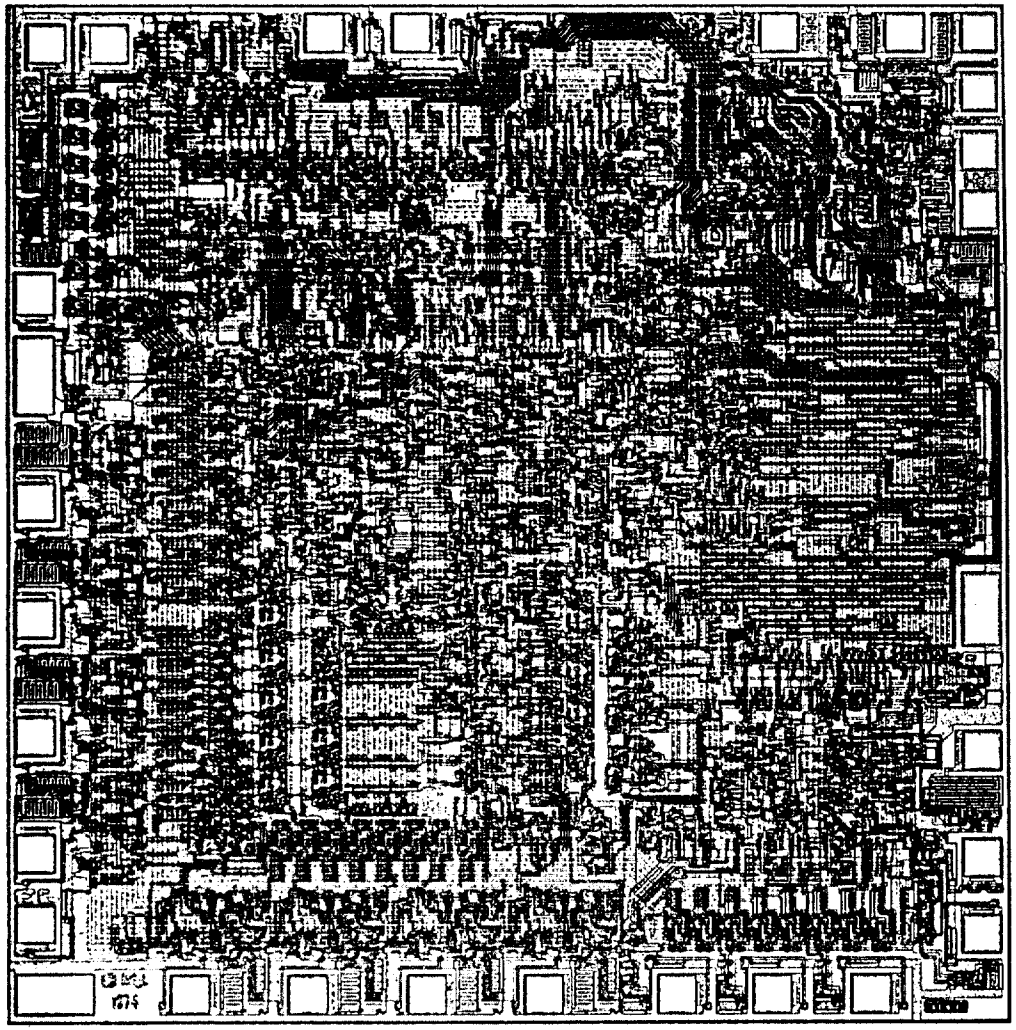


8255A Programmable Peripheral Interface Applicaitons

by Alan Ebright

INTRODUCTION 2-64

OVERVIEW 2-64

8080 CPU MODULE INTERFACE 2-64

PERIPHERAL INTERFACE SECTION 2-66

INTERNAL LOGIC SECTION 2-67

- Mode Definition 2-67
- Bit Set/Reset 2-68

INTERRUPT CONTROL LOGIC STATUS WORDS 2-69

SOFTWARE CONSIDERATION 2-71

MODE 0 — STATUS DRIVEN PERIPHERAL INTERFACE 2-73

- 8255A To Peripheral Hardware Interface 2-73
- 8080 CPU Module To 8255A Interface 2-73
- Mode 0 Interface Software 2-75
- Summary/Conclusions 2-76

MODE 1 — INTERRUPT DRIVEN PRINTER INTERFACE 2-78

- CPU Module To 8255A Interface 2-78
- 8255A To Peripheral Interface 2-78
- Mode 1 Software Driver 2-79
- Summary/Conclusions 2-80

MODE 2 — 8080 TO 8080 INTERFACE 2-83

- Hardware Discussion 2-83
- Software Discussion 2-83
- Summary/Conclusions 2-83

APPENDIX A — 8255A QUICK REFERENCE 2-90

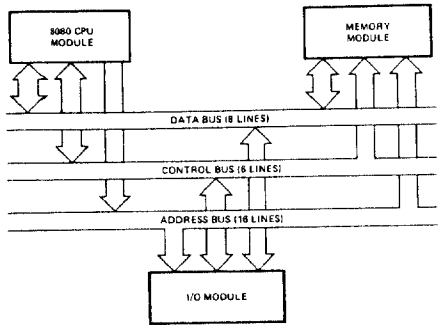
INTRODUCTION

Microprocessor-based system designs are a cost-effective solution to a wide variety of problems. When a system designer is presented with the task of selecting a microprocessor for a design, the capabilities of the microprocessor should not be the only consideration. The microprocessor should be an element of a compatible family of devices. The MCS-80 component family is a group of compatible devices which have been designed to directly address and solve the problems of microprocessor-based system design. One member of the MCS-80 component family is Intel's 8255A programmable peripheral interface chip. This device replaces a significant percentage of the logic required to support a variety of byte oriented Input/Output interfaces. Through the use of the 8255A, the I/O interface design task is significantly simplified, the design flexibility is increased, and the number of components required is reduced.

This application note presents detailed design examples from both the hardware and software points of view. Since the 8255A is an extremely flexible device, it is impossible to list all of the applications and configurations of the device. A number of designs are presented which may be modified to fulfill specific user interface requirements.

Detailed design examples are discussed within the context of the 8080 system shown in Figure 1. The basic 8080 system is composed of the CPU module, memory module, and the I/O module. CPU module and memory module design are discussed within other Intel publications. This application note deals exclusively with I/O module design.

It is assumed that the reader is familiar with the MCS-80 User's Manual and/or the MCS-85 User's Manual, particularly the 8255A device description.



OVERVIEW OF THE 8255A

The 8255A block diagram shown in Figure 2 has been divided into three sections: 8080 CPU Module Interface, Peripheral Interface, and the Internal Logic.

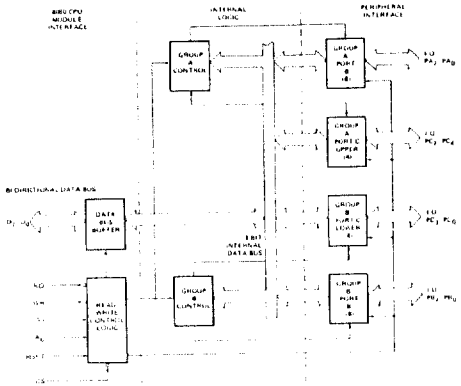


Figure 2. 8255A Block Diagram

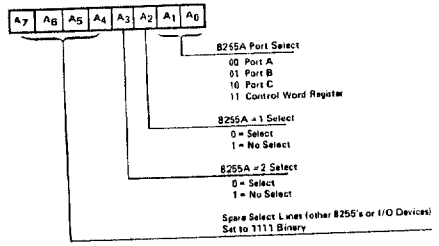
8080 CPU MODULE INTERFACE

The 8255A is a compatible member of the MCS-80 component family and, therefore, may be directly interfaced to the 8080. Figure 3 displays one method of interconnecting the 8255A and an 8080 CPU module. The 8080 CPU module consists of the 8080A CPU, the 8224 Clock Generator, and the 8228 System Controller. The system shown in Figure 3 utilizes a linear select scheme which dedicates an address line as an exclusive enable (chip select) for each specific I/O device. The chip select signal is used to enable communication between the selected 8255A and the 8080 CPU. I/O Ports A, B, C, or the Control Word Register are selected by the two port select signals (A_1, A_0). These signals (A_1 and A_0) are driven by the least significant bits of the address bus. The I/O port select characters required by this configuration are shown in Figure 4.

When a system utilizing the linear select scheme is implemented, a maximum of six I/O devices may be selected. If more than six I/O devices must be addressed, the six device select bits must be encoded to generate a maximum of 64 device select lines. Note that when large systems are implemented, bus loading considerations may require that bus drivers be included in the CPU module. The MCS-80 component family contains parts which are designed to perform this function (8216, 8226).

The 8255A I/O read ($\overline{RD}$) and I/O write ($\overline{WR}$) signals may be directly driven by the 8228. This results in an isolated I/O architecture where 8080 Input/Output instructions are used to reference an independent I/O address space. An alternate approach is memory mapped I/O. This architecture treats an area of memory as the I/O address space. The memory mapped I/O architecture utilizes 8080 memory reference instructions to access the I/O address space. Interfacing with the 8080 is outlined in Chapter 3 of the "8080 Microcomputer User's Manual".

The most important feature of the 8255A to 8080 CPU Module Interface is that for small system designs the 8255A may be interfaced directly to the standard MCS-80 component family with no external logic. Minimum external logic is required in large system designs.



Port Selected	Hexadecimal Port Select Character (Used with IN or OUT Instructions)
Port A 8255A #1	F8
Port B 8255A #1	F9
Port C 8255A #1	FA
Control Word Register 8255A #1	FB
Port A 8255A #2	F4
Port B 8255A #2	F5
Port C 8255A #2	F6
Control Word Register 8255A #2	F7

Figure 4. I/O Port Select Characters

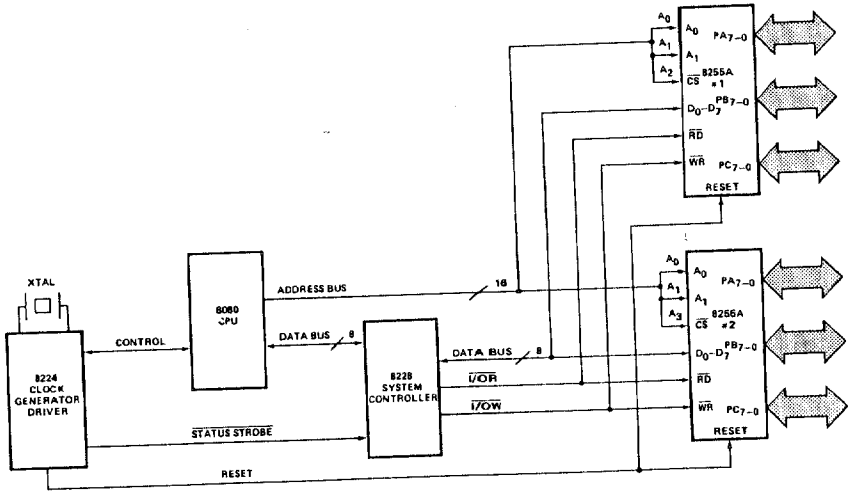


Figure 3. Linear Select 8255A Interconnect

PERIPHERAL INTERFACE SECTION

The peripheral interface section contains 24 peripheral interface lines, buffers, and control logic. The characteristics and functions of the interface lines are determined by the operating mode selected under program control. The flexibility of the 8255A is due to the fact that the device is programmable. Three modes of operation may be selected under program control: Mode 0 – Basic Input/Output, Mode 1 – Strobed Input/Output with interrupt support, and Mode 2 – Bidirectional bus with interrupt support. Through selecting the correct operating mode, the interface lines may be configured to fulfill specific interface requirements. The characteristics of the interface lines within each mode must be understood so that the designer may utilize the 8255A to achieve the most efficient design. Table I lists the basic features of the peripheral interface lines within each mode group. Figure 5 shows the grouping of the peripheral interface lines within each mode.

Table I. Features of Peripheral Interface Lines

Mode 0 – Basic Input/Output
Two 8-bit ports Two 4-bit ports with bit set/reset capability Outputs are latched Inputs are not latched
Mode 1 – Strobed Input/Output
One or two strobed ports Each Mode 1 port contains: 8-bit data port 3 control lines Interrupt support logic Any port may be input or output If one Mode 1 port is used, the remaining 13 lines may be configured in Mode 0. If two Mode 1 ports are used, the remaining 2 bits may be input or output with bit set/reset capability.
Mode 2 – Strobed Bidirectional Bus
One bidirectional bus which contains: 8-bit bidirectional bus supported by Port A 5 control lines Interrupt support logic Inputs and outputs are latched The remaining 11 lines may be configured in either Mode 0 or Mode 1.

One feature of Port C is important to note. Each Port C bit may be individually set and reset. Through the use of this feature, device strobes may be easily generated by software without utilizing external logic. The Mode 1 and Mode 2 configurations use a number of the Port C lines for interrupt control lines. Thus, the 8255A contains a large portion of the logic required to implement an interrupt driven I/O interface. This feature simplifies interrupt driven hardware design and saves a significant amount of the external logic that is normally required when less powerful I/O chips are used. In fact, the design examples contained in this application note describe how interrupt driven interfaces may be designed such that the only interrupt control logic required is that contained in the 8255A.

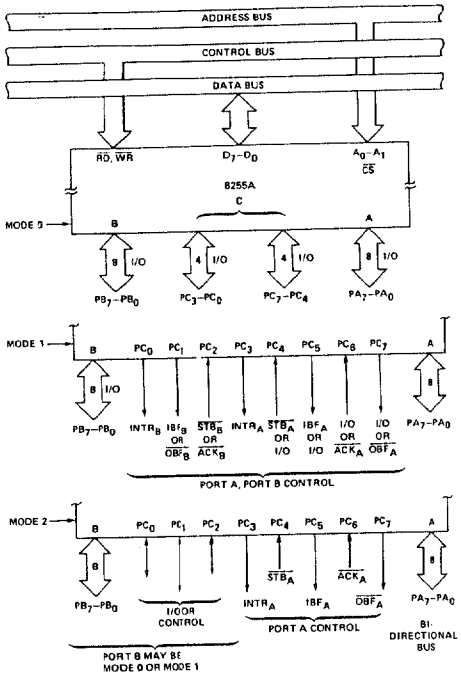


Figure 5. Grouping of Peripheral Interface Lines

INTERNAL LOGIC SECTION

The internal logic section manages the transfer of data and control information on the internal data bus (refer to Figure 2). If the port select lines (A_1 and A_0) specify Ports A, B, or C, the operation is an I/O port data transfer. The internal logic will select the specified I/O port and perform the data transfer between the I/O port and the CPU interface. As was previously mentioned, both the functional configuration of each port and bit set/reset on Port C are controlled by the system's software. When the control word register is selected, the internal logic performs the operation described by the control word. The control word contains an opcode field which defines which of the two functions are to be performed (mode definition or bit set/reset).

Mode Definition

When the opcode field (Bit 7) of the control word is equal to a one, the control word is interpreted by the 8255A as a mode definition control word. The mode definition control word (shown in Figure 6) is used to specify the configuration of the 24 8255A peripheral interface lines. The system's software may specify the modes of Port A and Port B independently. Port C may be treated independently or divided into two portions as required by the Port A and Port B mode definitions.

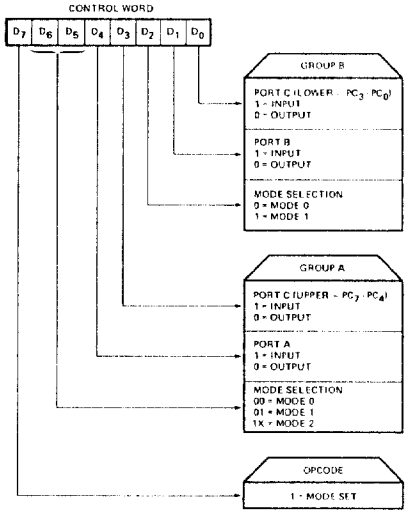


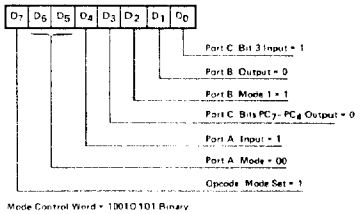
Figure 6. Mode Definition Control Word

Example #1: This example demonstrates how a mode control word is constructed and issued to an 8255A. The mode control word is passed to the device through the use of an output instruction that references an 8080 I/O port address. The value of the I/O port address is determined by the 8080 CPU interface implemented. This example references the I/O port addresses realized by the simple 8080 to 8255A interface shown in Figure 3.

If an 8255A is to be configured through the use of the mode control word interface as:

Port A	Mode 0 Input
Port B	Mode 1 Output
Port C	Bits PC_7–PC_4 Output
Port C	Bit 3 Input

The following mode control word is used:



The assembly language program is:

```
CWR     EQU   0FBH          ;8255A #1 CONTROL WORD REGISTER
;*****
;       ISSUE MODE CONTROL WORD
;*****
        MVI   A,10010101B   ;GET MODE CONTROL WORD
        OUT   CWR           ;OUTPUT TO 8255A #1 CONTROL WORD
                            ;REGISTER
```

Bit Set/Reset

When the opcode field (Bit 7) of the control word is equal to a zero, the control word is interpreted by the 8255A as a Port C bit set/reset command word (see Figure 7). Through the use of the bit set/reset command, any of the 8 bits on Port C may be independently set or reset. Note that control word bits 6–4 are not used. Bits 6–4 should be set to zero.

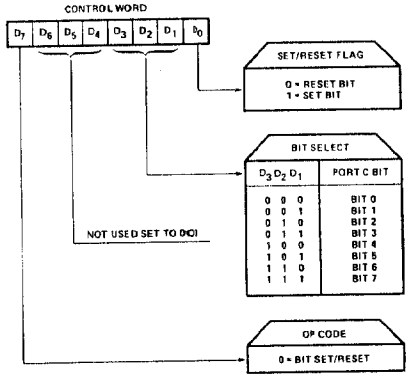
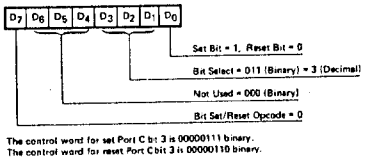


Figure 7. Bit Set/Reset Control Word

Example #2: This example demonstrates how a Port C bit set/reset control word is constructed and issued to an 8255A. The bit set/reset control word is passed to the device through the use of an output instruction that references an 8080 I/O port address. The value of the I/O port address is determined by the 8080 CPU interface implemented. This example references the I/O port addresses realized by the simple 8080 to 8255A interface shown in Figure 3.

Control word (see Figure 7).



The assembly language program is:

```
CWR     EQU     0F8H            ;8255A #1 CONTROL WORD REGISTER
;*****  SET BIT 3
        MVI     A,00000111B     ;GET SET BIT 3 CONTROL WORD
        OUT     CWR             ;OUTPUT TO 8255A #1 CONTROL WORD REGISTER
;*****  RESET BIT 3
        MVI     A,00000110      ;GET RESET BIT 3 CONTROL WORD
        OUT     CWR             ;OUTPUT TO 8255A #1 CONTROL WORD REGISTER
```

NOTE: An MVI instruction is used to load the reset bit 3 control word into the A register. Since it is known that the set bit control word is already in the A register, a "DCR A" Instruction could be used to generate the correct control word and save one byte of code.

00000111 − 1 = 00000110 (RESET BIT 3 CONTROL WORD)

Example #3: This example demonstrates one simple method of performing a bit set/reset operation on Ports A and B. The state of any output port may be determined by reading the port. The assembly language program which may be used to set/reset Port A or B bits is:

```
PORTA   EQU     0F8H            ;8255A #1 PORT A
;*****  SET BIT 0
        IN      PORTA           ;GET STATE OF PORT
        ORI     01H             ;SET BIT 0
        OUT     PORTA           ;OUTPUT TO PORT
;*****  RESET BIT 0
        IN      PORTA           ;GET STATE OF PORT
        ANI     0FEH            ;RESET BIT 0
        OUT     PORTA           ;OUTPUT TO PORT
```


INTERRUPT CONTROL LOGIC STATUS WORDS

As previously mentioned, the 8255A Mode 1 and Mode 2 configurations support interrupt control logic. If a read of Port C is issued when the 8255A is configured in Mode 1, the software will receive the Mode 1 status word shown in Figure 8. The bits in the status word correspond to the state of the associated Port C lines (buffer full, interrupt request, etc.). The INTE bit shown in the status word corresponds to the interrupt enable flip-flop contained in the 8255A. This signal is not available externally. The structure of the Mode 1 status word varies as a function of the mode of the 8255A. Example #4 shows the status word which results from reading Port C from an 8255A which is configured with Port A Mode 1 input and Port B Mode 1 output.

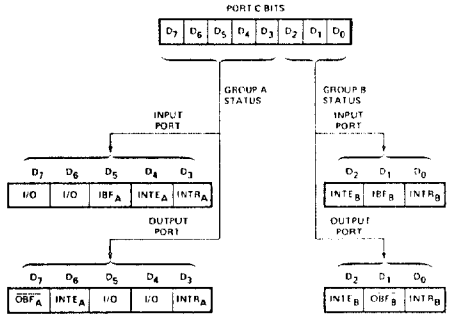


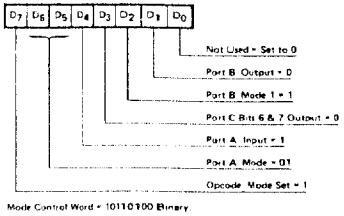
Figure 8. Mode 1 Status Word

Example #4 – MODE 1 STATUS WORD

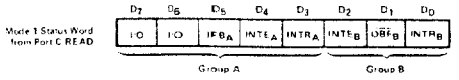
If an 8255A is to be configured through the use of the mode control word interface as:

Port A	Mode 1 Input
Port B	Mode 1 Output
Port C	Bits 6 & 7 Output

The following mode control word is used:



After the 8255A mode control word has been issued, a READ of Port C will obtain the following Mode 1 status word:



NOTE: The Port C I/O bits D_7 and D_6 should be modified through the use of the Port C bit set/reset command word. If a write to Port C is issued, the $INTE_A$ and $INTE_B$ bits may be inadvertently modified by the user. The IBF_A, $INTR_A$, $\overline{OBF}_B$, and $INTR_B$ bits will not be modified by either a write to Port C or a bit set/reset command. These four bits always reflect the state of the interrupt control logic.

Note that the Mode 2 status word (shown in Figure 9) differs from the Mode 1 status word. The format of the status word data bits D_2–D_0 are defined by the specification of the Port B configuration. Example #5 shows the structure of the Mode 2 status word when the 8255A is configured with Port A Mode 2 (bidirectional bus) and Port B Mode 1 input.

The Mode 1 and Mode 2 status words reflect the state of the interrupt logic supported by the 8255A.

Example #6 demonstrates how the interrupt enable bits are controlled through the use of the Port C bit set/reset feature. The application examples provide a more detailed explanation of the use of the Port C status word in the Mode 1 and Mode 2 configurations.

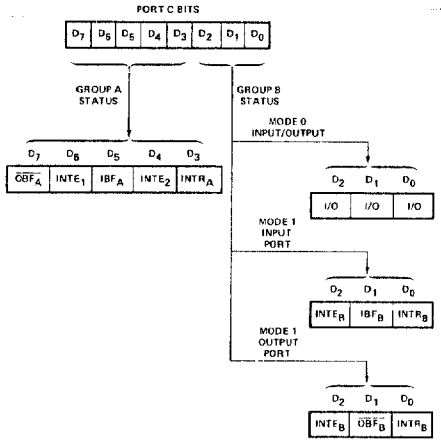


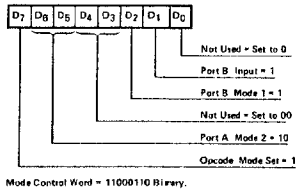
Figure 9. Mode 2 Status Word

Example #5 – MODE 2 STATUS WORD

If the 8255A is to be configured as follows:

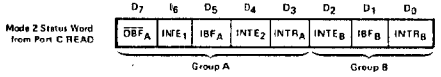
Port A	Mode 2 Bidirectional Bus
Port B	Mode 1 Input

The following mode control word is used:



Mode Control Word = 11000110 Binary.

After the 8255A mode control word has been issued, a read of Port C will obtain the following Mode 2 status word:



Example #6 – MODE 2 INTERRUPT ENABLE/DISABLE

The Mode 2 status word shown in Figure 9 contains two interrupt enable bits:

$INTE_1$ – Bit 6 – Enable output interrupts
$INTE_2$ – Bit 4 – Enable input interrupts

Bit set/reset control words may be constructed which may be used to control the INTE bits.

Set Bit 6 (Enable Output Interrupts) = 00001101 Binary

Reset Bit 6 (Disable Output Interrupts) = 00001100 Binary

Set Bit 4 (Enable Input Interrupts) = 00001001 Binary

Reset Bit 4 (Disable Input Interrupts) = 00001000 Binary

The control words shown were constructed from the standard bit set/reset format shown in Figure 7.

The value of CWR used in the following program example corresponds to the 8080 configuration shown in Figure 3.

```
CWR     EQU     0F8H            ;8255A #1 CONTROL WORD REGISTER
;*****  ENABLE INTERRUPTS FOR MODE 2 OUTPUT (SET PORT C BIT 6)
;*****
        MVI     A,00001101B     ;GET SET BIT 6 CONTROL WORD
        OUT     CWR             ;OUTPUT TO 8255 #1 CONTROL WORD REGISTER
;*****  DISABLE INTERRUPTS FOR MODE 2 OUTPUT (RESET PORT C BIT 6)
;*****
        MVI     A,00001100B     ;GET RESET BIT 6 CONTROL WORD
        OUT     CWR             ;OUTPUT TO 8255A #1 CONTROL WORD REGISTER
```


SOFTWARE CONSIDERATIONS

Regardless of the mode selected, the software must always issue the correct mode control word after a reset of the device. Generally, an initialization routine is constructed which issues the correct mode control word, sets up the initial state of the control lines, and initializes any program internal data.

Many of the software requirements of the 8255A vary as a function of the mode selected. The simplest mode supported by the device is Mode 0 (Basic Input/Output). Generally, Mode 0 is used for simple status driven device interfaces (no interrupts). Figure 10 illustrates sample software that could be used to support such interfaces. Most devices support a BUSY or READY signal which is used to determine when the device is ready to input or output data and a DATA STROBE which is used to request data transfer (DATA STROBE may easily be generated with the Port C bit set/reset feature). In the Mode 0 configuration, Ports A and B are used to input/output byte oriented data. Port C is used to input 8255A status, peripheral status and to drive peripheral control lines.

When the Mode 1 and Mode 2 configurations are used, the software is generally required to support interrupts. Software routines written for an interrupt driven environment tend to be more complex than status driven routines. The added complexity is due to the fact that interrupt driven systems are constructed such that other software tasks are run while the I/O transaction is in progress. A software routine that controls a peripheral device is generally referred to as a device driver. One method of implementing an interrupt driven device driver is to partition the device driver into a "Command Processor" and an "Interrupt Service Routine". The command processor is the module that validates

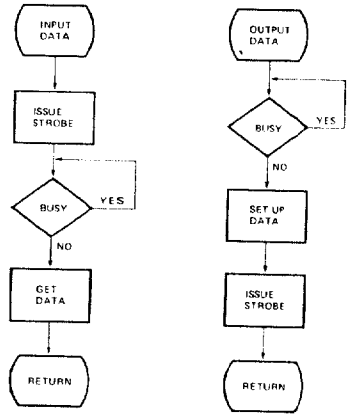


Figure 10. Sample Status Driven Software Flowchart

and initiates user program requests to the device driver. A common method of passing information between the various software programs is to have the requesting routine provide a device control block in memory. A sample device control block is shown in Table II.

Table II. Sample Device Control Block

NAME	DESCRIPTION
Status	This 1-byte field is used to transmit the status of the I/O transaction (busy, complete, etc.).
Opcode	This 1-byte field defines the type of I/O (READ, WRITE, etc.).
Buffer Address	This 2-byte field specifies the source/destination of the data block.
Character Count	This 1-byte field is a count of the number of characters involved in the transaction.
Character Transferred Count	This 1-byte count of the number of characters which were actually transferred.
Completion Address	This 2-byte field is the address of the user supplied completion routine which will be called after the I/O has been performed.

The command processor validates the transaction and initiates the operation described by the control block. Control is then returned to the requestor so that other processing may proceed. The interrupt service routine processes the remainder of the transaction.

The interrupt service routine supports the following functions:

1. The state of the machine (registers, status, etc.) must be saved so that it may be restored after the interrupt is processed.
2. The source of the interrupt must be determined. The hardware may support a register which indicates the interrupting device, or the software may poll the devices through interrogating the Port C status word of each 8255A.
3. Data must be passed to or from the device.
4. Control must be passed to the requesting routine at the completion of the I/O.
5. The state of the machine must be restored before returning to the interrupted program.

Figures 11 and 12 are simplified flowcharts of one of the many methods of implementing command processor and interrupt service routine modules.

The rest of this application note presents specific application examples. All of the 8080 assembly language programs supplied with the application examples use the standard Intel 8080 assembly language mnemonics. The programs discussed use the program equate statement to specify all hardware related data. Equate statements are used so that all references to an I/O port may be changed through a simple reassignment of the port address in the equate statement.

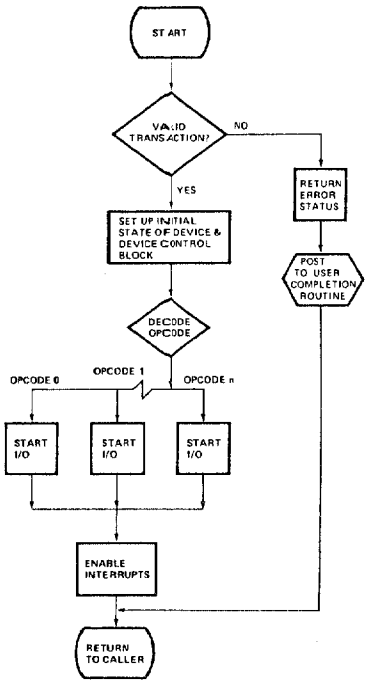


Figure 11. Command Processor

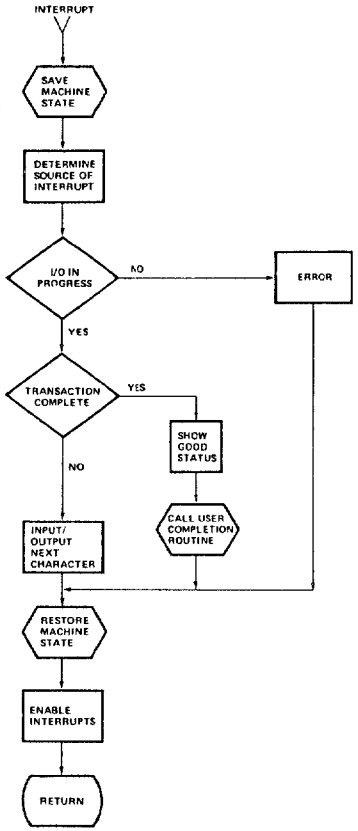


Figure 12. Interrupt Service Routine

MODE 0 – STATUS DRIVEN PERIPHERAL INTERFACE

This design example shows how a single 8255A in Mode 0 may be used to develop a status driven interface (no interrupts) for the Centronics 306 character printer, the Remex paper tape punch, and the Remex paper tape reader.

8255A To Peripheral Hardware Interface

The first step in the design is to examine the specification for the peripheral devices and identify the control and data signals which must be supported by the interface. Table III lists the signals which were chosen to be supported by the 8255A interface. All three of the devices support the standard BUSY/DATA STROBE interface discussed previously (see Figure 10). Figure 13 is a block diagram of the interface design. The 8255A Port A is configured as a Mode 0 output port which is used to support the printer and the paper tape punch data bus. Port B is configured as a Mode 0 input port and is used to input the paper tape reader data. Three of the Port C lower bits (PC_2–PC_0) configured in input mode are used to input the device busy indications. Three of the Port C upper bits (PC_6–PC_4) configured in output mode are used to support the device strobe signals required by each device.

Table III. Peripheral Interface Signals

CHARACTER PRINTER	
Name:	DATA 0–DATA 7
Definition:	Input data levels. A high signal represents a binary 1 and a low signal represents a binary 0. These eight lines are the data lines to the printer.
Name:	$\overline{\text{DATA STROBE}}$
Definition:	A 0.5 μsec pulse (minimum) used to transfer data from the 8255A to the printer.
Name:	BUSY
Definition:	The level indicating that the printer cannot receive data.
PAPER TAPE PUNCH	
Name:	TRACKS 1–8 DATA INPUT
Definition:	Input data levels. A high signal causes a hole to be punched on the associated track. These eight lines are the data lines to the printer.
Name:	PUNCH COMMAND INPUT
Definition:	A true condition moves the tape and initiates punching the tape. This signal is actually a data strobe.
Name:	PUNCH READY OUTPUT
Definition:	True signal indicates that the punch is ready to accept a punch command. This is the punch busy line.
PAPER TAPE READER	
Name:	DATA TRACK OUTPUTS
Definition:	True signal indicates data track hole. These eight lines are the data lines from the punch.
Name:	$\overline{\text{DRIVE RIGHT}}$
Definition:	True signal drives the tape to the right and reads a character. This signal is actually the data strobe (initiate read signal).
Name:	DATA READY OUTPUT
Definition:	True signal indicates data track outputs are in "On character" condition. This signal is the reader busy line.

The drive requirements of the interface lines are a function of the peripheral interface circuitry, the length of the interface cable, and the environment in which the unit is running. In this particular design example, all output lines from the 8255A to the peripherals were buffered through a 7407 buffer/driver. The input lines from the peripherals were fed directly into the Port C and Port B inputs.

8080 CPU Module To 8255A Interface

The schematic of the completed hardware design is shown in Figure 14. The CPU module design shown is the design which was implemented for Intel's SDK 80 kit board. The 8255A is addressed through the use of an isolated I/O architecture utilizing a linear select scheme. Address bits A_1 and A_0 are used to select the 8255A port. Address bit A_3 is the exclusive enable for 8255A #1. Examination of the schematic shows that all of the 8255A interface lines are directly driven by the CPU module.

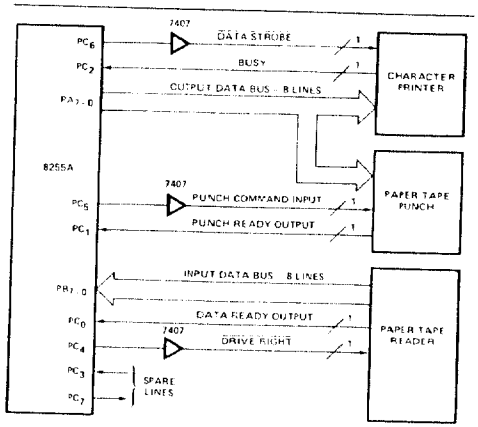


Figure 13. Interface Block Diagram

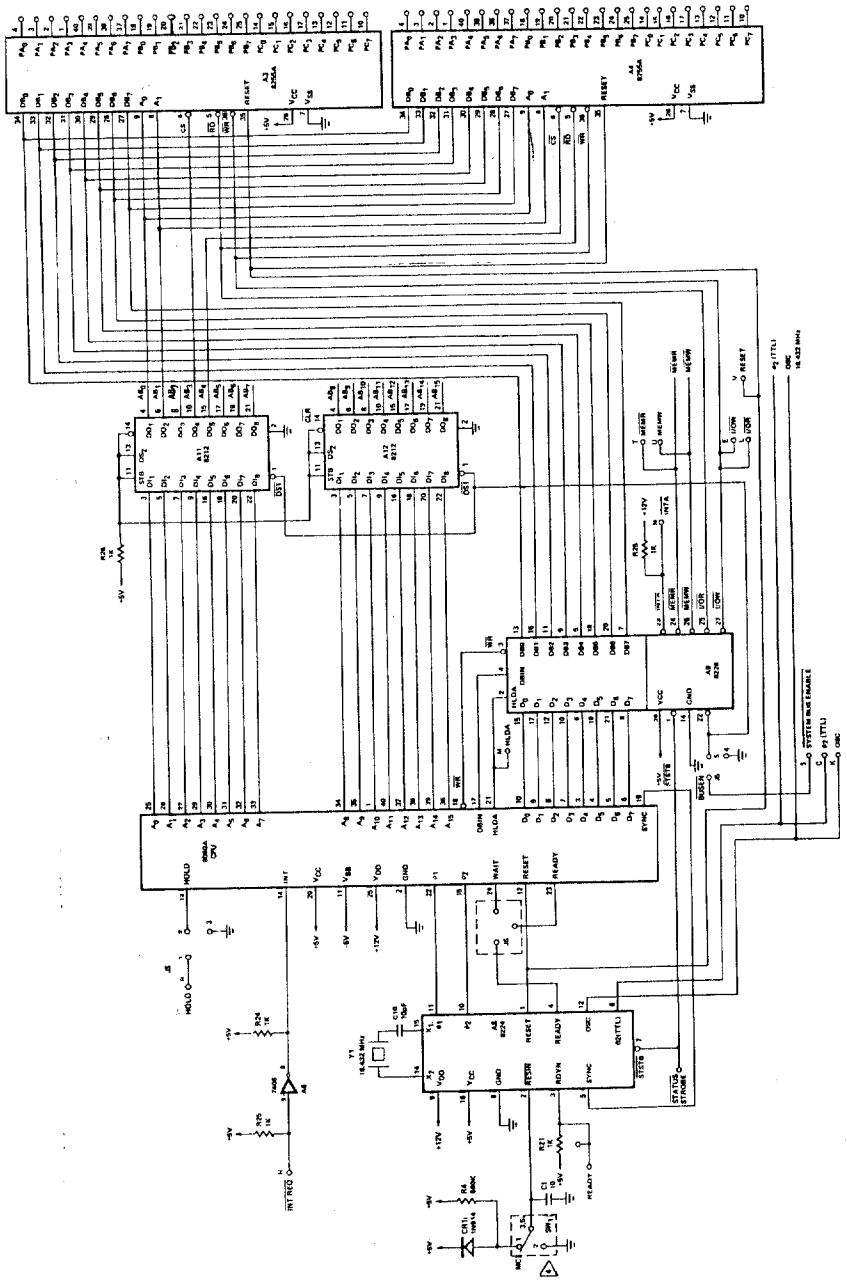
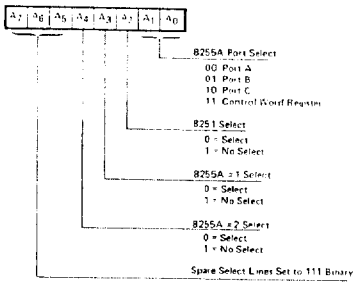


Figure 14. SDK 80 Schematic

Mode 0 Interface Software

An initialization routine and three device drivers (one for each peripheral device) are required to support the peripheral interface. The I/O port addresses implemented by the hardware are shown in Figure 15. The unused chip select bits are set to one so that chip select conflicts will not result if the unused bits are required by an expanded system.



Port Selected	Port Select Character (In Hexadecimal)
Port A 8255A #1	F4
Port B 8255A #1	F5
Port C 8255A #1	F6
Control Word Register 8255A #1	F7
Port A 8255A #2	EC
Port B 8255A #2	ED
Port C 8255A #2	EE
Control Word Register 8255A #2	EF

Figure 15. I/O Port Addresses

```
ISIS 8080 MACRO ASSEMBLER, V4.0                         PAGE 1
MODE ZERO EXAMPLE

                        TITLE 'MODE ZERO EXAMPLE'
                    ;*****
                    ;       CHARACTER PRINTER, PAPER TAPE PUNCH, PAPER TAPE READER
                    ;       MODE ZERO EXAMPLE
                    ;*****
                    ;*****
                    ;       PROGRAM EQUATES
                    ;*****
00F4                PORTA   EQU     0F4H    ; 8255 PORT A
00F5                PORTB   EQU     0F5H    ; 8255 PORT B
00F6                PORTC   EQU     0F6H    ; 8255 PORT C
00F7                CWR     EQU     0F7H    ; 8255 CONTROL WORD REGISTER
                    ;*****
                    ;       INITIALIZATION CONTROL WORD
                    ;
                    ;       USED TO CONFIGURE THE 8255 AS FOLLOWS:
                    ;
                    ;               PORT A - OUTPUT MODE ZERO
                    ;               PORT B - INPUT MODE ZERO
                    ;               PORT C (UPPER) - OUTPUT
                    ;               PORT C (LOWER) - INPUT
                    ;
                    ;*****
00B1                ICW     EQU     10110001B       ; INITIALIZATION CONTROL WORD
                    ;*****
                    ;       SET/RESET CONTROL WORDS FOR GENERATION OF DATA STROBES
                    ;       ON PORT C.
                    ;*****
00CF                PRSON   EQU     00001111B       ; PRINTER DATA STROBE ON
00CE                PRSOF   EQU     00001110B       ; PRINTER DATA STROBE OFF
00CB                PNCHN   EQU     00001011B       ; PUNCH DATA STROBE ON
00CA                PNCOF   EQU     00001010B       ; PUNCH DATA STROBE OFF
00C9                RDRON   EQU     00001001B       ; READER DATA STROBE ON
00C8                RDROF   EQU     00001000B       ; READER DATA STROBE OFF
                    ;*****
                    ;       BIT MASK FOR READER BUSY FLAG
                    ;*****
0080                PRBSY   EQU     10000000B       ; PRINTER BUSY
0040                PNBSY   EQU     01000000B       ; PUNCH BUSY
0020                RDRBY   EQU     00100000B       ; READER BUSY
```

Note that the initialization routine issues the mode control word (shown in Figure 16). It also sets the low true DATA STROBE signals to an inactive (high) state.

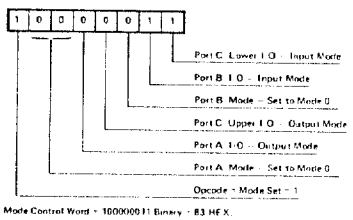


Figure 16. Mode Control Word

The three peripheral drivers which follow all have the basic structure discussed previously. Consider the printer routine. Here the user routine places an ASCII data character in the C-register and passes control to the LPST location through a subroutine call. The printer driver interrogates the status of the printer by reading Port C. If the printer is busy, the routine will loop until the printer is idle. When the printer is ready to accept a data character, the character is placed on the Port A lines and a DATA STROBE is generated. After generating the DATA STROBE, the driver executes a subroutine return to the caller.

The DATA STROBE signals to the devices are generated through the use of the Port C bit set/reset feature. The bit set/reset control words used are shown in Figure 17.

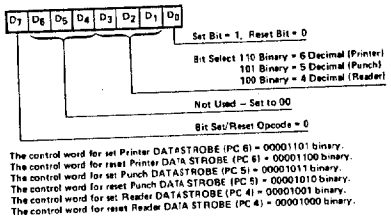
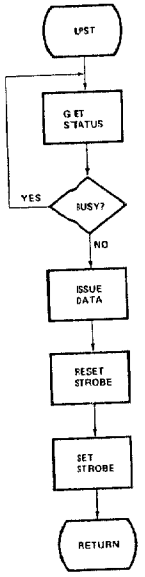


Figure 17. Bit Set/Reset Control Words

Summary/Conclusions

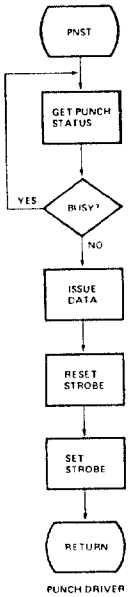
This design example discussed the basic hardware and software required to handle a simple device interface. The 8255A will easily accommodate a more complex interface design which utilizes additional interface lines supported by the peripheral. For instance, one of the spare Port C output lines may be used to control the punch direction. Support of this additional feature would require minor modification of the device driver so that the punch direction line could be specified by the user routine.

Through consideration of this example, the use of the 8255A in Mode 0 should become evident.



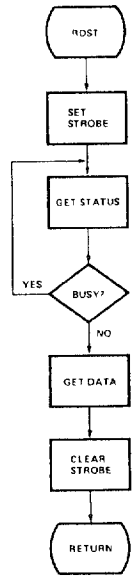
PRINTER DRIVER

```
8085 8080 MACRO ASSEMBLER, V1.0                    PAGE 3
MODE ZERO EXAMPLE - CHARACTER PRINTER DRIVER

                ;*****
                ;
                ;       CHARACTER PRINTER DRIVER
                ;
                ;               INPUTS : CHARACTER TO PRINT IN C-REG
                ;               OUTPUTS: CHARACTER TO PRINTER
                ;
                ;       A REGISTER MODIFIED
                ;
                ;*****
                LPST:
3000 DBF6               IN      PORTC   ; GET STATUS OF PRINTER
3002 E604               ANI     LPBIT   ; SEE IF BUSY
3004 C200               JNZ     LPST    ; IF BUSY - JUMP TO LPST (WAIT LOOP)
                ;
                ;*****
                ;       PRINTER IS IDLE - OUTPUT A CHARACTER
                ;*****
3007 79                 MOV     A,C     ; GET DATA BYTE SUPPLIED BY CALLER
3008 D3F5               OUT     PORTA   ; OUTPUT DATA TO DATA LINES
300A 3E0C               MVI     A,LPSOF ; GET DATA STROBE CONTROL WORD
300C D3F7               OUT     CWR     ; RESET DATA STROBE (LOW TRUE SIGNAL)
300E 3C                 INR     A       ; GENERATE SET DATA STROBE CONTROL WORD
300F D3F7               OUT     CWR     ; SET DATA STROBE
3011 C9                 RET             ; RETURN TO CALLER
```

PUNCH DRIVER

```
8085 8080 MACRO ASSEMBLER, V1.0                    PAGE 4
MODE ZERO EXAMPLE - PAPER TAPE PUNCH DRIVER

                ;*****
                ;
                ;       PAPER TAPE PUNCH DRIVER
                ;
                ;               INPUTS : DATA TO PUNCH IN C-REGISTER
                ;               OUTPUTS: DATA TO PUNCH
                ;
                ;       A REGISTER MODIFIED
                ;
                ;*****
                PNST:
3011 DBF6               IN      PORTC   ; GET STATUS OF PUNCH
3013 E602               ANI     PNBSY   ; SEE IF BUSY
3015 C21130             JNZ     PNST    ; IF BUSY - JUMP TO PNST (WAIT LOOP)
                ;
                ;*****
                ;       PUNCH IS IDLE - OUTPUT A CHARACTER
                ;*****
3018 79                 MOV     A,C     ; GET DATA BYTE SUPPLIED BY CALLER
3019 D3F5               OUT     PORTA   ; OUTPUT DATA TO DATA LINES
301B 3E0B               MVI     A,PNSON ; SET DATA STROBE CONTROL WORD
301D D3F7               OUT     CWR     ; SET DATA STROBE
301F 3D                 DCR     A       ; GENERATE RESET DATA STROBE CONTROL WORD
3020 D3F7               OUT     CWR     ; RESET DATA STROBE
3022 C9                 RET             ; RETURN TO CALLER
```



READER DRIVER

MODE 1 INTERRUPT DRIVEN PRINTER INTERFACE

The status driven interface described in the previous example required the software driver to poll the device status for completion. An alternate approach is to construct the device interface such that an interrupt is used to signal the completion of the operation. When an interrupt driven interface is utilized, the time that was dedicated to polling can be used to perform other functions and the effective processor through-put is increased. This example demonstrates how an 8255A configured in Mode 1 may be used to develop an interrupt driven interface for the Centronics 306 character printer.

CPU Module To 8255A Interface

The 8080 bus interface implemented for this example is the same as the Mode 0 example with the addition of interrupt support. Interrupt support is implemented through the use of a special feature of the 8228 System Controller. If only one interrupt vector is required (such as in small systems), the 8228 can automatically assert an RST 7 instruction onto the data bus at the proper time. This option is selected by connecting the $\overline{INTA}$ output of the 8228 to the +12-volt supply through a 1K ohm series resistor.

The Mode 1 interrupt support logic of the 8255A provides an interrupt request line for each port. The 8255A interrupt request line ($INTR_A$) must be connected to the INT line of the 8080. A 10K ohm pullup resistor is used to insure that the V_{IH} requirements of the 8080 are met.

8255A To Peripheral Interface

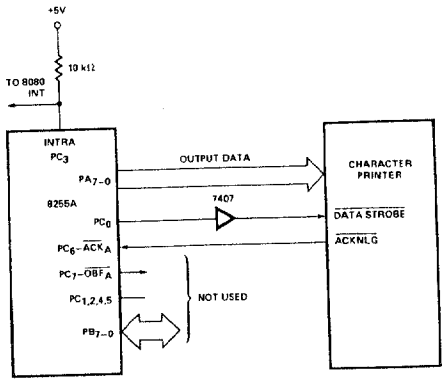
The interrupt driven configuration control signal interface to the printer is different than the status driven interface. Instead of a BUSY/DATA STROBE interface, a DATA STROBE/ACK interface is supported. The ACK signal notifies the 8255A that a character transferred to the printer by a DATA STROBE has been accepted. After an ACK is issued the printer is considered idle. The block diagram shown in Figure 18 displays the interface signals used.

The Mode 1 interrupt driven peripheral support signals used are:

PA_7–PA_0 – Output Data
Used to support the printer data port.

$\overline{OBF}$ – Output Buffer Full
This line goes low when data is placed in the output buffer. The OBF signal may be used as a data strobe signal when interfacing to peripherals which do not require a pulsed input. The Centronics 306 requires a pulsed DATA STROBE signal. This signal is supported by Port C bit 0.

$\overline{ACK}$ – ACKnowledge
This line is used to signal the 8255A that the device has accepted the data. This line is supported by the printer ACKNLG signal.



NOTES:
1. DATA BUS BUFFERED WITH 7407.
2. ALL 8255A OUTPUT LINES ARE PULLED UP TO +5 V AT THE PERIPHERAL.

Figure 18. Interface Block Diagram

Mode 1 Software Driver

The software driver implemented for this example utilizes the typical interrupt driven software structure outlined previously. The initialization routine issues the mode control word (shown in Figure 19) to the 8255A after reset of the device. The initialization routine also places a jump to the interrupt service routine in the interrupt location for RST 7. The command processor is started by the user routine through a subroutine call to PSTRT, with the address of the control block in the D and E registers (the control block format is shown in Table IV). The command processor insures that an I/O operation is not already in progress, starts the I/O, enables interrupts, and returns to the caller so that other processing may proceed.

After a character is placed in the output buffer, the DATA STROBE signal is generated through the use of the Port C bit set/reset feature. When the ACK is generated by the printer, the buffer full indication is cleared and the 8255A generates an interrupt. If interrupts are enabled, the interrupt request is serviced by the 8080 CPU through disabling processor interrupts and then executing the instruction at location 38 hexadecimal in program memory. The interrupt service routine saves the processor state and polls the 8255A to determine the source of the interrupt. Once the interrupting device is located, the control block is used to locate the next data character for transfer to the 8255A output buffer. After the entire buffer has been printed, the interrupt service routine passes control to the user-supplied completion routine. Before returning from the interrupt, the state of the processor is restored.

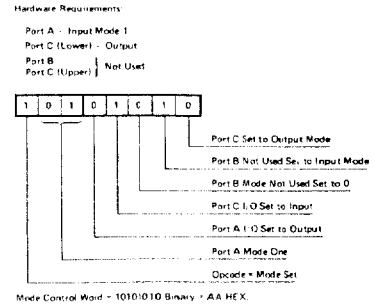


Figure 19. Mode Control Word

Table IV. Printer Software Control Block

NAME	POSITION	DEFINITION
Status	Byte 0	A 1-byte field which defines the completion status of an I/O. 00 = Good completion 01 = Error – command already in progress
Buffer Address	Byte 1, 2	Pointer to the start of the data to print.
Character Count	Byte 3	Count of the number of characters to print.
Character Transferred Count	Byte 4	The number of characters transferred.
Completion Address	Byte 5, 6	Address of a user supplied routine which will be called after the I/O has been performed.

NOTES:
1. An opcode field is not required because WRITE is the only operation performed.
2. The control block must reside above location FF Hex.

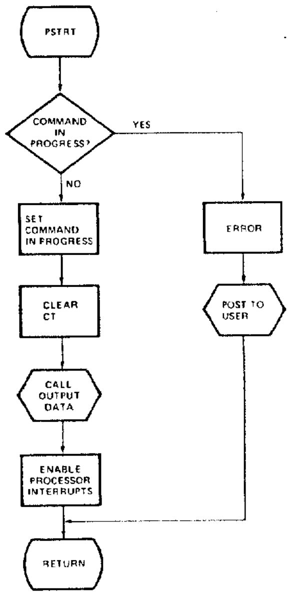
There are a number of error conditions which may occur, such as an interrupt from a device which does not have a control block in progress, or an interrupt when polling establishes that no device requires service. Neither of these errors should occur, but if they do, the driver should perform in a consistent fashion. The recovery routines implemented to handle error conditions are determined by the particular applications environment.

Summary/Conclusions

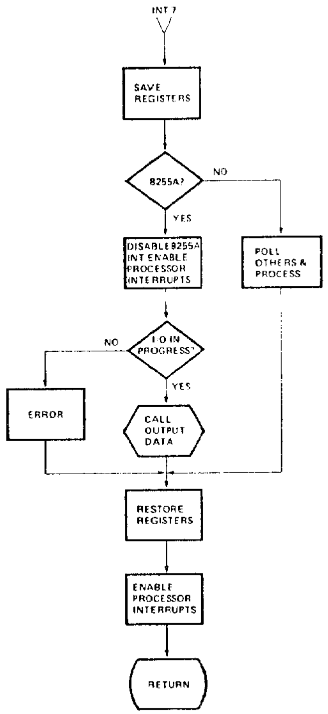
When utilized in a small system design, the 8255A interrupt support logic provides all of the capabilities required to implement an interrupt driven hardware interface without the use of external logic. In larger system designs, the designer may chose to use additional hardware to determine the source of interrupt requests without software polling. The software design required by an interrupt driven system is inherently more complex than the status driven interface. If an interrupt driven system is required the added complexity is a small price to pay for a significant increase in system through-put.

```
ISIS 8080 MACRO ASSEMBLER, V1.0                PAGE 1
MODE ONE EXAMPLE

                        TITLE 'MODE ONE EXAMPLE'
               ;*****
               ;
               ;        CHARACTER PRINTER - INTERRUPT DRIVEN
               ;        MODE ONE EXAMPLE
               ;
               ;*****
               ;
               ;*****
               ;        PROGRAM EQUATES
               ;*****
00F4           PORTA    EQU     0F4H    ; 8255 PORT A
               PORTB    EQU     0F5H    ; 8255 PORT B
00F6           PORTC    EQU     0F6H    ; 8255 PORT C
00F7           CWR      EQU     0F7H    ; 8255 CONTROL WORD REGISTER
0038           RST7     EQU     038H    ; RESTART 7 ADDRESS
               ;
               ;*****
               ;
               ;        INITIALIZATION CONTROL WORD
               ;
               ;                USED TO CONFIGURE THE 8255 AS FOLLOWS:
               ;
               ;                        PORT A - OUTPUT MODE 1
               ;                        PORT B - INPUT MODE 0 (NOT USED)
               ;                        PORT C LOWER - OUTPUT
               ;
               ;
               ;*****
00AA           ICW      EQU     10101010B       ; INITIALIZATION CONTROL WORD
               ;*****
               ;        SET/RESET CONTROL WORDS
               ;*****
0001           STBON    EQU     00000001B       ; SET STROBE
0000           STBOF    EQU     00000000B       ; RESET STROBE
               ;*****
               ;        8255 ENABLE/DISABLE INTERRUPT CONTROL WORDS
               ;*****
000D           IEN      EQU     00001101B       ; ENABLE INTERRUPTS
000C           IDN      EQU     00001100B       ; DISABLE INTERRUPTS
               ;*****
               ;        DEVICE STATUS EQUATES
               ;*****
0080           LPBSY    EQU     080H    ; BUFFER FULL (LINE PRINTER BUSY)
0008           INTRA    EQU     08H     ; INTERRUPT REQUEST
```

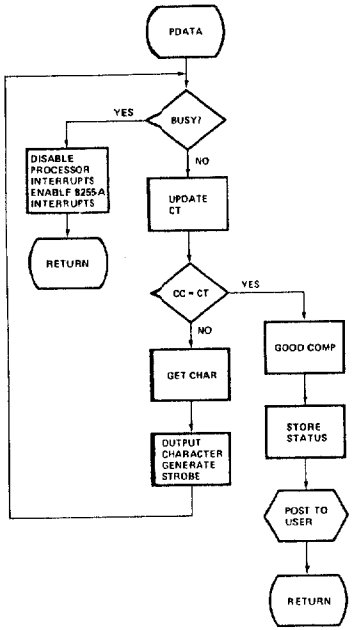
```
ISIS 8080 MACRO ASSEMBLER, V1.0                PAGE 2
MODE ONE EXAMPLE

               ;*****
               ;        CONTROL BLOCK EQUATES
               ;*****
0000           CBST     EQU     00H             ; STATUS BYTE
0001           CBUF     EQU     01H             ; BUFFER ADDRESS
0003           CBCC     EQU     03H             ; CHARACTER COUNT
0004           CBCT     EQU     04H             ; CHARACTER TRANSFERED COUNT
0005           CBCMP    EQU     05H             ; COMPLETION SERVICE ADDRESS
               ;*****
               ;        COMPLETION STATUS EQUATES
               ;*****
0000           STGD     EQU     00H             ; GOOD COMPLETION
0001           STE1     EQU     01H             ; ERROR - COMMAND ALREADY IN PROGRESS
               ;*****
               ;        PROGRAM ORIGIN
               ;*****
3000                    ORG     03000H
               ;*****
               ;
               ;        INITIALIZATION ROUTINE
               ;
               ;        A,H,L REGISTERS MODIFIED
               ;
               ;*****
               INIT:
3000 3EAA               MVI     A,ICW   ; GET MODE CONTROL WORD
3002 D3F7               OUT     CWR     ; OUTPUT TO CONTROL WORD REGISTER
3004 3E01               MVI     A,STBON ; GET SET DATA STROBE CONTROL WORD
3006 D3F7               OUT     CWR     ; SET DATA STROBE (LOW TRUE SIGNAL)
               ;*****
               ;        SET UP RESTART 7 LOCATION WITH JUMP TO PINT
               ;*****
3008 3EC3               MVI     A,0C3H  ; GET "JMP"
300A 323800             STA     RST7    ; PLACE IN RST7 LOCATION
300D 213030             LXI     H,PINT  ; GET ADDRESS OF INTERRUPT SERVICE ROUTINE
3010 223900             SHLD    RST7+1  ; STORE ADDRESS
3013 C9                 RET             ; RETURN TO CALLER
```


```
ISIS 8080 MACRO ASSEMBLER, V1.0                PAGE 3
COMMAND PROCEDURE

               ;*****
               ;
               ;        COMMAND PROCEDURE
               ;
               ;                INPUTS: CONTROL BLOCK ADDRESS IN D AND E REGISTERS
               ;
               ;                OUTPUTS: STATUS [illegible] OR ERROR STATUS IN CONTROL BLOCK
               ;
               ;        A,H,L REGISTERS MODIFIED
               ;
               ;*****
               PSTRT:
[illegible]             LDA     PIPRG+1 ; GET PRINT IN PROGRESS BLOCK ADDRESS
[illegible]             ANA     A       ; SET IF ZERO (PRINT IN PROGRESS)
                                        ; IF BLOCK ADDRESS NOT EQUAL TO ZERO THEN
                                        ; PRINT IN PROGRESS
[illegible]             JNZ     PSTE    ; IF SET - BRANCH TO ERROR
[illegible] EB          XCHG
[illegible]             SHLD    PIPRG   ; SAVE CONTROL BLOCK ADDRESS
[illegible] EB          XCHG
[illegible]             LXI     H,CBCT  ; GET INDEX TO CT
[illegible]             DAD     D       ; COMPUTE ADDRESS OF CT
[illegible]             MVI     M,00H   ; CLEAR CT
[illegible]             CALL    PDATA   ; START I/O
[illegible] FB          EI              ; ENABLE PROCESSOR INTERRUPTS
[illegible]             RET             ; RETURN TO CALLER
               ;*****
               ;        ERROR - TRANSACTION ALREADY IN PROGRESS
               ;*****
               PSTE:
[illegible]             MVI     A,STE1  ; GET ERROR STATUS CODE
[illegible]             JMP     POST    ; PASS CONTROL TO COMPLETION ROUTINE
```




```
ISIS 8080 MACRO ASSEMBLER, V1.0                PAGE 4
PRINTER INTERRUPT SERVICE ROUTINE

               ;*****
               ;        PRINTER INTERRUPT SERVICE ROUTINE
               ;        ALL REGISTERS SAVED AND RESTORED
               ;*****
               PINT:
[illegible] F5          PUSH    PSW     ; SAVE PSW
[illegible]             PUSH    B       ; SAVE REGISTER PAIR B AND C
[illegible]             PUSH    D       ; SAVE REGISTER PAIR D AND E
[illegible]             PUSH    H       ; SAVE REGISTER PAIR H AND L
               ;*****
               ;        [illegible]
               ;*****
[illegible]             IN      PORTC   ; GET STATUS [illegible] DEVICE
[illegible]             ANI     INTRA   ; SEE IF INT
[illegible]             JZ      PPOLL   ; NO - BRANCH TO POLL OTHER DEVICES IF ANY
[illegible]             MVI     A,IDN   ; [illegible]
[illegible]             OUT     CWR     ; DISABLE DEVICE INTERRUPTS
[illegible] FB          EI              ; ENABLE PROCESSOR INTERRUPTS
[illegible]             LHLD    PIPRG   ; GET CONTROL BLOCK ADDRESS
[illegible] AF          XRA     A       ; CLEAR A REG
[illegible]             CMP     M       ; [illegible] PRINT IN PROGRESS
[illegible]             JZ      [illegible] ; [illegible] ERROR ROUTINE
[illegible] EB          XCHG
[illegible]             CALL    PDATA   ; PRINT DATA
               ;*****
               ;        RESTORE REGISTERS AND RETURN FROM INTERRUPT
               ;*****
               PRTN:
[illegible]             POP     H       ; RESTORE REGISTER PAIR H AND L
[illegible]             POP     D       ; RESTORE REGISTER PAIR D AND E
[illegible]             POP     B       ; RESTORE REGISTER PAIR B AND C
[illegible]             POP     PSW     ; RESTORE PSW
[illegible] FB          EI              ; ENABLE PROCESSOR INTERRUPTS
[illegible]             RET             ; RETURN TO INTERRUPTED PROCESS
               ;*****
               ;        POLL OTHER DEVICES IF ANY
               ;                IF NO OTHER DEVICES TO POLL - USER SUPPLIED ERROR
               ;                RECOVERY ROUTINE
               ;*****
               PPOLL:
[illegible]             JMP     PRTN    ; RETURN
               ;*****
               ;        ERROR - INTERRUPT FROM IDLE DEVICE
               ;                USER SUPPLIED ERROR RECOVERY ROUTINE
               ;*****
               PIERR:
[illegible]             JMP     PRTN    ; RETURN
```




```
ISIS 8080 MACRO ASSEMBLER, V1.0                PAGE 5
PRINTER OUTPUT DATA ROUTINE


                        ;*****
                        ;
                        ;       PRINTER OUTPUT DATA ROUTINE
                        ;
                        ;       CONTROL BLOCK ADDRESS IN D AND E REG
                        ;
                        ;*****
                        PDATA:
3058 DBF6                       IN      PORTC   ; GET STATUS OF DEVICE
305A E680                       ANI     1FEH    ; SEE IF BUSY (BUFFER FULL)
305C CA6830                     JZ      PRTD    ; IF BUSY - BRANCH
305F 216900                     LXI     H,OBCT  ; SET INDEX TO CT
3062 19                         DAD     D       ; COMPUTE ADDRESS OF CT
3063 7E                         MOV     A,M     ; GET CT
3064 3D                         DCR     M       ; DEC CT
3065 2B                         DCX     H       ; SEE IF EQUAL
3066 BE                         CMP     M
3067 CA8A30                     JZ      PCOMP   ; IF EQUAL - DONE SO TELL USER
306A 216900                     LXI     H,OBUF  ; SET INDEX TO BUFFER ADDRESS
306D 19                         DAD     D       ; COMPUTE ADDRESS OF BUFFER ADDRESS
306E D5                         PUSH    D       ; SAVE D AND E REGISTERS
306F 5E                         MOV     E,M     ; GET LSB OF BUFFER ADDRESS
3070 23                         INX     H       ; INC TO NEXT BYTE
3071 56                         MOV     D,M     ; GET BUFFER MSB
3072 2B00                       MVI     H,00H   ; CLEAR H REG
3074 6F                         MOV     L,A     ; GET CT
3075 19                         DAD     D       ; COMPUTE CHARACTER ADDRESS
3076 7E                         MOV     A,M     ; GET CHARACTER
3077 D3FA                       OUT     PORTA   ; OUTPUT CHARACTER TO PRINTER
3079 3E00                       MVI     A,1TROF ; RESET DATA STROBE (LOW TRUE SIGNAL)
307B D3F7                       OUT     CWP
307D 3C                         INR     A       ; GENERATE SET CONTROL WORD
307E D3F7                       OUT     CWP     ; SET DATA STROBE
3080 D1                         POP     D       ; RESTORE CONTROL BLOCK ADDRESS
3081 C35830                     JMP     PDTA    ; LOOP UNTIL BUSY


                        ;*****
                        ;
                        ;       PRINTER BUSY - RETURN
                        ;
                        ;*****
                        PRTD:
3084 F3                         DI              ; DISABLE INTERRUPTS
3085 3E0D                       MVI     A,1EN   ; ENABLE DEVICE INTERRUPTS
3087 D3F7                       OUT     CWP     ; SET INTERRUPT ENABLE
3089 C9                         RET             ; RETURN TO CALLER
                        ;*****
                        ;       POST GOOD COMPLETION TO USER
                        ;
                        ;*****
                        PCOMP:
308A 3E00                       MVI     A,STGD  ; GET GOOD STATUS CODE
                                CALL    POST    ; POST TO USER
308F AF                         XRA     A       ; CLEAR A REG
3090 32A230                     STA     PIPRG-1 ; CLEAR COMMAND IN PROGRESS ADDRESS
3093 C9                         RET             ; RETURN TO CALLER


                        ;*****
                        ;
                        ;       POST TO USER COMPLETION ROUTINE
                        ;
                        ;       INPUTS : STATUS CODE IN A REG
                        ;                CONTROL BLOCK ADDRESS IN D AND E REG
                        ;       OUTPUTS: PASSES CONTROL TO USER COMPLETION ADDRESS
                        ;                SPECIFIED IN CONTROL BLOCK
                        ;                WITH CONTROL BLOCK ADDRESS IN D AND E
                        ;
                        ;       A,H,L,B,C REG MODIFIED
                        ;
                        ;*****
                        POST:
3094 EB                         XCHG
3095 77                         MOV     M,A     ; UPDATE STATUS
3096 EB                         XCHG
3097 215960                     LXI     H,OBCOMP ; GET INDEX TO COMPLETION ADDRESS
309A 19                         DAD     D       ; COMPUTE ADDRESS
309B 4E                         MOV     C,M     ; GET LSB OF COMPLETION ADDRESS
309C 23                         INX     H       ; INC TO NEXT BYTE
309D 46                         MOV     B,M     ; GET MSB  OF COMPLETION ADDRESS
309E C5                         PUSH    B       ; PUSH ADDRESS INTO STACK
309F C9                         RET             ; PASS CONTROL TO USER ROUTINE
                        ;*****
                        ;       DATA AND TABLES
                        ;*****
30A1 0000               PIPRG:  DW      0       ; IN PROGRESS CONTROL BLOCK ADDRESS
                                                ; IF DATA = 0 NO CONTROL BLOCK IN PROGRESS
                                                ; IF DATA NOT EQUAL TO ZERO CONTROL BLOCK IN PROGRESS
                        ;*****
                        END OF MODE ONE EXAMPLE
                        ;*****
0000                            END
```


MODE 2 – 8080 TO 8080 INTERFACE

Due to the drastic reduction of hardware costs, system designs which utilize multiple CPU Modules are becoming more common. An 8080 may be configured as a master CPU and used to control multiple 8080 slave modules which act as intelligent I/O controllers. When multiple CPUs are utilized, a method of processor intercommunication must be supported. Figure 20 is a block diagram of one method of implementing a master/slave interface through the use of the 8255A Mode 2 bidirectional bus.

Hardware Discussion

Two complete 8080 systems are required for this example. Intel's SBC 80/10 OEM board is used as the master CPU module and Intel's SDK 80 board is used as the slave CPU. The SBC 80/10 supports an 8255A which is configured in Mode 2. The 8255A is selected through the use of a decoded select scheme. Through the use of the 8228 RST 7 interrupt feature, a simple interrupt structure is supported. The SDK 80 is configured without interrupts for this example. The external logic required for this example is associated with the slave CPU. Simple logic is implemented which allows the slave CPU to generate the $\overline{ACK}$ and $\overline{STB}$ signals required to READ from and WRITE to the 8255A bidirectional bus with a single I/O instruction.

The system shown in Figure 20 utilizes SSI logic to read the 8255A IBF and OBF signals. If two spare 8255A input lines are available they could be used to input the IBF and OBF signals and eliminate the SSI logic.

Software Discussion

Two sets of software are required to support the processor to processor interface. The master resident software which follows conforms to the simple interrupt driven software structure outlined previously. The initialization routine issues the Mode 2 control word to the 8255A after device reset. The command processor accepts READ/WRITE control blocks which provide a simple user interface for transferring data to/from the slave CPU. The master software is capable of processing both a read and a write control block simultaneously. The slave resident software shown at the end of this example utilizes the status driven approach.

Summary/Conclusions

It is important to note that this design may be expanded to include more slave CPUs by simply adding another 8255A to the master module for each slave. The software drivers discussed address only the passing of data between the two processors. Specific applications generally dictate a software protocol be implemented for information transfer.

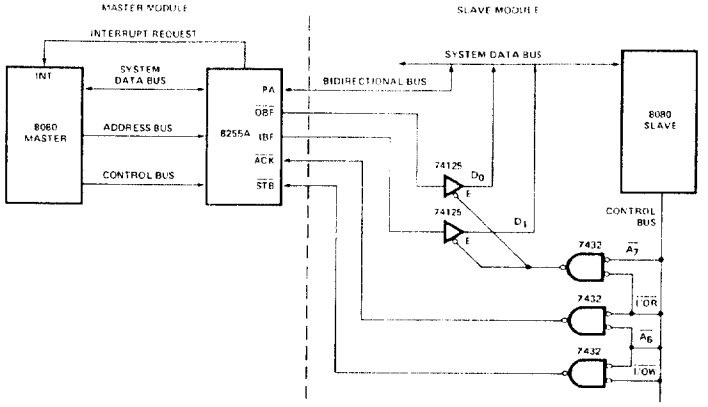
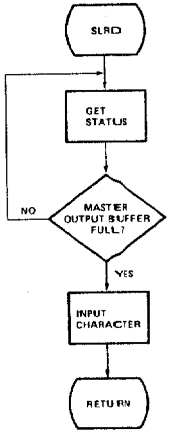
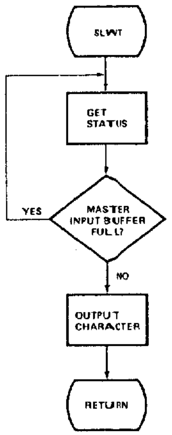


Figure 20. Interface Block Diagram



SLAVE READ ROUTINE

```
ISIS 8080 MACRO ASSEMBLER, V1.0                    PAGE 1
MODE TWO EXAMPLE - SLAVE SOFTWARE

                             TITLE 'MODE TWO EXAMPLE - SLAVE SOFTWARE'
                     ;*****
                     ;
                     ;       8080 MASTER TO 8080 SLAVE INTERFACE
                     ;               - SLAVE SOFTWARE -
                     ;               MODE TWO EXAMPLE
                     ;
                     ;*****
                     ;
                     ;*****
                     ;       PROGRAM EQUATES
                     ;*****
 00BF                PDATA   EQU     0BFH    ; INTERPROCESSOR DATA PORT
 007F                PSTS    EQU     07FH    ; STATUS
                     ;*****
                     ;       BUFFER STATUS MASKS
                     ;*****
 0001                OBF     EQU     01H     ; OUTPUT BUFFER FULL
 0002                IBF     EQU     02H     ; INPUT BUFFER FULL
                     ;*****
                     ;       PROGRAM ORIGIN
                     ;*****
 3000                        ORG     03000H
                     ;*****
                     ;
                     ;       SLAVE READ ROUTINE
                     ;
                     ;               INPUTS:  NONE
                     ;               OUTPUTS: CHARACTER READ IN C-REGISTER
                     ;
                     ;       A,C REG MODIFIED
                     ;
                     ;*****
                     SLRD:
 3000 DB7F                   IN      PSTS    ; GET STATUS
 3002 E601                   ANI     OBF     ; SEE IF BUFFER FULL
 3004 C20030                 JNZ     SLRD    ; NO - LOOP UNTIL FULL
 3007 DBBF                   IN      PDATA   ; GET CHARACTER
 3009 4F                     MOV     C,A     ; PLACE IN C-REG
 300A C9                     RET             ; RETURN TO CALLER
```



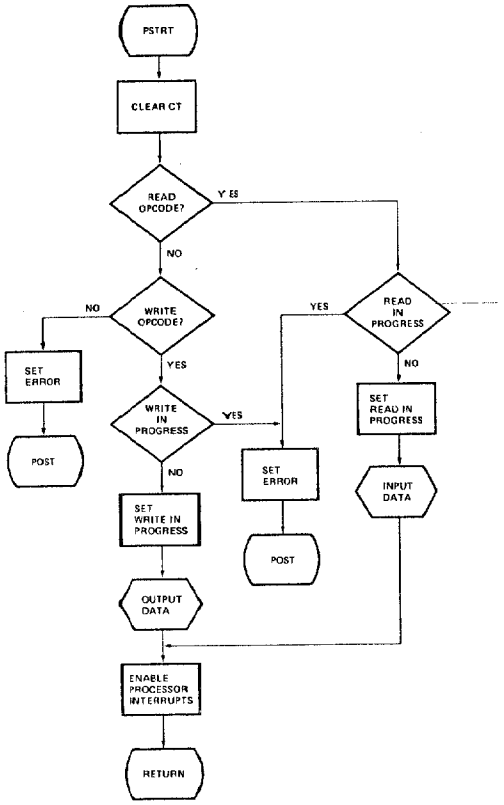
SLAVE WRITE ROUTINE

```
ISIS 8080 MACRO ASSEMBLER, V1.0                    PAGE 2
MODE TWO EXAMPLE - SLAVE SOFTWARE

                     ;*****
                     ;
                     ;       SLAVE WRITE ROUTINE
                     ;
                     ;               INPUTS:  CHARACTER TO WRITE IN C-REGISTER
                     ;               OUTPUTS: NONE
                     ;
                     ;       A REG MODIFIED
                     ;
                     ;*****
                     SLWT:
 300B DB7F                   IN      PSTS    ; GET STATUS
 300D E602                   ANI     IBF     ; SEE IF BUFFER FULL
 300F C20B30                 JNZ     SLWT    ; YES - LOOP UNTIL EMPTY
 3012 79                     MOV     A,C     ; GET DATA CHARACTER
 3013 D3BF                   OUT     PDATA   ; OUTPUT DATA
 3015 C9                     RET             ; RETURN TO CALLER
                     ;*****
                     ;       END OF SLAVE SOFTWARE DRIVER
                     ;*****
 0000                        END
```

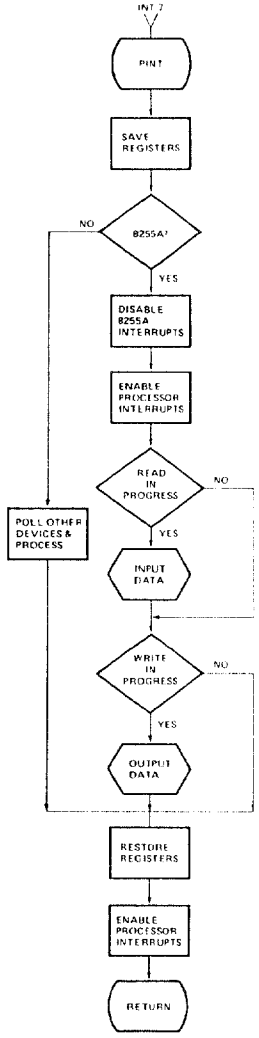


```
ISIS 8080 MACRO ASSEMBLER, V1.0                    PAGE 1
MODE TWO EXAMPLE - MASTER SOFTWARE

                             TITLE 'MODE TWO EXAMPLE - MASTER SOFTWARE'
                     ;*****
                     ;
                     ;       8080 MASTER TO 8080 SLAVE INTERFACE
                     ;               - MASTER SOFTWARE -
                     ;               MODE TWO EXAMPLE
                     ;
                     ;*****
                     ;
                     ;*****
                     ;       PROGRAM EQUATES
                     ;*****
 00E4                PORTA   EQU     0E4H    ; 8255 PORT A
                     PORTB   EQU     0E5H    ; 8255 PORT B
 00E6                PORTC   EQU     0E6H    ; 8255 PORT C
 00E7                CWR     EQU     0E7H    ; 8255 CONTROL WORD REGISTER
 0038                RST7    EQU     038H    ; RESTART 7 ADDRESS
                     ;
                     ;*****
                     ;
                     ;       INITIALIZATION CONTROL WORD
                     ;
                     ;               USED TO CONFIGURE THE 8255 AS FOLLOWS:
                     ;
                     ;                       PORT A - MODE 2 BIDIRECTIONAL BUS
                     ;                       PORT B - INPUT MODE 0 (NOT USED)
                     ;                       REMAINING PORT C LINES - INPUT MODE (NOT USED)
                     ;
                     ;*****
 00CB                ICW     EQU     11001011B       ; INITIALIZATION CONTROL WORD
                     ;*****
                     ;       8255 ENABLE/DISABLE INTERRUPT CONTROL WORDS
                     ;*****
 000D                IENI    EQU     00001101B       ; ENABLE INPUT INTERRUPTS
 0009                IENO    EQU     00001001B       ; ENABLE OUTPUT INTERRUPTS
 000C                IDNI    EQU     00001100B       ; DISABLE INPUT INTERRUPTS
 0008                IDNO    EQU     00001000B       ; DISABLE OUTPUT INTERRUPTS
                     ;*****
                     ;       STATUS EQUATES
                     ;*****
 0008                INTRA   EQU     08H     ; INTERRUPT REQ. BIT
 0080                OBFA    EQU     80H     ; OUTPUT BUFFER FULL
 0020                IBFA    EQU     20H     ; INPUT BUFFER FULL
```

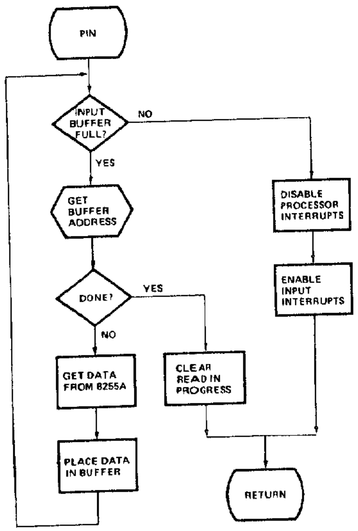
```
ISIS 8080 MACRO ASSEMBLER, V1.0                    PAGE 2
MODE TWO EXAMPLE - MASTER SOFTWARE

                     ;*****
                     ;       CONTROL BLOCK EQUATES
                     ;*****
 0000                CBST    EQU     00H             ; STATUS BYTE
                     CBOP    EQU     01H             ; OPCODE = 0 READ
                     ;                                        = 1 WRITE
 0002                CBUF    EQU     02H             ; BUFFER ADDRESS
 0004                CBCC    EQU     04H             ; CHARACTER COUNT
 0005                CBCT    EQU     05H             ; CHARACTER TRANSFERRED COUNT
 0006                CBCMP   EQU     06H             ; COMPLETION SERVICE ADDRESS
                     ;*****
                     ;       OPCODE EQUATES
                     ;*****
 0000                OPRD    EQU     00H     ; READ OPCODE
 0001                OPWT    EQU     01H     ; WRITE OPCODE
                     ;*****
                     ;       COMPLETION STATUS EQUATES
                     ;*****
 0000                STGD    EQU     00H             ; GOOD COMPLETION
 0001                STE1    EQU     01H             ; ERROR - COMMAND ALREADY IN PROGRESS
 0002                STE2    EQU     02H             ; ERROR - INVALID OPCODE
                     ;*****
                     ;       SET UP INTERRUPT VECTOR
                     ;*****
 0038                        ORG     RST7
 0038 C34630                 JMP     PINT            ; JUMP TO INTERRUPT SERVICE ROUTINE
                     ;*****
                     ;       PROGRAM ORIGIN
                     ;*****
 3000                        ORG     03000H
                     ;*****
                     ;
                     ;       INITIALIZATION ROUTINE
                     ;
                     ;       A REGISTER MODIFIED
                     ;
                     ;*****
                     INIT:
 3000 3ECB                   MVI     A,ICW   ; GET MODE CONTROL WORD
 3002 D3E7                   OUT     CWR     ; OUTPUT TO CONTROL WORD REGISTER
 3004 C9                     RET             ; RETURN TO CALLER
```



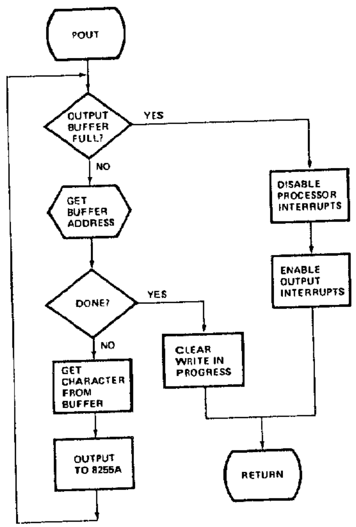

```
ISIS 8080 MACRO ASSEMBLER, V1.0                PAGE 3
COMMAND PROCESSOR

                        ;*****
                        ;
                        ;       COMMAND PROCESSOR
                        ;
                        ;               INPUTS: CONTROL BLOCK ADDRESS IN D AND E REGISTERS
                        ;
                        ;               OUTPUS: START I/O OR ERROR STATUS IN CONTROL BLOCK
                        ;       A,H,L   MODIFIED
                        ;
                        ;*****
                        PSTRT:
3005 210500                     LXI     H,CBCT  ; GET INDEX TO CT
3008 19                         DAD     D       ; COMPUTE ADDRESS OF CT
3009 3600                       MVI     M,OPR   ; CLEAR CT
300B 210100                     LXI     H,CBOC  ; GET INDEX TO OPCODE
300E 19                         DAD     D       ; COMPUTE ADDRESS
                                MOV     A,M     ; GET OPCODE
3010 FE00                       CPI     00H     ; SEE IF READ
3012 CA2430                     JZ      PSRD    ; YES - GO PROCESS READ
3015 FE01                       CPI     OPWT    ; SEE IF WRITE
3017 CA3530                     JZ      PSWT    ; YES - GO PROCESS WRITE
                        ;*****
                        ;       ERROR - INVALID OPCODE
                        ;*****
301A 3E02                       MVI     A,STE   ; GET ERROR STATUS CODE
                                JMP     POST    ; CALL COMPLETION ROUTINE
                        ;*****
                        ;       ERROR - TRANSICTION ALREADY IN PROGRESS
                        ;*****
                        PSTE:
301F 3ED1                       MVI     A,STE   ; GET ERROR STATUS CODE
3021 C3DC30                     JMP     POST    ; CALL COMPLETION ROUTINE
                        ;*****
                        ;       PROCESS READ COMMAND
                        ;*****
                        PSRD:
3024 3AEA30                     LDA     PRGRD+1 ; GET READ IN PROGRESS ADDRESS
3027 A7                         ANA     A       ; SEE IF READ IN PROGRESS (TEST FOR ZERO)
3028 C21F30                     JNZ     PSTE    ; IF YES - BRANCH
302B EB                         XCHG
302C 22E930                     SHLD    PRGRD   ; SAVE CONTROL BLOCK ADDRESS
302F EB                         XCHG
3030 CD7C30                     CALL    PIN     ; START I/O
3033 FB                         EI              ; ENABLE INTERRUPTS
3034 C9                         RET             ; RETURN TO CALLER
```



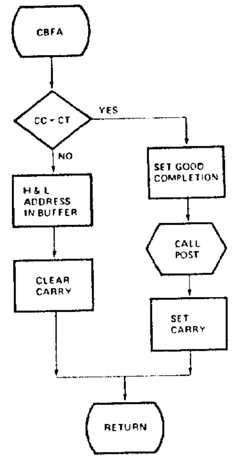

```
ISIS 8080 MACRO ASSEMBLER, V1.0                PAGE 4
COMMAND PROCESSOR

                        ;*****
                        ;
                        ;       PROCESS WRITE COMMAND
                        ;*****
                        PSWT:
3035 3AEC30                     LDA     PRGWT+1 ; GET WRITE IN PROGRESS ADDRESS
3038 A7                         ANA     A       ; SEE IF WRITE IN PROGRESS (TEST FOR ZERO)
3039 C21F30                     JNZ     PSTE    ; IF YES - BRANCH
303C EB                         XCHG
303D 22EB30                     SHLD    PRGWT   ; SAVE CONTROL BLOCK ADDRESS
3040 EB                         XCHG
3041 CD9C30                     CALL    PORT    ; START I/O
3044 FB                         EI              ; ENABLE INTERRUPTS
3045 C9                         RET             ; RETURN TO CALLER
```




```
ISIS 8080 MACRO ASSEMBLER, V1.0                PAGE 7
INPUT DATA ROUTINE

                ;*****
                ;       INPUT DATA ROUTINE
                ;*****
                PIN:
307C DBE6               IN      PORTC   ; GET STATUS OF DEVICE
307E E620               ANI     IBFA    ; SEE IF INPUT BUFFER FULL
3080 CA9630             JZ      PRTI    ; NO - BRANCH
3083 CDBC30             CALL    CBFA    ; GET ADDRESS IN BUFFER
3086 DABF30             JC      PIDON   ; IF DONE - BRANCH
3089 DBE4               IN      PORTA   ; GET DATA
308B 77                 MOV     M,A     ; PLACE IN BUFFER
308C C37C30             JMP     PIN     ; LOOP
                ;*****
                ;       END OF INPUT TRANSACTION
                ;*****
                PIDON:
308F AF                 XRA     A       ; CLEAR A
3090 32EA30             STA     PRGRD+1 ; CLEAR READ IN PROGRESS
3093 C39630             JMP     PRTI    ; RETURN
                ;*****
                ;       RETURN FROM INPUT
                ;*****
                PRTI:
3096 F3                 DI              ; DISABLE PROCESSOR INTERRUPTS
3097 3E0D               MVI     A,IEN1  ; GET ENABLE INPUT INTERRUPTS CONTROL WORD
3099 D3E7               OUT     CWR     ; OUTPUT TO CONTROL WORD REGISTER
309B C9                 RET             ; RETURN TO CALLER
```



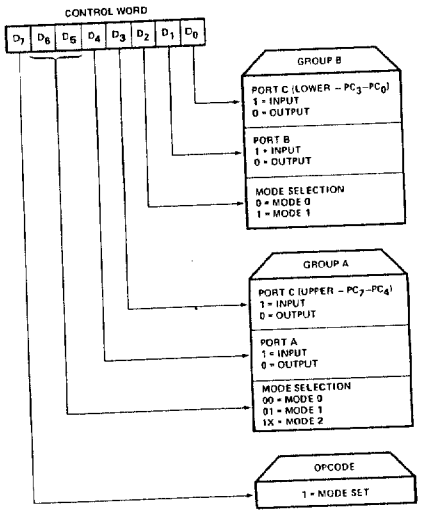

```
ISIS 8080 MACRO ASSEMBLER, V1.0                PAGE 8
OUTPUT DATA ROUTINE

                ;*****
                ;       OUTPUT DATA ROUTINE
                ;*****
                POUT:
309C DBE6               IN      PORTC   ; GET PORTC STATUS
309E E620               ANI     IBFA    ; SEE IF OUTPUT BUFFER FULL
                        JNZ     PRTO    ; YES - BRANCH
30A3 CDBC30             CALL    CBFA    ; SET UP ADDRESS OF DATA
30A6 DAAF30             JC      PODON   ; IF DONE - BRANCH
30A9 7E                 MOV     A,M     ; GET DATA FROM BUFFER
30AA D3E4               OUT     PORTA   ; OUTPUT DATA
30AC C39C30             JMP     POUT    ; LOOP
                ;*****
                ;       END OF OUTPUT TRANSACTION
                ;*****
                PODON:
30AF AF                 XRA     A       ; CLEAR A REG
30B0 32EC30             STA     PRGWT+1 ; CLEAR WRITE IN PROGRESS
30B3 C3B630             JMP     PRTO    ; RETURN
                ;*****
                ;       RETURN FROM OUTPUT
                ;*****
                PRTO:
30B6 F3                 DI              ; DISABLE PROCESSOR INTERRUPTS
30B7 3E09               MVI     A,IEN0  ; GET ENABLE OUTPUT INTERRUPTS CONTROL WORD
30B9 D3E7               OUT     CWR     ; OUTPUT TO CONTROL WORD REGISTER
30BB C9                 RET             ; RETURN TO CALLER
```

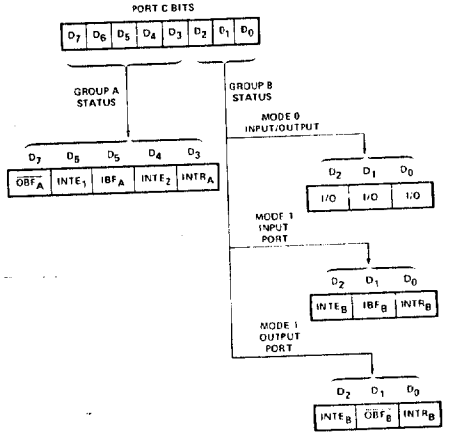
Setup Buffer Address Subroutine

```
ISIS 8080 MACRO ASSEMBLER, V1.0                PAGE 9
COMPUTE BUFFER ADDRESS ROUTINE

                ;*****
                ;       COMPUTE BUFFER ADDRESS ROUTINE
                ;*****
                CBFA:
30BC [illegible]        LXI     H,CBCT  ; GET INDEX TO CT
30BF 19                 DAD     D       ; COMPUTE ADDRESS OF CT
[illegible] 7E          MOV     A,M     ; GET CT
[illegible] 34          INR     M       ; INC CT
30C2 2B                 DCX     H       ; DEC TO CC
30C3 BE                 CMP     M       ; SEE IF EQUAL
[illegible]             JZ      PCOMP   ; IF EQUAL - DONE GO TELL USER
[illegible]             LXI     H,CBBUF ; GET INDEX TO BUFFER ADDRESS
[illegible]             DAD     D       ; COMPUTE ADDRESS OF BUFFER ADDRESS
[illegible]             PUSH    D       ; SAVE D AND E REGISTERS
[illegible]             MOV     E,M     ; GET LSB OF BUFFER ADDRESS
[illegible]             INX     H       ; INC TO NEXT BYTE
[illegible]             MOV     D,M     ; GET BUFFER MSB
[illegible] AC          XRA     H       ; CLEAR H REG
[illegible]             MOV     L,A     ; GET CT
[illegible]             DAD     D       ; COMPUTE CHARACTER ADDRESS
[illegible]             POP     D       ; RESTORE CONTROL BLOCK ADDRESS
[illegible] AF          ORA     A       ; CLEAR CARRY
[illegible]             RET             ; RETURN TO CALLER
```

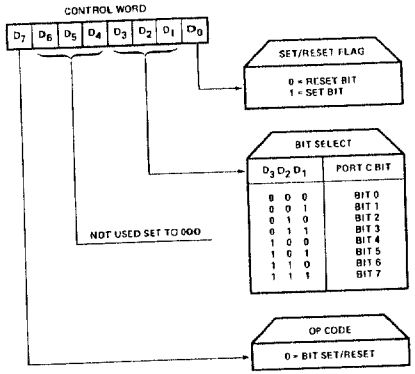
```
ISIS 8080 MACRO ASSEMBLER, V1.0                PAGE 10
POST TO USER COMPLETION ROUTINE

                ;*****
                ;       POST GOOD COMPLETION TO USER
                ;*****
                PCOMP:
[illegible]             MVI     A,SGOOD ; GET GOOD STATUS CODE
[illegible]             CALL    POST    ; CALL USER ROUTINE
[illegible]             STC             ; SET CARRY
[illegible]             RET             ; RETURN TO CALLER
                ;*****
                ;       POST TO USER COMPLETION ROUTINE
                ;
                ;               INPUTS - STATUS CODE IN A REG
                ;                        CONTROL BLOCK ADDRESS IN D AND E REG
                ;               OUTPUTS- PASSES CONTROL TO USER COMPLETION ADDRESS
                ;                        SPECIFIED IN CONTROL BLOCK
                ;*****
                POST:
30DC EB                 XCHG
30DD 77                 MOV     M,A     ; UPDATE STATUS
30DE EB                 XCHG
30DF 210600             LXI     H,CBCMP ; GET INDEX TO COMPLETION ADDRESS
30E2 19                 DAD     D       ; COMPUTE ADDRESS
30E3 4E                 MOV     C,M     ; GET LSB OF COMPLETION ADDRESS
30E4 23                 INX     H       ; INC TO NEXT BYTE
30E5 46                 MOV     B,M     ; GET MSB BYTE OF COMPLETION ADDRESS
30E6 C5                 PUSH    B       ; PUSH ADDRESS INTO STACK
30E7 C9                 RET             ; PASS CONTROL TO USER ROUTINE
30E8 C9                 RET             ; RETURN TO CALLER
                ;*****
                ;
                ;       DATA AND TABLES
                ;
                ;               IF DATA NON ZERO CONTROL BLOCK IN PROGRESS
                ;
                ;*****
30E9 0000       PRGRD:  DW      0       ; IN PROGRESS READ CONTROL BLOCK
30EB 0000       PRGWT:  DW      0       ; IN PROGRESS WRITE CONTROL BLOCK
                ;*****
                ;       END OF MASTER SOFTWARE DRIVER
                ;*****
0000                    END
```



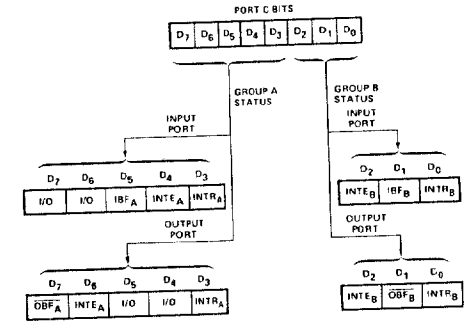
MODE CONTROL WORD



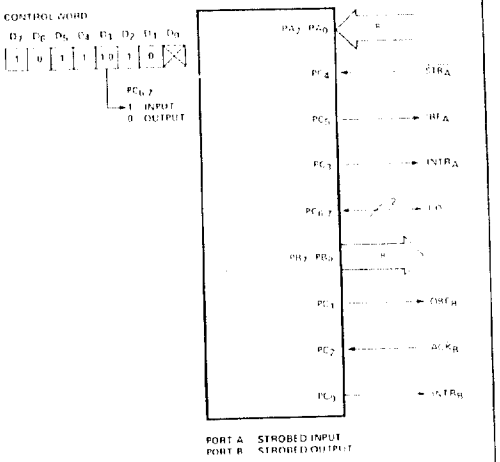
MODE 1 STATUS WORD

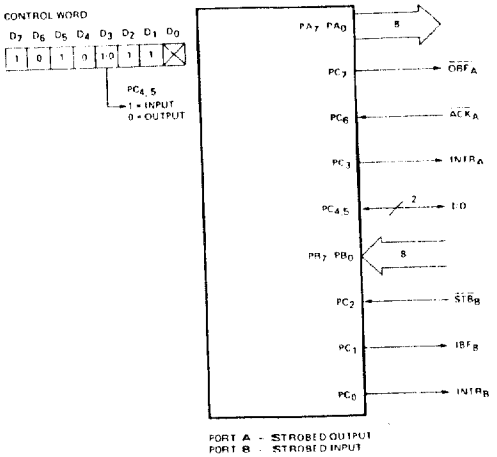


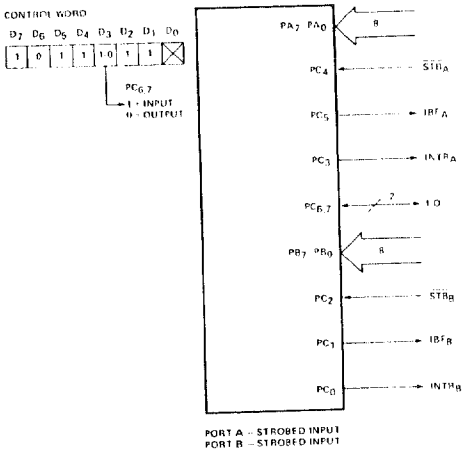
BIT SET/RESET CONTROL WORD

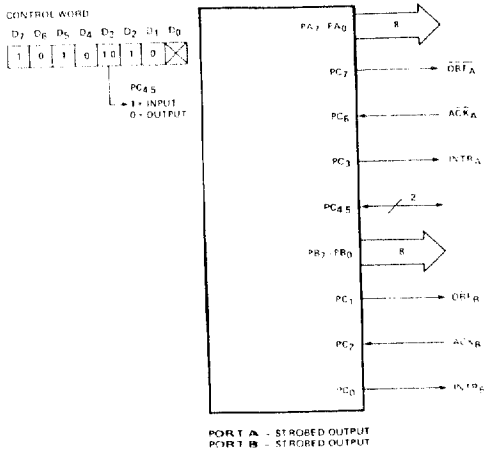


MODE 2 STATUS WORD

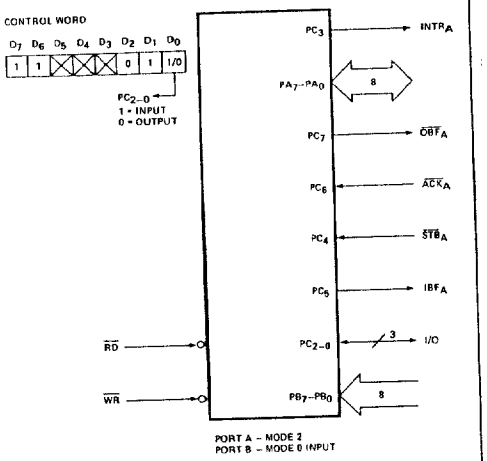


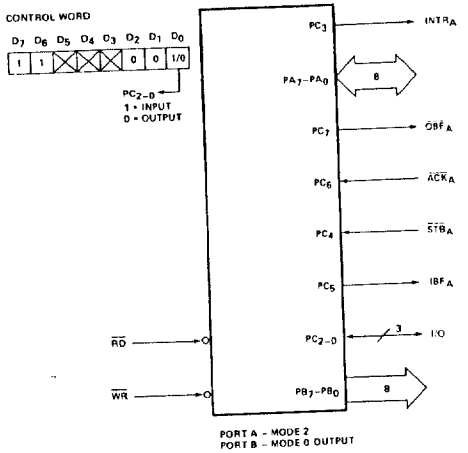


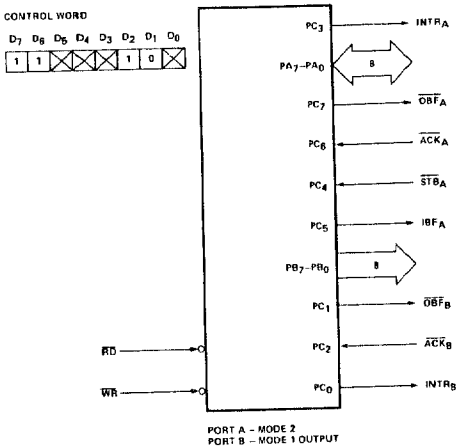


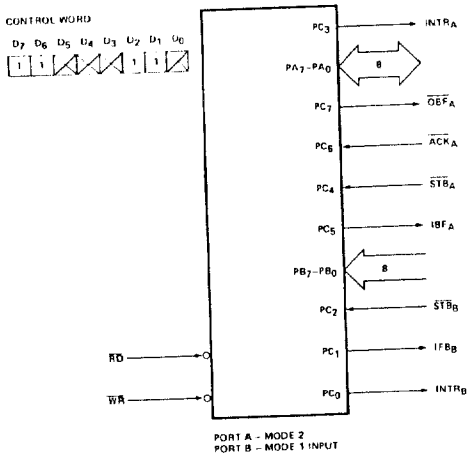


MODE 2 CONFIGURATIONS









Using The 8259 Programmable Interrupt Controller

by John Beaston

INTRODUCTION 2-94

CONCEPTS 2-94

- 8080 INTERRUPTS 2-95
- 8259—8080 OVERVIEW 2-96

8259 BLOCK DIAGRAM 2-97

- INPUT CIRCUIT 2-98
- PRIORITY CELL 2-99
- DATA BUS BUFFER 2-100
- READ/WRITE CONTROL LOGIC 2-100
- CASCADE BUFFER/COMPARATOR 2-100
- PIN DEFINITIONS 2-100

PROGRAMMING THE 8259 2-101

- INITIALIZATION COMMAND WORDS (ICSs) 2-101
- OPERATION COMMAND WORDS (OCWs) 2-103
  - Fully Nested Mode 2-103
  - Rotating Priority Commands 2-104
  - Interrupt Masks (OCW1) 2-106
  - Special Mask Mode (OCW3) 2-106
  - Polled Mode (OCW3) 2-107
  - Reading the 8259 Status (OCW3) 2-107

CASCADING THE 8259 2-108

APPLICATION EXAMPLES 2-109

- POWER FAIL/AUTO-START WITH BATTERY BACK-UP RAM 2-109
- 78 LEVEL INTERRUPT SYSTEM 2-114

CONCLUSION 2-118