input. In addition, in order to insure that all CPUs begin executing in the same clock cycle, the reset must satisfy a 25 ns setup time before the rising edge of the CLKOUT signal of all the processors.

LOCAL BUS CONTROLLER

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

Memory/Peripheral Control

The 80188 provides ALE, \overline{RD} , and \overline{WR} bus control signals. The \overline{RD} and \overline{WR} signals are used to strobe data from memory to the 80188 or to strobe data from the 80188 to memory. The ALE line provides a strobe to address latches for the multiplexed address/data bus. The 80188 local bus controller does not provide a memory/I/O signal. If this is required, the user will have to use the $\overline{S_2}$ signal (which will require external latching), make the memory and I/O spaces nonoverlapping, or use only the integrated chip-select circuitry.

Transceiver Control

The 80188 generates two control signals to be connected to 2946/2947 transceiver chips. This capability allows the addition of transceivers for extra buffering without adding external logic. These control lines, DT/ \overline{R} and DEN, are generated to control the flow of data through the transceivers. The operation of these signals is shown in Table 6.

Table 6. Transceiver Control Signals Description

Pin Name	Function
DEN (Data Enable)	Enables the output drivers of the transceivers. It is active LOW during memory, I/O, or INTA cycles.
DT/ \overline{R} (Data Transmit/Receive)	Determines the direction of travel through the transceivers. A HIGH level directs data away from the processor during write operations, while a LOW level directs data toward the processor during a read operation.

Local Bus Arbitration

The 80188 uses a HOLD/HLDA system of local bus exchange. This provides an asynchronous bus exchange mechanism. This means multiple masters utilizing the same bus can operate at separate clock frequencies. The 80188 provides a single HOLD/HLDA path through which all other bus masters may gain control of the local bus. This requires external circuitry to arbitrate which external device will gain control of the bus from the 80188 when there is more than one alternate local bus master. When the 80188 relinquishes control of the local bus, it floats \overline{DEN} , \overline{RD} , \overline{WR} , $\overline{S0-S2}$, \overline{LOCK} , $AD0-AD7$, $A8-A19$, $\overline{S7}$ and DT/\overline{R} to allow another master to drive these lines directly.

The 80188 HOLD latency time, i.e., the time between HOLD request and HOLD acknowledge, is a function of the activity occurring in the processor when the HOLD request is received. A HOLD request is the highest-priority activity request which the processor may receive: higher than instruction fetching or internal DMA cycles. However, if a DMA cycle is in progress, the 80188 will complete the transfer before relinquishing the bus. This implies that if a HOLD request is received just as a DMA transfer begins, the HOLD latency time can be as great as 4 bus cycles. This will occur if a DMA word transfer operation is taking place from an odd address to an odd address. This is a total of 16 clocks or more, if WAIT states are required. In addition, if locked transfers are performed, the HOLD latency time will be increased by the length of the locked transfer.

Local Bus Controller and Reset

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

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

NOTE: \overline{RE} is also provided with an internal pull-up device to prevent the processor from inadvertently entering Queue Status mode during reset.

- Drive $\overline{S0-S2}$ to the passive state (all HIGH) and then float.
- Drive \overline{LOCK} HIGH and then float.
- Tristate $AD0-AD7$, $A8-A19$, $\overline{S7}$, DT/\overline{R}
- Drive ALE LOW (ALE is never floated).
- Drive HLDA LOW.

INTERNAL PERIPHERAL INTERFACE

All the 80188 integrated peripherals are controlled via 16-bit registers contained within an internal 256-byte control block. This control block may be mapped into either memory or I/O space. Internal logic will recognize the address and respond to the bus cycle. During bus cycles to internal registers, the bus controller will signal the operation externally (i.e., the \overline{RD} , \overline{WR} , status, address, data, etc., lines will be driven as in a normal bus cycle), but $D7_0$, $SRDY$, and $ARDY$ will be ignored. The base address of the control block must be on an even 256-byte boundary (i.e., the lower 8 bits of the base address are all zeros). All of the defined registers within this control block may

be read or written by the 80186 CPU at any time. The location of any register contained within the 256-byte control block is determined by the current base address of the control block.

The control block base address is programmed via a 16-bit relocation register contained within the control block at offset FEH from the base address of the control block (see Figure 8). It provides the upper 12 bits of the base address of the control block. Note that mapping the control register block into an address range corresponding to a chip-select range is not recommended (the chip select circuitry is discussed later in this data sheet). In addition, bit 12 of this register determines whether the control block will be mapped into I/O or memory space. If this bit is 1, the control block will be located in memory space, whereas if the bit is 0, the control block will be located in I/O space. If the control register block is mapped into I/O space, the upper 4 bits of the base address must be programmed as 0 (since I/O addresses are only 16 bits wide).

In addition to providing relocation information for the control block, the relocation register contains bits which place the interrupt controller into iRMX mode, and cause the CPU to interrupt upon encountering ESC instructions. At RESET, the relocation register is set to 20FFH. This causes the control block to start at FF00H in I/O space. An offset map of the 256-byte control register block is shown in Figure 9.

The integrated 80188 peripherals operate semiautonomously from the CPU. Access to them for the most part is via software read/write of the control and data locations in the control block. Most of these registers can be both read and written. A few dedicated lines, such as interrupts and DMA request provide real-time communication between the CPU and peripherals as in a more conventional system utilizing discrete peripheral blocks. The overall interaction and function of the peripheral blocks has not substantially changed.

CHIP-SELECT/READY GENERATION LOGIC

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

Memory Chip Selects

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

The range for each chip select is user-programmable and can be set to 2K, 4K, 8K, 16K, 32K, 64K, 128K (plus 1K and 256K for upper and lower chip selects). In addition, the beginning or base address of the midrange memory chip select may also be selected. Only one chip select may be programmed to be active for any memory location at a time. All chip select sizes are in bytes, whereas 80188 memory is arranged in words. This means that if, for example, 16 64K x 1 memories are used, the memory block size will be 128K, not 64K.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET: FEH	ET	RMX		M/IO	Relocation Address Bits R19 - R8											

ET = ESC Trap/No ESC Trap (1/0)

M/IO = Register block located in Memory / I/O Space (1/0)

RMX = Normal Interrupt Controller mode / iRMX compatible Interrupt Controller mode (0/1)

Figure 8. Relocation Register

Relocation Register	OFFSET FEH
DMA Descriptors Channel 1	DAH D0H
DMA Descriptors Channel 0	CAH C0H
Chip-Select Control Registers	A8H A0H
Timer 2 Control Registers	66H 60H
Timer 1 Control Registers	5EH 58H
Timer 0 Control Registers	56H 50H
Interrupt Controller Registers	3EH 20H

Figure 9. Internal Register Map

Upper Memory \overline{CS}

The 80188 provides a chip select, called \overline{UCS} , for the top of memory. The top of memory is usually used as the system memory because after reset the 80188 begins executing at memory location FFFF0H.

The upper limit of memory defined by this chip select is always FFFFFH, while the lower limit is programmable. By programming the lower limit, the size of the select block is also defined. Table 7 shows the relationship between the base address selected and the size of the memory block obtained.

Table 7. UMCS Programming Values

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

The lower limit of this memory block is defined in the UMCS register (see Figure 10). This register is at offset A0H in the internal control block. The legal values for bits 6-13 and the resulting starting address and memory block sizes are given in Table 7. Any combination of bits 6-13 not shown in Table 7 will result in undefined operation. After reset, the UMCS register is programmed for a 1K area. It must be reprogrammed if a larger upper memory area is desired.

Any internally generally 20-bit address whose upper 16-bits are greater than or equal to UMCS (with bits 0-5 "0") will cause UCS to be activated. UMCS bits R2-R0 are used to specify READY mode for the area of memory defined by this chip-select register, as explained below.

Lower Memory \overline{CS}

The 80188 provides a chip select for low memory called LCS. The bottom of memory contains the interrupt vector table, starting at location 00000H.

The lower limit of memory defined by this chip select is always 0H, while the upper limit is programmable. By programming the upper limit, the size of the memory block is also defined. Table 8 shows the relationship between the upper address selected and the size of the memory block obtained.

Table 8. LMCS Programming Values

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

The upper limit of this memory block is defined in the LMCS register (see Figure 11). This register is at offset A2H in the internal control block. The legal values for bits 6-15 and the resulting upper address and memory block sizes are given in Table 8. Any combination of bits 6-15 not shown in Table 8 will result in undefined operation. After reset, the LMCS register value is undefined. However, the \overline{LCS} chip-select line will not become active until the LMCS register is accessed.

Any internally generated 20-bit address whose upper 16 bits are less than or equal to LMCS (with bits 0-5 "1") will cause \overline{LCS} to be active. LMCS register bits R2-R0 are used to specify the READY mode for the area of memory defined by this chip-select register.

Mid-Range Memory \overline{CS}

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

The size of the memory block defined by the midrange select lines, as shown in Table 9, is determined by bits 8-14 of the MPCS register (see Figure 12). This register is at location A8H in the internal control block. One and only one of bits 8-14 must be set at a time. Unpredictable operation of the \overline{MCS} lines will otherwise occur. Each of the four chip-select lines is active for one of the four equal contiguous divisions of the mid-range block. Thus, if the total block size is 23K, each chip select is active for 8K of memory with $\overline{MCS0}$ being active for the first range and $\overline{MCS3}$ being active for the last range.

The EX and MS in MPCS relate to peripheral functionality as described in a later section.

Table 9. MMCS Programming Values

Total Block Size	Individual Select Size	MMCS Bits 14-8
8K	2K	000001B
16K	4K	0000010B
32K	8K	0000100B
64K	16K	0001000B
128K	32K	0010000B
256K	64K	0100000B
512K	128K	1000000B

The base address of the mid-range memory block is defined by bits 15-9 of the MMCS register (see Figure 13). This register is at offset A6H in the internal control block. These bits correspond to bits A19-A13 of the 20-bit memory address. Bits A12-A0 of the base address area always 0. The base address may be set at any integer multiple of the size of the total memory block selected. For example, if the midrange block size is 32K (or the size of the block for which each MCS line is active is 8K), the block could be located at 10000H or 18000H, but not at 14000H, since the first few integer multiples of a 32K memory block are 0H, 8000H, 10000H, 18000H, etc. After reset, the contents of both of these registers is undefined. However, none of the MCS lines will be active until both the MMCS and MPCS registers are accessed.

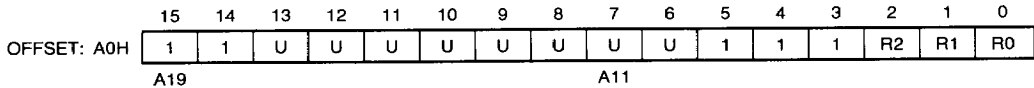


Figure 10. UMCS Register

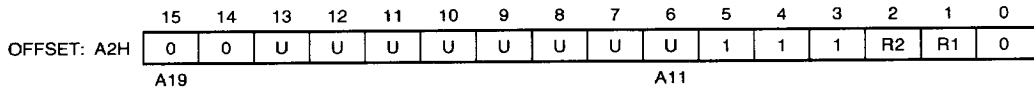


Figure 11. LMCS Register

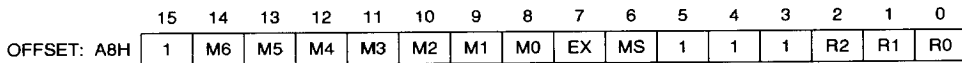


Figure 12. MPCS Register

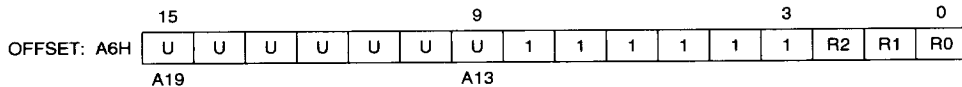


Figure 13. MMCS Register

MMCS bits R2-R0 specify READY mode of operation for all mid-range chip selects. All devices in mid-range memory must use the same number of WAIT states.

The 512K block size for the mid-range memory chip selects is a special case. When using 512K, the base address would have to be at either locations 00000H or 80000H. If it were to be programmed at 00000H when the LCS line was programmed, there would be an internal conflict between the LCS ready generation logic and the MCS ready generation logic. Likewise, if the base address were programmed at 80000H, there would be a conflict with the UCS ready generation logic. Since the LCS chip-select line does not become active until programmed, while the UCS line is active at reset, the memory base can be set only at 00000H. If this base address is selected, however, the LCS range must not be programmed.

Peripheral Chip Selects

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

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

PCS5 and PCS6 can also be programmed to provide latched address bits A1, A2. If so programmed, they cannot be used as peripheral selects. These outputs can be connected directly to the A0, A1 pins used for selecting internal registers of 8-bit peripheral chips. This scheme simplifies the hardware interface because the 8-bit registers of peripherals are simply

treated as 16-bit registers located on even boundaries in I/O space or memory space where only the lower 8-bits of the register are significant: the upper 8-bits are "don't cares."

The starting address of the peripheral chip-select block is defined by the PACS register (see Figure 14). This register is located at offset A4H in the internal control block. Bits 15-6 of this register correspond to bits 19-10 of the 20-bit Programma-

ble Base Address (PBA) of the peripheral chip-select block. Bits 9-0 of the PBA of the peripheral chip-select block are all zeros. If the chip-select block is located in I/O space, bits 12-15 must be programmed zero, since the I/O address is only 16 bits wide. Table 10 shows the address range of each peripheral chip select with respect to the PBA contained in PACS register.

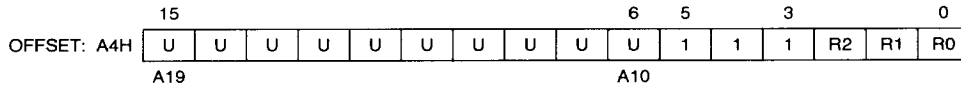


Figure 14. PACS Register

The user should program bits 15-6 to correspond to the desired peripheral base location. PACS bits 0-2 are used to specify READY mode for PCS0-PCS3.

Table 10. PCS Address Ranges

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

The mode of operation of the peripheral chip selects is defined by the MPCS register (which is also used to set the size of the mid-range memory chip-select block, see Figure 15). This register is located at offset A8H in the internal control block. Bit 7 is used to select the function of PCS5 and PCS6, while bit 6 is used to select whether the peripheral chip selects are mapped into memory or I/O space. Table 11 describes the programming of these bits. After reset, the contents of both the MPCS and the PACS registers are undefined, however none of the PCS lines will be active until both of the MPCS and PACS registers are accessed.

Table 11. MS, EX Programming Values

Bit	Description
MS	1 = Peripherals mapped into memory space. 0 = Peripherals mapped into I/O space.
EX	0 = 5 PCS lines. A1, A2 provided. 1 = 7 PCS lines. A1, A2 are not provided.

MPCS bits 0-2 are used to specify READY mode for PCS4-PCS6 as outlined below.

READY Generation Logic

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

READY control consists of 3 bits for each CS line or group of lines generated by the 80188. The interpretation of the ready bits is shown in Table 12.

Table 12. READY Bits Programming

R2	R1	R0	Number of WAIT States Generated
0	0	0	0 wait states, external RDY also used.
0	0	1	1 wait state inserted, external RDY also used.
0	1	0	2 wait states inserted, external RDY also used.
0	1	1	3 wait states inserted, external RDY also used.
1	0	0	0 wait states, external RDY ignored.
1	0	1	1 wait state inserted, external RDY ignored.
1	1	0	2 wait states inserted, external RDY ignored.
1	1	1	3 wait states inserted, external RDY ignored.

The internal ready generator operates in parallel with external READY, not in series if the external READY is used (R2 = 0). This means, for example, if the internal generator is set to insert two wait states, but activity on the external READY lines will insert four wait states, the processor will only insert four wait states, not six. This is because the two wait states generated by the internal generator overlapped the first two wait states generated by the external ready signal. Note that the external ARDY and SRDY lines are always ignored during cycles accessing internal peripherals.

R2-R0 of each control word specifies the READY mode for the corresponding block, with the exception of the peripheral chip selects: R2-R0 of PACS set the PCS0-3 READY mode, R2-R0 of MPCS set the PCS4-6 READY mode.

Chip Select/Ready Logic and Reset

Upon reset, the Chip-Select/Ready Logic will perform the following actions:

- All chip-select outputs will be driven HIGH.
- Upon leaving RESET, the UCS line will be programmed to provide chip selects to a 1K block with the accompanying READY control bits set at 011 to allow the maximum number of internal wait states in conjunction with external Ready consideration (i.e. UMCS resets to FFFBH).

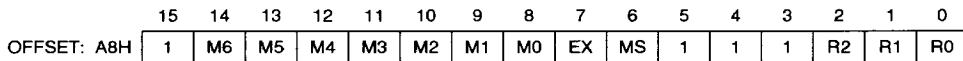


Figure 15. MPCS Register

- No other chip select or READY control registers have any predefined values after RESET. They will not become active until the CPU accesses their control registers. Both the PACS and MPCS registers must be accessed before the PCS lines will become active.

DMA CHANNELS

The 80188 DMA controller provides two independent high-speed DMA channels. Data transfers can occur between memory and I/O spaces (e.g., Memory to I/O) or within the same space (e.g., Memory to Memory or I/O to I/O). Data can be transferred either in bytes (8 bits) or in words (16 bits) to or from even or odd addresses. Each DMA channel maintains both a 20-bit source and destination pointer which can be optionally incremented or decremented after each data transfer (by one or two depending on byte or word transfers). Each data transfer consumes two bus cycles (a minimum of eight clocks), one cycle to fetch data and the other to store data. This provides a maximum data transfer rate of one Mword/sec or 2 MBytes/sec.

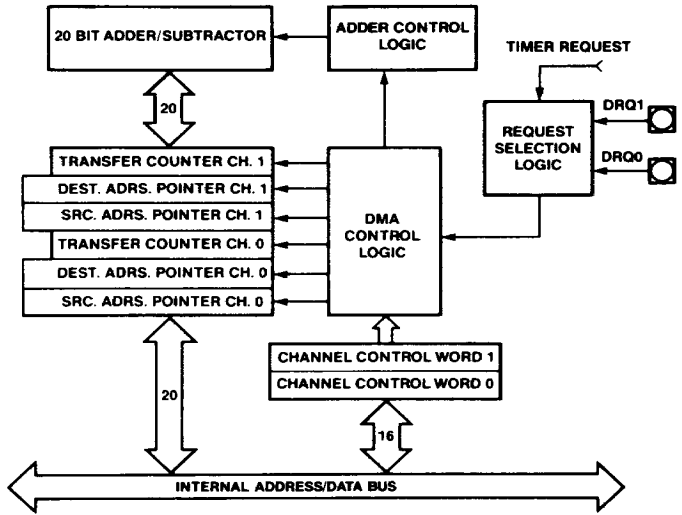
DMA Operation

Each channel has six registers in the control block which define each channel's specific operation. The control registers

consist of a 20-bit Source pointer (2 words), a 20-bit Destination pointer (2 words), a 16-bit Transfer Counter, and a 16-bit Control Word. The format of the DMA Control Blocks is shown in Table 13. The Transfer Count Register (TC) specifies the number of DMA transfers to be performed. Up to 64K byte or word transfers can be performed with automatic termination. The Control Word defines the channel's operation (see Figure 17). All registers may be modified or altered during any DMA activity. Any changes made to these registers will be reflected immediately in DMA operation.

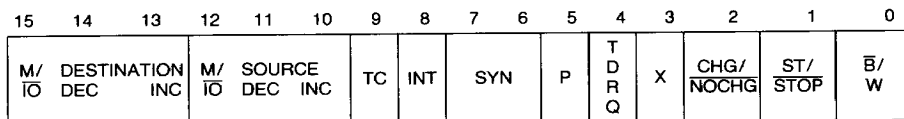
Table 13. DMA Control Block Format

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



BD003570

Figure 16. DMA Unit Block Diagram



X = DON'T CARE

Figure 17. DMA Control Register

DMA Channel Control Word Register

Each DMA Channel Control Word determines the mode of operation for the particular 80188 DMA channel. This register specifies:

- the mode of synchronization;
- whether bytes or words will be transferred;
- whether interrupts will be generated after the last transfer;
- whether DMA activity will cease after a programmed number of DMA cycles;
- the relative priority of the DMA channel with respect to the other DMA channel;
- whether the source pointer will be incremented, decremented, or maintained constant after each transfer;
- whether the source pointer addresses memory or I/O space;
- whether the destination pointer will be incremented, decremented, or maintained constant after each transfer; and
- whether the destination pointer will address memory or I/O space.

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

DMA Control Word Bit Descriptions

- B/W:** Byte/Word (0/1) Transfers.
- ST/STOP:** Start/stop (1/0) Channel.
- CHG/NOCHG:** Change/Do not change (1/0) ST/STOP bit. If this bit is set when writing to the control word, the ST/STOP bit will be programmed by the write to the control word. If this bit is cleared when writing the control word, the ST/STOP bit will not be altered. This bit is not stored; it will always be a 0 on read.
- INT:** Enable Interrupts to CPU on byte count termination.
- TC:** If set, DMA will terminate when the contents of the Transfer Count register reach zero. The ST/STOP bit will also be reset at this point if TC is set. If this bit is cleared, the DMA unit will decrement the transfer count register for each DMA cycle, but the DMA transfer will not stop when the contents of the TC register reach zero.

- SYN:** (2 bits)
- 00 No synchronization
 - NOTE:** The ST bit will be cleared automatically when the contents of the TC register reach zero regardless of the state of the bit.
 - 01 Source synchronization.
 - 10 Destination synchronization.
 - 11 Unused.
- SOURCE: INC** Increment source pointer by 1 or 2 (depends on B/W) after each transfer.
- M/I/O** Source pointer is in M/I/O space (1/0).
- DEC** Decrement source pointer by 1 or 2 (depends on B/W) after each transfer.
- DEST: INC** Increment destination pointer by 1 or 2 (B/W) after each transfer.
- M/I/O** Destination pointer is in M/I/O space (1/0).
- DEC** Decrement destination pointer by 1 or 2 (depending on B/W) after each transfer.
- P** Channel priority-relative to other channel.
- 0 low priority.
 - 1 high priority.
- Channels will alternate cycles if both set at same priority level.
- TDRQ** 0: Disable DMA requests from timer 2.
1: Enable DMA requests from timer 2.
- Bit 3** Bit 3 is not used.
- If both INC and DEC are specified for the same pointer, the pointer will remain constant after each cycle.

DMA Destination and Source Pointer Registers

Each DMA channel maintains a 20-bit source and a 20-bit destination pointer. Each of these pointers takes up two full 16-bit registers in the peripheral control block. The lower four bits of the upper register contain the upper four bits of the 20-bit physical address (see Figure 18). These pointers may be individually incremented or decremented after each transfer. If word transfers are performed the pointer is incremented or decremented by two. Each pointer may point into either memory or I/O space. Since the DMA channels can perform transfers to or from odd addresses, there is no restriction on values for the pointer registers. Higher transfer rates can be obtained if all word transfers are performed to even ad-

addresses, since this will allow data to be accessed in a single memory access.

DMA Transfer Count Register

Each DMA channel maintains a 16-bit transfer count register (TC). This register is decremented after every DMA cycle, regardless of the state of the TC bit in the DMA Control Register. If the TC bit in the DMA control word is set, however, DMA activity will terminate when the transfer count register reaches zero.

DMA Requests

Data transfers may be either source or destination synchronized, that is either the source of the data or the destination of the data may request the data transfer. In addition, DMA transfers may be unsynchronized; that is, the transfer will take place continually until the correct number of transfers has occurred. When source or unsynchronized transfers are performed, the DMA channel may begin another transfer immediately after the end of a previous DMA transfer. This allows complete transfer to take place every 2 bus cycles or eight clock cycles (assuming no wait states). No prefetching occurs

when destination synchronization is performed, however. Data will not be fetched from the source address until the destination device signals that it is ready to receive it. When destination synchronized transfers are requested, the DMA controller will relinquish control of the bus after every transfer. If no other bus activity is initiated, another DMA cycle will begin after two processor clocks. This is done to allow the destination device time to remove its request if another transfer is not desired. Since the DMA controller will relinquish the bus, the CPU can initiate a bus cycle. As a result, a complete bus cycle will often be inserted between destination synchronized transfers. These lead to the maximum DMA transfer rates shown in Table 14.

Table 14. Maximum DMA Transfer Rates with 8 MHz 80188

Type of Synchronization Selected	CPU Running	CPU Halted
Unsynchronized	1 Mbytes/sec	1 Mbytes/sec
Source Synch	1 Mbytes/sec	1 Mbytes/sec
Destination Synch	.65 Mbytes/sec	.75 Mbytes/sec

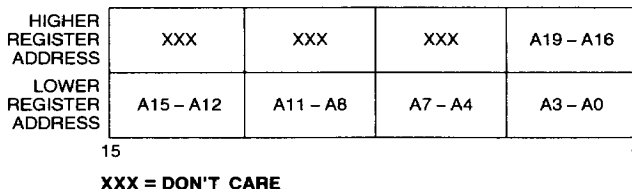


Figure 18. DMA Memory Pointer Register Format

DMA Acknowledge

No explicit DMA acknowledge pulse is provided. Since both source and destination pointers are maintained, a read from a requesting source, or a write to a requesting destination, should be used as the DMA acknowledge signal. Since the chip-select lines can be programmed to be active for a given block of memory or I/O space, and the DMA pointers can be programmed to point to the same given block, a chip-select line could be used to indicate a DMA acknowledge.

DMA Priority

The DMA channels may be programmed such that one channel is always given priority over the other, or they may be programmed such as to alternate cycles when both have DMA requests pending. DMA cycles always have priority over internal CPU cycles except between locked memory accesses or word accesses the odd memory locations; however, an external bus hold takes priority over an internal DMA cycle. Because an interrupt request cannot suspend a DMA operation and the CPU cannot access memory during a DMA cycle, interrupt latency time will suffer during sequences of continuous DMA cycles. An NMI request, however, will cause all internal DMA activity to halt. This allows the CPU to quickly respond to the NMI request.

DMA Programming

DMA cycles will occur whenever the ST/STOP bit of the Control Register is set. If synchronized transfers are programmed, a

DRQ must also have been generated. Therefore, the source and destination transfer pointers, and the transfer count register (if used) must be programmed before this bit is set.

Each DMA register may be modified while the channel is operating. If the CHG/NOCHG bit is cleared when the control register is written, the ST/STOP bit of the control register will not be modified by the write. If multiple channel registers are modified, it is recommended that a LOCKED string transfer be used to prevent a DMA transfer from occurring between updates to the channel registers.

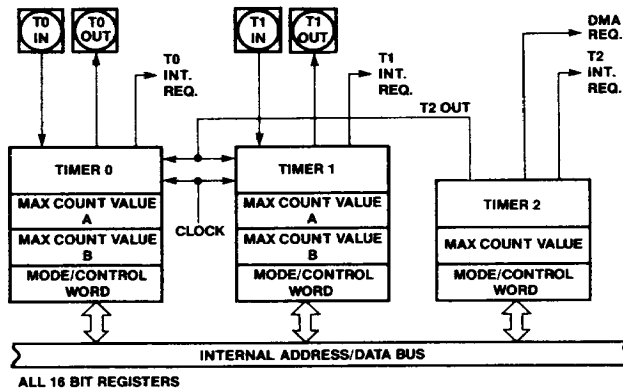
DMA Channels and Reset

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

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

TIMERS

The 80188 provides three internal 16-bit programmable timers (see Figure 19). Two of these are highly flexible and are connected to four external pins (2 per timer). They can be used to count external events, time external events, generate nonrepetitive waveforms, etc. The third timer is not connected to any external pins, and is useful for real-time coding and time delay applications. In addition, this third timer can be used as a prescaler to the other two, or as a DMA request source.



BD003580

Figure 19. Timer Block Diagram

Timer Operation

The timers are controlled by 11 16-bit registers in the internal peripheral control block. The configuration of these registers is shown in Table 15. The count register contains the current value of the timer. It can be read or written at any time independent of whether the timer is running or not. The value of this register will be incremented for each timer event. Each of the timers is equipped with a MAX COUNT register, which defines the maximum count the timer will reach. After reaching the MAX COUNT register value, the timer count value will reset to zero during that same clock, i.e., the maximum count value is never stored in the count register itself. Timers 0 and 1 are, in addition, equipped with a second MAX COUNT register, which enables the timers to alternate their count between two different MAX COUNT values programmed by the user. If a single MAX COUNT register is used, the timer output pin will switch LOW for a single clock, two clocks after the maximum count value has been reached. In the dual MAX COUNT register mode, the output pin will indicate which MAX COUNT register is currently in use, thus allowing nearly complete freedom in selecting waveform duty cycles. For the timers with two MAX COUNT registers, the RIU bit in the control register determines which is used for the comparison.

Each timer gets serviced every fourth CPU-clock cycle, and thus can operate at speeds up to one-quarter the internal clock frequency (one-eighth the crystal rate). External clocking of the timers may be done at up to a rate of one-quarter of the internal CPU-clock rate (2 MHz for an 8 MHz CPU clock). Due to internal synchronization and pipelining of the timer circuitry, a timer output may take up to six clocks to respond to any individual clock or gate input.

Since the count registers and the maximum count registers are all 16-bits wide, 16 bits of resolution are provided. Any Read or Write access to the timers will add one wait state to the minimum four-clock bus cycle. However, this is needed to synchronize and coordinate the internal data flows between the internal timers and the internal bus.

The timers have several programmable options.

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

These options are selectable via the timer mode/control word.

Timer Mode/Control Register

The mode/control register (see Figure 20) allows the user to program the specific mode of operation or check the current programmed status for any of the three integrated timers.

Table 15. Timer Control Block format

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

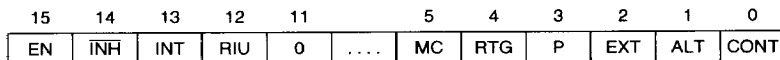


Figure 20. Timer Mode/Control Register

ALT:

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

CONT:

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

EXT:

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

P:

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

RTG:

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

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

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

EN:

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

INH:

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

INT:

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

MC:

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

RIU:

The Register In Use bit indicates which MAX COUNT register is currently being used for comparison to the timer count value. A zero value indicates register A. The RIU bit cannot be written, i.e., its value is not affected when the control register is written. It is always cleared when the ALT bit is zero.

Not all mode bits are provided for timer 2. Certain bits are hardwired as indicated below:

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

Count Registers

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

Max Count Registers

Timers 0 and 1 have two MAX COUNT registers, while timer 2 has a single MAX COUNT register. These contain the number of events the timer will count. In timers 0 and 1, the MAX COUNT register used can alternate between the two max count values whenever the current maximum count is reached. The condition which causes a timer to reset is equivalent between the current count value and the max count being used. This means that if the count is changed to be above the max count value, or if the max count value is changed to be below the current value, the timer will not reset to zero, but rather will count to its maximum value, "wrap around" to zero, then count until the max count is reached.

Timers and Reset

Upon RESET, the Timers will perform the following actions:

- All EN (Enable) bits are reset preventing timer counting.
- All SEL (Select) bits are reset to zero. This selects MAX COUNT register A, resulting in the Timer Out pins going HIGH upon RESET.

INTERRUPT CONTROLLER

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

The interrupt controller will resolve priority among requests that are pending simultaneously. Nesting is provided so interrupt service routines for lower priority interrupts may themselves be interrupted by higher priority interrupts. A block diagram of the interrupt controller is shown in Figure 21.

The interrupt controller has a special iRMX 86 compatibility mode that allows the use of the 80188 within the iRMX 86 operating system interrupt structure. The controller is set in this mode by setting bit 14 in the peripheral control block relocation register (see iRMX 86 Compatibility Mode section). In this mode, the internal 80188 interrupt controller functions as a "slave" controller to an external "master" controller. Special initialization software must be included to properly set up the 80188 interrupt controller in iRMX 86 mode.

MASTER (NON-iRMX) MODE OPERATION

Interrupt Controller External Interface

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

External sources in the cascade mode use externally generated interrupt vectors. When an interrupt is acknowledged, two \overline{INTA} cycles are initiated and the vector is read into the 80188 on the second cycle. The capability to interface to external 8259A programmable interrupt controllers is thus provided when the inputs are configured in cascade mode.

Interrupt Controller Modes of Operation

The basic modes of operation of the interrupt controller in non-iRMX mode are similar to the 8259A. The interrupt controller responds identically to internal interrupts in all three modes: the difference is only in the interpretation of function of the

four external interrupt pins. The interrupt controller is set into one of these three modes by programming the correct bits in the $\overline{INT0}$ and $\overline{INT1}$ control registers. The modes of interrupt controller operation are as follows:

Fully Nested Mode

When in the fully nested mode four pins are used as direct interrupt requests. The vectors for these four inputs are generated internally. An in-service bit is provided for every interrupt source. If a lower-priority device requests an interrupt while the in-service bit (IS) is set, no interrupt will be generated by the interrupt controller. In addition, if another interrupt request occurs from the same interrupt source while the in-service bit is set, no interrupt will be generated by the interrupt controller. This allows interrupt service routines to operate with interrupts enabled without being themselves interrupted by lower-priority interrupts. Since interrupts are enabled, higher-priority interrupts will be serviced.

When a service routine is completed, the proper IS bit must be reset by writing the proper pattern to the EOI register. This is required to allow subsequent interrupts from this interrupt source and to allow servicing of lower-priority interrupts. An EOI command is issued at the end of the service routine just before the issuance of the return from interrupt instruction. If the fully nested structure has been upheld, the next highest-priority source with its IS bit set is then serviced.

Cascade Mode

The 80188 has four interrupt pins and two of them have dual functions. In the fully nested mode the four pins are used as direct interrupt inputs and the corresponding vectors are generated internally. In the cascade mode, the four pins are configured into interrupt input-dedicated acknowledge signal pairs. The interconnection is shown in Figure 22. $\overline{INT0}$ is an interrupt input interfaced to an 8259A, while $\overline{INT2}/\overline{INTA0}$ serves as the dedicated interrupt acknowledge signal to that peripheral. The same is true for $\overline{INT1}$ and $\overline{INT3}/\overline{INTA1}$. Each pair can selectively be placed in the cascade or non-cascade mode by programming the proper value into $\overline{INT0}$ and $\overline{INT1}$ control registers. The use of the dedicated acknowledge signals eliminates the need for the use of external logic to generate \overline{INTA} and device select signals.

The primary cascade mode allows the capability to serve up to 128 external interrupt sources through the use of external master and slave 8259As. Three levels of priority are created, requiring priority resolution in the 80188 interrupt controller, the master 8259As, and the slave 8259As. If an external interrupt is serviced, one IS bit is set at each of these levels. When the interrupt service routine is completed, up to three end-of-interrupt commands must be issued by the programmer.

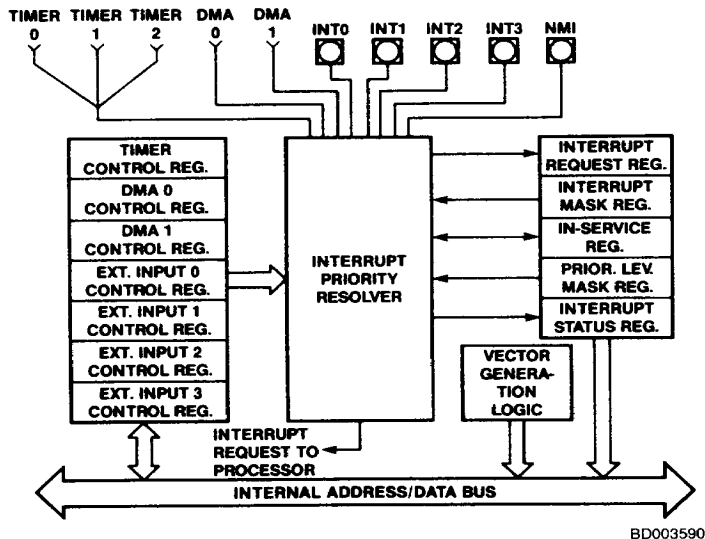


Figure 21. Interrupt Controller Block Diagram

Special Fully Nested Mode

This mode is entered by setting the SFNM bit in INT0 or INT1 control register. It enables complete nestability with external 8259A masters. Normally, an interrupt request from an interrupt source will not be recognized unless the in-service bit for that source is reset. If more than one interrupt source is connected to an external interrupt controller, all of the interrupts will be funneled through the same 80188 interrupt request pin. As a result, if the external interrupt controller receives a higher-priority interrupt, its interrupt will not be recognized by the 80188 controller until the 80188 in-service bit is reset. In special fully nested mode, the 80188 interrupt controller will allow interrupts from an external pin regardless of the state of the in-service bit for an interrupt source in order to allow multiple interrupts from a single pin. An in-service bit will continue to be set, however, to inhibit interrupts from other lower-priority 80188 interrupt sources.

Special procedures should be followed when resetting IS bits at the end of interrupt service routines. Software polling of the external master's IS register is required to determine if there is more than one bit set. If so, the IS bit in the 80188 remains active and the next interrupt service routine is entered.

Operation in a Polled Environment

The controller may be used in a polled mode if interrupts are undesirable. When polling, the processor disables interrupts and then polls the interrupt controller whenever it is convenient. Polling the interrupt controller is accomplished by reading the Poll Word (Figure 9), bit 15 in the poll word indicates to the processor that an interrupt of high enough priority is requesting service. Reading the Poll Word causes the In-Service bit of the highest-priority source to be set.

It is desirable to be able to read the Poll Word information without guaranteeing service of any pending interrupt, i.e., not set the indicated in-service bit. The 80188 provides a Poll

Status Word in addition to the conventional Poll Word to allow this to be done. Poll Word information is duplicated in the Poll Status Word, but reading the Poll Status Word does not set the associated in-service bit. These words are located in two adjacent memory locations in the register file.

Master (NON-iRMX) Mode Features

Programmable Priority

The user can program the interrupt sources into any of eight different priority levels. The programming is done by placing a 3-bit priority level (0-7) in the control register of each interrupt source. (A source with a priority level of 4 has higher priority over all priority levels from 5 to 7. Priority registers containing values lower than 4 have greater priority.) All interrupt sources have preprogrammed default priority levels (see Table 4).

If two requests with the same programmed priority level are pending at once, the priority ordering scheme shown in Table 4 is used. If the serviced interrupt routine reenables interrupts, it allows other requests to be serviced.

End-of-Interrupt Command

The end-of-interrupt (EOI) command is used by the programmer to reset the In-Service (IS) bit when an interrupt service routine is completed. The EOI command is issued by writing the proper pattern to the EOI register. There are two types of EOI commands, specific and nonspecific. The nonspecific command does not specify which IS bit is reset. When issued, the interrupt controller automatically resets the IS bit of the highest priority source with an active service routine. A specific EOI command requires that the programmer send the interrupt vector type to the interrupt controller indicating which source's IS bit is to be reset. This command is used when the fully nested structure has been disturbed or the highest priority IS bit that was set does not belong to the service routine in progress.

Trigger Mode

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

Interrupt Vectoring

The 80188 Interrupt Controller will generate interrupt vectors for the integrated DMA channels and the integrated Timers. In addition, the Interrupt Controller will generate interrupt vectors for the external interrupt lines if they are not configured in Cascade or Special Fully Nested Mode. The interrupt vectors generated are fixed and cannot be changed (see Table 4).

Interrupt Controller Registers

The Interrupt Controller register model is shown in Figure 23. It contains 15 registers. All registers can both be read or written unless specified otherwise.

In-Service Register

This register can be read from or written into. The format is shown in Figure 24. It contains the In-Service bit for each of the interrupt sources. The In-Service bit for each of the interrupt sources. The In-Service bit is set to indicate that a source's service routine is in progress. When an In-Service bit is set, the interrupt controller will not generate interrupts to the CPU when it receives interrupt requests from devices with a

lower programmed priority level. The TMR bit is the In-Service bit for all three timers; the D0 and D1 bits are the In-Service bits for the two DMA channels; the 10-13 are the In-Service bits for the external interrupt pins. The IS bit is set when the processor acknowledges an interrupt request either by an interrupt acknowledge or by reading the poll register. The IS bit is reset at the end of the interrupt service routine by an end-of-interrupt command issued by the CPU.

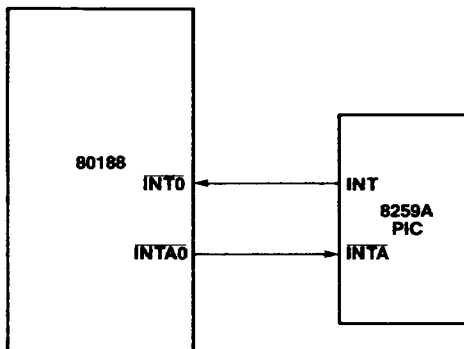
Interrupt Request Register

The internal interrupt sources have interrupt request bits inside the interrupt controller. The format of this register is shown in Figure 24. A read from this register yields the status of these bits. The TMR bit is the logical OR of all timer interrupt requests. D0 and D1 are the interrupt request bits for the DMA channels.

The state of the external interrupt pins is also indicated. The state of the external interrupt pins is not a stored condition inside the interrupt controller, therefore the external interrupt bits cannot be written. The external interrupt request bits show exactly when an interrupt request is given to the interrupt controller, so if edge-triggered mode is selected, the bit in the register will be HIGH only after an inactive-to-active transition. For internal interrupt sources, the register bits are set when a request arrives and are reset when the processor acknowledges the requests.

Mask Register

This is a 16-bit register that contains a mask bit for each interrupt source. The format for this register is shown in Figure 24. A one in a bit position corresponding to a particular source serves to mask the source from generating interrupts. These mask bits are the exact same bits which are used in the individual control registers; programming a mask bit using the mask register will also change this bit in the individual control registers, and vice versa.



AF002801

Figure 22. Cascade Mode Interrupt Connection

Timer, DMA 0, 1; Control Registers

These registers are the control words for all the internal interrupt sources. The format for these registers is shown in Figure 27. The three bit positions PR0, PR1, and PR2 represent the programmable priority level of the interrupt source. The MSK bit inhibits interrupt requests from the interrupt source. The MSK bits in the individual control registers are the exact same bits as are in the Mask Register; modifying them in the individual control registers will also modify them in the Mask Register, and vice versa.

INT0-INT3 Control Registers

These registers are the control words for the four external input pins. Figure 28 shows the format of the INT0 and INT1 Control registers; Figure 29 shows the format of the INT2 and INT3 Control registers. In cascade mode or special fully nested mode, the control words for INT2 and INT3 are not used.

The bits in the various control registers are encoded as follows:

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

LTM: Level-trigger mode bit. 1 = level-triggered; 0 = edge-triggered. Interrupt Input levels are active high. In level-triggered mode, an interrupt is generated whenever the external line is high.

In edge-triggered mode, an interrupt will be generated only when this level is preceded by an inactive-to-active transition on the line. In both cases, the level must remain active until the interrupt is acknowledged.

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

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

SFNM: Special fully nested mode bit, 1 = SFNM; 0 = normal nested mode.

EOI Register

The end of the interrupt register is a command register which can only be written into. The format of this register is shown in Figure 30. It initiates an EOI command when written to by the 80188 CPU.

The bits in the EOI register are encoded as follows:

Sx: Encoded information that specifies an interrupt source vector type as shown in Table 4. For example, to reset the In-Service bit for DMA channel 0, these bits should be set to 01010, since the vector type for DMA channel 0 is 10. Note that to reset the single In-Service bit for any of the three timers, the vector type for timer 0(8) should be written in this register.

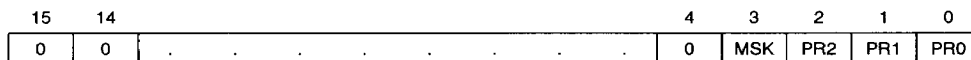


Figure 27. Timer/DMA Control Register Formats

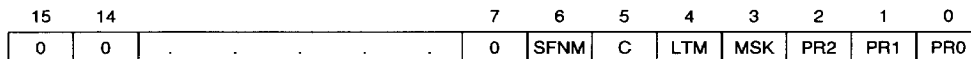


Figure 28. INT0/INT1 Control Register Formats

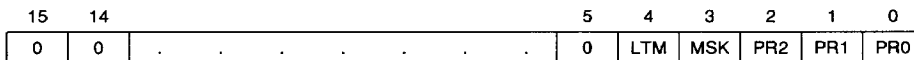


Figure 29. INT2/INT3 Control Register Formats

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

Poll and Poll Status Registers

These registers contain polling information. The format of these registers is shown in Figure 31. They can only be read. Reading the Poll register constitutes a software poll. This will set the IS bit of the highest priority pending interrupt. Reading the poll status register will not set the IS bit of the highest priority pending interrupt; only the status of pending interrupts will be provided.

Encoding of the Poll and Poll Status register bits are as follows:

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

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

iRMX 86 COMPATIBILITY MODE

This mode allows iRMX 86-80188 compatibility. The interrupt model of iRMX 86 requires one master and multiple slave 8259As in cascaded fashion. When iRMX mode is used, the internal 80188 interrupt controller will be used as a slave controller to an external master interrupt controller. The internal 80188 resources will be monitored through the internal interrupt controller, while the external controller functions as the system master interrupt controller.

Upon reset, the 80188 interrupt controller will be in the non-iRMX 86 mode of operation. To set the controller in the iRMX 86 mode, bit 14 of the Relocation Register should be set.

Because of pin limitations caused by the need to interface to an external 8259A master, the internal interrupt controller will no longer accept external inputs. There are however, enough 80188 interrupt controller inputs (internally) to dedicate one to each timer. In this mode, each timer interrupt source has its own mask bit, IS bit, and control word.

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

levels for each source. The required priority levels for the internal interrupt sources in iRMX mode are shown in Table 16.

Table 16. Internal Source Priority Level

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

These level assignments must remain fixed in the iRMX 86 mode of operation.

iRMX 86 Mode External Interface

The configuration of the 80188 with respect to an external 8259A master is shown in Figure 32. The INT0 input is used as the 80188 CPU interrupt input. INT3 functions as an output to send the 80188 slave-interrupt-request to one of the 8 master-PIC-inputs.

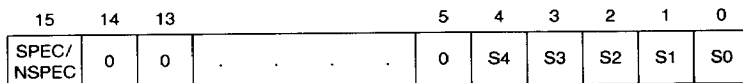
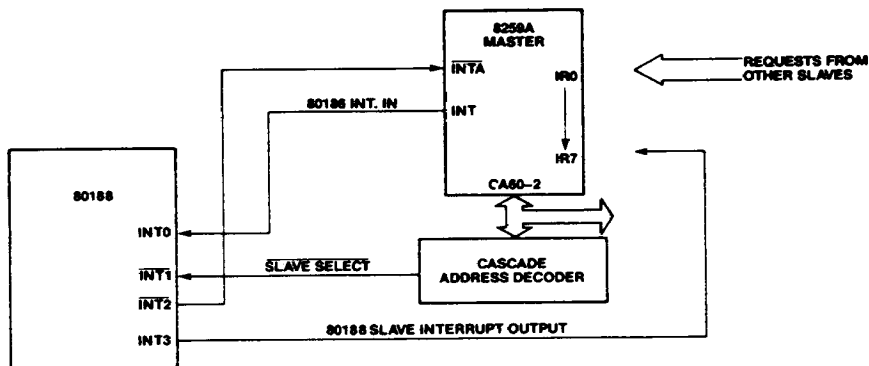


Figure 30. EOI Register Format



Figure 31. Poll Register Format



AF002811

Figure 32. iRMX 86 Interrupt Controller Interconnection

Correct master-slave interface requires decoding of the slave addresses (CAS0-2). Slave 8259As do this internally. Because of pin limitations, the 80188 slave address will have to be decoded externally. $\overline{INT1}$ is used as a slave-select input. Note that the slave vector address is transferred internally, but the READY input must be supplied externally.

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

Interrupt Nesting

iRMX 86 mode operation allows nesting of interrupt requests. When an interrupt is acknowledged, the priority logic masks off all priority levels except those with equal or higher priority.

Vector Generation in the iRMX 86 MODE

Vector generation in iRMX mode is exactly like that of an 8259A slave. The interrupt controller generates an 8-bit vector which the CPU multiplies by four and uses as an address into a vector table. The significant five bits of the vector are user-programmable while the lower three bits are generated by the priority logic. These bits represent the encoding of the priority level requesting service. The significant five bits of the vector are programmed by writing to the Interrupt Vector register at offset 20H.

Specific End-of-Interrupt

In iRMX mode the specific EOI command operates to reset an in-service bit of a specific priority. The user supplies a 3-bit priority-level value that points to an in-service bit to be reset. The command is executed by writing the correct value in the Specific EOI register at offset 22H.

Interrupt Controller Registers in the iRMX 86 Mode

All control and command registers are located inside the internal peripheral control block. Figure 33 shows the offsets of these registers.

End-of-Interrupt Register

The end-of-interrupt register is a command register which can only be written. The format of this register is shown in Figure 34. It initiates an EOI command when written by the 80188 CPU.

The bits in the EOI register are encoded as follows:

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

In-Service Register

This register can be read from or written into. It contains the in-service bit for each of the internal sources. The format for this register is shown in Figure 35. Bit positions 2 and 3 correspond to the DMA channels; positions 0, 4, and 5 correspond to the integral timers. The source's IS bit is set when the processor acknowledges its interrupt request.

Interrupt Request Register

This register indicates which internal peripherals have interrupt requests pending. The format of this register is shown in

Figure 35. The interrupt request bits are set when a request arrives from an internal source, and are reset when the processor acknowledges the request.

Mask Register

This register contains a mask bit for each interrupt source. The format for this register is shown in Figure 35. If the bit in this register corresponding to a particular interrupt source is set, any interrupts from that source will be masked. These mask bits are exactly the same bits which are used in the individual control registers, i.e., changing the state of a mask bit in this register will also change the state of the mask bit in the individual interrupt control register corresponding to the bit.

Control Registers

These registers are the control words for all the internal interrupt sources. The format of these registers is shown in Figure 36. Each of the timers and both of the DMA channels have their own Control Register.

The bits of the Control Registers are encoded as follows:

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

m_{sk} : mask bit for the priority level indicated by pr_x bits.

	OFFSET
LEVEL 5 CONTROL REGISTER (TIMER 2)	3AH
LEVEL 4 CONTROL REGISTER (TIMER 1)	38H
LEVEL 3 CONTROL REGISTER (DMA 1)	36H
LEVEL 2 CONTROL REGISTER (DMA 0)	34H
LEVEL 0 CONTROL REGISTER (TIMER 0)	32H
INTERRUPT-REQUEST REGISTER	2EH
IN-SERVICE REGISTER	2CH
PRIORITY-LEVEL MASK REGISTER	2AH
MASK REGISTER	28H
SPECIFIC EOI REGISTER	22H
INTERRUPT VECTOR REGISTER	20H

Figure 33. Interrupt Controller Registers (iRMX 86 Mode)

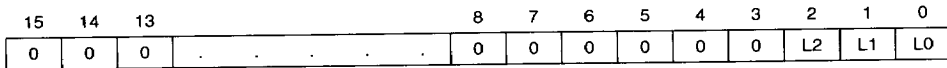


Figure 34. Specific EQI Register Format

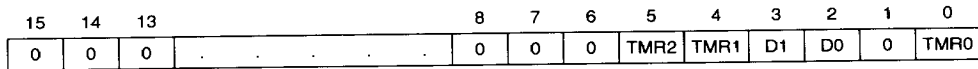


Figure 35. In-Service, Interrupt Request, and Mask Register Format

Interrupt Vector Register

This register provides the upper five bits of the interrupt vector address. The format of this register is shown in Figure 37. The interrupt controller itself provides the lower three bits of the interrupt vector as determined by the priority level of the interrupt request.

The format of the bits in this register is:

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

Priority-Level Mask Register

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

The encoding of the bits in this register is:

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

Interrupt Controller and Reset

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

- All SFNM bits reset to 0, implying Fully Nested Mode.
- All PR bits in the various control registers set to 1. This places all sources at lowest priority (level 111).
- All LTM bits reset to 0, resulting in edge-sense mode.
- All Interrupt Service bits reset to 0.
- All Interrupt Request bits reset to 0.
- All MSK (Interrupt Mask) bits set to 1 (mask).
- All C (Cascade) bits reset to 0 (non-cascade).
- All PRM (Priority Mask) bits set to 1, implying no levels masked.
- Initialized to NON-IRMX 86 mode.

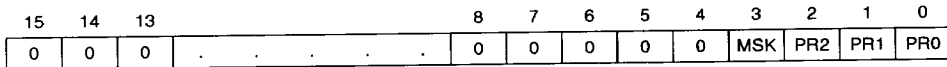


Figure 36. Control Word Format

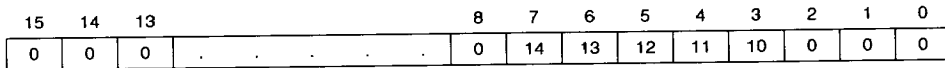


Figure 37. Interrupt Vector Register Format

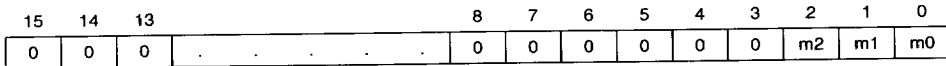
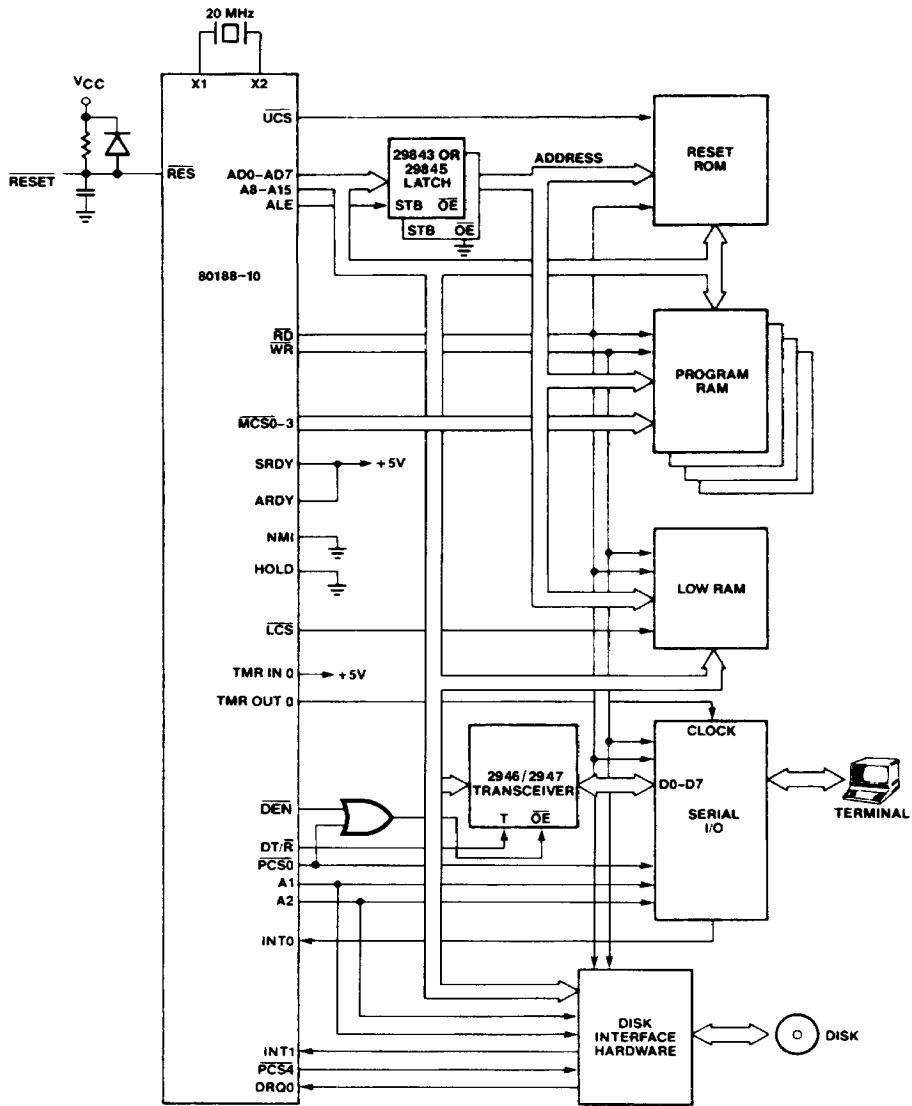
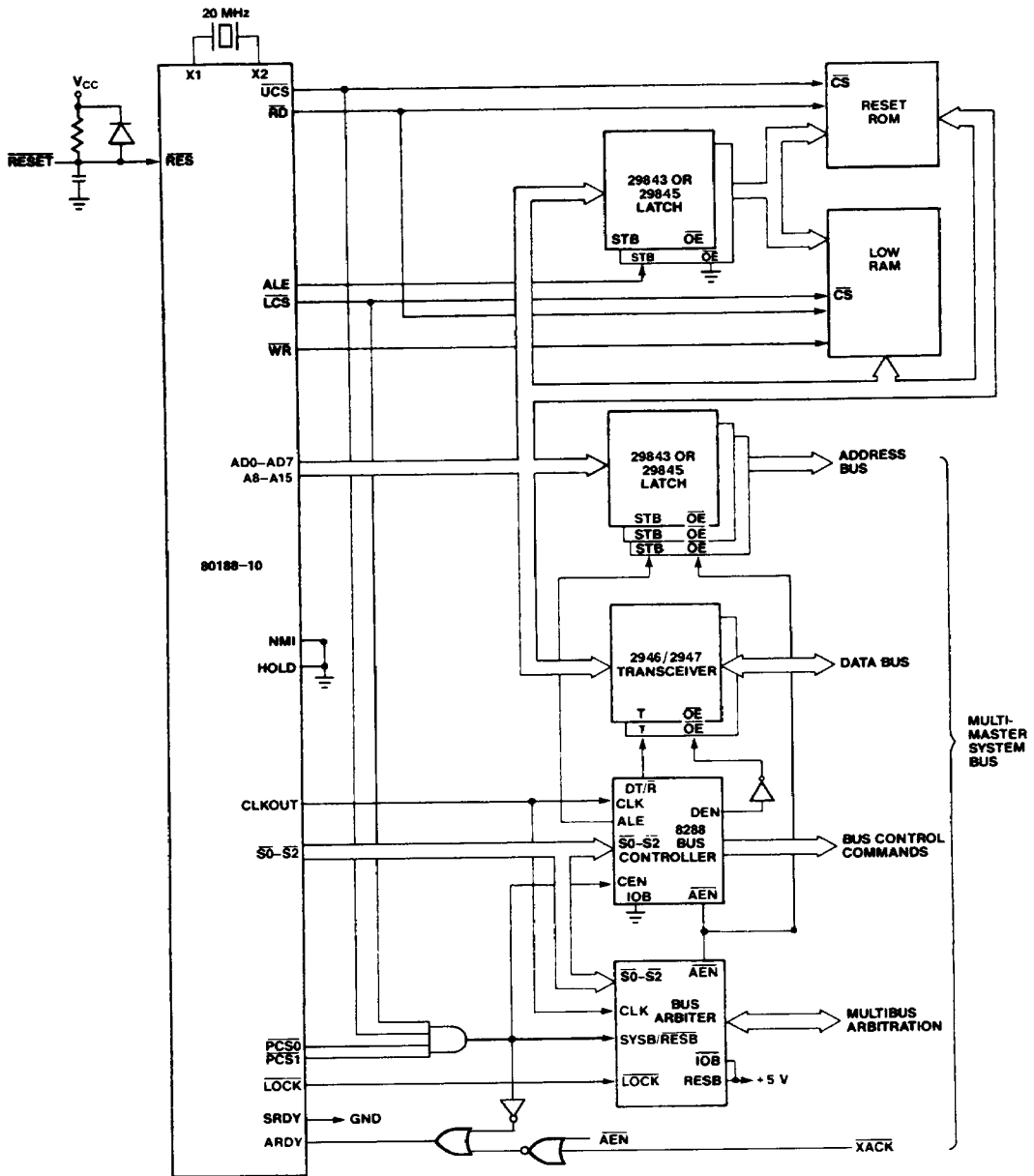


Figure 38. Priority Level Mask Register



AF002826

Figure 39. Typical 80188 Computer



AF002835

Figure 40. Typical 80188 Multi-Master Bus Interface

ABSOLUTE MAXIMUM RATINGS

Storage Temperature -65°C to +150°C
 Voltage on Any Pin with
 Respect to Ground -1.0 V to +7 V
 Power Dissipation (steady state 70°C) 2.0 W

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Commercial (C) Devices
 Temperature (T_A) 0 to +70°C
 (T_C) 0 to +110°C
 Supply Voltage (V_{CC}) 5 V ± 10%

Industrial (I) Devices
 Temperature (T_A) -40 to +85°C
 Supply Voltage (V_{CC}) 5 V ± 5%

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS (over operating range unless otherwise specified)

Parameters	Description	Test Conditions	Min.	Max.	Units
V _{IL}	Input LOW Voltage		-0.5	+0.8	Volts
V _{IH}	Input HIGH Voltage (All except X1 and RES)		2.0	V _{CC} + 0.5	Volts
V _{IH1}	Input HIGH Voltage (RES)		3.0	V _{CC} + 0.5	Volts
V _{OL}	Output LOW Voltage	I _A = 2.5 mA for S0-S2 I _A = 2.0 mA for all other outputs		0.45	Volts
V _{OH}	Output HIGH Voltage	I _{OA} = -400 μA	2.4		Volts
I _{CC}	Power Supply Current	T _A = -40°C		600**	mA
		T _A = 0°C		500	
		T _A = 70°C		375	
I _{LI}	Input Leakage Current	0 V < V _{IN} < V _{CC}		±10	μA
I _{LO}	Output Leakage Current	0.45 V < V _{OUT} < V _{CC}		±10	μA
V _{CLO}	Clock Output LOW	I _A = 4.0 mA		0.6	Volts
V _{CHO}	Clock Output HIGH	I _{OA} = -200 μA	4.0		Volts
V _{CLI}	Clock Input LOW Voltage		-0.5	0.6	Volts
V _{CHI}	Clock Input HIGH Voltage		3.9	V _{CC} + 1.0	Volts
C _{IN}	Input Capacitance			10	pF
C _{IQ}	I/O Capacitance			20	pF

SWITCHING CHARACTERISTICS

PIN TIMING

80188 Timing Requirements All Timings Measured At 1.5 Volts Unless Otherwise Noted

Parameters	Description	Test Conditions	80188-10 (10 MHz)		80188 (8 MHz)		Units
			Min.	Max.	Min.	Max.	
T _{DVCL}	Data in Setup (A/D)		15		20		ns
T _{CLDX}	Data in Hold (A/D)		8		10		ns
T _{ARYHCH}	Asynchronous Ready (AREADY) active setup time*		15		20		ns
T _{ARYLCL}	AREADY inactive setup time		25		35		ns
T _{CHARYX}	AREADY hold time		15		15		ns
T _{ARYCHL}	Asynchronous Ready inactive hold time		15		15		ns
T _{SRVCL}	Synchronous Ready (SREADY) transition setup time		20		20		ns
T _{CLSRV}	SREADY transition hold time		15		15		ns
T _{HVCL}	HOLD Setup*		20		25		ns
T _{INVCH}	INTR, NMI, TEST, TIMERIN, Setup*		25		25		ns
T _{INVCL}	DRQ0, DRQ1, Setup*		20		25		ns

*To guarantee recognition at next clock.
 For Industrial Grade Parts only.

SWITCHING CHARACTERISTICS (Cont'd.)
80188 Master Interface Timing Responses

Parameters	Description	Test Conditions	80188-10 (10 MHz)		80188 (8 MHz)		Units
			Min.	Max.	Min.	Max.	
TCLAV	Address Valid Delay	C _L = 20 – 200 pF all outputs	5	44	5	55	ns
TCLAX	Address Hold		10		10		ns
TCLAZ	Address Float Delay		TCLAX	30	TCLAX	35	ns
TCHCZ	Command Lines Float Delay			40		45	ns
TCHCV	Command Lines Valid Delay (after float)			45		55	ns
TLHLL	ALE Width		TCLCL-30		TCLCL-35		ns
TCHLH	ALE Active Delay			30		35	ns
TCHLL	ALE Inactive Delay			30		35	ns
TLLAX	Address Hold to ALE Inactive		TCHCL-20		TCHCL-25		ns
TCLDV	Data Valid Delay		10	40	10	44	ns
TCLDOX	Data Hold Time		10		10		ns
TWHDX	Data Hold after WR		TCLCL-34		TCLCL-40		ns
TCVCTV	Control Active Delay1		5	40	5	50	ns
TCHCTV	Control Active Delay2		10	44	10	55	ns
TCVCTX	Control Inactive Delay		5	44	5	55	ns
TCVDEX	DEN Inactive Delay (Non-Write Cycle)		10	56	10	70	ns
TAZRL	Address Float to RD Active		0		0		ns
TCLRRL	RD Active Delay		10	56	10	70	ns
TCLRHL	RD Inactive Delay		10	44	10	55	ns
TRHAV	RD Inactive to Address Active		TCLCL-40		TCLCL-40		ns
TCLHAV	HLDA Valid Delay		5	40	5	50	ns
TRLRH	RD Width		2TCLCL-46		2TCLCL-50		ns
TWLWH	WR Width		2TCLCL-34		2TCLCL-40		ns
TAVAL	Address Valid to ALE LOW		TCLCH-19		TCLCH-25		ns
TCHSV	Status Active Delay		10	45	10	55	ns
TCLSH	Status Inactive Delay		10	50	10	65	ns
TCLTMV	Timer Output Delay	100 pF max		48		60	ns
TCLRO	Reset Delay			48		60	ns
TCHQSV	Queue Status Delay			28		35	ns
TCHDX	Status Hold Time		10		10		ns
TAVCH	Address Valid to Clock HIGH		10		10		ns
TCLLV	LOCK Valid/Invalid Delay		5	60	5	65	ns

80188 Chip-Select Timing Responses

Parameters	Description	Test Conditions	80188-10 (10 MHz)		80188 (8 MHz)		Units
			Min.	Max.	Min.	Max.	
TCLCSV	Chip-Select Active Delay			45		66	ns
TCXCSX	Chip-Select Hold from Command Inactive		35		35		ns
TCHCSX	Chip-Select Inactive Delay		5	32	5	35	ns

SWITCHING CHARACTERISTICS (Cont'd.)
80188 CLKIN Requirements

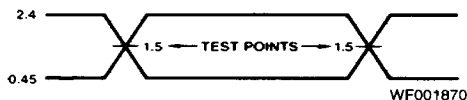
Parameters	Description	Test Conditions	80188-10 (10 MHz)		80188 (8 MHz)		Units
			Min.	Max.	Min.	Max.	
TCKIN	CLKIN Period		50	250	62.5	250	ns
TCKHL	CLKIN Fall Time	3.5 to 1.0 volts		10		10	ns
TCKLH	CLKIN Rise Time	1.0 to 3.5 volts		10		10	ns
TCLCK	CLKIN LOW Time	1.5 volts	20		25		ns
TCHCK	CLKIN HIGH Time	1.5 volts	20		25		ns

80188 CLKOUT Timing (200 pF load)

Parameters	Description	Test Conditions	Min.	Max.	Min.	Max.	Units
TCKO	CLKIN to CLKOUT Skew			25		50	ns
TCLCL	CLKOUT Period		100	500	125	500	ns
TCLCH	CLKOUT LOW Time	1.5 volts	$\frac{1}{2}T_{CLCL}-6.0$		$\frac{1}{2}T_{CLCL}-7.5$		ns
TCHCL	CLKOUT HIGH Time	1.5 volts	$\frac{1}{2}T_{CLCL}-6.0$		$\frac{1}{2}T_{CLCL}-7.5$		ns
TCH1CH2	CLKOUT Rise Time	1.0 to 3.5 volts		12		15	ns
TCL2CL1	CLKOUT Fall Time	3.5 to 1 volts		12		15	ns

All timings measured at 1.5 volts unless otherwise noted.

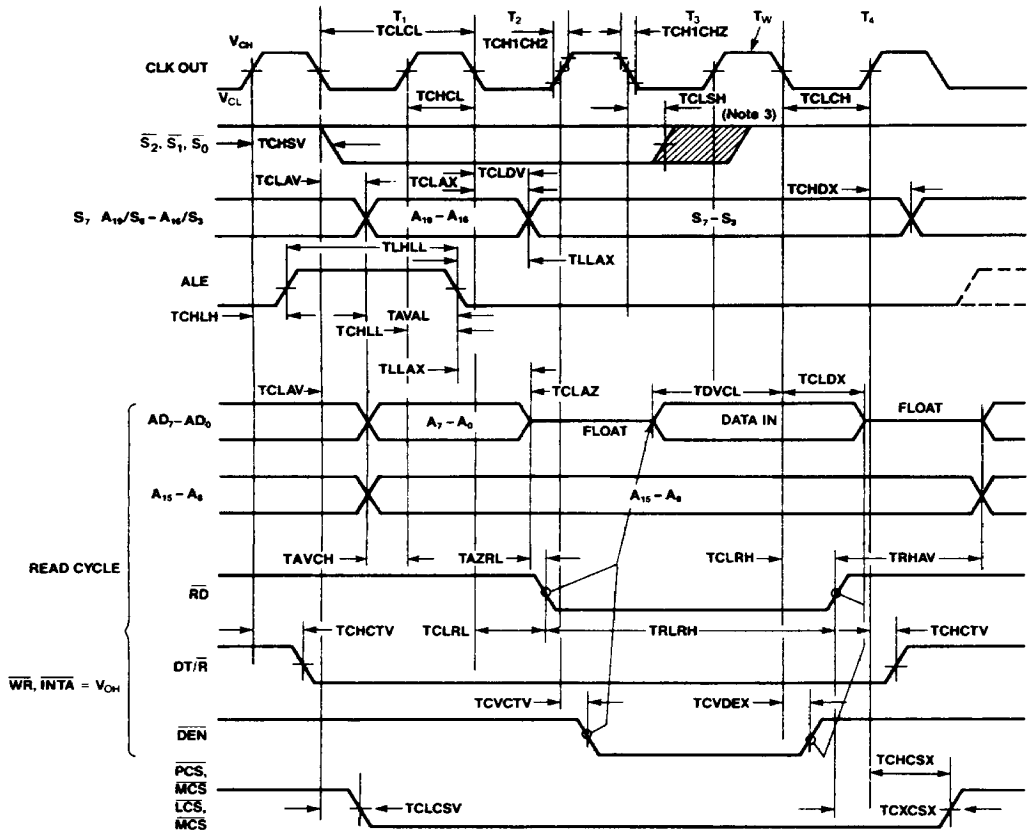
SWITCHING TEST INPUT/OUTPUT WAVEFORM



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

SWITCHING WAVEFORMS (Cont'd.)

MAJOR CYCLE TIMING (Cont'd.)

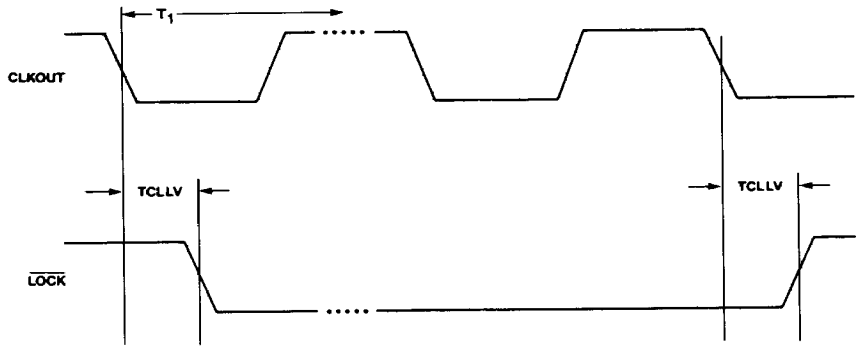


WF006226

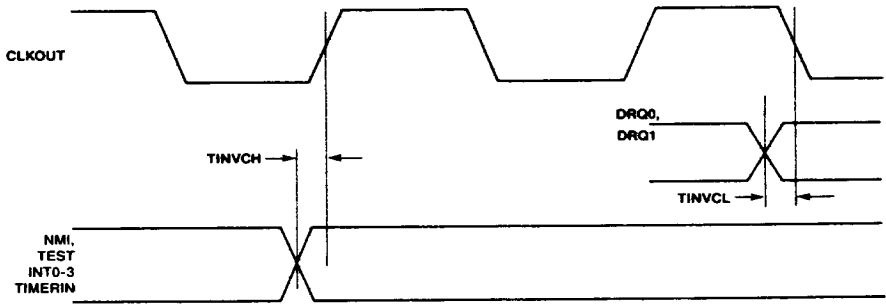
Notes:

- 1. Following a Write cycle, the Local Bus is floated by the 80188 only when the 80188 enters a "Hold Acknowledge" state.
- 2. INTA occurs one clock later in RMX-mode.
- 3. Status inactive just prior to T₄.

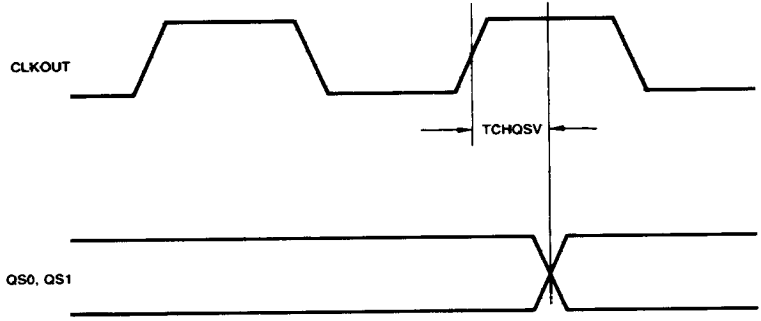
SWITCHING WAVEFORMS (Cont'd.)



WF006232

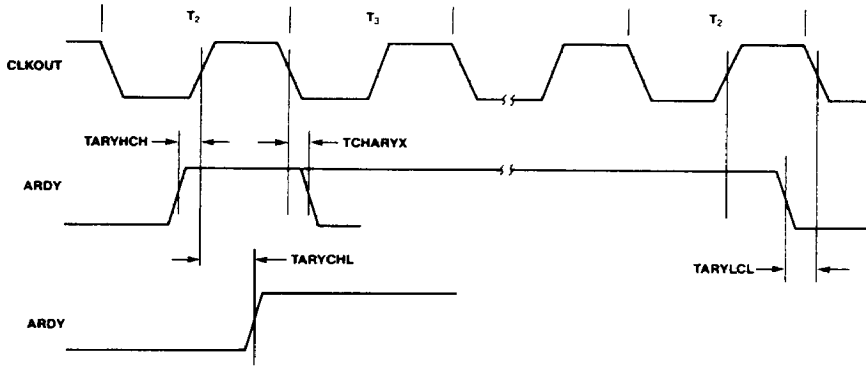


WF006240

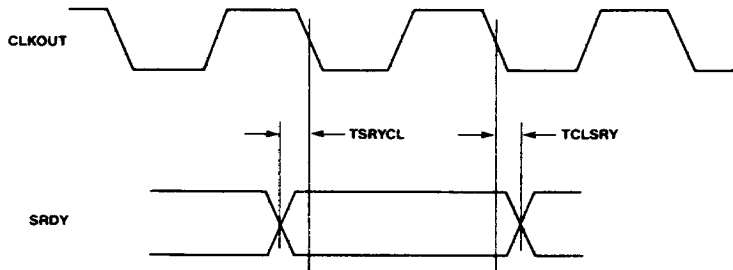


WF006250

SWITCHING WAVEFORMS (Cont'd.)

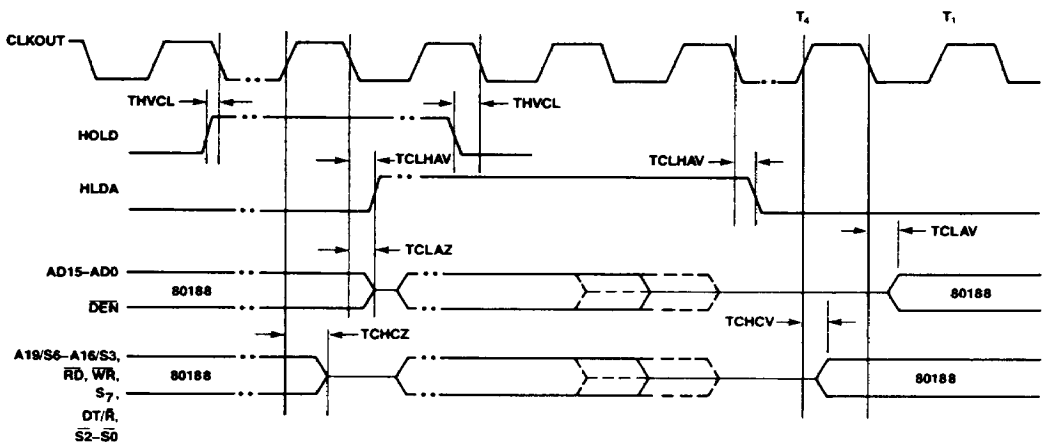


WF006262



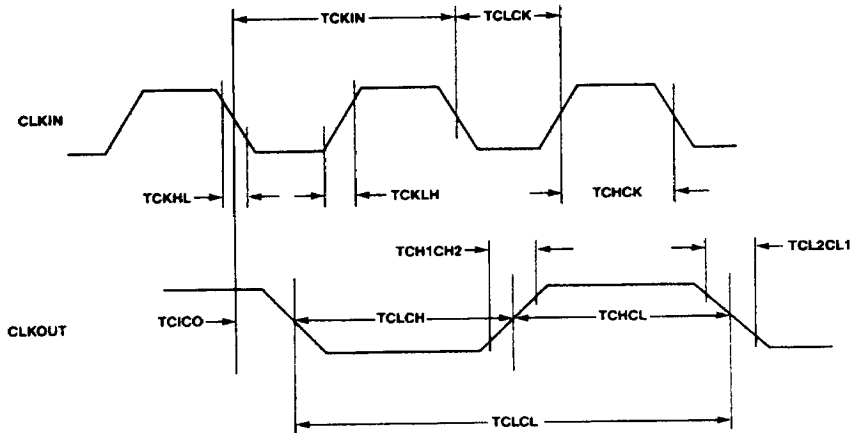
WF006270

HOLD-HLDA TIMING



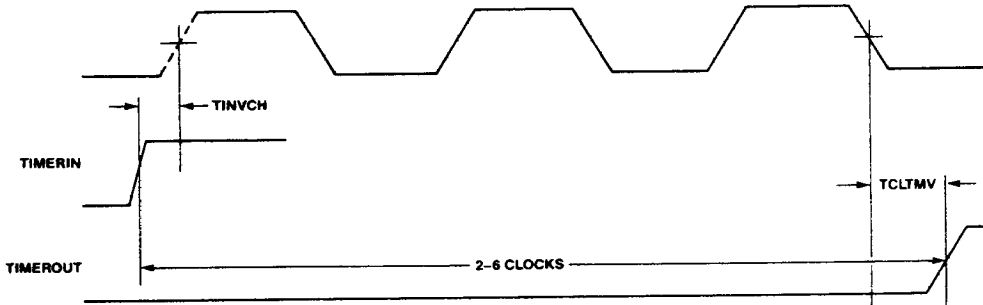
WF006282

SWITCHING WAVEFORMS (Cont'd.)



WF006293

TIMER ON 80188



WF006300

80188 INSTRUCTION TIMINGS

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

- The opcode, along with any data or displacement required for execution of a particular instruction, has been pre-fetched and resides in the queue at the time it is needed.
- No wait states or bus HOLDS occur.

- All word-data is located on even-address boundaries.

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

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

INSTRUCTION SET SUMMARY

FUNCTION	FORMAT	Clock Cycles	Comments				
DATA TRANSFER							
MOV = Move:							
Register to Register/Memory	<table border="1" style="display: inline-table;"><tr><td>1 0 0 0 1 0 0 w</td><td>mod reg r/m</td></tr></table>	1 0 0 0 1 0 0 w	mod reg r/m	2/12			
1 0 0 0 1 0 0 w	mod reg r/m						
Register/memory to register	<table border="1" style="display: inline-table;"><tr><td>1 0 0 0 1 0 1 w</td><td>mod reg r/m</td></tr></table>	1 0 0 0 1 0 1 w	mod reg r/m	2/9			
1 0 0 0 1 0 1 w	mod reg r/m						
Immediate to register/memory	<table border="1" style="display: inline-table;"><tr><td>1 1 0 0 0 1 1 w</td><td>mod 0 0 0 r/m</td><td>data</td><td>data if w = 1</td></tr></table>	1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1	12-13	8/16-bit
1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1				
Immediate to register	<table border="1" style="display: inline-table;"><tr><td>1 0 1 1 w</td><td>reg</td><td>data</td><td>data if w = 1</td></tr></table>	1 0 1 1 w	reg	data	data if w = 1	3-4	8/16-bit
1 0 1 1 w	reg	data	data if w = 1				
Memory to accumulator	<table border="1" style="display: inline-table;"><tr><td>1 0 1 0 0 0 0 w</td><td>addr-low</td><td>addr-high</td></tr></table>	1 0 1 0 0 0 0 w	addr-low	addr-high	9		
1 0 1 0 0 0 0 w	addr-low	addr-high					
Accumulator to memory	<table border="1" style="display: inline-table;"><tr><td>1 0 1 0 0 0 1 w</td><td>addr-low</td><td>addr-high</td></tr></table>	1 0 1 0 0 0 1 w	addr-low	addr-high	8		
1 0 1 0 0 0 1 w	addr-low	addr-high					
Register/memory to segment register	<table border="1" style="display: inline-table;"><tr><td>1 0 0 0 1 1 1 0</td><td>mod 0 reg r/m</td></tr></table>	1 0 0 0 1 1 1 0	mod 0 reg r/m	2/9			
1 0 0 0 1 1 1 0	mod 0 reg r/m						
Segment register to register/memory	<table border="1" style="display: inline-table;"><tr><td>1 0 0 0 1 1 0 0</td><td>mod 0 reg r/m</td></tr></table>	1 0 0 0 1 1 0 0	mod 0 reg r/m	2/11			
1 0 0 0 1 1 0 0	mod 0 reg r/m						
PUSH = Push:							
Memory	<table border="1" style="display: inline-table;"><tr><td>1 1 1 1 1 1 1 1</td><td>mod 1 1 0 r/m</td></tr></table>	1 1 1 1 1 1 1 1	mod 1 1 0 r/m	16			
1 1 1 1 1 1 1 1	mod 1 1 0 r/m						
Register	<table border="1" style="display: inline-table;"><tr><td>0 1 0 1 0</td><td>reg</td></tr></table>	0 1 0 1 0	reg	10			
0 1 0 1 0	reg						
Segment register	<table border="1" style="display: inline-table;"><tr><td>0 0 0</td><td>reg 1 1 0</td></tr></table>	0 0 0	reg 1 1 0	9			
0 0 0	reg 1 1 0						
Immediate	<table border="1" style="display: inline-table;"><tr><td>0 1 1 0 1 0 1</td><td>data</td><td>data r/m + B</td></tr></table>	0 1 1 0 1 0 1	data	data r/m + B	10		
0 1 1 0 1 0 1	data	data r/m + B					
PUSHA = Push All							
	<table border="1" style="display: inline-table;"><tr><td>0 1 1 0 0 0 0 0</td></tr></table>	0 1 1 0 0 0 0 0	20				
0 1 1 0 0 0 0 0							
POP = Pop:							
Memory	<table border="1" style="display: inline-table;"><tr><td>1 0 0 0 1 1 1 1</td><td>mod 0 0 0 r/m</td></tr></table>	1 0 0 0 1 1 1 1	mod 0 0 0 r/m	20			
1 0 0 0 1 1 1 1	mod 0 0 0 r/m						
Register	<table border="1" style="display: inline-table;"><tr><td>0 1 0 1 1</td><td>reg</td></tr></table>	0 1 0 1 1	reg	10			
0 1 0 1 1	reg						
Segment register	<table border="1" style="display: inline-table;"><tr><td>0 0 0</td><td>reg 1 1 1</td><td>(reg ≠ 01)</td></tr></table>	0 0 0	reg 1 1 1	(reg ≠ 01)	8		
0 0 0	reg 1 1 1	(reg ≠ 01)					
POPA = Pop All							
	<table border="1" style="display: inline-table;"><tr><td>0 1 1 0 0 0 0 1</td></tr></table>	0 1 1 0 0 0 0 1	20				
0 1 1 0 0 0 0 1							
XCHG = Exchange							
Register/memory with register	<table border="1" style="display: inline-table;"><tr><td>1 0 0 0 0 1 1 w</td><td>mod reg r/m</td></tr></table>	1 0 0 0 0 1 1 w	mod reg r/m	4/17			
1 0 0 0 0 1 1 w	mod reg r/m						
Register with accumulator	<table border="1" style="display: inline-table;"><tr><td>1 0 0 1 0</td><td>reg</td></tr></table>	1 0 0 1 0	reg	3			
1 0 0 1 0	reg						
IN = Input from:							
Fixed port	<table border="1" style="display: inline-table;"><tr><td>1 1 1 0 0 1 0 w</td><td>port</td></tr></table>	1 1 1 0 0 1 0 w	port	10			
1 1 1 0 0 1 0 w	port						
Variable port	<table border="1" style="display: inline-table;"><tr><td>1 1 1 0 1 1 0 w</td></tr></table>	1 1 1 0 1 1 0 w	8				
1 1 1 0 1 1 0 w							
OUT = Output to:							
Fixed port	<table border="1" style="display: inline-table;"><tr><td>1 1 1 0 0 1 1 w</td><td>port</td></tr></table>	1 1 1 0 0 1 1 w	port	9			
1 1 1 0 0 1 1 w	port						
Variable port	<table border="1" style="display: inline-table;"><tr><td>1 1 1 0 1 1 1 w</td></tr></table>	1 1 1 0 1 1 1 w	7				
1 1 1 0 1 1 1 w							
XLAT = Translate byte to AL	<table border="1" style="display: inline-table;"><tr><td>1 1 0 1 0 1 1 1</td></tr></table>	1 1 0 1 0 1 1 1	11				
1 1 0 1 0 1 1 1							
LEA = Load EA to register	<table border="1" style="display: inline-table;"><tr><td>1 0 0 0 1 1 0 1</td><td>mod reg r/m</td></tr></table>	1 0 0 0 1 1 0 1	mod reg r/m	6			
1 0 0 0 1 1 0 1	mod reg r/m						
LDS = Load pointer to DS	<table border="1" style="display: inline-table;"><tr><td>1 1 0 0 0 1 0 1</td><td>mod reg r/m</td><td>(mod ≠ 11)</td></tr></table>	1 1 0 0 0 1 0 1	mod reg r/m	(mod ≠ 11)	18		
1 1 0 0 0 1 0 1	mod reg r/m	(mod ≠ 11)					
LES = Load pointer to ES	<table border="1" style="display: inline-table;"><tr><td>1 1 0 0 0 1 0 0</td><td>mod reg r/m</td><td>(mod ≠ 11)</td></tr></table>	1 1 0 0 0 1 0 0	mod reg r/m	(mod ≠ 11)	18		
1 1 0 0 0 1 0 0	mod reg r/m	(mod ≠ 11)					
LAHF = Load AH with flags	<table border="1" style="display: inline-table;"><tr><td>1 0 0 1 1 1 1 1</td></tr></table>	1 0 0 1 1 1 1 1	2				
1 0 0 1 1 1 1 1							
SAHF = Store AH into flags	<table border="1" style="display: inline-table;"><tr><td>1 0 0 1 1 1 1 0</td></tr></table>	1 0 0 1 1 1 1 0	3				
1 0 0 1 1 1 1 0							
PUSHF = Push flags	<table border="1" style="display: inline-table;"><tr><td>1 0 0 1 1 1 0 0</td></tr></table>	1 0 0 1 1 1 0 0	9				
1 0 0 1 1 1 0 0							
POPF = Pop flags	<table border="1" style="display: inline-table;"><tr><td>1 0 0 1 1 1 0 1</td></tr></table>	1 0 0 1 1 1 0 1	8				
1 0 0 1 1 1 0 1							

Shaded areas indicate new 80188 instructions not available in 8086 or 8088 microprocessors.

All mnemonics copyright of Intel Corp. 1982

INSTRUCTION SET SUMMARY (Cont'd.)

FUNCTION	FORMAT	Clock Cycles	Comments
ARITHMETIC			
ADD = Add:			
Reg/memory with register to either	0 0 0 0 0 0 d w mod reg r/m	3/10	
Immediate to register / memory	1 0 0 0 0 0 s w mod 0 0 0 r/m data data if s w = 01	4/16	
Immediate to accumulator	0 0 0 0 0 1 0 w data data if w = 1	3/4	8/16-bit
ADC = Add with carry:			
Reg/memory with register to either	0 0 0 1 0 0 d w mod reg r/m	3/10	
Immediate to register/memory	1 0 0 0 0 0 s w mod 0 1 0 r/m data data if s w = 01	4/16	
Immediate to accumulator	0 0 0 1 0 1 0 w data data if w = 1	3/4	8/16-bit
INC = Increment:			
Register/memory	1 1 1 1 1 1 1 w mod 0 0 0 r/m	3/15	
Register	0 1 0 0 0 reg	3	
SUB = Subtract:			
Reg/memory and register to either	0 0 1 0 1 0 d w mod reg r/m	3/10	
Immediate from register/memory	1 0 0 0 0 0 s w mod 1 0 1 r/m data data if s w = 01	4/16	
Immediate from accumulator	0 0 1 0 1 1 0 w data data if w = 1	3/4	8/16-bit
SBB = Subtract with borrow:			
Reg/memory and register to either	0 0 0 1 1 0 d w mod reg r/m	3/10	
Immediate from register/memory	1 0 0 0 0 0 s w mod 0 1 1 r/m data data if s w = 01	4/16	
Immediate from accumulator	0 0 0 1 1 1 0 w data data if w = 1	3/4	8/16-bit
DEC = Decrement:			
Register/memory	1 1 1 1 1 1 1 w mod 0 0 1 r/m	3/15	
Register	0 1 0 0 1 reg	3	
CMP = Compare:			
Register/memory with register	0 0 1 1 1 0 1 w mod reg r/m	3/10	
Register with register/memory	0 0 1 1 1 0 0 w mod reg r/m	3/10	
Immediate with register/memory	1 0 0 0 0 0 s w mod 1 1 1 r/m data data if s w = 01	3/10	
Immediate with accumulator	0 0 1 1 1 1 0 w data data if w = 1	3/4	8/16-bit
NEG = Change sign	1 1 1 1 0 1 1 w mod 0 1 1 r/m	3	
AAA = ASCII adjust for add	0 0 1 1 0 1 1 1	8	
DAA = Decimal adjust for add	0 0 1 0 0 1 1 1	4	
AAS = ASCII adjust for subtract	0 0 1 1 1 1 1 1	7	
DAS = Decimal adjust for subtract	0 0 1 0 1 1 1 1	4	
MUL = Multiply (unsigned)			
Register-Byte	1 1 1 1 0 1 1 w mod 1 0 0 r/m	26 - 28	
Register-Word		35 - 37	
Memory-Byte		32 - 34	
Memory-Word		41 - 43	
IMUL = Integer multiply (signed):			
Register-Byte		25 - 28	
Register-Word		34 - 37	
Memory-Byte		31 - 34	
Memory-Word		40 - 43	
IMUL = Integer immediate multiply (signed)			
(signed)	1 1 1 1 0 1 1 w mod 1 1 0 r/m	26 - 28	
DIV = Divide (unsigned):			
Register-Byte		29	
Register-Word		38	
Memory-Byte		35	
Memory-Word		44	

Shaded areas indicate new 80188 instructions not available in 8086 or 8088 microprocessors.

All mnemonics copyright of Intel Corp. 1982

INSTRUCTION SET SUMMARY (Cont'd.)

FUNCTION	FORMAT	Clock Cycles	Comments																
ARITHMETIC (Continued):																			
IDIV = Integer divide (signed): Register-Byte Register-Word Memory-Byte Memory-Word	1 1 1 1 0 1 1 w mod 1 1 1 r/m	44 - 52																	
AAM = ASCII adjust for multiply	1 1 0 1 0 1 0 0 0 0 0 0 1 0 1 0	19																	
AAD = ASCII adjust for divide	1 1 0 1 0 1 0 1 0 0 0 0 1 0 1 0	15																	
CBW = Convert byte to word	1 0 0 1 1 0 0 0	2																	
CWD = Convert word to double word	1 0 0 1 1 0 0 1	4																	
LOGIC																			
Shift/Rotate Instructions:																			
Register/Memory by 1	1 1 0 1 0 0 0 w mod TTT r/m	2/15																	
Register/Memory by CL	1 1 0 1 0 0 1 w mod TTT r/m																		
Register/Memory by Count	1 1 0 0 0 0 0 w mod TTT r/m count	5 + n/17 + n																	
	<table style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding-right: 20px;">TTT</td> <td>Instruction</td> </tr> <tr> <td>0 0 0</td> <td>ROL</td> </tr> <tr> <td>0 0 1</td> <td>ROR</td> </tr> <tr> <td>0 1 0</td> <td>RCL</td> </tr> <tr> <td>0 1 1</td> <td>RCR</td> </tr> <tr> <td>1 0 0</td> <td>SHL/SAL</td> </tr> <tr> <td>1 0 1</td> <td>SHR</td> </tr> <tr> <td>1 1 1</td> <td>SAR</td> </tr> </table>	TTT	Instruction	0 0 0	ROL	0 0 1	ROR	0 1 0	RCL	0 1 1	RCR	1 0 0	SHL/SAL	1 0 1	SHR	1 1 1	SAR		
TTT	Instruction																		
0 0 0	ROL																		
0 0 1	ROR																		
0 1 0	RCL																		
0 1 1	RCR																		
1 0 0	SHL/SAL																		
1 0 1	SHR																		
1 1 1	SAR																		
AND = And:																			
Reg/memory and register to either	0 0 1 0 0 0 d w mod reg r/m	3/10																	
Immediate to register/memory	1 0 0 0 0 0 0 w mod 1 0 0 r/m data data if w = 1	4/16																	
Immediate to accumulator	0 0 1 0 0 1 0 w data data if w = 1	3/4	8/16-bit																
TEST = And function to flags, no result:																			
Register/memory and register	1 0 0 0 0 1 0 w mod reg r/m	3/10																	
Immediate data and register/memory	1 1 1 1 0 1 1 w mod 0 0 0 r/m data data if w = 1	4/10																	
Immediate data and accumulator	1 0 1 0 1 0 0 w data data if w = 1	3/4	8/16-bit																
OR = Or:																			
Reg/memory and register to either	0 0 0 0 1 0 d w mod reg r/m	3/10																	
Immediate to register/memory	1 0 0 0 0 0 0 w mod 0 0 1 r/m data data if w = 1	4/16																	
Immediate to accumulator	0 0 0 0 1 1 0 w data data if w = 1	3/4	8/16-bit																
XOR = Exclusive or:																			
Reg/memory and register to either	0 0 1 1 0 0 d w mod reg r/m	3/10																	
Immediate to register/memory	1 0 0 0 0 0 0 w mod 1 1 0 r/m data data if w = 1	4/16																	
Immediate to accumulator	0 0 1 1 0 1 0 w data data if w = 1	3/4	8/16-bit																
NOT = Invert register/memory	1 1 1 1 0 1 1 w mod 0 1 0 r/m	3																	
STRING MANIPULATION:																			
MOVS = Move byte/word	1 0 1 0 0 1 0 w	8 + 8n																	
CMPS = Compare byte/word	1 0 1 0 0 1 1 w	5 + 22n																	
SCAS = Scan byte/word	1 0 1 0 1 1 1 w	5 + 15n																	
LODS = Load byte/wd to AL/AX	1 0 1 0 1 1 0 w	6 + 11n																	
STOS = Stor byte/wd from AL/A	1 0 1 0 1 0 1 w	6 + 9n																	
INS = Input byte/wd from DX port	0 1 0 1 1 0 w																		
OUTS = Output byte/wd to DX port	0 1 0 1 1 1 w																		

Shaded areas indicate new 80188 instructions not available in 8086 or 8088 microprocessors.

All mnemonics copyright of Intel Corp. 1982

INSTRUCTION SET SUMMARY (Cont'd.)

FUNCTION	FORMAT	Clock Cycles	Comments				
STRING MANIPULATION (Continued):							
Repeated by count in CX							
MOVS = Move string	<table border="1" style="display: inline-table;"><tr><td>1 1 1 1 0 0 1 0</td><td>1 0 1 0 0 1 0 w</td></tr></table>	1 1 1 1 0 0 1 0	1 0 1 0 0 1 0 w	14			
1 1 1 1 0 0 1 0	1 0 1 0 0 1 0 w						
CMPS = Compare string	<table border="1" style="display: inline-table;"><tr><td>1 1 1 1 0 0 1 z</td><td>1 0 1 0 0 1 1 w</td></tr></table>	1 1 1 1 0 0 1 z	1 0 1 0 0 1 1 w	22			
1 1 1 1 0 0 1 z	1 0 1 0 0 1 1 w						
SCAS = Scan string	<table border="1" style="display: inline-table;"><tr><td>1 1 1 1 0 0 1 z</td><td>1 0 1 0 1 1 1 w</td></tr></table>	1 1 1 1 0 0 1 z	1 0 1 0 1 1 1 w	15			
1 1 1 1 0 0 1 z	1 0 1 0 1 1 1 w						
LODS = Load string	<table border="1" style="display: inline-table;"><tr><td>1 1 1 1 0 0 1 0</td><td>1 0 1 0 1 1 0 w</td></tr></table>	1 1 1 1 0 0 1 0	1 0 1 0 1 1 0 w	12			
1 1 1 1 0 0 1 0	1 0 1 0 1 1 0 w						
STOS = Store string	<table border="1" style="display: inline-table;"><tr><td>1 1 1 1 0 0 1 0</td><td>1 0 1 0 1 0 1 w</td></tr></table>	1 1 1 1 0 0 1 0	1 0 1 0 1 0 1 w	10			
1 1 1 1 0 0 1 0	1 0 1 0 1 0 1 w						
INS = Input string	<table border="1" style="display: inline-table;"><tr><td>1 1 1 1 0 0 1 0</td><td>0 1 1 0 1 1 0 w</td></tr></table>	1 1 1 1 0 0 1 0	0 1 1 0 1 1 0 w	8 + 2n/16	Repeated/Not Repeated		
1 1 1 1 0 0 1 0	0 1 1 0 1 1 0 w						
OUTS = Output string	<table border="1" style="display: inline-table;"><tr><td>1 1 1 1 0 0 1 0</td><td>0 1 1 0 1 1 1 w</td></tr></table>	1 1 1 1 0 0 1 0	0 1 1 0 1 1 1 w	8 + 2n/16	Repeated/Not Repeated		
1 1 1 1 0 0 1 0	0 1 1 0 1 1 1 w						
CONTROL TRANSFER							
CALL = Call:							
Direct within segment	<table border="1" style="display: inline-table;"><tr><td>1 1 1 0 1 0 0 0</td><td>disp-low</td><td>disp-high</td></tr></table>	1 1 1 0 1 0 0 0	disp-low	disp-high	14		
1 1 1 0 1 0 0 0	disp-low	disp-high					
Register memory indirect within segment	<table border="1" style="display: inline-table;"><tr><td>1 1 1 1 1 1 1 1</td><td>mod 0 1 0 r/m</td></tr></table>	1 1 1 1 1 1 1 1	mod 0 1 0 r/m	13/19			
1 1 1 1 1 1 1 1	mod 0 1 0 r/m						
Direct intersegment	<table border="1" style="display: inline-table;"><tr><td>1 0 0 1 1 0 1 0</td><td>segment offset</td></tr><tr><td></td><td>segment selector</td></tr></table>	1 0 0 1 1 0 1 0	segment offset		segment selector	23	
1 0 0 1 1 0 1 0	segment offset						
	segment selector						
Indirect intersegment	<table border="1" style="display: inline-table;"><tr><td>1 1 1 1 1 1 1 1</td><td>mod 0 1 1 r/m</td><td>(mod ≠ 11)</td></tr></table>	1 1 1 1 1 1 1 1	mod 0 1 1 r/m	(mod ≠ 11)	38		
1 1 1 1 1 1 1 1	mod 0 1 1 r/m	(mod ≠ 11)					
JMP = Unconditional jump:							
Short/long	<table border="1" style="display: inline-table;"><tr><td>1 1 1 0 1 0 1 1</td><td>disp-low</td></tr></table>	1 1 1 0 1 0 1 1	disp-low	13			
1 1 1 0 1 0 1 1	disp-low						
Direct within segment	<table border="1" style="display: inline-table;"><tr><td>1 1 1 0 1 0 0 1</td><td>disp-low</td><td>disp-high</td></tr></table>	1 1 1 0 1 0 0 1	disp-low	disp-high	13		
1 1 1 0 1 0 0 1	disp-low	disp-high					
Register/memory indirect within segment	<table border="1" style="display: inline-table;"><tr><td>1 1 1 1 1 1 1 1</td><td>mod 1 0 0 r/m</td></tr></table>	1 1 1 1 1 1 1 1	mod 1 0 0 r/m	11/17			
1 1 1 1 1 1 1 1	mod 1 0 0 r/m						
Direct intersegment	<table border="1" style="display: inline-table;"><tr><td>1 1 1 0 1 0 1 0</td><td>segment offset</td></tr><tr><td></td><td>segment selector</td></tr></table>	1 1 1 0 1 0 1 0	segment offset		segment selector	13	
1 1 1 0 1 0 1 0	segment offset						
	segment selector						
Indirect intersegment	<table border="1" style="display: inline-table;"><tr><td>1 1 1 1 1 1 1 1</td><td>mod 1 0 1 r/m</td><td>(mod ≠ 11)</td></tr></table>	1 1 1 1 1 1 1 1	mod 1 0 1 r/m	(mod ≠ 11)	26		
1 1 1 1 1 1 1 1	mod 1 0 1 r/m	(mod ≠ 11)					
RET = Return from CALL:							
Within segment	<table border="1" style="display: inline-table;"><tr><td>1 1 0 0 0 0 1 1</td></tr></table>	1 1 0 0 0 0 1 1	16				
1 1 0 0 0 0 1 1							
Within seg adding immed to SP	<table border="1" style="display: inline-table;"><tr><td>1 1 0 0 0 0 1 0</td><td>data-low</td><td>data-high</td></tr></table>	1 1 0 0 0 0 1 0	data-low	data-high	18		
1 1 0 0 0 0 1 0	data-low	data-high					
Intersegment	<table border="1" style="display: inline-table;"><tr><td>1 1 0 0 1 0 1 1</td></tr></table>	1 1 0 0 1 0 1 1	22				
1 1 0 0 1 0 1 1							
Intersegment adding immediate to SP	<table border="1" style="display: inline-table;"><tr><td>1 1 0 0 1 0 1 0</td><td>data-low</td><td>data-high</td></tr></table>	1 1 0 0 1 0 1 0	data-low	data-high	25		
1 1 0 0 1 0 1 0	data-low	data-high					

Shaded areas indicate new 80188 instructions not available in 8086 or 8088 microprocessors.

INSTRUCTION SET SUMMARY (Cont'd.)

FUNCTION	FORMAT	Clock Cycles	Comments
CONTROL TRANSFER (Continued):			
JE/JZ = Jump on equal zero	0 1 1 1 0 1 0 0 disp	4/13	13 if JMP taken 4 if JMP not taken
JL/JNGE = Jump on less not greater or equal	0 1 1 1 1 1 0 0 disp	4/13	
JLE/JNG = Jump on less or equal not greater	0 1 1 1 1 1 1 0 disp	4/13	
JB/JNAE = Jump on below not above or equal	0 1 1 1 0 0 1 0 disp	4/13	
JBE/JNA = Jump on below or equal not above	0 1 1 1 0 1 1 0 disp	4/13	
JP/JPE = Jump on parity parity even	0 1 1 1 1 0 1 0 disp	4/13	
JO = Jump on overflow	0 1 1 1 0 0 0 0 disp	4/13	
JS = Jump on sign	0 1 1 1 1 0 0 0 disp	4/13	
JNE/JNZ = Jump on not equal not zero	0 1 1 1 0 1 0 1 disp	4/13	
JNL/JGE = Jump on not less greater or equal	0 1 1 1 1 1 0 1 disp	4/13	
JNLE/JG = Jump on not less or equal greater	0 1 1 1 1 1 1 1 disp	4/13	
JNB/JAE = Jump on not below above or equal	0 1 1 1 0 0 1 1 disp	4/13	
JNBE/JA = Jump on not below or equal above	0 1 1 1 0 1 1 1 disp	4/13	
JNP/JPO = Jump on not par / par odd	0 1 1 1 1 0 1 1 disp	4/13	
JNO = Jump on not overflow	0 1 1 1 0 0 0 1 disp	4/13	
JNS = Jump on not sign	0 1 1 1 1 0 0 1 disp	4/13	
LOOP = Loop CX times	1 1 1 0 0 0 1 0 disp	5/15	
LOOPZ/LOOPE = Loop while zero equal	1 1 1 0 0 0 0 1 disp	6/16	
LOOPNZ/LOOPNE = Loop while not zero equal	1 1 1 0 0 0 0 0 disp	6/16	
JCXZ = Jump on CX zero	1 1 1 0 0 0 1 1 disp	16 5	
ENTER = Enter Procedure L = 0 I = 1 L = 1	1 1 0 0 1 0 0 0 data byte data byte	16 5 5	22 - 1000 - 11 8
LEAVE = Leave Procedure	1 1 1 0 0 1 0 0 1	16 5	
INT = Interrupt:			
Type specified	1 1 0 0 1 1 0 1 type	47	
Type 3	1 1 0 0 1 1 0 0	45	
INTO = Interrupt on overflow	1 1 0 0 1 1 1 0	48/4	if INT. taken/ if INT. not taken
IRET = Interrupt return	1 1 0 0 1 1 1 1	28	
BOUND = Detect value out of range	0 1 1 0 0 0 1 0 mod reg / r/m	39 - 95	

Shaded areas indicate new 80188 instructions not available in 8086 or 8088 microprocessors.

INSTRUCTION SET SUMMARY (Cont'd.)

FUNCTION	FORMAT	Clock Cycles	Comments
PROCESSOR CONTROL			
CLC = Clear carry	1 1 1 1 1 0 0 0	2	
CMC = Complement carry	1 1 1 1 0 1 0 1	2	
STC = Set carry	1 1 1 1 1 0 0 1	2	
CLD = Clear direction	1 1 1 1 1 1 0 0	2	
STD = Set direction	1 1 1 1 1 1 0 1	2	
CLI = Clear interrupt	1 1 1 1 1 0 1 0	2	
STI = Set interrupt	1 1 1 1 1 0 1 1	2	
HLT = Halt	1 1 1 1 0 1 0 0	2	
WAIT = Wait	1 0 0 1 1 0 1 1	6	if test = 0
LOCK = Bus lock prefix	1 1 1 1 0 0 0 0	2	
ESC = Processor Extension Escape	1 0 0 1 1 T T T mod LLL r/m	6	
(TTT LLL are opcode to processor extension)			

FOOTNOTES

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

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

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

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

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

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

reg is assigned according to the following:

reg	Segment Register
00	ES
01	CS
10	SS
11	DS

REG is assigned according to the following table:

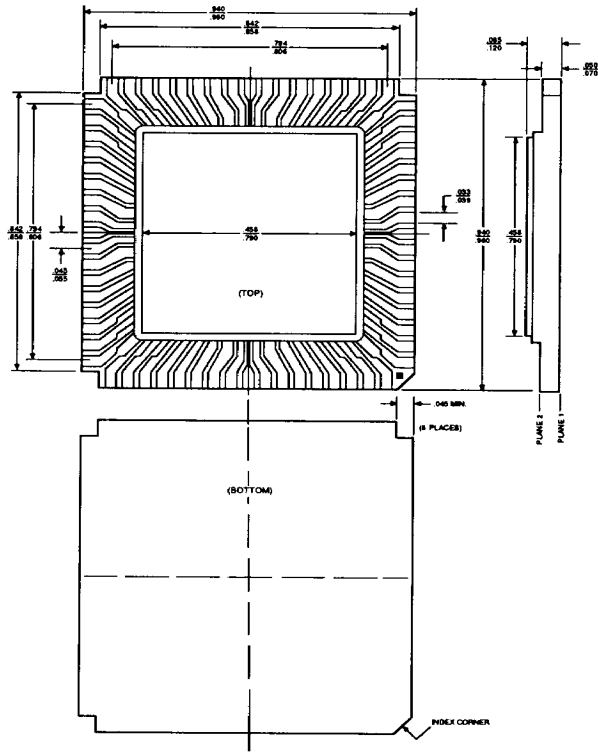
16-Bit (w = 1) 8-Bit (w = 0)

000 AX	000 AL
001 CX	001 CL
010 DX	010 DL
011 BX	011 BL
100 SP	100 AH
101 BP	101 CH
110 SI	110 DH
111 DI	111 BH

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

PHYSICAL DIMENSIONS*

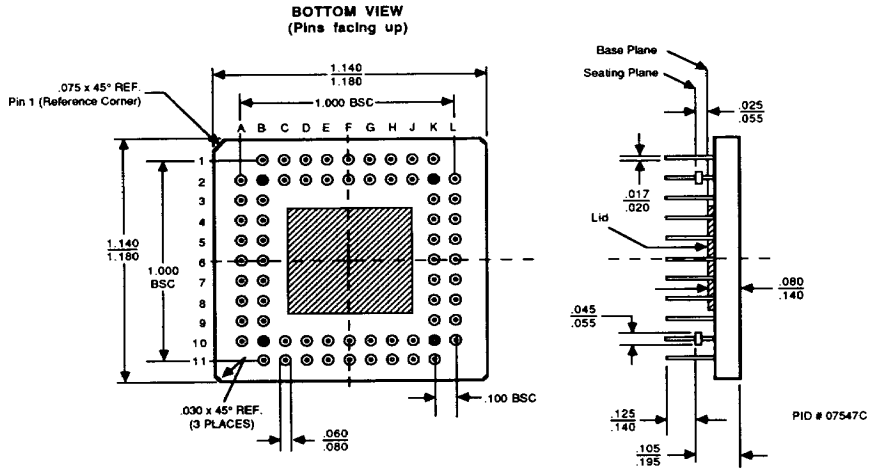
CA2086



*For reference only. All dimensions are measured in inches. BSC is an ANSI standard for Basic Space Centering.

PHYSICAL DIMENSIONS (Cont'd.)

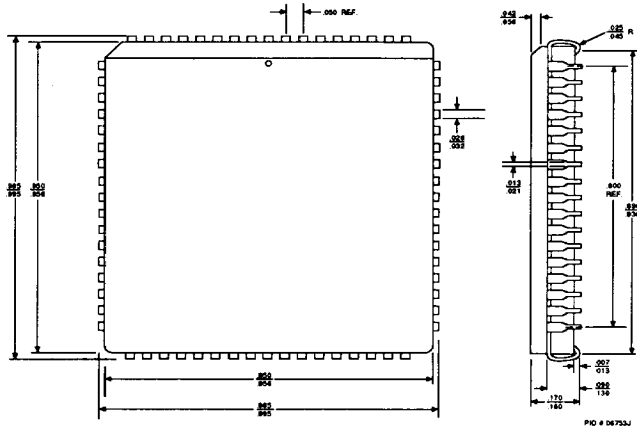
CGX068



*For reference only. All dimensions are measured in inches. BSC is an ANSI standard for Basic Space Centering.

PHYSICAL DIMENSIONS (Cont'd.)

PL 068



*For reference only. All dimensions are measured in inches. BSC is an ANSI standard for Basic Space Centering.

North American

ALABAMA	(205) 882-9122
ARIZONA	(602) 242-4400
CALIFORNIA	
Culver City	(213) 645-1524
Newport Beach	(714) 752-6262
Roseville	(916) 786-6700
San Diego	(619) 560-7030
San Jose	(408) 452-0500
Woodland Hills	(818) 992-4155
CANADA, Ontario	
Kanata	(613) 592-0060
Willowdale	(416) 224-5193
COLORADO	(303) 741-2900
CONNECTICUT	(203) 264-7800
FLORIDA	
Clearwater	(813) 530-9971
Ft. Lauderdale	(305) 776-2001
Orlando	(407) 830-8100
GEORGIA	(404) 449-7920
ILLINOIS	
Chicago	(312) 773-4422
Naperville	(312) 505-9517
KANSAS	(913) 451-3115
MARYLAND	(301) 796-9310
MASSACHUSETTS	(617) 273-3970
MINNESOTA	(612) 938-0001
MISSOURI	(913) 451-3115
NEW JERSEY	
Cherry Hill	(609) 662-2900
Parsippany	(201) 299-0002
NEW YORK	
Liverpool	(315) 457-5400
Poughkeepsie	(914) 471-8180
NORTH CAROLINA	(919) 878-8111
OHIO	
Columbus	(614) 891-6455
Dayton	(513) 439-0470
OREGON	(503) 245-0080
PENNSYLVANIA	(215) 398-8006
SOUTH CAROLINA	(803) 772-6760
TEXAS	
Austin	(512) 346-7830
Dallas	(214) 934-9099
Houston	(713) 785-9001

International

BELGIUM, Bruxelles	TEL (02) 771-91-42
	FAX (02) 762-37-12
	TLX 61028
FRANCE, Paris	TEL (1) 49-75-10-10
	FAX (1) 49-75-10-13
	TLX 263282
WEST GERMANY, Hannover area	TEL (0511) 736085
	FAX (0511) 721254
	TLX 922850
München	TEL (089) 4114-0
	FAX (089) 406490
	TLX 523883
Stuttgart	TEL (0711) 62 33 77
	FAX (0711) 625187
	TLX 721882
HONG KONG	TEL 852-5-8654525
	FAX 852-5-8654335
	TLX 67955AMDAPHX
ITALY, Milan	TEL (02) 3390541
	(02) 3533241
	FAX (02) 3498000
	TLX 315286
JAPAN, Kanagawa	TEL 462-47-2911
	FAX 462-47-1729
Tokyo	TEL (03) 345-8241
	FAX (03) 342-5196
	TLX J24064AMDTKOJ
Osaka	TEL 06-243-3250
	FAX 06-243-3253

International (Continued)

KOREA, Seoul	TEL 822-784-0030
	FAX 822-784-8014
LATIN AMERICA, Ft. Lauderdale	TEL (305) 484-8600
	FAX (305) 485-9736
	TEL 5109554261 AMDFTL
NORWAY, Hovik	TEL (02) 537810
	FAX (02) 591959
	TLX 79079
SINGAPORE	TEL 65-3481188
	FAX 65-3480161
	TLX 55650 AMDMMI
SWEDEN, Stockholm	TEL (08) 733 03 50
	FAX (08) 733 22 85
	TLX 11602
TAIWAN	TEL 886-2-7213393
	TLX 886-2-7122066
	FAX 886-2-7723422
UNITED KINGDOM, Manchester area	TEL (0925) 828008
	FAX (0925) 827693
	TLX 628524
London area	TEL (0483) 740440
	FAX (0483) 756196
	TLX 859103

North American Representatives

CANADA, Burnaby, B.C.	
DAVETEK MARKETING	(604) 430-3680
Calgary, Alberta	
DAVETEK MARKETING	(403) 291-4984
Kanata, Ontario	
VITEL ELECTRONICS	(613) 592-0060
Mississauga, Ontario	
VITEL ELECTRONICS	(416) 676-9720
Lachine, Quebec	
VITEL ELECTRONICS	(514) 636-5951
IDAHO	
INTERMOUNTAIN TECH MKTG	(208) 888-6071
ILLINOIS	
HEARTLAND TECHNICAL MARKETING	(312) 577-9222
INDIANA	
ELECTRONIC MARKETING CONSULTANTS, INC	(317) 921-3452
IOWA	
LORENZ SALES	(319) 377-4666
KANSAS	
Merriam - LORENZ SALES	(913) 384-6556
Wichita - LORENZ SALES	(316) 721-0500
KENTUCKY	
ELECTRONIC MARKETING CONSULTANTS, INC	(317) 921-3452
MICHIGAN	
Holland - COM-TEK SALES, INC	(616) 399-7273
Novi - COM-TEK SALES, INC	(313) 344-1409
MISSOURI	
LORENZ SALES	(314) 997-4558
NEBRASKA	
LORENZ SALES	(402) 475-4660
NEW MEXICO	
THORSON DESERT STATES	(505) 293-8555
NEW YORK	
East Syracuse - NYCOM, INC	(315) 437-8343
Woodbury - COMPONENT CONSULTANTS, INC	(516) 364-8020
OHIO	
Centerville - DOLFUSS ROOT & CO	(513) 433-6776
Columbus - DOLFUSS ROOT & CO	(614) 885-4844
Strongsville - DOLFUSS ROOT & CO	(216) 238-0300
PENNSYLVANIA	
DOLFUSS ROOT & CO	(412) 221-4420
PUERTO RICO	
COMP. REP. ASSOCIATES	(809) 746-6550
UTAH	
R ² MARKETING	(801) 595-0631
WASHINGTON	
ELECTRA TECHNICAL SALES	(206) 455-3600
WISCONSIN	
HEARTLAND TECHNICAL MARKETING	(414) 796-1128

Advanced Micro Devices reserves the right to make changes in its product without notice in order to improve design or performance characteristics. The performance characteristics listed in this document are guaranteed by specific tests, guard banding, design and other practices common to the industry. For specific testing details, contact your local AMD sales representative. The company assumes no responsibility for the use of any circuits described herein.



Advanced Micro Devices, Inc. 901 Thompson Place, P.O. Box 3453, Sunnyvale, CA 94088, USA
Tel: (408) 732-2400 • TWX: 910-339-9280 • TELEX: 34-6306 • TOLL FREE: (800) 538-8450
APPLICATIONS HOTLINE TOLL FREE: (800) 222-9323 • (408) 749-5703

© 1989 Advanced Micro Devices, Inc.
2/15/89
AIS-WCP-12M-3/89-0 Printed in USA

57

02670B ✓