**Contents**

INTRODUCTION ........................................... 1

OVERVIEW ................................................. 1

8080 CPU MODULE INTERFACE, ......................... 1

PERIPHERAL INTERFACE SECTION ................. 3

INTERNAL LOGIC SECTION ......................... 4

Mode Definition .............................. 4

Bit Set/Reset ..................................... 5

INTERRUPT CONTROL LOGIC STATUS
WORDS, .................................................. 6

SOFTWARE CONSIDERATIONS ................. 8

MODE 0 __ STATUS DRIVEN
PERIPHERAL INTERFACE ......................... 10

8255 To Peripheral Hardware Interface, 10

8080 CPU Module To 8255 Interface 10

Mode 0 Interface Software ........................ 12

Summary/Conclusions, ............................ 13

MODE 1 __ INTERRUPT DRIVEN
PRINTER INTERFACE ............................... 15

CPU Module To 8255 Interface, 15

8255 To Peripheral Interface, 15

Mode 1 Software Driver ......................... 16

Summary/Conclusions: ......................... 17

MODE 2 __ 8080 TO 8080 INTERFACE ........... 20

Hardware Discussion ......................... 20

Software Discussion .......................... 20

Summary/Conclusions ......................... 20

APPENDIX A __ 8255 QUICK
REFERENCE ........................................ 27
Related Intel Publications

"Intel 8080 Microcomputer Systems User's Manual"
"Memory Design Handbook"
"Using the 8251 Universal Synchronous/Asynchronous Receiver/Transmitter"
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 8255 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 8255, 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 8255 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 "8080 Microcomputer Systems User's Manual", particularly the 8255 device description.

OVERVIEW OF THE 8255

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

8080 CPU MODULE INTERFACE

The 8255 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 8255 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 designates 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 8255 and the 8080 CPU. I/O Ports A, B, C or the Control Word Register are selected by the two port select signals (A1, A0). These signals (A1 and A0) 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, 8228).

The 8255 I/O read (RD) and I/O write (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 8255 to 8080 CPU Module Interface is that for small system designs the 8255 may be interfaced directly to the standard MCS-80 component family with no external logic. Minimum external logic is required in large system designs.

---

**Figure 4. I/O Port Select Characters**

---

**Figure 3. Linear Select 8255 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 8255 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 8255 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.

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 strobe 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 8255 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 8255.

Table 1. Features of Peripheral Interface Lines

<table>
<thead>
<tr>
<th>Mode 0 – Basic Input/Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two 8-bit ports</td>
</tr>
<tr>
<td>Two 4-bit ports with set/reset capability</td>
</tr>
<tr>
<td>Outputs are latched</td>
</tr>
<tr>
<td>Inputs are not latched</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mode 1 – Strobed Input/Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>One or two strobed ports</td>
</tr>
<tr>
<td>Each Mode 1 port contains:</td>
</tr>
<tr>
<td>8-bit data port</td>
</tr>
<tr>
<td>3 control lines</td>
</tr>
<tr>
<td>Interrupt support logic</td>
</tr>
<tr>
<td>Any port may be input or output</td>
</tr>
<tr>
<td>If one Mode 1 port is used, the remaining 13 lines may be configured in Mode 0.</td>
</tr>
<tr>
<td>If two Mode 1 ports are used, the remaining 2 bits may be input or output with bit set/reset capability.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mode 2 – Strobed Bidirectional Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>One bidirectional bus which contains:</td>
</tr>
<tr>
<td>8-bit bidirectional bus supported by Port A</td>
</tr>
<tr>
<td>5 control lines</td>
</tr>
<tr>
<td>Interrupt support logic</td>
</tr>
<tr>
<td>Inputs and outputs are latched</td>
</tr>
<tr>
<td>The remaining 11 lines may be configured in either Mode 0 or Mode 1.</td>
</tr>
</tbody>
</table>

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 function and 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 8255 as a mode definition control word. The mode definition control word (shown in Figure 6) is used to specify the configuration of the 24 8255 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.

Example #1: This example demonstrates how a mode control word is constructed and issued to an 8255. 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 8255 interface shown in Figure 3.

If an 8255 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 PC7–PC4 Output
- Port C Bit 3 Input

The following mode control word is used:

```
Port A Mode 0
Port B Mode 1
Port C Bits PC7–PC4 Output
Port C Bit 3 Input

The assembly language program is:

CWR EQU OFBH ;8255 @1 CONTROL WORD REGISTER
***** ISSUE MODE CONTROL WORD
*****
MVI A, (40) ;GET MODE CONTROL WORD
OUT CWR ;OUTPUT TO 8255 #1 CONTROL WORD REGISTER
```

Figure 6. Mode Definition Control Word
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 8255 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.

![Control word diagram](image)

The assembly language program is:

```assembly
CWR EOU OFBH 8255 =1 CONTROL WORD REGISTER

... SET BIT 3
MVI A,00000111B GET SET BIT 3 CONTROL WORD
OUT CWR OUTPUT TO 8255 =1 CONTROL WORD REGISTER

... RESET BIT 3
MVI A,00000110B GET RESET BIT 3 CONTROL WORD
OUT CWR OUTPUT TO 8255 =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 "DCK A" Instruction could be used to generate the correct control word and save one byte of code.

00000111 \( \times 1 = 00000110 \) (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 EOU DFH 8255 =1 PORT A

... SET BIT 0
IN PORTA OEH :GET STATE OF PORT

... RESET BIT 0
IN ANI OEH PORTA \( \times \) GET STATE OF PORT
OUT PORTA \( \times \) RESET BIT 0
```

Example #2: This example demonstrates how a Port C bit set/reset control word is constructed and issued to an 8255. 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 8255 interface shown in Figure 3.
INTERRUPT CONTROL LOGIC STATUS WORDS

As previously mentioned, the 8255 Mode 1 and Mode 2 configurations support interrupt control logic. If a read of Port C is issued when the 8255 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 8255. This signal is not available externally. The structure of the Mode 1 status word varies as a function of the mode of the 8255. Example #4 shows the status word which results from reading Port C from an 8255 which is configured with Port A Mode 1 input and Port B Mode 1 output.

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

- D1 = Mode 1 Input
- D0 = Mode 1 Output
- D7 = Mode 0 Bit
- D6 = Mode 1 Bit
- D5 = Mode 2 Bit
- D4 = Mode 0 Bit
- D3 = Mode 1 Bit
- D2 = Mode 2 Bit
- D1 = Mode 0 Bit
- D0 = Mode 1 Bit

Example #4 – MODE 1 STATUS WORD

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

<table>
<thead>
<tr>
<th>Port</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1 Input</td>
</tr>
<tr>
<td>B</td>
<td>1 Output</td>
</tr>
<tr>
<td>C</td>
<td>Bits 6 &amp; 7 Output</td>
</tr>
</tbody>
</table>

The following mode control word is used:

- Mode Control Word = 10110100 Binary

NOTE: The Port C I/O bits D7 and D6 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 IBFA, INTRA, OBF_B, and INTRB 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 D2-D0 are defined by the specification of the Port B configuration. Example #5 shows the structure of the Mode 2 status word when the 8255 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 8255.
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.

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

![Mode 2 Status Word Diagram]

**Example #6 - MODE 2 INTERRUPT ENABLE/DISABLE**

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

- **INTE₁** - Bit 6 - Enable output interrupts
- **INTE₂** - 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.

```assembly
CWR EQU OFBH 8255 = CONTROL WORD REGISTER
   
   ; Enable interrupts for Mode 2 output
   ; set Port C bit 6
   MOV A, 00001101B
   OUT CWR
   ; output to 8255 = 1, control word register

   ; Disable interrupts for Mode 2 output
   ; reset Port C bit 6
   MOV A, 00001100B
   OUT CWR
   ; output to 8255 = 0, 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 8255 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 8255 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 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.

![Figure 10. Sample Status Driven Software Flowchart](image)

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>This 1-byte field is used to transmit the status of the I/O transaction (busy, complete, etc.).</td>
</tr>
<tr>
<td>Opcode</td>
<td>This 1-byte field defines the type of I/O (READ, WRITE, etc.).</td>
</tr>
<tr>
<td>Buffer Address</td>
<td>This 2-byte field specifies the source/destination of the data block.</td>
</tr>
<tr>
<td>Character Count</td>
<td>This 1-byte field is a count of the number of characters involved in the transaction.</td>
</tr>
<tr>
<td>Character Transferred Count</td>
<td>This 1-byte count of the number of characters which were actually transferred.</td>
</tr>
<tr>
<td>Completion Address</td>
<td>This 2-byte field is the address of the user supplied completion routine which will be called after the I/O has been performed.</td>
</tr>
</tbody>
</table>
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 8255.
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.
**MODE 0 – STATUS DRIVEN PERIPHERAL INTERFACE**

This design example shows how a single 8255 in Mode 3 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.

8255 To Peripheral Hardware Interface

The first step in the design is to examine the specifications 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 8255 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 8255 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 (PC3–PC0) configured in input mode are used to input the device busy indications. Three of the Port C upper bits (PC5–PC4) configured in output mode are used to support the device strobe signals required by each device.

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 8255 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 8255 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 8255 is addressed through the use of an isolated I/O architecture utilizing a linear select scheme. Address bits A1 and A0 are used to select the 8255 port. Address bit A3 is the exclusive enable for 8255 #1. Examination of the schematic shows that all of the 8255 interface lines are directly driven by the CPU module.

<table>
<thead>
<tr>
<th>CHARACTER PRINTER</th>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA – DATA?</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>DATA STROBE</td>
<td>A 0.5 μsec pulse (minimum) used to transfer data from the 8255 to the printer.</td>
<td></td>
</tr>
<tr>
<td>BUSY</td>
<td>The level indicating that the printer cannot receive data.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PAPER TAPE PUNCH</th>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRACKS 1–8 DATA INPUT</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>PUNCH COMMAND INPUT</td>
<td>A true condition moves the tape and initiates punching the tape. This signal is actually a data strobe.</td>
<td></td>
</tr>
<tr>
<td>PUNCH READY OUTPUT</td>
<td>True signal indicates that the punch is ready to accept a punch command. This is the punch busy line.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PAPER TAPE READER</th>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA TRACK OUTPUTS</td>
<td>True signal indicates data track hole. These eight lines are the data lines from the punch.</td>
<td></td>
</tr>
<tr>
<td>DRIVE RIGHT</td>
<td>True signal drives the tape to the right and reads a character. This signal is actually the data strobe (initiate read signal).</td>
<td></td>
</tr>
<tr>
<td>DATA READY OUTPUT</td>
<td>True signal indicates data track outputs are in &quot;On character&quot; condition. This signal is the reader busy line.</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 13. Interface Block Diagram**
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.

<table>
<thead>
<tr>
<th>Port Selected</th>
<th>Port Select Character (In Hexadecimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port A 8255 =1</td>
<td>F6</td>
</tr>
<tr>
<td>Port B 8255 =1</td>
<td>F6</td>
</tr>
<tr>
<td>Port C 8255 =1</td>
<td>F7</td>
</tr>
<tr>
<td>Control Word Register 8255 =1</td>
<td>C0</td>
</tr>
<tr>
<td>Port A 8255 =2</td>
<td>E0</td>
</tr>
<tr>
<td>Port B 8255 =2</td>
<td>E0</td>
</tr>
<tr>
<td>Port C 8255 =2</td>
<td>EE</td>
</tr>
<tr>
<td>Control Word Register 8255 =2</td>
<td>E0</td>
</tr>
</tbody>
</table>

**Figure 15. I/O Port Addresses**

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.

![Mode Control Word](image)

**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.

Summary/Conclusions

This design example discussed the basic hardware and software required to handle a simple device interface. The 8255 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 8255 in Mode 0 should become evident.
PAPER TAPE PUNCH DRIVER

INPUTS: DATA TO PUNCH IN C-REGISTER
OUTPUTS: DATA TO PUNCH
A REGISTER MODIFIED

START:
MOVF DBR6
IN DBR6
SET STATUS OF PUNCH

MOVF WR
AND WR
SET IF BUSY

JMP PUSHPJ
JMP IF BUSY - JUMP TO PUSHPJ (WAIT LOOP)

PUNCH IS IDLE - OUTPUT 1 CHARACTER

MOV A,C
OUT PORTB - OUTPUT DATA TO DATA LINES

MOV WR
OUT PORTD - OUTPUT DATA TO CONTROL WORD

OUT PORTD - OUTPUT DATA TO CONTROL WORD

OUT PORTD - OUTPUT DATA TO DATA LINES

OUT PORTB
RET - RETURN TO CALLER

PUNCH DRIVER

PAPER TAPE READER DRIVER

INPUTS: DATA FROM READER
OUTPUTS: CHARACTER TO USER IN C-REGISTER
A AND C REGISTER MODIFIED

START:
MOVF WR
OUT PORTD - OUTPUT DATA TO DATA LINES

IN DBR6
SET STATUS OF DEVICE

AND WR
SET IF BUSY

JMP PUSHPJ
JMP IF BUSY - JUMP TO PUSHPJ (WAIT LOOP)

HEADER NOT BUSY - GET CHAR AND CLEAR STROBE

IN PORTBR - GET CHARACTER

MOV A,A
GET CHARACTE

MOV A,WR
GET DATA FROM DEVICE

OUT PORTB
RET - RETURN TO CALLER

LOGEO

READER DRIVER

PAGE 5

MODE ZERO EXAMPLE - PAPER TAPE READER DRIVER

PAGE 4

MODE ZERO EXAMPLE - PAPER TAPE PUNCH 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 throughput is increased. This example demonstrates how an 8255 configured in Mode 1 may be used to develop an interrupt driven interface for the Centronics 306 character printer.

**CPU Module To 8255 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 INTA output of the 8228 to the +12-volt supply through a 1K ohm series resistor.

The Mode 1 interrupt support logic of the 8255 provides an interrupt request line for each port. The 8255 interrupt request line (INTRA) 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.

**8255 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 8255 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:

- **PA7–PA0** – Output Data  
  Used to support the printer data port.

- **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.

- **ACK** – ACKnowledge  
  This line is used to signal the 8255 that the device has accepted the data. This line is supported by the printer ACKNLG signal.

---

![Figure 18. Interface Block Diagram](image-url)
Mode I 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 8255 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 8255 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 hexadeciaml in program memory. The interrupt service routine saves the processor state and polls the 8255 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 8255 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](image)

Table IV. Printer Software Control Block

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

**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 8255 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 choose 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.
PROGRAM OUTPUT DATA ROUTINE

CONTROL BLOCK ADDRESS IN D AND E REG

INPUTS:
- STATUS CODE IN A REG
- CONTROL BLOCK ADDRESS IN D AND E REG
- OUTPUTS:
  - PASSED CONTROL TO USER COMPLETION ROUTINE
  - SPECIFIED IN CONTROL BLOCK
  - WITH CONTROL BLOCK ADDR IN D AND E

A,R,L,E,C MSG MODIFIED

POST-PROCEDURE

DATA AND TABLES

END OF USER COMPLETION ROUTINE

GO TO PROGRESS CONTROL BLOCK ADDRESS
IF DATA = 0 VIII CONTROL BLOCK IN PROGRESS
IF DATA NOT EQUAL TO 00H CONTROL BLOCK IN PROGRESS

END OF USER INTERFACE

0000 0EE
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 8255 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 8255 which is configured in Mode 2. The 8255 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 ACK and STB signals required to READ from and WRITE to the 8255 bidirectional bus with a single I/O instruction.

The system shown in Figure 20 utilizes SSI logic to read the 8255 IBF and OBF signals. If two spare 8255 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 8255 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 8255 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
**TITLE** "MORE TWO EXAMPLES - SLAVE SOFTWARE"

---

**SLRD**

**GET STATUS**

**NO**

**MASTER INPUT BUFFER FULL?**

**YES**

**INPUT CHARACTER**

**RETURN**

**SLAVE READ ROUTINE**

---

```
0000 D000 ORG D3000

; SLAVE READ ROUTINE
; INPUTS: NONE
; OUTPUTS: CHARACTER READ IN C-REGISTER
; A, C REG MODIFIED

SLRD:
0000 D07F IN D07S ; GET STATUS
0002 D001 ANI D0F ; SEE IF BUFFER FULL
0004 C0030 JNB D13D ; NO - LOOP UNTIL FULL
0007 D0BF IN D0F4 ; GET CHARACTER
0009 D9 MOV C,A ; PLACE IN C-REG
000A C9 RET ; RETURN TO CALLER
```

---

**SLWT**

**GET STATUS**

**YES**

**MASTER INPUT BUFFER FULL?**

**NO**

**OUTPUT CHARACTER**

**RETURN**

**SLAVE WRITE ROUTINE**

---

```
0000 D07F IN D07S ; GET STATUS
0002 D005 ANI D0F ; SEE IF BUFFER FULL
0004 2C0010 JNB D13D ; NO - LOOP UNTIL EMPTY
0007 D103 OUT D103 ; OUTPUT DATA CHARACTER
0009 D150 MOV A,C ; GET DATA CHARACTER
000A C9 RET ; RETURN TO CALLER

; END OF SLAVE SOFTWARE DRIVER
```

---

21
TITLE "MODE TWO EXAMPLE - MASTER SOFTWARE"

PROGRAM ENURES

0045 PORT_A EQU OEH ; 635 PORT A
0046 PORT_B EQU OEH ; 635 PORT B
0047 CMW EQU OEH ; 87H CONTROL WORD REGISTER
0048 NRTT EQU OEH ; RSTARY 7 ADRESSED

; INITIALIZATION CONTROL WORD
; TO CONFIGURE THE 675X AS FOLLOW:
; PORT A - MODE 2 BIDIRECTIONAL BUS
; PORT B - INPUT MODE 0 (NO I/Os)
; REMAINING PORT C LINES - INPUT MODE (NOT USED)

004B FM EQU 11001011B ; INITIALIZATION CONTROL WORD

; Enablers/Disablers INTERRUPT CONTROL WORDS

005D ENX EQU 00001010B ; ENABLE INTERRUPTS
0059 ENO EQU 00001010B ; ENABLE OUTPUT INTERRUPTS
005C LIM EQU 00001000B ; DISABLE INPUT INTERRUPTS
0058 LM EQU 00001000B ; DISABLE OUTPUT INTERRUPTS

; STATUS REGISTERS

0068 INTRA EQU 04H ; INTERRUPT REQUEST
0060 INTR EQU 02H ; INTERRUPT BUFFER FULL

CONTROL BLOCK ENURES

0060 CMW EQU OEH ; STATUS BITS
0064 CMW EQU OEH ; WRITE ADDRESS
0065 CMW EQU OEH ; READ ADDRESS
0064 CMW EQU OEH ; CHARACTER COUNT
0065 CMW EQU OEH ; CHARACTER-TRANSMITTED COUNT
0066 CMW EQU OEH ; COMPLETION SERVICE ADDRESS

; Opcode ENURES

0060 CMW EQU OEH ; WRITE OPCODE
0061 CMW EQU OEH ; READ OPCODE

; COMPLETION STATUS ENURES

006D CMW EQU OEH ; GOOD COMPLETION
0071 CMW EQU OEH ; ERRORS - COMMAND ALREADY IN PROGRESS
0072 CMW EQU OEH ; INVALID CODE

; Setup Interrupt Vector

0083 CMW EQU OEH ; JMP FIRST ; JUMP TO INTERRUPT SERVICE ROUTINE

PROGRAM ORIGIN

0000 CMW EQU OEH ; INITIALIZATION ROUTINE

; A REGISTER MODIFIED

3000 CMW EQU OEH ; GET MODE CONTROL WORD
3001 CMW EQU OEH ; OUTPUT TO CONTROL WORD REGISTER
3004 CMW RET ; RETURN TO CALLER
ILS IBM MACRO ASSEMBLER, V1.0

**COMMAND PROCESSOR**

```
; process read command
1024 times 00h, 01h, 02h, 03h
1025 times 04h, 05h, 06h, 07h
1026 times 08h, 09h, 0ah, 0bh
1027 times 0ch, 0dh, 0eh, 0fh
1028 times 10h, 11h, 12h, 13h
1029 times 14h, 15h, 16h, 17h
102a times 18h, 19h, 1ah, 1bh
102b times 1ch, 1dh, 1eh, 1fh
102c times 20h, 21h, 22h, 23h
102d times 24h, 25h, 26h, 27h
102e times 28h, 29h, 2ah, 2bh
102f times 2ch, 2dh, 2eh, 2fh
1030 times 30h, 31h, 32h, 33h
1031 times 34h, 35h, 36h, 37h
1032 times 38h, 39h, 3ah, 3bh
1033 times 3ch, 3dh, 3eh, 3fh
1034 times 40h, 41h, 42h, 43h
1035 times 44h, 45h, 46h, 47h
1036 times 48h, 49h, 4ah, 4bh
1037 times 4ch, 4dh, 4eh, 4fh
1038 times 50h, 51h, 52h, 53h
1039 times 54h, 55h, 56h, 57h
103a times 58h, 59h, 5ah, 5bh
103b times 5ch, 5dh, 5eh, 5fh
103c times 60h, 61h, 62h, 63h
103d times 64h, 65h, 66h, 67h
103e times 68h, 69h, 6ah, 6bh
103f times 6ch, 6dh, 6eh, 6fh
1040 times 70h, 71h, 72h, 73h
1041 times 74h, 75h, 76h, 77h
1042 times 78h, 79h, 7ah, 7bh
1043 times 7ch, 7dh, 7eh, 7fh
1044 times 80h, 81h, 82h, 83h
1045 times 84h, 85h, 86h, 87h
1046 times 88h, 89h, 8ah, 8bh
1047 times 8ch, 8dh, 8eh, 8fh
1048 times 90h, 91h, 92h, 93h
1049 times 94h, 95h, 96h, 97h
104a times 98h, 99h, 9ah, 9bh
104b times 9ch, 9dh, 9eh, 9fh
104c times a0h, a1h, a2h, a3h
104d times a4h, a5h, a6h, a7h
104e times a8h, a9h, aaa, abh
104f times acb, adh, aeh, afh
1050 times b0h, b1h, b2h, b3h
1051 times b4h, b5h, b6h, b7h
1052 times b8h, b9h, bah, bbh
1053 times bch, bdh, beh, bhf
1054 times c0h, c1h, c2h, c3h
1055 times c4h, c5h, c6h, c7h
1056 times c8h, c9h, caa, cab
1057 times ccb, cdb, ceb, cff
1058 times d0h, d1h, d2h, d3h
1059 times d4h, d5h, d6h, d7h
105a times d8h, d9h, daa, dbb
105b times dcb, ddb, deb, dff
105c times e0h, e1h, e2h, e3h
105d times e4h, e5h, e6h, e7h
105e times e8h, e9h, eag, ebh
105f times ecf, edb, ede, efh
1060 times f0h, f1h, f2h, f3h
1061 times f4h, f5h, f6h, f7h
1062 times f8h, f9h, faa, fbb
1063 times fcb, fdb, fde, fff
```
INPUT DATA ROUTINE

*** INPUT DATA ROUTINE

FIN:  

SETC 706: IN F07C: GET STATUS OF DEVICE
JMP 80D: IN 9F1: READ INPUT BUFFER FULL
JMP 8AD: JNC. PJT: NO = BRANCH
JMP 8A9: CALL. CMPA: GET ADDRESS IN BUFFER
JMP 84B: CALL 9F: IF HOME = BRANCH
JMP 810: IN PORTA: GET DATA
JMP 871: MOV A, A: PLACE IN BUFFER
JMP 80C: CJNE 0F, PCH: LOOP

*** END OF INPUT TRANSACTION

FIN:  

JMP 80C: CJP 25: RETURN

*** RETURN FROM INPUT

PJ:  

DI 09H: F3: DISABLE PROCESSOR INTERRUPTS
MO3 707: MOV B, ENC: GET ENABLE INPUT INTERRUPTS, CONTROL WORD
JMP 890: PCH: RETURN TO CALLED

CLEAR HEAD IN PROGRESS

GET DATA FROM 8500

PLACE DATA IN BUFFER

RETURN

OUTPUT DATA ROUTINE

*** OUTPUT DATA ROUTINE

FIN:  

CLR 706: IN 705C: GET PORT STATUS
CLR 82D: ANL 8BP: SET OF OUTPUT BUFFER FULL
JMP 849: CALL. CMPA: GET PORT ADDRESS
JMP 804: CALL 9F: IF DONE = BRANCH
JMP 810: MOV A, M: GET DATA FROM BUFFER
JMP 849: JMP. PGT: OUTPUT DATA
JMP 804: CJNE 0F, PCH: LOOP

*** END OF OUTPUT TRANSACTION

FIN:  

JMP 804: CJP 25: RETURN

*** RETURN FROM OUTPUT

PJ:  

DI 09H: F3: DISABLE PROCESSOR INTERRUPTS
MO3 707: MOV B, ENC: GET ENABLE OUTPUT INTERRUPTS, CONTROL WORD
JMP 890: PCH: RETURN TO CALLED

CLEAR WRITE IN PROGRESS

GET CHARACTER FROM BUFFER

OUTPUT TO 8255

RETURN