Universal Serial Bus

LEARNING OBJECTIVES

This chapter aims at describing the following:

- *Features of USB*
- *USB cable, USB host, USB peripheral, and USB hub*
- *Basics and different types of USB transfer*
- *Terms: enumeration, endpoint, functions, transaction, packet, and pipe.*
- *Basics of the USB controller Intel 8x931*

11.1 INTRODUCTION

The Universal Serial Bus, USB, is a recent development in PC hardware. Basically, USB simplifies the addition and removal of peripherals to the system. The technical details of USB are described in the following sections.

11.2 FEATURES OF USB

The USB has the following features:

(a) Simplifies the connection process and enables instantaneous addition of peripherals. Also it does not require opening of CPU, installing add-on card, selecting switch setting, and configuring the board, etc. as being done with PC bus based peripherals. It also does not require switching OFF the PC to attach or detach peripherals. The peripherals are just plugged into the USB ports provided at the rear side of the CPU. The PC automatically detects the peripherals and configures them.

(b) Establishes communication between PC and peripheral over a simple 4-wire cable.

(c) Supports three data transfer rates, 480 Mb/s (high-speed), 12 Mb/s (full-speed) and 1.5 Mb/s (low-speed).

(d) Many peripherals can be connected to one port using a special peripheral device called *USB hub*. PCs usually have only two ports. USB hubs provide additional USB ports.
(e) Allows up to 127 peripherals to be connected to a PC.
(f) Peripherals share the available bandwidth through a token-based protocol.
(g) Uses only one interrupt line from PC. All the peripherals connected to USB use the same interrupt line unlike devices connected to other interfaces.
(h) Conforms to Plug and Play specification.
(i) Distributes electrical power to many low power peripherals. The PC automatically senses the power requirement and delivers it.
(j) Allows data flow both ways between the PC and peripheral. It means that the PC can control the peripherals.
(k) Replaces serial and parallel port connectors with one standardized plug-and-play combination.
(l) Windows 98 and Windows NT have full USB support.

11.3 USB SYSTEM

USB is a fast, bidirectional, isochronous, low-cost, dynamically attachable/detachable serial interface for connecting peripherals. A USB system is the combination of three units—USB host, USB device, and USB cable as shown in Figure 11.1. USB host refers to a PC and there will be only one host in the complete USB system. USB device refers to peripherals such as scanner, printer etc. connected to the USB ports.

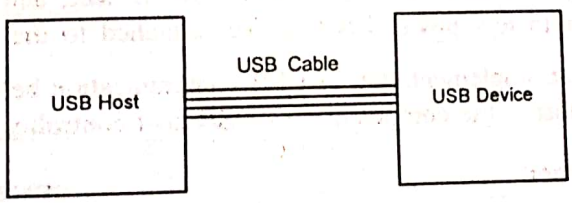


Figure 11.1 USB system.

11.3.1 USB Cable

The structure of USB cable and connectors are shown in Figure 11.2. The USB cable has four wires and carries signals on D+ and D– data lines between the host and device. It delivers power on V_{BUS} and GND lines to devices. USB allows supply voltages to be between 5.25 V and 4.2 V. The host supplies limited amount of power to bus-powered USB devices. Self-powered USB devices have their own power supply. USB devices implement power management features that allow the host to manage the power requirements of the devices. The terminations used at each end of the cable permits the detection of devices attached or detached and the identification of high, full and low-speed devices.

There are two different types of USB connectors, which are designated as type-A and type-B. PCs use type-A sockets and peripherals use type-B sockets. The USB cable connecting a host and a device has type-A plug at one end and type-B plug at another end.

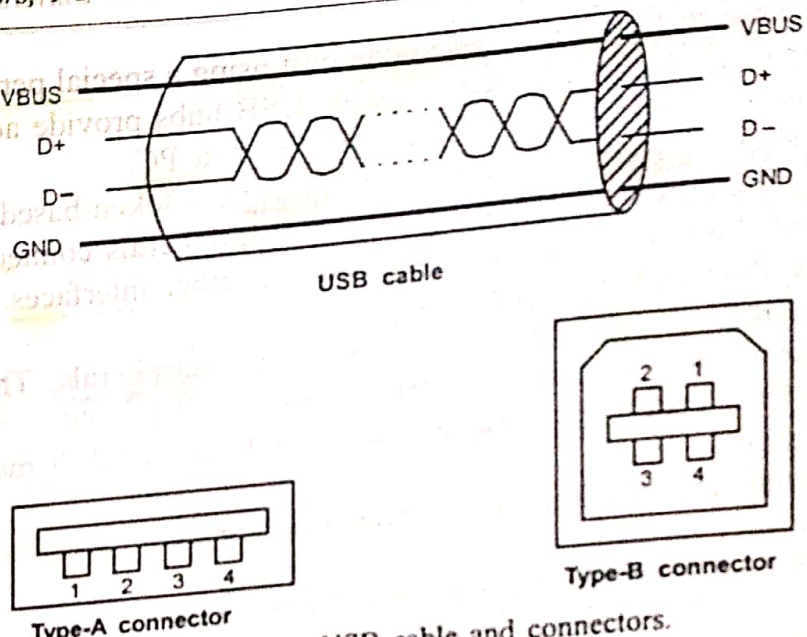


Figure 11.2 USB cable and connectors.

11.3.2 USB Host

The basic operations of a USB host are that it

(a) detects the attachment and removal of USB devices,
(b) manages flow of control information between the host and USB devices,
(c) manages flow of data between the host and USB devices,
(d) collects activity and status information of USB devices, and
(e) provides power to low power USB devices attached to the host.

Figure 11.3 shows an implementation of USB communication between the USB host and USB device. The USB host is the composition of USB host controller hardware, USB system

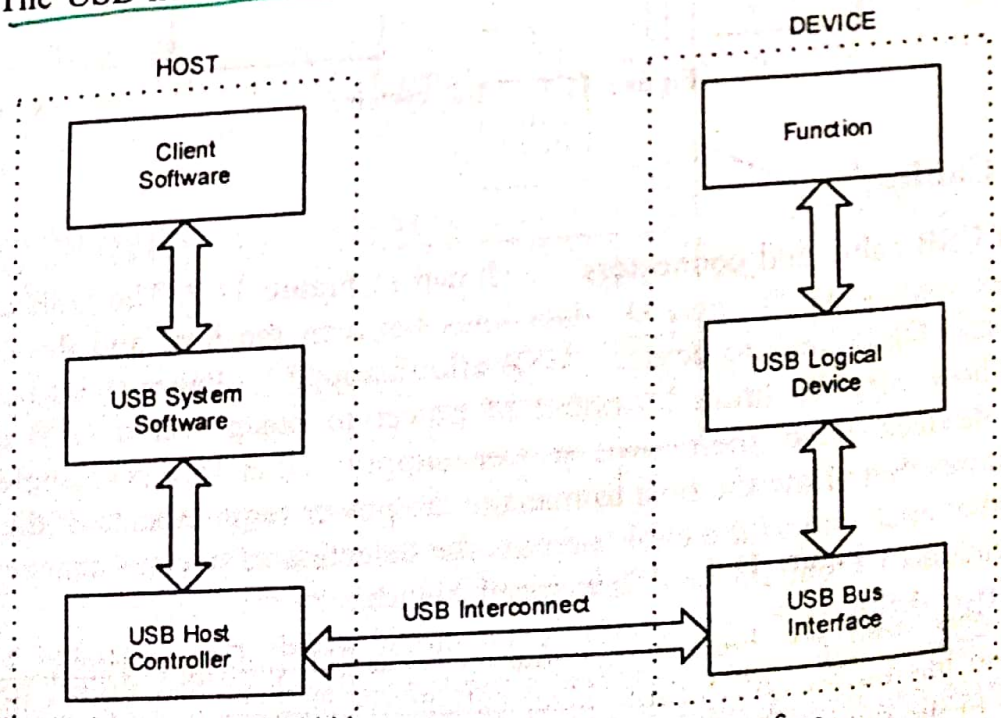


Figure 11.3 Implementation of USB interface.

software (operating system, controller driver, and USB driver) and the client software (device-driver for the USB device that remains resident and executes on the host).

11.3.3 USB Device

The basic operations of a USB device are as follows:

(a) It monitors the device address in each communication and selects itself for communication if it is the addressed device.
(b) It responds to all requests made by the host.
(c) It adds bits for checking errors for the data being sent by the device. It checks error-checking bits for the data being received and determines if any error has occurred in the transmission. It requests retransmission, if error is detected.
(d) The self-powered device manages its own power supply.

USB device is the composition of USB bus interface hardware, USB logical device and functions. A simple hardware for a USB I/O device is shown in Figure 11.4.

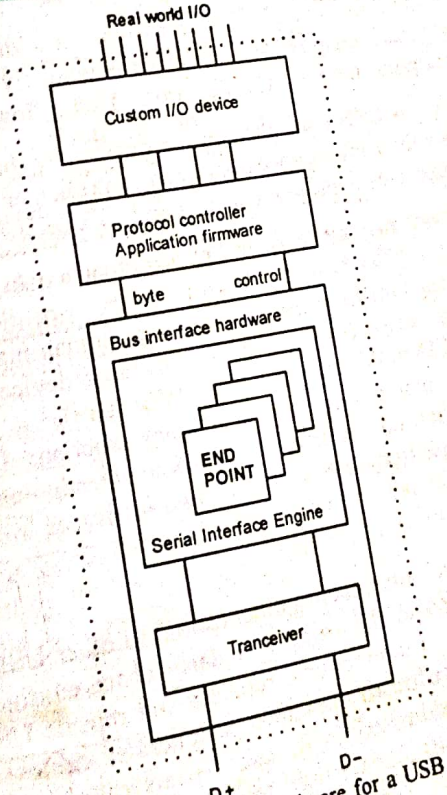


Figure 11.4 Simple hardware for a USB device.

Endpoint. The bus interface hardware contains a Serial Interface Engine (SIE) and a transceiver. It also has buffers of definite size, called *endpoints*, that store configuration and data information in the device. The bus interface hardware in the device transmits or receives

the USB data/command packets from or into the endpoints. The USB host in the USB system delivers data into an OUT endpoint and the USB device places data to be supplied to the host in an IN endpoint in the device. A USB device can have up to 16 IN and 16 OUT endpoints. Each endpoint has a unique endpoint number. The collection of endpoints is called an *interface*.

Endpoints are classified into *control endpoint* and *data endpoint*. Endpoint-0 is usually a control endpoint in the device. The control endpoint supports bi-directional flow and holds the configuration and setup information. A data endpoint supports unidirectional flow of data, i.e. a data endpoint supports either input (from device to host) or output (from host to device).

The transceiver in the bus interface hardware takes care of the electrical requirements and the SIE takes care of the bit timings of the bus.

Enumeration. The USB logical device handles routing data between the bus interface and various endpoints. The host system software gets the identity and capabilities of the device from the control endpoint and controls the operation. When the USB device is attached to the host, a series of handshaking takes place between the host system software and the control endpoint and finally the host system software assigns an address to the device. This process is called *enumeration*. Each endpoint in the device is uniquely referenced by the combination of device address, endpoint number and direction of data flow.

USB device controller. A USB controller on the device side handles many of the operations of the device. It is usually a microprocessor, microcontroller or a Digital Signal Processor in the USB device and executes a program to respond to various requests from the host.

Functions. A USB device may serve as an I/O device or a hub. A USB I/O device provides various capabilities to the USB host. USB refers to the capabilities as functions. USB I/O devices providing different functions are classified into human interface device HID (such as keyboard, mouse), printer device (such as printer), imaging device (such as scanner) or mass storage device (such as CD-ROM, floppy drive, DVD drive).

A single USB device may implement one or many functions. Devices offering more than one function have more than one interface (collection of endpoints). For example, an USB keyboard with built-in mouse/track ball will have two interfaces in a single device.

11.3.4 USB Hub

A USB hub provides additional attachment points (ports) to other USB devices in the system. USB devices can be connected to the ports in a daisy chain configuration as illustrated in Figure 11.5. A hub operates as a bidirectional repeater and repeats USB signals on upstream (towards host) cable and downstream (towards peripherals) cables. It monitors the USB signals, handles transactions addressed to it, and repeats other transactions to respective devices. USB hubs have status bits that are used to report to the host controller the attachment and removal status on one of their ports. Hubs may be cascaded up to seven levels deep. It results in connection up to 127 devices. A PC already has a root hub on its motherboard and implements two USB ports.

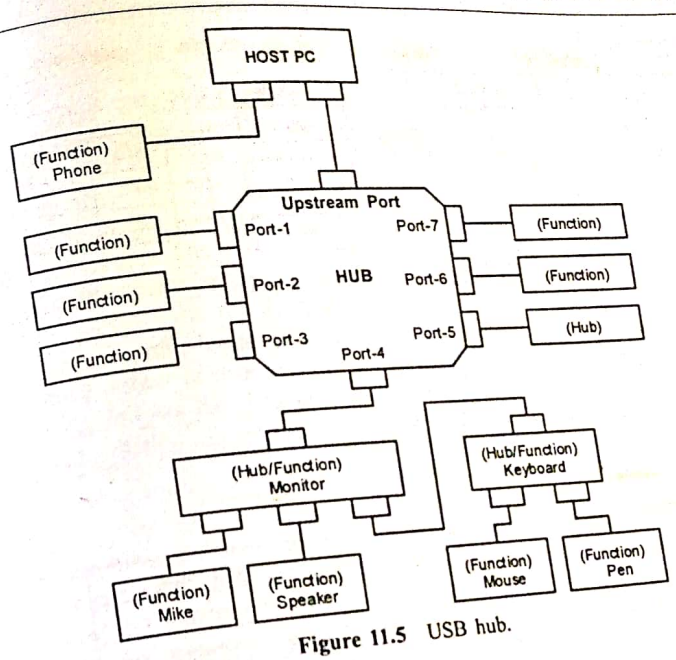


Figure 11.5 USB hub.

11.4 USB TRANSFER

USB transfer refers to exchange of control and data information between the USB host and a peripheral on the bus. The host controller initiates and manages all USB transfers. A USB transfer consists of one or more transactions and each transaction contains up to three packets. There are three different types of packets such as, token, data, and handshake packets. Each packet contains packet ID, block of information and error-checking codes (CRC) in a defined format. Figure 11.6(a) shows a typical USB transfer.

USB schedules all transactions in 1 ms time base called a frame on a full-/low-speed bus and in a 125 µs time base called a micro-frame on a high-speed bus. The transactions allowed within a frame or a micro-frame depend on the transfer types. Each frame begins with a start of frame packet followed by transactions that transfer data to or from device endpoints. It is shown in Figure 11.6(b).

11.4.1 Transaction

Transactions are the building blocks of a transfer. They are classified into three types based on the purpose and the direction of data flow. The different types are known as SETUP transaction, IN transaction, and OUT transaction. Setup transaction is used for sending control-transfer requests to a device, IN transaction is used for receiving data from a device, and OUT transaction is used for sending data to a device. Transfer of a small amount of data may be completed in just one transaction where as transfer of huge amount of data may be many transactions with part of data in each transaction. Transactions include, ID, CRC (error checking), status, and control information.

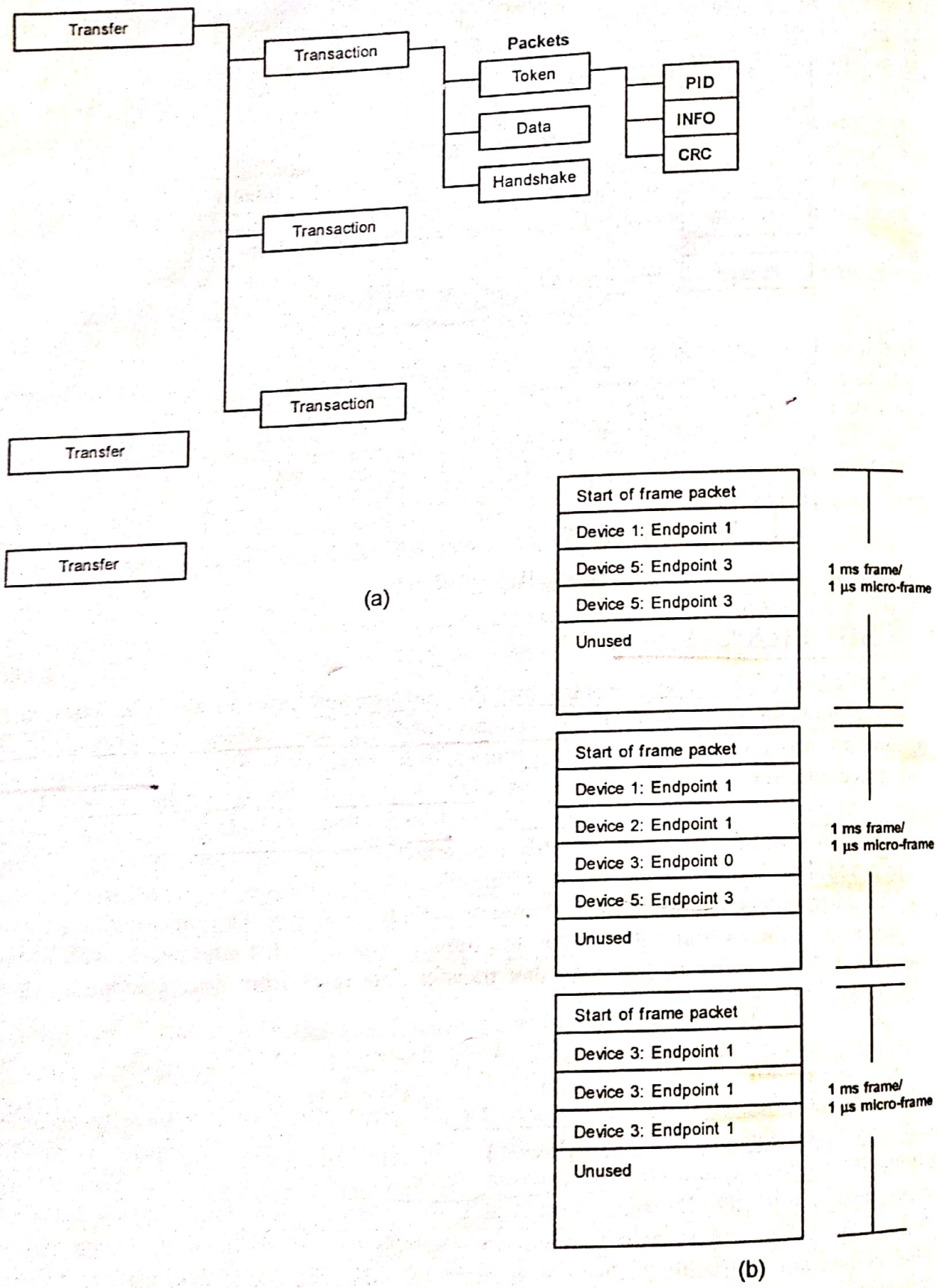


Figure 11.6 (a) A typical USB transfer, (b) frame.

11.4.2 Packet

Each transaction contains one, two or three packets of information. The packets are classified into three different types, called *token, data* and *handshake* packets. All packets contain a Packet ID, followed by the endpoint number, data, status information, or a frame number, along with error-checking bits.

Each transaction has a token packet. The host controller sends the token packet at the beginning of the transaction. The token packet describes the type and direction of transaction, the address of USB device, and endpoint number. The USB device decodes the address fields in the packet and selects itself for the transaction if it is the addressed device. In a given transaction, data may be transferred in either direction between the host and an endpoint on a device. The source (either host or device) then sends a data packet if it has any to transfer. A handshake packet follows the data packet. The destination sends the handshake packet to indicate the source whether the transfer was successful.

11.4.3 Pipe

USB transfers take place between host software and a particular endpoint on a USB device. Before a transfer occurs on the bus, both the host and the device establish a pipe. The pipe is not a physical object, but an association between the host software and a device's endpoint. The pipes are classified into two types called *stream pipe* and *message pipe*, based on whether or not the information travels in one or both directions. Only the control transfers use bidirectional message pipes and all other transfers use unidirectional stream pipes.

Pipes come into existence after the host received the configuration information when the system boots-up or when a device is attached. The pipe is removed when the device is detached. A USB device may have many pipes. As an example, a given USB device could have an endpoint that supports a pipe for transporting data to the USB device and another endpoint that supports a pipe for transporting data from the USB device.

11.4.4 Transfer Types

The USB transfer types describe the manner in which data move in the system over USB between USB host and USB peripheral. There are four basic types, such as the following:

Control transfers. The control transfer is used to configure a device at attach time and for other device-specific purposes, including control of other pipes in the device. They have high priority, automatic error protection and high data rate.

Bulk data transfers. The bulk data transfer is used for transfer of bulk data to a printer or from a scanner, where time is not a critical parameter. It implements error detection in hardware and hence the transfer is reliable. Bulk data transfer is sequential. Bandwidth taken up by bulk data transfers can vary, depending on other bus activities.

Interrupt data transfers. The interrupt data transfer is used for timely but reliable delivery of data. Data may be presented for transfer by a device at any time and is delivered by the USB at a rate no slower than is specified by the device. Interrupt data typically consists of event

notification, characters, or coordinates that are organized as one or more bytes. Devices such as keyboards and mouse generally use this method.

Isochronous data transfers. The isochronous data transfer is used for streaming real time transfers. Isochronous data is continuous and real-time in creation, delivery, and consumption. Devices transferring large amount of data at definite rate such as soundcard (which receives digitized data at definite rate and convert to analog signal) use this method. It does not implement error detection.

Low speed devices support control and interrupt transfer methods. Full-speed and high-speed devices support all the four methods. Measurement and control instrumentation applications can use either control transfer or interrupt transfer modes.

11.5 USB CONTROLLER

USB peripherals use USB controllers to handle USB communications. Many of the USB controllers on the peripheral side are based on the popular Intel 8051 microcontroller. The basics of Intel 8x931 USB peripheral controller IC are described in the following.

11.5.1 The 8x931 USB Peripheral Controller

There are two version of the 8x931 USB peripheral controller. They are 8x931HA and 8x931AA. The 8x931HA USB peripheral controller consists of both the standard MCS®51 controller and a USB module. The USB module provides USB hub and embedded functional capabilities. It contains one internal and four external downstream ports and integrates the USB transceivers, serial bus interface engine (SIE), hub interface unit (HIU), function interface unit (FIU), and transmit/receive FIFOs. The 8x931HA operates as a full-speed USB device. Root port data transfers with the PC are always at full-speed and downstream ports' data transfers are matched to USB device attached, i.e. to full-speed or low-speed.

The 8x931AA has the same features as 8x931HA but it excludes the hub interface. It is the hubless USB peripheral controller and provides only functions. It provides interfaces for one USB peripheral and three function endpoint pairs with respective transmit/receive FIFO pairs. It can operate as a full-speed (12 Mb/s) or a low-speed (1.5 Mb/s) USB device. Both the 8x931AA and 8x931HA controllers, use the standard instruction set of the MCS®51 architecture.

Architectural overview of 8x931

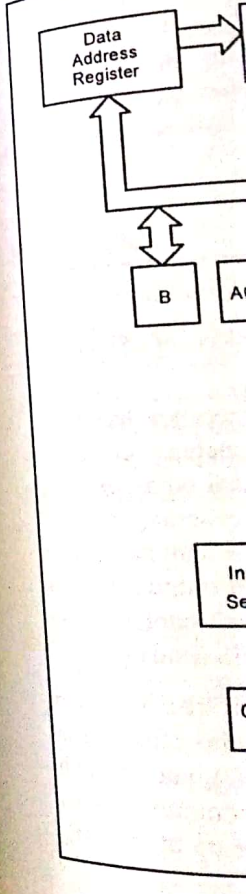
The 8x931 contains a microcontroller CPU core, a USB module, a keyboard control interface, and on-chip ROM (optional 8 KB), RAM (256 bytes), four 8-bit parallel ports, and on-chip timer/counter and serial port peripherals. The USB module operates in conjunction with the CPU and offers functional capabilities of a USB device. It supports all the four data transfer types. The functional block diagram and the blocks of USB module of 8x931 are shown in Figure 11.7.

Microcontroller core. The 8x931 has all the basic features and code compatibility with the MCS51 microcontroller. It has separate program memory and data memory space. It can

address 64 KB ROM and
located at lowest ROM a
address space. The CPU
four banks of eight gene
registers (SFRs), ACC,
transfer is selected via t
RAM locations in the d
(from 20H to 2FH) are
FFH.

The clock signal
through XTAL1 or can
crystal resonator conne
resets the 8x931 to a

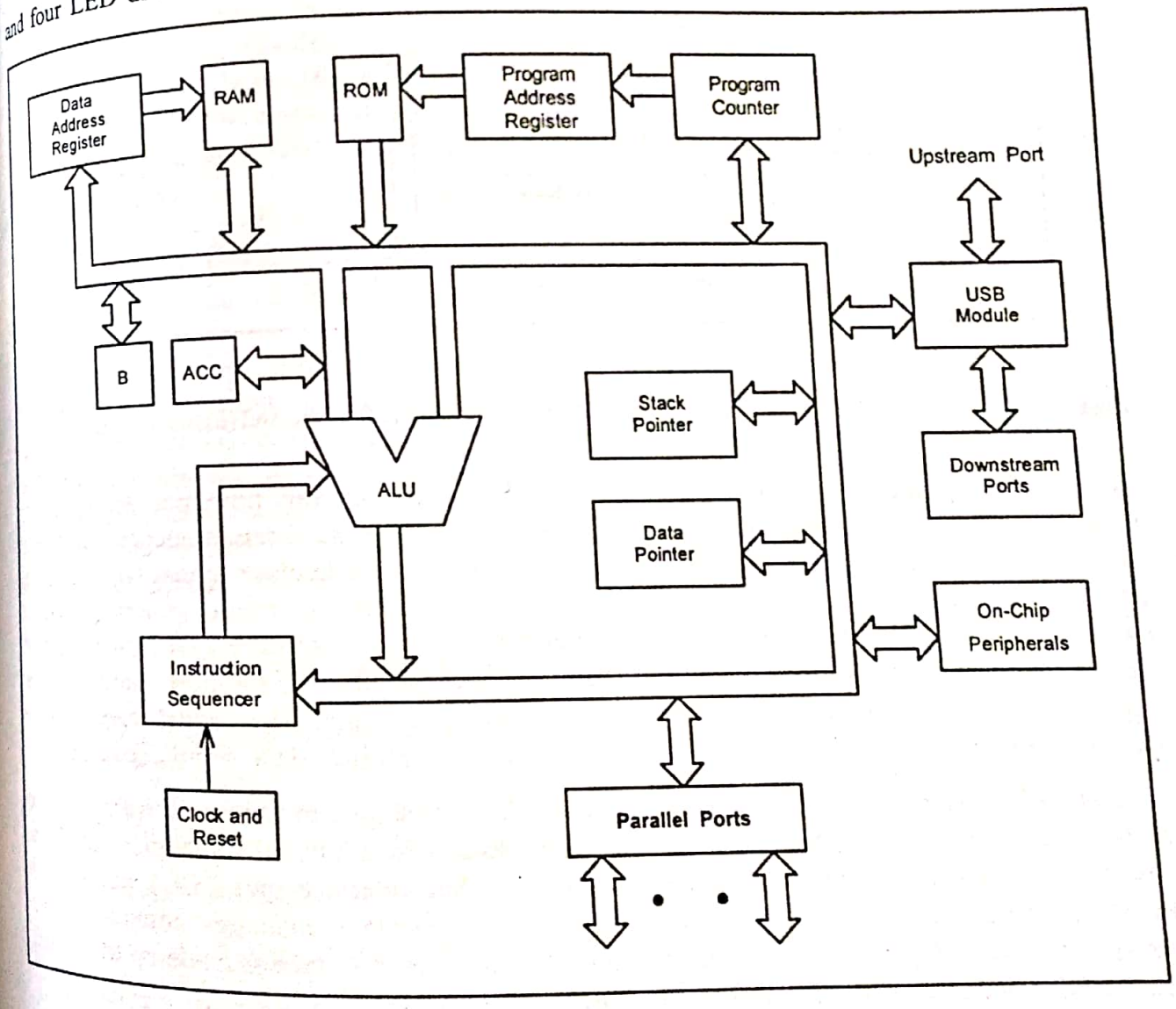
The 8x931 conta
and four LED drivers.



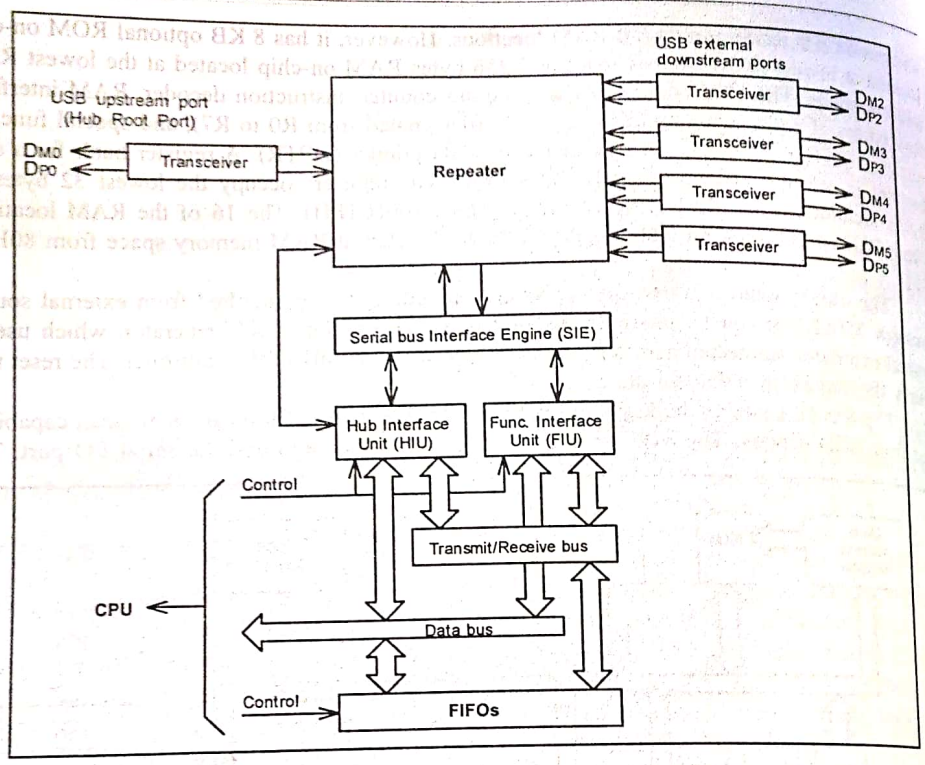
address 64 KB ROM and 64 KB RAM locations. However, it has 8 KB optional ROM on-chip located at lowest ROM address space and 256 bytes RAM on-chip located at the lowest RAM address space. The CPU contains ALU, program counter, instruction decoder, RAM interface, four banks of eight general-purpose registers (designated from R0 to R7), and special function registers (SFRs), ACC, B, stack pointer (SP), data pointer (DPTR). A register bank for a data transfer is selected via the program status word. The registers occupy the lowest 32 bytes of RAM locations in the data memory address space (00H-1FH). The 16 of the RAM locations (from 20H to 2FH) are bit addressable. SFRs are located at RAM memory space from 80H to FFH.

The clock signal required for the 8x931 operation can be supplied from external source through XTAL1 or can be internally generated by the on-chip clock generator, which uses a crystal resonator connected across the XTAL1 and XTAL2 pins of the controller. The reset unit resets the 8x931 to a known state.

The 8x931 contains the keyboard control interface with a 20-bit by 8-bit scan capability and four LED drivers. The 8x931 has on-chip timer/counter unit and the serial I/O port. The



(a)



(b)

Figure 11.7 The 8x931 USB peripheral controller—(a) functional block diagram, and (b) blocks of USB module.

two peripherals are controlled in program via SFRs. The on-chip timer/counter has three programmable 16-bit timer/counters. It can be programmed for application such as, capturing the time of an event on an external pin, outputting a programmable clock signal on an external pin, or generating a baud rate for the serial I/O port. Timer/counter events generate interrupt requests. The serial I/O port provides one synchronous and three asynchronous communication modes. The synchronous mode (mode 0) is half-duplex in which the serial port outputs a clock signal on one pin and transmits or receives data on another pin. The asynchronous modes (modes 1–3) are full-duplex in which the port sends and receives data simultaneously.

USB module. The functional blocks of the hub module are shown in Figure 11.7(b). The USB module operates in conjunction with the controller and provides hub and function capabilities. The hub repeater, the serial interface engine (SIE), the hub interface unit (HIU), and the hub FIFOs provide the hub capability. The USB function interface manages communications between the host PC and the embedded function. The function interface is made up of the SIE, the function interface unit (FIU), and the function FIFOs.

The USB module communi
(USB port 0) and communicates
downstream port (USB port 1)
transceivers for each external U
The operation of the USB
are made to/from endpoint pair
function endpoint pairs, a hub
endpoint. Hub endpoint-0 supp
transmit hub status change inf
The hub repeater is th
disconnection of devices on
downstream connectivity for
connectivity, and performs po
supports both full-speed (12 M
The SIE is shared by th
the bus. For data transfers fro
transfers to the host, it provi
the operation of the hub and
The CPU runs the program a
It also reads the receive FI
requests for the hub.

SUMMARY

The Universal Serial Bus, U
of peripherals to communic
two connectors called USB
ports may be connected t
USB peripherals use
8x931 is a UBB periphe
microcontroller and a USI

RE

1. List the importan
2. How many USB
3. What are the co
4. How many inter
5. What is the fun
6. What are the b

The USB module communicates with the host PC by means of an upstream data port (USB port 0) and communicates with the devices attached to the USB by means of an internal downstream port (USB port 1) and the four external downstream ports. It provides on-chip transceivers for each external USB port.

The operation of the USB module is controlled via the SFRs. Data transfers with the host are made to/from endpoint pairs (EPPs) on the USB module. The 8x931HA provides three function endpoint pairs, a hub control endpoint pair, and a transmit-only hub status change endpoint. Hub endpoint-0 supports only control data transfers. Hub endpoint-1 is used to transmit hub status change information to the host.

The hub repeater is the connectivity manager, which detects the connection or disconnection of devices on the external downstream ports and manages the upstream/downstream connectivity for data packets. It keeps track of hub port status, manages connectivity, and performs power management for external downstream ports. The repeater supports both full-speed (12 Mb/s) and low-speed (1.5 Mb/s) data traffic.

The SIE is shared by the hub and function interfaces. It puts data and gets data from the bus. For data transfers from the host, it provides serial to parallel conversion and for data transfers to the host, it provides parallel to serial conversion. The HIU uses SFRs to control the operation of the hub and to maintain the status of the hub and its downstream ports. The CPU runs the program associated with the operation of the hub and the function interface. It also reads the receive FIFOs, loads the transmit FIFOs, and decodes and executes USB requests for the hub.

SUMMARY

The Universal Serial Bus, USB, is a recent development in PC hardware. It allows several types of peripherals to communicate with PC and simplifies the connection process. A PC provides two connectors called USB ports for this purpose. A device called USB hub providing additional ports may be connected to a port to increase the number of ports.

USB peripherals use USB peripheral controllers and implement USB communications. Intel 8x931 is a UBB peripheral controller based on Intel 8051 microcontroller. It consists of a microcontroller and a USB module.

REVIEW QUESTIONS AND PROBLEMS

1. List the important features of USB.
2. How many USB peripherals can be attached to a USB host?
3. What are the compositions of USB host?
4. How many interrupt lines are used by the USB?
5. What is the function of a USB hub?
6. What are the basic functions of USB host and USB devices?