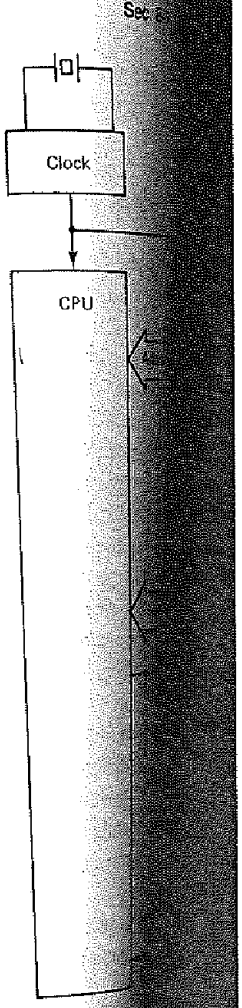
8 System Bus Structure



A set of conductors used for communicating information between the components in a computer system is called a *bus*. If a bus connects two minor components within a major component (e.g., the control unit to the set of working registers within the CPU), it is called an *internal bus*. When a bus connects two major components, such as a CPU and an interface, it is called an *external bus*. Because an internal bus is ordinarily internal to an IC device and its construction is dependent on the device, the exact construction of these buses is of little interest to us. External buses, on the other hand, have common characteristics that must be understood when designing the overall architecture of a computer system. Some systems include more than one external bus and they will be considered in Chap. 11. Others contain only one bus, which is referred to as the *system bus*, and it is the basic structure of system buses, particularly those in 8086- and 8088-based systems, that is discussed in this chapter.

Figure 8-1 illustrates the fundamental structure of a system bus and its relationships to the various components of the computer system. In this chapter we will be concerned primarily with the interface between the CPU and system bus, the bus control logic. Precisely how the other interfaces connected to the bus respond to and interpret the signals on the bus is studied in Chaps. 9 and 10. The complexity of the bus control logic depends on the amount of translation needed between the system bus and the pins on the CPU, the timing requirements, whether or not interrupt management is included, and the size of the overall system. If a system component other than the CPU can be master of the bus, then all of the address and data lines and most of the control lines must be capable of being logically disconnected from the CPU or bus control logic, i.e., most pins connected to the bus must be such that they can enter a high-impedance state.

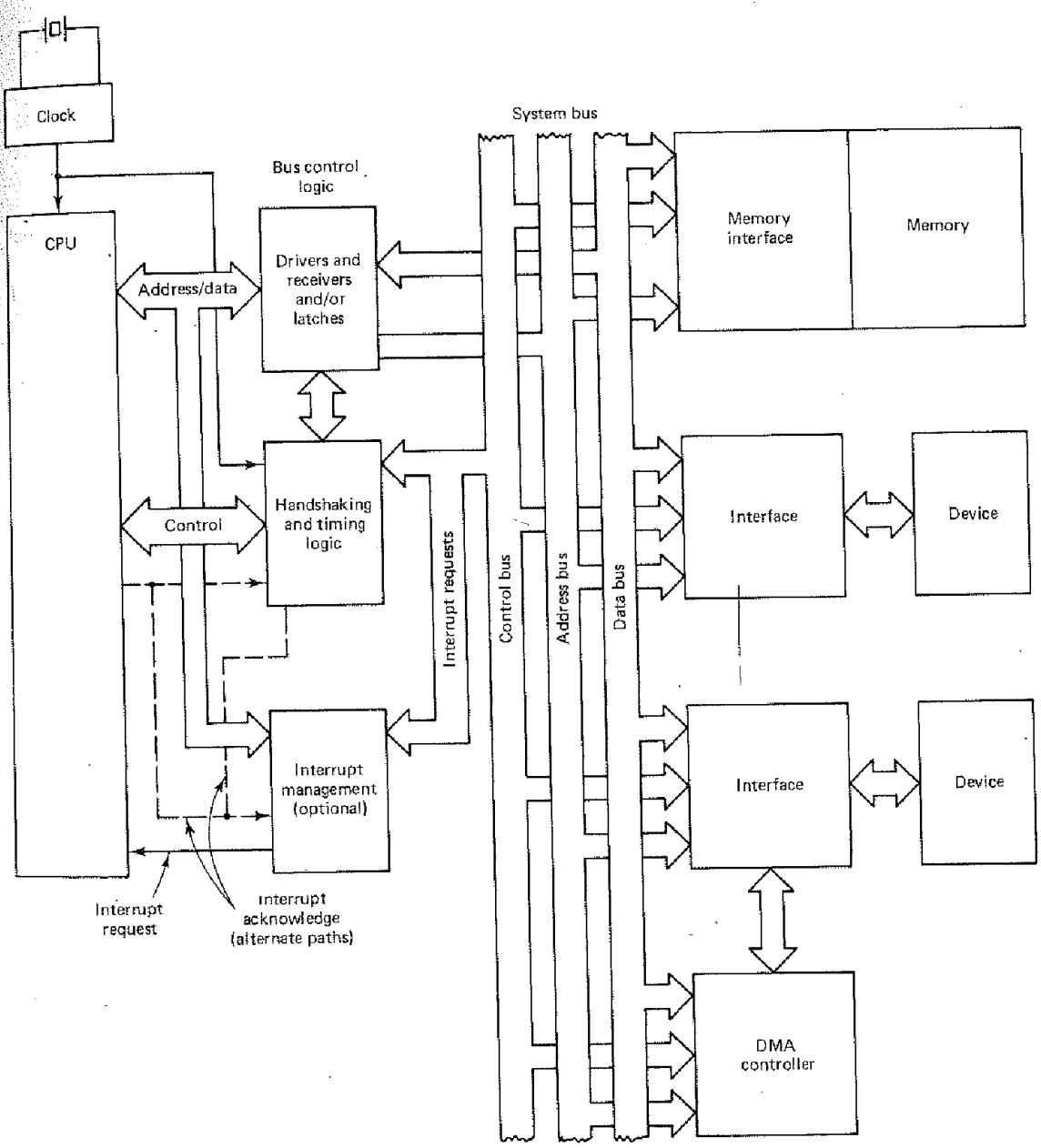


Figure 8-1 Typical system bus architecture.

Generally, the circuits that drive the pins on a single-chip CPU have a quite limited driving capability and can be connected to only a few interfaces. For a small system, many or all of the CPU control pins could be used directly and the handshaking logic could be reduced or, perhaps, eliminated. Similarly, drivers and

receivers would not be needed for the data and address lines. However, systems with several interfaces would need bus driver and receiver circuits connected to the bus in order to maintain adequate signal quality.

In addition to drivers and receivers, timing considerations may be such that latches are needed to hold the address that is output by the CPU during one part of the bus cycle but must be maintained throughout the cycle. Some processors, including the 8086 and 8088, use the same pins for both data and addresses. This means that during a data transfer operation the address, which is output first, must be latched before the bus is used to transfer the data.

The timing of the signals within the CPU and bus control logic is controlled by a clock. The bus cycles and CPU activity are controlled by groups of clock pulses. The exact number of clock pulses, or cycles, within a bus cycle varies. A typical CPU input transaction would proceed by outputting the address of the data during the first clock cycle, signaling that a read is to take place during the second clock cycle, waiting an indeterminate number of clock cycles for the addressed device to put the data on the data lines, inputting the data, and signaling the device that the transfer is complete during the last clock cycle. For an output transaction the address would again be output during the first cycle and the data would be output during the second cycle along with a write signal. After the addressed device has accepted the data it would return a transfer complete acknowledgment which causes the CPU to enter the last cycle.

As in previous chapters the discussion in this chapter will center on the 8086, and, for the most part, the 8088 is considered only by pointing out its differences from the 8086. Keep in mind that the 8088 is fully software compatible with the 8086 and, as an 8-bit processor, is hardware compatible with the peripherals in the 8080/8085 family. By using the 8088, an existing 8080/8085-based system can be updated with minimal hardware modification.

Section 8-1 considers the modes and pin assignments of the 8086 and 8088 microprocessors and the single-bus configurations of systems based on these processors. The second section describes the timing of bus transfers on 8086/8088 systems and the third section examines the Intel 8259A priority interrupt controller. Section 8-4 discusses bus standards by focusing on the Intel MULTIBUS bus standard.

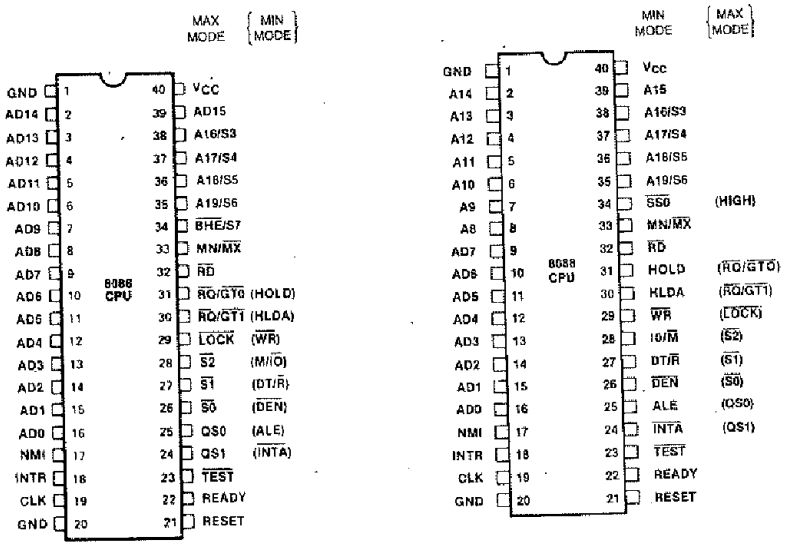
8-1 BASIC 8086/8088 CONFIGURATIONS

In order to adapt to as many situations as possible both the 8086 and 8088 have been given two modes of operation, the minimum mode and the maximum mode. The minimum mode is used for a small system with a single processor, a system in which the 8086/8088 generates all the necessary bus control signals directly (thereby minimizing the required bus control logic). The maximum mode is for medium-size to large systems, which often include two or more processors. In the maximum mode, the 8086/8088 encodes the basic bus control signals into 3 status

bits, and uses the remaining control pins to provide the additional information that is needed to support a multiprocessor configuration.

Pin diagrams for the 8086 and 8088 and the pin definitions that are common to both modes are given in Fig. 8-2. Pin 33 (MN/$\overline{MX}$) determines the configuration option. When it is strapped to ground the processor is to be used in a maximum mode configuration and when it is strapped to +5 V it is to be operated in its minimum mode. Both processors multiplex the address and data signals and both have 20 address pins with address and status signals being multiplexed on the 4 most significant address pins. However, because the 8088 can only transfer 8 bits

Figure 8-2 Pin assignment summary. [Parts (a) and (b) reprinted by permission of Intel Corporation. Copyright 1981.]



(a) 8086 pin diagram

(b) 8088 pin diagram

Pin(s)	Symbol	In/Out 3-State	Description
1	GND		Ground
2-16[1]	AD14-AD0	I/O-3	Outputs address during the first part of the bus cycle and inputs or outputs data during the remaining part of the bus cycle.
17	NMI	I	Nonmaskable interrupt request - positive-going edge triggered.
18	INTR	I	Maskable interrupt request - level triggered
19[2]	CLK	I	Clock - 33% duty cycle, maximum rate depends on CPU model 5 MHz for 8086 8 MHz for 8086-2 10 MHz for 8086-1

20	GND		Ground
21	RESET	I	Terminates activity, clears PSW, IP, DS, SS, ES, and the instruction queue, and sets CS to FFFF. Processing begins at FFFF0 when signal is dropped. Signal must be 1 for at least 4 clock cycles.
22	READY	I	Acknowledgment from memory or I/O interface that CPU can complete the current bus cycle.
23	$\overline{\text{TEST}}$	I	Used in conjunction with the WAIT instruction in multiprocessing environments. A WAIT instruction will cause the CPU to idle, except for processing interrupts, until a 0 is applied to this pin - see Chap. 11.
24-31	-	-	Definition depends on mode - see Figs. 8-3 and 8-8.
32	$\overline{\text{RD}}$	O-3	Indicates a memory or I/O read is to be performed.
33	MN/$\overline{\text{MX}}$	I	CPU is in minimum mode when strapped to +5 V and in maximum mode when grounded.
34[3]	$\overline{\text{BHE}}$/S7	O-3	If 0 during first part of bus cycle this pin indicates that at least one byte of the current transfer is to be made on pins AD15-AD8; if 1 the transfer is made on AD7-AD0. Status S7 is output during the latter part of bus cycle, but, presently, S7 has not been assigned a meaning.
35-38	A19/S6-A16/S3	O-3	During the first part of the bus cycle the upper 4 bits of the address are output and during the remainder of the bus cycle status is output. S3 and S4 indicate the segment register being used as follows: S4 S3 Register 0 0 ES 0 1 SS 1 0 CS or none 1 1 DS S5 gives the current setting of IF. S6 is always 0.
39[1]	AD15	I/O-3	Same as AD14-AD0
40	VCC	-	Supply voltage - +5 V ± 10%

[1] On 8088, AD15-AD8 are A15-A8 and are only for outputting address bits.

[2] 5 MHz for the 8088, and 8 MHz for the 8088-2.

[3] On 8088, this pin is denoted $\overline{\text{SS0}}$ and is used in minimum mode to denote status. Logically equivalent to $\overline{\text{S0}}$ - see Fig. 8-8. It is always 1 in maximum mode.

(c) Pin definitions

Figure 8-2 Continued.

of data at a time, only eight of its pins are used for data, as opposed to 16 for the 8086. Except for pins 28 and 34 the two processors have the same control pin definitions. Pin 28 differs only in the minimum mode. For the 8088 this minimum mode signal is inverted from that of the 8086, so that the 8088 is compatible with the Intel 8085 microcomputer chip.

On the 8086, pin 34 ($\overline{\text{BHE}}$) designates whether or not at least 1 byte of a

transfer is to be made on AD15 through AD8. A 0 on this pin indicates that the more significant data lines are to be used; otherwise, only AD7 through AD0 are used. Together the $\overline{BHE}$ and A0 signals indicate to the interfaces connected to the bus how the data are to appear on the bus. The four possible combinations are defined as follows:

Operation	$\overline{BHE}$	A0	Data pins used
Write/read a word at an even address	0	0	AD15–AD0
Write/read a byte at an even address	1	0	AD7–AD0
Write/read a byte at an odd address	0	1	AD15–AD8
Write/read a word at an odd address	0	1	AD15–AD8 (First bus cycle: puts the least significant data byte on AD15–AD8)
	1	0	AD7–AD0 (Next bus cycle: puts the most significant data byte on AD7–AD0)

where 0 is low and 1 is high.

Because, on the 8088, only AD7–AD0 can transfer data, this pin is not needed to indicate the upper or lower half of the data bus and is free to provide status information.

Pins 1 and 20 are grounded. Pins 2 through 16 and 39 (AD15–AD0) hold the address needed for the transfer during the first part of the bus cycle, and are free to transfer the data during the remaining part of the cycle.

Pins 17 and 18 (NMI and INTR) are for interrupt requests, which were discussed in Chap. 6. Pin 19 (CLK) is for supplying the clock signal that synchronizes the activity within the CPU.

Pin 21 (RESET) is for inputting a system reset signal. Most systems include a line that goes to all system components and a pulse is automatically sent over this line when the system is turned on, or the reset pulse can be manually generated by a switch that allows the operator to reinitialize the system. A 1 on the reset line causes the components to go to their "turn on" state. For the processor this state is having the PSW, IP, DS, SS, ES, and instruction queue cleared and CS set to FFFF. With (IP) = 0000 and (CS) = FFFF the processor will begin executing at FFFF0. Normally, this location would be in a read-only section of memory and would contain a JMP instruction to a program for initializing the system and loading the application software or operating system. Such a program is referred to as a *bootstrap loader.*

Pin 22 (READY) is for inputting an acknowledge from a memory or I/O interface that input data will be put on the data bus or output data will be accepted from the data bus within the next clock cycle. In either case, the CPU and its bus control logic can complete the current bus cycle after the next clock cycle. Pin 23

($\overline{TEST}$) is used in conjunction with the WAIT instruction and is employed primarily in multiprocessing situations (see Chap. 11). Pins 24 through 31 are mode dependent and are considered later. Pin 32 ($\overline{RD}$) indicates that an input operation is to be performed and, in minimum mode, is used along with pin 28, which distinguishes a memory transfer from an I/O transfer, and pin 29, which indicates an output operation, to determine the type of transfer.

During the first part of a bus cycle pins 35–38 (AD19/S6–AD16/S3) output the 4 high-order bits of the address, and during the remaining part of the cycle they output status information. Status bits S3 and S4 indicate the segment register that is being used to generate the address and bit S5 reflects the contents of the IF flag. S6 is always held at 0 and indicates that an 8086/8088 is controlling the system bus (see page 460).

Pin 40 (VCC) receives the supply voltage, which must be +5 V ± 10%. Systems based on an 8086 or 8088 are ordinarily designed so that only a TTL-compatible +5-V supply voltage and ground are needed, thus simplifying the design of the power supply.

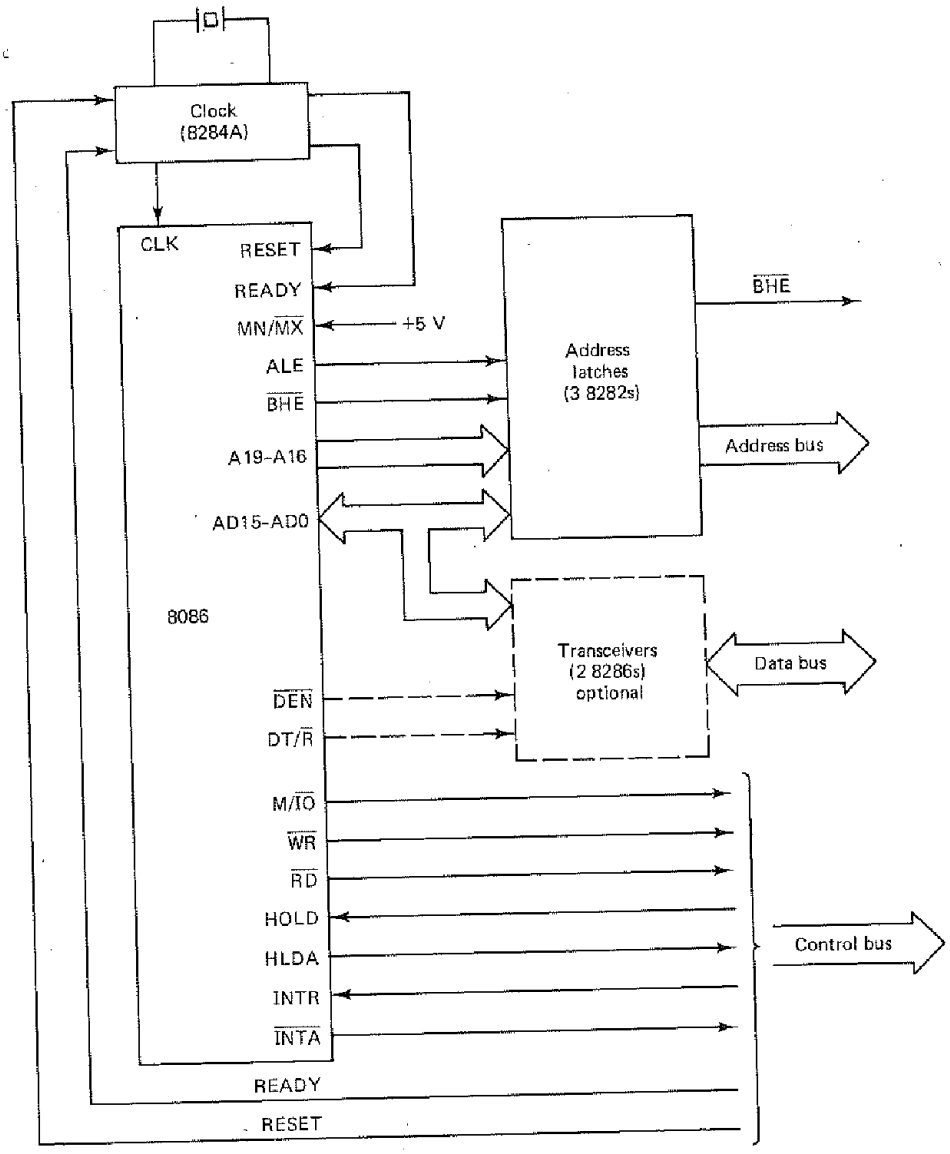
8-1-1 Minimum Mode

A processor is in minimum mode when its MN/$\overline{MX}$ pin is strapped to +5 V. The definitions for pins 24 through 31 for the minimum mode are given in Fig. 8-3 and a typical minimum mode configuration is shown in Fig. 8-4. The address must be

Figure 8-3 Pin definitions for the minimum mode.

Pin	Symbol	In/Out 3-State	Description
24	$\overline{INTA}$	O-3	Indicates recognition of an interrupt request. Consists of two negative going pulses in two consecutive bus cycles.
25	ALE	O	Outputs a pulse at the beginning of the bus cycle and is to indicate an address is available on the address pins.
26	$\overline{DEN}$	O-3	Output during the latter portion of the bus cycle and is to inform the transceivers that the CPU is ready to send or receive data.
27	DT/$\overline{R}$	O-3	Indicates to the set of transceivers whether they are to transmit (1) or receive (0) data.
28[1]	M/$\overline{IO}$	O-3	Distinguishes a memory transfer from an I/O transfer. For a memory transfer it is 1.
29	$\overline{WR}$	O-3	When 0, it indicates a write operation is being performed. It is used in conjunction with pins 28 (M/$\overline{IO}$) and 32 ($\overline{RD}$) to specify the type of transfer.
30	HLDA	O	Outputs a bus grant to a requesting master. Pins with tristate gates are put in high impedance state while HLDA=1.
31	HOLD	I	Receives bus requests from bus masters. The 8086/8088 will not gain control of the bus until this signal is dropped.

[1]For the 8088, the symbol is IO/$\overline{M}$ and a 1 indicates an I/O transfer.



Note: In an 8088 system $\overline{BHE}$ is $\overline{SSO}$, M/$\overline{IO}$ is IO/$\overline{M}$, and only one 8286 is needed.

Figure 8-4 Minimum mode system.

latched since it is available only during the first part of the bus cycle. To signal that the address is ready to be latched a 1 is put on pin 25, the address latch enable (ALE) pin. Typically, the latching is accomplished using Intel 8282s, as shown in Fig. 8-5. Because 8282 is an 8-bit latch, two of them are needed for a 16-bit address and three are needed if a full 20-bit address is used. In an 8086 system, $\overline{BHE}$ would

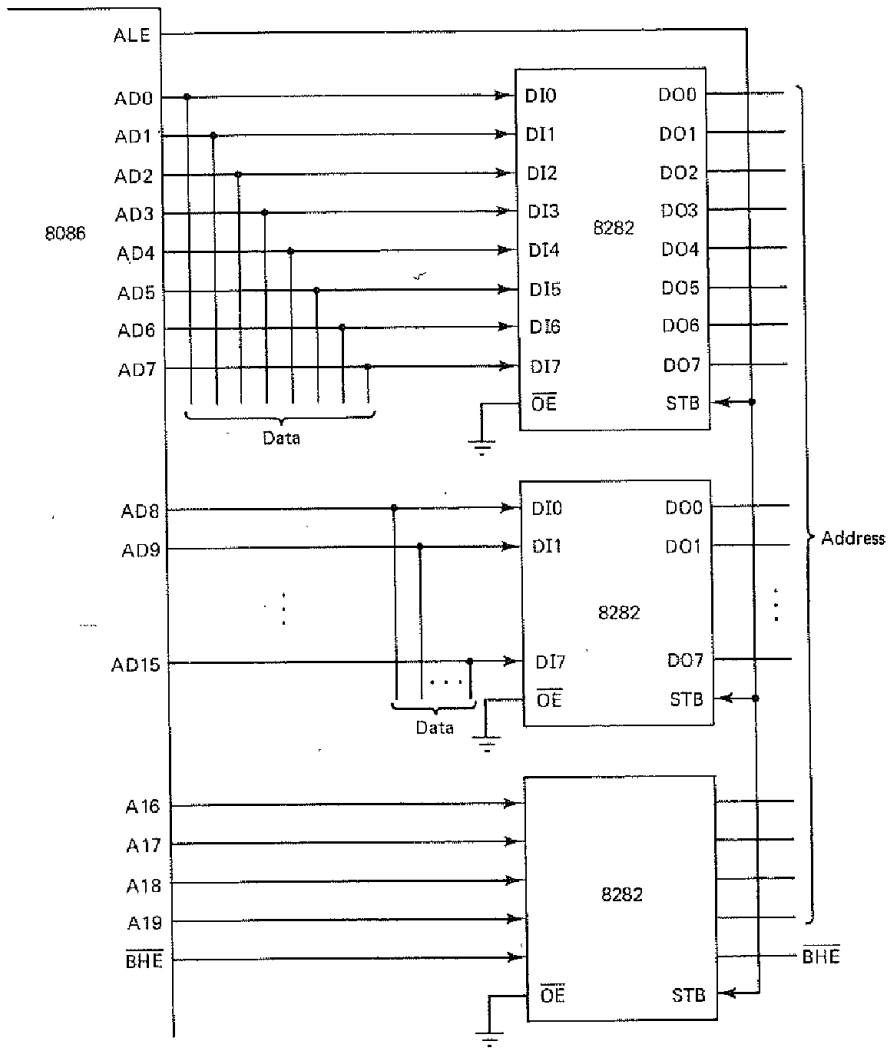


Figure 8-5 Application of 8282 latches.

also have to be latched. For a small 8088 system that has only 64k bytes of memory, only two 8282s would be required. A signal on the STB pin latches the bits applied to the input data lines DI7–DI0. Therefore, STB is connected to the 8086's ALE pin and DI7–DI0 are attached to eight of the address lines. An active low signal on the $\overline{OE}$ enables the latch's outputs DO7–DO0, and a 1 at this pin forces the outputs into their high-impedance state. In an 8086/8088 single-processor system that does not include a DMA controller (DMA is discussed in Chap. 9) this pin is grounded.

If a system includes several interfaces, then drivers and receivers, which may

not be needed on small, single-board systems, will be required for the data lines. The Intel IC device for implementing the transceiver (driver/receiver) block shown in Fig. 8-4 is the 8286 transceiver device. The 8286 contains 16 tristate elements, eight receivers, and eight drivers. Therefore, only one 8286 is needed to service all of the data lines for an 8088, but two are required in an 8086 system. Figure 8-6 shows how 8286s are connected into a system and a logic diagram of one of its cells. The 8286 is symmetric with respect to its two sets of data pins, either the pins A7–A0 can be the inputs and B7–B0 the outputs, or vice versa. The output enable ($\overline{OE}$) pin determines whether or not data are allowed to pass through the 8286 and the transmit (T) pin controls the direction of the data flow. When $\overline{OE} = 1$, data are not transmitted through the 8286 in either direction. If it is 0, then $T = 1$ causes A7–A0 to be the inputs and $T = 0$ results in B7–B0 being the inputs. In an 8086/8088-based system the $\overline{OE}$ pin would be connected to the $\overline{DEN}$ pin, which is active low whenever the processor is performing an I/O operation. The A7–A0 pins are connected to the appropriate address/data lines and the T pin is tied to the processor's DT/$\overline{R}$ pin. Thus, when the processor is outputting the data flow is from A7–A0 to B7–B0, and when it is inputting the flow is in the other direction. The processor floats the $\overline{DEN}$ and DT/$\overline{R}$ pins in response to a bus request on the HOLD pin.

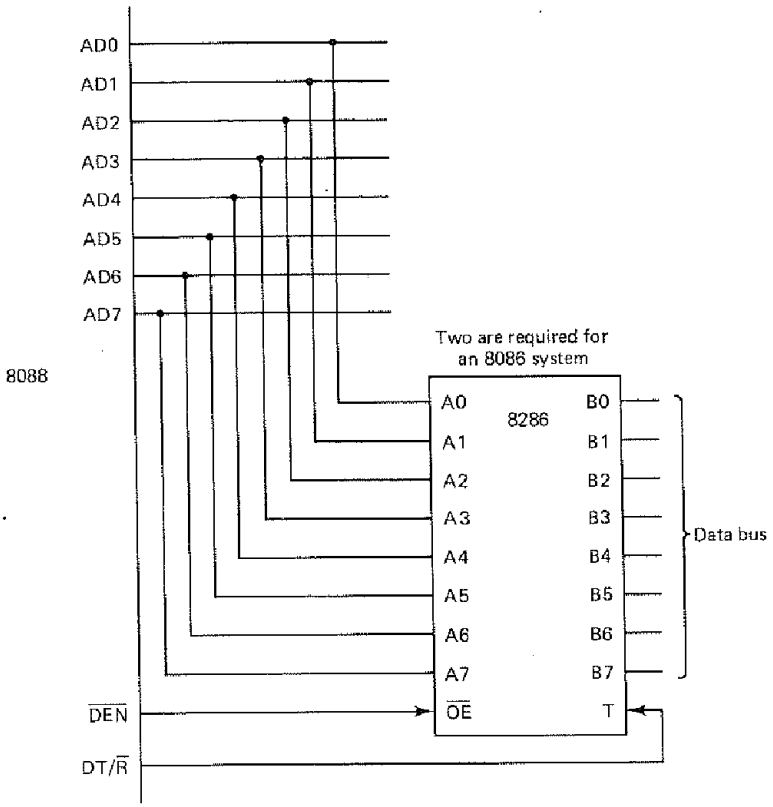
Sometimes a system bus is designed so that the address and/or data signals are inverted. Therefore, the 8282 and 8286 both have companion chips that are the same as the 8282 and 8286 except that they cause an inversion between their inputs and outputs. The companion for the 8282 is the 8283 and the companion for the 8286 is the 8287.

The third component, other than the processor, that appears in Fig. 8-4 is an 8284A clock generator. This device, which is actually more than just a clock, is detailed in Fig. 8-7. In addition to supplying a train of pulses at a constant frequency it synchronizes ready (RDY) signals, which indicate an interface is ready to complete a transfer, and reset ($\overline{RES}$) signals, which initialize the system, with the clock pulses. Although these two signals may be sent at any time, the 8284A will not reflect them in its READY and RESET outputs until the trailing edge of the clock pulse in which they are received.

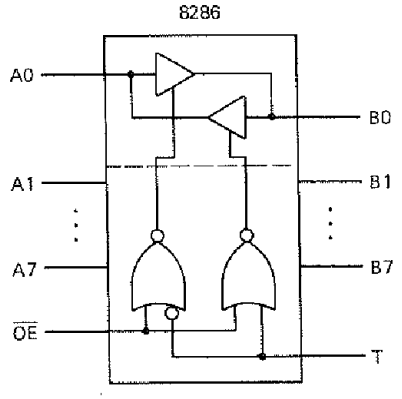
The frequency source applied to the 8284A may be from a pulse generator that is connected to the EFI pin or an oscillator that is connected across X1 and X2. If the input to F/$\overline{C}$ is 1, then the EFI input determines the frequency; otherwise, it is the oscillator input. In either case the clock output CLK is one-third of the input frequency.

All three of the devices considered above, the 8282, 8286, and 8284A, require only +5-V supply voltages. Their inputs and outputs are TTL compatible and, therefore, the devices are compatible with each other and with the 8086 and 8088. Many of the details of these devices have not been included here and, for more information, one should refer to the Intel manuals—see Reference 1. This comment applies particularly to the 8284A clock.

In a minimum system the control lines do not need to be passed through transceivers, but can be used directly. The M/$\overline{IO}$, $\overline{RD}$, and $\overline{WR}$ lines specify the



(a) 8088 connections



(b) Internal logic

Figure 8-6 Application and internal logic of an 8286.

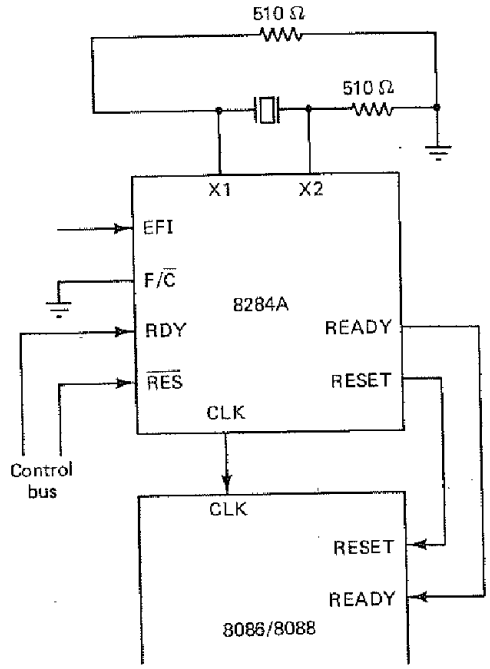


Figure 8-7 Typical 8284A clock connection.

type of transfer according to the following table:

M/IO	RD	WR	
0	0	1	I/O read
0	1	0	I/O write
1	0	1	Memory read
1	1	0	Memory write

where 0 is low and 1 is high.

The purposes of the bus request (HOLD), bus grant (HLDA), interrupt request (INTR), and interrupt acknowledge (INTA) lines were discussed in Chap. 6. The INTA signal consists of two negative pulses output during two consecutive bus cycles. The first pulse informs the interface that its request has been recognized, and upon receipt of the second pulse, the interface is to send the interrupt type to the processor over the data bus.

8-1-2 Maximum Mode

A processor is in maximum mode when its MN/MX pin is grounded. The maximum mode definitions of pins 24 through 31 are given in Fig. 8-8 and a typical maximum mode configuration is shown in Fig. 8-9. It is clear from Fig. 8-9 that the main difference between minimum and maximum mode configurations is the need for additional circuitry to translate the control signals. This circuitry is for converting the status bits S0, S1, and S2 into the I/O and memory transfer signals needed to

Pin	Symbol	In/Out 3-State	Description
24,25	QS1,QS0	O	Reflects the status of the instruction queue. This status indicates the activity in the queue during the previous clock cycle - see Chap. 11.
26,27,28	$\overline{S0}$,$\overline{S1}$,$\overline{S2}$	O-3	Indicates the type of transfer to take place during the current bus cycle: $\overline{S2}$ $\overline{S1}$ $\overline{S0}$ 0 0 0 Interrupt acknowledge 0 0 1 Read I/O port 0 1 0 Write I/O port 0 1 1 Halt 1 0 0 Instruction fetch 1 0 1 Read memory 1 1 0 Write memory 1 1 1 Inactive - passive (1 represents high and 0 represents low.) The status becomes active prior to the beginning of a bus cycle and returns to inactive during the later part of the cycle.
29	$\overline{LOCK}$	O-3	Indicates the bus is not to be relinquished to other potential bus masters. It is initiated by a LOCK instruction prefix and is maintained until the end of the next instruction - see Chap. 11. It is also active during and between the two $\overline{INTA}$ pulses.
30	$\overline{RQ/GT1}$	I/O	For inputting bus requests and outputting bus grants.
31	$\overline{RQ/GT0}$	I/O	Same as $\overline{RQ/GT1}$ except that a request on $\overline{RQ/GT0}$ has higher priority.

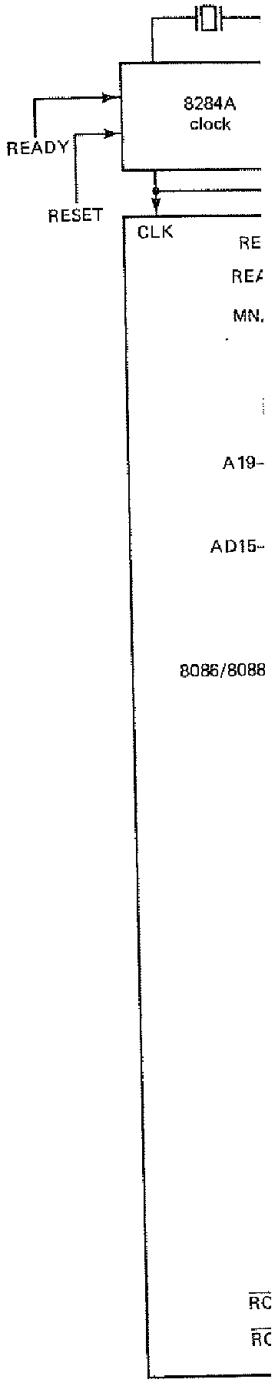
Note: In maximum mode the 8086 and 8088 pins have the same definitions except for pin 34, which on the 8088 is always 1.

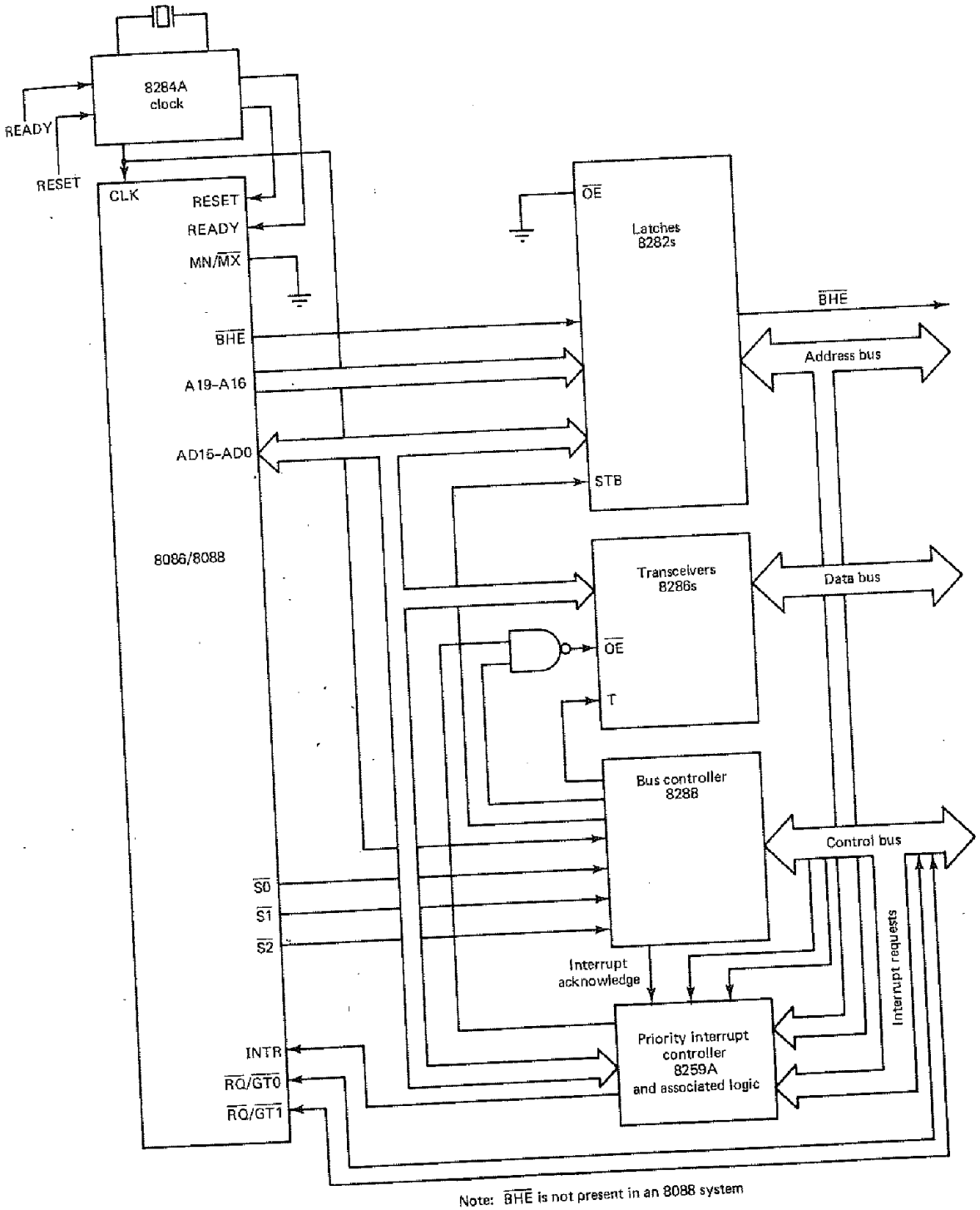
Figure 8-8 Maximum mode pin definitions.

direct data transfers, and for controlling the 8282 latches and 8286 transceivers. It is normally implemented with an Intel 8288 bus controller. Also included in the system is an interrupt priority management device; however, its presence is optional.

The $\overline{S0}$, $\overline{S1}$, and $\overline{S2}$ status bits specify the type of transfer that is to be carried out and when used with an 8288 bus controller they obviate the need for the M/$\overline{IO}$ (or IO/$\overline{M}$), $\overline{WR}$, $\overline{INTA}$, ALE, DT/$\overline{R}$, and $\overline{DEN}$ signals that are output over pins 24 through 29 when the processor is operating in minimum mode. Except for the case $\overline{S1}$ = $\overline{S0}$ = 1, $\overline{S2}$ = 0 indicates a transfer between an I/O interface and the CPU and $\overline{S2}$ = 1 implies a memory transfer. The $\overline{S1}$ bit specifies whether an input or output is to be performed. From the status the 8288 is able to originate the address latch enable signal to the 8282s, the enable and direction signals to the 8286 transceivers, and the interrupt acknowledge signal to the interrupt controller.

The QS0 and QS1 pins are to allow the system external to the processor to interrogate the status of the processor instruction queue so that it can determine which instruction it is currently executing, and the $\overline{LOCK}$ pin indicates that an instruction with a LOCK prefix is being executed and the bus is not to be used by





Note: $\overline{BHE}$ is not present in an 8088 system

Figure 8-9 Typical maximum mode configuration.

another potential master. These pins are needed only in multiprocessor systems and, along with the LOCK prefix, are discussed in detail in Chap. 11.

The HOLD and HLDA pins become the $\overline{RQ}/\overline{GT0}$ and $\overline{RQ}/\overline{GT1}$ pins. Both bus requests and bus grants can be given through each of these pins. They are exactly the same except that if requests are seen on both pins at the same time, then the one on $\overline{RQ}/\overline{GT0}$ is given higher priority. A request consists of a negative pulse arriving before the start of the current bus cycle. The grant is a negative pulse that is issued at the beginning of the current bus cycle provided that:

1. The previous bus transfer was not the low byte of a word to or from an odd address if the CPU is an 8086. For an 8088, regardless of the address alignment, the grant signal will not be sent until the second byte of a word reference is accessed.
2. The first pulse of an interrupt acknowledgment did not occur during the previous bus cycle.
3. An instruction with a LOCK prefix is not being executed.

If condition 1 or 2 is not met, then the grant will not be given until the next bus cycle, and if condition 3 is not met, the grant will wait until the locked instruction is completed. In response to the grant the three-state pins are put in their high-impedance state and the next bus cycle will be given to the requesting master. The processor will be effectively disconnected from the system bus until the master sends a second pulse to the processor through the $\overline{RQ}/\overline{GT}$ pin.

An expanded view of a maximum mode system which shows only the connections to an 8288 is given in Fig. 8-10. The $\overline{S0}$, $\overline{S1}$, and $\overline{S2}$ pins are for receiving the corresponding status bits from the processor. The ALE, DT/$\overline{R}$, and DEN pins provide the same outputs that are sent by the processor when it is in minimum mode (except that DEN is inverted from $\overline{DEN}$). The CLK input permits the bus controller activity to be synchronized with that of the processor. The $\overline{AEN}$, IOB, and CEN pins are for multiprocessor systems and are considered in Chap. 11. In a single-processor system $\overline{AEN}$ and IOB are normally grounded and a 1 is applied to CEN. The meaning of the MCE/$\overline{PDEN}$ output depends on the mode, which is determined by the signal applied to IOB. When IOB is grounded it assumes its master cascade enable (MCE) meaning and can be used to control cascaded 8259As (see Sec. 8-3). In the event that +5 V is connected to IOB, the peripheral data enable ($\overline{PDEN}$) meaning, which is used in multiple-bus configurations, is assumed.

The remaining pins given in Fig. 8-10 have the following definitions:

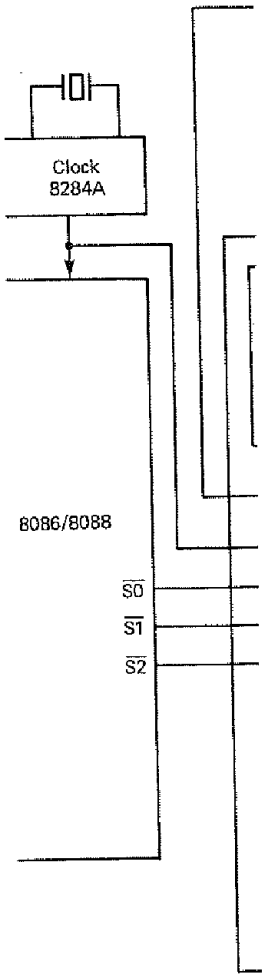
$\overline{INTA}$—Issues the two interrupt acknowledgment pulses to a priority interrupt controller or an interrupting device when $\overline{S0} = \overline{S1} = \overline{S2} = 0$.

$\overline{IORC}$ (I/O Read Command)—Instructs an I/O interface to put the data contained in the addressed port on the data bus.

$\overline{IOWC}$ (I/O Write Command)—Instructs an I/O interface to accept the data on the data bus and put the data into the addressed port.

$\overline{MRD}$
of th
$\overline{MW}$
on th

These sign
cycle. Cle
Not
$\overline{AMWC}$ (a



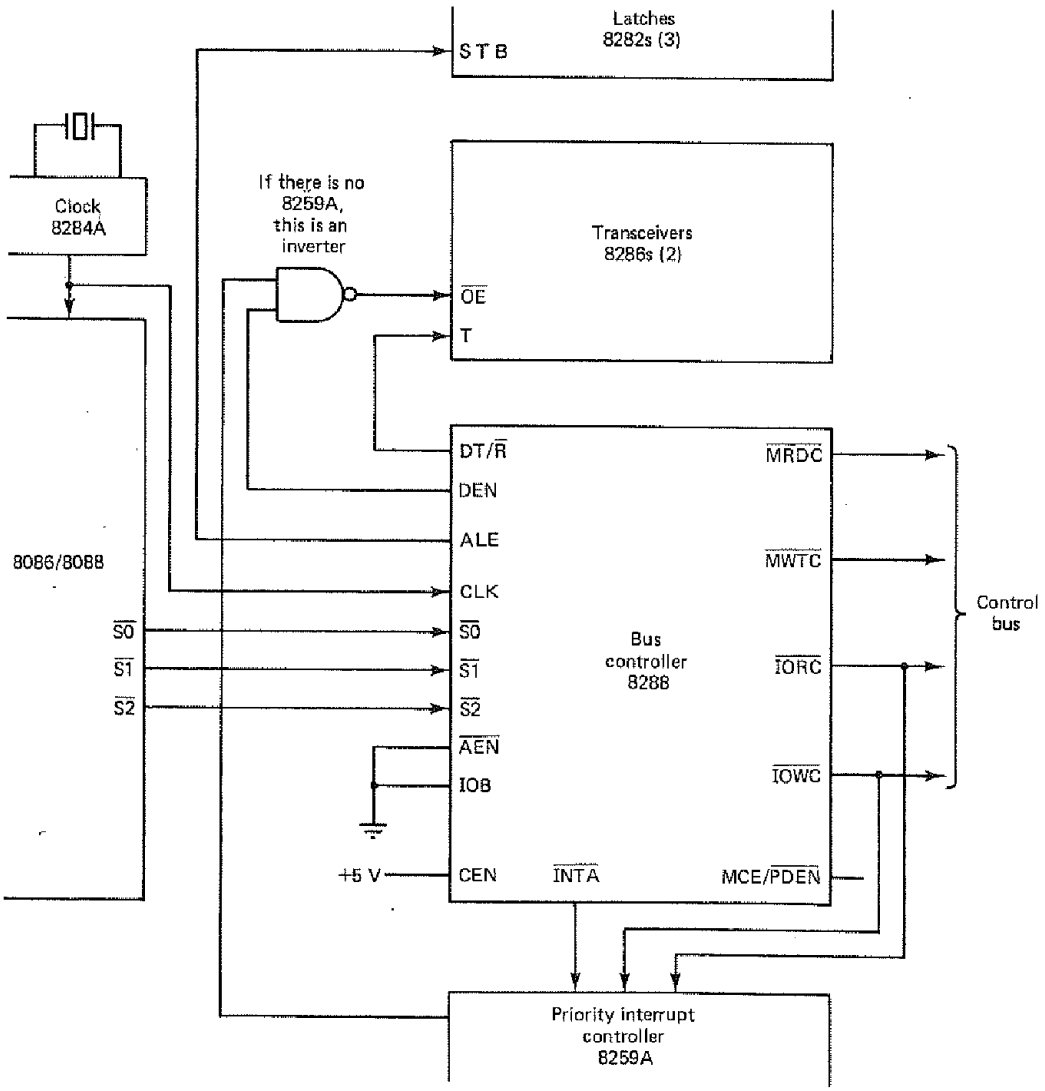
$\overline{\text{MRDC}}$ (Memory Read Command)—Instructs the memory to put the contents of the addressed location on the data bus.

$\overline{\text{MWTC}}$ (Memory Write Command)—Instructs the memory to accept the data on the data bus and put the data into the addressed memory location.

These signals are active low and are output during the middle portion of a bus cycle. Clearly, only one of them will be issued during any given bus cycle.

Not shown in Fig. 8-10 are the $\overline{\text{AIOWC}}$ (advanced I/O write command) and $\overline{\text{AMWC}}$ (advanced memory write command) pins. They serve the same purposes

Figure 8-10 Connections to an 8288 bus controller.



as the $\overline{IOWC}$ and $\overline{MWTC}$ pins except that they are activated one clock pulse sooner. This gives slow interfaces an extra clock cycle to prepare to input the data.

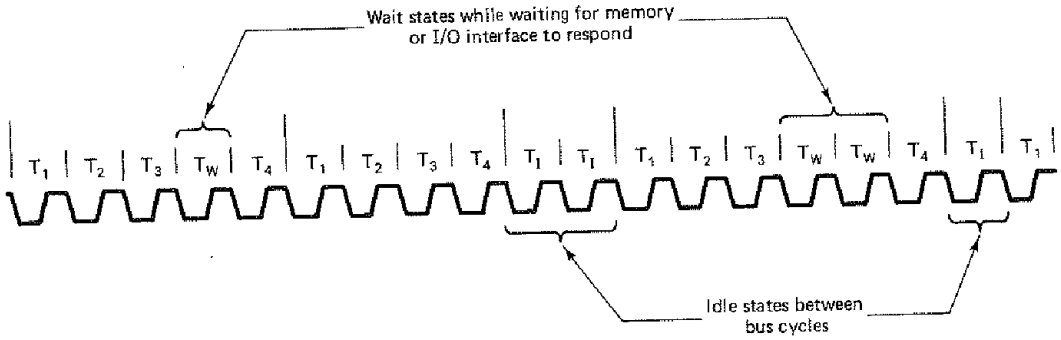
As with the other 8086 supporting devices, the 8288 requires a +5-V supply voltage and has TTL-compatible inputs and outputs. For more detailed information one should refer to the Intel manuals—see Reference 1.

8-2 SYSTEM BUS TIMING

Until now we have vaguely referred to the first part, middle part, and last part of the bus cycle. It is the objective of this section to put the discussion of timing on a more precise footing. The length of a bus cycle in an 8086/8088 system is four clock cycles, denoted T_1 through T_4, plus an indeterminate number of wait state clock cycles, denoted T_W. If the bus is to be inactive after the completion of a bus cycle, then the gap between successive cycles is filled with idle state clock cycles represented by T_I. Wait states are inserted between T_3 and T_4 when a memory or I/O interface is not able to respond quickly enough during a transfer. A typical succession of bus cycles is given in Fig. 8-11.

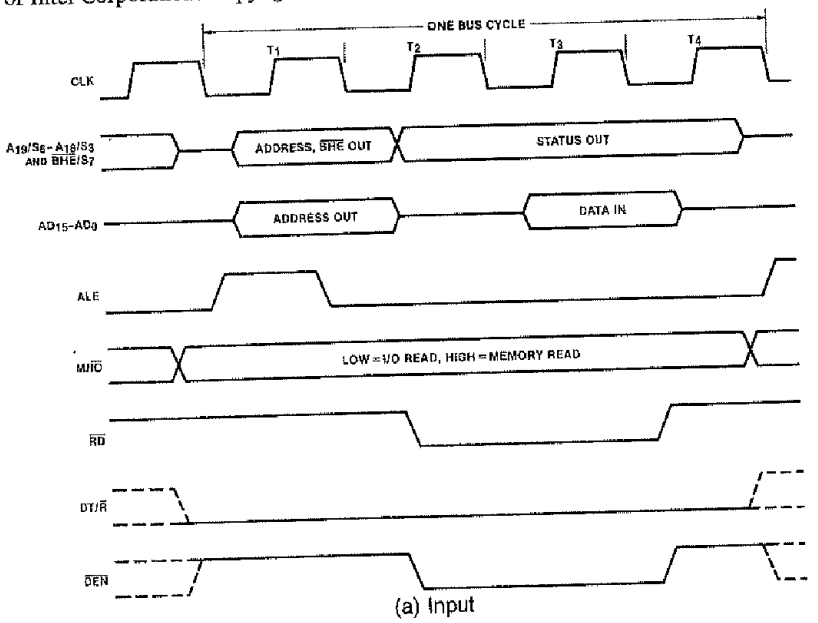
The timing diagrams for 8086 minimum mode input and output transfers that require no wait states are shown in Fig. 8-12. When the state of the processor is such that it is ready to initiate a bus cycle it applies a pulse to the ALE pin during T_1. Before the trailing edge of the ALE signal the address, $\overline{BHE}$, M/$\overline{IO}$, $\overline{DEN}$, and DT/$\overline{R}$ signals should be stable, with $\overline{DEN}$ = 1 and DT/$\overline{R}$ = 0 for an input and DT/$\overline{R}$ = 1 for an output. At the trailing edge of the ALE signal the 8282s latch the address. During T_2 the address is dropped and S3 through S7 are output on AD16/S3–AD19/S6 and $\overline{BHE}$/S7, and $\overline{DEN}$ is lowered to enable the 8286 transceivers. If an input is being conducted, $\overline{RD}$ is activated low during T_2 and AD15–AD0 enter a high-impedance state in preparation for input. If the memory or I/O interface can perform the transfer immediately, there are no wait states and the data are put on the bus during T_3. After the input data are accepted by the processor, $\overline{RD}$ is raised to 1 at the beginning of T_4 and, upon detecting this transition, the

Figure 8-11 Typical sequence of bus cycles.

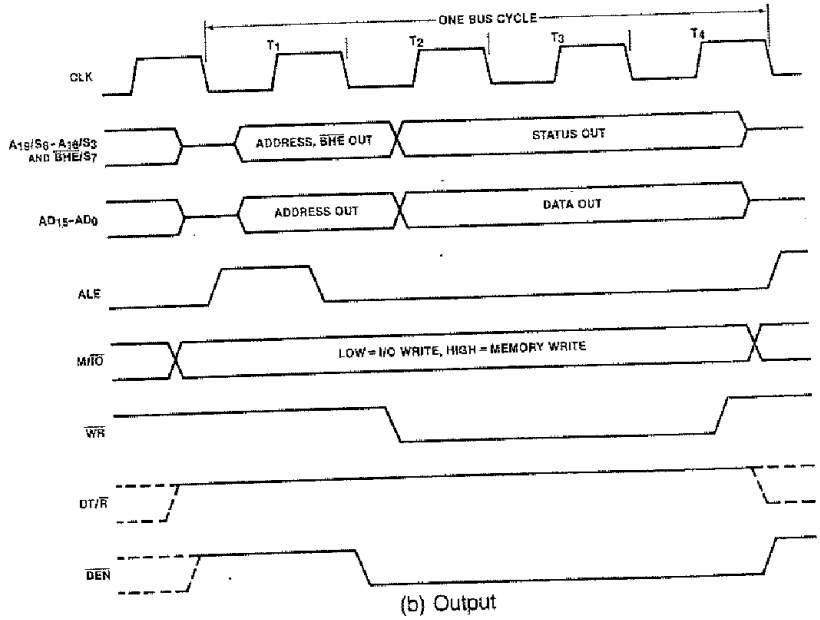


memory or I/O interface will drop its data signals. For an output, the processor applies the $\overline{WR} = 0$ signal and then the data during T_2, and in T_4 $\overline{WR}$ is raised and the data signals are dropped. For either an input or output, $\overline{DEN}$ is raised

Figure 8-12 8086 minimum mode bus timing diagrams (Reprinted by permission of Intel Corporation. Copyright 1979.)



(a) Input



(b) Output

Note: For an 8088, M/$\overline{IO}$ is IO/$\overline{M}$ and $\overline{BHE}$/S7 becomes $\overline{SSO}$ which is present throughout the bus cycle (i.e., it changes at the same time as IO/$\overline{M}$). Also, only AD7-AD0 carry data.

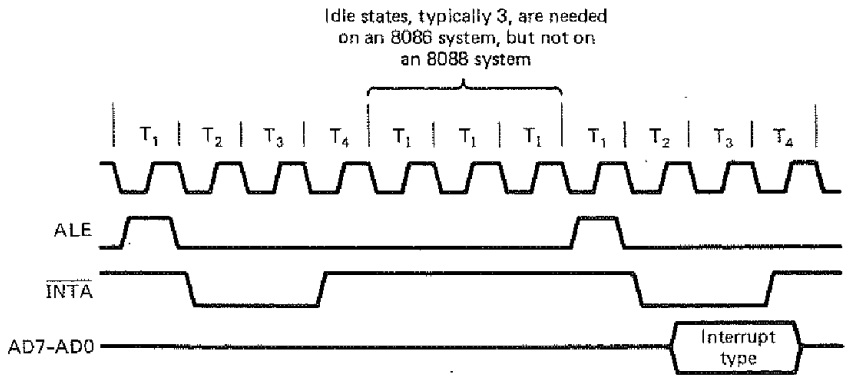
during T_4 to disable the transceivers and the M/$\overline{IO}$ signal is set according to the next transfer at this time or during a subsequent T_I state.

The bus timing has been designed so that the memory or I/O interface involved in a transfer can control when data are to be placed on or taken from the bus by the interface. This is done by having the interface send a READY signal to the processor (perhaps via an 8284A) when it has made data available or accepted data. If a READY signal has not been received by the processor by the beginning of T_3, then one or more T_W states will be inserted between T_3 and T_4 until a READY has been received. The bus activity during T_W is the same as during T_3. A signal applied to an RDY input of an 8284A will cause a READY output to the processor at the trailing edge of the current clock cycle; therefore, if a wait state is to be avoided, an RDY input must be received before the beginning of the T_3 clock cycle.

The timing diagram for an interrupt acknowledge is shown in Fig. 8-13. If an interrupt request has been recognized during the previous bus cycle and an instruction has just been completed, then a negative pulse will be applied to $\overline{INTA}$ during the current bus cycle and the next bus cycle. Each of these pulses will extend from T_2 to T_4. Upon receiving the second pulse, the interface accepting the acknowledgment will put the interrupt type on AD7–AD0, which are floated the rest of the time during the two bus cycles. The type will be available from T_2 to T_4.

Figure 8-14 shows the timing of a bus request and bus grant in a minimum mode system. The HOLD pin is tested at the leading edge of each clock pulse. If a HOLD signal is received by the processor before T_4 or during a T_I state, then the CPU activates HLDA and the succeeding bus cycles will be given to the requesting master until that master drops its request. The lowered request is detected at the rising edge of the next clock cycle and the HLDA signal is dropped at the trailing edge of that clock cycle. While HLDA is 1, all of the processor's three-state outputs are put in their high-impedance state. Instructions already in

Figure 8-13 Interrupt acknowledgment.



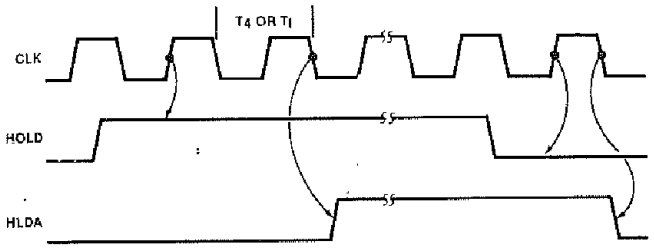


Figure 8-14 Bus request and bus grant timing on a minimum mode system. (Reprinted by permission of Intel Corporation. Copyright 1979.)

the instruction queue will continue to be executed until one of them requires the use of the bus. The instruction

```
MOV     AX,BX
```

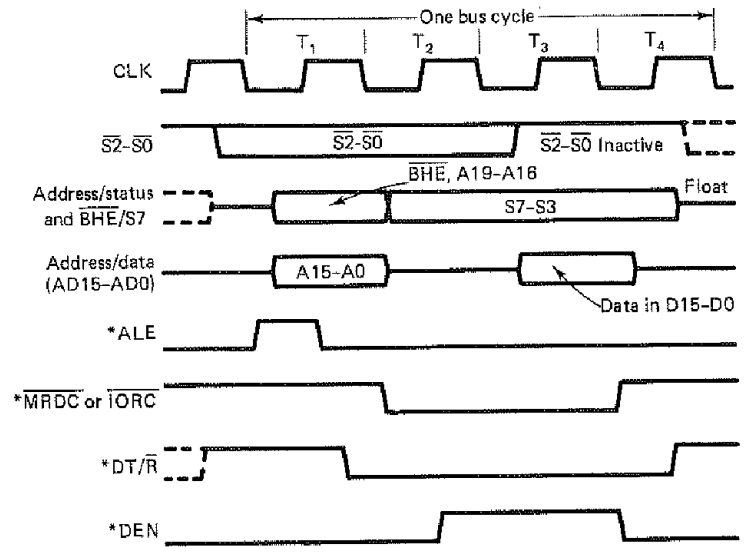
could execute completely, but

```
MOV     AX,NUMBER
```

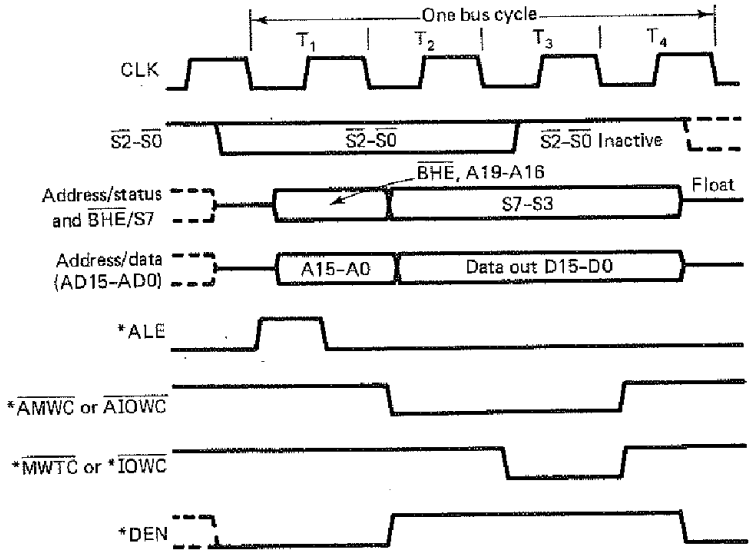
would only execute until it is necessary to bring in data from the location NUMBER.

The timing diagrams for input and output transfers on a maximum mode system are given in Fig. 8-15. The $\overline{S0}$, $\overline{S1}$, and $\overline{S2}$ bits are set just prior to the beginning of the bus cycle. Upon detecting a change from the passive $\overline{S0} = \overline{S1} = \overline{S2} = 1$ state, the 8288 bus controller will output a pulse on its ALE pin and apply the appropriate signal to its DT/$\overline{R}$ pin during T_1. In T_2, the 8288 will set $\overline{DEN} = 1$, thus enabling the transceivers, and, for an input, will activate either $\overline{MRDC}$ or $\overline{IORC}$. These signals will be maintained until T_4. For an output, the $\overline{AMWC}$ or $\overline{AIOWC}$ is activated from T_2 to T_4 and the $\overline{MWTC}$ or $\overline{IOWC}$ is activated from T_3 to T_4. The status bits $\overline{S0}$, $\overline{S1}$, and $\overline{S2}$ will remain active until T_3 and will become passive (all 1s) during T_3 and T_4. As with the minimum mode, if the READY input is not activated before the beginning of T_3, wait states will be inserted between T_3 and T_4.

Interrupt acknowledgment signals are the same as in the minimum mode case except that a 0 is applied to the $\overline{LOCK}$ pin from T_2 of the first bus cycle to T_2 of the second bus cycle. Bus requests and grants are handled differently, however, and the timing on an $\overline{RQ}/\overline{GT}$ pin is shown in Fig. 8-16. A request/grant/release is accomplished by a sequence of three pulses. The $\overline{RQ}/\overline{GT}$ pins are examined at the rising edge of each clock pulse and if a request is detected (and the necessary conditions discussed previously are met), the processor will apply a grant pulse to the $\overline{RQ}/\overline{GT}$ immediately following the next T_4 or T_I state. When the requesting master receives this pulse it seizes control of the bus. This master may control the bus for only one bus cycle or for several bus cycles. When it is ready to relinquish the bus it will send the processor the release pulse over the same line that it made its request. As noted before, $\overline{RQ}/\overline{GT0}$ and $\overline{RQ}/\overline{GT1}$ are the same except that $\overline{RQ}/\overline{GT0}$ has higher priority.



(a) Input



*8288 bus controller outputs.

(b) Output

Note: For an 8088, $\overline{BHE}$ is not present and data are transferred only over AD7–AD0.

Figure 8-15 Timing diagrams for a maximum mode system.

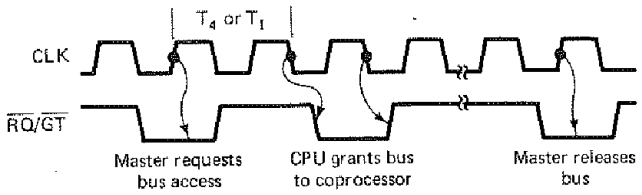


Figure 8-16. Timing for maximum mode bus requests and grants. (Reprinted by permission of Intel Corporation. Copyright 1979.)

8-3 INTERRUPT PRIORITY MANAGEMENT

The interrupt priority management logic indicated in Fig. 8-1 can be implemented in several ways. It does not need to be present in systems which use software priority management or simple daisy chaining, but more complex systems may require the efficiency gained by including hardware for managing the I/O interrupts. Many manufacturers have made priority management devices available and Intel is no exception. Although such a device made by one manufacturer could be used with processors made by other manufacturers, generally there are fewer compatibility problems if the CPU and interrupt priority device are produced by the same company. Therefore, we will be concerned with the Intel 8259A programmable interrupt controller (PIC), which has been specifically designed to work with the 8086/8088 as well as other members of the Intel microprocessor family.

The 8259A has been designed so that it can operate alone or in concert with other 8259As. In order to limit the initial discussion, an interrupt system involving a single 8259A device is considered first; then the discussion is extended to systems that can include as many as nine 8259As.

8-3-1 Interrupt System Based on a Single 8259A

The 8259A is contained in a 28-pin dual-in-line package that requires only a +5-V supply voltage. Its organization is shown in Fig. 8-17 along with its connections to a maximum mode system. Its pins (other than the supply voltage and ground pins) are defined as follows:

D7-D0—For communicating with the CPU over the data bus. On a few systems bus drivers may be needed, but on other systems direct connections can be used.

INT—To send interrupt request signals to the CPU.

$\overline{\text{INTA}}$—To receive interrupt acknowledge signals from the CPU. The 8259A assumes that an acknowledgment consists of two negative pulses, thus making it compatible with 8086/8088 systems.

$\overline{\text{RD}}$—To signal the 8259A that it is to place the contents of the IMR, ISR, or IRR register or a priority level on the data bus. Which of these possibilities is placed on the bus depends on the state of the 8259A and is discussed below.

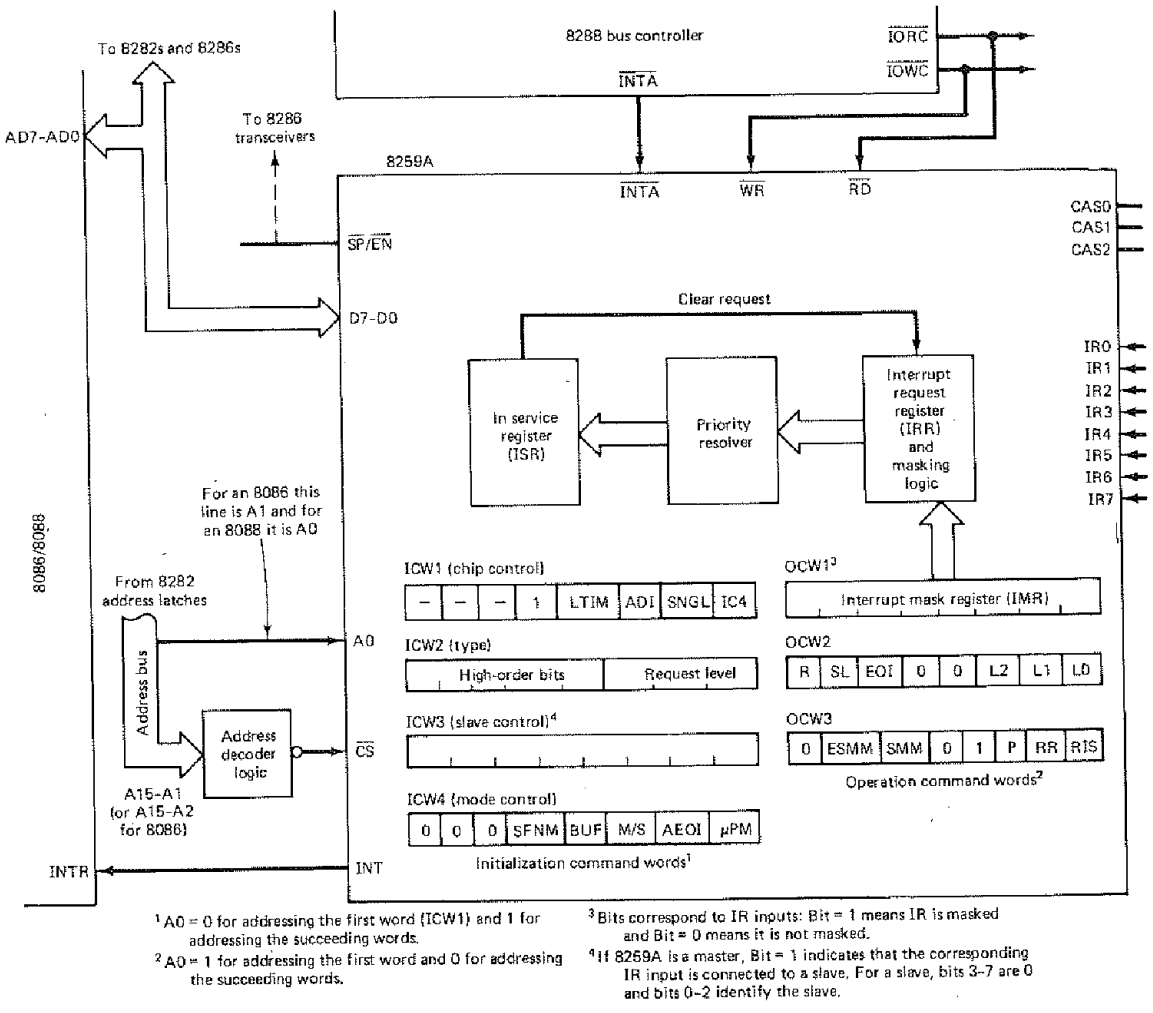


Figure 8-17 Organization of the 8259A programmable interrupt controller.

$\overline{\text{WR}}$—To signal the 8259A that it is to accept data from the data bus and use the data to set the bits in the command words. How the received data are distributed is discussed later.

$\overline{\text{CS}}$—For indicating that the 8259A is being accessed. This pin is connected to the address bus through the decoder logic that compares the high-order bits of the address of the 8259A with the address currently on the address bus. Input to this pin can be combined with $\overline{\text{S2}}$ to give the ready signal.

A0—For indicating which port of the 8259A is being accessed. Two addresses must be reserved in the I/O address space for each 8259A in the system.

IR7–IR0—For receiving interrupt requests from I/O interfaces or other 8259As referred to as slaves.

CAS2–CAS0—To identify a particular slave device. Their function is considered in Sec. 8-3-2.

$\overline{\text{SP}}/\overline{\text{EN}}$—For one of two purposes; either as an input to determine whether the 8259A is to be a master ($\overline{\text{SP}}/\overline{\text{EN}} = 1$) or as a slave ($\overline{\text{SP}}/\overline{\text{EN}} = 0$), or as an output to disable the data bus transceivers when data are being transferred from the 8259A to the CPU. Whether the $\overline{\text{SP}}/\overline{\text{EN}}$ pin is used as an input or output depends on the buffer mode discussed below.

For an 8088 the two addresses associated with an 8259A are normally consecutive, and the A0 line is connected to the A0 pin, but because there are only eight data pins on the 8259A and the 8086 always inputs the interrupt pointer from the lower 8 bits of its 16-bit data bus, all data transfers to and from the 8259A must be made over the lower byte of the bus. The easiest way to guarantee that all transfers will use the lower half of the bus is to connect the A1 line to A0 and use two consecutive even addresses, with the first being divisible by 4. However, to simplify the following discussion, the second address will be referred to as the odd address for both cases.

The control portion of the 8259A contains several programmable bits that can be viewed as being contained in seven 8-bit registers. These registers are divided into two groups, with one group containing the initialization command words (ICWs) and the other group containing the operation command words (OCWs). The *initialization command words* are normally set by an initialization routine when the computer system is first brought up and remain constant throughout its operation. By contrast, the *operation command words* are used to dynamically control the processing of interrupts.

The IRR (and its associated masking logic), priority resolver, and ISR are for receiving and controlling the interrupts that arrive at the IR7–IR0 pins. The IRR latches the incoming requests and, in conjunction with the priority resolver, allows unmasked requests with sufficient priority to put a 1 on the INT pin. The priority resolver logic determines the priorities of the requests in the IRR and the ISR is for holding the requests currently being processed.

After a bit in the IRR is set to 1 it is compared with the corresponding mask bit in the IMR. If the mask bit is 0, the request is passed on to the priority resolver, but if it is 1, the request is blocked. When an interrupt request is input to the priority resolver its priority is examined and, if according to the current state of the priority resolver the interrupt is to be sent to the CPU, the INT line is activated.

Assuming that the IF flag in the CPU is 1, the CPU will enter its interrupt sequence at the completion of the current instruction and return two negative pulses over the $\overline{\text{INTA}}$ line. Upon the arrival of the first pulse, the IRR latches are disabled so that the IRR will ignore further signals on the IR7–IR0 lines. This state is maintained until the end of the second $\overline{\text{INTA}}$ pulse. Also, the first $\overline{\text{INTA}}$ pulse will cause the appropriate ISR bit to be set and the corresponding IRR bit to be cleared. The second $\overline{\text{INTA}}$ pulse causes the current contents of ICW2 to be placed

on D7–D0, and the CPU uses this byte as the interrupt type. If the automatic end of interrupt (AEOI) bit in ICW4 is 1, at the end of the second $\overline{\text{INTA}}$ pulse the ISR bit that was set by the first $\overline{\text{INTA}}$ pulse is cleared; otherwise, the ISR bit is not cleared until the proper end of interrupt (EOI) command is sent to OCW2.

As indicated above, the initialization command words are normally filled by an initializing routine when the system is turned on and contain the control bits that are held constant throughout the system's operation. The 8259A has an even address (A0 = 0) and an odd address (A0 = 1) associated with it and the initialization command words must be filled consecutively by using the even address for ICW1 and the odd address for the remaining ICWs.

The definitions of the bits in ICW1 are:

Bits 7–5—Not used in an 8086/8088 system, only in an 8080 or 8085 system.

Bit 4—Always set to 1. It directs the received byte to ICW1 as opposed to OCW2 or OCW3, which also use the even address (A0 = 0).

Bit 3 (LTIM)—Determines whether the edge-triggered mode (LTIM = 0) or the level-triggered mode (LTIM = 1) is to be used. The edge-triggered mode causes the IRR bit to be cleared when the corresponding ISR bit is set.

Bit 2 (ADI)—Not used in an 8086/8088 system, only in an 8080 or 8085 system.

Bit 1 (SNGL)—Indicates whether or not the 8259A is cascaded with other 8259As. SNGL = 1 when only one 8259A is in the interrupt system.

Bit 0 (IC4)—Is set to 1 if an ICW4 is to be output to during the initialization sequence. For an 8086/8088 system this bit must always be set to 1 because bit 0 in ICW4 must be set to 1.

Bits 7–3 of ICW2 are filled from bits 7–3 of the second byte output by the CPU during the initialization of the 8259A, and bits 2–0 are set according to the level of the interrupt request, e.g., a request on IR6 would cause them to be set to 110. ICW3 is significant only in systems including more than one 8259A and is output to only if SNGL = 0. This case is discussed in Sec. 8-3-2. ICW4 is output to only if IC4 (ICW1) is set to 1; otherwise, the contents of ICW4 is cleared. The bits in ICW4 are defined as follows:

Bits 7–5—Always set to 0.

Bit 4 (SFNM)—If set to 1, the special fully nested mode is used. This mode is utilized in systems having more than one 8259A and is discussed below.

Bit 3 (BUF)—BUF = 1 indicates that the $\overline{\text{SP}}/\overline{\text{EN}}$ is to be used as an output to disable the system's 8286 transceivers while the CPU inputs data from the 8259A. If no transceivers are present, BUF should be set to 0 and, in systems involving only one 8259A, a 1 should be applied to the $\overline{\text{SP}}/\overline{\text{EN}}$ pin.

Bit 2 (M/S)—This bit is ignored if BUF = 0. For a system having only one 8259A, this bit should be 1; otherwise, it should be 1 for the master and 0 for the slaves.

Bit 1 (AEOI)—If AEOI = 1, then the ISR bit that caused the interrupt is cleared at the end of the second $\overline{\text{INTA}}$ pulse.

Bit 0 (μPM)—μPM = 1 indicates the 8259A is in an 8086/8088 system. This bit being 0 implies an 8080 or 8085 system.

A typical program sequence for setting the contents of the ICWs, which assumes that the even address of the 8259A is 0080, is:

```
MOV   AL,13H
OUT   80H,AL
MOV   AL,18H
OUT   81H,AL
MOV   AL,0DH
OUT   81H,AL
```

The first two instructions cause the requests to be edge triggered, denote that only one 8259A is used, and inform the 8259A that an ICW4 will be output. The next two instructions cause the 5 most significant bits of the interrupt type to be set to 00011. ICW3 is not output to because SNGL = 1; therefore, the last two instructions set ICW4 to 0D, which informs the 8259A that the special fully nested mode is not to be used, the $\overline{SP}/\overline{EN}$ is used to disable transceivers, the 8259A is a master, EOI commands must be used to clear the ISR bit, and the 8259A is part of an 8086/8088 system.

There are three OCWs. The command word OCW1 is used for masking interrupt requests; when the mask bit corresponding to an interrupt request is 1, then the request is blocked. OCW2 and OCW3 are for controlling the mode of the 8259A and receiving EOI commands. A byte is output to OCW1 by using the odd address associated with the 8259A and bytes are output to OCW2 and OCW3 by using the even addresses. OCW2 is distinguished from OCW3 by the contents of bit 3 of the data byte. If bit 3 is 0, the byte is put in OCW2, and if it is 1, it is put in OCW3. Both OCW2 and OCW3 are distinguished from ICW1, which also uses the even address, by the contents of bit 4 of the data. If bit 4 is 0, then the byte is put on OCW2 or OCW3 according to bit 3. There is no ambiguity in ICW2, ICW3, ICW4, and OCW1 all using the odd address because the initialization words must always follow ICW1 as dictated by the initialization sequence, and an output to OCW1 cannot occur in the middle of this sequence.

Referring back to Fig. 8-17, bits L2–L0 of OCW2 are for designating an IR level, bit 5 is for giving EOI commands, and bits 6 and 7 are for controlling the IR levels. Recall that when the AEOI bit in ICW4 is 1, the ISR bit, which is set by the interrupt request, is reset automatically at the end of the second $\overline{INTA}$ pulse, but if AEOI = 0, then the ISR bit must be explicitly cleared by an EOI command, which consists of sending an OCW2 with bit 5 equal to 1. When an EOI command is given the meanings of the four possible combinations of bit 7, the R (rotate) bit, and bit 6, the SL (set level) bit, are:

R	SL	
0	0	Nonspecific, normal priority mode
0	1	Specifically clears the ISR bit indicated by L2–L0
1	0	Rotate priority so that a device after being serviced has the lowest priority
1	1	Rotate priority until position specified by L2–L0 is lowest

The bits in OCW2 are only temporarily retained by the 8259A until the actions specified by them are carried out. This statement is particularly important with regard to the EOI bit. Let us now examine these possibilities in greater detail beginning with the normal priority mode (00).

Ordinarily, a request on IR0 has the highest priority, one on IR1 has the next highest priority, and so on. When the first $\overline{\text{INTA}}$ pulse arrives the priority resolver allows only the unmasked request having the highest priority to set its ISR bit. Because the 3 least significant bits of ICW2, where ICW2 indicates the interrupt type and determines the interrupt pointer address, are determined by which ISR bit is set, the address of the interrupt routine depends on which ISR bit is set. Therefore, the interrupt routine associated with the device connected to the highest-priority IR pin is begun first and the other requests must wait until further interrupts are allowed.

Under the normal priority mode, if ISRn is set, the priority resolver will not recognize any requests on IR7 through IR(n+1), but will recognize unmasked requests on IR(n−1) through IR0. Consequently, if the IF flag in the CPU has been reset to 1, requests having higher priority than the one being processed may cause the current interrupt routine to be interrupted while those having lower priorities are kept waiting. The lower-priority requests are processed according to their priorities as the higher-priority ISR bits are cleared. If AEOI = 1, the ISR bit corresponding to an interrupt is automatically cleared at the end of the second $\overline{\text{INTA}}$ pulse. When AEOI = 0 the ISR must be cleared by the interrupt routine by setting bit 5 of OCW2.

As an example of the normal priority mode, suppose that initially AEOI = 0 and all ISR and IMR bits are clear. Also suppose that, as shown in Fig. 8-18, requests occur simultaneously on IR2 and IR4, then a request arrives at IR1, and last a request arrives at IR3, and that these are the only requests. First, ISR2 will be set and the interrupt routine associated with IR2 will start executing. After this routine resets the IF flag to 1 and when the IR1 request is made, ISR1 will be set and the IR1 routine will be executed in its entirety. While it is executing it should reset the IF bit to 1 and send the necessary command to clear ISR1. Upon the return to the IR2 routine, ISR2 is cleared. Then ISR4 is set and its routine is begun. While this routine is executing IR3 is made. It is acknowledged as soon as IF is reset to 1, and ISR3 is set. Then the IR3 routine is initiated. Before the IR3 routine is completed it should clear ISR3 and set IF. The return is made to the IR4 routine, which should clear ISR4 before returning to the IR2 routine. The IR2 routine, having already cleared the ISR2 bit, would simply return to the interrupted program. (Note that if IF is not reset to 1 within the interrupt routine, further interrupts will not be processed until the routine is completed, i.e., the IRET instruction is encountered.)

Although a 1 sent to bit 5 of OCW2 normally causes the highest-priority ISR bit (i.e., the last ISR bit to be set) to be cleared, any ISR bit can be explicitly cleared by sending an OCW2 with the R, SL, and EOI bits set to 011 and putting

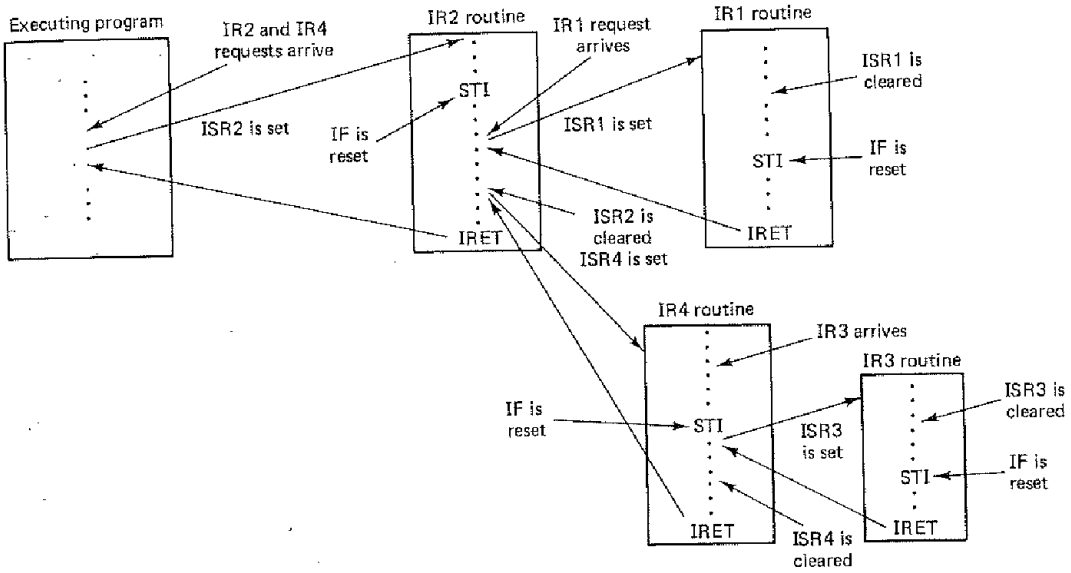


Figure 8-18 Actions taken in the normal operating mode when a typical sequence of interrupts occurs.

the number of the bit to be cleared in L2–L0. If

0	1	1	0	0	0	1	1

is sent to OCW2, then ISR3 will be cleared.

In addition to the normal priority mode discussed above, OCW2 can rotate the priority by assigning bottom priority to any one of the IR levels. In this case the other priorities will follow as if the normal ordering had been rotated. For instance, if the lowest priority is given to IR4, then the order of priorities will be:

IR5, IR6, IR7, IR0, IR1, IR2, IR3, IR4

(i.e., IR5 is rotated into the top-priority position). A rotation by one can be obtained by letting the combination for the R and SL bits be 10. If the R and SL bit combination is 11, then the IR level with the lowest priority is the one specified by L2–L0. If IR5 currently has top priority and

1	0	1	0	0	0	0	0

is sent to OCW2, then the new priority ordering would be

IR6, IR7, IR0, IR1, IR2, IR3, IR4, IR5

but if

1	1	1	0	0	0	1	0

is sent, the new ordering would be

IR3, IR4, IR5, IR6, IR7, IR0, IR1, IR2

The R and SL bits may also have significance when EOI = 0. In this case R = 1 and SL = 0 cause automatic rotations when AEOI = 1, and R = SL = 0 turns off this action so that automatic rotations do not take place. R = SL = 1 and EOI = 0 result in the lowest priority being designated by L2–L0 without an EOI command being sent. The remaining combination, R = 0 and SL = 1, causes no action.

In OCW3 the ESMM (enable special mask mode) and SMM (special mask mode) bits can be used to negate the priority modes discussed above. If a byte is sent to OCW3 in which both ESMM and SMM are set to 1, then unmasked interrupt requests are processed as they arrive (provided that the processor's IF bit is 1) and the priority order is ignored. By subsequently sending a byte to OCW3 in which ESMM = 1 and SMM = 0, a switch back to the priority ordering of interrupts can be made. If a byte with the ESMM bit equal to 0 is sent to OCW3, then the SMM bit will have no effect and the special mask mode will not change.

The P (polling) bit is used to place the 8259A in polling mode. This mode assumes that the CPU is not accepting interrupts (IF = 0) and it is necessary for the interrupt requests in the IRR to be polled. When the P bit is 1 the next $\overline{RD}$ signal would cause the appropriate bit in the ISR to be set just as if $\overline{INTA}$ signal had been received, and would return to the AL register in the CPU a byte of the form

I	—	—	—	—	W2	W1	W0

where I = 1 indicates that an interrupt is present and W2, W1, and W0 give the IR level of the highest-priority interrupt. For example, if P = 1, the priority ordering is

IR3, IR4, IR5, IR6, IR7, IR0, IR1, IR2

there are unmasked interrupts on IR4 and IR1, and the instruction

```
IN    AL,80H
```

(where 0080 is the even address of the 8259A) is executed, then

1	—	—	—	—	1	0	0

is input to the AL register.

When P = 0, the contents of IRR or ISR can be read into the AL register by setting RR = 1 and executing the instruction

```
IN    AL,80H
```

If at the time the IN instruction is executed RIS = 0, then IRR is input; otherwise, ISR is read. The contents of IMR can always be read by using the odd address of 8259A, e.g., for the address assignment indicated above,

```
IN    AL,81H
```

would input the contents of IMR to AL.

Because the OCW3 bits (except for ESMM) are used to specify whether or not the 8259A is in special mask mode and which information is to be put on the data bus during a read, these bits are retained until they are reset by the next output to OCW3. For example, if P = 0, RR = 1, and RIS = 0, any read from the even address of the 8259A before a new byte is sent to OCW3 will cause IRR to be read.

As a final note regarding the internal workings of the 8259A, let us examine what happens when noise interferes with a request. An IR input must remain high until after the trailing edge of the first $\overline{\text{INTA}}$ pulse. If it does not, then the 8259A will simulate a 1 on IR7. Therefore, provided that no device is connected to IR7, requests on this line would indicate improper drops in signals on the other request lines and the interrupt routine associated with IR7 would serve as a noise "cleanup" routine. If a device is connected to IR7, this noise detection scheme could still be used because an IR7 request from a device will set ISR7, while a noise-related request on IR7 will not affect ISR7. Thus, the IR7 routine could distinguish between the two events by reading the ISR and testing bit 7.

8-3-2 Interrupt System Based on Multiple 8259As

A multiple 8259A interrupt system is diagrammed in Fig. 8-19. In this figure data bus drivers are not shown, but they could be inserted as shown in Fig. 8-9. Although the $\overline{\text{SP}}/\overline{\text{EN}}$ pin on the master 8259A is connected to the data bus transceivers, a 0 is applied to the $\overline{\text{SP}}/\overline{\text{EN}}$ pins of the slaves. Only one slave is shown, but up to seven more slaves could be similarly connected into the system, permitting up to 64 distinct interrupt request lines. When designing the address decoder logic, each 8259A must be given its own address pair in the I/O address space. The drivers inserted in the CAS2–CAS0 lines may or may not be needed, depending on the proximity of the master to the slaves.

In a multiple 8259A system the slaves must be initialized as well as the master. The master would be initialized in the same way as indicated above except that SNGL would be set to 0 and ICW3 would need to be filled. A 1 would be put in each ICW3 bit for which the corresponding IR bit is connected to a slave and 0s would be put in the remaining bits. The SFNM bit may be set to 1 to activate the special fully nested mode. The SNGL bit should also be set to 0 when initializing the slaves. Thus, an ICW3 will be required for each slave, but for a slave ICW3 has a different meaning. For a slave, ICW3 has the form

0	0	0	0	0	ID2	ID1	ID0

where the 3 least significant bits provide the slave with an identification number. The identification number given to the slave should be the same as the number of the master request line to which its INT pin is connected.

When a slave puts a 1 on its INT pin this signal is sent to the appropriate IR pin on the master. Assuming that the IMR and priority resolver do not block this signal, it is sent to the CPU through the INT pin on the master. When the CPU returns the $\overline{\text{INTA}}$ signal the master will not only set the appropriate ISR bit and

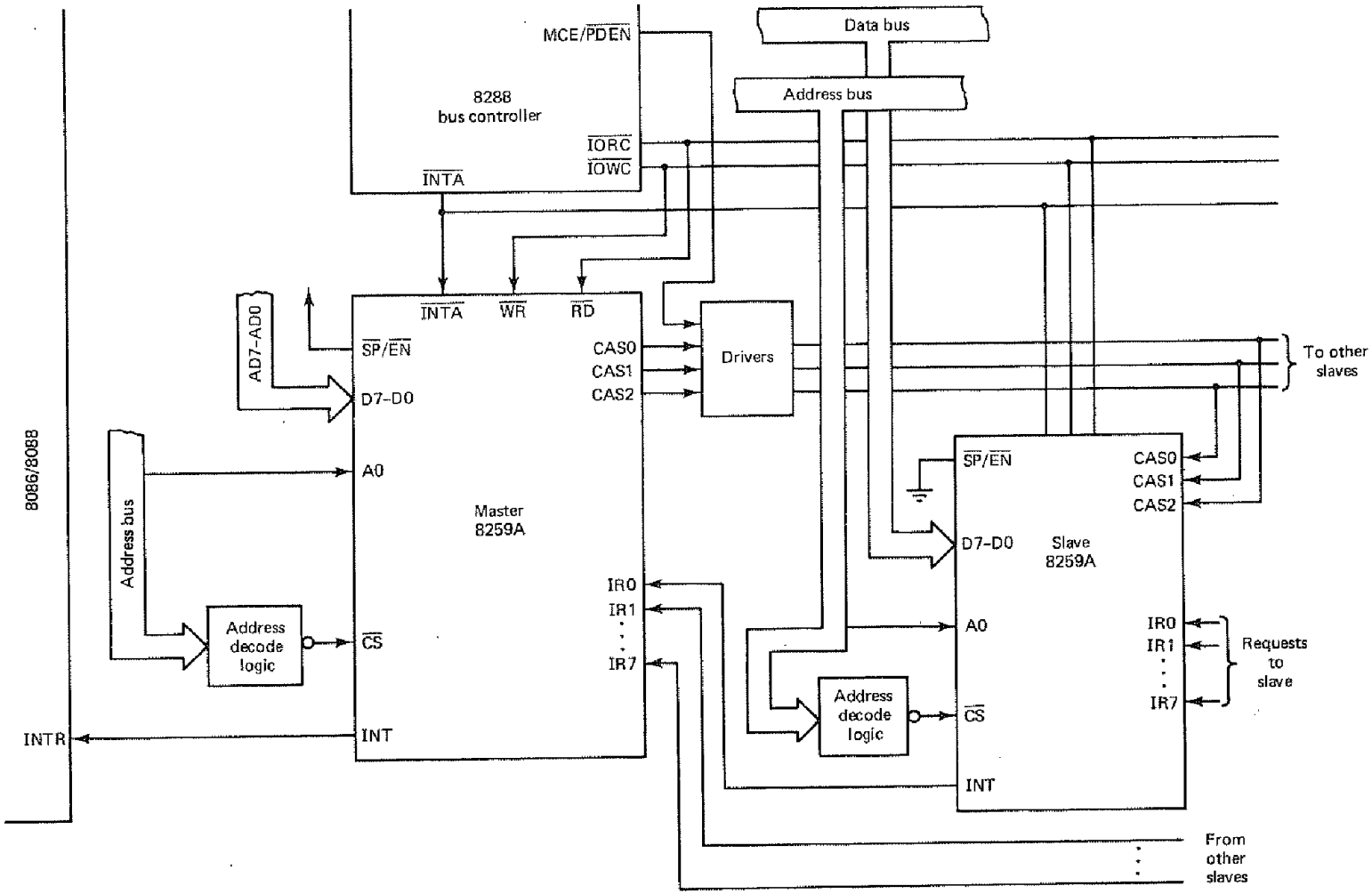
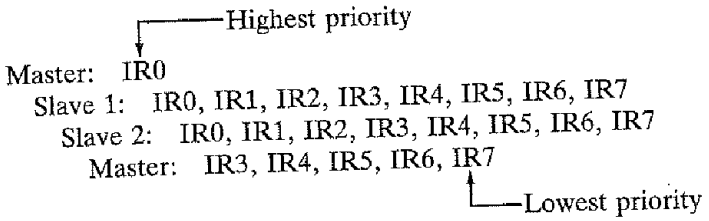


Figure 8-19 Multiple 8259A-based interrupt system.

clear the corresponding IRR bit, it will also check the corresponding bit in ICW3 to determine whether or not the interrupt came from a slave. If so, the master will place the number of the IR level on the CAS2–CAS0 lines; if not, it will put the contents of ICW2 on the data bus and no signals will be applied to the CAS2–CAS0 lines. The $\overline{\text{INTA}}$ signal is also received by all of the slaves, but only that slave whose ID matches the number sent to it by the master over the CAS2–CAS0 lines will accept the signal. In the selected slave the appropriate ISR bit will be set, the corresponding IRR bit will be cleared, and its ICW2 will be put on the data bus. Because ICW2 contains the interrupt type, it is important that unique combinations be put in the master and slave ICW2s during the initialization process. EOI commands are required for both the master and the slave if their AEOI bits are 0.

Except for the response to the $\overline{\text{INTA}}$ signal discussed above, the actions taken by all 8259A devices in the system are the same. Also, their modes are controlled and their registers are read in the same way. There is one exception, however. If the SFNM bit in the ICW4 of the master is initialized to 1, the master will enter the special fully nested mode. This mode is ordinarily used with the normal priority mode and AEOI = 0. In this case, the master will allow unmasked requests of sufficient priority to be passed on to the INT pin even if the corresponding ISR bit is already 1. This means that if a higher-priority request arrives at a slave while one or more of the slave's requests is being processed, the new request will be allowed to send its INT signal through the master. When using the special fully nested mode, two EOI commands may need to be sent by the interrupt routine. First, a nonspecific EOI command would be sent to the slave that caused the interrupt and then the slave's ISR would be tested. If and only if the ISR contains all 0s would a nonspecific EOI command be sent to the master. Therefore, assuming that slave 1 is connected to IR1 and slave 2 is connected to IR2 of a master, they are the only slaves, and the highest priority in all three 8259As is assigned to IR0, the fully nested order of priorities would be:



The masks in the master and slaves may, of course, be used to block out some of the requests.

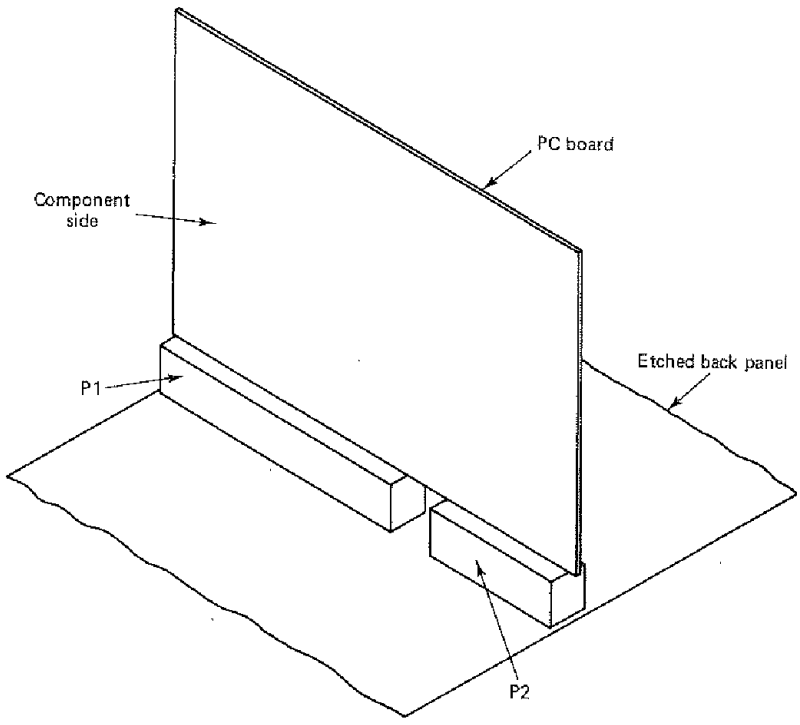
8-4 BUS STANDARDS

For reasons of simplicity, flexibility, and low cost, most microcomputers, including those involving multiprocessor configurations, are built around a primary system bus which connects all of the major components in the system. In order to obtain

a foundation while designing its products, a microcomputer manufacturer makes assumptions about the bus that is to be used to connect its devices together. Frequently, these assumptions become formalized and constitute what is referred to as a *bus standard*. If a line of products becomes popular, then its bus standard is widely accepted and other manufacturers may also design devices that satisfy the standard, particularly if the standard is adopted by an organization such as the Institute of Electrical and Electronics Engineers (IEEE). Because of the availability of a large number of compatible devices, a user can sometimes significantly reduce the system development time and cost by choosing a popular and well-designed bus. In this section we will briefly consider the Intel MULTIBUS.

The Intel MULTIBUS has gained wide industrial acceptance and several manufacturers offer MULTIBUS-compatible modules. This bus is designed to support both 8-bit and 16-bit devices and can be used in multiprocessor systems in which several processors can be masters. At any point in time, only two devices may communicate with each other over the bus, one being the master and the other the slave. The master/slave relationship is dynamic with bus allocation being accomplished through the bus allocation (i.e., request/grant) control signals. The MULTIBUS has been physically implemented on an etched backplane board which is connected to each module using two edge connectors, denoted P1 and P2, as shown in Fig. 8-20. The connector P1 consists of 86 pins which provide the major

Figure 8-20 Illustration of a module being plugged into a MULTIBUS.



bus signals, and P2 is an optional connector consisting of 60 auxiliary lines, primarily used for power failure detection and handling. (Although P1 and P2 are touched on below, they are not fully described and the reader should go to Reference 3 for the details.)

The P1 lines can be divided into the following groups according to their functions:

1. Address lines.
2. Data lines.
3. Command and handshaking lines.
4. Bus access control lines.
5. Utility lines.

The MULTIBUS has 20 address lines, labeled $\overline{\text{ADR0}}$ through $\overline{\text{ADR13}}$, where the numeric suffix represents the address bit in hexadecimal. The address lines are driven by the bus master to specify the memory location or I/O port being accessed. The MULTIBUS standard allows a 16-bit master or an 8088 to use all 20 lines to access memory and only the lower 12 address lines ($\overline{\text{ADRB}}$–$\overline{\text{ADR0}}$) to select an I/O port, despite the fact that both the 8086 and 8088 permit 16-bit I/O addresses. For 8-bit masters, memory addressing and I/O port selection use only 16 lines ($\overline{\text{ADRF}}$–$\overline{\text{ADR0}}$) and 8 lines ($\overline{\text{ADR7}}$–$\overline{\text{ADR0}}$), respectively.

In addition to the address lines, the bus provides three address control signals. They are the byte high enable ($\overline{\text{BHEN}}$), inhibit RAM ($\overline{\text{INH1}}$), and inhibit ROM ($\overline{\text{INH2}}$) signals. $\overline{\text{BHEN}}$ is used by 16-bit devices to indicate that data are to be transferred on the high byte of the data bus; this is required for word transfers. The MULTIBUS standard calls for all single bytes to be communicated over only the lower 8 bits of the bus; therefore, any 16-bit interface must include a swap byte buffer so that only the lower data lines are used for all byte transfers. (It should be pointed out that because an 8086 expects a byte to be put on the high-order byte of the bus when $\overline{\text{BHE}}$ is active, one may want to permit nonstandard MULTIBUS transfers between memory and an 8086.)

The two inhibit signals are provided for overlaying RAM, ROM, and auxiliary ROM in a common address space. For example, a bootstrap loader may be stored in an auxiliary ROM and a monitor in a ROM. Because the loader is needed only after a reset, $\overline{\text{INH1}}$ and $\overline{\text{INH2}}$ could both be activated while the loader is executing. Then, when the monitor is in control, $\overline{\text{INH2}}$ could be raised while $\overline{\text{INH1}}$ remains low. If control is passed to the user, $\overline{\text{INH1}}$ and $\overline{\text{INH2}}$ could both be deactivated, thus allowing the RAM to fill the entire memory space during normal operation.

There are 16 bidirectional data lines ($\overline{\text{DATF}}$–$\overline{\text{DAT0}}$), only eight of which are used in an 8-bit system. Data transfers on the MULTIBUS bus are accomplished by handshaking signals in a manner similar to that described in the preceding sections. The memory read ($\overline{\text{MRDC}}$), memory write ($\overline{\text{MWTC}}$), I/O read ($\overline{\text{IORC}}$), and I/O write ($\overline{\text{IOWC}}$) lines are defined to be the same as they were in the discussion of the 8288 bus controller. There is an acknowledge ($\overline{\text{XACK}}$) signal which serves the same purpose as the READY signal in the discussion of the bus control logic,

i.e., to verify the end of a transfer. In a general setting it may be received by any bus master. Because a master must wait to be notified of the completion of a transfer, the duration of a bus cycle varies depending on the speed of the bus master and the slave. This asynchronous nature enables the system to handle slow devices without penalizing fast devices.

When an 8086 or 8088 is configured in its maximum mode, the 8288 generates bus commands that are fully compatible with the electrical requirements of the MULTIBUS. An 8086 or 8088 in minimum mode can also communicate with a MULTIBUS, provided that external logic is included to generate the necessary command signals. An 8086 or 8088 system with more than one CPU must be configured in maximum mode. Also, in order to obtain the inverted signals specified by the MULTIBUS standard, 8283 latches and 8287 transceivers are used instead of 8282s and 8286s. The command and handshaking signal lines also include 8 interrupt request lines ($\overline{\text{INT7}}$–$\overline{\text{INT0}}$) and one interrupt acknowledge line ($\overline{\text{INTA}}$) to support priority management.

In order to support multiprocessor configurations, there are 6 lines designed for use by bus access logic. These bus access lines are typically connected to bus arbiters which ensure that only one bus master can have control of the MULTIBUS at a time. The bus arbiter is discussed in Chap. 11. The utility group consists of lines for ground, +5 V, −5 V, +12 V, −12 V, and timing.

The auxiliary bus which is connected to PC boards through the 60-pin P2 edge connector is optional. It is needed primarily to provide the system with the capability to detect and to handle power failures (although it also includes some seldom used control lines). When the ac voltage level drops below a certain level, the power supply generates a power failure signal. This signal causes the user-supplied power-failure-handling logic to generate an interrupt to the bus master, which then saves the current program's vital information. After the ac power is restored, the power-failure-handling logic signals the bus master to initiate a power-up sequence. During this sequence, the processor restores the information and resumes the execution of the suspended program. With a standby power source, normally provided by batteries, memory and other modules can remain active during power failures for reasonably long periods of time.

BIBLIOGRAPHY

1. *1982 Data Component Catalog* (Santa Clara, Calif.: Intel Corporation, 1982).
2. *The 8086 Family User's Manual* (Santa Clara, Calif.: Intel Corporation, 1979).
3. *Intel MULTIBUS Interfacing*, Application Note AP-28A (Santa Clara, Calif.: Intel Corporation, 1979).

EXERCISES

1. Use the logic diagram of the 8286 in Fig. 8-6 and truth tables to verify the statements regarding the data flow direction and enabling and disabling of the data flow.

2. Assume that the instruction

```
XCHG    BL,DATA
```

is already in the queue and is ready to execute, and give a timing diagram of the bus activity during its execution [similar to the ones in Figs. 8-12(a) and 8-15] for each of the following cases:

(a) Minimum mode 8086 with no wait states.
(b) Minimum mode 8088 with no wait states.
(c) Maximum mode 8086 with no wait states.
(d) Maximum mode 8086 if a ready signal two clock periods long is received in the middle of T_3. (Also include the timing diagram of the ready signal.)
(e) Maximum mode 8086 with no wait states, but assume that a bus request is detected during the input transfer and is dropped after only one bus cycle.

3. Consider a maximum mode 8086 system that is executing the instructions

```
MOV     AX,DATA (DATA has an even address)
CMP     AX,1
```

Suppose that an interrupt request arrives while the first instruction is executing and draw a timing diagram of the bus activity from the beginning of the first instruction until the interrupt type is received by the processor.

4. Assuming that the instruction is already in the queue, give the number of bus cycles needed to execute each of the following instructions:

```
(a) PUSH    AX
(b) CALL    NEAR PTR PROC_A
(c) CALL    FAR PTR PROC_B
(d) MOV     DATA,AX     (DATA has an even address)
(e) MOV     DATA,AX     (DATA has an odd address)
(f) OUT     60H,AX
```

5. What is the minimum number of bus cycles that can occur between the time an interrupt request is recognized and the first instruction in the interrupt routine is fetched?

6. Give an 8086 timing diagram that shows the activity on the clock line and all of the address/data lines while the instruction

```
ADD     AX,TAX
```

is being executed. First assume that TAX begins at an even address and then assume that it begins at an odd address.

7. Given that the instruction

```
JMP     NEAR PTR BEGIN
```

is already in the instruction queue and the branch is made to

```
BEGIN:  CMP     AL,0
```

describe the actions taken during each bus cycle from the beginning of the execution of the JMP instruction until the end of the execution of the CMP instruction. Answer the exercise for each of the following cases:

(a) An 8088 system.
(b) An 8086 system with BEGIN at an even address.
(c) An 8086 system with BEGIN at an odd address.

8. Write an initialization sequence for an 8259A, which is the only 8259A in an 8086 system and has an even address of 2130, that will cause:
   1. Requests to be level triggered.
   2. The interrupt type for an IR0 request to be 28.
   3. $\overline{SP}/\overline{EN}$ to output a disable signal to the data bus transceivers.
   4. The ISR bits to be automatically cleared at the end of the second $\overline{INTA}$ pulse.
   5. The IMR to be cleared.
9. Write an instruction sequence that will cause the priority of an 8259A, whose even address is 08A0, to be

   IR4, IR5, IR6, IR7, IR0, IR1, IR2, IR3

   Solve this exercise twice, once assuming that the highest priority is currently IR0 and once assuming that it is IR3.
10. Write a sequence of instructions that will transfer the contents of the IRR, ISR, and IMR in an 8259A, whose even address is 0500, to an array in memory that begins at REG_ARR.
11. Give an instruction sequence that will cause the requests on IR3, IR4, and IR6 of an 8259A, whose even address is 1208, to be blocked.
12. Suppose that IF = 0. Give a sequence of instructions that will poll the 8259A, whose even address is 1000, and branch to the *n*th offset in an array of eight offsets, where *n* is the number of the highest-priority interrupt request line that has received a request. If there are no requests, execution is to continue in sequence.
13. Given the following sequence of events, normal priority, and that EOI commands must be output, draw a diagram similar to the one in Fig. 8-18:
   1. Request on IR3.
   2. Request on IR2.
   3. Request on IR6.
   4. IF reset to 1.
   5. IF reset to 1.
   6. EOI to clear ISR2.
   7. EOI to clear ISR3.
   8. IF reset to 1.
   9. EOI to clear ISR6.
14. Consider a maximum mode 8086 system containing a master 8259A and one slave 8259A that is connected to IR1 of the master. The even addresses of the master and slave are 0580 and 0582, respectively. The interrupt type for IR0 of the master is to be 30 and for IR0 of the slave is to be 38, and all requests are to be edge triggered. EOI commands are to be used to clear the ISR bits. For both the master and slave the IMRs are to be cleared and $\overline{SP}/\overline{EN}$ is used as an input. Give a program sequence for initializing the interrupt system.
15. For the system described in Exercise 14, write a program sequence for clearing the appropriate ISR bits after an interrupt request is made by the slave. Remember that the slave must be checked for all 0s before ISR1 in the master is cleared.
16. Consider an interrupt system containing a master and three slaves, with the slaves being connected to IR2, IR3, and IR6 of the master. Assume that the IMR in the master is set to

01010000

and that the IMRs in the slaves are cleared. Normal priority is used in all 8259As except the slave connected to IR3, and its highest-priority request line is IR5. List the unmasked request lines in the order of their priorities, with the highest priority listed first. Repeat the solution for the case when the master's highest-priority line is IR3.