INPUT/OUTPUT INTERFACE CIRCUITS AND LSI PERIPHERAL DEVICE

10.1 Core and Special-Purpose I/O Interfaces
10.2 Byte-Wide Output Ports Using Isolated I/O
10.3 Byte-Wide Input Ports Using Isolated I/O
10.4 Input/Output Handshaking and a Parallel Printer Interface
10.5 82C55A Programmable Peripheral Interface
10.6 82C55A Implementation of Parallel Input/Output Ports
INPUT/OUTPUT INTERFACE CIRCUITS AND LSI PERIPHERAL DEVICE

10.7 Memory-Mapped Input/Output Ports
10.8 82C54 Programmable Interval Timer
10.9 82C37A Programmable Direct Memory Access Controller
10.10 Serial Communication Interface
10.11 Programmable Communication Interface Controller
10.12 Keyboard and Display Interface
10.13 8279 Programmable Keyboard/Display Controller

10.1 Core and Special-Purpose I/O Interfaces

- Special-Purpose I/O interfaces
  - Keyboard interface
  - Display interface
  - Parallel printer interface
  - Serial communication interface
  - Local area network interface

- Core I/O interfaces
  - Parallel input/output ports
  - Interval timers
  - Direct memory access control
10.2 Byte-Wide Output Ports Using Isolated I/O

Sixty-four-line parallel output circuit for an 8088-based microcomputer

<table>
<thead>
<tr>
<th>I/O Port</th>
<th>I/O Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 0</td>
<td>1XXXXXXXXXXX0000₂</td>
</tr>
<tr>
<td>Port 1</td>
<td>1XXXXXXXXXXX0010₂</td>
</tr>
<tr>
<td>Port 2</td>
<td>1XXXXXXXXXXX0100₂</td>
</tr>
<tr>
<td>Port 3</td>
<td>1XXXXXXXXXXX0110₂</td>
</tr>
<tr>
<td>Port 4</td>
<td>1XXXXXXXXXXX1000₂</td>
</tr>
<tr>
<td>Port 5</td>
<td>1XXXXXXXXXXX1010₂</td>
</tr>
<tr>
<td>Port 6</td>
<td>1XXXXXXXXXXX1100₂</td>
</tr>
<tr>
<td>Port 7</td>
<td>1XXXXXXXXXXX1110₂</td>
</tr>
</tbody>
</table>

I/O Address decoding for ports 0 through 7
**10.2 Byte-Wide Output Ports Using Isolated I/O**

Sixty-four-line parallel output circuit for an 8086-based microcomputer

---

**EXAMPLE**

To which output port in the 8088-based microcomputer are data written when the address put on the bus during an output cycle is $8002_{16}$?

Solution:

Express the address in binary form, we get

$$A_{15}\ldots A_0 = A_{15L}\ldots A_{0L} = 100000000000010_2$$

That is $A_{15L} = 1$, $A_{0L} = 0$ and $A_{3L}A_2A_1L = 001$

Moreover, whenever an output bus cycle is in progress, IO/M is logic 1. Therefore the enable inputs of the 74F138 decoder are

- $G_{2B} = A_{0L} = 0$
- $G_{2A} = IO/M = 0$
- $G_1 = A_{15L} = 1$
10.2 Byte-Wide Output Ports Using Isolated I/O

These inputs enable the decoder for operation. At the same time, its select inputs are supplied with the code 001. This input causes output $P_1$ to switch to logic 0:

$$P_1 = 0$$

The gate at the CLK input of port 1 has as its inputs $P_1$ and WR. When valid output data are on the bus, WR switches to logic 0. Since $P_1$ is also 0, the CLK input of the 74F374 for port 1 switches to logic 0. At the end of the WR pulse, the clock switches from 0 to 1, a positive transition. This causes the data on $D_0$ through $D_7$ to be latched and become available at output lines $O_8$ through $O_{15}$ of port 1.

### EXAMPLE

Write a series of instructions that will output the byte contents of the memory address DATA to output port 0 in the circuit shown in the previous figure of the 8088-base microcomputer.

**Solution:**

To write a byte to output port 0, the address is

$$A_{15}A_{14} \ldots A_0 = 1XXXXXXXXXXX0000_2$$

Assuming that the don’t-care bits are all made logic 0, we get

$$A_{15}A_{14} \ldots A_0 = 1000000000000000_2 = 8000_{16}$$

The instruction sequence is

- MOV DX, 8000H
- MOV AL, [DATA]
- OUT DX, AL
10.2 Byte-Wide Output Ports Using Isolated I/O

- Time-delay loop and blinking an LED at an output port

```
MOV DX, 8000H ; Initialize address of port 0
MOV AL, 00H      ; Load data with bit 7 as logic 0
ON_OFF OUT DX, AL        ; Output the data to port 0
MOV CX, 0FFFFH ; Load delay count of FFFFH
HERE: LOOP HERE ; Time delay loop
XOR AL, 80H      ; Complement bit 7 of AL
JMP ON_OFF    ; Repeat to output the new bit 7
```

Driving an LED connected to an output port
10.3 Byte-Wide Input Ports Using Isolated I/O

EXAMPLE

What is the I/O address of port 7 in the circuit of the previous figure? Assume all unused address bits are at logic 0

Solution:

For the I/O address decoder to be enable, address bits $A_{15}$ and $A_0$ must be

$$A_{15} = 1 \quad \text{and} \quad A_0 = 0$$

To select port 7, the address applied to the CBA inputs of the decoder must be

$$A_{2L}A_{2L}A_{1L} = 111$$

Using 0s for the unused bits gives the address

$$A_{15} \ldots A_{1L}A_{0L} = 1000000000001110_2 = 800E_{16}$$
10.3 Byte-Wide Input Ports Using Isolated I/O

**EXAMPLE**

For the circuit in the previous figure, write an instruction sequence that inputs the byte contents of input port 7 to the memory location \( \text{DATA}_7 \).

**Solution:**

In the previous example we found that the address of port 7 is \( 800E_{16} \). This address is loaded into the DX register by:

\[
\text{MOV DX, 800EH}
\]

Now the contents of this port are input to the AL register by:

\[
\text{IN AL, DX}
\]

Finally, the byte of data is copied to memory location \( \text{DATA}_7 \) by:

\[
\text{MOV DATA}_7, \text{AL}
\]
10.3 Byte-Wide Input Ports Using Isolated I/O

- Polling the setting of a switch

  \[
  \text{MOV} \; \text{DX}, \; 8000H \\
  \text{POLL}\_I7: \; \text{IN} \; \text{AL}, \; \text{DX} \\
  \text{SHL} \; \text{AL}, \; 1 \\
  \text{JC} \; \text{POLL}\_I7
  \]

CONTINUE:

10.4 Input/Output Handshaking and a Parallel Printer Interface

- I/O synchronization is achieved by implementing what is known as handshaking as part of the input/output interface.
- Three general types of signals at the printer interface:
  - Data
  - Control
  - Status

Parallel printer interface

Parallel Printer Port

Data

Control

Status

Printer

Status

False

True

Send data

Send control
10.4 Input/Output Handshaking and a Parallel Printer Interface

<table>
<thead>
<tr>
<th>Pin</th>
<th>Assignment</th>
<th>Pin</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Strobe</td>
<td>14</td>
<td>Auto Fox</td>
</tr>
<tr>
<td>2</td>
<td>Data 0</td>
<td>15</td>
<td>Error</td>
</tr>
<tr>
<td>3</td>
<td>Data 1</td>
<td>16</td>
<td>Initialize</td>
</tr>
<tr>
<td>4</td>
<td>Data 2</td>
<td>17</td>
<td>Select</td>
</tr>
<tr>
<td>5</td>
<td>Data 3</td>
<td>18</td>
<td>Ground</td>
</tr>
<tr>
<td>6</td>
<td>Data 4</td>
<td>19</td>
<td>Ground</td>
</tr>
<tr>
<td>7</td>
<td>Data 5</td>
<td>20</td>
<td>Ground</td>
</tr>
<tr>
<td>8</td>
<td>Data 6</td>
<td>21</td>
<td>Ground</td>
</tr>
<tr>
<td>9</td>
<td>Data 7</td>
<td>22</td>
<td>Ground</td>
</tr>
<tr>
<td>10</td>
<td>Ack</td>
<td>23</td>
<td>Ground</td>
</tr>
<tr>
<td>11</td>
<td>Busy</td>
<td>24</td>
<td>Ground</td>
</tr>
<tr>
<td>12</td>
<td>Paper Empty</td>
<td>25</td>
<td>Ground</td>
</tr>
</tbody>
</table>

Parallel printer port in assignments and types of interface signals:

- **Data**: Data0, Data1, ..., Data7
- **Control**: Strobe, Auto Fox, Initialize, Select
- **Status**: Ack, Busy, Paper Empty, Select, Error

I/O interface that employs handshaking:
10.4 Input/Output Handshaking and a Parallel Printer Interface

Handshake sequence flowchart

Handshaking printer interface circuit
10.4 Input/Output Handshaking and a Parallel Printer Interface

EXAMPLE

What are the addresses of the ports that provide the data lines, strobe output, and busy input in the circuit shown in the previous figure? Assume that all unused address bits are 0s.

Solution:

The I/O address that enable port 0 for the data lines, port 1 for the strobe output, and port 2 for the busy input are found as follows:

Address of port 0 = 1000000000000000₂ = 8000₁₆
Address of port 1 = 1000000000000010₂ = 8002₁₆
Address of port 2 = 1000000000000100₂ = 8004₁₆

EXAMPLE

Write a program that will implement the sequence for the circuit in the previous figure. Character data are held in memory starting at address PRNT BUFF, and the number of characters held in the buffer is identified by the count at address CHAR_COUNT. Use the port address from the previous example.

Solution:

First, the character counter and the character points are setup with the instructions

MOV CL, CHAR_COUNT ; (CL) = character count
MOV SI, PRNT BUFF ; (SI) = character pointer
Next, the BUSY input is checked with the instructions:

```
POLL_BUSY     MOV  DX, 8004H    ; Keep polling till busy = 0
              IN   AL, DX
              AND  AL, 01H
              JNZ  POLL_BUSY
```

The character is copied into AL, and then it is output to port 0:

```
MOV   AL, [SI]    ; Get the next character
MOV   DX, 8000H
OUT   DX, AL     ; and output it to port 0
```

Now, a strobe pulse is generated at port 1 with the instructions:

```
MOV   AL, 00H    ; STB = 0
MOV   DX, 8002H
OUT   DX, AL
MOV   BX, 0FH    ; Delay for STB duration
STROBE:     DEC  BX
              JNZ  STROBE
MOV   AL, 01H    ; STB = 1
OUT   DX, AL
```
10.4 Input/Output Handshaking and a Parallel Printer Interface

At this point, the value of PRNT_BUFF must be incremented, and the value of CHAR_COUNT must be decremented:

```
INC SI ; Update character counter
DEC CL ; and pointer
```

Finally, a check is made to see if the printer buffer is empty. If it is not empty, we need to repeat the prior instruction sequence. To do this, we execute the instruction

```
JNZ POLL_BUSY ; Repeat till all character
```

DONE: -

The program comes to the DONE label after all characters are transferred to the printer.

10.5 82C55A Programmable Peripheral Interface

- The 82C55A is an LSI peripheral designed to permit easy implementation of parallel I/O in the 8088- and 8086-microcomputer system.
- Flexible parallel interface:
  - Single-bit, 4-bit, and byte-wide input and output ports
  - Level sensitive inputs
  - Latched outputs
  - Strobed inputs or outputs
  - Strobed bidirectional input/outputs
- Timing of the data transfers to the 82C55A is controlled by the read/write control (RD and WR) signals.
10.5 82C55A Programmable Peripheral Interface

- The source or destination register within the 82C55A is selected by a 2-bit register select code \((A_1A_0)\).
- The chip-select (CS) input must be logic 0 during all read or write operations to the 82C55A.
- The reset (RESET) is used to initialize the 82C55A.
- Three byte-wide ports (port A, port B, port C) can be configured for input or output operation. This gives us a total of 24 I/O lines.
- The 82C55A contains an 8-bit internal control register for software control.
- A write bus cycle to the 82C55A with register-select code \(A_1A_0 = 11\), and an appropriate control word is used to modify the control register.
10.5 82C55A Programmable Peripheral Interface

Addressing an 82C55A using the microcomputer interface signals

EXAMPLE

What is the addresses of port A, port B, port C of the 82C55A device?

Solution:

To access port A, A1A0 = 00, A15 = A14 = 1, A13 = A12 = … = A2 = 0, which gives the port A address as

1100 0000 0000 00002 = C00016

Similarly, it can be determined that the address of port B equals C00116, that of port C is C00216, and the address of the control register is C00316.
10.5 82C55A Programmable Peripheral Interface

- Control-word bit functions

10.5 82C55A Programmable Peripheral Interface

- Mode 0 – Simple I/O operation

<table>
<thead>
<tr>
<th>Pin</th>
<th>Mode 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IN</td>
</tr>
<tr>
<td>PA0</td>
<td>IN</td>
</tr>
<tr>
<td>PA1</td>
<td>IN</td>
</tr>
<tr>
<td>PA2</td>
<td>IN</td>
</tr>
<tr>
<td>PA3</td>
<td>IN</td>
</tr>
<tr>
<td>PA4</td>
<td>IN</td>
</tr>
<tr>
<td>PA5</td>
<td>IN</td>
</tr>
<tr>
<td>PB0</td>
<td>IN</td>
</tr>
<tr>
<td>PB1</td>
<td>IN</td>
</tr>
<tr>
<td>PB2</td>
<td>IN</td>
</tr>
<tr>
<td>PB3</td>
<td>IN</td>
</tr>
<tr>
<td>PB4</td>
<td>IN</td>
</tr>
<tr>
<td>PB5</td>
<td>IN</td>
</tr>
<tr>
<td>PC0</td>
<td>IN</td>
</tr>
<tr>
<td>PC1</td>
<td>IN</td>
</tr>
<tr>
<td>PC2</td>
<td>IN</td>
</tr>
<tr>
<td>PC3</td>
<td>IN</td>
</tr>
<tr>
<td>PC4</td>
<td>IN</td>
</tr>
<tr>
<td>PC5</td>
<td>IN</td>
</tr>
</tbody>
</table>

Mode 0 port pin functions
**10.5 82C55A Programmable Peripheral Interface**

**EXAMPLE**

What is the mode and I/O configuration for ports A, B, and C of an 82C55A after its control register is loaded with $82_{16}$?

**Solution:**

Expressing the control register contents in binary form, we get

$$D_7D_6D_5D_4D_3D_2D_1D_0 = 10000010_2$$

Since $D_7$ is 1, the modes of operation of the ports are selected by the control word.

- $D_0 = 0$ Lower four bits of port C are outputs.
- $D_1 = 1$ Port B is an input port.
- $D_2 = 0$ Mode 0 for both port B and the lower four bits of port C.

---

The next four bits configure the upper part of port C and port A:

- $D_3 = 0$ Upper four bits of port C are outputs.
- $D_4 = 0$ Port A is an output port.
- $D_5D_6 = 00$ Mode 0 for both port A and the upper four bits of port C.
10.5 82C55A Programmable Peripheral Interface

- Mode 0 – Simple I/O operation

Mode 0 control words and corresponding input/output configurations.
10.5 82C55A Programmable Peripheral Interface

- Mode 0 – Simple I/O operation

Mode 0 control words and corresponding input/output configurations.
10.5 82C55A Programmable Peripheral Interface

- Mode 1 – Strobed I/O
  - In mode 1, the A and B ports are configured as two independent byte-wide I/O ports, each of which has a 4-bit control/data port associated with it. The control/data ports are formed from the lower and upper nibbles of port C, respectively.
  - In mode 1, data applied to an input port must be strobed in with a signal produced in external hardware.
  - An output port in mode 1 is provided with handshake signals that indicate when new data are available at its outputs and when an external device has read these values.

---

10.5 82C55A Programmable Peripheral Interface

- Mode 1 – Strobed I/O

<table>
<thead>
<tr>
<th>Pin</th>
<th>IN</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA0</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PA1</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PA2</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PA3</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PA4</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PA5</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PA6</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PA7</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PB0</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PB1</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PB2</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PB3</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PB4</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PB5</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PB6</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PB7</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PC0</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PC1</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PC2</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PC3</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PC4</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PC5</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PC6</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>PC7</td>
<td>IN</td>
<td>OUT</td>
</tr>
</tbody>
</table>

Mode 1 port pin functions
10.5 82C55A Programmable Peripheral Interface

- Mode 1 – Strobed I/O

Mode 1, port A output and input configuration

10.5 82C55A Programmable Peripheral Interface

- Mode 1 – Strobed I/O

Mode 1, port A input and output timing diagram
10.5 82C55A Programmable Peripheral Interface

EXAMPLE

The following figures show how port B can be configured for mode 1 operation. Describe what happens in the left figure when the STB_b input is pulsed to logic 0. Assume that INTE_B is already set to 1.

SOLUTION:

As STB_B is pulsed, the byte of data at PB_7 through PB_0 is latched into the port B register. This causes the IBF_B output to switch to 1. Since INTE_B is 1, INTR_B switches to logic 1.
10.5 82C55A Programmable Peripheral Interface

- Mode 2 – Strobed bidirectional I/O

Mode 2 port pin functions

Mode 2 input/output configuration
10.5 82C55A Programmable Peripheral Interface

- Mode 2 – Strobed bidirectional I/O

**EXAMPLE**

The interrupt-control flag INTEₐ for output port A in mode 1 is controlled by PC₆. Using the set/reset feature of the 82C55A, what command code must be written to the control register of the 82C55A to set it to enable the control flag?

**Solution:**

To use the set/reset feature, D₇ must be logic 0. Moreover, INTEₐ is to be set; therefore, D₀ must be logic 1. Finally, to select PC₆, the code at bits D₃D₂D₁ must be 110. The rest of the bits are don’t-care states. This gives us the control word

\[ D₇D₆D₅D₄D₃D₂D₁D₀ = 0XXX1101₂ \]

Replacing the don’t-care states with the 0 logic level, we get

\[ D₇D₆D₅D₄D₃D₂D₁D₀ = 00001101₂ = 0D₁₆ \]
10.5 82C55A Programmable Peripheral Interface

- Mixed modes

Combined mode 2 and mode 0 (input) control word and I/O configuration

10.5 82C55A Programmable Peripheral Interface

- Mixed modes

Combined mode 2 and mode 1 (output) control word and I/O configuration
10.5 82C55A Programmable Peripheral Interface

EXAMPLE

What control word must be written into the control register of the 82C55A such that port A is configured for bidirectional operation and port B is set up with mode 1 outputs?

Solution:

To configure the operating mode of the ports of the 82C55A, D7 must be 1. Port A is set up for bidirectional operation by making D6 logic 1. In this case, D5 through D3 are don’t-care states:

D7D6D5D4D3 = XXX2

Mode 1 is selected for port B by logic 1 in D2 and output operation by logic 0 in D1. D0 is a don’t-care state.

This gives the control word

D7D6D5D4D3D2D1D0 = 11XXX10X2

= 110001002 = C416

EXAMPLE

Write the sequence of instructions needed to load the control register of an 82C55A with the control word formed in the previous example. Assume that the control register of the 82C55A resides at address 0F16 of the I/O address space?

Solution:

First we must load AL with C416. This is the value of the control word that is to be written to the control register at address 0F16. The move instruction used to load AL is

MOV AL, 0C4H

These data are output to the control register with OUT instruction

OUT 0FH, AL

Because the I/O address of the control register is less than FF16, this instruction uses direct I/O.
10.5 82C55A Programmable Peripheral Interface

When the 82C55A is configured in mode 1 or mode 2 operations, most of the pins of port C perform I/O control functions.

10.6 82C55A Implementation of Parallel Input/Output Ports
EXAMPLE

What must be the address bus inputs of the circuit in the previous figure if port C of PPI 14 is to be accessed?

Solution:

To enable PPI 14, the 74F138 must be enabled for operation and its O2 output switched to logic 0. This requires enable input G2B = 0 and chip select code CBA = 111. This in turn requires from the bus that

\[ A_0 = 0 \] to enable 74F138

and \[ A_2A_1A_0 = 111 \] to select PPI 14

Port C of PPI 14 is selected with \[ A_1A_0 = 10 \], which from the bus requires that

\[ A_2A_1 = 10 \]

The rest of the address bits are don’t-care states.

EXAMPLE

Assume that in the previous figure, PPI 14 is configured so that port A is an output port, both ports B and C are input ports, and all three ports are set up for mode 0 operation. Write a program that will input that data at port B and C, find the difference (port C) – (port B), and output this difference to port A.

Solution:

Port A address = 001110002 = 3816

Similarly, Port B address = 3A16, Port C address = 3C16

Therefore,

\[
\begin{align*}
\text{IN} & \quad \text{AL}, 3\text{AH} ; \quad \text{Read port B} \\
\text{MOV} & \quad \text{BL}, \text{AL} ; \quad \text{Save data from port B} \\
\text{IN} & \quad \text{AL}, 3\text{CH} ; \quad \text{Read port C} \\
\text{SUB} & \quad \text{AL}, \text{BL} ; \quad \text{Subtract B from C} \\
\text{OUT} & \quad 38H, \text{AL} ; \quad \text{Write to port A}
\end{align*}
\]
10.6 82C55A Implementation of Parallel Input/Output Ports

82C55A parallel I/O ports in an 8086-based microcomputer

10.7 Memory-Mapped Input/Output Ports

- The full 20-bit address is available for addressing I/O. Therefore, memory-mapped I/O devices can reside anywhere in the 1Mbyte memory address space of the 8088.
- During I/O operations, memory read and write bus cycles are initiated instead of I/O bus cycles.
- Memory instructions, not input/output instructions, are used to perform data transfer.
10.7 Memory-Mapped Input/Output Ports

EXAMPLE

Which I/O port in the previous figure is selected for operation when the memory address output on the bus is 0040216?

Solution:

We begin by converting the address to binary form. This gives

\[ A_{19} \ldots A_0 = 00000000010000000010_2 \]

In this address, bits \( A_{10} = 1 \) and \( A_0 = 0 \). Therefore, the 74F138 address decoder is enabled whenever \( IO/M = 0 \).

\[ A_5A_4A_3 = 000 \]

This input code switches decoder output \( O_0 \) to logic 0 and chip selects PPI 0 for operation.

The address bits applied to the port select inputs of the PPI are \( A_2A_1 = 01 \). These inputs cause port B to be accessed. Thus, the address 0040216 selects port B on PPI 0 for memory-map I/O.
10.7 Memory-Mapped Input/Output Ports

EXAMPLE

Write the sequence of instructions needed to initialize the control register of PPI 0 in the circuit of the previous figure so that port A is an output port, ports B and C are input ports, and all three ports are configured for mode 0 operation.

Solution:

The control byte required to provide this configuration is:

```
000000000100000001102 = 0040616
```

From the circuit diagram, the memory address of the control register for PPI 0 is found to be

```
000000000100000001102 = 0040616
```

Since PPI 0 is memory mapped, the following move instructions can be used to initialized the control register:

```
MOV AX, 0 ; Create data segment at 00000H
MOV DS, AX
MOV AL, 8BH ; Load AL with control byte
MOV [406H], AL ; Write control byte to PPI 0 control register
```
10.7 Memory-Mapped Input/Output Ports

**EXAMPLE**

Assume that PPI 0 in the previous figure is configured as described in the previous example. Write a program that will input the contents of ports B and C, AND them together, and output the results to port A.

**Solution:**

The addresses of the three I/O ports on PPI 0 are:

- Port A = 0040016
- Port B = 0040216
- Port C = 0040416

Now we set up a data segment at 00000H and the program is:

```assembly
AND AX, 0 ; Create data segment at 00000H
MOV DS, AX
MOV BL, [402H] ; Read port B
MOV AL, [404H] ; Read port C
AND AL, BL ; AND data at port B and C
MOV [400H], AL ; Write to port A
```

Memory-mapped 82C55A parallel I/O ports in an 8086-based microcomputer
10.8 82C54 Programmable Interval Timer

- Block diagram of the 82C54
  - The 82C54 is an LSI peripheral designed to permit easy implementation of *timer* and *counter functions* in a microcomputer system.
  - The 82C54C can be memory-mapped into the memory address space or I/O-mapped into the I/O address space.
  - The microcomputer interface of the 82C54 allows the MPU to read from and write into its internal registers.
  - The 3 counters in 82C54 are each 16 bits in length and operate as *down counters*. 

---

**Block diagram and pin layout of the 82C54 interval timer**
10.8 82C54 Programmable Interval Timer

Block diagram of the 82C54

<table>
<thead>
<tr>
<th>Signals</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>D&lt;sub&gt;7&lt;/sub&gt;–D&lt;sub&gt;0&lt;/sub&gt;</td>
<td>8-bit bidirectional data bus</td>
</tr>
<tr>
<td>A&lt;sub&gt;0&lt;/sub&gt;, A&lt;sub&gt;1&lt;/sub&gt;</td>
<td>Register address inputs used to select the register to be accessed</td>
</tr>
<tr>
<td>RD, WR</td>
<td>Control signals indicating whether 82C54 is to be read from or written into</td>
</tr>
<tr>
<td>CS</td>
<td>Chip-select input to enable the 82C54’s microprocessor interface</td>
</tr>
<tr>
<td>CLK&lt;sub&gt;0&lt;/sub&gt;</td>
<td>Pulses applied to the clock input are used to decrement counter 0</td>
</tr>
<tr>
<td>GATE&lt;sub&gt;0&lt;/sub&gt;</td>
<td>Gate must be switched to logic 1 to enable the counter 0</td>
</tr>
<tr>
<td>OUT&lt;sub&gt;0&lt;/sub&gt;</td>
<td>Clock or pulse output of counter 0</td>
</tr>
</tbody>
</table>

Architecture of the 82C54

Internal architecture of the 82C54 interval timer
10.8 82C54 Programmable Interval Timer

### Architecture of the 82C54

**Control word format**

<table>
<thead>
<tr>
<th>SC1</th>
<th>SC0</th>
<th>RW/R/W</th>
<th>M0</th>
<th>M1</th>
<th>M2</th>
<th>BCD</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D8</td>
<td>D9</td>
<td>D4</td>
<td>D5</td>
<td>D6</td>
<td>D7</td>
</tr>
</tbody>
</table>

**Definition of control**

- SC: Select counter
  - SC1, SC0
  - 00: Select counter 0
  - 01: Select counter 1
  - 10: Select counter 2
  - 11: Read back command

- RW: Read/write
  - RW/R/W
  - 00: Counter latch command
  - 01: Read/write least significant byte only
  - 10: Read/write least significant byte only
  - 11: Read/write least significant byte first, then most significant byte

- BCD: Binary coded decimal
  - 0: Binary counter 16 bits
  - 1: Binary coded decimal (BCD) counter (of octave)

---

**EXAMPLE**

An 82C54 receive the control word 10010000₂. What configuration is set up for the timer?

**Solution:**

- SC bits = 10 indicating counter 2 is selected.
- RW bits = 01 sets counter 2 for the read/write sequence identified as the least significant byte only.
- The mode code M2M1M0 is 000, this selects mode 0 operation for counter 2.
- The last bit, BCD, is also set to 0 and selects binary counting.
10.8 82C54 Programmable Interval Timer

Architecture of the 82C54

<table>
<thead>
<tr>
<th>CS</th>
<th>RD</th>
<th>WR</th>
<th>A1</th>
<th>A0</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Write into Counter 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Write into Counter 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Write into Counter 2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Write Control Word</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Read from Counter 0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Read from Counter 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Read from Counter 2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>No-Operation (3-State)</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>No-Operation (3-State)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>No-Operation (3-State)</td>
</tr>
</tbody>
</table>

Accessing the registers of the 82C54 interval timer

EXAMPLE

Write an instruction sequence to set up the three counters of the 82C54 in the figure that follows:

Counter 0: Binary counter operating in mode 0, value = 1234H
Counter 1: BCD counter operating in mode 2, value = 0100H
Counter 2: Binary counter operating in mode 4, value = 1FFFH

Solution:

First, we need to determine the base address of the 82C54. The base address, which is also the address of counter 0, is determined with A1A0 set to 00. In the figure, we find that to select 82C54, CS must be logic 0. This requires that

\[ A_{i5}A_{i4}...A_2A_6A_5A_4A_3 = 000000000100002 \]
Combining this part of the address with the 00 at A1A0, gives the base address as

\[ \text{00000000010000002 = 40H} \]

Since the base address of the 82C54 is 40H, and to select the mode register requires \( A_1A_0 = 11 \), its address is 43H. Similarly, the three counters 0, 1, and 2 are at addresses 40H, 41H, 42H, respectively. Let's determine the mode words for the three counters.

- Mode word for counter 0 = \( 001100002 = 30_{16} \)
- Mode word for counter 1 = \( 010101012 = 55_{16} \)
- Mode word for counter 2 = \( 101110002 = \text{B8}_{16} \)
10.8 82C54 Programmable Interval Timer

MOV AL, 30H ; Set up counter 0 mode
OUT 43H, AL
MOV AL, 55H ; Set up counter 1 mode
OUT 43H, AL
MOV AL, 0B8H ; Set up counter 2 mode
OUT 43H, AL
MOV AL, 1234H ; Initialize counter 0 with 1234H
OUT 40H, AL
MOV AL, 12H
OUT 40H, AL
MOV AL, 0100H ; Initialize counter 1 with 0100H
OUT 41H, AL
MOV AL, 01H
OUT 41H, AL
MOV AL, 1FFFH ; Initialize counter 2 with 1FFFFH
OUT 42H, AL
MOV AL, 1FH
OUT 42H, AL

10.8 82C54 Programmable Interval Timer

EXAMPLE

Write an instruction sequence to read the contents of counter 2 on the fly. The count is to be loaded into the AX register. Assume that the 82C54 is located at I/O address 40H.

Solution:

First, we latch that contents of counter 2 and then read this value from the temporary storage register.

MOV AL, 1000XXXXB ; Latch counter 2. XXXX must be as per
; the mode and counter type
OUT 43H, AL
IN AL, 42H ; Read the low byte
MOV BL, AL
IN AL, 42H ; Read the high byte
MOV AH, AL
MOV AL, BL ; (AX) = counter 2 value
10.8 82C54 Programmable Interval Timer

Architecture of the 82C54

- Read-back mode permits a programmer to capture the current count values and status information of all three counters with a single command.
- A read back command has bits D6 and D7 both set to 1.

Read-back command format

```
A0, A1 = 11  CB = 0  RD = 1  WR = 0
```

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 0 0 0 0 1 0</td>
<td>Read back count and status of Counter 0</td>
<td>Count and status latched for Counter 0</td>
</tr>
<tr>
<td>1 1 1 0 1 0 0 0</td>
<td>Read back status of Counter 1</td>
<td>Status latched for Counter 1</td>
</tr>
<tr>
<td>1 1 1 0 1 1 0 0</td>
<td>Read back status of Counters 2, 1</td>
<td>Status latched for Counters 1 and 2</td>
</tr>
<tr>
<td>1 1 0 1 1 0 0 0</td>
<td>Read back count of Counter 2</td>
<td>Count latched for Counter 2</td>
</tr>
<tr>
<td>1 1 0 0 0 1 0 0</td>
<td>Read back count and status of Counter 1</td>
<td>Count and status latched for Counter 1</td>
</tr>
<tr>
<td>1 1 1 0 0 0 1 0</td>
<td>Read back status of Counter 0</td>
<td>Status latched for Counter 0</td>
</tr>
</tbody>
</table>

Read-back command examples
10.8 82C54 Programmable Interval Timer

- Architecture of the 82C54

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT</td>
<td>NULL COUNT</td>
<td>RW1</td>
<td>RW0</td>
<td>M2</td>
<td>M1</td>
<td>M0</td>
<td>BCD</td>
</tr>
</tbody>
</table>

- Status byte format

D7: 1 = Out pin is 1
0 = Out pin is 0
D6: 1 = Null count
0 = Count available for reading
D5-D0: Counter Programmed Mode

10.8 82C54 Programmable Interval Timer

- Operating modes of 82C54 counters

10.8 82C54 Programmable Interval Timer

- Operating modes of 82C54 counters

10.8 82C54 Programmable Interval Timer

- Operating modes of 82C54 counters
### 10.8 82C54 Programmable Interval Timer

#### Operating modes of 82C54 counters

<table>
<thead>
<tr>
<th>Signal Status Modes</th>
<th>LOW Or Gating Logic</th>
<th>Rising</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disable counting</td>
<td>—</td>
<td>Enable counting</td>
</tr>
</tbody>
</table>
| 1                   | Disable counting    | 1) Initiates counting  
2) Replaces output after next clock | —      |
| 2                   | Disable counting    | 1) Initiates counting  
2) Sets output immediately high | Enable counting |
| 3                   | Disable counting    | 1) Initiates counting  
2) Sets output immediately high | Enable counting |
| 4                   | Disable counting    | —      | Enable counting |
| 5                   | —                   | —      | —    |

**Effect of the GATE input for each mode**

---

**EXAMPLE**

The counter in the following figure is programmed to operate in mode 0. Assuming that the decimal value 100 is written into the counter, compute the time delay ($T_D$) that occurs until the positive transition takes place at the counter 0 output. The counter is configured for BCD counting. Assume the relationship between the $GATE_0$ and the $CLK_0$ signal as shown in the figure.

**Solution:**

Once loaded, counter 0 needs to count down for 100 pulses at the clock input. During this period, the counter is disabled by logic 0 at the $GATE_0$ input for two clock periods. Therefore, the time delay is calculated as

$$T_D = (n + 1 + d) \times T_{CLK0}$$

$$= (100 + 1 + 2) \times (1/1.19318) \ \mu s$$

$$= 86.3 \ \mu s$$
EXAMPLE

Counter 1 of an 82C54 is programmed to operate in mode 1 and is loaded with the decimal value 10. The gate and clock inputs are as shown in the figure below. How long is the output pulse? Assume that the counter is configured for BCD counting.

Solution:

The GATE1 input in the figure show that the counter is operated as a nonretriggerable one-shot. Therefore, the pulse width is

\[ T = (\text{counter contents})(\text{clock period}) \]

\[ = (10)(\frac{1}{1.19318}) \mu s \]

\[ = 8.38 \mu s \]
**10.8 82C54 Programmable Interval Timer**

**EXAMPLE**

Counter 1 of an 82C54, as shown, is programmed to operate in **mode 2** and is loaded with the decimal value 18. Describe the signal produced at OUT\(_1\). Assume that the counter is configured for BCD counting.

Solution:

In mode 2 the output goes low for one period of the input clock after the counter contents decrement to 0. Therefore,

\[ T_2 = \frac{1}{1.19318 \text{ MHz}} = 838 \text{ ns} \]

and

\[ T = 18 \times T_2 = 15.094 \mu\text{s} \]

**EXAMPLE**

The 82C54 counter, as shown, is programmed to operate in **mode 3** and is loaded with the decimal value 15. Determine the characteristics of the square wave at OUT\(_1\). Assume that the counter is configured for BCD counting.

Solution:

\[ T_{CLK1} = \frac{1}{1.19318 \text{ MHz}} = 838 \text{ ns} \]

\[ T_1 = \frac{T_{CLK1}(N+1)}{2} \]

\[ = 838 \text{ ns} \times \frac{15+1}{2} = 6.704 \mu\text{s} \]

\[ T_2 = \frac{T_{CLK1}(N-1)}{2} \]

\[ = 838 \text{ ns} \times \frac{15-1}{2} = 5.866 \mu\text{s} \]

\[ T = T_1 + T_2 = 6.704 \mu\text{s} + 5.866 \mu\text{s} = 12.57 \mu\text{s} \]
10.8 82C54 Programmable Interval Timer

EXAMPLE

The 82C54 counter, as shown, is programmed to operate in mode 4. What value must be loaded into the counter to produce a strobe signal 10 μs after the counter is loaded?

Solution:

The strobe pulse occurs after counting down the counter to zero. The number of input clock periods required for a period of 10 μs is given by

\[ N = \frac{T}{T_{CLK}} \]

\[ = \frac{10 \text{ μs}}{1/1.19318 \text{ MHz}} \]

\[ = 1210 = C_{16} = 000011002 \]

Thus, the counter should be loaded with the number \( n=0B_{16} \) to produce a strobe pulse 10 μs after loading.

10.9 82C37A Programmable Direct Memory Access Controller

- The 82C37A is an LSI controller IC that is widely used to implement the **direct memory (DMA)** function in the 8088/8086 microcomputer.
- DMA capability permits devices to perform high-speed data transfers between either two sections of memory or between memory and an I/O device.
- The memory or I/O bus cycles initiated as part of a DMA transfer are not performed by the MPU; instead, they are performed by DMA controllers, such as 82C37A.
- A single DMA device supports up to four peripheral devices for DMA operation.
10.9 82C37A Programmable Direct Memory Access Controller

- Microprocessor interface of the 82C37A

![Diagram showing Microprocessor interface of the 82C37A]

Block diagram and pin layout of the 82C37A DMA controller

---

10.9 82C37A Programmable Direct Memory Access Controller

- Microprocessor interface of the 82C37A

![Diagram showing Microprocessor interface of 82C37A to the 8088]

Microprocessor interface of 82C37A to the 8088
### 10.9 82C37A Programmable Direct Memory Access Controller

- **DMA interface of the 82C37A**

- 82C37A contains four independent channels, channel 0 through 3.
- When a peripheral device wants to perform DMA, it makes a request for service at 82C37A DREQ input by switching it to login 1.
- During DMA bus cycles, the DMA controller, not the MPU, drives the system bus. The 82C37A generates the address and all control signals to perform the memory or I/O data transfer.
- The 82C37A performs both the memory-to-I/O and I/O-to-memory DMA bus cycles in just four clock periods. The memory-to-memory data transfer takes 8 clock periods.
10.9 82C37A Programmable Direct Memory Access Controller

Internal architecture of the 82C37A

- The timing and control part of the 82C37A generates the timing and control signals needed by the external bus interface.
- The priority logic circuitry resolves priority for simultaneous DMA requests from peripheral devices based on either fixed priority or rotating priority scheme.
- The command control circuit decodes the register commands applied to the 82C37A through the microprocessor interface.
- Each DMA channel has two address register: the base address register and the current address register.
- Each DMA channel has two word-count register to specify the number of bytes of data to be transferred.
10.9 82C37A Programmable Direct Memory Access Controller

- Internal architecture of the 82C37A

  - 82C37A has 12 different types of internal registers.

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Address Registers</td>
<td>16 bits</td>
<td>4</td>
</tr>
<tr>
<td>Base Word Count Registers</td>
<td>16 bits</td>
<td>4</td>
</tr>
<tr>
<td>Current Address Registers</td>
<td>16 bits</td>
<td>4</td>
</tr>
<tr>
<td>Current Word Count Registers</td>
<td>16 bits</td>
<td>4</td>
</tr>
<tr>
<td>Temporary Address Register</td>
<td>16 bits</td>
<td>1</td>
</tr>
<tr>
<td>Temporary Word Count Register</td>
<td>16 bits</td>
<td>1</td>
</tr>
<tr>
<td>Status Register</td>
<td>8 bits</td>
<td>1</td>
</tr>
<tr>
<td>Command Register</td>
<td>8 bits</td>
<td>1</td>
</tr>
<tr>
<td>Temporary Register</td>
<td>8 bits</td>
<td>1</td>
</tr>
<tr>
<td>Mode Register</td>
<td>6 bits</td>
<td>4</td>
</tr>
<tr>
<td>Mask Register</td>
<td>4 bits</td>
<td>1</td>
</tr>
<tr>
<td>Request Register</td>
<td>4 bits</td>
<td>1</td>
</tr>
</tbody>
</table>

- Accessing the registers of the 82C37A
10.9 82C37A Programmable Direct Memory Access Controller

Internal architecture of the 82C37A

- The command register is used to control operating modes that apply to all channels of the DMA controller.

### Command register format

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Memory-to-memory enable</td>
</tr>
<tr>
<td>6</td>
<td>Memory-to-memory disable</td>
</tr>
<tr>
<td>5</td>
<td>Channel 0 address hold enable</td>
</tr>
<tr>
<td>4</td>
<td>Channel 0 address hold disable</td>
</tr>
<tr>
<td>3</td>
<td>Controller enable</td>
</tr>
<tr>
<td>2</td>
<td>Controller disable</td>
</tr>
<tr>
<td>1</td>
<td>Normal timing</td>
</tr>
<tr>
<td>0</td>
<td>Compressed timing</td>
</tr>
<tr>
<td></td>
<td>High priority</td>
</tr>
<tr>
<td></td>
<td>Low priority</td>
</tr>
<tr>
<td></td>
<td>Late write selection</td>
</tr>
<tr>
<td></td>
<td>Extended write selection</td>
</tr>
<tr>
<td></td>
<td>High priority</td>
</tr>
<tr>
<td></td>
<td>Low priority</td>
</tr>
<tr>
<td></td>
<td>DREQ sense active high</td>
</tr>
<tr>
<td></td>
<td>DREQ sense active low</td>
</tr>
<tr>
<td></td>
<td>DACK sense active high</td>
</tr>
<tr>
<td></td>
<td>DACK sense active low</td>
</tr>
</tbody>
</table>

EXAMPLE

If the command register of an 82C37A is loaded with 0116, how does the controller operate?

Solution:

- Bit 0 = 1 = Memory-to-memory transfers are disabled
- Bit 1 = 0 = Channel 0 address increment/decrement normally
- Bit 2 = 0 = 82C37A is enabled
- Bit 3 = 0 = 82C37A operates with normal timing
- Bit 4 = 0 = Channels have fixed priority, channel 0 having the highest priority and channel 3 the lowest priority
- Bit 5 = 0 = Write operation occurs late in the DMA bus cycle
- Bit 6 = 0 = DREQ is an active high (logic 1) signal
- Bit 7 = 0 = DACK is an active low (logic 0) signal
10.9 82C37A Programmable Direct Memory Access Controller

- Internal architecture of the 82C37A
  - The mode register is used to configure operation features of the 82C37A.

```
+---+---+---+---+---+---+---+
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
+---+---+---+---+---+---+---+
| 00 | 01 | 10 | 11 | 11 | 11 | 11 | 11 |
+---+---+---+---+---+---+---+
| 00 | 00 | 01 | 10 | 11 | 11 | 11 | 11 |
+---+---+---+---+---+---+---+
| 00 | 00 | 01 | 10 | 11 | 11 | 11 | 11 |
+---+---+---+---+---+---+---+
| 00 | 00 | 01 | 10 | 11 | 11 | 11 | 11 |
+---+---+---+---+---+---+---+
| 00 | 00 | 01 | 10 | 11 | 11 | 11 | 11 |
+---+---+---+---+---+---+---+
| 00 | 00 | 01 | 10 | 11 | 11 | 11 | 11 |
+---+---+---+---+---+---+---+
```

---

**EXAMPLE**

Specify the mode byte for DMA channel 2 if it is to transfer data from an input peripheral device to a memory buffer starting at address A00016 and ending at AFFF16. Ensure that the microprocessor is not completely locked off the bus during the DMA cycle. Moreover, at the end of each DMA cycle, the channel is to be reinitialized so that the same buffer is filled when the next DMA operation is initiated.

Solution:

For DMA channel 2, \( B_1B_0 = 10 \)

Transfer of data from an I/O device to memory represents a write bus cycle. Therefore, \( B_2B_3 = 01 \)
10.9 82C37A Programmable Direct Memory Access Controller

Selecting autoinitiation will set up the channel to automatically reset. Making bit 4 equal to 1 enables this feature:

\[ B_4 = 1 \]

The address that points to the memory buffer must increment after each data transfer. Therefore

\[ B_5 = 0 \]

To ensure that the 8088 is not locked off the bus during the complete DMA cycle, we will select the single-transfer mode:

\[ B_7B_6 = 01 \]

Thus, the mode register byte is

\[ B_7B_6B_5B_4B_3B_2B_1B_0 = 01010110_2 = 56_{10} \]

---

10.9 82C37A Programmable Direct Memory Access Controller

- Internal architecture of the 82C37A
  - The request register is used to respond to software-initiated requests for DMA services.
  - Any channel used to software-initiated DMA must be programmed for block-transfer mode of operation.

![Request register format](image)
10.9 82C37A Programmable Direct Memory Access Controller

- **Internal architecture of the 82C37A**
  - The 4-bit mask register is used to mask out (to ignore hardware request) the DREQ input to the DMA channels.

```
    Don't Care
  7 8 6 5 4 3 2 1 0
    |   |   |   |   |   |   |
  0: Select channel 0 mask bit
  1: Select channel 1 mask bit
  2: Select channel 2 mask bit
  3: Select channel 3 mask bit
  4: Clear mask bit
  5: Set mask bit
```

Single-channel and four-channel mask-register command format

- The status register contains the information about the operating state of the four channels of the 82C37A.

```
    Don't Care
  7 6 5 4 3 2 1 0
    |   |   |   |   |   |   |
  0: Clear channel 0 mask bit
  1: Set channel 0 mask bit
  2: Clear channel 1 mask bit
  3: Set channel 1 mask bit
  4: Clear channel 2 mask bit
  5: Set channel 2 mask bit
  6: Clear channel 3 mask bit
  7: Set channel 3 mask bit
```

Status register
EXAMPLE

Write an instruction sequence to issue a master clear to the 82C37A and then enable all its DMA channels. Assume that the device is located at base I/O address DMA < F0H.

Solution:

The master clear command is performed by simply writing into the register a relative address $D_{16}$. For instance, the instruction

```
OUT DMA+0DH, AL
```

To enable the DMA request inputs, all 4 bits of the mask register must be cleared. The clear-mask register command is issued by performing a write to the register at relative address $E_{16}$.

```
OUT DMA+0EH, AL
```
10.10 Serial Communication Interface

- Synchronous and asynchronous data communication

![Diagram showing synchronous communications interface and data-transmission format]

Asynchronous communications interface and data-transmission format

![Diagram showing asynchronous communications interface and data-transmission format]
10.10 Serial Communication Interface

- Simplex, half-duplex, and full-duplex communication links

**Simplex communication link**
- Microcomputer
- RS-232 Transmit line
- RS-232 Printer

**Half-duplex communication link**
- Microcomputer
- RS-232 Transmit/Receive line
- CRT terminal with keyboard
10.10 Serial Communication Interface

- Simplex, half-duplex, and full-duplex communication links

![Diagram of full-duplex communication link]

- Baud rate and the baud-rate generator
  - The rate at which data transfers take place over the receive and transmit lines is known as the **baud rate**.
  - By baud rate we mean the number of bits of data transferred per second.
  - Baud rate is set by a part of the serial communication interface called the **baud-rate generator**.
10.10 Serial Communication Interface

EXAMPLE

The data transfer across an asynchronous serial data communications line is observed and the bit time is measured as 0.883 ms. What is the baud rate?

Solution:

Baud rate is calculated from the bit time as

\[
\text{Baud rate} = \frac{1}{t_{bt}} = \frac{1}{0.833 \text{ ms}} = 1200 \text{ bps}
\]

10.10 Serial Communication Interface

- The RS-232C interface
  - The RS-232C interface is a standard hardware interface for implementing asynchronous serial data communication ports on devices such as printers, CRT terminals, keyboards, and modems.
  - Three signal lines can be used to connect the peripheral to the MPU via RS-232C interface: a receive-data line, a transmit-data line, and signal common.
  - The RS-232C standard defines a 25-pin interface.
  - The RS-232C is specified to operate correctly over a distance of up to 100 feet.
10.10 Serial Communication Interface

The RS-232C interface

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Protective Ground</td>
</tr>
<tr>
<td>2</td>
<td>Transmitted Data</td>
</tr>
<tr>
<td>3</td>
<td>Received Data</td>
</tr>
<tr>
<td>4</td>
<td>Request to Send</td>
</tr>
<tr>
<td>5</td>
<td>Clear to Send</td>
</tr>
<tr>
<td>6</td>
<td>Data Set Ready</td>
</tr>
<tr>
<td>7</td>
<td>Signal Ground (Common Return)</td>
</tr>
<tr>
<td>8</td>
<td>Received Line Signal Detector</td>
</tr>
<tr>
<td>9</td>
<td>Reserved for Data Set Testing</td>
</tr>
<tr>
<td>10</td>
<td>Reserved for Data Set Testing</td>
</tr>
<tr>
<td>11</td>
<td>Unassigned</td>
</tr>
<tr>
<td>12</td>
<td>Secondary Received Line Signal Detector</td>
</tr>
<tr>
<td>13</td>
<td>Secondary Clear to Send</td>
</tr>
<tr>
<td>14</td>
<td>Secondary Transmitted Data</td>
</tr>
<tr>
<td>15</td>
<td>Transmission Signal Element Timing</td>
</tr>
<tr>
<td>16</td>
<td>Secondary Received Data</td>
</tr>
<tr>
<td>17</td>
<td>Receiver Signal Element Timing</td>
</tr>
<tr>
<td>18</td>
<td>Unassigned</td>
</tr>
<tr>
<td>19</td>
<td>Secondary Request to Send</td>
</tr>
<tr>
<td>20</td>
<td>Data Terminal Ready</td>
</tr>
<tr>
<td>21</td>
<td>Signal Quality Detector</td>
</tr>
<tr>
<td>22</td>
<td>Ring Indicator</td>
</tr>
<tr>
<td>23</td>
<td>Data Signal Rate Selector</td>
</tr>
<tr>
<td>24</td>
<td>Transmit Signal Element Timing</td>
</tr>
<tr>
<td>25</td>
<td>Unassigned</td>
</tr>
</tbody>
</table>

A DTE-to-DTE serial communication connection
10.11 Programmable Communication Interface Controller

- USART – Universal Synchronous and Asynchronous Receiver Transmitter
  - The programmability of the USART provides for a very flexible asynchronous communication interface.
  - Typical USART can be configured through software for communication of data using formats with character length between 5 and 8 bits, with even or odd parity, and with 1, 1.5, or 2 stop bits.
  - A USART has the ability to automatically check characters during data reception to detect the occurrence of **parity**, **framing**, and **overrun errors**.

10.11 Programmable Communication Interface Controller

- 8251A USART
  - 8251A USART includes 4 key sections
    - Bus interface section
    - Transmit section
    - Receive section
    - Modem-control section
  - A UART cannot stand alone in a communication operation; its operation must typically be controlled by a microprocessor.
  - Data transfers over the bidirectional data bus (D₀ through D₇) are controlled by the signals C/D, RD, WR, CS.
10.11 Programmable Communication Interface Controller

- 8251A USART

Block diagram and pin layout of the 8251A

<table>
<thead>
<tr>
<th>C/D</th>
<th>RD</th>
<th>WR</th>
<th>CS</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>8251A Data → Data bus</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Data bus → 8251A Data</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Status → Data bus</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Data bus → Control</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Data bus → 3-state</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>Data bus → 3-state</td>
</tr>
</tbody>
</table>

Read/Write operations
10.11 Programmable Communication Interface Controller

EXAMPLE

What type of data transfer is taking place over the bus if the control signals are at $CS = 0$, $C/D = 1$, $RD = 0$, and $WR = 1$?

Solution:

Looking at the previous table, we see that $CS = 0$ means that the 8251A’s data bus has been enabled for operation. Since $C/D$ is 1 and $RD$ is 0, status information is being read from the 8251A.

---

10.11 Programmable Communication Interface Controller

- 8251A USART
  - The baud rate of the 8251A must be externally generated and applied to the $R_{xc}$ input of the receiver.
  - Through software the 8251A can be set up to internally divide the clock signal input by 1, 16 or 64 to obtain the desired baud rate.
  - The receiver performs serial data to parallel data operation while the transmitter performs parallel to serial data operation.
10.11 Programmable Communication Interface Controller

- 8251A USART

**Receiver and transmitter driven at the same baud rate**

The 8251A can be configured for various modes of operation through software.

**Control registers:**
- Mode-control register
- Command register
- Status register

**Instruction format**
- Baud rate factor (D0, D1)
- Character length (L1, L2)
- Parity enable (PE)
- Even parity check (EP)
- Number of stop bits (S1, S2)
10.11 Programmable Communication Interface Controller

EXAMPLE

What value must be written to the mode-control register in order to configure the 8251A such that it works as an asynchronous communication controller with the baud rate clock internally divided by 16? Character size is 8 bits; parity is odd; and one stop bit is used.

Solution:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud rate factor</td>
<td>B_1B_0 = 10</td>
</tr>
<tr>
<td>Character length</td>
<td>L_2L_1 = 11</td>
</tr>
<tr>
<td>Odd parity</td>
<td>EP PEN = 01</td>
</tr>
<tr>
<td>Stop bit</td>
<td>S_2S_1 = 01</td>
</tr>
</tbody>
</table>

Therefore, the complete control word is

\[ D_7D_6\ldots D_0 = 01011110_2 = 5E_{16} \]

10.11 Programmable Communication Interface Controller

8251A USART

*Command register format*

- Transmit enable, TxEN
- Data terminal, DTR
- Receiver enable, RxEN
- Send break character, SBRK
- Error reset, ER
- Request to send, RTS
- Internal reset, IR
- Enter hunt mode, EH
10.11 Programmable Communication Interface Controller

8251A USART

- Status register format
  - Parity error, PE
  - Overrun error, OE
  - Framing error, FE

8251A initialization flowchart
10.11 Programmable Communication Interface Controller

EXAMPLE

The circuit in the figure below implements serial I/O for the 8088 microprocessor using an 8251A. Write a program that continuously reads serial characters from the RS-232 interface, complements the received characters with software, and sends them back through the RS-232 interface. Each character is received and transmitted as an 8-bit character using 2 stop bits and no parity.

Solution

We must first determine the addresses for the registers in the 8251A that can be accessed from the microprocessor interface. Chip select (CS) is enabled for I/O read or write operations to addresses for which

\[ A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0 = 10000000 \]

Bit \( A_0 \) of the address bus is used to select between the data and control (or status) registers. As shown in the figure below, the addresses for the data and control register are XX80H and XX81H.

<table>
<thead>
<tr>
<th>8251A Register</th>
<th>I/O Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control/Status</td>
<td>( A_{15} A_{14} A_{13} A_{12} A_{11} A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0 )</td>
</tr>
<tr>
<td>Data</td>
<td>( x \times x \times x \times x \times x \times 1 \times 0 \times 0 \times 0 \times 0 \times 0 = \text{xx81H} )</td>
</tr>
<tr>
<td></td>
<td>( x \times x \times x \times x \times x \times 1 \times 0 \times 0 \times 0 \times 0 \times 0 = \text{xx80H} )</td>
</tr>
</tbody>
</table>
10.11 Programmable Communication Interface Controller

Solution

Next we must determine the mode word to select an 8-bit character with 2 stop bits and no parity. As shown in the figure below, the mode word is EEH. Here we have used a baud-rate factor of 16, which means that the baud rate is given as

\[
\text{Baud rate} = \frac{\text{Baud-rate clock}}{16} = \frac{19,200}{16} = 1200 \text{ bps}
\]

To enable the transmitter as well as receiver operation of the 8251A, the command word is equal to 15H.

Flow chart and program for the initialization of the 8251A is shown below.
10.11 Programmable Communication Interface Controller

Solution

The receive operation starts by reading the contents of the status register at address 81H and checking if the LSB, \( R_{xRDY} \) is at logic 1. If it is not 1, the routine keeps reading and checking until it does become 1. Next we read the data register at 80H for the received data. The byte of data received is complemented and then saved for transmission.

The transmit operation also starts by reading the status register at address 81H and checking if bit 1, \( T_{xRDY} \), is logic 1. If it is not, we again keep reading and checking until it becomes 1. Next, the byte of data that was saved for transmission is written to the data register at address 81H. This causes it to be transmitted at the serial interface. The receive and transmit operations are repeated by jumping back to the point where the receive operation begins.

---

8250/16450 UART

- 8250 and 16450 are newer devices than the 8251A UART and implement a more versatile I/O operation.
- New functions include a built-in programmable baud-rate generator, double buffering on communication data registers, and enhanced status and interrupt signaling.
10.11 Programmable Communication Interface Controller

*8250/16450 UART*

![Pin layout and RS-232 interface of the 8250/16450 UART](image)

**Register-select codes of the 8250/16450 UART**

<table>
<thead>
<tr>
<th>DLAB</th>
<th>A3</th>
<th>A2</th>
<th>A1</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Receiver buffer (read),</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Transmitter Holding</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Register (write)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Interrupt Enable</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Interrupt identification (read only)</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Line Control</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Modern Control</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Line Status</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Modern Status</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Scratch</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Division Latch (least significant byte)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Division Latch (most significant byte)</td>
</tr>
</tbody>
</table>
### 10.11 Programmable Communication Interface Controller

#### 8250/16450 UART

**Register bit functions and word-length select bits of the 8250/16450 UART**

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Word Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>5 bits</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>6 bits</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>7 bits</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>8 bits</td>
</tr>
</tbody>
</table>

Baud rates and corresponding divisors of the 8250/16450 UART

<table>
<thead>
<tr>
<th>Desired Baud-rate</th>
<th>Divisor Used to Generate 16 x Clock</th>
<th>Percent Error Difference Between Desired and Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>3840</td>
<td>-</td>
</tr>
<tr>
<td>75</td>
<td>2560</td>
<td>-</td>
</tr>
<tr>
<td>110</td>
<td>1745</td>
<td>0.026</td>
</tr>
<tr>
<td>134.5</td>
<td>1428</td>
<td>0.084</td>
</tr>
<tr>
<td>150</td>
<td>1280</td>
<td>-</td>
</tr>
<tr>
<td>180</td>
<td>107</td>
<td>0.312</td>
</tr>
<tr>
<td>200</td>
<td>96</td>
<td>-</td>
</tr>
<tr>
<td>240</td>
<td>80</td>
<td>-</td>
</tr>
<tr>
<td>300</td>
<td>640</td>
<td>-</td>
</tr>
<tr>
<td>360</td>
<td>54</td>
<td>0.628</td>
</tr>
<tr>
<td>450</td>
<td>40</td>
<td>-</td>
</tr>
<tr>
<td>720</td>
<td>27</td>
<td>1.33</td>
</tr>
<tr>
<td>960</td>
<td>20</td>
<td>-</td>
</tr>
<tr>
<td>1920</td>
<td>10</td>
<td>-</td>
</tr>
<tr>
<td>38400</td>
<td>5</td>
<td>-</td>
</tr>
</tbody>
</table>

**0.034**

**0.026**

**0.084**

**0.312**

**0.628**

**1.33**

**5.00**

**10.00**

**25.00**

**50.00**

**100.00**

**200.00**

**320.00**

**540.00**

**960.00**

**1920.00**

**3840.00**
### 10.11 Programmable Communication Interface Controller

#### EXAMPLE

What count must be loaded into the divisor latch registers to set the data communication rate to 2400 baud? What register-select code must be applied to the 8250/16450 when writing the bytes of the divider count into the DLL and DLM register?

**Solution:**

For 2400 baud rate, the divisor is 80. When writing into DLL, the address must take

\[
A_2A_1A_0 = 000_2 \text{ with } DLAB = 1
\]

and the value that is written is \( DLL = 80 = 50H \)

For DLM, the address must take

\[
A_2A_1A_0 = 001_2 \text{ with } DLAB = 1
\]

and the value is \( DLM = 0 = 00H \)

---

### 8250/16450 UART

- RS-232 interface with EIA drivers
10.12 Keyboard and Display Interface

- The size of the keyboard array is usually described in terms of the number of rows and columns.
- The microcomputer scans the keyboard array to determine which key is pressed.
- **Keyboard debouncing** is achieved by resampling the column lines a second time, about 10 ms later, to assure the same column line is at the 0 logic level.
- **Two-key lockout method** and **N-key rollover method** are usually used to resolve the problem of multiple key depression.
- The way in which the display is driven by the microcomputer is said to be **multiplexed**.
10.12 Keyboard and Display Interface

- Display Interface

Display interface to a microcomputer

- Seven-segment LED

Seven-segment display
10.13 8279 Programmable Keyboard/Display Controller

- The 8279 can drive an 8x8 keyboard switch array and a 16-digit, eight-segment display.
- 8279 has four signal sections:
  - The MPU interface
  - The key data inputs
  - The display data outputs
  - Scan lines used by both the keyboard and display
- The operation of the 8279 must be configured through software. Eight command words are provided for this purpose.
10.13 8279 Programmable Keyboard/Display Controller

- The scan lines (SL0-SL3) are used as row-drive signals for the keyboard and digit-drive signals for the display.
- The scan line can be configured for two different modes of operation through software:
  - Decoded mode
  - Encoded mode

Decoded-mode scan line signals

Encoded-mode scan line signals

System configuration using the 8086 and 8279
If logic 0 is detected at a return line during key scanning, the number of the column is coded as 3-bit binary number and combined with the 3-bit row number to make a 6-bit key code. This key code input is first debounced and then loaded into an 8x8 key code FIFO within the 8279.

Two other input signals, CNTR and SHIFT, are also stored as part of the key code when a switch closure is detected.
10.13 8279 Programmable Keyboard/Display Controller

- A status register is provided within the 8279 that contains the flags indicating the status of the key code FIFO.

**FIFO STATUS WORD**

<table>
<thead>
<tr>
<th>D</th>
<th>O</th>
<th>F</th>
<th>N</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>E</td>
<td>O</td>
<td>F</td>
<td>N</td>
</tr>
</tbody>
</table>

**Status register**

- **FIFO Full**
- **Error-Underrun**
- **Error-Overrun**
- **Sensor Closure/Error Flag for Multiple Closures**
- **Display unavailable**

10.13 8279 Programmable Keyboard/Display Controller

- The **command word 0** is used to set the mode of operation for the keyboard and display.

<table>
<thead>
<tr>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>K</td>
</tr>
<tr>
<td>0</td>
<td>K</td>
</tr>
<tr>
<td>0</td>
<td>K</td>
</tr>
</tbody>
</table>

**Command word 0 format**

<table>
<thead>
<tr>
<th>D</th>
<th>D</th>
<th>Display operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>8-bit character display – Left entry</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>16-bit character display – Left entry</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8-bit character display – Right entry</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>16-bit character display – Right entry</td>
</tr>
</tbody>
</table>

**Display mode select code**
10.13 8279 Programmable Keyboard/Display Controller

The command word 0

<table>
<thead>
<tr>
<th>K</th>
<th>K</th>
<th>K</th>
<th>Keyboard operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Encoded Scan Keyboard – 2-Key Lockout</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Decoded Scan Keyboard – 2-Key Lockout</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Encoded Scan Keyboard – N-Key Rollover</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Decoded Scan Keyboard – N-Key Rollover</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Encoded Scan Sensor Matrix</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Decoded Scan Sensor Matrix</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Strobed Input, Encoded Display Scan</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Strobed Input, Decoded Display Scan</td>
</tr>
</tbody>
</table>

Keyboard select codes

10.13 8279 Programmable Keyboard/Display Controller

EXAMPLE

What should be the value of command word 0 if the display is to be set for eight 8-segment digits with right entry and the keyboard for decoded scan with N-key rollover?

Solution:

The three MSBs of the command word are always 0. The next 2 bits, DD, must be set to 10 for eight 8-segment digits with right entry. Finally, the three LSBs are set to 011 for decoded keyboard scan with N-key rollover. This gives

Command word 0 = 000DDKKK

= 000100112

= 13_{16}
10.13 8279 Programmable Keyboard/Display Controller

The command word 1 is used to set the frequency of operation of the 8279. It is designed to run at 100 kHz; however, in most applications a much higher frequency signal is available to supply its CLK input. For this reason, a 5-bit programmable prescaler is provided within the 8279 to divide down the input frequency.

```
0 0 1 P P P P
```

Command word 1 format

---

10.13 8279 Programmable Keyboard/Display Controller

The command word 6 is used for initialization of the 8279. It is used to initialize the complete display memory, the FIFO status, and the interrupt request output line.

```
1 1 0 C_D C_D C_D C_F C_A
```

Command word 6 format

```
C_D C_D
0 X All zeros (X = Don’t Care)
1 0 AB = Hex 20 (0010 0000)
1 1 All ones
```

Enable clear display when = 1 (or by C_A = 1)

C_D coding
10.13 8279 Programmable Keyboard/Display Controller

EXAMPLE

What clear operations are performed if the value of command word 6 written to the 8279 is D2\textsubscript{16}?

Solution:

First, we express the command word in binary form. This gives

$$\text{Command word 6} = \text{D2}_{16} = 10010010_2$$

Note that the three C\textsubscript{D} bits are 100. This combination causes display memory to be cleared. The C\textsubscript{F} bit is also set, and this causes the FIFO status and IRQ output to be reset.

10.13 8279 Programmable Keyboard/Display Controller

- Only one bit of command word 7 is functional. This bit is labeled E and is an enable signal for what is called the special-error mode. When this mode is enabled and the keyboard has N-key rollover selected, a multiple-key depression causes the S/E flag of the FIFO status register to be set. This flag can be read by the microprocessor through software.

<table>
<thead>
<tr>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>E</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Command word 7 format
10.13 8279 Programmable Keyboard/Display Controller

- The **command word 2** is used to issue the read FIFO command for accessing the key code FIFO.

  MSB  |  LSB  | X = Don’t Care
  ---   |      | A
  010   | A X   | A

  Command word 7 format

- The **command word 4** is used to send new data to the display RAM.

  MSB  |  LSB
  ---   |      
  100   | A A A A

  Command word 4 format

- The **command word 3** is used to read the contents of the display RAM.

  MSB  |  LSB
  ---   |      
  011   | A A A A

  Command word 3 format