Pico Commander

Final Report and User Guide

Author: Albert Jansen

Prepared for: Randy Gallant

Course Code: ELIC350

Date: April 28, 2016

Table of Contents

Final Report

User Guide

Appendices

Introduction

Final Report and User Manual

The Pico Commander Final Report is a combination of a project report and user manual. The first half of the report focuses on the technology included in the project. Project planning and technological challenges are discussed alongside the solutions to those challenges. Planning for future development and recommendations are given so that moving forward, the project can run more smoothly.

The format of the final report is as follows:

1. A general overview of all technologies, and a discussion of problems and solutions at the system level
2. A closer examination of the software. Discussion of relevant technologies, problems and solutions.
3. A discussion of the firmware and hardware, as well as any changes and challenges that were encountered.
4. A project planning overview, including pitfalls and recommendations for moving forward in future development.
5. A preliminary user manual for both the hardware and software. Safety ratings and warnings, and a link for further reading.

Pico Commander

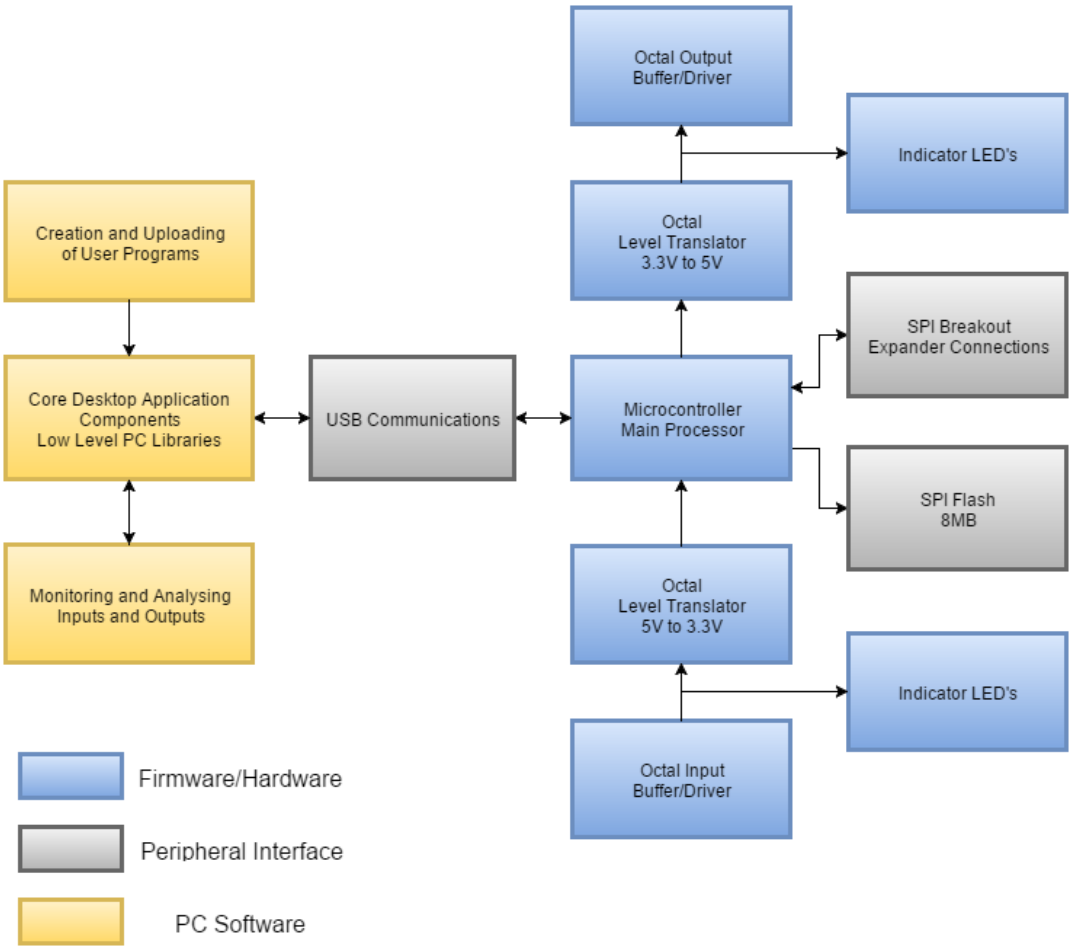
The Pico Commander is a PC to systems interface, providing solutions to control problems through software, firmware and hardware. Using a PC based program, users can create programs using a drag and drop interface. They can then upload these programs via a USB connection to the Pico Commander module. The Pico Commander module then analyzes these commands and runs the program in a loop.

The programs have full control over the 8 digital inputs and 8 digital outputs on the Pico Commander module. Further, the hardware module has 8Mb of Flash memory to store programs, greatly increasing the size and complexity of programs that users can upload.

Finally, the Pico Commander hardware and software are agnostic to one another. It is possible to create many different boards that can use the Pico Commander specification. This combination of flexibility, ease of use, and access to lower level technologies when needed makes the Pico Commander the perfect platform for control solutions.

The System

Block Diagram and Project Technologies



The block diagram on the previous page contains the core components of the Pico Commander system. The PC Software side of the system is written in Python using PyQt4, and communicates over a virtual serial port connection using USB to the microcontroller. The microcontroller was selected for it's on-board USB. There are D+ and D- pins inherent on the microcontroller, and the entire USB stack is exposed via libraries in the firmware.

The octal level translators and buffers were added to increase the output drive and sink capability of the Pico Commander system. LED's were included on each input/output for easy system monitoring, and SPI flash was added as well to store programs permanently, and increase the size of available user programs.

System Level Design Challenges and Solutions

1. PCB Design and Prototyping

   Problem

   A large part of the project design revolved around creating a printed circuit board that could perform that tasks and host the technologies associated with the project. Including the main microcontroller, there were 6 different IC's, (9 IC's in total). This number of components (and their associated passives) were very difficult to fit onto a PCB of a reasonable size to reduce manufacturing costs. Further, the system complexity was increased greatly. All of the IC's involved were connected in some way, many of them directly. This increased the chance a single failure could damage the entire system.

   Solution

   The design of a single board was reserved for future development. Several of the subsystems had boards created, so they could be tested independently. This modular design decreased prototyping costs, as well as the cost of circuit error. The subsystem PCB's are discussed further in the hardware section of the report.

2. Version Control and Systems Level Testing

   Problem

   Throughout the project consistent use of Git was essential in maintaining separate workflows for hardware, software, and firmware. However, when it came time to test, these

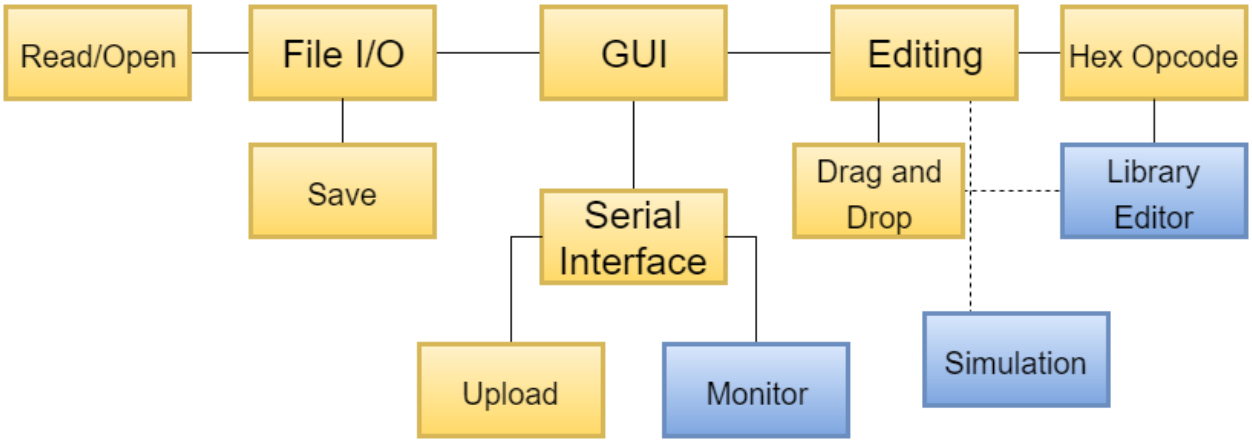
separate branches, still in development, were difficult to manage as a single master branch.

Solution

Using other technologies that perform in similar ways, tests can be done without interfering with the development of other systems. For example, in order to test the PC side of USB communication with the Pico Commander, the firmware must be complete. However, using an Arduino to create a virtual serial port and some test programs is a good analogue for the Pico Commander module communications. Using these different methods of test, the branches were kept separate for longer, and their development was kept separate. This led to better development organization, and more modular testing.

The Software

Software Block Diagram



The diagram above gives an overview of the software and how it was designed. The yellow blocks are sections that are complete: They are fully functional and available in this first release. The blue blocks are features that will be added in the future.

As shown in the diagram, everything stems from the main Graphical User Interface (GUI). From the GUI, one is able to edit files in two ways: Drag and Drop, and Hex Opcode. The Drag and Drop format allows users to move blocks around a workspace with their mouse, and connect them to make programs. The Hex Opcode version allows users to enter hex style commands directly.

Additionally, the GUI allows the user to connect to the module and upload programs. In the future, it will also allow for monitoring of those programs, with the results being displayed on the screen. This upload function makes use of a virtual serial port to pass data back and forth (the communication rates can be slow, 9600 baud, as the amount of data being passed back and forth is small).

Finally, the File I/O section ensures that files can be opened and read, as there are several different file formats being used in any Pico Commander project. These file formats and their uses are explained in further detail in the User Manual accompanying this report.

Technologies Used

All of the PC software was programmed using Python, and many of the freely available Python libraries. The table below lists the software technologies and their purpose.

Technology	Use
Python	Programming language used to create the PC software.
PyQt4	A GUI toolkit ported for Python. Used for GUI creation.
PySerial	Provides libraries to connect to serial ports through Python.

Software Challenges and Solutions

1. USB Bulk vs. Serial Port communication

   Problem

   Initially, the design called for the use of a bulk USB transfer from the PC to the Pico Commander module. However, the PyUSB library proved to be in it's early stages of development, and with constant updates, the library had many inherent small bugs that needed fixing. It proved too unreliable to be used for rapid development, which was one of the main reasons Python was selected in the first place.

   Solution

   Instead of using PyUSB, PySerial was used instead. The functionality is slightly different: PyUSB provides a full USB style connection, while PySerial emulates and connects to serial

ports. This change meant that more work would need to be done in the firmware, but the end result would be more stable, since PySerial had been released for several years, and no longer had as rapid, active development.

2. Graphical Toolkit – PyQt4 vs Kivy

   Problem

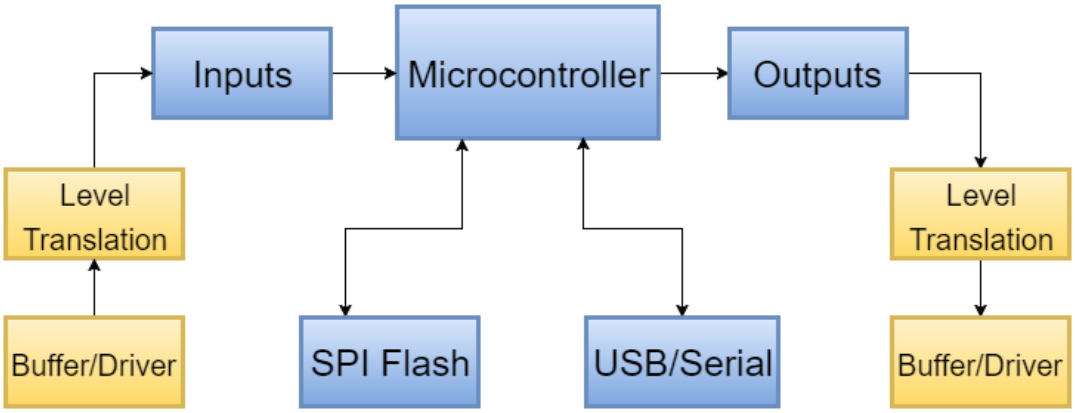
   Initially, the graphical toolkit that was selected was Kivy. It works with Python, and is designed to create modern, clean interfaces for programs. However, with the Python distribution being used for the project, there would be version discrepancies between different libraries.

   Solution

   Instead of Kivy, PyQt4 was selected. There is significantly more documentation on PyQt4, and it has been a stable library for much longer. After talking with the developers of Kivy, it may prove useful for a future project, however for Pico Commander it was not the best solution for the GUI toolkit.

Hardware and Firmware

Hardware Block Diagram



The block diagram on the previous page shows the signal paths and major components of the firmware and hardware on the Pico Commander module. The yellow blocks represent components that are purely hardware, while the blue blocks contain a mixture of firmware and hardware design.

The main purpose of the input and output stages is to provide a higher power output than that native to the microcontroller. The IC's used are a 3.3V-5V level translation IC, and a 5V, 25Ω line driver IC. Combined, these two IC's provide a more stable input and output capability, and make the Pico Commander a solution to a wider range of problems.

The microcontroller itself is a TMS320F28069. This microcontroller offers real time control capabilities, as well as built in USB. This allowed for a better learning experience with USB protocol, and also allowed for exploitation of features in the silicon that are unique to the field of control.

The SPI flash was used to increase program memory size, as well as increase the time that a project can be monitored. Along with the USB connection, it performs all the tasks related to the management of a user's program.

Finally, two PCB's were created. One was created to manage the SPI flash and USB connection, and another was created for the input and output stages. Both of these PCB's are used in conjunction with the dev board to create a single unit system. The buffer PCB conforms to the Texas Instruments Booster Pack standard, and will retain its functionality across the line of Launchpads.

Technologies Used

Every chip selected for the project was manufactured by Texas Instruments, except for the SPI Flash, which was manufactured by MicroChip. The firmware incorporated new code, as well as re-used code from examples provided with the development board.

Technology	Part Number	Purpose
Microcontroller	TMS320F28069U	To execute user created programs and manage peripherals such as USB, SPI, and the I/O lines.
Buffer/Driver	SN64BCT25244	Used to drive inputs or outputs at 5V, 80mA-180mA
Level Translator	SN74LVC4245A	Translates 5V to 3.3V for the inputs to the microcontroller

Technology	Part Number	Purpose
SPI Flash	SST25VF080B	Store user programs and monitoring data
One Bit Level Translator	TXB0101	Improve GPIO efficiency by managing multiple different enable pins from a single output.

Hardware and Firmware Challenges and Solutions

1. Design Errors on Received Boards

   Problem

   Throughout the design of several PCB's, errors were noticed after manufacturing and during testing. These errors were sometimes fatal to the design, and towards the end of the project, the budget had been depleted, not allowing for a re-fabrication.

   Solution

   The first PCB failure was absolute: There were several critical errors with the board that were either impossible to fix, or doing so would prove to take more time than re-ordering a new board.

   After the first failure, a re-evaluation was done. Two smaller boards were created, reducing system complexity significantly. There were no errors with the USB and SPI Flash PCB, however, there were several errors on the Input/Output buffer PCB. These errors were able to be corrected through creative soldering and rework of the PCB. After testing for errors, the IC's were removed, and new order of IC's was made, and they were replaced. This reduced the rework cost (instead of ordering a new PCB), and provided a testable, working module.

2. Lack of USB Source Code

   Problem

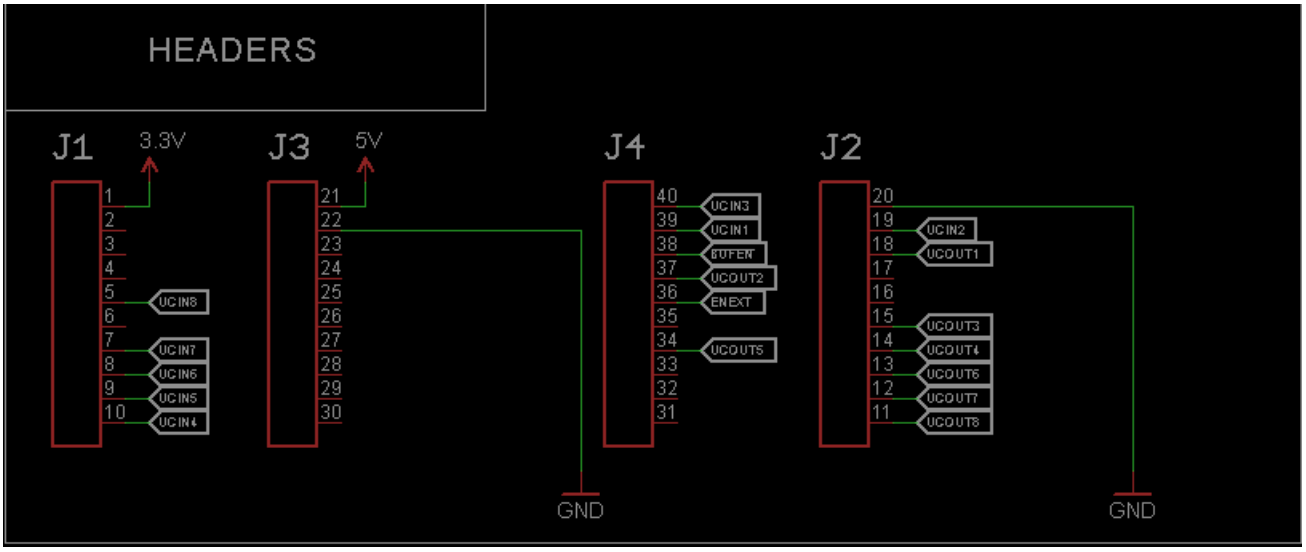
   The source code for the microcontroller was extensive; there were several libraries and examples for every feature of the microcontroller. However, there was no example code or library support for the transfer of data to and from the microcontroller over a virtual serial port using the built in USB connection.

Solution

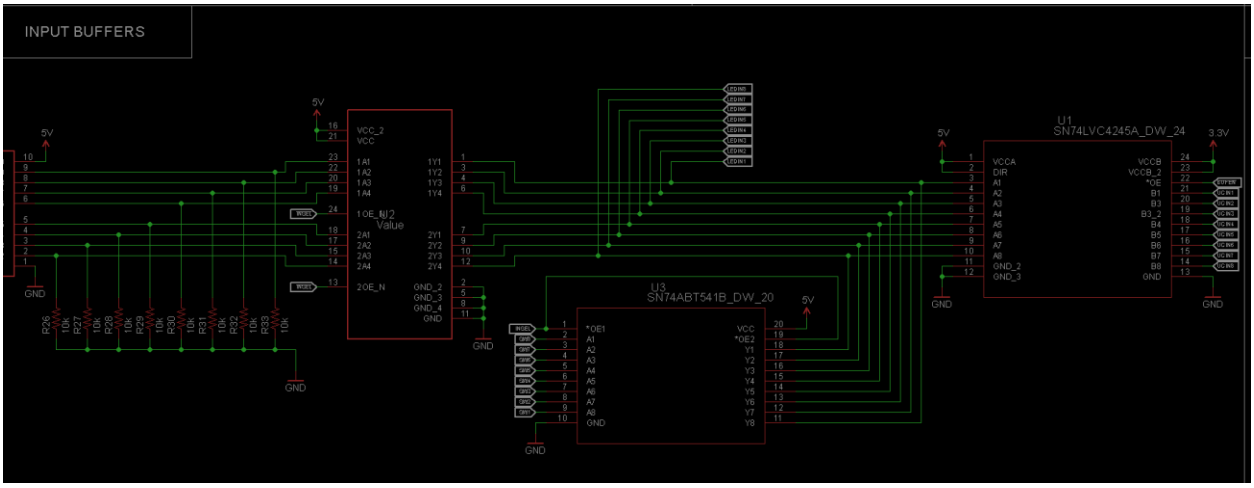
An older version of the compiler and microcontroller contained libraries that could be refactored and used for the current version. The example code and libraries were updated, and modified to work with the most recent compiler and microcontroller.

Schematics and Board Layouts – Input/Output PCB

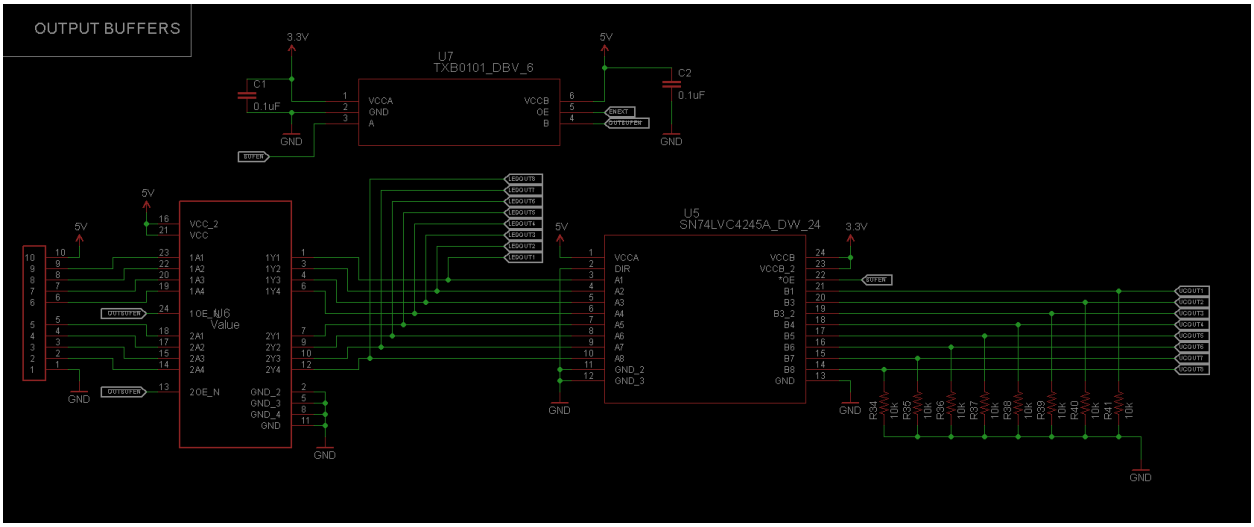
The following schematics and layout are for the Buffer BoosterPack, which provides the input and output capability of the Pico Commander.



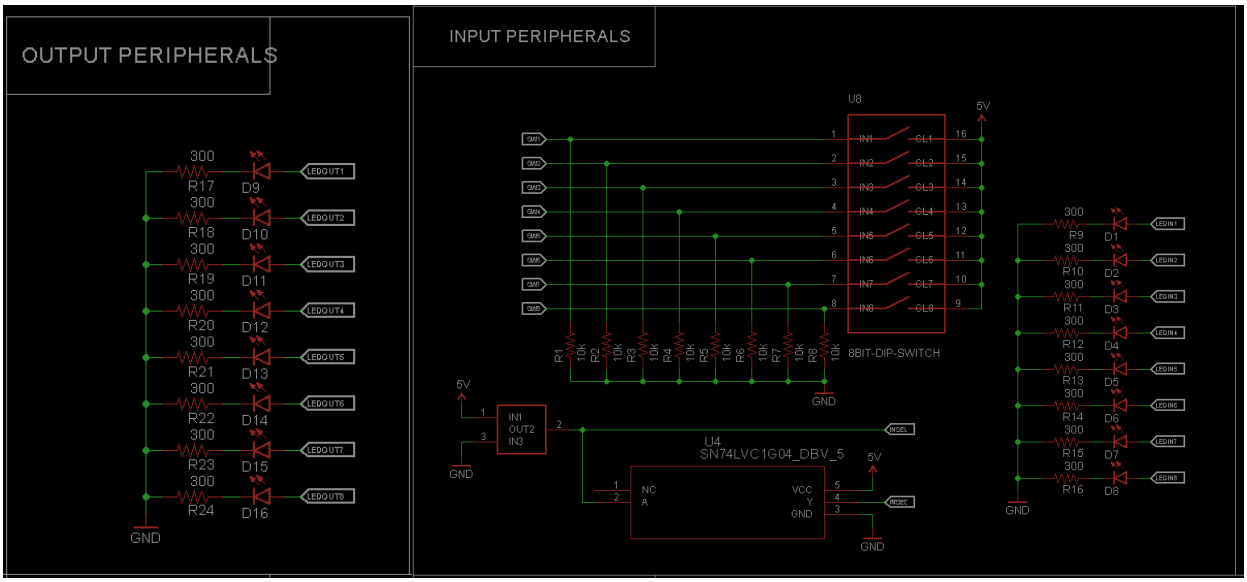
The above image displays how the connections are made to the development board, under the TI LaunchPad line.



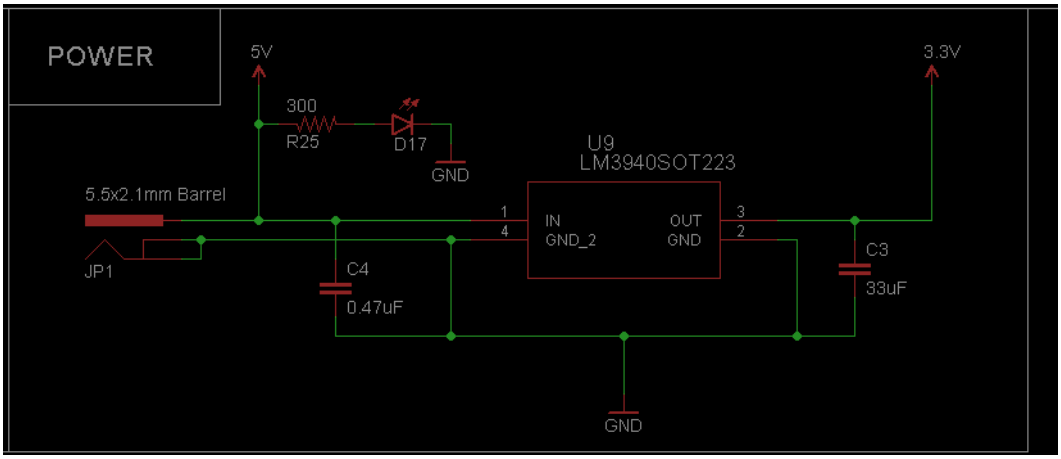
The Input buffer schematic is shown on the previous page. From left to right: Input connector with pulldown resistors, the 5V buffer line driver, an 8-bit bus which is connected to the indicator LED's, the toggle switches, and the 5V->3.3V level shifter. The level shifter feeds the microcontroller the 8 inputs.



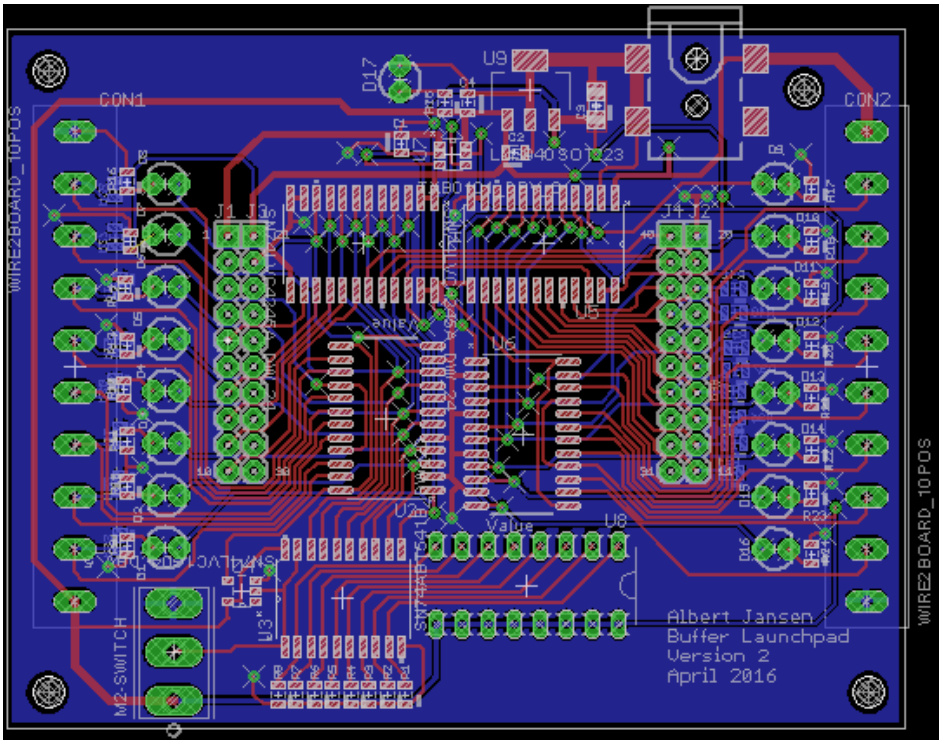
Above is the output buffer schematic. From right to left: The microcontroller has pulldown resistors and is connected to the 3.3-5V level shifter, which feeds the indicator LED's and 5V buffer line driver. The buffer line driver is connected to an output connecter. At the top of this section is a 1-bit level shifter: This allows the 3.3V enable signal on the level shifter to also activate the 5V enable signal on the buffer driver.



On the previous page the output peripherals and input peripherals are shown. These peripherals are the LED's and toggle switch connections. Additionally, in the input peripheral schematic is a 1 bit NOT gate. This gate allows chips that have an active low enable pin to be activated at the same time as the chips with the active high enable pin with needing a new GPIO.



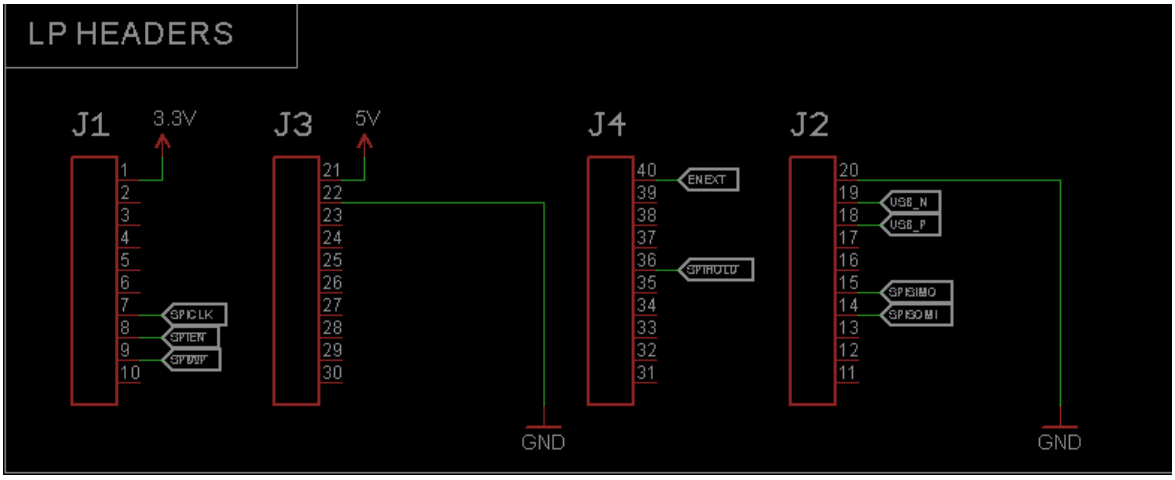
Power regulation is achieved with the use of a LM3940 5V-3.3V LDO regulator. There is an indicator LED to indicate when power is connected. The main power connector is a barrel jack.



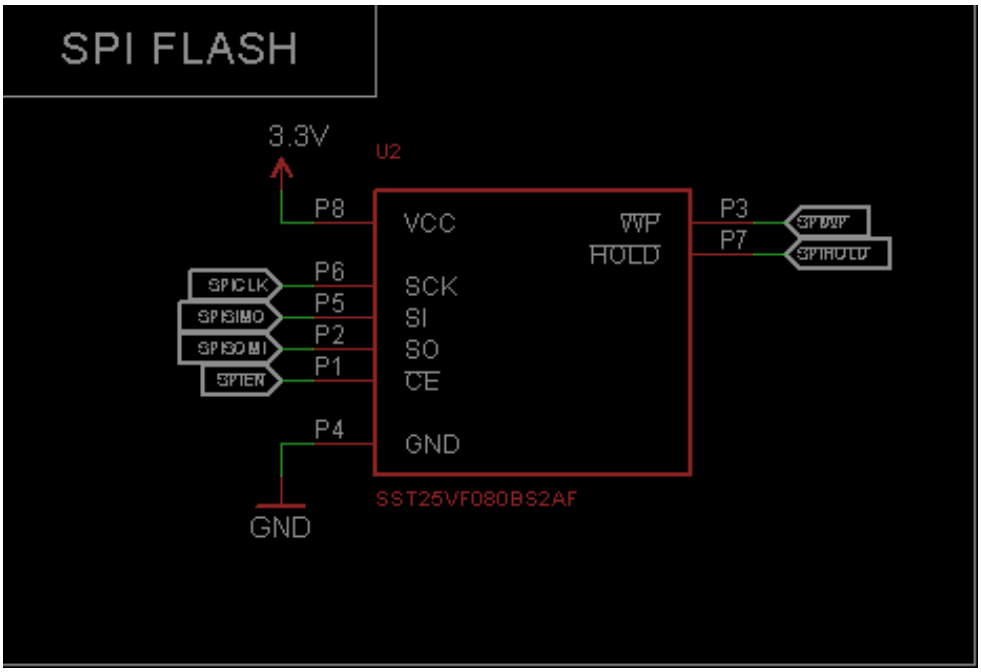
Above is the PCB layout of the buffer booster pack.

Schematics and Board Layouts – USB/SPI Flash PCB

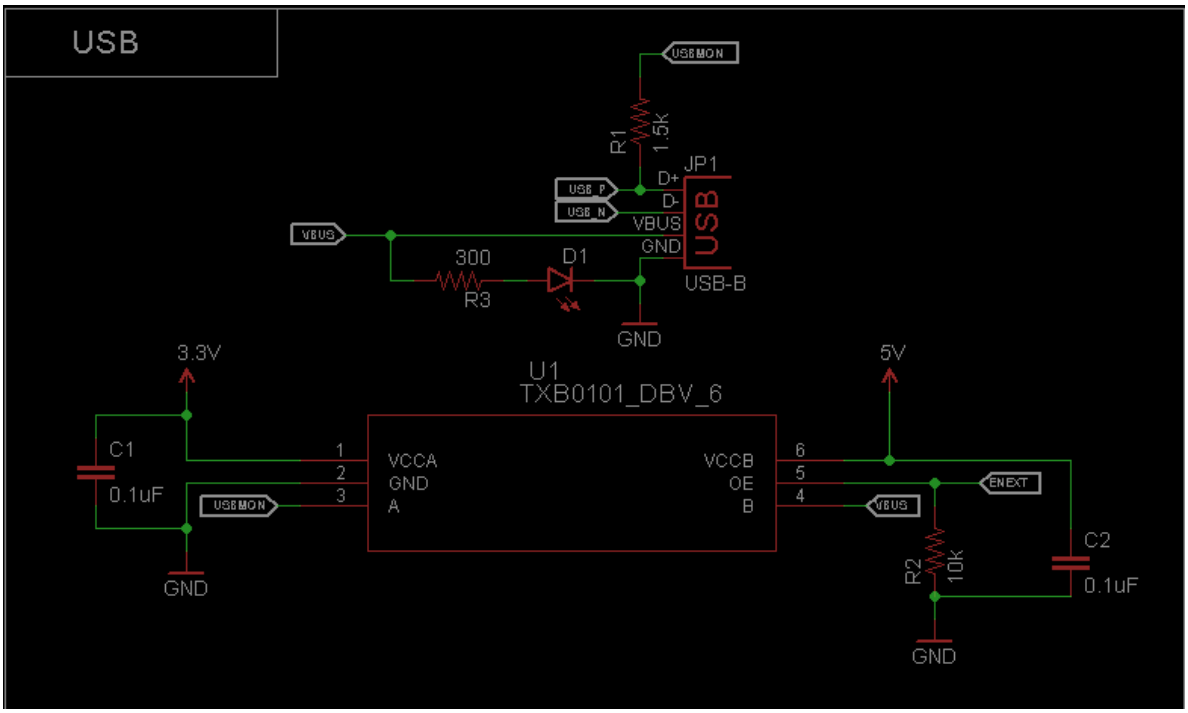
The following schematics and layout are for the USB Booster Pack, which provides breakout access to the USB pins via a USB connector, and an SPI connection for the FLASH memory.



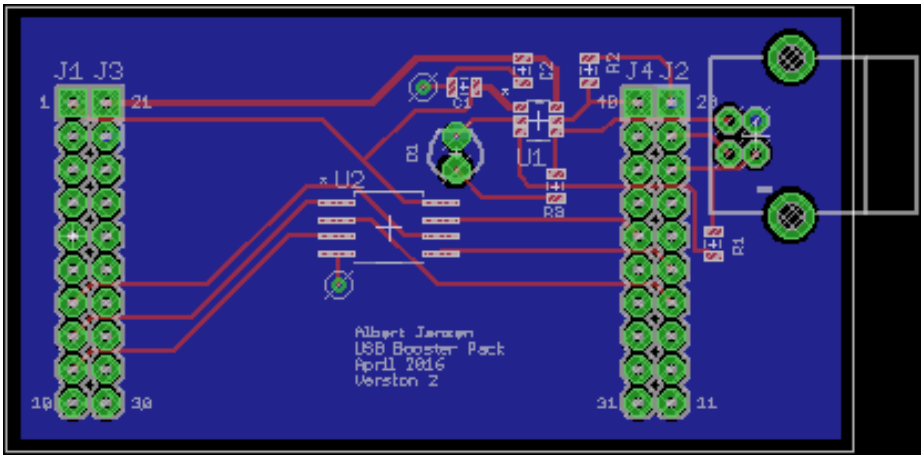
The above image shows the LaunchPad headers, which connect the Booster Pack to the development board.



The above image displays the SPI Flash connections.



The above schematic shows the layout of the USB connection. The connector is displayed at the top of the schematic. The 1.5kΩ resistor to the USBMON signal provides the necessary indication that this is a full-speed device. The indicator LED indicates whether the USB is connected. The 1-bit level shifter below the connecter is used to translate the 5V to 3.3V. This translation takes place to ensure that the microcontroller can remove power from the D+ and D- lines within 10s of removing the USB connecter, as per USB device specification.

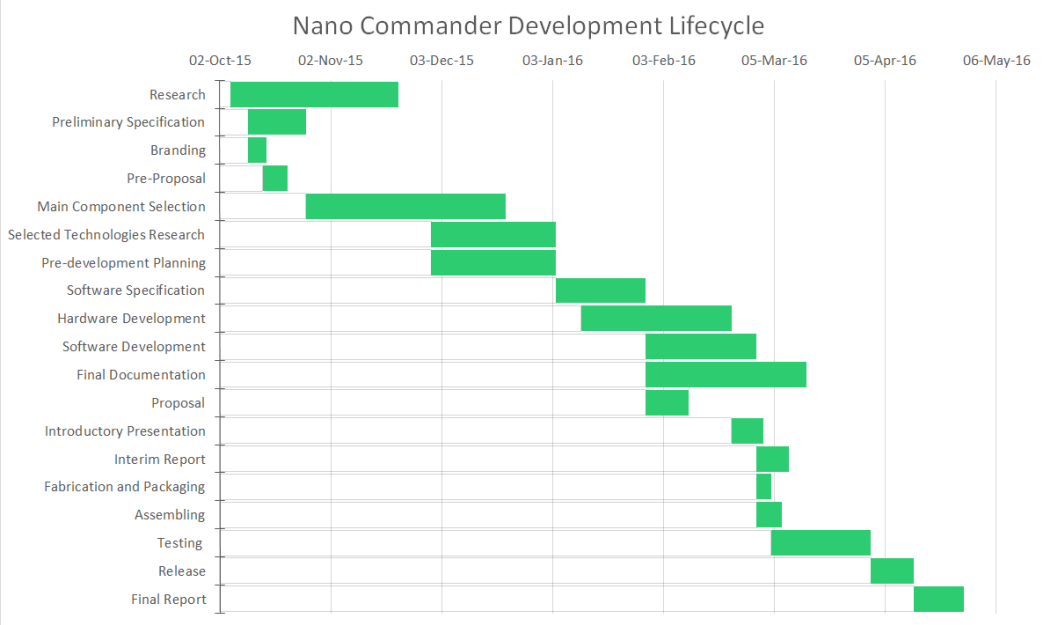


The above image shows the layout of the USB Booster Pack. The USB connecter is a USB female Type-A.

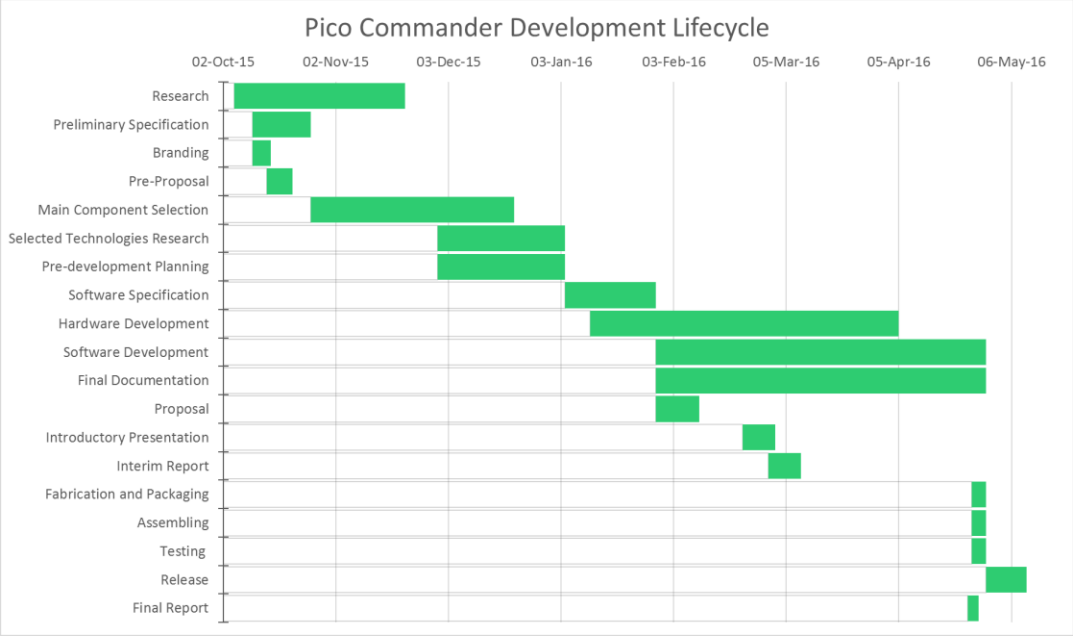
Project Planning

Gantt Chart

Initial Gantt Chart



Realized Gantt Chart



The first Gantt chart on the previous page shows the initial expected timeline. The second Gantt chart above has been updated with the actual times, and changed in several ways. The largest changes have come in software and hardware development, which have taken significantly longer than initially anticipated. This has forced the packaging, assembling, and testing stages to shrink, with release happening the weekend before the presentation.

Budget

Cost Type	Part Name	Quantity	Cost	Status
Incident	Wall Wart Power Supply	1	13.58	Purchased
	F28069 LaunchPad Development Kit	1	38.12	Purchased
	XDS100v2 JTAG Debug Probe	1	100	Purchased
Active Components	C2000 Piccolo Microcontroller	1	28.83	Purchased
	Level Translator	2	3.92	Purchased
	I/O Buffer/Driver	2	20.98	Purchased
	1-Bit Level Translator/Buffer	1	1.16	Purchased
	8-Bit Level Translator/Buffer	1	2.86	Purchased
	5V Octal Buffer	1	1.26	Purchased
	2:4 Active Low Decoder	1	0.85	Purchased
	8MBit SPI Flash Memory	1	1.52	Purchased
	LDO Voltage Regulator	1	2.27	Purchased
Passive Components	DIP Toggle Switch	1	1.56	Purchased
	Blue Indicator LED	20	6	Purchased
	Other Passive Components	50	15	Purchased
Fabrication and Assembly	Acrylic	1	10	Purchased
	PCB Fabrication	5	150	Purchased
	Solder Materials	1	20	Purchased
TOTAL			417.91	

The above budget has remained largely the same since the interim report and update at that time. Two major changes occurred during the assembly and fabrication process. First, all assembly was done by hand, eliminating the need for a solder paste stencil. Second, there were two PCB's needed instead of 1. These costs are reflected in the budget above.

Project Planning Challenges and Solutions

1. The Human Component of Development

   Problem

   The initial timeline given for the project assumed that there would be constant work on the project. There was no assumed downtime for the developer, simply downtime during

fabrication or testing. This proved to be very ineffective and made it very easy to burn out early in the project with the large workload.

Solution

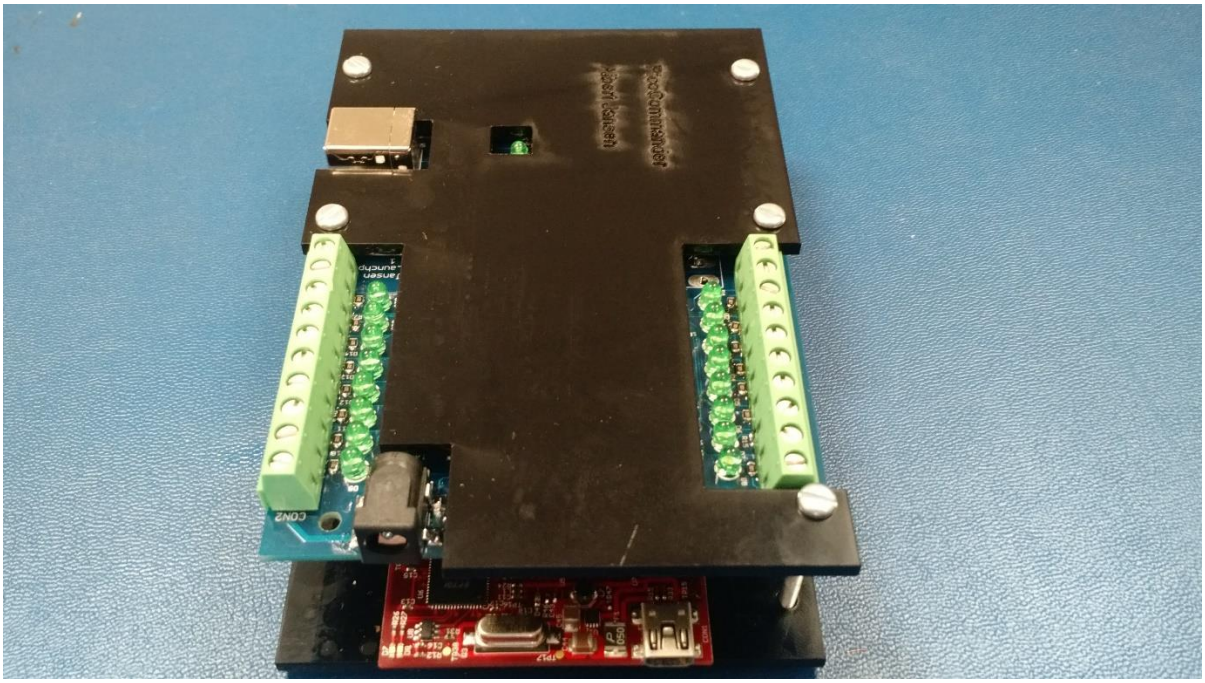
The solution came by allowing some of the extra time in the project to be used for a lighter workload. The development timeline became slightly more realistic for what one person could handle, however, this reduced the amount of buffer time.

2. Underestimation of Development Costs

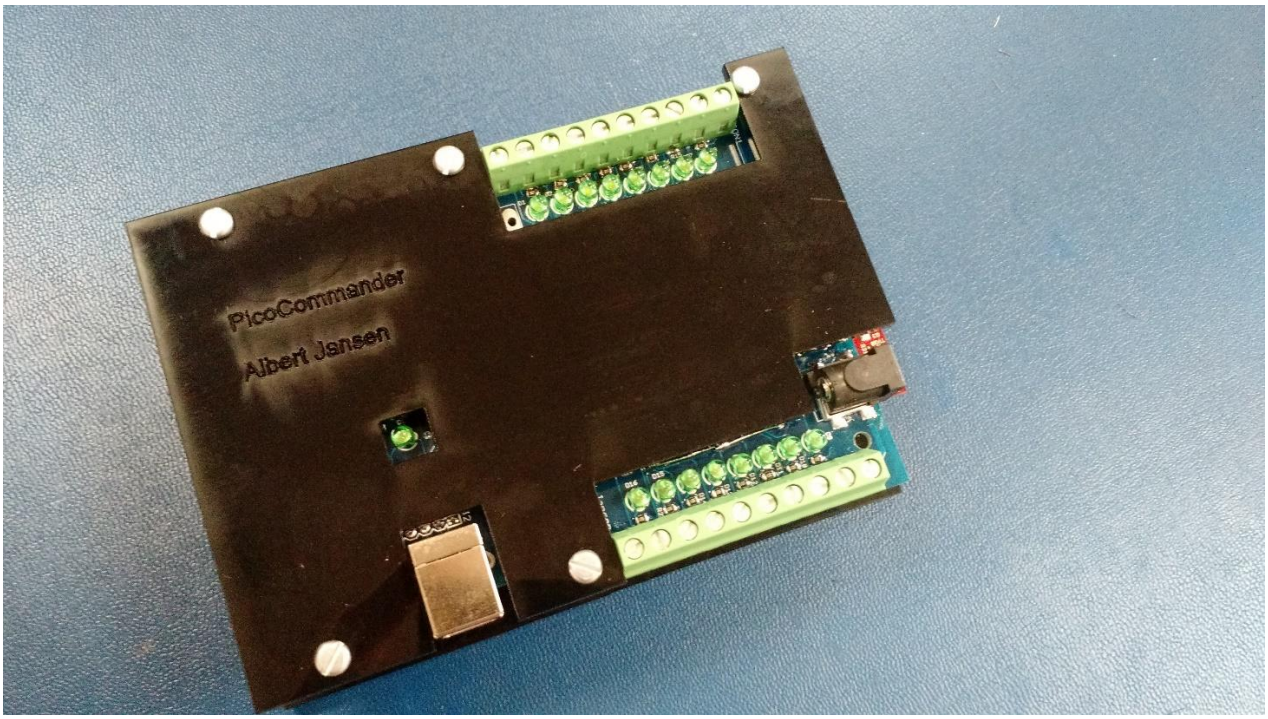
   Problem

   There was less time and money than was initially anticipated at the start of the project, and more was needed than was initially anticipated. This was mitigated by making compromises throughout the project on certain features that would have improved the overall user experience, but did not provide the base functionality needed by the Pico Commander.

Final Product



The above photo demonstrates the two different methods of providing power to the Pico Commander. The first is a DC jack on the module, and the second is a micro USB port.



The above photo shows the top view of the Pico Commander. There is a USB port and LED to indicate that the USB is connected. There are also screw terminals and LED's to indicate program functionality.

Future Development

Proposed Future Technological Development

1. Expansion of User Libraries

   The libraries of functions that can be used by the Pico Commander are currently quite limited. To expand its application, it would be useful to continually expand the variety and complexity of functions available for Drag and Drop program creation.
2. Re-configuration of input and output buffers

   Depending on the future purpose of the Pico Commander, the inputs and outputs will be reduced in number in favor of analog inputs and outputs. The current configuration of 8

inputs and 8 outputs is slightly excessive, and it would be beneficial to add 2 analog I/O, while reducing the digital side to 4 inputs and 4 outputs to conserve on materials and costs.

Additionally, a variable buffer voltage will be integrated. Currently the user only has the option of interfacing with 5V systems. Increasing the voltage range and making the inputs and outputs user selectable will increase the broad applications of the Pico Commander module.

3. Addition of wireless technology (Wi-Fi or Bluetooth)

   The addition of wireless would be used to perform any functions that a USB connection could perform, but without the wired connection. Part selection needs to be done to reduce firmware overhead, as the full USB stack and a wireless stack, on top of the Pico Commander firmware take up a significant portion of the on board microcontroller memory.

4. Monitoring and Simulation

   These two features are closely related. They both involve the ability to update the display for the user. Allowing the user to monitor and simulate their programs will give them greater control over testing and program operation.

Recommendations for Future Development

1. Prototype before final build

   While developing the final version of Pico Commander it is important to create modular systems that can be easily tested in isolation and within the system. The costs of hardware prototyping are large, and mistakes can be costly if final versions are consistently incorrect.

2. Create software specifications before further programming

   To increase the ease of use of the Pico Commander, software standards must be identified. This will allow users to add their own libraries and expand the Pico Commander system for their own needs, while sharing their changes with the community.

Conclusion

This final report serves two main purposes. The first is to explain the technologies associated with each area of the Pico Commander. The second is to provide a post-project analysis of both technological and planning challenges that were encountered, as well as their solutions. Looking forward, proposals for future development are given alongside recommendations for future development and planning, to ensure that the experience gained during the project is put to good use. The Pico Commander project has been a great success, and the experience gained throughout the several months spent on the project will prove invaluable going forward.

User Guide

Introduction

The following user guide captures the core features of the Pico Commander, and provides an overview of how to develop for the system. It also provides key safety features, and hardware interfacing suggestions. The user guide is split into four sections.

1. Hardware Interfacing and Setup
2. Software Development
3. Warnings and Ratings
4. Contribution and File Standards

Each section provides the information needed to begin development using the Pico Commander system. Section 4 provides the material needed to start developing the Pico Commander to suit your needs.

Hardware Interfacing and Setup

Inputs

There are 8 digital inputs available to the user, and they are marked as such on the packaging of the Pico Commander module. VCC (5V) and Ground are also marked, and sit at either end of the input connector.

To connect a signal to a digital input, simply insert a wire into the digital output of your choice, and secure the wire using by tightening the screw terminal block. This connection will provide the basis for all interfacing, and makes it easy to connect many different circuits to the Pico Commander.

Note that any circuit external to the Pico Commander must share a common ground to function correctly. Always connect the input ground the ground of your circuit.

Outputs

The outputs of the Pico Commander function in the reverse way of the inputs. To connect an output to the Pico Commander, simply follow the same process for the above circuit. Insert a wire into the desired output, and secure the wire by tightening the screw terminal.

Note that any circuit external to the Pico Commander must share a common ground to function correctly. Always connect the output ground the ground of your circuit.

USB

The USB connection is a female USB type A connector. The USB connection can be completed by connecting the USB cable to your PC with the Pico Commander software installed.

Note that this is not a USB powered device. Power must be supplied via the DC Jack.

DC Jack

The DC Jack on the Pico Commander accepts 5V. The wall wart or other power supply must be able to supply 1A of current. It is recommended that a wall wart is selected that can provide up to

2A of current. Simply plug in the power from the DC Jack and the program that you created will start running.

Software Development

The software development stages are split in to several parts. First, a general and quick start guide are given on how to create, compile, and upload a program. After this quick start guide, there is a more in-depth discussion of the different components of the graphical user interface, as well as their function and purpose.

Program Creation

To create and upload a program, open the Pico Commander GUI, and follow the steps below.

1. From the file menu, select "New >Empty File", (Ctrl+Shift+N). When the file window pops up, select "Drag and Drop". Click ok.
2. In the window that appears, add a tile by going to "Editor > Add Tile". The tile can be moved around the workspace. Position it where desired.
3. Once the tile is positioned, import a library by clicking on the "Import Library" button in the resource manager, and selecting a .lib file.
4. Double click on the tile you positioned earlier, and assign it a function from the drop down menu.
5. Repeat the process in steps 2 and 4 for a second tile. At this point, there should be two tiles, each with a function assigned to them.
6. To begin a connection between two tiles, right click on a tile. That tile will now have its output connected to one of the inputs to another tile.
7. To complete the connection, right click on the tile to which the output will be passed. Select the input that you would like to connect to in the pop-up message.
8. Repeat the tile adding and connecting process until the desired function is achieved.
9. Save the file through the file menu ("File > Save"). Once the file is saved, select the compile function. If everything processes correctly, there will be no error message.

10. Connect the Pico Commander module, and select the "Detect and Connect" button to connect to the Pico Commander. A message will appear if the connection is successful.
11. Finally, click the upload button, and browse to the .upl file generated by the compile process. Select the file.
12. If the upload completes successfully, the program will begin running immediately.

Menu and Toolbar

The menu and toolbar are found at the top of the application. From here, you are able to perform any of the functions available in the Pico Commander system, such as: opening and editing files, building programs, and uploading and connecting to the Pico Commander.

Workspace Manager

The workspace manager is a collection of buttons found above the file browser on the left side of the application. There are two main functions as of this release. The first, "Change Workspace Path" allows the user to change the folder that they are working in. The second, "Import a Library", will let the user import a library for use in their program creation.

File Browser

The file browser allows users to view and open files by browsing through the file structure of their workspace.

Library Browser

The library browser allows the user to see their available functions and the library they are associated with.

Workspace

The workspace is where users can create programs via a drag and drop creation method. The workspace can contain two types of objects: tiles and arrows. Tiles are blocks that calculate or manipulate a single output, based on one or more inputs. Their functions are assigned after they are added to the workspace.

Arrows denote the passing of a variable from the output of one tile, to one of the inputs of another tile. As stated above, each tile only has a single output, however that output can be connected to many different inputs if necessary.

Further reading

For further documentation, and access to the source code and hardware designs, refer to the following link: eelbot.github.io/PicoCommander

Warnings and Ratings

Safety

The Pico Commander module is capable of emitting higher than average levels of current for electronics. While this level of current and voltage are not at levels that are harmful, it is still necessary to practice safe circuit techniques when connecting the Pico Commander to a system.

1. Read and observe the maximum ratings when building a circuit. Do not exceed these ratings as doing so can lead to fire, excessive current, and other catastrophic failure of the Pico Commander module.
2. Do not work on circuits that contain voltages greater than 50V unless licensed to do so. Always consult an expert before creating high voltage circuits and connections.
3. Verify operation of the program before uploading, and ensure that safety standards for your application are met.

Ratings

Value	Recommended Rating	Maximum Rating
Current per input	80mA source, 180mA sink	100mA source, 376mA sink
Total input current	500mA	600mA
Input Voltage	0V – 5.5V	0V – 5V
Current per output	24mA source, 24mA sink	50mA source, 50mA sink
Total output current	150mA	200mA
Output Voltage	0V – 5V	0V – 5V

There is potential for electric shock when working with the Pico Commander. Observe the above recommendations to reduce risk of damage to equipment and lives.

Appendix A - Sources

- Data Sheet, Texas Instruments, "TMS320F2806X Piccolo Microcontrollers," TMS320F28069 datasheet, Nov. 2010 [Revised July 2014].
- Data Sheet, Texas Instruments, "25-Ω Octal Buffer/Driver With 3-State Outputs" SN64BCT25244 Datasheet, Dec. 1992 [Revised January 1994].
- Technical Reference, Texas Instruments, "TMS320x2806x Piccolo Technical Reference Manual," TMS320F28069 Reference Manual, Jan. 2011 [Revised March 2014].
- Reference Design, Texas Instruments, "Factory Automation and Control System Solutions", 2012. Available from: http://www.ti.com/lsds/ti/apps/automation/overview.page
- Reference and Technical Solutions, Maxim Integrated, "Programmable Logic Controllers", 2013. Available from: https://www.maximintegrated.com/en/solutions/industrial/control-and-automation/plc.html/tb_tab1
- Pit Documentation, Python Wiki, "About PyQt", June 2015. Available from: https://wiki.python.org/moin/PyQt
- PySerial Documentation, Chris Liechti, "Welcome to pySerial's documentation", 2015. Available from: https://pythonhosted.org/pyserial/
- Beyond Logic, Craig Peacock, "USB in a Nutshell", September 2010. Available from: http://www.beyondlogic.org/usbnutshell/usb1.shtml
- Taylor Singletary, "Writing Great Documentation", January 15, 2016. Available from: https://medium.com/life-tips/writing-great-documentation-44d90367115a#.sz21wcryi
- PyUSB Documentation, walac, "PyUSB - Easy USB access on Python", 2016. Available from: http://walac.github.io/pyusb/
- PyQt4 Documentation, "PyQt Class Reference", 2016. Available from: http://pyqt.sourceforge.net/Docs/PyQt4/classes.html