This chapter introduces the current line of Motorola microprocessors. It is important to learn the operation and interfacing of either the MC6800 or MC6809 microprocessor first, since the remainder of this text emphasizes these microprocessors. Once they are learned, transition to the MC68000 or any other microprocessor manufactured by any of the IC houses is easy. For example, the 8085A is also covered in some detail.

Whichever microprocessor you choose to study, you will find subsequent chapters interesting and useful.
4-1 PINOUTS

Figure 4-1 illustrates the MC6800, MC6809, and MC68000 microprocessors' pinouts. The MC6800 and MC6809 are both packaged in 40-pin dual in-line packages, while the MC68000 is integrated into a 64-pin dual in-line package. All three devices operate from a single 5V power supply with power dissipations of less than 1.5 W.

Output Loading
The MC6800 and MC6809 microprocessors are capable of providing 1.6 mA of sink current and 400 μA of source current at any of the output pins. This current will drive one 74LSXX TTL unit load, four 74LSXX TTL unit loads, or about ten NMOS or CMOS unit loads. Only ten NMOS or CMOS unit loads may be connected because each MOS input places a fairly large amount of capacitance on an output connection. Too much bus capacitance will degrade the timing signals issued by the microprocessor, causing performance problems. To prevent this, MOS loads are limited to ten or less. Refer to Table 3-1 in Chapter 3 for a detailed look at unit loading.

The MC68000 is capable of sinking 1.6 mA on the HALT pin, 3.2 mA on the address pins, and about 5.0 mA on the data bus connections. With this device, more TTL components can be driven directly from the output pins of the microprocessor. This means that a larger system may be connected directly to the MC68000 without the addition of external bus buffers. Since the source current at these outputs remains at 400 μA, the maximum number of MOS loads remains at ten or less.

It is interesting to note that the MC6800 and MC6809 microprocessors will not drive a 74SXXX series TTL load. This limitation can be overcome by using a 74ASXXX series gate or a FAST gate from Fairchild.

Input Loading
Input connections on all three microprocessors sink and source a maximum of 2.5 μA of current and present about 10 pF of capacitance. In addition to low loading, they are also compatible with the standard TTL voltage levels.

Noise Immunity
The system noise immunity in any of the Motorola processors is about 400 mV and is directly compatible with standard TTL noise immunities. In systems that contain heavy capacitive loads, long bus connections, or excessive current loads, bus buffers at the output connections are recommended. With additional buffering, it is possible to connect up to 100 MOS or 74LSXXX TTL unit loads to an output connection. Most buffers contain an enhanced pullup network that has been designed to drive capacitive loads.

FIGURE 4-1 Pin diagrams or pinouts of the Motorola MC6800, MC6809, and MC68000 microprocessors.

SOURCE: Courtesy of Motorola, Inc.

CLOCK CIRCUITRY

The MC6809 contains an internal clock generator that, in most cases, generates the basic timing for the microprocessor. The MC6800 and the MC68000 both require the addition of an external clock generator to provide their basic timing.

MC6809 Clock Circuitry
Under normal operation, a crystal with a frequency of 8 MHz would be attached between the EXTAL and XTAL input pins of the MC6809 (as pictured in Figure 4-2). The crystal is internally divided by a factor of four to produce the 2 MHz basic operating frequency. The range of allowable crystal frequencies is between 8.0 MHz and about 400 kHz for reliable operation.

4 MHz

FIGURE 4-2 The MC6809 clock generation circuitry.
an operating frequency outside this range is chosen. Motorola will not guarantee the proper operation of the MC6809.

In addition to a crystal, the MC6809 may be driven from an external TTL source. This is accomplished by grounding the XTAL pin and connecting the external TTL clock signal to the EXTL clock connection. This method of operation is used in multiple processor systems, where one timing source drives all of the processors.

MC6800 and MC68000 Clock Circuits

The MC6800 and MC68000 require an external clock generator for proper operation. The MC6875 clock generator, as illustrated in figure 4-3, can generate the required multiphase clock inputs for the MC6800. The MC68000 requires a TTL compatible clock input of up to 8.0 MHz for proper operation. A circuit that can be used to generate this clock is illustrated in figure 4-4.

![Figure 4-3](image)

**Figure 4-3** The MC6875 clock generator connected to the MC6800 microprocessor.

![Figure 4-4](image)

**Figure 4-4** The MC68000 clock generation circuitry.

### ADDRESS AND DATA BUS CONNECTIONS 4-3

**The Address Bus**

Both the MC6800 and the MC6809 contain 16 pins that have been dedicated to addressing the memory and I/O. This feature allows either of these microprocessors to address 64K bytes of memory and I/O space directly. The MC68000 contains 23 address connections, which allow it to access an ascending 16M bytes of memory and I/O directly. This is equal to eight million 16-bit words of memory information. In addition to the number of address connections present, the amount of drive current available is triple that of the MC6800 or MC6809.

**The Data Bus**

The data bus of the MC6800 and MC6809 microprocessors is 8 bits in width, whereas the MC68000 uses a 16-bit data bus. This bus, in all three cases, is a bidirectional, three-state bus that passes information out of, or into, the microprocessor.

As with the address bus, the data bus on the MC68000 possesses an enhanced drive capability. This capability allows the microprocessor to be structured into a larger system before bus buffering is required.

**MC68000 Bus Buffing**

Figure 4-5 illustrates the inclusion of a set of data and address bus buffers for the MC68000 microprocessor. The AS or address strobe output is connected to the enable (G) input on the address buffers. The AS signal becomes a logic zero whenever the address bus contains a valid memory address. In this circuit, AS switches the three-state buffers to their enabled, or on, condition.

The bidirectional bus transceivers, which are connected to the data bus, are controlled by the R/W control signal. Since the R/W signal selects the direction of data flow on the data bus, it is usable as a directional control input to the transceivers. During a memory or I/O read, R/W is high and causes the data to flow in from the data bus; during a memory or I/O write, this line is low and causes the data to flow out to the memory and I/O.

### CONTROL BUS CONNECTIONS 4-4

The control bus structures of the MC6800, MC6809, and the MC68000 are almost identical when only the major control signals are examined. If the MC6800 and MC6809 are compared, they differ only in the way that direct memory access I/O is controlled. Comparing all three demonstrates many more differences. Table 4-1 contrasts these differences.

**The Basic Memory and I/O Control Signals**

All three microprocessors use the R/W signal to command the memory or I/O to read or write data. In the MC68000 this signal also works in conjunction
with the LDS and UDS data strobes, which indicate how the microprocessor will react with the data bus during the current bus cycle. Table 4-2 illustrates how the MC68000 interprets the data bus for each combination of the R/W, LDS, and UDS signals.

In addition to these three control signals, the VMA and E signals are also present on all three microprocessors. The VMA signal indicates that the address bus contains a "valid memory address"; the E signal, or enable, is used to enable the memory or I/O device. The E signal is actually not present on the MC68000, but it is the phase two TTL output of the clock generator circuitry.

Additional MC6809 Control Signals
The MC6809 has a few other control signals that are not present on the MC6800. These include BS, FIRQ, MRDY, and DMA/BREQ. The BA and BS signals indicate the present state of the MC6809, as illustrated in Table 4-3.

The MRDY signal extends the access time provided for the memory by extending the current read or write cycle. The extension may be anything from one clocking period up to 10 μs in duration. This is most useful if a slower external device is to be interfaced to the MC6809. An example is an analog-to-digital converter, as illustrated for the 8085A in Chapter 6.

The DMA/BREQ input is used for a direct memory access or bus arbitration; it is covered along with the FIRQ input, later in this chapter.

Additional MC68000 Control Signals
The advanced architecture of the MC68000 microprocessor includes some additional control pins, such as BGACK, VPA, DTACK, BERR, BR, BG.
IPL0, IPL1, IPL2, FC0, FC1, and FC2. These pins control such features as interrupts and direct memory access or bus arbitration. The FC0, FC1, and FC2 signals indicate the status of the MC68000 as depicted in Table 4-4.

<table>
<thead>
<tr>
<th>FC2</th>
<th>FC1</th>
<th>FC0</th>
<th>Cycle Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Undefined</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>User data</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>User program</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Undefined</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Undefined</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Supervisor data</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Supervisor program</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Interrupt acknowledge</td>
</tr>
</tbody>
</table>

FC0, FC1, and FC2 are basically used to indicate the mode of operation, either supervisor or user. In the supervisor state, the MC68000 can control an external memory management device and system software. This capacity provides security, since the memory management unit and system software cannot be accessed by the normal user. The access requires a shift to the supervisor state, which is a privileged state.

The BERR input signal informs the processor of a bus error and is provided by the external hardware. The type of hardware most likely to generate this signal is a memory parity checking circuit. If a parity error is detected, this signal becomes active, and the processor executes an exception sequence or an interrupt. This sequence reads the address of the user supplied bus error handling subroutine from memory address 800000. Control is then transferred to this error-handling subroutine for a possible repeat of the bus cycle.

The BR, BG, and BGACK signals are used when more than one MC68000 or a DMA controller is connected to a system. BR is an input that requests the use of the bus. BG is an output that indicates that the MC68000 will release bus control at the end of the current cycle. The BGACK input indicates that some other device has become the bus master. These signals are discussed in greater detail in the section on bus arbitration.

The VPA input is activated whenever an external MC68000 peripheral device is addressed. This is provided so that the wealth of MC68000 8-bit peripheral devices can function with the MC68000. It also signals the processor to use automatic vectoring for an interrupt, as described in the interrupt section of this chapter.

4-5 RESET OR RESTART

If the MC68000 or MC6809 microprocessors are reset, they look at memory location SFFFE for the restart vector. The restart vector holds the starting address of the system program.

BUS TIMING

Resetting or restarting the MC68000 is completely different because two vectors apply to this function. When the MC68000 is first powered up, locations zero through three must contain the supervisor stack pointer (SSP). Locations four through seven must contain the location of the first instruction to be executed after a reset. These vectors are used only during a power up sequence.

The RESET instruction in the MC68000 will not cause the reset vectors to be called. This instruction will only cause the RESET output pin to become active for 124 clocking periods after it has been executed. This instruction and the resulting signal on the RESET pin are only used for resetting the external peripheral components in the system. It has absolutely no effect on the internal registers of the MC68000.

If reprogramming the processor is desirable, it can be accomplished by using the reset vectors stored in the vector table.

The standard operating frequency for the MC68000 and the MC6809 is 1 MHz. At this rate they are capable of transferring 1 byte of information per clocking period or 1 byte every microsecond. The MC68000 works with an internal clock frequency of 8 MHz and can transfer 1 byte of data every 500 ns since the internal timing is set up so that four external clock pulses are required for a bus transfer.

MC68000 Read and Write Timing

Figure 4-6 illustrates the basic read and write timing diagrams of the MC68000 microprocessor and its AC characteristics. In the MC68000 timing diagrams, the address is presented to memory and I/O during the logic zero portion of the phase two clock. When the phase two clock becomes a logic one, data is transferred into the processor or sent out from it.

The time allowed for a memory access (Tacc) is equal to 540 ns worst case. In other words, the memory plus the time interval introduced by buffers should have an access time of no longer than 540 ns. In addition to this time constraint, it is also important to note that data must be held for 10 ns minimum after the phase two clock returns to the logic zero level. If the phase two clock is used as an enable (or E) signal, the amount of time required to enable the memory device must not exceed 350 ns. Since the output buffers in a memory device typically take 120 ns to enable, this is generally ample time.

MC68000 Memory Read and Write Signals

The circuit depicted in Figure 4-7 allows the MC6800 or MC6809 to be used with most of this text. It also allows it to be used, without effort, with most of the industrywide standard memory components, such as the 2114 RAM, 2716 EPROM, and others.

By combining the phase two TTL signal or E signal with the VMA output and the R/W signal, we obtain the MEMR or RD and MEMW or WR control signals that are used throughout this book. These pulses are approxi-
FIGURE 4-6 The read and write timing diagrams and characteristics of the MC6800 microprocessor.

SOURCE: Courtesy of Motorola, Inc.

FIGURE 4-6 continued
WRITE IN MEMORY OR PERIPHERALS

**BUS TIMING**

![Bus Timing Diagram](image)

* E is the phase 1 TTL CLOCK

**FIGURE 4-7** Using the MC6800 to generate the MEMR and MEMW control signals. They are approximately 500 ns in width and are compatible with many standard memory components. Since the 680XX series microprocessors do not support isolated I/O, no attempt has been made to develop the I/O control signal T0R and T0W. For I/O control and its application, refer to the section in chapter 6 on memory mapped I/O.

**MC68000 Read and Write Timing**

Figure 4-8 illustrates the timing diagrams for the MC68000 microprocessor. The MC68000 will transfer one word, or 16 bits, of information every 500 ns, since it operates at a basic clock frequency of 8 MHz. The amount of time allowed to the memory component attached to the MC68000 is approximately 300 ns. This means that higher-speed memory components must be selected for use with this processor.

**FIGURE 4-8** Basic read and write timing for the MC6800 microprocessor.

**SOURCE:** Courtesy of Motorola, Inc.
The AS, or address strobe, signal activates a memory component. It is normal to use this signal to supply the MC68000 with its DTACK signal in systems that contain memory that can access data within 350 ns.

The MC6809 MRDY and the MC68000 DTACK
The MRDY connection on the MC6809 prolongs the processor bus cycle for low-speed memory or I/O devices. This input can be held at a logic zero level for up to 10 μs for these slower devices. If held longer than 10 μs, Motorola will not guarantee the validity of the data stored in the MC6809 internal register array.

The DTACK input, or data acknowledge, of the MC68000 can serve about the same purpose as the MRDY input of the MC6809. The difference is that the MRDY input is an optional feature that can be ignored by connecting it to a logic one, while the DTACK input must be used.

During a read operation, for example, the MC68000 sends out the control signals and waits for the external device (usually memory) to send the DTACK signal back to the microprocessor. In fact, if the DTACK signal does not occur, the system waits just as it does with MRDY. Once the processor accepts the information, the DTACK signal must be returned to its inactive state before another bus cycle can occur. Without this timing, the MC68000 will not function.

### MC6809 AND MC68000 BUS ARBITRATION (DMA)

**MC6809 Bus Arbitration**
The MC6809 microprocessor has an input labeled DMA/BREQ that requests access to the MC6809 system bus. When this pin is active, the microprocessor releases control of the system bus by three-stating the address, data, and control buses. This allows an external device to access the memory and I/O connected to the MC6809 directly.

The BA and BS signals grant or acknowledge the bus request when they are both at logic one levels. This same level also indicates that the microprocessor may be halted.

**MC68000 Bus Arbitration**
If more than one microprocessor or similar device is to function on the same bus system, a need for bus arbitration arises. The set of connections described in this segment determines which device will control the bus so that no conflict can occur. Bus conflicts will almost always result in a loss of data if they are allowed to occur.

The BR, or bus request signal, is an input to the MC68000 that asks for or requests the system bus. If the MC68000 is at the end of its current bus cycle, it will grant the bus request by sending out the BG, or bus grant signal. Once the requesting device notices the BG signal, it returns a BGACK, or bus grant acknowledge signal, back to the MC68000 to indicate that it has taken over the system buses.

### INTERRUPT STRUCTURES

This arbitration dialog is normally carried out between the MC68000 and an external DMA controller. During the bus grant, the MC68000 relinquishes control of the system by floating the address, data, and control buses. This of course will allow the external device to gain complete control over the system buses. The typical three wire handshake is illustrated in the timing diagram of figure 4-9.

![Figure 4-9 MC68000 bus arbitration timing.](source: Courtesy of Motorola, Inc.)

**MC6800 Interrupt Structure**
The MC6800 microprocessor has two hardware and one software interrupt that are vectored through the top part of the memory. Table 4-5 illustrates the interrupt vectors for the MC6800 microprocessor. These vectors contain the location of the software that will be executed in response to these input signals.

<table>
<thead>
<tr>
<th>Vector Location</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFFFE, SFFFT</td>
<td>Reset</td>
</tr>
<tr>
<td>SFFFC, SFFFD</td>
<td>NMI</td>
</tr>
<tr>
<td>SFFT, SFFBD</td>
<td>SWI</td>
</tr>
<tr>
<td>SFFPA, SFFPS</td>
<td>IRQ</td>
</tr>
</tbody>
</table>

**MC6809 Interrupt Structure**
The MC6809 microprocessor has three hardware interrupts and three software interrupts that are vectored through the top portion of the memory. A new hardware interrupt labeled PIRQ, has been added to the IRQ and NMI inputs of the MC6800. The only difference between the new interrupt and the two old interrupts is that the PIRQ input will only store the program counter and the status register on the stack. The IRQ and NMI inputs place all of the internal registers, except the hardware stack pointer, on the stack.
Table 4-6 illustrates the vector locations for the interrupt inputs to the MC6809.

**MC6800 Interrupt Structure**

The interrupt structure for the MC6800 is quite different from the structure for the MC6800 and MC6809. A complete listing of the many different interrupts appears in Table 4-7.

<table>
<thead>
<tr>
<th>Vector Location</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFTE, SFTE</td>
<td>Reset</td>
</tr>
<tr>
<td>SFTEC, SFTEE</td>
<td>NMI</td>
</tr>
<tr>
<td>SFTEA, SFTEB</td>
<td>SW1</td>
</tr>
<tr>
<td>SFTEC, SFTEC</td>
<td>TRQ</td>
</tr>
<tr>
<td>SFTEA, SFTEC</td>
<td>PRIQ</td>
</tr>
<tr>
<td>SFTEF, SFTEG</td>
<td>SW13</td>
</tr>
<tr>
<td>SFTE2, SFTE3</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**MC6800 Interrupt Vectors**

<table>
<thead>
<tr>
<th>Vector Number</th>
<th>Address</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00000</td>
<td>Reset, initial SSP</td>
</tr>
<tr>
<td>1</td>
<td>00001</td>
<td>Reset, initial PC</td>
</tr>
<tr>
<td>2</td>
<td>00002</td>
<td>Bus error</td>
</tr>
<tr>
<td>3</td>
<td>00003</td>
<td>Address error</td>
</tr>
<tr>
<td>4</td>
<td>00010</td>
<td>Illegal instruction</td>
</tr>
<tr>
<td>5</td>
<td>00014</td>
<td>Divide by zero</td>
</tr>
<tr>
<td>6</td>
<td>00018</td>
<td>CHK instruction</td>
</tr>
<tr>
<td>7</td>
<td>0001C</td>
<td>TRAP instruction</td>
</tr>
<tr>
<td>8</td>
<td>00020</td>
<td>Privilege violation</td>
</tr>
<tr>
<td>9</td>
<td>00024</td>
<td>Trace</td>
</tr>
<tr>
<td>10</td>
<td>00028</td>
<td>Line 1010 emulator</td>
</tr>
<tr>
<td>11</td>
<td>0002C</td>
<td>Line 1111 emulator</td>
</tr>
<tr>
<td>12-23</td>
<td>00036-000AF</td>
<td>Reserved by Motorola</td>
</tr>
<tr>
<td>24</td>
<td>00060</td>
<td>Spurious interrupt</td>
</tr>
<tr>
<td>25</td>
<td>00064</td>
<td>Level 1 interrupt</td>
</tr>
<tr>
<td>26</td>
<td>00068</td>
<td>Level 2 interrupt</td>
</tr>
<tr>
<td>27</td>
<td>0006C</td>
<td>Level 3 interrupt</td>
</tr>
<tr>
<td>28</td>
<td>00070</td>
<td>Level 4 interrupt</td>
</tr>
<tr>
<td>29</td>
<td>00074</td>
<td>Level 5 interrupt</td>
</tr>
<tr>
<td>30</td>
<td>00078</td>
<td>Level 6 interrupt</td>
</tr>
<tr>
<td>31</td>
<td>0007C</td>
<td>Level 7 interrupt</td>
</tr>
<tr>
<td>32-37</td>
<td>00080-000BF</td>
<td>TRAP instruction vectors</td>
</tr>
<tr>
<td>38-63</td>
<td>00080-000FF</td>
<td>Reserved by Motorola</td>
</tr>
<tr>
<td>64-255</td>
<td>00100-001FF</td>
<td>USER interrupt vectors</td>
</tr>
</tbody>
</table>

This vector table occupies the first 1024 bytes of memory or first 512 words of memory. Seven of these vectors are used for external interrupts; the remaining vectors are used for reset, for various Motorola system functions, and for TRAPs. TRAPs are used by the system program to call up error handling routines; they may also be used as short form subroutine jumps if so desired. The trap number references a vector in the vector table that indicates the address of the TRAP subroutine.

External interrupts are caused by applying the interrupt device number, one through seven binary, on the three interrupt inputs IPI1, IPI2, and IPI3. Level seven has the highest priority, while level one has the lowest. A zero binary on these 3 pins indicates that no interrupt is being requested.

These interrupts reference the seven vectors listed in Table 4-7 if the VPA input is asserted. Notice that these vectors are only 1 byte in length. One-byte vector locations are normally used for interrupts and contain memory address 0000 0000 0000 0000 0000 0000 0000 0000, where XXXX XXXX is the vector stored at vector locations one through seven.

If desired, the external hardware may apply the interrupt vector location by not asserting the VPA input. If an external interrupt vector is supplied through the least significant 8 bits of the data bus, a vector to any of the 256 possible table entries can be used. This is useful if multiple interrupt processed I/O devices exist at each interrupt priority level.

Masking various interrupt levels is accomplished through the status register and the 3 bits assigned to perform this function. Interrupts are prohibited if the masks are the same priority level or greater than the currently requested interrupt level. The level seven interrupt cannot be inhibited or masked by the mask bits. It is equivalent to the NMI interrupt input on the MC6800 and MC6809.

**INSTRUCTION TIMING**

This section includes a list of the instructions and the number of clock cycles required to execute them. Only the MC6800 instructions are provided in this chapter. They are given to allow the student to calculate some of the time delays required for homework problems or outside development. The complete instruction set for the MC6800 is listed in Table 4-8. To calculate the amount of time required to execute an instruction, multiply the number of instruction cycles by 1 µs. This is, of course, for the standard 1 MHz version of the MC6800.

**THE MC6800 AND THE LOGIC ANALYZER**

The logic analyzer is an extremely useful device in microprocessor testing. In fact, it is the only device that can be used to view the timing of a microprocessor while it is functioning in a system. It is even possible to view the pro-
### TABLE 4-8  MC6800 instruction timing.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Dual Operated</th>
<th>ACCX</th>
<th>Immediate</th>
<th>Direct</th>
<th>Extended</th>
<th>Indirect</th>
<th>Relative</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABA</td>
<td></td>
<td></td>
<td>1 2 3 4 5</td>
<td>6 7 8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC</td>
<td>x</td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD</td>
<td>x</td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AND</td>
<td>x</td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASL</td>
<td>2</td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCC</td>
<td></td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCS</td>
<td></td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BFA</td>
<td></td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BGT</td>
<td></td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BHI</td>
<td></td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIT</td>
<td>x</td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLE</td>
<td></td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLS</td>
<td></td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BFE</td>
<td></td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMI</td>
<td></td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BNE</td>
<td></td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BPL</td>
<td></td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRA</td>
<td></td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSR</td>
<td></td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EOR</td>
<td></td>
<td>1 2 3 4 5</td>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:** Interrupt time is 12 cycles from the end of the instruction being executed, except following a WAI instruction. Then it is 4 cycles.

**SOURCE:** Courtesy of Motorola, Inc.

### Summary

This chapter provides a working knowledge of the hardware signals and major timing of the microprocessor: The MC6800, MC6809, and MC68020 Motorola microprocessors were covered in fair detail, but less comprehen-
sively than in the data sheets from the manufacturer. If more detail is required on any of these or other microprocessors manufactured by Motorola, please refer to The Complete Motorola Microcomputer Data Library published by Motorola.

This chapter, of course, does supply enough detail so that you can explore the wonders of these fantastic devices. This book also allows a glimpse of another microprocessor, if you happen to be studying the Intel 8085A microprocessor discussed in chapter 3. In fact, I would strongly urge the student to make a comparative analysis of both Intel and Motorola microprocessors by perusing both chapters.

Glossary

Access time The amount of time required by a memory component to access or retrieve information.

Bus arbitration An access technique used when more than one bus controller or microprocessor exists on the same memory and I/O bus structure.

Bus cycle Whenever information is moved out of or into the microprocessor through its bus.

Direct memory access (DMA) A computer's ability to store or retrieve information directly from the memory without the intervention of the microprocessor.

Instruction cycle Equal to one clocking period in the MC6800, MC6809, and MC68000.

Interrupt An I/O technique that allows a slower external I/O device to interrupt the instruction flow of the microprocessor. This is accomplished through a hardware subroutine jump.

Memory management A technique whereby available memory space can be increased to an unlimited amount.

Parity A technique used to check for the validity of data.

Pinout The pictorial view of an integrated circuit defining each pin connection.

Read cycle Whenever the microprocessor reads data from the memory or an I/O device.

Sink current The amount of current available at an output whenever that output is a logic zero.

Source current The amount of current available at an output whenever that output is a logic one.

Transceiver A digital device that can either drive a bus line or receive data from a bus line.

Vector A number stored in the memory that is used to point to another location in the memory.

Write cycle Whenever the microprocessor writes information to a memory or an I/O device.

Questions and Problems

1. List the number of pin connections on each of the following microprocessors: MC6800, MC6809, and MC68000.
2. How many TTL unit loads can the MC6800 or MC6809 microprocessor directly drive?
3. How many TTL unit loads can the MC68000 microprocessor directly drive? Explain your answer.
4. What is the noise immunity for the MC6800, MC6809, and MC68000?
5. What crystal frequency would be selected to operate the MC6800 at 1 MHz?
6. Which crystal frequency would be selected to operate the MC6809 at 1 MHz?
7. How many memory locations can the MC6800 or MC6809 directly address?
8. How many memory locations can the MC68000 directly address?
9. How many data bus connections are available on the MC6800 microprocessor?
10. Which MC6800 bus is a bidirectional bus?
11. What is the purpose of the AS pin on the MC68000?
12. What is the purpose of the EDS and UDS strobes on the MC68000?
13. The BERR signal on the MC68000 indicates which condition?
14. Which three signals control a DMA action on the MC68000 microprocessor?
15. Which signals control the DMA action of the MC6809 microprocessor?
16. Where must the RESET vector be stored in the MC6800 or MC6809 microprocessor?
17. Where must the RESET vector be stored in the MC68000 microprocessor?
18. How much time is allowed for memory access in a MC68080 based system?
19. How much time is allowed for memory access in a MC68000 based system?
20. Explain the operation of the circuit in figure 4-7.
21. What is the purpose of the DTACK signal in the MC68000 microprocessor?
22. List the types of interrupts available for the MC6800 microprocessor.
23. List the types of interrupts available for the MC6809 microprocessor.
24. List the types of interrupts available for the MC68000 microprocessor.
25. What is the difference between the FIRQ and the IRQ inputs on the MC6809?
26. Which MC68000 interrupt input level has the highest priority?
28 Given the following MC6800 program, determine how long it takes to execute if a 1 MHz clock is used.

```
LDAA #$10
LOOP DECA
BNE LOOP
```

29 The logic analyzer can monitor the instruction flow in a subroutine or a program. Write a short program to test I/O location SC000.

30 If you were to use the internal clock on the logic analyzer and you set it for a 1 μs sample rate, what would you view on the screen if the data bus were connected to the analyzer's data inputs?
This chapter collects all of the separate techniques that were learned throughout this textbook. Example problems that include memory interface, various forms of I/O interface, and digital communications have been illustrated. It is very important that the student go through each of the example problems for ideas on hardware and software implementation.

For more examples, see the end of this chapter, which contains a series of projects that illustrate many of the techniques discussed in this text.
DATA CONCENTRATOR

Data concentrators are used in data communications environments to pack many slow channels of digital data onto one high-speed channel. For example, a department store may have 20 point of sales terminals that must be connected to a computer in another city. Instead of leasing 20 telephone lines for the fairly intermittent data from these in-store terminals, a data concentrator can be connected between the POS terminals and the computer in the other city. This connection does not reduce the speed of the system as far as the user is concerned; it only reduces the total system cost by replacing the 20 leased lines with 1.

6800 Data Concentrator Example

In this example two low-speed channels are concentrated onto one high-speed channel for transmission to another system. The data on the low-speed channels is serial asynchronous data transmitted at 300 baud, and the data on the high-speed channel is asynchronous data transmitted at 4800 baud. For this example, we will only consider one way communications between the two terminals and the remote system.

Figure 12-1 illustrates the protocol between the concentrator and the host computer system. The data is preceded by an ID byte that indicates which terminal is transmitting the data. The ID byte is always followed by 15 bytes of information, allowing for a fairly efficient means of data transmission between each terminal and the remote computer system.

6800 Data Concentrator Hardware

The hardware for this application is pictured in the schematic of figure 12-2. The 6800 is surrounded by three 6850 ACIA#s that receive serial data from the terminals and transmit serial data to the remote computer system. In addition to the ACIA#s, a 128-byte RAM for data storage and a 1K-byte EPROM for program storage exist.

The decoder selects the EPROM for memory locations SFXXX, the RAM for locations SBXXX, and the ACIA#s for locations SCXXX, SCXXX, and SCXXX. Data channel one uses SBXXX; data channel two uses SCXXX; data channel is used as the link between the remote computer and the data concentrator.

Data Concentrator Initialization Dialog

In this system the three ACIA# must be initialized to start the communications between the terminals and the remote system. This dialog resides at the location pointed to by the reset vector in locations SFFFE and SFFFF.

FIGURE 12-1 The protocol for the MC6800 based data concentrator.

```
<table>
<thead>
<tr>
<th>ID byte</th>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
<th>Byte 4</th>
<th>Byte 5</th>
<th>Byte 6</th>
<th>Byte 7</th>
<th>Byte 8</th>
<th>Byte 9</th>
<th>Byte 10</th>
<th>Byte 11</th>
<th>Byte 12</th>
<th>Byte 13</th>
<th>Byte 14</th>
<th>Byte 15</th>
</tr>
</thead>
</table>
```
CHAPTER 12: MC6800 APPLICATION EXAMPLES

1 *DATA CONCENTRATOR INITIALIZATION DIALOG
2 *
3 START LDAA *#09 SETUP ACIA COMMAND WORD
4 STA A *#0000 PROGRAM THREE ACIAS
5 STA A *#0000
6 CLR *#0000 SETUP COMMAND FOR ACIA THREE
7 LDS *#0007F SETUP STACK AREA
8 LDX COUNT POINT TO BUFFERS: POINTER AND FLAGS
9 LOOP CLR X CLEAR BUFFERS: POINTERS AND FLAGS
10 DEX
11 BNE LOOP
12 CLR X
13 LDAA *#32 SETUP QUEUE TWO POINTERS
14 STAR IPNT2+1
15 STAR OPNTZ+1

(System software begins here)

The terminal ACIAS are programmed to divide the external clock source by 16, to transmit 7 data bits with even parity, and to send 1 stop bit. The high-speed ACIA is programmed with the same data format, except that its internal divider is setup to divide by 1. The resulting transmission speed is 4800 baud.

Data Storage for the Data Concentrator

The data storage consists of two separate buffer areas in the memory. These hold data as it comes from the two terminal devices and function as FIFOs, or queue memories.

16 *RAM STORAGE
17 *
18 BUF1 RMB 32 TERMINAL ONE BUFFER
19 BUF2 RMB 32 TERMINAL TWO BUFFER
20 IPNT1 RMB 02 QUEUE ONE POINTERS
21 OPNT1 RMB 02
22 IPNT2 RMB 02 QUEUE TWO POINTERS
23 OPNT2 RMB 02
24 FLG FCB 00 TRANSMIT FLAG
25 COUNT FCB 00 BYTE COUNTER

The queue pointers are all initialized for the empty condition; that is, both the input and output pointers are equal in value. A full condition is indicated when the IPNT is one less than the OPNT.

ACIA Status Scanning Software

The purpose of this software is to determine when an ACIA has received information or when it is ready to transmit information. This software immediately follows the system initialization dialog presented earlier. Figure 12-3 illustrates the flowchart of this software, which is the main program for the data concentrator.

26 *STATUS SCANNING SOFTWARE
27 *
28 SYST LDSB *#1 SET ACIA NUMBER
29 LDA A *#0000 GET STATUS ONE
30 JSR CHKI GO CHECK ACIA STATUS
31 INCB SET ACIA NUMBER
32 LDA A *#0000 GET STATUS TWO
33 JSR CHKI GO CHECK ACIA STATUS
34 LDA A *#0000 GET STATUS THREE
35 JSR CHKO GO CHECK ACIA STATUS
36 BRK SYST KEEP CHECKING

The software is looped through continually until a ready condition on any receiver is detected or a ready condition in the transmitter is detected. Once detected, data is transmitted or received by subroutines presented later in this text.

37 *SUBROUTINE TO CHECK ACIA RECEIVER STATUS
38 *
39 CHKI RARA RORF INTO CARRY
40 BCS READ
41 RTS IF NO DATA IN THE RECEIVER
The subroutine illustrated above checks the receiver of the selected ACIA to determine if it is ready with data. If it is not ready, a return from the subroutine occurs. If it is ready, the subroutine continues at location READ.

Reception Software

As the data is read from each terminal, it is stored in the terminal's queue, where it is held for later transmission by the high-speed ACIA. No attempt is made to detect a queue full condition, since this can never happen because of the speeds involved. Refer to figure 12-4 for a complete flowchart of this subroutine.

42 SUBROUTINE TO READ DATA FROM AN ACIA
43 "AND SAVE IT IN THE APPROPRIATE QUEUE"
44 *
45 READ CMPB #2 CHECK FOR ACIA 2
46 EEO READ2
47 LDAA #5000 GET DATA
48 LDAA #0000 READ3
50 READ2 LDX IPNT2 POINT TO QUEUE TWO
51 LDAA #1000 GET DATA
52 READ3 STAA X SAVE DATA
53 LSR UPDAT INCREMENT AND WRAP POINTER
54 STX IPNT2 SAVE IPNT2
55 RTS

FIGURE 12-4 The flowchart of the READ subroutine.

DATA CONCENTRATOR

Transmission Software

56 *SUBROUTINE TO CHECK TRANSMITTER STATUS
57 *
58 CHKO BITA #02 TEST DTE
59 BNE SEND
60 RTS

This short subroutine determines if the transmitter in the high-speed ACIA is ready for another byte of information. If it is not, a return from the subroutine occurs so that the remaining ACIAS can be tested.

The SEND subroutine transmits data to the remote system through the high-speed ACIA. The flowchart for this routine is pictured in figure 12-5, and the program itself follows.

61 SEND SUBROUTINE FOR TRANSMITTING DATA THROUGH
62 THE HIGH SPEED ACIA
63 *
64 SEND LDAA FLAG GET TRANSMITTER BUSY FLAG
65 ENE BUSY IF BUSY
66 LDAA IPNT1 GET IPNT1
67 CMPA DPNT1 COMPARE WITH OPNT1
68 BNE ST1 IF FIFO ONE IS NOT EMPTY
69 LDAA IPNT2 GET IPNT2
70 CMPA DPNT2 COMPARE WITH OPNT2
71 BNE ST2 IF FIFO TWO IS NOT EMPTY
72 RTS IF BOTH FIFOS ARE EMPTY

This portion of the SEND subroutine checks whether the transmitter is currently sending data; if it is not, it continues on to check whether data is available to transmit. If no data is present to transmit and the transmitter is not busy, it returns to scanning for input data through the two low-speed data channels.

73 *CONTINUATION OF SEND WHEN FIFOS ARE NOT EMPTY
74 *
75 ST1 LDAA #01 LOAD TERMINAL ID NUMBER
76 STA ST GO SEND IT
77 ST2 LDAA #02 LOAD TERMINAL ID NUMBER
78 ST STA #001 SEND TERMINAL ID NUMBER
79 STA FLAG SAVE TERMINAL NUMBER IN FLAG
80 LDAA #15 SETUP BYTE COUNTER
81 STA COUNT SAVE IN COUNTER
82 RTS CONTINUE SCANNING

If the transmitter is not busy but data is available to transmit, this portion of the software sends the terminal number through the high-speed ACIA and also sets a byte counter to 15. The byte counter contains the number of bytes that must follow the ID number. After transmitting the ID number, a return to scanning occurs so that additional data may be received.
CHAPTER 12: MC6850 APPLICATION EXAMPLES

FIGURE 12-5 The flowchart of the SEND subroutine.

DATA CONCENTRATOR

02 CONTINUATION OF SEND
06 *
05 BUSY LDAA FLAG CHECK TERMINAL NUMBER
06 TSTA #02 CHECK FOR TERMINAL TWO
07 BNE T1 GO TO TERMINAL ONE
08 LDAA IFNT2+1 GET IFNT2
09 CMPA IFNT2+1 COMPARE WITH IFNT2
10 BNE SEN2 GO SEND A BYTE FROM TWO
11 RTS CONTINUE SCANNING
12 T1 LDAA IFNT1+1 GET IFNT1
13 CMPA IFNT1+1 COMPARE WITH IFNT1
14 BNE SEN1 GO SEND A BYTE FROM ONE
15 RTS CONTINUE SCANNING

If the transmitter has been sending data, it arrives at this section of the software to determine whether any data has been received. If it has, a transfer occurs to either SEN1 or SEN2 to send the information. If it has not, control is returned, and it continues to search for more input data.

06 DATA TRANSMISSION PORTION OF SEND
07 *
06 SEN1 LDX OPNT1 GET OPNT1
07 LDAA X GET A BYTE OF DATA
08 STAA #0001 SEND THE DATA
09 JSE UPDAT INCREMENT AND WRAP POINTER
10 STX OPNT1 SAVE OPNT1
11 SENK DEC COUNT DECREMENT BYTE COUNT
12 BNE RTS RETURN FROM SUBROUTINE
13 CLR FLAG CLEAR BUSY FLAG
14 RETS RTS CONTINUE SCANNING
15 SEN2 LDX OPNT2 GET OPNT2
16 LDAA X GET DATA
17 STAA #0001 SEND DATA
18 JSE UPDAT INCREMENT AND WRAP POINTER
19 STX OPNT2 SAVE POINTER
20 BRA SENX FINISH UP

This software sends information through the high-speed ACIA and then decrements the byte counter. If the byte counter reaches zero, which indicates that all 15 bytes have been transferred, the FLAG is cleared so that the transmitter can start transmitting the next 15 bytes of data.

113 INCREMENT A POINTER AND WRAP IT IF NEEDED
114 *
115 UPDAT STX COUNT+1 SAVE TEMP
116 LDAA COUNT+2 GET POINTER
117 ANDA #0001 STRIP MOST SIGNIFICANT
118 PSHA SAVE IT
12-2 TRAFFIC LIGHT CONTROLLER

Traffic light control by microprocessors is becoming commonplace in many large cities because these units are easily adjusted for different timing sequences and can be controlled by an external computer system. External computer control has increased traffic flow during peak hours and reduced the number of accidents in the cities where it has been tested.

The system illustrated in this text receives its timing sequence through a keyboard located at the controlled intersection. The keyboard also enters the time of day and other information, such as the times the traffic light should flash. This system also includes a set of trip plates to trip the light for one direction.

Traffic Light Controller Hardware

The hardware for this controller includes an MC6821, which scans the keyboard and controls the traffic lamps. In addition to the MC6821, an oscillator is included to provide the MC6800 with its clock and to act as a timing source for the nonmaskable interrupt input (NMI). Also included is a trip plate sensor that causes an interrupt to occur whenever a vehicle is in proximity with the trip plate. The trip plate itself is a loop of wire located just below the surface of the roadway. When a vehicle sits over it, the metal in the vehicle changes the inductance of the loop, which can be sensed by the interface.

Figure 12-4 illustrates the MC6800 controller hardware, including the memory required and the appropriate device selection logic. The outputs of
the decoder select the 1K-byte EPROM at address $FXXX, the 128-byte RAM at address $0XXX, and the $0821 at address $8XXX.

RAM Storage Assignment:

1. CLOCK RMB 7
2. TIME RME 1
3. NSRED RME 1
4. NSGRE RME 1
5. NSREL RME 1
6. EWRD RMB 1
7. EWGRE RMB 1
8. EWRD RMB 1
9. EWGRE RMB 1
10. MAXTR RMB 1
11. MINTR RMB 1
12. FLEND RMB 0

The basic clock timer is allocated 7 bytes of memory: 6 to keep track of the time in hours, minutes, and seconds and 1 to divide the 10 Hz input signal into 1-second pulses. The time is kept in unpacked BCD form for ease in software development.

Memory location TIME is used as a down counter that is decremented once per second. TIME is used by the software to time a particular light and is in standard binary form.

The six locations for light timing, NSRED, NSGRE, and so on, are each programmable for times of up to 255 seconds, which should be more than enough time for a lamp in any direction.

Minimum trip time and maximum number of trips indicate how long a light may remain tripped and the minimum amount of time required to cause a trip. A typical minimum time may be 20 seconds, and a typical maximum number of trips may be five. This, of course, depends on the traffic flow pattern at the intersection.

In many cases it is normal to remove a light from service in the wee hours of the morning by programming the start and end flash times into the 12 bytes of memory allocated for this purpose.

Initialization Dialog

Since this is a programmable device, it must be initialized whenever power is applied or whenever a change in the sequence of the lights is to be affected. The dialog that follows is executed whenever the microprocessor is restarted. The initialization dialog programs the PIA and branches to the keyboard entry portion of the software.

15. *INITIALIZATION DIALOG
16.
17. RESET LDS #007F SET STACK AREA
18. LDA #FFD SETUP PORT B
19. STA #E002 CONFIGURE PORT B

Traffic Controller Setup

The controller must be programmed to function after a restart. Programming is accomplished through the keyboard and consists of entering the time of day, the duration of each light, trip times, and flash times.

Each one of these pieces of information must be entered without visual feedback, since this unit contains no display. A display is unnecessary because the sequence is relatively short and can be entered again if an error is detected.

An example programming sequence is illustrated in figure 12-7.

20. LDA #004 SELECT PERIPHERAL DATA REGISTERS
21. STA #E001 SEND TO PIA
22. STA #E003 SEND TO PIA

23. PORTION OF THE SYSTEM PROGRAM THAT SETS ALL OF THE PROGRAMMABLE FEATURES
24.
25. SETUP LDS CLOCK +1 POINT TO TIME OF DAY
26. JCR INTIM GET TIME OF DAY
27. CLR CLOCK CLEAR CLOCK
28. LDS NSRED POINT TO NSRED
29. JCR INSEC GET SECONDS COUNT FOR NSRED
30. JCR INSEC GET SECONDS COUNT FOR NSGRE
31. JCR INSEC GET SECONDS COUNT FOR NSREG
32. LDS NSGRE GET NSGRE
33. LDS NSREG GET NSREG
34. STA EWER DELAY EWER
35. STA EWSDE DEVELOP EWER
36. STA EWER SAVE EWER
37. LDS NSREG GET NSREG
38. STA EWER SAVE EWER
39. STA EWSDE DEVELOP EWER
40. LDS NSGRE GET NSGRE
41. LDS MAXTR POINT TO MAXTR
42. JCR INSEC GET COUNT FOR MAXTR
43. JCR INSEC GET SECONDS COUNT FOR MINTR
44. LDS FLSTR POINT TO FLSTR
45. JCR INTIM GET FLASH START TIME
46. JCR INTIM GET FLASH END TIME
47. LDS MAXTR CHECK FOR A TRIP PLATE
48. DRE SYST IF NO TRIP PLATE

FIGURE 12-7 The setup sequence that programs the traffic light controller.
CHAPTER 12: MC6800 APPLICATION EXAMPLES

47 SET CLI

ENABLE TRIP PLATE INTERRUPT

(Continues at SYST program)

This software accepts all of the programming data from the keyboard and stores it in the appropriate memory locations. See figure 12-8 for a flowchart. It also calculates the duration of each of the traffic lamps facing east-west and determines whether or not a trip plate is connected to the system. If MAXTRIP is a zero, it is assumed by the software that no trip plate is connected in the system, and the trip plate interrupt is left disabled.

INTIM Subroutine

The INTIM subroutine reads the time through the keyboard for the time of day, flash starting time, and flash ending time. It stores the time in the format HH:MM:SS. HH is a two digit number for hours, MM is for minutes, and SS for seconds. This six digit number is stored in six contiguous memory locations, which are indexed by the X register in unpacked BCD form.

48 SUBROUTINE TO SAVE THE TIME IN UNPACKED BCD FORM

49 *

50 INTIM LDAB $000 SETUP COUNTER

51 INTIS JSR INKEY GET A DIGIT

52 STAX X SAVE IT

53 INX POINT TO NEXT LOCATION

54 DECX DECREMENT COUNT

55 BNE INTIS REPEAT UNTIL SIX DIGITS

56 RTS RETURN FROM SUBROUTINE

INSEC Subroutine

The INSEC subroutine, illustrated in the flowchart of figure 12-9, accepts a three digit number from the keyboard and converts it from BCD to binary. It is then stored in the memory location that is indexed by the X register. INSEC is used to get and save data for the timing of the lights and the trip times, if required.

FIGURE 12-8 The flowchart of the SETUP portion of the traffic light controller system program.

FIGURE 12-9 The flowchart of the INSEC subroutine.
57  *INPUTS DATA IN BCD FROM THE KEYBOARD THEN
58  *CONVERTS IT TO BINARY AND SAVES IT
59  *
60  INSEC LDAB #002        SETUP COUNTER
61  CLR FLx+6              CLEAR TEMP
62  INTI1 LDAA FLx+6       MULTIPLY BY 10
63  ASLA                   DOUBLE ACC
64  STAA FLx+6             SAVE
65  ASLA                   SAVE
66  ASLA                   SAVE
67  ADDA FLx+6             ADD
68  STAA FLx+6             SAVE
69  JSR INKEY              GET DIGIT
70  ADDA FLx+6             CREATE BINARY NUMBER
71  STAA FLx+6             SAVE
72  DECB                   DECREMENT COUNT
73  ENE INTI1              REPEAT FOR THREE DIGITS
74  LDAA FLx+6             SET BCD DIGIT
75  STAA X                 SAVE IT
76  INX                    POINT TO NEXT
77  RTS                    RETURN FROM SUBROUTINE

This subroutine converts from BCD to binary by multiplying the previous binary number by ten and then adding in the new BCD digit. This will generate a binary number for a BCD number of up to 255. In example 12-1, a 103 is converted to binary using this algorithm.

**EXAMPLE 12-1**

\[
\begin{array}{c|c|c|c}
& \times 10 & 0000 & 0001 \\
First & -1 & 0000 & 0001 \\
& \times 10 & 0000 & 0001 \\
Second & 0 & 0000 & 1010 \\
& \times 10 & 0000 & 1010 \\
Third & 3 & 0110 & 0011 \\
\end{array}
\]

**RESULT** 0110 0111

**INKEY Subroutine**

The INKEY subroutine is used to retrieve information from the ten key numeric keypad interfaced to the MC6800 through a MC6821 PIA. This procedure is accomplished by using the basic INKEY subroutine that was discussed in chapter 7. A flowchart for this subroutine is depicted in figure 12-10.

78  *SUBROUTINE TO READ A CHARACTER FROM THE KEYBOARD
79  *
80  INKEY JSR CHECK       CHECK FOR A KEYSTROKE
81  BNE INKEY              IF A KEYSTROKE

**FIGURE 12-10** The flowchart of the INKEY subroutine.

82  JSR DELAY              DEBOUNCE
83  JSR CHECK              CHECK FOR A KEYPAD
84  BNE INKEY              IF A KEYPAD
85  INKEY1 CALL CHECK     CHECK FOR A KEYPAD
86  BNE INKEY1             IF NO KEYPAD
87  JSR DELAY              DEBOUNCE
88  JSR CHECK              CHECK FOR A KEYPAD
89  BEQ INKEY1             IF NO KEYPAD
90  PSRB                   STACK ACC 8
91  LDAB **FF              SETUP BCD CODE
92  LDAB $E000              GET 0 TO 7
93  INCA                   CHECK FOR ANY
94  BNE INKEY3             IF 0 THROUGH 7
95  LDAB $E07              IF 8 OR 9
96  LDAB $E002              GET 8 AND 9
97  INKEY3 INCB
CHAPTER 12: MC6800 APPLICATION EXAMPLES

98  ROR A  
99  BCS INKEY2  IF NOT FOUND
100 OUT TBA  GET BCD CODE
101 PUL B  RESTORE ACC B
102 RTS  RETURN FROM SUBROUTINE
103 CHECK LDA A #0000  GET 0 TO 7
104 INCA
105 BNE CHK1  GET A 0 TO 7
106 LDA A #002  SET B AND 9
107 ORA A #FC
108 INCA
109 CHK1  RTS  RETURN FROM SUBROUTINE
110 DELAY PSH B  SAVE ACC B
111 LDA B #14  CAUSE 10 MSEC, DELAY
112 CLRA
113 DECA
114 BNE DEL1
115 DECB
116 BNE DEL1
117 PUL B  RESTORE ACC B
118 RTS  RETURN FROM SUBROUTINE

Nonmaskable Interrupt Service Subroutine
This subroutine is used for keeping the correct time by modifying \textit{CLOCK}; it also, once per second, decrements whichever number happens to be in Location \textit{TIME}. This feature provides the traffic light controller with a real-time clock that not only contains the time of day but can also time events. Location \textit{TIME} is used as a timer and can time events in 1 second intervals. See figure 12-11 for a flowchart of the interrupt service subroutine.

119 *NONMASKABLE INTERRUPT SERVICE SUBROUTINE FOR
120 THE REAL TIME CLOCK
121 *
122 NMI INC CLOCA  SET DIVIDE BY TEN
123 LDA A #0A  CHECK FOR A TEN
124 CMP A CLOCA
125 BNE NM13  EXIT
126 CLR CLOCA  CLEAR COUNT
127 LDX TIME  POINT TO TIME
128 DECA  DECREMENT TIME
129 DLX  POINT TO SECONDS
130 JSR INCR  GO INCREMENT SECONDS
131 BNE NM13  RETURN FROM INTERRUPT
132 LDA A #05  SET WRAP
133 JSR INCR  GO INCREMENT TENS OF SECONDS
134 BNE NM13
135 LDA A #0A  SET WRAP
136 JSR INCR  GO INCREMENT MINUTES

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{traffic_light_controller.png}
\caption{The flowchart of the nonmaskable interrupt service subroutine.}
\end{figure}

137 BNE NM13
138 LDA A #06  SET WRAP
139 JSR INCR  GO INCREMENT TENS OF MINUTES
140 BNE NM13
141 DLX
142 LDA A X  GET TENS OF HOURS
143 LDA A #0A  SET WRAP
144 INX
145 CMP A #02  CHECK FOR 20 HOURS
The subroutine INCR has been developed to increment the count in the memory location indexed by the X register. If the count equals the number in ACC B, or wrap, the count is cleared. Number wrap indicates the modulus of the counter to the subroutine. A return with the CCR indicating an equal condition means that the next higher order digit of time must be incremented. If a return with the CCR indicating a not equal condition occurs, it means that no further counters need be updated.

Traffic Light System Software

The purpose of this segment of the software is to change the traffic lights. The software scans the times programmed into the controller and changes the indicator lamps at the appropriate time. Figure 12-12 illustrates the flowchart for the system software.

```
146 BNE NMI12  IF NOT 20 TO 23 HOURS
147 LDAB #04  SET WRAP
148 NMI12 JSR INCR  GO INCREMENT HOURS
149 BNE NMI13
150 LDAB #03
151 JSR INCR  GO INCREMENT TENS OF HOURS
152 NMI13 RTI  RETURN FROM INTERRUPT
153 INCR INC X  GET COUNTER
154 CMPB X  CHECK FOR A WRAP
155 BNE INCR1  IF NO WRAP AROUND
156 CLR X
157 INCR1 DEX
158 TST X
159 RTS  RETURN FROM SUBROUTINE

160 SYSTEM SOFTWARE
161 SYS1 LDAA #01  Synchronize with clock
162 STAA TIME
163 SYS1 TST TIME
164 SYS2 LAS #84  Set NS-Green, EW-Red
165 STAA #0002  Change Lights
166 LDAA NSGR
167 JSR TMO  Go Time Out Light
168 LAS #84  Set NS-Yellow, EW-Red
169 STAA #0002  Change Lights
170 LDAA NSYL
171 JSR TMO  Go Time Out Light
172 LDAA #30  Set NS-Red, EW-Green
173 STAA #0002  Change Lights
174 LDAA ENGR
175 JSR TMO  Go Time Out Light
176 LDAA ENBL
177 JSR TMO  Go Time Out Light
178 LDAA #28  Set NS-Red, EW-Yellow
```

**Figure 12-12** The flowchart of the main traffic light system program.
The only feature of the above software that may be a little difficult to understand is the very first portion. Line numbers 161 to 165 are used to synchronize the internal interrupt service clock with the software listed. If this is not accomplished, timing may be inaccurate by 1 second occasionally.

### Trip Plate Software

The trip plate interrupt service subroutine only takes effect if its maximum number of trips is programmed into the controller. If the maximum number is zero, the interrupt remains disabled, and the sequence illustrated in SYST takes complete control. The plate itself produces a pulse on the IRQ pin of the MC6800 every time that a vehicle rests on or crosses the plate.

This interrupt service subroutine must be able to determine if the east-west light is in the red condition; if it is, it must then determine how much time remains before the light changes to green. If this time is equal to or less than the minimum time for tripping, no action is taken.

Once tripped, the software must continue to trip the light for up to the maximum amount of time before changing back to red. This is accomplished by counting how many times the light has been tripped during the on cycle. A flowchart for this interrupt service subroutine is illustrated in Figure 12-13.

### Summary

This chapter applies the Motorola MC6800 series microprocessor in two fairly typical examples, a data concentrator and a traffic light controller. Both appli-
Suggested Projects

1. Develop the MC6800 hardware and software to implement a coin changer mechanism. It must be able to accept coins in any denomination from 1 cent to 50 cents and dispense change in the fewest number of pennies, nickels, and dimes.

   The amount of money to be accepted is programmed through a set of switches located inside the vending machine. The programmable amount can be anything from 1 cent to $1.99.

   Your software must accept coins until the amount indicated on the internal switches has been either reached or exceeded. If the amount has been exceeded, dispense the fewest number of coins as change and send an active low pulse out the VEND pin for 20 ms.

   As coins are inserted, a mechanical assembly sorts them and signals the microprocessor with a pulse indicating the denomination of the coin. Once your program has detected and remembered the coin, it must drop it into the internal coin box by pulsing the DROP pin for 100 ms. There is also a "bent coin" signal in case a defective coin is inserted into the machine. If a bent coin is detected, you must pulse the EJECT line for 120 ms to clear the coin slot.

   To dispense change, the appropriate CH control line is activated for 100 ms, dropping a coin out of the change slot of the vending machine. You must only return one coin at a time with a pause of at least 50 ms between coins for the mechanical ejection mechanism to function properly. Table 12-1 illustrates all of the TTL input and TTL output connections that are to be interfaced to the MC6800.

2. Develop an IC test fixture that will automatically test the 7490 TTL decade counter. The pinout of this decade counter is pictured in Figure 12-14 with a brief description of its operating characteristics.

   Your system must completely test this device. If it is found faulty, the red LED must be lit; if good, the green LED must be lit. The test sequence must test the counter's clear to zero, clear to nine, and count sequence of the counter at least 20 times without failure for a good indication. The test socket and two LED indicators are pictured in Figure 12-15.
3. If the above system is to be able to test any 14-pin integrated TTL circuit, which changes would have to be made to the hardware?

4. Create a darkroom timer that will control the length of time that the enlarger exposes the paper. The timer must be capable of exposing the paper in increments of 0.1 second up to 10 min.
   Time settings are dialed in on a series of rotary switches that are labeled in one-tenth seconds, seconds, and minutes (as illustrated in figure 12-16). The push button starts the timing sequence that applies AC power to the lamp in the enlarger for the preset amount of time.

5. Your neighbor's son is a cub scout and wants you to build a timer for the annual pinewood derby. The box must be able to determine who wins each heat and to display the winning time on a set of LED numeric readouts.
   Figure 12-17 pictures the ramp, which accommodates two cars at one time, and the location of the beginning and ending trip points.
   Your software should start timing when either of the first trip points is tripped and continue timing until either of the second trip points is tripped. The hardware should indicate who has won the race and should light up the elapsed time on a set of displays.

6. Modify the system developed in question 5 so that it can accommodate a four lane ramp.
KEYBOARDS

7-1

Keyboards are interfaced in two different ways in most microprocessor based systems. The first method uses a keyboard encoder that detects keystrokes, converts the keystrokes into ASCII code, and signals the microprocessor that information is available. The second method requires an input port and an output port, which are used with some software to multiplex the keys in a keyboard matrix.

![Keyboard Encoder Diagram](image)

**FIGURE 7-1** The pinout and block diagram of the AY-5-2376 keyboard encoder.

**SOURCE:** Courtesy of General Instruments, Inc.

**FIGURE 7-2** An AY-5-2376 interface to the 8155 through Port A using the strobed input mode of operation.

**KEYBOARDS**

**KEYBOARD Encoder**

The AY-5-2376, a typical keyboard encoder, is illustrated in Figure 7-1. The keys are attached to the encoder through an eleven by eight keyboard matrix, which allows 88 keys to be connected to the encoder.

The encoder, under normal operation, scans the keyboard matrix for a key closure. Once the closure is detected, the internal circuitry addresses a ROM, which provides the ASCII address of the key at the data output connections. This information is not considered valid until the AY-5-2376 generates the output strobe signal after time is allowed for the key switch to stop bouncing.

In addition to the 88 key switches connected in the keyboard matrix, 2 additional switch connections accomplish the shift and control functions. These additional inputs select different ASCII codes for the key switches. The internal ROM is a 256 word read only memory, which provides three sets of ASCII codes, depending upon the conditions of shift and control.

**KEYBOARD Encoder to 8155 Interface**

Figure 7-2 pictures the AY-5-2376 connected to an 8155 peripheral interface adapter. The strobe output, which becomes active after a valid keystroke, strobos the keyboard data into the I/O port for use by the microprocessor. Once the software detects this event, data is input to the microprocessor and
The I/O port is again ready for another byte of information from the keyboard.

Pins A, B, and C on the AY-5-2376 are used as timing inputs for an internal oscillator. This oscillator times the basic keyboard scanning rate. The SC pin connection develops a time delay internally, which debounces the keys on the keyboard.

The subroutine that is used to test the AY-5-2376 for data follows:

1. **THIS SUBROUTINE CHECKS FOR KEYBOARD DATA**
2. **IF DATA IS FOUND IT RETURNS WITH IT IN THE ACC**
3. **IF NO DATA IS FOUND IT WAITS FOR THE DATA**
4. **THE ACC AND FLAGS ARE DESTROYED**
5. **INKEY: IN STATUS IGET THE BUFFER FULL FLAG**
6. **ANI 02H IISOLATE ADF**
7. **JZ INKEY ILOOP IF THERE IS NO DATA**
8. **IN PORTA IINPUT THE ASCII DATA**
9. **RET IRETURN FROM THE SUBROUTINE**

Hexadecimal Keypad Interface

The keyboard encoder is only used when a full keyboard is connected to the microprocessor. Most applications do not require a complete keyboard, so this circuit is not found. In its place you would probably find the circuit of figure 7-3 with a small keyboard matrix of 16 keys.

For this interface to fit many different types of parallel interface adapters, the diagram identifies only port A and port B.

This keyboard is organized as a 2-by-8-bit matrix. Port A must be programmed as an input port, while the 2 bits used in port B must be programmed as outputs. This is accomplished with the initialization dialog discussed in the last chapter and with the subroutines for this circuit.

**FIGURE 7-3** A hexadecimal keypad connected to Port A of a PIA

---

**KEYBOARDS**

The subroutines that will scan the keyboard must be capable of selecting a column of eight keys, detecting if any of the eight keys is depressed, debouncing the keystroke, and providing a code to identify the key's location. The flowchart provided in figure 7-4 illustrates this sequence of events.

**8085A Keypad Software**

When developing the software for this application, binary bit patterns 0000 0010 and 0000 0001 are chosen as codes to select the columns, and binary bit patterns 0000 0000 and 0000 1000 are chosen as an indicator for the first key in the selected column.

The time required for debouncing the keys depends upon the type of push button switches selected for the keyboard. In general, push button switches will stop bouncing after 10-20 ms.

The 8155 is initialized by programming the command register so that port A is an input port and port B is an output port. The initialization dialog is placed at the start of the system software at the reset location.

**FIGURE 7-4** The flowchart for scanning the keyboard illustrated in figure 7-3.
CHAPTER 7 INPUT/OUTPUT SYSTEMS

After the 8155 is initialized, it can be controlled to scan the keyboard. The INKEY subroutine that scans this keyboard follows:

1: 8085A ASSEMBLY LANGUAGE VERSION
2: SUBROUTINE TO DETECT A KEYSTROKE AND RETURN
3: WITH THE KEY CODE IN THE C-REGISTER.
4: i
5: CALL ALL REGISTERS EXCEPT HL ARE DESTROYED
6: USE THE SCAN AND DELAY SUBROUTINES
7:
8: ICHECK FOR KEY RELEASE
9: INKEY: CALL SCAN ICHECK ALL KEYS
10: JNZ INKEY IIF KEY IS DEPRESSED
11: CALL DELAY IODEBOUNSE
12: CALL SCAN ICHECK ALL KEYS
13: JNZ INKEY IIF KEY IS DEPRESSED
14: ICHECK FOR A KEY
15: LOOP: CALL SCAN ICHECK ALL KEYS
16: JZ LOOP IIF NO KEY IS DEPRESSED
17: CALL DELAY IODEBOUNSE
18: CALL SCAN ICHECK ALL KEYS
19: JZ LOOP IIF IT WAS NOISE
20: IDETERMINE WHICH KEY WAS DEPRESSED
21: LOOP1: PRC ILOCATE ROW
22: RNZ IRETURN IF FOUND
23: INR D I MODIFY KEY CODE
24: JMP LOOP1 CONTINUE TO LOOK

Lines 9 through 13 in the 8085A version of the keyboard software check whether the previous key has been released. This check is necessary because the software that uses this subroutine may call it before the person using the keyboard has had time to remove a finger from the button. If the key is released, lines 15 through 19 scan, or search, for another key closure. Once a key closure is detected, the subroutine searches the binary bit pattern for the closed contact; as it does, it modifies the key code in the C-register. When the code of the keystroke has finally been calculated, a return occurs with C equal to the key's code number.

24: 120 MSEC. TIME DELAY SUBROUTINE
25: 1CLOCK CYCLE TIME = 333 MSEC.
26: 1ACC, F, G, AND E ARE DESTROYED
27:
28: DELAY: LXI B, 166BD ILOAD COUNT
29: DELAY1: DEC D IINCREMENT COUNT
30: MOV A, D ITEST DE FOR A ZERO
31: ORA E
32: JNZ DELAY1 ICOUNT ≠ ZERO?
33: RET

KEYBOARDS

The amount of time used for the contact de-bounce delay is left up to the user, since it varies with different switches. The count 1508 in the DELAY subroutine is chosen for a 20 ms time delay for this example.

34: 1KEYBOARD SCANNING SUBROUTINE
35: 1MODIFIES B AND C; DESTROYS ACC AND F
36: 1RETURN ZERO = NO KEYSTROKE
37: 1RETURN NOT ZERO = KEYSTROKE
38: 
39: SCAN: MOV A, 02H ISELECT A COLUMN
40: MOV C, 00H ISET ROW STARTING KEY CODE
41: OUT PORTB
42: IN PORTA ICHECK ROWS
43: CPI OFFH
44: RNZ IRETURN ON KEY
45: MOV A, 01H ISELECT NEXT COLUMN
46: OUT PORTB
47: MOV C, 08H ISET ROW STARTING KEY CODE
48: IN PORTA ICHECK ROWS
49: CPI OFFH
50: RET

The keyboard scanning subroutine selects a column by modifying the data at port B. Once a column of eight keys is selected, port A is input and checked for a keystroke. If 1 or more bits are logic zeros at this time, it indicates that a key is depressed and the subroutine returns with the accumulator containing the raw bit pattern. If no key is depressed, the column selection bit pattern and the row beginning key code are modified and the next column of eight keys is checked.

6600 Keypad Software

To implement the hex keypad with the MC6860 and MC6821 PLA, the PLA must first be programmed or initialized at the reset location for the system program. The dialog that follows programs port A as an input port and port B as an output port.

1: *6821 HEX KEYPAD INITIALIZATION
2: 
3: RESET CLR CRA SELECT PORT A DDR
4: CLR PORTA PORT A = INPUT
5: CLR CRB SELECT PORT B DDR
6: LDAA #1FF PORT B = OUTPUT
7: STA PORTB
8: LDD #00D
9: STA CRA SELECT PORT A DATA REGISTER
10: STA CRB SELECT PORT B DATA REGISTER
11: 
12: 
13: 
14: 
15: 
16: 
17: 
18: 
19: 
20: 
21: 
22: 
23: 
24: 
25: 
26: 
27: 
28: 
29: 
30: 
31: 
32: 
33: 
34: 
35: 
36: 
37: 
38: 
39: 
40: 
41: 
42: 
43: 
44: 
45: 
46: 
47: 
48: 
49: 
50: 
51: 
52: 
53: 
54: 
55: 
56: 
57: 
58: 
59: 
60: 
61: 
62: 
63: 
64: 
65: 
66: 
67: 
68: 
69: 
70: 
71: 
72: 
73: 
74: 
75: 
76: 
77: 
78: 
79: 
80: 
81: 
82: 
83: 
84: 
85: 
86: 
87: 
88: 
89: 
90: 
91: 
92: 
93: 
94: 
95: 
96: 
97: 
98: 
99: 
100: 
101: 
102: 
103: 
104: 
105: 
106: 
107: 
108: 
109: 
110: 
111: 
112: 
113: 
114: 
115: 
116: 
117: 
118: 
119: 
120: 
121: 
122: 
123: 
124: 
125: 
126: 
127: 
128: 
129: 
130: 
131: 
132: 
133: 
134: 
135: 
136: 
137: 
138: 
139: 
140: 
141: 
142: 
143: 
144: 
145: 
146: 
147: 
148: 
149: 
150: 
151: 
152: 
153: 
154: 
155: 
156: 
157: 
158: 
159: 
160: 
161: 
162: 
163: 
164: 
165: 
166: 
167: 
168: 
169: 
170: 
171: 
172: 
173: 
174: 
175: 
176: 
177: 
178: 
179: 
180: 
181: 
182: 
183: 
184: 
185: 
186: 
187: 
The keypad scanning subroutine, which follows, checks to see whether a key is released. This is done because the software jumping to this subroutine may execute in a very short period of time. If it jumps to the subroutine before the operator releases the key, multiple keystrokes are entered into the system. Once the key is released, the INKEY subroutine detects which key has been pressed and returns with the code of the key in accumulator B.

1. *8000 ASSEMBLY LANGUAGE VERSION
2. *SUBROUTINE TO DETECT A KEYSTROKE AND RETURN
3. *WITH THE KEY CODE IN ACCUMULATOR B.
4. *
5. *WAIT FOR KEY RELEASE
6. INKEY JSR SCAN CHECK ALL KEYS
7. BNE INKEY IF KEY IS DEPRESSED
8. JSR DELAY DEBOUNCING KEY
9. JSR SCAN CHECK ALL KEYS
10. BNE INKEY IF KEY IS DEPRESSED
11. *WAIT FOR A NEW KEYSTROKE
12. LOOP JSR SCAN CHECK ALL KEYS
13. BNE LOOP IF NO KEY DEPRESSED
14. JSR DELAY DEBOUNCING
15. JSR SCAN CHECK ALL KEYS
16. BNE LOOP IF NO KEY DEPRESSED
17. *DETERMINE KEY CODE
18. LOOP1 BSF LOCATE KEYSTROKE
19. BCC RET RETURN WHEN FOUND
20. INCB MODIFY KEY CODE
21. JMP LOOP1 KEEP CHECKING

The time delay subroutine uses nested loops to achieve a time delay of 20 ms. This time delay is required to debounce the mechanical key switches in the keyboard matrix.

22. *20 MSEC. TIME DELAY SUBROUTINE
23. *
24. DELAY LOAD *B14 LOAD COUNT
25. DELAY1 LOAD *B15
26. DELAY2 DEC B DECIMUM B COUNT
27. BNE DELAY2 COUNT B = ZERO
28. DECA DECIMUM COUNT A = COUNT A
29. BNE DELAY1 COUNT A = ZERO
30. RTS RETURN FROM DELAY

The SCAN subroutine selects a column of eight keys and determines whether or not a key is depressed. If a key is detected, a return equal occurs; if no key is detected, a return not equal occurs.

31. *CHECK FOR ANY KEY SUBROUTINE
32. *RETURN EQUAL = KEYSTROKE DETECTED
33. *RETURN NOT EQUAL = NO KEYSTROKE DETECTED
34. *
35. SCAN LDA #02 SELECT COLUMN

MULTIPLEXED DISPLAYS

Display devices are normally multiplexed to reduce the component count in a microcomputer-based system. In microprocessors, the seven segment code is developed with software to further reduce the amount of external hardware required in the system.

BCD to Seven Segment Code Conversion

Code conversion from binary coded decimal to seven segment code is usually done via a table lookup subroutine. The BCD coded number forms the address of the seven segment coded character stored in a table in the memory. This method of code conversion is widely used because of its speed and relatively low cost. Table 7-1 illustrates the typical lookup table for a common anode seven segment display. The display and driver circuitry is pictured in figure 7-5. When a logic one is applied to the base of the segment driver, it becomes forward biased. This sinks current for the cathode of the display, which then lights.

FIGURE 7-5 A seven segment LED display illustrating the segment drivers.
TABLE 7-1 Common anode seven segment lookup table.

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
<th>Displayed Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE</td>
<td>011111110</td>
<td>0</td>
</tr>
<tr>
<td>TABLE-1</td>
<td>001110000</td>
<td>1</td>
</tr>
<tr>
<td>TABLE-2</td>
<td>011011010</td>
<td>2</td>
</tr>
<tr>
<td>TABLE-3</td>
<td>011111001</td>
<td>3</td>
</tr>
<tr>
<td>TABLE-4</td>
<td>011001011</td>
<td>4</td>
</tr>
<tr>
<td>TABLE-5</td>
<td>010111011</td>
<td>5</td>
</tr>
<tr>
<td>TABLE-6</td>
<td>011100000</td>
<td>6</td>
</tr>
<tr>
<td>TABLE-7</td>
<td>011111000</td>
<td>7</td>
</tr>
<tr>
<td>TABLE-8</td>
<td>011111111</td>
<td>8</td>
</tr>
<tr>
<td>TABLE-9</td>
<td>011111011</td>
<td>9</td>
</tr>
</tbody>
</table>

68085A Table Lookup Software
Software to convert the unpacked or single BCD digit in the accumulator of an 8085A into a seven segment coded number follows:

1: 8085A ASSEMBLY LANGUAGE PROGRAM
2: SUBROUTINE TO CONVERT THE ACCUMULATOR FROM BCD TO SEVEN SEGMENT CODE
3: BCD TO SEVEN SEGMENT CODE
4: HL IS DESTROYED
5: REFERENCES TABLE 7-1
6: i
7: CONV: ANDE #0FH IMPART LEFT NIBBLE
8: LDA A, TBLADD POINT TO LOOKUP TABLE
9: ADD L, H ADD BCD TO ADDRESS (HL)
10: MOV L,A
11: MOV W,A
12: ADD 00H
13: MOV H,A
14: MOV A,W GET SEVEN SEGMENT CODE
15: RET

6800 Table Lookup Software
Software to convert the contents of accumulator B in the MC6800 from a single unpacked BCD digit into seven segment code follows:

1: 6800 ASSEMBLY LANGUAGE PROGRAM
2: SUBROUTINE TO CONVERT ACCUMULATOR B FROM BCD
3: INTO SEVEN SEGMENT CODE
4: X IS DESTROYED
5: REFERENCES TABLE 7-1
6: i
7: CONV: ANDE #0FH MASK LEFT NIBBLE
8: LDX #TABLE GET TABLE ADDRESS
9: LDX #TABLE GET TABLE ADDRESS
10: STX TEMP SAVE TABLE ADDRESS
11: ADDB TEMP+1 DEVELOP ADDRESS
12: STAB TEMP
13: INC TEMP
14: CONV2 LDX TEMP GET TABLE ADDRESS
15: LDAD X GET 7 SEGMENT CODE
16: RTS

Location TEMP in the above software is 2 bytes of memory somewhere in the base page. This reduces the length of the subroutine. The extra work allows this subroutine to be stored in a ROM. If a ROM will not be used, the subroutine can be shortened considerably.

Multiple Digit Display
The table lookup technique for code conversion, along with other software, multiplexes the two digit display illustrated in figure 7-6. Port A supplies both displays with seven segment code through a set of drivers, and port B selects either the zero or digit one. Again the type of peripheral interface adapter is not specified, so that any can be utilized.

Port A provides seven segment data for both displays through a set of current amplifiers. These amplifiers are required to provide enough drive current for the displays, which typically require 10 mA per segment. Since this is a two digit multiplexed display, each display segment requires twice this amount of current to remain illuminated at normal intensity. A three digit display requires three times the current, and so forth.

FIGURE 7-6 A two digit multiplexed seven segment LED display.
The software developed to drive the displays will make one pass only; that is, it will display each digit only one time. It is the responsibility of the software using this subroutine to call it continually to maintain a constantly displayed number. If you wish to do quite a bit of processing between calls, it is important to blank the displays to prevent damage. The displays may be blanked by turning off both displays.

Figure 7-7 illustrates the flowchart for the DISP subroutine. Port B selects the digit that displays the information at port A. The two 'digit' selection pins at port B are connected to transistor switches that select a digit. These switches must be capable of passing the current from all seven segments in the selected display. In this circuit that amounts to 140 mA peak for each seven segment display, with an average current of 70 mA.

The subroutine that causes the 1 ms time delay is not illustrated but can be developed in the same manner as the DELAY subroutine in the section on keyboards. The DELAY subroutine is included to reduce the switching time to the displays. Without it, RF is generated and propagated from the displays, causing a problem with the Federal Communications Commission (FCC).

8085 Version of the Display Software

Before the display can be used, the 8155 must be programmed. In this application, ports A and B must be programmed as output ports for the displays. As with the prior software, the initialization dialog is found at the reset location.

1: 8155 INITIALIZATION DIALOG
2:
3  RESET: MVI A,00000016 PROGRAM PORT A & B
4  OUT COMMAND AS OUTPUT PORTS

Once the 8155 is programmed, the DISP subroutine can be used whenever data is to be displayed on the two digit display.

1: 8085 ASSEMBLY LANGUAGE PROGRAM
2: SUBROUTINE TO DISPLAY THE PACKED BCD NUMBER
3: IN THE ACCUMULATOR ON THE TWO DIGIT DISPLAY.
4:
5  DISP: PUSH PSW ISAVE BCD
6  CALL CONVERT ICONVERT TO SEVEN SEG
7  OUT PORTA ISEND DATA
8  MVI A,01H ISELECT DIGIT 0
9  OUT PORTB
10  CALL DELAY IWAIT 1 MSEC.
11  POP PSW IGET BCD
12  RRC IPOSITION NEXT DIGIT
13  RRC
14  RRC
15  RRC
16  CALL CONVERT ICONVERT TO SEVEN SEG
17  OUT PORTA ISEND DATA
18  MVI A,01H ISELECT DIGIT 1
19  OUT PORTB
20  CALL DELAY IWAIT 1 MSEC.
21  RET IRETURN FROM DISP
6800 Version of the Display Software

Before the display can be used, the MC6821 must be programmed. In this application ports A and B must be programmed as output ports for the display. As with the prior software, the initialization dialog is found at the reset location. Steps 3 and 4 are only required if the MC6821 is not reset. This may be the case in some systems; so it may be better to include these steps as a matter of practice.

```
*6821 INITIALIZATION DIALOG
*
3 RESET CLR CRA SELECT DDR PORT A
4 CLR CRB SELECT DDR PORT B
5 LDAA #$FF SET ALL BITS TO OUTPUT
6 STAA DDRA PROGRAM PORT A
7 STAA DDRB PROGRAM PORT B
8 LDAA #$04 SELECT DATA FOR PORT A
9 STAA CRA
10 STAA CRB SELECT DATA FOR PORT B
```

After the MC6821 is programmed, the DISP subroutine can be used whenever data is to be displayed on the two digit display.

```
*6800 ASSEMBLY LANGUAGE PROGRAM
*SUBROUTINE THAT TAKES THE PACKED BCD FROM
*ACC B AND DISPLAYS IT ON THE DISPLAYS.
*
5 DISP PSHB SAVE BCD DATA
6 JSR CONVERT CONVERT TO SEVEN SEG.
7 STAB PORTA SEND DATA
8 LDAB #$02 SELECT DIGIT 0
9 STAB PORTB
10 JSR DELAY WAIT FOR 1 MSEC.
11 PULB RESTORE BCD
12 LSRB POSITION NEXT DIGIT
13 LSRB
14 LSRB
15 LSRB
16 JSR CONVERT CONVERT TO SEVEN SEG.
17 STAB PORTA SEND DATA
18 LDAB #$01 SELECT DIGIT 1
19 STAB PORTB
20 JSR DELAY WAIT FOR 1 MSEC.
21 RTS RETURN FROM DISP
```
tive for any application requiring this type of sensitivity. The RST 6.5, RST 5.5, and INTR inputs are level sensitive; they must be held at their active levels until they are recognized at the end of the current instruction. The time required to recognize these three inputs varies with different instructions and clock speeds of the 8085A. It is also important to note that the HOLD input causes an interrupt to be delayed until after the HOLD condition has ended.

The INTR Input and INTA Output
The INTR input does not call an interrupt service subroutine directly. Instead the 8085A issues an INTA pulse when this input is acknowledged, as illustrated in figure 8-3. It is the designer's responsibility to add hardware that will force an instruction onto the data bus in response to the INTA output of the 8085A. For most applications, a RST 1 through RST 7 is forced onto the data bus; on occasion, a CALL instruction is. (Note that the RST 0 instruction is normally used for a software and hardware RESET.) Figure 8-4 pictures the application of a RST 5 in response to an INTR interrupt request. The RST 5 instruction, an EFH, is hardwired to the inputs of the eight three-state buffers. Whenever the INTR input is placed at the logic one level requesting an interrupt, the microprocessor responds with an INTA pulse. This procedure enables the buffers and applies the EFH or RST 5 op-code on the data bus. The microprocessor responds by executing the RST 5 or it calls the subroutine that begins at memory location 28H.

8-3 MC6800 AND MC6809 INTERRUPT STRUCTURE

The MC6809 has three interrupt inputs: one is a nonmaskable interrupt input, NMI, and the others are maskable interrupts, IRQ and FIRQ. The MC6800 has all the same inputs except the FIRQ. The NMI input causes the MC6809 to look at memory locations SFF8 and SFF9 for the address of the interrupt service subroutine. The IRQ input uses SFF8 and SFF9, and the FIRQ input uses SFF6 and SFF7 for their service subroutine vectors.

When the interrupt input is accepted by the MC6809 or MC6800, it automatically saves the contents of all internal registers on the stack and looks to the appropriate interrupt vector for the starting location of the interrupt service subroutine. The exception to this rule is the FIRQ, or fast interrupt request input, which only saves the contents of the program counter and condition code register.

At the end of the interrupt service subroutine, a special return instruction (RTI) reloads the registers saved on the stack and returns to the program that was interrupted. This return instruction is different from RTS, which does not restore any register but the program counter. An extra flag bit in the status register indicates whether the interrupt is a FIRQ or normal interrupt for the MC6809. This is looked at by RTI to determine which registers must be unloaded from the stack.

In the MC6800 the IRQ interrupt input is enabled by the CLI instruction and disabled by the SEI instruction. These instructions control the interrupt
enable bit (I) in the condition code register, which in turn controls whether or not the interrupt is accepted by the microprocessor.

In the MC6809, the (I) and (F) interrupt masks are controlled by the ORCC instruction, which sets or disables them, and the ANDCC instruction, which clears or enables these bits. The (F) condition code bit controls the FIRQ input, and the (I) condition code bit controls the IRQ input.