8086 processor is a parallel device. But the parallel data transmission over a long distance is very expensive. Hence, before transmission, the data is always converted into serial data bits. And the interfacing device should provide the following capabilities.

1. An input port and output port are required for interfacing.
2. An information must be provided to recognize the beginning and end of transmission.
3. In data transmission, the data should be converted from parallel word mode to serial mode.
4. In data reception, the data should be converted from serial to parallel mode.
5. Data transfer should be synchronized between processing unit and slow responding peripherals.

Depending on the modes that are supported by the hardware, the name of the communication sub-system will usually include an A if it supports Asynchronous communications, and a S if it supports Synchronous communications.

UART Universal Asynchronous Receiver/Transmitter
USART Universal Synchronous-Asynchronous Receiver/Transmitter

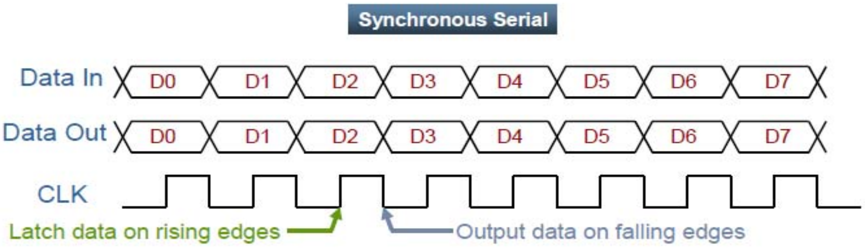
The Universal Asynchronous Receiver/Transmitter (UART) controller is the key component of the serial communications subsystem of a computer. The UART takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes.

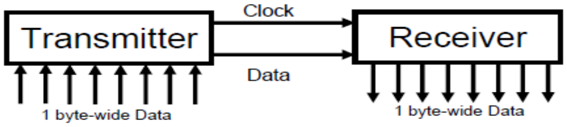
Serial transmission

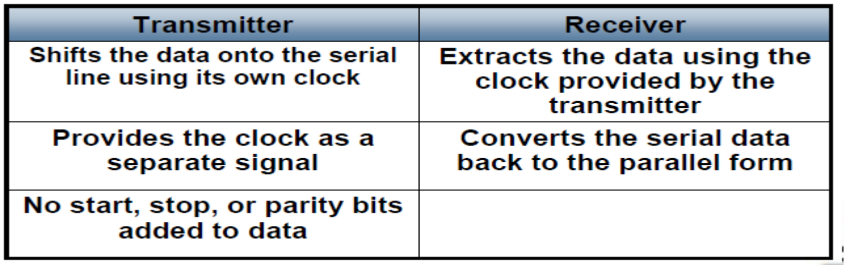
Serial transmission is commonly used with modems and for non-networked communication between computers, terminals and other devices. There are two primary forms of serial transmission: **Synchronous and Asynchronous**.

Synchronous Serial transmission

In synchronous format, a receiver and transmitter are synchronized. In this format, a block of characters are transmitted with a clock signal (synchronizing information) to synchronize the transmission and reception. It allows characters to be sent back to back but must include special “sync” characters at the beginning of each message and special “idle” characters in the data stream to fill up time when no information is being sent. Synchronous communication is usually more efficient because only **data bits** are transmitted between sender and receiver, and synchronous communication can be more costly if extra wiring and circuits are required to share a clock signal between the sender and receiver. Synchronous serial requires the clock signal to be transmitted from the source along with the data. Data rate for the link must be the same for the transmitter and the receiver. The transmitter typically provides the clock as a separate signal in addition to the serial data.





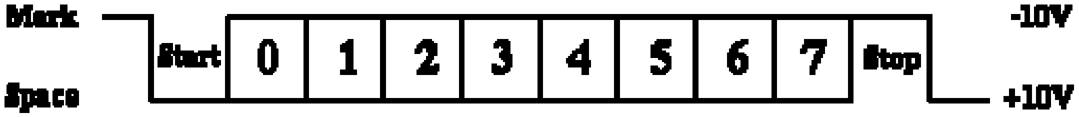


Transmitter	Receiver
Shifts the data onto the serial line using its own clock	Extracts the data using the clock provided by the transmitter
Provides the clock as a separate signal	Converts the serial data back to the parallel form
No start, stop, or parity bits added to data	

As mentioned earlier, in synchronous serial communications, the transmitter sends the data out the pin at a specific rate determined by the internal clock system. This clock is also output along with the data in order for a receiving device to know when to latch the incoming data bits. By sending the clock output along with the data there are no additional bits required to provide a synchronization time for a receiver. The receiving device just latches the data based on the received clock and converts the data internally to a parallel value for use by the CPU. Clock speed, clock polarity and data width are all established at design time.

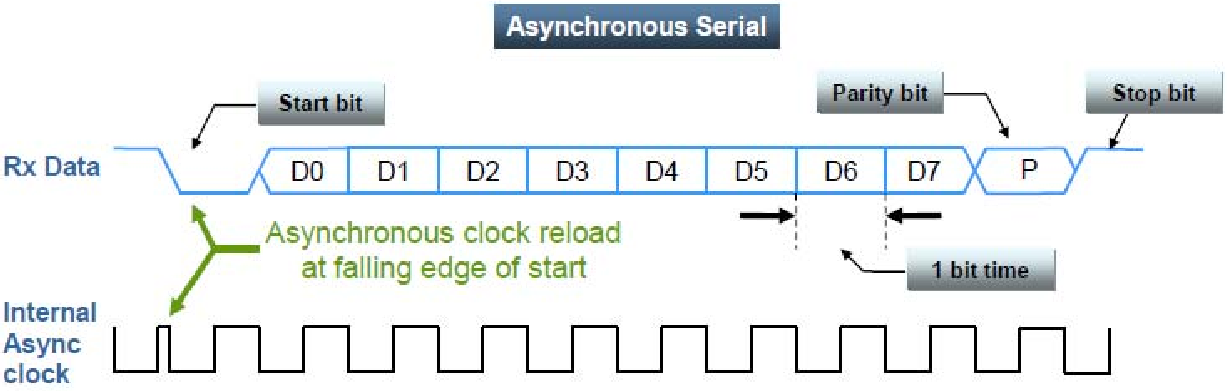
Asynchronous Serial Transmission

Asynchronous transmission allows data to be transmitted without the sender having to send a clock signal to the receiver. Instead, the sender and receiver must agree on timing parameters in advance and special bits are added to each word which is used to synchronize the sending and receiving units. It is character oriented mode of transmission. Before a character begins the line must be in its “1” state, called **mark** state. The transition from the mark state to “0” , or space ,states indicates the beginning of character. When a word is given to the UART for Asynchronous transmissions, the first bit is ‘0’ and is called the "Start Bit" is added to the beginning of each word that is to be transmitted. The Start Bit is used to alert the receiver that a word of data is about to be sent, and to force the clock in the receiver into synchronization with the clock in the transmitter. Transmission begins with the start bit followed by a character and one or two stop bits. Usually start bits are low signals and stop bits are continuous high signals. Asynchronous transmission is easy to implement but less efficient as it requires an extra 2-3 control bits for every 8 data bits .This method is usually used for low volume transmission.

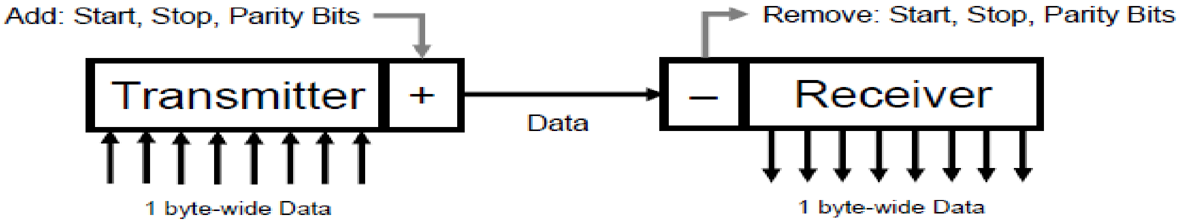


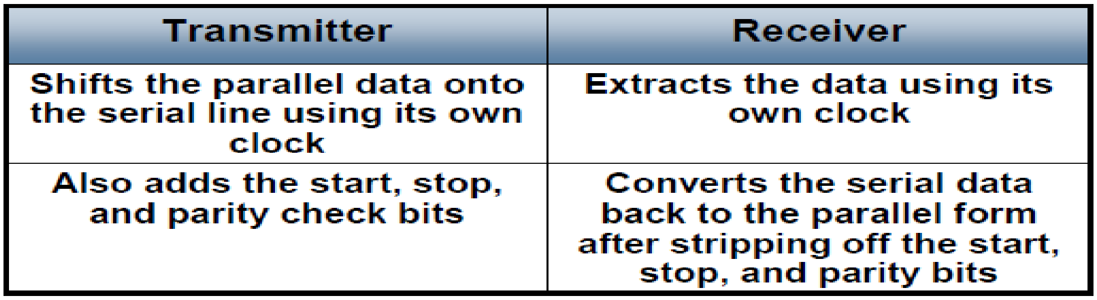
Generally it involves
Start bit—indicates the beginning of the data word
Stop bit—indicates the end of the data word
Parity bit—added for error detection (optional)
Data bits—the actual data to be transmitted

Baud rate—the bit rate of the serial port
Throughput—actual data transmitted per sec(total bits transmitted-overhead)
Example: 115200 baud = 115200 bits/sec
If using 8-bit data, 1 start, 1 stop, and no parity bits, the effective throughput is: 115200 * 8 / 10 = 92160 bits/sec



In asynchronous serial communications, data is transmitted without the clock. Therefore, it is up the CPU to synchronize its internal baud rate clock to the incoming data. In the case of RS232 the start bit is used to start the process. The start bit tells the receiver the phase relationship required for the internal clock used to latch the data bits. The rate of this clock is typically determined at design time; however, there are some applications where devices are required to determine the baud rate from the incoming data stream. The example above shows one way for devices to align their internal clock with the incoming data. Once the start bit is received then the clock system restarts its count sequence allowing for the clock edges to line up according to the bit time specified. With **asynchronous** communication, the transmitter and receiver do not share a common clock.


Add: Start, Stop, Parity Bits
Transmitter
+
Data
–
Receiver
Remove: Start, Stop, Parity Bits
1 byte-wide Data
1 byte-wide Data



Transmitter	Receiver
Shifts the parallel data onto the serial line using its own clock	Extracts the data using its own clock
Also adds the start, stop, and parity check bits	Converts the serial data back to the parallel form after stripping off the start, stop, and parity bits

Communication Standards :

There are various standards developed to communicate devices . For serial communication most widely used standard is RS 232

The RS232-C and V.24 Standards

In most computer systems, the UART (Universal Asynchronous Receiver Transmitter) is connected to circuitry that generates signals that comply with the EIA RS232-C specification. There is also a CCITT standard named V.24 that mirrors the specifications included in RS232-C.

RS-232 is defined as the “Interface between data terminal equipment and data communications equipment using serial binary data exchange.” This definition defines data terminal equipment (DTE) as the computer, while data communications equipment (DCE) is the modem. A modem cable has pin-to-pin connections, and is designed to connect a DTE device to a DCE device.

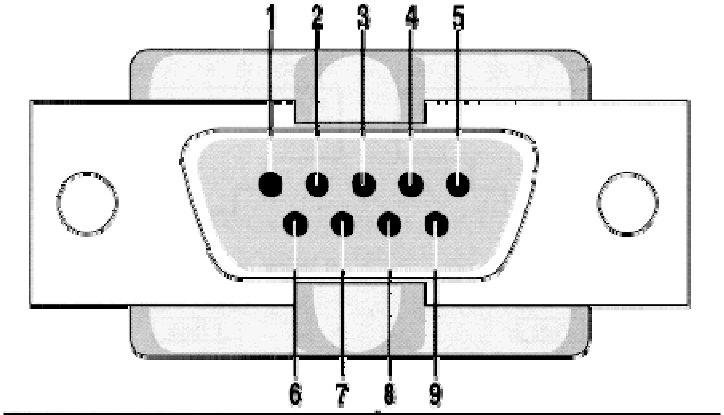
The RS232-C specification defines two types of equipment: the Data Terminal Equipment (DTE) and the Data Carrier Equipment (DCE). Usually, the DTE device is the terminal (or computer), and the DCE is a modem. Across the phone line at the other end of a conversation, the receiving modem is also a DCE device and the computer that is connected to that modem is a DTE device. The DCE device receives signals on the pins that the DTE device transmits on, and vice versa.

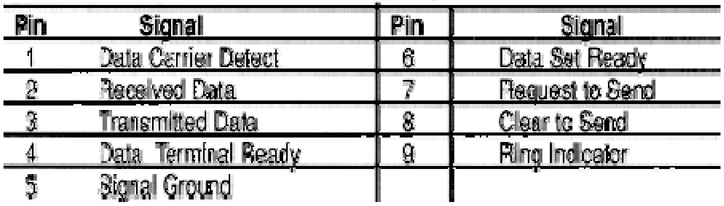
When two devices that are both DTE or both DCE must be connected together without a modem or a similar media translator between them, a NULL modem must be used. The NULL modem electrically re-arranges the cabling so that the transmitter output is connected to the receiver input on the other device, and vice versa. Similar translations are performed on all of the control signals so that each device will see what it thinks are DCE (or DTE) signals from the other device.

The number of signals generated by the DTE and DCE devices is not symmetrical. The DTE device generates fewer signals for the DCE device than the DTE device receives from the DCE

RS-232 Specifications

TRANSMITTED SIGNAL VOLTAGE LEVELS:	**RECEIVED SIGNAL VOLTAGE LEVELS:**	**DATA FORMAT:**
Binary 0: +5 to +15 Vdc (called a “space” or “on”) Binary 1: -5 to -15 Vdc (called a “mark” or “off”)	Binary 0: +3 to +13 Vdc Binary 1: -3 to -13 Vdc	Start bit: Binary 0 Data: 5, 6, 7 or 8 bits Parity: Odd, even, mark or space (not used with 8-bit data) Stop bit: Binary 1, one or two bit



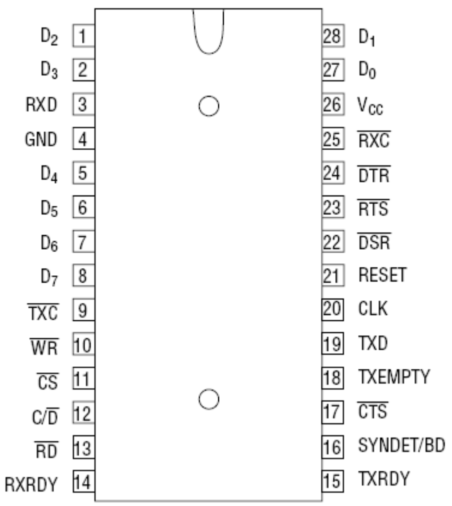


Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

8251A programmable Communication Interface

The 8251A is a USART (Universal Synchronous Asynchronous Receiver Transmitter) for serial data communication. As a peripheral device of a microcomputer system, the 8251 receives parallel data from the CPU and transmits serial data after conversion. This device also receives serial data from the outside and transmits parallel data to the CPU after conversion.

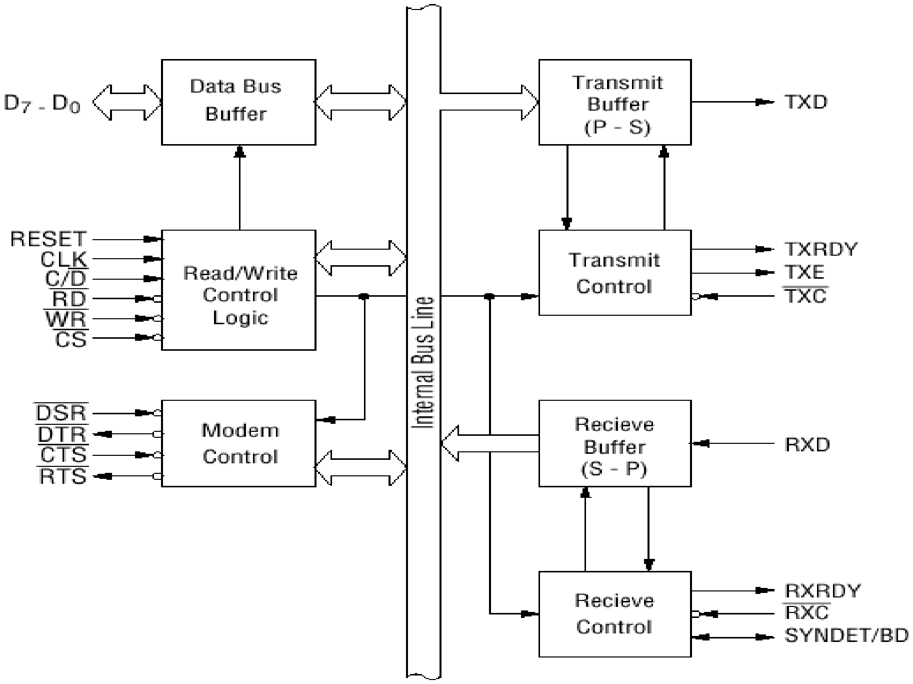
28 pin Plastic DIP



UART Functions

In addition to the basic job of converting data from parallel to serial for transmission and from serial to parallel on reception, a UART will usually provide additional circuits for signals that can be used to indicate the state of the transmission media, and to regulate the flow of data in the event that the remote device is not prepared to accept more data. For example, when the device connected to the UART is a modem, the modem may report the presence of a carrier on the phone line while the computer may be able to instruct the modem to reset itself or to not take calls by raising or lowering one more of these extra signals.

Block Diagram



Block diagram of the 8251 USART (Universal Synchronous Asynchronous Receiver Transmitter)

8251A is a programmable peripheral designed for synchronous /asynchronous serial data communication, packaged in a 28-pin DIP. It includes five sections:

1) Read/Write Control Logic
2) Modem Control
3) Data Bus buffer
4) Transmitter
5) Receiver

Read/Write Control logic

This section includes a control logic, six input signals, and three buffer registers: Data register, control register and status register. The control logic interfaces the chip with MPU, determines the functions of the chip according to the control word in the control register and monitors the data flow.

Input signals

- $\overline{\text{CS}}$ – Chip Select: When this signal goes low, the 8251A is selected by the MPU for communication.
- C/$\overline{\text{D}}$ – Control/Data: When this signal is high, the control or status register is addressed; when it is low, data buffer is addressed. The control register and status register are differentiated by $\overline{\text{WR}}$ and $\overline{\text{RD}}$ signals.
- $\overline{\text{WR}}$: When this signal is low, the MPU either writes in the control register or sends output to the data buffer
- $\overline{\text{RD}}$: When this signal goes low, the MPU either reads a status from the status register or accepts data from data buffer.
- RESET: A high on this signal reset 8252A and forces it into the idle mode.
- CLK: This is the clock input, usually connected to the system clock for communication with the microprocessor.

Control Register

The 16-bit register for a control word consist of two independent bytes. : the first byte is called mode word and the second byte is called the command word. The mode word specifies the general characteristics of operation such as baud, parity, number of bits etc. But the command word enables the data transmission and reception. The register can be accessed as an output port when the Control/Data pin is high.

Status register

This input register checks the ready status of the peripheral. The status word in the status register provides the information concerning register status and transmission errors.

Data buffer

This bidirectional register can be used as an input and output port when the C/$\overline{\text{D}}$ is low.

$\overline{\text{CS}}$	C/$\overline{\text{D}}$	$\overline{\text{WR}}$	$\overline{\text{RD}}$	Operation
0	0	1	0	MPU reads data from data buffer
0	0	0	1	MPU writes data from data buffer
0	1	0	1	MPU writes a word to control register
0	1	1	0	MPU reads a word from status register
1	×	×	×	Chip is not selected for any operation

Transmitter section

The transmitter accepts parallel data from MPU and converts them into serial data. It has two registers:

- Buffer register to hold eight bits
- Output register to convert eight bits into a stream of serial bits.

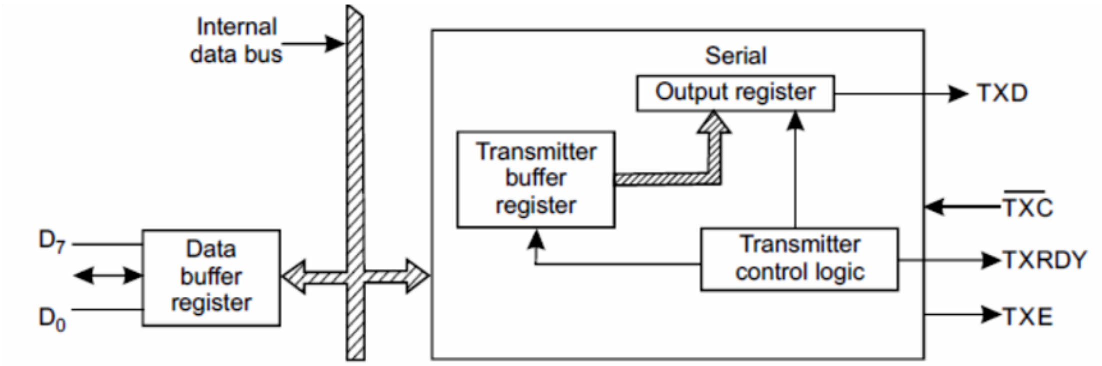
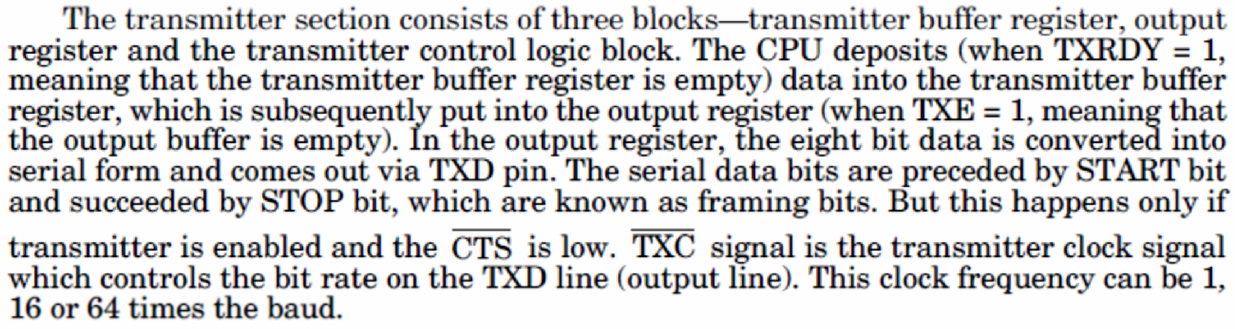


Fig Transmitter section


The transmitter section consists of three blocks—transmitter buffer register, output register and the transmitter control logic block. The CPU deposits (when TXRDY = 1, meaning that the transmitter buffer register is empty) data into the transmitter buffer register, which is subsequently put into the output register (when TXE = 1, meaning that the output buffer is empty). In the output register, the eight bit data is converted into serial form and comes out via TXD pin. The serial data bits are preceded by START bit and succeeded by STOP bit, which are known as framing bits. But this happens only if transmitter is enabled and the $\overline{\text{CTS}}$ is low. $\overline{\text{TXC}}$ signal is the transmitter clock signal which controls the bit rate on the TXD line (output line). This clock frequency can be 1, 16 or 64 times the baud.

The MPU writes a byte in the buffer register, whenever the output register is empty; the contents of buffer register are transferred to output register. And transmit the data on the TxD pin with appropriate framing bits. Three output and one input signals are associated with this transmitter section.

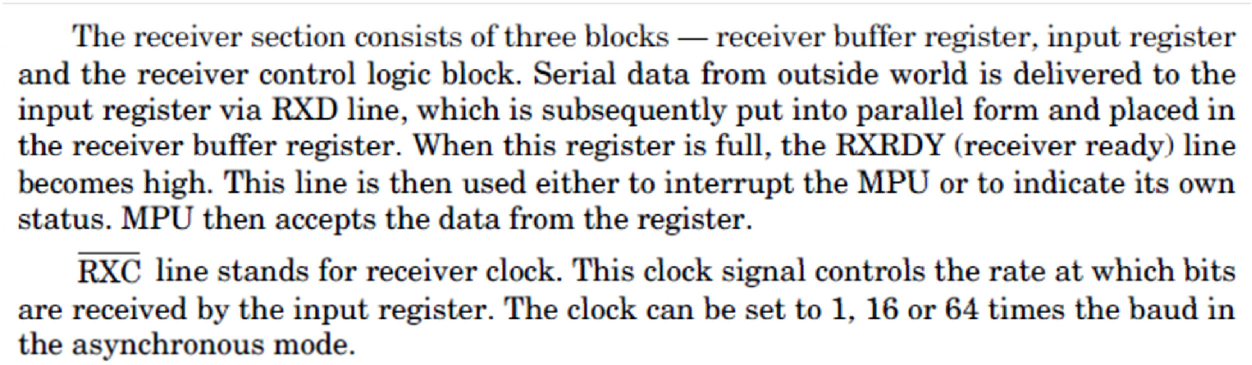
TxD: This is an output signal to transmit the data to peripherals

TxC: An in put signal controls the rate of transmission.

TxRDY: An output signal, indicate the buffer register is empty and the USART is ready to accept the next data byte.

TxE: An output signal to indicate the output register is empty and the USART is ready to accept the next data byte.

Receiver Section

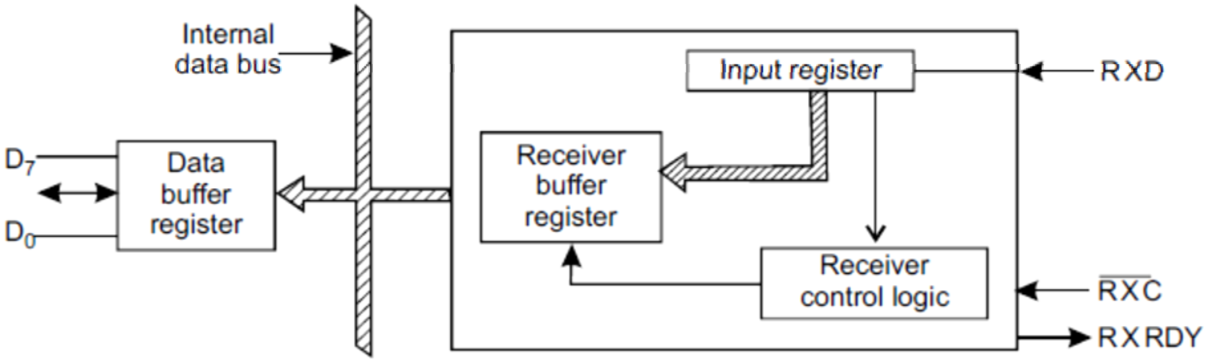


The receiver section consists of three blocks — receiver buffer register, input register and the receiver control logic block. Serial data from outside world is delivered to the input register via RXD line, which is subsequently put into parallel form and placed in the receiver buffer register. When this register is full, the RXRDY (receiver ready) line becomes high. This line is then used either to interrupt the MPU or to indicate its own status. MPU then accepts the data from the register.

$\overline{\text{RXC}}$ line stands for receiver clock. This clock signal controls the rate at which bits are received by the input register. The clock can be set to 1, 16 or 64 times the baud in the asynchronous mode.

Receiver section accepts serial data on the RxD pin and converts them to parallel data. This section has two registers:

- The receiver input register
- The buffer registers.



When the RxD goes low, the control logic assumes it is a start bit, waits for half bit time, and samples the line again. If the line is still low, the input register accepts the following data, and loads it into buffer register at the rate determined by the receiver clock.

RxRDY- receiver Ready: This is an output signal. It goes high when the USART has a character in the buffer register and is ready to transfer it to the MPU.

RxD- Receive data: Bits are received serially on this line and converted into a parallel byte in the receiver input register.

RxC- Receiver clock: This is clock signal that controls the rate at which bits are received by the USART.

Parallel Interface:

Parallel communication is a method of sending several data signals simultaneously over several parallel channels. It is accomplished by simultaneously transferring bits over separate lines Parallel communications are used over very short distances; typically inside the computer itself and to printers. This method, together with the connector, was developed by Centronics and used by IBM in its first PC. The parallel port greatly increases transfer speeds by using an eight wire connector which transmits the eight bits in a byte of data simultaneously, thus sending an entire byte of data in the time it takes to send a single bit in a serial system. This byte of data is supplemented by several other handshaking signals, each sent on its own wire, which ensures that data transfer takes place smoothly.

METHODS OF PARALLEL DATA TRANSFER

1. SIMPLE INPUT AND OUTPUT

- When you need to get digital data from a simple switch, all you have to do is connect the switch to an I/P port line and read the value.
- Likewise, when you need to O/P data to simple LED, all you have to do is connect the LED to an O/P port and send the value.
- The LED is always ready, so you can send data at any time.

2. SIMPLE STROBE I/O

- In many applications, valid data is present on an external device only at a certain time, so it must be read in at that time.
- When a key is pressed, circuitry on the keyboard sends out the ASCII code for the pressed key on eight parallel data line , and then sends out a strobe signal on another line to indicate that valid data is present on the eight data lines.
- For higher speed data transfer this method does not work.
- The sending system might send data bytes faster than the receiving system could read them. To prevent this handshake data transfer is required.

3. SINGLE HANDSHAKE I/O DATA TRANSFER

- The peripheral outputs some parallel data and sends STB signal to MPU.
- The MPU detects STB signal on a polled or interrupt basis and reads data byte.
- Then the MP sends on ACK signal to the peripheral to indicate that the data has been read and the peripheral can send to next byte of data.

4. DOUBLE HANDSHAKE I/O DATA TRANSFER

- For data transfers where coordination is required between sending system and the receiving system, a double handshake is used.
- The sending device asserts its STB low to ask “**Are you ready?”**
- The receiving system raises its ACK line high to say “**I’m ready”.**
- The peripheral device then sends the data byte and raises its STB signal high to say **“Here is valid data for you”.**
- When the receiving system finishes to read the data, receiving system drop its ACK line low to say **“I have data than you and I await your request to send the next byte of data”.**

8255A PROGRAMMABLE PERIPHERAL INTERFACE

INTRODUCTION

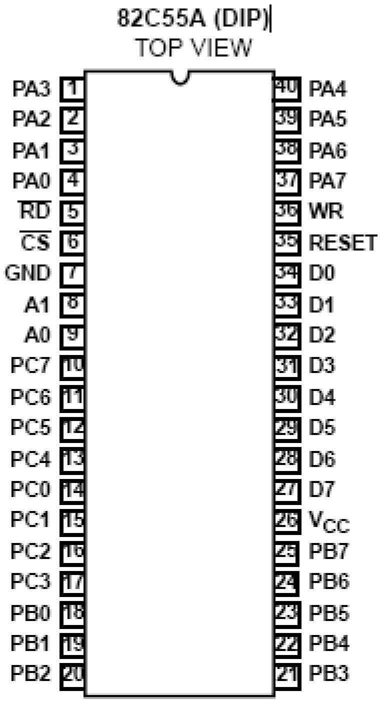
- The 8255A PPI provides three 8 bit input/output ports in one 40 pin package.
- The chip can be interfaced directly to the data bus of the processor allowing its function to be programmed.
- In one application, a port may appear as an output, but in another by reprogramming it an be as input.

DESCRIPTION

- The 8255A is a general purpose parallel I/O interfacing device.
- It provides 24 I/O lines organized as three 8 bit input/output ports labeled A,B and C.
- Each of the ports A or B can be programmed as an 8 bit input or output port.
- Port C can be divided in half, with the topmost or bottommost four bits programmed as inputs or outputs.
- Port C can be used as an 8 bit input or output port as two 4 bit ports or to produce handshake signals for Port A and B.
- The 8255A is a very versatile device.
- It can be programmed to look like
  - Three simple I/O ports (called MODE 0)
  - Two handshaking I/O ports (called MODE 1)
  - Port A as a bidirectional I/O port with 5 handshaking signals (called MODE 2)

- The modes can also be intermixed.
- For example, Port A can be programmed to operate in MODE 2 while Port B in MODE 0.

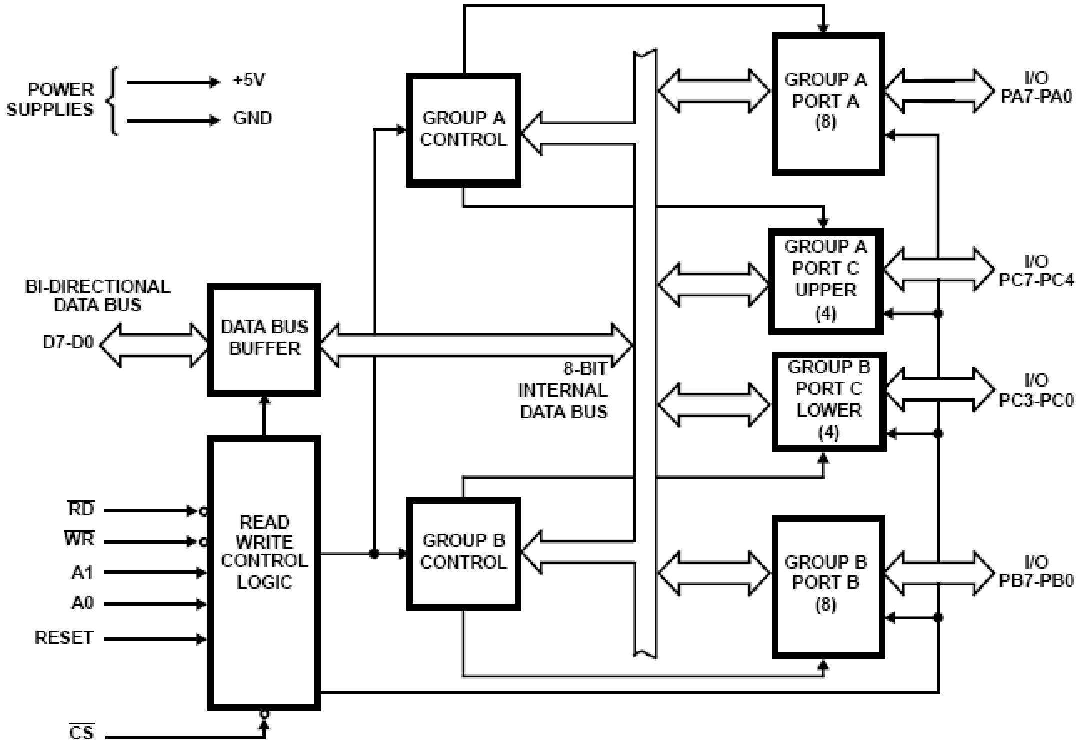
PIN CONFIGURATION OF 8255 PPI



PIN NAMES WITH DESCRIPTIONS

D7-D0	DATA BUS [BIDIRECTIONAL]
RESET	RESET INPUT
CS	CHIP SELECT
RD	READ INPUT
WR	WRITE INPUT
A0,A1	PORT ADDRESS
PA7-PA0	PORT A
PB7-PB0	PORT B
PC7-PC0	PORT C
VCC	+5V
GND	GROUND

Internal Block Diagram of 8255 PPI



- The address inputs A0, A1 allow you to selectively access one of the three ports or the control register.

- The internal addresses for the devices are tabulated below.

DESCRIPTION	A1	A0
PORT A	0	0
PORT B	0	1
PORT C	1	0
CONTROL	1	1

- The CS input of 8255A enables it for reading or writing

- This input is driven by an address decoder.

- The RD and WR input pins determine the direction of data flow over the chips 8 bit bidirectional data bus.

- The RESET input of 8255A is connected to the system reset line so that, when the system is reset all the port lines are initialized as input lines. This is done to prevent destruction of circuitry connected to port lines.

TRUTH TABLE FOR THE 8255A PPI

1. INPUT OPERATION

A1	A0	RD	WR	CS	INPUT OPERATION
0	0	0	1	0	PORT A->DATA BUS
0	1	0	1	0	PORT B->DATA BUS
1	0	0	1	0	PORT C->DATA BUS

2. OUTPUT OPERATION

A1	A0	RD	WR	CS	INPUT OPERATION
0	0	1	0	0	DATA BUS->PORT A
0	1	1	0	0	DATA BUS->PORT B
1	0	1	0	0	DATA BUS->PORT C
1	1	1	0	0	DATA BUS ->CONTROL

3. DISABLE FUNCTION

A1	A0	RD	WR	CS	INPUT OPERATION
X	X	X	X	1	DATA BUS TRISTATE
1	1	0	1	0	ILLEGAL CONDITION
X	X	1	1	0	DATA BUS TRISTATE

8255A OPERATIONAL MODES AND INITIALIZATION

1. MODE 0

- When programmed for MODE 0, the PPI offers three simple I/O ports with no handshaking signals.

- This mode is appropriate for I/O devices that do not need special synchronizing signals to exchange data with the processor.

- A common example is a keyboard used for data entry.

- When used as O/Ps, the PORT c lines can be individually set or reset by sending a special control word to control register address.

- Two halves of PORT C are independent so one half can be initialized as input and the other half as output.

2. MODE 1

- When programmed for MODE 1, the PPI offers PORT A or PORT B for a handshake input/output operation.

- Pins PC0,PC1 and PC2 function as handshake lines for PORT B

- Pins PC3,PC4 and PC5 function as handshake signal for PORT A (input).

- Pins PC6 and PC7 are available for use as input/output lines for PORT A

- If PORT A is initialized as handshake O/P port, then PORT C pins.
- PC3, PC6 and PC7 function as handshake signals.
- PORT C pins PC4 and PC5 are used as input/output lines for PORT A

3. MODE 2

- Only PORT A can be initialized in MODE 2.
- In MODE 2, PORT A can be used as bidirectional handshake data transfer.
- This means that data can be outputed/inputed from same eight lines.
- If PORT A is initialized in MODE 2, then pins PC3 through PC7 are used as handshake lines for PORT A.
- The other three pins PC0 through PC2 can be used for I/O port if PORT B is in MODE 0.
- The same 3 pins will be used for PORT B handshake signals if PORT B is initialized in MODE 1.

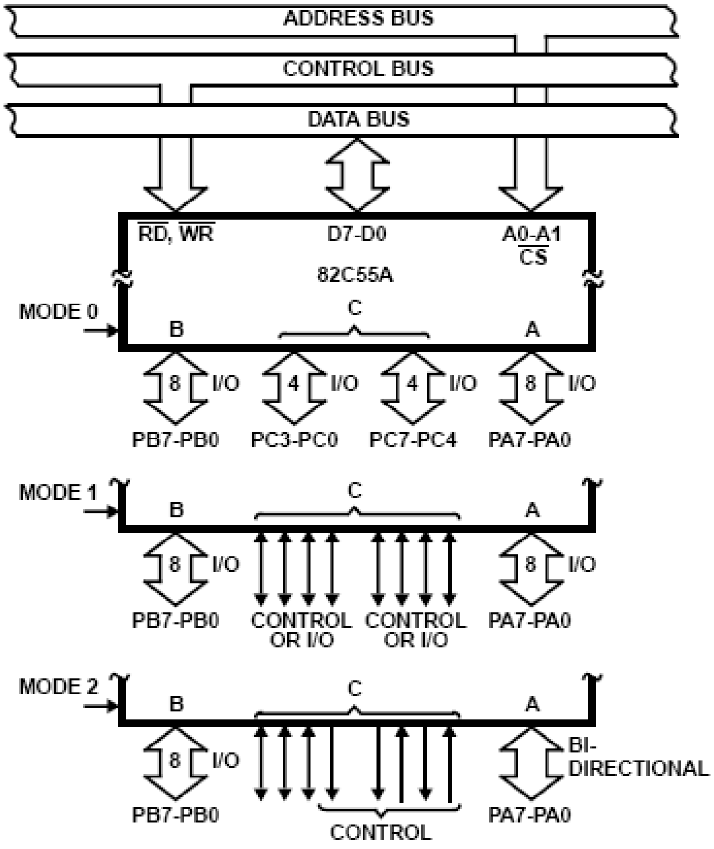
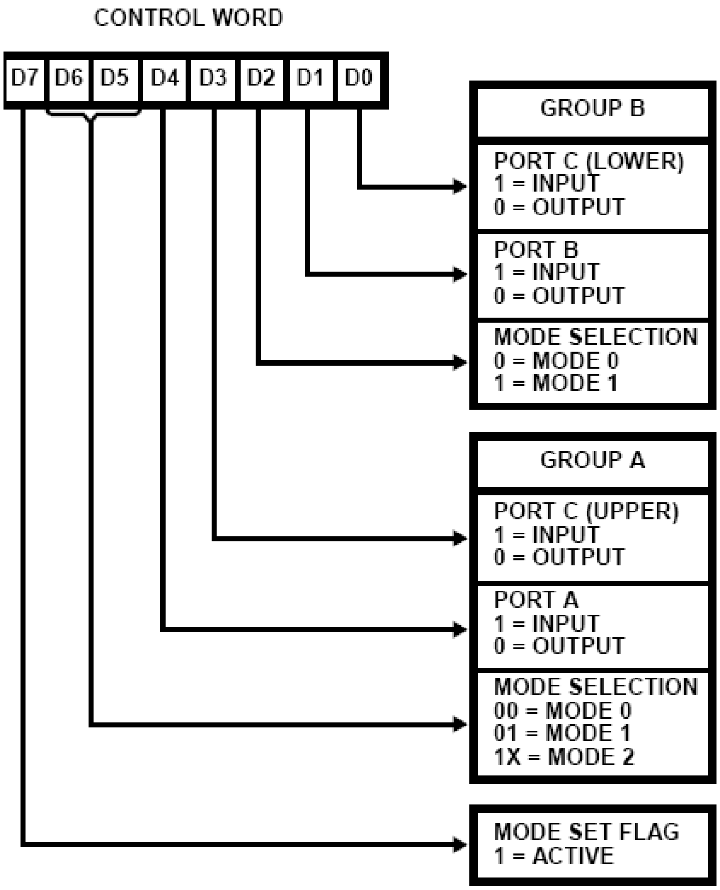
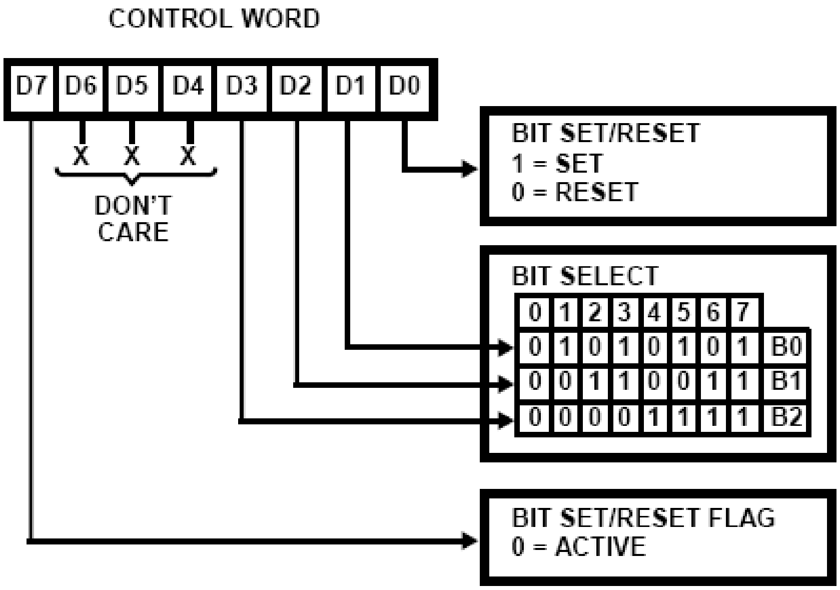


FIGURE 3. BASIC MODE DEFINITIONS AND BUS INTERFACE



Mode set control Word



0	1	2	3	4	5	6	7	
0	1	0	1	0	1	0	1	B0
0	0	1	1	0	0	1	1	B1
0	0	0	0	1	1	1	1	B2

FIGURE 5. BIT SET/RESET FORMAT

Port C Set Reset Control Word

PROBLEM 1

Write the 80x86 initialization routine required to program the 8255A for mode 0, with PORT A as an output port and PORT B and C as an input ports.

SOLUTION

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	1	0	1	1

D7 **:** 1 -> Mode set
D6 D5 **:** 00 -> Mode 0
D4 **:** 0 -> Port A Output
D3 **:** 1 -> Port C Upper Input
D2 **:** 0 -> Mode 0
D1 **:** 1 -> Port B Input
D0 **:** 0 -> Port C Lower Input

Assume the port address of control port is 0FFH

```
MOV AL,8BH
OUT 0FFH,AL
```

PROBLEM 2

Write an 80x86 program to input a byte from PORT B of PPI chip and output this byte to PORT A of the same chip. Assume it is already initialized.

SOLUTION

Assume the port address of control port is 0FDH
Assume the port address of PORT B as 0FDHH

```
IN AL,0FDH      ;Read from PORT B
OUT 0FCH,AL   ; Write to PORT B
```

PROBLEM 3

Write instruction to initialize 8255 to configure PORT A as simple output port, PORT B as simple input port, PORT C upper as output and port c lower as an input port.

PROBLEM 4

Write a program to take input from the 8 switches connected to PORT B and display the status of the switches thus red in the 8 LEDS connected to PORT A. Show how you derive the control word.

PROBLEM 5

Write instruction to initialize 8255 to configure PORT A and display the status of the switches thus read in the 8 LEDS connected to PORT B. show how do you derive the control word.

Timers and Event Counters

A device is sometime necessary to mark intervals of time for both the processor and external devices , count external events and make the count available to the processor , and provide external timing that can programmed from the processor . This device is called a programmable interval timer / event counter .

The 8254 is a programmable interval timer/counter designed for use with Intel microcomputer systems.It is a general purpose, multi-timing element that can be treated as an array of I/O ports in the system software. The 8254 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control. Instead of setting up timing loops in software, the programmer configures the 8254 to match his requirements and programs one of the counters for the desired delay. After the desired delay, the 8254 will interrupt the CPU. Software overhead is minimal and variable length delays can easily be accommodated.
Some of the other counter/timer functions common to microcomputers which can be implemented with the 8254 are:
# Real time clock
#Measure time delay between events
# Even counter
# Programmable baud rate generator
# Square wave generator
# Binary rate multiplier
# Waveform generator for A/D converter
# Complex motor controller

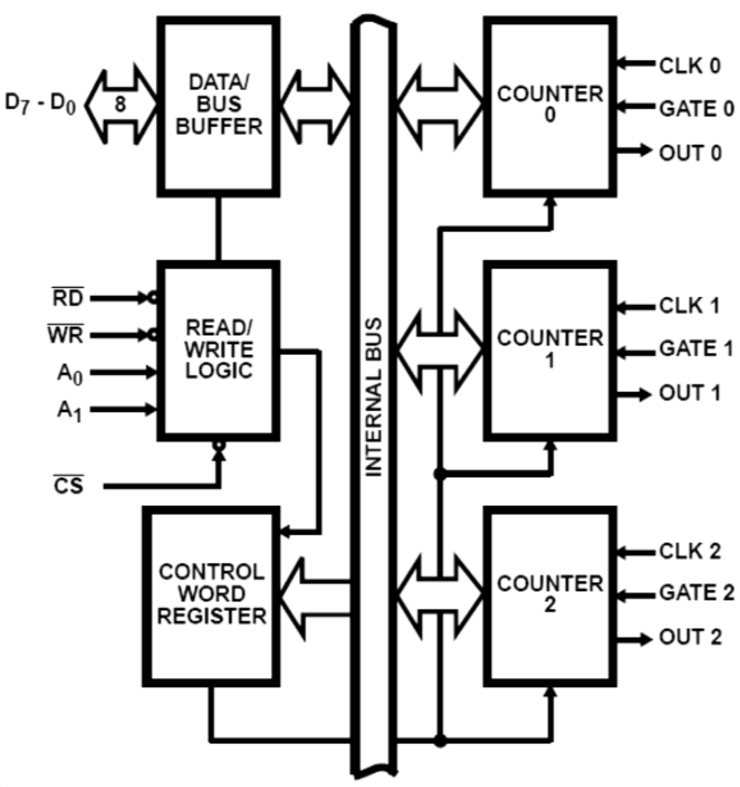
BLOCK DIAGRAM

DATA BUS BUFFER
This 3-state, bi-directional, 8-bit buffer is used to interface the 8254 to the system bus.

READ/WRITE LOGIC
The Read/Write Logic accepts inputs from the system bus and generates control signals for the other functional blocks of the 8254. A1 and A0 select one of the three counters or the Control Word Register to be read from/written into. A ``low'' on the RD input tells the 8254 that the CPU is reading one of the counters. A ``low'' on the WR input tells the 8254 that the CPU is writing either a Control Word or an initial count. Both RD and WR are qualified by CS; RD and WR are ignored unless the 8254 has been selected by holding CS low. The WRÝ and CLK signals should be synchronous. This is accomplished by using a CLK input signal to the 8254 counters which is a derivative of the system clock source. Another technique is to externally synchronize the WRY and CLK input signals. This is done by gating WRÝ with CLK.

CONTROL WORD REGISTER

The Control Word Register is selected by the Read/Write Logic when A1, A0 e 11. If the CPU then does a write operation to the 82C54, the data is stored in the Control Word Register and is interpreted as a Control Word used to define the operation of the Counters. The Control Word Register can only be written to;status information is available with the Read-Back Command.



COUNTERS (0 , 1 and 2):

These three functional blocks are identical in operation, so only a single Counter will be described.The Counters are fully independent. Each Counter may operate in a different Mode. The Control Word Register is shown in the figure; it is not part of the Counter itself, but its contents determine how the Counter operates.

GENERAL OPERATION

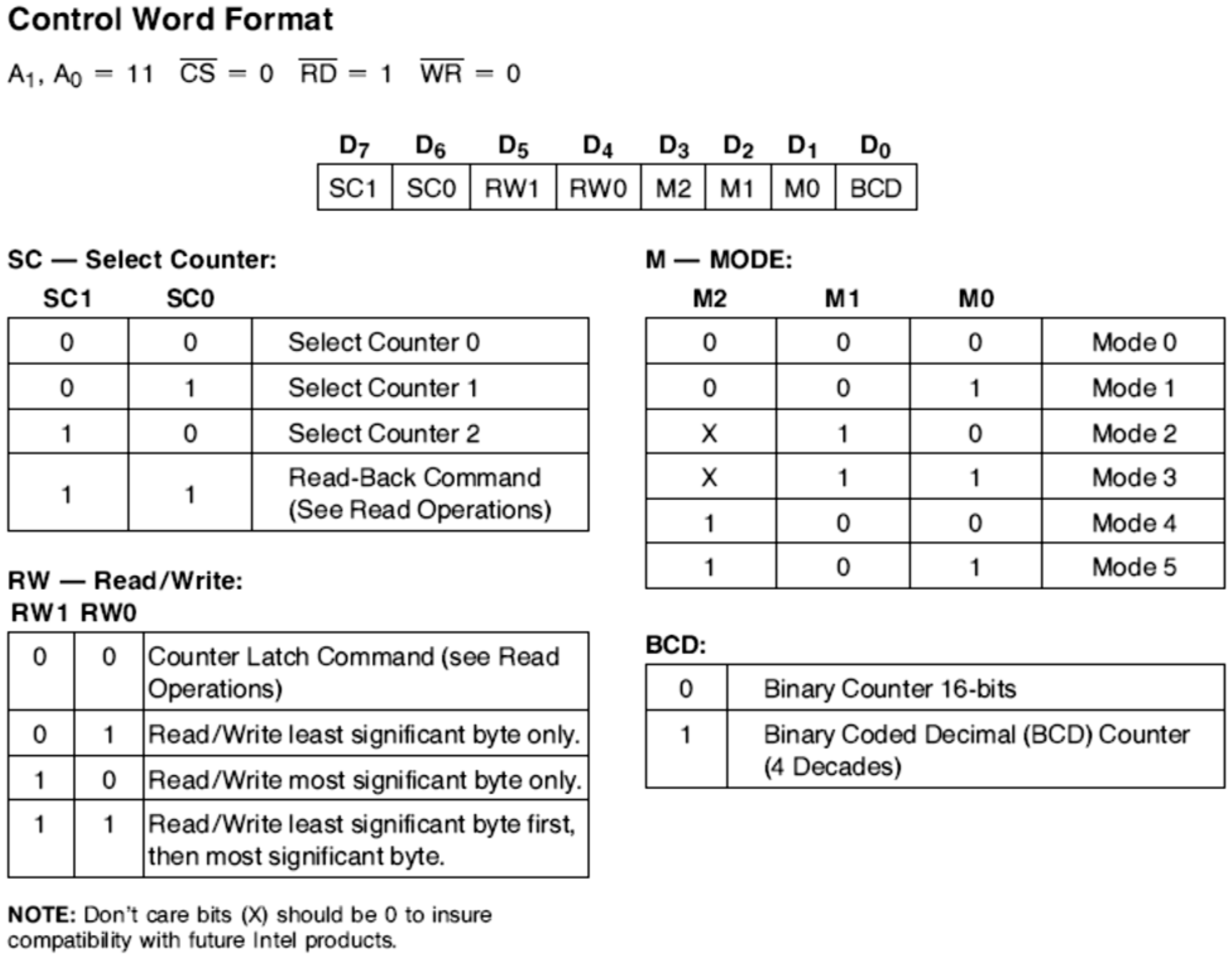
After power-up, the state of the 82C54 is undefined. The Mode, count value, and output of all Counters are undefined. How each Counter operates is determined when it is programmed. Each Counter must be programmed before it can be used. Unused counters need not be programmed.

Counters are programmed by writing a Control Word and then an initial count.

The programming procedure for the 82C54 is very flexible. Only two conventions need to be remembered:
1) For each Counter, the Control Word must be written before the initial count is written.
2) The initial count must follow the count format specified in the Control Word (least significant byte only, most significant byte only, or least significant byte and then most significant byte).

Since the Control Word Register and the three Counters have separate addresses (selected by the A1, A0 inputs), and each Control Word specifies the Counter it applies to (SC0, SC1 bits), no special instruction sequence is required. Any programming sequence that follows the conventions above is acceptable. A new initial count may be written to a Counter at any time without affecting the Counter's programmed Mode in any way. Counting will be affected as described in the Mode definitions. The new count must follow the programmed count format. If a Counter is programmed to read/write two-byte counts, the following precaution applies: A program must not transfer control between writing the first and second byte to another routine which also writes into that same Counter. Otherwise, the Counter will be loaded with an incorrect count.

CONTROL WORD FORMAT



Control Word Format

$A_1, A_0 = 11$ $\overline{CS} = 0$ $\overline{RD} = 1$ $\overline{WR} = 0$

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC — Select Counter:

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Read-Back Command (See Read Operations)

M — MODE:

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

RW — Read/Write:

RW1	RW0	
0	0	Counter Latch Command (see Read Operations)
0	1	Read/Write least significant byte only.
1	0	Read/Write most significant byte only.
1	1	Read/Write least significant byte first, then most significant byte.

BCD:

0	Binary Counter 16-bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

NOTE: Don't care bits (X) should be 0 to insure compatibility with future Intel products.

D0

 Binary number divisor 0000-FFFFH (D0 =0)

• Divisor = 65536 if the counter is loaded with 0 for both the low and the high bytes.

 BCD divisor 0000-9999H (D0=1)

• Divisor = 10000 if the counter is loaded with 0 for both the low and the high bytes.

 D1, D2, and D3: Mode selection

 Mode 0: Interrupt on terminal count

 Mode 1: Programmable one-shot

 Mode 2: Rate generator

 Mode 3: Square wave rate generator

 Mode 4: Software triggered strobe

 Mode 5: Hardware triggered strobe

- Read/write the most significant byte (MSB) only
- Read/write the LSB first then followed by the MSB

- We can write the value of the divisor into 8253/54 timer and read the contents of the counter at any time.
- D6 and D7 are used to select which of the three counters: counter 0 (00), counter 1 (01), and counter 2 (10).

Mode 0: Interrupt on terminal count

- It is used to generate an interrupt to the microprocessor after a certain interval of time.
- The output is initially low after the mode is set. The output remains LOW after the count value is loaded in the counter.
- The process of decrementing the counter continues till the terminal count is reached, i.e., the count become zero and the output goes HIGH. The output remains high until it reloads a new mode of operation or new count.
- The GATE signal is high for normal counting. When GATE goes low counting is terminated and the current count is latched till the GATE goes high again.

Mode 1: Programmable one-shot

- The 8253/54 can be used as a monostable multivibrator.
- The gate input is used as trigger input in this mode. Normally,the output remains high until the count is loaded and a trigger is applied.
- The duration of the quasistable of the monostable multivibrator is decided by the count loaded in the count register.

Mode 2: Rate generator

- Divide by N counter.
- The output is normally high after initialization.
- If N is loaded as the count value, after N pulses, the output becomes low for one clock cycle.
- Whenever the count becomes zero another low pulse is generated at the output.

Mode 3: Square wave rate generator

- It is similar to mode 2.
- When, the count N loaded is EVEN, half of the count will be high and half of the count will be low.
- When, the count N loaded is ODD, the first clock pulse decrements it by 1. Then half of the remaining count will be high and half of the remaining count will be low.

Mode 4: Software triggered strobe

- After the mode is set, the output goes high.
- The counter automatically begins to decrement (count down) one clock pulse after it is loaded with the initial value through software.

 When the GATE signal goes low the count is latched.
 On the terminal count, the output goes low for one clock cycle, and then again goes high. This low pulse can be used as a strobe.

Mode 5: Hardware triggered strobe
 This mode generates a strobe in response to an externally generated signal.
 It is similar to mode 4 except that the counting is initiated by a signal at the gate input, i.e., it is hardware triggered instead of software triggered.
 After it is initialized, the output goes high.
 The counter starts counting after the rising edge of the trigger input (GATE).
 When the terminal count is reached, the output goes low for one clock cycle.

Example 1
 Pin CS of a given 8253/54 is activated by binary address
A7-A2 = 100101.
 Find the port addresses assigned to this 8253/54.
Counter 0 Port address = 1001 01 00 = 94H
Counter 1 Port address = 1001 01 01 = 95H
Counter 2 Port address = 1001 01 10 = 96H
Control register Port address = 1001 01 11 = 97H
 Find the configuration for this 8253/54 if the control register is
programmed as follows.

```
MOV AL, 00110110
OUT 97H, AL
```

D7-D6 = 00 select counter 0
D5-D4 = 11 the low byte read/write is followed by the high byte.
D3-D1 = 011 select mode 3 (square wave)
D0 = 0 select the binary counting

Example 2
 Use the port addresses in Ex. 1 to program counter 2 for binary count of mode 3 (square wave) to divide CLK2 by number C26AH and find the frequency of OUT2 if CLK2 = 1.8 MHz.

 The control word = 10110110

```
MOV AL, B6H ; counter 2, mode 3, binary
OUT 97H, AL ; send it to control register
MOV AX,C26AH ; load the divisor
OUT 96H, AL ; send the low byte to counter 2
MOV AL, AH
OUT 96H, AL ; send the high byte to counter 2
```

 The output frequency for OUT2 = 1.8 MHz is divided by C26AH (49770 D). OUT 2 frequency is a square wave of 36 Hz.

Example 3

 Using the port address in Ex.1, show the programming of counter 1 to divide CLK1 by 10,000, producing the mode 3 square wave. Use the BCD option in the control byte.

```
MOV AL, 77H ; counter 1, mode 3, BCD
OUT 97H, AL ; send it to control register
SUB AL, AL ; AL = 0 load the divisor for 10,000
OUT 95H, AL ; send the low byte
OUT 95H, AL ; send the high byte
```

Direct Memory Access (DMA)

The transfer of data between a fast storage device such as magnetic disk and memory is often limited by the speed of the CPU. Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer . This transfer technique is called direct memory access(DMA).

DMA is used in large blocks of data transfer at a high speed to or from high speed devices, magnetic drums, disks, tapes, etc. We use DMA controller which is interface that provides I/O transfer of data directly to and from the memory and the I/O device. CPU initializes the DMA controller by sending a memory address and the number of words to be transferred.
Actual transfer of data is done directly between the device and memory through DMA controller freeing CPU for other tasks.

The 8237A Multimode Direct Memory Access (DMA) Controller is a peripheral interface circuit for microprocessor systems. It is designed to improve system performance by allowing external devices to directly transfer information from the system memory. Memory-to-memory transfer capability is also provided. The 8237A offers a wide variety of programmable control features to enhance data throughput and system optimization and to allow dynamic reconfiguration under program control.

A DMA controller is designed to service one or more I/O or mass storage interfaces and each interface is connected to the controller by a set of conductors. A portion of a DMA controller for servicing a single interface is called a channel.
Each channel will have

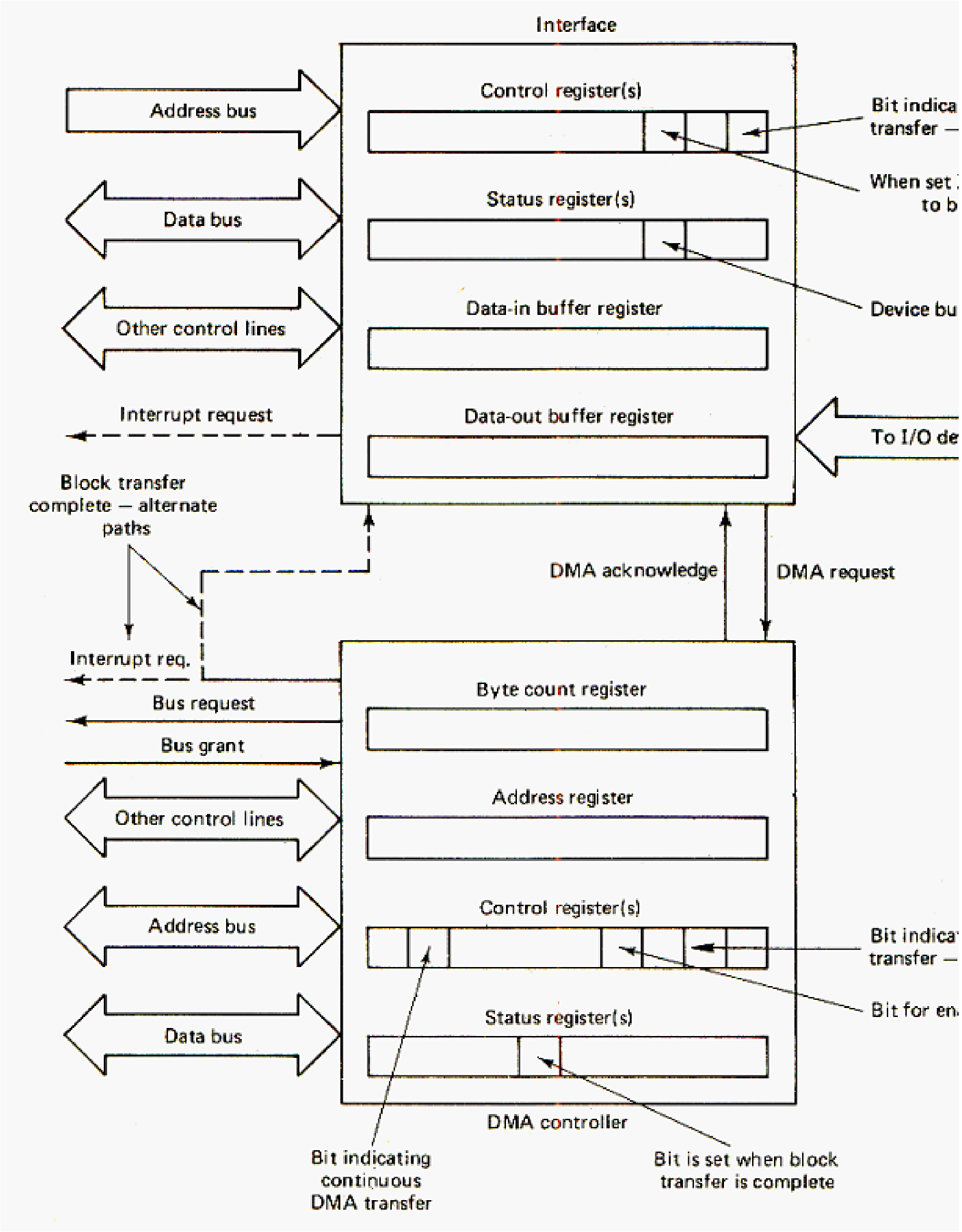
a) Control Register
b) Status Register
c) Address Register
d) Byte Count Register

When data are being put in or taken out from its control registers by the CPU,it is acting a s slave.When it is in the control of the bus , it is acting as the master.

1. Initilalizing the controller consists of filling these registers with the beginning address of the memory address that is used as a buffer and number of bytes to be transferred.
2. For an input to memory, each time the interface has the data to transfer it makes a DMA request through DREQ PIN of the DMA chip.
3. The controller then makes a bus request through HOLD pin of DMA .
4. The CPU after completing the current instruction, relinquishes the control of the system bus and send the DMA the bus grant signal through the HLDA pin of DMA.
5. After receiving a bus grant , it puts the contents of the address register on the address bus, sends an acknowledge back to the interface of the device who has issued the memory operation throught the DACK pin of the DMA.
6. It then issues the I/O read and memory write signals. The interface then puts the data on the data bus and drops its request.
7. When the memory accepts the data it returns a ready signal to the controller, which then increments the address register
8. It then decrements the byte count
9. Then it drops its bus request
10. Upon the count reaching zero, the process stops and a signal is sent to the processor as an interrupt requestor to the interface to notify it that the transfers have terminated.
11. An output is similarly executed ,except that controller issues I/O write and memory read signals and the data are transferred in the other direction

The 8237 includes Control and status register and four channels. Each channel contains a

1) mode register
2) current address register
3) base address register
4) current byte counter
5) base byte counter
6) request flag and
7) mask flag



Block diagram of 8237

Bit 5 of the mode register specifies whether the contents of the address register are to be incremented(0)or decremented(1) after each data transfer , thus determining the order in which the data is stored in the memory.
If bit 4 is 1 , then autoinitialization is enabled.When the current address and current byte count registers are initially loaded, their contents are also put in the base address and base byte count registers. If autoinitialization is enabled, the current registers are automatically reloaded from the base registers whenever the count goes to zero.

Each channel may be put in the one of the four modes with its current mode being determined by bits 7 and 6 of the channel's mode register. The four possible modes are

1) Single Transfer mode or cycle stealing mode (01)

Afer each transfer the controller will release the bus to the processor for at least one bus cycle, but will immediately begin testing for DREQ inputs and proceed to steal another cycle as soon as a DREQ line becomes active.

Cycle Stealing

While DMA I/O takes place, CPU is also executing instructions .When DMA Controller and CPU both access Memory, Memory Access Conflict arises. This is managed by Memory Bus controller. It coordinates the activities of all devices requesting memory access using priority system. Memory accesses by CPU and DMA Controller are interwoven, with the top priority given to DMA Controller. Cycle stealing is the mode in which DMA controller take over the bus for each byte of data to be transferred and then return control to the CPU. Since the processor originates most memory access cycles, it is often stated that DMA steals memory cycles from the processor. CPU is usually much faster than I/O(DMA), thus CPU uses the most of the memory cycles, DMA Controller steals the memory cycles from CPU.For those stolen cycles, CPU remains idle.

2) Block Transfer mode or burst transfer mode (10)

The bus control is not released by the DMA until the entire block of data has been transferred.

In Block Transfer mode the device is activated by DREQ to continue making
transfers during the service until a TC, caused by word count going to FFFFH, or an external End of Process (EOP) is encountered. DREQ need only be held active until DACK becomes active. Again, an Autoinitialization will occur at the end of the service if the channel has been programmed for it.

3) Demand Transfer Mode (00)

This mode is similar to the block mode except that DREQ is tested after each transfer.If DREQ is inactive, transfers are suspended until DREQ once again becomes active, at which time the block transfer continues from the point at which it was suspended. This allows the interface to stop the transfer in the event that its device cannot keep up. This is due to memry latency or slow speed of the interface.

Demand Transfer mode the device is programmed to continue making transfers until a TC or external EOP is encountered or until DREQ goes inactive. Thus transfers may continue until the I/O device has exhausted its data capacity. After the I/O device has had a chance to
catch up, the DMA service is re-established by means of a DREQ. During the time between services when the microprocessor is allowed to operate, the intermediate values of address and word count are stored in the 8237A Current Address and Current

Word Count registers. Only an EOP can cause an Autoinitialize at the end of the service. EOP is generated either by TC or by an external signal. DREQ has to be low before S4 to prevent another Transfer.

4) Cascade Mode(11)

In this mode 8237's are cascaded. This mode is used to cascade more than one 8237A together for simple system expansion. The HRQ and HLDA signals from the additional 8237A are connected to the DREQ and DACK signals of a channel of the initial 8237A. This allows the DMA requests of the additional device to propagate through the priority network circuitry of the preceding device. The priority chain is preserved and the new device must wait for its turn to acknowledge requests. Since the cascade channel of the initial 8237A is used only for prioritizing the additional device, it does not output any address or control signals of its own. These could conflict with the outputs of the active channel in the added device. The 8237A will respond to DREQ and DACK but all other outputs except HRQ will be disabled. The ready input is ignored.

KEYBOARD/DISPLAY CONTROLLER - INTEL 8279

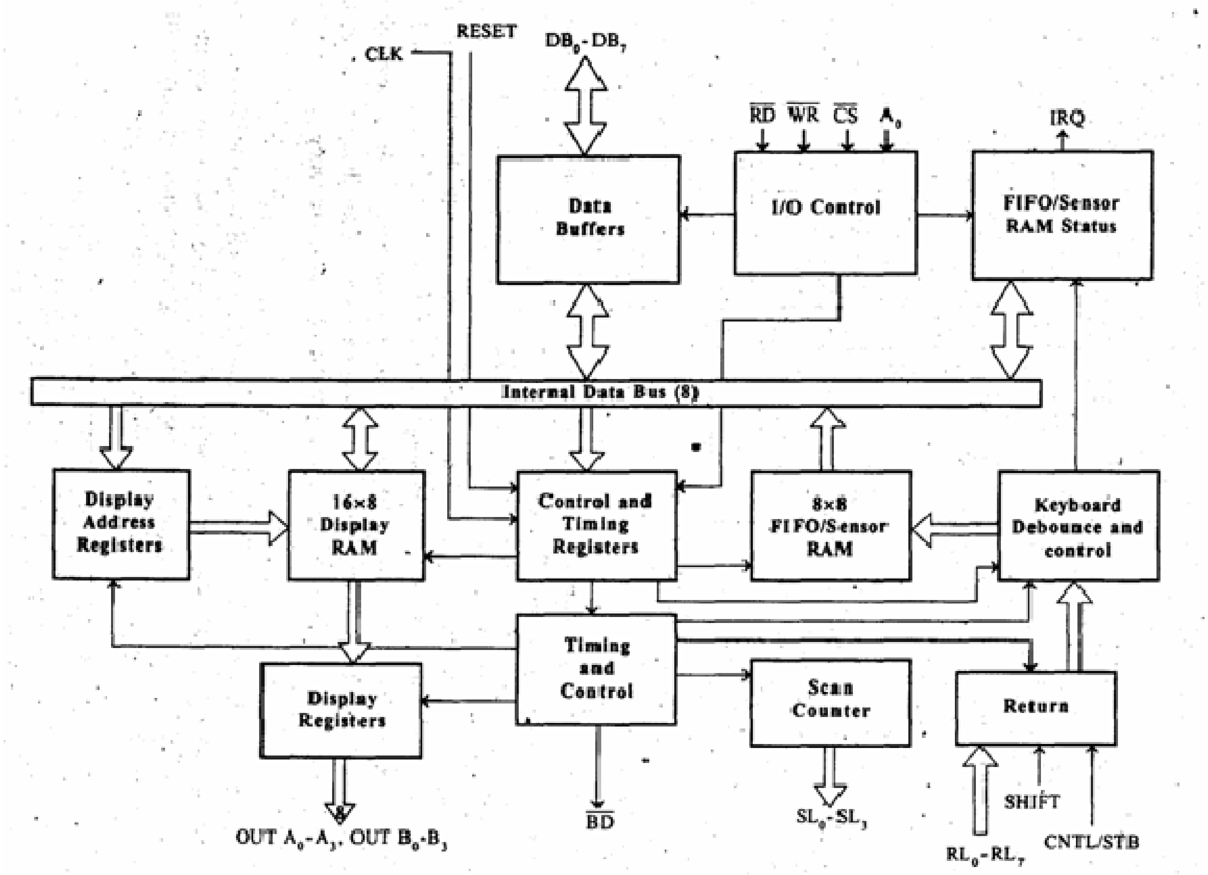
The INTEL 8279 is specially developed for interfacing keyboard and display devices to 8085/8086/8088 microprocessor based system. The important features of 8279 are,

- Simultaneous keyboard and display operations.
- Scanned keyboard mode.
- Scanned sensor mode.
- 8-character keyboard FIFO.
- 1 6-character display.
- Right or left entry 1 6-byte display RAM.
- Programmable scan timing.

Keyboard section:

- The keyboard section consists of eight return lines RL0 - RL7 that can be used to form the columns of a keyboard matrix.
- It has two additional input : shift and control/strobe. The keys are automatically debounced.
- The two operating modes of keyboard section are 2-key lockout and N-key rollover.
- In the 2-key lockout mode, if two keys are pressed simultaneously, only the first key is recognized.
- In the N-key rollover mode simultaneous keys are recognized and their codes are stored in FIFO.

- The keyboard section also have an 8 x 8 FIFO (First In First Out) RAM.
- The FIFO can store eight key codes in the scan keyboard mode. The status of the shift key and control key are also stored along with key code. The 8279 generate an interrupt signal when there is an entry in FIFO. The format of key code entry in FIFO for scan keyboard mode is,

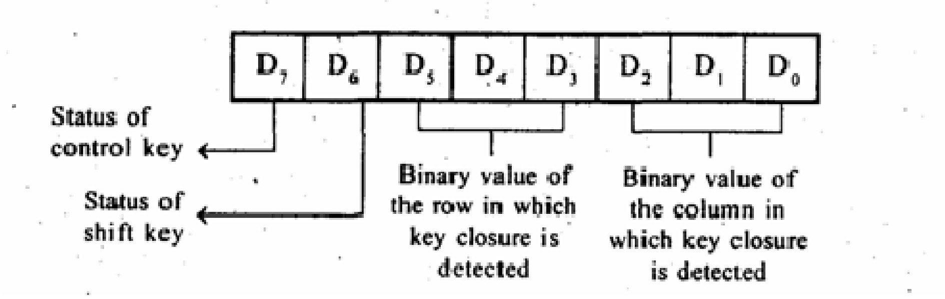


Functional block diagram

KEYBOARD AND DISPLAY INTERFACE USING 8279

Block diagram of 8279:

- The functional block diagram of 8279 is shown.


D7
D6
D5
D4
D3
D2
D1
D0
Status of control key
Status of shift key
Binary value of the row in which key closure is detected
Binary value of the column in which key closure is detected

- In sensor matrix mode the condition (i.e., open/close status) of 64 switches is stored in FIFO RAM. If the condition of any of the switches changes then the 8279 asserts IRQ as high to interrupt the processor.

Display section:

- The display section has eight output lines divided into two groups A0-A3 and B0-B3.
- The output lines can be used either as a single group of eight lines or as two groups of four lines, in conjunction with the scan lines for a multiplexed display.
- The output lines are connected to the anodes through driver transistor in case of common cathode 7-segment LEDs.
- The cathodes are connected to scan lines through driver transistors.
- The display can be blanked by BD (low) line.
- The display section consists of 16 x 8 display RAM. The CPU can read from or write into any location of the display RAM.

Scan section:

- The scan section has a scan counter and four scan lines, SL0 to SL3.
- In decoded scan mode, the output of scan lines will be similar to a 2-to-4 decoder.
- In encoded scan mode, the output of scan lines will be binary count, and so an external decoder should be used to convert the binary count to decoded output.
- The scan lines are common for keyboard and display.
- The scan lines are used to form the rows of a matrix keyboard and also connected to digit drivers of a multiplexed display, to turn ON/OFF.

CPU interface section:

- The CPU interface section takes care of data transfer between 8279 and the processor.
- This section has eight bidirectional data lines DB0 to DB7 for data transfer between 8279 and CPU.
- It requires two internal address A =0 for selecting data buffer and A = 1 for selecting control register of8279.
- The control signals WR (low), RD (low), CS (low) and A0 are used for read/write to 8279.
- It has an interrupt request line IRQ, for interrupt driven data transfer with processor.
- The 8279 require an internal clock frequency of 100 kHz. This can be obtained by dividing the input clock by an internal prescaler.
- The RESET signal sets the 8279 in 16-character display with two -key lockout keyboard modes.