```
                              .REPEAT           ;wait until DMA complete

0043  E4 78                   IN     AL,STATUS
                              .UNTIL AL &1

                              RET

              TRANS           ENDP
```

Programming the DMA controller requires a few steps, as illustrated in Example 12–1. The leftmost digit of the 5-digit address is sent to latch B. Next, the channels are programmed after the F/L flip-flop is cleared. Note that we use channel 0 as the source and channel 1 as the destination for a memory-to-memory transfer. The count is next programmed with a value that is one less than the number of bytes to be transferred. Next, the mode register of each channel is programmed, the command register selects a block move, channel 0 is enabled, and a software DMA request is initiated. Before return is made from the procedure, the status register is tested for a terminal count. Recall that the terminal count flag indicates that the DMA transfer is completed. The TC also disables the channel, preventing additional transfers.

Sample Memory Fill Using the 8237. In order to fill an area of memory with the same data, the channel 0 source register is programmed to point to the same address throughout the transfer. This is accomplished with the channel 0 hold mode. The controller copies the contents of this single memory location to an entire block of memory addressed by channel 1. This has many useful applications.

For example, suppose that a video display must be cleared. This operation can be performed using the DMA controller with the channel 0 hold mode and a memory-to-memory transfer. If the video display contains 80 columns and 25 lines, it has 2000 display positions that must be set to 20H (an ASCII space) to clear the screen.

Example 12–2 shows a procedure that clears an area of memory addressed by ES:DI. The CX register transfers the number of bytes to be cleared to the CLEAR procedure. Notice that this procedure is nearly identical to Example 12–1, except that the command register is programmed so the channel 0 address is held. The source address is programmed as the same address as ES:DI, and then the destination is programmed as one location beyond ES:DI. Also note that this program is designed to function with the hardware in Figure 12–12 and will not function in the personal computer unless you have the same hardware.

EXAMPLE 12–2

```
                    ;A procedure that clears an area of memory using the
                    ;8237A DMA controller in Figure 12-12. This is a
                    ;memory-to-memory block transfer with a channel 0 hold.
                    ;
                    ;Calling parameters:
                    ;    DI = offset address of area cleared
                    ;    ES = segment address of area cleared
                    ;    CX = number of bytes cleared
                    ;
= 0010                 LATCHB  EQU   10H         ;latch B
= 007C                 CLEAR_F EQU   7CH         ;F/L flip flop
= 0070                 CH0_A   EQU   70H         ;channel 0 address
= 0072                 CH1_A   EQU   72H         ;channel 1 address
= 0073                 CH1_C   EQU   73H         ;channel 1 count
= 007B                 MODE    EQU   7BH         ;mode
= 0078                 CMMD    EQU   78H         ;command
= 007F                 MASKS   EQU   7FH         ;masks
= 0079                 REQ     EQU   79H         ;request register
= 0078                 STATUS  EQU   78H         ;status register
= 0000                 ZERO    EQU   0H          ;zero

0000                   CLEAR   PROC  FAR USES AX

0001  8C C0                    MOV   AX,ES       ;program latch B
```

```
0003  8A C4              MOV   AL,AH
0005  C0 E8 04           SHR   AL,4
0008  E6 10              OUT   LATCHB,AL

000A  E6 7C              OUT   CLEAR_F,AL ;clear F/L flip-flop

000C  2E: A0 0000        MOV   AL,CS:ZERO
0010  26: 88 05          MOV   ES:[DI],AL ;save zero in first byte

0013  8C C0              MOV   AX,ES      ;program source address
0015  C1 E0 04           SHL   AX,4
0018  03 C7              ADD   AX,DI      ;form source offset
001A  E6 70              OUT   CH0_A,AL
001C  8A C4              MOV   AL,AH
001E  E6 70              OUT   CH0_A,AL

0020  8C C0              MOV   AX,ES      ;program destination address
0022  C1 E0 04           SHL   AX,4
0025  03 C7              ADD   AX,DI      ;form destination offset
0027  48                 INC   AX
0028  E6 72              OUT   CH1_A,AL
002A  8A C4              MOV   AL,AH
002C  E6 72              OUT   CH1_A,AL

002E  8B C1              MOV   AX,CX      ;program count
0030  48                 DEC   AX         ;adjust count
0031  48                 DEC   AX
0032  E6 73              OUT   CH1_C,AL
0034  8A C4              MOV   AL,AH
0036  E6 73              OUT   CH1_C,AL

0038  B0 88              MOV   AL,88H     ;program mode
003A  E6 7B              OUT   MODE,AL
003C  B0 85              MOV   AL,85H
003E  E6 7B              OUT   MODE,AL

0040  B0 03              MOV   AL,3       ;enable block hold transfer
0042  E6 78              OUT   CMMD,AL

0044  B0 0E              MOV   AL,0EH     ;unmask channel 0
0046  E6 7F              OUT   MASKS,AL

0048  B0 04              MOV   AL,4       ;start DMA transfer
004A  E6 79              OUT   REQ,AL

                         .REPEAT          ;wait until DMA complete
004C  E4 78              IN    AL,STATUS
                         .UNTIL AL &1
                         RET

0054            CLEAR    ENDP
```

12–3 SUMMARY

1. The HOLD input is used to request a DMA action, and the HLDA output signals that the hold is in effect. When a logic 1 is placed on the HOLD input, the microprocessor (1) stops executing the program; (2) places its address, data, and control bus at their high-impedance state; and (3) signals that the hold is in effect by placing a logic 1 on the HLDA pin.

2. A DMA read operation transfers data from a memory location to an external I/O device. A DMA write operation transfers data from an I/O device into the memory. Also available is a memory-to-memory transfer that allows data to be transferred between two memory locations by using DMA techniques.
3. The 8237 direct memory access (DMA) controller is a four-channel device that can be expanded to include an additional channel of DMA.

12–4 QUESTIONS AND PROBLEMS

1. Which microprocessor pins are used to request and acknowledge a DMA transfer?
2. Explain what happens whenever a logic 1 is placed on the HOLD input pin.
3. A DMA read transfers data from ___________ to ___________.
4. A DMA write transfers data from ___________ to ___________.
5. The DMA controller selects the memory location used for a DMA transfer through what bus signals?
6. The DMA controller selects the I/O device used during a DMA transfer by which pin?
7. What is a memory-to-memory DMA transfer?
8. Describe the effect on the microprocessor and DMA controller when the HOLD and HLDA pins are at their logic 1 levels.
9. Describe the effect on the microprocessor and DMA controller when the HOLD and HLDA pins are at their logic 0 levels.
10. The 8237 DMA controller is a _______ channel DMA controller.
11. If the 8237 DMA controller is decoded at I/O ports 2000H–200FH, what ports are used to program channel 1?
12. Which 8237 DMA controller register is programmed to initialize the controller?
13. How many bytes can be transferred by the 8237 DMA controller?
14. Write a sequence of instructions that transfer data from memory location 21000H–210FFH to 20000H–200FFH by using channel 2 of the 8237 DMA controller. You must initialize the 8237 and use the latch described in Section 12–1 to hold A19–A16.
15. Write a sequence of instructions that transfer data from memory to an external I/O device by using channel 3 of the 8237. The memory area to be transferred is at location 20000H– 20FFFH.

CHAPTER 13

Bus Interface

INTRODUCTION

Many applications require knowledge of the bus systems located within the personal computer. At times, main boards from personal computers are used as core systems in industrial applications. These systems often require custom interfaces that are attached to one of the buses on the main board. This chapter presents the ISA (industry standard architecture) bus, the VESA local bus, the PCI (peripheral component interconnect) bus, the USB (universal serial bus). Also provided are some simple interfaces to many of these bus systems as design guides.

CHAPTER OBJECTIVES

Upon completion of this chapter, you will be able to:

1. Detail the pin connections and signal bus connections on the parallel port and on the ISA, VESA local, and PCI buses.
2. Develop simple interfaces that connect to the parallel port and on the ISA, VESA local, and PCI buses.
3. Program interface places on boards that connect to the ISA, VESA local, and PCI buses.
4. Describe the operation of the USB and develop some short programs that transfer data.

13–1 THE ISA BUS

The ISA, or **industry standard architecture,** bus has been around since the very start of the IBM-compatible personal computer system (circa 1982). In fact, any card from the very first personal computer will plug into and function in any of the most modern Pentium 4-based computers. This is all made possible by the ISA bus interface found in all these machines, which is still compatible with the early personal computers.

Evolution of the ISA Bus

The ISA bus has changed from its early days. Over the years, the ISA bus has evolved from its original 8-bit standard to the 16-bit standard found in most systems today. Along the way, there was even a 32-bit version called the EISA bus **(extended ISA),** but that seems to have all but disappeared. What remains today in most personal computers is an ISA slot **(connection)** on the main board that can accept either an 8-bit ISA card or a 16-bit ISA printed circuit card. The 32-bit printed circuit cards are more often PCI or, in some older 80486-based machines, the VESA cards. The ISA bus has all but vanished recently in home computers, but it is available as a special order in most main boards. The ISA bus is still found in many industrial applications, but its days now seem limited.

The 8-Bit ISA Bus Ouput Interface

Figure 13–1 illustrates the 8-bit ISA connector found on the main board of all personal computer systems (again, this may be combined with a 16-bit connector). The ISA bus connector contains the entire de-multiplexed address bus (A19–A0) for the 1M byte 8088 system, the 8-bit data bus (D7–D0), and the four control signals MEMR, MEMW, IOR, and IOW for controlling I/O and any memory that might be placed on the printed circuit card. Memory is seldom added to any ISA bus card today because the ISA card only operates at an 8 MHz rate. There might be an EPROM or Flash memory used for setup information on some ISA cards, but never any RAM.

Other signals, which are useful for I/O interface, are the interrupt request line IRQ2–IRQ7. Note that IRQ2 is redirected to IRQ9 on modern systems and is so labeled on the connector in Figure 13–1. The DMA channels 0–3 control signals are also present on the connector. The **DMA request inputs** are labeled DRQ1–DRQ3 and the **DMA acknowledge** outputs are labeled DACK0–DACK3. Notice that the DRQ0 input pin is missing because the early personal computers used the DACK0 output as a refresh signal to refresh any DRAM that might be located on the ISA card. Today, this output pin contains a 15.2 μs clock signal. The remaining pins are for power and RESET.

Back of Computer

Pin #	Solder Side	Component Side
1	GND	$\overline{\text{IO CHK}}$
2	RESET	D7
3	+5V	D6
4	IRQ9	D5
5	−5V	D4
6	DRQ2	D3
7	−12V	D2
8	OWS	D1
9	+12V	D0
10	GND	IO RDY
11	$\overline{\text{MEMW}}$	AEN
12	$\overline{\text{MEMR}}$	A19
13	$\overline{\text{IOW}}$	A18
14	$\overline{\text{IOR}}$	A17
15	$\overline{\text{DACK3}}$	A16
16	DRQ3	A15
17	$\overline{\text{DACK1}}$	A14
18	DRQ1	A13
19	$\overline{\text{DACK0}}$	A12
20	CLOCK	A11
21	IRQ7	A10
22	IRQ6	A9
23	IRQ5	A8
24	IRQ4	A7
25	IRQ3	A6
26	$\overline{\text{DACK2}}$	A5
27	T/C	A4
28	ALE	A3
29	+5V	A2
30	OSC	A1
31	GND	A0

FIGURE 13–1 The 8-bit ISA bus.

Suppose that a series of four 8-bit latches must be interfaced to the personal computer for 32 bits of parallel data. This is accomplished by purchasing an ISA interface card (part number 4713-1) from a company like Vector Electronics or other companies. In addition to the edge connector for the ISA bus, the card also contains room at the back for interface connectors. A 37-pin sub-miniature D-type connector can be placed on the back of the card to transfer the 32 bits of data to the external source.

Figure 13–2 shows a simple interface for the ISA bus, which provides 32 bits of parallel TTL data. This example system illustrates some important points about any system interface. First, it is extremely important that the loading to the ISA bus is kept to one low power (LS) TTL load. In this circuit, a 74LS244 buffer is used to reduce the loading on the data bus. If the 74LS244 were not present, this system would present the data bus with four unit loads. If all bus cards were to provide heavy loads, the system would not operate properly (or perhaps not at all).

Output from the ISA card is provided in this circuit by a 37-pin connector labeled P1. The output pins from the circuit connect to P1, and a ground wire is attached. You must provide ground to the outside world, or else the TTL data on the parallel ports are useless. If needed, the output control pins (OC) on each of the 74LS374 latch chips can also be removed from ground and connected to the four remaining pins on P1. This allows an external circuit to control the outputs from the latches.

A small DIP switch is placed on two of the outputs of U7, so the address can be changed if an address conflict occurs with another card. This is unlikely, unless you plan to use two of these cards in the same system. Address connection A2 is not decoded in this system so it becomes a don't care. See Table 13–1 for the addresses of each latch and each position of the S1. Note that only one of the two switches may be on at a time and that each port has two possible addresses for each switch setting because A2 is not connected.

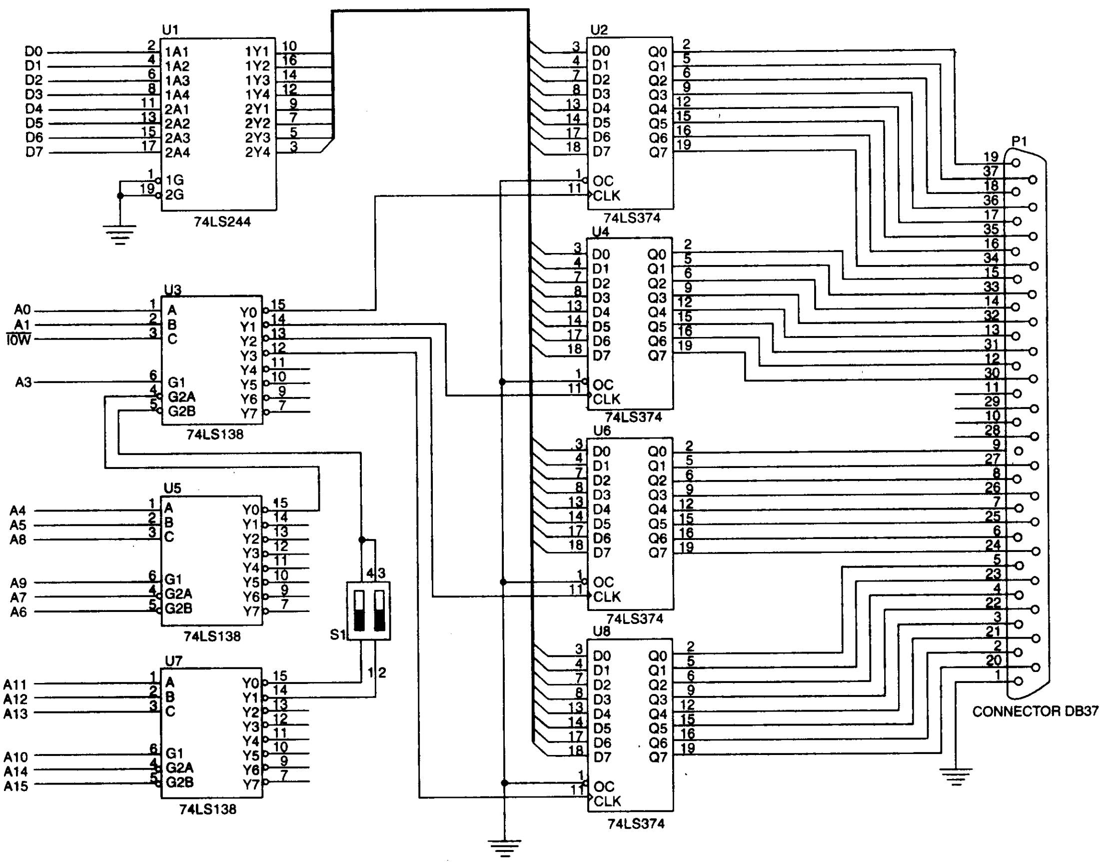


FIGURE 13–2 A 32-bit parallel port interfaced to the 8-bit ISA bus.

TABLE 13–1 The I/O port assignments for Figure 13–2

DIP Switch	*Latch U2*	*Latch U4*	*Latch U6*	*Latch U8*
1–4 On	0608H or 060CH	0609H or 060DH	060AH or 060EH	060BH or 060FH
2–3 On	0E08H or 0E0CH	0E09H or 0E0DH	0E0AH or 0E0EH	0E0BH or 0E0FH

In the personal computer, the ISA bus is designed to operate at I/O address 0000H through 03FFH. Depending on the version and manufacturer of the main-board, ISA cards may or may not function above these locations. Newer systems often allow ISA ports at locations above 03FFH, but older systems may not. The ports in this example may need to be changed for some systems. Some older cards only decode I/O addresses 0000H–03FFH and may have address conflicts if the port addresses above 03FFH conflict. The ports are decoded in this example by three 74LS138 decoders. It would be more efficient and cost-effective to decode the ports with a programmable logic device.

Figure 13–3 shows the circuit of Figure 13–2 reworked using a PAL16L8 to decode the addresses for the system. Notice that address bits A15–A4 are decoded by the PAL and the switch is connected to two of the PAL inputs.

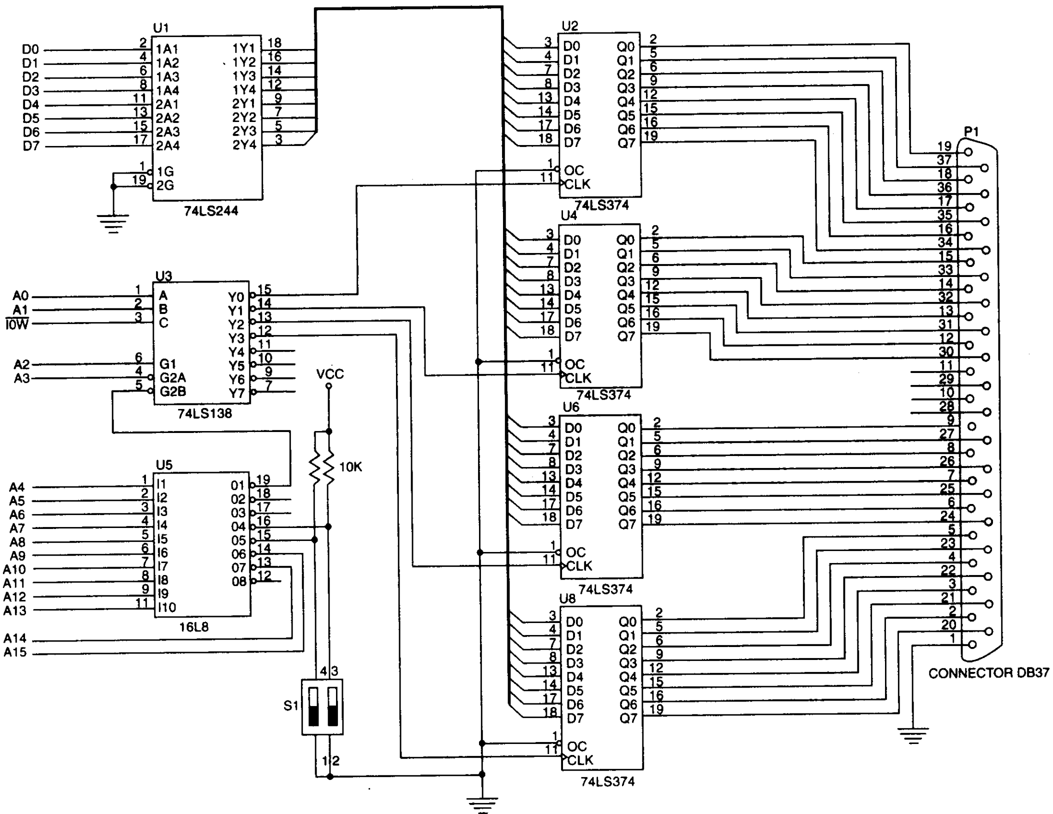


FIGURE 13–3 A 32-bit parallel port interfaced to the 8-bit ISA bus using a PAL16L8 for a decoder.

TABLE 13–2 Port assignments in Figure 13–3.

Switch 1–4	*Switch 2–3*	*Latch U2*	*Latch U4*	*Latch U6*	*Latch U8*
Off	Off	0604H	0605H	0606H	0607H
Off	On	0624H	0625H	0626H	0627H
On	Off	0644H	0645H	0646H	0647H
On	On	0664H	0665H	0666H	0667H

Note: On = a closed switch and Off = an open switch.

This change allows four different I/O port addresses for each latch, making the circuit more flexible. Table 13–2 shows the port number selected by switch 1–4 and switch 2–3. Example 13–1 shows the program for the PAL16L8 that causes the port assignments of Table 13–2.

EXAMPLE 13–1

```
CHIP      ISA1 PAL16L8
;pins  1  2  3  4  5  6  7   8   9   10
       A4 A5 A6 A7 A8 A9 A10 A11 A12 GND

;pins  11  12 13  14  15 16 17 18 19  20
      A13 NC A14 A15 S1 S2 NC NC DC VCC
```

```
EQUATIONS

/DC = /A15*/A14*/A13*/A12*/A11*A10*A9*/A8*/A7*/S1*/S2*/A6*/A5*/A4
    + /A15*/A14*/A13*/A12*/A11*A10*A9*/A8*/A7*/S1*S2*/A6*A5*/A4
    + /A15*/A14*/A13*/A12*/A11*A10*A9*/A8*/A7*S1*/S2*A6*/A5*/A4
    + /A15*/A14*/A13*/A12*/A11*A10*A9*/A8*/A7*S1*S2*A6*A5*/A4
```

Notice in Example 13–1 how the first product term generates a logic 0 on the output to the decoder only when both switches are in their off positions for I/O ports 0600H–060FH. The 74LS138 further refines the ports to 604H, 605H, 606H, or 607H for the latches. The second product term is active when switch 1–4 is off and switch 2–3 is on. The other two combinations on the switch select the last two port address assignments in Table 13–2.

Example 13–2 shows a small program that sends data to the ports in a pattern that could be used for testing. The pattern selected places a logic 1 on bit 0 of U2, and all zeros on the remaining latches. This pattern is then rotated through all 32 bits until one minute has elapsed. The timing is handled by the counter located at the 32-bit memory location starting at 0000:046C, which increments 18.2 times per second. This test program assumes that the latches appear at I/O ports 0604H–0607H, as selected by the switches.

EXAMPLE 13–2

```
                        ;A program that sends a test pattern to the I/O ports of
                        ;Figure 13-3.
                        ;
                        .MODEL TINY
0000                    .CODE
                        .STARTUP
0100  B8 0000                MOV   AX,0
0103  8E D8                  MOV   DS,AX          ;address segment 0000H
0105  BB 0001                MOV   BX,1           ;setup starting bit pattern
0108  BA 0000                MOV   DX,0           ;in registers DX-BX
010B  B9 0444                MOV   CX,1092        ;set count for 1 minute
                             .REPEAT
010E  BA 0604                  MOV   DX,0604H     ;address latch U2
0111  8A C3                    MOV   AL,BL        ;send BL to U2
0113  EE                       OUT   DX,AL
0114  42                       INC   DX           ;address latch U4
0115  8A C7                    MOV   AL,BH        ;send BH to U4
0117  EE                       OUT   DX,AL
0118  42                       INC   DX           ;address latch U6
0119  8A C2                    MOV   AL,DL        ;send DL to U6
011B  EE                       OUT   DX,AL
011C  42                       INC   DX           ;address latch U8
011D  8A C6                    MOV   AL,DH        ;send DH to U8
011F  D1 E2                    SHL   DX,1         ;rotate number in DX-BX
0121  D1 D3                    RCL   BX,1
                               .IF CARRY?
0125  83 C2 01                 ADD   DX,1
                               .ENDIF
0128  B8 046C                  MOV   AX,[46CH]     ;get counter
012B  BD 046E                  MOV   BP,[46EH]
012E  40                       INC   AX
012F  83 D5 00                 ADC   BP,0
                               .REPEAT
                               .UNTIL    AX==[46CH] && BP==[46EH]
                             .UNTILCXZ
                        .EXIT
                             END
```

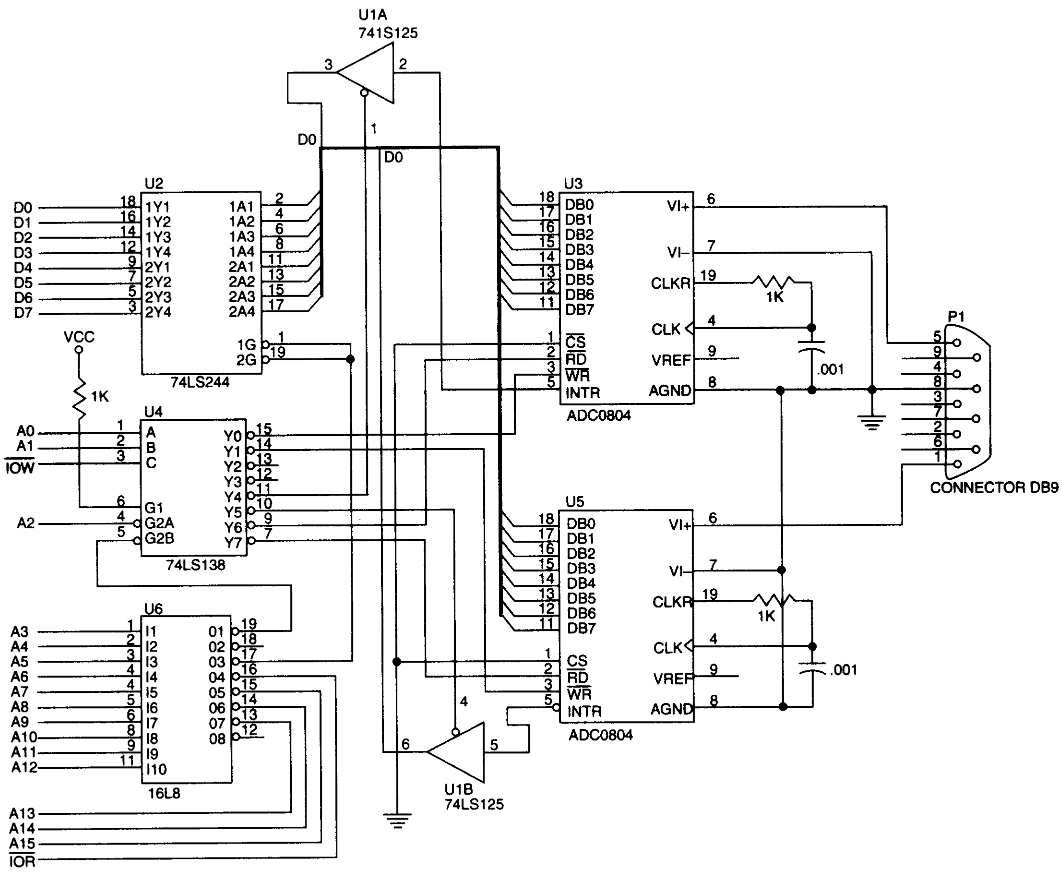


FIGURE 13–4 A pair of analog-to-digital converters interfaced to the ISA bus.

The 8-Bit ISA Bus Input Interface

To illustrate the input interface to the ISA bus, a pair of ADC804 analog-to-digital converters are interfaced to the ISA bus in Figure 13–4. The connections to the converters are made through a 9-pin DB9 connector. The task of decoding the I/O port addresses is more complex, because each converter needs a write pulse to start a conversion, a read pulse to read the digital data once it has been converted from the analog input data, and a pulse to enable the selection of the INTR output. Notice that the INTR output is connected to data bus bit position D0. When INTR is input to the microprocessor, the rightmost bit of AL is tested to see whether the converter is busy.

As before, great care is taken so that the connections to the ISA bus present one unit load to the system. Table 13–3 illustrates the I/O port assignment decoded by the PAL16L8 in Example 13–3.

TABLE 13–3 I/O port assignments for Figure 13–4.

Device	*Port Number*
Start ADC (U3)	1200H
Read INTR (U3)	1200H
Read ADC (U3)	1202H
Start ADC (U5)	1201H
Read INTR (U5)	1201H
Read ADC (U5)	1203H

EXAMPLE 13–3

```
CHIP      ISA2 PAL16L8

;pins  1  2  3  4  5  6  7  8   9   10
       A3 A4 A5 A6 A7 A8 A9 A10 A11 GND

;pins 11  12 13  14  15  16  17 18 19  20
      A12 NC A13 A14 A15 IOR G  NC DC VCC

EQUATIONS

/DC = /A15*/A14*/A13*A12*/A11*/A10*A9*/A8*/A7*/A6*/A5*/A4*/A3
/G  = /A15*/A14*/A13*A12*/A11*/A10*A9*/A8*/A7*/A6*/A5*/A4*/A3*/IOR
```

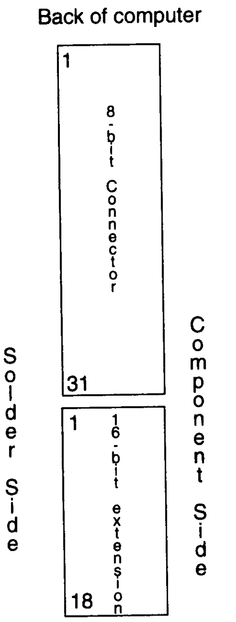
Example 13–4 lists a macro that can be used to read either ADC U3 or U5. The address is generated by passing either a 0 for U3 or a 1 for U5 to the macro as a parameter. The macro starts the converter by writing to it, and then waits until the INTR pin returns to a logic 0, indicating that the conversion is complete before the data are read and returned in the AL register.

EXAMPLE 13–4

```
;A macro that operates either converter 0 or converter 1 and
;returns the digital data in AL.  Note that converter 0 is U3
;and converter 1 is U5.
;
ADC    MACRO    WHICH
       MOV      DX,1200H
       ADD      DX,WHICH          ;address converter
       OUT      DX,AL             ;start converter
       .REPEAT
          IN       AL,DX          ;get busy signal
          TEST     AL,1
       .UNTIL   ZERO?
       ADD      DX,2              ;address data port
       IN       AL,DX
       ENDM
```

The 16-Bit ISA Bus

The only difference between the 8- and 16-bit ISA bus is that an additional connector is attached behind the 8-bit connector. A 16-bit ISA card contains two edge connectors: one plugs into the original 8-bit connector and the other plugs into the new 16-bit connector. Figure 13–5 shows the pin-out of the additional connector and its placement in the computer in relation to the 8-bit connector. Unless additional memory is added on the ISA card, the extra address connections A23– A20 do not serve any function for I/O operations. The added features that are most often used are the additional interrupt request inputs and the DMA request signals. In some systems, 16-bit I/O uses the additional eight data bus connections (D8–D15), but more often today, the EISA bus, VESA local bus, or PCI bus is used for peripherals that are wider than eight bits. The ISA bus is beginning to be used less than it once was and will probably be replaced in the near future by the USB **(universal serial bus).** About the only recent interfaces found for the ISA bus are modems and sound cards. Modems are serial devices that lend themselves well to the USB; sound cards are affected by noise. The computer power supply is a switching supply that generates noise that cannot be filtered. If the sound function is placed external to the computer with its own separate power supply, this annoying noise can be removed from the audio signal. The sound card will benefit greatly when placed on the USB.



(a)

Back of computer

Solder Side | Component Side

Pin#		
1	$\overline{MCS16}$	BHE
2	$\overline{IOCS16}$	A23
3	IRQ10	A22
4	IRQ11	A21
5	IRQ12	A20
6	IRQ15	A19
7	IRQ14	A18
8	$\overline{DACK0}$	A17
9	DRQ0	$\overline{MEMR}$
10	$\overline{DACK5}$	$\overline{MEMW}$
11	DRQ5	D8
12	$\overline{DACK6}$	D9
13	DRQ6	D10
14	$\overline{DACK7}$	D11
15	DRQ7	D12
16	+5V	D13
17	$\overline{MASTER}$	D14
18	GND	D15

(b)

FIGURE 13–5 The 16-bit ISA bus. (a) Both 8- and 16-bit connectors and (b) the pin-out of the 16-bit connector.

13–2 THE EXTENDED ISA (EISA) AND VESA LOCAL BUS ARCHITECTURES

The **extended ISA architecture** (EISA) is a 32-bit modification to the ISA bus. As computers became larger and had wider data buses (80386–Pentium 4), a new bus was needed that would transfer 32-bit data. Although the EISA bus seems to be fading, it is a steppingstone in the evolution of the computer system bus. The main problem with the EISA bus is that even though the data bus width is increased to 32 bits, the clocking speed remains at 8 MHz, which is why this interface standard has all but vanished. Note that the newer VESA local bus and PCI bus both operate at a higher speeds. The VESA local and PCI buses both presently operate at 33 MHz.

The most common application for the EISA bus is as a disk controller or as a video graphics adapter. These applications benefit from the wider data bus width because the data transfer rates for these devices are high.

EISA Bus Pin-Out

One interesting change from ISA to EISA is that the pin spacing is 0.05" instead of 0.1", as on the ISA bus edge connector. The new pins for the EISA bus are interspersed with the older pins in the standard 16-bit ISA connector pair. This makes insertion a little difficult, but it preserves compatibility with the older ISA standard. Figure 13–6 illustrates the pin-out of the EISA bus connectors and details the way that this new set of connectors is placed with the old ISA connector. Most of the new EISA connections are used for a 32-bit data bus and a latched 32-bit address bus. Also present are the bank enable signals used in the 80386 and 80486 microprocessors. The EISA standard does not support the 64-bit data bus found in the Pentium–Pentium 4 microprocessors.

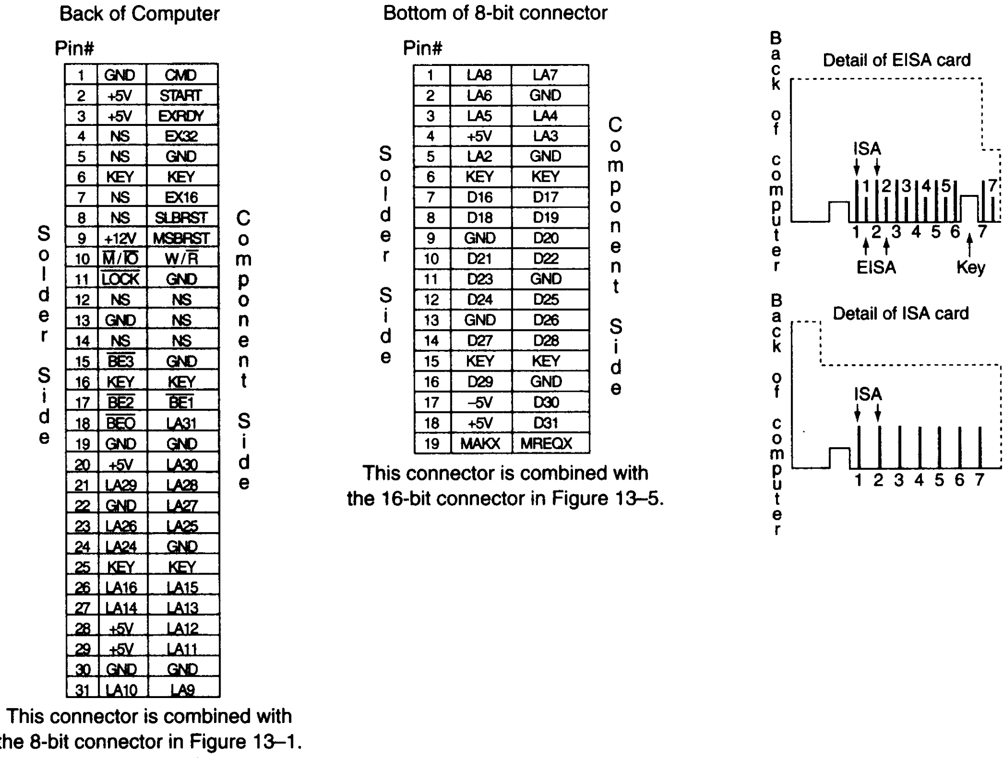


FIGURE 13–6 The EISA connectors and detail comparing the EISA card with the ISA card.

EISA Bus Interface Example

Figure 13–7 shows the interface to a 32-bit events counter. This system uses four 74LS590 octal (8-bit) binary counters with built-in latches. Each counter also has a direct clear input that is used to asynchronously clear the counter before any events are counted. The counter itself is wired as a 32-bit synchronous counter that uses ripple output to enable the next octal counter when the counter rolls over to 00000000, so the next stage counts up by one.

A PAL16L8 and 74LS138 decoder are used to decode the I/O addresses at ports 0308H, 0309H, and 030CH. Port 0308H is used to clear the counter, port 0309H latches the count, and port 030CH enables the three-state output buffers to apply the contents of the 32-bit latch to the microprocessor data bus. Port 030CH is used because 32-bit data should fall at a 32-bit address boundary for proper and efficient I/O operation. Note that the PAL16L8 program is not illustrated for this example because it is essentially the same as other examples in this chapter.

Events Counter. Example 13–5 illustrates software required to use this system as an events counter. Because the counter is 32-bits wide, it can accumulate up to 4 billion or so events before the count becomes incorrect. This is large enough for most applications that require event counts.

EXAMPLE 13–5

```
                .MODEL TINY
                .386
0000            .CODE
                .STARTUP
                ;
                ;Procedure that starts the events counter.
                ;
0100            START   PROC    NEAR
```

```
0100  BA 1308          MOV   DX,1308H     ;address the clear port
0103  EC               IN    AL,DX        ;clear the counter
0104  C3               RET

0105           START   ENDP
               ;
               ;Procedure that reads the count and returns it in EAX.
               ;
0105           READC   PROC  NEAR

0105  BA 1309          MOV   DX,1309H     ;address the latch port
0108  EC               IN    AL,DX        ;latch the count
0109  BA 130C          MOV   DX,130CH     ;address the count port
010C  66| ED           IN    EAX,DX       ;read the count to EAX
010E  C3               RET

010F           READC   ENDP
                       END
```

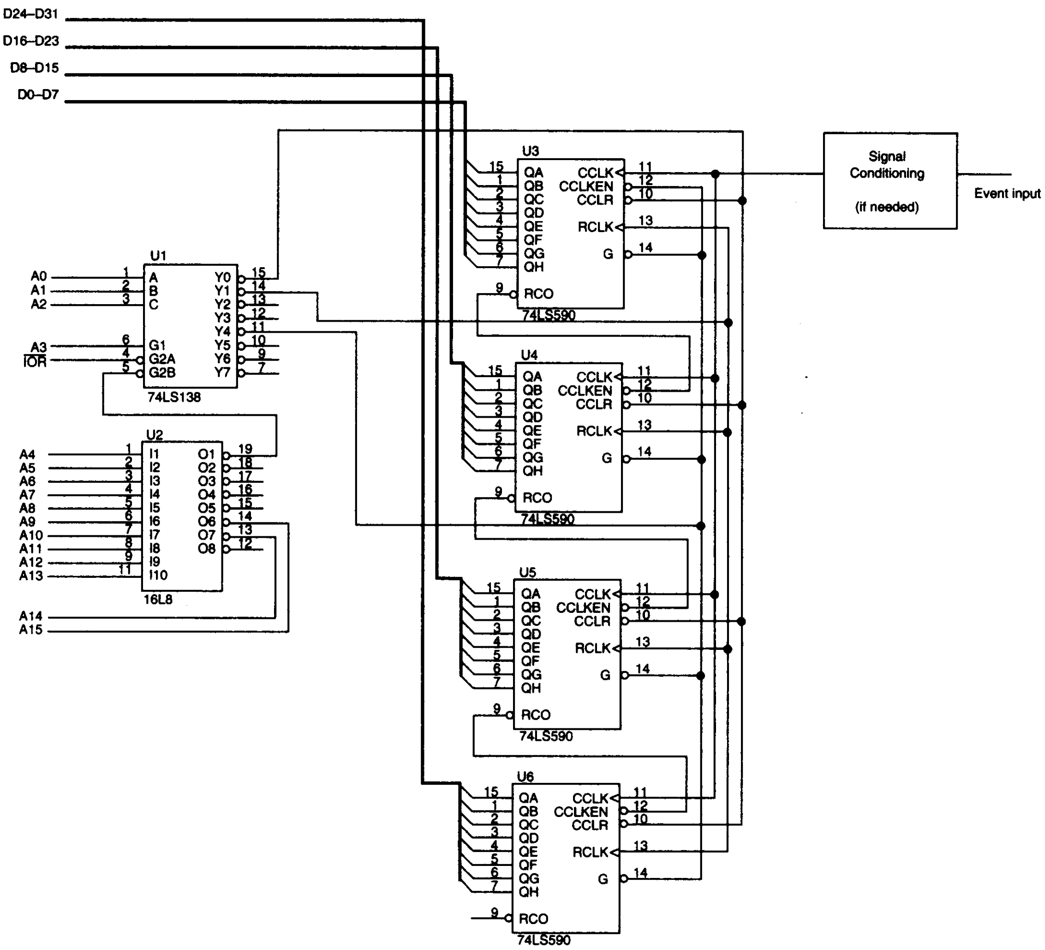


FIGURE 13–7 A 32-bit events and general purpose counter interfaced to the EISA bus.

The only part of this interface and software that appears odd is that an IN instruction is used to clear the counter and to latch the count. The reason is that no data are transferred between the microprocessor and the counter for these events, and it made the circuitry a little less complicated.

Frequency Counter. Another application for this same circuit is a frequency counter. The only difference between counting events and counting a frequency is the way the circuit is operated. For example, suppose that the counter is cleared and then read exactly one second later. The counter contains a number that represents the frequency at the events input. If a frequency is needed in kilo-Hertz range instead of Hertz, the counter is reset and then read exactly one ms later.

There are some limitations to the frequency, which can be measured by this circuitry. The accuracy of the measurement can be off by one count. The highest frequency is due to the speed of the counter and the conditioning circuitry, if needed. The 74LS590 can count to at least 32 MHz. If the input is purely TTL (no conditioning), then this circuit can measure frequency up to 32 MHz. The conditioning circuit chosen may also limit the upper frequency limit. The accuracy is determined by the time between the reset and the latching of the count. If the clock in the computer system is very accurate, the frequency is very accurate. It is assumed that the clock in the personal computer is about 0.1 percent accurate, which would be the accuracy of any frequency measured by this system. This is good enough for many applications that require frequency measurements.

Example 13–6 shows how the system can measure a frequency if you are willing to wait for one second. Notice how the SYNC macro is used to synchronize to the clock tick and how the result is scaled by multiplying the count by 18.2. To obtain a more accurate reading, the scaling factor could be changed by measuring the frequency with a laboratory frequency counter and then modifying the contents of NUM. The sample time (point where the clear and latch operations occur) can be reduced to make more samples per second.

EXAMPLE 13–6

```
                        .MODEL SMALL
                        .386
                        .387
0000                    .DATA
0000  4191999A          NUM    DD  18.2
0004  00000000          TEMP   DD  ?
0000                    .CODE
                        .STARTUP
                        ;
                        ;A simple frequency counter.
                        ;
                        ;This procedure returns the frequency in EAX.
                        ;The accuracy should be fairly good.
                        ;
                        SYNC   MACRO               ;sync to clock tick
                               MOV    AH,0
                               INT    16H          ;get clock tick information into CX:DX
                               MOV    BX,CX
                               MOV    BP,DX
                               .REPEAT             ;sync to clock tick
                                  MOV    AH,0
                                  INT    16H
                               .UNTIL CX!=BX || DX!=BP
                               ENDM
```

```
0010                FREQ    PROC    NEAR

                            SYNC                ;sync to clock tick

0024  BA 1308               MOV     DX,1308H    ;address clear counter port
0027  EC                    IN      AL,DX       ;clear counter

                            SYNC                ;wait for next tick

003C  BA 1309               MOV     DX,1309H    ;address latch port
003F  EC                    IN      AL,DX       ;latch count
0040  BA 130C               MOV     DX,130CH    ;address counter
0043  66| ED                IN      EAX,DX      ;get count
0045  66| A3 0004 R         MOV     TEMP,EAX
0049  DB 06 0004 R          FILD    TEMP        ;convert to Hertz
004D  D8 0E 0000 R          FMUL    NUM
0051  DB 1E 0004 R          FISTP   TEMP        ;save as integer
0055  66| A1 0004 R         MOV     EAX,TEMP
0059  C3                    RET
005A                FREQ    ENDP
                            END
```

13–3 THE PARALLEL PRINTER INTERFACE (LPT)

The parallel printer interface (LPT) is located on the rear of the personal computer, and as long as it is a part of the PC it can be used as an interface to the PC. LPT stands for Line Printer. The printer interface gives the user access to eight lines that can be programmed to receive or send data.

Port Details

The parallel port (LPT1) is normally at I/O port addresses 378H, 379H, and 37AH. The secondary (LPT2) port, if present, is located at I/O port addresses 278H, 279H, and 27AH. The following information applies to both ports, but LPT1 port addresses are used throughout.

The Centronics Interface implemented by the parallel port uses two connectors, a 25-pin D-type on the back of the PC and a 36-pin Centronics on the back of the printer. The pin-outs of these connectors are listed in Table 13–4, and the connectors are shown in Figure 13–8.

The parallel port can work as both a receiver and a transmitter at its data pins (D0–D7). This allows devices other than printers, such as CD-ROMs, to be connected to and used by the PC through the parallel port. Anything that can receive and/or send data through an 8-bit interface can and often does connect to the parallel port (LPT1) of a PC.

Figure 13–9 illustrates the contents of the data port (378H), the status register (379H), and an additional status port (37AH). Some of the status bits are true when they are a logic zero.

Using the Parallel Port Without ECP Support

In most systems since the PS/2 was released by IBM, you can basically follow the information presented in Figure 13–9 to use the parallel port without ECP. To read the port, you must first intialize it by sending a 20H to register 37AH as illustrated in Example 13–7.

TABLE 13–4 Parallel Port (LPT) pin and signal connections.

Signal	*Description*	*25-pin*	*36-pin*
#STR	Strobe to printer	1	1
D0	Data bit 0	2	2
D1	Data bit 1	3	3
D2	Data bit 2	4	4
D3	Data bit 3	5	5
D4	Data bit 4	6	6
D5	Data bit 5	7	7
D6	Data bit 6	8	8
D7	Data bit 7	9	9
#ACK	Acknowledge from printer	10	10
BUSY	Busy from printer	11	11
PAPER	Out of paper	12	12
ONLINE	Printer is online	13	13
#ALF	Low if printer issues a LF after a CR	14	14
#ERROR	Printer error	15	32
#RESET	Resets the printer	16	31
#SEL	Selects the printer	17	36
+5V	5V from printer	—	18
Protective Ground	Earth ground	—	17
Signal Ground	Signal Ground	All other pins	All other pins

Note: # indicates an active low signal.

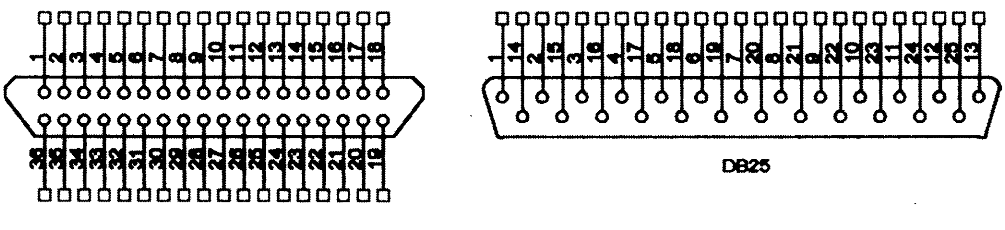


FIGURE 13–8 The connectors used for the parallel port.

EXAMPLE 13–7

```
MOV     AL,20H
MOV     DX,37AH
OUT     DX,AL
```

Once the port is initialized, it can be read as illustrated in Example 13–8. Notice that reading the port is very easy.

EXAMPLE 13–8

```
MOV     DX,378H
IN      AL,DX
```

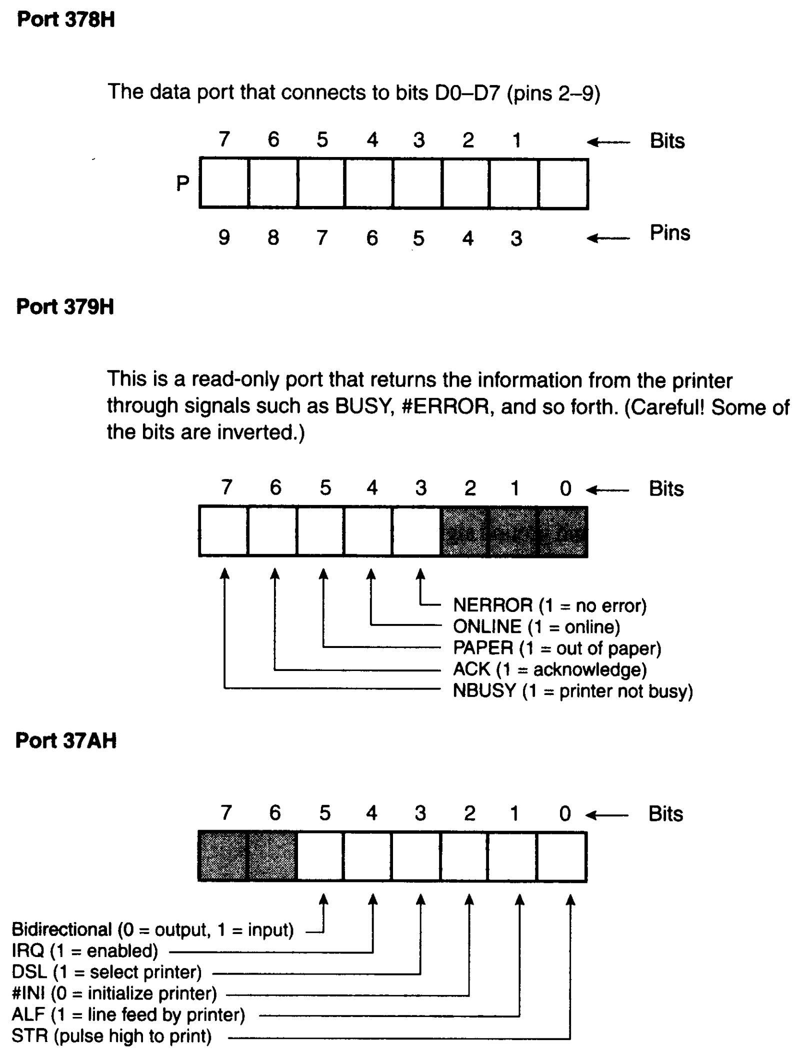


FIGURE 13–9 Ports 378H, 379H, and 37AH as used by the parallel port.

To write data to the port, it must be initialized as in Example 13–7, but instead of sending it a 20H we send it a 00H to set it up for writing data. Example 13–9 shows how to write data to the parallel port.

EXAMPLE 13–9

```
MOV     DX,378H
MOV     AL,WRITE_DATA
OUT     DX,AL
```

On some older systems without a bidirectional bit you may need to output an 0FFH to port 378H before you begin to input information. This action takes the place of programming the control register 37AH. Once the FFH is written, you can read the port.

13–4 THE UNIVERSAL SERIAL BUS (USB)

The universal serial bus (USB) has solved a problem with the personal computer system. The current ISA sound cards use the internal PC power supply, which generates a tremendous amount of noise. Because the USB allows the sound card to have its own power supply, the noise associated with the PC power supply can be eliminated, allowing for high-fidelity sound. Other benefits are ease of user connection and access to up to 127 different connections through a four-connection serial cable. This interface is ideal for keyboards, sound cards, simple video-retrieval devices, and modems. Data transfer speeds are 12 Mbps for full-speed operation and 1.5 Mbps for slow-speed operation.

Cable lengths are limited to five meters maximum for the full-speed interface and three meters maximum for the low-speed interface. The maximum power available through these cables is rated at 100 mA maximum current at 5.0 V. If the amount of current exceeds 100 mA, Windows will display a yellow exclamation point next to the device.

The Connector

Figure 13–10 illustrates the pin-out of the USB connector. There are two types of female connectors specified and both are in use. In either case, there are four pins on each connector, which contain the signals indicated in Table 13–5. As mentioned, the +5.0 V and ground signals can be used to power devices connected to the bus as long as the amount of current does not exceed 100 mA per device. The data signals are biphase signals. When +data are at 5.0 V, –data are at zero volts and vice versa.

USB Data

The data signals are biphase signals that are generated using a circuit such as the one illustrated in Figure 13–11. The line receiver is also illustrated in Figure 13–11. Placed on the transmission pair is a noise-suppression circuit that is available from Texas Instruments (SN75240). Once the transceiver is in place, interfacing to the USB is

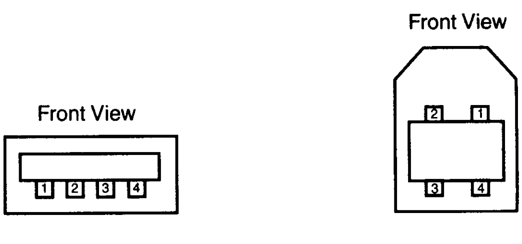


FIGURE 13–10 The front view of the two common types of USB connectors.

TABLE 13–5 USB pin configuration.

Pin Number	*Signal*
1	5.0 V
2	– Data
3	+ Data
4	Ground

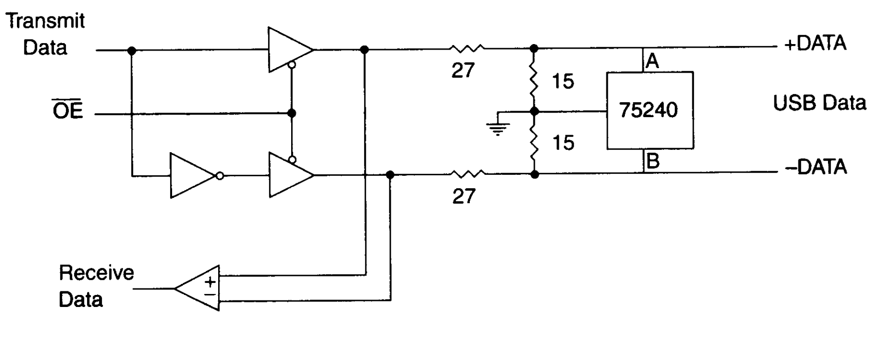


FIGURE 13–11 The interface to the USB using a pair of CMOS buffers.

complete. The 75773 integrated circuit from Texas Instruments functions as both the differential line driver and receiver for this schematic.

The next phase is learning how the signals interact on the USB. These signals allow data to be sent and received from the host computer system. The USB uses NRZI (non-return to zero, inverted) data encoding for transmitting packets. This encoding method does not change the signal level for the transmission of a logic 1, but the signal level is inverted for each change to a logic 0. Figure 13–12 illustrates a digital data stream and the USB signal produced using this encoding method.

The actual data transmitted includes sync bits using a method called *bit stuffing*. If a logic 1 is transmitted for more than six bits in a row, the bit stuffing technique adds an extra bit (logic 0) after six continuous 1s in a row. Because this lengthens the data stream, it is called bit stuffing. Figure 13–13 shows a bit-stuffed serial data stream

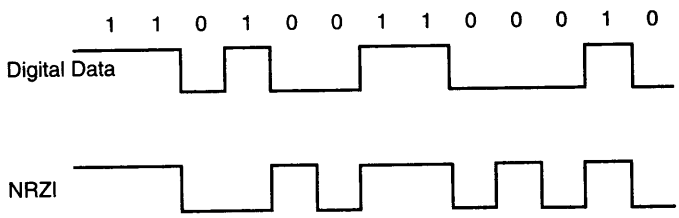


FIGURE 13–12 NRZI encoding used with the USB.

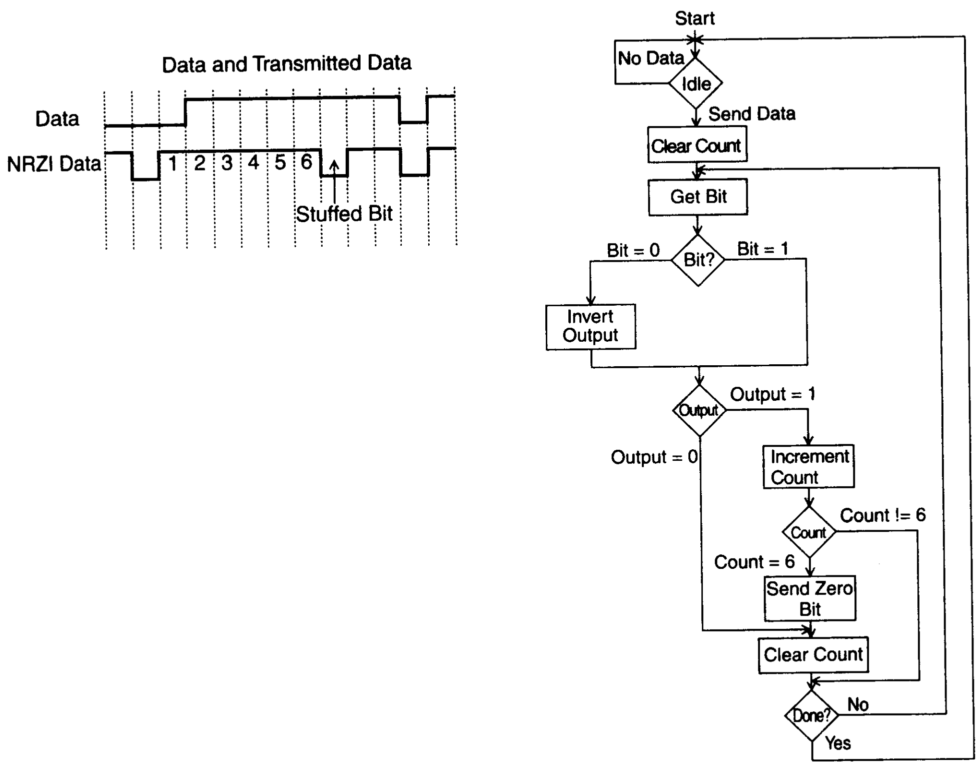


FIGURE 13–13 The data stream and the flow chart used to generate USB data.

and the algorithm used to create it from raw digital serial data. Bit stuffing ensures that the receiver can maintain synchronization for long strings of 1s. Data are always transmitted beginning with the least-significant bit first, followed by subsequent bits.

USB Commands

Now that the USB data format is understood, the commands used to transfer data and select the receptor are discussed. To begin communications, the sync byte (80H) is transmitted first, followed by the packet identification byte (PID). The PID contains eight bits, but only the rightmost four bits contain the type of packet that follows, if any. The leftmost four bits of the PID are the ones complementing the rightmost four bits. For example, if a command of 1000 is sent, the actual byte sent for the PID is a 0111 1000. Table 13–6 shows the available 4-bit PIDs and their 8-bit codes. Notice that there are PIDs used as token indicators, as data indicators, and for handshaking.

Figure 13–14 lists the formats of the data, token, handshaking, and start-of-frame packets found on the USB. In the token packet, the ADDR (address field) contains the 7-bit address of the USB device. As mentioned earlier, there are up to 127 devices present on the USB at a time. The ENDP (endpoint) is a 4-bit number used by the USB. Endpoint 0 is used for initialization, while other endpoint numbers are unique to each USB device.

There are two types of CRC (cyclic redundancy checks) used on the USB: one is a 5-bit CRC and the other (used for data packets) is a 16-bit CRC. The 5-bit CRC is generated with the $X^5 + X^2 + 1$ polynomial; the 16-bit CRC is generated with the $X^{16} + X^{15} + X^2 + 1$ polynomial. When constructing circuitry to generate or detect the CRC, the plus signs represent exclusive-OR circuits. Note that a CRC is a serial checking mechanism. When using the 5-bit CRC, a residual of 01100 is received for no error in all five bits of the CRC and the data bits. With the 16-bit CRC, the residual is 1000000000001101 for no error.

The USB uses the ACK and NAK tokens to coordinate the transfer of data packets between the host system and the USB device. Once a data packet is transferred from the host to the USB device, the USB device either transmits an ACK (acknowledge) or a NAK (not acknowledge) token back to the host. If the data and CRC are received correctly, the ACK is sent; if not, the NAK is sent. If the host receives a NAK token, it retransmits the data packet until the receiver finally receives it correctly. This method of data transfer is often called **stop and wait flow control.** The host must wait for the client to send an ACK or NAK before transferring additional data packets.

TABLE 13–6 PID codes.

PID	*Name*	*Type*	*Description*
E1H	OUT	Token	Host ∅ function transaction
D2H	ACK	Handshake	Receiver accepts packet
C3H	Data0	Data	Data packet PID even
A5H	SOF	Token	Start of frame
69H	IN	Token	Function ∅ host transaction
5AH	NAK	Handshake	Receiver does not accept data
4BH	Data1	Data	Data packet PID odd
3CH	PRE	Special	Host preamble
2DH	Setup	Token	Setup command
1EH	Stall	Token	Stalled

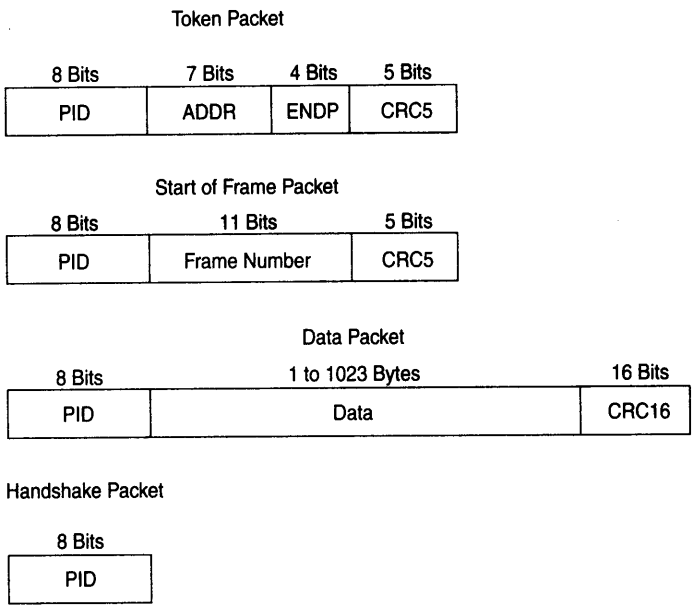


FIGURE 13–14 The types of packets and contents found on the USB.

13–5 SUMMARY

1. The bus systems (ISA, EISA, VESA, PCI, and USB) allow I/O and memory systems to be interfaced to the personal computer.
2. The ISA bus is either 8- or 16-bits, and supports either memory or I/O transfers at rates of 8 MHz.
3. The EISA bus is an extended version of the ISA bus that supports 8-, 16-, and 32-bit transfers between the personal computer and memory or I/O at rates of 8 MHz.
4. The VESA (Video Electronics Standards Association) local bus supports 32-bit transfers between the personal computer and I/O or memory at rates of 33 MHz.
5. The universal serial bus (USB) promises to replace the ISA bus in the most advanced system. The USB has two data transfer rates: 1.5 Mbps and 12 Mbps.
6. The USB uses the NRZI system to encode data, and uses bit stuffing for logic 1 transmission more than six bits long.

13–6 QUESTIONS AND PROBLEMS

1. The letters ISA are an acronym for what phrase?
2. The ISA bus system supports what size data transfers?
3. Is the ISA bus interface often used for memory expansion?

4. What data rates are available for use on the USB?
5. How are data encoded on the USB?
6. What is the maximum cable length for use with the USB?
7. Will the USB ever replace the ISA bus?
8. How many device addresses are available on the USB?
9. What is NRZI encoding?
10. What is a stuffed bit?
11. If the following raw data are sent on the USB, draw the waveform of the signal found on the USB: (110011000011001101 1010)
12. What is the maximum length of a data packet on the USB.
13. What is the purpose of the NAK and ACK tokens on the USB?

CHAPTER 14

The 80186, 80188, and 80286 Microprocessors

INTRODUCTION

The Intel 80186/80188 and the 80286 are enhanced versions of the earlier 8086/8088 microprocessors. The 80186/80188 and 80286 are all 16-bit microprocessors that are upward-compatible to the 8086/8088. Even the hardware of these microprocessors is similar to the earlier versions. This chapter presents an overview of each microprocessor, and points out the differences or enhancements that are present in each version. The first part of the chapter describes the 80186/80188 microprocessors, and the last part shows the 80286 microprocessor.

New to this edition is an expanded coverage of the 80186/80188 family. Intel has added four new versions of each of these embedded controllers to its lineup of microprocessors. Each is a CMOS version and designated with a two-letter suffix: XL, EA, EB, and EC. The 80C186XL and 80C188XL models are most similar to the earlier 80186/80188 models.

CHAPTER OBJECTIVES

Upon completion of this chapter, you will be able to:

1. Describe the hardware and software enhancements of the 80186/80188 and the 80286 microprocessors as compared to the 8086/8088.
2. Detail the differences between the various versions of the 80186 and 80188 embedded controllers.
3. Interface the 80186/80188 to memory and I/O.
4. Develop software using the enhancements provided in these microprocessors.

14–1 80186/80188 ARCHITECTURE

The 80186 and 80188, like the 8086 and 8088, are nearly identical. The only difference between the 80186 and 80188 is the width of their data buses. The 80186 (like the 8086) contains a 16-bit data bus, while the 80188 (like the 8088) contains an 8-bit data bus. The internal register structure of the 80186/80188 is virtually identical to the 8086/8088. About the only difference is that the 80186/80188 contain additional reserved interrupt vectors and some very powerful built-in I/O features. The 80186 and 80188 are often called **embedded controllers** because of their application as a controller, not as a microprocessor-based computer.

TABLE 14–1 The four versions of the 80186/80188 embedded controller.

Feature	*80C186XL 80C188XL*	*80C186EA 80C188EA*	*80C186EB 80C188EB*	*80C186EC 80C188EC*
80286-like instruction set	✔	✔	✔	✔
Power-save (green mode)	✔	✔		✔
Power down mode		✔	✔	✔
80C187 interface	✔	✔	✔	✔
ONCE mode	✔	✔	✔	✔
Interrupt controller	✔	✔	✔	✔ 8259-like
Timer unit	✔	✔	✔	✔
Chip selection unit	✔	✔	✔ enhanced	✔ enhanced
DMA controller	✔ 2-channel	✔ 2-channel		✔ 4-channel
Serial communications unit			✔	✔
Refresh controller	✔	✔	✔ enhanced	✔ enhanced
Watchdog timer				✔
I/O ports			✔ 16-bits	✔ 22-bits

Versions of the 80186/80188

As mentioned, the 80186 and 80188 are available in four different versions, which are all CMOS microprocessors. Table 14–1 lists each version and the major features provided. The 80C186XL and 80C188XL are the most basic versions of the 80186/80188, while the 80C186EC and 80C188EC are the most advanced. This text details the 80C186XL/80C188XL, and then describes the additional features and enhancements provided in the other versions.

80186 Basic Block Diagram

Figure 14–1 provides the block diagram of the 80188 microprocessor that generically represents all versions except for the enhancements and additional features outlined in Table 14-1. Notice that this microprocessor has a great deal more internal circuitry than the 8088. The block diagrams of the 80186 and 80188 are identical except for the pre-fetch queue, which is four bytes in the 80188 and six bytes in the 80186. Like the 8088, the 80188 contains a bus interface unit (BIU) and an execution unit (EU).

In addition to the BIU and EU, the 80186/80188 family contains a clock generator, a programmable interrupt controller, programmable timers, a programmable DMA controller, and a programmable chip selection unit. These enhancements greatly increase the utility of the 80186/ 80188 and reduce the number of peripheral components required to implement a system. Many popular subsystems for the personal computer use the 80186/80188 microprocessors as caching disk controllers, local area network (LAN) controllers, etc. The 80186/80188 also finds application in the cellular telephone network as a switcher.

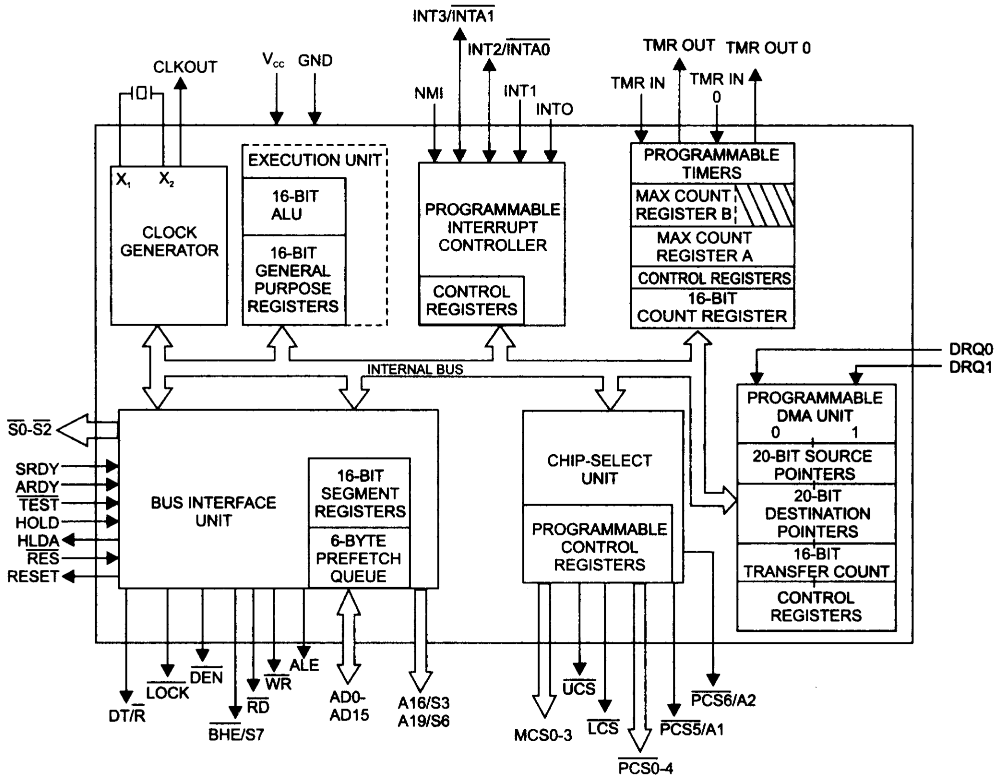


FIGURE 14–1 The block diagram of the 80186 microprocessor. Note that the block diagram of the 80188 is identical, except that BHE/S7 is missing and AD15–AD8 are relabeled A15–A8. (Courtesy of Intel Corporation.)

Software for the 80186/80188 is identical to the 80286 microprocessor, without the memory management instructions. This means that the 80286-like instructions immediate multiplication, immediate shift counts, string I/O, PUSHA, POPA, BOUND, ENTER, and LEAVE all function on the 80186/80188 microprocessors.

80186/80188 Basic Features

In this segment of the text, we introduce the enhancements of the 80186/80188 microprocessors or embedded controllers that apply to all versions except where noted, but we do not provide an exclusive coverage. More details on the operation of each enhancement and details of each advanced version are provided later in the chapter.

Clock Generator. The internal clock generator replaces the external 8284A clock generator used with the 8086/8088 microprocessors. This reduces the component count in a system.

The internal clock generator has three pin connections: X1, X2, and CLKOUT (or on some versions: CLKIN, OSCOUT, and CLKOUT). The X1 (CLKIN) and X2 (OSCOUT) pins are connected to a crystal that resonates at twice the operating frequency of the microprocessor. In the 8 MHz version of the 80186/80188, a 16 MHz crystal is attached to X1 (CLKIN) and X2 (OSCOUT). The 80186/80188 is available in 6 MHz, 8 MHz, 12 MHz, 16 MHz, or 25 MHz versions.

The CLKOUT pin provides a system clock signal that is one-half the crystal frequency, with a 50 percent duty cycle. The CLKOUT pin drives other devices in a system and provides a timing source to additional microprocessors in the system.

In addition to these external pins, the clock generator provides the internal timing for synchronizing the READY input pin, whereas in the 8086/8088 system, READY synchronization is provided by the 8284A clock generator.

Programmable Interrupt Controller. The programmable interrupt controller (PIC) arbitrates the internal and external interrupts, and controls up to two external 8259A PICs. When an external 8259 is attached, the 80186/80188 microprocessors function as the master and the 8259 functions as the slave. The 80C186EC and 80C188EC models contain an 8259A-compatible interrupt controller in place of the one described here for the other versions (XL, EA, and EB).

If the PIC is operated without the external 8259, it has five interrupt inputs: INT0–INT3 and NMI. Note that the number of available interrupts depends on the version: the EB version has six interrupt inputs and the EC version has 16. This is an expansion from the two interrupt inputs available on the 8086/8088 microprocessors. In many systems, the five interrupt inputs are adequate.

Timers. The timer section contains three fully programmable 16-bit timers. Timers 0 and 1 generate waveforms for external use, and are driven by either the master clock of the 80186/ 80188 or by an external clock. They are also used to count external events. The third timer, timer 2, is internal and clocked by the master clock. The output of timer 2 generates an interrupt after a specified number of clocks and can provide a clock to the other timers. Timer 2 can also be used as a watchdog timer because it can be programmed to interrupt the microprocessor after a certain length of time.

The 80C186EC and 80C188EC models have an additional timer called a *watchdog*. The watchdog timer is a 32-bit counter that is clocked internally by the CLKOUT signal (one-half the crystal frequency). Each time the counter hits zero, it reloads and generates a pulse on the WDTOUT pin that is four CLKOUT periods wide. This output can be used for any purpose: it can be wired to the reset input to cause a reset or to the NMI input to cause an interrupt. Note that if it is connected to the reset or NMI inputs, it is periodically reprogrammed so that it never counts down to zero. The purpose of a watchdog timer is to reset or interrupt the system if the software goes awry.

Programmable DMA Unit. The programmable DMA unit contains two DMA channels or four DMA channels in the 80C186EC/80C188EC models. Each channel can transfer data between memory locations, between memory and I/O, or between I/O devices. This DMA controller is similar to the 8237 DMA controller discussed in Chapter 12. The main difference is that the 8237 DMA controller has four DMA channels, as does the EC model.

Programmable Chip Selection Unit. The chip selection is a built-in programmable memory and I/O decoder. It has six output lines to select memory, seven lines to select I/O on the XL and EA models, and 10 lines that select either memory or I/O on the EB and EC models.

On the XL and EA models, the memory selection lines are divided into three groups that select memory for the major sections of the 80186/80188 memory-map. The lower memory select signal enables memory for the interrupt vectors, the upper memory select signal enables memory for reset, and the middle memory select signals enable up to four middle memory devices. The boundary of the lower memory begins at location 00000H and the boundary of the upper memory ends at location FFFFFH. The sizes of the memory areas are programmable, and wait states (0–3 waits) can be automatically inserted with the selection of an area of memory.

On the XL and EA models, each programmable I/O selection signal addresses a 128-byte block of I/O space. The programmable I/O area starts at a base I/O address programmed by the user, and all seven 128-byte blocks are contiguous.

On the EB and EC models, there is an upper and lower memory chip selection pin, and eight general-purpose memory or I/O chip selection pins. Another difference is that from 0–15 wait states can be programmed in these two versions of the 80186/80188 embedded controllers.

Power Save/Power Down Feature. The power save feature allows the system clock to be divided by 4, 8, or 16 to reduce power consumption. The power-saving feature is started by software and exited by a hardware event such as an interrupt. The power down feature stops the clock completely, but it is not available on the XL version. The power down mode is entered by executing a HLT instruction and is exited by any interrupt.

Refresh Control Unit. The refresh control unit generates the refresh row address at the interval programmed. The refresh control unit does not multiplex the address for the DRAM—this is still the responsibility of the system designer. The refresh address is provided to the memory system at the end of the programmed refresh interval, along with the RFSH control signal. The memory system must run a refresh cycle during the active time of the RFSH control signal. More on memory and refreshing is provided in the section that explains the chip selection unit.

Pin-out

Figure 14–2 illustrates the pin-out of the 80C186XL microprocessor. Note that the 80C186XL is packaged in either a 68-pin lead-less chip carrier (LCC) or in a pin grid array (PGA). The LCC package and PGA packages are illustrated in Figure 14–3.

Pin Definitions. The following list defines each 80C186XL pin, and notes any differences between the 80C186XL and 80C188XL microprocessors. The enhanced versions are described later in this chapter.

Vcc This is the system **power** supply connection for ±10%, +5.0 V.

Vss This is the system **ground** connection.

X1 and X2 These pins are generally connected to a fundamental-mode parallel resonant crystal that operates an internal crystal oscillator. An external clock signal may be connected to the X1 pin. The internal master clock operates at one-half the external crystal or clock input signal. Note that these pins are labeled CLKIN (X1) and OSCOUT (X2) on some versions of the 80186/80188.

CLKOUT This pin provides a **timing signal** to system peripherals at one-half the clock frequency with a 50 percent duty cycle.

RES This pin **resets** the 80186/80188. For a proper reset, the RES must be held low for at least 50 ms after power is applied. This pin is often connected to an RC circuit that generates a reset signal after power is applied. The reset location is identical to that of the 8086/8088 microprocessor—FFFF0H.

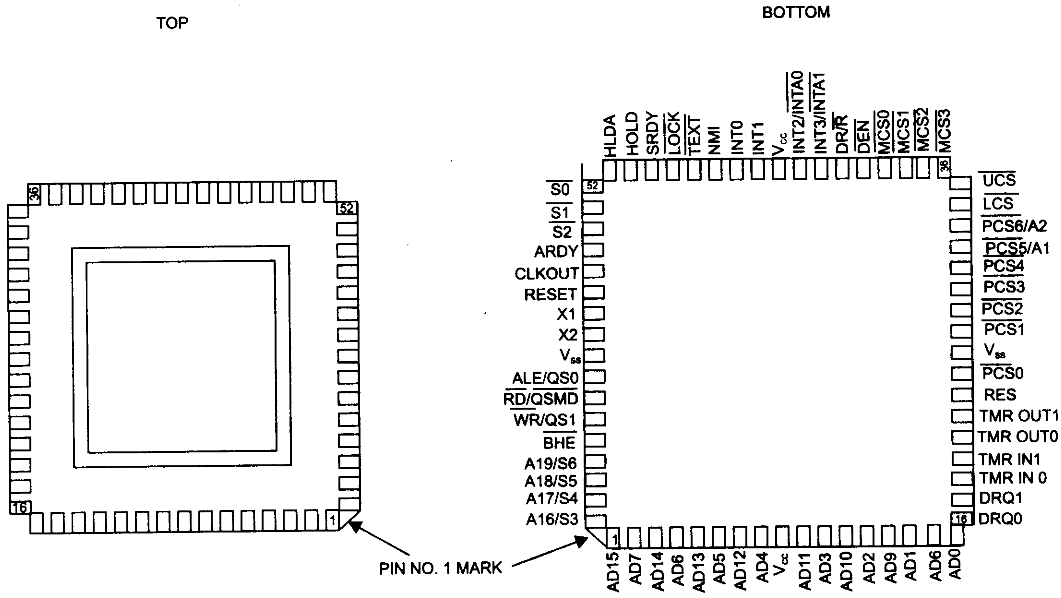


FIGURE 14–2 Pin-out of the 80186 microprocessor. (Courtesy of Intel Corporation.)

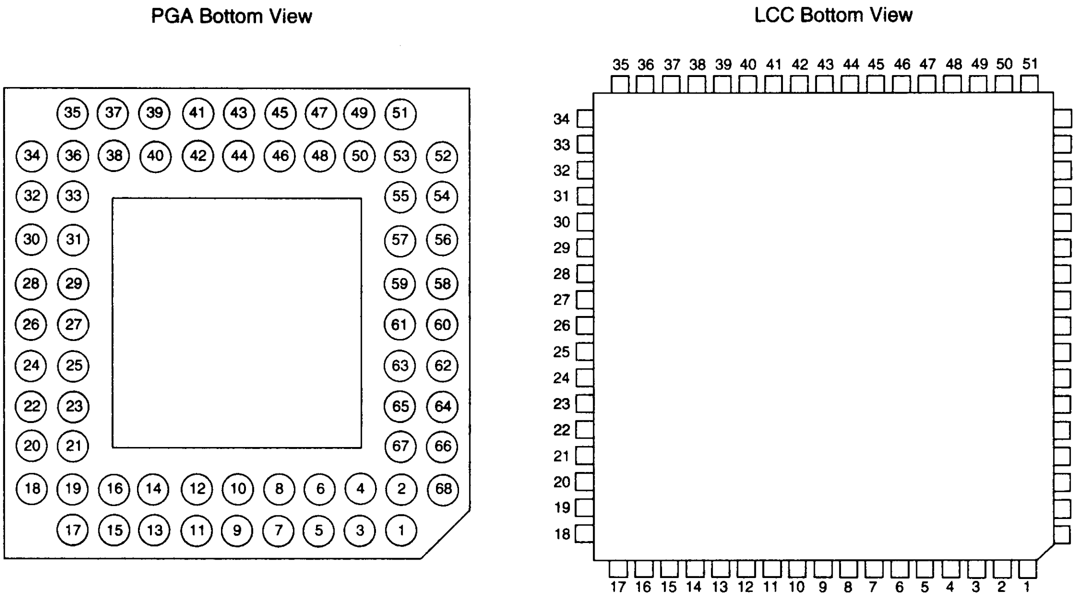


FIGURE 14–3 The bottom views of the PGA and LCC style versions of the 80C188XL microprocessor.

RESET	The companion **reset output** pin (goes high for a reset) connects to system peripherals to initialize them whenever the RES input goes low.
TEST	This **test** pin connects to the BUSY output of the 80187 numeric coprocessor. The TEST pin is interrogated with the WAIT instruction.
TMRIN0, TMRIN1	These pins are used as **external clocking sources** to timers 0 and 1.
TMROUT0 and TMROUT1	These pins provide the **output signals** from timers 0 and 1, which can be programmed to provide square waves or pulses.
DRQ0 and DRQ1	These pins are active-high level triggered **DMA request lines** for DMA channels 0 and 1.
NMI	This is a **non-maskable interrupt** input. It is positive edge-triggered and always active. When NMI is activated, it uses interrupt vector 2.
INT0, INT1, INT2/INTA0, and INT3/INTA1	These are **maskable interrupt** inputs. They are active-high, and are programmed as either level or edge-triggered. These pins are configured as four interrupt inputs if no external 8259 is present, or as two interrupt inputs if 8259s are present.
A19/ONCE, A18, A17, A16	These are multiplexed **address status connections** that provide the address (A19–A16) and status (S6–S3). Status bits found on address pins A18–A16 have no system function and are used during manufacturing for testing. The A19 pin is an input for the ONCE function on a reset. If ONCE is held low on a reset, the microprocessor enters a testing mode.
AD15–AD0	These are multiplexed **address/data bus connections.** During T1, the 80186 places A15–A0 on these pins; during T2, T3, and T4, the 80186 uses these pins as the data bus for signals D15–D0. Note that the 80188 has pins AD7–AD0 and A15–A8.

BHE This pin indicates (when a logic 0) that valid data are transferred through data bus connections D15–D8.

ALE This is a multiplexed output pin that contains ALE one-half clock cycle earlier than in the 8086. It is used to de-multiplex the address/ data and address/status buses. (Even though the status bits on A19– A16 are not used in the system, they must still be de-multiplexed.)

WR This **write** pin causes data to be written to memory or I/O.

RD This **read** pin causes data to be read from memory or I/O.

ARDY The **asynchronous READY** input informs the 80186/80188 that the memory or I/O is ready for the 80186/80188 to read or write data. If this pin is tied to +5.0 V, the microprocessor functions normally; if it is grounded, the microprocessor enters wait states.

SRDY The **synchronous READY** input is synchronized with the system clock to provide a relaxed timing for the ready input. Like ARDY, SRDY is tied to +5.0 V for no wait states.

LOCK This **lock** pin is an output controlled by the LOCK prefix. If an instruction is prefixed with LOCK, the LOCK pin becomes a logic 0 for the duration of the locked instruction.

S2, S1, and S0 These are **status bits** that provide the system with the type of bus transfer in effect. See Table 14-2 for the states of the status bits.

TABLE 14–2 The S2, S1, and S0 status bits.

S2	*S1*	*S0*	*Function*
0	0	0	Interrupt acknowledge
0	0	1	I/O read
0	1	0	I/O write
0	1	1	Halt
1	0	0	Opcode fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive

UCS The **upper-memory chip select** pin selects memory on the upper portion of the memory map. This output is programmable to enable memory sizes of 1K–256K bytes ending at location FFFFFH. Note that this pin is programmed differently on the EB and EC versions and enables memory between 1K and 1M long.

LCS The **lower-memory chip select** pin enables memory beginning at location 00000H. This pin is programmed to select memory sizes from 1K–256K bytes. Note that this pin functions differently for the EB and EC versions and enables memory between 1K and 1M bytes long.

MCS0–MCS3 The **middle-memory chip select** pins enable four middle memory devices. These pins are programmable to select an 8K–512K byte block of memory, containing four devices. Note that these pins are not present on the EB and EC versions.

PCS0–PCS4 These are five different **peripheral selection lines.** Note that the lines are not present on the EB and EC versions.

PCS5/A1 and PCS6/A2 These are programmed as **peripheral selection lines** or as internally latched address bits A2 and A1. These lines are not present on the EB and EC versions.

DT/R This pin controls the direction of data bus buffers if attached to the system.

DEN This pin **enables** the external data bus buffers.

80186 Master Interface Timing Responses

		80188 (8 Mhz)		80188-6 (6 Mhz)			
Symbol	**Parameters**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**	**Test Conditions**
T_{CLAV}	Address Valid Delay	5	44	5	63	ns	C_L = 20-200$_p$F all outputs
T_{CLAX}	Address Hold	10		10		ns	
T_{CLAZ}	Address Float Delay	T_{CLAX}	35	T_{CLAX}	44	ns	
T_{CHCZ}	Command Lines Float Delay		45		56	ns	
T_{CHCV}	Command Lines Valid Delay (after float)		55		76	ns	
T_{LHLL}	ALE Width	$T_{CLCL-35}$		$T_{CLCL-35}$		ns	
T_{CHLH}	ALE Active Delay		35		44	ns	
T_{CHLL}	ALE Inactive Delay		35		44	ns	
T_{LLAX}	Address Hold to ALE Inactive	$T_{CHCL-25}$		$T_{CHCL-30}$		ns	
T_{CLDV}	Data Valid Delay	10	44	10	55	ns	
T_{CLDOX}	Data Hold Time	10		10		ns	
T_{WHDX}	Data Hold after WR	$T_{CLCL-40}$		$T_{CLCL-50}$		ns	
T_{CVCTV}	Control Active Delay 1	5	70	5	87	ns	
T_{CHCTV}	Control Active Delay 2	10	55	10	76	ns	
T_{CVCTX}	Control Inactive Delay	5	55	5	76	ns	
T_{CVDEX}	$\overline{DEN}$ Inactive Delay (Non-Write Cycle)		70		87	ns	
T_{AZRL}	Address Float to RD Active	0		0		ns	
T_{CLRL}	$\overline{RD}$ Active Delay	10	70	10	87	ns	
T_{CLRH}	$\overline{RD}$ Inactive Delay	10	55	10	76	ns	
T_{RHAV}	$\overline{RD}$ Inactive to Address Active	$T_{CLCL-40}$		$T_{CLCL-50}$		ns	
T_{CLHAV}	HLDA Valid Delay	10	50	10	67	ns	
T_{RLRH}	$\overline{RD}$ Width	$2T_{CLCL-50}$		$2T_{CLCL-50}$		ns	
T_{WLWH}	$\overline{WR}$ Width	$2T_{CLCL-40}$		$2T_{CLCL-40}$		ns	
T_{AVAL}	Address Valid to ALE Low	$T_{CLCH-25}$		$T_{CLCH-45}$		ns	
T_{CHSV}	Status Active Delay	10	55	10	76	ns	
T_{CLSH}	Status Inactive Delay	10	55	10	76	ns	
T_{CLTMV}	Timer Output Delay		60		75	ns	100 pF max
T_{CLRO}	Reset Delay		60		75	ns	
T_{CHQSV}	Queue Status Delay		35		44	ns	

80186 Chip-Select Timing Responses

Symbol	Parameters	Min.	Max.	Min.	Max.	Units	Test Conditions
T_{CLCSV}	Chip-Select Active Delay		66		80	ns	
T_{CXCSX}	Chip-Select Hold from Command Inactive	35		35		ns	
T_{CHCSX}	Chip-Select Inactive Delay	5	35	5	47	ns	

Symbol	Parameters	Min.	Max.	Units	Test Conditions
TDVCL	Data in Setup (A/D)	20	20	ns	ns
TCLDX	Data in Hold (A/D)	10		ns	
TARYHCH	Asynchronous Ready (AREADY) active setup time*	20		ns	
TARYLCL	AREADY inactive setup time	35		ns	
TCHARYX	AREADY hold time	15		ns	
TSRYCL	Synchronous Ready (SREADY) transition setup time	35		ns	
TCLSRY	SREADY transition hold time	15		ns	
THVCL	HOLD Setup*	25		ns	
TINVCH	INTR, NMI, $\overline{TEST}$, TIMERIN, Setup*	25		ns	
TINVCL	DRQ0, DRQ1, Setup*	25		ns	

*To guarantee recognition at next clock

FIGURE 14–5 80186 AC characteristics. (Courtesy of Intel Corporation.)

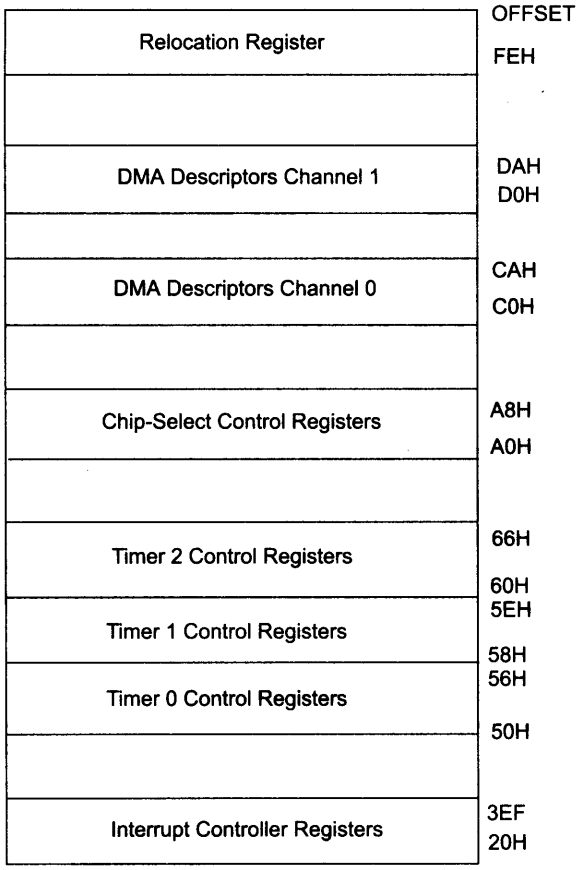
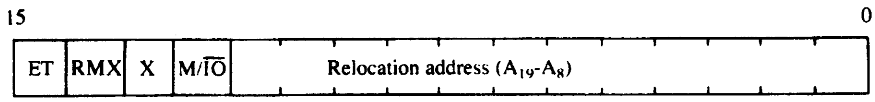


FIGURE 14–6 Peripheral control block (PCB) of the 80186/80188. (Courtesy of Intel Corporation.)



ET = ESC/NO ESC TRAP
RMX = iRM × 86 mode/master mode
M/$\overline{IO}$ = Memory/IO space
X = Unused

FIGURE 14–7 Peripheral control register.

FFFEH with a new bit pattern. For example, to relocate the PCB to memory locations 20000H–200FFH, a 1200H is sent to I/O address FFFEH. Notice that M/IO is a logic 1 to select memory, and that a 200H selects memory address 20000H as the base address of the PCB. Note that all accesses to the PCB must be word accesses because it is organized as 16-bit wide registers. Example 14–1 shows the software required to relocate the PCB to memory location 20000H–200FFH. Note that either an 8- or 16-bit output can be used to program the 80186; in the 80188, never use the OUT DX,AX instruction because it takes additional clocking periods to execute.

EXAMPLE 14–1

```
0100  BA FFFE        MOV   DX,0FFFEH      ;address relocation register
0103  B8 1200        MOV   AX,1200H       ;code for new PCB location
0106  EE             OUT   DX,AL          ;this can also be OUT DX,AX
```

The EB and EC versions use a different address for programming the PCB location. Both versions have the PCB relocation register stored at offset XXA8H, instead of at offset XXFEH for the XL and EA versions. The bit patterns of these versions is the same as for the XL and EA versions, except that the RMX bit is missing.

Interrupts in the 80186/80188

The interrupts in the 80186/80188 are identical to the 8086/8088, except that there are additional interrupt vectors defined for some of the internal devices. A complete listing of the reserved interrupt vectors appears in Table 14–3. The first five are identical to the 8086/8088.

The array BOUND instruction interrupt is requested if the boundary of an index register is outside the values set up in the memory. The unused opcode interrupt occurs whenever the 80186/80188 executes any undefined opcode. This is important if a program begins to run awry. Note that the unused opcode interrupt can be accessed

TABLE 14–3 80186/80188 interrupt vectors.

Name	*Type*	*Address*	*Priority*
Divide error	0	00000H–00003H	1
Single-step	1	00004H–00007H	1A
NMI pin	2	00008H–0000BH	1
Breakpoint	3	0000CH–0000FH	1
Overflow	4	00010H–00013H	1
BOUND instruction	5	00014H–00017H	1
Unused opcode	6	00018H–0001BH	1
ESCape opocde	7	0001CH–0001FH	1
Timer 0	8	00020H–00023H	2A
Reserved	9	00024H–00027H	
DMA 0	10	00028H–0002BH	4
DMA 1	11	0002CH–0002FH	5
INT0	12	00030H–00033H	6
INT1	13	00034H–00037H	7
INT2	14	00038H–0003BH	8
INT3	15	0003CH–0003FH	9
80187	16	00040H–00043H	1
Reserved	17	00044H–00047H	
Timer 1	18	00048H–0004BH	2B
Timer 2	19	0004CH–0004FH	2C
Serial 0 receiver (EB only)	20	00050H–00053H	3A
Serial 0 transmitter (EB only)	21	00054H–00057H	3B

Note: Priority level 1 has the highest priority and level 9 the lowest. Some interrupts have the same priority. Only the EB and EC versions contain the serial unit.

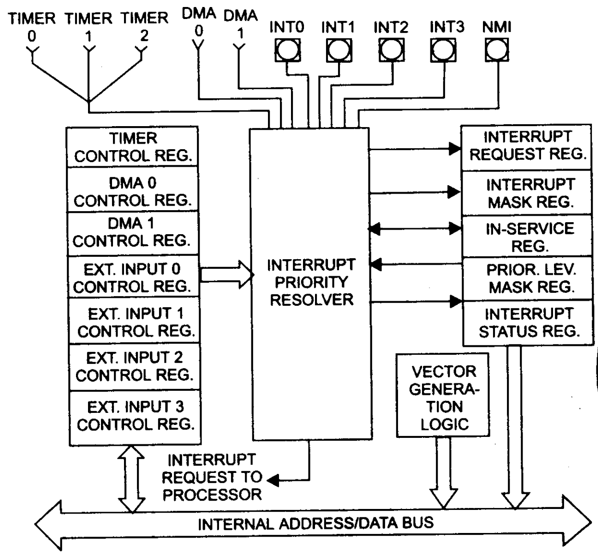


FIGURE 14–8 80186/80188 programmable interrupt controller. (Courtesy of Intel Corporation.)

by an instruction, but the assembler does not include it in the instruction set. On the Pentium Pro/Pentium 4 and some earlier Intel microprocessors, the 0F0BH or 0FB9H instruction will cause the program to call the procedure whose address is stored at the unused opcode interrupt vector.

The ESC opcode interrupt occurs if ESC opcodes D8H–DFH are executed. This occurs only if the ET (escape trap) bit of the relocation register is set. If an ESC interrupt occurs, the address stored on the stack by the interrupt points to the ESC instruction or to its segment override prefix, if one is used.

The internal hardware interrupts must be enabled by the I flag bit and must be unmasked to function. The I flag bit is set (enabled) with STI and cleared (disabled) with CLI. The remaining internally decoded interrupts are discussed with the timers and DMA controller, later in this section.

Interrupt Controller

The interrupt controller inside the 80186/80188 is a sophisticated device. It has many interrupt inputs that arrive from the five external interrupt inputs, the DMA controller, and the three timers. Figure 14–8 provides a block diagram of the interrupt structure of the 80186/80188 interrupt controller. This controller appears in the XL, EA, and EB versions, but the EC version contains the exact equivalent to a pair of 8259As, as found in Chapter 11. In the EB version, the DMA inputs are replaced with inputs from the serial unit for receive and transmit.

The interrupt controller operates in two modes: master and slave mode. The mode is selected by a bit in the interrupt control register (EB and EC versions) called the *CAS bit.* If the CAS bit is a logic 1, the interrupt controller connects to external 8259A programmable interrupt controllers (see Figure 14–9); if CAS is a logic 0, the internal interrupt controller is selected. In many cases, there are enough interrupts within the 80186/80188, so the slave mode is not normally used. Note that in the XL and EA versions, the master and slave modes are selected in the peripheral control register at offset address FEH.

This portion of the text does not detail the programming of the interrupt controller. Instead, it is limited to a discussion of the internal structure of the interrupt controller. The programming and application of the interrupt controller is discussed in the sections that describe the timer and DMA controller.

Interrupt Controller Registers. Figure 14–10 illustrates the interrupt controller's registers. These registers are located in the peripheral control block beginning at offset address 22H. For the EC version, which is compatible with the 8259A, the interrupt controller ports are at offset addresses 00H and 02H for the master and ports 04H and 06H

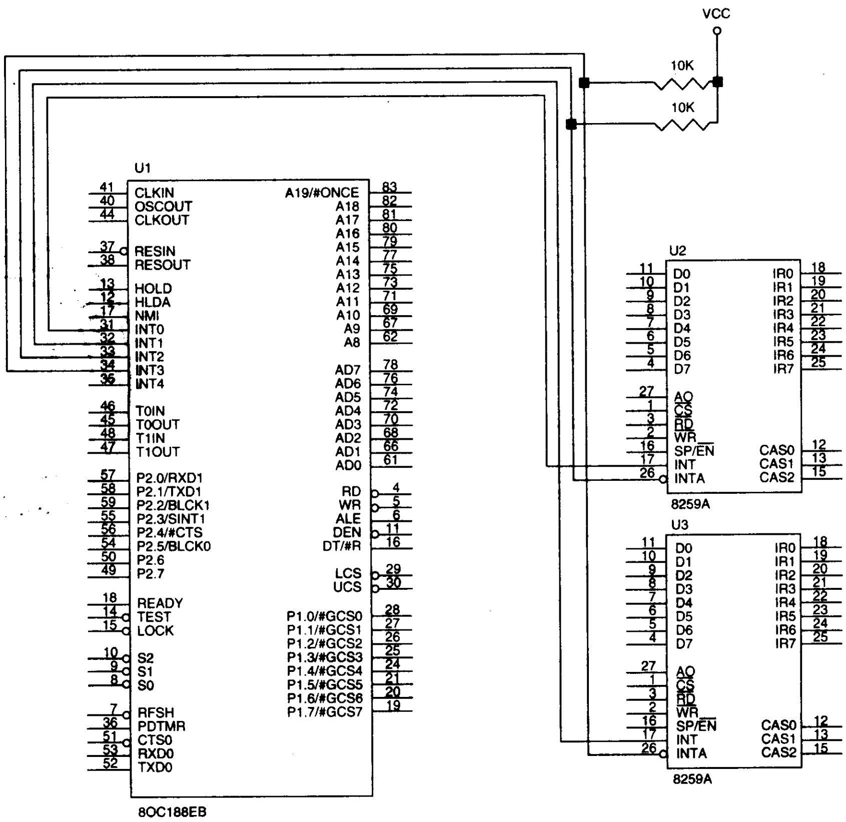


FIGURE 14–9 The interconnection between the 80C188EB and two 8259A programmable interrupt controllers. *Note:* Only the connections vital for this interface are shown.

for the slave. In the EB version, the interrupt controller is programmed at offset address 02H. Note that the EB version has an additional interrupt input (INT4).

Slave Mode. When the interrupt controller operates in the slave mode, it uses up to two external 8259A programmable interrupt controllers for interrupt input expansion. Figure 14–9 shows how the external interrupt controllers connect to the 80186/80188 interrupt input pins for slave operation. Here, the INT0 and INT1 inputs are used as external connections to the interrupt request outputs of the 8259s, and INTA0 (INT2) and INTA1 (INT3) are used as interrupt acknowledge signals to the external controllers.

Interrupt Control Registers. There are interrupt control registers in both modes of operation, which each control a single interrupt source. Figure 14–11 depicts the binary bit pattern of each of these interrupt control registers. The mask bit enables (0) or disables (1) the interrupt input represented by the control word, and the priority bits set the priority level of the interrupt source. The highest priority level is 000, and the lowest is 111. The CAS bit is used to enable slave or cascade mode (0 enables slave mode), and the SFNM bit selects the special fully nested mode. The SFNM allows the priority structure of the 8259A to be maintained.

Interrupt Request Register. The interrupt request register contains an image of the interrupt sources in each mode of operation. Whenever an interrupt is requested, the corresponding interrupt request bit becomes a logic 1, even if the interrupt is masked. The request is cleared whenever the 80186/80188 acknowledges the interrupt. Figure 14–12 illustrates the binary bit pattern of the interrupt request register for both the master and slave modes.

XL and EA Versions		EB Version	
3EH	INT3 Control Register	1EH	INT3 Control Register
3CH	INT2 Control Register	1CH	INT2 Control Register
3AH	INT1 Control Register	1AH	INT1 Control Register
38H	INT0 Control Register	18H	INT0 Control Register
36H	DMA1 Control Register	16H	INT4 Control Register
34H	DMA0 Control Register	14H	Serial Control Register
32H	Timer Control Register	12H	Timer Control Register
30H	Interrupt Status	10H	Interrupt Status
2EH	Request	0EH	Request
2CH	In Service	0CH	In Service
2AH	PRIMSK	0AH	PRIMSK
28H	Interrupt Masks	08H	Interrupt Masks
26H	POLL Status	06H	POLL Status
24H	POLL	04H	POLL
22H	EOI	02H	EOI

FIGURE 14–10 The I/O offset port assignment for the interrupt control unit.

Mask and Priority Mask Registers. The interrupt mask register has the same format as the interrupt register illustrated in Figure 14–12. If a source is masked (disabled), the corresponding bit of the interrupt mask register contains a logic 1; if enabled, it contains a logic 0. The interrupt mask register is read to determine which interrupt sources are masked and which are enabled. A source is masked by setting the source's mask bit in its interrupt control register.

The priority mask register, illustrated in Figure 14–13, shows the priority of the interrupt currently being serviced by the 80186/80188. The level of the interrupt is indicated by priority bits P2–P0. Internally, these bits prevent an interrupt by a lower priority source. These bits are automatically set to the next lower level at the end of an interrupt, as issued by the 80186/80188. If no other interrupts are pending, these bits are set (111) to enable all priority levels.

In-Service Register. The in-service register has the same binary bit pattern as the request register of Figure 14–12. The bit that corresponds to the interrupt source is set if the 80186/80188 is currently acknowledging the interrupt. The bit is reset at the end of an interrupt.

The Poll and Poll Status Registers. Both the interrupt poll and interrupt poll status registers share the same binary bit patterns as those illustrated in Figure 14–14. These registers have a bit (INT REQ) that indicates an interrupt is pending. This bit is set if an interrupt is received with sufficient priority, and cleared when an interrupt is acknowledged. The S bits indicate the interrupt vector type number of the highest priority pending interrupt.

The poll and poll status registers may appear to be identical because they contain the same information. However, they differ in function. When the interrupt poll register is read, the interrupt is acknowledged. When the

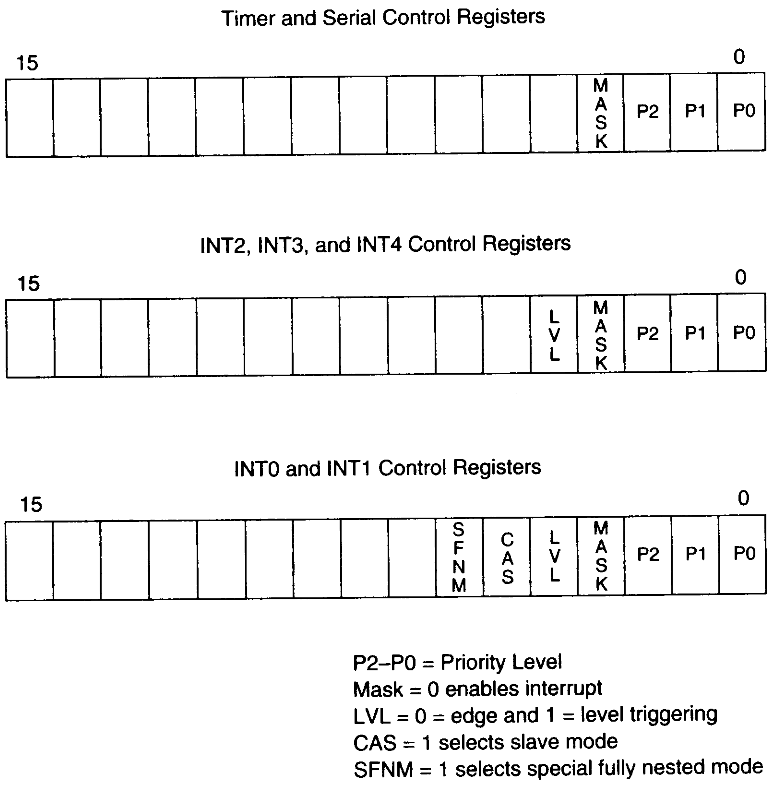


FIGURE 14–11 The interrupt control registers.

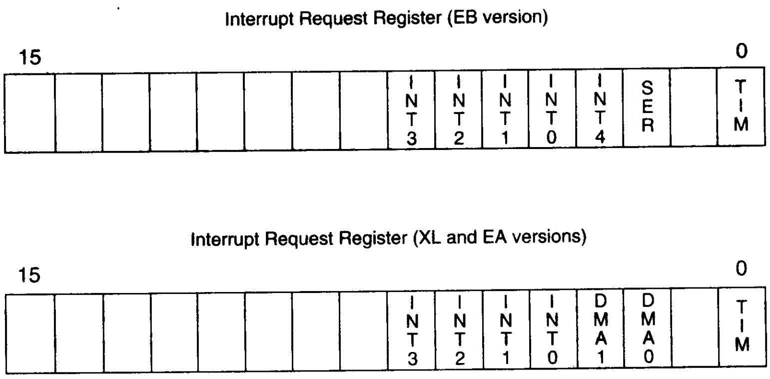


FIGURE 14–12 The interrupt request register.

interrupt poll status register is read, no acknowledge is sent. These registers are used only in the master mode, not in the slave mode.

End-of-interrupt Register. The end-of-interrupt (EOI) register causes the termination of an interrupt when written by a program. Figure 14–15 shows the contents of the EOI register for both the master and slave mode.

Priority Mask Register

15															0
													P2	P1	P0

P2–P0 = Priority Level

FIGURE 14–13 The priority mask register.

Poll and Poll Status Registers

15															0
IREQ											VT4	VT3	VT2	VT1	VT0

IREQ = 1 = Interrupt pending
VT4–VT0 = Interrupt type number of highest priority pending interrupt

FIGURE 14–14 The poll and poll status registers.

End-of-Interrupt Register

15															0
NSPEC											VT4	VT3	VT2	VT1	VT0

FIGURE 14–15 The end-of-interrupt (EOI) register.

In the master mode, writing to the EOI register ends either a specific interrupt level or whatever level is currently active (nonspecific). In the nonspecific mode, the NSPEC bit must be set before the EOI register is written to end a nonspecific interrupt. The nonspecific EOI clears the highest level interrupt bit in the in-service register. The specific EOI clears the selected bit in the in-service register. The nonspecific mode is used unless there is a special circumstance that requires a different order for interrupt acknowledges.

In the slave mode, the level of the interrupt to be terminated is written to the EOI register. The slave mode does not allow a nonspecific EOI.

Interrupt Status Register. The format of interrupt status register is depicted in Figure 14–16. In the master mode, T2–T0 indicates which timer (timer 0, timer 1, or timer 2) is causing an interrupt. This is necessary because all three timers have the same interrupt priority level. These bits are set when the timer requests an interrupt and are cleared when the interrupt is acknowledged. The DHLT (DMA halt) bit is only used in the master mode; when set, it stops a DMA action. Note that the interrupt status register is different for the EB version.

Interrupt Vector Register. The interrupt vector register is present only in the slave mode, and only in the XL and EA versions at offset address 20H. It is used to specify the most significant five bits of the interrupt vector type number. Figure 14–17 illustrates the format of this register.

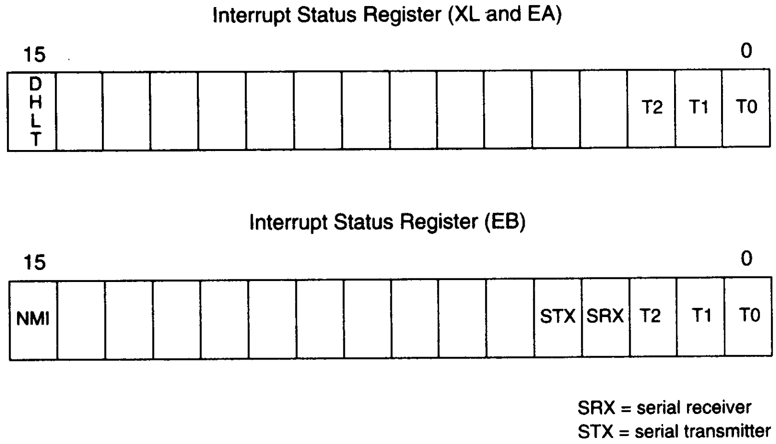


FIGURE 14–16 The interrupt status register.

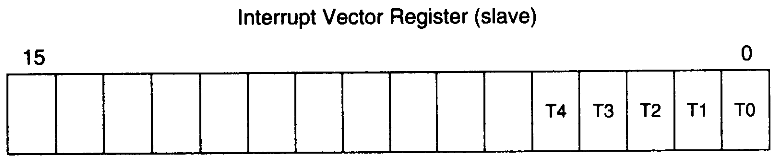


FIGURE 14–17 The interrupt vector register.

Timers

The 80186/80188 microprocessors contain three fully programmable 16-bit timers and each is totally independent of the others. Two of the timers (timer 0 and timer 1) have input and output pins that allow them to count external events or generate wave-forms. The third timer (timer 2) connects to the 80186/80188 clock. Timer 2 is used as a DMA request source, as a prescaler for other timers, or as a watchdog timer.

Figure 14–18 shows the internal structure of the timer unit. Notice that the timer unit contains one counting element that is responsible for updating all three counters. Each timer is actually a register that is rewritten from the counting element (a circuit that reads a value from a timer register and increments it before returning it). The counter element is also responsible for generating the outputs on the pins T0OUT and T1OUT, reading the T0IN and T1IN pins, and causing a DMA request from the terminal count (TC) of timer 2 if timer 2 is programmed to request a DMA action.

Timer Register Operation. The timers are controlled by a block of registers in the peripheral control block (see Figure 14–19). Each timer has a count register, maximum-count register or registers, and a control register. These registers may all be read or written at any time because the 80186/80188 microprocessors ensure that the contents never change during a read or write.

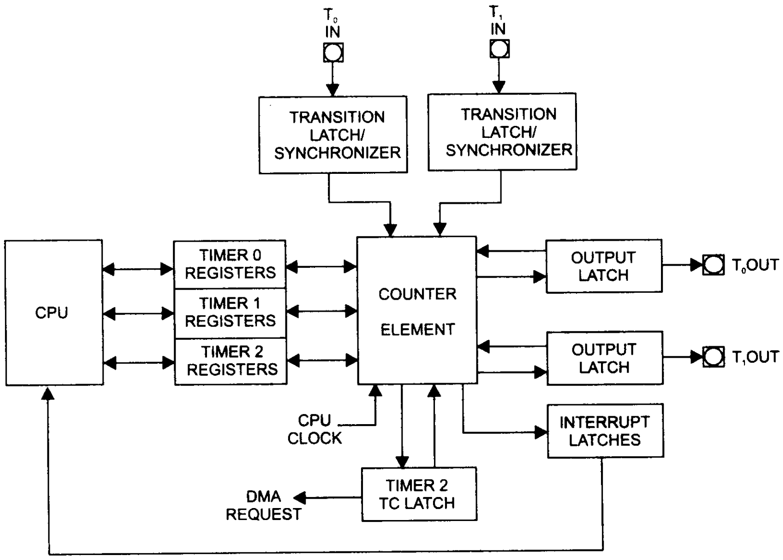


FIGURE 14–18 Internal structure of the 80186/80188 timers. (Courtesy of Intel Corporation.)

The timer count register contains a 16-bit number that is incremented whenever an input to the timer occurs. Timers 0 and 1 are incremented at the positive edge on an external input pin, every fourth 80186/80188 clock, or by the output of timer 2. Timer 2 is clocked on every fourth 80186/80188 clock pulse and has no other timing source. This means that in the 8 MHz version of the 80186/80188, timer 2 operates at 2 MHz, and the maximum counting frequency of timers 0 and 1 is 2 MHz. Figure 14–20 depicts these four clocking periods, which are not related to the bus timing.

Each timer has at least one maximum-count register, called a **compare** register (compare register A for timers 0 and 1) that is loaded with the maximum count of the count register to generate an output. Note that a timer is an up counter. Whenever the count register is equal to the maximum-count compare register, it is cleared to 0. With a maximum count of 0000H, the counter counts 65,536 times. For any other value, the timer counts the true value of the count. For example, if the maximum count is 0002H, then the counter will count from 0 to 1 and then be cleared to 0—a modulus 2 counter has 2 states.

Timers 0 and 1 each have a second maximum-count compare register (compare register B) that is selected by the control register for the timer. Either maximum-count compare register A or both maximum-count compare registers A and B are used with these timers, as programmed by the ALT bit in the control register for the timer. When both maximum-count compare registers are used, the timer counts up to the value in maximum-count compare register A, clears to 0, and then counts up to the count in maximum-count compare register B. This process is then repeated. Using both maximum-count registers allows the timer to count up to 131,072.

The control register (refer to Figure 14–19) of each timer is 16 bits wide and specifies the operation of the timer. A definition of each control bit follows:

EN The **enable** bit allows the timer to start counting. If EN is cleared, the timer does not count; if it is set, the timer counts.

INH The **inhibit** bit allows a write to the timer control register to affect the enable bit (EN). If INH is set, then the EN bit can be set or cleared to control the counting.
If INH is cleared, EN is not affected by a write to the timer control register. This allows other features of the timer to be modified without enabling or disabling the timer.

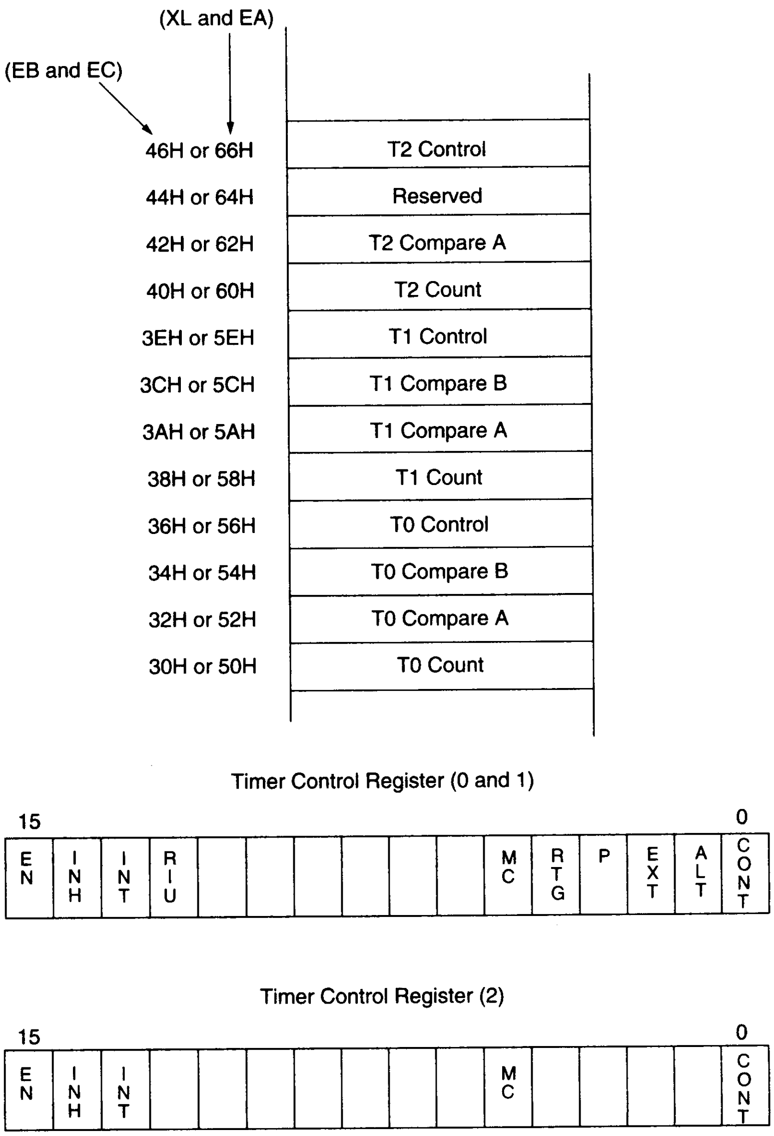


FIGURE 14–19 The offset locations and contents of the registers used to control the timers.

INT The **interrupt** bit allows an interrupt to be generated by the timer. If INT is set, an interrupt will occur each time that the maximum count is reached in either maximum-count compare register. If this bit is cleared, no interrupt is generated. When the interrupt request is generated, it remains in force, even if the EN bit is cleared after the interrupt request.

RIU The **register in use** bit indicates which maximum-count compare register is currently in use by the timer. If RIU is a logic 0, then maximum-count compare register A is in use. This bit is a read-only bit, and writes do not affect it.

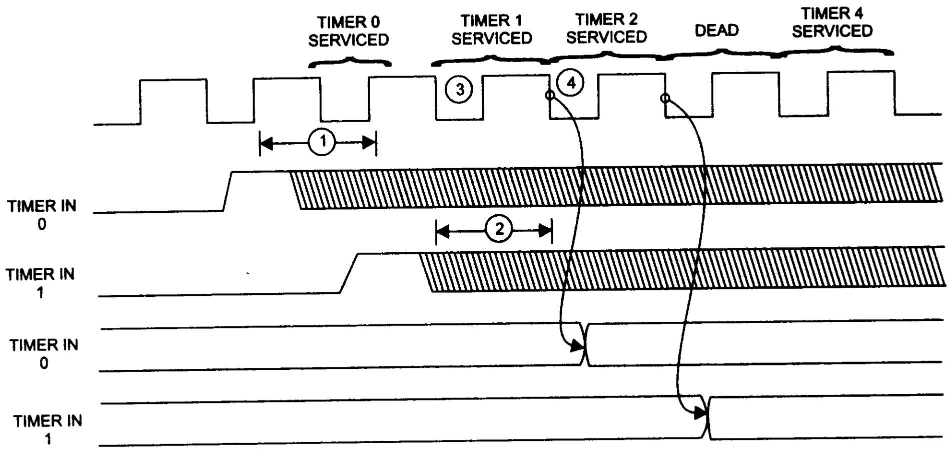


FIGURE 14–20 Timing for the 80186/80188 timers. (Courtesy of Intel Corporation.)

MC The **maximum count** bit indicates that the timer has reached its maximum count. This bit becomes a logic 1 when the timer reaches its maximum count and remains a logic 1 until the MC bit is cleared by writing a logic 0. This allows the maximum count to be detected by software.

RTG The **re-trigger** bit is active only for external clocking (EXT = 0). The RTG bit is used only with timers 0 and 1 to select the operation of the timer input pins (T0IN and T1IN). If RTG is a logic 0, the external input will cause the timer to count if it is a logic 1; the timer will hold its count (stop counting) if it is a logic 0. If RTG is a logic 1, the external input pin clears the timer count to 0000H each time a positive-edge occurs.

P The **prescaler** bit selects the clocking source for timers 0 and 1. If $EXT = 0$ and $P = 0$, the source is one-fourth the system clock frequency. If $P = 1$, the source is timer 2.

EXT The **external** bit selects internal timing ($EXT = 0$) or external timing ($EXT = 1$). If $EXT = 1$, the timing source is applied to the T0IN or T1IN pins. In this mode, the timer increments after each positive-edge on the timer input pin. If $EXT = 0$, the clocking source is from one of the internal sources.

ALT The **alternate** bit selects single maximum-count mode (maximum-count compare register A) if a logic 0, or alternate maximum-count mode (maximum-count compare registers A and B) if a logic 1.

CONT The **continuous** bit selects continuous operation if a logic 1. In continuous operation, the counter automatically continues counting after it reaches its maximum count. If CONT is a logic 0, the timer will automatically stop counting and clear the EN bit. Note that whenever the 80186/80188 are reset, the timers are automatically disabled.

Timer Output Pin. Timers 0 and 1 have an output pin used to generate either square waves or pulses. To produce pulses, the timer is operated in single maximum-count mode ($ALT = 0$). In this mode, the output pin goes low for one clock period when the counter reaches its maximum count. By controlling the CONT bit in the control register, either a single pulse or continuous pulses can be generated.

To produce square waves or varying duty cycles, the alternate mode (*ALT* = 1) is selected. In this mode, the output pin is a logic 1 while maximum-count compare register A controls the timer; it is a logic 0 while maximum-count compare register B controls the timer. As with the single maximum-count mode, the timer can generate either a single square wave or continuous square waves. See Table 14–4 for the function of the ALT and CONT control bits.

TABLE 14–4 Function of ALT and CONT in the timer control register.

ALT	*CONT*	*Mode*
0	0	Single pulse
0	1	Continuous pulses
1	0	Single square wave
1	1	Continuous square waves

Almost any duty cycle can be generated in the alternate mode. For example, suppose that a 10 percent duty cycle is required at a timer output pin. Maximum-count register A is loaded with a 10 and maximum-count register B is loaded with a 90 to produce an output that is a logic 1 for 10 clocks and a logic 0 for 90 clocks. This also divides the frequency of the timing source by a factor of 100.

Real-Time Clock Example. Many systems require the time of day. This is often called a **real-time clock.** A timer within the 80186/80188 can provide the timing source for software that maintains the time of day.

The hardware required for this application is not illustrated. All that is required is that the T1IN pin be connected to +5.0 V through a pull-up resistor to enable timer 1. In the example, timers 1 and 2 are used to generate a 1-second interrupt that provides the software with a timing source.

The software required to implement a real-time clock is listed in Examples 14–2 and 14–3. Example 14–2 illustrates the software required to initialize the timers. Example 14–3 shows an interrupt service procedure, which keeps time. There is a another procedure in Example 14–3 that increments a BCD modulus counter. None of the software required install the interrupt vector, and time of day is illustrated here.

EXAMPLE 14–2

```
;software for a real-time clock using the 80C188EA microprocessor

= FF62          T2_CA  EQU 0FF62H        ;address of timer 2 compare A
= FF66          T2_CON EQU 0FF66H        ;address of timer 2 control
= FF60          T2_CNT EQU 0FF60H        ;address of timer 2 count
= FF5A          T1_CA  EQU 0FF5AH        ;address of timer 1 compare A
= FF58          T1_CON EQU 0FF58H        ;address of timer 1 control
= FF5E          T1_CNT EQU 0FF5EH        ;address of timer 1 count

0010            CLOCK_UP   PROC   FAR

0010  B8 4E20          MOV    AX,20000      ;count for timer 2
0013  BA FF62          MOV    DX,T2_CA      ;address timer 2 compare A
0016  EE               OUT    DX,AL         ;program for 10 ms

0017  B8 0064          MOV    AX,100        ;count for timer 1
001A  BA FF5A          MOV    DX,T1_CA      ;address timer 1 compare A
001D  EE               OUT    DX,AL         ;program for 1 second

001E  B8 0000          MOV    AX,0
0021  BA FF60          MOV    DX,T2_CNT     ;address timer 2 count
0024  EE               OUT    DX,AL         ;clear count

0025  BA FF5E          MOV    DX,T1_CNT     ;address timer 1 count
0028  EE               OUT    DX,AL         ;clear count
0029  B8 C001          MOV    AX,0C001H     ;enable timer 2
002C  BA FF66          MOV    DX,T2_CON     ;address timer 2 control

002F  EE               OUT    DX,AL
0030  B8 E009          MOV    AX,0E009H     ;enable timer 1
```

```
0033  BA FF58              MOV   DX,T1_CON     ;address timer 1 control
0036  EE                   OUT   DX,AL

0037  CB                   RET

0038            CLOCK_UP   ENDP

                           END
```

Timer 2 is programmed to divide by a factor of 20,000. This causes the clock (2 MHz on the 8 MHz version of the 80186/80188) to be divided down to one pulse every 10 ms. The clock for timer 1 is derived internally from the timer 2 output. Timer 1 is programmed to divide by 100 and generates a pulse once per second. The control register of timer 1 is programmed so that the one-second pulse internally generates an interrupt.

The interrupt service procedure is called once per second to keep time. The interrupt service procedure adds a one to the content of memory location SECONDS. Once every 60 seconds, the content of the next memory location (SECONDS + 1) is incremented. Finally, once per hour, the content of memory location SECONDS + 2 is incremented. The time is stored in these three consecutive memory locations in BCD, so the system software can easily access the time.

EXAMPLE 14–3

```
0000  00             SECONDS   DB  ?                          ;time
0001  00             MINUTES   DB  ?
0002  00             HOURS     DB  ?

0100                 INTRS     PROC  FAR USES AX SI

0102  BE 0000 R                MOV   SI,OFFSET SECONDS ;address time
0105  B4 60                    MOV   AH,60H
0107  E8 000F                  CALL  UP_COUNT          ;increment seconds
010A  75 0A                    JNZ   ENDI
010C  E8 000A                  CALL  UP_COUNT          ;increment minutes
010F  75 05                    JNZ   ENDI
0111  B4 24                    MOV   AH,24H
0113  E8 0003                  CALL  UP_COUNT          ;increment hours
0116                 ENDI:
                               IRET

0119                 INTRS     ENDP
0119                 UP_COUNT  PROC  NEAR

0119  2E: 8A 04                MOV   AL,CS:[SI]        ;get count
011C  46                       INC   SI
011D  04 01                    ADD   AL,1              ;increment count
011F  27                       DAA                     ;make it BCD
0120  2E: 88 44 FF             MOV   CS:[SI-1],AL      ;save new count
0124  2A C4                    SUB   AL,AH             ;test modulus
0126  75 04                    JNE   ENDU              ;if no roll-over needed
0128  2E: 88 44 FF             MOV   CS:[SI-1],AL      ;clear count
012C                 ENDU:
012C  C3                       RET

012D                 UP_COUNT  ENDP

                               END
```

DMA Controller

The DMA controller within the 80186/80188 has two fully independent DMA channels. Each has its own set of 20-bit address registers, so any memory or I/O location is accessible for a DMA transfer. In addition, each channel is programmable for auto-increment or auto-decrement to either source or destination registers. This controller is

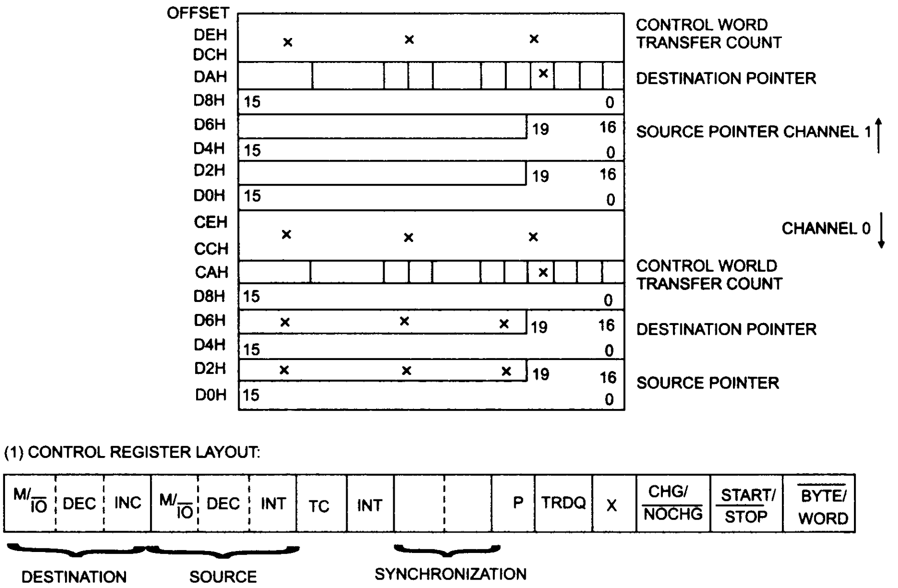


FIGURE 14–21 Register structure of the 80186/80188 DMA controller. (Courtesy of Intel Corporation.)

not available in the EB or EC versions. The EC version contains a modified four-channel DMA controller, while the EB version contains no DMA controller. This text does not describe the DMA controller within the EC version.

Figure 14–21 illustrates the internal register structure of the DMA controller. These registers are located in the peripheral control block at offset addresses C0H–DFH.

Notice that both DMA channel register sets are identical; each channel contains a control word, a source and destination pointer, and a transfer count. The transfer count is 16 bits wide and allows unattended DMA transfers of bytes (80188/80186) and words (80186 only). Each time that a byte or word is transferred, the count is decremented by one until it reaches 0000H—the terminal count.

The source and destination pointers are each 20 bits wide, so DMA transfers can occur to any memory location or I/O address without concern for segment and offset addresses. If the source or destination address is an I/O port, bits A19–A16 must be 0000 or a malfunction may occur.

Channel Control Register. Each DMA channel contains its own channel control register (refer to Figure 14–21), which defines its operation. The leftmost six bits specify the operation of the source and destination registers. The M/IO bit indicates a memory or I/O location, DEC causes the pointer to be decremented, and INC causes the pointer to be incremented. If both the INC and DEC bits are 1, then the pointer is unchanged after each DMA transfer. Notice that memory-to-memory transfers are possible with this DMA controller.

The TC (terminal count) bit causes the DMA channel to stop transfers when the channel count register is decremented to 0000H. If this bit is a logic 1, the DMA controller continues to transfer data, even after the terminal count is reached.

The INT bit enables interrupts to the interrupt controller. If set, this bit causes an interrupt to be issued when the terminal count of the channel is reached.

The SYN bit selects the type of synchronization for the channel: 00 = no synchronization, 01 = source synchronization, and 10 = destination synchronization. When either unsynchronized or source synchronization is selected, data are transferred at the rate of 2M bytes per second. These two types of synchronization allow transfers to occur without interruption. If destination synchronization is selected, the transfer rate is slower (1.3M bytes per second), and the controller relinquishes control to the 80186/80188 after each DMA transfer.

The P bit selects the channel priority. If $P = 1$, the channel has the highest priority. If both channels have the same priority, the controller alternates transfers between channels.

The TRDQ bit enables DMA transfers from timer 2. If this bit is a logic 1, the DMA request originates from timer 2. This can prevent the DMA transfers from using all of the microprocessor's time for the transfer.

The CHG/NOCHG bit determines whether START/STOP changes for a write to the control register. The START/STOP bit starts or stops the DMA transfer. To start a DMA transfer, both CHG/NOCHG and START/STOP are placed at a logic 1 level.

The BYTE/WORD selects whether the transfer is byte- or word-sized.

Sample Memory-to-Memory Transfer. The built-in DMA controller is capable of performing memory-to-memory transfers. The procedure used to program the controller and start the transfer is listed in Example 14–4.

EXAMPLE 14–4

```
                    .MODEL SMALL
                    .186
0000                .CODE

                    ;Memory-to-memory DMA transfer procedure
                    ;
                    ;Calling parameters:
                    ;
                    ;   DS:SI = source address
                    ;   ES:DI = destination address
                    ;   CX = count
                    ;

                    GETA    MACRO   SEGA,OFFA,DMAA
                            MOV     AX,SEGA         ;;get segment
                            SHL     AX,4            ;shift segment left 4 places
                            ADD     AX,OFFA         ;;add in offset
                            MOV     DX,DMAA         ;;address DMA controller
                            OUT     DX,AL           ;;program rightmost 16-bits
                            PUSHF                   ;;save possible carry
                            MOV     AX,SEGA         ;;get segment
                            SHR     AX,12           ;;for leftmost 4-bits
                            POPF
                            ADD     AX,0            ;;add in possible carry
                            ADD     DX,2
                            OUT     DX,AL
                            ENDM

0000                MOVES   PROC    FAR

                    GETA    DS,SI,0FFC0H            ;program source address
                    GETA    ES,DI,0FFC4H            ;program destination address

0032  BA FFC8               MOV     DX,0FFC8H       ;program count
0035  8B C1                 MOV     AX,CX
0037  EE                    OUT     DX,AL

0038  BA FFCA               MOV     DX,0FFCAH       ;program control
003B  B8 B606               MOV     AX,0B606H
003E  EE                    OUT     DX,AL           ;start transfer

003F  CB                    RET

0040                MOVES   ENDP
                            END
```

The procedure in Example 14–4 transfers data from the data segment location addressed by SI into the extra segment location addressed by DI. The number of bytes transferred is held in register CX. This operation is identical to the REP MOVSB instruction, but execution occurs at a much higher speed.

Chip Selection Unit

The chip selection unit simplifies the interface of memory and I/O to the 80186/80188. This unit contains programmable chip selection logic. In small- and medium-sized systems, no external decoder is required to select memory and I/O. Large systems, however, may still require external decoders. There are two forms of the chip selection unit; one form found in the XL and EA versions differs from the unit found in the EB and EC versions.

Memory Chip Selects. Six pins (XL and EA versions) or 10 pins (EB and EC versions) are used to select different external memory components in a small- or medium-sized 80186/80188-based system. The UCS (upper chip select) pin enables the memory device located in the upper portion of the memory map that is most often populated with ROM. This programmable pin allows the size of the ROM to be specified and the number of wait states required. Note that the ending address of the ROM is FFFFFH. The LCS (lower chip select) pin selects the memory device (usually a RAM) that begins at memory location 00000H. As with the UCS pin, the memory size and number of wait states are programmable. The remaining four or eight pins select middle memory devices. The four pins in the XL and EA version (MCS3–MCS0) are programmed for both the starting (base) address and memory size. Note that all devices must be of the same size. The 8 pins (GCS7–GCS0) in the EB and EC versions are programmed by size and also by starting address, and can represent a memory device or an I/O device.

Peripheral Chip Selects. The 80186/80188 addresses up to seven external peripheral devices with pins PCS6–PCS0 (in the XL and EA versions). The GCS pins are used in the EB and EC versions to select up to eight memory or I/O devices. The base I/O address is programmed at any 1K-byte interval with port address block sizes of 128 bytes.

Programming the Chip Selection Unit for XL and EA Versions. The number of wait states in each section of the memory and the I/O are programmable. The 80186/80188 microprocessors have a built-in wait state generator that can introduce between 0–3 wait states. Table 14–5 lists the logic levels required on bits R2–R0 in each programmable register to select various numbers of wait states. These three lines also select if an external READY signal is required to generate wait states. If READY is selected, the external READY signal is in parallel with the internal wait state generator. For example, if READY is a logic 0 for three clocking periods but the internal wait state generator is programmed to insert two wait states, three wait states are inserted.

Suppose that a 64K-byte EPROM is located at the top of the memory system and requires two wait states for proper operation. To select this device for this section of memory, the UCS pin is programmed for a memory range of F0000H–FFFFFH with two wait states. Figure 14–22 lists the control registers for all memory and I/O selections in the peripheral control block at offset addresses A0–A9H. Notice that the rightmost three bits of these control registers are from Table 14–5. The control register for the upper memory area is at location PCB offset

TABLE 14–5 Wait state control bits R2, R1, and R0 (XL and EA versions).

R2	*R1*	*R0*	*Number of Waits*	*READY required*
0	X	X	–	Yes
1	0	0	0	No
1	0	1	1	No
1	1	0	2	No
1	1	1	3	No

Upper Memory Control Register (offset = A0H)

15 0

0	0	A17	A16	A15	A14	A13	A12	A11	A10	0	0	0	R2	R1	R0

Lower Memory Control Register (offset = A2H)

15 0

0	0	A17	A16	A15	A14	A13	A12	A11	A10	0	0	0	R2	R1	R0

Peripheral Chip Select Base Address (offset = A4H)

15 0

A19	A18	A17	A16	A15	A14	A13	0	0	0	0	0	0	R2	R1	R0

Mid-range Memory Base Address (offset = A6H)

15 0

A19	A18	A17	A16	A15	A14	A13	0	0	0	0	0	0	R2	R1	R0

Mid-range Memory Size (offset = A8H)

15 0

0	M6	M5	M4	M3	M2	M1	M0	EX	MS	0	0	0	R2	R1	R0

FIGURE 14–22 The chip selection registers for the XL and EA versions of the 80186/80188.

address A0H. This 16-bit register is programmed with the starting address of the memory area (F0000H, in this case) and the number of wait states. Please note that the upper two bits of the address must be programmed as 00, and that only address bits A17–A10 are programmed into the control register. See Table 14–6 for examples illustrating the codes for various memory sizes. Because our example requires two wait states, the basic address is the same as in the table for a 64K device, except that the rightmost three bits are 110 instead of 100. The datum sent to the upper memory control register is 3006H.

Suppose that a 32K-byte SRAM that requires no waits and no READY input is located at the bottom of the memory system. To program the LCS pin to select this device, register A2 is loaded in exactly the same manner as register A0H. In this example, a 07FCH is sent to register A2H. Table 14–7 lists the programming values for the lower chip-selection output.

The central part of the memory is programmed via two registers: A6H and A8H. Register A6H programs the beginning or base address of the middle memory select lines (MCS3—MCS0) and number of waits. Register A8H defines the size of the block of memory and the individual memory device size (see Table 14–8). In addition to block size, the number of peripheral wait states are programmed as with other areas of memory. The EX (bit 7) and MS (bit 6) specify the peripheral selection lines, and will be discussed shortly.

TABLE 14–6 Upper memory programming for register A0H (XL and EA versions).

Start Address	*Block Size*	*Value for No Waits, No READY*
FFC00H	1K	3FC4H
FF800H	2K	3F84H
FF000H	4K	3F04H
FE000H	8K	3E04H
FC000H	16K	3C04H
F8000H	32K	3804H
F0000H	64K	3004H
E0000H	128K	1004H
C0000H	256K	0004H

TABLE 14–7 Lower memory programming for register A2H (XL and EA versions).

Ending Address	*Block Size*	*Value for No Waits, No READY*
003FFH	1K	0004H
007FFH	2K	0044H
00FFFH	4K	00C4H
01FFFH	8K	01C4H
03FFFH	16K	03C4H
07FFFH	32K	07C4H
0FFFFH	64K	0FC4H
1FFFFH	128K	1FC4H
3FFFFH	256K	3FC4H

TABLE 14–8 Middle memory programming for register A8H (XL and EA versions).

Block Size	*Chip Size*	*Value for No Waits, No READY, and EX=0 MS=1*
8K	2K	0144H
16K	4K	0344H
32K	8K	0744H
64K	16K	0F44H
128K	32K	1F44H
256K	64K	3F44H
512K	128K	7F44H