APPLICATION NOTE

AN405

FTDI USB Adapter Interfacing for Rabbit

Universal Serial Bus (USB) has replaced the standard RS232 port on many PCs, especially in portable situations such as Laptop/Notebook computers and PDAs.

There are three components to USB. The host adapter, which usually resides on a host such as a PC, and the target, which is implemented in the peripheral. In addition to the host and target there is a hub, which acts as an intermediary to extend the number of USB ports and to provide power for the downstream (host -> target) targets. A maximum of seven hubs for one chain can be used, including the root hub that is usually present on the host adapter.

USB is designed to put much of the complexity on the host side (PC) thus making it feasible to have a very simple target (peripherals) implementation. This strategy is cost effective in an implementation with one host and several peripherals.

Peripherals come in many flavors and are grouped in classes of target types that perform similar functions. A popular class is the human interface device (HID), which includes keyboards, mice and the like. HID devices are very simple. They provide a description of themselves (device capabilities) and provide simple data, such as the scan codes of the keys on a keyboard via USB.

Another popular HID is the communication device class of which the serial port is a subclass. A USB serial device is simple by convention: once initialized it just passes bytes between the host and the target. The details of how USB works at the lower level can be found in some of the references mentioned below.

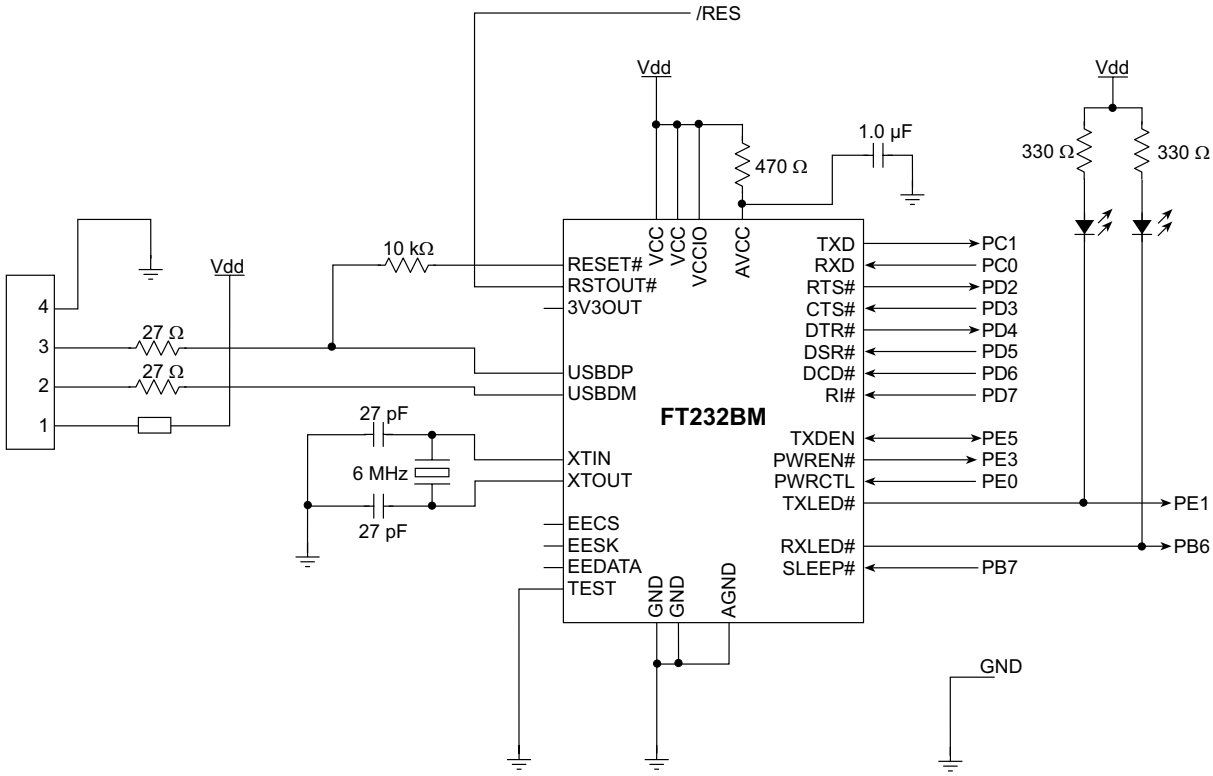
USB-Based Interface Chips

Future Technology Devices International, Ltd. (FTDI) provides several USB-based interface chips to use when interfacing peripherals to USB-based PCs. There are two chips, the FT232BM and the FT245BM, that implement USB serial devices. FTDI also provides free software drivers for PC-based platforms, as well as MacOS and Linux. The drivers provide a COM-like interface to software running on the host.

FT232BM

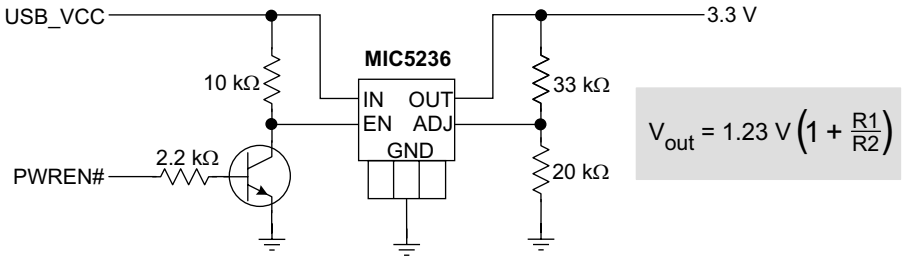
The FT232BM is a USB-to-asynchronous serial interface. Conceptually, it behaves like a modem to the peripheral device. The FT232BM provides an asynchronous serial transmit and receive interface as well as modem control signals ready to interface to an asynchronous UART. High baud rates of up to 920K are supported by this chip set and the drivers. The FT232BM is easily interfaced to the Rabbit using one of its internal UARTs. The CMOS voltage levels of the 5 V FT232BM are compatible with the Rabbit CPU and thus it presents a glueless interface to the Rabbit. A schematic of the FT232MB interfaced to the RCM3000 prototyping board is shown in Figure 1.

Figure 1. FT232BM interfaced to the RCM3000 prototyping board



Depending on the application, it may not be necessary to interface the modem control signals, or manage the power features of the FT232BM chip. The FT232BM can be USB-bus powered and provide an on-board 3.3 V regulated output that can be used to power the VIO side of the FT232BM chip. However, it does not have enough current capability to power a Rabbit CPU running at full speed. At reduced speeds a Rabbit CPU will draw less current and can be USB bus powered. A separate LDO regulator can be used to power the CPU from the USB power bus. Figure 2 shows such an arrangement.

Figure 2. Using a Separate LDO Regulator for Power



The PWREN# signal can come from the FT232BM or FT245BM. This circuit turns on the power to the external circuit (+3.3 V) when Windows has initialized the FT2xx chip. It turns off the power when Windows suspends USB devices to save power.

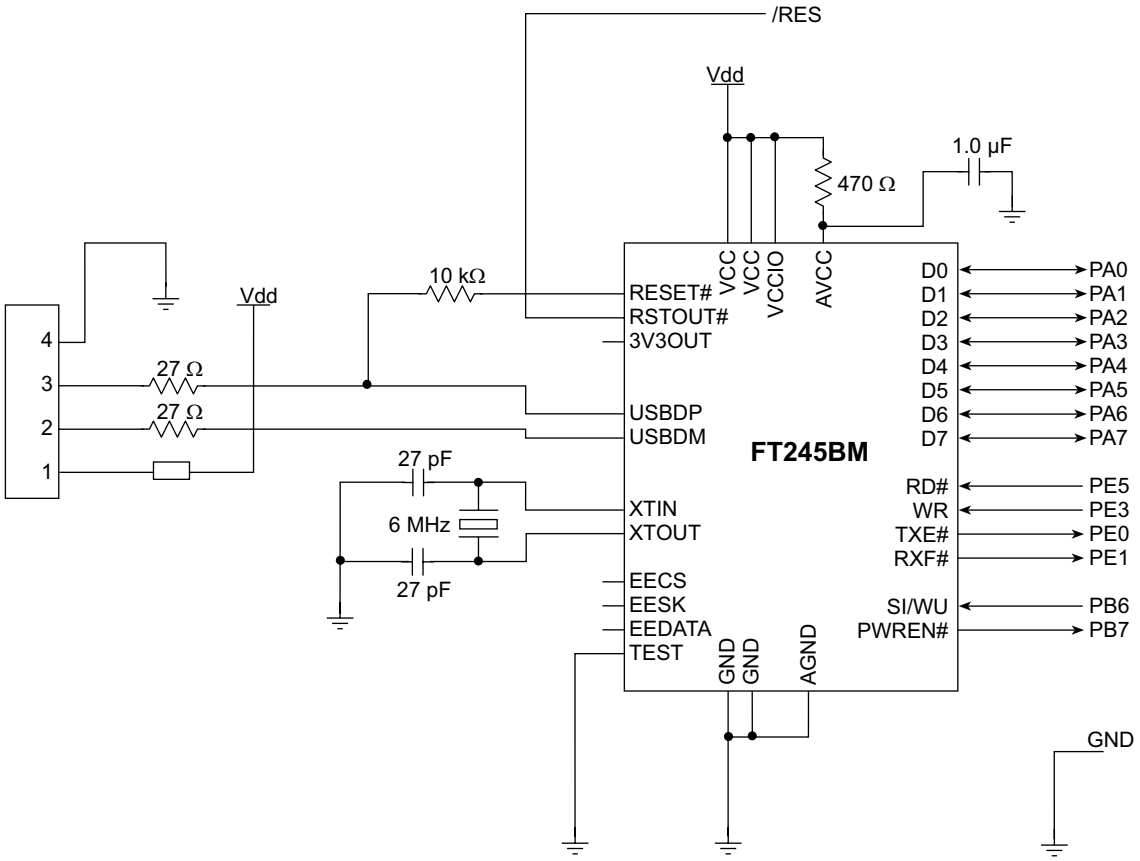
Using the internal serial ports of the Rabbit, it is possible to use standard serial library functions (`serXopen()`, `serXputc()`, `serXputc()`, etc.) to communicate with a host via USB and the FT232BM. In many cases you can port a RS232-based Rabbit application to a USB-based one without software changes. All it takes is the FT232BM interface on the Rabbit side and the FTDI-provided virtual COM drivers (VCOM) for the host end. The FTDI-provided drivers are royalty-free USB drivers.

The sample code to configure the Rabbit pins and then read and write a byte is simple and is done by sample program `ftdi232.c`. Details for this program are given below. The source code, shown in Listing 1, is available in the zip file accompanying this application note, `AN405.zip`.

FT245BM

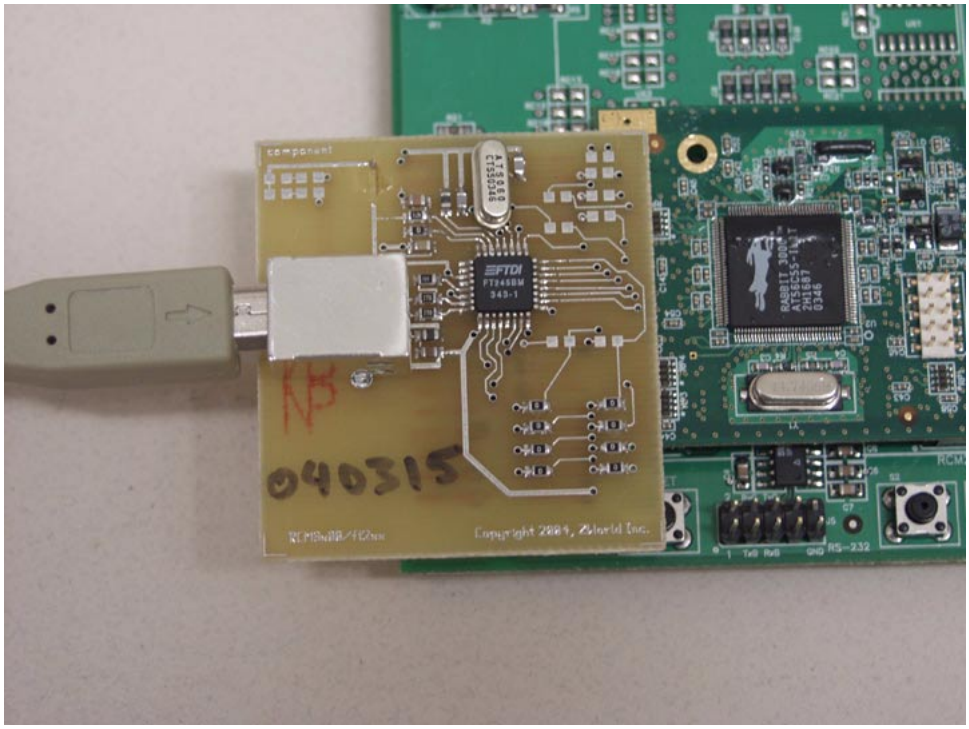
The FT245BM is similar to the FT232BM, except that the interface to the peripheral device is via a bi-directional 8 bit data bus. In addition to a read and write strobe, which the Rabbit uses to signal whether the Rabbit is transferring data to or from the FT245BM's internal FIFO, there are signals indicating FIFO status to the device. Figure 3 shows how a FT245BM is interfaced to a Rabbit.

Figure 3. Schematic of FT245BM Interface to a Rabbit



The FT245BM needs to have drivers on the Rabbit to read and write data from/to the USB host. The code to read and write a byte is simple and is done by sample program `ftdi245.c`. Details for this program are given below. The source code, shown in Listing 2, is available in `AN405.zip`, the zip file accompanying this application note. Unlike the FT232BM, the FT245BM has no modem control signals and automatically implements flow control using the devices FIFO interface.

Figure 4. Implementation of FT245BM Interface to a Rabbit



When the PC has data to write to the Rabbit, and the FT245BM's receive FIFO is full, the VCOM interface on the PC will signal that it cannot accept any more characters. When the Rabbit reads data from the FT245BM's receive FIFO, space is freed up and more bytes can be sent from the PC. If the PC is not ready to receive data, the Rabbit will continue to fill the FT245BM's transmit buffer until it is full, at which time the Rabbit will have to wait until bytes are read out of the FIFO by the PC.

On the host side, the same driver from FTDI will present a virtual COM interface. The buffer management is transparently handled by a combination of the VCOM interface, the USB subsystem and the FT245BM's FIFOs.

The interface board described in Figure 3 and pictured in Figure 4 plugs into a RCM3000/3100/3200 prototyping board. Using this design you can run the sample programs described in the next section.

Sample Programs Instructions

Some sample programs are provided to demonstrate the FTDI and Rabbit interface. These programs are described here and provided in source code form in the accompanying zip file, `AN405.zip`. Download the sample program (`ftdi232.c` or `ftdi245.c`) to the Rabbit-based board via the programming cable. After connecting the Rabbit-based board and the host via a USB cable, and running a terminal emulation program under Windows or `tstserial.c` under Linux, run the sample program on the Rabbit-based system.

`ftdi232.c - echo program for FT232BM implementation`

This program is a simple asynchronous sample program for FT232BM implementation. While it is aware of a full modem interface, it sets the modem control signals into a default state and operates in 3-wire asynchronous serial mode. It uses serial port D and a default baud rate of 921600 baud, the maximum that the PC VCOM driver supports. To test this program attach a FT232BM as shown in Figure 1, and use a terminal program on the host after the VCOM driver from FTDI's website has been installed. The terminal program must open the COM port associated with the FT232BM/Rabbit combination and select the same baud rate as selected by the Rabbit.

`ftdi245.c - echo program for FT245BM implementation`

This program works with the FT245BM as attached in Figure 3. It initializes Port A to works as a AUXIO port with two external chip enables. The external chip enables are mapped to the FT245BM's RD# and WR strobe line. An external read from address 0xa000 will cause the RD strobe to be asserted and a byte be transferred from the FT245BM to the Rabbit. An external write to address 0x6000 will present a byte on Port A and assert the FT245BMs WR strobe. The program enters a loop where it checks to see if any characters are available in the receive FIFO, and then read it and transfer it to the transmit FIFO of the FT245BM. This program can be used with a terminal program on the host in the same manner as `ftdi232.c` and the FT232BM, except that it doesn't matter what the baud rate of the VCOM is set to.

`tstserial.c - echo program for Linux to use with usbserial driver`

This program works with both the FT232BM and the FT245BM implementations. The VCOM drivers from FTDI are not needed; their function is implemented by the usbserial driver.

To compile the program under Linux:

```
%cc tstserial.c -o tstserial
```

Open the specified USB serial port (usually `/dev/ttyUSB?`) to run the program under Linux:

```
% ./tstserial /dev/ttyUSB0
```

To test the connection, compile and download `ftdi232.c` or `ftdi245.c` to the Rabbit, depending on your implementation. `tstserial`.c running on the host will send a block of bytes and read them back, then check the accuracy using the function `memcmp()`. Once 100 blocks of data have been transferred, the program will report the number of errors encountered, if any, as well as the number of bytes that were transferred.

Summary

Both FTDI chips are suitable for USB target implementation with Rabbit and have their advantages, depending on the application. The FT232BM has reduced I/O requirements, since it can be operated with minimal modem signals.

A 3-wire asynchronous interface is possible using only two I/O pins on the Rabbit. While maximum asynchronous speeds of over 900K baud are implementable, flow control has to be handled in software, unless the modem flow control signals (RTS/CTS) are used with the FT232BM. The asynchronous baud rate on the Rabbit has to match the baud rate selected by the host via the VCOM driver.

The FT245BM uses more I/O signals in its minimum configuration, as well as the AUXIO port of the Rabbit. One of the advantages of the FT245BM, however, is that it is baud rate independent and implements flow control transparently. The baud rate selected via the VCOM interface is ignored; the FT245BM transfers data at the maximum rate the Rabbit and host can handle.

Appendix

The sample code provided with this application note in `AN405.zip` is shown here for convenience.

Listing 1: ftdi232.c

```
/*************************************************************
* Docs\refs\AN405\AN405.zip:ftdi232.c
* Copyright (c) 2004, Z-World
* This program demonstrates setting up the Rabbit registers
* to interface with the FT232BM. The main program loop echos
* any characters received over the USB connection.
*************************************************************/

main(){
   int c;

   BitWrPortI(PBDDR,&PBDDRShadow,1,7);
   BitWrPortI(PBDR,&PBDRShadow,1,7);            // SLEEP#

   BitWrPortI(PDDDR,&PDDDRShadow,1,3);
   BitWrPortI(PDDR,&PDDRShadow,0,3);            // CTS#

   BitWrPortI(PDDDR,&PDDDRShadow,1,6);
   BitWrPortI(PDDR,&PDDRShadow,0,6);            // DCD#

   BitWrPortI(PDDDR,&PDDDRShadow,1,5);
   BitWrPortI(PDDR,&PDDRShadow,0,5);            // DSR#

   BitWrPortI(PDDDR,&PDDDRShadow,1,7);
   BitWrPortI(PDDR,&PDDRShadow,1,7);            // RI#

   BitWrPortI(PDDDR,&PDDDRShadow,0,4);          // DTR#

   BitWrPortI(PDDDR,&PDDDRShadow,0,2);          // RTS#

   BitWrPortI(PEDDR,&PEDDRShadow,0,1);          // RXF#

   BitWrPortI(PBDDR,&PBDDRShadow,0,6);          // TXE#

   serDopen(921600);

   while(1){
      if ((c = serDgetc()) != -1) {
         serDputc(c);
      }
   }
}
```

Note that PE1 and PB6 are unused. They are configured as inputs to allow the LEDs to work.

Listing 2 - ftdi245.c

```
/*************************************************************
* Docs\refs\AN405\AN405.zip:ftdi245.c
* Copyright (c) 2004, Z-World
* This program demonstrates setting up the Rabbit registers
* and external I/O mapping to interface with the FT245BM.
* The main program loop echos any characters received over
* the USB connection.
*
* PE5 - rd# strobe, active low, address range starting at 0xa000
* PE3 - wr strobe, active high, address range starting at 0x6000
* PE1 - rxf# low when data is ready
* PE0 - txe# low when transmitter is ready
*************************************************************/
#define FT_WR_BASE 0x6000
#define FT_RD_BASE 0xa000

main(){
   unsigned char c;

   WrPortI(SPCR,&SPCRShadow,0x8c);        //  slave off, ext. I/O
   WrPortI(PEFR,&PEFRShadow,0x28);        //  external chip enables


   WrPortI(IB3CR,&IB3CRShadow,0x6c);      //  7 wait, write, active high
   WrPortI(IB5CR,&IB5CRShadow,0x58);      //  7 wait, read, active low


   BitWrPortI(PBDDR,&PBDDRShadow,1,6);    //  Pull SI/WU up when not used:
   BitWrPortI(PBDR,&PBDRShadow,1,6);      //    (from FT245BM data sheet)


   while(1){
      costate {
         waitfor((RdPortI(PEDR)&0x03)==0x00);
         c = RdPortE(FT_RD_BASE);
         WrPortE(FT_WR_BASE,NULL,c);
      }
   }
}
```

References

1. Axelson, Jan. USB Complete. Lakeview Press
2. The FTDI Home Page: this is where you can download the free VCOM drivers and view the data sheets for the interface chips:

   www.ftdichip.com

3. Distributors of FTDI chips FT232BM, FT245BM

   www.saelig.com
   www.parallax.com

4. FTDI application note on debugging FT232BM and FT245BM based designs:

   www.ftdichip.com/Documents/AppNotes.htm

   or

   www.ftdichip.com/Documents/AppNotes/AN232B-06_11.pdf