

## 4. Bus Structure and Memory Devices

### 4.1 Bus Structure, synchronous and asynchronous data bus, address bus, bus timing

#### Bus Structure:

A microprocessor unit performs basically four operations: memory read, memory write, I/O read, I/O write. These operations are part of communication between MPU & peripheral devices.

A communication includes identifying peripheral or memory location, transfer of data & control functions. These are carried out using address bus, data bus and control bus respectively. All the buses together is called the system bus.

In case of 8085 MPU we have

- 8 unidirectional address pins
- 8 bi directional multiplexed address/ data pins
- 11 control output pins
- 11 control input pins

#### Data bus:

The data bus provides a path for data flow between the system modules. It consists of a number of separate lines, generally 8, 16, 32 or 64. The number of lines is referred to as width of the data bus. A single line can only carry one bit at a time, the number of lines determine how many bits can be transmitted at a time. The width also determines the overall system performance.

An 8 bit data bus would require twice the time required by 16 bit data bus to transmit 16 bit data

- 8085 – 8 bit data bus
- 8086 – 16 bit data bus

#### Asynchronous bus:

In an asynchronous data bus the timing is maintained in such a way that occurrence of one event on the bus follows and depends on the occurrence of previous event. Let us consider a simple memory read:

- The CPU places the memory read and address signals on the bus.
- After allowing for these two signals to stabilize, it issues Master synchronous signal (MSYNC) to indicate the presence of valid address and control signals on the bus.
- The addressed memory module responds with the data and the slave synchronous (SSYNC) signal.

Thus at a certain time it cannot be identified when there is data or an address on the bus.

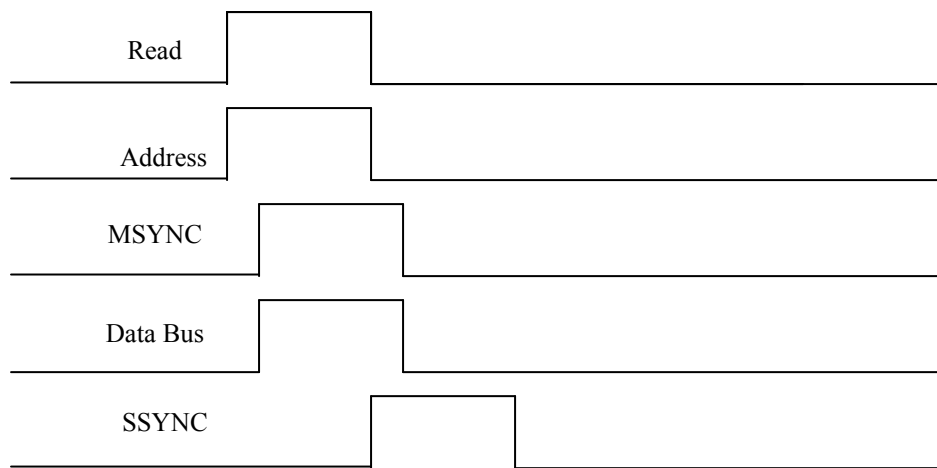


Fig: Asynchronous Bus

### Synchronous Bus:

A synchronous bus has its events tied by a clock. The clock transmits a regular sequence of 0's and 1's of equal duration. A single 1-0 transmission is called clock cycle or bus cycle. All other devices work with the clock cycle. The events start at the beginning of the clock cycle. A memory read in case of a synchronous bus progresses as:

- The CPU issues a start signal to indicate the presence of address and control information on the bus.
- Then it issues the memory read signal and places the memory address on the address bus.
- The addressed memory module recognizes the address and after a delay or one clock cycle it places the data and acknowledgement signals on the buses.

In a synchronous bus, all the devices are tied to a fixed rate.

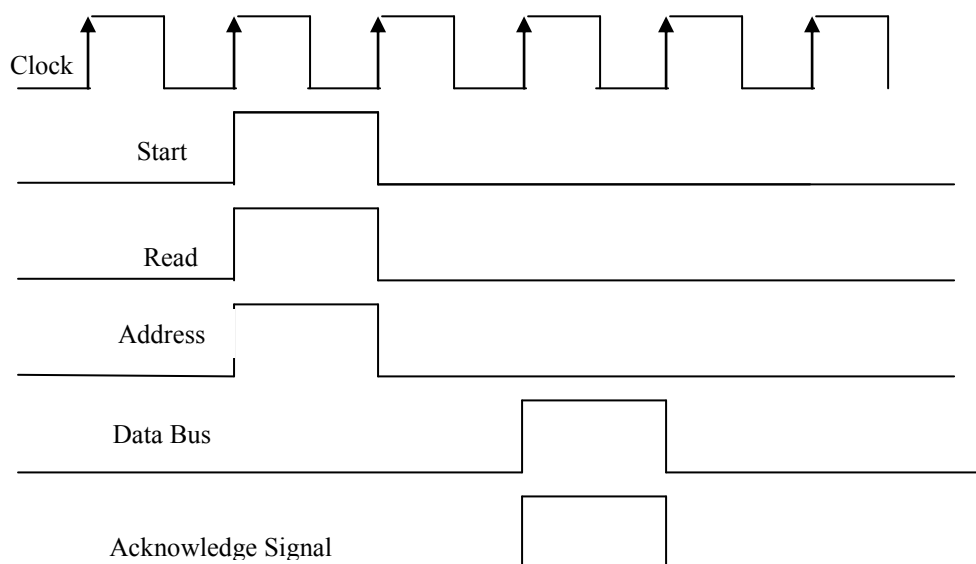


Fig: Synchronous Bus

### **Address bus:**

The address bus is used to designate the source and destination of data in data bus. In a computer system, each peripheral or memory location is identified by a binary number called an address.

The width of the address bus determines the maximum possible memory capacity of the system. The address bus is also used to address I/O ports. Usually higher order bits are used to select particular modules and lower order bit select a memory location or I/O port within a module.

8085 – 16 bit address bus

Thus, maximum amount of memory locations it can address

$$2^{16} = 65,536 \text{ or } 64 \text{ Kb}$$

### **Control bus:**

These are group of lines used to control the data and address bus. Since this bus is shared by all the component of the microcomputer system; there must be come control mechanism to distinguish between data and address. The timing signals indicate the validity of data and address information; while command signal specify operations to be performed. Some of the control signals are:

- Memory write
- Memory Read
- I/O write
- I/O read
- ALE
- Interrupt Request
- Interrupt Acknowledge

## **4.2 Static and Dynamic RAM, ROM**

Memory is like the pages of a notebook with space for a fixed number of binary numbers on each line. Memory is an essential component of microcomputer system; storing binary instructions and data for the microprocessor. Memory locations are identified by an address. The address bus of the microprocessor defines the maximum amount of memory a microprocessor can address.

Microcomputer memory can be grouped as

- Processor Memory
- Primary or Main memory
- Secondary Memory

### **Processor Memory:**

Processor memory is a group of register along with processor. They temporarily hold data during computation since processor and registers are fabricated with same technology they have same speed. Although the use of such registers increase speed of the processor; the cost factor allows only few registers to be included.

Now, a separate memory is kept alongside the processor to keep the most frequently needed information. The performance could be improved if this new memory could be improved if this new memory could be included within the processor. This newly added memory is known as 'cache memory'.

### **Primary Memory:**

This is the memory the microprocessor uses in executing & storing programs. Usually the size of this memory is larger and slower than processor memory.

Primary Memory can be grouped as:

- Random Access Memory (RAM)
- Read Only Memory (ROM)

### **RAM**

It is used primarily for information that is likely to be altered, such as writing programs or receiving data. This memory is volatile. Two types of RAM are available.

#### **Static RAM (SRAM)**

This memory is made up of flip-flops, and it stores the bit as a voltage. Each memory cell requires six transistors; they have low packing density but high speed and consume more power.

#### **Dynamic RAM (DRAM)**

DRAM stores data in capacitors; so it can hold data for a few milliseconds. Hence DRAM must be refreshed from time to time. In DRAM greater number of bits can be stored in small chips but the interfacing circuit of DRAM is complicated because of refreshing. They are cheaper than SRAM.

#### **ROM (Read Only Memory)**

The ROM is a non-volatile memory. This memory is used for programs and data that need not be altered. As the name suggests, the information is read only, which means once a bit pattern is stored it is permanent or at least semi permanent. The different types of ROM are:

### **4.3 PROM, UVEPROM, EEPROM, PROM programmer and erasure**

#### **Masked ROM:**

They are permanent ROM recorded by masking. Generally manufacturers use this process to produce ROM in large numbers.

#### **PROM:**

These are un-programmed ROM. The fuses on the ROM are not burned. A programmer can program this ROM according to his needs. The process is known as 'burning the PROM', and the information stored is permanent.

## **EPROM:**

These ROM can be reprogrammed and erased. Two types of such EPROM are available.

## **UV-EPROM**

The memory of such ROM can be erased by exposing the chip via a lid or window on the chip to ultraviolet light. The erase time generally varies between 10 to 30 minutes. The EPROM can be programmed by inserting the chip into a socket of the PROM programmer and providing proper addresses. The programming time varies from 1 to 2 minutes.

## **EEPROM**

These are similar to UVEPROMs, except that the memory is altered by electrical signals at register level.

## **Secondary Memory**

Secondary memory are storage devices. These devices have high data holding capacity. They store programs that are not frequently used by the processor. They are slow and have larger size. Some of the examples are: Hard disk, Floppy disk, Magnetic Tapes, CD, DVD etc.

### **4.4 Address decoding, memory interface (8, 16, 32, 64 bit)**

A microcomputer has microprocessor, memory & I/O devices attached. A processor communicates with all parts interconnected in the system through common address and data bus. Hence only one device can transmit data through the bus at a time & others can only receive data. Now, to ensure that the proper device gets addressed at proper time, the method of address decoding is used

In this process memory blocks & I/O units are assigned a specific address. The address is determined by the way the device is connected to processor. The signal used to drive the device is called chip select. Since only one chip select is activated at a time only one block of memory or I/O device is activated. The purpose of address decoding circuit is to ensure that the Chip select signals to other devices are not activated. There are two methods for mapping address of these devices. They are:

- I/O mapped I/O
- Memory mapped I/O

In I/O mapped I/O method the I/O devices are addressed with 8-bit address. It means that the Chip Select signals for the devices are derived by using the 8 address lines. In 8085 microprocessor, the general procedure involves the use of lower 8 bit address lines for deriving the Chip Select signal. Thus only 256 devices or addresses can be mapped. The higher order address bits are don't care conditions. Instruction used for input and output are IN and OUT.

In memory mapped memory I/O mode the I/O devices are addressed with 16 bit data. It means that in this mode the Chip Select signals for the devices are derived using all 16 address lines from the processor. Thus a total of 64 KB can be addressed.

Usually in microcomputers based on 8085 processor the memories like RAM, ROM, EPROM etc uses memory mapping whereas the devices like 8255A, 8251A, input/output latches etc use I/O mapping. Now depending on the addresses that are allocated to the device the address decoding can be categorized as

### 1. Unique address decoding:

If all of the address lines available on that mapping mode are used for address decoding, then that decoding is called unique address decoding. Thus, in memory mapped memory I/O all 16 lines are used and in I/O mapped I/O method all 8 lines are used for address decoding.

### 2. Non-Unique address decoding:

If all of the address lines available on that mode are not used address decoding, then that decoding is called non-unique address decoding. Such practice of address decoding is simple but rather conflicting as the device is activated for a wide range of addresses.

### Memory Interface

Using the proper address and data bus, a control bus must also be used to control the operation of the memory circuitry. The operations to be considered for memory interface are:

- Read data from memory (Access memory)
- Write data from memory (Access memory)
- Do no access memory

## **5 Input/Output Interfaces**

### **5.1 Serial Communication**

Serial Communication involves the transmission of data from one place to another using serial device. In serial communication the data is transmitted bit by bit on a single line. This minimizes the interconnecting wires, thereby also reducing the number of line drivers and receivers. Thus, only single bit can be transmitted at a time reducing the data transfer rate as the time required to transmit increases. However, since only one wire is used for data transfer the amount of cross-talk (interference between different lines) decreases. This in turn allows us to increase data rate which was the limiting factor in parallel communication.

#### **5.1.1 Asynchronous interface: ASCII code, baud rate, start bit, stop bit, parity bit**

##### **ASCII code**

The acronym ASCII stands for the American Standard Code for Information Interchange. It is an 8-bit code commonly used with microprocessors (including those of computers) for representing alphanumeric codes. Out of the 8 bits first 7 are used to represent the character while the eighth bit is used to test for errors and is referred to as parity bit. The 7 bit code provide with 128 ( $2^7$ ) combinations. The parity bit used can be that of even parity or odd parity.

##### **Baud rate**

The term baud rate is used to indicate the rate at which serial data is being transferred. Baud rate is defined as  $1/(\text{bit interval})$ . The rate is usually expressed as Bd or bits/second.

##### **Start bit & Stop bit**

In serial communication the data are sent serially so to differentiate between proper data and garbage data the data is enclosed in start bit and stop bit. The start bit indicates the beginning of a data character and indicated by the line going low for 1 bit time. The stop bit is indicated by line going continuously high after the data is over.

##### **Parity bit**

The parity bit follows the final data bit. Depending on the type of parity its value may be 0 or 1. This bit is used to check errors in received data.

#### **5.1.2 Synchronous interface**

Synchronous Interface is a widely used interface standard for industrial applications between a master (e.g. controller) and a slave (e.g. sensor). Synchronous Serial Interface (SSI) is based on RS422.

#### **5.1.3 8255 Programmable Peripheral Interface (Block diagram and Modes only)**

The 8255A is a widely used programmable parallel I/O device. It can be programmed to transfer data under various conditions from simple I/O to interrupt I/O. It is a general purpose

I/O device that can be used with almost any microprocessor. The PPI can work in two modes depending on the bit D7 on the control word. These are:

- BSR (Bit Set Reset) mode
- I/O (input/output) mode

24 I/O ports available in 8255 are divided into two groups (Group A and Group B). The 24 I/O ports form three 8-bit parallel ports (Port A, Port B and Port C). The eight bits of port C can be used as individual bits or be grouped in two four bit ports. The functions of these ports are defined by writing a control word in the control register. The different modes available in 8255 are shown below.

### Block Diagram of 8255

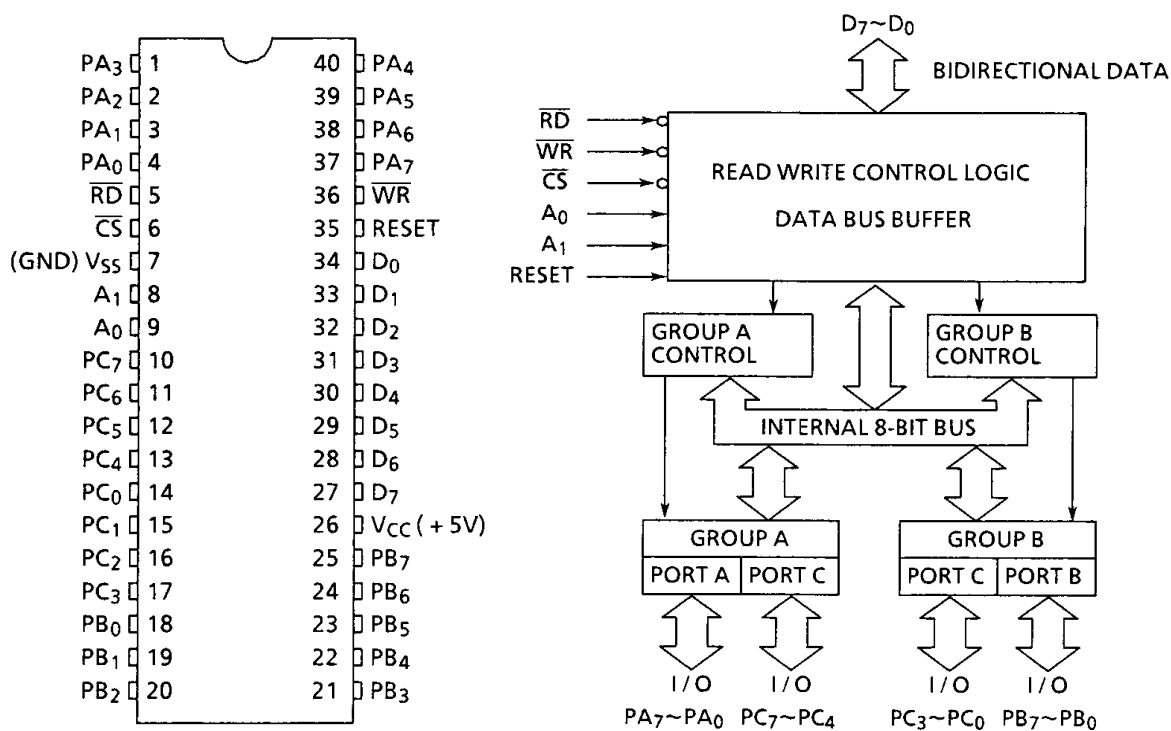


Fig: Block diagram of 8255 PPI

The figure above shows the pin configuration and the block diagram of 8255 PPI. We see that there are 24 I/O lines. Port A and Port B can be used as 8 bit input or output port. Port C is a special supplementary port which can be used as an 8 bit I/O port, two 4 bit ports, as individual lines, or to produce handshake signals for port A and B.

We also see that there are eight data lines which allow you to write data bytes to a port or the control register and read bytes from the port or the status register under the control of the RD' and WR' lines. The address inputs, A<sub>0</sub> and A<sub>1</sub>, allow you to selectively access one of the three ports or the control register. The internal addresses for the device are: port A, 00; port B, 01; port C, 10; control register, 11. Selecting the CS' input will enable 8255 for reading or writing. It is connected to address decoder circuitry to select the device when addressed. The RESET input of the 8255 is connected to the system reset line. When reset the device is programmed as input.



## 8255 Operational Modes and Initialization

The ports in 8255 can attain three modes of operation that can be selected by control words.

- |                                    |                    |
|------------------------------------|--------------------|
| Mode 0- Basic I/O                  | (Group A, Group B) |
| Mode 1- Strobe input/Strobe output | (Group A, Group B) |
| Mode 2- Two way bus                | (Port A only)      |

### MODE 0

When we need a port for simple input or output without handshaking, we initialize the port in mode 0. If both port A and port B are initialized in mode 0, then the two halves of port C can be used together as an additional 8 bit port, or they can be used individually as two 4 bit ports. When used as outputs, the port C lines can be individually set or reset by sending a special control word to control register address (Bit set/reset mode). Also the two halves of port C are independent so one half can be initialized as input and the other half initialized as output.

- Output are latched
- Inputs are not latched
- Ports don't have handshake or interrupt capability

### MODE 1

When we need to use strobed input or output then we can initialize port A and port B in mode 1. In this mode, some of the pins of port C function as handshake lines. Pins PC0, PC1, and PC2 function as handshake lines for port B if it is initialized in mode 1. If port A is initialized as a input handshake port, then pins PC3, PC4, and PC4 function as handshake signals. Pins PC6 and PC7 are available for used as input or output lines. If port A is initialized as a handshake output port, then port C pins PC3, PC6 and PC7 function as handshake signals, Port C pins PC4 and PC5 are available for use as input or output lines.

- Input and output data are latched
- Interrupt logic is supported

### MODE 2

Only port A can be initialized in mode 2. In mode 2, port A can be used for bidirectional handshake data transfer. This means that data can be output or input on the same eight lines. If port A is initialized in mode 2, then pins PC3 through PC7 are used as handshake lines for port A.

### ***Precaution for use in Mode 1 and 2***

*When used in Mode 1 and 2, bits which are not used as control or status in Port C can be used as follows:*

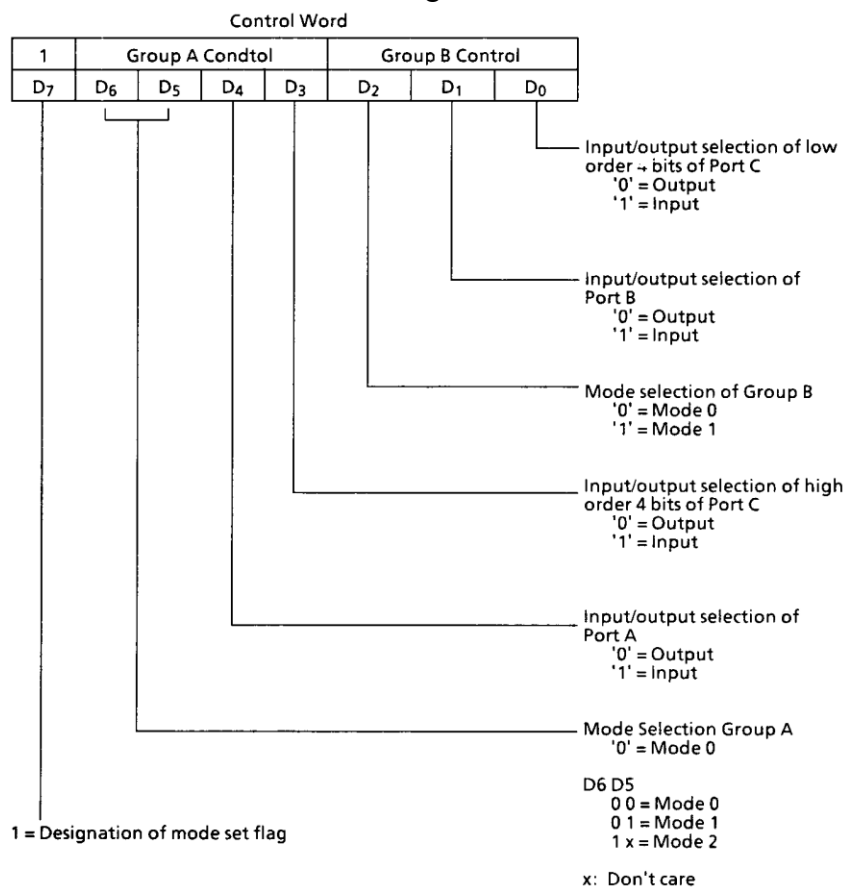
- *If programmed as the input, they are accessed by normal Port C read.*
- *If programmed as the output, high order bits of Port C (PC7-PC4) are accessed using the bit set/reset function. As to low order bits of Port C (PC3-PC0), in addition to access by the bit set/reset function, 3 bits only can be accessed by normal writing.*

## Constructing 8255 Control Word

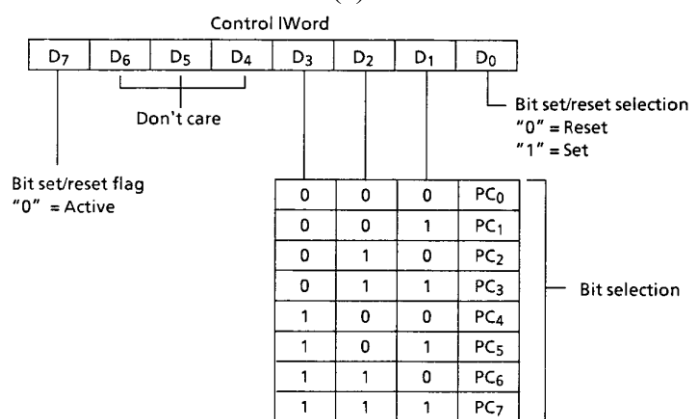
The following figure shows the formats for the two 8255 control words. The MSB of the control word distinguishes between the two control words. The *mode definition control word* format in the figure tells the device what modes you want the ports to operate in. We use the

bit set/reset control word format in the latter figure when we want to set or reset the output on a pin of port C or when you want to enable the interrupt signals for handshake data transfers.

Both the control word are sent to the control register of 8255



(a)



(b)

Fig: 8255 control word formats. (a) Mode set control word  
 (b) Port C bit set/reset control word.

Example 1:

Construct a control word to configure Ports A and port C<sub>U</sub> as output ports and port B and port C<sub>L</sub> as input ports.

D7	D6	D5	D4	D3	D2	D1	D0	Control Word
1	0	0	0	0	0	1	1	83H
I/O function	Port A in mode 0		Port A as output	Port C <sub>U</sub> as output	Port B in mode 0	Port B as input	Port C <sub>L</sub> as input	

Example2:

Write Control word to set and reset bits PC7 and PC3.

	D7	D6	D5	D4	D3	D2	D1	D0	Control Word
To set bit PC7	0	0	0	0	1	1	1	1	0FH
To reset bit PC7	0	0	0	0	1	1	1	0	0EH
To set bit PC3	0	0	0	0	0	1	1	1	07H
To reset bit PC3	0	0	0	0	0	1	1	0	06H

#### 5.1.4 8251 Programmable Communication Interface (Block diagram and Modes only)

The 8251 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 parallel conversion.

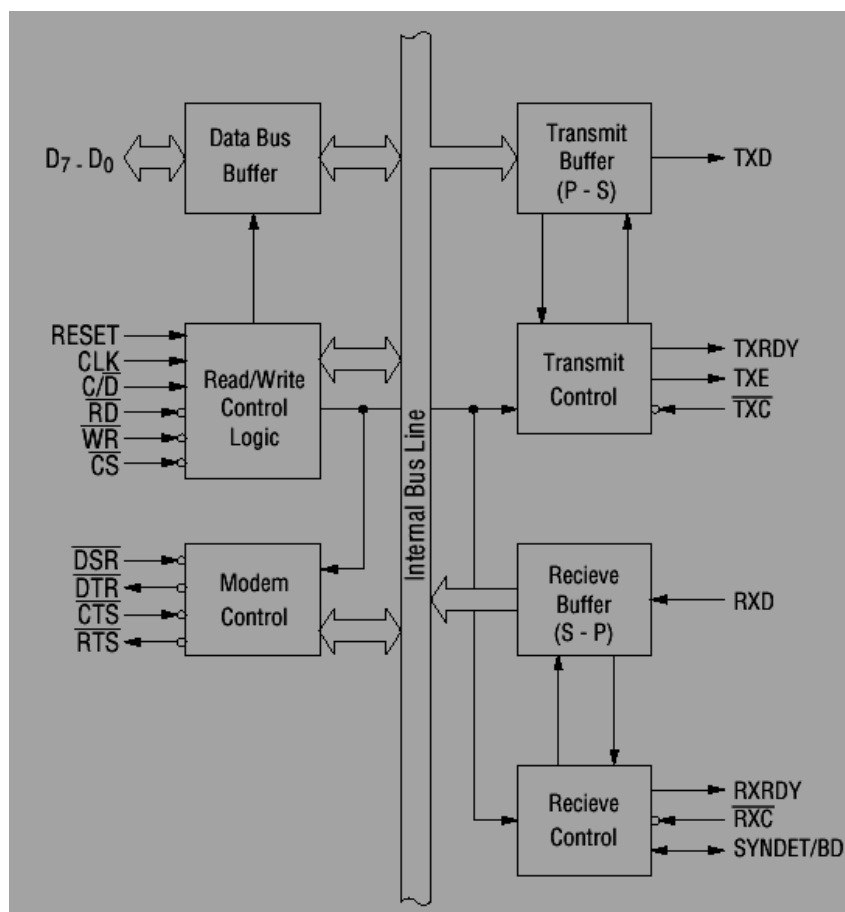


Fig: Block Diagram of Intel 8251

The internal block diagram of 8251 is shown above and the different pin functions of 8251 are listed on table below:

Pin Name	Pin Function
D7-D0	Data bus (8 bit)
C/D'	Control or data is to be written or read
RD'	Read data command
WR'	Write data or control command
CS'	Chip select
CLK	Clock pulse (TTL0
REST	Reset
TxC'	Transmitter clock
TxD	Transmitter data
RxC'	Receiver clock
RxD	Receiver data
RxRDY	Receiver ready (has character for CPU)
TxRDY	Transmitter ready (ready for char from CPU)
DSR'	Data set ready
DTR'	Data terminal ready
SYNDET/BD	Sync detect/ break detect
RTS'	Request to send data
CTS'	Clear to send data
TxEMPTY	Transmitter empty
Vcc	+5 V supply
GND	Ground

Fig: Pin Description of 8251

The eight parallel lines, D7-D0, connect to the system data bus so that data words and control/status words can be transferred to and from the device. The chip select (CS') input is connected to an address decoder so the device is enabled when addressed. The 8251 has two internal addresses, a control address which is selected when C/D is high, and a data address which is selected when C/D input is low. The RESET, RD', and WR' lines are connected to the system signals with the same names. The clock input of the 8251 is usually connected to a signal derived from the system clock to synchronize internal operations of the USART with the processor timing.

The signal labeled TxD on the upper right corner of the 8251 block diagram is the actual serial data output. The pin labeled RxD is the serial data input. The additional circuitry is needed to convert the TTL logic levels from 8251A to current loop or RS232C signals and opposite to receive the data. We will discuss it on section 5.4.

The shift registers in the USART require clocks to shift the serial data in and out. TxC is the transmit shift register clock input, and RxC' is the receive shift register clock input. Usually these two inputs are tied together so they are driven at same clock frequency. The frequency chosen must be 1, 16, or 64 times the transmit and receive baud rate depending upon the mode in which the 8251 is initialized. Using a clock frequency higher than the baud rate allows the receive shift register to be clocked at the centre of a bit time rather than at a transition. This reduces the chance of noise at a transition causing a read error.

The 8251A is double buffered. This means that one character can be loaded into a holding buffer while another character is being shifted out of the actual transmit shift register. The TxRDY output from the 8251 will go high when the holding buffer is empty and another character can be sent from CPU. The TxEMPTY pin on the 8251 will go high when both the holding buffer and the transmit shift register are empty. The RxRDY pin of the 8251A will go high when a character has been shifted into the receive buffer and is ready to be readout by the CPU. Incidentally, if a character is not read out before another character is shifted in, the first character will be over-written and lost.

The sync-detect/break-detect (SYNBDET/BD) pin has two uses. When the device is operating in asynchronous mode, this pin will go high if the 3 serial data input line, RxD, stays low for more than two character times. This signal then indicates an intentional break in data transmission, or a break in the signal line. When programmed for synchronous data transmission this pin will go high, when the 8251 finds a specified Sync character(s) in the incoming string of data bits.

### Initialization

To initialize an 8251A you must send first a mode word and then a command word to the control register address for the device. The following figure shows the formats for these words and for the 8251A status word which is read from the same address.

In the mode word Baud rate factor is specified by the two least significant bits of the mode word, is the ratio between the clock signal applied to the TxC'-RxC' inputs and the desired baud rate. For example, if you want to use TxC' of 19200 Hz and transmit data at 1200Bd, the baud rate factor is 19200/1200 or 16x. If bits D0 and D1 are both made 0's, the 8251A is programmed for synchronous data transfer. In this case the baud rate will be the same as the applied TxC' and RxC'. The other three combinations for these 2 bits represent asynchronous transfer. A baud rate factor of 1 can be used for asynchronous transfer only if the transmitting system and the receiving system both use the same TxC' and RxC'. The character length is specified by D2 and D3 in the mode word includes only the actual data bits not the start bit, parity bit or stop bit(s). If parity (D4) is disabled then no parity bit is inserted in the transmitted bit string. The bit D5 represents the type of parity used for error correction. The last two bits signifies the number of stop bits used after the data is sent.

After the mode word is send, you must then send it a command word.

Bit D0=1 enables the transmitter section and TxRDY output. When enabled, the 8251A TxRDY output will be asserted high if the CTS input has been asserted low, and the transmitter holding buffer is ready for another character from CPU. When a character is written to the 8251A data address, the TxRDY signal will go low and remain low until the holding buffer is again ready for another character.

Bit D1=1 will cause the DTR output of 8251 to be asserted low. This signal is used to tell a modem that a terminal or computer is operational.

Bit D2=1 enables the RxRDY output pin of 8251A. If enabled, the RxRDY pin will go high when the 8251 has a character in its receiver buffer ready to be send. This signal can be connected to an interrupt input so that characters can be read in on an interrupt basis. The RxRDY output is reset when a character is read from the 8251A.

Bit D3=1 causes the 8251 to output a character of all 0's which is called a break character. A break character is sometimes used to indicate the end of a block of transmitted data.

Bit D4=1 causes 8251 to reset the parity, overrun and framing error flags in the 8251 status register.

Bit D5=1 causes 8251 to assert its RTS output low. This signal is sent to a modem to ask whether the modem and the receiving system are ready for a data character to be sent.

Bit D6=1 causes the 8251 to internally reset when the command word is sent. After the software reset command is sent in this way, a new mode word must be sent

Bit D7=1 tells the 8251 to look for specified sync characters in a system of bits being shifted in. If 8251 finds specified sync characters, it will assert its SYNDET/BD pin high.

## 5.2 Parallel Communication

In all preceding chapters, we have used port devices to input and output parallel data to and from the microprocessor and other devices. Although serial communication is also popular parallel data transfer takes place between any two devices like microprocessor and memory, microprocessor and I/O devices and memory and I/O devices.

### Methods of Parallel Communication

- Simple Input and Output:

When you need to get digital data from a simple switch such as a thermostat, into a microprocessor, all you have to do is connect the switch to an input port line and read the port. The thermostat data is always present and ready, so you can read it at any time. Similarly, to output data to a simple device such as LED, all you have to do is connect the input of the LED buffer on an output port in and output the logic level required to turn on the light. The LED is always there and ready, so you can send data to it at any time. The timing waveform below shows the data transfer sequence.

- 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. An example of this is the ASCII-encoded keyboard. When a key is pressed circuitry on the keyboard sends out the ASCII code for the pressed key on eight parallel data lines, and then sends out a strobe signal on another line to indicate that valid data is present on the eight data lines as shown in the figure. You can connect this strobe line to an input port line and poll it to determine when you can input valid data from the keyboard. Another alternative is to connect the strobe line to an interrupt input on the processor and have an interrupt service procedure read in the data when the processor receives an interrupt. The scheme behind this arrangement is that the transfer is time dependent. Thus you can only read data when a strobe pulse tells you that the data is valid.

- Single handshake I/O

The figure shows an example for a handshake data transfer from a peripheral device to a microprocessor. The peripheral outputs some parallel data and sends an STB signal to the microprocessor. The microprocessor detects the asserted STB

signal on a polled or interrupt basis and reads in the byte of data. Then the microprocessor sends an Acknowledge signal (ACK) to the peripheral to indicate that the data has been read and that the peripheral can send the next byte of data. Such transfer is strobed input from microprocessor's point of view.

Similarly, for an output the microprocessor outputs a character to the printer and raises the STB signal to the printer to tell "Here is a character for you." When the printer is ready, it answers back with the ACK signal to tell "I got that one; send me another".

- **Double Handshake Data Transfer**

Such mode of data transfer is used where even more coordination is required between the sending systems and the receiving system. Such mode of data transfer can be much easily understood as a conversation between two people. In the waveforms each signal edge has meaning. The sending device asserts its STB line to ask, "Are you ready?" The receiving system raises its ACK line high to say, "I'm ready." Then the peripheral device sends the byte of data and raises its STB' line high to say, "Here is some valid data for You." After it has read in the data the receiving system drops its ACK line low to say, "I have the data. Thank you. and I await your request to send the next byte of data".

### **5.3 Data transfer using wait interface**

To understand wait interface, let us consider a simple keyboard consisting of 8 switches, connected to a microprocessor through a parallel interface circuit. In this case, each switch is of dip switches. The microprocessor should be able to detect that a key has been activated. This can be done by observing that all bits are equal to 1 when none of the key is pressed. The processor should repeatedly read the state of the input section until it finds that one of the eight bits is equal to 0.

The microprocessor should ensure that when a key is activated, it is read only once. This can be done in the software. When one of the keys is activated and the input byte is read, the software program could repeatedly check the input byte until it reaches to all 1s indicating that the pressed key has been released

The other problem occurs from the nature of the operation of the mechanical switches. Almost all mechanical contacts are subject to vibrations or bouncing, while the switch is being opened or closed. The time required for a switch to settle to fully open or a fully closed position is about 2ms. This may not seem much as it is  $1/500^{\text{th}}$  of a second, however, a microprocessor can execute several hundred of instruction in this time. Thus there must be a mechanism to ignore the keyboard unit till the keyboard denounce is stopped. This is done by checking the input until it is the same before accepting it as valid. This can be visualized in the following flowchart.

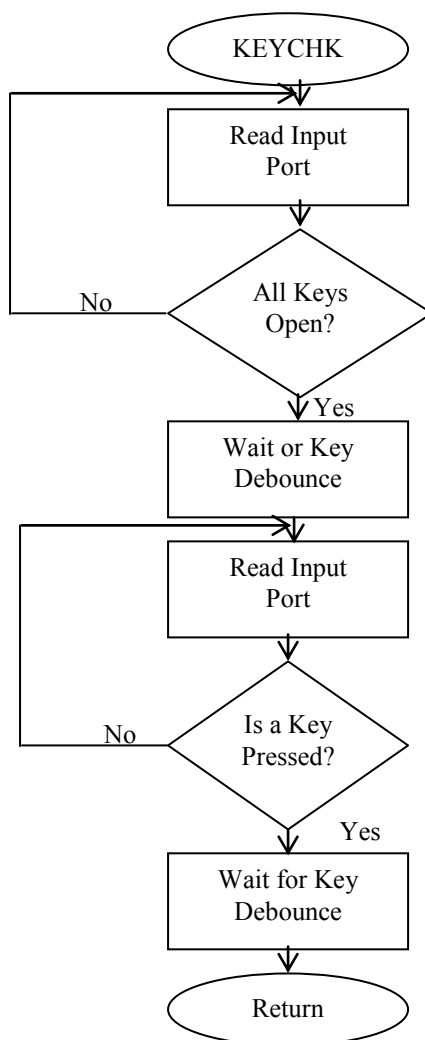


Fig: flowchart of Switch Interface

## 5.4 RS-232 and IEEE 488-1978 general purpose interface standard

### RS-232C Serial Data Standard

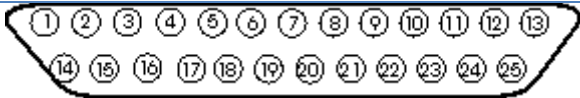
In 8251(USART), we discussed how serial communication takes place. The TTL signals output by a USART, however, are not suitable for transmission over long distances, so these signals are converted to some other form to be transmitted. In this section we discuss device used to send serial data signals over long distances.

The 1960s saw widespread use of timeshare computer terminals. However, to install new communication lines was a Herculean task. Thus modems were developed so that the terminals could use phone lines to communicate, which were already in existence. Modems are often referred to as *data communication equipment* or DCE. The terminals or computers that are sending or receiving the data are referred to as *data terminal equipment* or DTE. In response to the need for signal and handshake standards between DTE and DCE, the Electronic Industries Association (EIA) developed standard RS-232C. This standard describes the function of 25 signals and handshake pins for serial data transfer. It also



describes the voltage levels, impedance levels, rise and fall times, maximum bit rate, and maximum capacitance for these signal lines.

RS-232 stands for Recommend Standard number 232 and C is the latest revision of the standard. RS-232 is a 25 pin standard but for systems where many of 25 pins are not needed, a 9 pin connector is also used. The voltage levels for all RS-232 signals are as follows. A logic high, or mark, is a voltage between -3V and -15V. A logic low or space is a voltage between +3V and +15V. Voltages such as  $\pm 12V$  are commonly used.

Male RS232 DB25	
<b>Pin Number</b>	<b>Direction of signal:</b>
1	Protective Ground
2	Transmitted Data (TD) Outgoing Data (from a DTE to a DCE)
3	Received Data (RD) Incoming Data (from a DCE to a DTE)
4	Request To Send (RTS) Outgoing flow control signal controlled by DTE
5	Clear To Send (CTS) Incoming flow control signal controlled by DCE
6	Data Set Ready (DSR) Incoming handshaking signal controlled by DCE
7	Signal Ground Common reference voltage
8	Carrier Detect (CD) Incoming signal from a modem
20	Data Terminal Ready (DTR) Outgoing handshaking signal controlled by DTE
22	Ring Indicator (RI) Incoming signal from a modem


Male RS232 DB9	
<b>Pin Number</b>	<b>Direction of signal:</b>
1	Carrier Detect (CD) (from DCE) Incoming signal from a modem
2	Received Data (RD) Incoming Data from a DCE
3	Transmitted Data (TD) Outgoing Data to a DCE
4	Data Terminal Ready (DTR) Outgoing handshaking signal
5	Signal Ground Common reference voltage
6	Data Set Ready (DSR) Incoming handshaking signal
7	Request To Send (RTS) Outgoing flow control signal
8	Clear To Send (CTS) Incoming flow control signal
9	Ring Indicator (RI) (from DCE) Incoming signal from a modem

Fig: 25 pin and 9 pin connectors used for RS 232 standards on DTE device

### IEEE 488 standard

The IEEE-488 bus was developed to connect and control programmable instruments, and to provide a standard interface for communication between instruments from different sources. Hewlett-Packard originally developed the interfacing technique, and called it HP-IB (Hewlett-Packard Instrument Bus). The interface quickly gained popularity in the computer industry. Because the interface was so versatile, the IEEE committee renamed it GPIB (General Purpose interface Bus).

IEEE-488 allows up to 15 devices to share a single 8-bit parallel electrical bus by daisy chaining connections. The slowest device participates in control and data transfer handshakes to determine the speed of the transaction. The maximum data rate is about one Mbyte/sec in the original standard, and about 8 Mbyte/sec with IEEE-488.1-2003 (HS-488).

The IEEE-488 bus employs 16 signal lines eight bi-directional used for data transfer, three for handshake, and five for bus management plus eight ground return lines. The main features of IEEE-488 standard are as follows.

- Parallel 8 – bit data path for data read and write
- Three flow control lines (handshake signals)
- Five special lines for control of bus and interrupts
- Data speed of 1M bytes/sec limited by the speed of the slowest listener
- Special stackable connectors.
- A maximum of 15 instruments connected to a common bus
- Up to 20m maximum cable length

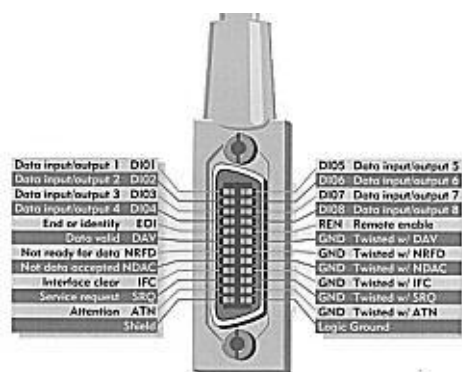


Fig: The IEEE-488 Connector

*[SAP1 and SAP2 Coming Soon]*



<http://www.facebook.com/bsccsit.com>