IBM 1403-N1 Printer Controller
Design Report

Watson Capstone Project WCP07
Sponsor: IEEE Binghamton Section
2015-05-01
Revision: -

Submitted by:
Nicholas Hekman, Lead, EE
Alena Yampolskaya, COE
John Wiseman, COE

Faculty Advisor: Professor Jack Maynard
External Advisor: Tommy Lam
Client Advisor: Arthur Law
Program Manager: Professor Jack Maynard

Submitted in partial fulfillment of EECE 487-488 / ME 493-494 requirements.

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

Executive Summary

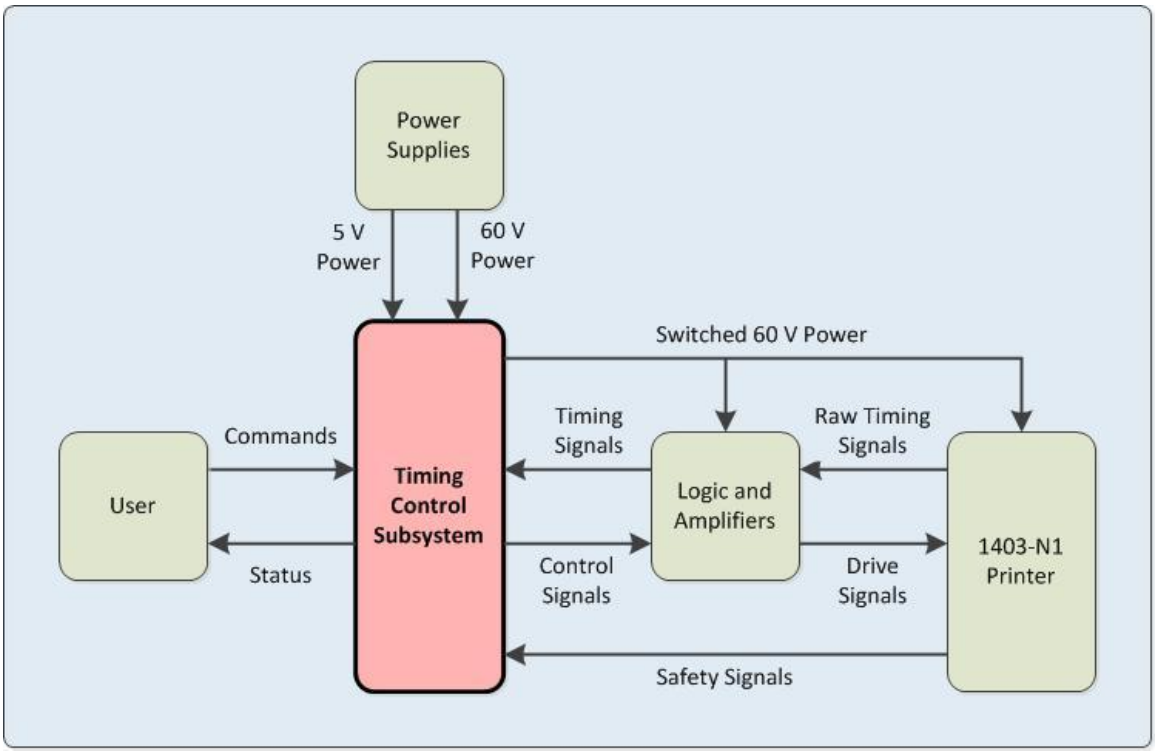


Figure 1: Context Diagram

The IBM 1403-N1 line printer was introduced in October 1959, and could print up to 1100 lines per minute (LPM). The Center for Technology and Innovation (CT&I), located at 321 Water Street in Binghamton New York, has obtained a 1403-N1 printer and begun restoration. CT&I intends to bring the printer to working order and gain the ability to print. The necessary control components for the printer are not in place, so alternatives are being developed. This was the goal of a previous Watson Capstone Project (WCP) effort, and was the goal of this team's effort.

The printer utilizes 132 solenoid-driven hammers, which press paper into a spinning character train. Paper is advanced using a hydraulic motor and tractor, which pull continuous form paper between the hammers and character train. Hardware amplifiers were developed during the previous WCP effort to interface with all inputs and outputs of the printer. CT&I has enlisted the help of Triple Cities Makerspace (TCM) in scaling up and implementing the hardware solution WCP developed. TCM has modified the driver board design to incorporate logic components (see figure 1 above.) The redesigned hardware will accept timing and sync information from the Timing Control Subsystem and use that information to drive the printer solenoids.

The Institute of Electrical and Electronics Engineers (IEEE) Binghamton section has sponsored a WCP effort to produce a Timing Control Subsystem. The subsystem is responsible for determining which hammer must be fired and at what time. This information must be communicated to the hammer driver boards. The subsystem needs to accept user input specifying the data to print. The subsystem is also responsible for interruption of 60V printer power in the case of any error.

Table of Contents

List of Figures

List of Tables

1 Problem Definition

The 1403 printer that the CT&I has is currently unable to print. Rather than abandon this historic device, the CT&I has decided to undergo a restoration process that will bring this printer to working order. Engineers from CT&I, Triple Cities Makerspace (TCM), and this WCP team are working together to accomplish this. This subsystem must be able to take print data as input from a user, and properly interface with the TCM logic Printed Circuit Boards (PCBs) in order to send information to the printer regarding which hammers to fire, and in what order.

The 1403-N1 originally required an additional rack of equipment, the IBM 2821 control unit, to interface with a computer. This device included a set of high power transistor switches and a control system to determine when to operate each switch. Each switch, when closed, would provide 60Vdc at 5A to a single hammer, which would print a single print position. These switches were referred to as "drivers" which contrasts with the modern definition of a printer driver: software which communicates with self-contained printer hardware.

By carefully obeying the timing requirements that the 2821 controller met, the team will be able to achieve the rated print speed of 1100 lines per minute. A previous WCP effort developed a printer controller prototype that showed it was possible to interface with the vintage printer using modern hardware controlled by modern software. The current WCP effort will work with TCM and CT&I to bring the printer to working order. WCP's prime responsibility is to work with the previous WCP effort's hardware and expand it to full capability.

1.1 Problem Scope

The two main pieces of this design are the timing control subsystem and the hardware safety interlock. The timing control board acts similarly to the printer driver in most modern computers; it will take input from a modern PC through a Host PC Application and figure out the output signals required in order to print it. It will send output signals to control the printer's hammers, and these signals must adhere to real time specifications in order to ensure correct printing as well as the safety of the printer. Spacing between lines of text is also controlled by this board. These signals go to the TCM logic driver cards. In order to ensure that the print timings are correct, the Timing Control board receives sync pulses from the printer (that are amplified by the TCM Amplifier Boards). These pulses ensure that the Timing Control board does not lose track of what it is printing. The hardware safety interlock uses information about the printer's signals to decide if a dangerous state exists and to stop power if that occurs.

1.2 Technical Review

Printing written text and pictures has been around for thousands of years. Whether this process has been done by hand, or with devices ranging from wooden blocks, to moveable type, to modern inkjet printers, humans have always wanted to write down information. Obviously, this gives us the ability to view information at a later time, with other people. Writing information down has allowed humans to record histories over generations.

Some of the earliest printing was done using a technique called woodblock printing. This method, used extensively in antiquity can be done by stamping a block onto paper or fabric, or by rubbing the medium against the block. Moveable type eventually became the most efficient method of

printing with Johannes Gutenberg, and the invention of the printing press. Such a device allowed humans to print the same thing over and over very quickly. If something else needed to be printed, it was as simple as rearranging the metal blocks to write different words. This allowed for the mass production of books, specifically the Bible, as someone no longer had to rewrite the entire thing by hand. The next major step in printing technology came in the late 18th century, with lithography. This process used chemicals to create pictures. With the invention of computers, new devices needed to be developed in order to create data and documents in the physical space. One such device was the 1403 line printer. Line printers were large devices that printed a single line of text at a time on one large piece of paper. These printers were capable of printing upwards of 1000 lines of text per minute and would be used by an entire company. With the majority of households having computers in the modern time, newer, smaller inkjet printers have been developed in order to allow anybody to print from their home computer.

1.3 Design Requirements

The requirements for this subsystem were sourced from a full-system client specification. The Host PC needs to be operable by someone familiar with a Windows style interface. The Host PC needs to be able to open a file, and send all of the text to print, including the ability to space up to three lines. Any sort of status information coming from the controller must also be displayed by the Host PC. Hammer firing order and timing is determined by the Printer Controller. This is important for the printer to output the proper text. If incorrect hammers are fired, or the correct hammers are fired at the wrong time, then incorrect data will be printed. In a full system test, the printer must be able to print up to 1100 lines of text per minute. This requirement is very important in proving that the modified printer is able to print as well as it could in its original form. This implies various restrictions, such as each line taking at least 55msec to print. A hardware safety subsystem is required to ensure the printer is unharmed by user error. Appendix A contains details about these requirements, as well as the remainder of requirements for this project.

2 Design Description

2.1 Overview

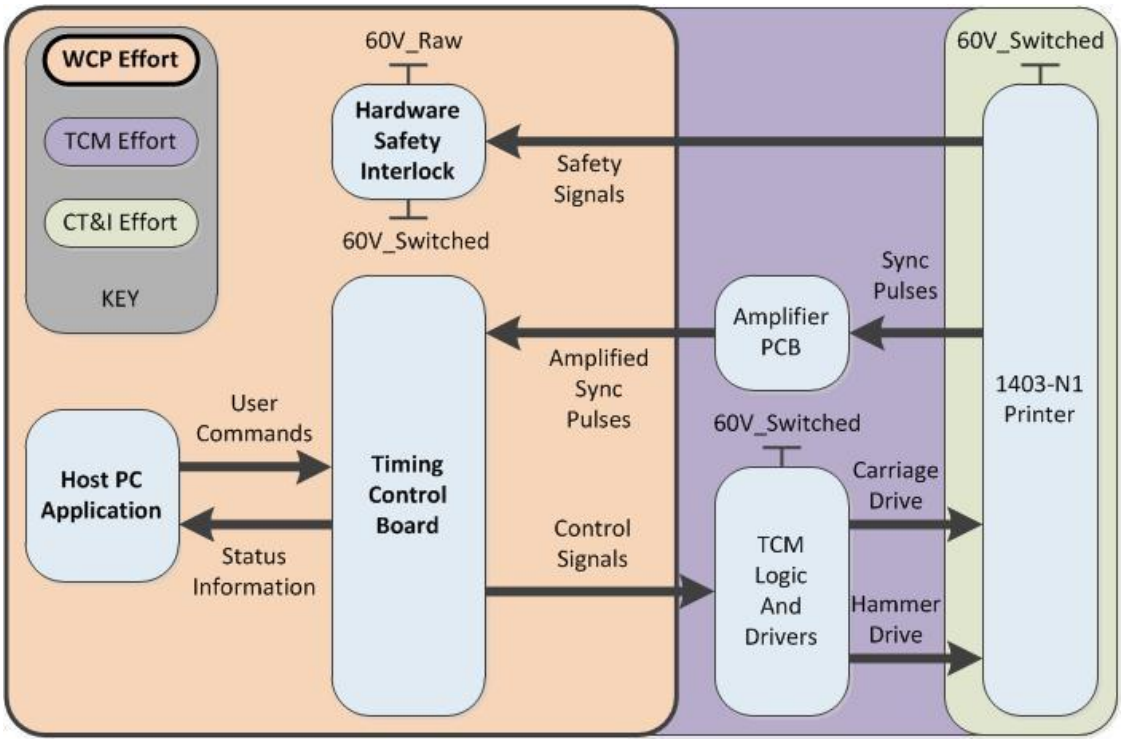


Figure 2: System Diagram

The timing control subsystem sends control signals to, and receives sync signals from, the hammer and carriage drivers developed by TCM. TCM is responsible for those physical interfaces, and for the delivery of sync pulses at Transistor-Transistor Logic (TTL) levels. The Timing Control Board will act as the interface with the TCM hardware, and will determine all hammer timing information. The Host PC application, running on a Windows XP PC, serves as a user interface to the system. An additional board, the Hardware Safety Interlock, removes system power in the case of any safety switch activation.

2.2 Detailed Description

2.2.1 Interfaces

The Host PC and Timing Control Board interface over USB 2.0, and all communication between them uses high speed serial. Messages are converted to a packet before being sent. Packets begin with the 16 bit size of the message and end with the Fletcher-16 checksum of the message. The commands are shown in appendix K.

The Timing Control Board interfaces with the driver boards using a set of 8 address lines and a single sync line. The address lines convey the hammer to fire, and the sync line is toggled at the moment the hammer needs to be fired. The carriage commands work in the same way. This parallel data transmission allows the TCB and drivers to communicate at the high speeds required. The address line controls are summarized in Table 1 below.

Table 1: Address Line Commands

Proposed Address Allocation	
Address Setting	Command
0xFF	Idle (perform no action)
0x08 - 0x8C	Fire the indicated hammer
0xF0	Toggle carriage command active
0xF1	Toggle carriage command idle
0x00	Reset all drivers

The Timing Control Board also receives signals from the amplifier board. The amplifier produces Pulse Sub-Scan (PSS) pulses and Universal Character Set (UCS) pulses, which are used to maintain synchronization with the printer. The printer uses a pair of drive gears to move the print train, and each of these gears generates a series of pulses during operation. The amplifier detects and amplifies these pulses so that they can be detected by the Timing Controller.

The Hardware Safety Interlock PCB interfaces with the printer in two ways. It controls 60V power to the printer using a contactor, and receives safety inputs from the printer. See the sections below for more information.

2.2.2 Host PC Application

The Host PC Application is written in Visual Basic .NET. It takes input from a file or allows users to type in their own requested string. The timing control software is set up to take input one 132-character line at a time, so the Host PC application splits the input stream or string into 132-character strings. The strings are sent to the timing control board over USB utilizing a UART. The Host PC either waits for a finished signal from the timing control software in order to send the next line.

The Host PC application has options available for different modes of operation, including different character trains, a debug mode that sends the information to be printed as the entire file and forces the timing control board to split it up and other important modes.

The Host PC application is also able to query the timing control board for status and fault information. This results in the timing control board dumping its debug file back to the Host PC.

2.2.3 Timing Control Board

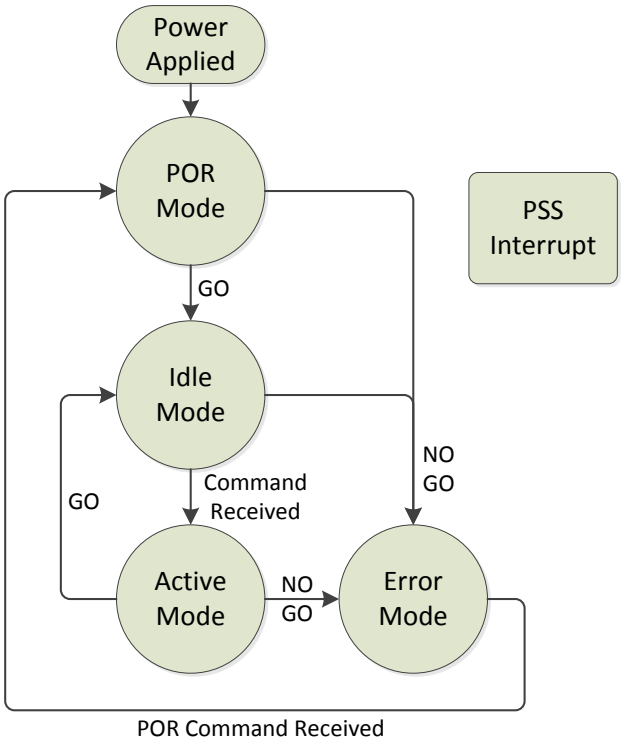


Figure 3: Required Modes

The Timing Control Board (TCB) interfaces with both the Host PC and the hammer/carriage drivers. It is responsible for receipt and interpretation of the signals coming from the printer, as well as generating signals to the driver cards based on that timing information. This device is essentially the core of the Timing Control Subsystem, with real-time software requirements.

Upon the application of system power, the TCB enters a Power On Reset (POR) mode and begins receiving PSS interrupts. POR mode brings the system into a working state, where all interfaces are properly running and initialized. The PSS interrupt will sync the system with the printer as described in section 2.2.3.5.

The TCB will need to await user commands from the Host PC, then act on them. As such, the device will spend the majority of its time in an "Idle" mode awaiting commands. When commands are received, the device will enter "Active" mode, where all the work is really done. If the command was a query, such as for status information, active mode will just reply before returning to Idle mode. If the command is to print a line, active mode will perform that action as outlined in section 2.2.3.3.

We have purchased a chipKIT WiFire development board to act as the TCB. This device utilizes 4 modes and a single interrupt, as seen in the figure above. These modes are described in more detail in the following subsections.

2.2.3.1 POR Mode

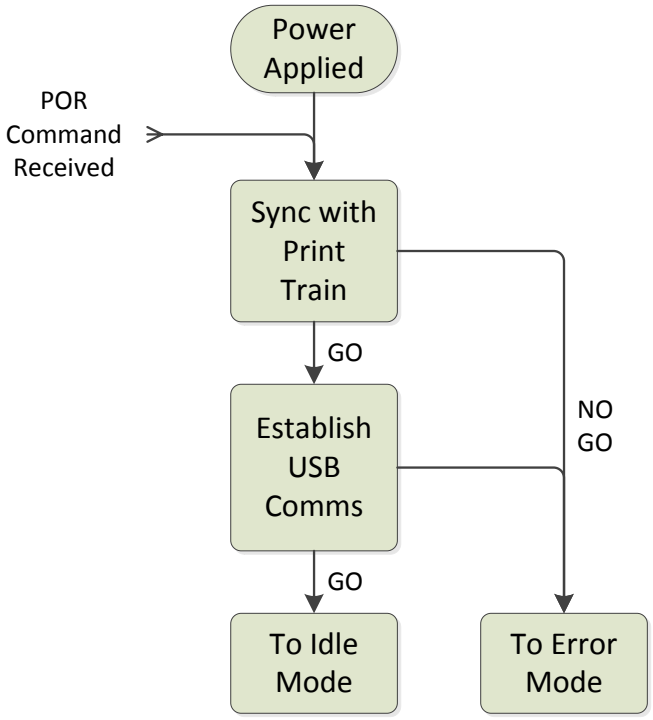


Figure 4: POR Mode

The TCB turns on when power is applied either through the USB/PC interface or another power source and the switch is turned on. The chipKit WiFire has some errata and a delay is required in order to allow it to work properly. It will spend at least three seconds in this mode to allow the USB to work properly. See the PIC32MZ errata datasheet for specifics.

In this mode, the chipkit waits for a PSS pulse and starts its 5us interval timer in order to keep in sync with the print train. This is done far before the actual printing begins in order to ensure PSS pulses are being received. After 3 seconds, USB comms will be established with the Host PC and the chipkit will move into Idle mode. Error mode will occur if the chipKit cannot communicate with the Host PC or no PSS pulses are received.

2.2.3.2 Idle Mode

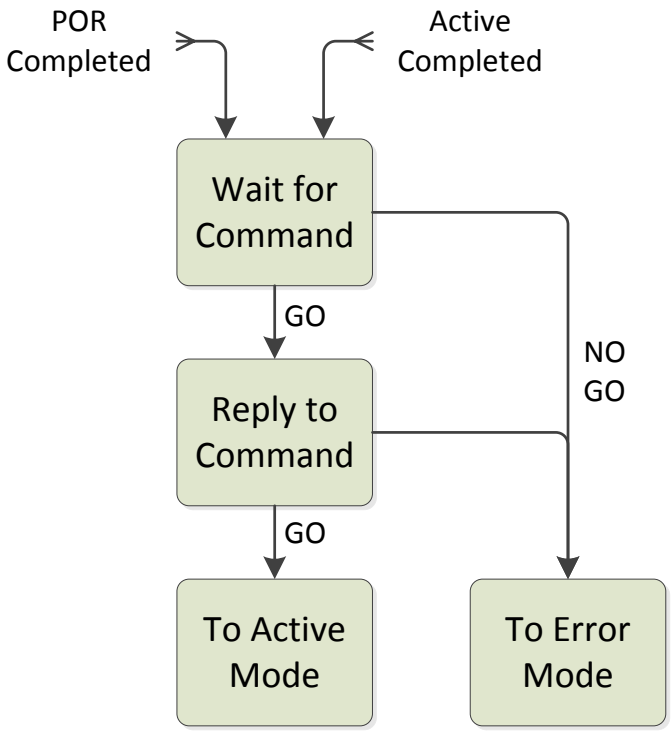


Figure 5: Idle Mode

The idle mode waits for information from the Host PC application. The commands it can receive from the Host PC are shown in appendix K. The TCB will send an acknowledge back to the Host PC before moving on to act on whatever command it was given. These commands are not time-sensitive and so the format does not matter much. The TCB has a built-in Universal Asynchronous Receiver-Transmitter (UART) on the micro-USB port used to communicate with the Host PC. It receives commands serially using the chipKit serial libraries.

2.2.3.3 Active Mode

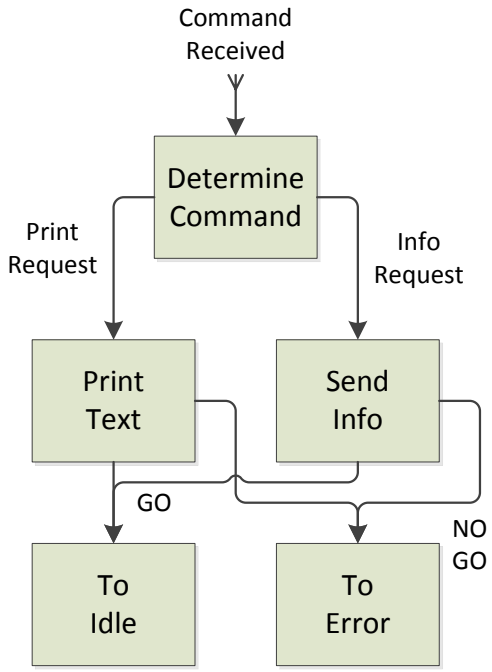


Figure 6: Active Mode

Active mode performs all actions specified by the user via the Host PC. If the specified command is a request for information, the TCB will access said information and reply over USB. If the command is a request to print text, the TCB will print that text by sending signals to the hammer and carriage drivers.

The TCB will take the command from the Host PC and determine if it will start printing text or send info to the Host PC. This information is written to the micro SD card on the TCB and when the PC requests info, it dumps this file to the PC. The primary function of send info is to send error information so that the operator may debug the issue. This includes the mode it was in beforehand and whatever data exists (PSS interrupt timing data, line print data, UART buffer contents, locations of pointers in character and hammer arrays, etc.)

The TCB receives data one line at a time in the production design. Each line is 132 characters. This data is received through the same UART used to send commands which has the alias of Serial in MPIDE. The Arduino Serial library is used in order to transfer data into a string, which can be run through other algorithms.

When the timing control board has a 132-character input, it will first determine if all characters exist on the character train. If a non-printable character is entered, it must save that character and change to error mode. This was done to keep in line with the original printer's behavior which lit the error light if an unprintable character was given.

If the line has only printable characters, the TCB must determine which hammers need to be fired. The algorithm will mark a 132-array of booleans as true or false depending on if a character exists on that hammer's position. It will also count the number of hammers marked in this way and store them in an integer in order to determine the total number that need to be fired. This array of booleans will be referred to as the hammer array. There is also an array of characters

which corresponds to the current train mounted on the printer, or the character array. For the rest of this description, it will be assumed that this is the UCS.

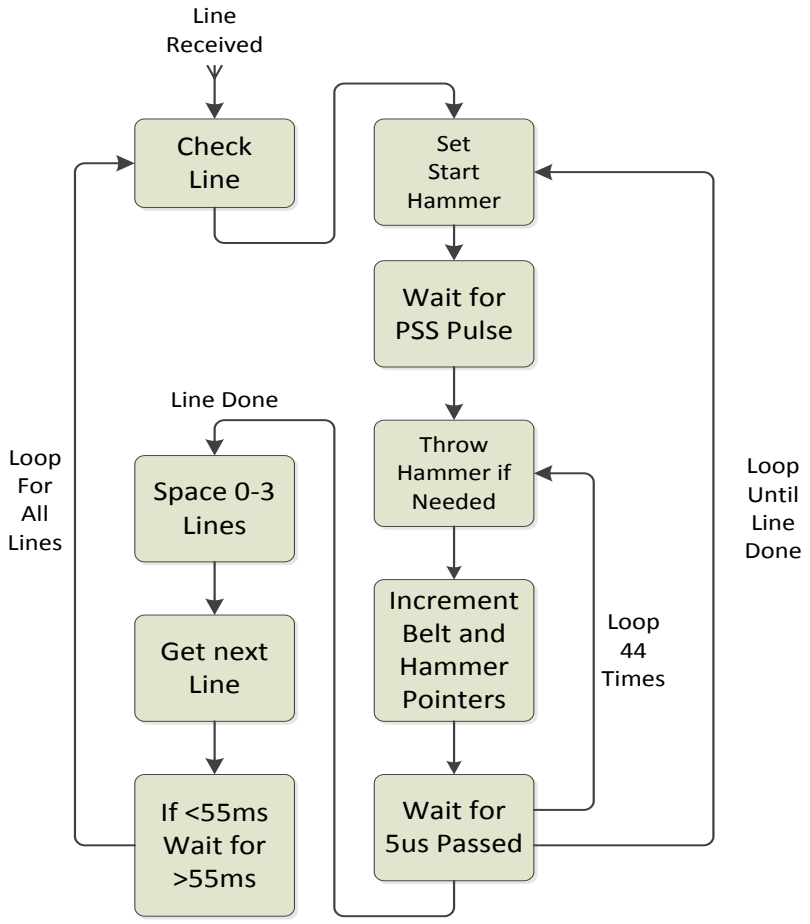


Figure 7: Print Text Detail

Printing text is the main task of Active mode, and the most important task of the TCB. As seen in appendix A, the timing of print signals is complex. The time between consecutive hammer firings may be as short as 5uS. The inner software loop must, therefore, be less than this time. In addition to this timing, the TCB will be receiving PSS pulses approximately every 243uS. These pulses are used to determine printer timing, as described in section 2.2.3.5.

When the printer receives a print command, it must await the next PSS interrupt. After this interrupt, a new hammer will be aligned with a letter every 5uS for 44 hammers. Using an interval timer set to 5us, a pointer is incremented through the hammer and character arrays. The character pointer is incremented by 2 and the hammer pointer is incremented by 3 in order to mimic the physical train's movement. See Figure 1-5 in the 1403-N1 maintenance manual for an example of how the printer hits each character. A short (~13uS) empty time will follow while awaiting the next interrupt. This process repeats until all locations have been printed.

After every increment, the current hammer that can be fired (index of hammer array + 1) is checked against the array of bools. If that hammer is marked as true, a character exists in that position on the input string. Therefore the character array pointer must be checked and the item at its current position compared with the character that needs to be printed from the input string. If the characters match, the hammer address is sent to the hammer driver cards and the number of hammers to be fired is decremented by 1. This process continues until the number of hammers to be fired is 0 which corresponds to all marked hammers having been fired.

The process of firing a hammer has been greatly simplified over last year's prototype system by the addition of the TCM logic hardware. Rather than toggling each hammer high for a specified time, the TCB only needs to signal the TCM designed hammer drivers with the address of the hammer to fire at the correct time. The drivers will then toggle the hammer input for 1170uS, causing the hammer to strike the page which prints one character.

Following completion of each line, the page needs to be advanced. The timing control software determines the number of new lines, from 1-3. This task has also been simplified by the addition of the TCM designed logic. The TCB sends a command to the carriage driver, as described above, with an address that signifies the start of carriage motion, and then sends another command when it is time to stop the carriage. The driver handles the toggling of the specific discretes which advance the page. The paper will take at least 20ms to move, depending on the number of lines moved. During this time, the timing control software will take the next 132-character string from the UART and check which hammers need to be fired for that string.

Each line must take at least 55ms to print. This is a mechanical limit imposed by the printer hardware and is out of this design's control. This lower bound on time per line will be kept by checking the time taken to decrease the hammers required count to 0 and adding 20 since that is the time taken to move the carriage. If this final value is less than 55, the timing control software will have to ignore PSS pulses until it is greater than or equal to 55. The pulses must still be recorded since they will determine what the starting hammer and character are, but the software cannot go through and attempt to assert outputs until enough time has passed.

2.2.3.4 Error Mode

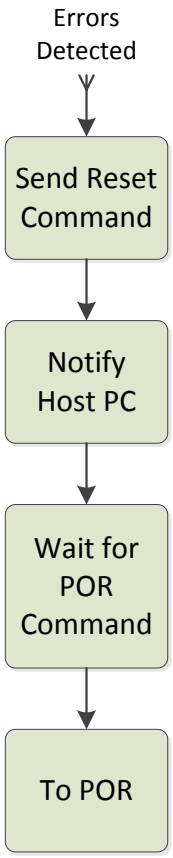


Figure 8: Error Mode

If the TCB encounters an unknown condition in any mode, it enters error mode. The TCB sends a reset command to the hammer drivers by setting the appropriate address and toggling the sync line. This causes the drivers to all be toggled low and the carriage to be stopped. It must record the issue which resulted in error mode, if known, and the status of the TCB. It waits for a reset command of its own from the Host PC in order to go into POR mode again. Error mode necessitates user intervention. In order to obtain information for debugging, the operator must query the TCB to send information (see 2.2.3.3).

2.2.3.5 PSS Interrupt

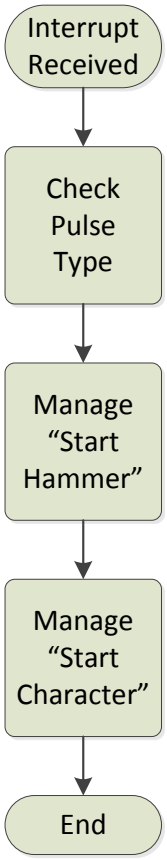


Figure 9: PSS Interrupt

One of the external interrupt pins on the TCB is configured to interpret rising edges as interrupts. Section 10.6 in the chipKit wiFire manual covers external interrupts. Only one pin is required. The other sync signals received from the printer are not as crucial.

The PSS interrupt will is activated every 243uS by a signal from the printer. This signal indicates that the print train is aligned at the start of a sub-scan, and that 44 hammers will be aligning as described in section 2.2.3.3. There is also an extra pulse, the home pulse, which occurs just before sub-scan 1 and signals a full train revolution. There is a difference in timing between the normal PSS pulses and the extra pulse, as shown in Figure 10.

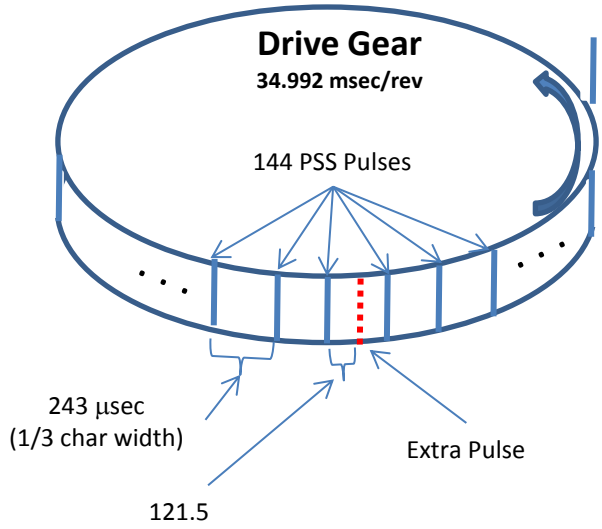


Figure 10: Pulse Source & Timing

PSS pulses are the main method of keeping the train and timing control software in sync. This also requires that the software have a method of determining which pulse is the extra pulse so that it can be ignored. This is done by comparing the time between pulses. If a pulse comes much too soon, it is the extra pulse and will be ignored. The TCB also keeps track of times between pulses in order to ensure that no pulses are missed or erratic. This information is used to decide when an error has occurred, which is evidenced by too long a time between pulses or otherwise bad timings.

Other pulses which do not cause interrupts will indicate the state of the other printer components, and are necessary to determine which sub-scan is being triggered. These signals are checked after entering the interrupt. Once this information is received, the interrupt determines what state the belt is in and records that position for use by Active mode.

2.2.4 Hardware Safety Interlock

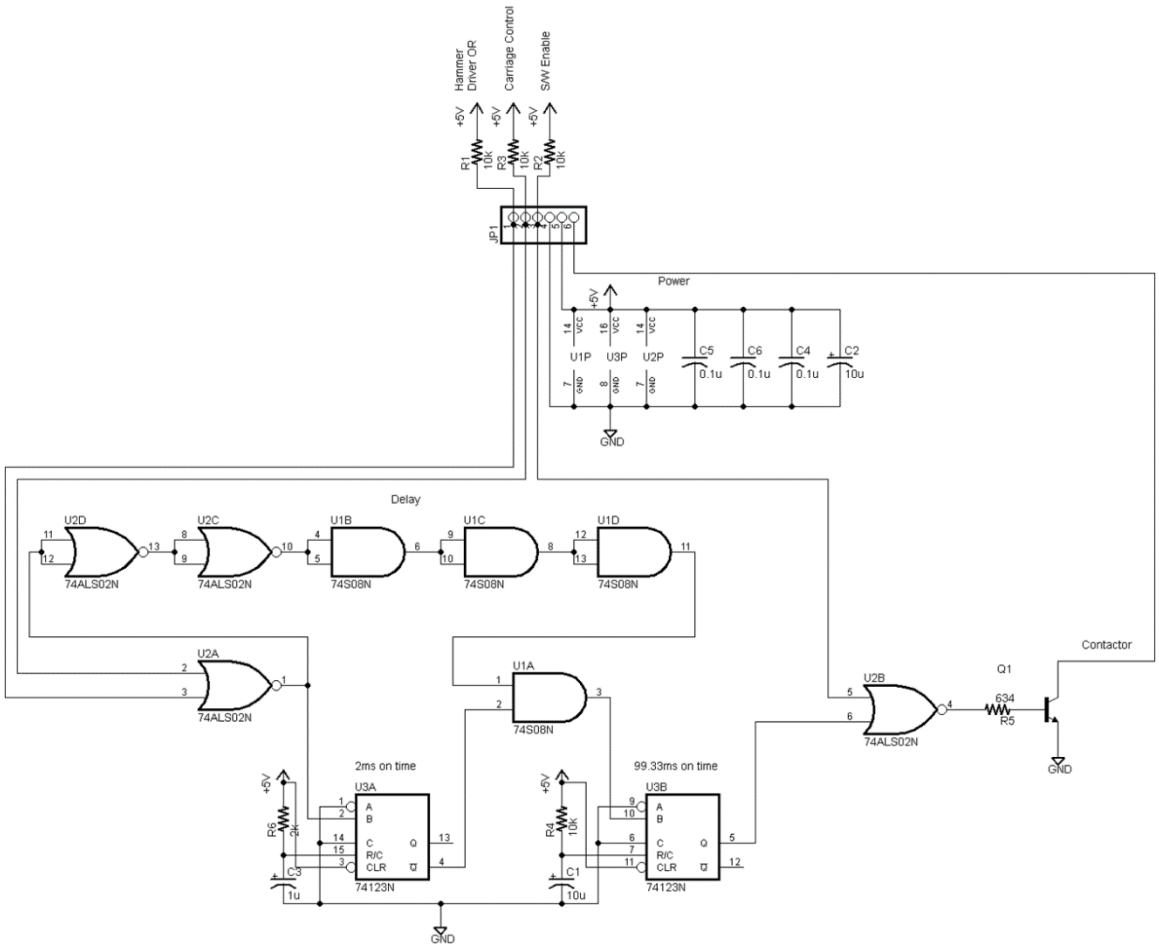


Figure 11: Hardware Safety Interlock

The interlock PCB is the only hardware which the WCP team produced. It will remove power when unsafe conditions occur. Rather than remove power using software, and since this system is meant to protect the printer from damage, it was decided that pure hardware should be used to decrease the chances of failure.

The printer controller rack contains a contactor for removing power when hardware errors occur. One side of the contactor coil is powered by 60Vdc, and the other needs to be grounded before power is applied to the printer. This ground is supplied by the WCP interlock device, and will be removed in the case of error. When the interlock device removes this ground, power will be removed from the printer.

The printer contains a set of switches which will open when the printer enters an unsafe condition. For example, if the printer is overheating a thermal safety switch will open. Rather than detect these failures with the interlock device, all of these switches will be placed in series with the provided ground signal. When any switch opens, it will break the ground and open the contactor, resulting in the removal of power.

The interlock will detect if the carriage is signaled to move while any hammer is currently engaged, as this could damage the printer. This task is complicated by the fact that a 1.2ms

coincidence between carriage and hammer signals is expected on every printed line. This means that coincidences of less than ~2ms must be ignored, and longer coincidences must result in removal of power.

To prevent power from being applied while the control signals are in an unknown state, the carriage driver includes an enable signal to be detected by the interlock device. This active low signal results in removal of power when removed.

2.3 Use

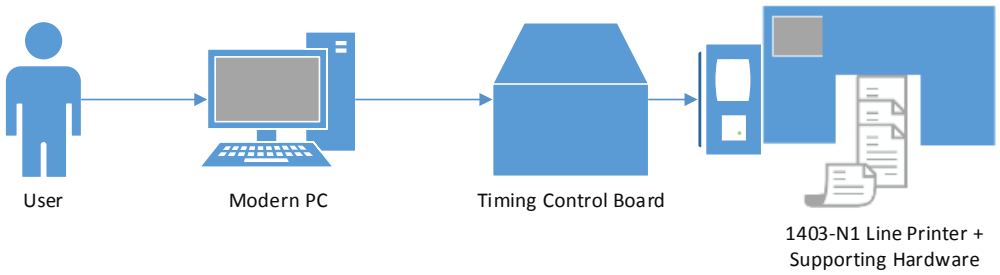


Figure 12: User Interfacing

This design effectively replaces the rack of hardware which this printer required in order to interface with a CPU. It allows a modern PC to interface with the 1403-N1 line printer. To the user, most of this design's functions are hidden. A user need only interface directly with the Host PC application. They must have some text in mind to print and either type it directly into the Host PC application or have a file they wish to print. An operator who wishes to investigate printer errors may query the TCB for fault information and edit the configuration files, but the end-user will only print text.

3 Implementation

3.1 Timing Controller

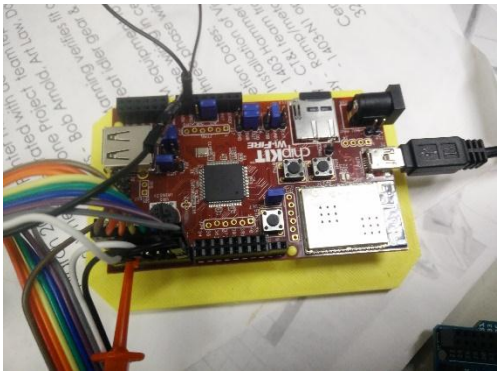


Figure 13: chipKIT WiFire and Mounting

The timing control software was implemented on a chipKIT WiFire board using MPIDE, which is similar in form to the Arduino development environment. The program makes it complicated to include multiple source files into a single build, so all code is contained in a single file with no headers. The language is C++, but many functions are included for manipulating the chipKIT.

All wait loops were implemented using the PIC32 core timer, a register which is incremented on every other clock cycle. Since the WiFire runs at an internal clock speed of 200MHz, this means that the core timer has a resolution of 10ns. This made it very convenient to get tight timings on all our loops, notably the 5us inner loop during the cascade. Given the 32 bit register size, the maximum delay time of any wait loop would be around 43s, so we never came close to having issues there.

Rather than use the MPIDE port manipulation functions, direct port manipulation was performed using the PIC32 LAT, PORT, and TRIS registers. This was necessary because the MPIDE port manipulation functions have a lot of overhead, and we wanted to keep both the inner loop and interrupt times down as low as possible. Since this adds some complexity to the code, all port manipulation functions were implemented in #define statements, which were prefaced with a comment block explaining the use of each function.

The chipKIT will be mounted inside of a rack mount chassis that CT&I and TCM designed. The chassis is intended to hold all the hammer and carriage driver cards, a power supply for them, the chipKIT, and all wiring necessary for interconnection. TCM also designed and 3D printed a holder for the chipKIT which will be adhered to the floor of the chassis. Panel mount USB and SD card connectors were ordered and fitted to the chassis so that chipKIT connections can be made without opening the case. CT&I is currently in the process of chassis assembly and wiring, following which they will physically integrate the WCP subsystem with all other hardware.

3.2 Host PC

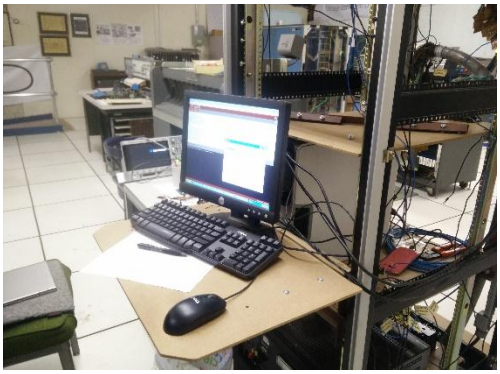


Figure 14, Host PC Hardware

The Host PC software was implemented on a Windows XP machine that CT&I had allocated for the purpose, and was written in VB.NET using Microsoft Visual Studio (VS) Express 2010. This was the most recent version of VS that would run on an XP computer, and it was necessary to write all code in the .NET Framework version 4.0 for the same reason. The code should therefore be compatible with any current Windows version (7 or 8) without modification, as they are compatible with Framework 4.0.

The Host PC software UI can be seen in appendix H. Screens were included for managing connection, creating configuration files, and printing. Each screen has a clear indicator of the connection status, including any errors that may occur. The software was implemented using event driven code, so errors can be detected and displayed at any time should they occur.

3.3 Hardware Safety Interlock

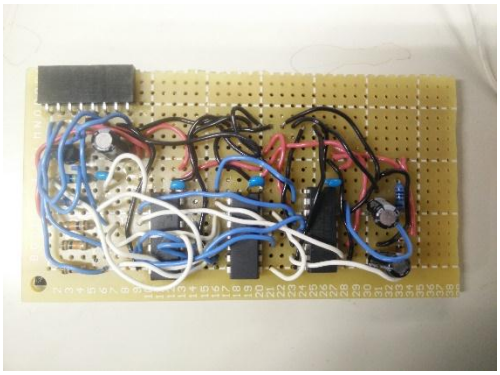


Figure 15: Hardware Safety Interlock Prototype

The Interlock device was designed with the help of Robert Arnold, a volunteer at the client site and retired IBM engineer who had assisted in the original design of the 1403 printer. A scheme was developed without any special circuitry for the printer interlocks, and the switches themselves were placed in the forward path of the 60Vdc supply to the contactor. This would ensure that the contactor opened if any of the interlocks were opened.

One function that remained on the interlock card itself was detection of carriage controls coinciding with engaged hammers. The complication was that coincidence is expected for the first ~1170us of the carriage control signal. A pair of mono-stable “single-shots” were implemented to ignore coincidence of less than ~2ms, while longer coincidence would trigger a ~100ms disable pulse to the contactor, opening it. Test results for this circuitry can be seen in appendix I.

CT&I also requested that a function be added to the interlock device. This was a software enable, which would need to be held low for the printer to receive power. This enable was added and will be connected to a spare output of the carriage driver card for use by the TCM software.

A single prototype was assembled using perf board. The prototype was made to match the layout of the routed board to facilitate debug if it were necessary. The prototype was tested by inputting pulses of different lengths to the various inputs of the unit, and the results were delivered to CT&I. Those results were approved by client engineers, and gerber files for a routed board were delivered to the client. The gerber files include a second routing which includes two sets of circuitry on a single board which could be cut in half, resulting in two interlock bare boards. This was provided to reduce client costs should they ever decide to create a final board and spare.

4 Evaluation

4.1 Overview

In order to demonstrate that the system can meet all of its requirements, a series of acceptance tests will be performed. Each test will validate one or more requirements of the system, as shown in appendix B. When all tests in appendix B have been performed, the system will be considered complete and ready to be delivered to the client.

Testing will take place at the client site during one or more selloff events. These events should take place no later than April 29 2015 but may take place as soon as system integration has completed. If possible a client representative will be present at the selloff events to approve test approach and witness test results.

The test procedures were devised in such a way that each test has as little reliance as possible on hardware outside the scope of the requirement. For example, requirements for the Timing Controller alone should not have tests which rely on the Host PC or on the hammer/carriage drivers. For that reason, the test procedures are separated into sections based on the scope of the requirements within.

4.2 Testing and Results

4.2.1 Selloff Event One

On April 23 2015 a selloff event was attempted and partially completed at the client site. The results were later determined to be erroneous, as explained in the following paragraphs, and this event cannot be used as validation of any requirement. A signoff sheet, included in this document as appendix G, was filled in by Arthur Law, who was present as a client representative. During the event, 20 of the 64 total tests were completed successfully. Given the result of the passed tests, additional 10 tests were deemed unnecessary by the client and signed off.

Shortly after the start of the event, it was determined that we could not meet certain requirements with the loaded Timing Controller software. We had implemented enhancements to the Error mode and SD card code, and these changes were causing unexpected problems with the rest of the functions. The team decided to roll back to a previous release for the event, since the enhancements could be proven out at a later date. The previous week's code was quickly reviewed, and several essential updates were made.

During test, it was determined that several portions of the loaded code had been mistakenly left in a debug mode. For example, the entry point to error mode had been commented out to facilitate debug the prior week. These determinations were made in the later portion of the event, and likely invalidated the test results of some performed tests. Given the irregularity of the circumstances, it was decided that all testing would need to be repeated with a new release of the code, and the results of selloff 1 were to be considered invalid.

The event was started at 18:00 without a Control Signal Detector (CSD) present, and all tests which required that device were left unperformed. The tests which were completed included many of those proving serial communication between the Host PC and Timing Controller, print signal timing, transition into the different operational modes, and detection of sync errors. The

voltage levels of the command outputs were also tested, but the majority of the signals were signed off by the client without test because of the unlikelihood of failure.

Jim Ulrich, a TCM representative and the designer of the hammer/carriage driver software, arrived during selloff, and the procedure was interrupted to implement a CSD. The team decided that it would be best if a driver card was used as a CSD, and the hammer drive signals were used to verify timing and address. Jim quickly modified his release so that a "Reset" command would cause the character 'R' to be sent over serial. This would allow the team to more easily detect reset signals, which cause no other change in driver board outputs. At this point it was determined that selloff should be aborted and testing ended.

During this event, two additional findings were made. Firstly, the PSS interrupt was too error sensitive and needed improvement. Intermittently the Timing Controller would detect a timing error in the PSS pulses and enter error mode. Secondly, the serial approach needed enhancement. Occasionally during serial transmission a byte was lost, resulting in a comm error. These issues were to be addressed at a future date, ideally prior to any future selloff event.

4.2.2 Selloff Event Two

On April 29 and 30 2015, a second selloff event was attempted and completed successfully. A signoff sheet, included in this document as appendix J, was filled in by Nicholas Hekman, with Robert Lusch present as a client representative. During the event, 34 of the 64 tests were completed successfully. 18 additional tests were deemed unnecessary given successful completion of the aforementioned tests. The remaining 12 tests were not performed due to partial completion of the rack wiring, and are expected to pass when they are performed.

Following failure of selloff 1, fixes for all complications had been implemented. Serial communication was improved by prepending the message size and appending a Fletcher-16 checksum to the message. Message receipt was then acknowledged following checksum calculation and comparison. The PSS interrupt was made more robust by adding additional error handling, especially the ability to recover from a single missed PSS pulse. The concept of "minor" errors was also implemented, meaning that errors which did not directly imply loss of sync (glitches) would be ignored until a threshold amount had occurred.

The event was started at 18:00 using a logic analyzer as the CSD. After the arrival of Jim Ulrich, a carriage driver and hammer driver were provided to be used as CSDs. All tests except for 3060 were completed without complication. Given the ability of the hammer and carriage driver cards to properly detect control signals, paired with the success from the previous selloff, tests 1010 and 1020 were determined to be unnecessary and were not performed.

Test 3060, sending of carriage space commands, could not be completed during the first night of the event. Space commands were sent and detected, but the timing of the commands was too short. Instead of a 2ms spacing, the 1 line space had ~2.2us spacing. After leaving the client site for the night, the problem was found to be a line of code which was off by a factor of 1000. The code was corrected, and 3060 retest was completed on the following afternoon.

During the event, one additional finding was made. The serial enhancements had been designed to operate within the 20ms of free time following each printed line, and were expected to take around 10ms. After acceptance testing was completed, this belief was tested and found to be incorrect. The entire process of sending one line over serial was seen to take ~30ms, which

exceeded the time available when printing full lines. This will limit maximum print speed to less than 950 LPM.

It is theorized that this unexpectedly long serial time is due to a delay in the Host PC software serial following the receipt of each line. The Host PC software is currently event driven, and an event occurs when serial data has been received which triggers evaluation of the data and if required response. The event will not occur for some time after serial data is received while the PC waits for additional data which is not coming. The team's proposal for reducing the latency is to move from event driven code to multi-threaded code, which will allow us to respond very quickly following the receipt of serial data. This will have the added benefit of more closely matching the Timing Controller serial approach, which operates synchronously on the main thread.

4.3 Assessment

Although the system was able to pass all acceptance testing, it is unable to meet the throughput desired by the client. With the current serial approach, robust though it is, the maximum throughput of the device is approximately 950 LPM, compared to the client specified print speed of 1100 LPM. WCP has, however, determined a method for meeting this specification, and the team will try to reach that goal prior to handing off the software.

The team was able to implement a robust approach to maintaining sync with the printer. The system now rarely, if ever, loses sync with the print train, and any loss of sync should indicate an issue with the printer and generation of sync pulses. This was the biggest concern of the CT&I team, and it is a major accomplishment in its own right.

The Host PC user interface is now clean and user friendly, offering instant feedback following the occurrence of any error. The Host PC is capable of performing all required tasks, including the printing of a text file as specified by the user.

The Interlock device has not been fully integrated with the contactor or switches, and has never been connected to the carriage driver. As such, it would be hard to accurately assess the functionality of the board. However, given the customer requirements and the results of unit testing on the prototype, it is reasonable to expect that the device will operate as intended.

5 Schedule and Budget

For this project, our team was given a budget of $800 from the IEEE Binghamton Section. In the beginning of the year, we made an initial budget based on what we thought we would need. We began by thinking that we would need a single chipKIT WiFire board and planned our budget around this. It became apparent later on in the project that we would in fact need two in order to debug our software efficiently. This has impacted our budget significantly, but we have been able to absorb this increase in cost due to the fact that we had overestimated the cost of certain other aspects of our project. At the end of this project, we are predicted to come in significantly under budget, as seen in table 2 below.

Table 2: Final Budget

Item	Original Estimate $	Actual to Date $	Estimate to Completion $	Estimate at Completion $
chipKIT WiFire (x2)	100	188	0	188
Hardware Safety Interlock and Mounting	200	30	70	100
Prototyping, Rework, and Misc. Hardware	300	92	208	300
Total $	600	310	278	588
Budget $	800		Remaining $	212

Over the course of this school year, our team has been required to complete certain project related milestones, summarized in table 3 below. These milestones started in October, with the beginning of this project, and will end in May as we conclude our work. We have been able to complete every milestone on time so far, with some being completed ahead of time. All of these milestones have been completed in accordance with the WCP requirements. Our team is also on track to complete our future milestones and present our work to CT&I. A detailed project schedule is available in appendix C.

Table 3: Summarized Schedule

Milestone	Date Due	Percent Completed
Project Launch	2014-10-03	100
Requirements Analysis / SRR	2014-10-17	100
System Design and Planning / SDR	2014-11-14	100
Detailed Design / CDR	2014-12-05	100
Interim Presentation	2014-12-12	100
Prototype Operational	2015-04-17	100
Testing Complete	2015-04-24	90
Final Presentation and Report	2015-05-08	25
Delivery to Client	2015-05-16	25

6 Future Plans

Before our subsystems can be utilized, the CT&I Printer Refurbishment Project must complete system integration and test at their overall project level. A total of 17 hammer driver boards and one carriage driver board will need to be assembled for the chassis, which represents a lot of wiring. Each card will receive connections to the 9 timing controller outputs, and will have outputs to drive 8 hammers.

Once the chassis components are mounted, all control signals will need to be validated. Timing controller outputs have been validated when connected to a single hammer driver via a foot of wire, and it is currently unknown whether additional hammer driver connections will affect the signals. One BAE systems engineer who was consulted suggested that 50ohm terminations may be necessary to reduce ringing in longer lines, but that the current approach should be tried before implementing any change.

Once the system is fully assembled and mechanically fit, the line spacing times will need to be adjusted in software. Client engineers have informed the team that line space timings are variable and may need to be varied, so these have been left variable in the software and will need to be found empirically before being set.

If the client wishes, they could at a future time implement the high speed skip function of the carriage. This would allow them to space much faster than the current implementation, but is only meant for spaces of more than three lines. Due to the complexity of the process, and because it is considered unnecessary for normal printer operation, the client did not want the feature added at this time. The carriage driver was, however, developed with the high speed skip in mind, and there are control signals mapped to the enabling and disabling of the carriage high speed skip. This means that the WCP software could be updated to incorporate high speed skips with no hardware modifications necessary.

7 References

[1] Digilent Inc. (2014). *ChipKIT™WiFire™ Board Reference Manual*. Pullman, WA: Digilent Inc.

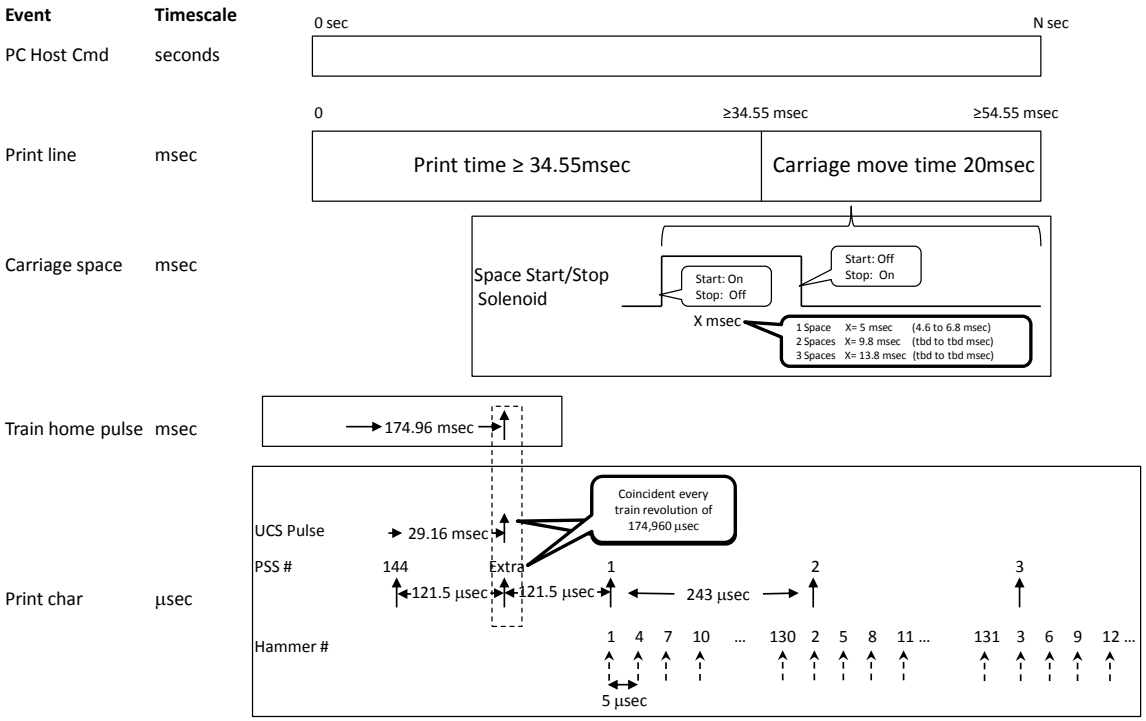
[2] WCP07 IBM Printer Driver Project Requirements Specification (PRS) Rev. A of 2014-11-08

Appendixes

Appendix A – Timing Chart
Appendix B – Test Procedures
Appendix C – Detailed Schedule
Appendix D – Preliminary Print Pseudo Code
Appendix E – Preliminary TCB Function List
Appendix F – Driver Operation Modes
Appendix G – Selloff 1 Results 2015-04-22
Appendix H – Host PC UI Screens
Appendix I – Interlock Layout and Results
Appendix J – Selloff 2 Results 2015-04-29
Appendix K – Command Definitions

Appendix A – Timing Chart

WCP07 IBM Printer Driver
Events & Timescales



Appendix B – Test Procedures

IBM 1403-N1 Printer Controller
Acceptance Test Procedure

Watson Capstone Project WCP07
Sponsor: IEEE Binghamton Section
2015-05-01
Revision: A

Submitted by:
Nicholas Hekman, Lead, EE
Alena Yampolskaya, COE
John Wiseman, COE

Faculty Advisor: Professor Jack Maynard
External Advisor: Tommy Lam
Client Advisor: Arthur Law
Program Manager: Professor Jack Maynard

Submitted in partial fulfillment of EECE 487-488 / ME 493-494 requirements.
Thomas J. Watson School of Engineering and Applied Science
Binghamton University
Binghamton, NY

Table of Contents

1 Notes

1. Tests are standalone and independent, therefore test sequence is not a requirement.
2. A measurement specified to be taken a measured amount of time after applying a stimulus is for reference only. The measurement may be performed at any time after the stimulus is applied.
3. As a minimum, all tests contained in this document shall be performed.
4. If at the time of testing it is impossible or inconvenient to attain sync pulses from the printer, these pulses may be simulated using other hardware (e.g. a microprocessor development board.)
5. If at the time of testing it is impossible or inconvenient to interface with the print hammers or hammer driver cards, Timing Controller commands and timing over that interface may be detected by other means (e.g. a microprocessor development board.)
6. If at the time of testing it is impossible or inconvenient to interface with the carriage driver card to receive the interlock software enable, the interlock input may be tied low for testing.

2 Special Test Equipment and Definitions

2.1 Control Signal Detector

In order to verify the Timing Controller control and sync outputs, it is necessary to detect the state of all nine TTL signals at the rising edge of the sync pulse. This may be done with a logic analyzer, but it may also be possible to use other means (e.g. a microprocessor development board.)

2.2 Standard Print Image

The Timing Controller determines control signal timings using a variable print train image. Unless otherwise stated, all test procedures require that the following print train image be loaded:

1234567890#@/STUVWXYZ+,%JKLMNOPQR-$*ABCDEFGHI&.]
1234567890#@/STUVWXYZ+,%JKLMNOPQR-$*ABCDEFGHI&.]
1234567890#@/STUVWXYZ+,%JKLMNOPQR-$*ABCDEFGHI&.]
1234567890#@/STUVWXYZ+,%JKLMNOPQR-$*ABCDEFGHI&.]
1234567890#@/STUVWXYZ+,%JKLMNOPQR-$*ABCDEFGHI&.]

2.3 Standard Test Line

To test controller print capability, a standard test line will be used. All procedures which refer to a standard test line should use the following string:

1234567890#@/STUVWXYZ+,%JKLMNOPQR-$*ABCDEFGHI&.]

3 Detailed Test Procedures

3.1 Timing Controller Test Procedures

3.1.1 Test 1000

3.1.1.1 Requirements

Upon application of power the system shall enter POR Mode. {WCP07-001}
The controller shall establish and maintain connection with the TCM hardware. {WCP07-029}
Upon entering this mode (POR), the system shall send a "Reset" command to the TCM Hardware. {WCP07-056}
The controller shall receive all power from a single 5Vdc supply. {WCP07-053}

3.1.1.2 Equipment

1. Control Signal Detector

3.1.1.3 Procedure

1. Ensure that P5V power is removed from the Timing Controller.
2. Connect the Control Signal Detector to the Timing Controller control outputs.
3. Apply P5V power to the Timing Controller.
4. Verify that a reset command was transmitted to the Control Signal Detector.

3.1.2 Test 1010

3.1.2.1 Requirements

The controller shall output sync pulses to the TCM hardware at TTL levels. {WCP07-049}

3.1.2.2 Equipment

1. Oscilloscope

3.1.2.3 Procedure

1. Connect the oscilloscope to the Timing Controller sync pulse output and configure it to trigger on the sync pulse.
2. Apply P5V power to the Timing Controller.
3. Verify that the signal low level falls between -0.1Vdc and 0.4Vdc.
4. Verify that the signal high level falls between 2.4Vdc and 5.5Vdc.

3.1.3 Test 1020

3.1.3.1 Requirements

The controller shall output hammer and carriage control signals to the TCM hardware at TTL levels. {WCP07-048}

3.1.3.2 Equipment

1. Oscilloscope

3.1.3.3 Procedure

Perform the following procedure for each bit of the Timing Controller control outputs.

1. Connect the oscilloscope to the Timing Controller output signal and configure it to trigger on the rising edge.
2. Apply P5V power to the Timing Controller.
3. Verify that the signal low level falls between -0.1Vdc and 0.4Vdc.
4. Verify that the signal high level falls between 2.4Vdc and 5.5Vdc.

3.1.4 Test 1030

3.1.4.1 Requirements

The controller shall detect a PC command to initiate POR Mode. {WCP07-002}
The controller shall detect a PC command in order to initiate Power On/Reset. {WCP07-026}

3.1.4.2 Equipment

1. Control Signal Detector

3.1.4.3 Procedure

1. Connect the Control Signal Detector to the Timing Controller control outputs.
2. Send the command “POR” to the Timing Controller via USB.
3. Verify that a reset command was transmitted to the Control Signal Detector.

3.1.5 Test 1040

3.1.5.1 Requirements

This mode (POR) shall establish synchronization with the printer train. {WCP07-003}
The controller shall accept the PSS/Extra Pulse, UCS, and home pulses at TTL levels. {WCP07-047}

3.1.5.2 Equipment

None

3.1.5.3 Procedure

1. Ensure that the Timing Controller timing pulse inputs are connected to the printer pulse signals.
2. Send the command “POR” to the Timing Controller via USB.
3. Wait a minimum of 1 second to enter Idle mode.
4. Send the command “ERR” to the Timing Controller via USB.
5. Verify that the timing controller responded over USB with the command “ERR 0” indicating no errors.

3.1.6 Test 1050

3.1.6.1 Requirements

This mode (POR) shall establish communication across all internal and external interfaces. {WCP07-004}
While in Idle Mode, the system shall maintain communication across all internal or external interfaces. {WCP07-008}
While in Idle Mode, the system shall maintain synchronization with the printer train. {WCP07-009}

3.1.6.2 Equipment

1. Control Signal Detector

3.1.6.3 Procedure

1. Ensure that the Timing Controller timing pulse inputs are connected to the printer pulse signals.
2. Connect the Control Signal Detector to the Timing Controller control outputs.
3. Send the command “POR” to the Timing Controller via USB.
4. Verify that a reset command was transmitted to the Control Signal Detector.
5. Wait a minimum of 1 second to enter Idle mode.
6. Send the command “ERR” to the Timing Controller via USB.
7. Verify that the timing controller responded over USB with the command “ERR 0” indicating no errors.

3.1.7 Test 1060

3.1.7.1 Requirements

If any error conditions occur, this mode (POR) shall transition to Error Mode. {WCP07-005}

3.1.7.2 Equipment

None

3.1.7.3 Procedure

POR mode has no error conditions. No testing will be performed.

3.1.8 Test 1070

3.1.8.1 Requirements

Upon successful completion, this mode (POR) shall transition to Idle Mode. {WCP07-006}
Upon entering Idle Mode, the system shall await user commands. {WCP07-007}
Upon receipt of user commands, this mode (Idle) shall transition to Active Mode. {WCP07-011}
Upon entering this mode (Active), the system shall perform all tasks specified by the user. {WCP07-012}

3.1.8.2 Equipment

None

3.1.8.3 Procedure

1. Send the command “POR” to the Timing Controller via USB.
2. Wait a minimum of 1 second to enter Idle mode.
3. Send the command “WHO” to the Timing Controller via USB.
4. Verify that the timing controller responded over USB with the command “WCP”.

3.1.9 Test 1080

3.1.9.1 Requirements

If any error conditions occur, this mode (Idle) shall transition to Error Mode. {WCP07-010}
The controller shall detect error conditions. {WCP07-027}

3.1.9.2 Equipment

None

3.1.9.3 Procedure

1. Ensure that the Timing Controller timing pulse inputs are connected to the printer pulse signals.
2. Send the command “POR” to the Timing Controller via USB.

3. Wait a minimum of 1 second to enter Idle mode.
4. Send the command “ERR” to the Timing Controller via USB.
5. Verify that the timing controller responded over USB with the command “ERR 0” indicating no errors.
6. Disconnect the printer UCS pulse signal from the Timing Controller.
7. Wait a minimum of 1 second to enter Error mode.
8. Verify that the timing controller generated the USB signal “ERR X” where X is a non-zero positive integer.

3.1.10 Test 1090

3.1.10.1 Requirements

If any error conditions occur, this mode (Active) shall transition to Error Mode. {WCP07-013}

3.1.10.2 Equipment

None

3.1.10.3 Procedure

Note that following this test it may be necessary to transmit a new configuration file to the Timing Controller.

1. Send the command “POR” to the Timing Controller via USB.
2. Wait a minimum of 1 second to enter Idle mode.
3. Send the command “WCF N” to the Timing Controller, where N is the Standard Test Line.
4. Verify that the timing controller generated the USB signal “ERR X” where X is a non-zero positive integer.

3.1.11 Test 1100

3.1.11.1 Requirements

Upon completion of all tasks, this mode (Active) shall transition to Idle Mode. {WCP07-014}

3.1.11.2 Equipment

None

3.1.11.3 Procedure

1. Send the command “POR” to the Timing Controller via USB.
2. Wait a minimum of 1 second to enter Idle mode.
3. Send the command “WHO” to the Timing Controller via USB.
4. Verify that the timing controller responded over USB with the command “WCP”.
5. Send the command “WHO” to the Timing Controller via USB.
6. Verify that the Timing Controller responded over USB with the command “WCP”.

3.1.12 Test 1110

3.1.12.1 Requirements

Signals to the TCM hardware shall be limited to the address of the hammer to fire, an Idle command, a reset command, a carriage start command, or a carriage stop command. {WCP07-033}

3.1.12.2 Equipment

None

3.1.12.3 Procedure

All undefined commands are considered Idle, as they will have no effect on the hammer driver.

3.1.13 Test 1120

3.1.13.1 Requirements

While in Error Mode, the system shall idle while awaiting a PC command to initiate POR Mode. {WCP07-018}

3.1.13.2 Equipment

None

3.1.13.3 Procedure

1. Ensure that the Timing Controller timing pulse inputs are connected to the printer pulse signals.
2. Send the command “POR” to the Timing Controller via USB.
3. Wait a minimum of 1 second to enter Idle mode.
4. Disconnect the printer UCS pulse signal from the Timing Controller.
5. Wait a minimum of 1 second to enter Error mode.
6. Reconnect the UCS pulse signal.
7. Send the command “WCF WCP” to the Timing Controller via USB.
8. Verify that the Timing Controller responded over USB with the command “ERR X” where X is a positive non-zero integer.

3.1.14 Test 1130

3.1.14.1 Requirements

The controller shall store debug, status, and performance information. {WCP07-038}

3.1.14.2 Equipment

None

3.1.14.3 Procedure

1. Ensure that the Timing Controller timing pulse inputs are connected to the printer pulse signals.
2. Send the command “POR” to the Timing Controller via USB.
3. Wait a minimum of 1 second to enter Idle mode.
4. Disconnect the printer UCS pulse signal from the Timing Controller.
5. Wait a minimum of 1 second to enter Error mode.
6. Remove the Timing Controller SD card and verify that it contains a file with data for debug.

3.2 Host PC Test Procedures

3.2.1 Test 2000

3.2.1.1 Requirements

The Host PC shall utilize a Microsoft Windows Operating System (OS). {WCP07-054}
The Host PC OS shall be Microsoft Windows XP or any newer version. {WCP07-055}

3.2.1.2 Equipment

None

3.2.1.3 Procedure

1. On the Host PC click on the "Start" button in the lower left of the screen.
2. Right click on the "My Computer" icon and select "Properties" in the options box that appears.
3. In the window that appears, verify that "Microsoft Windows XP" is listed under the heading "System:".

3.2.2 Test 2010

3.2.2.1 Requirements

The Host PC shall receive all power from a single NEMA 5-15 grounded wall outlet. {WCP07-052}

3.2.2.2 Equipment

None

3.2.2.3 Procedure

1. Verify that the Host PC is plugged into a standard 15A grounded American power outlet.

3.3 System Test Procedures

Unless otherwise stated, all procedures in this section require that the Timing Controller and Host PC be connected via USB and that the Timing Controller timing pulse sense lines be connected to the printer pulse outputs.

3.3.1 Test 3000

3.3.1.1 Requirements

The controller shall establish and maintain connection with the Host PC. {WCP07-028}
The Host PC shall be able to read and display status information from the controller. {WCP07-045}

3.3.1.2 Equipment

None

3.3.1.3 Procedure

1. Open the Host PC software and navigate to the “Connections” window.
2. Click the “Refresh List” button to list available com ports.
3. Select the Timing Controller com port in the list and then click the “Connect to Port” button.
4. Verify that the “Status” box to the upper right turns green.

3.3.2 Test 3010

3.3.2.1 Requirements

The Host PC shall be able to signal the controller to enter POR Mode. {WCP07-040}

3.3.2.2 Equipment

1. Control Signal Detector

3.3.2.3 Procedure

1. Connect the Control Signal Detector to the Timing Controller control outputs.
2. Ensure that the Host PC software is connected to the Timing Controller.
3. Using the Host PC software, send a POR command to the Timing Controller.
4. Verify that a reset command was transmitted to the Control Signal Detector.

3.3.3 Test 3020

3.3.3.1 Requirements

The Host PC shall be able to read and display fault information from the controller. {WCP07-046}
Upon entering this mode (Error), the system shall notify the user that an error has occurred. {WCP07-016}
The controller shall send state information to the Host PC. {WCP07-037}
The controller shall make debug, status, and performance information available to the Host PC. {WCP07-039}

3.3.3.2 Equipment

None

3.3.3.3 Procedure

1. Open the Host PC software and navigate to the “Connections” window.
2. Ensure that the Host PC software is connected to the Timing Controller.
3. Disconnect the printer UCS pulse signal from the Timing Controller.
4. Wait a minimum of 1 second to enter Error mode.
5. Verify that the “Status” indicator in the Host PC software turns red and contains text.

3.3.4 Test 3030

3.3.4.1 Requirements

The Host PC shall be able to signal the controller to print a character at a specified hammer. {WCP07-041}

3.3.4.2 Equipment

1. Control Signal Detector

3.3.4.3 Procedure

1. Connect the Control Signal Detector to the Timing Controller control outputs.
2. Ensure that the Host PC software is connected to the Timing Controller.
3. Using the Host PC software, signal the Timing Controller to print the character ‘A’ at hammer number 1.
4. Verify that the command for hammer number one was transmitted to the Control Signal Detector.

3.3.5 Test 3040

3.3.5.1 Requirements

The controller shall establish and maintain synchronization with the print train. {WCP07-030}
The controller shall determine the hammer firing order and timing. {WCP07-031}
The controller shall be capable of printing a standard text block at a speed of 1100 LPM. {WCP07-034}
The Host PC shall be able to signal the controller to print one or more lines of text. {WCP07-042}

3.3.5.2 Equipment

1. Control Signal Detector
2. Oscilloscope

3.3.5.3 Procedure

1. Connect the Control Signal Detector to the Timing Controller control outputs.
2. Connect the oscilloscope to the Timing Controller sync output and configure it to trigger on the rising edge to capture approximately 1ms of data afterward.
3. Ensure that the Host PC software is connected to the Timing Controller.
4. Using the Host PC software, signal the Timing Controller to print a line consisting of the first three characters in the Standard Print Line.
5. Verify that the command for hammers number one, two, and three were transmitted to the Control Signal Detector in that order.
6. Verify that the three sync pulses are spaced 243 microseconds apart with a tolerance of 15 microseconds.

3.3.6 Test 3050

3.3.6.1 Requirements

The controller shall limit the number of simultaneous active hammers to a TBD number. {WCP07-035}

3.3.6.2 Equipment

1. Oscilloscope

3.3.6.3 Procedure

1. Connect the oscilloscope to the Timing Controller sync output and configure it to trigger on the rising edge to capture approximately 2ms of data afterward.
2. Configure the Timing Controller to limit active hammers to one.
3. Using the Host PC software, signal the Timing Controller to print a line consisting of the first two characters in the Standard Print Line.
4. Verify that a no sync pulse is seen for at least 1.2 milliseconds following the triggering sync pulse.

3.3.7 Test 3060

3.3.7.1 Requirements

The controller shall manage carriage control signals to space 1-3 lines. {WCP07-036}
The Host PC shall be able to signal the controller to space 1-3 lines down a page. {WCP07-043}

3.3.7.2 Equipment

1. Control Signal Detector
2. Oscilloscope

3.3.7.3 Procedure

1. Connect the Control Signal Detector to the Timing Controller control outputs.
2. Ensure that the Timing Controller is configured for a 1 line space wait of 2ms and a 3 line space wait of 6ms.
3. Connect the oscilloscope to the Timing Controller sync output and configure it to trigger on the rising edge to capture approximately 8ms of data afterward.
4. Ensure that the Host PC software is connected to the Timing Controller.
5. Using the Host PC software, signal the Timing Controller to space one line.
6. Verify that the commands for begin spacing and end spacing were transmitted to the Control Signal Detector, in that order.
7. Verify that the two sync pulses were 2ms +/- 0.5ms apart.
8. Using the Host PC software, signal the Timing Controller to space three lines.
9. Verify that the commands for begin spacing and end spacing were transmitted to the Control Signal Detector, in that order.
10. Verify that the two sync pulses were 6ms +/- 0.5ms apart.

3.3.8 Test 3070

3.3.8.1 Requirements

The Host PC shall be able to open a text file and signal the controller to print all text contained in that file. {WCP07-044}

3.3.8.2 Equipment

1. Control Signal Detector
2. Oscilloscope

3.3.8.3 Procedure

1. Connect the Control Signal Detector to the Timing Controller control outputs.
2. Connect the oscilloscope to the Timing Controller sync output and configure it to trigger on the rising edge to capture approximately 1ms of data afterward.
3. Ensure that the Host PC software is connected to the Timing Controller.
4. Create a text file containing the first three characters in the Standard Print Line.

5. Using the Host PC software, signal the Timing Controller to print the contents of the created text file.
6. Verify that the command for hammers number one, two, and three were transmitted to the Control Signal Detector in that order.
7. Verify that the four sync pulses are spaced 243 microseconds apart with a tolerance of 15 microseconds.

3.4 Stretch Goal Test Procedures

3.4.1 Test 4000

3.4.1.1 Requirements

The system should interrupt 60V power if the “T-Casting” safety switch is activated.

3.4.1.2 Equipment

1. DMM

3.4.1.3 Procedure

1. With the system in a powered on state, verify that 60Vdc +/-10Vdc is being supplied to the printer.
2. Activate the “T-Casting” safety switch.
3. Wait a minimum of 1 second for power to be removed.
4. Verify that -0.5Vdc to 1Vdc is now being supplied to the printer.

3.4.2 Test 4010

3.4.2.1 Requirements

The system should interrupt 60V power if the “Train Installed” safety switch is activated.

3.4.2.2 Equipment

1. DMM

3.4.2.3 Procedure

1. With the system in a powered on state, verify that 60Vdc +/-10Vdc is being supplied to the printer.
2. Activate the “Train Installed” safety switch.
3. Wait a minimum of 1 second for power to be removed.
4. Verify that -0.5Vdc to 1Vdc is now being supplied to the printer.

3.4.3 Test 4020

3.4.3.1 Requirements

The system should interrupt 60V power if a thermal safety switch is activated.

3.4.3.2 Equipment

1. DMM

3.4.3.3 Procedure

1. With the system in a powered on state, verify that 60Vdc +/-10Vdc is being supplied to the printer.
2. Simulate activating a “T-Casting” thermal safety switch by opening a connection to one.
3. Wait a minimum of 1 second for power to be removed.
4. Verify that -0.5Vdc to 1Vdc is now being supplied to the printer.

3.4.4 Test 4030

3.4.4.1 Requirements

The system should interrupt 60V power if any of the 4 “Form Jam” safety switches is activated.

3.4.4.2 Equipment

1. DMM

3.4.4.3 Procedure

1. With the system in a powered on state, verify that 60Vdc +/-10Vdc is being supplied to the printer.
2. Activate one of the “Form Jam” safety switches.
3. Wait a minimum of 1 second for power to be removed.
4. Verify that -0.5Vdc to 1Vdc is now being supplied to the printer.

3.4.5 Test 4040

3.4.5.1 Requirements

The system should interrupt 60V power if the “End of Form” safety switch is activated.

3.4.5.2 Equipment

1. DMM

3.4.5.3 Procedure

1. With the system in a powered on state, verify that 60Vdc +/-10Vdc is being supplied to the printer.

2. Activate the “End of Form” safety switch.
3. Wait a minimum of 1 second for power to be removed.
4. Verify that -0.5Vdc to 1Vdc is now being supplied to the printer.

3.4.6 Test 4050

3.4.6.1 Requirements

The system should interrupt 60V power if any hammer coil is energized while the carriage is moving.

3.4.6.2 Equipment

1. DMM

3.4.6.3 Procedure

1. With the system in a powered on state, verify that 60Vdc +/-10Vdc is being supplied to the printer.
2. Tie the interlock inputs for “hammer or” and “Carriage or” to ground.
3. Wait a minimum of 1 second for power to be removed.
4. Verify that -0.5Vdc to 1Vdc is now being supplied to the printer.

3.4.7 Test 4060

3.4.7.1 Requirements

…the Host PC should be able to specify the print train to the Timing Controller.

3.4.7.2 Equipment

None

3.4.7.3 Procedure

Note that following this test it will be necessary to transmit a new train image to the Timing Controller.

1. Ensure that the Host PC software is connected to the Timing Controller.
2. Using the Host PC software, signal the Timing Controller to update its train image. The new image must not be the Standard Print Image.
3. Remove the Timing Controller SD card and verify that it contains an updated train image.

3.4.8 Test 4070

3.4.8.1 Requirements

The Host PC should be able to specify the maximum number of hammers to simultaneously engage.

3.4.8.2 Equipment

None

3.4.8.3 Procedure

1. Ensure that the Host PC software is connected to the Timing Controller.
2. Using the Host PC software, signal the Timing Controller to update the maximum hammer on. The new number must differ from the number previously stored.
3. Remove the Timing Controller SD card and verify that it contains an updated maximum hammer on.

3.4.9 Test 4080

3.4.9.1 Requirements

The Host PC UI should be operable by someone familiar with a Windows style interface.
The UI should utilize a hierarchical set of windows.

3.4.9.2 Equipment

None

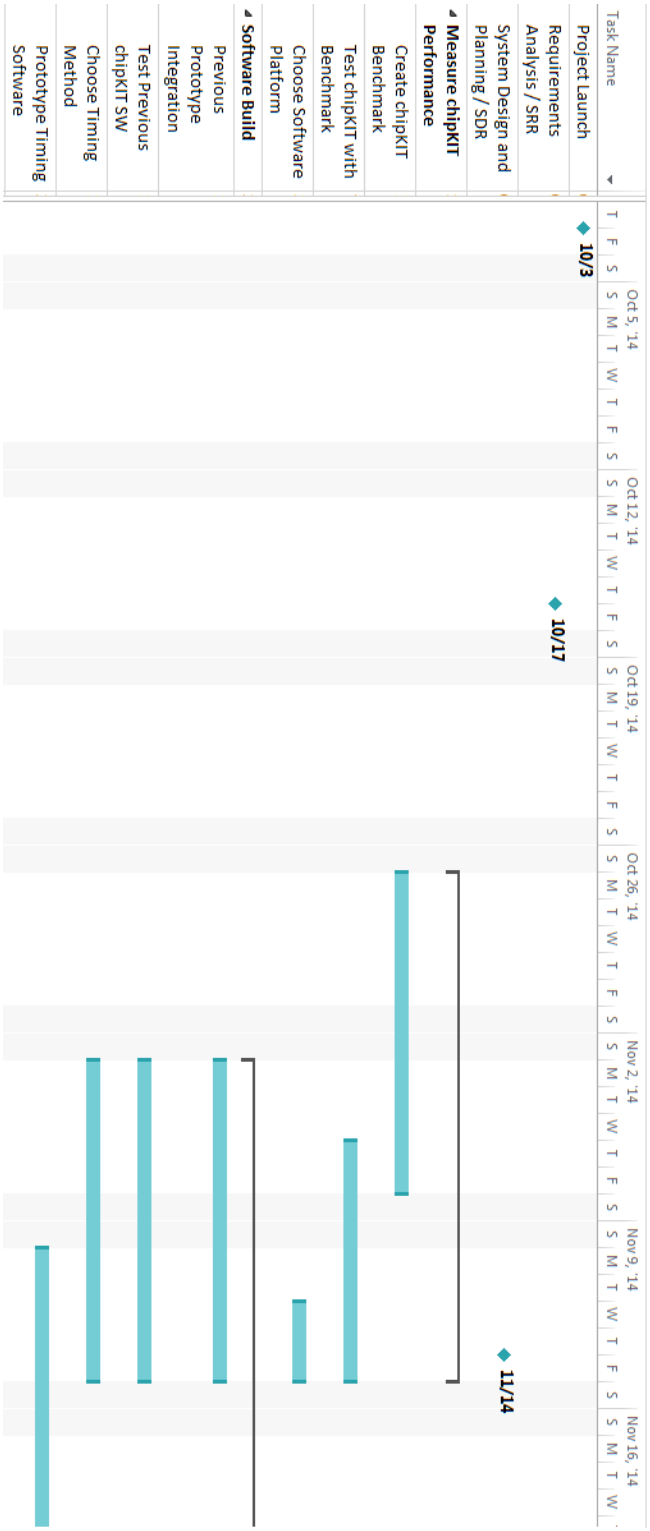
3.4.9.3 Procedure

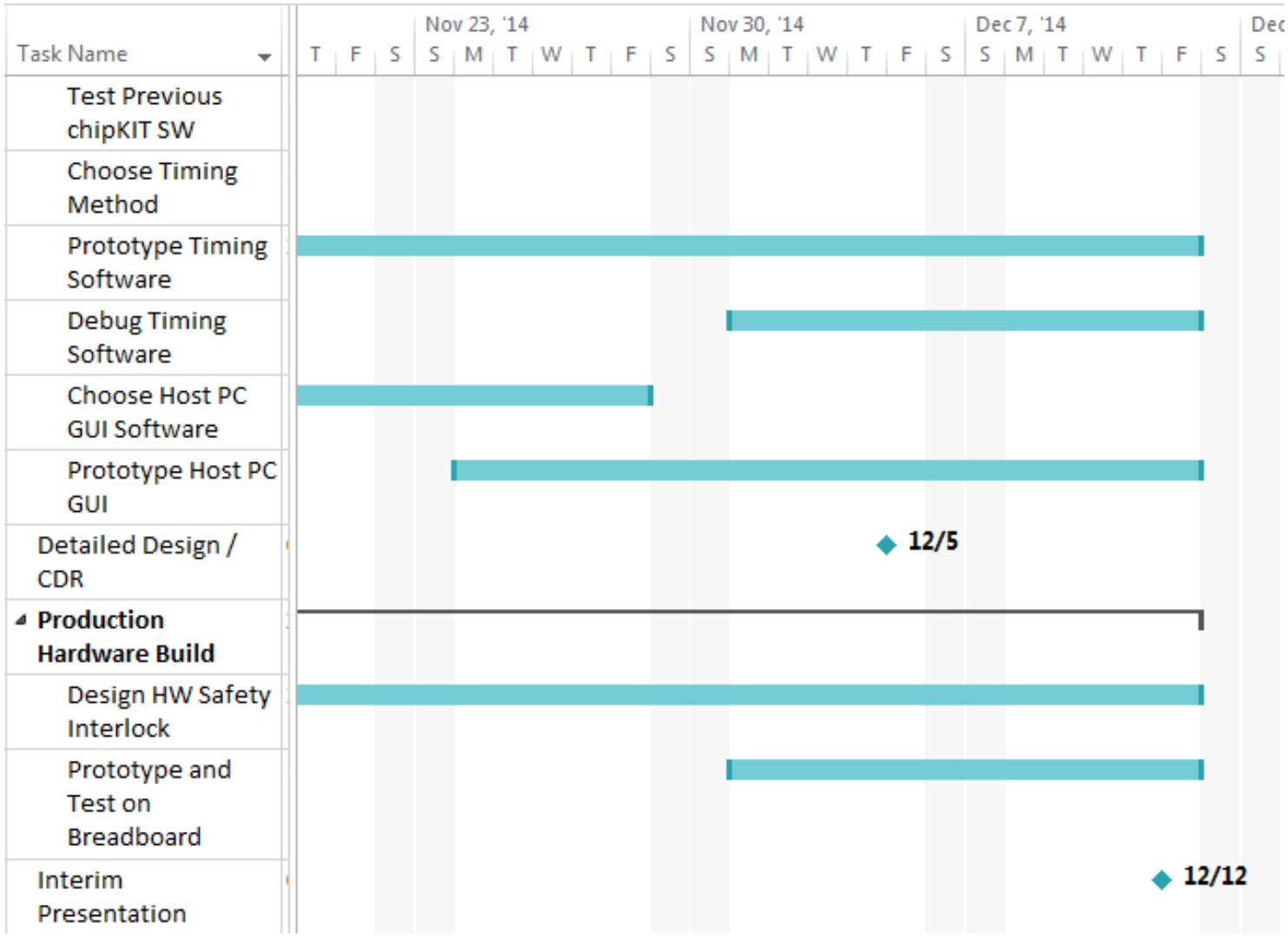
1. Open the Host PC software and navigate to the “Home” window.
2. Demonstrate the procedure for connecting to the Timing Controller.
3. Demonstrate the procedure for printing one line of text.
4. Demonstrate the procedure for printing text from a file.
5. Demonstrate the procedure for generating, saving, and sending a configuration file.

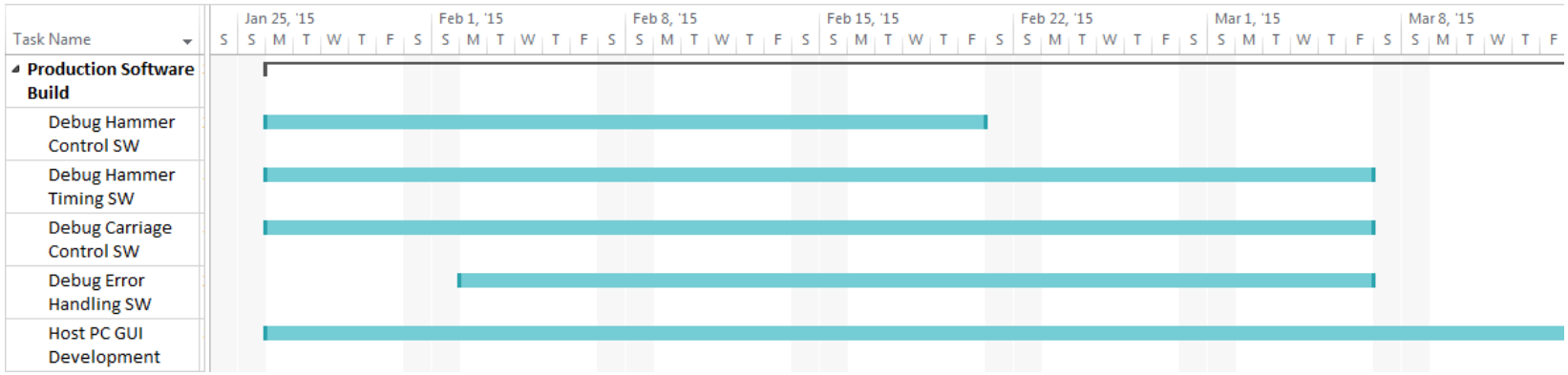
4 Acronyms Used

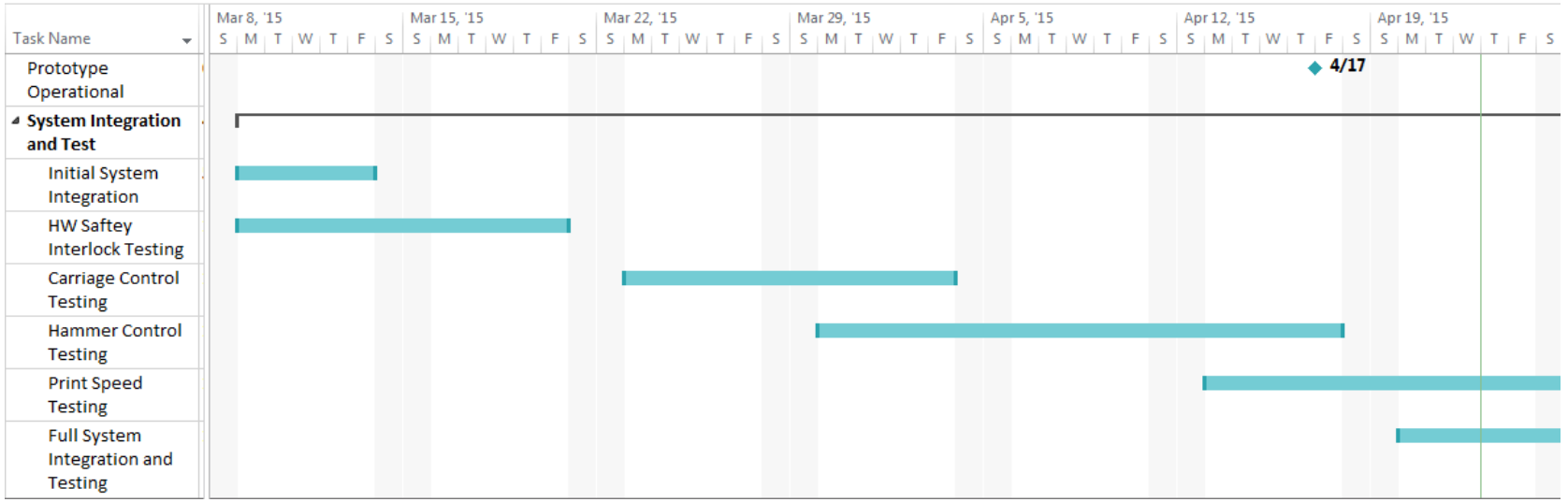
DMM	Digital Multi-Meter
OS	Operating System
POR	Power-On Reset
PSS	Pulse Sub-Scan
SD	Secure Digital
TCM	Triple Cities Makerspace
TTL	Transistor-Transistor Logic
UCS	Universal Character Set
USB	Universal Serial Bus

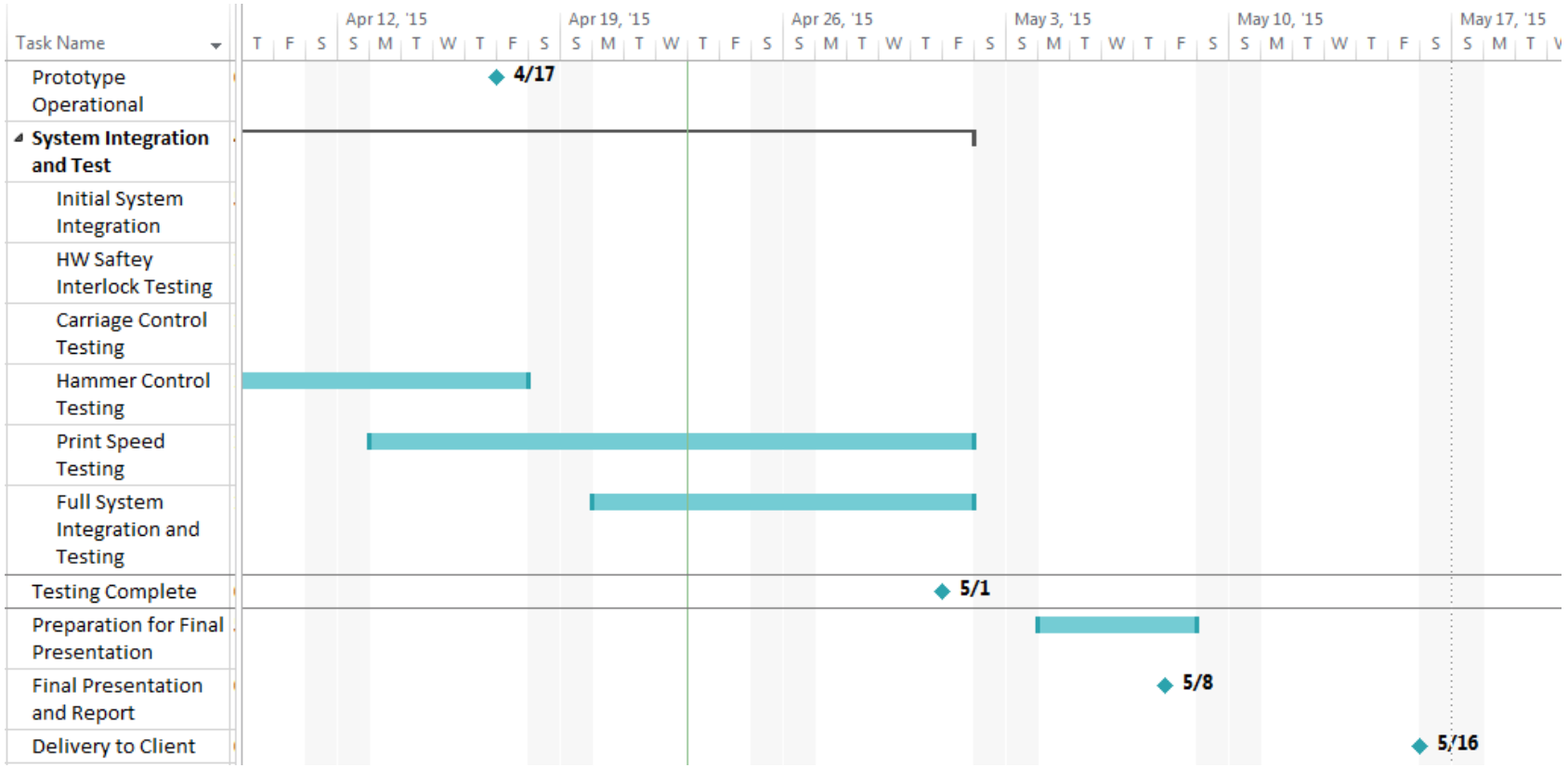
Appendix C – Detailed Schedule




Task Name
Nov 23, '14
Nov 30, '14
Dec 7, '14
T F S S M T W T F S S M T W T F S S M T W T F S S
Test Previous chipKIT SW
Choose Timing Method
Prototype Timing Software
Debug Timing Software
Choose Host PC GUI Software
Prototype Host PC GUI
Detailed Design / CDR
12/5
Production Hardware Build
Design HW Safety Interlock
Prototype and Test on Breadboard
Interim Presentation
12/12


Task Name
Jan 25, '15
Feb 1, '15
Feb 8, '15
Feb 15, '15
Feb 22, '15
Mar 1, '15
Mar 8, '15
S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F
Production Software Build
Debug Hammer Control SW
Debug Hammer Timing SW
Debug Carriage Control SW
Debug Error Handling SW
Host PC GUI Development


Task Name
Mar 8, '15
Mar 15, '15
Mar 22, '15
Mar 29, '15
Apr 5, '15
Apr 12, '15
Apr 19, '15
S M T W T F S
Prototype Operational
4/17
System Integration and Test
Initial System Integration
HW Saftey Interlock Testing
Carriage Control Testing
Hammer Control Testing
Print Speed Testing
Full System Integration and Testing


Task Name
Apr 12, '15
Apr 19, '15
Apr 26, '15
May 3, '15
May 10, '15
May 17, '15
T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T
Prototype Operational
4/17
System Integration and Test
Initial System Integration
HW Saftey Interlock Testing
Carriage Control Testing
Hammer Control Testing
Print Speed Testing
Full System Integration and Testing
Testing Complete
5/1
Preparation for Final Presentation
Final Presentation and Report
5/8
Delivery to Client
5/16

Appendix D – Preliminary Print Pseudo Code

PSS Interrupt:
- Check for Home or Extra Pulse
  - If not Home or Extra, record "PSS Occurred"
- Maintain "Start Hammer" variable (1, 2, or 3)
- Maintain "Start Char" Variable (any char in belt)

Print File:
- Receive SOF command from Host, reply
- Receive First line from host, reply
- Check Line
  - Check for characters not in belt
    - Replace with space
    - Flag error status
  - Check for hammers to throw
    - Mark all hammers that need attention in a/several bitfield(s)
- PRINT (loop marker)
- Print line
  - LINE (loop marker)
  - Set current hammer and character
  - Wait for "PSS Occurred"
  - Start "cascade"
    - CASCADE (loop marker)
    - Check hammer and char for coincidence
      - Set address bits
      - set sync active
      - wait for logic interrupt
      - set sync idle
      - Remove hammer from bitfield
    - Increment hammer by 3 and character by 2
    - Wait rest of 5us
    - GOTO CASCADE: loop 44x
  - GOTO LINE: Loop until the line is complete (bitfield empty)
- Space 0-3 lines
  - Send start carriage command (as with hammers above)
  - Wait the appropriate time for the specified spaces
  - Send end carriage command (as with hammers above)
- Get next line, reply with status
- Check line as above
- Wait remaining time for space operation to complete
- If <55ms since starting line, wait till >55ms
- GOTO PRINT: loop until EOF received from host
- EOF received, return to Idle Mode

Appendix E – Preliminary TCB Function List

- Send USB Command
- Receive USB Command
- Check one line
- PSS Interrupt
- Cascade (44 character alignment following PSS)
- Print one line
- Fire specified hammer
- Space N lines
- Load config from file
- Update config file

Appendix F – Driver Operation Modes

Operational Modes of the Hammer Driver Card

There are three distinct modes of operation of the Hammer Driver Card:

1. Service Mode
   This mode of operation is determined by sampling the service mode select input (referred to as "carriage" in the schematic) and detecting ground level. The service mode allows for testing of the command input lines, hammer output lines, and setting the unique identifier (hammer bank) of the board

Command Code	Action	Notes
0b00000NNN 0b11110NNN	Pulse hammer NNN	Pulse individual hammer
0bXXXXX001	Set ID to XXXXX	XXXXX != 0 XXXXX != 11111 Board responds to hammer addresses 0bXXXXXNNN where XXXXX determines the bank address and NNN is the hammer number. Bank Address 0b11110NNN is reserved for carriage control board.
0bXXXXX002	Sequence Hammers	Pulse each hammer from 0 to 7, repeat indefinitely.

2. Hammer Mode
   This mode of operation is determined by the service select pin being open (not connected to anything) and having an ID of 0b00001NNN to 0b11101NNN (see service mode).

Command Code	Action	Notes
0b00000000	Soft Reset	Clear all hammers
0bXXXXXNNN	Fire Hammer NNN	If 0bXXXXX000 matches board's ID the hammer indexed by NNN will fire for 1150 microseconds, then will turn off
0bYYYYYNNN	No Operation	YYYYYNNN != 0 YYYYY != board ID

3. Carriage Mode
   This mode of operation is determined by the service select pin being open (not connected to anything) and having an ID 0b11110NNN (see service mode).

   The carriage control has two pairs of complementary outputs, carriage and

skip. Carriage is driven by Hammer 0 and Hammer 1 outputs, Skip is driven by Hammer 2 and Hammer 3 outputs. Additionally Hammer output 6 will be driven on after Hammer outputs 0-3 and carriage control logic are initialized.

Setting and resetting the Carriage and Skip outputs are timed by the receipt of the idle/complement commands and are not self timed like the hammer mode.

Command Code	Action	Notes
0b11110000	Set Carriage to idle	Hammer 0 = 1 Hammer 1 = 0
0b11110001	Complement Carriage (advance)	Hammer 0 = 0 Hammer 1 = 1
0b11110010	Set Skip to idle	Hammer 2 = 1 Hammer 3 = 0
0b11110011	Complement Skip (advance)	Hammer 2 = 0 Hammer 3 = 1
0b00000000	Soft Reset	Set Carriage and Skip to idle states

Appendix G – Selloff 1 Results 2015-04-22

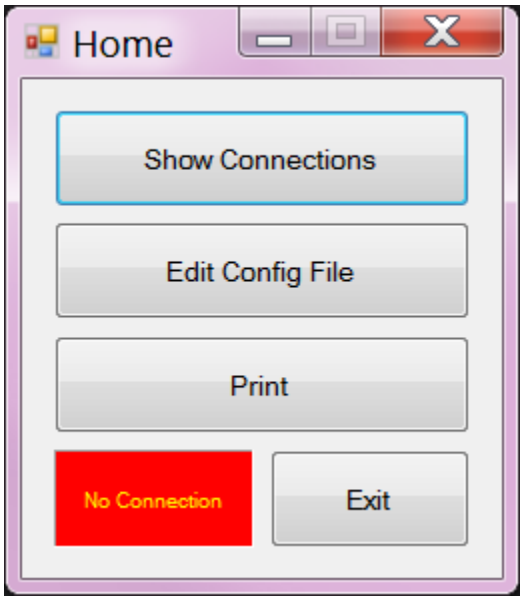
Note: These results are to be considered invalid, as described in section 4.2.1.

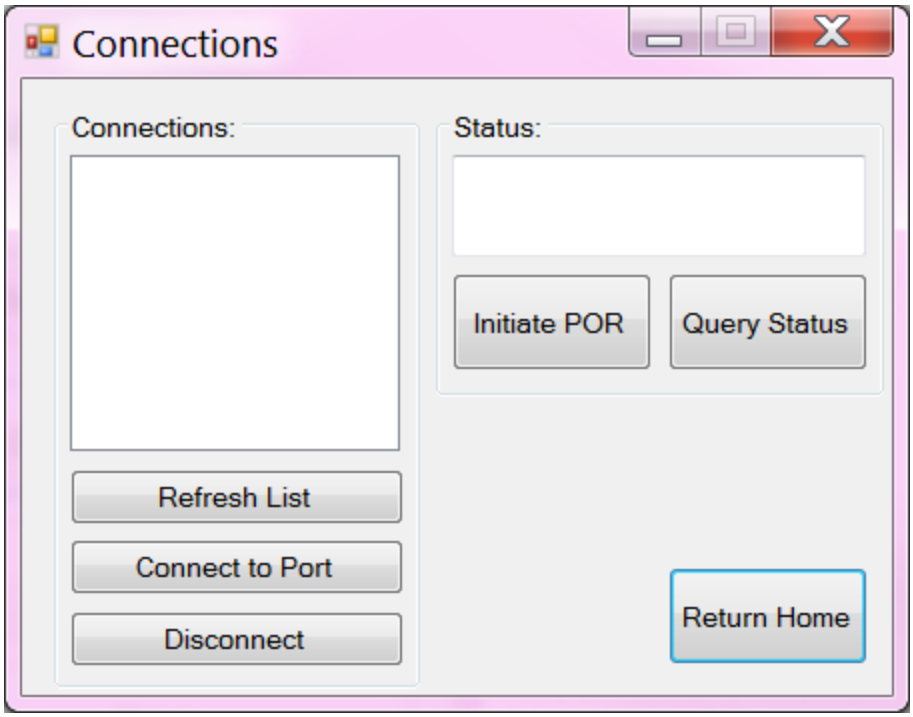
IBM 1403-N1 Printer Controller Test Data Sheet

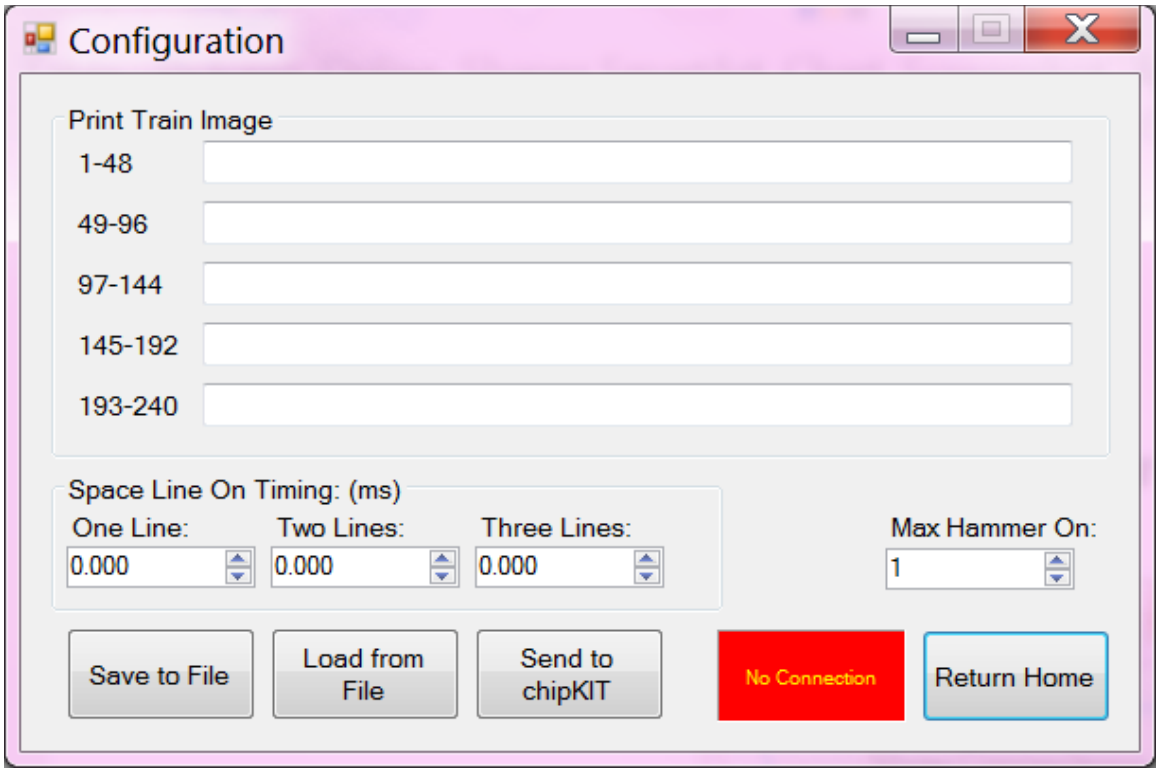
Test Number	Operator Initials	Date	Pass/Fail	Comments and Results
1000.00				
1010.00	AL		P	VALUE = 0 } SYNC PULSE
1010.01	AL		P	VALUE = 3.1 }
1020.00	AL			VALUE = 0 } ADDRESS 0
1020.01				VALUE = 3.1 }
1020.02	AL			} N/A PER A-LAN
1020.03				
1020.04				
1020.05				
1020.06	AL		P	VALUE = 0 } ADDRESS = 3
1020.07			P	VALUE = 3.1 }
1020.08				} N/A PER A-LAN
1020.09				
1020.10				
1020.11				
1020.12				
1020.13				
1020.14	AL			VALUE = 0 } ADDRESS 7
1020.15	AL			VALUE = 3.1 }
1030.00				
1040.00	AL		P	RECEIVED ERR = 0
1050.00				
1050.01				
1060.00	N/A	N/A	N/A	No testing required.
1070.00	AL		P	RECEIVED EXPECTED RESPONSE "WCP"
1080.00	AL		P	RECEIVED RESPONSE ERR 0
1080.01	AL		P	RECEIVED RESPONSE ERR 8
1090.00				
1100.00	AL		P	RECEIVED WCP ~~FULLS AS EXPECTED~~
1100.01			P	RECEIVED WCP
1110.00	N/A	N/A	N/A	No testing required.
1120.00	AL		P	STOPS & OUTPUT AS EXPECTED
1130.00				
	AL			
2000.00			P	WINDOWS XP DISPLAYED
2010.00				
3000.00	AL		P	GREEN BOX DISPLAYED
3010.00	AL		P	VERIFIED CHKPNT RESET
3020.00	AL		P	ERROR STATUS (RED INDICATOR) RECEIVED
3030.00	AL		P	LED FLASHED AS EXPECTED
3040.00	AL		P	
3040.01	AL		P	SYNCH PULSE TIMING AS EXPECTED
3050.00				
3060.00				
3060.01				
3060.02				
3060.03				
3070.00				
3070.01				
4000.00				
4000.01				

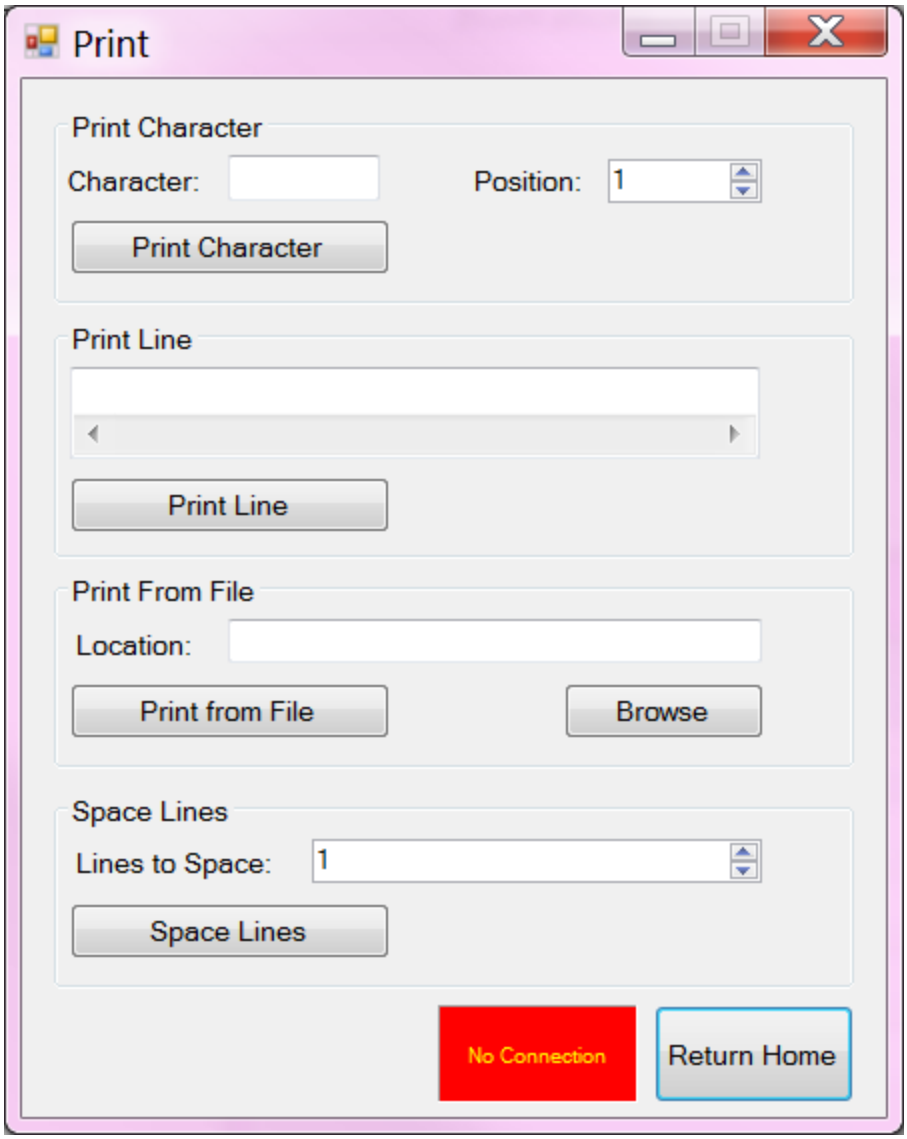
4010.00				
4010.01				
4020.00				
4020.01				
4030.00				
4030.01				
4040.00				
4040.01				
4050.00				
4050.01				
4060.00				
4070.00				
4080.00				
4080.01				
4080.02				
4080.03				

Appendix H – Host PC UI Screens

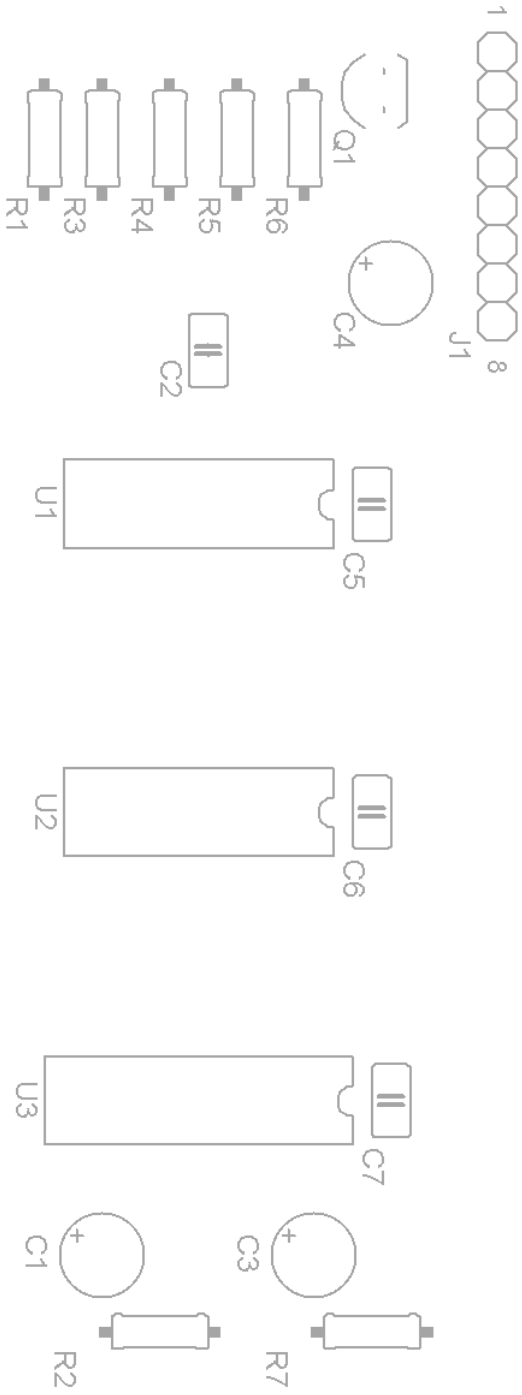


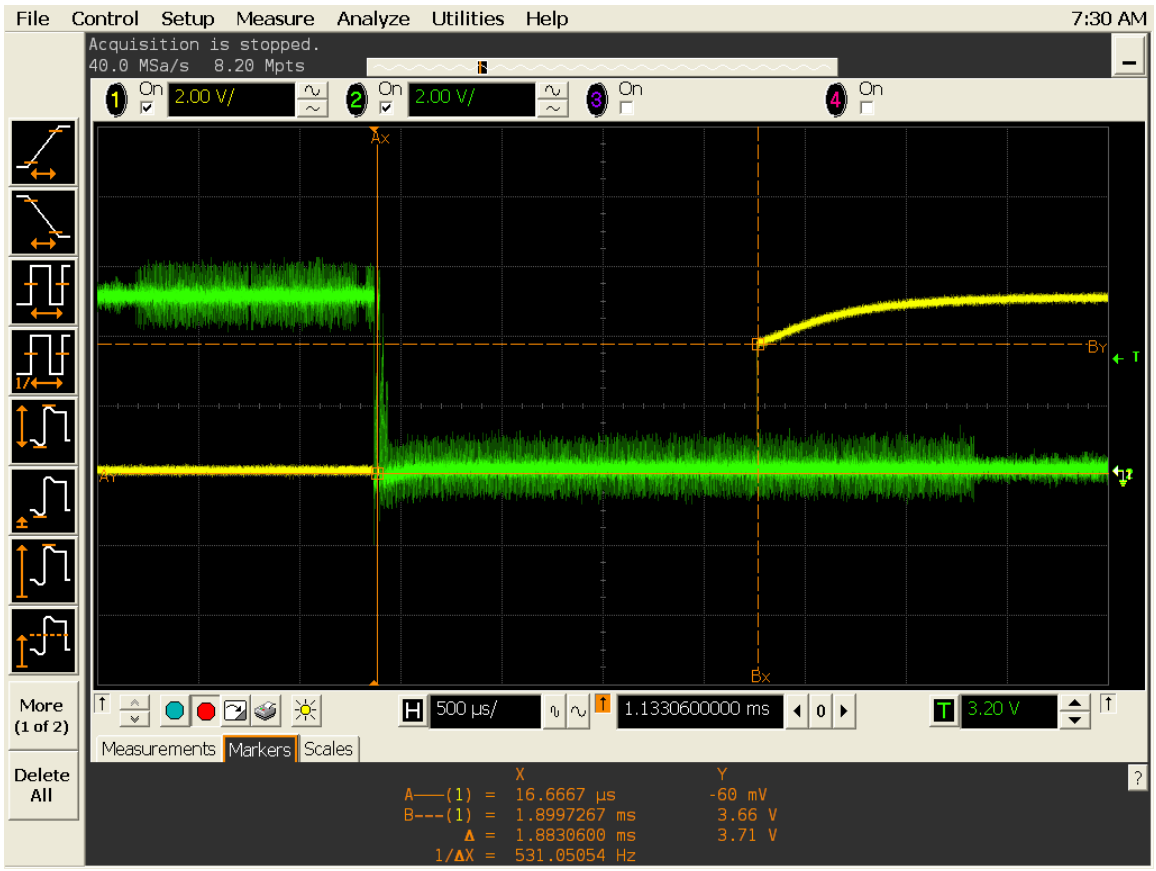

Connections
Connections:
Status:
Initiate POR
Query Status
Refresh List
Connect to Port
Disconnect
Return Home


Configuration
Print Train Image
1-48
49-96
97-144
145-192
193-240
Space Line On Timing: (ms)
One Line:
Two Lines:
Three Lines:
0.000
0.000
0.000
Max Hammer On:
1
Save to File
Load from File
Send to chipKIT
No Connection
Return Home

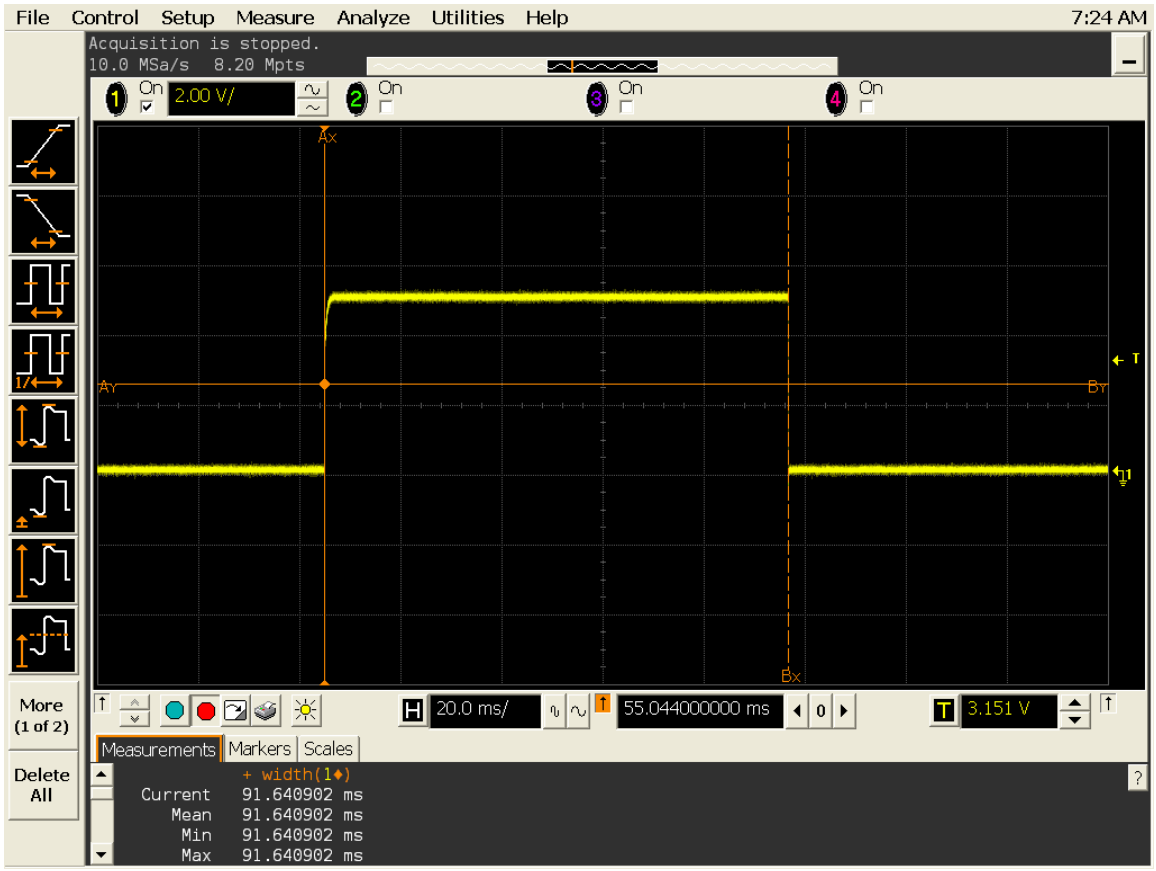

Print
Print Character
Character:
Position:
1
Print Character
Print Line
Print Line
Print From File
Location:
Print from File
Browse
Space Lines
Lines to Space:
1
Space Lines
No Connection
Return Home

Appendix I – Interlock Layout and Results





The 2ms delay between the start of hammer/carriage coincidence and the disable signal.



The 100ms disable time at the interlock output following hammer and carriage coincidence.

Appendix J – Selloff 2 Results 2015-04-29

IBM 1403-N1 Printer Controller Test Data Sheet				
Test Number	Operator Initials	Date	Pass/Fail	Comments and Results
1000.00	NH	4-29	P	
1010.00				
1010.01				
1020.00				
1020.01				
1020.02				
1020.03				
1020.04				
1020.05				
1020.06				
1020.07				
1020.08				
1020.09				
1020.10				
1020.11				
1020.12				
1020.13				
1020.14				
1020.15				N/A Per N. Hekman See previous Selloff results
1030.00	NH	4-29	P	
1040.00	NH	4-29-2015	P	None
1050.00	NH	4-29	P	
1050.01	NH	4-29	P	
1060.00	N/A	N/A	N/A	No testing required.
1070.00	NH	4-29	P	None
1080.00	NH	4-29	P	Err 8 received
1080.01	NH	4-29	P	X
1090.00	NH	4-29	P	~~ERR~~ WCF 64 rcvd
1100.00	NH	4-29	P	X
1100.01	NH	4-29	P	X
1110.00	N/A	N/A	N/A	X No testing required.
1120.00	NH	4-29	P	ERR 32
1130.00	NH	4-29	P	
2000.00	NH	4-29	P	Windows XP
2010.00	NH	4-29	P	X
3000.00	NH	4-29	P	
3010.00	NH	4-29	P	command 0x00 rcvd
3020.00	NH	4-29	P	"Pulse Timing Error" rcvd
3030.00	NH	4-29	P	command 0x08 for hammer #1
3040.00	NH	4-29	P	
3040.01	NH	4-29	P	Δ1=242μs Δ2=246 μs
3050.00	NH	4-29	P	
3060.00	NH	4-29 30	P	
3060.01	NH	4-30	P	Δ=2.02mS
3060.02	NH	4-30	P	
3060.03	NH	4-30	P	Δ=6.00 mS
3070.00	NH	4-29	P	
3070.01	NH	4-29	P	Δ1=242μ Δ2=246μ
4000.00				
4000.01				

4010.00				
4010.01				
4020.00				
4020.01				
4030.00				
4030.01				
4040.00				
4040.01				
4050.00				
4050.01				
4060.00	NA	4-29	P	
4070.00	NA	4-29	P	
4080.00	NA	4-29	P	
4080.01	NA	4-29	P	
4080.02	NA	4-29	P	
4080.03	NA	4-29	P	

Appendix K – Command Definitions

Baud	115200
data bits	8
stop bits	1
parity	none
Terminating char	none

Host PC initiated				
Function	Command	Args	Reply	Description
Print Page	PRP	[Text][N]	PRP [Error]	Following PRP, single lines of text will be sent. After printing each, the chipKIT will reply with any error code for the previous line.
End print page	END		END [Error]	Marks the end of the printed page. Chipkit replies with error of previous line.
Write Config File	WCF	[File Contents]	WCF	Overwrites the current config file stored on the SD card.
Get current error codes	ERR		ERR [Error]	Gets current error codes.
Power on reset	POR		POR [ERROR]	Cause power on reset.
Query	WHO		WPC	Query the chipKIT to make sure you're talking to the right guy. Should be done when establishing Comms.

chipKIT initiated				
Function	Command	Args	Reply	Description
Error Occurred	ERR	[Error]		Occurs when error mode is entered.