

DESIGN GUIDE

FOR

EMBEDDED SYSTEMS

WITH REMOVABLE USB FLASH DRIVES



The power of memory. **Secured.**

PRESENTED BY:



EXPERTS IN RUGGED
INDUSTRIAL MEMORY
SOLUTIONS

Tel: +44 (0) 1794 301439

E-mail: info@nexusindustrialmemory.com

Web: nexusindustrialmemory.com

Datakey products are exclusively distributed by Nexus in Austria, Denmark, Finland, Germany, Ireland, Norway, Sweden, Switzerland and the UK.

WHITE PAPER

Embedded Systems Design Guide for Removable USB Flash Drives

ABSTRACT

Implementing an embedded system with a removable USB flash drive can be a difficult task. Knowing the key concepts and considerations beforehand can make the job much easier.

Table of Contents

Introduction	2
An Overview of the Universal Serial Bus (USB)	2
Designing the System	3
Make/Buy	3
Compliance Testing.....	3
Implementation Overview	4
USB Host Controller Choices	4
USB Software Support	6
Designing Custom USB Hardware	8
USB Connectors.....	8
Power Supplies	8
Resets & Brownout.....	10
Electromagnetic Interference (EMI)	10
Controlled Impedance	11
Example Host Controller Solutions	12
Example Bolt-On Host Controller	12
Example Built-In Host Controller	14
Implementing the Embedded Firmware	15
The USB Flash Drive File System.....	15
Handling Surprise Removal and Power Loss.....	15
File System Licensing	15
About Datakey	16
Conclusion	16
Glossary	16
References	17

INTRODUCTION

The removable USB flash drive is a key technology in our embedded computing world. Among other things, it can be used to store collected data, perform secure authenticated access and automatically update firmware. A removable USB flash drive simply takes embedded computing to the next level.

Although a removable USB flash drive is easy to use, implementing a USB flash drive can be difficult. There are many issues that an embedded designer may encounter for the first time when implementing a USB flash drive. Unfortunately, when you don't know what you don't know, your project can get in trouble very quickly.

Designers need to understand the challenges of implementing a removable USB flash drive in an embedded system before they begin development. First, let's do a quick review of USB, then walk through the development process step by step and explore some of the considerations that make this task different from many others.

AN OVERVIEW OF THE UNIVERSAL SERIAL BUS (USB)

- USB is a standard developed in the early 1990s to make it fundamentally easier to connect external devices to PCs. Its development and acceptance consolidated the number and variety of connectors to a single, multi-purpose, external peripheral bus. As of 2014, three billion USB products are shipped into the market every year. (Intel, 2014)¹
- The USB Implementers Forum (USB-IF) administers the USB standards. Developers that desire to implement products that incorporate USB are encouraged (but not required) to become members of the USB-IF. USB-IF membership requires an annual fee.
- In order to use the USB logo, a company must become a signatory to the USB Adopter's Agreement and its devices must pass compliance testing.
- USB devices are required to incorporate a unique VID (Vendor ID) and PID (Product ID) to uniquely identify them to the host so that the correct device drivers may be loaded if necessary. Vendor IDs are assigned to USB-IF members and may be purchased by non-members.
- USB devices contain strings that can be accessed by a host. Strings are stored as Unicode to support internationalization and US English is the default language. Devices are required to implement a manufacturer string that contains the name of the device manufacturer. Other optional strings include a product name string and a serial number string. Devices may also implement private strings.
- USB has several speed versions: USB 1.1 low-speed @ 1.5Mbps (LS), full-speed @ 12Mbps (FS); USB 2.0, which includes FS and adds Hi-Speed @ 480Mbps (HS); USB 3.0, which includes FS and HS and adds SuperSpeed @ 5Gbps (SS); and USB 3.1, which boosts SS to 10Gbps. Each of these speed grades indicates the bus wire speed. None of them delivers full wire speed performance because of protocol overhead and operating system latency. As a rule of thumb, they typically provide a maximum throughput of 50-70% of the wire speed.
- There are three types of nodes allowed on the bus: a host, a device and a hub. A host has receptacles for hubs and devices and initiates all USB traffic. A device plugs into hubs and hosts to receive USB traffic and respond. A hub is a special type of device that connects one upstream port to one or more downstream ports by repeating the traffic on all the ports and returning responses.

- USB supports exactly one host per bus and up to 127 devices (including hubs). Hubs may be nested up to a maximum of seven layers (including the root hub).
- USB is a hot-pluggable bus, which allows devices to be plugged or unplugged while the bus is powered. USB provides 5VDC nominal power for its devices. USB 1.1 through 2.0 provide up to 500mA of current for devices while USB 3.0+ provides up to 900mA.
- USB devices may also be implemented as USB On-The-Go (OTG) devices. The OTG spec allows a device to function as either a USB host or USB device, depending on the type of cabling with which they are connected. This capability is useful for digital cameras that need to function as a USB flash drive when connected to a computer and need to function as a host when connected to a printer.
- USB devices may implement a USB class. A USB class is a standardized, generic device function that allows devices to operate identically on multiple operating systems without the need for a user-installed driver. OS vendors must implement a class driver for each class they support.
- The Mass Storage Class (MSC) is a USB class that implements a removable storage device. MSC makes USB flash drives possible.

DESIGNING THE SYSTEM

Before designing an embedded system that incorporates a removable USB flash drive, you'll need to know a few things and make some decisions like these:

MAKE/BUY

One of the first things that you need to think about is the decision to make or buy your system. The more likely it is that someone else might want to do what you need to do, the more likely it is that you can buy something to do it. It's a sad day for the designer who built custom hardware to learn after the project is complete that what he built could have been purchased. So take some time to research single-board computers and system-on-module solutions and compare their costs to that of custom hardware at your expected production quantities. Be sure to account for your development costs and consider risk, time-to-market and opportunity costs.

COMPLIANCE TESTING

USB compliance testing is a series of tests established by the USB-IF to determine whether a USB product meets the USB specification requirements. Although USB compliance testing is not mandatory, it is likely in your customers' best interest. Products that pass compliance testing are not only assured to work with other USB products, they are compliance listed by the USB-IF and are eligible to license the USB Logo for use on the product packaging.

How does compliance testing work? Compliance testing may be performed either at a USB-IF-sponsored compliance workshop (aka 'PlugFest') or at an independent test lab. Additionally, you can perform some of the preliminary testing yourself using free software tools provided by the USB-IF to help get you ready for compliance testing. You can [download the tools here](#)².

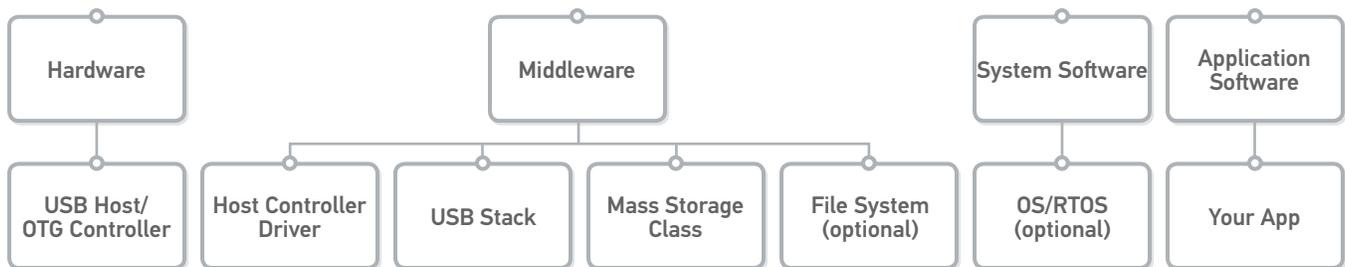
For more information about USB compliance testing, visit the [USB website here](#)³.

IMPLEMENTATION OVERVIEW

The USB specification dictates that the only way to read from or write to a USB flash drive is through a USB host. A USB host is specifically designed for other USB devices to be plugged into it and typically has one or more type A receptacle connectors for that purpose. If you need to implement a removable USB flash drive in an embedded system, you must implement a USB host.

Not all embedded USB hosts are created equally. In order for a USB host to be able to communicate with a class of devices, the designer must select the device drivers, class drivers and protocol stacks for the devices and services that the host will support. For instance, to support a USB flash drive, a USB host should include the components shown in **Figure 1**.

Figure 1: USB Flash Drive Implementation Block Diagram



As you can see in the preceding figure, an embedded host for a USB flash drive is a complex combination of hardware and software that works together as a system.

A key component of this system is software that we call ‘middleware.’ Middleware is a collection of software consisting of drivers, protocol stacks and class drivers that make it possible to communicate with the USB flash drive via the USB host controller. For embedded systems with no operating system, it may also include a file system to properly format the data written to the USB flash drive so that other computers can read it as if it were a disk drive. We’ll talk more about middleware later, but we mention it now because for our system-level discussion it’s important to know that middleware is a required part of our system. It is also important to realize that middleware imposes additional requirements on the system above and beyond the USB host controller hardware.

Now with the basic requirements understood, let’s take a look at the hardware and software options available to do the job.

USB HOST CONTROLLER CHOICES

Let’s say that you’re upgrading an existing custom embedded system that does not support USB, so you need to add a USB host controller to your embedded system. You can either add (or bolt on) a separate USB host controller chip to your existing system or switch to a new microcontroller that has a USB host controller built-in.

ADDING A 'BOLT-ON' HOST CONTROLLER

Adding what we're calling a 'bolt-on' USB host controller to your system comes with some benefits and costs. One benefit of sticking with your old processor is that you can use your existing development tools. Costs to your system include the USB host controller, any new system resources such as RAM (for code space required by the middleware) and the OS or RTOS necessary to run it.

USB host controllers can be divided into three types as shown in **Table 1** and described below:

1. **'Standardized' USB Host Controllers:** These host controllers implement a standard software interface and typically require an OS or an RTOS and middleware, which will require extra RAM and flash storage.
2. **Proprietary USB Host Controllers:** These host controllers implement a vendor-specific software interface and require the use of vendor-supplied or user-written libraries. These also may require middleware, but the middleware must include support for the required chip because the interface is proprietary.
3. **USB Host Peripherals:** These chips are actually self-contained embedded systems that contain a USB host controller and contain firmware that includes middleware, mass storage class and file system in a single chip. They are easy to use, and in keeping with their minimalistic system footprint, their feature set and performance are limited but quite capable for a wide variety of applications.

Table 1: Partial Listing of USB Host Controllers Chips

Vendor	Type	Part Number(s)	Bus	References
NXP	Standardized	SAF1562HL	PCI	NXP Interface Products ⁴
	Standardized	SAF1760BE	Generic	
	Standardized	SAF1761BE	Generic	
Cypress	Proprietary	SL811HS	Generic	Cypress USB Hosts ⁵
	Proprietary	CYC67200	Generic	
	Proprietary	CYC67300	Generic	
FTDI	Peripheral	VNC1L	Parallel,	FTDI Products ⁶
	Peripheral	VNC2	UART, SPI	
	Proprietary	FT313H	Parallel	

A deciding factor between these types of chips is whether or not your system has sufficient resources available to support an OS or RTOS and middleware. If your system requires high-performance such as Hi-Speed or SuperSpeed and can afford the cost of middleware and an OS or RTOS, perhaps a standardized host controller is a good choice. For a system that needs higher performance and can support middleware but not an OS or RTOS, a proprietary host controller may be a better fit. If your system is resource constrained and moderate performance is acceptable, a USB host peripheral may be the better choice since it is ready-made and puts a lighter load on the existing embedded system.

BUILT-IN USB HOST CONTROLLER

If the ‘bolt-on’ host controller doesn’t appeal to you, you may want to consider designing your system using a processor with a built-in USB host controller. Normally, this would not be a viable option for upgrading an existing system because it means essentially replacing the whole system, but recent years have seen a flood of low-cost, low-power, high-performance microcontrollers. Because these new chips include a suite of high-value peripherals, many of them include built-in USB host support with free middleware and RTOS support included as part of the design software for the chip. Factory-provided firmware is designed from the start to work with the chip, so integration will be easier. Another benefit is that there are hundreds of other developers using the same system, which makes it possible to get free support from other like-minded developers using the same system and tools. **Table 2** lists some of the many microcontrollers that have built-in USB OTG or Host support.

Table 2: Partial Listing of Microcontrollers with Built-In Host Support

Supplier	Family	Architecture	USB OTG or Host Support	References
NTI	Tiva	ARM	FS/HS Device & OTG	Tiva C Series MCU Page ⁷
TI	Hercules	ARM	LS/FS OHCI, FS Device	TI Safety Microcontrollers ⁸
TI	Sitara	ARM	FS/HS Host & Device	TI Sitara Microprocessors ⁹
TI	OMAP	ARM	FS/HS Host & Device	TI OMAP Application Processors ¹⁰
Freescale	Kinetis	ARM	FS/HS Host & Device	Freescale Kinetis Microcontrollers ¹¹
Freescale	Vybrid	ARM	FS/HS Host & Device	Freescale Vybrid Microprocessors ¹²
Freescale	i.MX	ARM	FS/HS Host & Device	Freescale i.MX Series Microprocessors ¹³
Atmel	AVR	AVR	FS OTG & Device	Atmel AVR Microcontrollers ¹⁴
Atmel	SAM	ARM	FS Host & Device	Atmel ARM Microcontrollers ¹⁵
NXP	LPC	ARM	FS/HS Host & Device	NXP LPC Microcontrollers ¹⁶
Microchip	PIC	PIC	FS/HS Host & Device	Microchip PIC Microcontrollers ¹⁷

USB SOFTWARE SUPPORT OPERATING SYSTEM

One option to get USB software support for your system is to use an operating system. From the software point of view, operating systems are an attractive option in that they include device drivers for most host controllers, the USB stack, a mass storage class driver and a file system as part of the OS. They also come with very powerful development tools. The down side is that an OS requires considerable system resources, large amounts of flash and RAM, and it boots slowly. Commercial operating systems may have a per-unit license fee, whereas open-source operating systems typically do not. **Table 3** lists a few embedded operating systems that include removable USB flash drive support.

Table 3: Partial Listing of Embedded Operating Systems

Name	Vendor	Architecture	License	References
Linux	N/A	Many	GNU	www.linux.org
Windows Embedded	Microsoft	x86	Windows EULA	Windows Embedded ¹⁸
NetBSD	N/A	Many	Berkeley	www.netbsd.org

THIRD-PARTY RTOS + MIDDLEWARE

Some applications require the use of an RTOS (Real Time Operating System) to meet determinism, throughput and/or response time requirements. Most RTOS products offer USB support through middleware that is ported to a development tool suite. Unfortunately, these middleware products tend to be proprietary, usually do not interoperate with tools from other suppliers and can cost several thousand dollars to license. **Table 4** lists some popular commercial RTOS solutions.

Table 4: Partial List of Third-Party RTOS + Middleware Solutions

Name	Vendor	Components	References
emWare	Segger	embOS, emboOS/IP, emWin, emFile, emUSB Device, emUSB Host, emLib	emUSB Embedded Host ¹⁹
MDK	Keil	RTX, CAN, Flash File System, USB Host, USB Device, TCP/IP, GUI	Keil ARM Tools ²⁰
INTEGRITY	Green Hills	Networking, Cryptography, File System, USB, Graphics	Green Hills Tools ²¹
MQX	Freescale	RTOS, RTCS, MFS, USB host/device	Freescale MQX Tools ²²
uC/OS	Micrium	RTOS, TCP/IP, USB host/device, CAN, Modbus, Bluetooth, Filesystem, GUI	www.micrium.com

MANUFACTURER RTOS + MIDDLEWARE

If the 'built-in' USB host controller option looks good to you, you may be able to get a special deal on RTOS and middleware. As mentioned earlier, microcontroller vendors are eager to encourage you to switch to their new products and some of them are including free RTOS and middleware to help you decide. **Table 5** shows which RTOS + Middleware resources are available for a variety of processors.

Table 5: Partial List of Vendor-Supplied RTOS + Middleware Solutions

Vendor	Family	Architecture	References
TI	Tiva	ARM	TivaWare Hardware / Software Resources Page ²³
Atmel	AVR	AVR/AVR32	ASF USB Host Stack Application Note ²⁴
Freescale	All	All	MQX Software Solutions ²⁵

BARE METAL

If you use a USB host peripheral for your host controller solution, you have the option of implementing what's known as a 'bare metal' embedded system. A bare metal embedded system has neither an OS nor an RTOS; it's just a program running on a microcontroller. Modern bare metal embedded systems are usually written in C and send commands to the USB host peripheral directly over standard peripheral interfaces. Bare metal systems tend to support just the bare essentials due to their minimalistic implementation, but the functions they implement tend to be extremely efficient because they pay no OS or RTOS overhead penalty in code space, RAM or computational resources.

DESIGNING CUSTOM USB HARDWARE

If you choose to design your own custom USB hardware, you need to consider a number of things. Here are some helpful tips to keep in mind:

USB CONNECTORS

Consumer-grade USB flash drives use the USB type A plug and plug into a type A receptacle. The signal names for a type A receptacle are shown in **Figure 2**. The type A USB receptacle is a low-cost, consumer-grade connector rated to last a minimum of 1,500 plug-unplug cycles. Although the USB specification includes specs for connectors, their use is not mandated. Special needs, such as high-reliability applications, may require different connectors to meet their requirements. For instance, type A USB connectors are not waterproof, so third parties have developed variants of standard USB connectors to meet the needs of their customers. **Figure 3** below shows a panel-mount receptacle that is capable of plugging into a standard USB type A receptacle, but it offers extremely high cycle life and is available in IP65 and IP67 rated versions. When you evaluate your embedded system requirements, make sure that you specify the right connector to meet your needs.

Figure 2: Type A USB Receptacle

Pin	Name
1	VBUS
2	D-
3	D+
4	GND

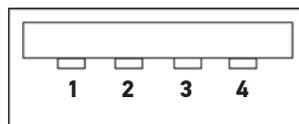


Figure 3: Datakey UR Series Receptacle and RUGGEDrive™ Flash Drive

Figure 3 shows a Datakey UR series receptacle. It connects to standard USB ports, but works with Datakey RUGGEDrive™ flash drives. These receptacles offer high cycle life (50,000 cycles versus only 1,500 cycles for a typical USB type A connector) and are available in IP-rated versions.



POWER SUPPLIES

The most important circuit of any electronic product is the power supply. Just as we need clean water to live, electronic devices need clean power to operate. A properly functioning power supply is an unsung hero, but even the best system will fail if its power supply goes bad. For embedded USB hosts, the power supply is doubly important because it not only powers the host, it also powers all the devices connected to it.

USB hosts must provide regulated, overcurrent-protected power on the VBUS line at 4.75V to 5.0V. Current is specified in units of 'unit load' where a unit load is 100mA. Externally powered Full and Hi-Speed hosts must supply up to five unit loads of power to devices, and SuperSpeed hosts must supply up to nine unit loads. Battery-powered hosts may supply as little as one unit load.

POWER REQUESTS

When a USB device is first plugged into a system, it should consume a maximum of one unit load (100mA). The host queries the device to find out how much power it needs to operate. Then, the device indicates the amount of current that

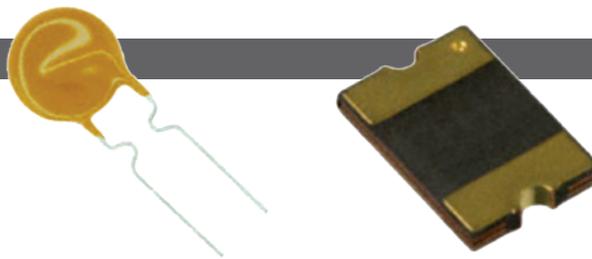
it intends to consume. If the amount of current that your host can provide is limited, it may deny a request for current by failing to send a configuration message to the device. If the device does not receive a configuration message, it simply will not connect. A user who observes this will not necessarily be able to tell that the connection failed. Therefore, it is recommended that a USB host notify the user when a power request is denied so the user will know what happened and potentially take corrective action.

The standard practice for externally powered embedded USB hosts is to connect VBUS to the +5V rail through an overcurrent protection device and acknowledge all requests for power. If your system is battery-powered or has stringent power consumption requirements, your system design should probably take a more rigorous approach to power management. For comprehensive information about USB Power Management, consult sections 7.2 and 9.2.5 of the USB 2.0 specification²⁶ and section 2.1.1 of the USB 3.0 On-The-Go and Embedded Host Supplement²⁷.

OVERCURRENT PROTECTION

The USB specification requires overcurrent protection for all USB hosts, embedded or otherwise. This requirement is imposed for safety reasons to handle the case where a device or a cable is plugged into the host that shorts the power lead (VBUS) to ground. Without current limiting, this situation could lead to a fire if the power supply is powerful enough.

Figure 4: PTC Resettable Fuses



The requirement states that overcurrent limiting must be resettable without user mechanical intervention, which rules out a fuse. PC hosts usually incorporate a root hub chip with current sensing capability and external VBUS switches that turn the power off to devices that draw too much current. Embedded systems can do this too, but to keep costs low most embedded hosts incorporate a PTC resettable fuse, like those shown in **Figure 4**, between the power source and the VBUS line. A PTC fuse (aka, polyfuse or polyswitch) is a solid-state, self-resetting circuit breaker device that goes to a high resistance state when the current through it exceeds a certain amount then returns to a low resistance state when the current is removed.

INRUSH CURRENT PROTECTION

When a device connects to a USB host, the host must charge up the bypass capacitance in the device while also powering the device. This scenario results in large inrush currents that, if left unmitigated, can lead to system instability caused by power supply voltage droops that result from high inrush currents.

One way for USB hosts to handle high device inrush currents is to have about 120 μ F of capacitance on the host side of VBUS. Large bulk capacitance serves as a charge reservoir to supply current during device plug-in that may normally swamp the host power supplies. Another mitigation to be used in combination with a large bulk capacitance is to provide a low value series resistor between the host bulk capacitance and the device VBUS. This technique limits the maximum amount of inrush current that a device can be supplied. It will also introduce a constant voltage drop across the series resistor during normal operation. The voltage drop can be ignored as long as the output voltage of the host port is between 4.75 to 5.0V.

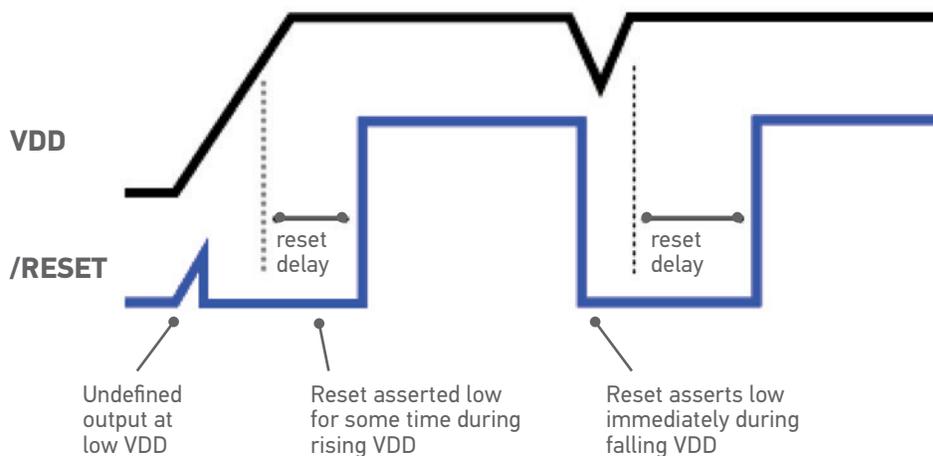
RESETS & BROWNOUT

Every microcontroller-based system should incorporate reset and brownout control. Reset control can be as simple as an RC circuit between power and ground to gently bring your microcontroller out of reset once the power supply is up and stable.

Did you know that without a reset controller, your microcontroller keeps right on running even after the power has been turned off? That's right. After the power supply has been turned off and while the supply voltage is bleeding down from its nominal voltage toward ground, your microcontroller still has enough voltage at its power supply pins to keep running. Some parts of the microcontroller don't use much power and they keep working for a long time while other more power-hungry parts of the microcontroller stop working relatively quickly. This phenomenon is known as brownout. Brownout is the condition where parts of the system work and other parts shut down as the power supply ramps up or down. This can be a big problem for flash-based microcontrollers because during brownout the microcontroller can execute random code that is capable of corrupting flash memory. **Figure 5** illustrates how a supervisor protects your microcontroller by resetting it when the input voltage is too low.

Figure 5: Brownout Due to Power Supply Glitch²⁸

HOW A SUPERVISOR PROTECTS AGAINST BROWNOUT



One solution for brownout is a supervisor or reset controller. A reset controller is a simple IC that controls reset based on the power supply voltage rather than time. A number of companies provide reset controllers that put an end to brownout. The diagram above in **Figure 5** shows how Texas Instruments' Supervisor and Reset ICs operate. Be sure that you design one into your system.

ELECTROMAGNETIC INTERFERENCE (EMI)

There are a variety of regulations (e.g., FCC, CE, etc.) that require electronic products to minimize EMI radiation and susceptibility. EMI can be radiated from the data signals or from the system ground connections between your system and the outside world. Here are some quick tips to help minimize radiated EMI:

1. Use power and ground planes on your PCBs. Planes provide the lowest inductance path for power and ground, and lower EMI.
2. Use grounded polygon pours on the top and bottom layers to provide a return path for unbalanced return currents.
3. Properly bypass all ICs with low ESR ceramic capacitors.
4. Use an EMI filter on the D+/D- lines of your USB connections. Make sure that the filter is designed for the speed of your USB connection.
5. Use ferrite beads on your VBUS and GND lines to filter noise generated by your system from getting into the power leads of the USB cable.
6. Do not connect the USB shield directly to ground. Tie it to ground through a 1 M Ω resistor in parallel with a 4,700 pF ceramic capacitor. This should provide sufficient filtering to avoid radiating noise through the shield.
7. Use a metal or metallized plastic enclosure to act as a 'Faraday cage' to enclose EMI radiated by your electronics.
8. If you use a metal or metallized plastic enclosure, make sure that the shield connections on your USB connector are electrically connected to your enclosure. This will help reduce radiated emissions.

CONTROLLED IMPEDANCE

USB signals are differential and must be routed using a 90 Ω +/- 10% differential impedance design. The signals that must be controlled are the D+ and D- lines between the connector and your host controller. Properly controlled impedance design is a good idea for low and full-speed designs, but it's absolutely required for Hi-Speed and SuperSpeed designs. Without it, the circuits simply won't work. If your design will be compliance tested, it will undergo TDR (Time-Domain Reflectometer) impedance measurement to assure that your design has the correct impedance.

Controlled impedance design is relatively straightforward. The general principle is that the D- and D+ signals should be routed side by side using a fixed spacing from beginning to end and over a continuous ground plane for the whole route. Try to keep D+ and D- on the same layer and the same length. The routing topology may be either microstrip or stripline as shown in **Figure 6**.

Figure 6: PCB Layout of Differential Pairs



Cypress Semiconductor has prepared a great [application note for Hi-Speed USB layout](#) that is definitely worth reading²⁹. Another useful tool for controlled impedance PCB design is an impedance calculator program such as the [Saturn PCB Toolkit](#)³⁰.

Another technique to get your PCB built with the proper controlled impedance USB D+/D- traces is to specify the controlled impedance signals to your PCB manufacturer as part of your fabrication instructions. Most PCB manufacturers have the software necessary to calculate impedance precisely and they can test the impedance of your PCB traces using a TDR. This is a relatively common practice and is not a big cost adder, so it is a great way to get controlled impedance PCBs with very low risk. Design the differential traces using the application note and the results of the impedance calculator. Next, add fabrication instructions to your PCB drawing instructing the manufacturer to build for 90 ohms differential impedance for USB traces. You should clearly mark the affected traces in your drawings so the manufacturer will tune the correct lines.

In summary, you can greatly simplify the task of achieving the correct controlled impedance on your PCB by using an impedance calculator and/or enlisting the help of your PCB supplier.

PCB TRACE WIDTH SIZING

PCB trace widths for signal lines are generally limited by the minimum trace geometry for the PCB process. But for PCB power traces, the trace width can have a significant impact on how well the circuit performs. The amount of current that a PCB trace can carry depends on the acceptable amount of temperature rise. Temperature rise is determined by the thickness of the trace copper and the width of the trace. An easy way to size your PCB traces is to use a calculator like the one [found here](#)³¹. Calculating PCB power trace width may seem like overkill, but it's better than getting surprised later on.

EXAMPLE HOST CONTROLLER SOLUTIONS

The next two sections showcase two types of USB host controller solutions in a bit more detail. The first will use a 'bolt-on' host controller and the second uses a microcontroller with a built-in USB host controller.

EXAMPLE BOLT-ON HOST CONTROLLER

For the 'Bolt-on' host controller example circuit, we'll take a look at the FTDI V2DIP1. The module is a little smaller than a stick of chewing gum, yet it contains all the circuitry needed for a USB host controller including the connector as shown in **Figure 7**.

Figure 7: V2DIP1 USB Host Module



The module is based on the Vinculum II, a second-generation, full-speed USB host system-on-a-chip that contains the host controller, middleware and FAT file system firmware all in a single chip.

From the hardware perspective, you may design the module or the chip into your circuit. Either way, you need to provide power and wire the UART, parallel FIFO or SPI interface to your microprocessor. Even though there is only one USB host connector, the module actually supports two USB host ports. All you need to do is wire up the second port if you need it. The schematic for the V2DIP1 is found in **Figure 8**.

USB host controller functions are executed when your microcontroller sends high-level commands to the Vinculum II over the microcontroller interface. For example, in order to read the directory of a USB flash drive, you send the 'DIR' command over the microcontroller interface and the module sends a list of filenames in response. The command protocol supports either ASCII or binary commands.

All in all, a USB host peripheral is a clever solution for applications where resources are limited and the required data rates are relatively low.

Figure 8: V2DIP1 Schematic

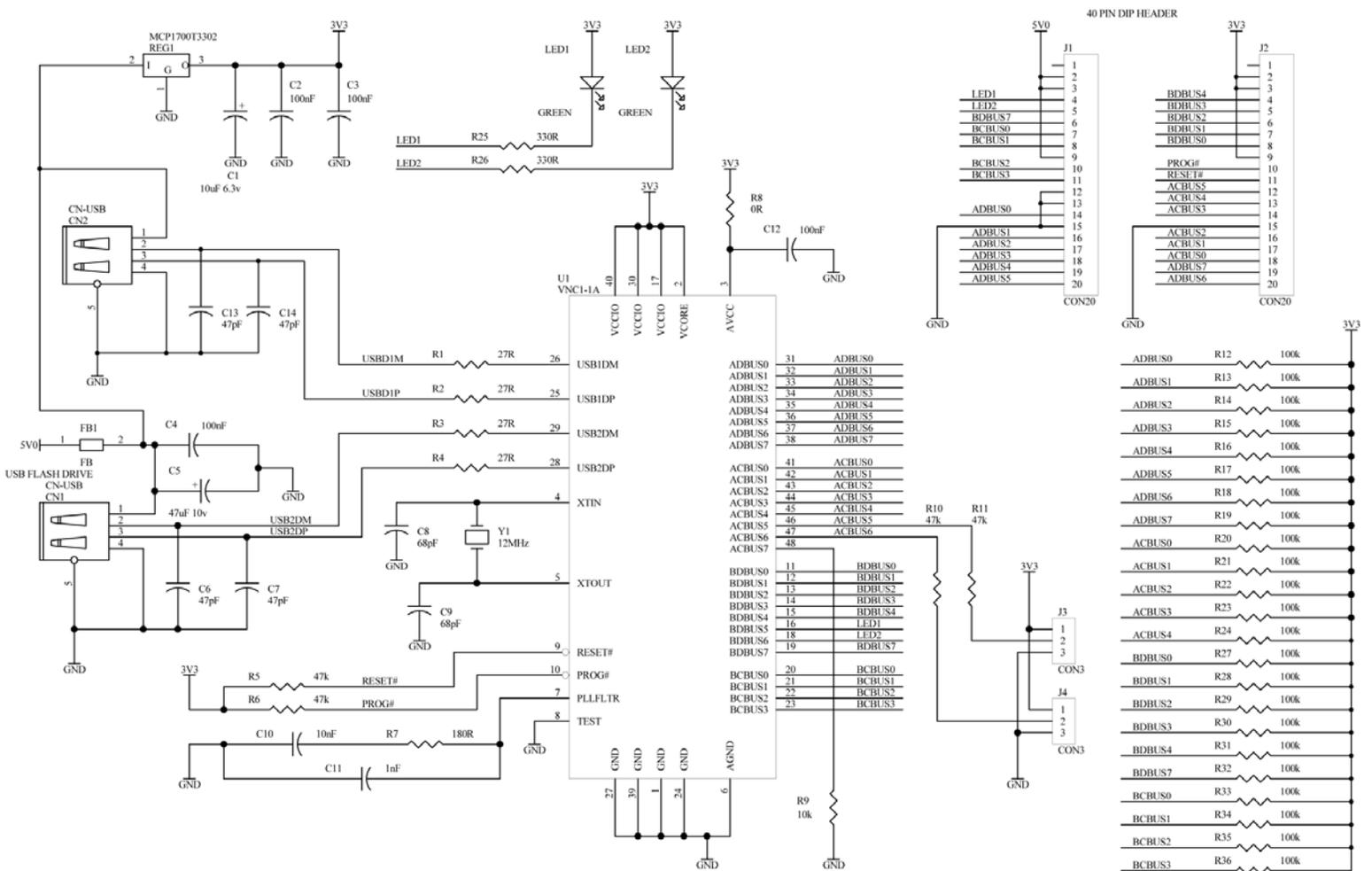


Image used with permission from FTDI.

EXAMPLE BUILT-IN HOST CONTROLLER

For the built-in host controller example, we'll be looking at the mbed NXP LPC1768 development board. It is a miniature single board computer based on the NXP LPC1768 ARM Cortex-M3 MCU. **Figure 9** shows a top view of the module along with descriptions for each pin. The hardware and software are licensed to users under the Apache 2.0 license, which allows royalty-free use for personal and commercial products.

The mini-USB port on the module is a device port that connects to a PC for software downloads and debugger functions. To use this board as a USB host controller, you need to wire up the USB host connector to the ports indicated in the [mbed HDK](#)³² as shown in **Figure 10**.

Figure 9: mbed NXP LPC1768 Development Board

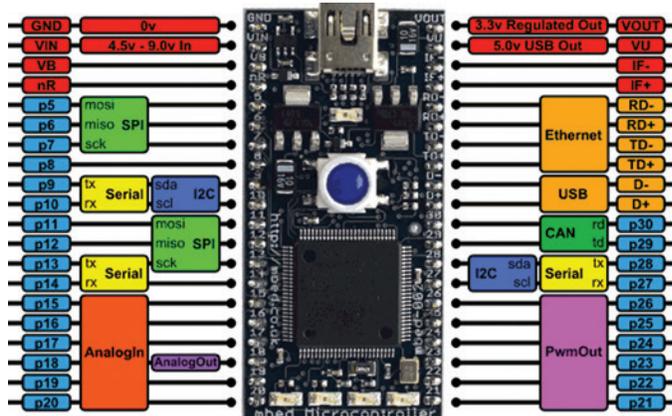
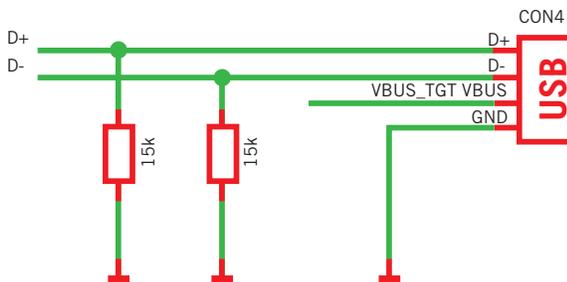


Image used with permission from mbed.

The software development tools for mbed are free and online, so developing code is as easy as creating a web account. The IDE, compilers, version control system and libraries are all online. The system is designed to make it easy to import source code from other developers who share their work. For instance, one developer has designed and shared a project named MSCUsbHost [found here](#)³³. The program uses the included libraries, middleware and FAT file system to read the directory of a USB flash drive, write a test file and read back the contents of the test file. This code can be easily imported into your online account to help you get started right away.

When it comes to building your own hardware using the NXP LPC1768, you can download the mbed NXP LPC1768 development board schematics from www.mbed.org.

Figure 10: mbed NXP LPC1768 USB Host Connection



IMPLEMENTING THE EMBEDDED FIRMWARE

Once the hardware is built or bought and system software is installed, it is time to write or modify your application. Here are some things to keep in mind:

THE USB FLASH DRIVE FILE SYSTEM

Most USB flash drives ship preformatted with the FAT32 or ExFAT file system. In all likelihood, the USB flash drives that plug into your system will be formatted with one of these file systems. ExFAT is not compatible with the FAT file system, so if your system only supports FAT, you will need to be able to detect whether newly inserted USB flash drives are formatted using FAT or not, and possibly prompt to reformat them.

HANDLING SURPRISE REMOVAL AND POWER LOSS

As convenient as a removable USB flash drive may be, that convenience introduces an element of uncertainty to the embedded application developer. The simple fact is that a removable USB flash drive opens up the possibility for the flash drive to be removed during a read or write operation. Embedded systems may also be subject to unpredictable power loss.

The FAT file system helps with surprise removal and power loss by maintaining two copies of the File Allocation Table (FAT). In case a removal or power loss happens while writing the table, chances are that only one table will be corrupted, so the disk can be recovered. ExFAT only maintains one copy of the FAT, so recovery is less likely with ExFAT.

Another way to minimize the possibility of media corruption due to disconnection or power loss is to design your software to perform file access operations in a manner that minimizes the duration of the file access. Rather than opening a file at the beginning of the program and leaving it open for a long period of time, it is better to open the file, read or write the data, then close the file immediately afterwards. By adopting this approach, the possibility of file corruption is minimized.

If your file system implements a caching scheme, it is best to disable write caching because it can lengthen the duration of a write operation due to asynchronous cache flushing.

FILE SYSTEM LICENSING

The FAT file system is one of the most popular file systems in use today. The FAT and ExFAT file systems are covered under software patents owned by Microsoft. If your product implements a FAT file system, you should check with Microsoft to see if you need a license to use it in your product. If your product needs to use ExFAT, you definitely need to purchase a license. For more information, visit [Microsoft's Technology Licensing Programs](#) page.

ABOUT DATAKEY

ATEK Access Technologies' Datakey brand has a long and distinguished history (dating back to 1976) of producing rugged portable memory products for the embedded electronics industry. The company's RUGGEDrive™ product line includes a variety of USB flash drive devices that addresses many of the weaknesses found in consumer USB flash drives. The RUGGEDrive™ line includes USB flash drives with ruggedized packaging, extended temperature operation, read-only operation, and other security and OEM-customizable features. The line also includes high cycle-life, IP65 and IP67-rated receptacles, for demanding, harsh-environment applications. For more information on the RUGGEDrive™ line or other Datakey portable memory products, please visit www.datakey.com.

CONCLUSION

We have covered many topics with regard to implementing an embedded system with USB flash drive support. After discussing high-level system design, we looked at many practical system implementation options available to you at the hardware, system software and application levels. It is our hope that this paper is helpful to you as you undertake your development and that you will avoid many difficulties that you may have encountered otherwise.

GLOSSARY

EHCI: EHCI is an acronym for Enhanced Host Controller Interface. EHCI is a register-level interface for a USB 2.0 host controller. The EHCI Specification may be found here:
www.intel.com/content/www/us/en/io/universal-serial-bus/ehci-specification.html

OHCI: OHCI is an acronym for Open Host Controller Interface. OHCI is a register-level interface for a USB 1.1 host controller. The OHCI specification may be found here:
ftp://ftp.compaq.com/pub/supportinformation/papers/hcir1_0a.pdf

Device Driver: A device driver is a software component that implements a predefined set of functions that allows an operating system to communicate with hardware devices in a uniform manner.

Class Driver: A class driver is a device driver for USB devices that provides a device-independent interface to an operating system subsystem.

Protocol Stack: A protocol stack is a collection of software components that implements a communication protocol.

Middleware: A collection of software or firmware that provides a software service that may not be provided by the operating system.

RTOS: RTOS is an acronym for Real Time Operating System. An RTOS is a special type of operating system specifically designed for systems that have specific determinism, throughput and/or response time requirements.

REFERENCES

- ¹ Universal Serial Bus Overview , Intel Website, <http://www.intel.com/content/www/us/en/io/universal-serial-bus/universal-serial-bus.html>
- ² USB-IF Hardware and Software Tools, USB-IF Website, www.usb.org/developers/tools
- ³ USB-IF Compliance Program, USB-IF Website, www.usb.org/developers/compliance
- ⁴ NXP Interface and Connectivity Products, NXP Website, www.nxp.com/products/interface_and_connectivity/usb_host_controllers/#products
- ⁵ Cypress USB Hosts, Cypress Website, <http://www.cypress.com/?id=186>
- ⁶ FTDI Products, FTDI Website, www.ftdichip.com/Products/ICs.htm
- ⁷ Tiva C Series MCUs, TI Website, www.ti.com/lscds/ti/microcontroller/tiva_arm_cortex/c_series/overview.page
- ⁸ TI Safety Microcontrollers, TI Website, www.ti.com/lscds/ti/microcontroller/safety_mcu/overview.page
- ⁹ TI Sitara Microprocessors, TI Website, www.ti.com/lscds/ti/arm/sitara_arm_cortex_a_processor/overview.page
- ¹⁰ TI OMAP Application Processors, TI Website, www.ti.com/lscds/ti/omap-applications-processors/overview.page
- ¹¹ Freescale Kinetis Microcontrollers, Freescale Website, www.freescale.com/webapp/sps/site/homepage.jsp?code=KINETIS
- ¹² Freescale Vybrid Microprocessors, Freescale Website, www.freescale.com/webapp/sps/site/homepage.jsp?code=VYBRID
- ¹³ Freescale i.MX Series Microprocessors, Freescale Website, www.freescale.com/webapp/sps/site/homepage.jsp?code=IMX_HOME
- ¹⁴ Atmel AVR Microcontrollers, Atmel Website, www.atmel.com/products/microcontrollers/avr/default.aspx
- ¹⁵ Atmel ARM Microcontrollers, Atmel Website, www.atmel.com/products/microcontrollers/arm/default.aspx
- ¹⁶ NXP LPC Microcontrollers, NXP Website, www.nxp.com/products/microcontrollers
- ¹⁷ Microchip PIC Microcontrollers, Microchip Website, <http://www.microchip.com/pagehandler/en-us/technology/usb/microcontrollers/home.html>
- ¹⁸ Windows Embedded, Microsoft Website, www.microsoft.com/windowseembedded
- ¹⁹ emUSB Host, Segger Website, www.segger.com/emusb-host.html
- ²⁰ Keil MDK-ARM, Keil Website, www.keil.com/arm/mdk.asp
- ²¹ Green Hills Products, Green Hills Website, www.ghs.com/products.html
- ²² MQX Tools, Freescale Website, www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MQX
- ²³ TivaWare Resources Page, TI Website, www.ti.com/lscds/ti/microcontroller/tiva_arm_cortex/c_series/tools_software.page#tivaware
- ²⁴ ASM USB Host Stack, Atmel Website, www.atmel.com/Images/doc8486.pdf
- ²⁵ MQX Software Solutions, Freescale Website, www.freescale.com/webapp/sps/site/homepage.jsp?code=MQX_HOME
- ²⁶ USB Specification v2.0, USB-IF Website, www.usb.org/developers/docs/usb20_docs
- ²⁷ On-The-Go and Embedded Host, USB-IF Website, www.usb.org/developers/onthego
- ²⁸ SVS Reset Delay, www.ti.com/lscds/ti/power-management/supervisor-reset-ic-overview.page
- ²⁹ 'Hi-Speed USB PCB Layout', Cypress Semiconductor, <http://www.cypress.com/?docID=47409>
- ³⁰ Saturn PCB Toolkit, Saturn Website, www.saturnpcb.com/pcb_toolkit.htm
- ³¹ PCB Trace Calculator, Advanced PCB Website, <http://www.4pcb.com/trace-width-calculator.html>
- ³² mBED HDK, mbed Website, <http://mbed.org/teams/mbed/code/mbed-HDK/>
- ³³ MSCUsbHost Example, mbed Website, www.mbed.org/users/igorsk/code/MSCUsbHost
- ³⁴ FAT File System Licensing Program, Microsoft Website, <http://www.microsoft.com/en-us/legal/IntellectualProperty/IPLicensing/Programs/Default.aspx>

PRESENTED BY:



**EXPERTS IN RUGGED
INDUSTRIAL MEMORY
SOLUTIONS**

Tel: +44 (0) 1794 301439

E-mail: info@nexusindustrialmemory.com

Web: nexusindustrialmemory.com

Datakey products are exclusively distributed by Nexus in Austria, Denmark, Finland, Germany, Ireland, Norway, Sweden, Switzerland and the UK.