SECTION 8
THE R6522 VERSATILE INTERFACE ADAPTER

8.1 MICROCOMPUTER INPUT/OUTPUT SECTIONS

Input/output is a major consideration in almost all microcomputer applications. The greatest difficulty is that each input and output device has unique features. Devices differ in the following ways:

1. Data rates
2. Codes
3. Response time
4. Data format, e.g., serial or parallel
5. Control signals required to synchronize transfers and determine operating modes
6. Status signals that reflect the progress of transfers and the state of the device
7. Error detection and correction facilities

Thus frequently the I/O section of a microcomputer is more complicated and more expensive than all the other sections. It is also different for each application. The I/O section

may contain latches, flip-flops, buffers, drivers, shift registers, counters, timers, decoders, and multiplexers. These components may easily occupy one or several circuit boards.

Even then, a complex I/O section may not be well-suited to a given application. It may not offer the required numbers of inputs and outputs, the proper signal polarities, or the desired combinations of status and control signals. Furthermore, a complex I/O board requires extra space and power and reduces system reliability.

An alternative is to build an I/O section on a chip. This chip should contain a variety of I/O circuit elements and connections with the precise choice of features determined by the contents of accessible registers. The user then, has the equivalent of a designer's casebook at his or her command. The program simply has to establish the values of the various registers and thus choose the desired logic connections.

Typical choices include:

1. Whether ports (or individual pins) are to be inputs or outputs.

2. The polarity of signals (i.e., active-high or active-low) and active transitions (i.e., rising edge or falling edge).

3. Polling or interrupt-driven modes of operation.

4. The sequence of status and control signals.

5. Methods for deactivating status and control signals.

Note the many advantages of this approach:

1. One board can handle a variety of applications.

2. Changes and corrections can be made in software rather than in hardware.

3. The chip will take less room, dissipate less power, and be more reliable than a circuit board full of components.

4. The chip can include all the circuitry required to communicate with the microprocessor.

The problems for the user are to learn:

1. What features the chip offers.

2. How a particular set of logic connections can be implemented.

3. How to take full advantage of the chip's capabilities.

8.2 FEATURES OF THE VERSATILE INTERFACE ADAPTER

The R6522 Versatile Interface Adapter used in the AIM 65 microcomputer is an example of an entire I/O section on a chip. It contains the following:

1. Two 8-bit I/O ports (A and B). Each pin can be individually selected to be either an input or an output.

2. Four status and control lines (two associated with each port).

3. Two 16-bit counter/timers which can be used to generate or count pulses. These timers can produce single pulses or a continuous series of pulses.

4. An 8-bit shift register which can convert data between serial and parallel forms.

5. Interrupt logic so that I/O can proceed on an interrupt-driven basis. This logic includes an interrupt flag register that tells whether particular interrupts have occurred and an interrupt enable register which determines whether particular interrupts are allowed.

The Versatile Interface Adapter occupies 16 memory locations as shown in Table 8-1. The addresses are those of the user VIA on the AIM 65 board. The way that it operates is determined by the contents of 4 registers.

1. Data Direction Register A (DDRA) determines whether the pins on port A are inputs or outputs.

2. Data Direction Register B (DDRB) determines whether the pins on port B are inputs or outputs.

3. The Peripheral Control Register (PCR) determines which polarity of transition (i.e., rising edge or falling

edge) is recognized on the input status lines (CA1 and CB1) and how the other status lines (CA2 and CB2) operate.

4. The Auxiliary Control Register (ACR) determines whether the data ports are latched and how the timers and shift register operate.

Table 8-1. R6522 Memory Assignments

Location	Function		
A000	Port B Output Data Register (ORB)		
A001	Port A Output Data Register (ORA)		**Controls handshake**
A002	Port B Data Direction Register (DDRB)		0 = Input
A003	Port A Data Direction Register (DDRA)		1 = Output
	Timer	**R/$\overline{W}$ = L**	**R/$\overline{W}$ = H**
A004	T1	Write T1L-L	Read T1C-L Clear T1 Interrupt Flag
A005	T1	Write T1L-H & T1C-H T1L-L → T1C-L Clear T1 Interrupt Flag	Read T1C-H
A006	T1	Write T1L-L	Read T1L-L
A007	T1	Write T1L-H Clear T1 Interrupt Flag	Read T1L-H
A008	T2	Write T2L-L	Read T2C-L Clear T2 Interrupt Flag
A009	T2	Write T2C-H T2L-L → T2C-L Clear T2 Interrupt Flag	Read T2C-H
A00A	Shift Register (SR)		
A00B	Auxiliary Control Register (ACR)		
A00C	Peripheral Control Register (PCR)		
A00D	Interrupt Flag Register (IFR)		
A00E	Interrupt Enable Register (IER)		
A00F	Port A Output Data Register (ORA)		***No effect on handshake***

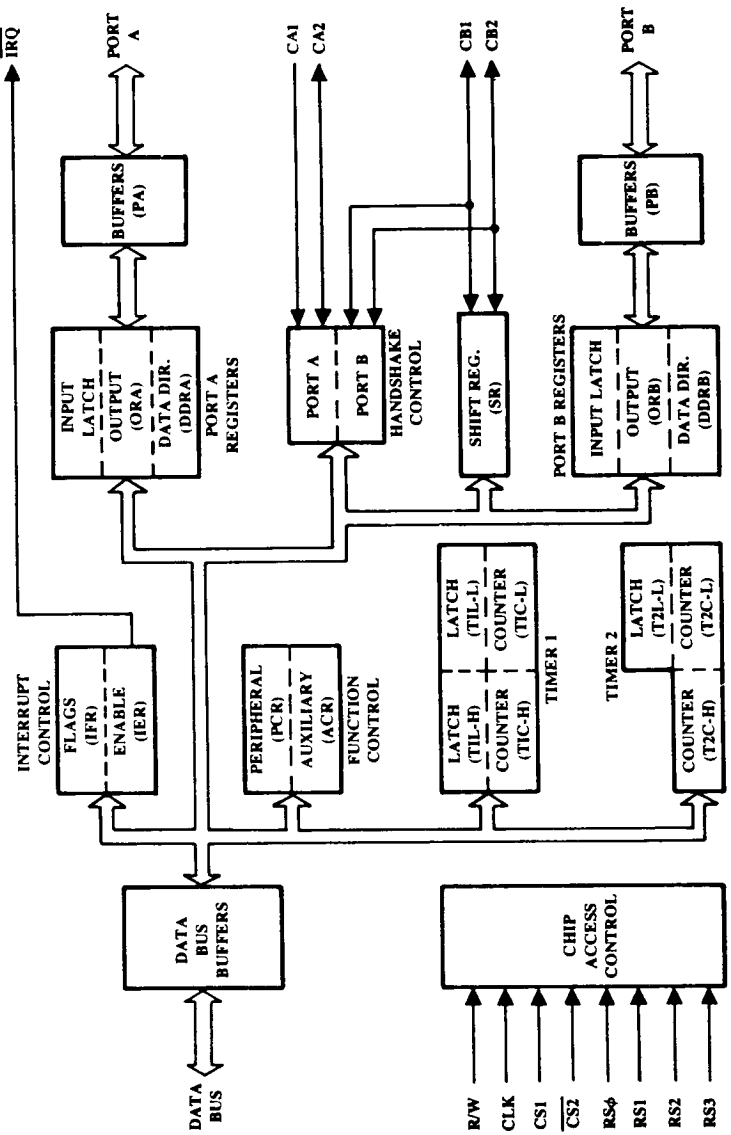


Figure 8-1. Versatile Interface Adapter

Note that there is a data direction register for each side but only one pair of control registers. Ports A and B are almost identical. One important difference is that port B can handle Darlington transistors which are used to drive solenoids and relays. We will generally use port A for input and port B for output in our example.

Figure 8-1 is a block diagram of the R6522 Versatile Interface Adapter.

8.3 SIMPLE I/O WITH THE VERSATILE INTERFACE ADAPTER

Since RESET clears all the VIA registers, disabling all interrupts and clearing all control lines, we can discuss simple I/O referring only to the data registers and the data direction registers. So simple I/O can be performed with the R6522 VIA as follows:

1. Establish the directions of the pins by storing the proper values in the data direction registers.

2. Transfer data by moving it to or from the data registers.

Note that most programs only have to execute Step 1 once since the directionality of most input and output devices is fixed (i.e., you never want to read data from a display or printer or write data to a switch or paper tape reader).

You can establish directions as follows:

1. A '0' in a bit in the data direction register makes the corresponding pin an input.

For example, a '0' in bit 4 of data direction register A makes pin PA4 into an input.

2. A '1' in a bit in the data direction register makes the corresponding pin an output.

   For example, a '1' in bit 6 of data direction register B makes pin PB6 into an output.

As for transferring data, remember that the R6502 microprocessor has no specific I/O instructions. Storing data in a VIA port that has been designated for output is equivalent to sending the data to the attached output device. Loading data from a VIA port that has been designated for input is equivalent to reading the data from the attached input device. But any instruction that acts on memory can serve as an I/O instruction if the specified address is actually an I/O device. You must be careful of the exact significance of such instructions in writing, reading, and documenting R6502 programs.

EXAMPLES

In these examples, we use the assembler designations (U in front of the register name) for the user VIA addresses in the assembler format. We use the actual addresses in the AIM 65 mnemonic format. Note that the labels for the user R6522 variables correspond to the labels in the AIM 65 memory map (Table 7-11) and Monitor program listing. The following equates were established at the beginning of the assembly source code:


```
==0000 UDRB
       =$A000
==0000 UDRAH
       =$A001
==0000 UDDRB
       =$A002
==0000 UDDRA
       =$A003
==0000 UT1L
       =$A004
==0000 UT1CH
       =$A005
==0000 UT1LL
       =$A006
==0000 UT1LH
       =$A007
==0000 UT2L
       =$A008
==0000 UT2H
       =$A009
==0000 USR
       =$A00A
==0000 UACR
       =$A00B
==0000 UPCR
       =$A00C
==0000 UIFR
       =$A00D
==0000 UIER
       =$A00E
==0000 UDRA
       =$A00F
```

1. Fetch data from a simple input port (e.g., from a set of switches or a keypad) and store it in memory location 40.

   (AIM 65 assembly format)

```
==0200 EX1
A500   LDA #0
;MAKE SIDE A INPUTS
8D03A0 STA UDDRA
AD01A0 LDA UDRAH
;GET AND STORE DATA
8540   STA $40
```

(AIM 65 disassembly format)

```
0200 A9 LDA #00
0202 8D STA A003
0205 AD LDA A001
0208 85 STA 40
```

2. Send data to a simple output port (e.g., to a set of displays or relays) from memory location 40.

(AIM 65 assembly format)

```
==020A EX2
A9FF   LDA ##FF
;MAKE SIDE B OUTPUTS
8D02A0 STA UDDRB
A540   LDA $40
;GET AND SEND DATA
8D00A0 STA UDRB
```

(AIM 65 disassembly format)

```
020A A9 LDA #FF
020C 8D STA A002
020F A5 LDA 40
0211 8D STA A000
```

You can mix inputs and outputs on a single port by establishing the directions of individual pins appropriately. Note that you can read the states of data pins (e.g., with LDA UDRA or LDA UDRB) even if they have been designated as outputs. The B side is buffered so that it can always be

read correctly, however, the A side is not buffered so that it can only be read correctly if it is lightly loaded (or designated as inputs).

8.4 RECOGNIZING STATUS SIGNALS

If the I/O device is more complex, we cannot simply transfer data to or from it at will. In the input case, the processor must know when new data is available (e.g., a key has been pressed on a keyboard or a tape reader has read another character). In the output case, the processor must know whether the device is ready to receive data (e.g., a printer has finished printing the last character or a modem has completed the previous transmission).

Normally, the input or output device provides a status signal. A transition on that line indicates the availability of data or the readiness of the device. The microcomputer I/O section must recognize the transition and allow the processor to determine that it has occurred.

You can handle this kind of I/O with the R6522 Versatile Interface Adapter as follows:

1. Attach the peripheral status input to input CA1 or CB1.

2. Determine which edge on the status line will be recognized by assigning a value to control register bit 0 (CA1) or 4 (CB1). A value of zero in that bit position means that the interrupt flag will be set by a high-to-low transition (or falling edge). A value of

one means that the interrupt flag will be set by a low-to-high transition (or rising edge).

3. Determine whether a transition has occurred by examining bit 1 (CA1) or 4 (CB1) of the interrupt flag register. The bit will be one if a transition has occurred.

4. Reset the interrupt flag by reading or writing the corresponding data register. The flag is then ready to be used in the next operation.

Let us now look at some examples:

1. Fetch data from an input port with an active high-to-low DATA READY strobe and place the data in memory location 40.

(AIM 65 assembly format)

```
          *=$0214
==0214 EA4
A900    LDA #0
;MAKE SIDE A INPUTS
8B03A0 STA UDDRA
;SET CA1 INT FLAG
;ON FALLING EDGE
8B00A0 STA UPCR
==0220 WTDATA
A00DA0 LDA UIFR
2902    AND #%00000001
0
;DATA READY?
F0F9    BEQ WTDATA
;YES, GET AND STORE
;DATA
A00FA0 LDA UORA-
8540    STA $40
```


(AIM 65 disassembly format)

```
0214 A9 LDA #00
0216 8D STA A03C
0219 8D STA A00C
021C AD LDA A00D
021F 29 AND #02
0221 F0 BEQ 021C
0223 AD LDA A001
0226 85 STA 40
```

Clearing the Peripheral Control Register is unnecessary if the routine is starting from a reset. Note that reading the Output Data Register with LDA UDRAH clears the interrupt flag so that it is available for the next DATA READY signal.

2. Send data to an output port with an active high-to-low PERIPHERAL READY strobe. Get the data from memory location 40 and send it when the peripheral is ready.

(AIM 65 assembly format)

```
==0229 EA4
A9FF    LDA #$FF
;MAKE SIDE B OUTPUTS
8B03A0 STA UDDRB
A900    LDA #0
;SET CB1 INT FLAG
;ON FALLING EDGE
8B00A0 STA UPCR
==0233 WTRDY4
A00DA0 LDA UIFR
2910    AND #%00001000
0
;PERIPHERAL READY?
F0F9    BEQ WTRDY4
;YES, GET AND SEND
;DATA
A540    LDA $40
8D00A0 STA UDRB
```

(AIM 65 disassembly format)

```
0218 A9 LDA #FF
021A 8D STA A002
021D A9 LDA #10
021F 8D STA A000
0222 AD LDA A00D
0225 29 AND #10
0227 F0 BEQ 0222
0229 A5 LDA 40
022B 8D STA A000
```

Note that sending the data to the Output Data Register (STA UDRB) clears the interrupt flag so that it is available for the next PERIPHERAL READY signal.

3. Fetch data from an input port with an active low-to-high DATA READY strobe and place the data in memory location 40.

(AIM 65 assembly format)

```
==023E EX5
A900   LDA #0
;MAKE SIDE A INPUTS
8D03A0 STA UDDRA
A901   LDA #1
;SET CA1 INT FLAG
;ON RISING EDGE
8D00A0 STA UPCR
==0248 WTDAT5
AD0DA0 LDA UIFR
2902   AND #%00000001
0
;DATA READY?
F0FB   BEQ WTDAT5
;YES, GET AND STORE
;DATA
AC01A0 LDA UDRAH
8540   STA $40
```


(AIM 65 disassembly format)

```
023E A9 LDA #00
0240 8D STA A003
0243 A9 LDA #01
0245 8D STA A001
0248 AD LDA A00D
024B 29 AND #02
024D F0 FB BEQ 0248
024F AD LDA A001
0252 85 STA 40
```

4. Send data to an output port with an active low-to-high PERIPHERAL READY strobe. Get the data from memory location 40 and send it when the peripheral is ready.

(AIM 65 assembly format)

```
==0254 EX6
A9FF   LDA #$FF
;MAKE SIDE B OUTPUTS
8D02A0 STA UDDRB
;SET CB1 INT FLAG ON
;RISING EDGE
A910   LDA #%00010000
0
8D0CA0 STA UPCR
==025E WTRDY6
AD0DA0 LDA UIFR
2910   AND #%00010000
0
;PERIPHERAL READY?
F0FB   BEQ WTRDY6
;YES, GET & SEND DATA
A540   LDA $40
8D00A0 STA UDRB
```

```
0254 A9 LDA #FF
0256 8D STA A002
0259 A9 LDA #10
025B 8D STA A00C
025E AD LDA A00D
0261 29 AND #10
0263 F0 BEQ 025E
0265 A5 LDA 40
0267 8D STA A000
```

Note that the VIA has both input and output latches. The output latches are always enabled; output data is latched when it is stored in an output data register. The input latches, if they are needed, can be enabled by setting Bit 0 (side A) or Bit 1 (side B) of the Auxiliary Control Register. The input data will then be latched by the active transition on CA1 or CB1.

8.5 PRODUCING OUTPUT STROBES

The peripheral may also require information about when a transfer has occurred or whether the port is ready to receive data. For example, devices such as digital-to-analog converters commonly require a LOAD pulse to enter data into the converter. A multiplexed display requires an output signal that directs the next output properly. A communications device may need a signal to indicate that an input buffer is available or that an output buffer is full. Output signals may also be needed to turn devices on or off, activate operator displays, or control operating modes.

You can handle this kind of I/O with the R6522 Versatile Interface Adapter as follows:

1. Attach the control output to CA2 or CB2.

2. Make CA2 (CB2) into an output by setting control register bit 3 (7).

3. Make CA2 (CB2) into a pulse by clearing control register bit 2 (6) or into a level by setting that bit.

4. If CA2 (CB2) is a pulse, make it into a handshake signal (low from the time the Output Register is read or written until the next active transition on CA1 (CB1) by clearing control register bit 1 (5) or into a single-cycle strobe by setting that bit.

5. If CA2 (CB2) is a level, determine its value by clearing or setting bit 1 (5).

So the options are:

1. CA2 goes low when the processor transfers data to or from Output Register A and goes high when the next active transition occurs on CA1. The signal can indicate that the port is ready for more data or that output data is available. The peripheral's response then indicates that it has sent more data or has processed the previous data.

2. CA2 goes low when the processor transfers data to or from Output Register A and goes high after one clock cycle. This signal indicates that an input or output operation has occurred and can be used for multiplexing.

3. CA2 is a level controlled by the value of control register bit 1. This signal can provide an active-high or low pulse of arbitrary length. It can be used to load registers, turn devices on or off, or control operating modes.

Let us now look at some examples:

1. Fetch data from an input device that requires a handshake signal and that produces an active high-to-low DATA READY strobe. Place the data in memory location $40.

(AIM 65 assembly format)

```
==026A EX7
A900   LDA #0
;MAKE SIDE A INPUTS
8D03A0 STA UDDRA
;SET CA2 HANDSHAKE
;OUTPUT MODE WITH
;CA1 INT FLAG SET
;ON FALLING EDGE
A908   LDA #%00000100
0
8D0CA0 STA UPCR
==0274 WTDAT7
AD0DA0 LDA UIFR
2902   AND #%00000001
0
;DATA READY?
F0F9   BEQ WTDAT7
;YES, GET AND STORE
;DATA AND SEND DATA
;TAKEN ON CA2
AD01A0 LDA UDRAH
8540   STA $40
```


(AIM 65 disassembly format)

```
026A A9 LDA #00
026C 8D STA A003
026F A9 LDA #08
0271 8D STA A00C
0274 AD LDA A00D
0277 29 AND #02
0279 F0 BEQ 0274
027B AD LDA A001
027E 85 STA 40
```

The Peripheral Control Register bits are:

bits 4-7 = 0 since CB1 and CB2 are not used

bit 3 = 1 to make CA2 an output

bit 2 = 0 to make CA2 a pulse

bit 1 = 0 to make CA2 a handshake acknowledgement that remains low until the next active transition on CA1

bit 0 = 1 to make the active transition on CA1 a falling edge (high-to-low transition).

2. Fetch data from an input device that requires a brief DATA ACCEPTED strobe for multiplexing or control purposes. Place the data in memory location $40.

(AIM 65 assembly format)

```
==0180 EX8
A900   LDA #0
;MAKE SIDE A INPUTS
8D03A0 STA UDDRA
;SET CA2 PULSE
;OUTPUT MODE WITH
;CA1 INT FLAG SET
;ON FALLING EDGE
A90A   LDA #%00000101
0
8D0CA0 STA UPCR
==028A WTRDY8
AD0DA0 LDA UIFR
2902   AND #%00000001
0
;DATA READY?
F0F8   BEQ WTRDY8
;YES, GET AND STORE
;DATA AND SEND DATA
;TAKEN STROBE ON CA2
AD01A0 LDA UDRAH
8540   STA $40
```

(AIM 65 disassembly format)

```
0280 A9 LDA #00
0282 8D STA A003
0285 A9 LDA #0A
0287 8D STA A00C
028A AD LDA A00D
028D 29 AND #02
028F F0 BE0 028A
0291 AD LDA A001
0294 85 STA 40
```

Here bit 1 of the Peripheral Control Register is set to 1 to make CA2 a brief strobe lasting one cycle after the reading of Port A Output Data Register.

3. Send data to an output device that requires a handshake signal and that produces an active low-to-high PERIPHERAL READY strobe. Get the data from memory location $40 and send it when the peripheral is ready.

(AIM 65 assembly format)

```
==0296 EX9
A9FF   LDA ##FF
;MAKE SIDE B OUTPUTS
8D02A0 STA UDDRB
;SET CB2 HANDSHAKE
;OUTPUT MODE WITH
;CB1 INT FLAG SET
;ON RISING EDGE
A990   LDA #%1001000
0
8D0CA0 STA UPCR
==02A0 WTRDY9
AD0DA0 LDA UIFR
2910   AND #%0001000
0
;PERIPHERAL READY?
F0F8   BEQ WTRDY9
;YES, GET AND SEND
;DATA AND SET
;ACKNOWLEDGE
A540   LDA $40
8D00A0 STA UDRB
```

(AIM 65 disassembly format)

```
0296 A9 LDA #FF
0298 8D STA A002
029B A9 LDA #90
029D 8D STA A00C
02A0 AD LDA A00D
02A3 29 AND #10
02A5 F0 BE0 02A0
02A7 A5 LDA 40
02A9 8D STA A000
```

The Peripheral Control Register bits are:

bit 7 = 1 to make CB2 an output

bit 6 = 0 to make CB2 a pulse

bit 5 = 0 to make CB2 a handshake acknowledgement that remains low until the next active transition on CB1

bit 4 - 1 to make the active transition on CB1 a rising edge (low-to-high transition)

bits 0-3 = 0 since CA1 and CA2 are not used.

4. Send data to an output device that requires a brief OUTPUT or DATA READY strobe for multiplexing or control purposes. Fetch the data from memory location $40.

(AIM 65 assembly format)

```
==02B0 EX10
A9FF   LDA #$FF
;MAKE SIDE B OUTPUTS
8D02A0 STA UDDRB
;SET CB2 PULSE
;OUTPUT MODE
A9A0   LDA #%1010000
0
8D0CA0 STA UPCR
;GET AND SEND DATA
;AND DATA STROBE
A540   LDA $40
8D00A0 STA UDRB
```

(AIM 65 disassembly format)

```
02E0 A9 LDA #FF
02E2 8D STA A002
02E5 A9 LDA #A0
02E7 8D STA A00C
02EA A5 LDA 40
02EC 8D STA A000
```


Here bit 5 of the Peripheral Control Register is set to 1 to make CB2 a brief strobe lasting one cycle after the writing of Port B Output Data Register.

5. Fetch data from an input device that requires an active-high START pulse. The device produces an active high-to-low DATA READY strobe. Place the data in memory location $40.

(AIM 65 assembly format)

```
==02BF EX11
A900   LDA #0
;MAKE SIDE A INPUTS
8D03A0 STA UDDRA
;RESET START PULSE
A90C   LDA #%00000110
0
8D0CA0 STA UPCR
;SET START PULSE
;HIGH ON CA2
A90E   LDA #%00000111
0
8D0CA0 STA UPCR
;RESET START PULSE
A90C   LDA #%00000110
0
==02D0
8D0CA0 STA UPCR
==02D3 WTDT11
AD0DA0 LDA UIFR
2902   AND #%00000001
0
;DATA READY?
F0F9   BEQ WTDT11
;YES, GET AND STORE
;DATA
AD01A0 LDA UDRAH
8540   STA $40
```

(AIM 65 disassembly format)

```
02BF A9 LDA #00
02C1 8D STA A003
02C4 A9 LDA #0C
02C6 8D STA A00C
02C9 A9 LDA #0E
02CB 8D STA A00C
02CE A9 LDA #0C
02D0 8D STA A00C
02D3 AD LDA A00D
02D6 29 AND #02
02D8 F0 BE0 02D3
02DA AD LDA A001
02DD 85 STA 40
```

Here bit 2 of the Peripheral Control Register is set to 1 to make CA2 a level with the value given by bit 1 of the Peripheral Control Register. This mode can be used to produce pulses of any length and polarity; it is called the manual output mode because there is no automatic pulse information.

In a typical application, an analog-to-digital converter or data acquisition system usually needs a START CONVERSION pulse to begin operations.

6. Send data to an output device that must be turned on before the data is sent and turned off after the data is sent (a logic 1 on a control line turns the device on). The peripheral produces an active low-to-high PERIPHERAL READY strobe. Get the data from memory location $40 and send it when the peripheral is ready.

(AIM 65 assembly format)

```
==02DF EX12
A9FF    LDA #$FF
;MAKE SIDE B OUTPUTS
8D02A0 STA UDDRB
A9F0    LDA #%1111000
0
;TURN PERIPHERAL ON
;BY SETTING CB2 HIGH
;AND SET CB1 INT
;FLAG ON RISING EDGE
8D0CA0 STA UPCR
==02E9 WTRD12
AD0DA0 LDA UIFR
2910    AND #%0001000
0
;PERIPHERAL READY?
F0F9    BEQ WTRD12
;YES, GET & SEND DATA
A540    LDA $40
8D00A0 STA UDRB
;TURN PERIPHERAL OFF
A9D0    LDA #%1101000
0
8D0CA0 STA UPCR
```

(AIM 65 disassembly format)

```
02FF A9 LDA #FF
0301 8D STA A002
0304 A9 LDA #F0
0306 8D STA A00C
0309 AD LDA A00D
030C 29 AND #10
030E F0 BE0 02E9
0310 A5 LDA 40
0312 8D STA A000
0315 A9 LDA #D0
0317 8D STA A00C
```

In many applications, such as portable equipment, the output peripheral is only turned on when data is to be sent to it. In other applications, the processor must issue an OUTPUT REQUEST and receive an acknowledgement before sending the data.

8.6 VIA INTERRUPTS

You can easily use the R6522 Versatile Interface Adapter in an interrupt-driven mode. Figure 8-2 shows the Interrupt Enable Register (IER). Any of the various interrupt sources can be enabled by setting the corresponding enable bit. Note that the most significant bit controls how the other enable bits are affected:

If IER7 = 0, each '1' in a bit position clears an enable bit and thus disables that interrupt.

If IER7 = 1, each '1' in a bit postion sets an interrupt bit and thus enables that interrupt.

Zeroes in the enabling bit positions always leave the enable bits as they were.

Some examples should help you see how this works.

1. Enable CA1 interrupt, disable all others.

(AIM 65 assembly format)

```
==02FA EX13
A97D   LDA #%01111110
1
;DISABLE OTHER
;INTERRUPTS
9D0EA0 STA UIER
A982   LDA #%10000001
0
;ENABLE CA1
;INTERRUPT
9D0EA0 STA UIER
```


(AIM 65 disassembly format)

```
02FA A9 LDA #7D
02FC 8D STA A00E
02FF A9 LDA #82
0301 8D STA A00E
```

The first operation clears all the interrupt enables except CA1. The second operation sets the CA1 interrupt enable.

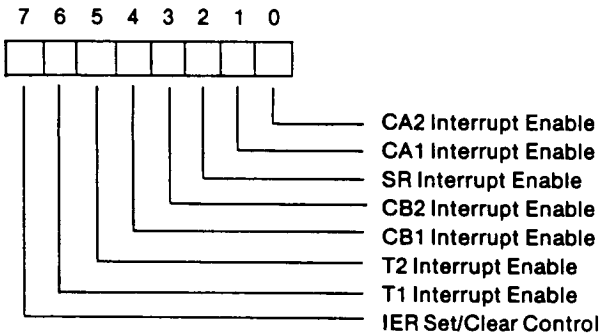
2. Enable CB1 and CB2 interrupts, disable all others.

(AIM 65 assembly format)

```
==0304 EX14
A967   LDA #%00110011
1
;DISABLE OTHER
;INTERRUPTS
9D0EA0 STA UIER
A995   LDA #%10011100
0
;ENABLE CB1 AND CB2
;INTERRUPTS
9D0EA0 STA UIER
```

(AIM 65 disassembly format)

```
0304 A9 LDA #67
0306 8D STA A00E
0309 A9 LDA #98
030B 8D STA A00E
```



INTERRUPT ENABLE BITS (IER0-6)

IERn = 0 Disable interrupt
= 1 Enable interrupt

IER SET/CLEAR CONTROL (IER7)

IER7 = 0 For each data bus bit set to logic 1, clear corresponding IER bit
= 1 For each data bus bit set to logic 1, set corresponding IER bit.

Note: IER7 is active only when R/$\overline{W}$ = L; when R/$\overline{W}$ = H, IER7 will read logic 1.

Figure 8-2. R6522 Interrupt Enable Register (IER)

Note that we could disable all interrupts in the first step.

3. Disable CA1 interrupt, leave others as they were.

(AIM 65 assembly format)

```
==0310 EX15
A902   LDA #%0000001
0
;DISABLE CA1
;INTERRUPT
8D0EA0 STA UIER
```

(AIM 65 disassembly format)

```
0310 A9 LDA #02
0312 8D STA A00E
```

4. Disable CB1 and CB2 interrupts, leave others as they were.

(AIM 65 assembly format)

```
==0310 EX16
A918   LDA #%0001100
0
;DISABLE CB1 AND CB2
;INTERRUPTS
8D0EA0 STA UIER
```

(AIM 65 disassembly format)

```
0210 A9 LDA #10
0212 8D STA A00E
```

The processor can determine which interrupt has occurred by examining the interrupt flag register (Figure 8-3). Note that examining bit 7 determines if any interrupts have occurred on the VIA. Note also the conditions for clearing the interrupt flags.

A typical polling sequence would be:

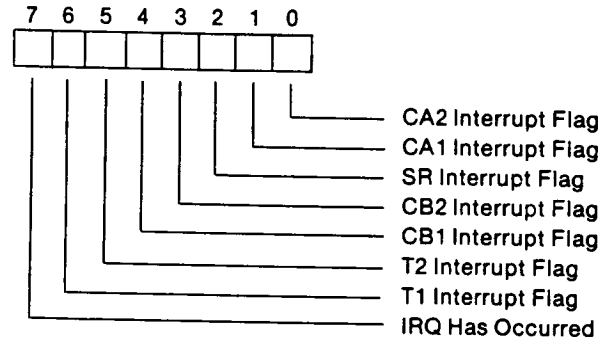
```
LDA     UIFR        ;ANY INTERRUPTS ON THIS VIA?
BPL     NEXTS       ;NO, LOOK AT NEXT SOURCE
ASL     A           ;IS INTERRUPT FROM T1?
BMI     TIM1        ;YES, GO SERVICE T1 INTERRUPT
ASL     A           ;IS INTERRUPT FROM T2?
BMI     TIM2        ;YES, GO SERVICE T2 INTERRUPT
ASL     A           ;IS INTERRUPT FROM CB1?
BMI     CB1         ;YES, GO SERVICE CB1 INTERRUPT
.
.
.
```

8.7 VIA TIMERS

The two 16-bit timer/counters in the R6522 Versatile Interface Adapters can be used to:

1. Generate a single time interval. The timer must be loaded with the number of clock pulses.

R6522 INTERRUPT FLAG REGISTER (IFR)



IFR Bit	Set By	Cleared By
0	Active transition on CA2	Reading or writing the ORA ($A001 or $A00F)
1	Active transition on CA1	Reading or writing the ORA ($A001 or $A00F)
2	Completion of eight shifts	Reading or writing the SR ($A00A)
3	Active transition on CB2	Reading or writing the ORB ($A000)
4	Active transition on CB1	Reading or writing the ORB ($A000)
5	Time-out of Timer 2	Reading T2C-L ($A008) or writing T2C-H ($A009)
6	Time-out of Timer 1	Reading T1C-L ($A004) or writing T1L-H ($A005 or $A007)
7	Any IFR bit set with its corresponding IER bit also set	Clearing IFR0-IFR6 ($A00D) or IER0-IER6 ($A00E)

Figure 8-3. R6522 Interrupt Flag Register (IFR)

2. Count pulses on pin PB6 (timer 2 only). The timer must be loaded with the number of pulses to be counted.

3. Generate continuous time intervals (timer 1 only). The timer must be loaded with the number of clock pulses per interval.

4. Produce a single pulse or a continuous series of pulses on pin PB7 (timer 1 only). The timer must be loaded with the number of clock pulses per interval.

Let us first look at timer 2, which can only be used to generate a single time interval (the so-called one-shot mode) or to count pulses on PB6. Bit 5 of the Auxiliary Control Register selects the mode:

Bit 5 = 0 for one-shot mode, 1 for pulse-counting mode

Note that 16-bit timer 2 occupies two memory locations (see Table 8-1). The first address (A008) is used to read or write the 8 least significant bits; reading this address also clears the timer 2 interrupt flag (Figure 8-3). The second address (A009) is used to read or write the 8 most significant bits; writing this address loads the counters, clears the timer 2 interrupt flag, and starts the timing operation. The completion of the operation sets the timer 2 interrupt flag.

EXAMPLES:

1. Generate a delay of 2048 (0800 hex) clock pulses.

(AIM 65 assembly format)

```
==031D EX17
A950   LDA #0
;SET T2 ONE-SHOT
;INTERVAL TIMER MODE
8D0BA0 STA UACR
;DELAY = 0800 CLOCKS
8D08A0 STA UT2L
A908   LDA #8
;LOAD & START TIMER
8D09A0 STA UT2H
A920   LDA #%00100000
0
;T2 COUNTED DOWN?
==0328 WAIT17
2C0DA0 BIT UIFR
F0F8   BEQ WAIT17
;YES,CLR T2 INT FLAG
AD08A0 LDA UT2L
```

(AIM 65 disassembly format)

```
031D A9 LDA #00
031F 8D STA A00B
0322 8D STA A008
0325 A9 LDA #08
0327 8D STA A009
032A A9 LDA #20
032C 2C BIT A00D
032F F0 BEQ 0328
0331 AD LDA A008
```

Note the final LDA UT2L. The sole purpose of this instruction is to clear the timer 2 interrupt flag so that it will be available for the next operation.

2. Count 5 input pulses on PB6.

(AIM 65 assembly format)

```
==0034 EX18
A500   LDA #0
;MAKE SIDE B INPUTS
8002A0 STA UDDRB
;SET T2 PULSE COUNT
;MODE
A920   LDA #%00010000
0
8B0BA0 STA UACR
;DELAY = 0005 CLOCKS
A505   LDA #5
8D09A0 STA UT2L
A500   LDA #0
==0045
;START T2
8D0AA0 STA UT2H
A920   LDA #%00010000
0
;T2 COUNTED DOWN?
==004A WAIT18
2C0DA0 BIT UIFR
F0FB   BEQ WAIT18
;YES:CLR T2 INT FLAG
AD08A0 LDA UT2L
```

(AIM 65 disassembly format)

```
0034 A9 LDA #00
0036 8D STA A002
0039 A9 LDA #20
003B 8D STA A00B
003E A9 LDA #05
0040 8D STA A008
0043 A9 LDA #00
0045 8D STA A005
0048 A9 LDA #20
004A 2C BIT A00D
004D F0 FB 004A
004F AD LDA A008
```


Note that you must load the least significant bits of the timer first, since loading the most significant bits loads the counters and starts the timing operation.

Timer 1 is somewhat more complex than timer 2 because it has four operating modes (see Table 8-2). It can be used to generate a single time interval (one-shot mode) or a continuous series of intervals (free-running mode). Furthermore, each loading operation can generate an output pulse on PB7. Bits 6 and 7 of the Auxiliary Control Register determine timer 1 mode.

Bit 7 = 1 to generate output pulses on PB7, 0 to disable such pulses (in the free-running mode, PB7 is inverted each time the counter reaches zero).

Bit 6 = 1 for free-running mode, 0 for one-shot mode.

Table 8-2. Timer 1 Control

ACR7	ACR6	Mode
0	0	T1 one-shot mode — Generate a single time-out interrupt each time T1 is loaded. Output to PB7 disabled.
0	1	T1 free-running mode — Generate continuous interrupts. Output to PB7 disabled.
1	0	T1 one-shot mode — Generate a single time-out interrupt and an output pulse on PB7 each time T1 is loaded.
1	1	T1 free-running mode — Generate continuous interrupts and a square wave output on PB7.

Timer 1 occupies four memory locations (see Table 8-1). The first two addresses (A004 and A005) are used to read or write the counters. Writing the second address loads the counters, clears the timer 1 interrupt flag, and starts the timing operation. The next two addresses (A006 and A007) are used to read or write the latches without affecting the counters. This allows the generation of complex waveforms in the free-running mode. Writing the most significant bits of the latches also clears the timer 1 interrupt flag.

EXAMPLES:

1. Generate a delay of 1024 (0400 hex) clock pulses, no output to PB7.

(AIM 65 assembly format)

```
==0152 EX19
A900   LDA #0
;SET T1 ONE-SHOT -
;PBT DISABLED MODE
8D0BA0 STA UACR
;DELAY = 0400 CLOCKS
8D04A0 STA UT1L
A904   LDA #4
;START T1
8D05A0 STA UT1CH
A940   LDA #%0100000
0
;T1 COUNTED DOWN?
==0161 WAIT19
2C0DA0 BIT UIFR
F0FB   BEQ WAIT19
;YES,CLR T1 INT FLAG
AD04A0 LDA UT1L
```


(AIM 65 disassembly format)

```
0152 A9 LDA #00
0154 8D STA A00B
0157 8D STA A004
015A A9 LDA #04
015C 8D STA A005
015F A9 LDA #40
0161 2C BIT A00D
0164 F0 BEQ 0161
0166 AD LDA A004
```

This program is the same as the one shown previously for timer 2 except for the changes in the addresses and the interrupt flag bit position.

2. Generate an interrupt every 4096 (1000 hex) clock pulses and produce a pulse output on PB7 with a pulse width of 4096 clock pulses. Note that timer 1 is loaded with the desired clock count minus two. e.g. ($0FFE).

(AIM 65 assembly format)

```
;SET IRQ VECTOR
==0000
       *=$A400
==A400
0001   .WOR IRQINT
;
==A402
       *=$0370
==0370 EX20
A9FF   LDA #$FF
;MAKE SIDE B OUTPUTS
8D02A0 STA UDDRB
;SET T1 FREE RUN -
;PBT ACTIVE MODE
A9C0   LDA #%1100000
0
8D0BA0 STA UACR
;ENABLE T1 IRQ INT
8D0EA0 STA UIER
;DELAY =1000 CLOCKS
;      =0FFE T1 LOAD
```

(AIM 65 assembly format) cont.

```
A9FE   LDA #$FE
8D04A0 STA UT1L
==0362
;START T1
A90F   LDA #$0F
8D05A0 STA UT1CH
;ENABLE IRQ INT
58     CLI
;CONTINUE
---------------------
IRQ INT PROCESSING

==0388 IRQINT
A940   LDA #%01000000
0
2C0DA0 BIT UIFR
;T1 CAUSE IRQ INT?
D008   BNE INTRET
;YES. CLR T1 INT FLAG
AD04A0 LDA UT1L
;CONTINUE T1 INT
;PROCESSING
==0392 INTRET
40     RTI
;RETURN
```

(AIM 65 disassembly format)

```
0370 A9 LDA #FF
0372 8D STA A002
0375 A9 LDA #C0
0377 8D STA A00B
037A 8D STA A00E
037D A9 LDA #F2
037F 8D STA A004
0382 A9 LDA #0F
0384 8D STA A005
0387 58 CLI
```


(AIM 65 disassembly format) cont.

```
0388 A9 LDA #40
038A 2C BIT A00D
038D D0 BNE 0392
038F AD LDA A004
0392 40 RTI

.M0=A400 88 03 7B E0
```

This program will cause the processor to be interrupted every 4096 clock cycles. The level on PB7 will be inverted at the end of each interval (it will go low when the first interval starts). Note that T1 runs continuously; whenever the counters reach zero, they are reloaded with the values in the latches.

8.8 VIA SHIFT REGISTER

The VIA also has a shift register which can be used to convert data between serial and parallel forms. Auxiliary control register bits 2, 3, and 4 control this register as shown in Table 8-3. Loading the register starts the shifting process and an interrupt flag (bit 2 of the interrupt flag register) is set after the completion of eight shifts.

Table 8-3. Shift Register Control

ACR4	ACR3	ACR2	Mode
0	0	0	Shift Register Disabled.
0	0	1	Shift in under control of Timer 2.
0	1	0	Shift in under control of ø2.
0	1	1	Shift in under control of external clock.
1	0	0	Free-running output at rate determined by Timer 2.
1	0	1	Shift out under control of Timer 2.
1	1	0	Shift out under control of ø2.
1	1	1	Shift out under control of external clock.

EXAMPLES:

1. Shift in 8 bits under control of timer 2 and store the data into memory location $40. Subtract 2 from the desired time value (in microseconds) to determine the times 2 count value.

(AIM 65 assembly format)

```
        *=$0390
==0390 EX21
A900   LDA #0
;DISABLE SR
8D0BA0 STA UACR
A904   LDA #%00000010
0
;SET SHIFT IN BY T2
8D0BA0 STA UACR
;DELAY = 0040 CLOCKS
==039A ST21
A928   LDA #40
8D08A0 STA UT2L
A900   LDA #0
8D09A0 STA UT2H
;START SHIFT IN
AD0AA0 LDA USR
==03A7 WTSH21
AD0DA0 LDA UIFR
2904   AND #%00000010
0
```


(AIM 65 assembly format) cont.

```
;SHIFT IN COMPLETE?
F0F9   BE0 WTSH21
;YES, DISABLE SR
A900   LDA #0
8D0BA0 STA UACR
;GET AND STORE DATA
AD0AA0 LDA USR
8540   STA $40
```

(AIM 65 disassembly format)

```
0390 A9 LDA #00
0392 8D STA A00B
0395 A9 LDA #04
0397 8D STA A00B
039A A9 LDA #28
039C 8D STA A008
039F A9 LDA #00
03A1 8D STA A009
03A4 AD LDA A00A
03A7 AD LDA A00D
03AA 29 AND #04
03AC F0 BE0 03A7
03AE A9 LDA #00
03B0 8D STA A00B
03B3 AD LDA A00A
03B6 85 STA 40
```

2. Shift in 8 bits under control of phase 2 (ø2) and store the data into memory location $40.

(AIM 65 assembly format)

```
==01B0 EX22
A500   LDA #0
;DISABLE ACR
8D0BA0 STA UACR
A906   LDA #%00000100
0
;SET SHIFT IN BY
;PHASE 2 MODE
8D0BA0 STA UACR
;START SHIFT IN
AD0AA0 LDA USR
;DELAY 13 CYCLES
A204   LDX #4
==01C7 WTSH22
CA     DEX
;SHIFT IN COMPLETE?
D0FD   BNE WTSH22
;YES. DISABLE SR
8D0BA0 STA UACR
;GET AND STORE DATA
AD0AA0 LDA USR
8540   STA $40
```

(AIM 65 disassembly format)

```
01B0 A9 LDA #00
01B2 8D STA A00B
01B5 A9 LDA #06
01B7 8D STA A00B
01BA AD LDA A00A
01BD A2 LDX #04
01BF CA DEX
01C0 D0 BNE 01C7
01C2 8D STA A00B
01C5 AD LDA A00A
01C8 85 STA 40
```


3. Shift in 8 bits under control of an external clock and store the data into memory location $40.

(AIM 65 assembly format)

```
==01D2 EX23
A500   LDA #0
;DISABLE SR
8D0BA0 STA UACR
A90C   LDA #%00000110
0
;SET SHIFT IN BY
;EXT CLOCK MODE
8D0BA0 STA UACR
==01DC WTSH23
AD0DA0 LDA UIFR
2904   AND #%00000010
0
;SHIFT IN COMPLETE?
F0FB   BEQ WTSH23
;YES. GET AND STORE
;DATA
AD0AA0 LDA USR
8540   STA $40
```

(AIM 65 disassembly format)

```
01D2 A9 LDA #00
01D4 8D STA A00B
01D7 A9 LDA #0C
01D9 8D STA A00B
01DC AD LDA A00D
01DF 29 AND #04
01E1 F0 BEQ 01DC
01E3 AD LDA A00A
01E6 85 STA 40
```

4. Continually shift out 8 bits from memory location $40 at timer 2 rate.

(AIM 65 assembly format)

```
==03E8 EX24
A900   LDA #0
;DISABLE SR
8D0BA0 STA UACR
A910   LDA #%0001000
0
;SET SHIFT OUT
;CONTINUOUSLY BY T2
8D0BA0 STA UACR
;DELAY =0080
A580   LDA #80
8D09A0 STA UT2L
A900   LDA #0
==03F9
8D09A0 STA UT2H
;LOAD SR WITH DATA
;AND START SHIFT OUT
A540   LDA $40
8D0AA0 STA USR
```

(AIM 65 disassembly format)

```
03D2 A9 LDA #00
03D4 8D STA A00E
03D7 A9 LDA #0C
03D9 8D STA A00E
03DC AC LDA A00A
03DF AD LDA A00D
03E2 29 AND #04
03E4 F0 BEQ 03DF
03E6 AD LDA A00A
03E9 85 STA 40
```


5. Shift out 8 bits from memory location $40 under control of timer 2. Subtract 2 from the desired time value (in microseconds) to determine the times 2 count value.

(AIM 65 assembly format)

```
==0401 EX25
A900   LDA #0
;DISABLE SR
8D0BA0 STA UACR
A914   LDA #%0001010
0
;SET SHIFT OUT BY
;T2 MODE
8D0BA0 STA UACR
;DELAY =0020
A920   LDA #$20
8D09A0 STA UT2L
A900   LDA #0
==0412
8D09A0 STA UT2H
;LOAD DATA INTO SR
;AND START SHIFT OUT
A540   LDA $40
8D0AA0 STA USR
```

(AIM 65 disassembly format)

```
0401 A9 LDA #00
0403 8D STA A00B
0406 A9 LDA #14
0408 8D STA A00B
040B A9 LDA #20
040D 8D STA A008
0410 A9 LDA #00
0412 8D STA A009
0415 A5 LDA 40
0417 8D STA A00A
```

6. Shift out 8 bits from memory location $40 under control of phase 2 (ø2).

(AIM 65 assembly format)

```
==041A EX26
A900   LDA #0
;DISABLE SR
8D0BA0 STA UACR
A918   LDA #%0001100
0
;SET SHIFT OUT BY
;PHASE 2 CLOCK MODE
8D0BA0 STA UACR
;LOAD SR WITH DATA
;AND START SHIFT OUT
A540   LDA $40
8D0AA0 STA USR
```

(AIM 65 disassembly format)

```
041A A9 LDA #00
041C 8D STA A00B
041F A9 LDA #18
0421 8D STA A00B
0424 A5 LDA 40
0426 8D STA A00A
```


7. Shift out 8 bits from memory location $40 under control of an external clock with IRQ interrupt handling.

(AIM 65 assembly format)

```
 SET IRQ VECTOR
==0000
        *=$A400
==A400
4594    WDR IRQINT
;
==A402
        *=$0430
==0430 EX27
A900    LDA #0
;DISABLE SR
8D0BA0 STA UACR
A911    LDA #%00001110
0
;SET SHIFT OUT BY
;EXT CLOCK MODE
8D0BA0 STA UACR
A984    LDA #%10000010
0
;ENABLE SR IRQ INT
8D0EA0 STA UIER
;LOAD SR WITH DATA
;AND START SHIFT OUT
A540    LDA $40
==0441
8D0AA0 STA USR
;ENABLE IRQ INT
58      CLI
;CONTINUE
```

(AIM 65 assembly format) cont.

```
    IRQ INT PROCESSING

    ==0445 IRQINT
    A404   LDA #%00000010
    @
    2001A0 BIT VIFR
    ;SR CAUSE IRQ INT?
    D003   BNE INTRET
    ;YES,CLR SR INT FLAG
    AD0AA0 LDA USR
    ;CONTINUE SR INT
    ;PROCESSING
    ==044F INTRET
    40     RTI
    ;RETURN
```

(AIM 65 disassembly format)

```
    0410 A9 LDA #00
    0412 8D STA A00B
    0415 A9 LDA #1C
    0417 8D STA A00B
    041A A9 LDA #84
    041C 8D STA A00E
    041F A5 LDA 40
    0441 8D STA A00A
    0444 58 CLI
    0445 A9 LDA #04
    0447 2C BIT A00D
    044A D0 BNE 044F
    044C AD LDA A00A
    044F 40 RTI

    <M;=A400 45 04 7B E0
```


8.9 INTERFACING REFERENCES

Microprocessor input/output and interfacing are discussed in:

Lesea, A. and R. Zaks, Microprocessor Interfacing Techniques, Sybex, Berkeley, Ca., 1977

Peatman, J., Microcomputer-based Design, McGraw-Hill, New York, 1977

Wakerly, J.F., "Microprocessor Input/Output Architecture", Computer, February, 1977, pp. 26-33

8.9 INTERFACING REFERENCES

Microprocessor input/output and interfacing are discussed in:

Lesea, A. and R. Zaks, Microprocessor Interfacing Techniques, Sybex, Berkeley, Ca., 1977

Peatman, J., Microcomputer-based Design, McGraw-Hill, New York, 1977

Wakerly, J.F., "Microprocessor Input/Output Architecture", Computer, February, 1977, pp. 26-33