Cornell University
School of Electrical and Computer Engineering

USB Host Controller for an MCU

Terry Young Hwa Kim, MEng '12
Electrical and Computer Engineering, Cornell University

Abstract

Using a microcontroller as a USB host device instead of as a USB peripheral device can be very helpful for students in ECE 4760 (Digital Systems Design Using Microcontrollers) to attach a USB mouse, a USB keyboard or a USB memory stick. Although a full software version of a USB host has been previously implemented on AVR Mega32 microcontrollers as a final project for ECE 4760 by students, the fact that it heavily loads the microcontroller forces us to look for an alternative solution, in other words, a hardware implementation. For example, dedicated host chips such as VNC1L and MAX3421 provide high-speed interface to unload the microcontroller. I employed the USB protocols and implemented a useable USB host interface to Mega1284 using a chip (VNC1L on VDIP1 module) that unloads the microcontroller. The final product includes the libraries that consist of APIs that a host uses to communicate with HID (human interface device) and mass storage class peripheral devices. The hardware implementation can be used by students in ECE 4760 to run a mouse, a keyboard and a memory stick. This will allow students to easily attach peripherals such as a mouse to their final project without having to use a host computer.

VDIP1 and Firmware

A development module for VNC1L is called VDIP1 and it ships pre-programed with firmware VDAP.

Extended Command Set	Function
Monitor Configuration Commands	
ECS (Extended Command Set)	Switches to the extended command set
IPA (Monitor Mode ASCII)	Monitor commands use ASCII values
FWV (Firmware Version)	Display firmware version
Disk Commands	
DIR file (Directory)	List specified file and size
OPW file (Open File for Write)	Open a file for writing or create a new file
SEK dword (Seek)	Seek to the byte position specified by the 1st parameter in the currently open file
WRF dword (Write to File) data	Write the number of bytes specified in the 1st parameter to the currently open file
CLF file (Close File)	Close the currently open file
OPR file (Open File for Read)	Open a file for reading
RDF dword (Read From File)	Read the number of bytes specified in the 1st parameter from the currently open file
USB Device Commands	
QP2 (Query Port)	Query port 2
QD byte (Query Device)	Query device specified in the 1st parameter
SC byte (Set Current)	Set device specified in the 1st parameter as the current device
SSU qword (Device Send Setup Data)	Send setup data to device control endpoint with optional follow-on data

USB Protocol Basics

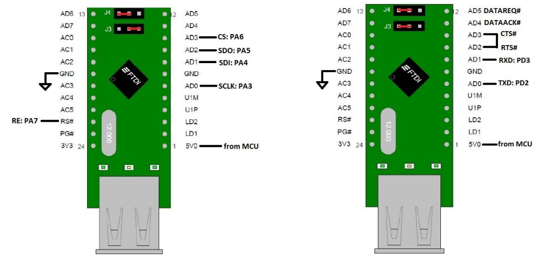
USB Transfers: A USB device uses a USB transfer to send and receive data for communication. Each transfer has a defined format for sending data, status and control information and more. There are four types of USB transfers: control, bulk, interrupt and isochronous transfers. All USB devices must support control transfer which is used for identification and configuration of a device. USB devices such as USB flash drives and printers use bulk transfer that does not guarantee latency while HID devices such as USB keyboards and USB mice use interrupt transfer that does guarantee low latency. Except for control transfer, interrupt transfer is the only way low-speed devices such as a USB mouse can transfer data.

Important elements of a USB transfer are called endpoints, and there is a pair of endpoints: IN and OUT. Once a transfer is scheduled, all data on a USB bus travels to or from a device endpoint. Simply put, endpoints are buffers that store bytes which are received or waiting to be transmitted. While a host device has buffers to hold received data and data waiting to be transmitted, only a slave device has device endpoints. Before any data can be exchanged between a host and a slave, they must establish a pipe. A pipe connects endpoints of a device to a host. Any transfer type is allowed to use IN and OUT transactions.

Enumeration: Once a USB slave device is connected to a USB host, the host needs to learn about the device before any application can communicate with the device. This process of initialization and exchanging information between a host and a slave device is called enumeration. During the enumeration, a host can assign an address to a device and read all kinds of descriptors from a device. Once the enumeration is completed, a slave device can be ready to transfer data to a host. USB descriptors are data structures that contain information about a device such as interface and endpoints of it. They enable a host to learn about a slave device, and all USB devices are required to respond to requests for the standard USB descriptors from a host. VNC1L from FTDI does execute the process of enumeration in the background for us as soon as it detects a device on its USB port.

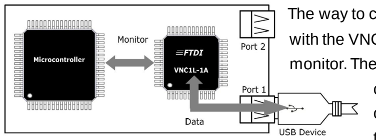
Human Interface Device (HID) Report: It was mentioned above that HID devices such as a USB keyboard or a USB mouse use interrupt transfer to send data to a host through an IN interrupt endpoint (the direction of any data transfer is always from a host point of view). This data that a device sends to a host is contained in a report, which HID devices use to exchange data. A report descriptor contains information about the data that is sent and received between a host and a device; however, the descriptor does not include a report itself in it. HID class specific requests can be used instead to get a report from a device and send a report to a device. For example, a report from a USB keyboard can tell a host which key has been pressed, and a report from a USB host can turn on an LED for a NUM Lock on a keyboard. The Windows HID API provides a set of functions that applications can use to get and send a report. Conveniently, the firmware for VNC1L comes with a command that can do the same.

SPI vs. USART



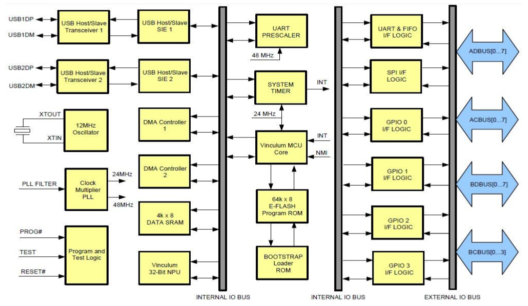
VNC1L provides configuration options to interface to the Command Monitor via UART, SPI or FIFO. When the data and control buses of VNC1L are configured in the UART mode, the interface implements an asynchronous serial UART port with flow control. When the buses are configured in the SPI mode, the interface operates as an SPI slave and thus it will need a master to provide the clock. In this project, both SPI and USART are employed for the serial communication between the microcontroller host (mega1284) and the VNC1L on the VDIP1, which is connected to a USB slave device. Both SPI and USART are tested with a USB flash drive, a USB keyboard and a USB mouse, and they proved to work fine equally.

Command Monitor



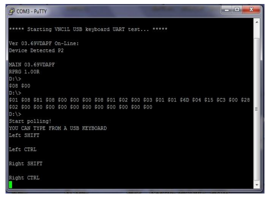
The way to control and communicate with the VNC1L is through a command monitor. The firmware activates a command monitor port on one of the USB ports, and this allows an embedded device such as an MCU to communicate with a USB peripheral device via the VNC1L's UART, SPI or FIFO interface. An MCU can send instructions (commands) to the command monitor and receive data from it. When the VNC1L is ready to take a command and after a successful execution of any command, the command monitor returns a prompt (D:\>).

Vinculum VNC1L Embedded USB Host Controller IC

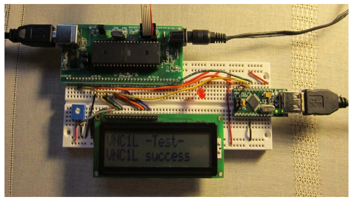


VNC1L from FTDI is a single chip embedded dual USB host controller that features two independent USB 2.0 Low/Full-speed USB host ports. This chip handles entire USB protocols such as enumeration and various transfers of descriptors at IN and OUT endpoints. It comes with 64k bytes of embedded Flash (E-FLASH) memory to store firmware, which could be programed or updated via UART interface. VNC1L also provides options to interface to external Command Monitor via UART, SPI or FIFO slave interface. It does not require external software control because the free firmware takes care of it all, which is the best advantage of using a VNC1L chip for the project.

Compatibility of the tested USB devices with VNC1L


COM3 - PuTTY

```
***** Starting VNC1L USB keyboard UART test... *****

Ver 03.69VDAPF On-Line:
Device Detected P2

MAIN 03.69VDAPF
RPRG 1.00R
D:\>
$08 $00
D:\>
$01 $08 $81 $08 $00 $00 $00 $08 $01 $02 $00 $03 $01 $01 $6D $04 $15 $C3 $00 $28
$02 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00
D:\>
Start polling!
YOU CAN TYPE FROM A USB KEYBOARD
Left SHIFT

Left CTRL

Right SHIFT

Right CTRL
```



USB flash drive		
SanDisk	16 GB, FAT 32 file system	Works
Cruzer mini	1 GB, FAT 32 file system	Works
Cornell ECE (unknown vendor)	2 GB, FAT file system	Works
Unknown manufacturer	256 MB, FAT 32 file system	Works
USB keyboard		
Logitech	Model #: Y-UR83	Works
HP	Model #: SK-2885	Does not work
DELL	Model #: SK-8115	Does not work
USB mouse		
DELL	Model #: M056U0A	Work
Logitech	Model #: M-U0007	Work except for scrolling the wheel
GE	Model #: unknown	Does not work
Unknown manufacturer	Model #: unknown	Does not work