Quick and EZ Guide to USB

Welcome to the world of USB! USB (Universal Serial Bus) promises to solve many of the legacy problems of the past. We've all experienced the frustration when we add a modem, scanner, or any other peripheral to our system and it simply didn't work. What's worse, we don't know where to start to find the problem and agonize over having to be on hold for hours at your nearest help line. With major industry support, USB is focused on eliminating these frustrations, making computer systems simpler and easier for the average consumer. USB has such sweeping implications that it will extend past PC platforms and into many household consumer products which have an embedded microprocessor and attachment configuration.

This quick and EZ guide is intended to get you familiar with the basics of USB in 30 minutes. We cover the basics of USB topology, hardware, and software issues. We also have pointers on how to you can quickly be up and running with USB traffic in hours using Cypress's innovative architecture. Cypress's focus is to make USB design the EZiest in the market so that the peripheral manufacturer can quickly get to market with a cost effective solution. There is a glossary so that you can talk USB speak.

USB Objectives

USB brings the ultimate in simplicity when connecting peripheral devices. The user no longer needs to determine the right connection for the mouse versus the keyboard. There isn't a different cable for the monitor, printer, or high storage capacity external disk drive. The user never has to determine if the connection is a parallel or serial port. In the world of USB, there are no longer dip switches, jumpers, IRQ conflicts, and DMA conflicts. In fact, users will always be able to connect or disconnect a new device when the PC is up and running. Now that's true Plug and Play! Because of this consistency and simplicity of connection, USB will make it easy to add peripheral types that were previously thought to be too difficult for the average user.

A major objective of USB is to ensure that it is all encompassing, supporting non-real time data transfers, such as those used in text, email, and graphics, as well as time sensitive data such as audio, voice, and compressed video. Another major objective of USB is to ensure the implementation is cost effective. While a USB host controller connection will support 12 Mbps serial data rates, a lower rate at 1.5 Mbps was created so that less expensive cables and components could be used. Such low-speed devices as mice and keyboards do not require the full 12 Mbps bandwidth. However, such low-speed peripherals benefit greatly from the standardization and ease of use of USB.

USB Hardware

USB Bus Topology

The USB bus topology is known as a tiered star topology. The root of the star is the host controller. The USB host is located on the motherboard and normally is integrated into the core logic chip set. There is only one host that controls the entire system in a standard USB system. The host controller schedules transactions that are passed back and forth throughout the system to the various peripherals. If the host controller is busy, then all the attached peripherals must wait for the host controller to free itself before further communications can occur.

While a peripheral can be connected directly to the host controller port, it is more likely that a USB hub will be connected to the host controller port to expand the number of peripheral ports available to the user. Hubs permit expansion of a USB system by providing one or more additional USB ports for attaching other USB devices. Peripherals (or another hub) is attached to the hub, creating the star topology. (See *Figure 1*.)

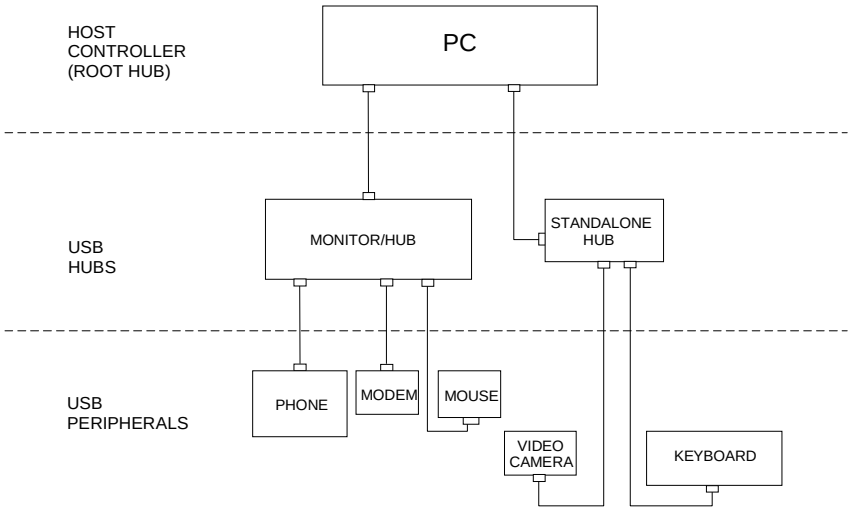

Figure 1.

Since a hub can connect with other hubs, this creates multiple tiers of peripheral stars in the bus topology. A USB hub consists of two major functions: a hub controller and repeater. The repeater function takes the communication packets and “boosts” them up to the host controller (or another hub) or back to the peripheral. USB hubs come in two types: standalone or compound device. Standalone hubs typically have 4 expansion ports. Alternatively, a hub that combines other functions such as the keyboard and monitor is known as a compound device.

The last leg of the star is the peripheral. Peripherals can have a single function, or can be combined with multiple functions. Multiple-function peripherals are known as composite devices (as opposed to the compound device for hubs). Cypress’s USB solutions are well-suited for composite peripheral devices since the configurable architecture allows the support of multiple functions. As mentioned previously, peripherals are categorized as fully rated at 12 Mbps or low-speed at 1.5 Mbps.

USB Connections

One of the great outcomes from the conversion to USB is the simplicity of connections. There are only two types of connections for all hubs and peripherals. It is literally impossible for the user to screw the connection up as the two types of connectors are physically incompatible. The connector that is nearest the host controller or the hub (the upstream side) is called a Series A connector and is shaped in a rectangular fashion (see *Figure 2*).

Figure 2.

The connector that is nearest the peripheral or downstream side is called the Series B connector and is a more square shape with two corners shaved for orientation purposes. While the host controller will only have Series A connections, hubs have both Series A and Series B connections. If the peripheral has a cable attachment it will have a Series A plug at the other end to attach to a hub or the host controller port. With the tiered star topology, the Series A connection is always at the upstream portion of the star while the Series B connection is at the downstream connection (see *Figure 3*).

For fully rated connections at 12 Mbps, the cable length can extend up to 5 meters. For low-speed peripherals, the cable is limited to 3 meters between connections. Fully rated USB cables are shielded, providing higher performance and longer length capability. Low-speed cables are unshielded, yielding lower cost.

Another aspect USB simplifies is the actual wire connection between devices. Instead of a variety of different cables which differ in the number of signal wires, all USB connections use four wires. Two are for the power supply (power and ground), and two for differential signaling (D+ and D–). A fully rated peripheral is identified by a 1.5-kΩ pull-up resistor on the D+ line. A low-speed peripheral is identified by the system by a 1.5-kΩ pull-up resistor on the D– line.

Power

All USB ports supply power for devices that are to be attached. Both hubs and peripheral may be self-powered, implementing your own power-supply, or be bus-powered. Bus-powered hubs and peripherals derive their entire power requirements from the USB cable. A fully rated port must be able to handle 500 mA of current to an attached device. Self-powered hubs can usually supply the maximum ratted power each port. However, bus-powered hubs have only the power they receive from the upstream cable, severely limiting their ability to supply full current needs of a peripheral. For devices that are attached to a bus-powered hub, the maximum amount of current they can draw is 100 mA.

Prior to being configured, USB devices must not draw more than 100 mA of current. After configuration, the peripheral can draw more current based on the configuration setting.

USB has numerous power conservation modes. One is suspend, an option whereby power consumption is curtailed through software control. There are two types of suspend: global and selective. All devices must support both types of suspend modes. Global suspend places all USB devices in a suspended state. Selective suspend places only devices which have been inactive in a suspended state. Devices enter suspend after 3 ms of no bus activity and must consume no more than 500 µA total. The suspend current of 500 µA is

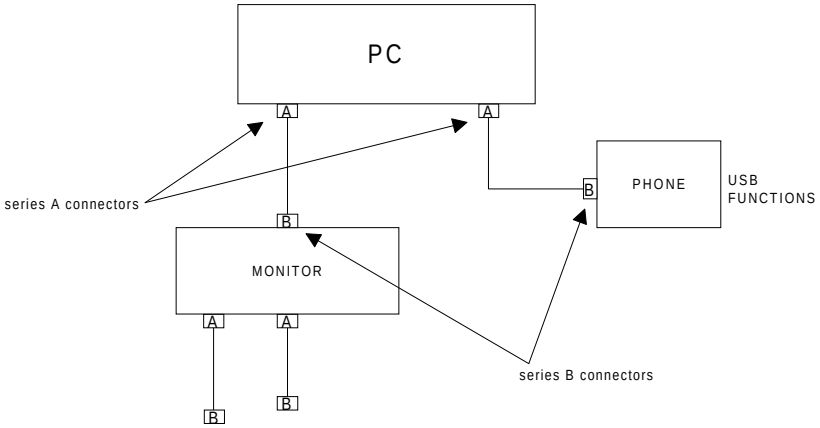

Figure 3.

more difficult to achieve than it may seem, as 200 µA is used due to the 1.5-kΩ pull-up resistor, leaving only 300 µA for the rest of the device peripheral.

The EZ-USB family is designed to operate at extremely low power; it is one of the few USB devices able to operate at 3.3V. With this low power, it is ideal for bus-powered high-speed peripherals.

USB Software

USB Communication Model

Now here is the hard part. Because USB simplifies hardware connections, the complexity shifts to a protocol that can accommodate the variety of peripheral devices. Although the bus topology of a USB system has multiple tiers of communications through hubs, the communication can be modeled from the USB host to the peripheral as a one-to-one connection. This is referred to as logical connections to the USB host.

At the center of the communication model is the USB host, which is the master of the USB system. The USB host acts as a traffic cop, controlling and scheduling all activities attached to the port. This is called a host-based communication model. The USB host is the only device that uses system resources requiring host memory locations, I/O address space, and IRQ lines. Remember, USB peripherals are no longer mapped into memory or I/O address space, nor do they use IRQ lines or DMA channels.

The USB host controller initiates transactions via its root hub or hubs. The host controller initiates a start of a frame (SOF) every 1 ms. After the SOF, communication transactions between the hubs/peripherals and the host occur within the 1 ms time frame. The host may schedule a single data transfer by the peripheral over one large block within a 1-ms frame or over several consecutive frames. The actual scheduling depends on a number of factors including: transaction traffic, type of transaction, and bandwidth requested by the peripheral.

When a USB peripheral initiates a transfer, it calls the USB system software and requests a transfer. The host will either accept the data transfer or reject the request. If the host acknowledges (ACK) the request, then an entire routine, which consist of setup, data transfer, and handshake occurs. If the host rejects the request, then a Not Acknowledgement (NAK) is sent back to the peripheral. The peripheral will then repeat the request, waiting for an acceptance from the host for the transfer request.

USB Data Types and Endpoints

In line with achieving simplicity, USB categorizes the various data types into four specific data forms used for all peripheral functions. USB can be connected to support non-time sensitive data types such as print, text, or graphics data, or real-time data such as audio, voice, and compressed video. To accommodate the various data types there are four USB data transfer types: control, isochronous, bulk and interrupt.

Control transfers are bidirectional and intended as the primary communication between host and device for configuration, command or status information. Control transfers consist of 2 or 3 stages; a setup stage, data stage (may or may not exist), and status stage. Cyclic redundancy checks (CRC) are performed on control packets. Since accuracy is important for control packets, retransmission occurs for unrecoverable errors. Because the control transfers have two or three stages, they are completed over a few USB frames. Control transfers are given a guaranteed 10% bus allocation. Control packets have a maximum length of 64 bytes.

Isochronous transfers can be unidirectional or bidirectional and are specifically targeted for streaming data such as audio or video. Since isochronous data is time sensitive, it is guaranteed access to USB with reasonable limitations on the bandwidth of the whole USB bus. Should an error occur, the peripheral device will not retry to transmit data since constant data rate is the more important than accuracy of data. Streaming data is more tolerant of errors. The maximum packet size for isochronous transfer is 1024 bytes per ms. This translates to a maximum data rate of 8.814 Mpbs.

Bulk transfers can be unidirectional or bidirectional. They are ideal for large amounts of data whose integrity must be guaranteed, but whose delivery is not time critical. Printer data or scanner data are natural candidates for transmission via a bulk transfer. A bulk transfer is designed to be a filler, claiming unused bandwidth of USB when other transfer requirements on the bus have been met. All forms of error detection and recovery are used. Maximum packet size for bulk transfers is 64 bytes.

Interrupt transfers are not interrupts in the standard sense of interrupts used by PC platforms. Instead they are used to poll devices to determine if they have data that needs to be transferred to the host. Hence, the direction of interrupt transfers are always from the USB peripheral to the host (IN only). If a device does not have data to send, then the device returns a no acknowledge (NAK) indicating no data available. Delivery is guaranteed. Maximum packet size for interrupt transfers is 64 bytes. Interrupt OUT packets have been proposed for the next version of USB (Version 1.1).

Central to the USB communication model is the abstract concept of transferring data using pipes between the host and peripherals. This pipe medium can be further distributed into even smaller pipes, with each type of data requiring a separate tiny pipe. Each tiny pipe (endpoint) carries a unique data type that is needed between the peripheral and the host. For instance, in a multimedia USB device, different endpoints would be required for voice (isochronous), data (bulk), and control information. Thus a total of 5 endpoints are required, since two endpoints are needed for bidirectional data (see *Figure 4*). All these data types must be treated differently and are separated through the use of endpoints.

The Cypress EZ-USB family has the most endpoints available in the market, supporting the maximum number of 31 endpoints allowed in the USB specification. With the vast number of endpoints, users have the flexibility to assign different buffers for each individual data stream, instead of consolidating various data streams into a single data type to be used for transmission. In addition, these endpoints can be programmed to be double-buffered, which improves transfer bandwidth in some applications.

Software Development

While the concept of USB is simple, the development of USB peripherals is not. Since the hardware scheme has been extremely simplified, the burden of complexity has moved toward software development. There will be a minimum of two software developers needed to address two areas of development required for USB: the firmware side and the driver side.

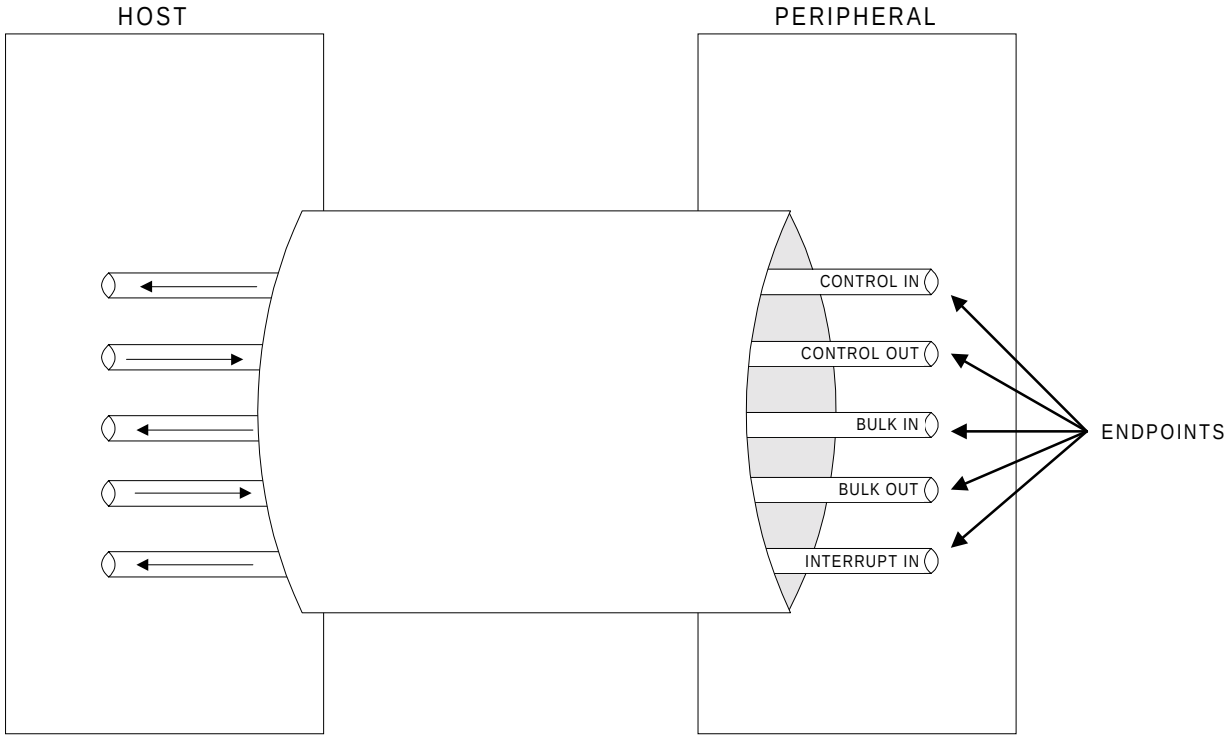

Figure 4.

Initialization

Enumeration/Renumeration

Host software is responsible for detecting and configuring the hub controller and peripheral devices. The process of identifying and assigning an address to each USB device on the hub is called device enumeration. Enumeration occurs during power-up or when the peripheral is first plugged into a USB port. Now each peripheral has a unique identifier so that the host controller can individually address each device in the star topology. After an address is assigned, the host reads and evaluates configuration information from the device descriptors passed from the device to the host. The host evaluates whether the resources requested by the device are available. If the host determines that bandwidth is available, the host assigns a configuration value to the device and is ready for use.

Renumeration is a proprietary Cypress-specific capability that allows a device to reconfigure itself after enumeration has occurred. When the driver is installed, a new set of firmware can also be loaded into the EZ-USB RAM core. This changes the “personality” of the device. Normally a device would have to be physically disconnected before another enumeration process will occur. However, with Cypress EZ-USB devices, a disconnect is simulated and another enumeration sequence occurs (see *Figure 5*). The new set of device descriptors are loaded into the USB device. This feature allows the peripheral manufacturer to provide continual updates to users via either floppy disk or through the Internet since 8051 firmware is contained in software files not in nonvolatile memory.

Firmware

The 8051 is the most popular microprocessor standard, making it a perfect fit for USB peripheral functions. With the numerous vendors supplying 8051-compatible devices and the plethora of 8051 development tools from third-party vendors, the 8051 provides a well understood and stable environment for code development. An excellent source for finding development tools for embedded processors is the Miller Freeman Directories. Their Web Site, located at www.directories.mfi.com, provides a listing of 8051 compilers, assemblers, and debuggers for development.

To efficiently develop 8051 firmware code, the designer will need a minimum of three items; 8051 assembler and/or compiler, ROM/RAM downloader, and 8051 debugger. To further decrease the learning curve and development, the designer should have access to example code, firmware library, and an USB bus analyzer. A listing below provides a quick summary and potential sources for these tools.

Item Needed	Source	Required or Optional?
8051 Assembler and/or Compiler	Keil, Tasking, or PLC	Required
ROM/RAM Downloader	Vendor	Required
8051 Debugger	Keil, Tasking, or PLC	Required
Example Code	Vendor	Optional
Firmware Library	Vendor	Optional
USB Bus Analyzer	CATC	Optional

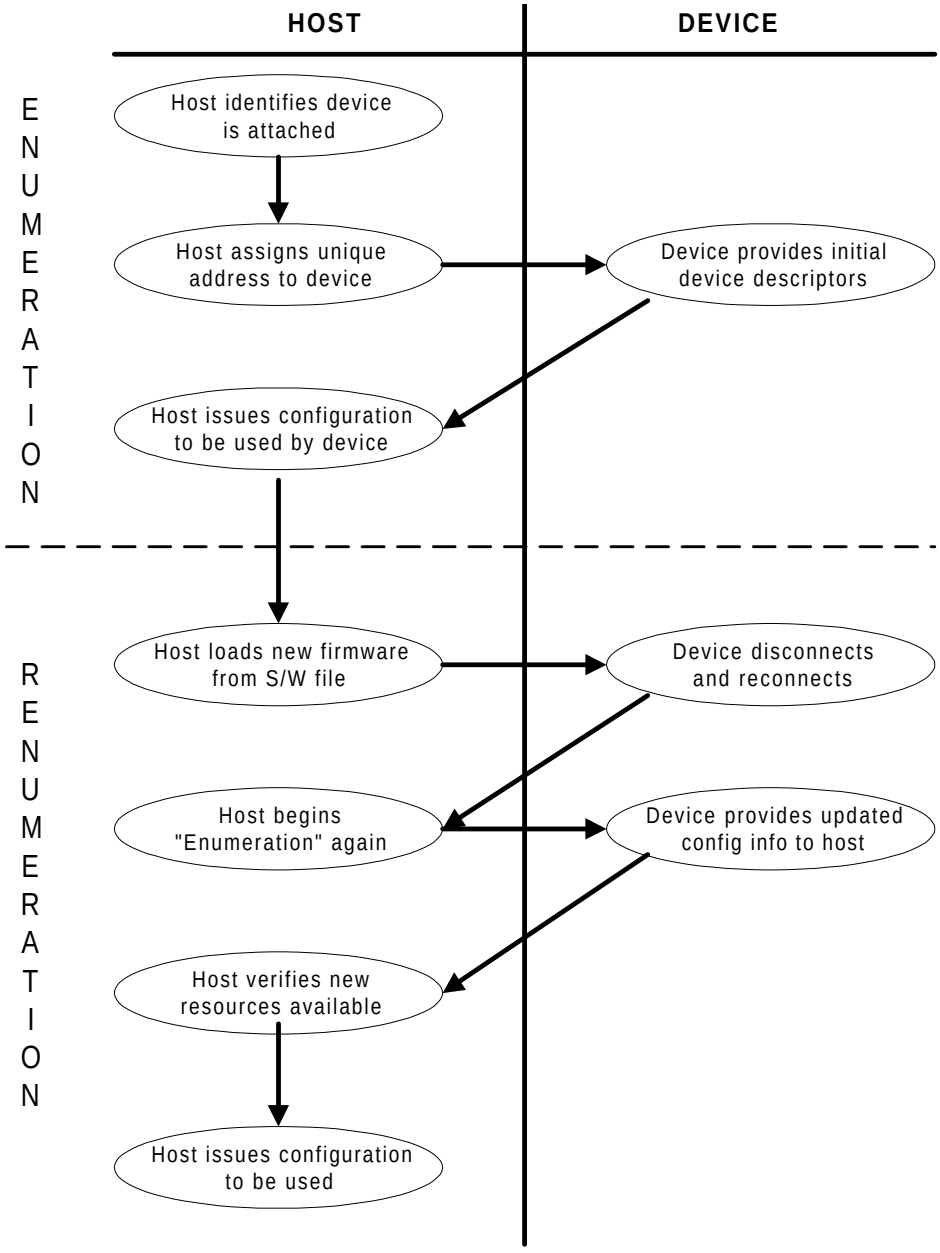

Figure 5.

One of the major features of the Cypress EZ-USB family is the significant reduction of 8051 firmware code. In most applications, endpoint zero is the most complicated to program as it is usually designated the CONTROL endpoint. Each device must implement a default CONTROL endpoint (always endpoint zero) used for configuring the device, controlling device states, and other aspects of the device's operation. Thus endpoint zero is the most complicated to program as it requires a setup stage, data stage (may or may not exist), and status stage. With other USB chip solutions, endpoint zero is handled similarly to other endpoints. The EZ-USB core provides extensive logic to simplify programming required to support endpoint zero. The use of this higher level protocol can reduce the number of standard 8051 assembly calls by as much as 80%. For example, the USB hardware core keeps track of the three phases for CONTROL (SETUP, DATA, and HANDSHAKE) not the 8051. Therefore, the CPU needs only to load an address pointer to requested data

When the designer first connects their USB prototype to the PC, most likely the system will not operate properly. When a peripheral device is initially connected to the PC, a host controller issues standard USB commands, inquiring about the resources of the peripheral. Normally, the designer cannot pretest responses to the host controller as there is no method for the designer to control commands issued by the host controller. However, with the EZ-USB Xcelerator Development Kit, a custom application developed by Cypress (EZ-USB Control Panel) provides the firmware developer the ability to manually issue host commands to the peripheral and see the

corresponding peripheral response. This allows the firmware to be coded and tested with all host controller commands before ever being coupled with a driver. With this unique tool, firmware designers can now see if their firmware has been properly coded without the necessity of a software driver. This allows more efficient and faster firmware code development.

Device Driver

If the driver is bundled with the operating system, the peripheral manufacturer does not need to develop the driver. All that is required is to follow the USB device class specifications as stated by the specific class that corresponds to the device. If the peripheral designer has capabilities that differentiate its capabilities above the standard class specifications, then a custom driver is required. Designers will need the standard development kits, DDK, from Microsoft to develop these drivers. The DDK is needed for a set of operating system libraries and Windows driver development tools. Of course, a standard C compiler for a number of different sources is needed to generate object code of your WDM driver. The driver needs to account the operating system during the development stages. It should operate in a Windows 95 operating system with an enhancement called OSR 2.1 (OEM Software Release 2.1). This is also codenamed “Detroit”. Should the software developer want device drivers focused on the next-generation operating system (Windows 98), codenamed “Memphis”, the developer will need to develop under the WDM (Windows Driver Model) stack.

An example code of drivers will accelerate the learning curve significantly. Cypress provides a set of example drivers in addition to a set of generic drivers to accelerate peripheral driver development. Of course users will still be able to utilize the standard set of Microsoft drivers that cover a family of class drivers. A listing below provides a quick summary and potential sources for tools needed for device driver development.

In addition, Cypress supplies source code for downloading 8051 firmware and device driver code during the Renumeration process. Cypress supplies this type of code to take advantage of this unique capability for “soft configuration”.

Item Needed	Source	Required or Optional?
WDM DDK	Microsoft	Required
C Compiler	Various	Required
Soft ICE (In-Circuit Emulator)	NuMega Technologies	Optional
Example Code	Vendor	Optional
Generic Compiled Drivers	Vendor	Optional
Driver Loader after Enumeration	Cypress	Optional
USB Bus Analyzer	CATC	Optional

Application/User Software

For the host application development, users only need to have a C Compiler or a Visual Basic software tool. A good tool is the Microsoft Visual C++ Developer Studio AppWizaard. In addition, application notes and applets to assist in the reprogramming and downloading new firmware would be helpful.

What Makes EZ-USB So Easy?

Cypress realizes that most peripheral manufacturers do not want to or need to become USB experts in order to take advantage of the benefits of USB. Because of proprietary capabilities and forethought to a much-improved USB architecture, Cypress can get peripheral designers up and running USB traffic within hours instead of the months from other USB chip vendors. A summary of these unique advantages is provided below.

1. Cypress has the competitive advantage of generating 8051 firmware code for the enumeration process without one line of user code written by the peripheral manufacturer. This allows the designer to analyze USB traffic, increasing the learning curve for understanding USB programming requirements. Because of its proprietary capability of “soft configuration” and Renumeration, user-developed 8051 firmware code can supplant the default 8051 firmware code residing inside the chip.
2. Cypress uses a higher level protocol for generating 8051 code simplifying firmware code by as much as 80%. With other USB chip solutions, endpoint zero is handled similarly to other endpoints. The EZ-USB core provides extensive logic to simplify programming required to support endpoint zero. For example, the USB hardware core keeps track of the three phases for CONTROL (SETUP, DATA, and HANDSHAKE) not the 8051. Therefore, the CPU needs only to load an address pointer to requested data. For further programming simplicity, the EZ-USB core provides two buffers for endpoint zero data.
3. The Cypress EZ-USB family uses an enhanced 8051 core for processing. The enhanced 8051 core runs 5 times faster than the industry-standard 8051, providing significant horsepower to handle the toughest of USB traffic and requirements. With the numerous vendors supplying 8051 compatible devices and the plethora of 8051 development tools from third-party vendors, the 8051 core provides a well understood and stable environment for code development The enhanced 8051 core contained in the EZ-USB family is binary code compatible and performs the same functions as with the industry-standard 8051. The effects of these instructions on bits, flags, and other status functions are identical to the standard 8051.
4. RAM architecture provides design and software flexibility. With 4, 8, or 16 Kbytes of SRAM, users have a complete solution that also provides a “soft” configuration capability. This “soft” configuration enables peripheral manufacturers complete flexibility with no design risks. Peripheral manufacturers can accommodate code changes due to field updates, last minute software code changes prior to production, or dynamic changes in peripheral properties as set by the user.
5. Pin-compatible and software-compatible family provides numerous options without design risk. RAM-based USB solution will allow customers to continue their design efforts even after production has begun and not worry about having adequate inventory for ROM-based devices during the critical ramp-up stage. With a growth path from 4K to 8K to 16K memory, parallel board development can occur without concern to program code expansion effecting hardware changes.

6. Tiny footprint provides more flexibility in board design. Requiring less than 1 square inch of board space for a total USB solution, the AN2131Q can minimize the precious board space of cost-sensitive peripherals.