# Module

# 3

# Embedded Systems I/O

# Lesson
# 17

# USB and IrDA

## Instructional Objectives

After going through this lesson the student would be able to learn basics of

- The Universal Serial Bus Signals
- The IrDA standard

## Pre-Requisite

Digital Electronics, Microprocessors

## 17(I) The USB Port

As personal computers and other microprocessor based embedded systems began handling photographic images, audio, video and other bulky data, the traditional communications buses are not enough to carry the data as fast as it is desired. So a group of leading computer and telecom firms including IBM, Intel, Microsoft, Compaq, Digital Equipment, NEC and Northern Telecom got together and developed USB.

The USB is a medium-speed serial data bus designed to carry relatively large amounts of data over relatively short cables: up to about five meters long. It can support data rates of up to 12Mb/s (megabits per second). The USB is an addressable bus system, with a seven-bit address code so it can support up to 127 different devices or nodes at once (the all zeroes code is not a valid address). However it can have only one host. The host with its peripherals connected via the USB forms a star network. On the other hand any device connected to the USB can have a number of other nodes connected to it in daisy-chain fashion, so it can also form the hub for a mini-star sub-network. Similarly you can have a device which purely functions as a hub for other node devices, with no separate function of its own. This expansion via hubs is because the USB supports a tiered star topology, as shown in Fig.17.1. Each USB hub acts as a kind of traffic cop for its part of the network, routing data from the host to its correct address and preventing bus contention clashes between devices trying to send data at the same time. On a USB hub device, the single port used to connect to the host PC either directly or via another hub is known as the upstream port, while the ports used for connecting other devices to the USB are known as the downstream ports. This is illustrated in Fig.17.2. USB hubs work transparently as far as the host PC and its operating system are concerned. Most hubs provide either four or seven downstream ports, or less if they already include a USB device of their own. Another important feature of the USB is that it is designed to allow hot swapping i.e. devices can be plugged into and unplugged from the bus without having to turn the power off and on again, re-boot the PC or even manually start a driver program. A new device can simply be connected to the USB, and the PC’s operating system should recognize it and automatically set up the necessary driver to service it.

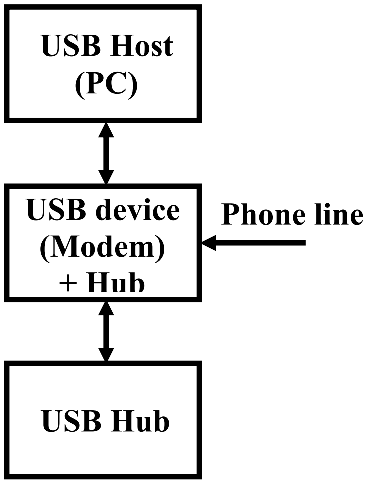


**Fig. 17.1 The USB is a medium speed serial bus used to transfer data between a PC and its peripherals. It uses a tiered star configuration, with expansion via hubs (either separate, or in USB devices).**

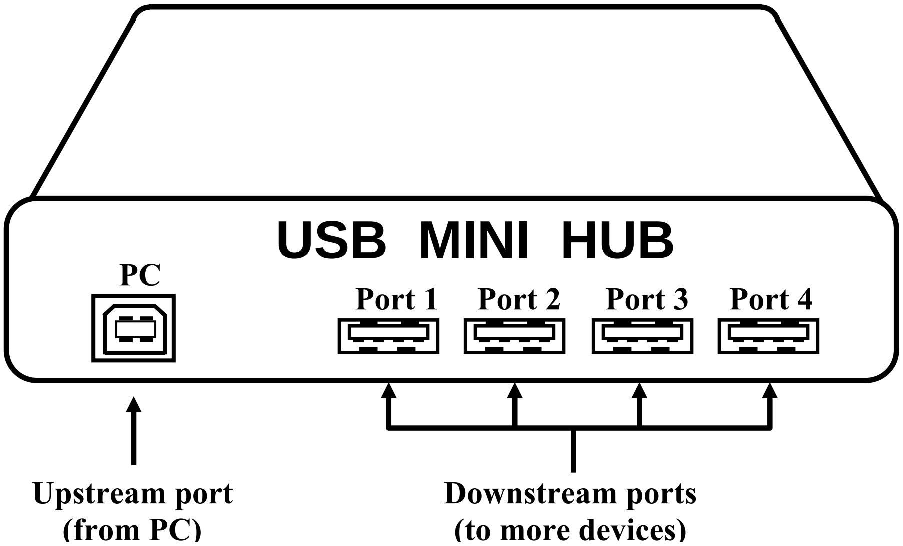


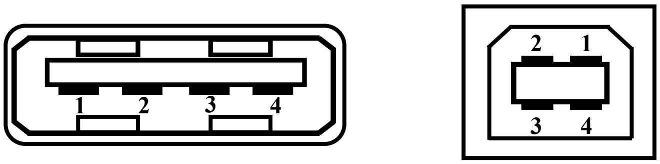
**Fig. 17.2 The port on a USB device or hub which connects to the PC host (either directly or via another hub) is known as the upstream port, while hub ports which connect to additional USB devices are downstream ports. Downstream ports use Type A sockets, while upstream ports use Type B sockets.**

## Power and data

USB cables consist of two twisted pairs of wires, one pair used to carry the bidirectional serial data and the other pair for 5V DC power. This makes it possible for low-powered peripherals such as a mouse, joystick or modem to be powered directly from the USB or strictly from the host (or the nearest hub) upstream, via the USB. Most modern PCs have two USB ports, and each can provide up to 500mA of 5V DC power for bus powered peripherals Individual peripheral devices (including hubs) can draw a maximum of 100mA from their upstream USB port, so if they require less than this figure for operation they can be bus powered. If they need more, they have to use their own power supply such as a plug-pack adaptor. Hubs should be able to supply up to 500mA at 5V from each downstream port, if they are not bus powered. Serial data is sent along the USB in differential or push-pull mode, with opposite polarities on the two signal lines. This improves the signal-to-noise ratio (SNR), by doubling the effective signal amplitude and also allowing the cancellation of any common-mode noise induced into the cable. The data is sent in non-return-to-zero (NRTZ) format, with signal levels of 3.3V peak (i.e., 6V peak differential). USB cables use two different types of connectors: Type-A plugs for the upstream end, and Type B plugs for the downstream end. Hence the USB ports of PCs are provided with matching Type-A sockets, as are the downstream ports of hubs, while the upstream ports of USB devices (including hubs) have Type B sockets. Type-A plugs and sockets are flat in shape and have the four connections in line, while Type B plugs and sockets are much squarer in shape and have two connections on either side of the centre spigot (Fig.17.3). Both types of connector are polarized so they cannot be inserted the wrong way around. Fig.17.3 shows the pin connections for both type of connector, with sockets shown and viewed from the front. Note that although USB cables having a Type-A plug at each end are available, they should never be used to connect two PCs together, via their USB ports. This is because a USB network can only have one host, and both would try to claim that role. In any case, the cable would also short their 5V power rails together, which could cause a damaging current to flow. USB is not designed for direct data transfer between PCs. All normal USB connections should be made using cables with a Type A plug at one end and a Type B plug at the other, although extension cables with a Type A plug at one end and a Type A socket at the other can also be used, providing the total extended length of a cable doesn't exceed 5m. By the way, USB cables are usually easy to identify as the plugs have a distinctive symbol molded into them (Fig.17.4).

## Data formats (Fig.17.5)

USB data transfer is essentially in the form of packets of data, sent back and forth between the host and peripheral devices. However because USB is designed to handle many different types of data, it can use four different data formats as appropriate. One of the two main formats is bulk asynchronous mode, which is used for transferring data that is not time critical. The packets can be interleaved on the USB with others being sent to or from other devices. The other main format is isochronous mode, used to transfer data that is time critical such as audio data to digital speakers, or to/from a modem. These packets must not be delayed by those from other devices. The two other data formats are interrupt format, used by devices to request servicing from the PC/host, and control format, used by the PC/host to send token packets to control bus operation, and by all devices to send handshake packets to indicate whether the data they have just received was OK (ACK) or had errors (NAK). Some of the data formats are illustrated in Fig.17.5. Note that all data packets begin with a sync byte (01hex), used to synchronize the PLL (phase-locked loop) in the receiving device's USB controller. This is followed by the packet identifier (PID), containing a four-bit nibble (sent in both normal and inverted form) which indicates the type of data and the direction it is going in (i.e., to or from the host). Token packets then have the 7-bit address of the destination device and a 4-bit end point field to indicate which of that device's registers it's to be sent to. On the other hand data packets have a data field of up to 1023 bytes of data following the PID field, while Start of Frame (SOF) packets have an 11-bit frame identifier instead and handshake packets have no other field. Most packets end with a cyclic redundancy check (CRC) field of either five or 16 bits, for error checking, except handshake packets which rely on the redundancy in the PID field. All USB data is sent serially, of course, and leastsignificant-bit (LSB) first. Luckily all of the fine details of USB handshaking and data transfer are looked after by the driver software in the host and the firmware built into the USB controller inside each USB peripheral device and hub



**Type A socket**
**(from front)**

| Pin connections | |
|---|---|
| **Pin No.** | **Signal** |
| 1 | + 5V Power |
| 2 | - Data |
| 3 | + Data |
| 4 | Ground |

**Fig. 17.3 Pin connections for the two different types of USB socket, as viewed from the front.**

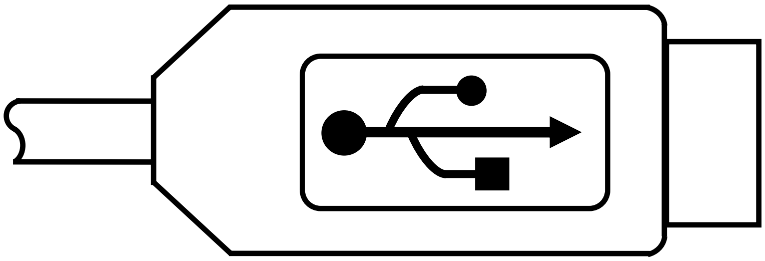


**Fig. 17.4 Most USB plugs have this distinctive marking symbol.**

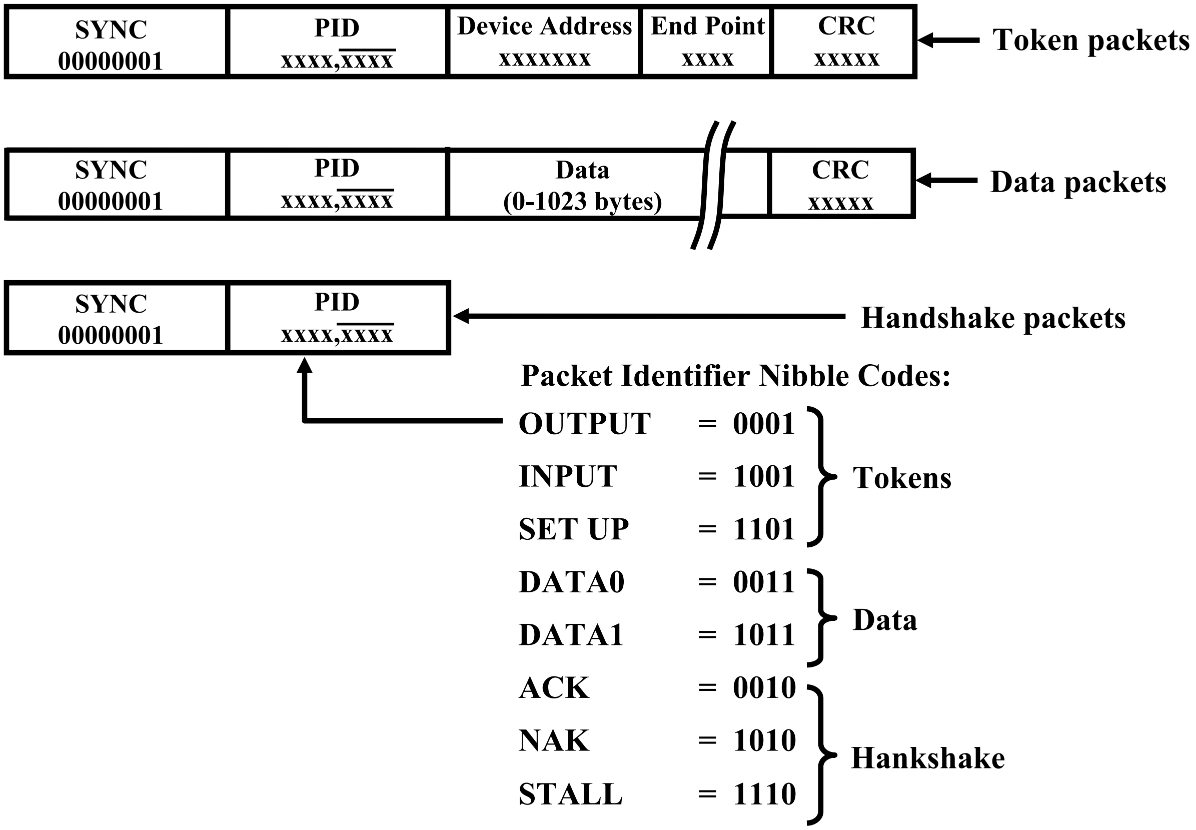


**Fig. 17.5 Examples of the various kinds of USB signaling and data packets.**

## 17(II) IrDA Standard

**IrDA** is the abbreviation for the Infrared Data Association, a non–profit organization for setting standards in IR serial computer connections.

The transmission in an IrDA–compatible mode (sometimes called SIR for serial IR) uses, in the simplest case, the RS232 port, a built–in standard of all compatible PCs. With a simple interface, shortening the bit length to a maximum of 3/16 of its original length for power–saving requirements, an infrared emitting diode is driven to transmit an optical signal to the receiver. This type of transmission covers the data range up to115.2 kbit/s which is the maximum data rate supported by standard UARTs (Fig.17.7). The minimum demand for transmission speed for IrDA is only 9600 bit/s. All transmissions must be started at this frequency to enable compatibility. Higher speeds are a matter of negotiation of the ports after establishing the links.

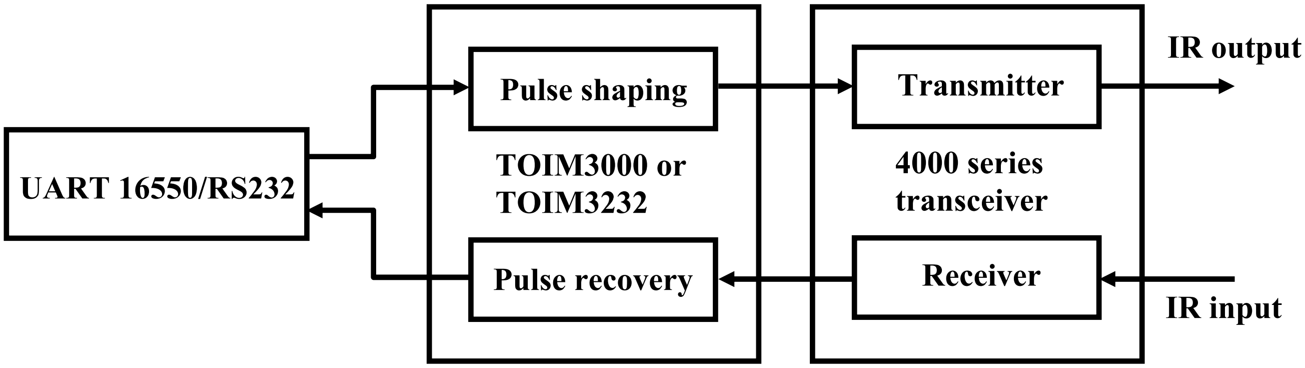


**Fig. 17.7 One end of the over all serial link.**

Please browse www.irda.org for details

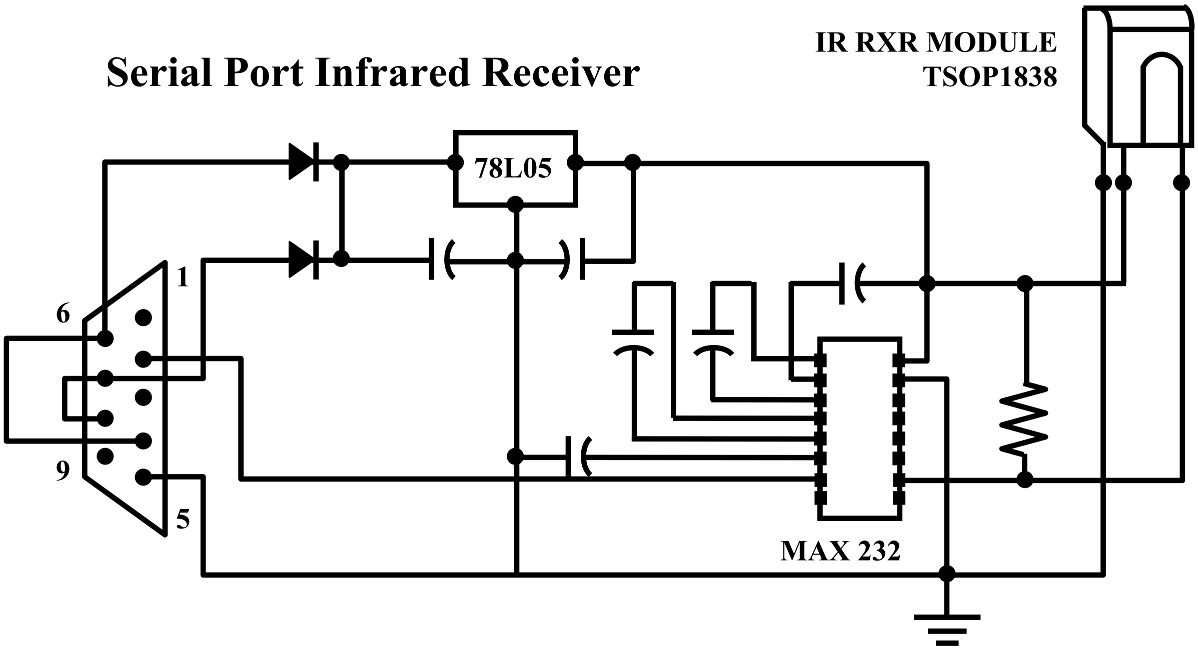


**Fig. 17.8(a) A simple circuit for Infrared interface to RS232 port.**

7805- is a voltage regulator which supplies 5V to the MAX232 the Level converter. It converts the signal which is at 5V and Ground to ±12V compatible with RS232 standard.

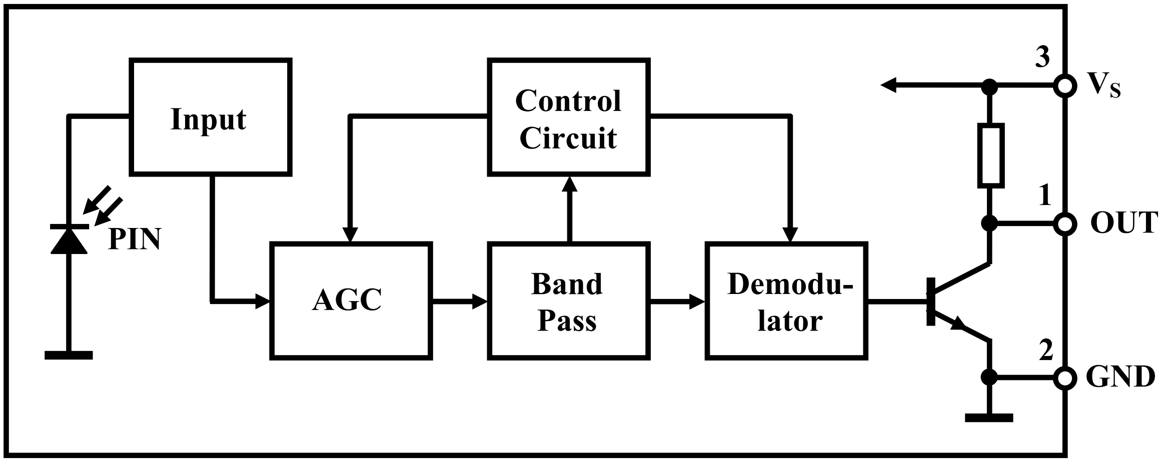
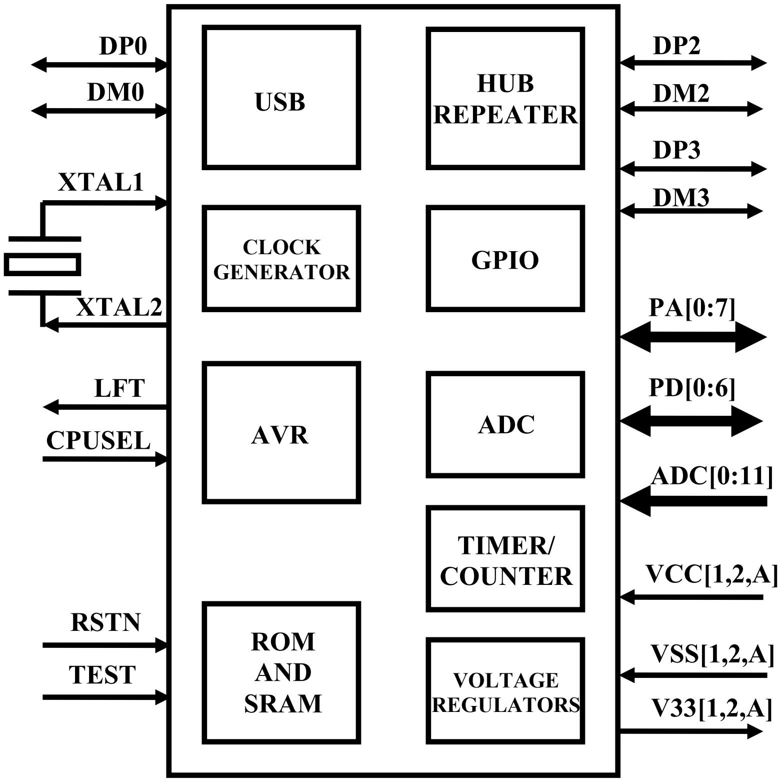


Fig. 17.8(b) The TSOP Receiver

## Question

Q.1. From the internet find out a microcontroller with in-built USB port and draw its architecture

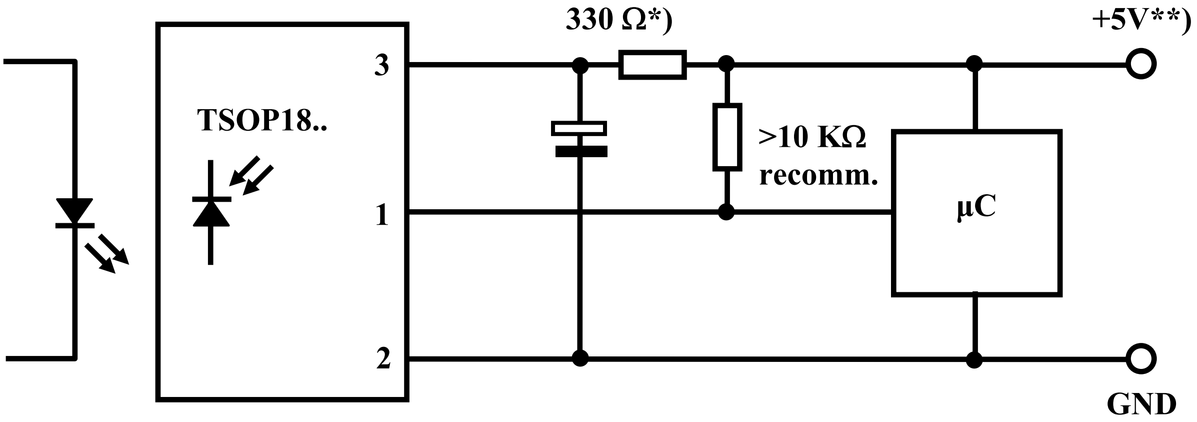
**Ans:**



The architecture of a typical microcontroller from **Atmel** with an on-chip USB controller

Q.2 Draw the circuit diagram for interfacing an IrDA receiver with a typical microcontroller

**Ans:**



A typical application circuit The Receiver Interface to a Microcontroller

## Further Reference

1. www.usb.org
2. www.irda.org