PIC32 Development Board

A Design Project Report

Presented to the School of Electrical and Computer Engineering of Cornell University

In Partial Fulfillment of the Requirements for the Degree of

Master of Engineering, Electrical and Computer Engineering

Submitted by

Michael David Ross

Project Advisor: Bruce R. Land

Degree Date: January 2014

ABSTRACT

Master of Electrical Engineering Program

School of Electrical and Computer Engineering

Cornell University

Design Project Report

Project Title: PIC32 Development Board for Teaching and Prototyping

Author: Michael David Ross

Abstract:

A PIC32 Development Board (Devboard) was designed, built, and tested. The process began with specifying requirements for the device. From there, hardware was selected and the schematic was generated. A layout was handmade for the device following practices meant to minimize noise. This layout was sent out for manufacture and after coming back, was populated and tested. A minimal amount of code was created to test the functionality of the board, and errors in previous steps were corrected. The resulting device is a fully functional development board capable of interfacing with a standard breadboard and with all PIC32 functions accessible.

EXECUTIVE SUMMARY

The purpose of this project is to create a custom platform for developing with a PIC Microcontroller. The intent is to create a system useful for teaching students about embedded systems, and the PIC architecture in particular. For this reason, it is designed to be easy to make, easy to use, and to have enough support that someone with little prior knowledge would be able to make use of it.

The intended end users, students with little experience with embedded systems, dictated the requirements of the project. A student that had never seen a soldering iron should be able to, with some instruction, assemble the entire board by hand. This meant that components would have to be through-hole, or be of a larger surface-mounted type. There needed to be a minimum amount of power regulation on the board, and any usable pins of the microcontroller needed to be broken out to a header compatible with a standard breadboard. Finally, the board should also include other peripherals that might be useful to a student or developer.

A very few microcontrollers in the PIC family exist that are 32-bits and available in a through-hole package. A PIC32MX250F128 was selected, having the largest memory of matching devices. Given that this device has USB functionality, one of the first additions made was a USB port. Other additions were based upon a comparison to the ATmega development board already in use. The PIC microcontroller selected only had half of the available pins of the ATmega, so one addition made was an IO Expander to add more. Another thing that the PIC lacked was a section of nonvolatile data memory, so a suitable EEPROM chip was selected. To minimize the loss of pins, both of these were selected to use an I^{2}C interface, resulting in a total of 2 pins given up. A set of switches was also added to allow users to connect or disconnect these peripherals. A final addition was made of an LED light, which can often be used to verify that a system is running.

With the schematic design finished, a layout was prepared. Components were placed to make them easy to solder, keeping distances between relatively large. Best practices were followed as much as possible to keep the noise in the layout minimal. The entire layout was made to fit on a board 2” wide by 3” long, and then it was verified and sent out for manufacturing.

Once the board came back, it was populated and tested for proper operation. A minimal amount of code was written to test out functions of the microcontroller and its peripherals. Issues were identified and corrected, and the board took its final form.

TABLE OF CONTENTS

INTRODUCTION

MOTIVATION

Currently, Cornell uses a 16-bit MSP430 microcontroller in the Embedded Systems class (ECE3140), and an 8-bit ATmega in Digital Design using Microcontrollers (ECE4760). This means that there is no representation of 32-bit systems or PIC architectures. To solve both of these problems at once, the PIC32 Devboard was envisioned as an option to replace, supplement, or enhance the current selection of development platforms. The goal was to ultimately produce a very powerful development system (high speed and 32-bit words) that would be comparable to the systems already in use.

REQUIREMENTS

The requirements of this system were derived by looking primarily at the existing ATmega Devboard used in ECE4760, and seeking to emulate and improve upon that design for a PIC microcontroller. The following list of requirements was generated:

1. 32-bit PIC architecture
2. Designed with high-frequency signaling in mind
3. Sufficient I/O for varied projects
4. Powered by existing DC sources
5. Accommodates in-system programming
6. Can be assembled with little or no prior experience
7. Interfaces with a solderless breadboard
8. Includes useful peripherals

Requirement (1) was a result of discussion with Bruce Land over what he might like to see in a new development board. The 32-bit architecture was the first goal, to enable students to have a more powerful system to work with. PIC was selected as to not repeat prior work (i.e. make a new board with an existing architecture family).

Requirement (2) evolved from (1). PIC architectures have an internal Phase Locked Loop (PLL) that allows the system to be clocked at up to 50 MHz. At these frequencies, noise can be a serious concern, so it was important to consider high-frequency signals at all points during the design process.

The remaining requirements (3)-(8) emulate features of the ATmega Devboard. Requirement (4) exists so that DC power supplies in use for the ATmega boards can be reused on the PIC boards. Requirement (5) is about ease-of-use in terms of developing new code. Requirement (6) restricts component choices to through-hole packages or relatively large surface-mount packages.

SYSTEM DESIGN

The PIC32 Devboard went through a full design process to go from definition of requirements to functional hardware. For this project, the first step was identification of parts based upon the requirements. Once the parts were identified, they were assembled in a schematic editor. The schematic was then transferred into a layout editor to run traces and prepare the design files for manufacture. Once the board came back in, it was finally populated and tested.

PART IDENTIFICATION

The first major part to identify was the microcontroller to be used. Requirement (1) identified the family of PIC32 microcontrollers. The next important requirement for selecting this part, requirement (6), specified that the microcontroller should be easy to solder. A through-hole package was selected for this requirement, as a chip holder could be soldered in, allowing for quick swaps of broken chips and avoiding direct heat contact to the microcontroller. With these requirements, only two sets of chips remained: the PIC32MX1xx and the PIC32MX2xx. The major difference between these groups was the presence of a USB controller on the PIC32MX2xx group. To enable the largest number of functions to be developed, the USB device was selected – the PIC32MX250F128B in particular, as it had the largest amount of memory. The USB could also be used in place of a serial connection for communications with a host computer.

The next important set of parts to identify was those involved with the power system. The DC supplies from requirement (3) have outputs between 5 and 12 VDC, which have to be dropped down to 3 VDC for the PIC microcontroller. Because power comes from the bench, efficiency is not a significant concern, and so a simple LDO Regulator was chosen. The particular device would need to allow for supply voltages up to 12 VDC, and to output a steady 3 VDC for the circuitry. With a great many devices meeting these specifications, an MCP1702 was selected because the microcontroller was already coming from Microchip. Other parts in the power system include a barrel jack (identical to that used by ECE4760), a basic on/off switch, and two diodes. One diode is for the barrel connector and prevents issues arising when a barrel jack is plugged in having opposite polarity than expected. The other diode is for pulling power from the USB port, and allows a bench power supply and a USB cable to be plugged in without interfering with one another.

The final group of important parts to identify was those defined by requirement (8). To determine what "useful peripherals" should be included, the ATmega Devboard was consulted again, and differences analyzed. One major difference between the ATmega Devboard and the PIC Devboard at this point was number of I/O lines. While the ATmega has 4 ports organized into 8 pins each, the PIC has 2 ports of different size and nonsequential numbering with a total of 17 pins, barely over half that of the ATmega. To counter this problem, an I/O Expander was added to the design, communicating over I2C as to use as few pins as possible. The MCP23008 was selected for this purpose, adding 8 pins and designating 2 as I2C lines. Another difference between the ATmega and the PIC is that the PIC lacks an EEPROM for nonvolatile memory. An external EEPROM chip was selected to solve this issue and give a nonvolatile data memory space for projects. The 24FC512 was used as it was also an I2C device, meaning no additional pins would be used. A final addition was a DIP switch with 3 DPST circuits. This would allow the I2C lines to be connected or disconnected as needed, and also enables extra signaling lines from the I/O Expander to be used. The third switch was connected to the write enable of the EEPROM, allowing it to be physically locked.

Schematic Design

The schematic for the PIC32 Devboard was generated using the gEDA gschem schematic editor. Where default symbols would not work, and public symbols available online did not match, custom symbols were generated.

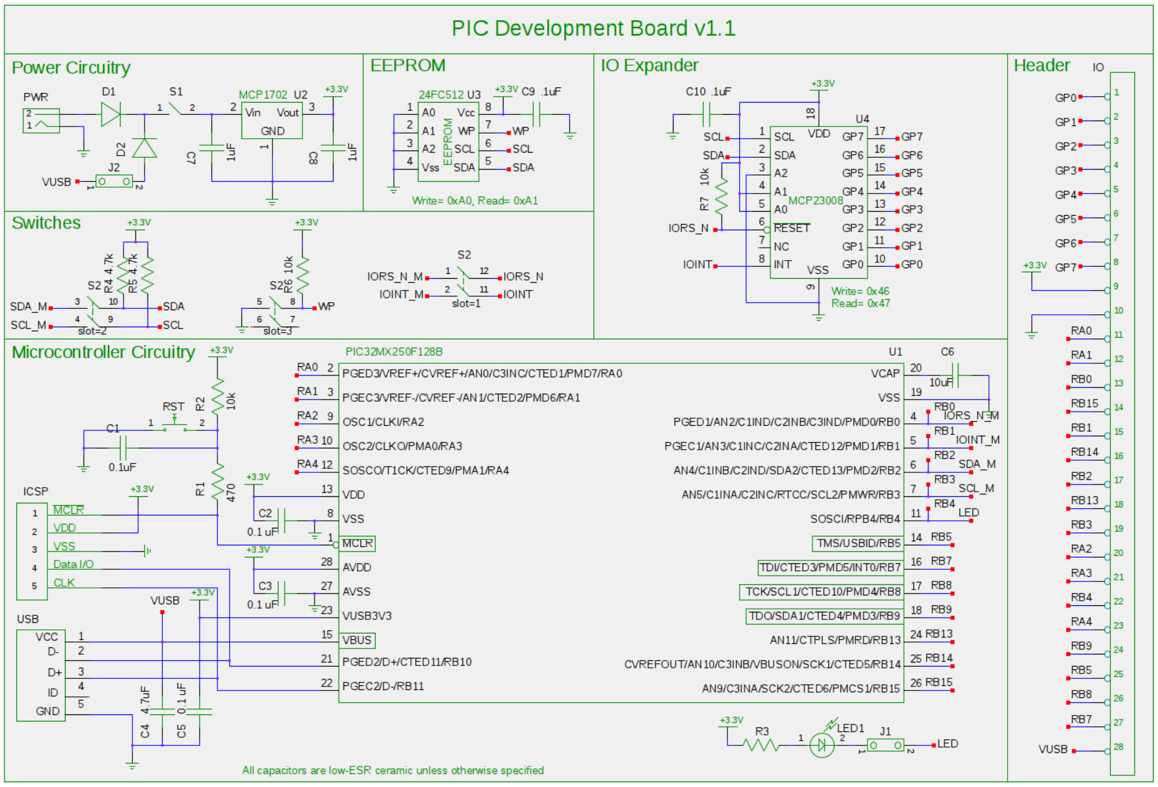

Figure 1. Completed Devboard Schematic

During development, the design went back and forth a few times between the schematic stage and the layout stage. This was in large part due to working out the proper order of output pins on the I/O Header.

Layout Design

Layout design for this project was generated using the gEDA PCB layout editor. The 'gsch2pcb' script was used to prepare the parts and netlist. Where default footprints did not work for parts, and public footprints available online did not match, custom footprints were generated.

The layout for the PIC32 Devboard was created trying to follow practices set forth in "PCB Design Guidelines for Reduced EMI" released by Texas Instruments. Parts were placed as logically as possible, with the I/O Header needing to be on the long edge, the USB socket and power connector going along any edges, and other placements following from those connections.

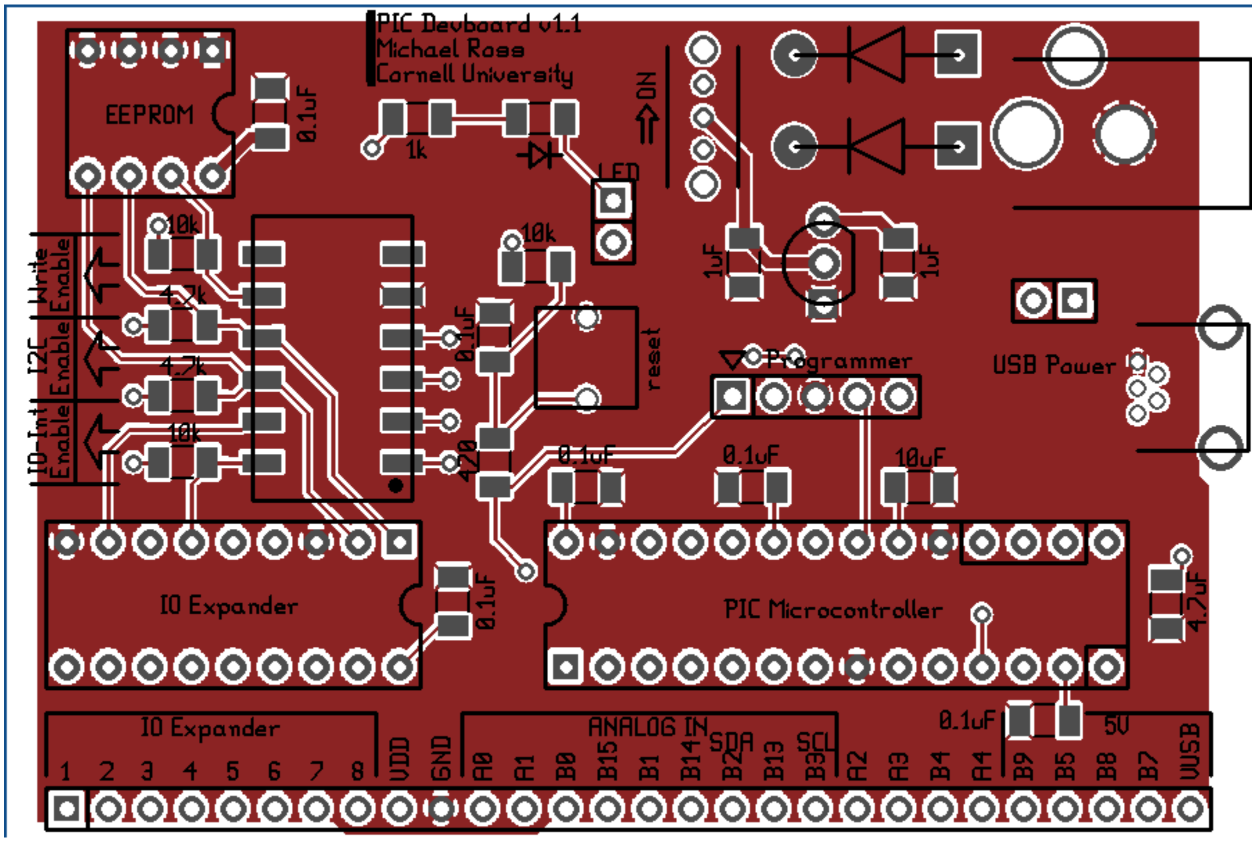

Figure 2. Top-side Traces

The above image shows the top side of the PIC32 Devboard. All parts are populated on this side of the board for the sake of simplicity. The PIC Microcontroller, I/O Expander, and EEPROM chips are all populated with sockets instead of being directly soldered in place.

Most of the topside of this board is one large ground plane. This is one of the major recommendations for reducing EMI, as it keeps signal return paths short and absorbs noise from under the microcontroller. Parts are spread out as much as possible to make assembling the board easier.

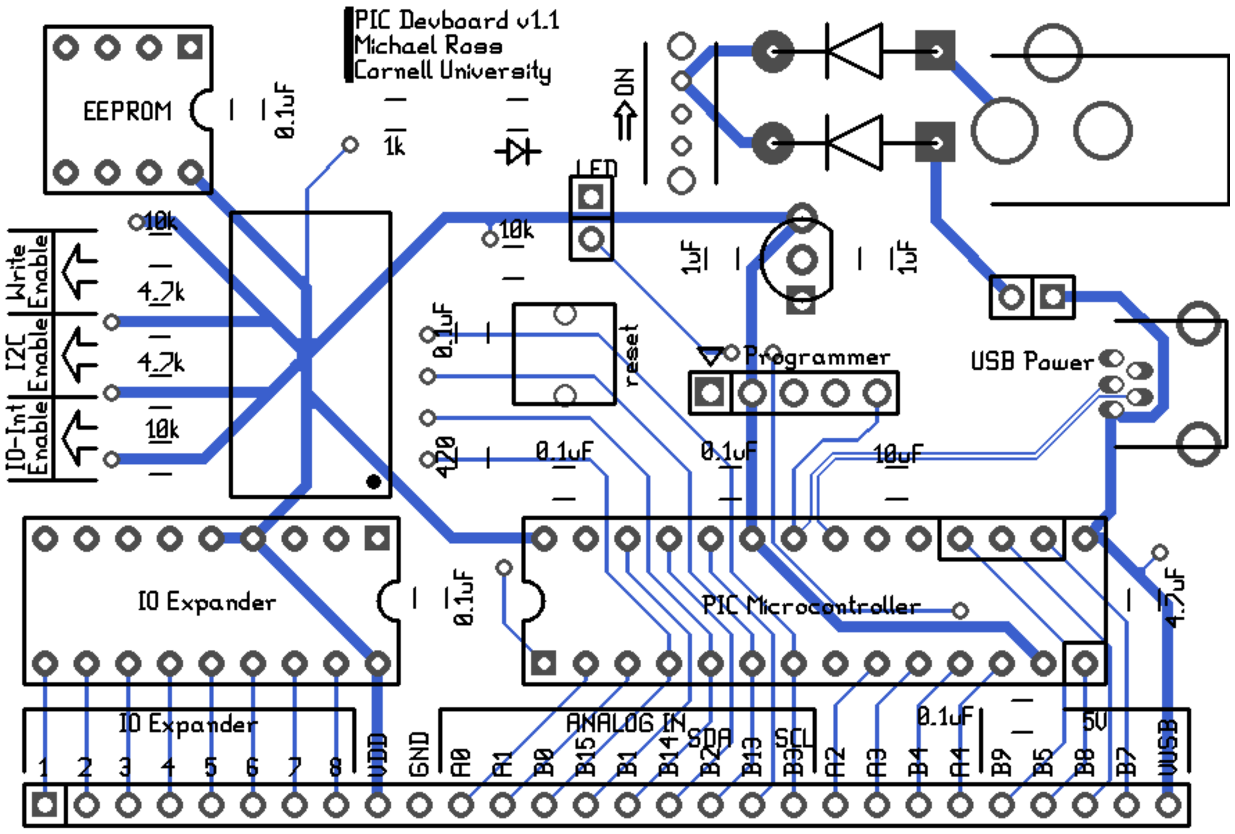

Figure 3. Bottom-side Traces

On the underside of the board, other practices can be observed to reduce EMI. Star distributions are used to deliver power across the board to keep differences in trace lengths minimal. Signal lines are as short and straight as possible, only passing to the other side when unavoidable. Additionally, the differential signaling lines for the USB connector are kept as close together as possible to keep EMI effects even between the traces.

Once the design was completed and reviewed, it was exported to gerber files and sent out to be manufactured.

Population and Testing

After the manufactured boards returned, and all parts were purchased or sampled, the board was incrementally populated and tested.

First to be populated was the power circuitry. Everything from the barrel connector to the LDO was assembled. The board was then plugged into bench power and a multimeter was used to verify correct voltage output. The voltage was within the 0.4% tolerance of the LDO, so the board passed this stage.

Next to be populated was the microcontroller, and all of its associated components. When populated, it would be connected to through the MPLAB IDE for programming to verify that the PIC worked. On the v1.0 board, this test failed, and MPLAB would not connect to the microcontroller. After some debugging, the issue was determined to be an incorrect pinout for the programming header, a result of misreading the PIC datasheet. Rework wire fixed this problem for the v1.0 board, and a fix was put together for the v1.1 board. Once the fix was in place, the programmer was able to connect properly.

Finally, the remaining peripherals on the board were populated. The LED was tested first, with code to light it, and then code to blink it. This verified that the PIC was being programmed correctly, and that the I/O was working. Then the I/O expander was tested with code that set outputs. Finally, the EEPROM was tested by writing a byte and then reading it back.

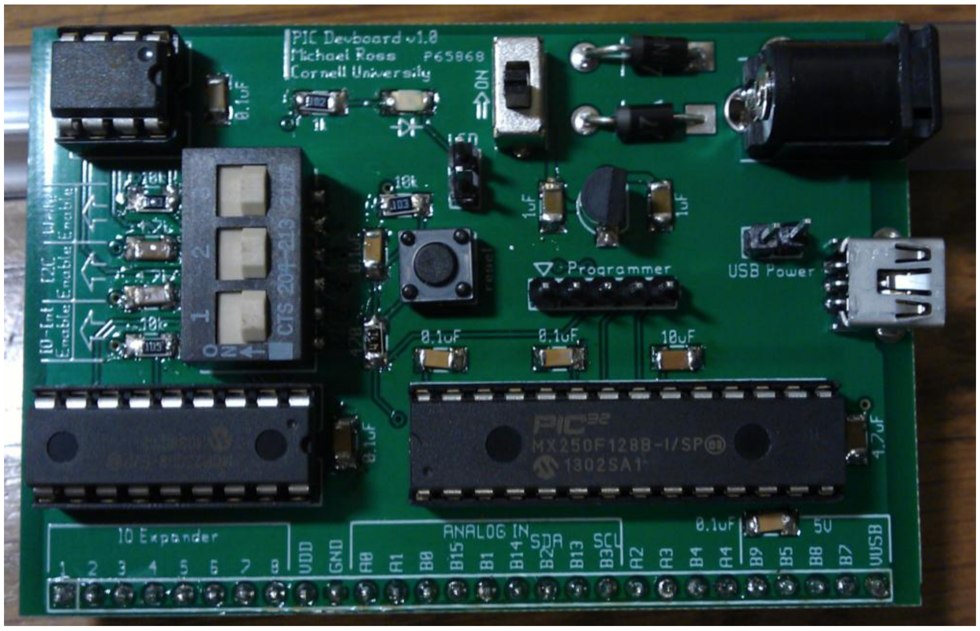

Figure 4. Fully Populated v1.0 Board

Summary and Conclusions

I believe that this PIC32 Development Board is an effective tool for teaching microcontroller programming and doing project prototyping. It has a large number of available features and a form factor that is easy to work with. If I had more time to work with this, I would probably have liked to implement a few test projects like would be used in ECE 4760, and at least one that would take full advantage of the system's 32-bit processing power. As it stands though, it should be perfectly capable of meeting any demands made by classes and users in the future.

Acknowledgements

I would like to thank my M.Eng advisor Bruce Land for helping me with each step of this project, from conception to completion. Bruce is without a doubt one of the best teachers at Cornell, as he is extremely knowledgeable, highly approachable, and inspires an interest in engineering topics in his students.

I would also like to thank Microchip, who makes and from whom I sampled most major components of this PIC32 Devboard.

References

[1] Texas Instruments, "PCB Design Guidelines for Reduced EMI," November 1999.

Resources

???