University of Idaho
College of Engineering

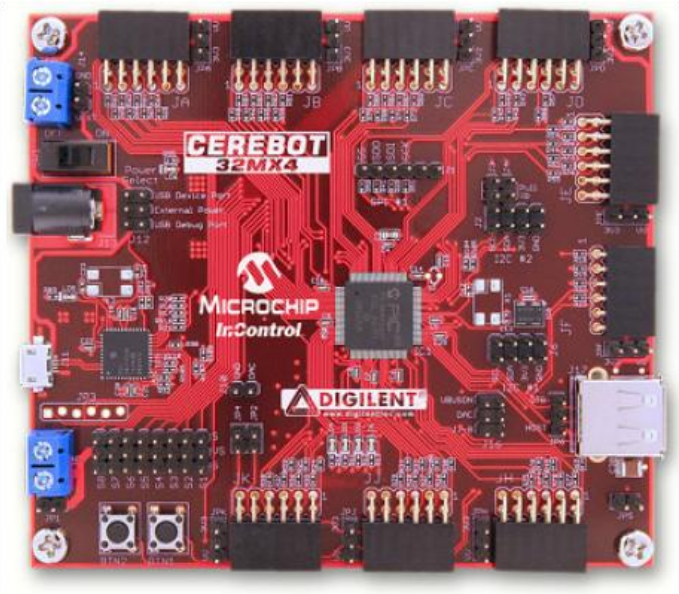
DIGILENT
BEYOND THEORY

Team Gen-X Micro

'Modernizing the microcontroller development tools and educational material used by students at the University of Idaho'

Timothy Melton, Chenchen Li, Sean Robertson
melt3010@vandals.uidaho.edu, chenchenli@vandals.uidaho.edu, seanrobertson@vandals.uidaho.edu

Department of Electrical and Computer Engineering
University of Idaho



Executive Summary

The purpose of this project is to update the hardware and software development tools and educational material used to teach University of Idaho students fundamental microcontroller concepts on an updated platform. Our client and sponsor, U of I professor Dr. Richard Wall has outlined the concepts that students learn in his lectures and must demonstrate in his labs to serve as guidelines for us to use in selecting a new system to replace the current microcontroller system. Some of these concepts include state machine design and control, timing and delays, polling, interrupt servicing, I^2C and SPI communications, and methods of handshaking.

In order to provide a modern, student-friendly set of tools to teach these concepts, Team Gen-X Micro started searching for a base development board that could be used to demonstrate all of the required concepts concurrently. We searched for complete/integrated hardware and software solutions and evaluated systems from well-known, reputable companies such as Atmel, Cypress Semiconductor, and Digilent. In the end we chose a board from Digilent, the Cerebot 32MX4, that uses the Microchip PIC32MX460 microcontroller as the base system for the next generation of microcontroller development tools to be used at the U of I.

In order to provide a system that could be used to demonstrate state machine control using a stepper motor, we were tasked with designing a stepper motor expansion module. Using the EAGLE CAD software we designed the circuit schematic and board layout. The stepper motor board design and layout is nearly finished and will be reviewed once more before being sent out for fabrication early next semester.

In addition to selecting hardware and designing expansion modules (Pmods) for this project, we are also required to design the code to implement the system features taught in Dr. Wall's microcontroller class as well as write the laboratories that students will be using to demonstrate the concepts learned on the new Cerebot 32MX4 board. The code for the first seven labs has been successfully implemented and demonstrated. The actual laboratory material is still in the process of being synthesized but will be completed by the end of next semester.

This project is important for the U of I because it will provide students at both the undergraduate and graduate level with the modern tools needed to design microcontroller-based solutions. The value of this project lies in the flexibility it will provide as a tool which can be used at any level to learn and demonstrate microcontroller concepts ranging from basic fundamentals to advanced research topics. It is our intent that the system we develop has value not only to the students in the microcontrollers lab, but also provides a solid platform for future senior design projects, graduate projects and advanced research.

Table of Contents

1. Introduction

1.1 Background

This project has many stakeholders that are interested in modernizing the microcontrollers lab. Dr. Wall at the U of I is the sponsor for our project and would like to update the lab and make the coursework easier for new instructors and students to be able to pick up. Digilent has become one of our surrogate sponsors and we hope to work with them to market the peripheral boards that we develop based on their microcontroller development boards. We hope future students will benefit from our work by being better equipped to develop modern microcontroller solutions. Finally, other universities looking for microcontroller curriculum may be interested in the coursework that we will be developing.

An integral piece of the electrical and computer engineering curriculum at the U of I is the microcontrollers class and accompanying lab. The objective of this class is to familiarize students with the process of designing microcontroller-based systems. For the last seven years the university has used the now dated Rabbit3000 microcontroller. As technology has progressed there is a need to update this platform to one that incorporates the new industry standards for microcontrollers. These new standards include 32bit MIPS architecture, multiple analog to digital converters and network communications to name a few.

To meet these new standards our group chose a microcontroller development board based on the PIC32MX460 family. The board we chose has many options for input and output but does not have some of the interfaces that the students will use in the microcontrollers class. To remedy this problem, our team is designing new peripheral interface boards that interface with the chosen microcontroller development environment.

The board that is currently used in lab was developed by Dr. Wall with another senior design team. This board was designed from the ground up and is an all inclusive development system. The microcontroller on board is the Rabbit3000—an older 8-bit microcontroller and is not a standard processor that is used in industry. Problems that are currently plaguing the design are the cost of repairs when parts break as well as the cost to develop an entire system from the ground up. Our project aims to remedy these two affordability issues as well as modernize the platform with a new 32-bit industry standard microcontroller.

1.2 Problem Definition

The purpose of our project is to create a new development toolset that incorporates the new industry standard features but satisfies the affordability, maintainability, scalability, observability, usability and sustainability. It is important that the design be affordable so that students can afford to purchase their own boards for work at home. Maintainability allows for quick, affordable repair of the system in the event of failure through good documentation and modular components. Scalability is the ability to add as few or as many optional functions that could be needed. Observability makes it possible to actually measure the outputs with a logic analyzer or oscilloscope on all I/O pins. Usability will be satisfied by having a system with a universal programming language that students will be able to pick up readily and a user-friendly IDE with integrated debugging. Sustainability is defined by our group as the ease of servicing parts on the board as well as the long term support from developers (including documentation).

In addition to these six abilities there is a list of criteria that must be satisfied by the board that we will be using. These requirements outline the concepts that are taught in the lab currently and are necessary for students to learn. These requirements will form the basis for our selection of development boards.

- Integrated Development Environment
    - Debug console
    - Variable watch
    - Breakpoints
    - Single-stepping
    - C-based programming environment
    - Trace mode
- Input/Output
    - Buttons
    - LEDs
    - LCD
- Communications
    - I^2C
    - SPI
    - UART (RS232)
    - Ethernet
- Real-time Control
    - Required
        - Hardware timers
        - External interrupts
        - Timer interrupts
- Mixed Signal Functions
    - Required
        - ADC
        - DAC/PWM
        - Closed-loop linear control

The specific deliverables for the project are the documentation of the boards we create, the laboratory work for students to do, a stepper motor control board and a fully capable system that implements all of the features described.

2. Project Description

2.1 Concept Consideration and Selection

In selecting a solution for modernizing the educational and research tools available at the U of I, two general areas were considered. The first area of consideration is the development tools, including

the hardware, software/IDE and expansion/peripheral modules (Pmods). The second general area of consideration is the educational material and documentation used to aid students in the development and implementation of microcontroller solutions.

2.1.1 Development Tools

Hardware

When evaluating development boards to use for our project, we considered the following features: input/output capabilities, communications, real-time control and mixed-signal processing. For I/O capabilities, we required at least 2 buttons on board, at least 2 LEDs on board and an LCD screen available on board or though an expansion module. We required the ability to use I^2C communications as well as SPI, UART and the ability to add Ethernet if it is not included on board. Hardware timers and interrupts were a must to implement real-time control with the microcontroller. To interface with the analog world, analog to digital converters as well as digital to analog converters were a requirement. We were looking to find a solution that included a feature-rich development board that we could expand on with a C-based integrated development environment. We wanted a system with built-in programming and debugging capabilities, so that little or no additional hardware or software packages were needed for microcontroller development.

When looking for candidates, we started by looking at several well known companies. These companies included Texas Instruments, Atmel, NXP, Microchip, Cypress Semiconductors, and Digilent. When looking for development systems, we quickly eliminated most of the Texas Instruments and NXP boards because of an obvious lack of features. Cypress' PSoC development boards looked to be the most desirable because they were packed full of features and was recommended by Dr. Wall. Digilent had the largest selection of development boards including those based on Cypress' PSoC as well as Microchip and Atmel chips. To determine which board best suited our needs, we put the top contenders into a decision matrix to evaluate which had the features we required and as well as

optional features. The boards that we considered had to meet the criteria of the six abilities (affordability, maintainability, scalability, observability, usability and sustainability) as well as all of the criteria in the list of functionality:

Table 1 Decision Matrix Results

Board Identification					
ID#	MFG	Series	MFG Part #	Micro Implemented	TOTAL SCORE
1	Digilent	PIC Series	Cerebot 32MX4	Microchip PIC32MX460F512L	280
2	Cypress	PSoC Series	CY8CKIT-001	PSoC CY8C28	278
3	Digilent	Atmel	Cerebot II	Atmel ATmega64 AVR	270
4	Digilent	Atmel	Cerebot Plus	Atmel ATmega2560 AVR	270
5	Olimex	Atmel	SAM7-P256	AT91SAM7S256 ARM7	230
6	BD Micro	Atmel	MAVRIC-IB	Atmel ATmega128 AVR	235

The details of the selection can be found in Appendix A. We selected the Digilent Cerebot 32MX4 because it met the requirements of the six abilities as well as the functional criteria outlined.

Expansion Modules

Some of the interfaces that students will be using were not available on the development boards we looked at. The required interfaces for the class were an RS232 connector module, a stepper motor controller board, LCD display module, Ethernet module, large I/O module, and a DC motor control module.

We were able to find prefabricated modules that satisfy the RS232 module, LCD display module, and DC motor control module requirements manufactured by Digilent. In order to satisfy the stepper motor module requirement, we had to design our own board. One feature of our design, was having a set of jumper selectable LEDs tied to the outputs from the microcontroller. As the microcontroller board can only source 200mA in total for all I/O, it would be desirable to make the LED's jumper selectable (enabling us to not draw current to light the LEDs and allowing that current to drive more critical functions). We designed three different methods for driving the LEDs.

The first method we considered was using the output current from the microcontroller output pins to directly drive the LEDs. This was a simple option as it required the least amount of board space and parts.

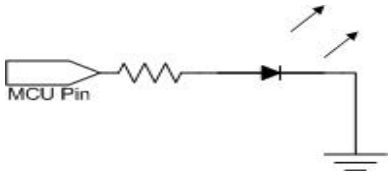


Figure 1 Direct LED Configuration

Driving the LEDs with the current output from the microcontroller pins did not easily give us the option to put a selection jumper on the LEDs without having a separate jumper for each LED. Our analysis of the circuit showed that the current draw for all of the LEDs would be almost 50mA, which significantly reduces the remaining current available for other digital outputs that we may require.

The next option we considered was using an inverter circuit on the output of the pins from the microcontroller and tying the anode of the LED to the positive voltage. This arrangement makes it so that when the output from the microcontroller was logic high, the LED would turn on; and when a logic zero was present there would be no current flow through the LED. This idea had its merits, but required voltage regulation for the inverter and requires more board space than desirable.

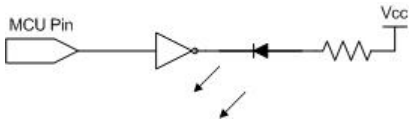


Figure 2 Inverter LED Configuration

The inverter also caused current to be drawn from each of the microcontroller output pins which could interfere with other peripherals that are needed.

The third option we considered was using MOSFETs to drive the LEDs with the output from the microcontroller connected to the gate. This solution added many more parts to be put on the board as a separate MOSFET is required for each LED.

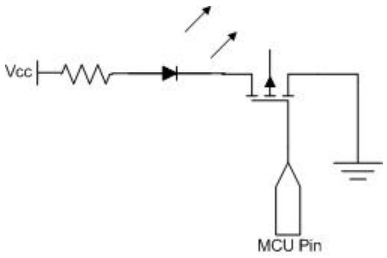


Figure 3 MOSFET LED Configuration

Ultimately, we chose the MOSFETs to drive the LEDs because they require very minimal current from the microcontroller pins (~100nA). This option allows to drive the LEDs with Vcc and not the microcontroller itself.

Software/IDE

When we looked into integrated development environments (IDE), we developed a list of requirements to determine which to use. The IDE needed to have a variable watch list, breakpoints, single stepping, code coverage, and we wanted the language used to be C-based. The variable watch list will aid in debugging code by allowing students to see what the value of variables is at any given time. Breakpoints allow the students to stop code execution at a specific line of code. Code coverage marks each line of code, making it easy to see which lines of code were actually executed by the microcontroller when debugging. Using C for the language allows for greater code portability between

IDEs in case the students want to reuse their code on a different platform later. C is much more standardized than a proprietary or processor-specific assembly language which will make it easier for students who have little or no programming experience coming into the lab.

We selected the MPLAB IDE from Microchip because it met all of these requirements and had more features such as a stopwatch for timing code execution as well as logic analyzers that allow you to view waveforms generated by the microcontroller. MPLAB does not have a built-in C compiler, but Microchip has a free academic version of their C compiler that is transparently integrated into the IDE. The documentation provided by MPLAB is outstanding; every function of the compiler and IDE is easy to find and thoroughly explained.

2.1.2 Educational Materials

Textbook and Labs

The book for the class will be '*Programming 32-bit Microcontrollers in C: Exploring the PIC32'* by Lucio Di Jasio. The selection was made by Dr. Wall because it is a step-by-step introduction to programming the PIC32 microcontroller in C. We outlined the material to compare when concepts are introduced in the textbook as opposed to when the concepts are introduced in lab. Of the 16 chapters, only the first 11 have content in the scope of the class. Below is an outline of the book:

Chapter 1
- Port direction
- Assign ports
- #includes
- Building Projects
- Watch Variables

Chapter 2
- Using timers for polling

Chapter 3
- Control Stuctures(loops)
- Variables

Chapter 4
- Integer division
- Floating point
- Stopwatch

Chapter 5 Interrupts
- Interrupts sources
- Interrupt priorities
- Interrupt handler declaration

Chapter 6 Memory
- Pointers

Chapter 7 Running
- Primary oscillator clock chain
- Peripheral bus clock

Chapter 8 Communication
- Synchronous Serial Interface
  - I^2C
  - SPI
- Asynchronous Serial Interface

Chapter 9 Asynchronous Communications

UART(RS232)
Serial Port Debugging

Chapter 10 LCD
LCD API

Chapter 11 ADC
ADC Configuration
Creating an ADC Library

The current ordering of the labs is slightly different than the outline of the book. Dr. Wall expressed the desire to follow the sequence of the textbook. The labs are currently ordered as follows:

Lab 0 – IDE
Debug console, variable watch, break-points, code stepping
Lab 1 – I/O
Buttons, LEDs
Lab 2 – Timing
Counters, delays, internal timers
Lab 3 – State Machines
Stepper motor control
Lab 4 – Polling
Hardware timers
Lab 5 - Interrupts
Lab 6 – Handshaking
LCD API
Lab 7 – UART
Lab 8 – ADC
Lab 9 – I^2C

These labs must be reordered in the following fashion:

Lab 0 – IDE
Debug console, variable watch, break-points, code stepping
Lab 1 – I/O
Buttons, LEDs
Lab 2 – Timing
Counters, delays, internal timers
Lab 3 – State Machines
Stepper motor control
Lab 4 – Polling
Hardware timers
Lab 5 - Interrupts
Lab 6 – I^2C
Lab 7 – UART
Lab 8 – Handshaking
LCD API
Lab 9 – ADC

One consideration that we had to look into for the labs is a standard coding convention. There are two methods for coding in MPLAB. One is to use the built in macros that are provided by MPLAB and the other is to directly access the special function registers and set the appropriate bits. A good example of a macro that is setup by the Microchip developers is the 'mPortASetDigitalInput(BIT_1 | BIT_2).' This macro sets bit one and two on port A to be digital inputs. The same functionality can be done using the statement 'TRISA = 0x0006.' The advantage to the second statement is that it is more concise. The advantage to the first example is it is more verbose which gives it easier readability. It is

important that when we write the labs that we use the same convention universally, otherwise students may get confused when looking at examples. The macros are what the textbook uses so we will follow suit and develop the labs with the macro approach.

2.2 Selected Design

The value of the system that we have developed lies in its flexibility and versatility. Critical to our system selection was the six abilities; in particular, affordability and scalability. The Digilent board exemplifies these characteristics. The Cerebot 32MX4 is designed with scalability and affordability in mind. All of the microcontrollers I/O lines have been routed in groups of eight to edge connectors with ground and power available at each port. Digilent, a local company, has already developed add-on Pmods that contain the functionality that is needed to demonstrate a majority of the concepts taught in the current microcontrollers lab at U of I.

Although the Pmods that Digilent currently markets provide a majority of the capabilities that we require, we still needed additional functionality. One of those functions is a stepper motor controller, which is currently used to demonstrate state-machine concepts. In order to provide this feature we have worked to design our own stepper motor Pmod. To drive the stepper motor, we chose to leverage off of the existing microcontroller board and decided to use the ST Micro L293DD surface-mount stepper motor controller IC (see Appendix E). The stepper motor board has become our design task for first semester.

While designing the stepper motor board, we compiled a bill of materials. We choose to go though Digi-Key for all parts because we could get them all from one distributor and keep our shipping costs to a minimum. For a complete list of the parts used to build the initial stepper motor board, see Appendix D. The total parts cost for each stepper motor board comes to $17.50.

Our estimated project cost for two full systems is just over $1500. This includes the fabrication and population of our custom stepper motor board, two 32MX4 development boards, two 32MX7 development boards, the Digilent LCD, RS232, Ethernet and H-Bridge Pmods, two Digilent linear motors and two presentation posters. A sales tax of six percent was estimated and ten percent for shipping costs. The total estimated cost for the project as of the end of first semester comes to $1796. See Appendix C for the full estimated project budget.

Another aspect of this project is the educational resources. Digilent is committed to providing affordable educational tools including development boards, example code and written educational material. One of the six abilities we focused on when selecting the Cerebot 32MX4 board was affordability. At only $80 a board, students can afford to own a personal development board, which can be used for pre-labs, additional learning at home, or class projects.

Also, the new textbook is a very affordable at only $51 – much lower than the cost of most college textbooks. Some desirable aspects of the new textbook are that it starts with an overview of C concepts, and builds up to advanced peripheral configuration and implementation. A student can easily pick up this textbook and start learning how to program microcontrollers with little or no background in C programming.

3. Future Work

To complete our project we still need to complete the following items:

- Review and fabricate the stepper motor module.
- Update the ECE341 labs to be tailored to the new microcontroller board.
- Evaluate the next generation Cerebot board (32MX7), which provides onboard Ethernet, as an upgrade from the Cerebot 32MX4.

- Implement Ethernet Pmod for use with Cerebot 32MX4 board if we decide not to upgrade to the 32MX7 board.
- Complete the following items as time allows (additional expansion modules):
    - Implement a linear feedback controller Pmod.
    - Implement a traffic controller system Pmod.
    - Implement micro-stepping with our stepper motor Pmod.
- Continue documenting the implementation of our Pmods.

Our budget already has funds allocated to complete two additional expansion modules. Continued support from Digilent will allow us to work with the Cerebot 32MX7 and implement the on-board Ethernet or their Ethernet Pmod with the Cerebot 32MX4 board. The greatest risk we face is running out of time to complete all these objectives. Therefore, we will prioritize our work by focusing on the core requirements of updating the labs and implementing the Ethernet first and then tackling the task of designing and implementing the two additional expansion modules as time allows.

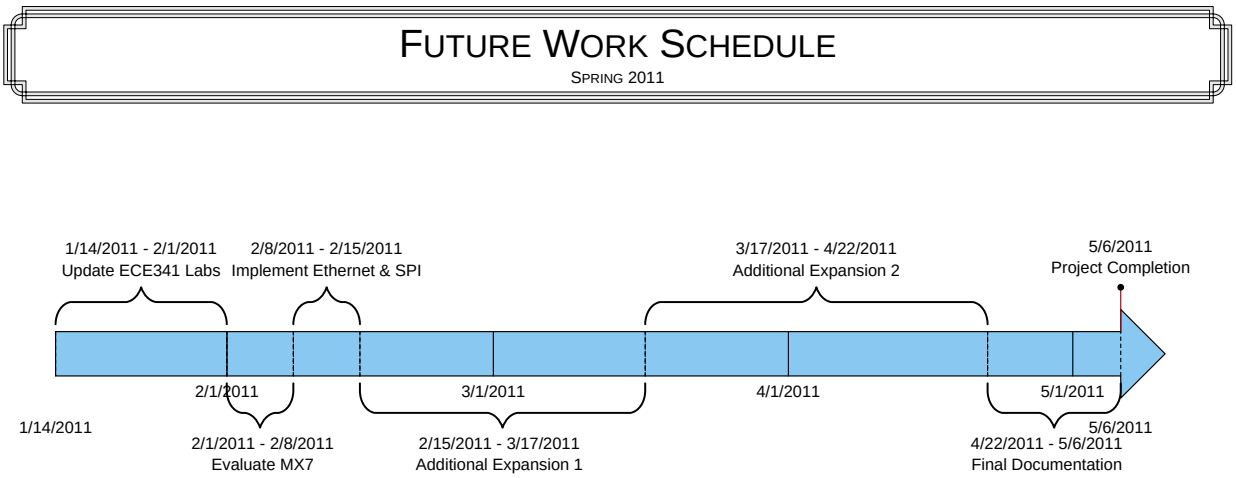


Figure 4 Future Work Schedule

Appendix A: Decision Matrix

Appendix B: EAGLE Schematic and Layout

Appendix C: Budget

Appendix D: Bill of Materials for Stepper Motor Pmod

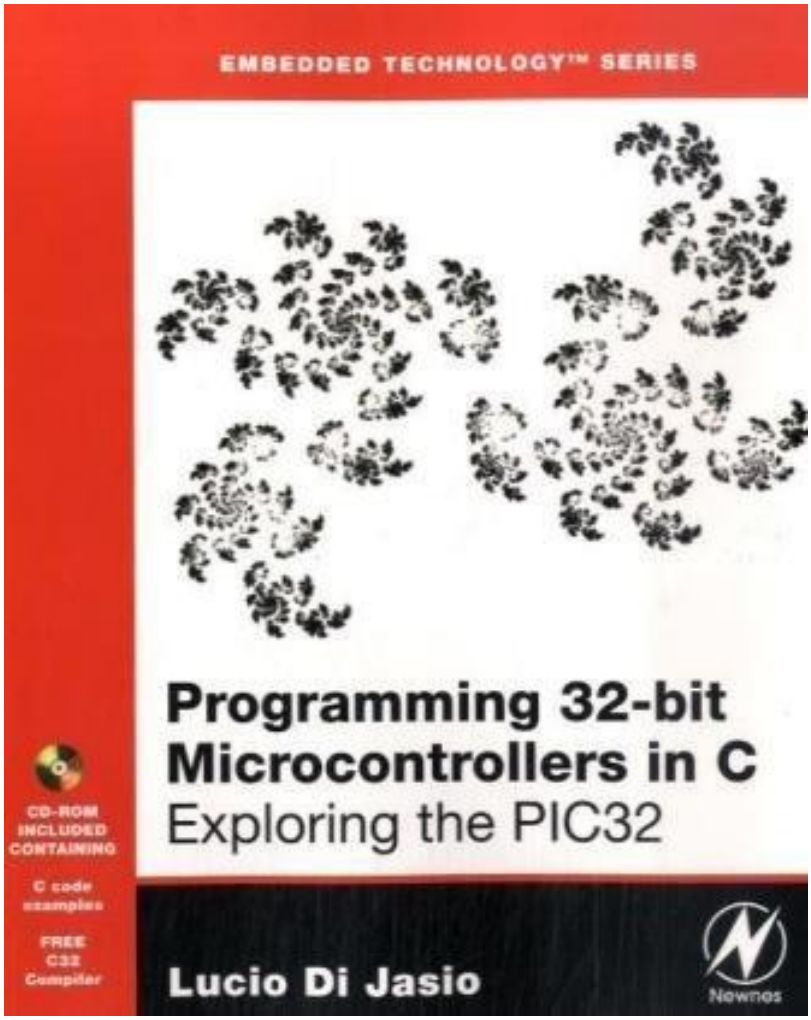
Appendix E: L293DD Datasheet

Appendix F: Parallel Display LCD Pmod Datasheet

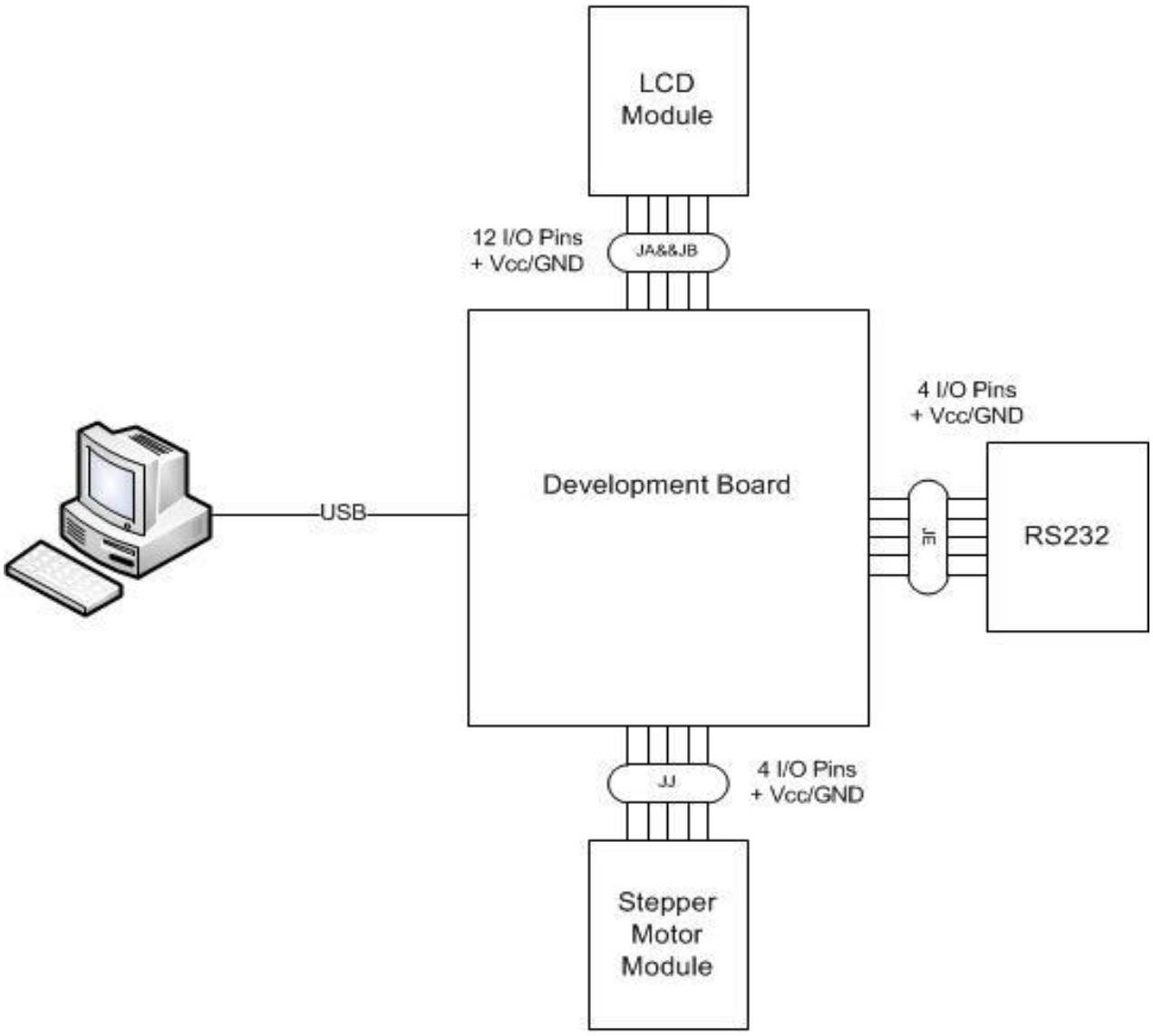
Appendix G: RS232 Pmod Datasheet

Appendix H: Cerebot 32MX4 Board Datasheet

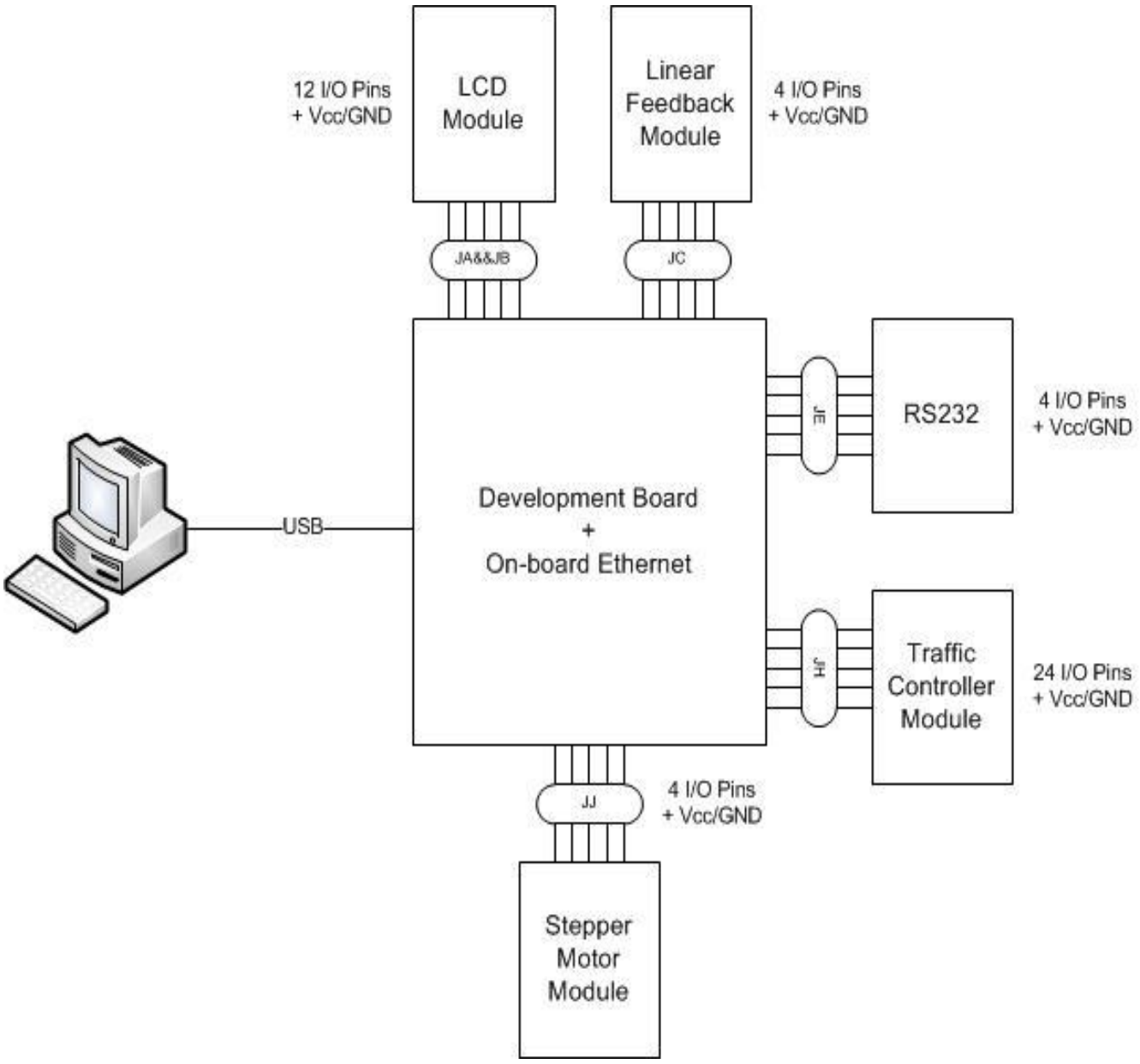
Appendix I: Programming 32-bit Microcontrollers in C



Appendix J: Current Setup



Appendix K: Projected Setup



Appendix L: Example Code