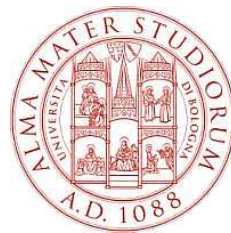


Design of a USB homebrew peripheral based on Microchip PIC 18F2455/2550/4455/4550

a feasibility study

by **Andrea Sottoriva**
2nd May 2006

Tutor:
Prof. Renzo Davoli



Department of computer science
University of Bologna (Italy)

Copyright ©1999-2006 MeTA-LAbS. All rights reserved.

This document is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This document is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this document; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Contents

1	Abstract	5
2	Microcontroller overview	6
3	Hardware configuration	7
3.1	Power on the device	7
3.2	Oscillator settings	8
3.3	Setting up the USB module	9
3.4	Build the microcontroller board	10
4	Availability and costs	11
5	Programmiers	12
6	Compilers and IDEs	13
7	Already available software	14
8	USB protocol configuration	15
9	Conclusions	16
10	Bibliography	18

List of Figures

1	<i>Device power circuit</i>	7
2	<i>Oscillator circuit</i>	8
3	<i>USB module circuit</i>	9
4	<i>Complete circuit</i>	10

1 Abstract

Recently, with the availability of low-cost microcontrollers with a great deal of enhanced complex peripherals, the interesting on such technologies from Computer Scientists grows a lot, in particular regarding interfacing such devices to a common computer.

Nowadays it is extremely easy to find out a cheap microcontroller with technologies like USART, A/D converters, PWM comparators, I2C, Ethernet and USB directly inside it.

The aim of this paper is to evaluate the feasibility of a little project which involves the Microchip microcontrollers family 18Fxxxx with enhanced USB controllers. The objective is to design a hardware and software framework on which a person with a little experience in electronics can work, using a few line of code and as simple as possible pre-designed circuits for various common needs. The idea consists on the development of a general purpose peripheral interfaced with the Linux kernel via the Universal Serial Bus that may be able to accept configuration commands and do I/O.

In particular, further in this document, will be discussed the controller-side part of the implementation problem, underling the aspects of PIC programming, availability and various hardware-related issues.

2 Microcontroller overview

The PICs 18F2455/2550/4455/4550 are 28/40/44 high performance, enhanced flash USB microcontrollers with nanoWatt technology. They are available in some different package flavours such as TQFP (44 pins), QFN (44 pins), SOIC (28 pins) and the more suitable PDIP (28 and 40 pins).

The matters discussed into this document can be considered appropriate for any of the microcontrollers listed above, however from now we will refer only to the microcontroller chose for the our project: the PIC18F2455.

This micro is the smallest and simplest of the family, so it provides all the needed features without add complexity to the project.

The main features of this PIC are described as follows.

- *General:*
 - up to 48 MHz operating frequency
 - RISC architecture with 75 instructions (83 with EIS enabled)
 - 24 KB of program memory (12288 instructions)
 - 2048 bytes of data memory (static RAM)
 - 256 bytes of EEPROM data memory
 - 19 interrupts sources
 - 8x8 single-cycle hardware multiplier
- *Provided features:*
 - three 8-bit I/O ports A, B, C and one input only port E
 - three 16 bit timers/counters and a 8 bit one
 - two capture/compare/PWM modules
 - enhanced USART
 - ten input channels for the 10-bit A/D converter
 - two comparators
 - Universal Serial Bus controller (enhanced transceiver)
- *USB features:*
 - USB v2.0 compliant
 - both low-speed and full-speed supported (1.5 and 12 Mbps)
 - 16 possible bidirectional endpoints

Obviously, even if implements the USB v2.0 specification, a 48 MHz microcontroller such as the 18F2455 is not able to support a 480 Mbps throughput of data provided by the full-speed USB specification, however 12 Mbps are enough for an enormous amount of applications. Moreover is not possible to use *all* the listed features simultaneously in a 28-pin packaged microcontroller (and some time neither in a 44-pin one), so a lot of them need to be multiplexed.

3 Hardware configuration

3.1 Power on the device

The first step of our analysis regards the power supply circuit.

The microcontroller needs a power source able to supply at most 25 mA with 5V. As we know the USB connector provides two wires for data and two wires for voltage that can supply a maximum of 100mA for low-power devices and 500mA for high-power devices. Therefore the USB bus can be used as a power source for the microcontroller connecting directly the V+ of USB to the Vdd pin of the PIC and the V- to the Vss pin.

For stability purposes a 10uF capacitor between Vdd and Vss is recommended. The MCLR pin is used for reset and it's active low so it can be simply connected to Vdd with a 1Kohm resistor and to Vss with a push button (normally open).

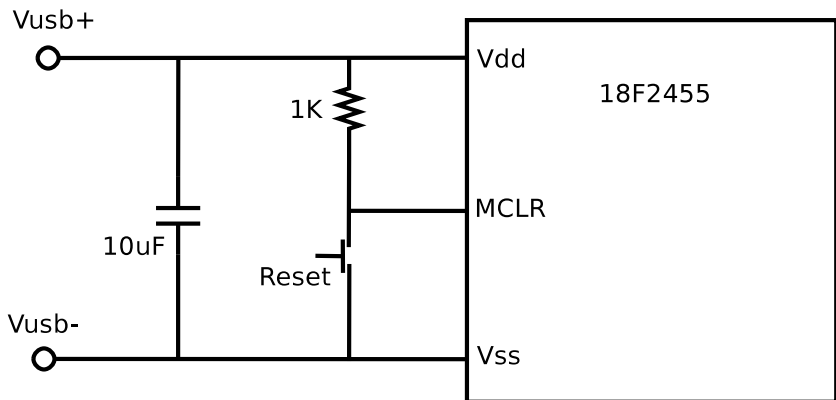


Figure 1: *Device power circuit*

It is recommended to insert additional components paying constantly attention to the power provided by the USB bus; the current supplied depends on the number of devices connected and on their consumption, so if high-power additional components are needed it is suggested to add also an extern power supply and to connect the circuits with safe optoisolators. The power management aspects are not discussed on this paper, but we can consider a value of 130mW as the maximum PIC consumptions.

3.2 Oscillator settings

The PIC18F2455 like all members of its family can be used choosing from a lot of different oscillator settings, however, in order to keep the things easy we have chosen XTPLL oscillator mode which permits to obtain 48Mhz of clock both for CPU and USB providing the maximum CPU speed configuration and the full-speed 12Mbps USB rate.

XTPLL needs to use a 4MHz quartz crystal between OSC1 and OSC2 pins, connected at the same time to Vss both with a 27pF ceramic capacitor. The 4MHz signal passes to a first PLL prescaler which divides the frequency by 1 (PLLDIV = 000) because it is responsible to give a fixed 4Mhz signal to the internal PLL.

This phase locked loop produces a fixed 96MHz frequency which is passed to the USB controller that divides it by 2 (USBDIV = 1, FSEN = 1) and to the CPU prescaler which divides the signal in the same way (CPUDIV = 00, OSCCON = 0x00).

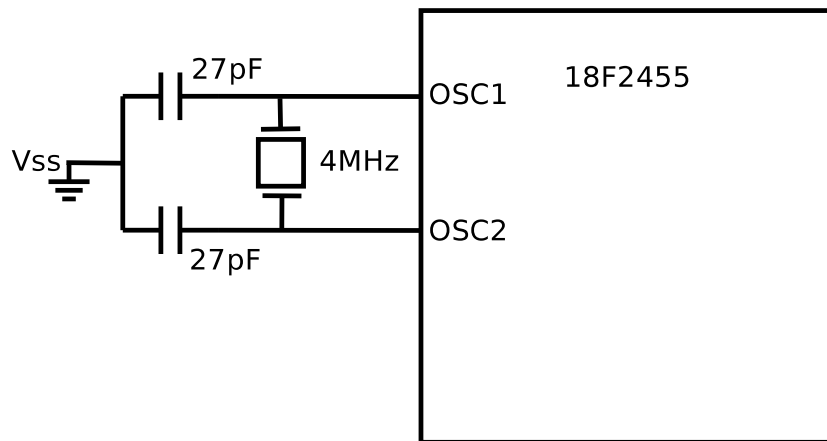


Figure 2: *Oscillator circuit*

The micro permits also to choose a lot of other oscillator settings but we have decided to use the simplest one which give the maximum performance, however you can refer to the PIC data sheet for different oscillator settings.

3.3 Setting up the USB module

The PIC has also an enhanced 3.3V regulator that can provide the appropriate voltage to the USB data bus. Internal pull-up resistors connected to D+ and D- are also available. Therefore the USB bus from the side of microcontroller doesn't need an hardware configuration, however is recommended to connect the Vusb (which is set to 3.3V) to Vss with a 10uF capacitor.

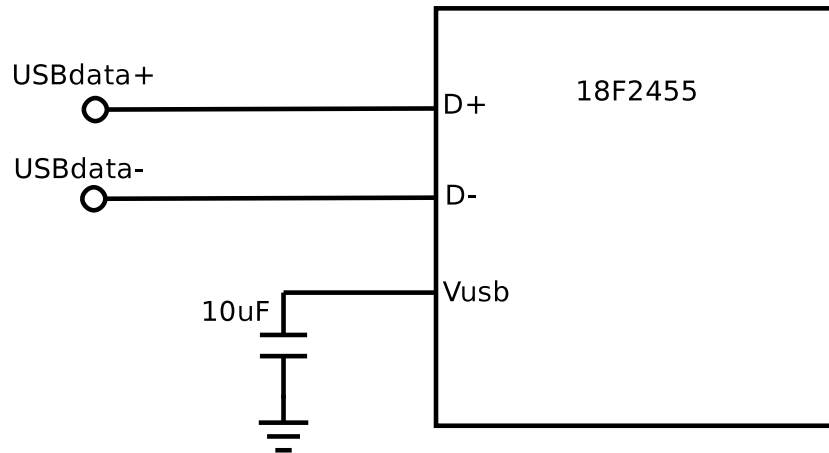


Figure 3: *USB module circuit*

If needed it is possible to configure the PIC also with external 3.3V USB power supply or place the pull-up resistors by hand, for a further discussion you may refer to the PIC data sheet.

3.4 Build the microcontroller board

At this stage, we have a completely functional microcontroller board and we can attach our device to the USB bus. It is possible to add a funny led to the circuit but remember that a common led needs at least 15/20mA, so very low power leds must be considered mandatory. Resuming, we need the following components:

- 1x pre-drilled experimenters board 120x80mm
- 2x ceramic 27pF capacitors
- 2x 10uF capacitors
- 1x 4MHz quartz crystal
- 1x USB connector
- strips and wires as needed

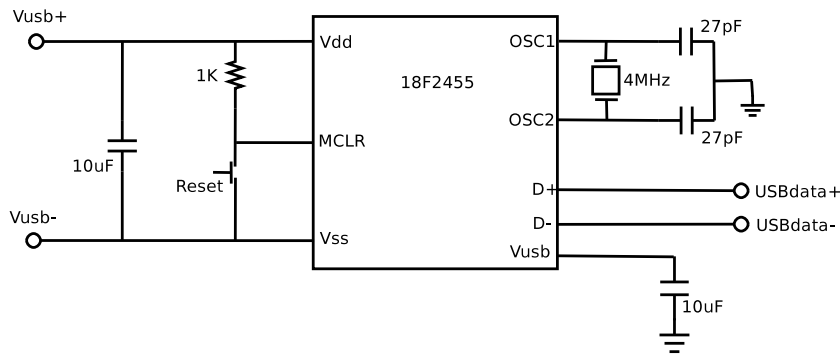


Figure 4: *Complete circuit*

And here it is our microcontroller development board. Simply ! Isn't it ? Now it's time to discuss the availability of this components and on PIC programming.

4 Availability and costs

One of the main requisites that we have imposed to this project is the low cost. Buy and build the discussed peripheral must be not only easy, but also cheap. Microchip's microcontrollers are quite diffused in Italy, especially those of the 16F family. Even though the situation for the 18F family isn't the same, however a certain growth of 18F microcontrollers inside the Italian retail market is expected in the near future. If any attempt of micro retrieve will fail, it is possible to get the samples from Microchip (at most 3 items) ordering directly from its website, they want only a valid name and address and, within few weeks you will get your little PICmicro box at home. The micro cost is approximately between 5E and 15E so with at maximum 20/25 at most we will be able to build the entire micro board. Microchip makes available also a USB demoboard with a 18F4550 microcontroller called *PICDEM* that is sell with manuals and various kinds of demo software but is also very expensive (at least 100E). A lot of PICs types are also available on ebay for very low prices (about 8/10E) so a preliminary searching on these types of auctions sites is always recommended.

5 Programmers

A serious problem that often appears when treating microcontrollers projects is the disposability of the programmer. Even if a great amount of programming features are supported by PICs like *In-Circuit Programming* it is a common problem to find a complete software/hardware solution because the resources are invested specially on business products rather than on homebrew.

Unfortunately there isn't a simple solution on this issue for 18F pics with enhanced USB, however a great deal of different programmers are available especially on ebay-like sites. In particular, south-east Asia countries like South Korea or Thailand often sell programming solution at very low cost with a specific support to the 18F USB pic family. A few months ago I have bought a PIC programmer on ebay which is able to program an incredible amount of microcontrollers for about 30€ from Thailand plus 7€ of custom duties. Despite the great variety of hardware solutions, the set of softwares used by PIC programmers is quite small and, unfortunately, only for Windows systems. I have tested some GNU/Linux programmers in the past which aren't specifically warranted for 18F PICs but with bad results: it is necessary to test a lot of different hardware and software configurations that may be dangerous for the microcontroller, with the constant risk that, finally, nothing could work. Notwithstanding all I think with some serious work it is possible to build a functional, flexible and portable programming solution for Linux because the programming method is approximately the same for every PIC, however this argument is not treated on this paper and neither on this project.

6 Compilers and IDEs

The scenario regarding the Integrated Development Environments available for PIC microcontrollers is fairly different than programmers one. A great amount of software compilers and assemblers are present in the common literature, a famous project which provides information regarding the opensource PIC software available is *gnupic* (www.gnupic.org). This web site collects a great deal of opensource PIC software spread all over the world which permits to generate binaries for PIC architectures from assembly and *High-Tech C* (the native C implementation for PICs microcontrollers). For the 18F microcontrollers family I suggest to use *PikDev* (<http://pikdev.free.fr/>), an IDE for KDE environment which supported a lot of PIC types. Microchip offers also a free development environment called *MPLAB*, freely downloadable from the Microchip website, however the Linux version seems to be no longer available and it must be used an older version or to try a wine emulation.

7 Already available software

A large set of papers and sources are downloadable in the *resource* section of the PIC 18F2455 home page: some already designed projects, various usage ideas and an interesting USB library which implements the entire enumeration protocol described in the Chapter 9 of the USB official protocol specification. This library provides three main calls:

- InitUSB()
- PutEP1()
- GetEP1()

The aim of this code is to provide an example of USB-HID setting which use only two EndPoints: the control endpoint 0 and a low speed output endpoint 1. An interesting example of application realized with this library is also distributed, it simulates a simply USB mouse connected as a HID device to the computer which drag around the mouse pointer on the screen when connected, the right code to use for the beta-testing of the assembled circuit. The main problem with this software is the licence that is not very restrictive but its use is permitted only on PICmicro environments, so creating GPL software based on this library must be avoided. A very interesting idea can be to develop a GPL version of the code which would not provides only a simply low speed HID-like configuration but also some different features and settings like bulk and isochronous data transfers at full-speed and multiple endpoints usage. This issue is fundamental because developing a GPL software solution was between the goals of our project, and we have to consider that the design of a peripheral driver must include the development of the entire pic-side USB firmware.

8 USB protocol configuration

The objectives of this project are to design a general purpose peripheral which has to be controllable in-software via the computer-side operating system. The idea is to have a PIC connected via the USB port that accept commands and data from the host. The commands flow uses the control endpoint in order to send to the pic a *configuration word*: a one or two byte tag which, recognized from the pic, sets the microcontroller into a specific configuration state, for example:

- 0x01: 8 bit output on PORTA
- 0x02: 8 bit input on PORTB
- ...
- 0xF0: 10 bit input from the 1st A/D channel
- ...
- 0xFF: output a specified PWM signal

Each word can need one or more parameters. Then, after the *configuration word*, a bulk-like data flow is expected from or to the PIC. Such a peripheral permits, using simply some *ioctl()*, *read()* and *write()*, a lot of experimentation and useful applications such as data acquisition and high speed I/O all over the USB bus, rather than an ancestral RS232 or a proprietary solution. An idea to start experimentation could be to use the Microchip USB firmware from the PIC side and treat the micro as a common HID device from the computer side; like the demo software discussed before but using bidirectional I/O. Develop such a driver for the Linux kernel become extremely simple and fast and permits to attempt the first steps through the project.

9 Conclusions

The project discussed suffers of a certain amount of problems, however I think that the exposed idea is so much interesting and, notwithstanding all...achievable. Actually there are a lot of solutions which involve the USB pics, but no one very open and general purpose for GNU/Linux systems.

Spreading the usage of cheap and funny technologies like USB microcontrollers in the daily experimentation, taking it out from the proprietary environments, is the main goal of the treated project and it opens a great deal of other interesting perspectives.

Andrea Sottoriva - hifi@metalabs.org, <http://www.metalabs.org/hifi>

10 Bibliography

- *Universal Serial Bus Specification, rev 2.0*, Compaq, Hewlett-Parkard, Intel, Lucent, Microsoft, NEC, Philips, 27th April 2000.
- *Device Class Definition for Human Interface Devices (HID), ver 1.11*, USB Implementers forum, 27th June 2001.
- *PIC18F2455/2550/4455/4550 Data Sheet - 28/40/44-Pin High-Performance, Enhanced Flash USB Microcontrollers with nanoWatt Technology*, Microchip Technology, 2004.