Basic USB Communications for PIC

by Ben Rowland, February 2009

Abstract

The code that drives the PICmicro USB functionality is sometimes confusing and difficult to navigate, however after the release of the USB pack for Flowcode the majority of the work is done for you, allowing you time to think about the program functionality rather than the inner workings of complicated USB code. In this article Ben demonstrates how to use Flowcode to control the USB communications.

Requirements

Software:

- Professional licence of Flowcode v3 or v4 for PIC.

Hardware:

- EB006 Multiprogrammer
- (or) HP488 Programmer board

With the release of the USB pack for Flowcode PICmicro we have suddenly been able to create a whole host of USB functional devices without the need for getting bogged down in the labyrinth of code that drives the PICmicro USB functionality. With the release of our EB055 USB E-Block we have been able to add this USB functionality to our EB006 and HP488 products. Therefore anyone with a EB006 Multiprogrammer or a HP488 V3 Development board can now obtain a USB enabled PICmicro device and start developing USB compatible embedded projects.

Here is a short list of common USB enabled PICmicro microcontrollers that will work directly with the EB055 E-Block via an EB006 or HP488 V3.

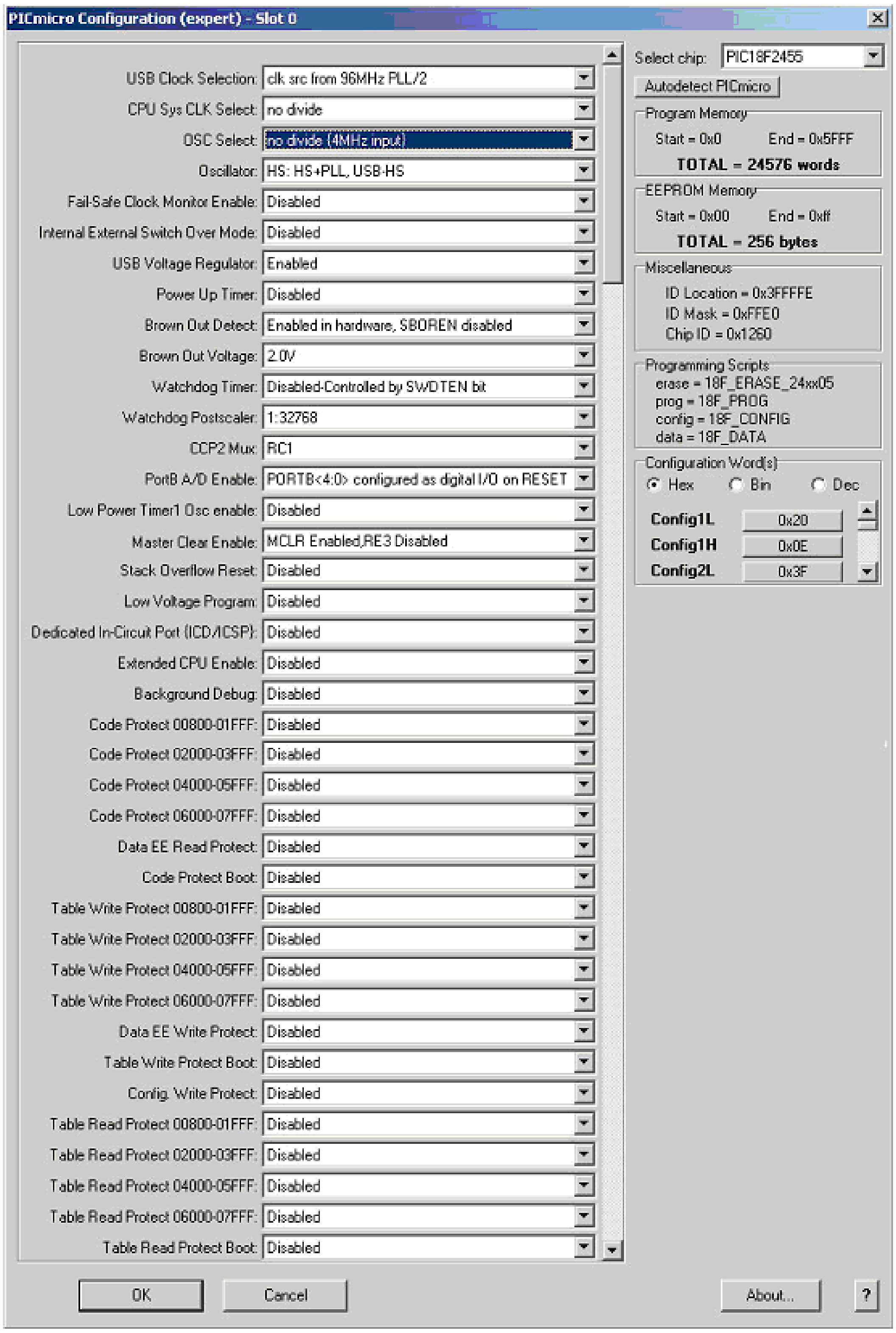
18F2455, 18F2550, 18F4455, 18F4550, 18F2450, 18F4450, 18F13K50, 18F14K50

Using the Flowcode USB components is relatively straight forward so I will skip directly to getting the code running on the device and allowing it to correctly synch up with the PC. For this synchronisation to happen the PICmicro must first be running at the correct speed. To establish the correct speed of 48MHz the USB enabled PICmicro devices have an onboard phased locked loop (PLL) connected to the oscillator circuit which allows the external clock speed to be boosted by a multiplier. The EB006 and HP488 boards come with a pre-socketed crystal making it possible to simply unplug the current crystal and insert a new one. To allow the PLL and USB peripheral to work correctly one of the following crystal speeds must be used.

Compatible Crystal Speeds
4, 8, 12, 16, 20, 24, 40, 48 MHz

Once you have changed your crystal from the standard 19.6608MHz device you are ready to edit the software configuration for the chip. The configuration is available on all PICmicro devices and controls things like MCLR and Watchdog functionality etc. Once you have created a program in Flowcode simply click on Chip -> Configure to access the configuration screen. Once you are here click on the switch to expert mode to allow the 18F PICmicro devices to be configured. If your USB enabled PICmicro device is already situated on the programmer board then clicking auto detect will load up the configuration settings for that particular device. Alternatively you can select the device from the drop down list at the top right.

Once you have defined your chip configuration settings for your Flowcode program they will be saved into the Flowcode FCF file so you will not have to repeat the configuration process every time you program the device. Following is an example configuration set to allow an 18F2455 to synchronise with the USB bus. Change the highlighted drop down menu to suit your crystal speed.



PICmicro Configuration (expert) - Slot 0

USB Clock Selection:	clk src from 96MHz PLL/2
CPU Sys CLK Select:	no divide
OSC Select:	no divide (4MHz input)
Oscillator:	HS: HS+PLL, USB-HS
Fail-Safe Clock Monitor Enable:	Disabled
Internal External Switch Over Mode:	Disabled
USB Voltage Regulator:	Enabled
Power Up Timer:	Disabled
Brown Out Detect:	Enabled in hardware, SBOREN disabled
Brown Out Voltage:	2.0V
Watchdog Timer:	Disabled-Controlled by SWDTEN bit
Watchdog Postscaler:	1:32768
CCP2 Mux:	RC1
PortB A/D Enable:	PORTB<4:0> configured as digital I/O on RESET
Low Power Timer1 Osc enable:	Disabled
Master Clear Enable:	MCLR Enabled,RE3 Disabled
Stack Overflow Reset:	Disabled
Low Voltage Program:	Disabled
Dedicated In-Circuit Port (ICD/ICSP):	Disabled
Extended CPU Enable:	Disabled
Background Debug:	Disabled
Code Protect 00800-01FFF:	Disabled
Code Protect 02000-03FFF:	Disabled
Code Protect 04000-05FFF:	Disabled
Code Protect 06000-07FFF:	Disabled
Data EE Read Protect:	Disabled
Code Protect Boot:	Disabled
Table Write Protect 00800-01FFF:	Disabled
Table Write Protect 02000-03FFF:	Disabled
Table Write Protect 04000-05FFF:	Disabled
Table Write Protect 06000-07FFF:	Disabled
Data EE Write Protect:	Disabled
Table Write Protect Boot:	Disabled
Config. Write Protect:	Disabled
Table Read Protect 00800-01FFF:	Disabled
Table Read Protect 02000-03FFF:	Disabled
Table Read Protect 04000-05FFF:	Disabled
Table Read Protect 06000-07FFF:	Disabled
Table Read Protect Boot:	Disabled

Select chip: PIC18F2455

Autodetect PICmicro

Program Memory
Start = 0x0 End = 0x5FFF
TOTAL = 24576 words

EEPROM Memory
Start = 0x00 End = 0xff
TOTAL = 256 bytes

Miscellaneous
ID Location = 0x3FFFFE
ID Mask = 0xFFE0
Chip ID = 0x1260

Programming Scripts
erase = 18F_ERASE_24xx05
prog = 18F_PROG
config = 18F_CONFIG
data = 18F_DATA

Configuration Word(s)
[✓ Hex] Bin Dec

Config1L	0x20
Config1H	0x0E
Config2L	0x3F

Whether you are working with an example from the USBpack download or a fresh project you will have to take a few steps to set up the Flowcode project for use with the USB. To do this you will have to follow these steps.

1. Change the target to the basic USB chip by clicking on Chip -> Target.
2. Change the clock speed to 48000000 by clicking on Chip -> Clock speed.
3. Change the target configuration by clicking on Chip -> Configuration.
4. Change the hardware RC / XTAL switch on the EB006 / HP488 boards to the XTAL position.

The example file provided for this article uses the Flowcode USB Serial component to create a virtual COM port connection on the PC. For instructions on how to set up hyperterminal for use with the USB Serial component please refer to the **“Communicating with a PC”** article.

The example Flowcode file waits for an incoming byte on the USB. When a byte is received it is checked to see if it is between the ASCII values ‘0’ and ‘7’ e.g. if you have typed the numbers 0 - 7 into hyperterminal.

If the incoming data is valid then the program will then output a logic high to the corresponding pin on Port B. Remember if you are changing the target chip specified in the example file then you will also have to check that the clock speed and configuration are correct in Flowcode before compiling the code to the chip.

Further reading

Below are some links to other resources and articles on related subjects, and technical documentation relating to the hardware used for this project...

Flowcode:	http://www.matrixmultimedia.com/flowcode.php
Eblocks:	http://www.matrixmultimedia.com/eblocks.php
Learning Centre:	http://www.matrixmultimedia.com/lc_index.php
User Forums:	http://www.matrixmultimedia.com/mmforums
Product Support:	http://www.matrixmultimedia.com/sup_menu.php