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

- Explain the interrupt structure of the Intel family of microprocessors.
- Explain the operation of software interrupt instructions INT, INTO, INT3, and BOUND.
- Explain how the interrupt enable flag bit (IF) modifies the interrupt structure.
- Describe the function of the trap interrupt flag bit (TF) and the operation of trap-generated tracing.

Chapter Objectives (cont.)
Upon completion of this chapter, you will be able to:

- Develop interrupt-service procedures that control lower-speed, external peripheral devices.
- Expand the interrupt structure of the microprocessor by using the 82S9A programmable interrupt controller and other techniques.
- Explain the purpose and operation of a real-time clock.

12–1 BASIC INTERRUPT PROCESSING

- This section discusses the function of an interrupt in a microprocessor-based system.
- Structure and features of interrupts available to Intel microprocessors.

The Purpose of Interrupts

- Interrupts are useful when interfacing I/O devices at relatively low data transfer rates, such as keyboard inputs, as discussed in Chapter 11.
- Interrupt processing allows the processor to execute other software while the keyboard operator is thinking about what to type next.
- When a key is pressed, the keyboard encoder debounces the switch and puts out one pulse that interrupts the microprocessor.

Introduction

- In this chapter, the coverage of basic I/O and programmable peripheral interfaces is expanded by examining a technique called interrupt-processed I/O.
- An interrupt is a hardware-initiated procedure that interrupts whatever program is currently executing.
- This chapter provides examples and a detailed explanation of the interrupt structure of the entire Intel family of microprocessors.
Interrupts

- Intel processors include two hardware pins (INTR and NMI) that request interrupts...
- And one hardware pin (INTA) to acknowledge the interrupt requested through INTR.
- The processor also has software interrupts INT, INTO, INT 3, and BOUND.
- Flag bits IF (interrupt flag) and TF (trap flag), are also used with the interrupt structure and special return instruction IRET
  - IRETD in the 80386, 80486, or Pentium

Interrupt Vectors

- Interrupt vectors and the vector table are crucial to an understanding of hardware and software interrupts.
- The interrupt vector table is located in the first 1024 bytes of memory at addresses 000000H–0003FFH.
  - contains 256 different four-byte interrupt vectors
- An interrupt vector contains the address (segment and offset) of the interrupt service procedure.

Intel Dedicated Interrupts

- Type 0
  The divide error whenever the result from a division overflows or an attempt is made to divide by zero.
- Type 1
  Single-step or trap occurs after execution of each instruction if the trap (TF) flag bit is set.
  - upon accepting this interrupt, TF bit is cleared so the interrupt service procedure executes at full speed
• **Type 4**
  - **Overflow** is a special vector used with the INTO instruction. The INTO instruction interrupts the program if an overflow condition exists.
    - as reflected by the overflow flag (OF)

• **Type 5**
  - The `BOUND` instruction compares a register with boundaries stored in the memory.
  - If the contents of the register are greater than or equal to the first word in memory and less than or equal to the second word, no interrupt occurs because the contents of the register are within bounds.
    - if the contents of the register are out of bounds, a type 5 interrupt ensues

• **Type 6**
  - An **invalid opcode** interrupt occurs when an undefined opcode is encountered in a program.

• **Type 7**
  - The **coprocessor not available** interrupt occurs when a coprocessor is not found, as dictated by the machine status word (MSW or CR0) coprocessor control bits.
    - if an ESC or WAIT instruction executes and no coprocessor is found, a type 7 exception or interrupt occurs

• **Type 8**
  - A **double fault** interrupt is activated when two separate interrupts occur during the same instruction.

• **Type 9**
  - The **coprocessor segment overrun** occurs if the ESC instruction (coprocessor opcode) memory operand extends beyond offset address FFFFH in real mode.

• **Type 10**
  - An **invalid task state segment** interrupt occurs in the protected mode if the TSS is invalid because the segment limit field is not 002BH or higher.
    - usually because the TSS is not initialized

• **Type 11**
  - The **segment not present** interrupt occurs when the protected mode P bit (P = 0) in a descriptor indicates that the segment is not present or not valid.

• **Type 12**
  - A **stack segment overrun** occurs if the stack segment is not present (P = 0) in the protected mode or if the limit of the stack segment is exceeded.
• **Type 13**
  The **general protection fault** occurs for most protection violations in 80286–Core2 in protected mode system.

  These errors occur in Windows as general protection faults.

  A list of these protection violations follows.

• **Type 14**
  **Page fault** interrupts occur for any page fault memory or code access in 80386, 80486, and Pentium–Core2 processors.

• **Type 16**
  **Coprocessor error** takes effect when a coprocessor error (ERROR = 0) occurs for ESCape or WAIT instructions for 80386, 80486, and Pentium–Core2 only.

• **Type 13** protection violations (cont.)
  – (a) Descriptor table limit exceeded
  – (b) Privilege rules violated
  – (c) Invalid descriptor segment type loaded
  – (d) Write to code segment that is protected
  – (e) Read from execute-only code segment
  – (f) Write to read-only data segment
  – (g) Segment limit exceeded
  – (h) CPL = IOPL when executing CTS, HLT, LGDT, LLDT, LMSW, or LTR
  – (i) CPL > IOPL when executing CLI, IN, INS, LOCK, OUT, OUTS, and STI

• **Type 17**
  **Alignment checks** indicate word and doubleword data are addressed at an odd memory location (or incorrect location, in the case of a doubleword).
  – interrupt is active in 80486 and Pentium–Core2

• **Type 18**
  A **machine check** activates a system memory management mode interrupt in Pentium–Core2.

**Interrupt Instructions: BOUND, INTO, INT, INT 3, and IRET**

• Five software interrupt instructions are available to the microprocessor:
  – INT and INT 3 are very similar.
  – BOUND and INTO are conditional.
  – IRET is a special interrupt return instruction.

• **BOUND** has two operands, and compares a register with two words of memory data.

• INTO checks or tests the overflow flag (O).
  – If O = 1, INTO calls the procedure whose address is stored in interrupt vector type 4
  – If O = 0, INTO performs no operation and the next sequential program instruction executes

• The **INT n** instruction calls the interrupt service procedure at the address represented in vector number n.
• INT 3 instruction is often used as a breakpoint-interrupt because it is easy to insert a one-byte instruction into a program.
  – breakpoints are often used to debug software
• The IRET instruction is a special return instruction used to return for both software and hardware interrupts.
  – much like a far RET, it retrieves the return address from the stack

### Operation of a Real Mode Interrupt

• When the processor completes executing the current instruction, it determines whether an interrupt is active by checking:
  – (1) instruction executions
  – (2) single-step
  – (3) NMI
  – (4) coprocessor segment overrun
  – (5) INTR
  – (6) INT instructions in the order presented

• If one or more are present:
  – 1. Flag register contents are pushed on the stack
  – 2. Interrupt (IF) & trap (TF) flags clear, disabling the INTR pin and trap or single-step feature
  – 3. Contents of the code segment register (CS) are pushed onto the stack
  – 4. Contents of the instruction pointer (IP) are pushed onto the stack
  – 5. Interrupt vector contents are fetched and placed into IP and CS so the next instruction executes at the interrupt service procedure addressed by the vector

### Operation of a Protected Mode Interrupt

• In protected mode, interrupts have the same assignments as real mode.
  – the interrupt vector table is different
• In place of interrupt vectors, protected mode uses a set of 256 interrupt descriptors stored in an interrupt descriptor table (IDT).
  – the table is 256 x 8 (2K) bytes long
  – each descriptor contains eight bytes

• The interrupt descriptor table is located at any memory location in the system by the interrupt descriptor table address register (IDTR).
• Each IDT entry contains the address of the interrupt service procedure
  – in the form of a segment selector and a 32-bit offset address
  – also contains the P bit (present) and DPL bits to describe the privilege level of the interrupt
• Fig 12–3 shows interrupt descriptor contents.
Interrupt Flag Bits
- The interrupt flag (IF) and the trap flag (TF) are both cleared after the contents of the flag register are stacked during an interrupt.
- The contents of the flag register and the location of IF and TF are shown here:
  - when IF is set, it allows the INTR pin to cause an interrupt.
  - when IF is cleared, it prevents the INTR pin from causing an interrupt.

Trace Procedure
- Assuming TRON is accessed by an INT 40H instruction and TROFF is by an INT 41H instruction, Example 12–3 traces through a program immediately following the INT 40H instruction.
- The interrupt service procedure illustrated in Example 12–3 responds to interrupt type 1 or a trap interrupt.

Storing an Interrupt Vector in the Vector Table
- To install an interrupt vector—sometimes called a hook—the assembler must address absolute memory.
- Example 12–4 shows how a new vector is added to the interrupt vector table using the assembler and a DOS function call.
- Function AX = 3100H for INT 21H, installs the NEW40 procedure until the PC is shut off.

12–2 HARDWARE INTERRUPTS
- The two processor hardware interrupt inputs:
  - non-maskable interrupt (NMI)
  - interrupt request (INTR)
- When NMI input is activated, a type 2 interrupt occurs:
  - because NMI is internally decoded
- The INTR input must be externally decoded to select a vector.
- Any interrupt vector can be chosen for the INTR pin, but we usually use an interrupt type number between 20H and FFH.
- Intel has reserved interrupts 00H - 1FH for internal and future expansion.
- INTA is also an interrupt pin on the processor:
  - it is an output used in response to INTR input to apply a vector type number to the data bus connections D7 – D0
- Figure 12–5 shows the three user interrupt connections on the microprocessor.
Figure 12-5 The interrupt pins on all versions of the Intel microprocessor.

- **NMI**
- **INTR**
- **INTA**

Interrupt inp  Interrupt out

- **The non-maskable interrupt** (NMI) is an edge-triggered input that requests an interrupt on the positive edge (0-to-1 transition).
  - After a positive edge, the NMI pin must remain logic 1 until recognized by the microprocessor.
  - Before the positive edge is recognized, NMI pin must be logic 0 for at least two clocking periods.
- The NMI input is often used for parity errors and other major faults, such as power failures.
  - Power failures are easily detected by monitoring the AC power line and causing an NMI interrupt whenever AC power drops out.

Figure 12-6 A power failure detection circuit.

- Figure 12-6 shows a power failure detection circuit that provides logic 1 to the NMI input whenever AC power is interrupted.
- In this circuit, an optical isolator provides isolation from the AC power line.
- The interrupt service procedure stores the contents of all internal registers and other data into a battery-backed-up memory.
- This assumes the PC power supply has a large enough filter capacitor to provide energy for at least 75 ms after the AC power ceases.

Figure 12-7 A battery-backed-up memory system using a NiCad, lithium, or gel cell.

- Figure 12-7 shows a circuit that supplies power to a memory after the DC power fails.
  - Diodes are used to switch supply voltages from the DC power supply to the battery.
  - When DC power fails, the battery provides a reduced voltage to the $V_{CC}$ connection on the memory device.
  - Most memory devices will retain data with $V_{CC}$ voltages as low as 1.5 V, so the battery voltage does not need to be +5.0 V.
**INTR and INTA**

- The interrupt request input (INTR) is level-sensitive, which means that it must be held at a logic 1 level until it is recognized.
  - INTR is set by an external event and cleared inside the interrupt service procedure.
  - INTR is automatically disabled once accepted.
    - re-enabled by IRET at the end of the interrupt service procedure.
  - 80386–Core2 use IRETD in protected mode.
    - in 64-bit protected mode, IRETQ is used.

- The processor responds to INTR by pulsing INTA output in anticipation of receiving an interrupt vector type number on data bus connections $D_7–D_0$.
- Fig 12–8 shows the timing diagram for the INTR and INTA pins of the microprocessor.
- Two INTA pulses generated by the system insert the vector type number on the data bus.
- Fig12–9 shows a circuit to apply interrupt vector type number FFH to the data bus in response to an INTR.

**Using a Three-State Buffer for INTA**

- Fig 12–10 shows how interrupt vector type number 80H is applied to the data bus ($D_0–D_7$) in response to an INTR.
- In response to INTR, the processor outputs the INTA to enable a 74ALS244 three-state octal buffer.
- The octal buffer applies the interrupt vector type number to the data bus in response.
- The vector type number is easily changed with DIP switches shown in this illustration.
Making INTR Input Edge-Triggered

- INTR input can be converted to an edge-triggered input by using a D-type flip-flop, as illustrated in Figure 12–11.
- Clock input becomes an edge-triggered interrupt request input, and the clear input is used to clear the request when the INTA signal is output by the microprocessor.
- The RESET signal initially clears the flip-flop so that no interrupt is requested when the system is first powered.

The 82C55 Keyboard Interrupt

- Fig 12–12 shows interconnection of an 82C55 with the microprocessor and keyboard.
- Every time a key is typed, 82C55 requests a type 40H interrupt through the INTR pin.
- Example 12–5 illustrates the interrupt service procedure for the keyboard.
- The procedure is short because the processor already knows that keyboard data are available when the procedure is called.

12–3 EXPANDING THE INTERRUPT STRUCTURE

- This section covers three common methods of expanding the interrupt structure of the processor.
- By method 1, it is possible to expand the INTR input so it accepts seven interrupt inputs.
- Also explained is how to “daisy-chain” interrupts by software polling.

Using the 74ALS244 to Expand Interrupts

- The modification shown in Fig 12–13 allows the circuit of Fig 12–10 to accommodate up to seven additional interrupt inputs.
- The only hardware change is the addition of an eight-input NAND gate, which provides the INTR signal to the microprocessor when any of the IR inputs becomes active.
Operation

- If any of the IR inputs becomes logic 0, the output of the NAND gate goes to logic 1 and requests an interrupt through the INTR input.
- The interrupt vector that is fetched during the pulse depends on which interrupt request line becomes active.
  - Table 12–1 shows the interrupt vectors used by a single interrupt request input
- If two or more interrupt requests are active, a new interrupt vector is generated.

Daisy-Chained Interrupt

- Expansion by a daisy-chained interrupt is in many ways better than using the 74ALS244.
  - because it requires only one interrupt vector
- Fig 12–14 shows a two 82C55 peripheral interfaces with their four INTR outputs daisy-chained and connected to the single INTR input of the processor.
- If any interrupt output becomes logic 1, so does INTR input, causing an interrupt.

8259A PROGRAMMABLE INTERRUPT CONTROLLER

- 8259A (PIC) adds eight vectored priority encoded interrupts to the microprocessor.
- Expandable, without additional hardware, to accept up to 64 interrupt requests.
  - requires a master 8259A & eight 8259A slaves
- A pair of these controllers still resides and is programmed as explained here in the latest chip sets from Intel and other manufacturers.

General Description of the 8259A

- 8259A is easy to connect to the microprocessor
- all of its pins are direct connections except the CS pin, which must be decoded, and the WR pin, which must have an I/O bank write pulse

12–4 8259A PROGRAMMABLE INTERRUPT CONTROLLER

- 8259A (PIC) adds eight vectored priority encoded interrupts to the microprocessor.
- Expandable, without additional hardware, to accept up to 64 interrupt requests.
  - requires a master 8259A & eight 8259A slaves
- A pair of these controllers still resides and is programmed as explained here in the latest chip sets from Intel and other manufacturers.

12–4 8259A PROGRAMMABLE INTERRUPT CONTROLLER

- 8259A (PIC) adds eight vectored priority encoded interrupts to the microprocessor.
- Expandable, without additional hardware, to accept up to 64 interrupt requests.
  - requires a master 8259A & eight 8259A slaves
- A pair of these controllers still resides and is programmed as explained here in the latest chip sets from Intel and other manufacturers.

General Description of the 8259A

- 8259A is easy to connect to the microprocessor
- all of its pins are direct connections except the CS pin, which must be decoded, and the WR pin, which must have an I/O bank write pulse
8259A Pin-Outs

**D0–D7**
- The bidirectional data connections are normally connected to the data bus on the microprocessor.

**IR0–IR7**
- Interrupt request inputs are used to request an interrupt and to connect to a slave in a system with multiple 8259As.

**WR**
- The write input connects to write strobe signal (IOWC) on the microprocessor.

**RD**
- The read input connects to the IORC signal.

**INT**
- The interrupt output connects to the INTR pin on the processor from the master and is connected to a master IR pin on a slave.

**INTA**
- Interrupt acknowledge is an input that connects to the INTA signal on the system. In a system with a master and slaves, only the master INTA signal is connected.

**A0**
- The A0 address input selects different command words within the 8259A.

**CS**
- Chip select enables the 8259A for programming and control.

**CAS₀–CAS₂**
- The cascade lines are used as outputs from the master to the slaves for cascading multiple 8259As in a system.

**SP/EN**
- Slave program/enable buffer is a dual-function pin.
  - when the 8259A is in buffered mode, this output controls the data bus transceivers in a large microprocessor-based system
  - when the 8259A is not in the buffered mode, this pin programs the device as a master (1) or a slave (0)

**Connecting a Single 8259A**
- Fig 12–16 shows a single 8259A connected to the microprocessor.
- The 8259A is decoded at I/O ports 0400H and 0401H by the PLD.
- The 8259A requires four wait states for it to function properly with a 16 MHz 80386SX
  - more for some other versions of the Intel microprocessor family
Cascading Multiple 8259As

- Fig 12–17 shows two 8259As connected in a way often found in the ATX-style system.
  - XT- or PC-style computers use a single 8259A controller at interrupt vectors 08H–0FH.
  - The ATX-style computer uses interrupt vector 0AH as a cascade input from a second 8259A located at vectors 70H through 77H.
  - Appendix A contains a table that lists the functions of all the interrupt vectors used.

Programming the 8259A

- 8259A is programmed by initialization and operation command words.
  - **Initialization command words** (ICWs) are programmed before the 8259A is able to function in the system.
    - dictate the basic operation of the 8259A
  - **Operation command words** (OCWs) are programmed during the normal course of operation.
    - OCWs control the operation of the 8259A

Initialization Command Words

- the four initialization command words (ICWs) are selected when the A0 pin is logic 1
  - if a single 8259A is used in a system, ICW1, ICW2, and ICW4 must be programmed when powered up
  - if programmed in cascade mode by ICW1, then ICW3 must be programmed

ICW Descriptions - ICW1

- **ICW1** programs basic operation of the 8259A.
  - selects single or cascade operation by programming the SNGL bit
  - if cascade operation is selected, ICW3 must also be programmed
  - the LTIM bit determines whether interrupt request inputs are positive edge-triggered or level-triggered
ICW Descriptions - ICW₂

- ICW₂ selects the vector number used with the interrupt request inputs.
  - if programming 8259A so it functions at vector locations 08H–0FH, place 08H into this command word
  - if programming for vectors 70H–77H, place 70H in this ICW

ICW Descriptions - ICW₃

- ICW₃ is used only when ICW₁ indicates the system is operated in cascade mode.
  - this ICW indicates where the slave is connected to the master. (In Figure 12–18 a slave was connected to IR₃)
  - to program ICW₃ for this connection, in both master and slave, place 04H in ICW₃
  - if two slaves connected using IR₀ and IR₁, the master is programmed with an ICW₃ of 03H
  - one slave is programmed with an ICW₃ of 01H and the other with an ICW₃ of 02H

ICW Descriptions - ICW₄

- ICW₄ is programmed for use with the 8086–Pentium 4 processors, and the rightmost bit must be logic 1 to select operation with them.
- Remaining bits are programmed as follows:
  - SFNM—Selects the special fully nested mode of operation if logic 1 is placed in this bit
  - BUF and M/S—Buffered and master slave are used together to select buffered operation or nonbuffered operation for the 8259A as a master or a slave

Operation Command Words

- used to direct 8259A operation once programmed with the ICW
- OCWs are selected when the A₀ pin is at logic 0 level
- except OCW₁, which is selected when A₀ is logic 1
- shown here are the binary bit patterns for all three operation command words of the 8259A

OCW Descriptions - OCW₁

- OCW₁ is used to set and read the interrupt mask register.
  - when a mask bit is set, it will turn off (mask) the corresponding interrupt input
  - the mask register is read when OCW₁ is read
  - because the state of the mask bits is unknown when 8259A is first initialized, OCW₁ must be programmed after programming the ICW upon initialization
**OCW Descriptions - OCW₂**

- **OCW₂** is programmed only when the AEOI mode is not selected for the 8259A.
- Selects the way 8259A responds to an interrupt. Modes are listed as follows:
  - **Nonspecific End-of-Interrupt**—Command sent by the interrupt service procedure to signal the end of the interrupt
  - 8259A determines which interrupt level was active
  - resets the correct interrupt status register bit
  - resetting the bit allows the interrupt to take action again or a lower priority interrupt to take effect

**OCW Descriptions - OCW₂ (cont.)**

- **OCW₂ interrupt modes:**
  - **Rotate-on-Nonspecific EOI**—functions exactly like the Nonspecific End-of-Interrupt command, except it rotates interrupt priorities after resetting the interrupt status register bit
  - the level reset by this command becomes the lowest priority interrupt
  - if IR₄ was just serviced by this command, it becomes the lowest priority interrupt input and IR₅ becomes the highest priority

**OCW Descriptions - OCW₂ (cont.)**

- **OCW₂** selects register to be read, operation of the special mask register & poll command.
  - if polling is selected, the P bit must be set and then output to the 8259A; the next read operation will read the poll word
  - the rightmost three bits of the word indicate the active interrupt request with the highest priority
  - the leftmost bit indicates if there is an interrupt and must be checked to determine whether the rightmost three bits contain valid information
**Status Register**

- Three status registers are readable in 8259A:
  - Interrupt request register (IRR).
    - an 8-bit register that indicates which interrupt request inputs are active
  - In-service register (ISR).
    - an 8-bit register that contains the level of the interrupt being serviced
  - Interrupt mask register (IMR).
    - an 8-bit register that holds the interrupt mask bits and indicates which interrupts are masked off

**8259A Programming Example**

- Fig 12–21 shows 8259A connected to a 16550 programmable communications controller.
  - the INTR pin from the 16550 is connected to the programmable interrupt controller’s interrupt request input IR\(_0\)
  - The 16550 is decoded at I/O ports 40H and 47H, and the 8259A is decoded at 8-bit I/O ports 48H and 49H.
  - Both are interfaced to the data bus of an 8088.

**Initialization Software**

- The first portion of the software for this system must program both the 16550 and the 8259A.
  - then enable the 8088 INTR pin so interrupts can take effect
- Example 12–8 lists the software required to program both devices and enable INTR.
- The software uses two memory FIFOs to hold data for the transmitter and receiver.
  - each FIFO is 16K bytes long and addressed by a pair of pointers (input and output)
- INIT programs the 16550 UART for operation with seven data bits, odd parity, one stop bit, and a baud rate clock of 9600.
- The second part of the procedure programs the 8259A, with its three ICWs and one OCW.
- The final part enables the receiver and error interrupts of the 16550.
  - transmitter interrupt not enabled until data are available for transmission
- See Figure 12–22 for the contents of the interrupt control register of the 16550 UART.
Handling the 16550 UART Interrupt Request

- 16550 generates only one interrupt request for various interrupts.
  - so the interrupt handler must poll the 16550 to determine what type of interrupt has occurred
- Accomplished by examining the interrupt identification register, shown in figure 12–23.
  - the interrupt identification register (read-only) shares the same I/O port as the FIFO control register (write-only)

Receiving data from the 16550 requires two procedures.
- one procedure reads the data register of the 16550 each time the INTR pin requests an interrupt and stores it into the memory FIFO
- the other reads data from the FIFO from the main program
- Example 12–10 lists the procedure used to read data from the memory FIFO from the main program.

Transmitting Data to the 16550

- Data are transmitted to the 16550 in much the same manner as they are received.
  - except the interrupt service procedure removes transmit data from a second 16K-byte FIFO
- Example 12–12 lists the procedure that fills the output FIFO.
  - it is similar to the procedure listed in Example 12–10.
  - except it determines whether the FIFO is full instead of empty
- 16550 also contains a general-purpose "scratch" register.
  - it can be used in any way deemed necessary by the programmer
  - also contained in the 16550 are a modem control register and a modem status register.
  - these registers allow the modem to interrupt and control the operation of the 16550 with a modem
  - See Figure 12–24 for contents of both the modem status register and the modem control register.
Figure 12–24 The 16550 modem control and modem status registers.

Figure 12–25 The 16550 interfaced to an RS-232C using 1488 line drivers and 1489 line receivers.

12–5 INTERRUPT EXAMPLES

- This section presents a real-time clock and an interrupt-processed keyboard as examples of interrupt applications.
- A real-time clock (RTC) keeps time in real time; also used for precision time delays.
- The interrupt-processed keyboard uses a periodic interrupt to scan through the keys of the keyboard.

Real-Time Clock

- Fig 12–26 shows a circuit using the 60 Hz AC power line to generate a periodic interrupt request signal for the NMI interrupt input pin.
- You must make certain that the power line ground is connected to the system ground in this schematic.
  - the power line neutral (white wire) connection is the wide flat pin on the power line
  - the narrow flat pin is the hot (black wire) side or 120 V AC side of the line

Figure 12–26 Converting the AC power line to a 60 Hz TTL signal for the NMI input.
• The software for the real-time clock contains an interrupt service procedure that is called 60 times per second and a procedure that updates the count located in four memory locations.
• Example 12–14 lists both procedures, along with the four bytes of memory used to hold the BCD time of day.
• Another way to handle time is to use a single counter to store the time in memory and then determine the actual time with software.

![Diagram of a telephone-style keypad interfaced to the 82C55.]

**Interrupt-Processed Keyboard**

• The interrupt-processed keyboard scans the keyboard through a periodic interrupt.
  – on each interrupt occurs, the interrupt-service procedure tests for a key or debounces the key
• Once a valid key is detected, the interrupt service procedure stores the key code into a queue for later reading by the system.
• Figure 12–27 shows the keyboard interfaced to an 82C55 and Example 12–18 lists the interrupt service procedure for the keyboard.

**SUMMARY**

• An interrupt is a hardware- or software-initiated call that interrupts the currently executing program at any point and calls a procedure.
• The procedure is called by the interrupt handler or an interrupt service procedure.
• Interrupts are useful when an I/O device needs to be serviced only occasionally at low data transfer rates.

**SUMMARY (cont.)**

• The microprocessor has five instructions that apply to interrupts: BOUND, INT, INT 3, INTO, and IRET.
• The INT and INT 3 instructions call procedures with addresses stored in the interrupt vector whose type is indicated by the instruction.
• The BOUND instruction is a conditional interrupt that uses interrupt vector type number 5.

• The INTO instruction is a conditional interrupt that interrupts a program only if the overflow flag is set.
• Finally, the IRET, IRETD, or IRETQ instruction is used to return from interrupt service procedures.
• The microprocessor has three pins that apply to its hardware interrupt structure.
SUMMARY (cont.)

- Real mode interrupts are referenced through a vector table that occupies memory locations 0000H-03FFH.
- Each interrupt vector is four bytes long and contains the offset and segment addresses of the interrupt service procedure.
- In protected mode, the interrupts reference the interrupt descriptor table (IDT) that contains 256 interrupt descriptors.

SUMMARY (cont.)

- Two flag bits are used with the interrupt structure of the microprocessor: trap (TF) and interrupt enable (IF).
- The IF flag bit enables the INTR interrupt input.
- TF flag bit causes interrupts to occur after the execution of each instruction, as long as TF is active.

SUMMARY (cont.)

- The first 32 interrupt vector locations are reserved for Intel use, with many predefined in the microprocessor.
- The last 224 interrupt vectors are for the user's use and can perform any function desired.

SUMMARY (cont.)

- Whenever an interrupt is detected, the following events occur:
  1. The flags are pushed onto the stack
  2. The IF and TF flag bits are both cleared
  3. The IP and CS registers are both pushed onto the stack
  4. The interrupt vector is fetched from the interrupt vector table and the interrupt service subroutine is accessed through the vector address.

SUMMARY (cont.)

- Tracing or single-stepping is accomplished by setting the TF flag bit.
- This causes an interrupt to occur after the execution of each instruction for debugging.
- The non-maskable interrupt input (NMI) calls the procedure whose address is stored at interrupt vector type number 2.
- This input is positive edge-triggered.

SUMMARY (cont.)

- The INTR pin is not internally decoded, as is the NMI pin.
- Methods of applying the interrupt vector type number to the data bus vary widely.
- One method uses registers to apply interrupt type number FFH to the data bus, while another uses a three-state buffer to apply any vector type number.
SUMMARY (cont.)

- The 8259A programmable interrupt controller (PIC) adds at least eight interrupt in-puts to the microprocessor.
- If more interrupts are needed, this device can be cascaded to provide up to 64 interrupt inputs.

SUMMARY (cont.)

- Programming the 8259A is a two-step process.
- First, a series of initialization command words (ICWs) are sent to the 8259A,
- Second, a series of operation command words (OCWs) are sent.

SUMMARY

- The 8259A contains three status registers:
  IMR (interrupt mask register)
  ISR (in-service register)
  IRR (interrupt request register).
- A real-time clock is used to keep time in real time. In most cases, time is stored in either binary or BCD form in several memory locations.