**13. Basic I/O Interface. Interface an Analog-to-Digital Converter and a Digital-to-Analog Converter to the microprocessor.**
**14. I8254 programmable internal timer. Architecture. Modes of operation. Application.**
**15. Direct Memory Access. System architecture with i8237 DMA controller. Modes of operation.**

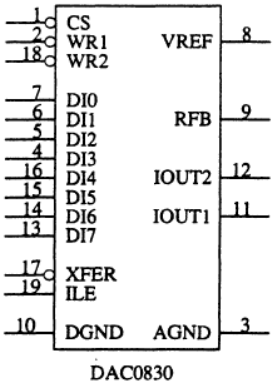
## ANALOG-TO-DIGITAL (ADC) AND DIGITAL-TO-ANALOG (DAC) CONVERTERS

Analog-to-digital (ADC) and digital-to-analog (DAC) converters are used to interface the microprocessor to the analog world. Many events that are monitored and controlled by the microprocessor are analog events. These often include monitoring all forms of events, even speech, to controlling motors and like devices. In order to interface the microprocessor to these events, we must understand the interface and control of the ADC and DAC, which convert between analog and digital data.

A fairly common and low-cost digital-to-analog converter is the DAC0830 (a product of National Semiconductor Corporation). This device is an 8-bit converter that transforms an 8-bit binary number into an analog voltage. Other converters are available that convert from 10-, 12-, or 16-bits into analog voltages. The number of voltage steps generated by the converter is equal to the number of binary input combinations. Therefore, an 8-bit converter generates 256 different voltage levels, a 10-bit converter generates 1,024 levels, and so forth. The DAC0830 is a medium-speed converter that transforms a digital input to an analog output in approximately 1 μs.

Figure 10–48 illustrates the pin-out of the DAC0830. This device has a set of eight data bus connections for the application of the digital input code and a pair of analog outputs labeled Iout1 and Iout2 that are designed as inputs to an external operational amplifier. Because this is an 8-bit converter, its output step voltage is defined as $-V_{REF}$ **(reference voltage)** divided by 255. For example, if the reference voltage is –5.0V, its output step voltage is +.0196V. Note that the output voltage is the opposite polarity of the reference voltage. If an input of 1001 $0010_2$ is applied to the device, the output voltage will be the step voltage times 1001 $0010_2$, or in this case +2.862V. By changing the reference voltage to –5.1V, the step voltage becomes +.02V. The step voltage is also often called the *resolution* of the converter.

**FIGURE 10–48** The pin-out of the DAC0830 digital-to-analog converter




*Connecting the DAC0830 to the Microprocessor.* The DAC0830 is connected to the microprocessor as illustrated in Figure 10–50. Here a PAL16L8 is used to decode the DAC0830 at 8-bit I/O port address 20H. Whenever an OUT 20H,AL instruction is executed, the contents of data bus connections $AD_0$–$AD_7$ are passed to the converter within the DAC0830. The 741 operational amplifier along with the –12V zener reference voltage causes the full-scale output voltage to equal +12V. The output of the operational amplifier feeds a driver that powers a 12V DC motor. This driver is a Darlington amplifier for large motors. This example shows the converter driving a motor, but other devices could be used as outputs.

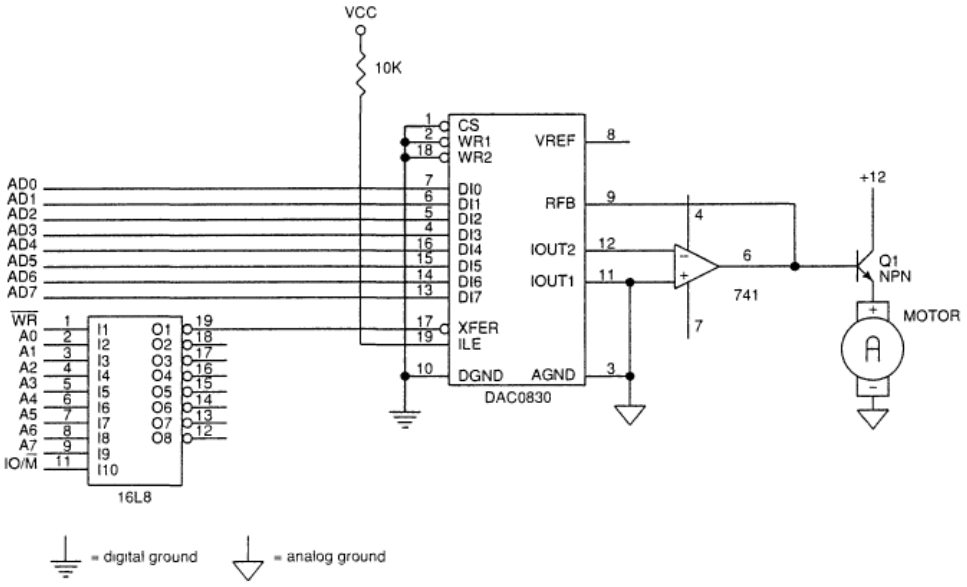



**FIGURE 10–50** A DAC0830 interfaced to the 8086 microprocessor at 8-bit I/O location 20H

A common low-cost ADC is the ADC0804, which belongs to a family of converters that are all identical except for accuracy. This device is compatible with a wide range of microprocessors such as the Intel family. There are faster ADCs available, and some with more resolution than 8-bits, but this device is ideal for many applications that do not require a high degree of accuracy. The ADC0804 requires up to 100 μs to convert an analog input voltage into a digital output code.

Figure 10–51 shows the pin-out of the ADC0804 converter (a product of National Semiconductor Corporation). To operate the converter, the $\overline{WR}$ pin is pulsed with $\overline{CS}$ grounded to start the conversion process. Because this converter requires a considerable amount of time for the conversion, a pin labeled INTR signals the end of the conversion. Refer to Figure 10–52 for a timing diagram that shows the interaction of the control signals. As can be seen, we start the converter with the $\overline{WR}$ pulse, we wait for INTR to return to a logic 0 level, and then we read the data from the converter. If a time delay is used that allows at least 100 μs of time, then we don't need to test the INTR pin. Another option is to connect the INTR pin to an interrupt input so an interrupt occurs when the conversion is complete.

*The Analog Input Signal.* Before the ADC0804 can be connected to the microprocessor, its analog inputs must be understood. There are two analog inputs to the ADC0804: $V_{IN}$ (+) and $V_{IN}$ (–). These inputs are connected to an internal operational amplifier and are differential inputs, as shown in Figure 10–53. The differential inputs are summed by the operational amplifier to produce a signal for the internal analog-to-digital converter. Figure 10–53 shows a few ways to use these differential inputs. The first way (Figure 10–53a) uses a single input that can vary between 0V and +5.0V. The second (Figure 10–53b) shows a variable voltage applied to the $V_{IN}$ (–) pin so the zero reference for $V_{IN}$ (+) can be adjusted.

*Generating the Clock Signal.* The ADC0804 requires a clock source for operation. The clock can be an external clock applied to the CLK IN pin, or it can be generated with an RC circuit. The permissible range of clock frequencies is between 100 KHz and 1460 KHz. It is desirable to use a frequency that is as close as possible to 1460 KHz so conversion time is kept to a minimum.

*Connecting the ADC0804 to the Microprocessor.* The ADC0804 is interfaced to the 8086 microprocessor as illustrated in Figure 10–55. Note the $V_{REF}$ signal is not attached to anything,

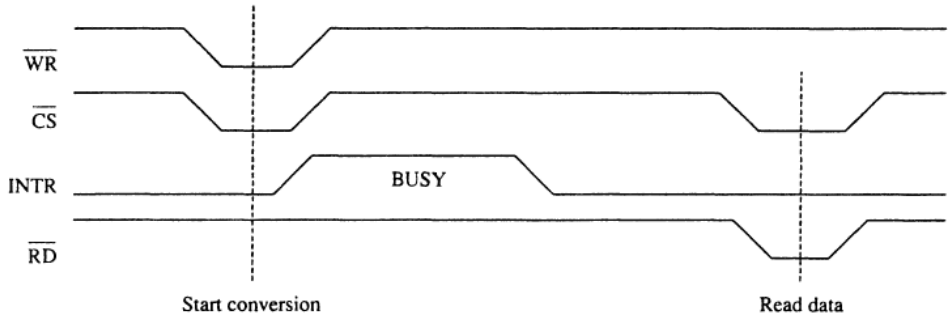
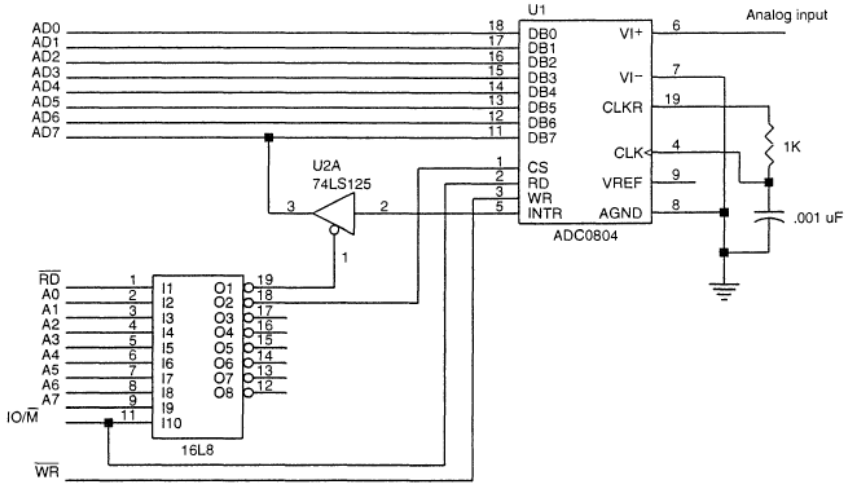



**FIGURE 10–52** The timing for the ADC0804 analog-to-digital converter

which is normal. Suppose that the ADC0804 is decoded at 8-bit I/O port address 40H for the data and port address 42H for the INTR signal and a procedure is required to start and read the data from the ADC. This procedure is listed in Example 10–27. Notice that the INTR bit is polled; if it becomes a logic 0, the procedure ends with AL containing the converted digital code.

**FIGURE 10–55** The ADC0804 interfaced to the microprocessor




This section of the text illustrates an example using both the ADC0804 and the DAC0830 to capture and replay audio signals or speech. In the past we often used a speech synthesizer to generate speech, but the quality of the speech was poor. For quality speech we can use the ADC0804 to capture an audio signal and store it in memory for later playback through the DAC0830.

## 8254 PROGRAMMABLE INTERVAL TIMER

The 8254 programmable interval timer consists of three independent 16-bit programmable counters *(timers)*. Each counter is capable of counting in binary or binary-coded decimal (BCD). The maximum allowable input frequency to any counter is 10 MHz. This device is useful wherever the microprocessor must control real-time events. Some examples of usage include real time clock, events counter, and motor speed and direction control.

This timer also appears in the personal computer decoded at ports 40H–43H to (1) generate a basic timer interrupt that occurs at approximately 18.2 Hz, (2) cause the DRAM memory system to be refreshed, and (3) provide a timing source to the internal speaker and other devices. The timer in the personal computer is an 8253 instead of an 8254.

The signals that connect to the microprocessor are the data bus pins ($D_7$–$D_0$), RD, WR, CS, and address inputs $A_1$ and $A_0$. The address inputs are present to select any of the four internal registers used for programming, reading, or writing to a counter. The personal computer contains an 8253 timer or its equivalent decoded at I/O ports 40H–43H. Timer zero is programmed to generate an 18.2 Hz signal that interrupts the microprocessor at interrupt vector 8 for a clock tick. The tick is often used to time programs and events. Timer 1 is programmed for a 15 μs output that is used on the PC/XT personal computer to request a DMA action used to refresh the dynamic RAM. Timer 2 is programmed to generate tone on the personal computer speaker.

### *Pin Definitions*

**$A_1$, $A_0$** The **address inputs** select one of four internal registerswithin the 8254. See Table 10–6 for the function of the $A_1$ and $A_0$ address bits.

**CLK** The **clock** input is the timing source for each of theinternal counters. This input is often connected to the PCLK signal from the microprocessor system bus controller.

**CS** **Chip select** enables the 8254 for programming and reading or writing a counter.

## 8254 Functional Description

Figure 10–33 shows the pin-out of the 8254, which is a higher-speed version of the 8253, and a diagram of one of the three counters. Each timer contains a CLK input, a gate input, and an output (OUT) connection. The CLK input provides the basic operating frequency to the timer, the gate pin controls the timer in some modes, and the OUT pin is where we obtain the output of the timer.

**G** The **gate** input controls the operation of the counter in some modes of operation.

**GND** **Ground** connects to the system ground bus.

**OUT** A **counter output** is where the wave-form generated by the timer is available.

**$\overline{RD}$** **Read** causes data to be read from the 8254 and often connects to the $\overline{IORC}$ signal.

**$V_{CC}$** **Power** connects to the +5.0V power supply.

**$\overline{WR}$** **Write** causes data to be written to the 8254 and often connects to the write strobe ($\overline{IOWC}$).

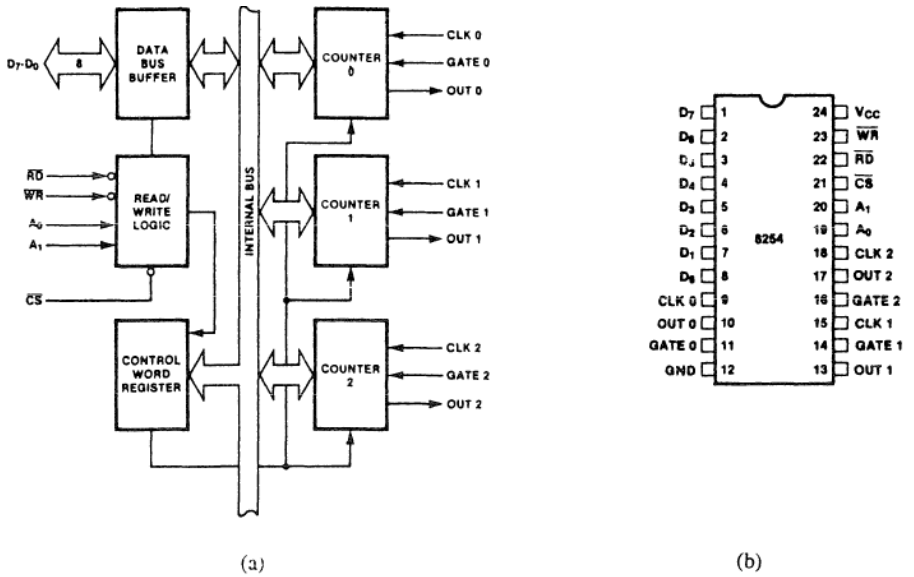


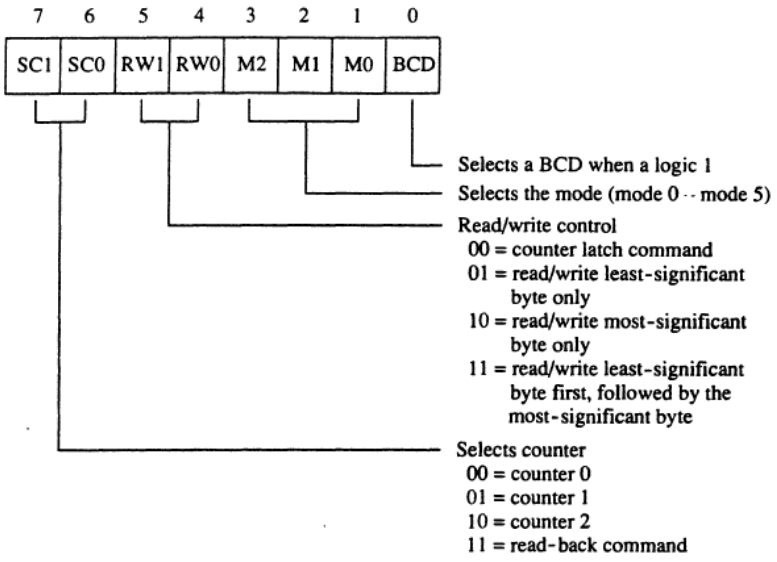

**FIGURE 10–33** The 8254 programmable interval timer. (a) Internal structure, and (b) pin-out

Each counter is individually programmed by writing a control word followed by the initial count. Figure 10–34 lists the program control word structure of the 8254. The **control word** allows the programmer to select the counter, mode of operation, and type of operation (read/write). The control word also selects either a binary or BCD count. Each counter may be programmed with a count of 1 to FFFFH. A count of 0 is equal to FFFFH+1 (65,536) or 10,000 in BCD. The minimum count of 1 applies to all modes of operation except modes 2 and 3, which have a minimum count of 2. Timer 0 is used in the personal computer with a divide by count of 64K (FFFFH) to generate the 18.2 Hz (18.196 Hz) interrupt clock tick. Timer 0 has a clock input frequency of 4.77 MHz ÷ 4, or 1.1925 MHz.

**TABLE 10–6** Address selection inputs to the 8254

| $A_1$ | $A_0$ | *Function* |
|---|---|---|
| 0 | 0 | Counter 0 |
| 0 | 1 | Counter 1 |
| 1 | 0 | Counter 2 |
| 1 | 1 | Control word |

**FIGURE 10–34** The control word for the 8254–2 timer




The control word uses the BCD bit to select a BCD count (BCD = 1) or a binary count (BCD = 0). The M2, M1, and M0 bits select one of the six different modes of operation (000–101) for the counter. The RW1 and RW0 bits determine how the data are read from or written to the counter. The SC1 and SC0 bits select a counter or the special read-back mode of operation discussed later in this section.

Each counter has a program control word used to select the way the counter operates. If two bytes are programmed into a counter, then the first byte (LSB) will stop the count, and the second byte (MSB) will start the counter with the new count. The order of programming is important for each counter, but programming of different counters may be interleaved for better control. For example, the control word may be sent to each counter before the counts for individual programming.

*Modes of Operation.* Six modes of operation (mode 0 – mode 5) are available to each of the 8254 counters. Figure 10–35 shows how each of these modes functions with the CLK input, the gate (G) control signal, and the OUT signal. A description of each mode follows.

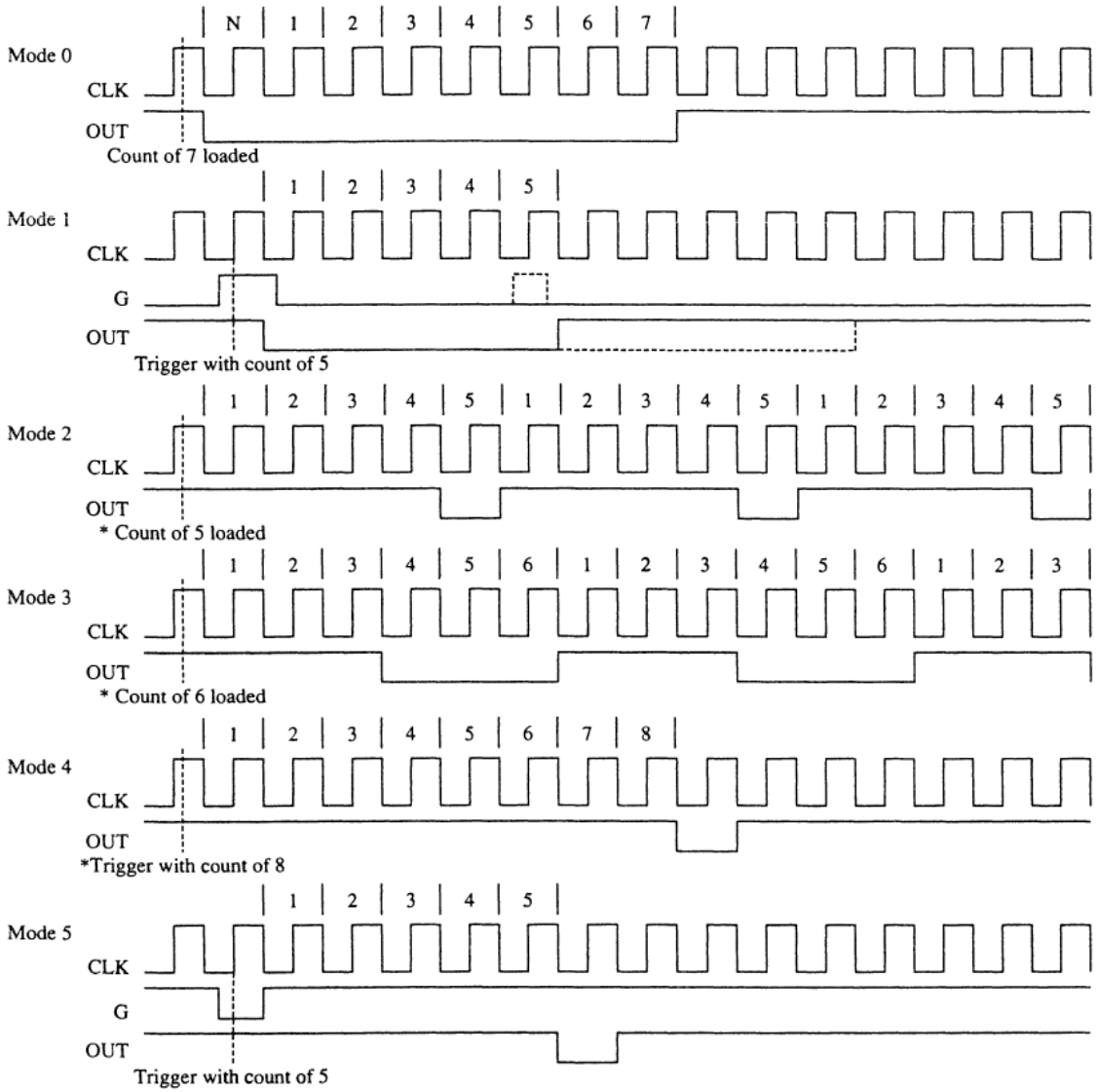



**FIGURE 10–35** The six modes of operation for the 8254–2 programmable interval timer. *The G input stops the count when 0 in modes 2, 3, and 4.

**Mode 0** Allows the 8254 counter to be used as an events counter. In this mode, the output becomes a logic 0 when the control word is written and remains until N plus the number of programmed counts. For example, if a count of 5 is programmed, the output will remain a logic 0 for 6 counts beginning with N. Note that the gate (G) input must be a logic 1 to allow the counter to count. If G becomes a logic 0 in the middle of the count, the counter will stop until G again becomes a logic 1.

**Mode 1** Causes the counter to function as a retriggerable monostable multivibrator (one-shot). In this mode, the G input triggers the counter so that it develops a pulse at the OUT connection that becomes a logic 0 for the duration of the count. If the count is 10, then the OUT connection goes low for 10 clocking periods when triggered. If the G input occurs within the duration of the output pulse, the counter is again reloaded with the count and the OUT connection continues for the total length of the count.

**Mode 2** Allows the counter to generate a series of continuous pulses that are one clock pulse in width. The separation between pulses is determined by the count. For

example, for a count of 10, the output is a logic 1 for 9 clock periods and low for 1 clock period. This cycle is repeated until the counter is programmed with a new count or until the G pin is placed at a logic 0 level. The G input must be a logic 1 for this mode to generate a continuous series of pulses.

**Mode 3** Generates a continuous square-wave at the OUT connection, provided the G pin is a logic 1. If the count is even, the output is high for one-half of the count and low for one-half of the count. If the count is odd, the output is high for one clocking period longer than it is low. For example, if the counter is programmed for a count of 5, the output is high for 3 clocks and low for 2.

**Mode 4** Allows the counter to produce a single pulse at the output. If the count is programmed as a 10, the output is high for 10 clocking periods and then low for 1 clocking period. The cycle does not begin until the counter is loaded with its complete count. This mode operates as a software triggered one-shot. As with modes 2 and 3, this mode also uses the G input to enable the counter. The G input must be a logic 1 for the counter to operate for these three modes.

**Mode 5** A hardware triggered one-shot that functions as mode 4 except that it is started by a trigger pulse on the G pin instead of by software. This mode is also similar to mode 1 because it is retriggerable.

*Generating a Wave-form with the 8254.* Figure 10–36 shows an 8254 connected to function at I/O ports 0700H, 0702H, 0704H, and 0706H of an 80386SX microprocessor. The addresses are decoded using a PAL16L8 that also generates a write strobe signal for the 8254, which is connected to the low order data bus connections. The PAL also generates a wait signal for the microprocessor that causes two wait states when the 8254 is accessed.

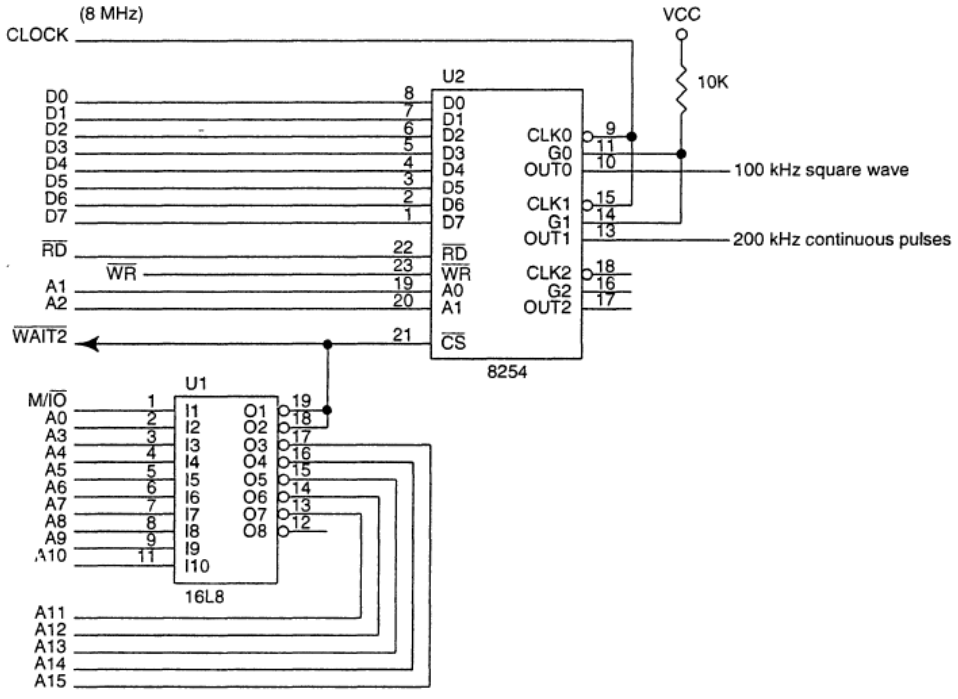



**FIGURE 10–36** The 8254 interfaced to an 8 MHz 8086 so that it generates a 100 KHz square wave at OUTO and a 200KHz continous pulse at OUT1

The program generates a 100 KHz square-wave at OUT0 and a 200 KHz continuous pulse at OUT1. We use mode 3 for counter 0 and mode 2 for counter 1. The count programmed into counter 0 is 80, and the count for counter 1 is 40. These counts generate the desired output frequencies with an 8 MHz input clock.

**Direct memory access** (DMA). The DMA I/O technique provides direct access to the memory while the microprocessor is temporarily disabled. This allows data to be transferred between memory and the I/O device at a rate that is limited only by the speed of the memory components in the system or the DMA controller. The DMA transfer speed can approach 15–20 M-byte transfer rates with today's high-speed RAM memory components.

DMA transfers are used for many purposes, but more common are DRAM refresh, video displays for refreshing the screen, and disk memory system reads and writes. The DMA transfer is also used to do high-speed memory-to-memory transfers.

Two control signals are used to request and acknowledge a direct memory access (DMA) transfer in the microprocessor-based system. The HOLD pin is an input used to request a DMA action, and the HLDA pin is an output that acknowledges the DMA action. Figure 12–1 shows the timing that is typically found on these two DMA control pins.

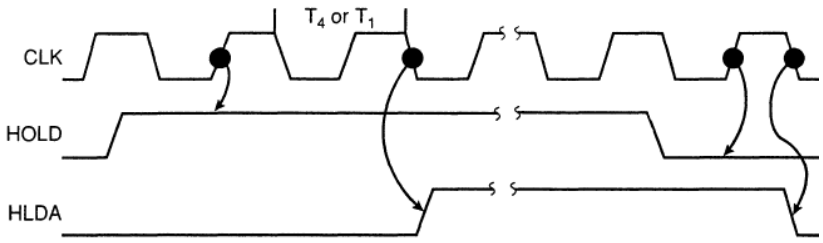



**FIGURE 12–1** HOLD and HLDA timing for the 8086/8088 microprocessors

Whenever the HOLD input is placed at a logic 1 level, a DMA action (hold) is requested. T microprocessor responds, within a few clocks, by suspending the execution of the program a by placing its address, data, and control bus at their high-impedance states. The high-impedar state causes the microprocessor to appear as if it has been removed from its socket. This st allows external I/O devices or other microprocessors to gain access to the system buses so memc can be accessed directly.

As the timing diagram indicates, HOLD is sampled in the middle of any clocking cycle. Thus, the hold can take effect at any time during the operation of any instruction in the microprocessor. As soon as the microprocessor recognizes the hold, it stops executing software and enters hold cycles. Note that the HOLD input has a higher priority than the INTR or NMI interrupt inputs. Interrupts take effect at the end of an instruction, while a HOLD takes effect in the middle of an instruction. The only microprocessor pin that has a higher priority than a HOLD is the RESET pin. Note that the HOLD input may not be active during a RESET or the reset is not guaranteed.

The HLDA signal becomes active to indicate that the microprocessor has indeed placed its buses at their high-impedance states, as can be seen in the timing diagram. Note that there are a few clock cycles between the time that HOLD changes until the time that HLDA changes. The HLDA output is a signal to the external requesting device that the microprocessor has relinquished control of its memory and I/O space. You could call the HOLD input a DMA request input and the HLDA output a DMA grant signal.

Direct memory accesses normally occur between an I/O device and memory without the use of the microprocessor. A **DMA read** transfers data from the memory to the I/O device. A **DMA write** transfers data from an I/O device to memory. In both operations, the memory and I/O are controlled simultaneously, which is why the system contains separate memory and I/O control signals. This special control bus structure of the microprocessor allows DMA transfers. A DMA read causes the $\overline{\text{MRDC}}$ and $\overline{\text{IOWC}}$ signals both to activate, transferring data from the memory to the I/O device. A DMA write causes the $\overline{\text{MWTC}}$ and $\overline{\text{IORC}}$ signals both to activate. These control bus signals are available to all microprocessors in the Intel family, except the 8086/8088 system. The 8086/8088 require their generation with either a system controller or a circuit such as the one illustrated in Figure 12–2. The DMA controller provides the memory with its address, and a signal from the controller ($\overline{\text{DACK}}$) selects the I/O device during the DMA transfer.

The data transfer speed is determined by the speed of the memory device or a DMA controller that often controls DMA transfers. If the memory speed is 100 ns, DMA transfers occur at rates of up to 1/100 ns or 10 M-bytes per second. If the DMA controller in a system functions at a maximum rate of 5 MHz and we still use 100 ns memory, the maximum transfer rate is 5 MHz because the DMA controller is slower than the memory. In many cases, the DMA controller slows the speed of the system when DMA transfers occur.

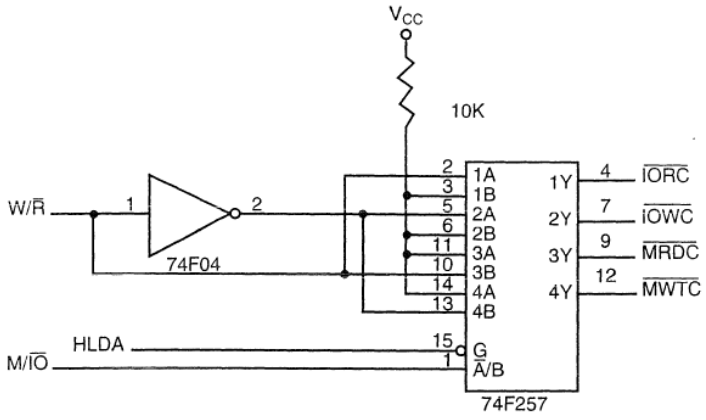



**FIGURE 12–2** A circuit that generates system control signals in a DMA environment

## THE 8237 DMA CONTROLLER

The 8237 DMA controller supplies the memory and I/O with control signals and memory address information during the DMA transfer. The 8237 is actually a special purpose microprocessor whole job is high-speed data transfer between memory and the I/O. Figure 12–3 shows the pin-out and block diagram of the 8237 programmable DMA controller. Although this device may not appear as a discrete component in modern microprocessor-based systems, it does appear within system controller chip sets found in most newer systems. Although not described because of its complexity, the chip set (82357 ISP or integrated peripheral controller), and its integral set of two DMA controllers, are programmed exactly as the 8237. The ISP also provides a pair of 8259A programmable interrupt controllers for the system.

The 8237 is a four-channel device that is compatible to the 8086/8088 microprocessors. The 8237 can be expanded to include any number of DMA channel inputs, although four channels seem to be adequate for many small systems. The 8237 is capable of DMA transfers at rates of up to 1.6M bytes per second. Each channel is capable of addressing a full 64K-byte section of memory and can transfer up to 64K bytes with a single programming.

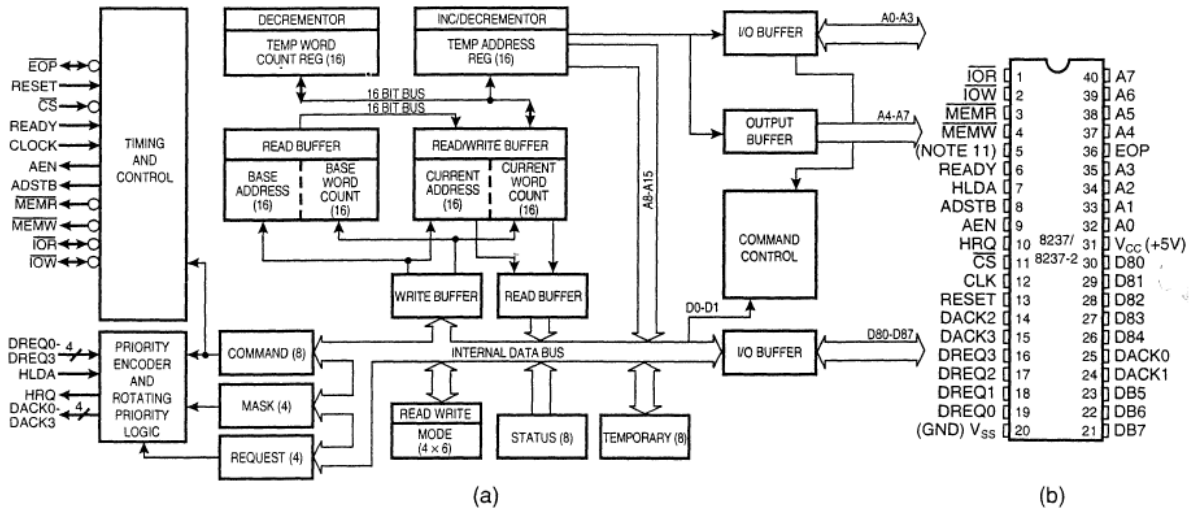



**FIGURE 12–3** The 8237A–5 programmable DMA controller. (a) Block diagram, and (b) pin-out

## *Pin Definitions*

**CLK** — The **clock** input is connected to the system clock signal as long as that signal is 5 MHz or less. In the 8086/8088 system, the clock must be inverted for the proper operation of the 8237.

**$\overline{CS}$** — **Chip select** enables the 8237 for programming. The $\overline{CS}$ pin is normally connected to the output of a decoder. The decoder does not use the 8086/8088 control signal IO/$\overline{M}$ (M/$\overline{IO}$) because it contains the new memory and I/O control signals ($\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$, and $\overline{IOW}$).

**RESET** — The **reset** pin clears the command, status, request, and temporary registers. It also clears the first/last flip-flop and sets the mask register. This input primes the 8237 so it is disabled until programmed otherwise.

**READY** — A logic 0 on the **ready** input causes the 8237 to enter wait states for slower memory and/or I/O components.

**HLDA** — A **hold acknowledge** signals the 8237 that the microprocessor has relinquished control of the address, data, and control buses.

**$DREQ_3$–$DREQ_0$** — The **DMA request** inputs are used to request a DMA transfer for each of the four DMA channels. Because the polarity of these inputs is programmable, they are either active-high or active-low inputs.

**$DB_7$–$DB_0$** — The **data bus** pins are connected to the microprocessor data bus connections and are used during the programming of the DMA controller.

**$\overline{IOR}$** — **I/O read** is a bi-directional pin used during programming and during a DMA write cycle.

**$\overline{IOW}$** — **I/O write** is a bi-directional pin used during programming and during a DMA read cycle.

**$\overline{EOP}$** — **End-of-process** is a bi-directional signal that is used as an input to terminate a DMA process or as an output to signal the end of the DMA transfer. This input is often used to interrupt a DMA transfer at the end of a DMA cycle.

**$A_3$–$A_0$** — These **address pins** select an internal register during programming and also provide part of the DMA transfer address during a DMA action.

**$A_7$–$A_4$** These **address pins** are outputs that provide part of the DMA transfer address during a DMA action.

**HRQ** **Hold request** is an output that connects to the HOLD input of the microprocessor in order to request a DMA transfer.

**$DACK_3$–$DACK_0$** **DMA channel acknowledge** outputs acknowledge a channel DMA request. These outputs are programmable as either active-high or active-low signals. The DACK outputs are often used to select the DMA-controlled I/O device during the DMA transfer.

**AEN** The **address enable** signal enables the DMA address latch connected to the $DB_7$–$DB_0$ pins on the 8237. It is also used to disable any buffers in the system connected to the microprocessor.

**ADSTB** **Address strobe** functions as ALE, except that it is used by the DMA controller to latch address bits $A_{15}$–$A_8$ during the DMA transfer.

**$\overline{MEMR}$** **Memory read** is an output that causes memory to read data during a DMA read cycle.

**$\overline{MEMW}$** **Memory write** is an output that causes memory to write data during a DMA write cycle.

*Internal Registers*

**CAR** The **current address register** is used to hold the 16-bit memory address used for the DMA transfer. Each channel has its own current address register for this purpose. When a byte of data is transferred during a DMA operation, the CAR is either incremented or decremented, depending on how it is programmed.

**CWCR** The **current word count register** programs a channel for the number of bytes (up to 64K) transferred during a DMA action. The number loaded into this register is one less than the number of bytes transferred. For example, if a 10 is loaded into the CWCR, then 11 bytes are transferred during the DMA action.

**BA and BWC** The **base address** (BA) and **base word count** (BWC) registers are used when auto-initialization is selected for a channel. In the auto-initialization mode, these registers are used to reload both the CAR and CWCR after the DMA action is completed. This allows the same count and address to be used to transfer data from the same memory area.

**CR** The **command register** programs the operation of the 8237 DMA controller. Figure 12–4 depicts the function of the command register.

**MR** The **mode register** programs the mode of operation for a channel. Note that each channel has its own mode register (see Figure 12–5) as selected by bit positions 1 and 0. The remaining bits of the mode register select the operation, auto-initialization, increment/decrement, and mode for the channel. Verification operations generate the DMA addresses without generating the DMA memory and I/O control signals.

The modes of operation include demand mode, single mode, block mode, and cascade mode. Demand mode transfers data until an external $\overline{EOP}$ is input or until the DREQ input becomes inactive. Single mode releases the HOLD after each byte

of data is transferred. If the DREQ pin is held active, the 8237 again requests a DMA transfer through the DRQ line to the microprocessor's HOLD input. Block mode automatically transfers the number of bytes indicated by the count register for the channel. DREQ need not be held active through the block mode transfer. Cascade mode is used when more than one 8237 is present in a system.

**RR** The **request register** is used to request a DMA transfer via software (see Figure 12–6). This is very useful in memory-to-memory transfers where an external signal is not available to begin the DMA transfer.

**MRSR** The **mask register set/reset** sets or clears the channel mask, If the mask is set, the channel is disabled. Recall that the RESET signal sets all channel masks to disable them.

**MSR** The **mask register** clears or sets all of the masks with one command instead of individual channels as with the MRSR.

**SR** The **status register** shows the status of each DMA channel. The TC bits indicate if the channel has reached its terminal count (transferred all of its bytes). Whenever the terminal count is reached, the DMA transfer is terminated for most modes of operation. The request bits indicate if the DREQ input for a given channel is active.

There are four steps required to program the 8237: (1) the F/L flip-flop is cleared using a clear F/L command, (2) the channel is disabled, (3) the LSB and then MSB of the address are programmed, and (4) the LSB and MSB of the count are programmed. Once these four operations are performed, the channel is programmed and ready to use. Additional programming is required to select the mode of operation before the channel is enabled and started.

## SUMMARY

1. The HOLD input is used to request a DMA action, and the HLDA output signals that the hold is in effect. When a logic 1 is placed on the HOLD input, the microprocessor (1) stops executing the program, (2) places its address, data, and control bus at their high-impedance states, and (3) signals that the hold is in effect by placing a logic 1 on the HLDA pin.
2. A DMA read operation transfers data from a memory location to an external I/O device. A DMA write operation transfers data from an I/O device into the memory. Also available is a memory-to-memory transfer that allows data to be transferred between two memory locations using DMA techniques.
3. The 8237 direct memory access (DMA) controller is a four-channel device that can be expanded to include additional channels of DMA.