Lecture-65

SERIAL DATA COMMMUNICATION

The data bus of a microcomputer system is designed to transfer data to and from I/O device in parallel - all bits of a data word are transformed simultaneously. This is the fastest way of transferring the data to and from I/O device. There are many cases when it is preferable to transfer data serially (1 bit at a time). In this case the data is transferred bit by bit by making signal line High or Low depending on whether logic ‘1’ or logic ‘0’ is to be transmitted. The data transmitted serially is often sent in groups of bits that constitute a character or word. The characters are generally coded in ASCII. Serial transfer requires only one signal line or communication channel and is used under following conditions:

1) The I/O device to/from which the data is transferred is inherently serial in operation. E.g. CRT terminals, magnetic tape cardridges and cassettes. These devices are designed to receive or transmit data bit at a time.
2) The distance between the microprocessor and the I/O device is great.

As the distance between a microprocessor and an I/O device increases, the cost of running a colored cable with a number of conductors equal to the data bus width increases compared to running a single cable. The parallel data transfer also requires multiple line drivers and receivers in order to take care of loading and noise problem. A point is reached beyond which it becomes more economical to use serial data transfer, even though it may require additional hardware and /or software, than to use a multiple conductor cable. In other applications, the distance may be so great that common facilities such as telephone lines are required & then the data must be transferred in serial form.

For serial data transfer between the microprocessor and an I/O device, an interface is required in between. The interface provides two functions:

1) The logical formatting of data, including serial to parallel /parallel to serial conversion. Parallel to serial conversion is required for transmission of data (information) and serial to parallel conversion is required so that it can be easily transferred to the microcomputer bus.
2) Translation of logic signals to the electrical signal appropriate for transmitting data over the communication channel connecting the microcomputer and I/O device.

Voltage and current levels used for data communication are rarely TTL compatible. Electrical signal translation is implemented by hardware. Logical formatting of data may be implemented by software or hardware.

In the software method of data transformation, the CPU executes a program to generate the serial waveform for transmission or to determine the data from the received data from a serial device. Although this method requires a minimum of external hardware, it does require a significant amount of program instruction for both transmitting & receiving of data. Furthermore, with the software method, the CPU time is occupied during the complete transmission or reception of serial data word. With slow serial devices like a teletype writer, the time required to transmit or receive a single –

ASCII coded character can be as long as 0.1 sec. In many applications this use of CPU time would be prohibited because it may have other tasks it could be performing while the serial communication is taking place. For these reasons the hardware approach is often preferred.

Serial Communication Systems:

Serial data transfer systems are of three types: simplex, half duplex and full duplex. In a simplex system, data line can transmit data only in one direction. For instance, communication link needed to connect a serial printer to a microcomputer. Printer is an output device; therefore, the CPU only needs to transmit data to the printer. Data are not transmitted back. In this case a single unidirectional communication line can be used to connect the printer and microcomputer. This is known as simplex communication link. The other common example is radio station in which information is broadcasted by the radio station.

In half duplex system, data communication can take place in either direction but can only occur in one direction at a time. For example CRT terminal with keyboard need to both transmit data and receive data from the CPU. This requirement can be satisfied with a single communication line set up in half duplex mode. Therefore, transmission and reception of data cannot take place at the same time. The other example is two-way radio system where one user always listens while the other talks because the receiver circuitry is turned off during transmission. Once the data is transmitted in one direction, the user say ‘over’ and then the direction is reversed to receive the data.

In a full duplex mode, the data can be transmitted in both directions simultaneously therefore, separate transmission and receive lines are used to connect the peripheral and microcomputer. For example CRT terminal with keyboard can be connected to microcomputer in full duplex mode. A normal phone conversation is another example.

In this chapter we will consider only full duplex serial communication.

Types of Serial Communication:

There are two basics basic types of serial communication. The data can be sent serially either in asynchronous mode or in synchronous mode.

In asynchronous serial communication a character is sent by the transmitter whenever it is available. Thus, the time internal between two characters is variable. In other words, there can be indefinite time intervals when the transmitter is not transmitting data to the receiver. However the time interval between bits of a single character is fixed. When no character is available for transmission, the line is said to in ‘IDLE’ state or HIGH state or ‘MARK’ state. Each character carries the information of the START & the STOP bits for identification. In this case, only three-wire cable is required to connect two systems for serial communication as shown in fig.12.1. The TxD (transmit data) line of one system is connected to RxD (receive data) line of other system. The third line is used to connect the signal ground of two systems.

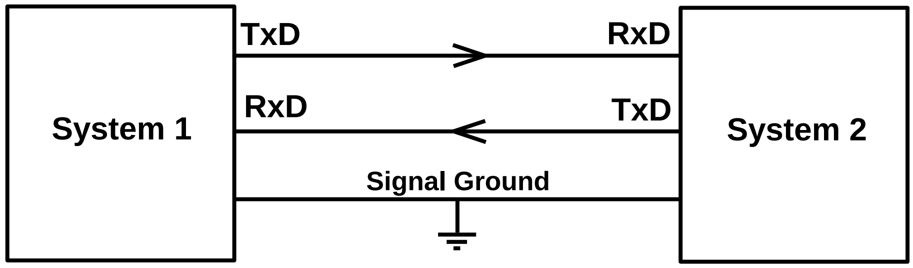

Fig.12.1 Connections for Asynchronous Serial Communication

In synchronous communication the transmitter continually transmitting data to the receiver irrespective of whether information is available or not. The data words are sent in group called blocks, and the blocks are separated by special data words called “Synchronizing Character(s)”. In other words, when the transmitter has no data to transmit, it continually transmits synchronizing characters as fillers. These synchronizing characters are used by the receiver to synchronize its clock signal to that of the receiver.

In synchronous communication, the two systems that are communicating with each other run synchronously. Therefore, the interface between two systems includes clock signal as well as TxD, RxD and signal common lines as shown in fig.12.2.

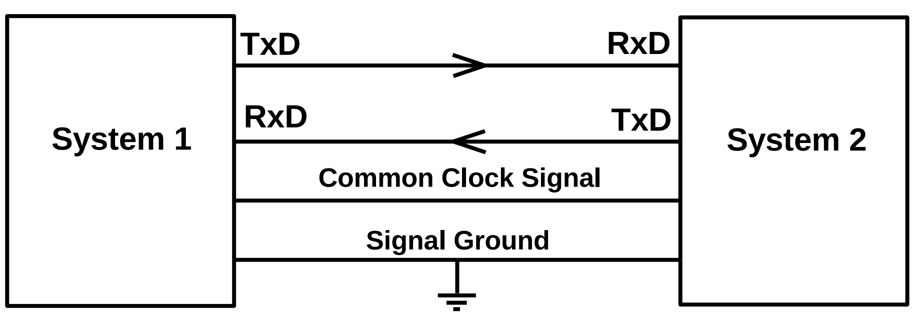

Fig.12.2 Connections for Synchronous Serial Communication

The clocks at the opposite ends of an asynchronous transmission line do not need to have exactly the same frequency because of re-synchronization at the beginning of each character. But in synchronous communication, it is the clock that determines the position of each bit. Therefore, the clocks of both transmitter and receiver must be same and normally generated by the transmitter for the receiver.

In synchronous serial communication, the characters are sent at constant rate though some characters are synchronous. Although both types of communication may waste time sending identification bits, the maximum information rate of synchronous line is higher than that of synchronous line with the same bit rate because asynchronous transmission must include extra bits with each character.

<u>Serial Communication Standards:</u>

As discussed above in serial communication the data is transmitted and received serially either in asynchronous mode or synchronous mode. To transmit the data serially it is converter to into serial form either through hardware or through software. Once the data is converted to serial form, it must be translated from TTL to appropriate logic levels so that it can be sent on serial link. Similarly once the data is received serially it must be converted into TTL form before converting it to parallel form for the microcomputer. There are several standards used for data translation:

A. Current Loops
B. RS-232C interface

Current Loops:

In current loop systems such as teletype writer a nominal current of 20 mA in signal line represents logic ‘1’ or mark state and no current represents logic ‘0’ or a space. The terms space and mask are often used to indicate the absence and presence of the current in teletype communication links. Some manufactures use 60mA as the nominal current level.

Fig.12.3 shows the schematic diagram of interfacing teletype writer with the system that uses TTL data levels. The teletype sends data to the device receiver data input line RxD, and receives data from the device transmit data output line TxD. The RxD and TxD lines might be SID or SOD of microprocessor or part of an interfacing device for serial to parallel or vice versa.

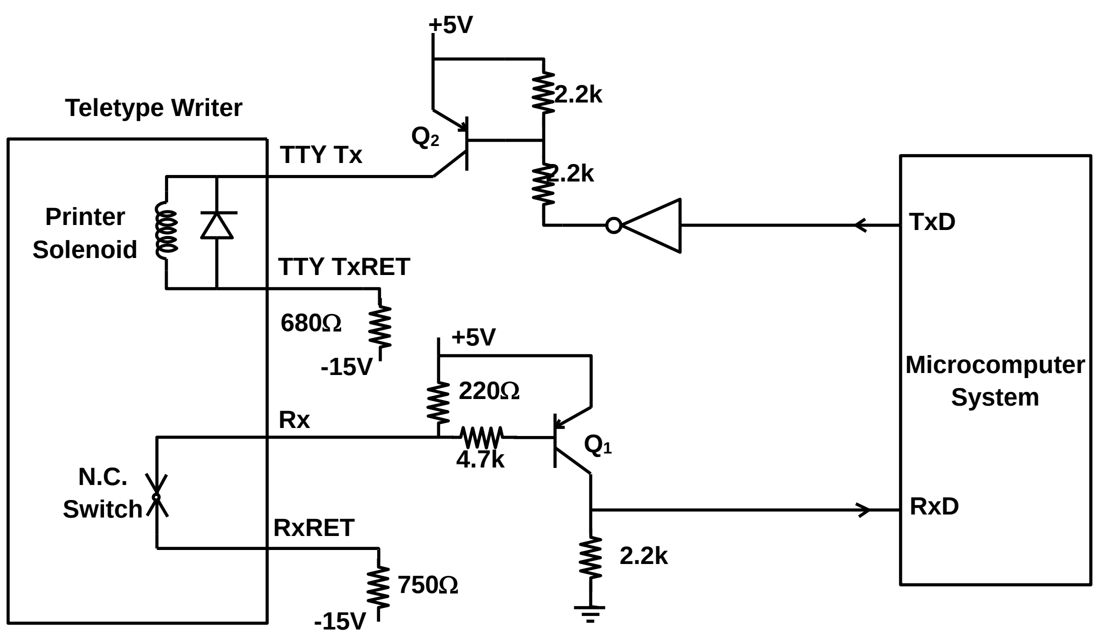

Fig.12.3 Interfacing of Teletype Writer with the Microcomputer

The TxD signal is a TTL- level signal (i.e., 0V and +5V) that controls Q_2. A high on TxD will produce a low on the PNP transistor. This will turn the transistor ON and cause a positive current of 20mA to flow out of the TTY Tx line. Inside a teletype writer this current flows through the printer solenoid and back to the TTY TxRET line. The current then flows on to -15V through 680Ω to complete the path. If TxD signal is ’0’ then no current flows through the solenoid. If the TxD is an ASCII–coded serial signal, the pattern of current and no current through the solenoid sets up internal electro-magnets to print the desired character.

The teletype sends data to the μc interface by opening and closing an internal switch. When it is not transmitting, the teletype switch is normal closed so that Q_1 is ON and +5V is applied to the RxD input. This is idling or masking condition. The teletype begins its transmission of opening the switch that turn OFF Q_1 and applies 0V to RxD. This in the START bit, which begins the serial transmission. The teletype then opens and closes the switch according to the ASCII code of the keyboard character it is transmitting.

EIA RS-232C Standard:

The devices used the serial data over long distance are called Data Communication Equipment (DCE). The terminal or computer that is sending and receiving the data is referred to Data Terminal Equipment (DTE). In response to the need for signal and handshake standards between DCE and DTE, the Electronic Industries Association (EIA) developed EIA standard RS-232C. This standard describes the function of 25 signal and handshake pins for serial data transfer. It also describes the voltage levels, rise and full times, maximum bit rate and maximum capacitance for these signals.

The voltage levels for all RS 232C signals are as follows:

1) A logic ‘1’ or mark, is a voltage between -3V and -15V under load (-25V no load).
2) A logic ‘0’ or space is a voltage between +3V and +15V under load (+25V no load).

Any voltage between -3V & +3V is undefined. Typically, an RS-232C system uses nominal voltage of -12V and +12V for logic ‘1’ and ‘0’ respectively. Most of standard ICs and microcomputer components use TTL logic level (0V, +5V) and are not directly compatible with peripheral equipment that uses RS-232C. This requires an interface circuitry to convert TTL level to RS232C and vice-versa. Two ICs are designed especially for this purpose the Motorola MC-1488 converts TTL inputs to RS-232C output and the MC-1489 converts RS-232C inputs to TTL outputs. It is shown in fig.12.4.

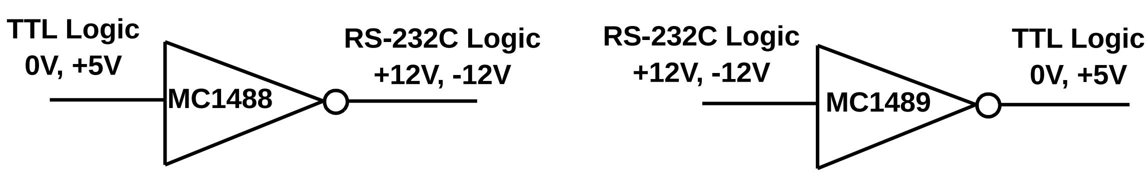

Fig.12.4 Conversion of TTL to RS-232C Logic Levels and Vice-versa

Whenever two devices, one uses TTL logic and other RS-232C logic, are to be connected, the above ICs are used in between for logic level conversion. If the data is to be transmitted using telephone lines, MODEM are used as DCE.