IBM1403-N1 Printer Interface
Project Report

Architectural Design, Detailed Design, Test Readiness, Interim, Final Version

Watson Capstone Project <WCP22>
Client: Center for Technology & Innovation (CT&I)
Sponsor: IEEE Binghamton Chapter

21st May, 2014

Revision: -

Submitted by:
Ryan Kulesza, Lead, EE
Peter Haviland, CoE
Mohammad Imran, CoE
Yu Chao Wang, EE

Faculty Advisor: Professor Maynard
External Advisor: Don Manning
Client Advisor: Susan Sherwood
Program Manager: Professor Jack Maynard
IEEE Chair: William Tracz

Thomas J. Watson School of Engineering and Applied Science
Binghamton University
Binghamton, NY

Abstract

The IBM1403-N1 printer was introduced in October 1959. Being able to print 1100 lines per minute, the IBM1403-N1 paved the way for a new era of high-speed and high-volume printing which was only surpassed by laser printers in the 1970s. It was a revolutionary printer in that it was the first of its kind to deploy what is known as on-the-fly printing. With the characters embedded within a fast moving chain, hammers pressed against the paper when right characters were aligned. Key to this kind of printing was to know when to activate which hammer.

In our project, attention was mainly focused on the driver cards within the printer. These drivers; hammer driver and carriage driver, are responsible for hammer firing and forms movement. Using modern day technology, a reconstruction of these drivers was carried out under the guided supervision of CT&I experts. Also essential to printing are feedback signals from the printer itself telling which character is in which position so that the right hammers get fired. Our project utilized a microcontroller which will be programmed to make logical decisions by processing these feedback signals.

This report discusses our project goal and our analysis and reconstruction of the driver circuits which has reached its completion and is ready for full scale fabrication. Also discussed in this report is a detailed analysis of print mechanism and the corresponding algorithm that will be implemented on the microcontroller.

Table of Contents

List of Figures

List of Tables

1 Scope

1.1 Identification

This is the interim project report for Watson Capstone Project WCP22, Interface for IBM1403-N1Printer of 2013-2014.

1.2 System Overview

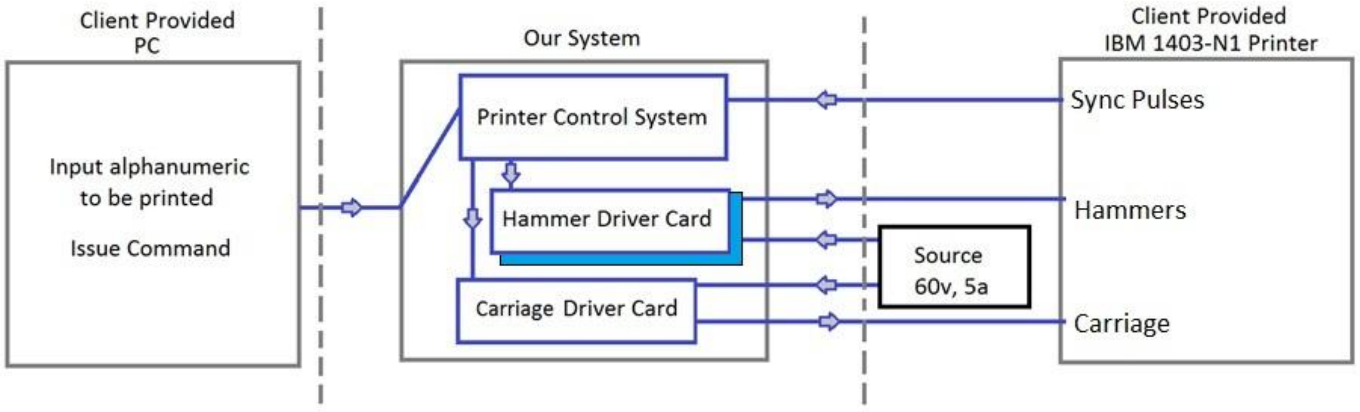


Figure 1 Contextual Flow Chart

In this project IBM's 1403-N1 Printer will be used to print data coming from a modern day PC. The project utilizes a Printer Robot Driver, compatible with the IBM1403-N1, for limited print capabilities. The robot driver will drive 12 printer hammers and will utilize timing and feedback to synchronize with the printer.

The IBM1403-N1 printer, developed in 1959, has not been in function for a number of years and hence this project is an attempt to revive the printer with a modern interface. Using readily available, modern components, hammer drivers will be redesigned and produced utilizing the existing IBM schematics and advise from CT&I technical experts.

The project is sponsored by IEEE's Binghamton Chapter, and will be used by CT&I for demonstration purposes at TechWorks located at 321 Water Street, Binghamton, NY.

1.3 Document Overview

This document will define the design features of the IBM1403-N1 Printer Interface project. The document has been created by WCP22, and it applies to the Architectural Design, Detailed Design, and Interim design phases. There are no security or privacy considerations or restrictions pertaining to this document.

2 Referenced Documents

The following documents of the exact issue shown form a part of this document to the extent specified herein.

No.	Title	Date created	Latest Revision
1.	IBM 1403 Printer Models N1 and 3 Maintenance Manual	December 1971	-
2.	IBM 1403 Printer Component Description	November 1964	-
3.	Project Requirement Specification (PRS) for IBM1403-N1 Printer Interface	November 2013	May 2, 2014
4.	Project Development Plan (PDP) for IBM1403-N1 Printer Interface	December 2013	May 2, 2014
5.	Project Test Plan and Procedures (PTPP) for IBM1403-N1 Printer Interface	March 2014	May 2, 2014
6.	WCP Software Design Description	May 2014	May 21, 2014
7.	WCP Software Interface Module (code)	May 2014	May 2, 2014

3 Project Overview

3.1 Project Definition

The IBM1403-N1 Printer Interface project as defined in the Project Requirements Specification (PRS) document will provide hardware and software necessary to operate 12 hammers and the carriage control of the IBM1403-N1 Printer. The input to the system will be meaningful text and line spacing arguments. The output will be the printing of the text and execution of line spacing using the IBM1403-N1 Printer. Delivered hardware will include an Amplifier PCB, for dissemination and collection of signals and the routing of these signals to the microcontroller, Hammer Driver PCBs (of which there will be two), or Carriage Driver PCB. The Hammer Driver PCBs will communicate timed hammer pulses in order to print meaningful text, and the Carriage Driver PCB will actuate line spacing depending on the line spacing argument provided, or if necessary- due to input type- when all 12 print positions have be printed.

3.2 System Design

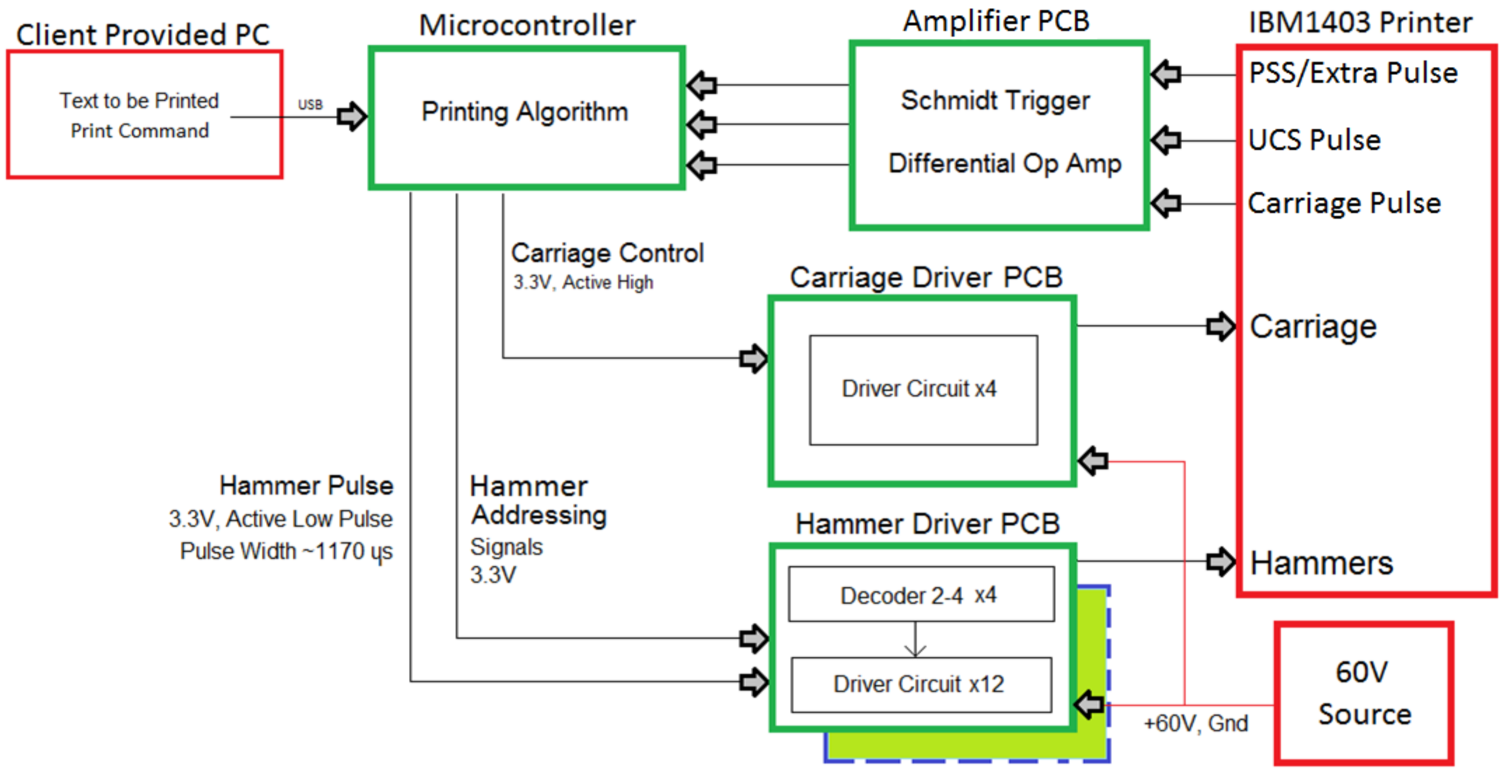


Figure 2 Simplified System Block Diagram

The above flow chart describes the major components of our design, as well as, the interface between our design and client provided systems.

Client provided systems appear in red, including a PC, a 60V 5A source, and an IBM1403-N1 Printer. The PC is required for acquisition of text data to be printed and the issuance of the print command. The 60V 5A source is required supply power to twelve hammer driver circuits and four carriage driver circuits. A Single Pull Signal Throw (SPST) switch will be provided by WCP22 in order to turn on and off the 60V 5A source where it interfaces with the Amplifier PCB. This switch eliminates the possibility of unintended hammer strikes or carriage movement due to unknown states in the microcontroller prior to initialization.

The IBM1403-N1 Printer is required to perform the printing function utilizing internal hammer solenoids which will propel the print hammers toward the paper form at specific text locations on the current line. The IBM1403-N1 printer is also required to perform the line spacing function utilizing internal carriage solenoids which will control four hydraulic valves, one valve to start a line space, one valve to stop line spacing, one valve to start line skipping, and one valve to stop line skipping. The IBM1403-N1 is also required to provide feedback to the WCP22 IBM1403-N1 Printer Interface.

There are three types of feedback that will allow our system to synchronize with the IBM1403-N1 Printer. The first type of feedback is a carriage emitter pulse ("E1"), as the carriage position changes the gear connected to the carriage hydraulics generates a signal that will be used to indicate the number of spaces the carriage has performed. The second type of feedback is the character train emitter pulse. As the characters revolve on the cartridge a gear connected to the cartridge will generate a signal that will be used to indicate when a new character has aligned with printer hammer one, two, or three. This emitter signal will be known as the PSS, or Print Sub-Scan, pulse. The third type of feedback is another emitter pulse that is generated by a second idler gear connected to the printer cartridge. This pulse is coincident with the PSS "extra" pulse when the characters on the cartridge have returned to their original position. This emitter signal will be known as the UCS pulse, or Universal Character Set pulse.

The Hammer Driver PCB will be delivered by WCP22. The function of this printed circuit board is to provide a 60V 5A pulse of approximately 1170 microseconds in width to any one of the twelve IBM1403-N1 internal hammer solenoids. In order to drive twelve hammers, two identical Hammer Driver PCBs will be created. Each of the Hammer Driver PCBs will contain six driver circuits, as well as, two 2-4 decoders. The decoder will be supplied with the hammer actuation pulse originating from the microcontroller, as well as, two addressing signals which also originate from the microcontroller. The addressing signals will indicate the circuit path pertaining to a specific hammer driver circuit. The hammer actuation pulse is active low due to the characteristics of the selected decoder. The decoder outputs a logic high signal to the addressed path when the inverted-enable input receives a logic low signal.

The Carriage Driver PCB will be delivered by WCP22. The function of this printed circuit board is to provide a low current signal of less than one amp to any one of the four IBM1403-N1 internal carriage solenoids. The carriage driver circuits will be supplied with four active high signals. These carriage control signals will originate from the microcontroller and will be used to control the IBM1403-N1 internal space start solenoid, space stop solenoid, skip start solenoid and skip stop solenoid. Either the space start or space stop signal will be active at all times to control the IBM1403-N1 hydraulic space valves. Likewise, either the skip start or skip stop signal will be active at all times to control the IBM1403-N1 hydraulic skip valves.

The Amplifier PCB will be delivered by WCP22. The function of this printed circuit board is to condition the three feedback signals from the IBM1403-N1 printer, to provide regulated 3.3V power to the microcontroller, and to disseminate incoming and outgoing logic signals to the Hammer Driver PCB, the Carriage Driver PCB, and the Microcontroller. All three feedback signals will be conditioned for use in the microcontroller by means of a differential operational amplifier stage and a one-sided Schmidt trigger, the output of which will be provided to the microcontroller. The differential stage is necessary to collect the signal data from each of the IBM1403-N1 internal emitters without the presence of a common ground. The one sided Schmidt trigger will generate a logic pulse when the output of the differential stage exceeds the Schmidt trigger threshold. Additionally the PSS pulse and the UCS pulse generated by the Schmidt trigger will be applied to an AND gate, since the "home" condition is satisfied when both the UCS pulse and the PSS pulse is high. The output of this AND gate will be provided to the microcontroller, in addition to the PSS pulse, and the carriage emitter pulse to allow for synchronization between the microcontroller algorithm and the IBM1403-N1 printer.

To aid in synchronization a Phase Locked Loop circuit was designed to accept as an input the PSS pulse and perform a divide-by-three function on this pulse effectively creating a clock divider. This was found to be unnecessary on this model of printer, but has been included to provide this divide-by-three capability necessary for controlling later models of printers. In later models of IBM printers, this clock-divided signal would have also been provided to the microcontroller to indicate the timing of three sub-scans. The Phase Locked Loop circuit will be capable of adjusting the timing of the sub-scan pulses as the printer hammers may change the velocity of the cartridge rotation as they make contact with the paper and characters.

The +5V -5V 1A supply will be delivered by WCP22. The purpose of this supply is to provide power to the 3.3V regulator, and all integrated circuits. The supply provides both +5V and -5V to accommodate the rail voltages of our operational amplifiers, without the introduction of an offset between the 60V 5A supply ground and the +5V – 5V 1A supply ground.

The microcontroller of the system is essentially the hub where signals from the PC and printer are received and processed. Its outputs are based on logical decisions which will be captured in our code and outputted to the driver cards.

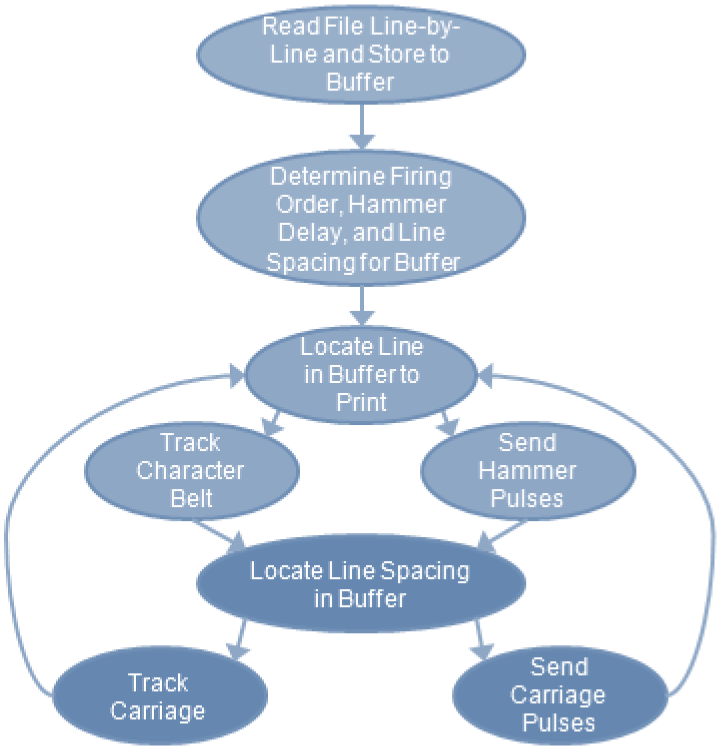

Read File Line-by-Line and Store to Buffer
Determine Firing Order, Hammer Delay, and Line Spacing for Buffer
Locate Line in Buffer to Print
Track Character Belt
Send Hammer Pulses
Locate Line Spacing in Buffer
Track Carriage
Send Carriage Pulses

Figure 3 Printing Algorithm Flow Chart

The above flow chart demonstrates the basic flow of operation for the printer interface.

Once the user has finished entering the print job in the form of a text file, the data is sent to the microcontroller via USB cable. The microcontroller will accept the data line-by-line and store it to a buffer before printing occurs. This buffer will store the entire page. When the page is completely buffered in the microcontroller, an algorithm will precisely determine which hammer needs to be fired and at which specific times utilizing feedback signals, discussed in detail in section 4, from the printer. The algorithm will also determine the individual line spacing for the page along with the printer hammer delays. To print the page, the microcontroller will move through the buffer line-by-line. To print a line of characters, the microcontroller will track the print cartridge while simultaneously sending hammer pulses as determined by the print algorithm. When a blank line is met, the microcontroller will utilize carriage signals as described in section 4 to move the page based on the determined line spacing for that individual line. The microcontroller will continue to move through the buffer until the entire page has been printed.

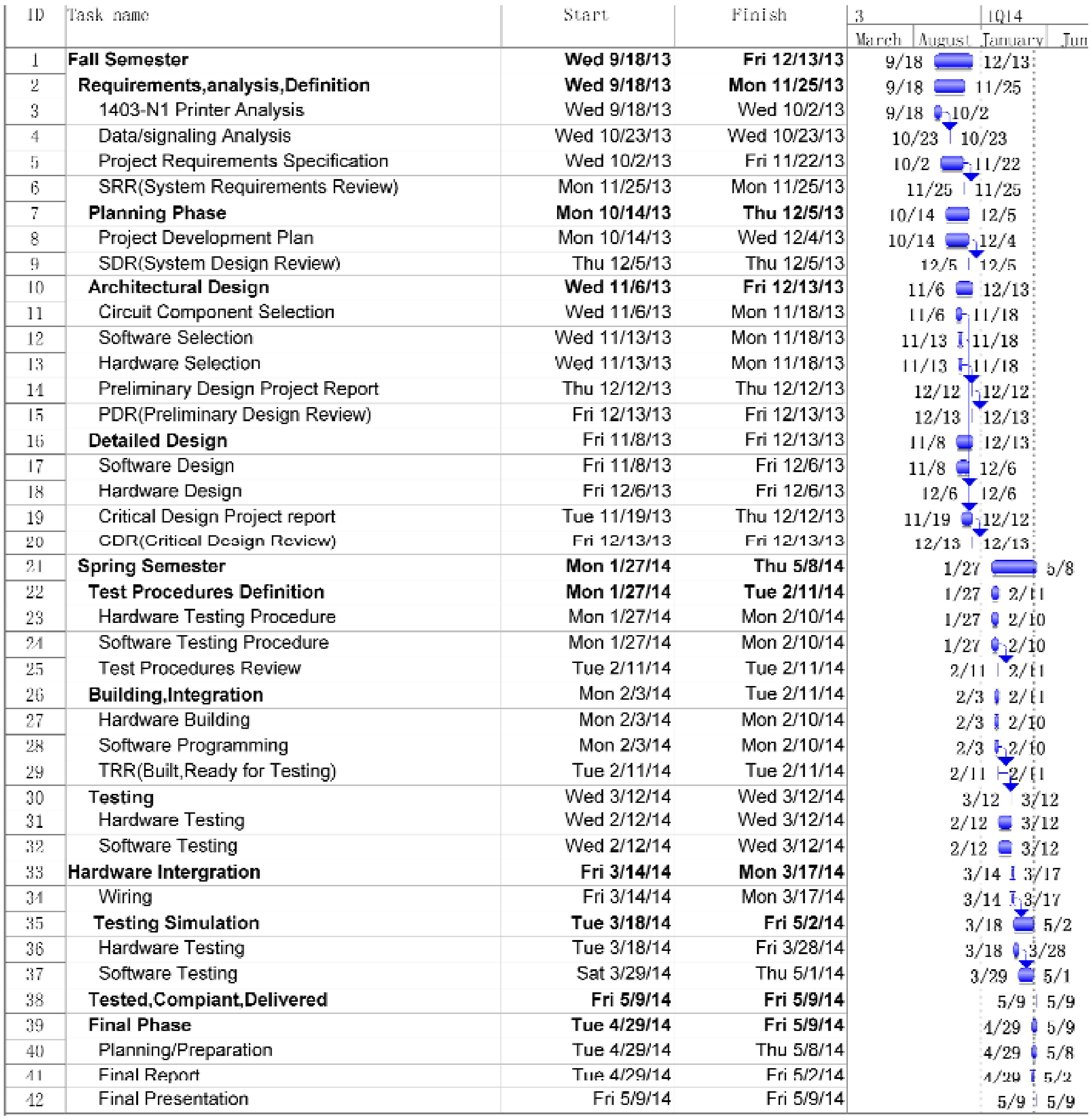
MPIDE is used to program the microcontroller in the C programming language. This program controls the printing algorithm, the receiving of the text file from the PC, and the interaction with the IBM1403-N1 printer. MPIDE is free software provided by Digilent and is cross-platform.

The Processing IDE is used to run the program required to send the text file from the PC to the microcontroller. This program reads the text file and sends the data to the microcontroller through the USB wire. Processing IDE is open source and cross-platform software that is programmed using the open source programming language called Processing.

The Arduino IDE is the third program utilized for simulation purposes. This simulation allows the system to function as if the printer was in operation, and allows the system to be verifiable through an oscilloscope and data logs implemented through and SD card.

The ChipKIT Pro Mx4 contains 74 I/O pins which will allow for future operable expansion to all 132 hammers. The implementation of the 2x4 decoders will allow for the control of all 132 hammers. The ChipKIT Pro Mx4 microcontroller has 512 KB of on board memory and 32 KB of SRAM memory. If the file to be printed is larger than the workable SRAM, then expandable memory will be implemented with the microcontroller to store the buffer for the file.

3.3 Project Schedule



ID	Task name	Start	Finish
1	**Fall Semester**	**Wed 9/18/13**	**Fri 12/13/13**
2	**Requirements,analysis,Definition**	**Wed 9/18/13**	**Mon 11/25/13**
3	1403-N1 Printer Analysis	Wed 9/18/13	Wed 10/2/13
4	Data/signaling Analysis	Wed 10/23/13	Wed 10/23/13
5	Project Requirements Specification	Wed 10/2/13	Fri 11/22/13
6	SRR(System Requirements Review)	Mon 11/25/13	Mon 11/25/13
7	**Planning Phase**	**Mon 10/14/13**	**Thu 12/5/13**
8	Project Development Plan	Mon 10/14/13	Wed 12/4/13
9	SDR(System Design Review)	Thu 12/5/13	Thu 12/5/13
10	**Architectural Design**	**Wed 11/6/13**	**Fri 12/13/13**
11	Circuit Component Selection	Wed 11/6/13	Mon 11/18/13
12	Software Selection	Wed 11/13/13	Mon 11/18/13
13	Hardware Selection	Wed 11/13/13	Mon 11/18/13
14	Preliminary Design Project Report	Thu 12/12/13	Thu 12/12/13
15	PDR(Preliminary Design Review)	Fri 12/13/13	Fri 12/13/13
16	**Detailed Design**	Fri 11/8/13	Fri 12/13/13
17	Software Design	Fri 11/8/13	Fri 12/6/13
18	Hardware Design	Fri 12/6/13	Fri 12/6/13
19	Critical Design Project report	Tue 11/19/13	Thu 12/12/13
20	CDR(Critical Design Review)	Fri 12/13/13	Fri 12/13/13
21	**Spring Semester**	**Mon 1/27/14**	**Thu 5/8/14**
22	**Test Procedures Definition**	**Mon 1/27/14**	**Tue 2/11/14**
23	Hardware Testing Procedure	Mon 1/27/14	Mon 2/10/14
24	Software Testing Procedure	Mon 1/27/14	Mon 2/10/14
25	Test Procedures Review	Tue 2/11/14	Tue 2/11/14
26	**Building,Integration**	Mon 2/3/14	Tue 2/11/14
27	Hardware Building	Mon 2/3/14	Mon 2/10/14
28	Software Programming	Mon 2/3/14	Mon 2/10/14
29	TRR(Built,Ready for Testing)	Tue 2/11/14	Tue 2/11/14
30	**Testing**	Wed 3/12/14	Wed 3/12/14
31	Hardware Testing	Wed 2/12/14	Wed 3/12/14
32	Software Testing	Wed 2/12/14	Wed 3/12/14
33	**Hardware Intergration**	**Fri 3/14/14**	**Mon 3/17/14**
34	Wiring	Fri 3/14/14	Mon 3/17/14
35	**Testing Simulation**	**Tue 3/18/14**	**Fri 5/2/14**
36	Hardware Testing	Tue 3/18/14	Fri 3/28/14
37	Software Testing	Sat 3/29/14	Thu 5/1/14
38	**Tested,Compiant,Delivered**	**Fri 5/9/14**	**Fri 5/9/14**
39	**Final Phase**	**Tue 4/29/14**	**Fri 5/9/14**
40	Planning/Preparation	Tue 4/29/14	Thu 5/8/14
41	Final Report	Tue 4/29/14	Fri 5/2/14
42	Final Presentation	Fri 5/9/14	Fri 5/9/14

Table 1 Project Schedule

3.4 Project Finances

Item	Estimate	Estimate to Completion	Actual
Printed Circuit Boards	175	0	325
Electrical Components	175	0	195
Microcontroller/MicroSD	80	0	80
Service Box	0	0	67
Connectors	20	0	40
Totals	450	0	707
Available Funds			1500

Table 2 Top-Level Financial Summary

The above table discusses the financial plans for the system with an estimated cost at completion.

The manageable budget for this system is $1600 dollars. Additional expenditures have been made to provide spare electrical components, four extra printed circuit boards, a clear plastic service box, and additional connectors.

3.5 Conclusion

This section shall summarize the current status of the project, noting the significant challenges, accomplishments, lessons learned, and if applicable, the work yet to be done.

WCP22's design approach was to create the driver circuits necessary to actuate hammers and the printer carriage. Another facet of our approach was to design a feedback system that would allow for synchronization with the state of the IBM1403-N1 character belt. Also, our design required a microcontroller and algorithm accept the print job, calculate the timing of events, and actuate the printing procedure.

The current status of the project is that it is working in accordance with the requirements of the project. WCP22 invited our external advisors observe our final run of the system based on a sample input that they provided us. We were able to verify the correctness of our output from reading the log data and oscilloscope reading vs the calculation that our advisors had. The final test run of the system demonstrated that our algorithm was working correctly on Wednesday April 30th. A separate run was carried out earlier, on Monday March 31st, which demonstrated correct functionality of the driver circuits. All driver circuits were individually selectable and capable of producing output pulse of any desired pulse width.

The acquisition of the feedback signals posed some difficulty when the direct outputs of the LM741 operational amplifiers were found to be distorted. This issue led to the generation of multiple solutions. One solution tried by WCP22 was to add smoothing capacitors to the supply rails of the LM741 operational amplifiers, this removed a small amount of noise however we were still not satisfied that the microcontroller would be able to correctly receive the signals. These capacitors were left in place to provide smoothing of the power supply when the Amplifier PCB was connected to the project service box by 14 feet of Cat 6 cable. This length of cable will allow the Amplifier PCB to be placed inside the printer while the rest of the WCP22 system will be located in a near-by rack. We then tried adding a capacitor between ground and the output of the LM741 operational amplifiers to solve the issue of our distorted inputs. This had very little effect on the noise level so the capacitors were removed to prevent the slowing of our output signals unnecessarily. Finally, upon observing that the direct output of the carriage emitter pulse and the output of the AND gate were not distorted, WCP22 tried routing the distorted signals through the AND gate from which clean digital signals had been observed. This solved the issue of our distorted inputs. Upon testing the newly stabilized outputs we found that the outputs were active-low instead of active-high as we had expected. This presented our team with the challenge of how to determine when the extra pulse and the UCS pulse were coincident. The hardware solution generated was to replace the AND gate with a NAND gate. The software solution was to check the UCS pulse when an extra pulse was detected. If the UCS pulse was low, our software would detect the train home condition. The software solution solved this issue and we were then able to synchronize our system with the simulated output of the IBM1403-N1 printer provided by an Arduino microcontroller.

Significant challenges were faced in changing our algorithm that used interrupts each time a PSS pulse was received vs our earlier version that used polling. With constant help from our advisors we were able to figure out a systematic approach to change our code that could use interrupts. Two important milestones were laid down to us by our external advisors for the final demonstration (order in terms of priority) (a). show that correct hammers are being fired to print the required text (b). timing of hammer firing such that two hammers can be fired after a gap of 5 us. We accomplished the first milestone and that was enough for our system acceptance and hence the work to be done is timing of hammer firing. As discussed by our external advisors this will be the work of the next WCP team.

4 Technical Details

4.1 Design Decisions

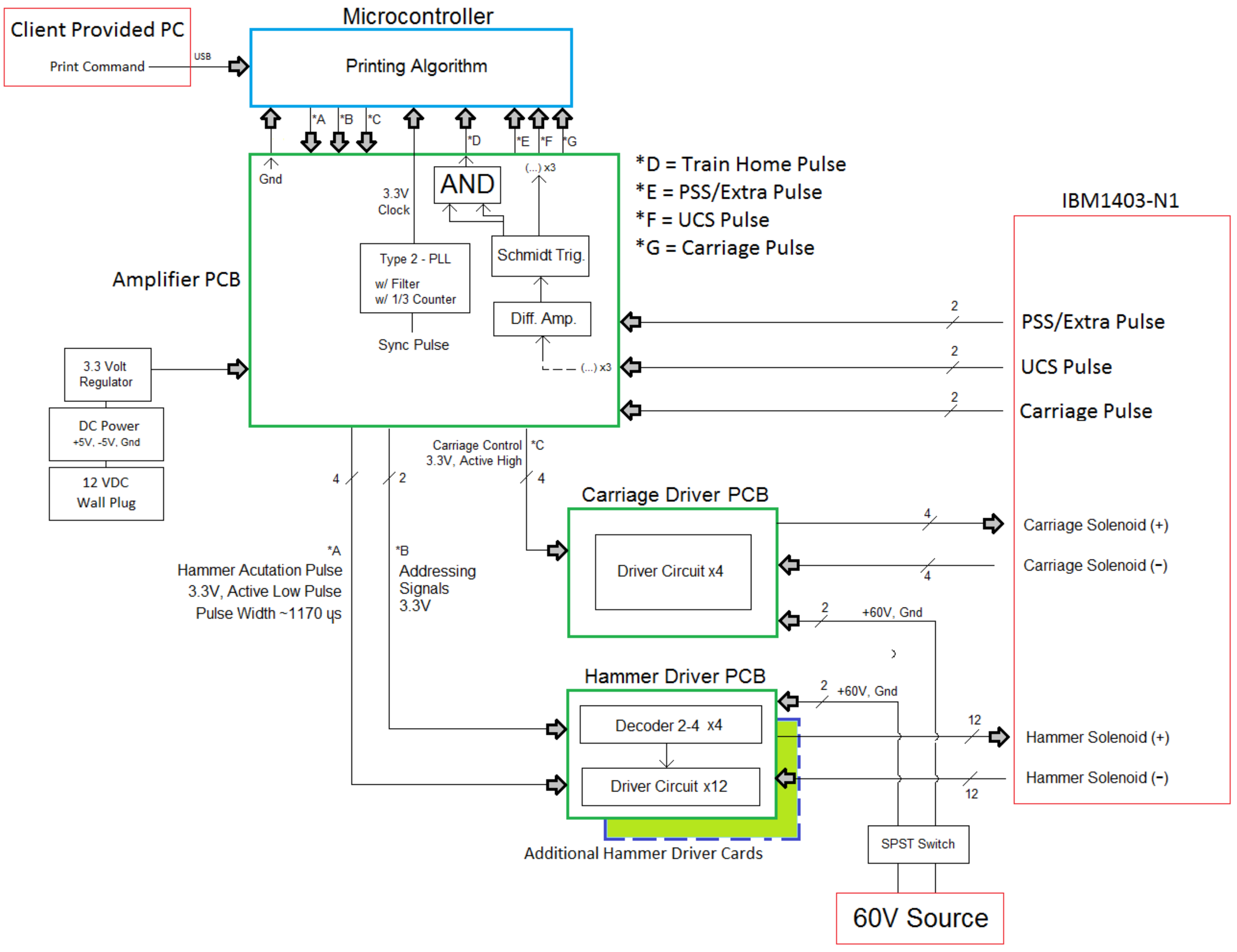


Figure 4 Detailed System Flow Chart

As shown in the Detailed System Flow Chart, figure 4, there are three inputs to the system; data to be printed delivered through a USB from the PC (utilizing a keyboard), feedback signals from the printer and a 60V 5A power supply for the hammer and carriage drivers. There are two major outputs; signals to be sent from the microcontroller to the hammer driver producing a 60V 5A pulse to control hammer solenoids, and signal from the microcontroller to the carriage driver producing a 60V <1A signal to control carriage solenoids. Printing begins by user typing in the desired data that needs to be printed using a modern keyboard and PC. This data is then sent to the microcontroller via USB. The microcontroller then sends out signals to the hammer driver and printer at correct timings to fire each hammer, and to control the carriage for line spacing.

The input from the keyboard will be buffered and printing will start immediately upon completion of PSS timing calculation. At this point the job of the keyboard is done and printing can begin. To determine correct characters are printed (hammered) onto the paper, correct logic will be implemented to determine rotational speed of the printer cartridge and pulse timings of the hammer driver. This logic will be carefully captured in the code algorithm (C language). If implemented correctly high performance can be achieved as the clock Rate of the microcontroller is quite high for the type of signals we are dealing with in this project.

The user will input data using a text file editor on a windows operating system.

WCP22 will utilize advice of CT&I technical experts in order to produce a prototype with a level of safety appropriate for demonstration at the TechWorks facility.

WCP22 will produce printed circuit boards, via a third party (Advanced Circuits), with markings that may be traced back to the circuit schematic for ease in population of the boards.

WCP22 has designed the IBM1403-N1 Printer Interface with provisions included for expandability. The hardware has been designed using decoders in order to reduce the necessary number of I/O pins so the IBM1403-N1 Printer Interface may be expanded to control all 132 print hammers using the same microprocessor with the addition of twenty Hammer Driver PCBs. If it is desired to activate multiple hammer-driver circuits that are currently connected to the same decoder, the decoder may be removed and the hammer drivers may be activated when an active-high pulse is applied to their inputs, rather than an active-low pulse as is currently used to control the output enable pin of the decoder.

		Arduino Due			chipKIT Pro MX4		
Criterion	Weight	Value	Score	Result	Value	Score	Result
# of I/O Pins	.2	54	2	0.4	74	5	1
Pin Voltage	.1	3.3V	4	0.4	3.3V	4	0.4
I/O Total Current	0.05	130 mA	4	0.2	200 mA	5	0.25
Input Voltage	0.05	7-12V	5	0.25	3.6-12V	5	0.25
Clock Speed	0.2	84 MHz	5	1	80 MHz	5	1
Flash Memory	0.2	512 kB	5	1	512 kB	5	1
SRAM	0.05	96 kB	5	0.25	32 kB	3	0.15
EEPROM	0.05	N/A	0	0	N/A	0	0
Cost	0.1	$58	3	0.3	$80	2	0.2
Sum	1			**3.8**			**4.25**

Table 3 Microcontroller Trade Study

The Microcontroller selected best fit for the system was the chipKIT Pro MX4. This microcontroller, which is provided by Digilent, is compatible with the majority of Arduino code while offering the large I/O required for the system. Although there is less available SRAM when compared to the Arduino Due, this will be combated with expandable memory to hold the potentially large files. The Pro MX4 is also more expensive than the Arduino; however the sheer number of I/O pins supplied by the microcontroller is one of the most important factors for future development of the system.

4.2 Sub-System Design

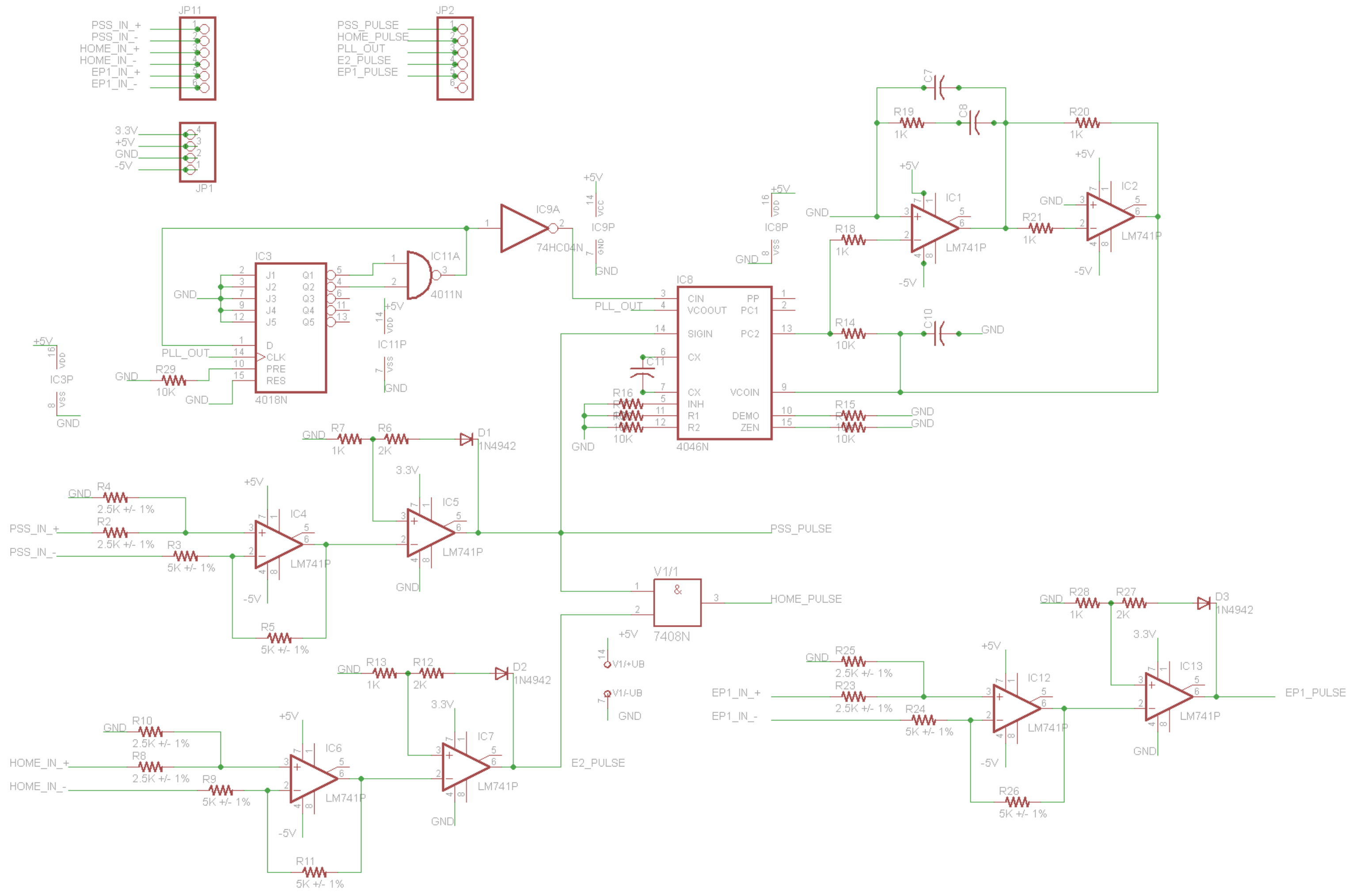


Figure 5 The Amplifier PCB

The Amplifier PCB is responsible for the dissemination of signals to the microcontroller, the Hammer Driver PCBs and the Carriage Driver PCB. The Amplifier PCB will also condition the emitter pulses received from the IBM1403-N1 Printer for use in the microcontroller. Our project will utilize a +5V -5V power supply for the operation of integrated circuits. An additional board has been constructed to provide a 3.3V regulator to supply 3.3V rail limits for the Amplifier PCB's Schmidt triggers.

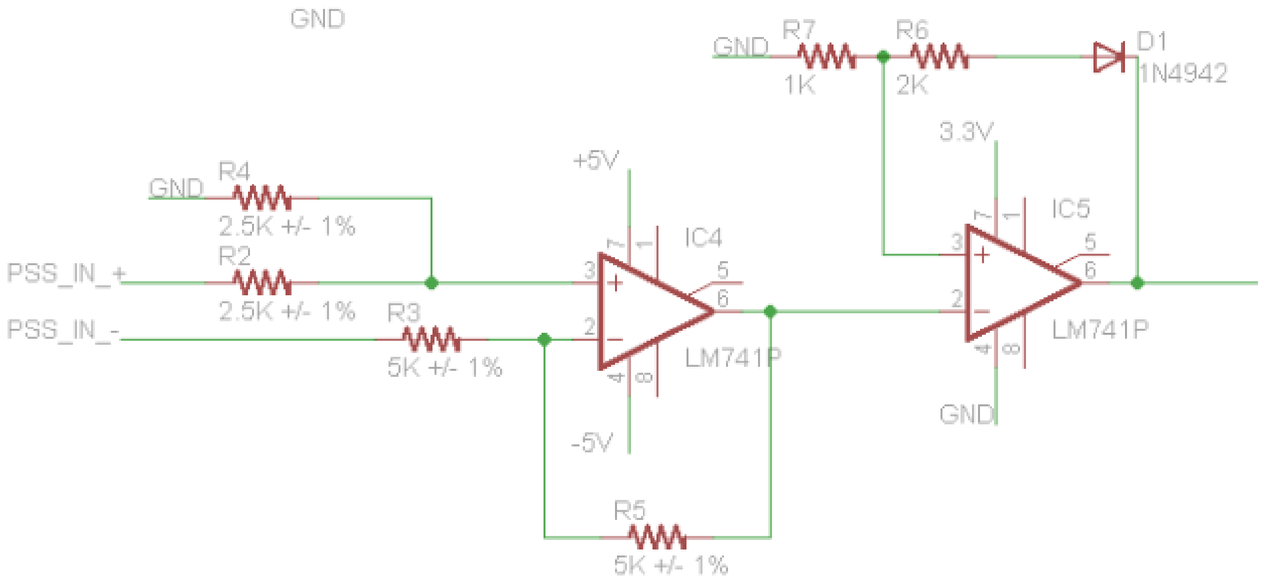


Figure 6 Emitter Input Conversion

The incoming emitter pulses are generated by the revolution of the cartridge gear in the case of the PSS pulse, an additional cartridge gear in the case of the UCS pulse, and the carriage gear in the case of the carriage pulse “E1.” These signals must be converted from periodic analog signals to periodic digital signals that can be used by the microcontroller. In order to accomplish this, the signal is first passed through a differential operational amplifier. This stage collects the potential difference created by the rotating gear with a net gain of 1. This stage is necessary because there is to common ground between the emitter and the IBM1403-N1 Printer Interface. The signal is then passed through a Schmidt trigger. The Schmidt trigger has a diode in the feedback loop to prevent negative feedback from entering the non-inverting input of the operational amplifier. This creates a one-sided Schmidt trigger necessary to provide a signal pulse which will represent the signal digitally. The output of the Schmidt trigger will then be passed to the microcontroller to represent the PSS pulse and the carriage pulse, “E1”. The final gain of the differential stage has been configured to be approximately 21. The final hysteresis point is configured to be approximately 0.66 Volts.

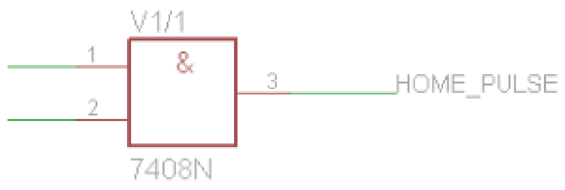


Figure 7 AND Gate

The UCS pulse and the PSS pulse are then passed through an AND gate, shown in figure 7. When the home and PSS pulses satisfy the AND condition this indicates the cartridge has returned to its starting position. In testing it was found that the outputs of the Schmidt triggers were distorted while the output of the AND gate remained a stable logic signal. Capacitors were placed across the supply inputs of each operational amplifier however this failed to fully remove the signal distortion. This signal is of great importance to our algorithm because it will be used to indicate when the algorithm may begin to queue print hammers. In the current configuration the PSS and UCS pulse outputs of the Amplifier card have been passed through the AND gate each with the second operand being true (logic high, 3.3V). This provided stabilization of the PSS and UCS pulse outputs from the Schmidt triggers. It was also found in testing that the outputs of the Schmidt Triggers were active low pulses. This presented a challenge for receiving the AND condition of the UCS and PSS extra pulse because both pulse were active low and had a logical value of false (logic low, 0V). A NAND gate was purchased as a precautionary measure, but ultimately the NAND function was successfully implemented in software and the Amplifier PCB was not repopulated with a NAND gate. If it is desirable in future iterations of this project to perform the AND or NAND function in hardware, the AND gate may be replaced by the provided NAND gate. This would provide an active high pulse when the NAND condition is detected. Additionally, if the PSS pulse and UCS pulse are passed through the NAND gate each with the second operand being false (logic low, 0V) an active high representation of the PSS and UCS pulses will be produced.

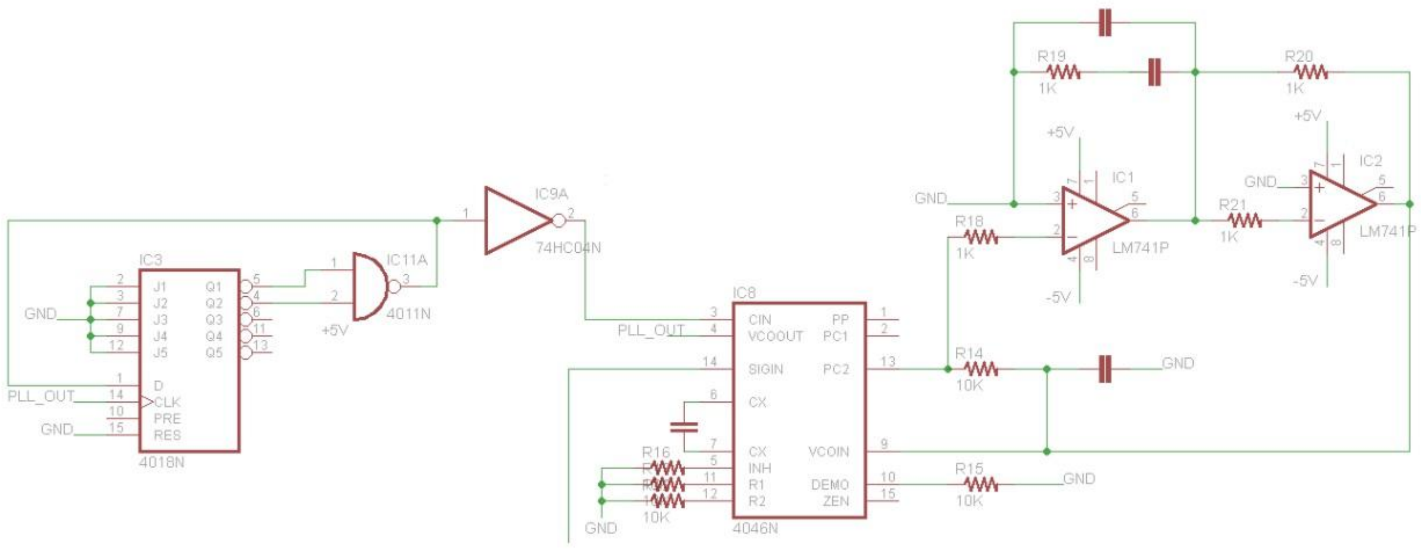


Figure 8 Type 2 Phase Locked Loop

Included on the Amplifier PCB is a Type 2 Phase Locked Loop . In subsequent printers the character belt was reduced in size to increase the number of characters on the character belt and the speed of printing. In these later printer designs a single pulse was generated per scan. The Phase Locked Loop was used in a divide-by-3 configuration to provide a unique PSS, Print Sub-Scan, pulse.

The Phase Locked Loop above has been included on our Amplifier PCB. However it will not be used in the control of the IBM1403N1 printer. In future iterations of this, or similar, projects the Phase Locked Loop may be useful in generating PSS pulses while controlling later printer models. The Phase Locked Loop provided performs a divide-by-three function, shown in figure 8. The current configuration may use either of two different varieties of filters in the forward control path. The first filter is a phantom integrator, and the second, a low pass filter. Currently the low pass filter has been by-passed by not populating the capacitor and resistor, "R14."

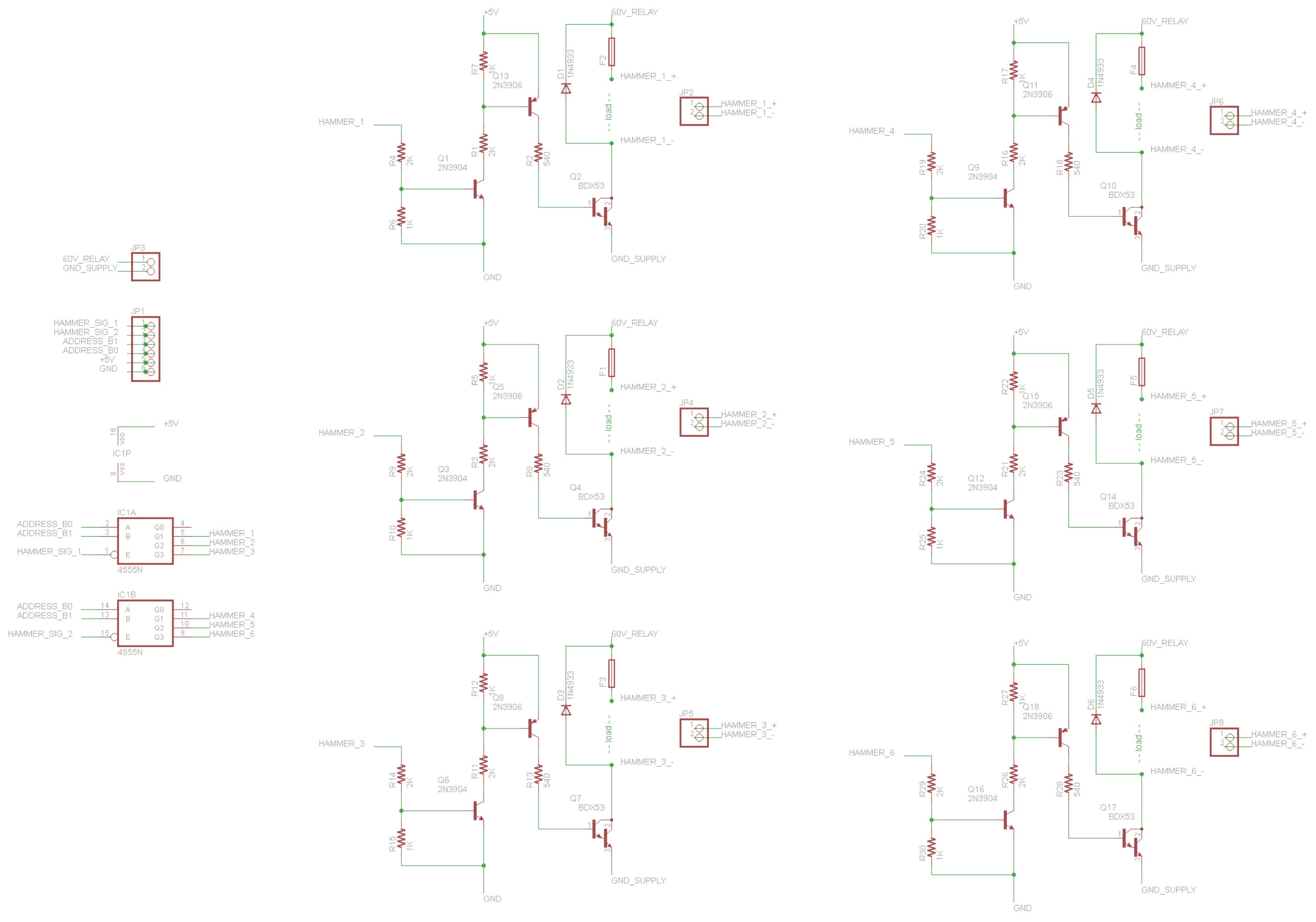


Figure 9 The Hammer Driver PCB

The Hammer Driver PCB, imaged above, will accept an active low pulse of approximately 1170 microseconds, originating from the microcontroller, via the Amplifier PCB. This pulse will be sent to the correct hammer driver circuit by utilizing an inverted-enable decoder that outputs a logic high value on the addressed output when the enable pin receives a low signal.

The decoder used in the design is a dual 2-4 decoder, shown in figure 11. Due to nature of the printer upon receiving the UCS pulse the address 01, may be sent to the addressing pins of all decoder. This will select hammers 1, 4, 7, and 10 in our project, and may be expanded to select all hammers satisfying the equation 3n-2

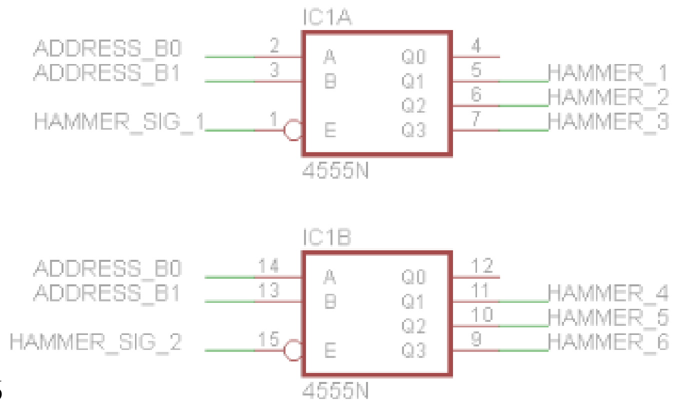


Figure 10 Decoder

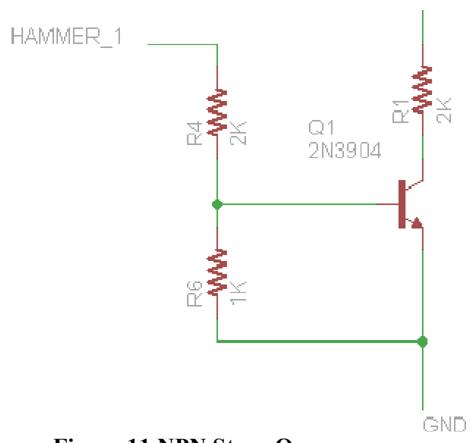


Figure 11 NPN Stage One

(where n is a positive integer) if the project is expanded to include all IBM1403-N1 Printer hammers. Upon receiving the first PSS pulse the address 10 will be sent to all decoder addressing pins, selecting hammers 2, 5, 8, and 11 in our project, and may be expanded to all hammers satisfying the equation 3n-1 (where n is a positive integer). When the third PSS pulse is received the address 11 will be sent to the addressing pins of all decoders. This will select hammers 3, 6, 9, and 12 in our project, and may be expanded to select all hammers satisfying the equation 3n (where n is a positive integer).

The active high output of the decoder will allow current to pass in the initial NPN stage of the driver circuit, shown in figure 12. This configuration allows for the use of a 3.3V logic signal to activate the driver circuit without shifting the logic level provided by the microcontroller as many methods of logic level conversion introduce undesirable delay.

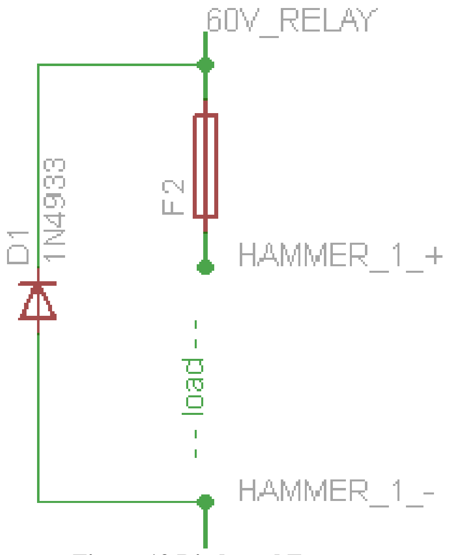


Figure 12 Diode and Fuse

The Hammer Driver PCB also contains protective measures, shown in figure 13, meant to disable a driver if the current signal becomes fixed in the "on" state. This may occur if the hammer is unable to return to its neutral position after contact. A 1.5A slow-blow fuse is included in the current path that will deliver the 60V 5A pulse to the IBM1403-N1 Printer hammer solenoids. This fuse was included in the original IBM1403-N1 Hammer Driver PCB, and is rated to 1.5A due to the very low duty cycle that will be imparted on the printer hammer. If this duty cycle becomes large enough due to any error the fuse will then blow disconnecting the affected hammer. The fuse will be mounted on fuse clips to allow for solder-less replacement of a blown fuse. Another protective measure, found in each driver circuit, is a reverse bias diode which is placed in parallel with the fuse and solenoid load. When the current is applied to the hammer solenoid the diode is reverse biased and no current will flow in the diode path. However when the current signal is removed from the hammer solenoid a negative voltage spike will be generated due to the inductive properties of a solenoid coil. The diode will then clamp the reverse voltage spike to approximately 0.7V (the built-in potential of the diode).

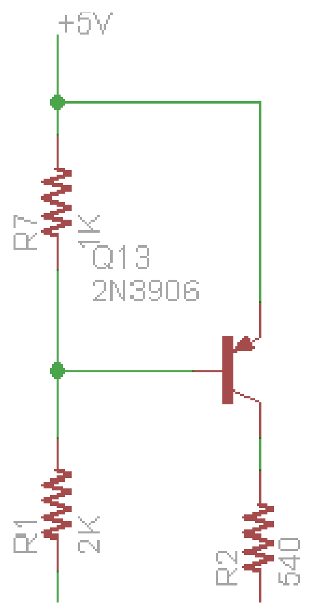
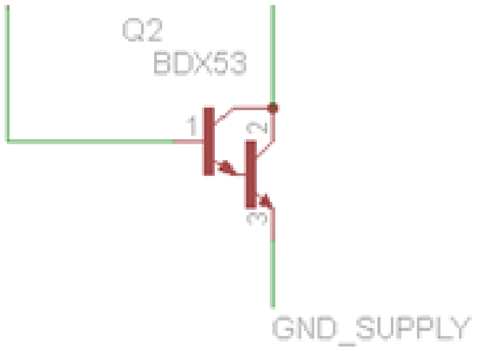


Figure 13 PNP Stage

Another protective measure is the PNP transistor, shown in figure 14, which will allow the circuit to remain in the "off" state if the input is floating. Since a floating signal tends to float in the positive voltage direction. This circuit prefference was requested by technical advisor Bob Arnold and is in keeping with IBM circuit design practices. Our decoder provides a similar protective quality although its active high input is not in keeping with the same preference. The decoder outputs are tied to ground when not selected, and additonally the address 00 does not correspond to a signal path. These design decisions are intended to prevent the unintentional firing of hammers if the microcontroller outputs are in an undefined or floating state prior to initialization. The SPST switch located on the Amplifier PCB serves a similar purpose. This switch will remain in the “off” position until sufficient time has passed, allowing the microcontroller to initialize the states of the output pins prior to the application of the 60V 5A source.



The final NPN stage, shown in figure 15, of the driver circuit is where the current signal is amplified, a darlington transistor was selected in order to ensure a large enough gain and a large enough maximum collector current to provide 5A draw to the hammer solenoid. A TO-220 package was selected for this transistor to allow the necessary thermal dissipation. The footprint of a TO-220 provides exposed copper on the surface of the PCB that may be used to increase the amount of thermal dissipation.

Figure 14 NPN Stage Two

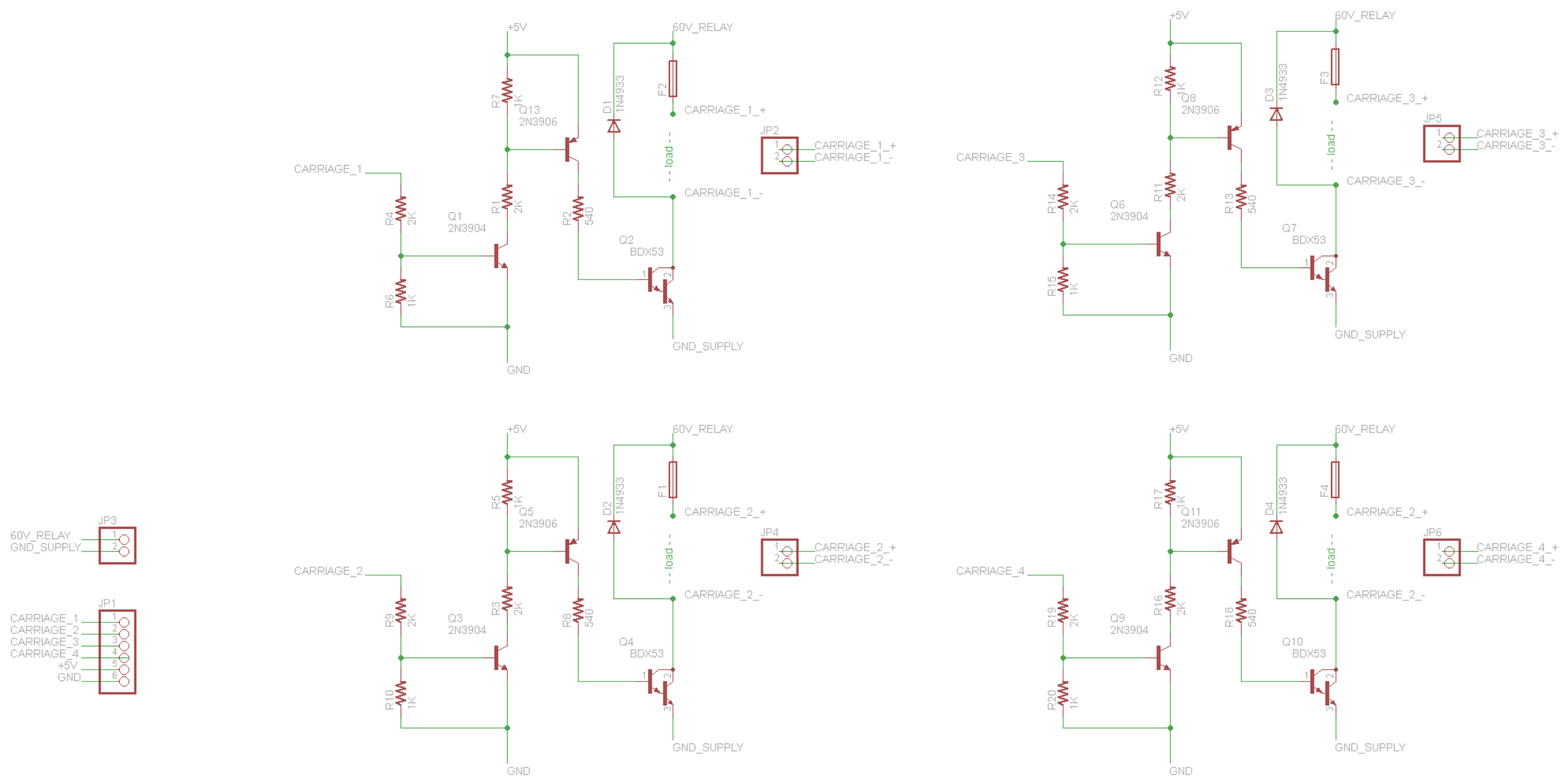


Figure 15 The Carriage Driver PCB

The Carriage Driver PCB receives an active high signal, originating from the microcontroller, via the Amplifier PCB. The signal “Carriage 1” will control the “space start” magnet within the IBM1403-N1 Printer, the signal “Carriage 2” will control the “space stop” magnet, the signal “Carriage 3” will control the “skip start” magnet, and the signal “Carriage 4” will control the “skip stop” magnet. These four solenoids each control a hydraulic valve which per forms the functions “space start,” “space stop,” “skip start,” and “skip stop.” A connection to the SPST switch controlled 60V 5A source is made via the Amplifier PCB. The Carriage Driver PCB outputs will interface with the IBM1403-N1 printer by wire connection to the corresponding wire in the “shoe” connector which was originally used to connect the IBM1403-N1 printer to a mainframe computer.

The various stages of transistors perform the same functions as in the Hammer Driver PCB. However, the gain of the transistors will be reduced in order to provide a low current signal to the solenoids controlling the “space start,” “space stop,” “skip start,” and “skip stop” valves. The signals produced by the Carriage Driver PCB will provide an “on” state to either the start or stop valves at all times for both the “space” valves and the “skip” valves.

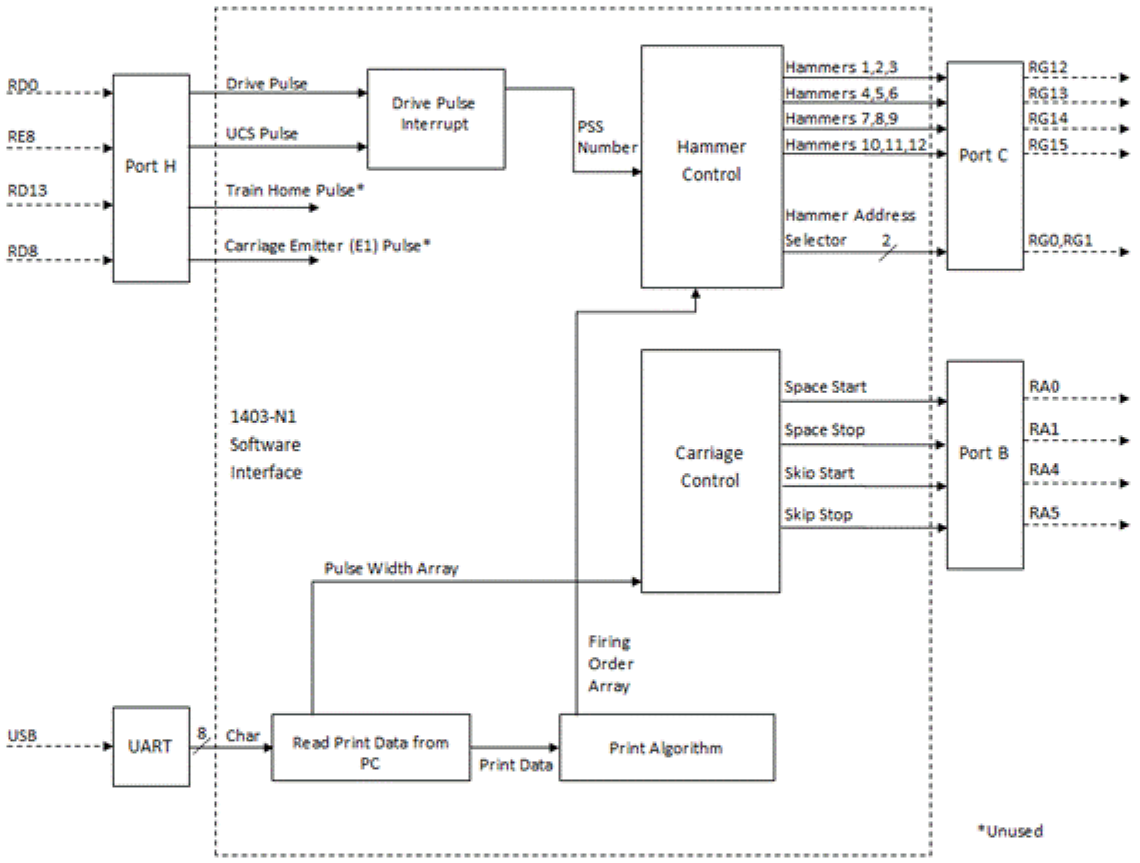
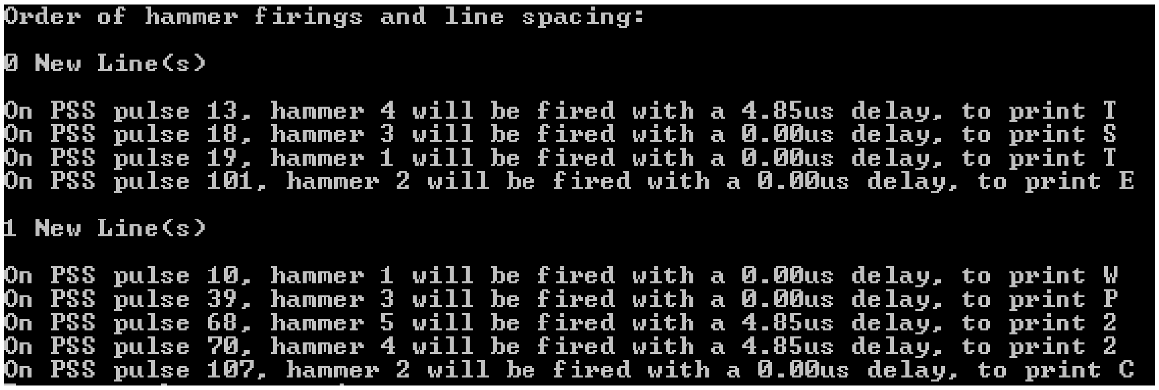


Figure 16: Microcontroller Software Architecture

The PC connected to the system will be loaded with the Character Transfer Module which will send the text file data to the microcontroller. The source code for this module is in the Appendix. The transmission of data will occur through the USB cable attached to the PC. After initialization of the serial port, the program will create a ‘BufferedReader’ type variable to read the file character-by-character and store it to a string. After the end of the file is reached, the program will write each character to the port with a 100 millisecond delay between each write.

In synchronization with this program is the IBM 1403-N1 Software Interface which received the text file. The software will load the characters into onboard memory and after receiving the end character from the PC, will move into the print algorithm. While the system is receiving characters, it will track how many new lines are between each line of characters. As seen in the screenshot below, the print algorithm assigns a PSS value to each of the hammers which signifies on which PSS pulse input they will be printed on. The printing algorithm will determine for each line when to fire the printer hammers based on the PSS pulse. This is accomplished in software by creating an array that mimics the character belt of the IBM1403-N1 printer. The program also mimics the operation of the printer by moving through the character belt in the same fashion that the physical belt rotates in the printer. The data accumulated from the printing algorithm will be utilized when the printing has started. By monitoring the PSS pulses during the actual printing, the program can use the data gained from the printing algorithm to determine when to fire the printer hammers without having to do much on the spot computation.



```
Order of hammer firings and line spacing:

0 New Line(s)

On PSS pulse 13, hammer 4 will be fired with a 4.85us delay, to print T
On PSS pulse 18, hammer 3 will be fired with a 0.00us delay, to print S
On PSS pulse 19, hammer 1 will be fired with a 0.00us delay, to print T
On PSS pulse 101, hammer 2 will be fired with a 0.00us delay, to print E

1 New Line(s)

On PSS pulse 10, hammer 1 will be fired with a 0.00us delay, to print W
On PSS pulse 39, hammer 3 will be fired with a 0.00us delay, to print P
On PSS pulse 68, hammer 5 will be fired with a 4.85us delay, to print 2
On PSS pulse 70, hammer 4 will be fired with a 4.85us delay, to print 2
On PSS pulse 107, hammer 2 will be fired with a 0.00us delay, to print C
```

Figure 17 Printing Algorithm Simulation

The above screenshot shows the print operation for a text file with “TEST” on the first line and “WCP22” on the second line.

After the print algorithm is performed the software will begin controlling the hammers and carriage to print the print data. An interrupt, which runs from the start of the program, is used to track the drive pulse from the printer in order to synchronize with the character belt. The UCS pulse in addition to the drive pulse is used to determine when the character belt has performed a complete revolution. The carriage control has the operation of moving the paper through the printer while the hammer control fires hammers on the correct PSS pulse values to print the data.

4.3 Concept of execution

Print Control (Cartridge) Mechanism:

The IBM1403-N1printer follows what is known as on the fly printing. The type, consisting of the characters, moves continuously behind a paper. Refer figure 18 below. The hammers press the paper against the moving type when the required character is in position. The key to this type of functionality is the knowledge of when to fire the right hammer.

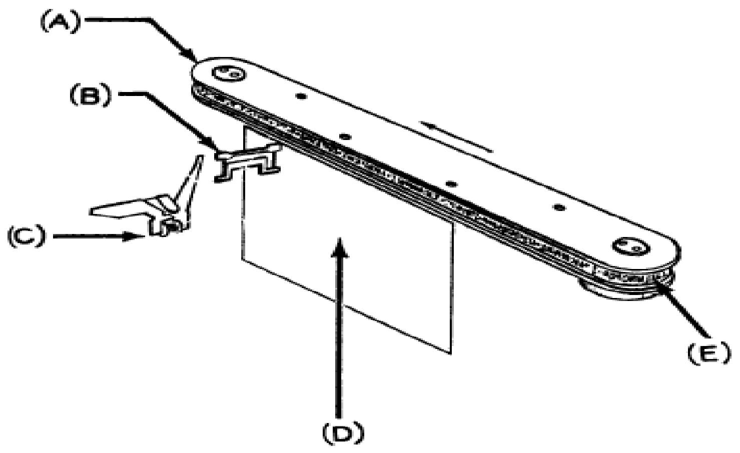


Figure 18 The Printer Cartridge

A – Type array
B – Hammer unit
C – Armature hammer magnet
D – Paper form – stationary during printing
E – Chain

The chain consists of 240 characters. The characters are divided into sets of 3 called slugs, so 3 characters per slug. The type moves at a speed of 206.4 inches per second. A print hammer is available at each print position. The distance between each print position or hammer unit is 0.100 inches. There are 132 hammer units and hence each line consists of 132 characters.

Important signals that need to be monitored are: emitter pulses, UCS pulses and PSS pulses. These signals come from the printer. In our project these signals are fed to a microcontoller that processes it and based on the timings of these pulse it makes decisions to fire which hammer and at what intervals.

Printing is carried out serially, one character at a time. At any given time one third of the hammers are lined up to some character on the type face. Hence 3 subscans, each containing 44 print positions, are needed to expose all characters to each print position. Figure 19 below outlines the printing operation.

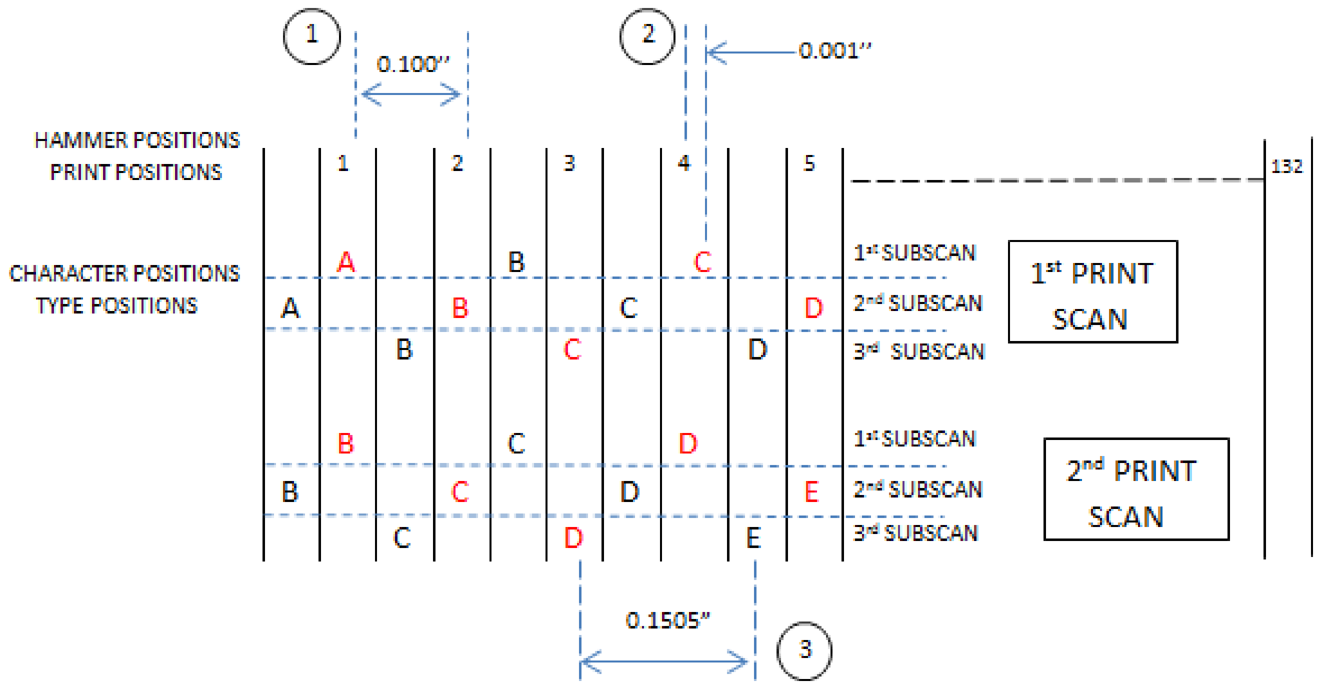


Figure 19 Scan and Subscan States

Characters in red are aligned with hammer.

1 – distance between two hammer/print postions
2 – offset of 2nd aligned character within a subscan
3 – distance between two characters on the type.

One complete scan brings character B to print position 1. Taking into account type speed and distance between two adjacent characters this takes 729 microseconds. Hence time for each subscan will be one third that value or 243 microseconds.

Emitter pulses are sent out at the beginning of each subscan. At the first emitter pulse hammers 1,4,7..,130 are aligned for printing. If character A is to be printed, hammer one can be fired and then hammer 4 is optioned for printing character C, 4.85 microseconds later. Print position 7 is then optioned for printing until one third or 44 hammer positions have been printed. Since type face is moving at a rate of 206.4 inches per second and offset is 0.001 inches it takes 4.85 microseconds for C to align with hammer 4. Offset is included for serial printing. In this fashion 44 hammer positions accumulate a total time of 44x4.85 = 213.4 microseconds of 243 microseconds available per subscan.
At the end of this time, second subscans begins with character B aligned with hammer two. In third subscan character C is aligned with hammer 3. At the end of the third subscan, one scan is complete and 44 characters have been exposed to 44 print posionts.
A home/PSS pulse is received once the type has taken one whole revolution that is A is back at print positions 1.

Carriage mechanism:

In order to print multiple lines, the IBM1403-N1 sends out carriage signals which control line spacing as well as line skipping. The IBM1403-N1 is capable of printing 1100 lines per minute that is it takes 54 milliseconds per line. Line spacing is carried out by two magnets, space start magnet and space stop magnet. Either one is energized at all times with 60 V supply. At the start of a new line feed space start pulse is turned on. It is then turned off and space stop pulse is turned on. The start of the space stop pulse also signals a carriage-settling single shot signal to be pulsed. The carriage settling single shot allows time for the carriage to settle down after spacing or skipping and to control the start of the next print operation. Single line spacing takes 20 milliseconds. Hence time allowed for printing a single line is 34.55 milliseconds. Our microcontroller is fast enough to pick up these timings and be able to print on a new line after it receives the next PSS pulse.

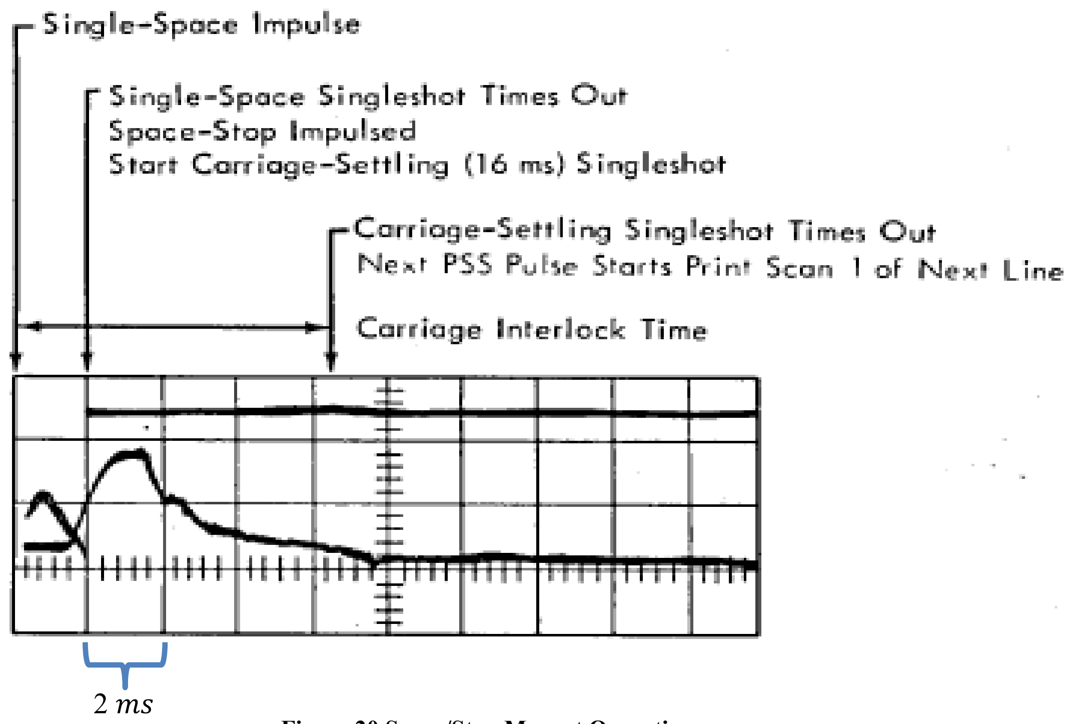
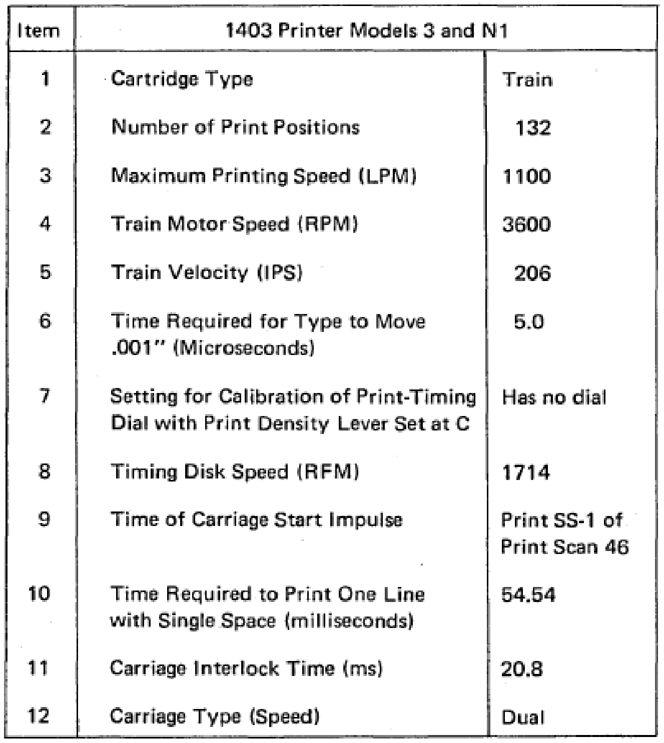


Figure 20 Space/Stop Magnet Operation



Item	1403 Printer Models 3 and N1	
1	Cartridge Type	Train
2	Number of Print Positions	132
3	Maximum Printing Speed (LPM)	1100
4	Train Motor Speed (RPM)	3600
5	Train Velocity (IPS)	206
6	Time Required for Type to Move .001" (Microseconds)	5.0
7	Setting for Calibration of Print-Timing Dial with Print Density Lever Set at C	Has no dial
8	Timing Disk Speed (RFM)	1714
9	Time of Carriage Start Impulse	Print SS-1 of Print Scan 46
10	Time Required to Print One Line with Single Space (milliseconds)	54.54
11	Carriage Interlock Time (ms)	20.8
12	Carriage Type (Speed)	Dual

Figure 21 Timings for IBM1403-N1 Printer

An emitter pulse called “E1,” also known in this document as the “carriage emitter pulse,” is used to verify the forms movement. For each line skipped it sends out one pulse. These emitter pulses are used to count how many lines have been skipped. They are primarily used for high speed skipping of more than 3 line spaces. For this purpose another set of magnets called skip start and skip stop magnets are used in conjecture with the space start and stop magnets. Both space and skip magnets are turned on at the same time. E1 pulses are counted and with 4 to 10 (adjustable) E1 pulses left, the skip magnet is turned off and with 3 to 6 (adjustable) E1 pulses to go space stop magnet is turned off.

Figure 21 above summarizes the timings discussed in this document regarding the print and carriage functionality.

4.4 Interface design

Internal interfacing of the microcontroller, Amplifier PCB, Carriage Driver PCB, and two Hammer Driver PCBs will be accomplished with ribbon cable and in the case of the Amplifier PCB a 14 foot Cat 6 cable will be used to allow for the Amplifier PCB board to be placed inside the printer cabinet while the other sub-systems remain in a near -by service box within an industrial rack. The interface between the Amplifier PCB and the IBM1403-N1 Printer will be accomplished by connection to the original “shoe” connector of the IBM1403-N1 Printer, or through the connections to the original sense amplifier cards. WCP22 will provide wires or wire-terminals depending on the needs of CT&I. The Hammer Driver PCB and the Carriage Driver PCB interface to the IBM1403-N1 Printer will be accomplished by connection to the original “shoe” connector, or through the original card slots on the IBM1403-N1 Printer.

5 Traceability and Testing

This project's Test Procedures, see WCP22 Project Test Plan and Procedures document and Results Report located in appendix A will provide traceability from the project-level requirements to the project's subsystems and major components.

6 Notes

6.1 Acronyms and Abbreviations

CDR	Critical Design Review
CT&I	Center for Technology and Innovation
IBM	International Business Machines
PCB	Printed Circuit Board
PDP	Project Development Plan
PDR	Preliminary Design Review
PRS	Project Requirements Specification
PSS	Print Sub-Scan
SAR	System Acceptance Review
SDR	System Design Review
SPST	Single Pull Single Throw
SRR	System Requirements Review
TRR	Test Readiness Review
UCS	Universal Character Set
WCP	Watson Capstone Projects

6.2 Bibliography

Figure 6 - ***ieeexplore***.*ieee.org A Development Study of the Print Mechanism on the IBM 1403 Chain Printer,B. J. GREENBLOTT January 1963.*

Figure 8 - SY24-3395-3, Printer Models N1 and 3Maintenance Manual, section 4.16

Figure 9 - SY24-3395-3, Printer Models N1 and 3Maintenance Manual, section 4.16.

6.3 Appreciation

WCP22 would like to extend thanks to the IEEE Binghamton Chapter, the Center for Technology and Innovation, as well as, Professor Jack Maynard of Binghamton University, for invaluable contribution to the success of the IBM1403-N1 Printer Interface project.

A Appendices

Test Results

Software Modules

Explanation of the different text files generated by the print algorithm

PRINT.txt – shows the contents of the file to be printed (Print Data),the number of lines to be printed (Total Number Of Lines), number of characters on each line(chars_on_line) and the spacing before each character (line_buffer)
Note: it removes multiple new lines for the display so if input text was
A

D
It would display: A
D

Also the commas after each letter are only added for displaying here and are not actually added for printing.

TEST.txt (log data) –
X New Line(s): $ST_{carriage}$->$ET_{carriage}$ = $ET_{carriage}$-$ST_{carriage}$
Hammer: H Selector: XX (Port #PP) Printing: <C> on PSS XXX ST_{hammer}-> ET_{hammer}= ET_{hammer} - ST_{hammer}
Printing Time: ST_{line}-> ET_{line}= $ET_{line} - ST_{line}$

X - # of new line(s) to be processed
$ST_{carriage}$ – Start Time of space start magnet
$ET_{carriage}$ –End Time of space start magnet
H – hammer # to be fired
XX – selector chosen
PP – the port to which H maps to
<C> - the character to be printed
XXX – PSS pulse #
ST_{hammer} – Start Time of hammer
ET_{hammer}- End Time of hammer
ST_{line} – Start Time of a character to be printer on a given line
ET_{line} – End Time of a character to be printed on a given line

Note: all timings are in micro seconds. All timings happen after ST_{line} of the first character on the first line, i.e. all timings are recorded once the timing of the first character to be printed is determined.

ORDER.txt – outputs the PSS number in which the hammer would be fired

PSS.txt – outputs the average of all PSS pulses in one revolution of the train.

Character Transfer Module Test

Test Level	Sub-System
Test Type or Class	Erroneous Input
Qualification Me\thod(s)	Test
System Requirement(s)	WCP22-2-3

This test will take place in the Watson Capstone Labs. This test shall check whether the file transferred for printing is sent to the microcontroller correctly. This test will be performed in fulfillment the requirement WCP22-2-3. The test data to be recorded is inputted to the microcontroller from PC as a .txt file.

Test Results

Test Date: 04/19/14
Location: Watson Capstone Labs

Test Date: 04/19/14
Location: Watson Capstone Labs

Print.txt (Print Data)	**Text File (From PC)**
`Total Number Of Lines: 3`	TESTING
`chars_on_line:`	
`7,7,8,`	A B C
`line_buffer:`	H E LLO
`0,2,1,`	
`Print Data:`	
`T,E,S,T,I,N,G,`	
`A, , ,B, , ,C,`	
`H, ,E, , ,L,L,O,`	

The verification is printed to the ‘Print.txt’ text file in the SD card that is attached to the microcontroller. The total number of lines counts how many lines of characters are in the text file. The chars_on_line header signifies the number of characters per line. The line_buffer header signifies the number of newlines before the current line of characters) The print data shows the text file unformatted. The data to be printed was successfully received by the microcontroller as can be verified by the content of the PRINT.txt .

Software Interface Module Test

Test Level	Sub-System
Test Type or Class	Timing
Qualification Method(s)	Test
System Requirement(s)	WCP22-2-4

This test will take place in the Watson Capstone Labs. The test shall check for the overall functionality of the print algorithm which takes inputs as text file and feedback signal from the simulated printer and outputs hammer firing signals and carriage signals at correct intervals. The test would need to run for at least two file inputs to check for consistency and check that correct outputs. This may take anywhere between 10 to 15 minutes. Data to be recorded would be length of output signals from the pins of the microcontroller. These would be cross checked with the values that have already been established to be accurate and precise. Outputs will be monitored with an oscilloscope.

Test Results

Test Date: 04/30/14
Location: Watson Capstone Labs

Test Date: 04/30/14
Location: Watson Capstone Labs

This test is divided in two categories

a. Test for minimum printing time – which includes testing the requirement that a minimum of 34 ms is elapsed from the first hammer being fired to when the carriage magnets are activated.

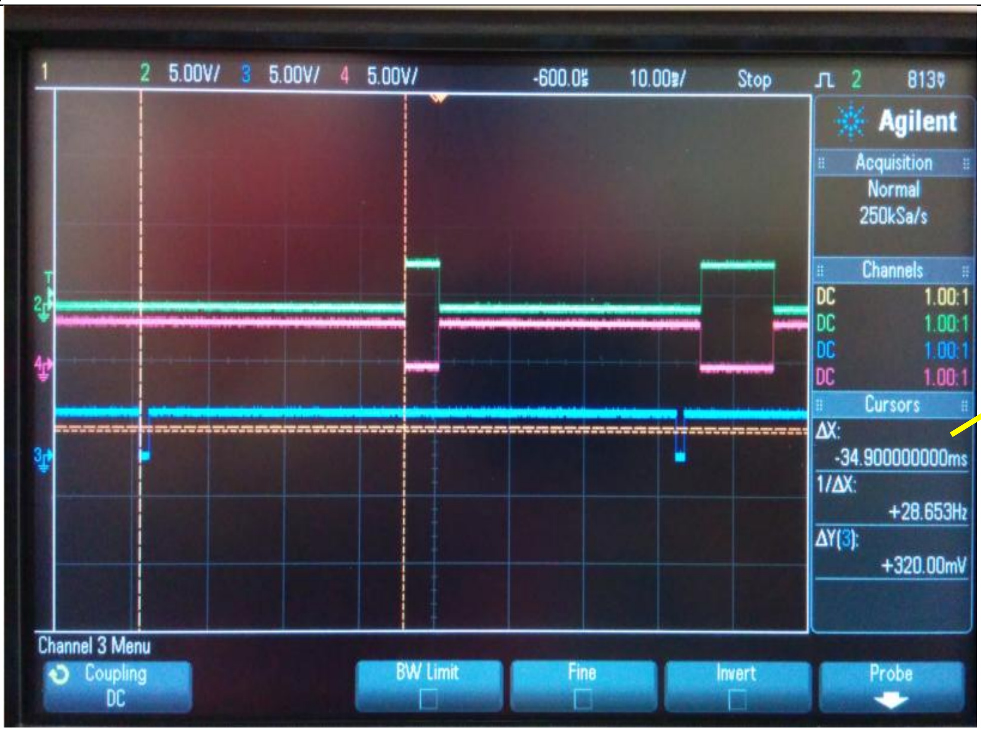
b. Test for required carriage on time – which includes the requirement that the single space magnet is on for 4.6 ms, 9.8 ms, 13.8 ms for single double and triple line spacing.

Test Results for (a).

PRINT.txt (Print Data)	ORDER.txt (PSS Pulse)	PSS.txt (PSS Pulse Period)

`Total Number Of Lines: 4`
`chars_on_line:`
`1,1,1,1,`
`line_buffer:`
`0,1,2,3,`
 	
`Print Data:`
`A,`
`D,`
`G,`
`J,`
	`109,`
	`118,`
	`127,`
	`73,`
		`Average: 240`

TEST.txt (Printing Data Log)
0 New Line(s): 0 -> 0 = 0 Hammer: 1 Selector: 01 (Port #16) Printing: A on PSS 253 4077415 -> 4078586 = 1171 Printing Time: 4077393 -> 4112388 = 34995 1 New Line(s): 4112392 -> 4116993 = 4601 Hammer: 1 Selector: 01 (Port #16) Printing: D on PSS 550 4148763 -> 4149934 = 1171 Printing Time: 4116996 -> 4151992 = 34996 2 New Line(s): 4151996 -> 4161797 = 9801 Hammer: 1 Selector: 01 (Port #16) Printing: G on PSS 703 4185516 -> 4186691 = 1175 Printing Time: 4161800 -> 4196795 = 34995 3 New Line(s): 4196803 -> 4210603 = 13800 Hammer: 1 Selector: 01 (Port #16) Printing: J on PSS 217 4241702 -> 4242878 = 1176 Printing Time: 4210606 -> 4245600 = 34994



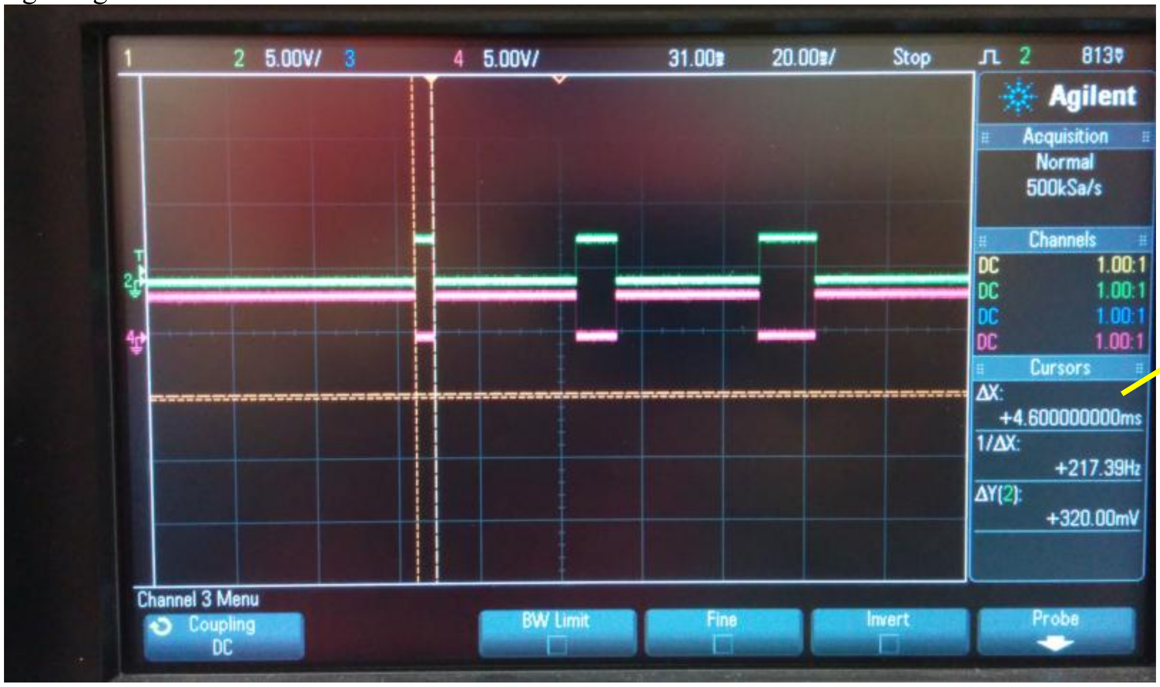
Required time

Minimum print timing

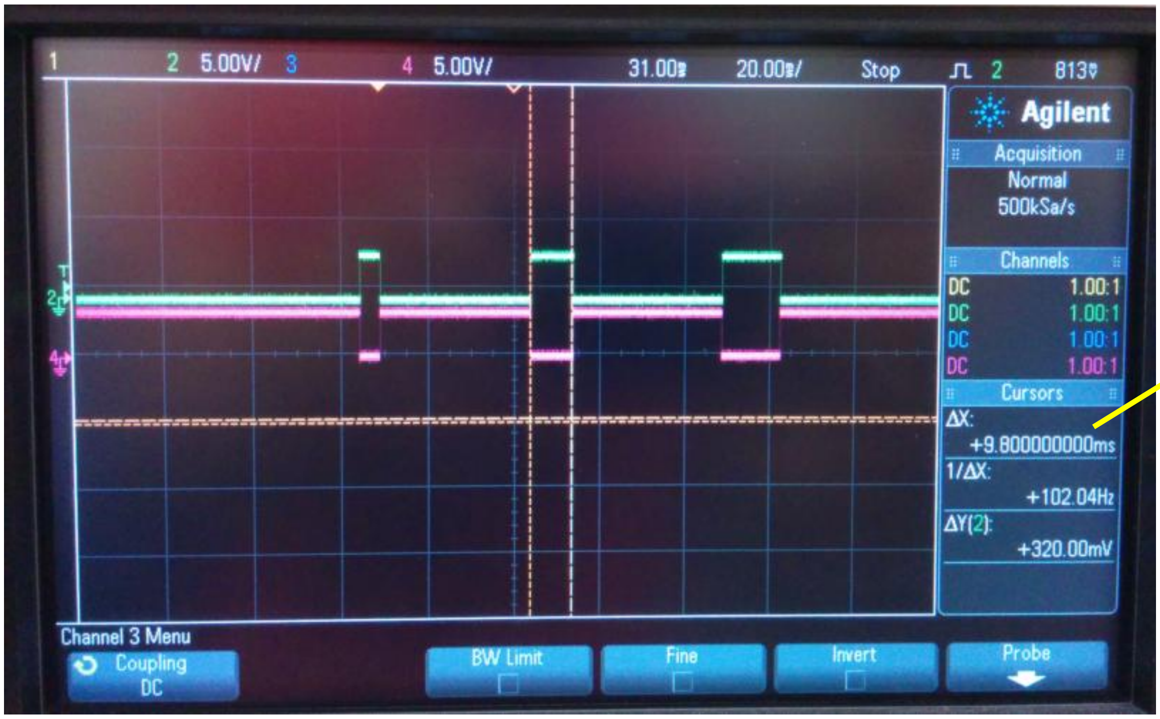
The content to be printed appears in the PRINT.txt and under Print Data subheading. From TEST.txt we see that A is fired on hammer 1 and it is fired soon as the printing has started has started. Since this is the only character being fired on that line, the rest of time is spent without firing any hammer and at the end of the allotted time of 34 ms (time to print one line) the carriage is activated. This is verifiable from the figure above. The first blue pulse(bottom pulse) signifies hammer 1 (printing character A) is fired. The green pulse(top) shows the space start pulse and the red (middle) is the space stop pulse (notice how one only one stays high at all time). The time from when hammer 1 is fired to when the space start pulse get active(and in turn apace stop pulse gets deactivated) is 34.9 ms.

Test Results for (b)

From PRINT.txt under subheading line_buffer, we see that there is 1 line space between A and D, 2 between D and G and 3 between G and J. The correct timing of these are verified from the three figures given below.

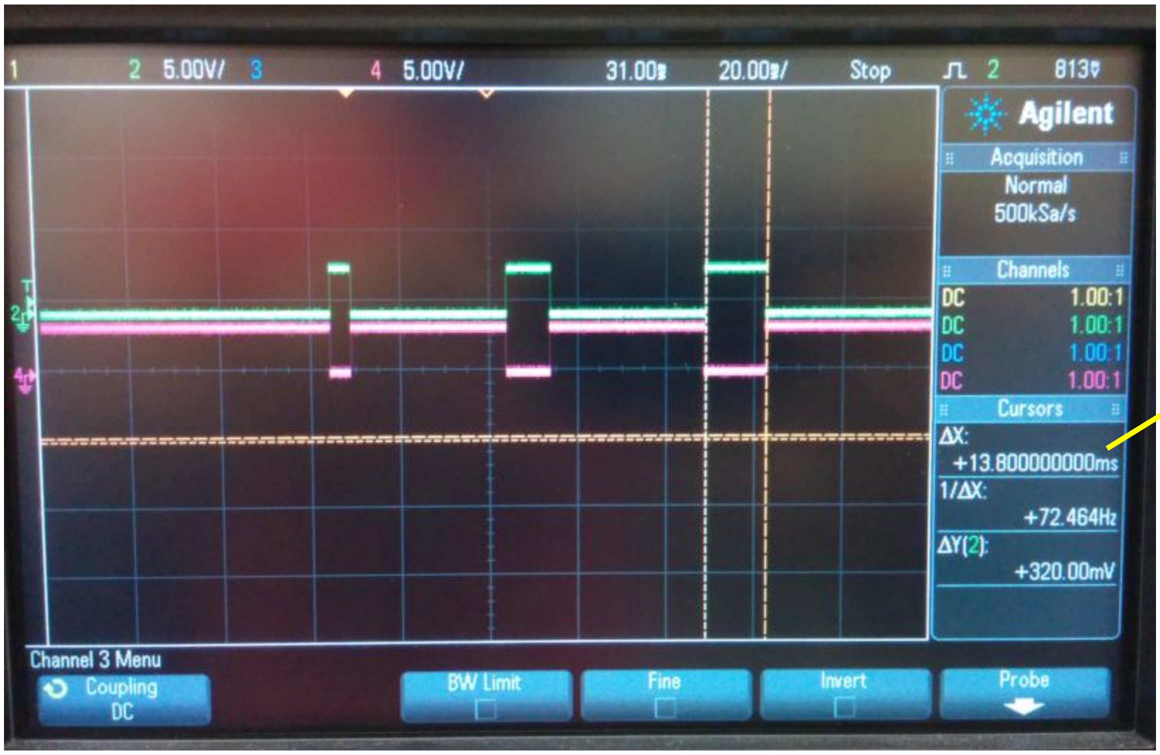


Carraiage timing (single line)



Required time

Carriage timing (double line)



Carriage timing (triple line)

In this test single character was fired per line and there are 3 lines.

Feedback Independent Module Test

Test Level	Sub-System
Test Type or Class	Timing
Qualification Method(s)	Test
System Requirement(s)	WCP22-2-4

This test will take place in the Watson Capstone Labs. The Feedback Independent Module is a variant of the Software Interface Module. It varies in that it receives as input only the text file and not feedback signals from the printer. The purpose of this module is only to check the correctness of the print algorithm by checking the output timing. The test would need to be implemented to run for multiple inputs as it would point out bugs (if any) with each run which would be handled as necessary. Data to be recorded would firstly be checking for any outputs at the pins. Then timings of the outputs are checked. Finally the length of the signals is checked. Passing these three requirements would mean that signals are being outputted as designed. Output pins will be monitored with an oscilloscope.

Test Results

Test Date: 04/25/14
Location: Watson Capstone Labs

Test Date: 04/35/14
Location: Watson Capstone Labs

PRINT.txt (Print Data)	ORDER.txt (PSS Pulse)
Total Number Of Lines: 1 chars_on_line: 2, line_buffer: 0, Print Data: W,P,	52,89,

The correctness of the print algorithm can be verified by comparing the PSS values with the IBM 1403-N1 Printer Visual Simulation. (Provided in the Appendix) After review, the PSS values were determined to be correct in addition to a number of tests.

Hammer Driver PCBs (2)

The Hammer Driver PCBs will be tested to verify output timing, selectivity and power in fulfillment of requirements WCP22-2-2, WCP22-2-4, WCP22-2-6

Hammer Driver PCBs Output Selectivity Test

Test Level	Sub-System
Test Type or Class	Output Selectivity
Qualification Method(s)	Test
System Requirement(s)	WCP22-2-4

The output selectivity test will verify the correct selection and connection of the Hammer Driver PCB outputs to printer hammers. The microcontroller will attempt to select and actuate hammer outputs sequentially. The input data and Hammer Driver PCB outputs will be monitored by an oscilloscope to verify the output selectivity.

Test Results

Test Date: 03/22/14
Location: Watson Capstone Labs

Additional Test Date: 03/31/14

Location: Techworks!

	Selectivity?		Selectivity?
Hammer Driver 1	YES	Hammer Driver 7	YES
Hammer Driver 2	YES	Hammer Driver 8	YES
Hammer Driver 3	YES	Hammer Driver 9	YES
Hammer Driver 4	YES	Hammer Driver 10	YES
Hammer Driver 5	YES	Hammer Driver 11	YES
Hammer Driver 6	YES	Hammer Driver 12	YES

Observations:
Each Hammer driver could be selected and controlled individually.
Hammer drivers activated when active low pulse was received.

Hammer Driver PCBs Output Timing Test

Test Level	Sub-System
Test Type or Class	Output Timing
Qualification Method(s)	Test
System Requirement(s)	WCP22-2-4

The timing test will be performed in the Watson Capstone Labs and will verify an output pulse of 1170 µs ±5%, in part, fulfilling requirement WCP22-2-4. Since this dummy load will determine the output current, WCP22 will choose this resistor to produce a low amount of current at the Watson Capstone Lab. A 100 ohm resistor may be used in the Watson Capstone Labs, with a 10 volt power-supply, to limit the output current to 0.1 amp. In this scenario the resistor(s) will be rated to 1 watt. The microcontroller will be used to actuate hammer driver circuits for 1170 µs ±5%.The input data and Hammer Driver PCB outputs will be monitored by an oscilloscope to verify the output timing.

Test Results

Test Date: 03/22/14
Location: Watson Capstone Labs

Additional Test Date: 03/31/14
Location: Techworks!

	Timing Correct?		Timing Correct?
Hammer Driver 1	YES	Hammer Driver 7	YES
Hammer Driver 2	YES	Hammer Driver 8	YES
Hammer Driver 3	YES	Hammer Driver 9	YES

Hammer Driver 4	YES	Hammer Driver 10	YES
Hammer Driver 5	YES	Hammer Driver 11	YES
Hammer Driver 6	YES	Hammer Driver 12	YES

Observations:
Hammer Drivers were able to produce an output of pulse width 1170 us.
Output durations of 500 us, 300 us, and 200 us were also achieved.
Hammer Driver timing can be predictably controlled to produce desired pulse widths.

Hammer Driver PCBs Output Power Test

Test Level	Sub-System
Test Type or Class	Output Current
Qualification Method(s)	Test
System Requirement(s)	WCP22-2-4

The power test will be performed at TechWorks! and will verify the capability of providing a 5 amp pulse of 1170 µs ±5%. In combination with the output timing test, the output power test fulfills the Hammer Driver PCB portion of requirement WCP22-2-4. For the power test a different dummy load will be required to simulate the presence of a printer hammer. WCP22 will choose this resistor to produce 5 amps of current at TechWorks!. A 2 ohm resistor may be used, with a 10 volt power-supply to produce an output current of 5 amps. The resistor(s) in this scenario will be rated for 50 watts. The microcontroller will be used to actuate hammer driver circuits for 1170 µs ±5%. The Hammer Driver PCB outputs will be monitored by an oscilloscope to verify the output current.

Test Results

Test Date: 03/31/14
Location: Techworks!

	Achieved 5A?		Achieved 5A?
Hammer Driver 1	YES	Hammer Driver 7	YES
Hammer Driver 2	YES	Hammer Driver 8	YES
Hammer Driver 3	YES	Hammer Driver 9	YES
Hammer Driver 4	YES	Hammer Driver 10	YES
Hammer Driver 5	YES	Hammer Driver 11	YES
Hammer Driver 6	YES	Hammer Driver 12	YES

Observations:
Test was performed using a 60V supply, and hall-effect current probe.
Load was an actual hammer unit from an IBM1403-N1, roughly 5 Ohms, 4.7 Henries
Hammer Drivers were able to produce an output of 5A, and up to 10A pulsed.

Hammer Drivers unintentionally output 10A with 99.9% duty-cycle, no damage to circuit. Fuse did not blow- Fuses may need to be changed to FAST-BLOW.

Hammer Driver PCB Inspection

Test Level	Sub-System
Test Type or Class	Hardware
Qualification Method(s)	Inspection
System Requirement(s)	WCP22-2-2 , WCP22-2-6

The Hammer Driver PCB will be inspected by WCP22 and CT&I experts to ensure logic levels have been transitioned safely and efficiently and the Hammer Driver Card has been reproduced using modern components. The system, circuit schematics, and bill of materials will be available for reference and inspection.

Test Results

WCP22 has performed the following in fulfillment of the above requirements.

It was not necessary to transition from one logic level to another. WCP22 has purchased all components new from Unicorn Electronics and Digi-Key. All were readily available.

Carriage Driver PCB

The Carriage Driver PCBs will be tested to verify output timing, selectivity and power in fulfillment of requirement WCP22-2-4

Carriage Driver Output Timing and Selectivity Test

Test Level	Sub-System
Test Type or Class	Output Pattern
Qualification Method(s)	Test
System Requirement(s)	WCP22-2-4

The output selectivity test will verify the correct selection and connection of the Carriage Driver PCB outputs to the carriage magnets. The microcontroller will attempt to select and actuate carriage magnets as in accordance with the magnet control parameters below. A large value resistor will be used as a dummy load to limit the current. A 1k ohm resistor will limit the current to 0.01 amps, when supplied with 10 volts. The output will begin in the No Movement state, then the Lane Space state for 9 ms, the return to the No Movement state. The Line Skip state will be tested in the same progression, but for longer durations in the Line Skip state, such as 100 ms. The input data and Carriage Driver PCB outputs will be monitored by an oscilloscope to verify the output selectivity.

Output States	Space Start	Space Stop	Skip Start	Skip Stop
No Movement	L	H	L	H
Line Space	H	L	L	H
Line Skip	H	L	H	L

Test Results

Test Date: 03/22/14
Location: Watson Capstone Labs (Selectivity and Timing)

Test Date: 03/31/14
Location: Techworks! (Selectivity)

	Selectivity?	Timing?
Carriage Driver 1	YES	YES
Carriage Driver 2	YES	YES
Carriage Driver 3	YES	YES
Carriage Driver 4	YES	YES

Observations:
Carriage Drivers can be selected and controlled independently.
Carriage Drivers produced all four carriage control patterns.

Carriage Driver Output Power Test

Test Level	Sub-Sub-System
Test Type or Class	Output Current
Qualification Method(s)	Test
System Requirement(s)	WCP22-2-4

The power test will be performed at Watson Capstone Labs, because the carriage control will require less than 2 amps of current. This test will verify the capability of providing a 0.5 amp output. In combination with the Hammer Driver PCB tests, and the carriage output timing and selectivity test, requirement WCP22-2-4 will be fulfilled. For the power test a different dummy load will be required to simulate the presence carriage magnet. WCP22 will choose this resistor to produce 0.5 amps of current. A 20 ohm resistor may be used, with a 10 volt power-supply to produce an output current of 0.5 amps. The resistor(s) in this scenario will be rated for 5 watts. The microcontroller will be used to provide input to the carriage driver circuits. The Carriage Driver PCB outputs will be monitored by an oscilloscope to verify the output current.

Test Results

Test Date: 03/31/14

Location: Techworks!

	Produce 0.5A - 2A?
Carriage Driver 1	YES
Carriage Driver 2	YES
Carriage Driver 3	YES
Carriage Driver 4	YES

Observations:
Carriage Drivers were able to produce an output of 5A, and up to 10A pulsed.

Carriage Driver PCB Inspection

Test Level	Sub-System
Test Type or Class	Hardware
Qualification Method(s)	Inspection
System Requirement(s)	WCP22-2-2 , WCP22-2-6

The Carriage Driver PCB will be inspected by WCP22 and CT&I experts to ensure logic levels have been transitioned safely and efficiently and the Carriage Driver Card has been reproduced using modern components. The system, circuit schematics, and bill of materials will be available for reference and inspection.

Test Results

WCP22 has performed the following in fulfillment of the above requirements.

It was not necessary to transition from one logic level to another. WCP22 has purchased all components new from Unicorn Electronics and Digi-Key. All were readily available.

Amplifier PCB

The Amplifier PCB will be tested for pulse detection in fulfillment of requirement WCP22-2-5

Pulse Detection Test

Test Level	Sub-System
Test Type or Class	Input Detection
Qualification Method(s)	Analysis
System Requirement(s)	WCP22-2-5

The pulse detection test will be performed at the Watson Capstone Labs. This test will verify the system's ability to recognize the feedback pulses received from the IBM1403-N1 printer. A signal generator will be used to create a sinusoid with amplitude of 5 to 15 mv. The output of the Amplifier PCB will be monitored with an oscilloscope to verify the signal has been collected, amplified, and digitized. The microcontroller will verify receiving the digitized pulses using LEDs.

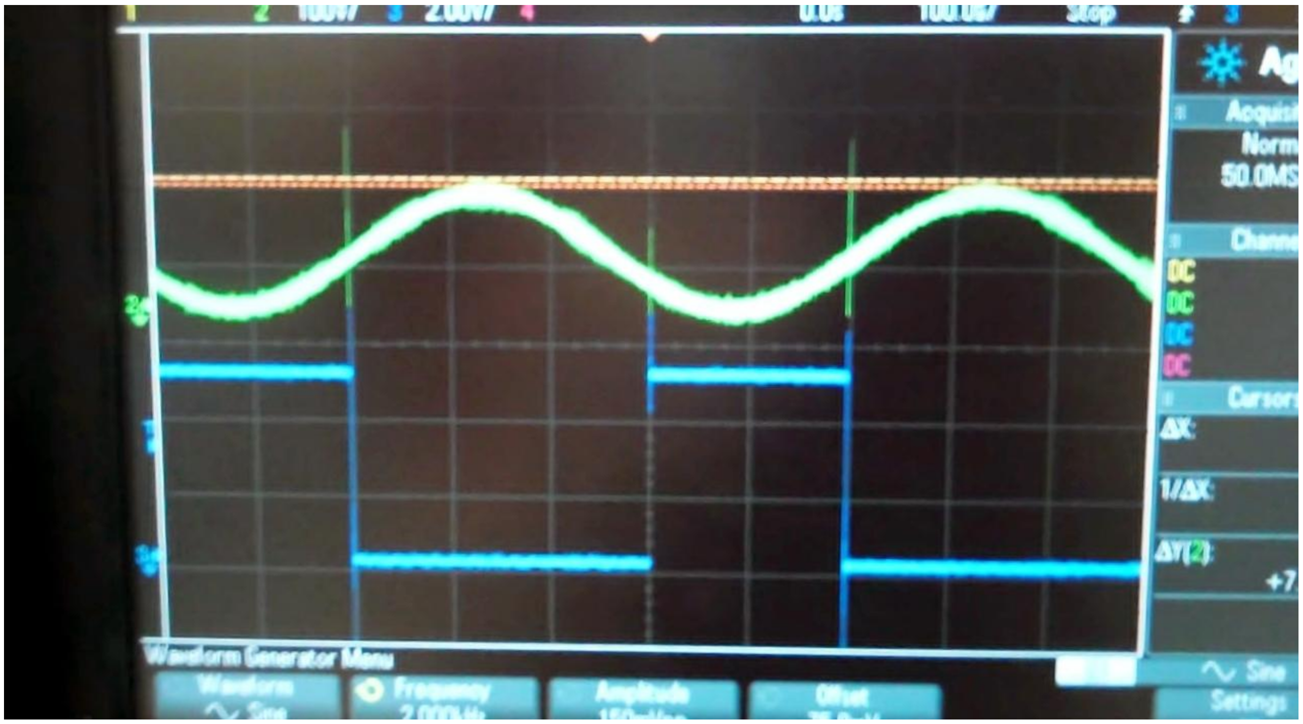
Test Results

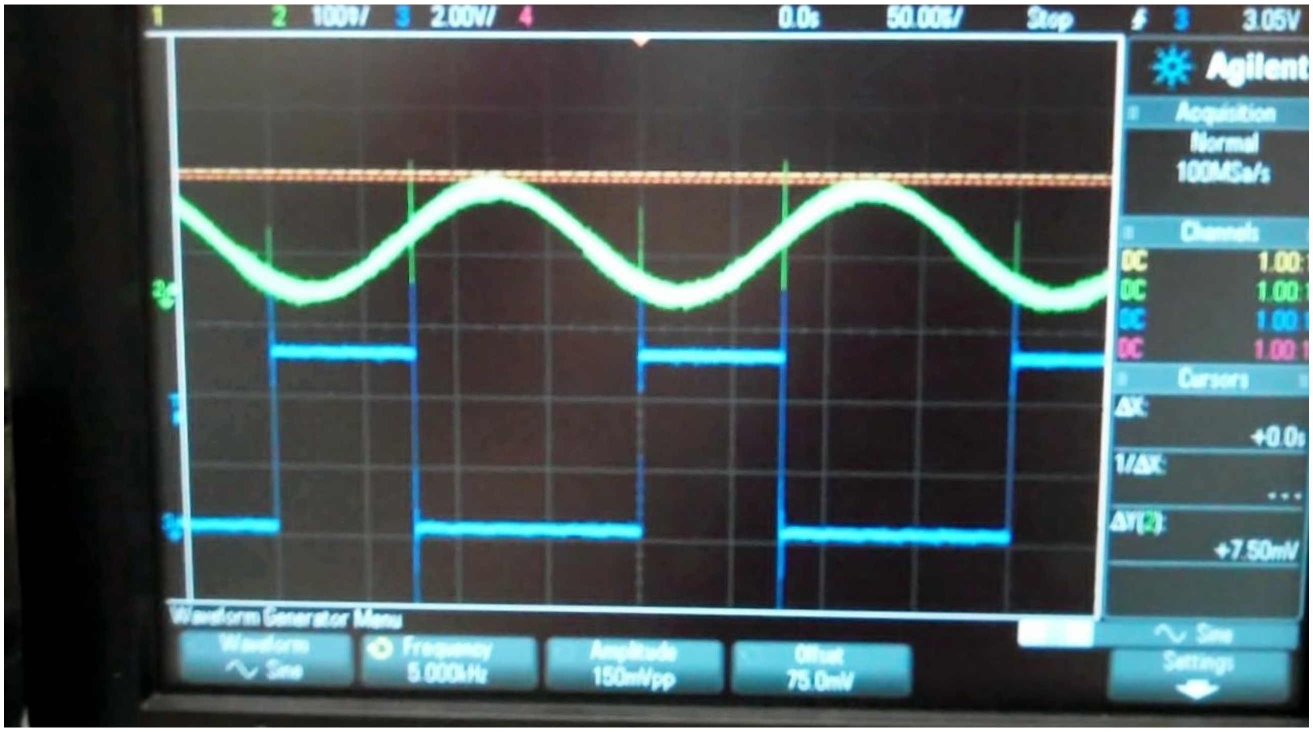
Test Date: 04/28/14
Location: Watson Capstone Labs

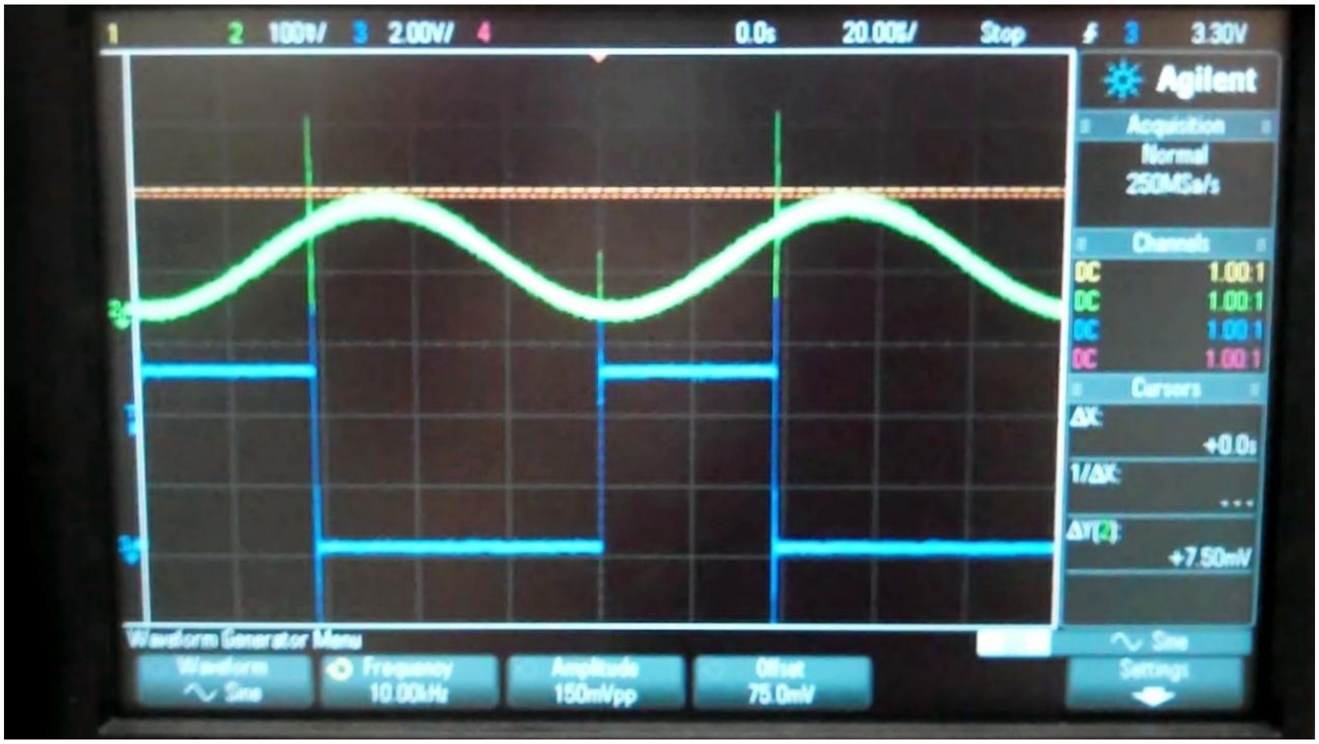
Test Date: 04/30/14
Location: Watson Capstone Labs

Observations:
Sync Pulses appear Active-Low instead of Active-High as expected.
By sending the PSS and UCS signals through the AND gate (AND "1") the output signals have been stablized.
Capacitors were added to the supply rails of the LM741 op-amps.
The AND function must now be performed in software, or a NAND gate must replace the AND.
AND was sucessfully implemented in software, system is now properly interpreting PSS, UCS, and HOME.

Three images provided are tests performed at 2 kHz, 5 kHz, and 10 kHz
respectively.
For frequencies greater than 10 kHz output signal begins to skrink in pulse width.




1
2 100V/
3 2.00V/
4
0.0s
50.00$/
Stop
3 3.05V
Agilent
Acquisition
Normal
100MSa/s
Channels
DC 1.00:1
DC 1.00:1
DC 1.00:1
DC 1.00:1
Cursors
ΔX: +0.0s
1/ΔX: ---
ΔY(2): +7.50mV
Waveform Generator Menu
Waveform Sine
Frequency 5.000kHz
Amplitude 150mVpp
Offset 75.0mV
Sine Settings


1
2 100V/
3 2.00V/
4
0.0s
20.00$/
Stop
3 3.30V
Agilent
Acquisition
Normal
250MSa/s
Channels
DC 1.00:1
DC 1.00:1
DC 1.00:1
DC 1.00:1
Cursors
ΔX: +0.0s
1/ΔX: ---
ΔY(2): +7.50mV
Waveform Generator Menu
Waveform Sine
Frequency 10.00kHz
Amplitude 150mVpp
Offset 75.0mV
Sine Settings

Amplifier PCB Inspection

Test Level	Sub-System
Test Type or Class	Hardware
Qualification Method(s)	Inspection
System Requirement(s)	WCP22-2-2 , WCP22-2-6

The Amplifier PCB will be inspected by WCP22 and CT&I experts to ensure logic levels have been transitioned safely and efficiently and the Amplifier PCB has been produced using modern components. The system, circuit schematics, and bill of materials will be available for reference and inspection.

Test Results

WCP22 has performed the following in fulfillment of the above requirements.

It was not necessary to transition from one logic level to another. WCP22 has purchased all components new from Unicorn Electronics and Digi-Key. All were readily available.

Microcontroller

The microcontroller hardware will be inspected in fulfillment of requirement WCP22-10-2

Microcontroller Demonstration

Test Level	Sub-System
Test Type or Class	Software
Qualification Method(s)	Demonstration
System Requirement(s)	WCP22-2-3

WCP22 will demonstrate for CT&I experts and/or Binghamton University officials the ability of the microcontroller to accept a print job from an external source.

Test Results

Microcontroller Inspection

Test Level	Sub-System

Test Type or Class	Hardware I/O
Qualification Method(s)	Inspection
System Requirement(s)	WCP22-10-2

CT&I experts will inspect the microcontroller I/O to verify there are enough outputs to expand the system later and control all 132 print hammers.

Test Results

The microcontroller was reviewed by CT&I before the purchase at a weekly meeting during the fall semester and was approved for purchase by the external and technical advisors.

Full System

Safety Analysis

The full system will be analyzed by experts at CT&I, and/or Binghamton University to ensure a safety level appropriate for display and demonstration at TechWorks!.

Test Level	System
Test Type or Class	Safety
Qualification Method(s)	Analysis
System Requirement(s)	WCP22-7-1

Test Results

WCP22 has performed the following in fulfillment of the above requirements.

A clear plastic box has been constructed to house the system. Rubber washers were used to hold the printed circuit boards on stand-offs. High Voltage inputs have been clearly labeled and positioned inside the unit where they are connected using large terminal blocks. A switch has been provided to activate a relay which will provide 60 volts to the WCP22 system. The relay will also throw a contactor which will activate three phase power to the IBM1403-N1 printer. While high-voltage is present the WCP22 system is not to be handled.

Documentation

Code and Documentation Inspection

The code will be inspected by CT&I to verify it is written in C programming language, and to ensure clarity in the commenting of unclear portions for code. Documentation will be inspected to ensure the delivery of a software design document, and detailed design documentation for future expansion of the project.

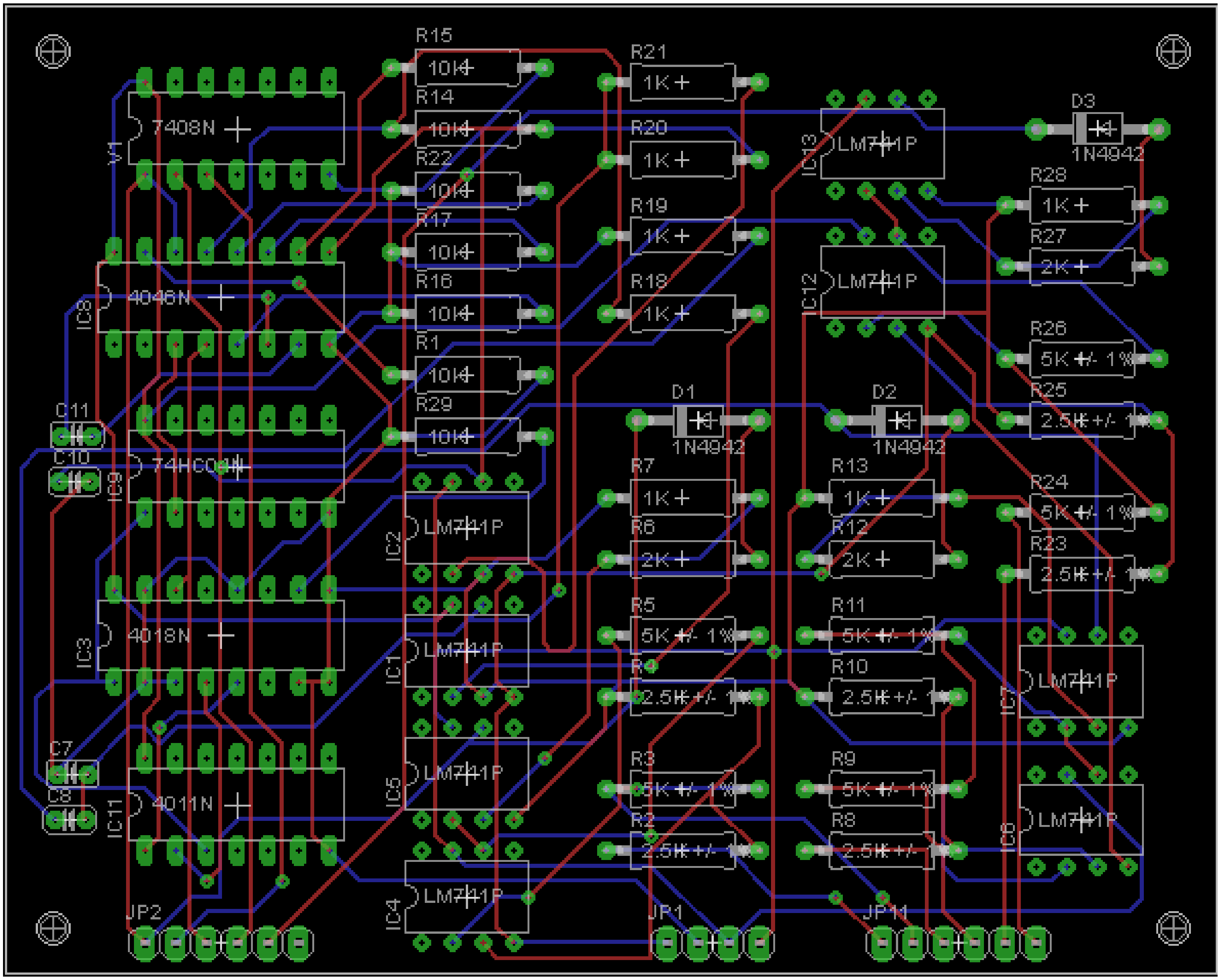
Test Level	System
Test Type or Class	Documentation
Qualification Method(s)	Analysis
System Requirement(s)	WCP22-2-7 , WCP22-2-8 , WCP22-10-1 , WCP22-14-1

Test Results

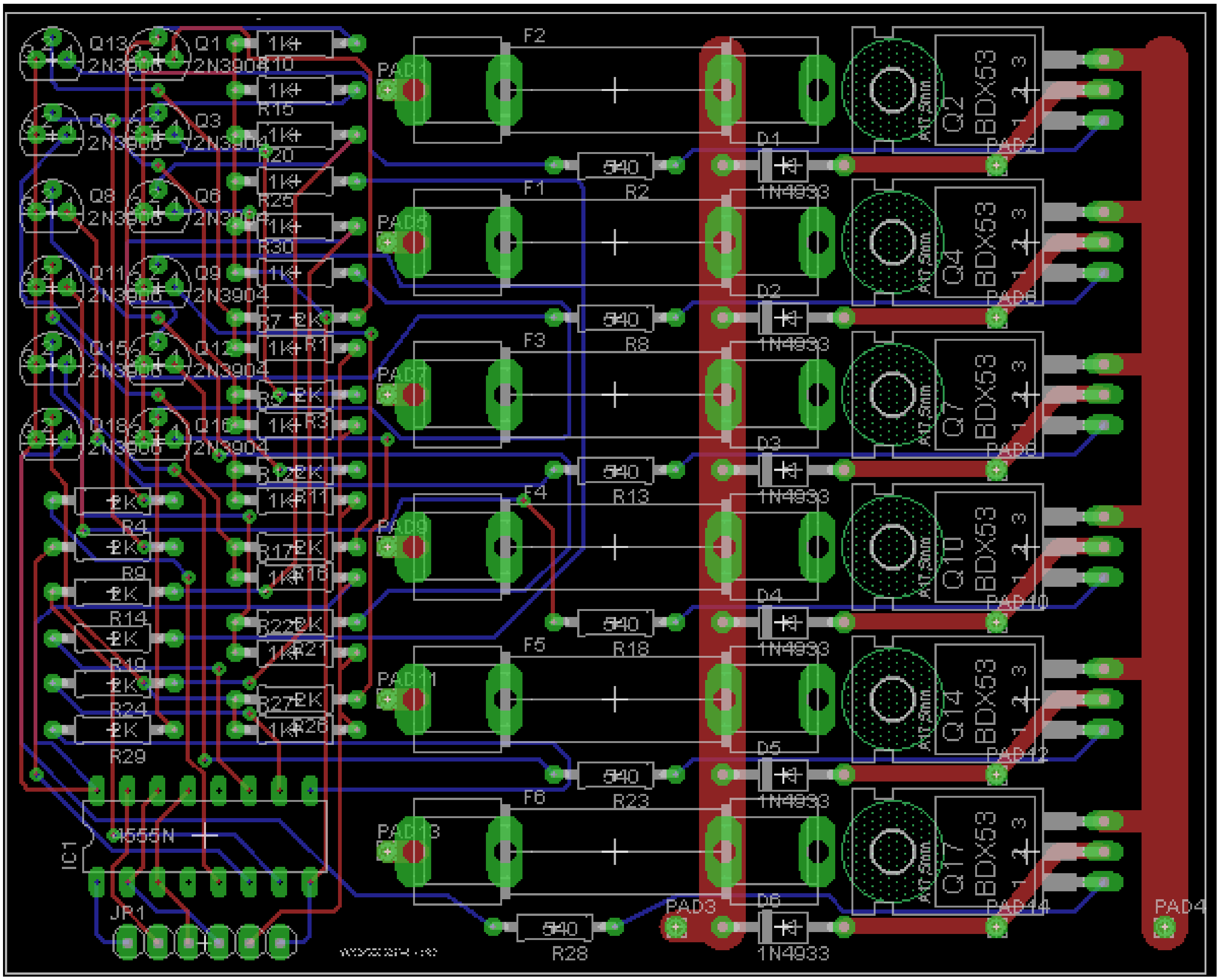
WCP22 has performed the following in fulfillment of the above requirements.

WCP22 has delivered well commented source code written in C programming language for use with our Arduino compatible microcontroller. A software design document and a user manual have also been delivered. The Center for Technology and Innovation will be given a chance to review the material, and request changes if necessary.

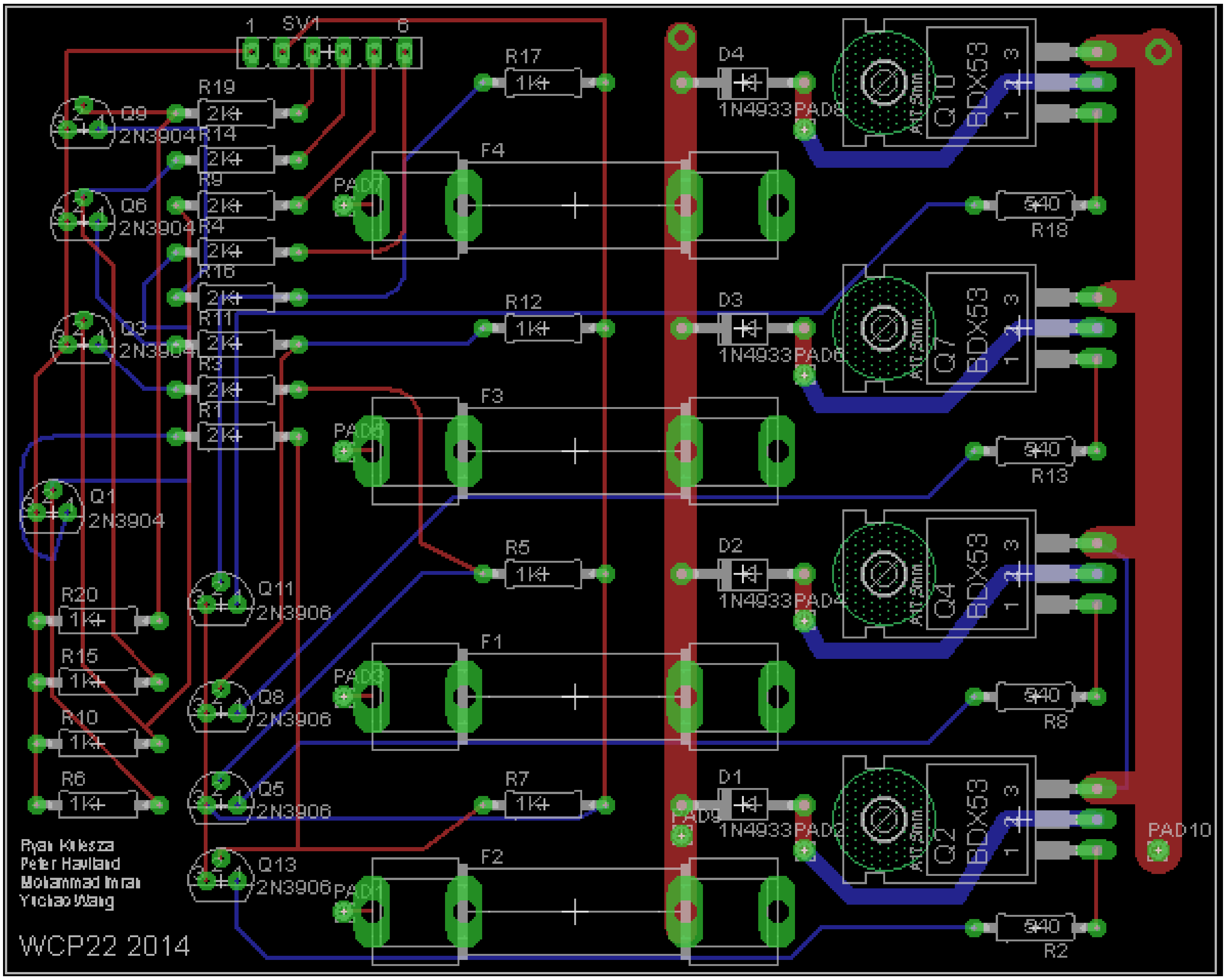
Amplifier PCB Layout



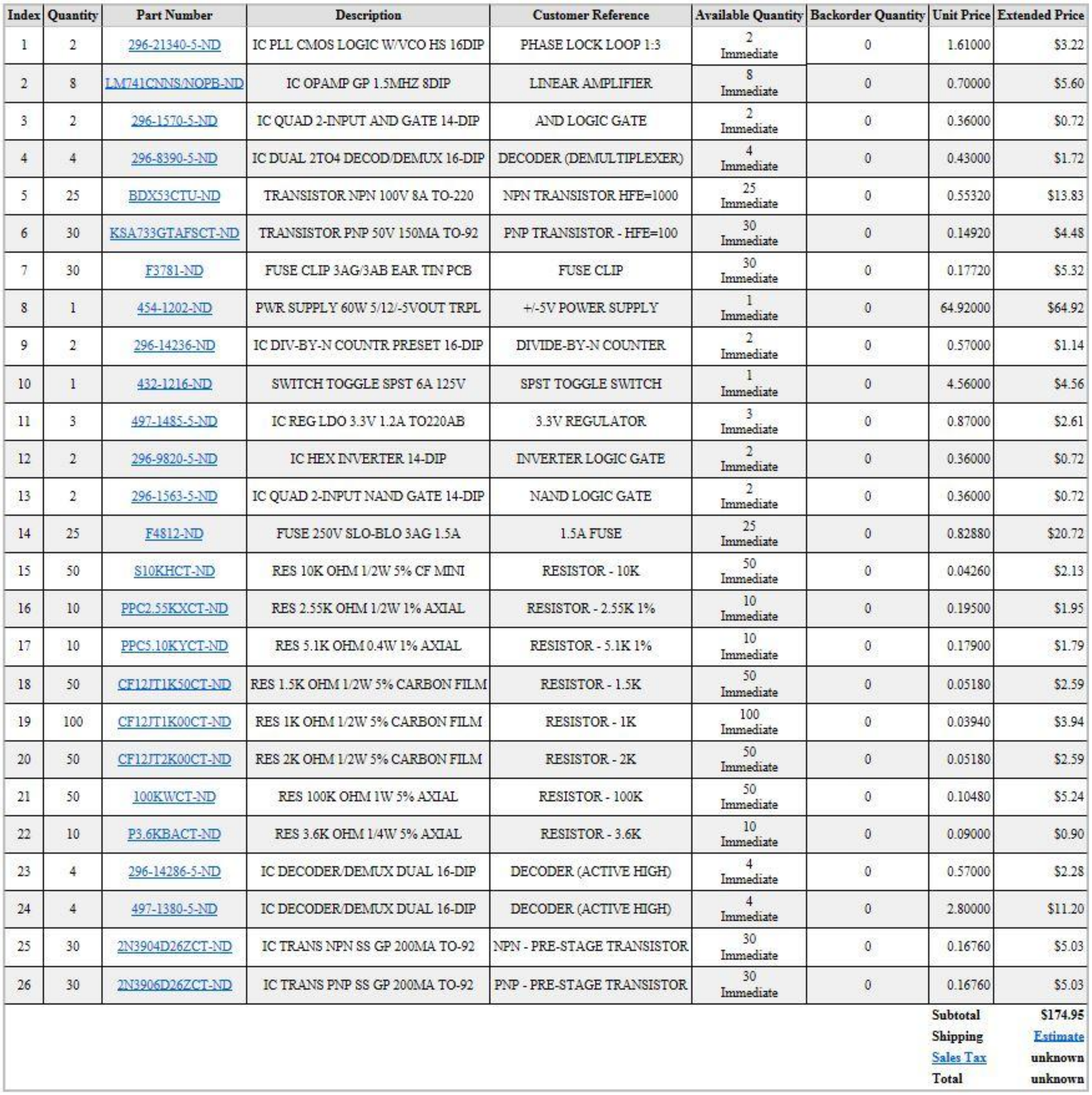
Hammer Driver PCB Layout



Carriage Driver PCB Layout



Electronic Components – Bill of Materials



Index	Quantity	Part Number	Description	Customer Reference	Available Quantity	Backorder Quantity	Unit Price	Extended Price
1	2	296-21340-5-ND	IC PLL CMOS LOGIC W/VCO HS 16DIP	PHASE LOCK LOOP 1:3	2 Immediate	0	1.61000	$3.22
2	8	LM741CNNS/NOPB-ND	IC OPAMP GP 1.5MHZ 8DIP	LINEAR AMPLIFIER	8 Immediate	0	0.70000	$5.60
3	2	296-1570-5-ND	IC QUAD 2-INPUT AND GATE 14-DIP	AND LOGIC GATE	2 Immediate	0	0.36000	$0.72
4	4	296-8390-5-ND	IC DUAL 2TO4 DECOD/DEMUX 16-DIP	DECODER (DEMULTIPLEXER)	4 Immediate	0	0.43000	$1.72
5	25	BDX53CTU-ND	TRANSISTOR NPN 100V 8A TO-220	NPN TRANSISTOR HFE=1000	25 Immediate	0	0.55320	$13.83
6	30	KSA733GTAFSCT-ND	TRANSISTOR PNP 50V 150MA TO-92	PNP TRANSISTOR - HFE=100	30 Immediate	0	0.14920	$4.48
7	30	F3781-ND	FUSE CLIP 3AG/3AB EAR TIN PCB	FUSE CLIP	30 Immediate	0	0.17720	$5.32
8	1	454-1202-ND	PWR SUPPLY 60W 5/12/-5VOUT TRPL	+/-5V POWER SUPPLY	1 Immediate	0	64.92000	$64.92
9	2	296-14236-ND	IC DIV-BY-N COUNTR PRESET 16-DIP	DIVIDE-BY-N COUNTER	2 Immediate	0	0.57000	$1.14
10	1	432-1216-ND	SWITCH TOGGLE SPST 6A 125V	SPST TOGGLE SWITCH	1 Immediate	0	4.56000	$4.56
11	3	497-1485-5-ND	IC REG LDO 3.3V 1.2A TO220AB	3.3V REGULATOR	3 Immediate	0	0.87000	$2.61
12	2	296-9820-5-ND	IC HEX INVERTER 14-DIP	INVERTER LOGIC GATE	2 Immediate	0	0.36000	$0.72
13	2	296-1563-5-ND	IC QUAD 2-INPUT NAND GATE 14-DIP	NAND LOGIC GATE	2 Immediate	0	0.36000	$0.72
14	25	F4812-ND	FUSE 250V SLO-BLO 3AG 1.5A	1.5A FUSE	25 Immediate	0	0.82880	$20.72
15	50	S10KHCT-ND	RES 10K OHM 1/2W 5% CF MINI	RESISTOR - 10K	50 Immediate	0	0.04260	$2.13
16	10	PPC2.55KXCT-ND	RES 2.55K OHM 1/2W 1% AXIAL	RESISTOR - 2.55K 1%	10 Immediate	0	0.19500	$1.95
17	10	PPC5.10KYCT-ND	RES 5.1K OHM 0.4W 1% AXIAL	RESISTOR - 5.1K 1%	10 Immediate	0	0.17900	$1.79
18	50	CF12JT1K50CT-ND	RES 1.5K OHM 1/2W 5% CARBON FILM	RESISTOR - 1.5K	50 Immediate	0	0.05180	$2.59
19	100	CF12JT1K00CT-ND	RES 1K OHM 1/2W 5% CARBON FILM	RESISTOR - 1K	100 Immediate	0	0.03940	$3.94
20	50	CF12JT2K00CT-ND	RES 2K OHM 1/2W 5% CARBON FILM	RESISTOR - 2K	50 Immediate	0	0.05180	$2.59
21	50	100KWCT-ND	RES 100K OHM 1W 5% AXIAL	RESISTOR - 100K	50 Immediate	0	0.10480	$5.24
22	10	P3.6KBACT-ND	RES 3.6K OHM 1/4W 5% AXIAL	RESISTOR - 3.6K	10 Immediate	0	0.09000	$0.90
23	4	296-14286-5-ND	IC DECODER/DEMUX DUAL 16-DIP	DECODER (ACTIVE HIGH)	4 Immediate	0	0.57000	$2.28
24	4	497-1380-5-ND	IC DECODER/DEMUX DUAL 16-DIP	DECODER (ACTIVE HIGH)	4 Immediate	0	2.80000	$11.20
25	30	2N3904D26ZCT-ND	IC TRANS NPN SS GP 200MA TO-92	NPN - PRE-STAGE TRANSISTOR	30 Immediate	0	0.16760	$5.03
26	30	2N3906D26ZCT-ND	IC TRANS PNP SS GP 200MA TO-92	PNP - PRE-STAGE TRANSISTOR	30 Immediate	0	0.16760	$5.03
							Subtotal	$174.95
							Shipping	Estimate
							Sales Tax	unknown
							Total	unknown

Character Transfer Module

```
//*****************************************//
//                                        //
//  WCP22: IBM 1403-N1 Printer Interface  //
//    Ryan Kulesza, Peter Haviland        //
//    Mohammad Imran, Yuchao Wang         //
//    Date: May 1, 2014                   //
//                                        //
//*****************************************//

/*
Operation:
This program will read from a text file (predetermined path).
The program will send one character at a time.
The program will send the entire file to the microcontroller.
*/
import processing.serial.*;
import java.io.*;
import java.util.*;
Serial port;

int position = 0; //tracks position in stringlist
StringList data = new StringList(); //holds the entire textfile
File readFile = new File("C:/filename.txt"); //filename and directory
BufferedReader reader = null; //reads from the textfile
String text = null; //stores line
int enableRead = 1; //toggles read

void setup()
{
  port = new Serial(this, "COM3", 9600);
  delay(5000); //need delay or microcontroller will miss first few characters
}

void draw()
{
 if(enableRead==1)
 {
  //==========Storing the Text File to an Array==========//
  try
  {

    reader = new BufferedReader(new FileReader(readFile));
    while((text=reader.readLine())!=null)
    {
     data.append(text);
    }

  }
  catch(FileNotFoundException instance)
  {
    instance.printStackTrace();
  }
  catch(IOException instance)
```

```
    {
      instance.printStackTrace();
    }
    finally
    {
      try
      {
        if (reader != null)
        {
          reader.close();
        }
      }
      catch (IOException instance)
      {
        instance.printStackTrace();
      }
    }
    enableRead = 0;
  }

    if(position<data.size() && data.get(position).length()!=0)
    {
      print("Sending: ");
      println(data.get(position));
      port.write(data.get(position)); //send line of print data

    }
    position++; //move to next position in print data

    if(position>=data.size())
    {
      println("Program Complete.");
      if(data.size() == 0) //test if print data was empty
        println("Warning: The print data is empty.");
      else
        println("The print data has been succesfully sent to the microcontroller.");
      port.write('\r'); //write end of file character
      noLoop();
    }
    else
    {
      println("Sending: (New Line)");
      port.write("\n"); //write newline character
    }
}
```

IBM 1403-N1 Printer Simulation

```
//******************************************//
//                                  //
//   WCP22: IBM 1403-N1 Printer Interface    //
//     Ryan Kulesza, Peter Haviland          //
//     Mohammad Imran, Yuchao Wang           //
//     Date: May 1, 2014                    //
//                                  //
//******************************************//

void setup() {
  // initialize the digital pin as an output.
  pinMode(7, OUTPUT); //Drive Pulse Output
  pinMode(6, OUTPUT); //UCS Pulse Output
  digitalWrite(6, LOW);
  digitalWrite(7, HIGH);
}

//This loop will generate a UCS and Drive Pulse
//Simulated from an IBM 1403-N1 Printer
void loop() {

  //current timing delays take into account delay from
  //instructions as run on the Arduino UNO
  for(int counter=0; counter<=720; counter++)
  {
    digitalWrite(7, HIGH);
    delayMicroseconds(119); //~112 half period
    digitalWrite(7, LOW);
    if(counter%144==0 && counter%120 == 0 && counter!=0) //train home pulse
    {
      digitalWrite(6, HIGH);
      delayMicroseconds(20);
      digitalWrite(7, HIGH);
      delayMicroseconds(27);
      digitalWrite(7, LOW);
      delayMicroseconds(21);
      digitalWrite(6, LOW);

      counter = 0;
    }
    else if(counter%144==0 && counter!=0) //extra pulse
    {

      delayMicroseconds(21);
      digitalWrite(7, HIGH);
      delayMicroseconds(27);
      digitalWrite(7, LOW);
      delayMicroseconds(27);
    }
    else if(counter%120==0 && counter!=0) //ucs pulse
    {
      digitalWrite(6, HIGH);
      delayMicroseconds(72);
      digitalWrite(6, LOW);
    delayMicroseconds(5);
   }
   else
    delayMicroseconds(83); //~112 half period
  }

}
```

IBM1403-N1 Printer Visual Simulation

```
/*This program simulates the IBM1403-N1 printer
for the printing of one 12 character maximum line.
This program includes updating graphics to follow
the movement of the character belt. When alignment
occurs, straight bars appear around the printer
hammer.*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>

char printerCartridge[] = "ZYXWVUTS/'$=RQP-,0987654321*.+IHGFEDCBAONM%$&-
,0987654321LKJZYXWVUTS/($&RQP-,0987654321*.+IHGFEDCBAONMLKJ-
,0987654321*.)ZYXWVUTS/@$#RQP-,0987654321*.+IHGFEDCBAONMLKJ-
,0987654321*.)ZYXWVUTS/($&RQP-,0987654321*.+IHGFEDCBAONMLKJ-,0987654321*.)";
char stringToPrint[] = "HELLO WORLD.";
int printed[12] = {0};

int main()
{
        int counter;
        char dummy;
        int startingPosition=0;
        int iterator=0;
        int PSS=1;
        int stringCounter;

        /*simulates only the first rotation of the belt*/
        while(startingPosition<240)
        {
                /*determines starting position of hammers*/
                stringCounter=iterator;
                if(iterator==2)
                        counter=startingPosition+1;
                else
                        counter=startingPosition;

                /*loops for first line*/
                while(counter<(startingPosition+8))
                {
                        /*compares to see if proper alignment occurs*/
                        if(printerCartridge[counter] == stringToPrint[stringCounter])
                        {
                                /*simulates firing hammer*/
                                printed[stringCounter] = 1;
                        }

                        /*increment the character and hammer counters*/
                        stringCounter+=3;
                        counter+=2;
                }

                /*below builds the graphical layout for the simulation*/
                printf("Printed:  ");
```

```
		for(counter=0; counter<12; counter++)
		{
			if(counter != 0)
				printf("     ");

			if(printed[counter] == 0)
				printf("_");
			else
				printf("%c", stringToPrint[counter]);
		}
		printf("\n\n");
		if(iterator == 0)
		{
			printf("Hammers:
|1|____2_____3____|4|____5_____6____|7|____8_____9____|10|___11____12\n");
			printf("PSS %3d:  ", PSS);
		}
		if(iterator == 1)
		{
			printf("Hammers:
1____|2|____3_____4____|5|____6_____7____|8|____9_____10___|11|___12\n");
			printf("PSS %3d:  ", PSS);
			printf("      ");
		}
		if(iterator == 2)
		{
			printf("Hammers:
1_____2____|3|____4_____5____|6|____7_____8____|9|____10____11___|12|\n");
			printf("PSS %3d:  ", PSS);
			printf("   ");
		}

		/*determines spacing of character belt print out*/
		for(counter=startingPosition; counter<(startingPosition+8); counter++)
		{
			if(iterator == 0)
				printf("%c        ", printerCartridge[counter]);
			else if(iterator == 1)
				printf("%c        ", printerCartridge[counter]);
			else if(iterator == 2)
				printf("%c        ", printerCartridge[counter]);
		}

		PSS++;

		if(iterator==0)
			startingPosition++;

		iterator++;
		if(iterator==3)
			iterator=0;

		/*pauses for user input*/
		dummy = getchar();
		printf("\n\n");
	}
	printf("\n");
	return 0;
}
```

Printing Algorithm with Descriptions

```
/*This program is the printer algorithm
with complete descriptions of the
print mechanism for corresponding
PSS pulses*/
#include <stdio.h>
#include <stdlib.h>

/*character belt*/
char cartridge[] = "ZYXWVUTS/'$=RQP-,0987654321*.+IHGFEDCBAONM%$&-
,0987654321LKJZYXWVUTS/($&RQP-,0987654321*.+IHGFEDCBAONMLKJ-
,0987654321*.)ZYXWVUTS/@$#RQP-,0987654321*.+IHGFEDCBAONMLKJ-
,0987654321*.)ZYXWVUTS/($&RQP-,0987654321*.+IHGFEDCBAONMLKJ-,0987654321*.)";
/*buffer of line to print*/
char *buffer;
/*tracks progress of printing*/
char *isPrinted;

int charCounter = 0;

int simulateRead(FILE *file);
FILE *simulateOpenFile();
void simulateCloseFile(FILE *file);
void determineLineRange();
int linePrinted(int start, int end);

int main()
{
        /*subscan pulses from printer*/
   int emitterPulse = 1;

        /*character belt trackers*/
   int charBeltCounter = 0;
   int charBeltPosition = 0;

        /*printer hammer trackers*/
        int offsetCounter = 0;
        int bufferCounterPosition = 0;
   int bufferCounter = 0;

        /*lines for carriage control*/
        int newLineCounter = 0;

        /*start and end of line in buffer*/
   int lineStart = 0;
   int lineEnd = 0;

   int isEOF = 0;
   int flag = 0;

        /*begin of simulate reading from PC*/
   FILE *simulateFile = simulateOpenFile();

   if(simulateFile==NULL)
   {
```

```
  printf("Simulation Error. This file could not be opened.\n");
  return 0;
}

int stillReading = 1;
while(stillReading == 1){
stillReading = simulateRead(simulateFile);
}
     /*end of simulate reading from PC*/


     printf("Order of hammer firings and line spacing:\n");

/*simulate microcontroller loop*/
while(isEOF==0)
{
            newLineCounter = 0;

            /*start finding start and end location inside buffer*/
            if(flag==1 || buffer[lineStart]=='\n')
            {
    flag = 0;
                   /*looking for start location*/
    while(true)
    {
      if(buffer[lineStart] == '\n' || (int)buffer[lineStart] == -1)
      {
                              newLineCounter++;
                              if(flag == 0)
                                     flag = 1;
      }

      if(buffer[lineStart] != '\n' && flag == 1)
      {
        break;
      }
      lineStart++;
    }
            }

  flag = 1;
  lineEnd = lineStart;
            /*looking for end location*/
  while(true)
  {
                   if(buffer[lineEnd] == '\n')
    {
                          lineEnd--;
      break;
    }
    if(lineEnd == charCounter)
    {
                          isEOF = 1;
      lineEnd--;
      break;
    }
```

```
        lineEnd++;
                }
                /*finished finding start and end location in buffer*/

                printf("\n%d New Line(s)\n\n", newLineCounter);

                /*reset variables for new line*/
                bufferCounter = lineStart;
                charBeltCounter=0;
                bufferCounterPosition=0;
                charBeltPosition=0;
                emitterPulse=1;

                /*printing algorithm performed here*/

                /*loops until line fully printed*/
                while(linePrinted(lineStart,lineEnd)==0)
                {
                        bufferCounterPosition = bufferCounter;
        charBeltPosition = charBeltCounter;

                        /*loops until reaches end of line*/
                        while(bufferCounter<=lineEnd)
        {
                                /*checks if correct hammer aligned to correct character*/
                                if(buffer[bufferCounter] == cartridge[charBeltCounter] &&
isPrinted[bufferCounter] == 'x')
            {
                                        printf("On PSS pulse %d, hammer %d will be fired with a
%.2fus delay, to print %c\n", emitterPulse, bufferCounter-lineStart+1,
offsetCounter*4.85,buffer[bufferCounter]);
              /*signifies character printed*/
                                        isPrinted[bufferCounter] = 'o';
            }

                                if(buffer[bufferCounter]==' ')
                                        isPrinted[bufferCounter] = 'o';

                                /*used for timing delay of hammers*/
                                offsetCounter++;

                                /*go to the next aligned character and hammer*/
            bufferCounter=bufferCounter+3;
            charBeltCounter=charBeltCounter+2;

                                /*implements circular array for character belt*/
                                if(charBeltCounter>=240)
                                {
                                        charBeltCounter=charBeltCounter-240;
                                }
                        }

                        /*reset timing delay for hammers*/
                        offsetCounter = 0;
                        /*reset hammer back to left most aligned position*/
                        bufferCounter = bufferCounterPosition;
                        /*move to the next aligned hammer for next pulse*/
                        bufferCounter++;

                        /*reset hammer alignment for next 3 PSS pulses*/
        if(bufferCounter==lineStart+3)
                                bufferCounter=lineStart;

                        /*reset character back to left most aligned position*/
                        charBeltCounter = charBeltPosition;

                        /*move to the next aligned character for next pulse*/
                        if(emitterPulse%3==0)
                                charBeltCounter=charBeltCounter-1;
                        else
                                charBeltCounter++;

                        /*implements circular array for character belt*/
                        if(charBeltCounter >= 240)
                                charBeltCounter = charBeltCounter-240;

                        /*goes to next PSS pulse*/
                        emitterPulse++;
                }
        }

        simulateCloseFile(simulateFile);
        return 0;
}



/*for simulation purposes only*/
/*will simulate reading a file from PC*/
int simulateRead(FILE *file)
{

  /*string to write*/
  char readChar;

  /*loops through file*/
  while((int)((readChar = getc(file)) != -1))
  {
    charCounter++;

                /*dynamically allocates memory for character*/
    buffer = (char*)realloc(buffer,charCounter);
    isPrinted = (char*)realloc(isPrinted,charCounter);

    isPrinted[charCounter-1] = 'x';
    buffer[charCounter-1] = readChar;

    if(readChar == '\n')
      return 1;
  }
```

```
    return 0;
}

/*for simulation purposes only*/
/*will simulate opening a file on PC*/
FILE *simulateOpenFile()
{
    char *fileReadName = "toPrint.txt";
    /*file to write to*/
    FILE *file;
    /*opens file pointer to write*/
    file = fopen(fileReadName, "r");

    return file;
}

/*for simulation purposes only*/
/*will simulate closing a file on PC*/
void simulateCloseFile(FILE *file)
{
    fclose(file);
}

/*tracks progress of printed line*/
/*0 means not fully printed, 1 fully printed*/
int linePrinted(int start, int end)
{
    int counter;
    for(counter=start; counter<=end; counter++)
    {
        if(isPrinted[counter] == 'x')
            return 0;
    }
    return 1;
}
```