THE UNIVERSITY OF ROCHESTER

UREAD BRAILLE

2014 CORNELL CUP

Team Members:

Christina Kayastha, Samantha Piccone, Doug Miller, Nicholas Gekakis, Tianyi Liu, Joel Howard

Advisors:

Prof. Randal Nelson, Prof. Ted Pawlicki, Prof. Chris Brown

CONTENTS

1. ABSTRACT

Tablets and e-readers have become a staple of modern life in the Information Age. The modern world has grown accustomed to on-demand access to information that is always within arm's reach. For the blind, accessing said information involves the inefficient and often frustrating use of screen readers and the inconvenience of being tied to a computer. While there do exist some options that can dynamically display output, for example the BrailleNote[1], these bulky devices are limited to one line of braille and cost upwards of $4,000, and thus do not truly parallel the touchscreens that have become the status quo for the rest of the population. The difficulty of creating such a device lies in efficiently producing a large number of braille sized pins that are capable of quickly moving up and down and staying in place. We propose a solution that would allow the production of a large array of pins at standard braille sizing using solenoid actuators created with a printed circuit board and an elegant magnetic bistable pin design.

2. CHALLENGE DEFINITION

In 1960, 50% of legally blind school-age children were able to read braille (Estimated Number of Adult Braille Readers in the United States). But, in 2007, only 10% used braille as their primary reading language (Facts and Figures on Americans with Vision Loss, 2008). The use of braille is on the decline, and it is easy to see why. Braille books are incredibly large and expensive. For example, a braille edition of the Harry Potter series consists of 56 volumes, and a single text book can cost $1000. A small book, less than half an inch thick, when translated into braille, results in an 11 by 11.5 inch book (the standard size of braille book paper) that is around 2 inches thick. Students are now turning to MP3 players, audiobooks, and computers with screen readers as a more affordable, accessible, and convenient option, disregarding the need to learn braille. Braille has become regarded by some as arcane and obsolete. A stigma surrounding braille has developed, with the idea that using braille somehow makes a person "more blind." So, in the modern era, why does braille really matter?

Audio learning is certainly not considered sufficient for sighted children, and neither should it be for blind children. A child that only listens to learn is not taught spelling, punctuation, or syntax, and cannot fully engage with the material being taught. Additionally, braille can communicate information that audio cannot, such as math and science notations. The impact of braille on the blind community is evident in the employment statistics. In the United States, only one third of blind adults are employed. Of those that are employed, 93% read and write braille. Braille is essential for the blind to be afforded the same opportunities as the sighted population.

[1] The BrailleNote is a refreshable braille display by HumanWare.

CHALLENGE REQUIREMENTS

In order for a device to solve these problems, it would need to fulfill the following requirements:

PRIMARY REQUIREMENTS

To make braille convenient and accessible, the device must be able to replace printed braille books and be affordable while having good portability, robustness, and usability.

BOOK REPLACEMENT

For a device to replace braille books, it must be able to express information in a similar manner with at least the same level of efficiency.

AFFORDABLE

With only one third of blind adults employed, it is essential for the cost of the device to be kept as low as possible. An expensive device will be inaccessible to many of the people that would most benefit from said device.

The BrailleNote, which only displays one line of braille, costs $3895. Based on that cost and the cost of a typical textbook, around $1000, it is reasonable to say that an affordable option should be below $4000, and ideally below $2000. Though high, even a $4000 device would be an affordable option in the long run because it would drastically reduce or potentially eliminate the need for expensive braille books.

PORTABLE

Though a stationary device would still be useful and an improvement to current options, in order to parallel the options available to the sighted world and allow the on-demand access to information described above, the device would have to be reasonably portable.

Portability will be judged based on size and weight in comparison with common portable technologies. A portable device will be comparable to a full-size laptop, with an ideal being comparable to large tablet.

ROBUST

Since the device will be beneficial for both children and the elderly, who may be more prone to dropping it, extra attention should be spent on finding a robust solution. Since the device will not be cheap enough to be easily replaceable by the average person, it is extremely important that it does not break easily and that is can withstand day-to-day use.

Robustness will be judged in comparison to modern laptops.

USEABLE

The device, both in its physical design/layout and braille user interface, must be easy to understand and convenient to use.

Usability will be assessed along the way using typical human computer interaction methods of assessment as described in the performance measures.

FUNCTIONALITY REQUIREMENTS

For a braille tablet to satisfy these primary requirements, it must satisfy the following:

ARRAY OF PINS

The device must have an array of braille pins, in the standard size and shape, and with the standard spacing. It must display a full page of braille similar to a printed page. A full page is essential for portraying format and syntax along with math and science notations, and simply creating a smooth reading experience. The most adept braille readers are even able to use two fingers at once along different parts of the page to read more efficiently.

A typical braille page has around 40 columns and 25 lines of cells (each cell has 2 columns and 3 rows of pins). Since the amount of cells makes the device larger, heavier and more expensive, this competes directly with the portability and affordability of the device. Because of this, a full screen will be defined as anything equal to or larger than 20 columns and 10 lines of braille, with an ideal around 30 columns and 25 lines.

Because books can use simple graphics to display information, an ideal solution would be a full grid of pins. A full grid is defined as a page of evenly spaced pins, such that when every third column and every fourth row of pins is off, the pins form normal braille cells. A full grid opens up the possibility for other exciting usages such as games.

INPUT BUTTONS

For a simple e-reader, the device would need buttons that would allow navigation and selection. Dynamic buttons that simulate web-browser functionality is outside of the scope of the proposed project but beneficial to keep in mind when assessing different technology options.

DATA INPUT

The device must be able to take in data from some external device, in some text form, to display. This would most likely be from a USB storage device, and ideally also from the internet.

INTERNAL STORAGE

Though not essential for use, for convenience the device would ideally be able to store documents along with information to improve usability.

PRACTICAL ENERGY CONSUMPTION

The device must consume a reasonable amount of energy comparative to current technology, and not generate too much heat.

3 PROJECT ENTRY SOLUTION

To solve this challenge, we present the BrailleShape.

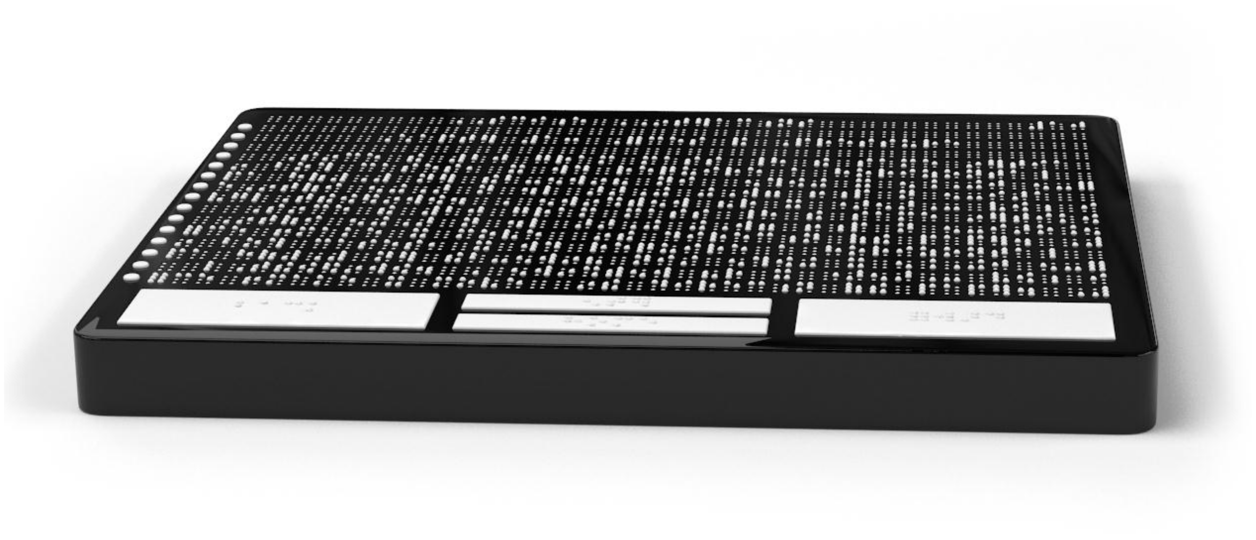


Figure 1 - BrailleShape Mockup

The BrailleShape is a full-page, refreshable braille display currently in development. Because the cost to prototype the actuators was beyond our budget, we focused on separately developing the components of the device without the full array of actuators, and hand making a functional single braille cell with actuators to prove our concept.

FUNCTIONALITY

PLANNED FUNCTIONALITY

The device will be able to quickly refresh a full page of braille to show documents, allow the user to take notes, and possibly allow the user to play games.

The devices will have two modes, reading and writing, and three main pages, the library, the document menu, and the document itself. When in the library, the user will be able to use the top navigation buttons to import a document from a USB drive, resume the last page they were viewing, and view their categories, bookmarks, and notes. Selection buttons corresponding to each line will allow the user to easily select which document they would like to view. When reading a document, it will be just like reading a book, with directional buttons to flip the page back and forth, and main navigation buttons to quickly go back to the table of contents or bookmarks. This is explained in more detail below. This is explained in more detail under

CURRENT FUNCTIONALITY

Current functionality is limited to the components as described below. Software and related functionality is implemented without the actuator array. The actuator array is reduced down to a single character for prototyping. The bistable pins are demonstrated both the one character prototype and a non-actuated demo.

KEY COMPONENTS

SOFTWARE

Software that converts text or formatted XML into a braille format to be used by the device.

The software is able to take plain text or an XML document and convert it into a format that the device can use and display while translating the text to braille, which can include special standard braille characters that are used to condense common words. The braille format uses pages and line numbers to allow navigation with table of contents and bookmarking.

MICROPROCESSOR

For each pin, the microprocessor determines whether the pin needs to be moved and sends the FPGA the address of the pin.

FPGA

Obtains next pin address from the microprocessor and activates the actuator.

ADDRESSING CIRCUIT

Decodes the address sent by the FPGA and depending on whether the pin has to be raised or lowered activate the correct actuator.

Activates the correct Actuator Driver Circuit.

ACTUATOR DRIVER CIRCUIT

When the addressing circuit matches the correct actuator, allow high current to pass through the solenoid.

Provides current to the driver circuit.

ACTUATOR ARRAY

A physical grid of solenoid actuators that uses passive matrix addressing. Two grids are used, stacked on top of each other, to reliably control the pin. Activating the top solenoid will move the pin up, and activating the bottom will move it down.

Using a multi-layer printed circuit board, an array of solenoids can be created efficiently. Though the cost to prototype such a board is great (over $10,000), the cost of each goes down dramatically with bulk ordering, making them affordable (approximately $500) when ordering quantities of 20 or more.

We initially abandoned the idea of a full page array of actuators. After receiving a quote of about $500,000 for a 30 layer board, we gave up on the PCB and looked into other ways of actuating the pins. Though we came up with a few designs, none were as elegant as the full PCB, and had problems with bulkiness and refresh speed. After experimenting again with solenoids, we realized that we could decrease the layers needed to bring down the price into a more reasonable range. Because of our new bistable pin design and how seamlessly it worked with the solenoids, we decided to refocus on the PCB and were able to design so that it would be affordable. Regrettably, even the affordable option was out of our budget for prototyping, so we created one cell using hand-made solenoids to the same specifications they would be made using the PCB.

BISTABLE PINS

An array of bistable pins that move up or down when the corresponding actuator is activated.

The bistable mechanism allows a pin to stay up once its solenoid is pulsed, meaning the power requirements are low and only needed when the page is being refreshed. Also, the drastically decreases and chance of the device overheating.

This designed is changed from our original proposal. We were initially worried about the pins being easy to push down when using magnets, and our previous designs were overcomplicated and used complex parts that would need to be custom made and require difficult, small scale machining that would be difficult to align. The new design is simple; no expensive custom parts are necessary, decreasing the cost of the device. The design also has great stability, ensuring that the pins will stay in position even with significant pressure applied. The entire length of the pins are used to their full advantage by occupying the entire actuator space, keeping the device thin and portable.

BUTTONS

Buttons for navigation and selection that check for feedback and send information back to the FPGA. We added main navigation buttons to the design of our device to make it more useable. The full grid device also has line section buttons for both the standard 6 pin braille, and 8 pin braille which is used to show extra formatting.

4 PERFORMANCE EVALUATION

PERFORMANCE MEASURES

CHALLENGE REQUIREMENTS (TABLE 1)

	Measure	Goal	Limit	Method
Affordability	Price	$2000	$4000	Total of parts, plus manufacturing estimates.
Resolution	Columns Rows	40 25	20 15	Simple count. Negatively affects portability and price.
Portability	Length Width Height Weight	10 in 8 in .75 in 4 lbs	14 in 10 in 1.5 in 7 lbs	Simple measurement. Comparable to a laptop or tablet.
Robustness	Strength Stability	The device should not break if dropped. The components should stay in place if the device is shaken.		
Usability	Learnability Efficiency Memorability Errors Satisfaction	The goal is to maximize each measure of usability to ensure the average user can pick up and easily and enjoyably use the device.		Usability testing will be done with common HCI techniques. User testing will be done with primarily paper prototypes before constructing final device due. Interface testing will also be done with the final device.

SUBSYSTEMS AND COMPONENTS (TABLE 2)

	Measure	Performance
Pin Strength	Flexural Strength	The pins do not break under reasonable conditions. Measured with MTS Criterion™ Universal Test Systems.
Pin Stability	Force	The pins stay up and down. The pin must be able to withstand the force a person can exert with his or her finger. The pin must also not fall into the on position when the device is flipped.
Pin Control	Error Count	The movement of the pins can be controlled accurately and reliably. The pins always move when sent the signal to move. The pins are always in the intended positions. The pins are up when signaled to be on, and down when signaled to be off.
Refresh Speed	Seconds	The page refreshes within a reasonable amount of time.
Energy Consumption	Power	The power consumption is reasonable compared to current technology. Measured with a kilowatt meter.
Heat	Temperature	The device does not get too hot to touch comfortably. Measured with an infrared thermometer.

PERFORMANCE

Because we were unable to create a fully functional device, we assess the performance of the components and estimate the impact this will have on the performance of the future device.

CHALLENGE REQUIREMENTS

AFFORDABILITY

Estimated cost: $2000 - $3000.

The primary expense is the PCB, which will cost approximately $500 in total. This cost is only modestly dependent of the size of the board; smaller boards will be cheaper but not significantly so.

The other components depend on how many pins are used for the device. Using a simplified pin design, the cost of all mechanical parts is kept low, and is limited to drilling and the pins themselves. The cost of each pin is approximately 14 cents. For a reasonably sized board (30 by 15 cell) using 6-pin braille cells, the cost for pins will be $400 total. For a full grid board, it is $735.

Different devices could easily be made for different budgets. While a large page of a full grid of pins has more possibilities for display than using braille spacing, the number of pins is almost doubled. Smaller devices may be able to be reduced to $1000 or less.

RESOLUTION

The resolution of the device is only constrained by the price – the currently technology can handle as many pins as is desired.

The ideal board will have a full grid of pins that can function as 32 by 15 cells with 6 pins, or 32 by 12 cells with 8 pins. This level of resolution allows not only simple text to be read, but additional formatting and even graphics.

PORTABILITY

The new bistable pin design gives the device great portability by allowing it to be thinner than current technologies. The current board design is approximately .5 inches thick. While a steel sheet used for the bistability does increase the weight of the device, the sheet is thin and it is still reasonable. The board can also be made with minimal margin space around the array of pins, keeping the size of the device as small as possible.

ROBUSTNESS

There are no fragile parts, and the board is primarily a steel sheet between layers of polycarbonate. As such, the device will be very strong.

The pins are also very stable, stay in place when the device is shaken. Even with great pressure applied to the pins, while they can be pushed down, they stay in the proper location, bouncing back up once pressure is released. The only way to push a pin down is to use a pin to push it all the way through the hole.

USABILITY

Usability was unable to be assessed due to not being able to create the device and focusing our energy on the technology itself. The only usability aspect that comes into play with the pin technology is the size of the pins. Braille users are accustomed to a certain size and shape of braille, and we worked hard to ensure our technology is capable of achieving these size constraints.

Other aspect of usability are dependent on the button layout and the software. Though these are important, they can be easily modified and changed. Usability testing will be done before a complete prototype board is produced.

SUBSYSTEMS AND COMPONENTS

PIN STRENGTH

Since the pins are so small, 1/16th in in diameter, the strength is critical. Since the pins need only be ¼ inch long and are 1/16th of an inch along the entire length, they are very strong.

PIN STABILITY

The pins are highly stable as described above in Robustness. They will not be able to be pushed down under any reasonable circumstances. In order to test this we pushed the pins flat down on the surface of the sample bistable mechanism. Due to the mechanism they rebounded back into place.

PIN CONTROL

They pins are highly controllable. By using two solenoids, we can set a pin up or down independent of its current position. The control is also highly reliable and will only fail if the solenoid fails to fire.

5 TECHNICAL DOCUMENTATION

The primary purpose of the BrailleShape can be summarized as the translation of basic text to a tactile braille output. In order to fulfill the requirements of this translation, the process was broken down into several subsections. Please see Key Components for more details. A summary of the process is listed below:

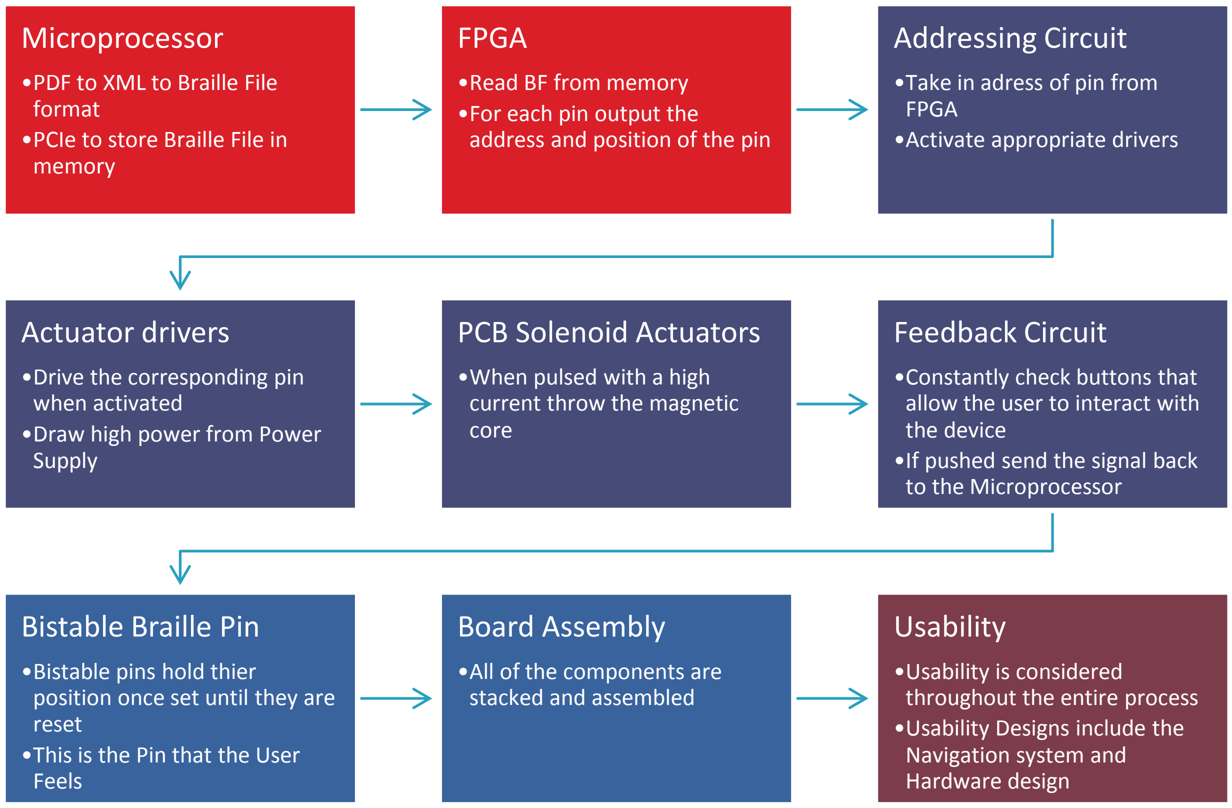

Microprocessor
•PDF to XML to Braille File format
•PCIe to store Braille File in memory
FPGA
•Read BF from memory
•For each pin output the address and position of the pin
Addressing Circuit
•Take in adress of pin from FPGA
•Activate appropriate drivers
Actuator drivers
•Drive the corresponding pin when activated
•Draw high power from Power Supply
PCB Solenoid Actuators
•When pulsed with a high current throw the magnetic core
Feedback Circuit
•Constantly check buttons that allow the user to interact with the device
•If pushed send the signal back to the Microprocessor
Bistable Braille Pin
•Bistable pins hold thier position once set until they are reset
•This is the Pin that the User Feels
Board Assembly
•All of the components are stacked and assembled
Usability
•Usability is considered throughout the entire process
•Usability Designs include the Navigation system and Hardware design

Figure 2 - Process

This section will examine each one of these components and explain in full the algorithms and processes used in order to complete their implementation. The different components have been grouped into four categories: Software, Electronics, Hardware and Usability. For each category, the subsections will be listed and defined. Each component will have a general overview, implementation procedure, design justification, and alternate suggestions sections.

The goal of this spread is to fully cover all technical aspects regarding the BrailleShape. Please note that there are supplemental details in the Appendix regarding the Test Plans, Assembly and Construction, as well as Software.

TECHNICAL OVERVIEW

Before explaining each section in detail, we will explain in brief the process flow from beginning to end of the BrailleShape. This basic overview does not list all of the specifics of the project but only serves to give a short background before delving into the technical areas. Following this section a more rigorous explanation of each component is given as well as the technical justification, and assembly process where needed.

PROCESS THROUGH EXAMPLE

The best way to describe the process would be to follow it by an example. Suppose we want to translate a PDF book to braille, this is what happens behind the scenes.

First, say the user plugs in a USB drive and presses import choosing a PDF book to import. On the software side, the BrailleShape will take the pdf file and translate it to an XML document containing only the relevant formatting information. This XML document is stored on the device for future use. After the XML document is created, a second step in the software parses the XML and outputs a Braille File format specific to the device. This file format is then stored in memory via PCIe connection.

Next, the FPGA reads the Braille File and for each pin listed in the file calculates the corresponding row and column of the pin to be activated, we will call this the address of the pin. The address of the pin is output to the off-board electronics.

The addressing circuit receives the address of the pin to be activated and decodes it. It then activates the drivers relating to the pin and its position. The actuator drivers pulse a high current through the solenoid coils which sets the magnetic core in either the top or bottom position. Due to the power necessity, the pins are activated one at a time. At the hardware category, the magnetic pin is bistable so that when it is placed in the top position it stays up and when it is set down it stays down.

After the BrailleShape is done setting the page it continues listening to commands from the feedback circuit to determine the next page to set. For example if the user then clicks the next page button, the device sends that signal to the microprocessor which will then output the braille file corresponding to the next page and the process will repeat. The design of this feedback system constitutes of the Usability category.

The next sections outline in detail the functionality and process of the different parts of the device grouping them by category.

5.1 SOFTWARE

The Software constitutes of the Microprocessor and the FPGA. Both of these combined are responsible for the total control of the BrailleShape. The software process begins with the raw PDF formatted file and ends with the FPGA’s output to the off-board circuitry telling it to set the pins for an entire page pin-by-pin.

We start the discussion with the Microprocessor then define the communication lines between the Microprocessor and the FPGA and finally discuss the role of the FPGA.

5.1.1 MICROPROCESSOR

The microprocessor is responsible for all the primary computational work of the system. The Atom processor on the Altera board was used. There are several parts of software that live in the microprocessor these are:

1. Navigational system
    a. Feedback buttons check
    b. PDF to XML translator
    c. XML to Braille File format translator
    d. Storage to memory through PCIe

The main piece of code is the Navigational system. This dictates what the next output should be and calls on all of the other three functions of the microprocessor.

NAVIGATIONAL SYSTEM OVERVIEW

In this section we outline the navigational system of the device as well as all of the software functionalities we have designed.

The device will have three main areas: the library, the document menu, and the document itself. These where chosen based on typical existing e-book designs.

In order to select links, the user simply presses the side button aligned with that line of braille.

Confirmation boxes can pop up using the bottom three lines, the top telling the user what they are confirming, the next for yes, and the last for no.

LIBRARY

The library is where all of the documents are displayed, each listed using on line. When in the library, the back button will function as the import button that will bring the user to the USB interface. The resume button will being the user back to the last page they left off on. The categories will allow the user to choose different categories to sort the books by. Bookmarks will display the most recent bookmarks. If the user wants to delete a document, the can hold down the side button on the line that document is listed on, and then press the delete button.

DOCUMENT MENU

The document menu will contain information about the book. When the user clicks the back button, they will be sent back to the library. Resume will send them to the most recent page they viewed in the current document. The categories button will display the table of contents, which will be links to the appropriate page. The bookmarks and notes buttons will show a list of bookmarks and notes, respectively, and link to those lines.

PAGE

While viewing a page, the back button will take the user back to the document menu. The categories button will be a shortcut to the table of contents. The user will be able to add a bookmark or a note to a page by clicking the button, or to specific lines by holding down the corresponding selection button then pressing that button.

IMPLEMENTATION

As explained above, the navigational system has 4 main functions. In this section we explore the implementation of the Navigational system. The following sections explain the implementation of each of the three sections pertaining to the implementation of the Navigational system.

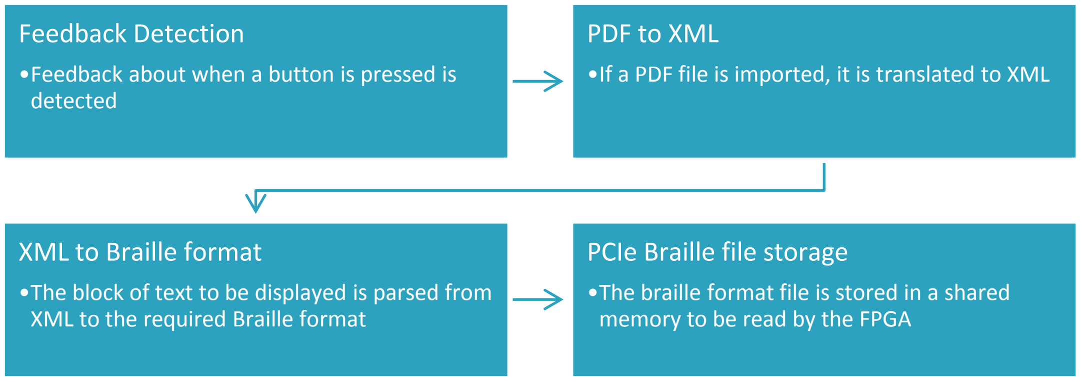

Feedback Detection
•Feedback about when a button is pressed is detected
PDF to XML
•If a PDF file is imported, it is translated to XML
XML to Braille format
•The block of text to be displayed is parsed from XML to the required Braille format
PCIe Braille file storage
•The braille format file is stored in a shared memory to be read by the FPGA

Figure 3 - the separate parts of the program and the process in which they are connected

Before explaining the algorithm of the navigational system, we summarize the major data structure of our code.

DATA STRUCTURES

The main way in which the files are stored on the device are through XML Pages. One book corresponds to a single XML document, an XML 'page.' The Main menu screens are also stored in XML Pages. The pseudo code for the Page object is listed below:

XML PAGE DATA STRUCTURE

```
Page(XMLfile, currentPosition, type):
        Xml = xmlfile
        currentPosition = currentPosition
        type=type
        if (type == main){
                menu.1 = new Page (MainMenu.xml, 0, main)
                menu.2 = LibraryPage
                menu.7 = SettingsPage
                …
        }
        Else if(type == book){
                menu.1 = Library
                menu.2 = Contents
                …
        }
```

Each Page contains a pointer to the XML file containing the xml text to be translated. The currentPosition refers to the current place that is being displayed. For example in a book the currentPosition might be some page number that was last read. This allows the program to calculate the next page or previous page.

The Page also contains information on what type of page it is. This is important because the menu bar for a book Page and the Library page are different. Given this information if the E-reader is currently on a certain page and a menu button is pressed it will know what page to redirect to (going from reading a book to the library) or if the position simply has to be changed (flipping a page).

NAVIGATION SYSTEM PSEUDOCODE

Given a certain page, the Navigation system is simply a way of tying them together. It will either detect feedback, ask for a pdf to be converted to an xml, or ask for a new xml page to be displayed. This can be summarized in the pseudocode.

PSEUDOCODE

```
Navigation System{
        currentPage=new Page (MainMenu.xml, 0, main)
        XMLtoBraille(currentPage, 0)
        while(on){
                checkButton= CheckButton();
                if (checkButton.pressed==True){
                        if(import){
                                PDFtoXML(pdf)
                        }
                        XMLtoBraille(currentPage, buttonPressed)}
}
```

First on initialization the currentPage, the XML page currently being displayed, is set to the beginning of the menu screen. This is then output to the device. Then for the duration of when the device is on, the navigation system checks for button presses. If a button has been pressed then it changes the display and outputs it. If the command is to import some new pdf then the PDF to XML converter is called.

The next sections outline each of the sub-functions called.

PDF TO XML TRANSLATOR

In order to translate from PDF to XML we used existing open source libraries. In our current implementation we are using the PDFMinder.py translator with the amagenerate.py open source translator.

First, Pdfminer.py takes an input PDF and outputs unformatted XML, amagenerate.py then formats the XML into paragraphs and pages. We edited the amagenerate.py because it was storing information that was not relevant to our translation such as subscripts and footers that we did not want to translate. In the long run we would write our own software to replace amagenerate because it is prone to errors and can be improved.

Please see the Software Appendix for more information about the libraries we are using as well as sample output code at each stage of the process.

The XML output from this proves stores formatting initially contained in the PDF text. The formatting that we are currently using is listed in the XML tags below:

Tag	Function: Justification
<b>	Bold: this is a standard in braille format
<i>	Italics: this is a standard in braille format
<u>	Underline: this is a standard in braille format
<h1>, <h2>, <h3>, <h4>	Header: four levels of headers are stored. These headers can be used in order to automatically generate table of contents pages for books or papers. This would allow the user to skim through the information without having to read it line by line which is one of the main goals of our full page display
<p>	Paragraph: this information allows us to format word wrapping properly and insures that the original paragraph blocks are translated accurately.
<page>, <tab>	Page, tab: the white space information allows us to format the braille appropriately. For example if the end of the chapter ends at half a page, we want to make sure that the next chapter starts on a new page.

Table 3

This XML file is then stored onto the local device for future use. For example if the user wants to read book #1 then once it has been translated from PDF to XML it does not have to be stored again.

XML TO BRAILLE FILE FORMAT TRANSLATOR + STORAGE TO MEMORY THROUGH PCIE

When the XMLtoBraille function is called, first the nextPage corresponding to the button press is called. Next the page is translated to BrailleOutput and that is printed to the screen. The pseudo code for this process is below:

PSEUDOCODE

```
nextPage XMLtoBraille(currentPage, buttonPressed):
        switch (buttonPressed){
                nextPage:       {nextPage = currentPage.currentPosition ++}
                prevPage:       { nextPage = currentPage.currentPosition --}
                menu#:                  {nextPage = currentPage.menu#}
                select#:        {nextPage = currentPage.currentPosition == select ID}
        }
        Output(nextPage)
Output(nextPage):
        BrailleOutput = TranslateXMLtobraille( nextPage)
        PCIe store =BrailleOutput
```

Once the XML file is generated, there is a stage that translates the XML to a braille file format. This braille file is then stored in memory. In order to complete the translation Grade 2 English braille is used as it is the most popular file format. For more information about the translation rules that were used please see the Software Appendix section.

The braille file is a binary file that stores the information about each pin on the current display. For example, in our current display there are 1920 pins, so the braille file contains 1920 bits referring to each pin. A 1 is considered a pin up state and a 0 is a pin down state. The XML to braille translator also deals with word wrapping and fills in whitespace in this braille format file. When the FPGA accesses this file, it does not need to change any of the pins on the display, but simply to read and set them according to this file.

The Braille file format is used only to communicate with the FPGA and is temporarily stored in a shared memory. The file is overwritten every time the page refreshes. Because we are translating from XML to braille text and the XML is already pre-generated, this process is not time consuming and can be done every time the page resets.

Currently in order to implement this, our Python code stores the braille file format to a txt file, and a C executable reads this and stores it to the FPGA. The C code acts as an interfacing piece connecting to the PCIe connection.

FEEDBACK BUTTONS CHECK

The C code checks the PCIe connection to see if any new button was pressed every 100 microseconds. If it receives information that the button was pressed then it relays this to the Python ButtonPressed() function as well as the information about what button is pressed.

In the main code, depending on what button was pressed and what the currentPage and currentPosition is, the next page is recalculated. For example, if in on the Library page one of the select buttons is pressed to open up a book, the XML file for the book is accessed and the currentPage is changed.

DESIGN JUSTIFICATION

There were several design choices that we made for our implementation of the navigational system.

USAGE OF XML AS THE PRIMARY BOOK FORMAT

One of the more important design decisions that we made was storing the generated file in the XML format. Initially our translator simply stored the text file from the PDF however we found that this meant that a lot of data about the formatting was lost. Information such as bold, italics, underlined formatting as well as header level could not be retried.

We decided to use XML to store this formatting information because XML is widely used for such formatting applications and is very easily adaptable. With XML we are able to easily parse through the code and add or remove formatting as necessary.

Furthermore, in the long term we want to be able to translate not just PDFs but also other forms of Rich text formatting and ebook formats such as the EPUB, KF8, AZW. The main long term goal is to create a browser-like experience for the user so that they can access webpages. XML is highly compatible with HTML and so using XML adds a level of simplicity when working with webpages. Please see the Next Steps section for more information on future steps.

USING PYTHON AS A PROGRAMMING LANGAUGE

Our current implementation uses a python translator to translate from PDF to XML. Initially we were planning on using a C library, XPDF, that does the same thing as PDFMiner. Both of these code bases output the same type of unformatted XML code. The reason that we decided to use PDFMiner is because we were able to find the amagenerate.py code that simplified the XML.

We are using Python as the main programming language for this device mainly because the code that takes the longest is the translation from PDF to XML. Since we chose a python library in order to do this we decided to continue using python for compatibility.

Since Altera provided PCIe libraries in C, it would be better for us to have used C all the way through. This would simplify crosstalk between programs. Currently we are only using C for the last step in the process, to access the PCIe connection.

BRAILLE FILE FORMAT

The braille file format was chosen to be a binary file containing a string of 0s and 1s. We used this format because this way the FPGA only has access to the most relevant information—what pins should it set when updating the page. In the long term, the BrailleShape would migrate away from the altera development board and use an onboard microprocessor. By separating all of the code that deals with calculating the next page from the code that governs the hardware will allow for a level of abstraction when implementing this without an FPGA.

ALTERNATIVE SUGGESTIONS:

Some places that we could improve our software are in the translation process from PDF to XML. Currently we are using open source software, amagenerate.py, to format the output of PDFMiner.py (unformatted XML) to formatted xml. This software had a lot of code that inaccurately recognized footers, and page numbers etc. The next step would be to write our own code that does this formatting storing only the information that we need.

If we write our own code using the XML output, we are no longer restricted to Python. It would be better to switch to a faster programming language such as C and use a C library for PDF to XML translation. We would also be able to interface our program with open source software for OCR (Optical Character Recognition) software in order to extract text from images etc. This would allow us to greatly expand the input formats that we allow.

5.1.2. COMMUNICATION BETWEEN THE MICROPROCESSOR AND THE FPGA

In the previous section it was explained that the PCIe connection will be used in order to interface the Microprocessor with the FPGA. In order to fully communicate all of the functions of the device, a system of handshaking logic was created.

OVERVIEW (TABLE 4)

The table below outlines the communication lines between the microprocessor and the FPGA.

Line Name	Category	Description	Direction
On	Communication	On turns on and stays on when the processor is turned on.	Processor→ FPGA
Ready	Communication	Ready goes high when the processor is ready to send the next signal and has set the next pins address in the address lines. Ready cannot turn on without want having turned on.	Processor→ FPGA
Done	Communication	Done is turned on by the FPGA when the pin is finished setting. See the next section for more details.	FPGA→ Processor
Want	Communication	Want can only turn on when 'On' is on. This like goes from the FPGA to the processor asking for the next address to be passed to it.	FPGA→ Processor
Braille File	Pin Positions	This is a stream of bits that is stored to memory specifying the pins that have to be set on the page as well as their position. This stream is sent via PCIe and stored in a common memory.	Processor→ FPGA
Feedback Set	Feedback Information	This will go high if the circuit detects any input in any of the buttons. Using this the code does not have to keep checking all of the Feedback addresses but only see if this goes high.	FPGA→Processor

Feedback 0-9	Feedback Address	These cells are the feedback address. If a button is pressed, then the FPGA sends a signal back to the processor specifying which button (for example the next page button) was pressed. Due to the circuitry, a feedback with 9 pins corresponds to 3x8=24 addresses. This is because of the 3:8 Decoders	FPGA→ Processor

IMPLEMENTATION

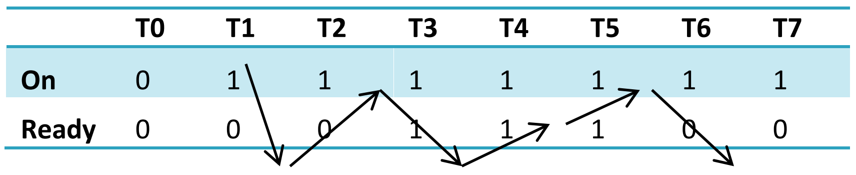
While the previous section lists out in general what the different lines of communication are, this section explores the communication lines in more detail and explains the algorithm that defines when certain actions are triggered by these communication lines.

The following describes each one of the communication lines in more detail:

1. On: This line is set by the processor when the processor is turned on and is ready to send signals to the FPGA. This line is an enabling line and is required for the FPGA to send signals to the solenoid actuators as well set any of the other communication lines.
2. Ready: The processor turns on ready when it has finished storing the new braille format file in memory. This signifies that the positions for all the pins are set. The processor turns on ready once it receives 'want' as high and has set the output file. Ready is deactivated when 'done' goes high.
3. Done: Done is a line that is set by the FPGA once it is finished setting an entire page. It represents that the currently stored file is done setting. When the processor receives 'Done' it triggers Ready to turn off. It then checks whether any of the Feedback lines are set, and if it is received recalculates the next set of Row and Col address lines to output when ready is turned on.
4. Want: Want represents the FPGA's state of wanting the next page. This turns on when the FPGA detects Done. Note that want and done are not opposites because when the FPGA is setting the page, Want and Done are both turned off. When Want is turned on, the processor prepares the next page address and prepares Ready.

FOR EXAMPLE:

The table below shows a generic example of what the values of each of these lines at a moment in time. The arrows represent when one communication line triggers an event or when an even triggers the line to change.



	T0	T1	T2	T3	T4	T5	T6	T7
On	0	1	1	1	1	1	1	1
Ready	0	0	0	1	1	1	0	0

Done	0	0	0	0	0	1	1	0
Want	0	0	1	1	0	0	0	1

Table 5

We can explain what happens at each line by looking at what events occur at each time interval:

T1. The processor is turned on, prior to this nothing is on, all lines are at a low state
T2. Once the processor is on the FPGA recognizes that no processes are currently going on, Ready and Done are both turned off. This triggers it to ask for the next page turning on Want
T3. The processor sees that the want signal is turned on and so it sets the new address and sets ready on.
T4. Once the FPGA sees that Ready is on it turns off Want. At this point it sends the signal out to the off board electronics to turn on all the pins specified.
T5. After the signal has been sent by the FPGA to the off board electronics, Done is turned on. This signifies that the specified pins have been set.
T6. Once done is turned on Ready is turned off. The Processor checks the feedback lines to see if any of them have been set, and recalculates ready if necessary. The Processor is now waiting for the FPGA to request the next signal
T7. The FPGA turns on Want when it recognizes that Ready and Done are both turned off. And the process repeats

This communication allows the processor to send address lines to the external electronics through the FPGA.

DESIGN JUSTIFICATION

One of the goals of the design process is to simplify all of the processes during implementation. A question therefore might arise: why not directly control the off- board addressing through the processor without using the FPGA? This section explains why we chose the design explained above.

There are two reasons that this design was chosen rather than connecting the processor directly to some output.

1. Safety: Using the FPGA we are able to perform some of the circuitry that is needed for the enable line. Because our actuators require high power, leaving the power supply flowing through them for an extended amount of time rather than a pulse could lead to metldowns. (See section on Solenoid Actuators). While a pulsed signal could be send through the processor, there are dangers in this because of possible spikes in the processor actions. For this reason the FPGA is used for the enabling signal. Because the FPGA is preprogrammed to solid hardware, there is less of a risk of getting lag spikes or unexpected errors.
2. Functionality: Because the FPGA is directly hooked up to the GPIO ports (Expansion header) on the Altera board, we could use these as the output from our board rather than finding some usb

extension. This allowed us to keep as much computation as possible on the board rather than off-board, making it both more compact and cheaper.

ALTERNATE SUGGESTIONS

It should be noted that our communication system between the FPGA and Processor still has room for improvement. This section explores some suggestions that can be implemented in future designs:

1. Additional 'On' line to indicate the state of the FPGA: While there is an enabling line from the processor to the FPGA, this design could be improved by having some sort of feedback from the FPGA regarding its status. This can be extended further to say that there could be an improvement made by adding some sort of on line coming from the off-board electronics indicating that all of the circuits are powered and ready for input.

5.1.3 FPGA

The FPGA serves mainly as bus— given the braille file format outputs a series of addresses corresponding to the pins and their locations. The FPGA that was used was the Cyclone IV that is provided on the Altera development board. Once the signal comes from the Processor, the FPGA uses it to send signals out of the GPIO port. This spread examines the functionality of the FPGA and its contribution overall to the project.

OVERVIEW

The FPGA performs two critical actions:

1. It maps the address lines obtained from the Processor to the GPIO outputs, and maps the feedback lines from the GPIO outputs back to the processor.
2. When it receives the address from the processor and all of the address lines have been set, it sends a 1ms timed pulse to activate the solenoid at that address. This timing is defines how long the solenoid is turned on for.

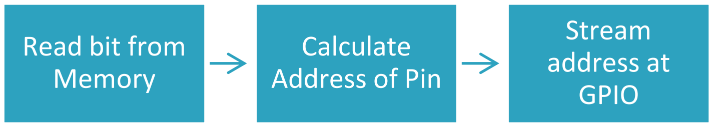
The FPGA sends outputs two main components to the external off-board circuit. The address of the pin to be activated and an enabling pulse to activate the actuators. Both of these outputs are connected to the Addressing circuit. Here we list in short how the FPGA was physically connected to the Addressing Circuit.

The addressing circuit is a logic circuit that decodes the binary address to activate certain drivers on the specified rows and columns. The addressing circuit is surface mounted onto a printed circuit board. The connection between the FPGA and Addressing circuit is completed with a regular IDE cable. This attaches onto the Altera board's expansion header on one side and on a similar header mounted on the printed circuit board. All connections are made through this cable.

This section explores these two functionalities, their implementation and justification as well as alternate suggestions for the FPGA.

ADDRESS MAPPING IMPLEMENTATION

When an input is received from the Processor and stored in memory the FPGA is able to access it at a later time. In order to set a page, the FPGA reads the in the bit stream and for each bit calculates the address. It then outputs the address to the corresponding pin. The following chart shows the flow of the process through the FPGA:



After the braille file is stored in memory the FPGA receives the signal from the processor to start outputting the page. At this point it starts reading the information stored one bit at a time. For each bit, it then proceeds to calculate the address of the pin.

The braille file is stored in serial order, so the bits can be read in series to output to the screen. Depending on the preset values of the page length and width the position of the pin can be determined from its address in memory. The address is output as two binary numbers with the position of the '1' corresponding to the active row and active column.

Once the address is calculated it is output at the GPIO pins. However because the GPIO pins are limited the address is streamed serially out. For example if the address for the row is 010 (second row) then the GPIO pin would have an output of '0' for the first clock cycle, a '1' for the second clock cycle, and a '0' for the last. This clock cycle is called clk. When outputting the FPGA will also output two offset clocks at a frequency of 4clk in order to give the external electronics a time reference for the serial output. See Addressing Circuit section for details.

FOR EXAMPLE

Consider a board with 10 pins, 2 rows of five. The braille file for a sample page might be:

Input: 0010101110

The FPGA will read these inputs and calculate the row and column of each pin as well as the position. In this case it will read the first pin as '0', so the row is 1, and col is 1. So the FPGA would output:

Output: 10 10000

corresponding to the first row and first column. For the second pin it would be:

Output: 10 01000

And so on and so forth.

In order to indicate whether the pin is in the up state or the down state the rows are offset by the length of the row. This is because the actuators for the top and bottom motion of the pin are tied in series. (See section Actuator Grid for more details)So the output for the third pin would be:

Output: 0010 00100

*Note that in the first two examples, since the pin is being set down the last two bits of the row column would be set to 0.

In order to output this to the external circuit the output line for the row is pulsed at 0,0,1,0 for one clk cycle each.

All programming done on the FPGA was done in VHDL. The software Quartus II that was provided with the Altera boards was used in order to program the board.

TIMING ENABLE PULSE IMPLEMENTATION

The solenoid actuators are activated when a short 1ms pulse of high current is passed through them. One of the key functionalities of the FPGA is to pass the enabling pulse to the addressing circuit. This pulse turns on the actuator for the brief amount of time. This section discusses how the FPGA fulfills this job.

Once the addressing lines are all set, the FPGA outputs a timed enable pulse that allows current to pass to the drivers.

JUSTIFICATION

There were some design considerations that were taken into account when choosing to do the timing circuit on the FPGA rather than having an external electronic circuit. There are several advantages and disadvantages to this approach that should be discussed:

ADVANTAGES

- Flexibility: Having the timing circuit on the FPGA rather than having a circuit meant that we were using the clock of the FPGA rather than using capacitors or a 555 timer chip in order to create the 1 ms pulse. Doing some backend calculations we found that the pulse of current necessary to activate the solenoid actuators was 1ms, however there were many parameters that had to be estimated. The pulse might be shortened or lengthened depending on the necessary force of the actuators. Due to this having the pulse timer on the FPGA meant that any changes could be made directly in the VHDL code and uploaded to the development board rather than doing an off-board timer which would have to be physically changed. This gives us room for running performance tests and optimizing the design.
- Less Hardware: Putting the timer on the FPGA means that the amount of hardware circuitry needed to assemble is shortened and unnecessary expenses are reduced.

DISADVANTAGES

- More prone to error: While programming on an FPGA is more flexible, at the same time it is also more prone to error. For example programming a 1 s pulse instead of a 1ms pulse could be dangerous to the circuit.

- There is a limit to the pulse width: While not so important in our instance, the FPGA is not able to time very short pulses (nanoseconds). This did not matter for our design because the actuator had to be activated for a millisecond, however might be an important consideration for future applications if FPGAs other than the Cyclone IV are used.
- If the power supply is constantly providing high voltage drop across the driver circuit, without proper circuit protection it might damage the FETs and consequently the FPGA. In order to avoid this problem the Power supply is not set to be a constant high current supply. Please see Driver Circuit for more details.

ALTERNATE SUGGESTIONS

Some suggestions can be made for the future implementations of this project with regards to the FPGA. The FPGA is not vital for this project, if necessary a bare bone microprocessor could be used to control the addressing lines directly with some intermediary safety logic. This could be important in order to reduce the overall size of the device and to shrink it down in size. This would also lower the power consumption overall of the device. It might be worth experimenting with alternates to the FPGA in the future.

5.2 ELECTRONICS

After the FPGA sets outputs at the GPIO pins, the signal is connected to an off-board circuit board. This circuit board contains all of the electronics necessary to drive the pins and receive feedback. The following image is a render of the printed circuit board design with all of the components on it.

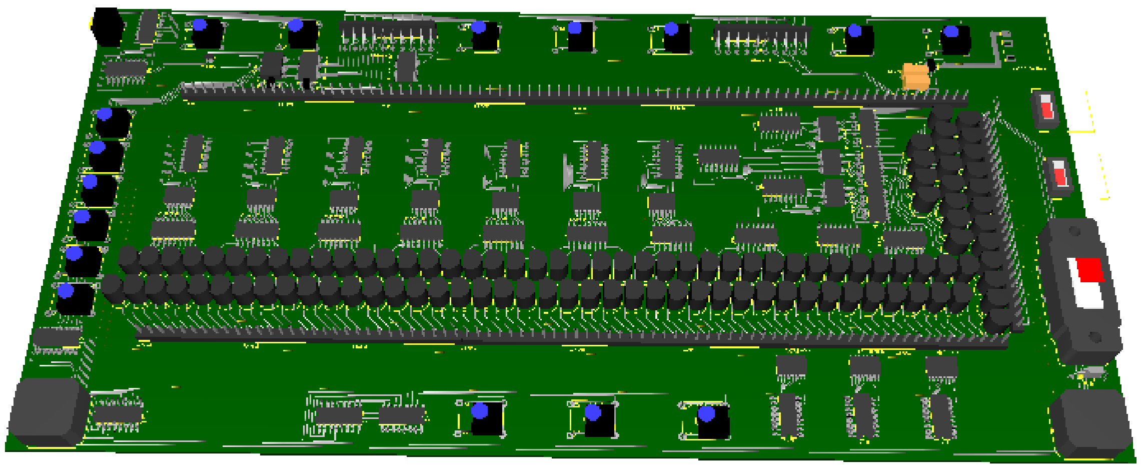


Figure 4 Circuit board

Please see the Assembly and Construction Appendix for details on the gerber files for this design. The Assembly and Construction Appendix also lists a total parts list for the components of all the circuits discussed in the electronics section.

Due to our limited resources we were not able to print out the Solenoid Actuator array. Please see section PCB Solenoid Actuators for more details. Thusly, we are using adapted versions of the circuit in order to demonstrate our device for the competition. The printed circuit board prepared for the demonstration and does not contain the high power supply as it is not hooked up to a solenoid actuator grid.

The electronics section will contain information regarding the Addressing circuit, Actuator Drivers, PCB Solenoid Actuators, as well as the Feedback circuitry. For each subsection the overview, implementation, design justification, and alternate suggestions will be given as well as a section explaining how that module was adapted for the Cornell Cup demonstration.

5.2.1 ADDRESSING CIRCUIT

The purpose of the addressing circuit is to minimize the number of lines going between the FPGA and the external circuit. Although we might want to control thousands of pins, we want to ensure that the number of lines contain the information about which pins are being activated and where they are being set to a minimum. The addressing circuit serves exactly this purpose. Once the FPGA outputs an address, the next step is for the addressing circuit to decode the signal.

There are several components that the addressing circuit consists of. This section examines the functionality of the addressing circuit, first a summary of the circuit is given and then its two main subsections are explored in detail.

ADDRESSING CIRCUIT OVERVIEW

The addressing circuit takes in two sets of inputs from the FPGA: the address lines and the enable line. This corresponds to the two functionalities of the addressing circuit:

1. Addressing line decoding
    a. The addressing circuit takes the encoded output of the FPGA and decodes it to activate the row and column corresponding to the pin that needs to be activated.
2. Passive Matrix addressing
    a. The grid of braille dots can be addressed by their row and column. For example if the first dot has to be turned on, then it is the first row and the first column that needs to be activated.
    b. The Matrix addressing is set up so that turning on a row and a column activates the component on the grid that lies on the cross-section of the two lines. Because the solenoid actuators are bistable and only have to be triggered by a 1ms pulse, a passive matrix addressing system is used to turn on one pin at a time.

The Addressing Circuitry decodes the input from the FPGA, and activates the corresponding row and column. When the row and column are activated the actuator at the intersection is turned on. It should be clarified however that the row and column are not just activated by turning them high, there is a driver circuit that when a row or column is turned on by the addressing circuit actuates the row pulling it to a high voltage or ground as necessary. There is a driver circuit for each row and column.

Since they are both on the Circuit board the connection between the addressing circuit and the driver circuit are just created in copper traces on the board. To see the full construction of the circuit please see the Appendix.

These two parts of the addressing circuit are discussed in the spread below.

ADDRESSING LINE DECODING

Throughout the course of the project we have worked with two addressing circuits. The spread below outlines each addressing circuit, lists its strengths and weaknesses and gives justification for why we chose the addressing circuit that uses shift registers instead of the one using decoders for our final design.

ADDRESSING USING DECODERS

If the output of the FPGA can be set to be an encoded binary coding that refers to a specific row and column rather than simply setting a '1' in the position of the active row and address position. Suppose there are two sets of addressing lines that come into the addressing circuit, 5 lines that represent which column the actuator to be activated is on and 3 lines that represent the row. The array of solenoid actuators has 32 columns and 6 rows. This section explains how these lines are decoded to address a certain col and row of the actuator grid.

IMPLEMENTATION

There are two parts to the decoding circuit, the circuit that decodes the rows and the circuit that decodes the columns. First let us consider the circuit that decodes the rows from 3 lines in to upto 8 lines out. The following diagram shows the 3-8 decoder that is used in order to achieve this. Please see the Assembly and Construction Appendix for the truth table for the decoder chip.

As can be seen in the truth table when a 3 digit binary input is taken in the output is the corresponding digital line high. This ensures that only one line can be selected at a time. This decodes the three address lines from FPGA into the corresponding row and indicates that it is selected by setting it low (This is an inverting decoder). In order to produce a high when the selected line is active, an inverting nor gate is added to the end of each of these outputs.

Note that there are three enable lines. The output is set according to input lines A,B,C only when all three of them match the criteria necessary. The enabling line from the FPGA that pulses 1ms to indicate the actuator that corresponds to the row and col is connected to this. This is explained in more detail in the next section: Passive matrix addressing.

For example if the input from the FPGA is 000 and the enable is turned on then the '0' pin out of the decoder is low and the rest are high. Similarly if it is 010 then the '2' pin out is turned off and the rest are high.

The decoding circuit for the columns uses the same components as the decoder for the row, however there are no decoders that decode 5 to 32. For this reason two decoding chips are combined in order to allow the great amount of input to output ration. The circuit diagram below portrays the circuit used for this decoding logic:

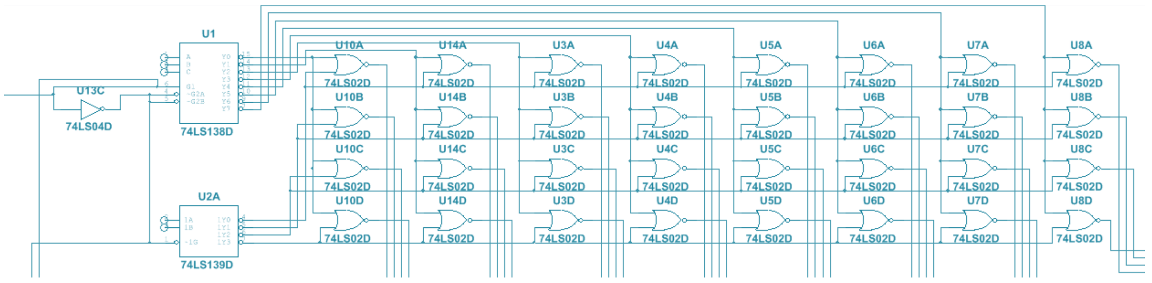


Figure 5 Addressing Lines

As can be seen here two decoders are used: a3 to 8 and a 2 to 4. The outputs of these decoders are then passed through nor gates, where one input to the nor gate is the output of one of the decoders and the other from another. This is repeated for each possible combination of decoder outputs.

Consider an example where the output of the first decoder is: 01111111, and the output of the second decoder is 0111. The nor is taken of every combination of these outputs, the output is only the nor gate that takes the input from the first and the second will turn on, the rest will be off. Similarly if it was 11110111 and 0111, the nor gate that takes in the 5th line from the first decoder and the first from the second will be on. Note that the enable lines have to be on in order for the output to be passed through.

From this it can be seen that the 5 line input can be converted to 8*4=32 outputs.

In summary, the addressing circuit uses a combination of decoder chips and logic gates in order to decode the input to output a high at the selected lines. These lines are enabled only when the enable lines are on. In order to see how these circuits were built please see the appendix on construction.

JUSTIFICATION

One important design decision arises from the visible complexity of the routing of the connections in the logic decoding circuit: What are the advantages of implementing the decoding in hardware rather than on the FPGA?

The main reason that it was decided to implement this part of the addressing circuitry in hardware rather than on the FPGA was because of safety reasons. The addressing system of the solenoid actuator grid works so that when a certain row and a certain column is activated, the actuator that falls on both of those lines is activated. A problem would arise if two rows and one col was activated. In this case two actuators would turn on. And if by some error all of the rows were activated, and all of the columns, the power would run through the entire grid. This would mean that the power necessary to activate a single actuator could not be met. Programming the decoding logic on the FPGA could therefore be risky. With this hardware implementation the use of the decoders ensures that only one line can possibly be selected regardless of the input from the FPGA.

Another reason that it would not be advisable to implement this on the FPGA is because of the sheer amount of outputs. Here an example of a 32 by 6 grid is discussed. This is only two half lines of braille. The goal of the final product is to have a full page of 15 lines of braille and have 40 characters across. This would not scale well with the number of outputs necessary from the FPGA. For this reason the decoding logic was separated from the FPGA.

ALTERNATE SUGGESTIONS

For a large scale production it might be suggested to design a decoding chip that performs a high input output ratio decoding rather than cascading and combining multiple chips. For industrial purposes combining logic chips like this can be inefficient due to the increase in number of necessary copper traces, connections as well as assembly time and propagation delay through the chips.

SHIFT REGISTER DECODING

The biggest problem with the previous design is that it's not scalable. The design required that there was an AND gate at each intersection. So if we had 80 columns, we would need 80 and gates. And for our device we need 80 columns and 24 rows. This means that if we want to change the number of rows we have then we need to change the number of ICs we have on the board and it increases linearly. This section explores the use of the similar devices in order to set the addressing lines to solve this problem.

IMPLEMENTATION

In order to solve this problem, rather than setting all of the pins at once, a shift register based circuit was designed. Specifically we used an 8 bit shift register which takes in a serial input and outputs it in parallel. By chaining these shift registers it was then possible to set any number of pins using a single serial input. Please see the Appendix for the truth table for this unit.

A basic 8-bit shift register has three input pins and 8 output pins. One of the inputs is the serial input, the second input is the clock(clk) and the last is the not clear(~clr). On the rising edge of the clock the input is sent to the Most Significant Bit (MSB) of the output and the previous MSB is shifted over. In this manner the shift register has the ability to remember the past inputs to the device.

FOR EXAMPLE:

Consider a 4-bit shift register depicted below[2]:

[2] Taken from: http://www.allaboutcircuits.com/vol_4/chpt_12/4.html

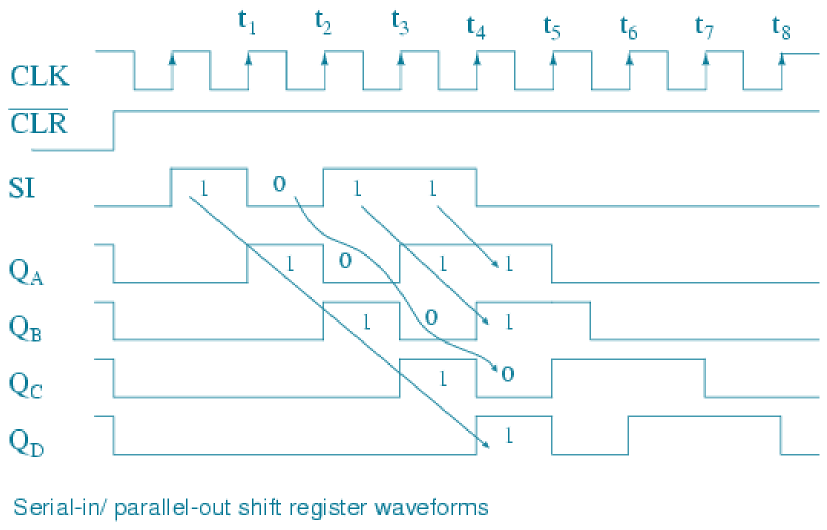


Figure 6 Shift Register

On every rising edge of the clk, the signal input (SI) is shifted to the MSB and all inputs before are shifted over. At the end if the input was 1011, then A= 1, B=1, C=0, D=1. When the clear is not low, it allows the bits to be set. These shift registers can be chained so that the last bit from the first register is the first bit for the next hence creating a long memory history needed for a many row device.

While the basic 8-bit shift register would allow us to set the output to be 1 at the row and col that we want to activate, an issue arises as we shift the bit over—the output is constantly being updated so the 1 would shift along. For this reason we use an 8-bit shift register with output latches. The diagram below[3] summarizes the three stages of the shift register.

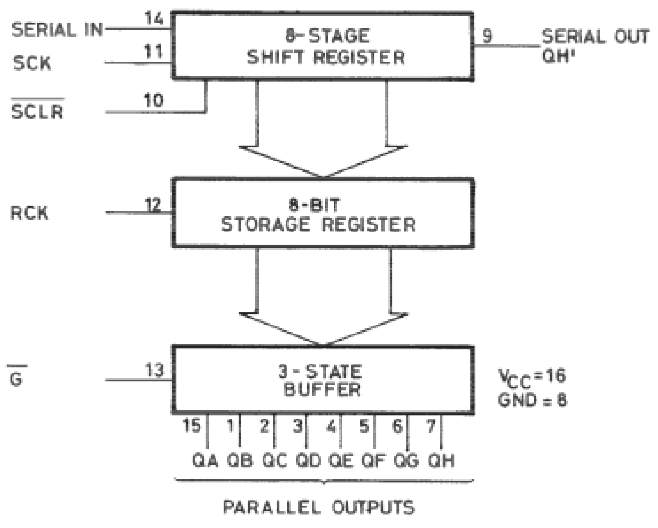


Figure 7 Shift register block diagram

[3] Taken from: http://pdf.datasheetcatalog.com/datasheet/SGSThomsonMicroelectronics/mXtuvqt.pdf

The first step is a regular 8 bit shift register. The second part is an 8-bit storage register. This is a series of latches that when the RCK is at a rising edge stores the output of the shift register. When ~G is set to low, the parallel outputs are then enabled.

This allows us to ensure that the bits aren't changing as we input the address as well as gives us the option of turning the output on and off with the ~G signal. Using the ~G signal we can dictate the 1ms pulse that transfers over to the driver circuit. We can further chain these shift registers to create many outputs in order to remember the position for a many bit grid.

However using this design is not enough. One of the strengths of the Decoding circuit driver was that it never allowed more than one output to be set to 1 at a time. This ensured that two pins were not accidentally turned on. With the shift register as it is, there is nothing to ensure that that won't happen.

Because of this our implementation uses the reset inputs in order to reset the outputs if a one is received. The idea is, if a one is received then wipe out anything prior to that that was a 1 and then store this address line value. Thus for a particular row if we try and set more than one row high it only takes the last input.

The circuit diagram below shows the shift register chain and the inputs given. This register chain could set 32 lines.

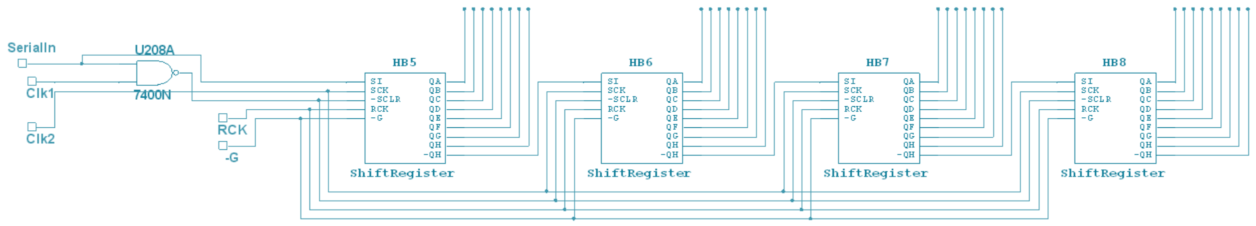


Figure 8 Shift Register Addressing circuit

FOR EXAMPLE:

This example shows how the reset of the register allows us to insure that the output only has one line active at a time. The waveform of the output at the register is:

*Note that this is the output of the basic shift register, to show the functionality of the logic in ensuring that two lines are never high. In the latched shift register the RCK and ~G would have to be set to get the output to change.

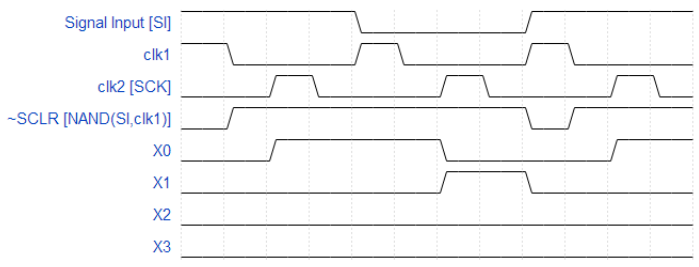


Figure 9 Shift register timing diagram

Time	Output
T0	At the initial point the Signal input(SI) is high. Clk1 is set to high as well so this resets all of the bits in the register to 0. X0-X3 are all 0.
T1	Clk1 is low this ensures that nothing is being reset as we set the shift register with the next clock tick.
T2	Clk2 is rising, this is the SCK which sets the register pins. Since the SI is high, it sets the first pin X0 to high.
T3	Both clocks are low to ensure that nothing is being set and reset at the same time, nothing changes
T4	Clk1 is high again but since SI is low, it does not reset the registers. Nothing changes.
T5	Both clocks are low
T6	Clk2 is high setting the SI value to the register and shifting outputs. X0 is now low, and X1 is set high.
T7	Both clocks are low
T8	Clk1 is high and SI is high. This means that the FPGA is trying to set another row high. In order to avoid this all previous entries are cleared through the ~SCLR going low. X0-X3 are all low.

JUSTIFICATION

The Decoder circuit had an issue where the number of input lines coming from the fpga was linear to the number of rows or columns there are. With this circuit, only 6 lines are ever required in order to set both the row and the columns: Row SI, Col SI, Clk1, Clk2, RCK, ~G. This means that there is a huge conservation in cost of production both due to space it takes up including the number of connections that need to be made on the circuit board, as well as the cost of the extra ICs. The tradeoff in our case is that the time it takes to set the address is higher.

However, although the time it takes to set the addressing lines is high, it is still negligible in comparison to the 1ms it takes to set the clock. With a MHz clock input for the values of Clk1 and Clk2 it takes 4 micro seconds in order to set one row, at our 80 row design that is 320 micro seconds. If we set the shift register as the previous pin is being set, the time is reduced to nothing since we can do both simultaneously.

Furthermore, with the logic circuit that blocks two rows from being high at the same time, there is no real advantage of the decoder addressing over this system. Due to these reasons the Shift Register circuit was chosen for our final design.

ALTERNATE SUGGESTIONS

If we want to keep the decoder system it would be possible to combine it with the shift register system in order to reduce the number of inputs. However at the time being no advantage to that approach can be seen and therefore is not persued.

PASSIVE MATRIX ADDRESSING

The second job of the addressing circuitry is to activate the appropriate solenoid. The Solenoid actuators work in a bistable manner, when we trigger the pin to stay up/down it holds its position even when the power is cutoff. A pulse input triggers their position but constant power does not have to be continuously provided. For this reason the passive matrix addressing system is able to be used. The spread below discusses how enabling one row and one column of the grid of actuators activates a single actuator.

IMPLEMENTATION

The circuit diagram below shows a small grid of inductors that uses passive matrix addressing:

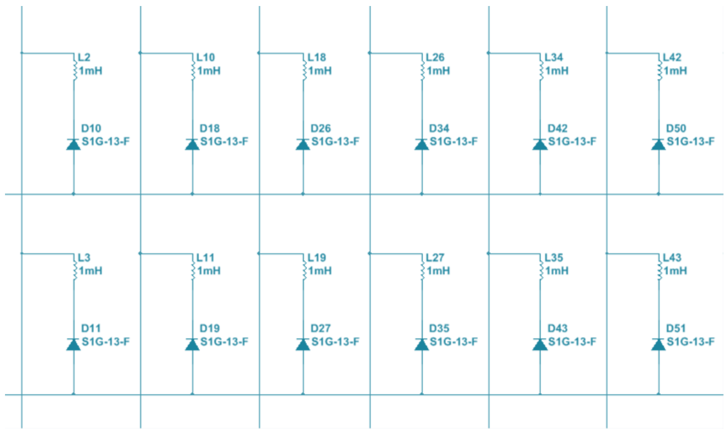


Figure 10 Matrix addressing

In this case suppose the first row and first column are 'activated.' By activated we mean we are allowing current to flow through them. Suppose the rows when activated are connected to a 15V source, and the columns when activated are connected to ground. When they are not connected, the lines are in a high impedance state. It can be seen from this therefore that when a certain row and column are activated, current can flow through the inductor. In this case the inductor represents the solenoid. The reason that there has to be a diode in parallel is to prevent sneak currents. For example consider the circuit without the diodes. When the first row and first column are activated, current will flow through inductor (1,1), but since there are no diodes, current can also flow through (2,1) ,(2,2),(1,2). This would mean that those solenoids could also be activated and pull power from the solenoid that we need to activate.

This matrix addressing system means that only one solenoid can be activated at a time. This works for our solenoid actuators because they are bistable.

JUSTIFICATION

One reason that the passive matrix system is better than an active matrix system, is that power does not have to be constantly provided to the system. This is especially useful for a device such as an E- Reader because a page of text can be loaded and might take some time to finish reading before it has to be refreshed. A drawback of this system however is that only one pin can be activated at a time, and so to refresh a page of 1000 pins (~4 lines) would take 3 or 4 seconds because of the time it takes to activate the pin and then load the information for the next one. This can be unusable to some extent.

ALTERNATE SUGGESTIONS

One way that we discussed improving this addressing system is to have several addressing grids run in parallel. For example instead of having a 32 by 6 grid, have 4 8 by 6 grids. Given multiple addressing systems and driver circuits, this would allow for multiple pins to be loaded at once. Pipelining the pin loading would also decrease the time needed to load a full page of text.

A possible disadvantage to this method however is that it would take a lot more power in order to activate multiple pins and the time saved from doing so might not be worth the bulk of the extra power supply necessary. Tests should be conducted to determine the minimum power necessary to throw the pins.

ADAPTATIONS FOR DEMONSTRATION

For our demonstration circuit we used the shift register method of actuation. Although we do not have solenoid coils to activate we are still able to show the design of the shift register for storing line values.

In order to show the passive matrix addressing, instead of using solenoid coils we are using an array of LEDs wired in the same formation. The following circuit shows the inside of the LED array that we are using. With this display we will be able to show that the addressing circuits work as a proof of concept. Please see the Appendix for the part number of this component.

The following renders show the Printed Circuit boards designed for testing this demonstration. The circuit board on top shows the passive matrix addressing. It is a grid of LED arrays. This connects with the bottom circuit through headers and alignment pins. The bottom part of the circuit contains a simplified version of the driver circuit because high power is not needed.

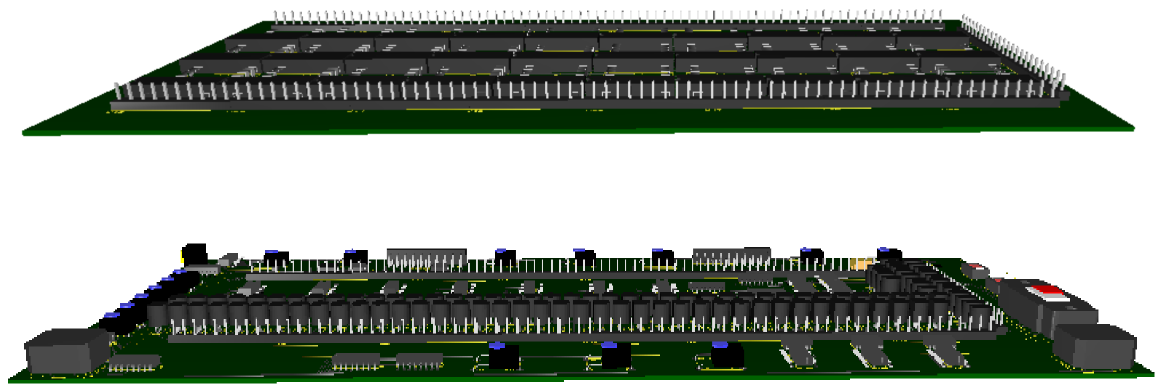


Figure 11 Circuitboards for demonstrating addressing and driver

5.2.2 ACTUATOR DRIVER AND POWER SUPPLY

The driver circuit is the part of the circuit that activates a row or column depending on whether the row or column is high coming in from the Addressing circuit. This section explains the details that pertain to the Driver Circuit. Fist an explanation of the circuit is given as a standalone without considering the addressing lines. Next this circuit is applied to the addressing circuit, and the final circuit is explained. The justification for this choice of the driver circuit is then given and alternative suggestions are supplied. At the end of the section the power supply to the actuator driver is outlined.

OVERVIEW

The purpose of the driver circuit is to provide the necessary power to the solenoid actuator. We can simplify this problem without taking into consideration the addressing grid. This spread explains the circuit necessary to provide the necessary power to a singly solenoid on an enable signal.

The circuit below portrays the necessary for this:

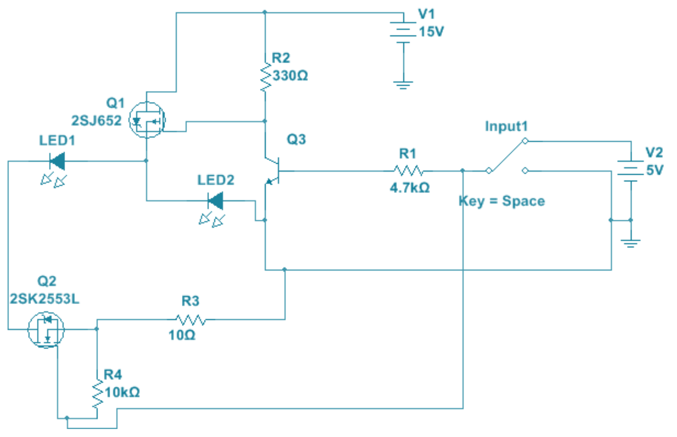


Figure 12 Single pin actuator driver

This design is a standard Push pull amplifier (Totem pole) design with two MOSFETS. When the enable is high (in this case Input1), the bottom n-channel Mosfet is activated and pulls the bottom part of the inductor to ground. At the same time on the top, when the enable is turned on, a BJT turns the p-cannel mosfet on pulling the top part of the inductor to power. This allows power to flow through the inductor. The Enable line is a 3.3V logic level where a high is above 2 V, whereas the power that flows through the inductor is our high power 15V 30A.

IMPLEMENTATION

We can now imagine that the enable lines are connected to a row and a column of the addressing circuit. When the specific row and column are turned on the addressing circuit, the driver is turned on and a high power is sent through the coil. The high power that flows through the solenoid actuator is a 1ms pulse. This is controlled by the enable being pulsed. When the FPGA activates the addressing circuit for 1ms this translates to the addressing circuit activating the driver circuit for 1ms.

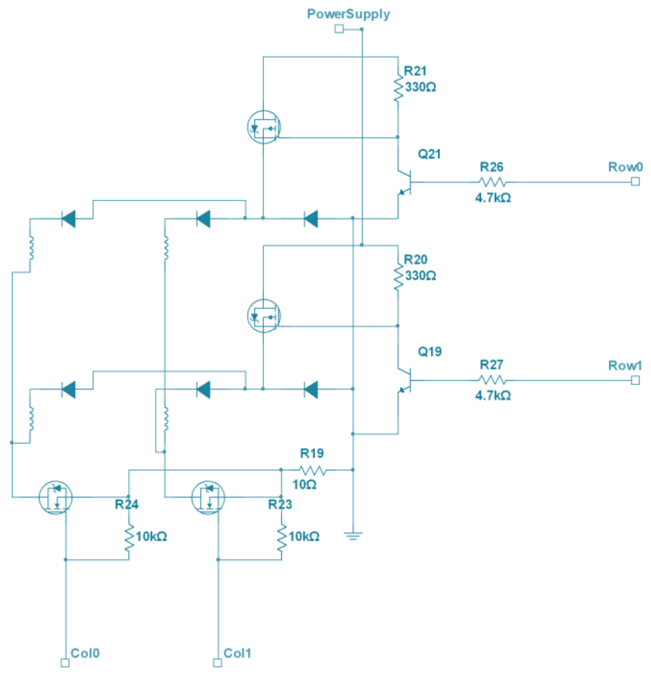


Figure 13 4 pin actuator driver

The circuit diagram below show a part of the circuit that connects the addressing lines to the driver circuit. Here 4 actuators are represented by the inductors.

As can be seen here only the driver circuit with corresponding to the activated row or column is turned on and allows a high power pulse to pass through.

FOR EXAMPLE:

When the Row0 and Col0 are set to high, the first coil is turned on.

JUSTIFICATION

The amplifier design is a common circuit used in such situations where a logic level enable is used to power a high power part of the circuit. One design consideration however was when choosing what MOSFETs to use or what type of transistors to use. The reason that these specific MOSFETs were used was because they are high power and have maximum power ratings much higher than the expected values. This was useful because this ensures that they will not burn out. Furthermore because MOSFETs act as open circuits between the gate and source voltage, there is no need for a high current to drive the circuit, a simple voltage amplifier and inverter do as shown in the diagrams.

ALTERNATE SUGGESTIONS

While we used very high power MOSFETs it would be a worthwhile test to check different MOSFETs and see how long some of the lower power ones can last. Furthermore one of the concerns that came up is the possible heat dissipated through the MOSFETS. Since all of the components are surface mounted, there is no space for heat sinks. This should be considered when choosing the transistors and Mosfets.

Other problems that might arise if the Power supply was directly connected to the Mosfets is if the inductance of the coil is very high. This might cause the voltage drop across the Mosfet to exceed its maximum ratings. The inductance of the coil should be found and the circuit protected from voltage spikes.

The diodes we are using in line with the solenoids are rated at 2A, however the pulsed rating is much higher than that. Tests should be conducted to test the maximum operating life of the diodes.

POWER SUPPLY

The driver circuit redirects a high power from the power source to the solenoid actuators. Other than that there is also a TTL 5V Power supply that connects to the other IC components of the circuit. These are connected to the printed circuit board through a PCB terminal Block of the series (MKDSN) see the Appendix on Construction for more details. There is one high power supply and ground relating to this high power supply and one 5V supply and a digital ground for this low power.

POWER SUPPLY

In order to provide the necessary high current to the solenoid actuator a charging based power supply was designed. Rather than having the power supply be a bulky piece that outputs an average high current, a two charging capacitors are used to charge and discharge the circuit. In this section we define a power supply for the actuator driver as well as provide justification for using such a circuit.

OVERVIEW

The following circuit describes the charging circuit that was designed. There is a big capacitor that holds the charge for multiple pulses. This is attached to a little capacitor through an on/off relay. The little capacitor contains the charge for a single pulse.

The big capacitor is stored up so that when the board is completely reset it can provide a high amount of power in a short amount of time. The smaller capacitor holds the charge necessary in order to pulse the circuit a single time and set the pin in the appropriate position.

FOR EXAMPLE:

If we want to set one of the pins up, initially the larger capacitor is connected to the output and the smaller capacitor is not connected to the large one. The smaller capacitor was previously charged. Now the pulse is fired to set the pin in the up or down state completely discharging the smaller capacitor. At this point the smaller capacitor is disconnected from the circuit and connected to the large capacitor to be quickly charged up. Once it reaches a certain voltage (15V) it is disconnected from the large capacitor and joined onto the main circuit again. It is now waiting to be discharged at the next pin change.

JUSTIFICATION

Besides the fact that this configuration eliminates a need for a large power supply, the smaller capacitor can be set up in such a way that it can only pulse for 1ms. This means that even if there was a circuit failure elsewhere in the device and one of the coils was continuously connected it cannot burn out because there is no more charge left on the capacitor. This would also solve the issue of the FPGA controlling the duration of the pulse since in this case even if the pulse is longer the capacitor is not able to discharge for more than 1ms.

ALTERNATE SUGGESTIONS

Although this circuit is more efficient in terms of power preservation, it does not necessarily mean that it is as fast. The process of charging and discharging the capacitor depends on the resistances of the circuit. It is also possible that the large capacitor will take a very long time to charge meaning that although the time it takes to refresh a page might be shortened, it is only shortened if the time between page changes is longer.

ADAPTATION FOR DEMONSTRATION

Because we were unable to get the solenoid actuators, the high power circuit is not used in the demonstration. The LED demonstration will need a low power supply and due to this the actuation circuit has been modified to have only BJT transistors. This circuit is portrayed below:

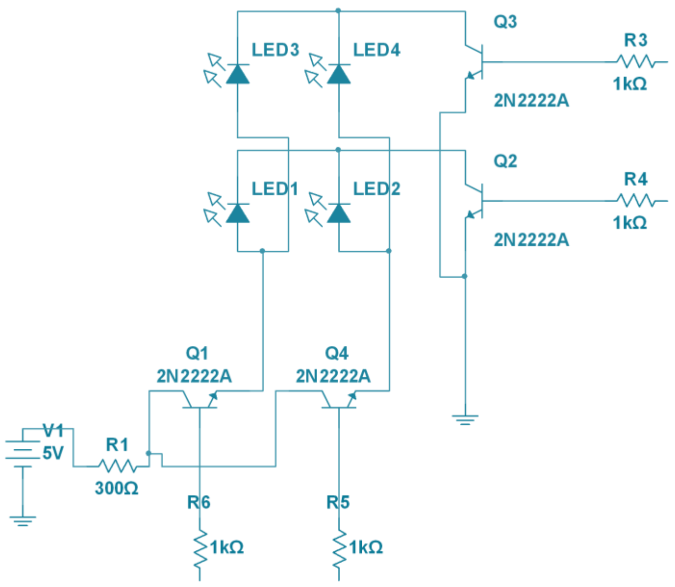


Figure 14 Demonstration actuator driver

The principles are the same and this push pull configuration is a proof of concept that the driver circuit worked.

For the demonstration a small 6 pin 1 character display is being put together. For this the high power driver circuit will be used as described above. The power supply for this section will not, however, be tested to the limit since only one character is being activated.

5.2.3 PCB SOLENOID ACTUATORS

The foundational hardware that allows the each pin of the braille dot to move up and down can be condensed to the solenoid actuators. This section explains the functionality of the solenoid actuators.

A solenoid actuator consists of a tube of a solenoid and a core of magnetic material. In general a solenoid actuator is solenoid with a magnet in it. When current is pulsed through the solenoid a magnetic field is induced and this reacts with the magnetic core pushing it into or out of the solenoid depending on the direction of the current.

In this case the main goal of the project was to create solenoid actuators that were printed onto a printed circuit board. First we explain how the solenoid is printed onto the circuit board. The design Implementation, justification and alternate suggestions are then given.

OVERVIEW

The foundation of this project is the Solenoid Actuators that are printed on a printed circuit board (PCB). The main issue with braille actuators is that the spacing between the pins is so limited that there is no space to add hand wound or wire wound solenoids. The idea of this project is to use the printed circuit board technology and print a loop of the coil on each layer using the copper traces.

When the magnet is less than half way through in the solenoid, when the solenoid is turned on, it is pushed to the opposite state. The magnet pin is positioned to be outside the solenoid by using a steel plates with holes in between two PCB solenoids. When the pin is ‘on’ the magnet’s bottom is attracted to the steel layer, when it is ‘off’ the top of the magnet is attracted to the steel layer. See the section on Braille Pins for an explanation of how this moves the braille pin up and down. By using two sets of printed circuit boards, it is possible to control when a pin moves up or down. When the top coil is activated it pulls the pin up, when the bottom one is activated, it pulls the pin down. If the pin is exactly half way through the solenoid it experiences no force. So if there was a board through the middle of the pin when it is in the up position and the board is activated, the pin will not move, but if it is on the bottom it will get pulled up. Using this we can control whether the pin is turned on or off.

Consider a multilayer printed circuit board. We can create a solenoid on this board by printing a loop of copper on each layer, and connecting them through hidden microvias. The following image portrays several sample layers of the printed circuit board:

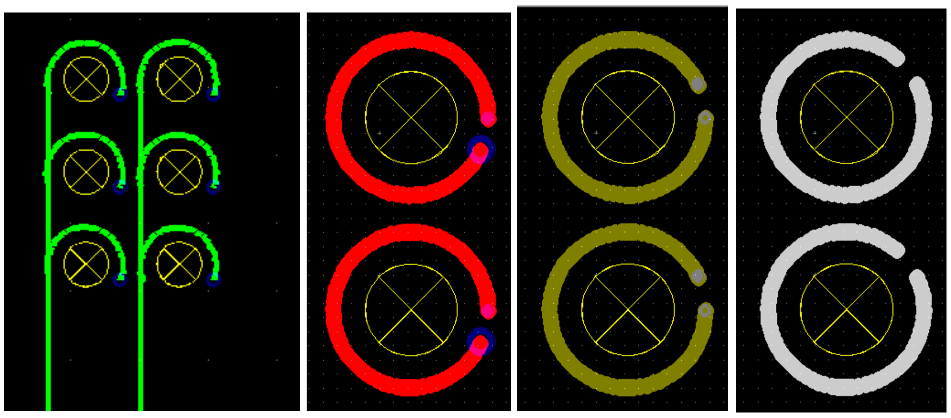


Figure 15 PCB layers

As can be seen the top layer in green all of the columns are connected because these correspond to the columns, the rows are not connected because each loop is connected to a unique diode.

If the red layer is the second layer, a via connects the top layer to a point in the red layer. The brown layer is below the red layer and is connected to it continuing the loop of wire, and then the silver layer connects to the brown layer continuing the loop. This process was continued to create a solenoid loop.

IMPLEMENTATION

During the design process several prototype boards were developed. Initially we were planning on creating a 30 layer PCB, then we simplified our process down to 10, and 4 layers. Our final design had 6 layers.

The problem with printed circuit board technology is that although the cost of mass-producing PCBs is very low, prototyping prices are very high. With our PCB design it required High Density Interconnects, or stacked vias. This means that we needed a via going in between each later making the board very expensive to produce. The following section explains the process for developing PCBS and explains why sequential buildup is expensive.

PCB MANUFACTURING PROCESS OVERVIEW

HDI PCBs are manufactured in general using the sequential buildup technology. The reason that the cost is driven up for our PCB design is due to the stacked vias. Say we want a 6 layer board with our vias on every layer. In order to manufacture such a board, the manufacturer will start with the middle two layers, a dielectric with the copper traces on the top and on the bottom.

Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6

Next they add the dielectric that would separate layer 2 from 3 and 4 from 5 and then laser drill the holes for the vias going between 2-3 and 4-5. This process of layering on the next two layers requires a lamination process.

This is repeated until all the layers are completed. So for a 10 layer board it would require 4 lamination.

Companies that offer HDI technologies (High Density Interconnects) offer stacked vias, most offer 1-X-1 or 2-X-2 which means that there is some middle core (in our case just layer 3-4 but it could be multiple layers) and then the number refers to the stacked layers. We would be looking for something like 4-X-4. This is done, but more rarely because of the lamination cycles. We need an any layer interconnect technology which is only offered by some companies.

PCB DESIGN AND QUOTES

The following 6 layer board was designed. The two images below show the full PCB with the addressing lines on the top and bottom layers. Note that the addressing lines were kept as wide as possible in order to decrease the voltage drop across the trace. The diodes were placed under every coil on the top side of the board.

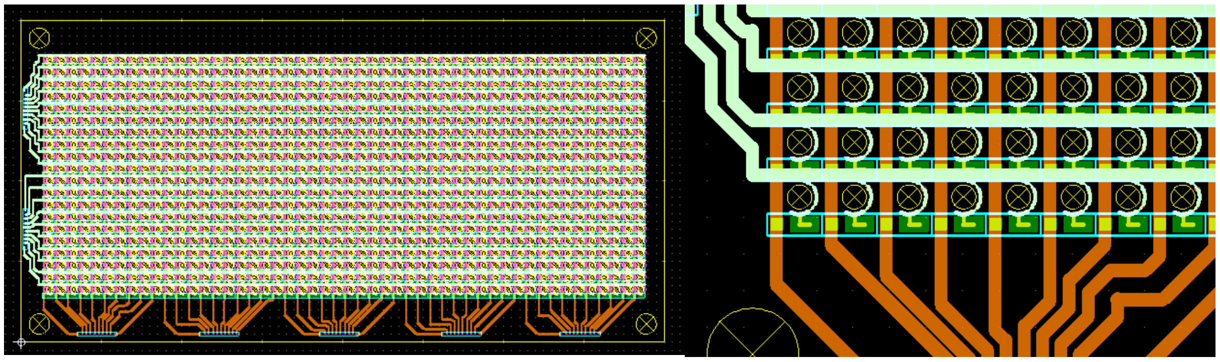


Figure 16 PCB Matrix addressing

The following images show the progression on a single coil from the top layer to the bottom from left to right. The first layer has the diodes for the passive matrix addressing, a microvia connects it to the loop below. There are two loops on the second layer that connect to the third fourth and so on until the row addressing is reached at the bottom layer.

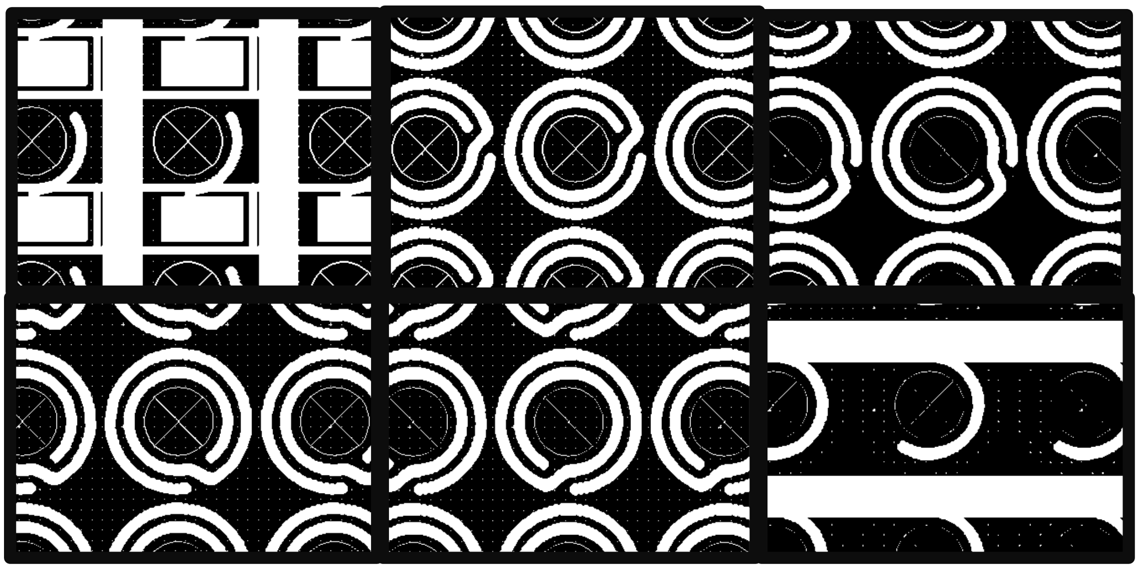


Figure 17 PCB Solenoids

For this PCB the cost for developing two boards was around $6000.

The following table summarizes the quotes we obtained for each PCB design from different manufacturers for the different prototypes:

Company	Layers	Quantity	Price	comments
Speedy Circuits	4	1	2200	
Sierra Circuits	10	1	12763.8	
Alpha Circuit Corporation	10	1	5000	This is the price for a 25 day lead time (cheapest)
Speedy Circuits	10	1	10000	
Viasystems Sales	10	8	19136	$2392 each
Sierra Circuits	4	1	2948.88	after 20% discount
Speedy Circuits	4	2	2250	$1125 each
AT&S	4	24	3997.44	$166.56 each
Alpha Circuit Corporation	4	1	1250	This is the price for a 20 day lead time (cheapest)
San Francisco Circuits	10	10	26325	$2632.5 each
Epec	4	3	4017.24	$1339.08 each

MODELING THE PCB COILS

Although we were not able to print out the PCB because it was outside of our budget, we put together a demo of the coil geometry using 35 gauge wire. We tested this circuit with a magnetic pin and were able throw it up and down our bistable positions. It required a 20V drop across the solenoid.

Because the geometry of the model of the coil that we developed is very close to the PCB geometry we expect the 6 layer design to work. We worked extensively with Sierra Circuits who offered to print a 6 layer board with $6000. While we were able to get a lower price on the 4 layer board, due to the resistance of the traces we need to put the addressing lines on their own layers. This is because if the traces are too thin, they run a chance of burning up.

SIMULATING THE PCB COILS

Finite element modeling software COMSOL Multiphysics was used in order to simulate the magnetic field of the 8 loop coil (6 layer board.) The following images show the results of this simulation.

First a 3d model of the coil was created using 3ds Max shown in the figure to the left. The 3d model was then imported to comsol. The image to the left depicts the potential difference across the coil.

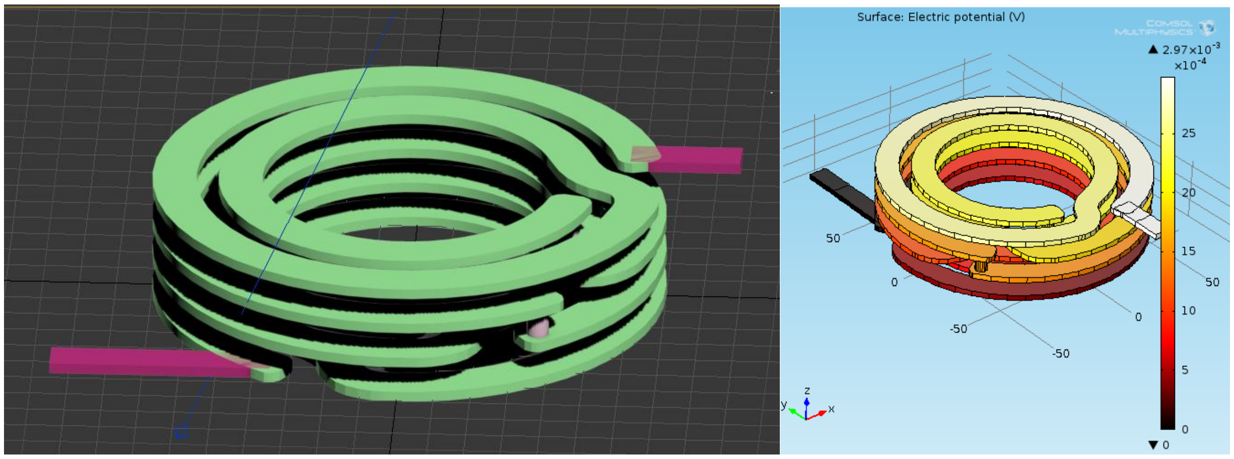


Figure 18 PCB Solenoid

The magnetic fields were then simulated for this coil. The following images portray the magnetic field densities throughout the loops.

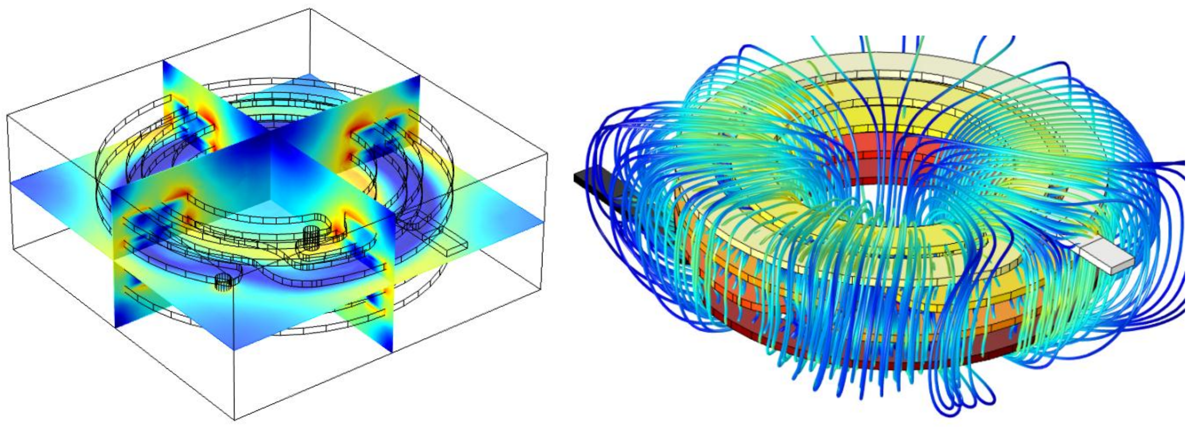


Figure 19 Solenoid Magnetic field

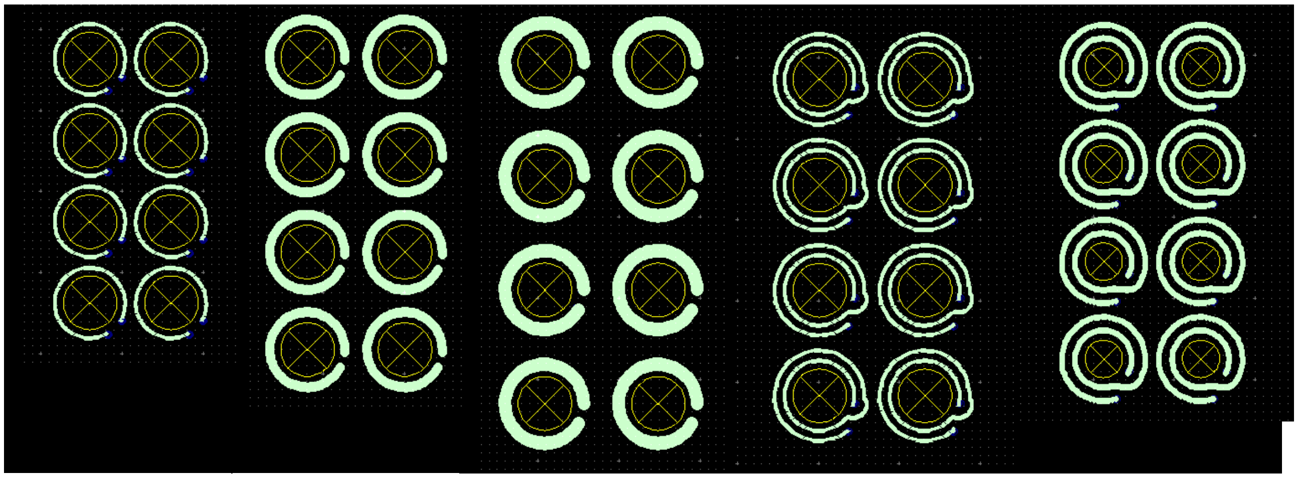
JUSTIFICATION

The PCB coil loop idea is something that mass produced will become significantly cheaper. Because of this along with the high reproducibility of the PCB and low assembly costs this is a good solution to our challenge definition.

ALTERNATE SUGGESTIONS

PCBs were designed for many trace configurations. In the long terms it would be worthwhile to test these different configurations. The following list a few including their measurements.

100 mil spacing 66 mil hole diameter 6 mil trace width	120 mil spacing 66 mil hole diameter 12 mil trace width	140 mil spacing 66 mil hole diameter 10 mil trace width	130 mil spacing 66 mil hole diameter 6 mil trace width	120 mil spacing 46 mil hole diameter 10 mil trace width



ADAPTATION FOR DEMONSTRATION

For the demonstration we will try to hand wind solenoids using the same technique as before to have a proof of concept single cell display working. This will show that the 6 layer board should work in the long run.

5.2.4 FEEDBACK CIRCUIT

A feedback circuit was designed for the buttons and input from the user. This spread explains the relevant design choices made for this circuit.

IMPLEMENTATION

The basic circuit design used for the feedback buttons is shown below.

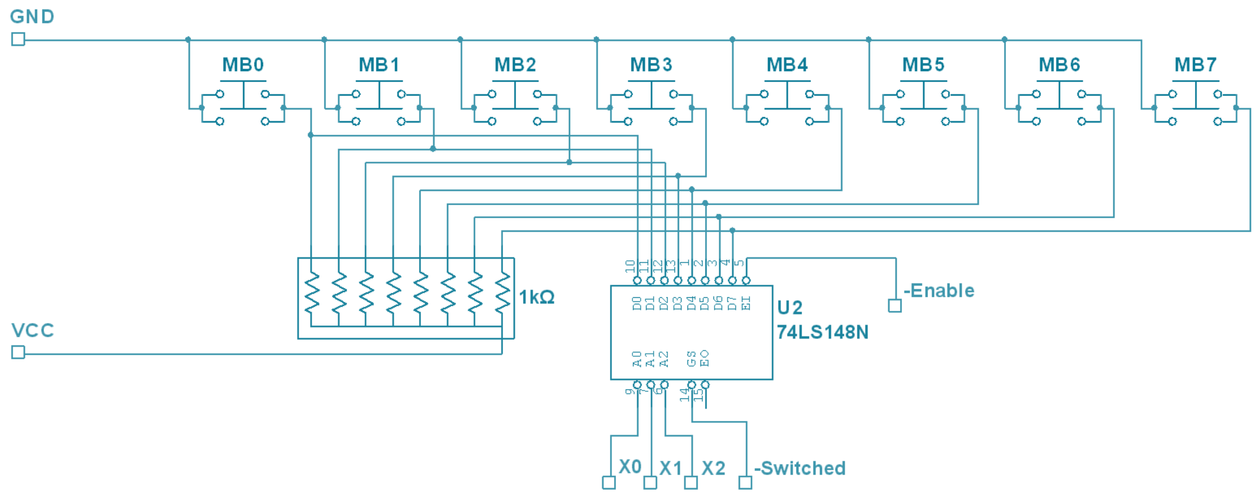


Figure 20 Feedback buttons

The 8 pushbutton switches momentarily bring the input voltage to the inverting encoder down to ground. When a button is pressed the ~Switched signal is low. This circuit is a basic pull down circuit coupled with an encoder to allow the output address of the button to be read by the FPGA.

Because he have many buttons on the page the following circuit summarizes how these are hooked together:

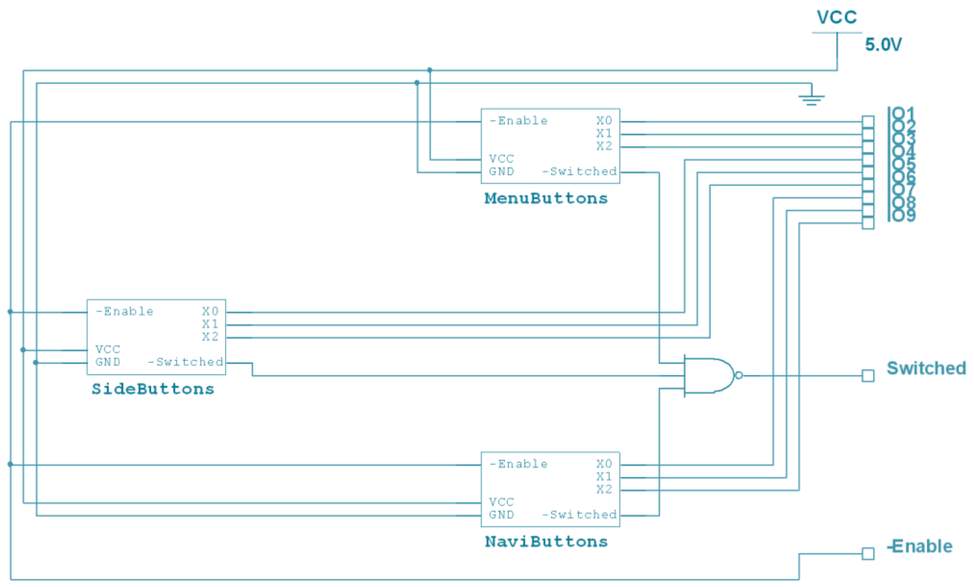


Figure 21 Feedback circuit layout

Because the ~Switch signals are NANDed together, the output of the Switched signal can simply be checked in order to determine if a button was pushed. This way the code does not have to check all addresses separately.

The output of these encoders is put through a debouncer in order to stabilize the input before it is read by the Microprocessor.

DESIGN JUSTIFICATION

The encoding circuit makes it easier to transfer the information over to the Microprocessor and reduces the addressing lines. In the long term this should be coupled with a parallel to serial shift register in order to further reduce the number of inputs.

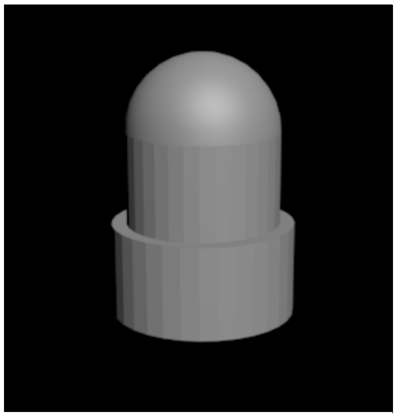
5.3 HARDWARE

This spread explains the Bistable Pin Design as well as the Layer Stackup of the board.

BISTABLE PIN DESIGN

The bistable pin mechanism uses ¼ inch magnetic pins with ball bearings to create a rounded top. The design is simple. A layer of steel is sandwiched between two PCBs that contain the actuators, which is then sandwiched between two players of polycarbonate. Holes are drilled through the whole thing where the pins will be. On top, there is a thin layer of metal with slightly smaller holes. This layer prevents the pins from shooting through the device. When the top solenoid is activated, the pin shoots up, then stays in place. Likewise, when the bottom solenoid is activated, the pin gets pulled down. The pins stay in place due to the magnetic attraction to the steel plate. If a pin that is up is manually pushed down it may move, but cannot be displaced enough for the other end of the magnet to attract to the steel, and thus will move back into place.

On top of the magnet we have a ball bearing to create the roundness of the ball. In the long terms the following custom shape would be created using cold press machining. This would let us better control the height of the ball.



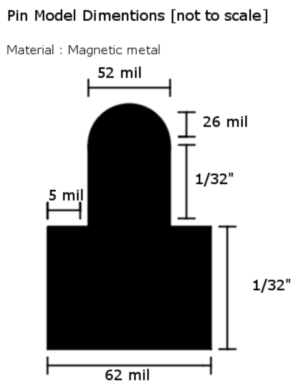
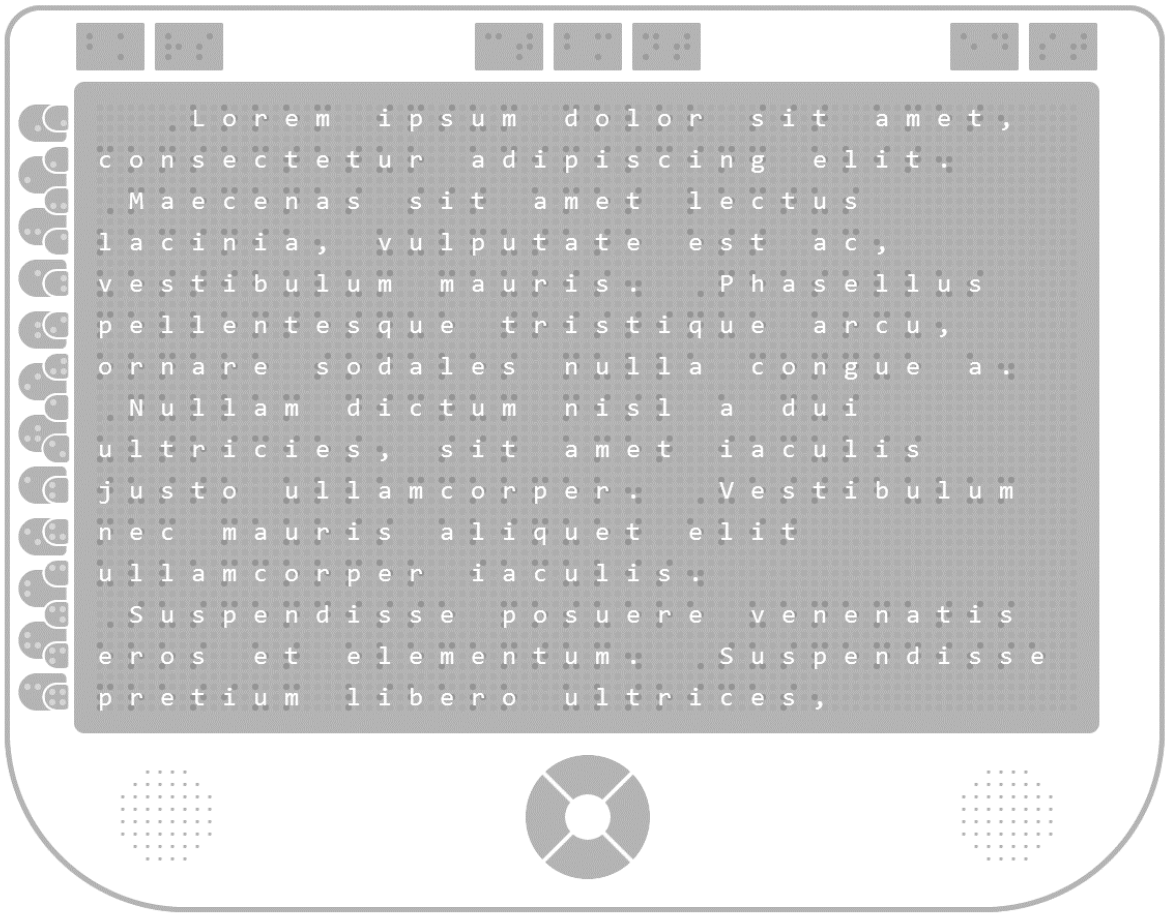

Pin Model Dimentions [not to scale]
Material : Magnetic metal
52 mil
26 mil
1/32"
5 mil
1/32"
62 mil

Figure 22 Steel Pin model

5.4 USABILITY



Lorem ipsum dolor sit amet,
consectetur adipiscing elit.
Maecenas sit amet lectus
lacinia, vulputate est ac,
vestibulum mauris. Phasellus
pellentesque tristique arcu,
ornare sodales nulla congue a.
Nullam dictum nisl a dui
ultricies, sit amet iaculis
justo ullamcorper. Vestibulum
nec mauris aliquet elit
ullamcorper iaculis.
Suspendisse posuere venenatis
eros et elementum. Suspendisse
pretium libero ultrices,

Figure 23 - The design of the potential device using the full grid of pins.

The main component of the device is the screen containing the array of pins.

There are 7 navigation buttons along the top, with 2-character braille codes to represent their functionality. In order, they are: Back (BK), Resume (RS), Categories (CT), Bookmarks (BM), Notes (NT), Edit/Delete (ED), and Settings (ST). We tried to keep the bulk of the navigation off the screen and done using the buttons to make the device more convenient and efficient to use.

Selection buttons along the side are designed to allow the device to display 6 or 8 pin braille. Depending on the setting, the selection buttons for the other cells size are disabled. Though most people read with 6-pin braille, and 6-pins would allow us to show more lines of text, 8 pins are often used for electronic devices to show if a line is selected, and to show a cursor for writing. Another benefit of 8-pin braille is shorter text through simplifications, and addition mathematic symbols. Since the full grid can change to use either, it is beneficial to set up the device to do both. The selection buttons make the device much more efficient to use than if they had to scroll through all of the links in order to select one.

There is also a circular button to scroll or flip pages. Flipping the page is an obvious function, but scrolling can also come in handy. For example, if a person is reading a page with math equations, they may get to a part where they want to know information on a previous line. If that line is on a separate page, then can scroll in order to get all the information they want on one page. Having up and down buttons also may be useful for games.

Ideally, the device would also have audio features. Audio can help when someone is initially learning to use the device, and can also be a convenient feature. For someone learning to read braille, they could use the selection buttons to get the device to read the line aloud, helping them learn braille more easily.

PROJECT EXECUTION OVERVIEW

Throughout the project, we faced many oppositions. The main component of our device is the PCB containing the solenoid actuators. When we submitted our board to different companies to get quotes, we were disappointed with companies saying they could not do it. The one company that did give us a quote was for $500,000. After this, we were forced to look into other technologies. After brainstorming many new ideas, we fused on two.

One idea we focused on was using bands under each row with bumps corresponding to all of the possible braille cells. Motors would rotate the bands into position for a given column and a pusher bar would push the bands up so that the bumps would push up with pins, setting a column at a time. This was a clunky solution, and would be slow and make the device large, so we continued to look into other solutions.

Another technology we looked into was shape-memory alloys to control the pins. While we prototyped a working pin, the device would have to be very thick, and it would be difficult to assemble. It also was expensive to get enough shape memory wire for the device. The alloy could also not reliably keep the pin up, so we would need a separate locking mechanism.

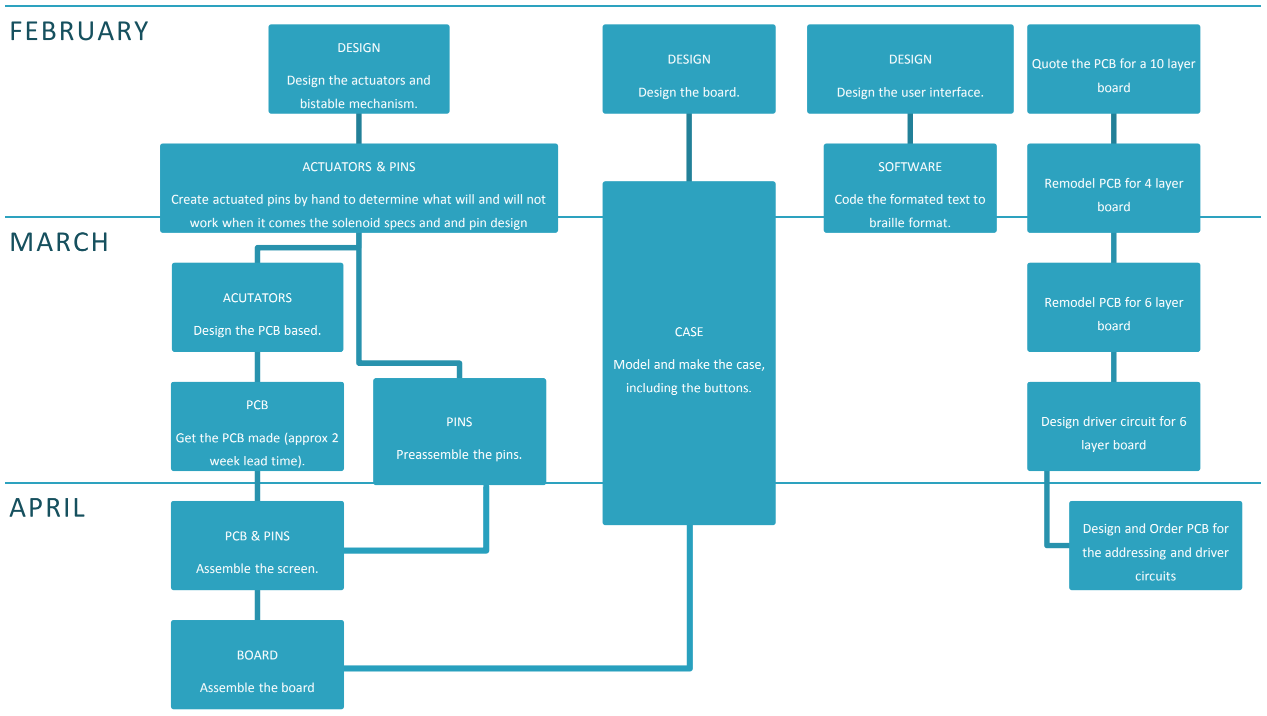
We were very unhappy with these two designs. After playing around with bistable mechanisms we discovered a pin design that would work very well with solenoids. With this new motivation, we looked back into the PCB. We figured out we could reduce the number of layers decrease the cost of the board significantly more than we expected. Sadly, even getting the board down to two for $5,000, we still could not afford the board. By the time we figured this out, we did not have time to generate funding, wait for the board to be made, and then assemble the device.

TIMELINE

Our timeline was greatly affected by setbacks, primarily by the PCB being out of budget.

NOVEMBER – JANURARY

Due to needing to change technologies, we had to drop all goals and focus all of our efforts into coming up with a new plan, since everything else would be meaningless without a board. We spent the rest of the fall semester brainstorming and trying to prove new concepts. We came up with a few ideas and started testing them, but we were unsatisfied by them. Around the end of February, we decided to go back to the PCB.



FEBRUARY
MARCH
APRIL
DESIGN Design the actuators and bistable mechanism.
ACTUATORS & PINS Create actuated pins by hand to determine what will and will not work when it comes the solenoid specs and and pin design
ACUTATORS Design the PCB based.
PCB Get the PCB made (approx 2 week lead time).
PINS Preassemble the pins.
PCB & PINS Assemble the screen.
BOARD Assemble the board
DESIGN Design the board.
CASE Model and make the case, including the buttons.
DESIGN Design the user interface.
SOFTWARE Code the formated text to braille format.
Quote the PCB for a 10 layer board
Remodel PCB for 4 layer board
Remodel PCB for 6 layer board
Design driver circuit for 6 layer board
Design and Order PCB for the addressing and driver circuits

EXPENDITURES

PART	$Price
Descripting/Justification	
MOTORS	$200
For pervious band design and moving line of actuators	
SHAPE MEMORY MATERIALS	$200
For pervious design using Shape memory alloy pins	
PRINTED CIRCUIT BOARD	$600
Proof of concept of driver and addressing circuitry as well as feedback	
MISC CIRCUIT/HARDWARE COMPONENTS	$200
Circuit components for the boards as well as for testing bistability	
LED ARRAYS	$100
Proof of concept of addressing circuit	

3D PRINTING	$500
3d printing the case as well as a resetting mechanism for the pins	
MAGNETS	$200
Magnets to demonstrate the bistable mechanism	
MOTORS	$200
For pervious band design and moving line of actuators	
TRAVEL	$2000
TOTAL	$4200

RECOMMENDATIONS & NEXT STEPS

COMPLETE THE BRAILLESHAPE

To complete the BrailleShape, the primary step is to get the PCB made. Since it is already designed, all that needs to be done is to get the funding required to prototype the device. We estimate it will take approximately $15,000 to create the functional prototype, mainly for the PCB.

Some more research needs to be done on the best way to assemble the device, since it will be slightly different than assembling the board without the actuators as we did for the prototype.

DEVELOP THE DEVICE

Once the fully functional prototype is made, the next step is to develop the final device. The key difficulty is manufacturing, due to the high tolerances needed. Manufacturing techniques will need to be developed to optimize the production of devices.

The device and interface design must be perfected using user testing.

INCREASING FUNCTIONALITY

NOTE TAKING

Current one-line braille displays function as note-takers. There are even special “keyboards” that have buttons for each finger instead of keys, meant for typing braille. By allowing this or a standard keyboard to interface with the device, we can give our device this note-taking ability.

GAMES

A very exciting possibility for the device is the ability to display some simple games, giving the blind access to games that would be impossible or extremely inconvenient for them to play without a full screen dynamic braille display. Many puzzle games would work well with the BrailleShape. For example, it is easy to see how the new hit puzzle game, 2048, which involves a grid of numbered blocks that the player slides to each side of the board to combine numbers, can be implemented using a full screen of braille. A game like this would be impossible for a blind person to play on a screen, but easy for them to play if they could feel the whole game board at once.

New games specifically created for this device would be especially exciting, and open up a wealth of possibilities. Adding this functionality does not require additional resources as it is all on the software side.

REFERENCES

Estimated Number of Adult Braille Readers in the United States. (n.d.). *American Foundation for the Blind: Programs and Policy Research*.

Facts and Figures on Americans with Vision Loss. (2008). *American Foundation for the Blind*.

APPENDIX A—SOFTWARE APPENDIX

PDF TO XML TRANSLATOR

The library that we used was the PDFMiner.py library with an additional open source library amagenerate.py. Information about the PDFMiner can be found at: http://www.unixuser.org/~euske/python/pdfminer/index.html, information about the amagenerate.py can be found at: http://dlbeer.co.nz/articles/pdf2html.html.

We only used the pdf2xml functionality of PDFMiner from this software. This converts from PDF to XML but the output code is not organized. Here is an example of what the output code looks like for a

Because of this we then feed the output of pdfminer to ama_generate.py. This library cleans up the XML code so that it groups lines and paragraphs together. Here is the output code after it has been sent to the ama_generate.py.

We further edited the ama_generate.py so that it does not store information such as the footers, page numbers, superscripts and subscripts etc because they are not always accurate and we do not need this information.

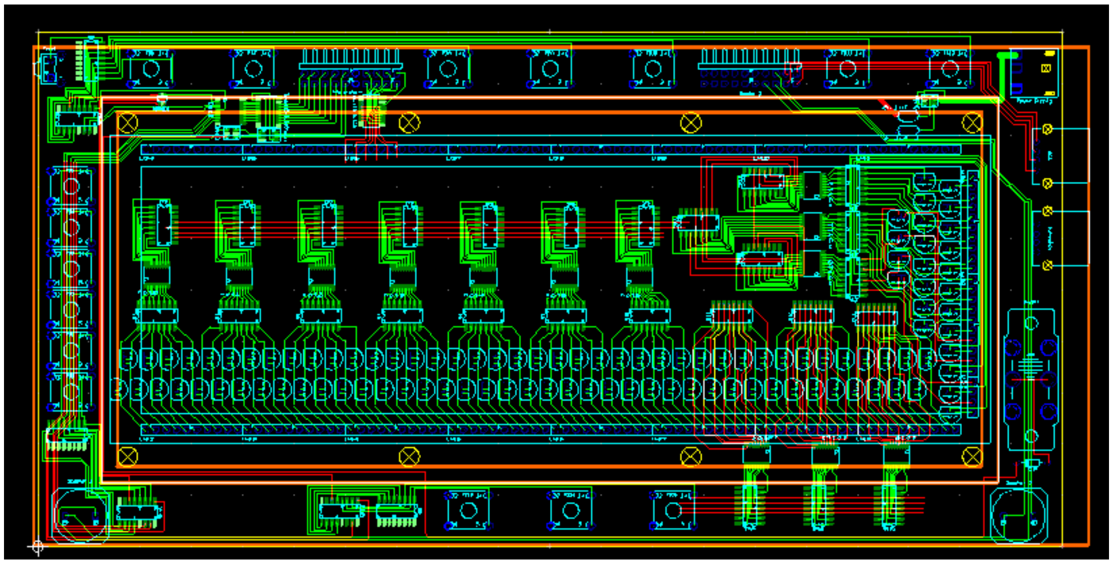
BRAILLE TRANSLATION KEY

Grade 2 braille was used in the translation. We used the rules that Duxbury systems published, the most popular braille embossing printer company in the US, uses. This translation is approved by the Braille Authority of North America. The table showing the braille exceptions and rules is shown below:

http://www.duxburysystems.com/images/bana_black.pdf

APPENDIX B—ASSEMBLY AND CONSTRUCTION APPENDIX

MAIN CIRCUIT PRINTED CIRCUIT BOARD



DECODER TRUTH TABLE

G1	G2A	G2B	A	B	C	0	1	2	3	4	5	6	7
0	X	X	**X**	**X**	**X**	1	1	1	1	1	1	1	1
X	1	1	**X**	**X**	**X**	1	1	1	1	1	1	1	1
X	1	0	**X**	**X**	**X**	1	1	1	1	1	1	1	1
X	0	1	**X**	**X**	**X**	1	1	1	1	1	1	1	1
1	0	0	**0**	**0**	**0**	0	1	1	1	1	1	1	1
1	0	0	**0**	**0**	**1**	1	0	1	1	1	1	1	1
1	0	0	**0**	**1**	**0**	1	1	0	1	1	1	1	1
1	0	0	**0**	**1**	**1**	1	1	1	0	1	1	1	1
1	0	0	**1**	**0**	**0**	1	1	1	1	0	1	1	1
1	0	0	**1**	**0**	**1**	1	1	1	1	1	0	1	1
1	0	0	**1**	**1**	**0**	1	1	1	1	1	1	0	1
1	0	0	**1**	**1**	**1**	1	1	1	1	1	1	1	0

SHIFT REGISTER TRUTH TABLE AND TIMING DIAGRAM

Timing Diagram

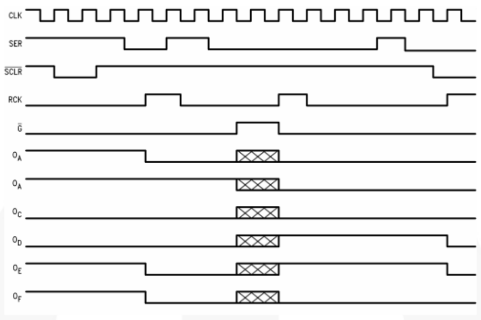
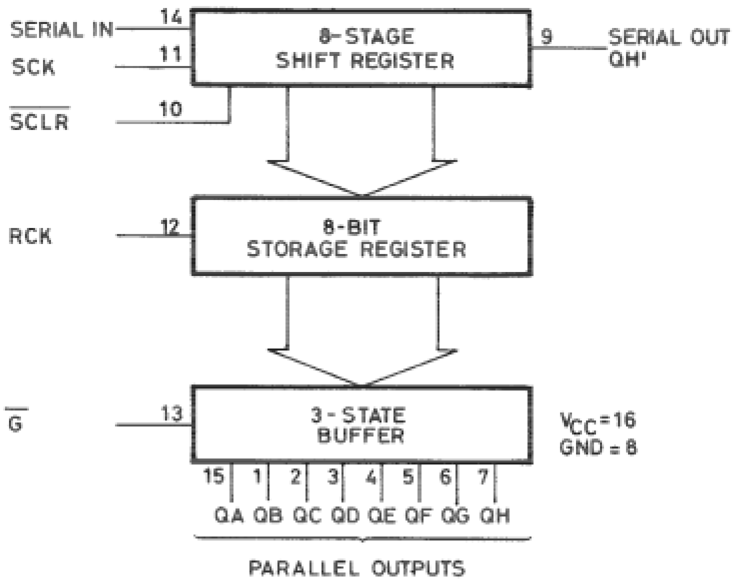


Figure 3. Timing Diagram

Note:

6. XXX Implies that the output is in 3-state mode.



http://pdf.datasheetcatalog.com/datasheet/SGSThomsonMicroelectronics/mXtuvqt.pdf

APPENDIX C—INTERVIEWS WITH POTENTIAL CUSTOMERS

Question Book

RECIPIENT 1

1: What is your age?

21 to 64

Q2: What is your average annual household income?

$25,000-$49,999

Q3: How much time do you spend reading braille each day?

less than 1 hour

Q4: How often do you read for fun?

everyday

Q5: What is your preferred reading method?

braille book

Other (please specify)audio

Q6: Where do you go to purchase reading material?

Bookshare and the BARD NLS website

Q7: How difficult is it for you to locate reading material?

somewhat difficult

Q8: Would you be interested in a full page, refreshable, and portable braille display that reads PDF documents and books? (The display is a reading device only, but it offers the most up-to-date books and converts online documents to braille.)

I would be interested in purchasing it, depending on the price.

Q9: How much would you be willing to pay for this portable braille e-reader?

$500-$1000

RECIPIENT 2

: What is your age?

21 to 64

Q2: What is your average annual household income?

$0-$24,999

Q3: How much time do you spend reading braille each day?

3-6 hours

Q4: How often do you read for fun?

everyday

Q5: What is your preferred reading method?

braille display

Q6: Where do you go to purchase reading material?

www.braillesuperstore.com

Q7: How difficult is it for you to locate reading material?

There are a lot of PDFs that are not accessible, so if it doesn't work with my iPhone, I don't usually read it unless it is through audible or the Victor Reader

Q8: Would you be interested in a full page, refreshable, and portable braille display that reads PDF documents and books? (The display is a reading device only, but it offers the most up-to-date books and converts online documents to braille.)

I would be interested in purchasing it, depending on the price.

Q9: How much would you be willing to pay for this portable braille e-reader?

$0-$500

Q10: At what email address would you like to be contacted?

dovin1128@live.missouristate.edu

What is your age?

21 to 64

Q2: What is your average annual household income?

$100,000 and up

Q3: How much time do you spend reading braille each day?

more than 6 hours

Q4: How often do you read for fun?

2-4 times a week

Q5: What is your preferred reading method?

braille book

Q6: Where do you go to purchase reading material?

I very rarely purchase reading material. I receive braille books from our local National Library Service lending library. Any material that is not available from the library I have transcribed into braille by our local rehab agency for the blind, but this is very rare.

Q7: How difficult is it for you to locate reading material?

The lending library has books that I enjoy, but wish they had more best sellers and current fiction and nonfiction.

Q8: Would you be interested in a full page, refreshable, and portable braille display that reads PDF documents and books? (The display is a reading device only, but it offers the most up-to-date books and converts online documents to braille.)

I would be interested in purchasing it, depending on the price.

Q9: How much would you be willing to pay for this portable braille e-reader?

$500-$1000