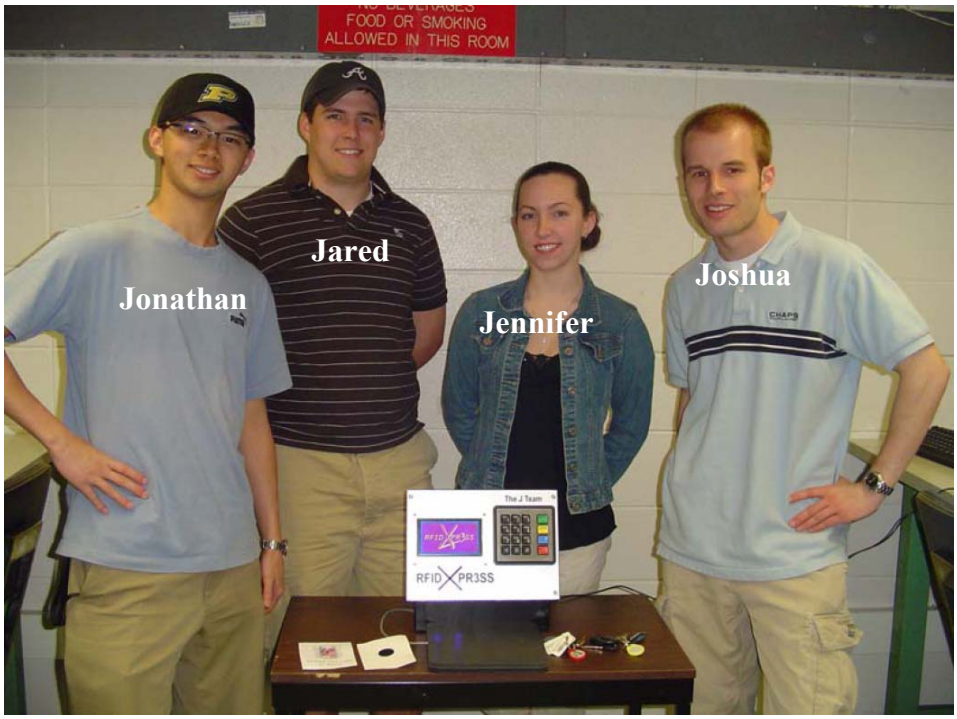
ECE 477 Final Report
Spring 2006



Team Code Name: RFID Xpress **Team ID:** 10

Team Members (#1 is Team Leader):

#1: Jennifer Tietz **Signature:** _____ **Date:** _____

#2: Jared Suttles **Signature:** _____ **Date:** _____

#3: Joshua Chapman **Signature:** _____ **Date:** _____

#4: Jonathan Chen **Signature:** _____ **Date:** _____

REPORT EVALUATION

Component/Criterion	Score	Multiplier	Points
Abstract	0 1 2 3 4 5 6 7 8 9 10	X 1	
Project Overview and Block Diagram	0 1 2 3 4 5 6 7 8 9 10	X 2	
Team Success Criteria/Fulfillment	0 1 2 3 4 5 6 7 8 9 10	X 2	
Constraint Analysis/Component Selection	0 1 2 3 4 5 6 7 8 9 10	X 2	
Patent Liability Analysis	0 1 2 3 4 5 6 7 8 9 10	X 2	
Reliability and Safety Analysis	0 1 2 3 4 5 6 7 8 9 10	X 2	
Ethical/Environmental Impact Analysis	0 1 2 3 4 5 6 7 8 9 10	X 2	
Packaging Design Considerations	0 1 2 3 4 5 6 7 8 9 10	X 2	
Schematic Design Considerations	0 1 2 3 4 5 6 7 8 9 10	X 2	
PCB Layout Design Considerations	0 1 2 3 4 5 6 7 8 9 10	X 2	
Software Design Considerations	0 1 2 3 4 5 6 7 8 9 10	X 2	
Version 2 Changes	0 1 2 3 4 5 6 7 8 9 10	X 1	
Summary and Conclusions	0 1 2 3 4 5 6 7 8 9 10	X 1	
References	0 1 2 3 4 5 6 7 8 9 10	X 2	
Appendix A: Individual Contributions	0 1 2 3 4 5 6 7 8 9 10	X 4	
Appendix B: Packaging	0 1 2 3 4 5 6 7 8 9 10	X 2	
Appendix C: Schematic	0 1 2 3 4 5 6 7 8 9 10	X 2	
Appendix D: Top & Bottom Copper	0 1 2 3 4 5 6 7 8 9 10	X 2	
Appendix E: Parts List Spreadsheet	0 1 2 3 4 5 6 7 8 9 10	X 2	
Appendix F: Software Listing	0 1 2 3 4 5 6 7 8 9 10	X 2	
Appendix G: FMECA Worksheet	0 1 2 3 4 5 6 7 8 9 10	X 2	
Technical Writing Style	0 1 2 3 4 5 6 7 8 9 10	X 8	
CD of Project Website	0 1 2 3 4 5 6 7 8 9 10	X 1	
		TOTAL	

Comments:

TABLE OF CONTENTS

Abstract

This is the final report for the J-Team's RFID Xpress senior design project. The project is a self-checkout device that uses radio frequency identification (RFID) to recognize the customer as well as the products being purchased. An external database is used to store the customer and item data and is accessed through an Ethernet connection. The project has the capability to print a paper receipt and will always e-mail a receipt to the customer. The overall design process and implementation of the project will be discussed in the remainder of this report. It will include information on the professional and technical aspects of the design process.

1.0 Project Overview and Block Diagram

Figure 1-1. RFID Xpress

RFID Xpress is a self-checkout system designed for use in grocery and retail stores. The purpose of this project is to improve upon the current UPC technology and develop a better system for retailers and shoppers. RFID technology improves upon UPC labels by allowing the customer to scan and detect items from a reasonable radial distance from the RFID reader. The item does not have to be oriented in a particular fashion in order to read a label, as with UPC. Also, RFID tags provide each item with a unique serial number embedded in the tag, which helps retailers maintain more accurate knowledge of individual product arrivals and departures. Finally, due to radial scanning of RFID tags, added security measures can be easily implemented to prevent shop-lifters from carrying items out of the store undetected.

The RFID Xpress system is controlled by a Freescale MC9S12NE64 microcontroller and consists of a slim profile external RFID reader, a graphical LCD, a PIN entry keypad, and a thermal receipt printer. The customer and product information are stored in an external database which is interfaced through UDP protocol on a Java server.

Once the customer is finished shopping, he or she initiates the checkout process by swiping a key fob with an embedded RFID tag within a few inches of the mouse pad-like receiver. The unique serial number on the key fob is used to query the external database and obtain the customer name, email address, and PIN number. The customer is then prompted to input his or her PIN on the 16-key keypad as an added security measure. Once authenticated, the customer can begin scanning products across the receiver. The serial numbers stored on each item's RFID tag are used to retrieve the product's name and price from the external database. As products are scanned, their information and other cart statistics (e.g., the number of items scanned and the total) are displayed on the LCD screen for the customer to view. After all of the customer's products are scanned, the customer chooses whether to print a receipt or to just receive one via e-mail.

The major components of the RFID Xpress system and their main communication paths are depicted in the following block diagram.

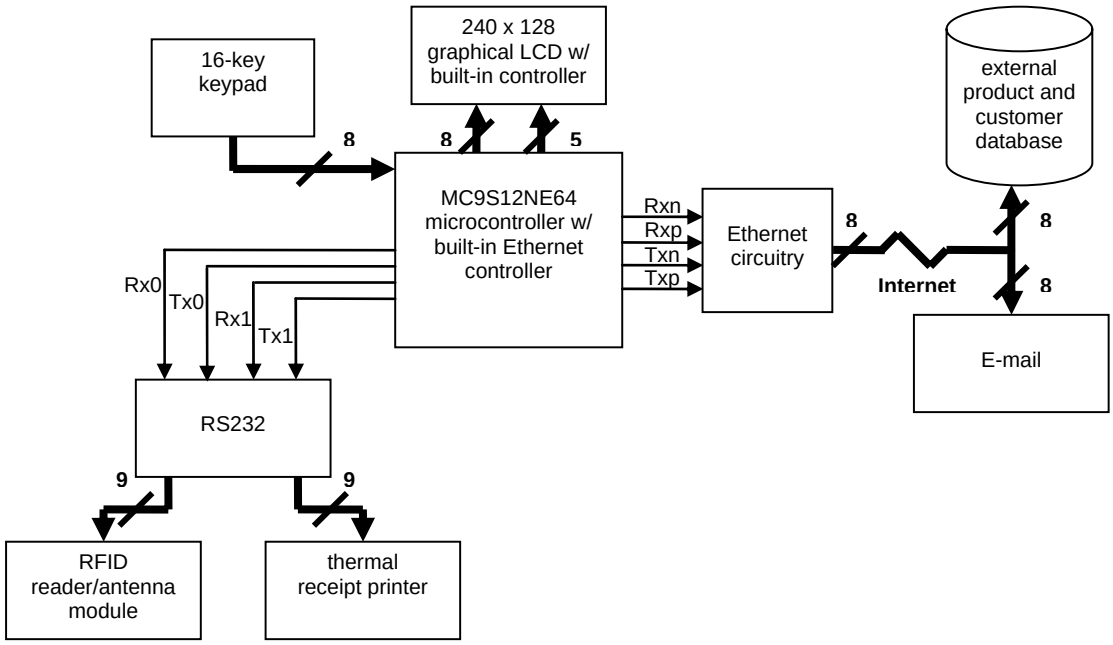


Figure 1-2. Block Diagram

2.0 Team Success Criteria and Fulfillment

All five of the J-Team's Project Specific Success Criteria (PSSC) were successfully fulfilled. The five PSSC were as follows:

1. **An ability to identify an item (and look up data on that item) based on its RFID tag.**
   This was demonstrated by scanning an RFID tag across the reader and showing that the correct item name and price were obtained from the external database and displayed on the LCD.
2. **An ability to identify a customer based on a key ring transponder and PIN code (entered on a keypad).**
   This was demonstrated by scanning a key fob RFID tag. The PIN that was obtained from the customer database is compared with the PIN entered by the customer. If they matched, the customer was allowed to proceed. If the incorrect PIN was entered, the customer was not allowed to proceed and the session was reset to the startup screen.
3. **An ability to display status information (e.g., item being scanned/price) on an LCD.**
   This was demonstrated by simply scanning items and showing their correct name and price on the LCD. Additional status information displayed on the LCD included the current date and time, total number of items in the cart, and the total price of the cart.
4. **An ability to print and/or e-mail receipts, based on customer selection (via keypad).**
   This was demonstrated at the end of the session. The customer was asked if they would like a printed receipt. If he or she selected yes on the keypad by pressing "Enter," the receipt printed below the keypad. Regardless of the print selection, an e-mail receipt was sent to the address obtained from the customer database. The receipts were verified to contain the proper items and amounts.
5. **An ability to gather product and customer data by querying an external database (using Ethernet).**
   This was demonstrated by scanning multiple key fobs and items and observing the data packets that were being received by the server over Ethernet. The serial numbers were sent, along with a command character which told the server what data was requested. Using the serial number as a key into the database, the server returned the appropriate data based upon the given command character.

3.0 Constraint Analysis and Component Selection

Design Constraint Analysis

The major design constraints of RFID Xpress were the appearance and ease of use of the product, as well as minimizing cost and PCB size. It was imperative that customers find the product simple to use. This meant that the LCD screen could not be dim and the keys on the keypad could not be small or hard to press. Other constraints included the number of pins that were required by the processor to interface with external devices, including the RFID reader, LCD, keypad, and printer. The need to communicate with an external database and send emails required a microcontroller with an integrated Ethernet module as well.

Computational Requirements

Most of the computational power was needed to operate the LCD, send and receive UDP packets to and from a remote database through the Ethernet controller, and interpret data received from the RFID reader. Additionally, the keypad needed to be polled at a high frequency to detect key presses, which also required some computational power. None of these major functions were as complex as image or audio processing, and hence there were few constraints on the processing power requirement of the microcontroller. The only time constraints seemed trivial, since such tasks as handling RFID interrupts and promptly displaying product information on the LCD could easily be handled by the fast 25 MHz clock needed by the Ethernet circuit. Should these actions be performed at a slower than desirable rate, it would simply result in customer dissatisfaction and would not cause adverse effects in the system. The largest latency was likely to be associated with accesses to the database through the Ethernet controller. Even so, the overall delay should be on the scale of milliseconds, and virtually undetectable by the user.

Interface Requirements

The finished product has no analog components in the design, and thus all I/O is digital. To interface with the LCD, 13 output pins are needed. They are used both to write to and control the LCD [1]. The LCD is provided power by the inverter, operating at 5 V with a maximum current draw of 450 mA. It dissipates a considerable amount of power, 2.25 W max [2]. The LCD logic uses 5 V, but only 15 mA [1]. The keypad is wired directly to the microcontroller

through general I/O in four column output and four row input lines. Internal pull-down resistors are required on the row input pins to pull their value low when the buttons are not being pressed. The RFID reader is interfaced via RS232 communication and is only capable of transmission. The printer also interfaces via RS232, and only receives data. Using the Maxim IC MAX3233E RS232 level translator, it will require only one input and one output pin on the microcontroller [3]. The total number of required I/O pins for these major components is 22. None of the pins in use on the microcontroller require optical isolation and have nominal sinking and sourcing requirements.

On-Chip Peripheral Requirements

One on-chip peripheral that is integral to the design is the built-in Ethernet controller. Since the MC9S12NE64 comes with only 64 kB of memory [4], it is not practical to embed the database of all products and customers into each chip, whether on the basis of memory capacity or convenience of future modification to the database. Therefore, it is necessary to communicate with an external database through a network. Another essential on-chip peripheral is the Serial Communication Interface for use with the RFID reader [5] and the thermal printer [6]. Timer channel interrupts are also used to maintain the system time and to poll the keypad.

Off-Chip Peripheral Requirements

The only off-chip peripheral needed is an RS232 level translator for the RFID reader and thermal printer. A Maxim MAX3233E was chosen because of its dual channel capabilities and 3.3 V logical high voltage level [3]. Space for a PLD and headers for flywires are also included for general use and backup I/O and logic expansion.

Power Constraints

Since RFID Xpress is designed to be stationed in a retail location where power supply should be abundant, there are no significant power constraints in the design. However, power conservation still played a role in component selection and software design. Increased power consumption can lead to component overheating, especially with voltage regulators, so care was taken to monitor the operating conditions of such components.

Packaging Constraints

As mentioned before, RFID Xpress is designed to maximize customer satisfaction. Under normal usage, none of the internal parts should be seen by the customer. This implies that no wires or PCB should be visible. Retail stores already using UPC self-checkout machines should have more than adequate space for this compact solution. Hence there are few volume or shape constraints, except that the RFID Xpress enclosure should be light and small enough to sit atop the counter of an existing self-checkout aisle. The keypad, LCD, RFID reader, and receipt printer are the only devices with which the user directly interacts, so these parts were carefully chosen to withstand normal usage.

Cost Constraints

The primary cost constraint for RFID Xpress is the minimization of production costs. This product is one of the first of its kind, and in order for it to have mass-market appeal to retail stores around the world, it has to be an affordable, yet efficient, solution. The prototype development was anticipated to cost approximately $500, the bulk of which was due to the RFID reader with built-in antenna, keypad, graphical LCD, and thermal receipt printer. However, subsequent production costs can be reduced by bulk ordering these items. For example, one graphical LCD retails for $112.63, while bulk orders of 1000+ drop the unit cost to only $45.39.

Another factor in RFID Xpress' mass-market appeal remains beyond the scope of this project - that is the current price of RFID tags and the cost associated with overhauling the retail industry and tagging all products with RFID labels. As RFID technology continues to improve, the cost of the labels will certainly decrease, making the RFID Xpress solution an even more economical one for retail stores worldwide.

RFID Xpress is unique in that existing market solutions do not utilize RFID tags for identification of products; rather, similar machines like the NCR FastLane use UPC labels to identify products and deactivate RFID tags for security purposes. FastLane units have been installed anywhere from grocery stores in Germany to Home Depots across America. Due to confidentiality, security, and company policy, it is not possible for NCR to release pricing information for FastLane; therefore, RFID Xpress has no competitive cost constraints. Regardless, because of the RFID implementation differences between RFID Xpress and

FastLane, it is apparent that RFID Xpress will be the only solution like it on the market and faces no direct competition.

Component Selection Rationale

Microcontroller: Freescale MC9S12 NE64

This microcontroller was compared with the TA neCore12 LAN DIP Module and the Rabbit RCM3750 RabbitCore Module. The design constraints required that there was an integrated Ethernet controller. All three choices had integrated Ethernet capabilities, but the later two came in packages which cost a lot more than the controller itself. Another design goal was to minimize board size, and the MC9S12NE64 microcontroller was significantly smaller than the other two, even considering the extra circuitry that was required to integrate a separate Ethernet jack. This controller also operated at a low 3.3 V, as compared to the 5 V Rabbit microcontroller [4]. Moreover, Freescale's MC9S12NE64 offered up to 40 I/O pins, far more than the 22 required for major components. The choice was even clearer when the price of the MC9S12NE64 ($7.92) was compared to the DIP module from TA ($79.00) and from Rabbit ($74.00).

RFID Reader: Intersoft USA Medium Range RFID Reader

This reader was compared with two other readers: the TI S4100 Multi-Function Reader Module and the Parallax RFID Reader Module. For the purpose of this design, long range was not required and was actually undesirable, since the reader might detect tags that weren't intended to be read yet. A medium range was ideal, as long as it didn't require the tag to be very close to the reader. The TI reader offered a very wide range in both low and high frequencies, but it didn't come with an antenna. Separate antennas increased the project cost by several hundred dollars, so this made the TI reader an unlikely solution. The Parallax reader offered a range of a mere three inches, which would be sufficient for most small merchandise, but would be insufficient for larger ones like trash cans, requiring the customer to flip the object around to scan it, hence defeating the purpose of using RFID. The logical choice was the Intersoft reader, with a decent maximum range of 19 inches, dependent upon tag size, and slim design with a built-in antenna, at a modest cost of $130.00 [5].

LCD: CrystalFontz CFAG240128D-FMI-T

There existed many options for the LCD. A display was needed to show multiple lines of text, and preferably some graphics capability to make the checkout process more appealing. All of the available character LCDs had a maximum of four lines and would not be able to handle a reasonable display of items that had been scanned. Hence, character LCDs were not an ideal solution. Other comparable graphical LCDs either cost significantly more or did not have a high enough contrast, and the CrystalFontz LCD was reasonably priced at $112.00 and provided high contrast. Also, the CFAG240128D did not have a wide viewing angle, meaning customer privacy was protected [1].

Keypad: Storm 6000 series PIN entry pad

Once again, there existed a wide array of keypads to choose from, ranging from sealed, weather-proof, and customizable to back-lit, mounted, and conductive rubber. The choice was based on two main concerns: an appealing appearance and an adequate number of keys to perform desired functions (PIN, Enter, Clear, Cancel). The Storm keypad had a pleasing appearance and a nice ergonomic design. The team acquired a sample of a 4 x 4 Grayhill series 84 keypad with customizable keys. However, the buttons were too small and were very hard to press, and the unit itself wasn't aesthetically pleasing. Therefore, the Storm 6000 series keypad was chosen for RFID Xpress [6].

4.0 Patent Liability Analysis

In the process of designing any product it is important to start off with a search for patents on which the product may infringe. In a search for such patents for this project, two were found. These two patents, as well as a commercially available product that performs a similar function, are outlined in the proceeding sections. In addition to the results of the patent search, the necessary steps to avoid patent infringement are discussed.

Results of Patent and Product Search

In the search for patents, only one was found that performed substantially the same function as this project.

United States Patent 6,547,040 is for a "self-service checkout with RFID capability" [7]. This patent was filed on April 2, 2001. The patent is assigned to NCR Corporation with John C. Goodwin, III listed as the inventor. This patent is for a self-service checkout system that weighs products with RFID labels for security purposes. The system uses a scale to weigh the product and a computer that looks up the weight and price that is stored on an RFID tag. The system compares the actual weight with the reference weight stored in the computer. If the weight is close enough the item scan is accepted. The claims include recording an indication of the product being on the scale, determining actual weight, scanning the RFID tag, obtaining the product information from the computer, comparing the actual weight with the reference weight and initiating the acceptance of the item onto the order if the difference of the weights is within a specific threshold.

One other patent that performs the function of a self-service checkout is described in United States Patent 5,992,570 [8]. This patent was filed on September 9, 1997 also by NCR Corporation. This patent says that it is for a self-service checkout device which includes a customer operated scanning device to identify each item purchased. It also includes credit card and cash accepting devices as well as a change return. It includes a terminal that displays a total amount owed. This device is also capable of being used as an ATM separately from its function as a self-service checkout. This patent claims an apparatus for processing items selected by the customer for purchase. The claims state that the transaction is done by the customer indicating each product using the scanning method and using an integrated terminal for accepting payment.

In searching for similar products, only one was found. This product is the NCR FastLane [9][10]. This product is made by the same company that holds United States Patent 6,547,040 and US Patent 5,992,570 which are mentioned previously in this paper. The actual product is a hybrid machine incorporating both RFID and barcode capabilities. It is a self-service checkout station that has an RFID reader, a barcode scanner, cash and credit card acceptors and a screen that displays pertinent information to the customer. This product seems to incorporate parts of both patents previously mentioned.

Analysis of Patent Liability

This product is fairly similar to both patents mentioned above. A potential for a doctrine of equivalents infringement was found with the first patent and a literal infringement with the second.

Even though the product being reviewed performs substantially the same function as the first patent mentioned, it is performed in a substantially different way. The claims for the first patent indicate that for each station, it includes a computer that stores all product information. The product being designed uses a microcontroller with Ethernet connectivity to gather the information from an external database. This allows several stations to access the same database. Even several stores could use one large database through an internet connection. If this were found to be substantially the same way of performing the same function, it would be an infringement under the doctrine of equivalents [11].

The product does literally infringe upon the second patent mentioned. The second patent applies to any form of self-service checkout that includes a device for identifying the items as well as their price. The product being designed includes a scanner for RFID tags, which are read when the user places items over the scanner, making the product a “user operated device.” It is because of this fact that it is a literal infringement of the second patent. Even thought the patent includes several more functions than the product being developed, it does not eliminate the infringement.

Action Recommended

Since the potential for infringement exists for the product under review, it will be necessary to eliminate this infringement before it could be manufactured. Since there is a literal

infringement with US Patent 5,992,570 [8], it would be necessary to either pay fees or purchase the patent from NCR Corporation. If this product were to be manufactured, neither of the aforementioned solutions would be ideal. It would be necessary to carefully redesign the application so that it does not infringe on any patent. Since there are substantial differences in the implementation, it would not infringe on the other patent (6,547,040) [7] so no action is needed for that patent.

In actual industry this patent search and analysis should have been completed soon after, if not during, the time when the project was only an idea. It would be substantial waste of money to complete a prototype of a product before knowing the legal issues associated with manufacturing the product.

5.0 Reliability and Safety Analysis

As the RFID Xpress is designed for interaction with the general public on a frequent basis, safety and reliability are a serious concern. The system needed to be manufactured reliably, including selection of well-tested components. It had to be properly enclosed so as to minimize the probability of harm to the customer should a malfunction occur, despite the fact that this system posed very few opportunities to compromise customer safety. The system must also adequately verify customer identification to decrease the likelihood of identity theft. A detailed Reliability Analysis of several main components in the design was performed, and their respective Mean Time to Failure (MTTF) was computed based on parameters and equations obtained in the Military Handbook of Reliability and Prediction of Electronic Equipment [12].

Reliability Analysis

In order to determine the reliability of the overall system, it was necessary to identify those components that were most likely to fail. This decision was based largely upon the common operating temperature of the components, as those that operate above room temperature are more likely to fail sooner, as well as the use of the components, since those found in power circuits are stressed more heavily.

The team identified the following four components and utilized the MIL-HDBK-217F Military Handbook to calculate each component's failure rate per 10^6 hours and the component's MTTF:

1. MC9S12NE64 Microcontroller
2. REG1117A 1 A LDO 3.3 V and 5.0 V Voltage Regulators
3. MC33063A 1.5 A Peak Boost/Buck/Inverting 12 V Switching Regulator
4. SMD High Frequency 25 MHz Crystal Unit

The following tables outline the reliability analysis of these four components. Each provides a specific equation for that component to compute the part failure rate, λ_p. MTTF was calculated according to the equation $1/\lambda_p$. The 3.3 V and 5.0 V regulators were evaluated as one component since they are the same part and have the same parameter values. The operating environment for each component was assumed to be ground benign since RFID Xpress is intended for use within a grocery or retail store, which is a non-mobile and temperature and

humidity controlled environment. Any other assumptions made in the calculations were clearly stated. Useful parameter definitions are found below in Table 5-1.

λ_p = part failure rate
λ_b = base failure rate
C_1 = die complexity failure rate
C_2 = package failure rate (utilizes N_p)
N_p = number of functional pins
π_T = temperature factor
π_E = environment factor
π_Q = quality factor
π_L = learning factor

Table 5-1. MTTF Parameter Definitions

For failure analysis, two levels of criticality were identified for RFID Xpress. Those levels, along with their explanation and maximum allowable probability, are detailed below in Table 5-2 and are discussed at length in the reliability analysis and FMECA.

Criticality Level	**Description**	**Maximum Probability**
High	Customer identification or safety compromised	$\lambda_p < 10^{-9}$
Low	Possible customer dissatisfaction	$\lambda_p < 10^{-5}$

Table 5-2. Criticality Levels

MC9S12NE64 microcontroller [13]

$$\lambda_p = (C_1 \pi_T + C_2 \pi_E) \times \pi_Q \times \pi_L \text{ failures per } 10^6 \text{ hours}$$

Parameter	Value	Justification
C_1	0.28	16-bit MOS microcontroller (MIL-HDBK-217F, Section 5.1)
C_2	0.041	80-pin SMT package $C_2 = 3.6 \times 10^{-4} (N_p)^{1.08}$, $N_p = 80$ (MIL-HDBK-217F, Section 5.9)
π_T	3.1	Digital MOS device, maximum operating temperature, $T_j = 125°C$ (MIL-HDBK-217F, Section 5.8)
π_E	0.50	Assumes ground benign environment, G_B (MIL-HDBK-217F, Section 5.10)
π_Q	10	Commercial product screening level (MIL-HDBK-217F, Section 5.10)
π_L	1.0	Generic device in production ≥ 2 years (MIL-HDBK-217F, Section 5.10)

Table 5-3. MC9S12NE64 Microcontroller MTTF Parameters

$\boldsymbol{\lambda_p}$ = 8.89 failures per 10^6 hours

MTTF = 112549.24 hours ≈ 12.85 years

The microcontroller was selected for reliability analysis due to its central role in the entire system. It is responsible for controlling the integrated peripherals such as the RFID reader, thermal printer, LCD, Ethernet, and keypad, as well as containing all of the software logic for the design. Therefore, proper operation of the microcontroller is integral to the success of RFID Xpress. While the microcontroller is relatively important compared to other components, the system does not pose any serious safety hazards to the customer should a failure occur. It is therefore considered a low criticality level, and a part failure rate of 8.89×10^{-6} hours is adequate. Also, taking into consideration the fact that the microcontroller will not be in continuous intensive operation during idle states and that it will normally be operating below the maximum rated temperature, this calculated MTTF is an upper-bound that will likely not pose a serious threat to system operation, and failure is reparable.

REG1117A 1A LDO 3.3 and 5.0 V Voltage Regulators [14]

$$\lambda_p = (C_1 \pi_T + C_2 \pi_E) \times \pi_Q \times \pi_L \text{ failures per } 10^6 \text{ hours}$$

Parameter	Value	Justification
C_1	0.01	Assumes 1 to 100 transistors in linear MOS device (MIL-HDBK-217F, Section 5.1)
C_2	0.0012	3-pin SMT package $C_2 = 3.6 \times 10^{-4} (N_p)^{1.08}$, $N_p = 3$ (MIL-HDBK-217F, Section 5.9)
π_T	58	Linear MOS device, maximum operating temperature, $T_j = 125°C$ (MIL-HDBK-217F, Section 5.8)
π_E	0.50	Assumes ground benign environment, G_B (MIL-HDBK-217F, Section 5.10)
π_Q	10	Commercial product screening level (MIL-HDBK-217F, Section 5.10)
π_L	1.0	Generic device in production ≥ 2 years (MIL-HDBK-217F, Section 5.10)

Table 5-4. REG1117A Voltage Regulator MTTF Parameters

$\boldsymbol{\lambda_p}$ = 5.81 failures per 10^6 hours

MTTF = 172235.62 hours ≈ 19.66 years

The 3.3 V and 5.0 V regulators were selected for reliability analysis because they provide two of the three supply voltages to system components and they are known to operate at higher than ambient temperatures. A thin copper strip was added to the heat dissipation pad of both voltage regulators to decrease their operating temperatures, as they increase noticeably when the circuit draws more current. The 3.3 V signal powers the microcontroller and RS232 level translator, and the 5.0 V signal powers the LCD. Due to the fact that failures on either voltage regulator would simply result in improper operation (potentially including no operation at all) of the microcontroller, serial communication, and LCD, this component is rated at a low criticality level that would cause customer dissatisfaction. Therefore, the part failure rate of 5.81×10^{-6} is acceptable. However, while very rare, the possibility of a short to ground on either voltage regulator does exist. This could potentially result in part overheating or a fire, which could put the customer at risk. Due to the fact that the customer only interacts with the device through the keypad, which would not quickly overheat, the risk to the customer would be low.

MC33063A 1.5 A Peak Boost/Buck/Inverting 12 V Switching Regulator [15]

$\lambda_p = (C_1 \pi_T + C_2 \pi_E) \times \pi_Q \times \pi_L$ failures per 10^6 hours

Parameter	Value	Justification
C_1	0.01	Assumes 1 to 100 transistors in linear MOS device (MIL-HDBK-217F, Section 5.1)
C_2	0.0034	8-pin DIP package $C_2 = 3.6 \times 10^{-4} (N_p)^{1.08}$, $N_p = 8$ (MIL-HDBK-217F, Section 5.9)
π_T	180	Linear MOS device, maximum operating temperature, $T_j = 150°C$ (MIL-HDBK-217F, Section 5.8)
π_E	0.50	Assumes ground benign environment, G_B (MIL-HDBK-217F, Section 5.10)
π_Q	10	Commercial product screening level (MIL-HDBK-217F, Section 5.10)
π_L	1.0	Generic device in production ≥ 2 years (MIL-HDBK-217F, Section 5.10)

Table 5-5. 12V Switching Regulator MTTF Parameters

$\boldsymbol{\lambda_p}$ = 18.02 failures per 10^6 hours

MTTF = 55503.14 hours ≈ 6.34 years

The 12 V switching regulator was selected for reliability analysis because it provides power for the RFID receiver, which is an integral part of the design. It also operates at higher than ambient temperatures, which makes it a greater candidate for failure. The relatively large number of resistors, capacitors, and inductors associated with the 12 V power circuit (compared to the other power circuits) also increases the opportunity for failure. If power somehow became shorted to ground in the circuit, excessive heat may be generated that could possibly overheat components and cause a fire, resulting in injury to the customer. In this high criticality situation, a part failure rate of 18.02×10^{-6} is unacceptable. However, in the more likely event that the voltage regulator fails to output the proper voltage or shuts down entirely, the RFID reader would simply cease to operate. While this would render the entire unit useless until it was repaired and would result in customer dissatisfaction, it poses no serious risk to the customer. In this low criticality situation, the calculated part failure rate is only slightly below an acceptable value. However, considering that this part will never operate at the maximum temperature,

which was used to determine the part failure rate, it is estimated that the true failure rate of the part under normal operating conditions will fall within a safe range.

SMD High Frequency 25 MHz Crystal Unit [16]

$$\lambda_p = \lambda_b \times \pi_Q \times \pi_E \text{ failures per } 10^6 \text{ hours}$$

Parameter	Value	Justification
λ_b	0.027	25 MHz (MIL-HDBK-217F, Section 19.1)
π_Q	2.1	Assumes lower than MIL-SPEC (MIL-HDBK-217F, Section 19.1)
π_E	1.0	Assumes ground benign environment, G_B (MIL-HDBK-217F, Section 19.1)

Table 5-6. 25 MHz Crystal MTTF Parameters

$\boldsymbol{\lambda_p}$ = 0.0567 failures per 10^6 hours

MTTF = 17636684.30 hours ≈ 2013.32 years

The 25 MHz crystal unit was selected for reliability analysis due to its sensitivity and the sensitivity of the components within the oscillator circuit. It is possible that simply probing the traces in the oscillator circuitry could cause permanent damage and result in erratic operation of the microcontroller. However, should a failure actually occur with the crystal unit, the effects would be limited to the microcontroller and the components that depend on proper timing. It is likely that the internal bus clock would not operate properly, thereby distorting the baud rate and causing serial communication to break down. The customer would not be at serious risk in this situation, so the failure rate of 0.0567×10^{-6} is not only acceptable, but results in the longest MTTF out of all the components identified as high-risk.

Overall Design Results

The results of this reliability analysis are organized in Table 5-7. The shortest MTTF is clearly the 12 V switching regulator, which seems logical. The microcontroller, 3.3 V, and 5.0 V regulators all have relatively similar MTTF values. Should a failure occur in the system, it is clear that these components are most likely to be at fault. The 25 MHz crystal, on the other

hand, is estimated to be orders of magnitude more reliable than the other components and won't be any more likely to experience failure than any other random minor component in the design. While the failure rate for the microcontroller and voltage regulators may seem high, it is important to remember that the worst-case operating temperatures were used in these calculations and that measures have already been taken to increase heat dissipation in necessary components. Therefore, it is believed that the calculated MTTF of the overall design is reasonable and safe.

Component	λ_p per 10^6 hours	MTTF (hours)
MC9S12NE64 microcontroller	8.89	1.12549×10^5
REG1117A 3.3 V, 5.0 V regulator	5.81	1.72235×10^5
12 V switching voltage regulator	18.02	5.5503×10^4
25 MHz crystal unit	0.0567	1.7636684×10^7

Table 5-7. RFID Xpress Reliability Analysis Results

Failure Mode, Effects, and Criticality Analysis (FMECA)

For the Failure Mode, Effects, and Criticality Analysis [17], the overall schematic was partitioned into seven functional blocks: 5.0 V Power Circuit (A), 3.3 V Power Circuit (B), 12 V Power Circuit (C), Microcontroller and Oscillator Circuit (D), Ethernet and Physical Transceiver Circuit (E), RFID and Printer RS232 Circuit (F), and Keypad and LCD Interface Circuit (G). Schematics of each block can be found in Appendix C. The potential failure modes of each block are examined in the FMECA worksheet in Appendix G, including possible causes and effects of failure, methods of detection, and criticality.

The criticality levels in this analysis correspond to those mentioned previously in Table 5-2. A failure with a criticality level of "high" ($\lambda_p < 10^{-9}$) either compromises customer safety or customer identity. A failure with a criticality level of "low" ($\lambda_p < 10^{-5}$) is likely to cause customer dissatisfaction.

Through this safety analysis, it was determined that the MTTF values obtained for the identified components were within acceptable limits for the predefined levels of criticality. The potential modes of failure by functional block were explored in the FMECA analysis. The RFID Xpress system was designed to improve the customer's retail shopping experience, all the while maintaining adequate levels of customer safety and security.

6.0 Ethical and Environmental Impact Analysis

Ethical Impact Analysis

The development of RFID Xpress has presented many interesting ethical questions. RFID technology as a whole is under fire from many sources because of privacy issues, as well as some more radical religious beliefs. Because RFID Xpress aims to bring this technology into all types of retail shopping locations, it will legitimately become a daily occurrence to interact with the technology, which means that these concerns must be considered with the utmost importance. There are also ethical issues that must be considered due to the possibility of identity theft and abuse of the self-checkout concept. As a system that allows users access to their store accounts without direct supervision, identity verification is integral to the security of customers' personal information.

RFID Privacy Considerations

Recently, RFID technology has taken on many negative connotations. It is being compared to "Big Brother" control [18], and in some radical religious circles, it is being referred to as the so called "mark of the beast" [19]. Though commercial applications are not the main concern of these fears, privacy is still a huge issue. On a small scale, RFID technology can be used to identify specific items at a very short range by reading a unique serial number embedded in a small integrated circuit. On a larger scale, it can be used to track entire shipments of products around the United States and the world. This has major implications for privacy concerns in the coming years.

In theory, any RFID tag can be read from a reasonably long distance if the tag and the reader are compatible and powerful enough. This was considered when designing RFID Xpress, and it was determined to be a significant problem. First, an individual could effectively scan another's shopping cart and receive the serial numbers of all of the tags in the cart at that moment. This was only of minimal concern, since the serial number is unique to every individual item. Hence, without access to the main database, the serial number would be useless in determining any information about the products. However, it is a possibility that a person could place incorrect RFID tags on products or in other people's carts to trick the RFID Xpress into thinking that a product was a different item. Some solutions on the market today use either

video or weight matching technology to make sure that the item being scanned corresponds to the RFID on it.

Because the tags are unique to each item, maintaining an inventory count is unnecessary, and the serial number may just be removed from the database once it is purchased. To prevent the possibility of items being unknowingly scanned after purchase, a solution was considered that involved adding a large magnetic pad to the design. This pad would be used to destroy the RFID circuit after the purchase was made. For this version of RFID Xpress, no video or weight verification is being done, nor are the tags being destroyed post-purchase, but they are all considerations for further iterations of the RFID Xpress system.

Another situation that is a very high risk to customers is the fact that by the same logic above, although unlikely, a malicious person could somewhat easily attain the unique serial number on a given key fob used to identify a customer. This is a rather serious concern in this iteration, since a 4-digit PIN is not unreasonable to guess or even just watch and record. If, however, the key fob were coupled with some other, more secure, verification system, such as biometric scanning, then the likelihood of this occurring would decrease significantly.

There are also two oddities with the RFID Xpress system that will require owner understanding, and thus documentation and warning stickers will be used to ensure that the owners don't place the unit or themselves in danger. First, the 12 V power supply will be fed to the RFID reader via an unused pin of theDB9 connector. Hence, should an owner accidentally attempt to connect a printer to the wrong port, they may encounter a situation where they are feeding 12 V into an inappropriate connection. A warning label will be placed near the port to remind them that it is a proprietary connection. Another connection that requires a warning label is the RJ45 Ethernet connection. This first version of the RFID Xpress system uses a proprietary CAT5 Ethernet cable. This is due to a small mistake made in the PCB design. This will be corrected in further iterations, but for now, a warning label will alert the users not to use a standard or cross-over Ethernet cable in order to avoid malfunctions in the Ethernet communication.

Environmental Impact Analysis

The development of RFID Xpress has also brought about a few considerable environmental concerns. In the manufacturing stage, many environmentally sensitive procedures are used to

produce the system. Energy usage was a concern during the normal operation of the system. Finally, the disposal of RFID tags has raised a great deal of concern in the United States as well as the world. The disposal of the RFID Xpress system may also pose some environmental challenges.

Manufacturing Considerations

RFID Xpress consists of an LCD, Keypad, RFID Reader, Thermal Printer, and a PCB mainboard. Due to the large number of printed circuit boards in the system, PCB fabrication is a great concern of the design team. PCB fabrication involves many steps that produce hazardous byproducts, and these materials must be handled and disposed of carefully [20]. Though the RFID Xpress design team is not directly responsible for PCB fabrication, it is important to select a reputable and environmentally friendly fabrication company to manufacture the printed circuit boards. RFID Xpress was designed to minimize PCB board size, which will help to reduce the amount of industrial waste generated during the manufacturing process. Surface mount parts were used in almost every possible location, and the PCB was redesigned multiple times to compact the placement of the parts and to minimize the trace lengths.

Normal Operation Considerations

During the normal operation of the RFID Xpress, there are no major environmental issues to consider. Of course, the system aims to conserve energy as much as possible. Hence, it uses a minimal supply voltage that can power the entire system. The system also disables all unused functionality in order to conserve energy.

Another slight concern during normal operation is customer exposure to radio frequency radiation. The RFID reader, though rather weak, is still a considerable source of radiation. The design uses passive RFID tags, which means that the power to transmit the tags serial number is provided by the radio frequency radiation coming from the reader. There has been much discussion and research done to determine the long term effects of RF exposure, and the basic consensus is that while RF radiation is inherently capable of causing damage to human and animal body tissue, the levels of radiation being produced by most products are nominal enough to be considered safe [21]. The level of RF radiation emitted from a typical RFID tag is well

below that of a typical cellular phone, and can hence be assumed to be as safe, if not safer than that technology, which has been thoroughly shown to be harmless to humans [22].

Recycling and Disposal

There are two cases to consider for recycling and disposal of the RFID Xpress system. The first and perhaps easiest to handle is the disposal of the actual system itself. When a retail store decides to upgrade to a newer system, or in the unlikely event that a failure causes the company to replace the product, average disposal methods for e-waste should be sufficient to ensure the safety of the environment. Due to the lead and mercury that can be present in the PCB, solder, and certain components of the current design, the main board and components on it must be disposed at special facilities [23]. A sticker will be placed inside the packaging, as well as on the main board, suggesting that the owner recycle the PCB with a reputable company. The provided documentation will also suggest the recycling of the PCB, as well as everything else in the system, with reputable companies.

The second case to consider is the disposal of the RFID tags themselves. An RFID tag is simply a small electronic circuit that is powered through RF radiation. The problem that this poses is that the metal in the circuit can cause damage and contamination in existing recycling systems for plastic, glass, and other products [24]. The current standard tags (UPC labels) are completely harmless, but RFID tags being attached to every product in a super market will cause significant issues to these existing recycling establishments. On the positive side though, the system is designed to use passive RFID tags. Since passive tags don't actually contain batteries, they can be disposed of in standard landfills and using standard procedures. Conversely, should active RFID tags ever be implemented, they do contain small batteries that must be disposed of properly. Active tags will likely never be implemented though, due to their high cost and the lack of any worthwhile advantages over passive tags.

7.0 Packaging Design Considerations

The RFID Xpress has an LCD screen, a keypad, a receipt printer and a scanning platform as its external user interfaces. It is important to make sure the LCD screen is large enough to read and that the keypad buttons are convenient for anyone to press. The printer is placed in an easy to reach location. Everything is arranged in a neat way that makes sense to the average customer and is easy for them to use. The RFID antenna is housed inside the scanning platform. The range of this antenna is important. If the range is too long, it may read things it should not, but if it is too short, the user might have a difficult time scanning an item. The network, RS232, and power cables are external interfaces that will not be used by the customer. These are hidden in the back and not easily accessible to the average shopper. The RFID Xpress should be packaged in something that is sturdy and will hold up over time. This product has the potential to be used several times per hour, so it should hold up to any abuse it receives. It is possible for the internal components to heat up significantly with continued use. For this reason, it may be necessary to include ventilation in the casing near the potentially hot components.

Commercial Product Packaging

Product #1

NCR Corporation makes a product called *FastLane-Mini* [26]. The FastLane-Mini (Figure 7-1) is a self-checkout station that consists of a 15" LCD touch screen, a bar code scanner, a bag holder with a scale, a money accepter and dispenser and a credit card transaction unit. NCR has made and is testing an RFID module to be added to the FastLane, but at this time it will only be used for deactivating security tags [9]. The packaging is a large metal box with everything mounted onto it. The good thing about this package is that it is an all-in-one solution. Everything one would ever need is on this stand. The entire package does not take up a whole lot of space with respect to traditional check out lanes. The FastLane design includes several payment options when really only one is needed per transaction. This is addressed in the design by integrating all the payment options into an account in the database, accessed by the serial number on the key fob. For the application RFID tags are used to identify products, as opposed to UPC labels.

Figure 7-1. FastLane-Mini

RFID Xpress uses the same basic layout of components as the NCR FastLane-Mini. The LCD is placed above the RFID scanner in a place that is very easy to view. The keypad for entering a PIN is located to the right of the LCD, similar to the design of NCR's product. Many of NCR's extra peripherals are excluded from the RFID Xpress design. RFID is solely used for customer and product identification, and online bill payment is encouraged. RFID Xpress is also intended for more of a desktop design and would affix to an already constructed desk, table, or checkout aisle.

Product #2

Fujitsu makes a self-checkout station called the *USCAN1* [27]. The USCAN1 (Figure 7-2) is the closest to our design. The USCAN1 incorporates a touch-screen, a UPC reader, cash and change accepters, a keypad and card swipe, a coupon accepter and cash and change dispensers much like the NCR FastLane-Mini. The LCD and keypads are in very convenient locations. A customer can scan items and read the feedback on the screen at the same time. The only flaw in this design is that the cash dispenser is quite low and could cause customers discomfort from bending over.

Figure 7-2. USCAN1

The RFID Xpress aims to optimally place integrated components. The cash dispenser and many of the other peripherals are not used because the customer will just swipe a store-issued key fob. This key fob will be linked to the customer's account and the method of payment previously decided upon will be used. RFID Xpress does not accept cash payments. Fujitsu's solution is also a stand-alone unit, while RFID Xpress will have to be placed on a desktop.

Project Packaging Specifications

The goal of the RFID Xpress packaging was to arrange components in the most convenient, easy-to-use fashion. It is divided into two major parts. The RFID reader is integrated with an antenna and packaged in a low-profile, pad-like package that is 215 x 215 x 19 mm. An RS232 cable connects the reader/antenna combo to the main unit. The main unit is an OKW DATEC TERMINAL L. It is approximately 11" x 12" x 4", which is just about perfect for the specifications. The face of the main unit slopes back so that the LCD and keypad face the user. The main unit has an LCD screen on the left side of its face and a keypad on the right (see

Appendix B). The receipt prints out below the keypad on the right side of the front of the box. Since the majority of people are right handed, this setup will be the most convenient to the most people. RFID Xpress does not incorporate the exact placement of the main unit and the scanning pad. This leaves room open to the individual retailers to decide where exactly to put the individual units. This placement will be limited by the three foot length of the RS232 line. The plastic box used for the main unit required that holes be cut into it to accommodate the LCD screen, the keypad, the printer and all the external ports on the PCB.

PCB Footprint Layout

The size of our printed circuit board (PCB) (see Appendix D) is only limited by the size of the enclosure for the main unit. With the LCD screen and the keypad, the box needs to be at least 10" x 9" x 2". The PCB is well under this size so it is not a significant constraint. A surface mount microcontroller was chosen because it was readily available and also because it should be fairly simple to work with on the PCB. The LCD is wired to the PCB via several headers. The RJ45 jack has a few different footprint options, but they are all very similar so the most accessible footprint was used. Taking into consideration the current component and header footprints, as well as additional space allotted for undetermined passive components and routing, the size of the PCB was estimated to be 88 x 56 mm.

Packaging Summary

The main goal of the RFID Xpress packaging was to make it as simple for the user as possible. There were virtually no size limitations as this is a desktop design. There also was not a weight restriction as the product would be secured to a fixed structure in an actual retail application. It did have to be slightly rugged to withstand continued use. NCR's FastLane-Mini and Fujitsu's U-SCAN are both examples of self-checkout systems that are successful and currently deployed in retail stores around the country. It would be ideal for RFID Xpress to emulate these products in the way they are set up and made simple for the user. However, the design of RFID Xpress has been modified to exclude several of the peripherals that these units currently use and implement RFID product and customer identification. It should technically be possible to take this product and expand it to include all the functionality of these two existing solutions. RFID Xpress is being used to show the feasibility of either adding RFID capabilities

to the already existing units or using RFID technology exclusively. This would eliminate the need for UPC labels on products and the bulky bar code readers on the existing solutions, as well as increase the efficiency of self-checkout lanes in general.

8.0 Schematic Design Considerations

The design of the schematic for RFID Xpress required considerations of all of the major components, including the multiple power supply circuits, the microcontroller, RFID reader, LCD, keypad, and printer. The theory of operation explains each of these modules in greater detail, including considerations that were specific to the design of RFID Xpress.

Theory of Operation

Power Supply Circuit

Due to the need for several different regulated supply voltages ranging from 3.3 V to 12 V, the design of the power supply circuit was very important. Since the printer chosen for the design was provided with its own wall-wart, the design utilizes a second power cube to produce these other signals. The printer is powered by a 9 V DC regulated supply. Rather than risk performance loss or power issues by running the entire circuit from this source, a 7 V DC, 1.6 A wall-wart is used for the circuit components. The printer supply is rated at only 1.2 A, which is on the low end of acceptable values if this source was to be used for the overall design. The 7 V supply is connected to a TI dc-dc switching regulator [15] that, with the help well-documented circuitry, increases the output voltage to a regulated 12 V DC at 175 mA. The 7 V supply also provides power for two TI REG1117A low drop-out voltage regulators [14] that decrease the supply down to 5 V at 1.5 A and 3.3 V at 1.5 A. Since only 80 mA at 12 V are needed for the RFID reader [5], 517 mA at 5 V are needed for the LCD [1][2] and keypad [6], and 285 mA at 3.3 V are needed for the microcontroller [13], the supplied voltages and currents will more than suffice. The parts associated with the power supply circuits were selected in accordance with the typical operating diagrams provided in the voltage regulator data sheets.

Microcontroller (Freescale MC9S12NE64)

The MC9S12NE64 80-pin microcontroller has been chosen for its selection of features that closely meet the design constraints of the RFID Xpress system. As suggested by the data sheet, it operates at 3.3 V and a frequency of 25 MHz [13], neither of which will have a severe impact on the power dissipation of the system. The presence of the background debug module (BDM) is ideal for in-circuit programming and debugging. The circuitry for the oscillator and the BDM

is designed in accordance with the microcontroller data sheet [13] and utilizes 1% precision components to ensure reliable operation.

One major criterion for microcontroller selection was the need for Ethernet connectivity to send emails and access external databases. The microcontroller's integrated Ethernet capabilities allowed it to operate as a UDP client via an RJ-45 connector to communicate with a Java server, which sends emails to customers and queries the databases. The circuitry used in this design was adapted directly from the Ethernet physical transceiver circuit on the microcontroller data sheet [13]. The RFID reader utilizes the SCI1 port to receive and transmit data to the microcontroller as the customer key fobs and products are scanned. When the scanned serial number on the RFID tag is transmitted to the microcontroller from the RFID reader, the external database is queried to find the customer or product information. Once that is successfully located and transmitted back to the microcontroller, it is transferred to the LCD through 8 data bus pins on Ports G, H, and J.

The 16-key PIN entry keypad is wired directly to the microcontroller through general I/O Ports H and J. Similar to the RFID reader, the thermal printer receives data to print receipts from the SCI0 port. That data will only be transmitted to the printer if the customer selects to print a receipt by pressing the appropriate button on the keypad. Both the RFID reader and printer are wired to a MAXIM IC MAX3233E RS232 level translator to ensure that the signals are properly converted to the required levels by the microcontroller and RS232 devices.

RFID Reader (Intersoft USA Medium Range WM-RO-MR-2 RFID Reader)

The RFID reader is intended for reading and decoding passive RFID tags. The reader and antenna are housed within a rugged, yet slim, enclosure that lays flat on a table-top surface. The reader operates on 12 V regulated DC with a maximum of 80 mA [5], which is supplied by the power circuitry as explained earlier. The estimated supply current of 175 mA at 12 V should be more than adequate for proper operation. The reader communicates with the microcontroller via RS232 protocol, and therefore a DB9 connector is used to route signals to a MAX3233 level translator [3] and then on to the microcontroller.

An interrupt is triggered when a valid key fob or product tag is scanned and detected within range of the reader, and a built-in green LED on the reader is illuminated. The decoded 12-character serial number (preceded by a colon start character) is transmitted to the microcontroller

via the SCI1 port [5]. The microcontroller then sends a UDP packet via the Ethernet to query the external database on the serial number and locate the customer or product information.

LCD (Crystalfontz CFAG240128D-FMI-T 240 x 128 Graphical LCD)

The LCD utilizes an 8-bit parallel data bus to receive data from the microcontroller. The signals are split between Port G (0 through 3), Port H (0 through 2) and Port J(0) of the microcontroller. While this isn't exactly ideal, it came as a necessity when it was determined that the ATD Port could only be used for digital input. The LCD module [1] has a built-in controller with a 4 kB external display RAM to store data. The RFID Xpress application will use the graphic mode of operation for displaying startup images. In this mode, each bit in RAM corresponds to a pixel on the LCD which will either illuminate or not. It will also utilize the text mode of operation for displaying product information in the shopping cart. In this mode, each byte of RAM represents a character to be displayed, with special registers that set the character width and line spacing. A Crystalfontz CFACCFL1 inverter module [2] is used to control power to the LCD backlight and requires 5 V and 450 mA. The LCD logic circuitry also requires a current source of 15 mA and an operating voltage of 5 V [1].

As valid customer key fobs or products are detected by the RFID reader and corresponding information is located within the external databases, the LCD is updated by placing the appropriate data bits onto the aforementioned I/O ports and then strobing the LCD controller in accordance with the specified timing characteristics. This will update the LCD RAM, refreshing the display and showing the customer his or her name, purchased products, running total, and other useful information.

Keypad (Storm 6000 Series PIN Entry Pad)

The 16-key keypad [6] is connected directly to Ports H (3 through 6) for the columns and J (6 through 7) and T (4 through 5) for the rows on the microcontroller. The interpreting of the key presses is performed in software on the microcontroller in order to minimize board size and eliminate the need for a keypad encoder. The software sets one column output to be high while the rows are pulled low by internal pull-down resistors when no buttons are pressed. The software loops through each of the columns, asserting one at a time, and if the customer pressed

a button in that column, the row also becomes asserted and the exact key pressed can be determined.

Printer (Star Micronics NP-211 Thermal Kiosk Receipt Printer)

The thermal printer is used to print 58 mm wide receipts for the customer, should he or she select the printed receipt option on the keypad by selecting the appropriate option after being prompted by the LCD. For minimum power consumption, the printer receives transmissions at 9600 baud via RS232 protocol [28] using a DB9 connector. Formatting of sent data is very flexible, making it an ideal solution for development. The printer maintains an idle state unless the customer chooses to print a receipt, at which time the data is simply transmitted through the SCI0 port.

A separate wall-wart rated at 9 V regulated output and 1.2 A is provided with the printer, which supplies enough power to function properly, as it satisfies the operating voltage requirement of 7 - 14 V regulated DC. It was advised that the printer be powered individually by its own regulated wall-wart, thereby requiring the aforementioned second power supply for the RFID Xpress system. The reasoning lies both in the fact that the printer was designed to run as the only load on the supply, and because the 1.2 A rating is on the low end of the total current design constraints. Another advantage to running the printer off of its own supply is that the current spikes that could occur during operation will not impact other components in the design.

When the customer selects the printed receipt option, the data log of the customer's purchases stored in SRAM is transmitted to the printer through the SCI0 port on the microcontroller. That data is stored into the printer's 5 kB receiving buffer before being printed.

9.0 PCB Layout Design Considerations

RFID Reader / Antenna

One of the most important design considerations in a project utilizing RFID technology is the placement of the RFID reader in relation to the other signals. Placing the antenna and transmitter too close to sensitive components could cause interference and result in unexpected circuit operation. This is not a concern in the design, due to the fact that the RFID reader and antenna are not built into the circuit, but rather exist in a separate housing connected by a three foot RS232 cable to a DB9 connector attached to the PCB [5].

Oscillator / PLL Circuits

An important analog component in the design is the crystal oscillator, which inherently must be placed close to the microcontroller. Not only must it also have short traces, but in order to provide the correct clock rate for the 100 Mbps Ethernet communication, the crystal must be connected in a Pierce configuration to the XTAL and EXTAL pins of the microcontroller [13]. It is also strongly recommended that, when using the Ethernet capabilities of the MC9S12NE64 microcontroller, the PLL circuits and filter capacitors must be configured in accordance with the data sheet diagrams [13]. The RFID Xpress circuit was designed to match these circuits exactly. Finally, in order to minimize any problems caused by noise, no components should be placed beneath the oscillator circuit, and hence the area is left completely empty.

Ethernet Circuit

The Ethernet circuitry also has a few more considerations to be taken into account. The microcontroller can be configured by software to provide LED signals that detail the status of the Ethernet communication. This design chooses to use two of those signals, link and activity, to drive LEDs mounted on the RJ45 jack. Another requirement is that the magnetics module needed for the Ethernet communication be placed as close as possible to the RJ45 jack; specifically, it must be less than one inch away from it. This is inherently achieved since the Molex HyperJack chosen for the design has the magnetics built into the jack [29]. This particular RJ45 module also satisfies the requirement that the magnetics module use a 1:1 turn ratio for both the receive and transmit lines. In order to ensure stability and reliability of the Ethernet communication, 12 mil traces and spacing are used in the design [13][30].

Power Supply

Another point of concern in the design was the power supply design. Due to the power dissipation and mixed voltage levels, the power supply is set away from the rest of the circuitry. Each part will generate a lot of heat, and hence will need to be given ample room to dissipate. There are copper pours beneath the heat-dissipation tabs of the two TI REG1117 voltage regulators in the SOT-223 surface mount package. These pours were made as large as possible, while maintaining the small overall size of the board. They will act as simple heat sinks to help draw out the heat generated in the voltage regulation [14].

The TI MC34063A Switching Regulator also requires some considerations to be taken into account. First, there are two relatively large inductors involved with the circuit, which must be placed apart from one another so that their mutual inductance doesn't affect the circuit. They are also part of the motivation to place the entire power circuit away from everything else because of their magnetic characteristics, which might otherwise cause interference [15].

The final power consideration was given to the LCD inverter which uses a large step up converter to provide the high voltage required to power the LCD backlight. This could cause interference if placed close to any sensitive circuits, and hence is located at the top of the PCB, away from any sensitive circuits [2].

Microcontroller

Many considerations also need to be made for the microcontroller part of the design. First, the MC9S12NE64 80-pin package has a small flag exposed on the underside of the chip. This is designed for heat dissipation, but because this application is not processing intensive, and heat will likely not be a problem, a copper pour was not placed beneath the chip to attach to that flag. This allowed for many vias to be placed beneath the chip, which helped to decrease the overall size of the PCB layout [13].

The microcontroller also has many power pins which need to be bypassed with capacitors so as to provide an instantaneous current source during transitions. It is very important that these capacitors be close to their associated pins, so most of them are placed on the bottom side of the board. The data sheet also mentioned the importance of providing the central power and ground to the VDDA and VSSA pins at the "top" of the chip. Hence, those pins are connected to the

main 3.3 V and ground rails. Finally, the area beneath the capacitors associated with the PLL and oscillator circuits was recommended to be left open and free of traces [13].

General Considerations

There were also many considerations taken into account in general when routing the PCB. Overall size was not a major design constraint, but compactness was a definite goal, both for functionality and cost practicality reasons. Hence, all design blocks are arranged around the central processor in order to save space. In order to make the use of the design more flexible, all of the external interfaces were also placed on one side of the board so that the board can be mounted on any face of the packaging.

Signal traces were routed with 12 mil copper to decrease line inductance and improve signal reliability, and 45 degree angles were used at all times. Main power and ground traces were routed with 60 mil copper to help drive their inherently longer nets. Due to the large number of surface mount parts used, a larger number of vias was likely, but with careful reworking, the number was minimized to be only 61. The number of vias is also offset by being able to place many components on the bottom side of the board to conserve space. Also whenever possible, power and ground traces were routed next to each other in order to decrease line noise [31].

In order to account for future modifications to the circuit design, a header was placed on the unused G port. This allows for 7 spare I/O pins that can be programmed later in development. Although the IRQ and XIRQ pins are not currently implemented, they are also routed to headers so as to make them available for future implementation or design changes.

After careful design and multiple iterations, the PCB was finalized to maximize precision and reliability, and to minimize size and hence cost. This was achieved through careful component placement and optimized trace width and routing. The final board was only 16.0 square inches, and had only 61 vias.

10.0 Software Design Considerations

Memory Mapping

$0000-$03FF	Register Space
$03FF-$1FFF	RAM (7k) (inaccessible)
$2000-$3FFF	RAM (8k)
>$2000-$2FFF	Variables (heap)*
>$3000-$3FFF	Stack (SP starts at $4000)*
$4000-$FF00	Flash EEPROM (<48k)
>$4000-$8E84	19.6kB
$FF00-$FFFF	Interrupt Vectors
	* Approximations

All of the software including the bitmap binary code for graphics sum up to about 19.6 kB and are all stored in the EEPROM. There are three stored bitmaps, one of which requires 1kB of memory (the logo). Since the microcontroller is programmed in embedded C, the compilation is done by the compiler. Therefore, the ability to estimate usage of accessible RAM space is limited.

External Interface Mapping

	Port:	Address:
Keypad:	Port H (3:6)	$0258 (3:6)
	Port J (6:7)	$0260 (6:7)
	Port T (4:5)	$0240 (4:5)
LCD:	Port G (0:6)	$0250 (0:6)
	Port H (0:2)	$0258 (0:2)
	Port J (0)	$0260 (0)
Printer:	SCI0 Tx	$00CF (SCI0_DRL)
RFID reader:	SCI1 Rx	$00D7 (SCI1_DRL)
Ethernet:	EPHY (RxN, RxP, TxN, TxP)	Multiple Ports*

*Ethernet communication is provided by included libraries which perform many functions, such as maintaining a FIFO queue of data to be sent, and also placing the data on the appropriate pins at the appropriate time.

Due to I/O limitations on the microcontroller, the keypad and LCD needed special attention for pin mapping. Since none of the general I/O ports on this scaled down version of the microcontroller have 8 externally available pins, the LCD's 8-pin data bus and the keypad's pins (4 in, 4 out, 8 total) could not fit into one port. The only pins used for input were the keypad's row signals, and since PAD was input only, we considered migrating these signals to the PAD port, saving 4 general I/O pins for the LCD. However, the PAD port didn't offer internal pull-down resistors, so the row signals needed to be placed elsewhere. It was concluded that the most optimal organization of pin mapping resulted from the organization listed in the External Interface Mapping above.

Utilization of Integrated Peripherals

The built-in Ethernet module enabled communication with a remote database that contained customer and item information. Communication with the thermal printer [28] and the RFID reader [5] utilized the Serial Communications Interface module (SCI0 and SCI1) to conserve open pins for other devices. The Timer Channel module was used to maintain system time and strobe the columns of the keypad.

Organization of Application Code

The main code is organized as an Interrupt-Flag-Driven, Polling State Machine (I.F.D.P.S.M). The state machine checks for certain conditions to determine state transitions. In each state, certain flags are cleared or asserted. For example, the check keypad flag is set if the keypad is needed in the current state. The main loop calls a function that polls the keypad, and if a key press is expected and one is found, another flag is set that notifies the main loop that a key has been entered. Polling is the best way to communicate with the keypad because the resolution of a key press is quite coarse and they don't require interrupts to be handled quickly. The main loop is a large `IF` structure, with each `IF` statement checking for assertions of various flags. If a relevant flag is set, then the condition it denotes will be handled. An interrupt service routine will be called when an SCI interrupt alerts the processor that RFID serial number data is being written to the SCI1 port. The interrupt captures the data and sets a flag in order to allow the loop to finish its execution in a timely manner. In this way, an RFID event will not consume large amounts of time and delay the main loop.

Provisions Made for Debugging

The BDM (Background Debug Module) header allowed the chip to be flashed with code, and real-time access to the various registers and other parameters within the processor during run-time. Headers were also left available for Port G, the ATD module, and the IRQ and XIRQ pins should they become necessary and/or useful.

Flowchart

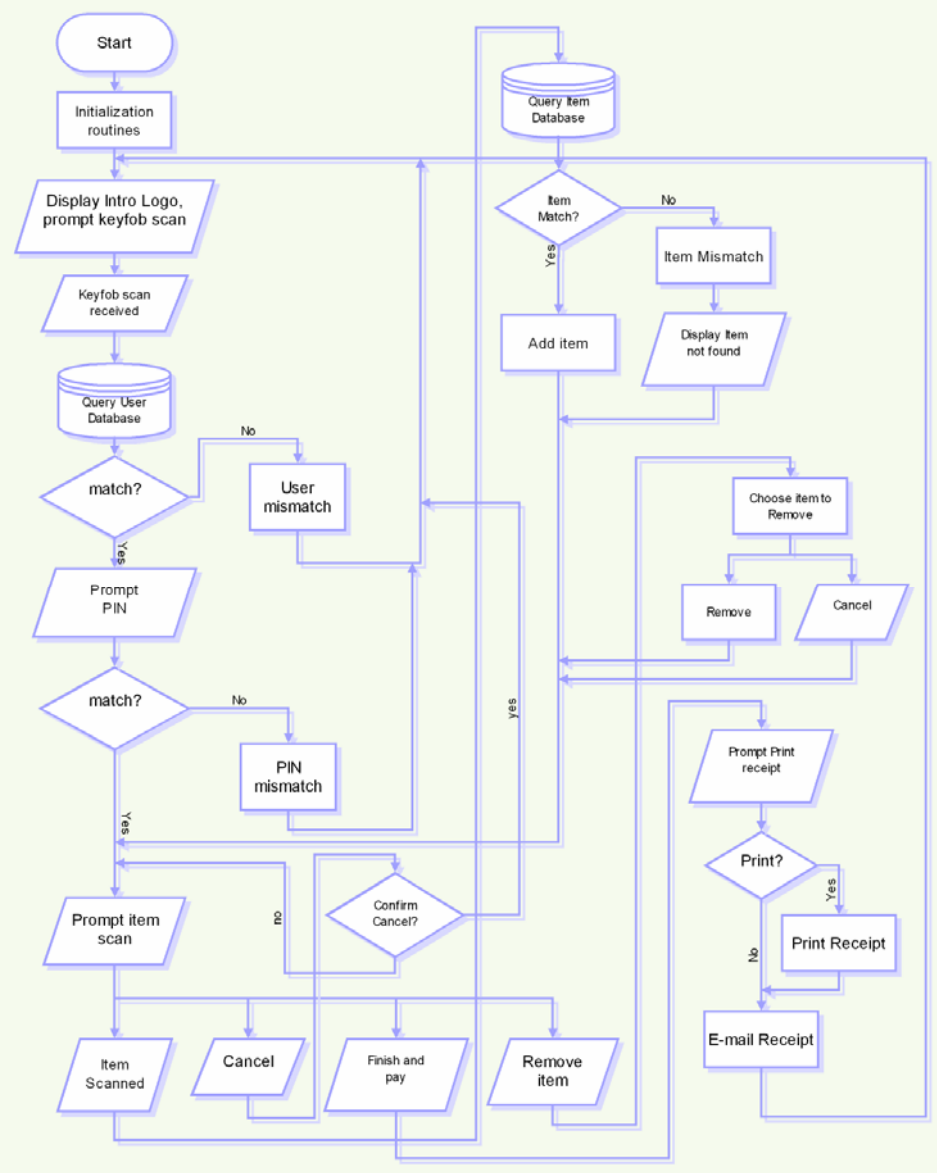


Figure 10-1. Software Flowchart

Software Hierarchical Diagram

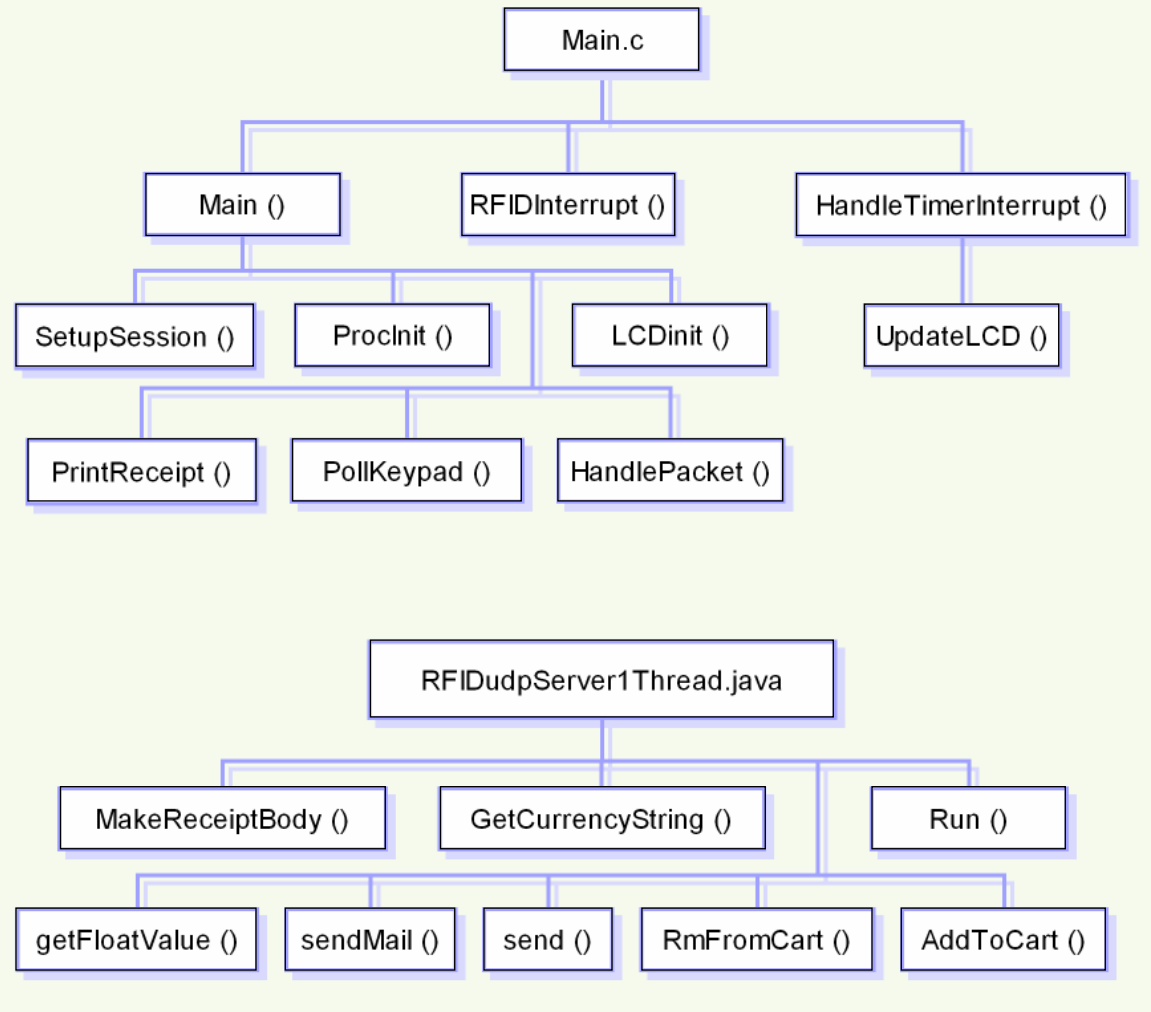


Figure 10-2. Software Hierarchical Diagram (Part 1)

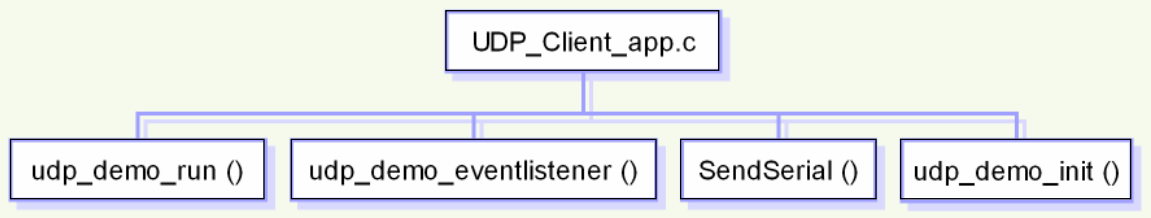


Figure 10-3. Software Hierarchical Diagram (Part 2)

Software Design Narrative

In the main function, the processor initialization and Ethernet initialization functions are called before the main polling loop begins. They initialize all necessary register values for use in the design. These include Data Direction registers, Pull Device registers, EPHY registers, and SCI configuration registers. Most of the initialization values were either created by the IDE (Metrowerks CodeWarrior for HC12) [32] or taken from demo code provided with the processor evaluation kit [33].

The LCD character and graphical mode initialization functions are called at various times throughout normal operation, as both character and graphical modes are used regularly. They initialize all of the registers necessary for the LCD [1]. The LCD is configured by writing to it with the RS signal asserted to select a configuration register in the LCD's RAM, and then subsequently writing to the LCD with the RS signal negated to write a specific value to those registers. Data is transmitted to the LCD using various control signals, and an 8-bit data bus. The data bus needed to be split between three ports, G, H, and J [13]. The control signals will be provided by the remaining pins on Port G, since only three pins are needed (LCDreset, RS, and Enable).

Though sending e-mail is possible via the integrated Ethernet controller, most companies that purchase the RFID Xpress system would require custom implementations of such software. It is not very cost-effective to flash custom versions of embedded code into the system, so it was decided that a proprietary communication protocol with the Java server would be a better universal solution. The microcontroller just tells the Java server to send an e-mail, and the custom Java code, which is much easier to change, does the actual e-mailing.

The RFID Interrupt method is called when a serial-data-ready interrupt is triggered on SCI1. It retrieves the data, in the form of a serial number, sent by the RFID reader [4] and sets a flag to alert the main loop that this has occurred. The main loop then takes appropriate action depending on the state of the system. If an RFID scan is irrelevant to the current state, the data is still read in order to clear the interrupt, however the main loop disregards the asserted flag and negates it before the next state transition.

The Print Receipt module is called when a printed receipt is desired. It sends data to the thermal printer over the SCI0 port. No hand-shaking is required, and in almost all cases, simply writing the ASCII value of a character to the port prints that character to the receipt. The printer

is configured manually using a proprietary button pressing sequence on the printer itself, as stated in the manufacturer's data sheet [28].

The Handle Packet module handles the responses sent from the Java server. The main code sends the RFID serial number to the Java server, which processes the data received from the database for user and item identification and verification. This module parses the response packets and stores it in appropriate data structures in SRAM.

The LCD methods are conveniently broken down into small functional blocks. The LCD graphical initialization module initializes required registers for graphical mode. It sets the display address and cursor address and clears the whole RAM to prepare for graphics information. It also sets the number of bits to be displayed on each 8-pixel block.

The LCD character initialization module initializes required registers for character mode. It sets the display address and cursor address and clears the whole RAM to prepare for text display, as well as character pitch, character spacing, and number of characters in a row.

The LCD write data module simply writes one byte into the LCD RAM. Input can be in integer or hexadecimal format for a better representation of the graphical perspective of the data.

The LCD print string function converts a string and sends it byte-by-byte to the LCD by calling the aforementioned LCD write data module. Since on any data write the LCD automatically increments the cursor address by one, there is no need to reset the cursor address every time a byte of data is written.

11.0 Version 2 Changes

If an opportunity arose to improve RFID Xpress further, one of the first changes would be to add administrator privileges and functionality to the design. Certain key fobs would be designated "administrator" access (with unique PINs for added security) where an employee could manually add an unrecognized item to the product database, approve age restricted item purchases, etc.

Another change would be to add cash and credit payment options to the design by using a bill acceptor and magnetic card reader. This would provide additional methods of payment beyond the online billing method that is recommended with the design. However, these changes would likely require more communication ports on the microcontroller. Since both SCI ports on the MC9S12NE64 are currently in use, these additions may require a change to a different microcontroller for the design.

The system could be improved by adding functionality to remove an item's serial number and information from the product database after purchase. Currently, the system compares each serial number scanned with the previously scanned serial number. This serves a dual purpose – to act as a "de-bouncer" for the rapid scanning of the RFID reader and to prevent a customer from scanning an item more than once. If items were removed entirely from the database after purchase, scanning the item again would result in an "item not recognized" message on the LCD regardless of when the item was scanned a second time. This would ensure that a customer would not be improperly charged and would maintain an accurate inventory for the store owner and employees.

Finally, the design could benefit from slight improvements in the packaging. While the final product looks finished and professional, the printer is still exposed on the rear of the enclosure. Easy access to the printer is necessary for the store employees to be able to change the paper roll, but it should be fully enclosed for safety reasons. The PCB also protrudes from the rear of the packaging slightly in order for the two DB9 connectors, RJ-45 jack, and power jack to be accessible. However, if the packaging enclosure had been selected and acquired earlier, the PCB could have been designed using the proper constraints so that it would fit perfectly inside with only the connectors exposed.

12.0 Summary and Conclusions

The RFID Xpress senior design project culminated in the successful and timely completion of all five Project Specific Success Criteria. The system was able to correctly identify a customer based on a key fob with an embedded RFID tag and authenticate that customer when a correct PIN was entered on the keypad. The RFID reader scanned products and transmitted the unique serial number on the tag to the microcontroller. The Ethernet module was used to query an external product and customer database to retrieve pertinent information. The scanned product names and items were displayed to the LCD, as well as the date, time, number of items in the cart, and cart total. At the end of the shopping session, the thermal printer generated a receipt upon request, and the Ethernet module was used to send an e-mail receipt to the customer. The system was packaged in a professional-looking enclosure that was nearly suitable to put directly into retail stores.

There were many opportunities for learning experiences during the development of RFID Xpress. PCB layout was an unfamiliar task at the start of the semester, but many hours in lab and several re-routings of the entire design resulted in a decent overall layout and much knowledge about routing considerations. Prototyping separate hardware modules of the design prior to PCB population proved to be a wise decision. As component functionality was verified, the software segments for those components were also completed. Designing the software for the system in a modular fashion resulted in very organized, easy to comprehend and debug software. The importance of placing debugging headers on the PCB was underestimated, as they would have been much more helpful in diagnosing communication problems. Instead, many unique forms of debugging were utilized, and while the problems were all rectified in the end, many hours of work were wasted. In conclusion, RFID Xpress was a challenging project which offered many opportunities to gain new experience and knowledge, and ultimately was a great success.

13.0 References

[1] Crystalfontz America, "CrystalFontz CFAG240128D Graphic LCD Specification," *Crystalfontz America*, 2006. [Online]. Available: http://www.crystalfontz.com/products/240128d/CFAG240128DFMIT.pdf. [Accessed: Feb. 2006].

[2] Crystalfontz America, "CFAICCFL1 Datasheet," *Crystalfontz America*, 2006. [Online]. Available: http://www.crystalfontz.com/products/240128d/CFAICCFL1.pdf. [Accessed: Feb. 2006].

[3] Maxim Integrated Products, "MAX3233E/3235E Datasheet," *Maxim IC*, Aug. 2004. [Online]. Available: http://pdfserv.maxim-ic.com/en/ds/MAX3233E-MAX3235E.pdf. [Accessed: Feb. 2006].

[4] Freescale Semiconductor, "MC9S12NE64 Microcontroller Fact Sheet," *Freescale Semiconductor*, Rev. 0, 2004. [Online]. Available: http://www.freescale.com/files/microcontrollers/doc/fact_sheet/MC9S12NE64FS.pdf. [Accessed: Feb. 2006].

[5] Intersoft Corp., "WM-RO-MR2 Datasheet," *Intersoft Corp.*, Nov. 2000. [Online]. Available: http://www.intersoft-us.com/dnload/wmromr2.pdf. [Accessed: Feb. 2006].

[6] Storm Interface, "Storm 6000 Series Keypad," *Storm Interface*, Oct. 2004. [Online]. Available: http://www.storm-interface.com/dyncat.cfm?catid=19038. [Accessed: Feb. 2006].

[7] J.C. Goodwin, III, "Self-service checkout system with RFID capability," U.S. Patent 6,547,040, April 2, 2001. [Online]. Available: http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=/netahtml/srchnum.htm&r=1&f=G&l=50&s1=6,547,040.WKU.&OS=PN/6,547,040&RS=PN/6,547,040. [Accessed: Mar. 2006].

[8] J.S. Walter, "Self-service checkout apparatus," U.S. Patent 5,992,570, September 9, 1997. [Online]. Available: http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=/netahtml/srchnum.htm&r=1&f=G&l=50&s1=5,992,570.WKU.&OS=PN/5,992,570&RS=PN/5,992,570. [Accessed: Mar. 2006].

[9] NCR, "Worlds First 'Hybrid' Self-Checkout Installed in METRO Group's RFID Innovation Center," *NCR*, Aug. 2004. [Online]. Available: http://www.ncr.com/media_information/2004/aug/pr080904.htm. [Accessed: Mar. 2006].

[10] NCR, "FastLane Overview," *NCR EB-1621-1104*, 2004. [Online]. Available: http://www.ncr.com/en/products/pdf/hardware/sa_FastLane.pdf. [Accessed: Mar. 2006].

[11] D. G. Meyer, "Module 10: Patent Infringement Liability," *Purdue University: Electrical and Computer Engineering*, ECE477 Class Notes, Spring 2006. [Online]. Available: http://shay.ecn.purdue.edu/~dsml/ece477/Notes/PDF/4-Mod10.pdf. [Accessed: Mar. 2006].

[12] Department of Defense, "MIL-HDBK-217F - Military Handbook of Reliability and Prediction of Electronic Equipment", *Department of Defense*, December 2, 1991. [Online]. Available: http://shay.ecn.purdue.edu/~dsml/ece477/Homework/Spr2006/Mil-Hdbk-217F.pdf. [Accessed: Apr. 2006].

[13] Freescale Semiconductor, "MC9S12NE64 Microcontroller datasheet," *Freescale Semiconductor*, Rev. 1, Sep. 2004. [Online]. Available: http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC9S12NE64V1.pdf. [Accessed: Feb. 2006].

[14] Texas Instruments, "REG1117A 1A LDO Voltage Regulator datasheet," *Texas Instruments*, July 2004. [Online]. Available: http://focus.ti.com/lit/ds/symlink/reg1117.pdf. [Accessed: Feb. 2006].

[15] Texas Instruments, "MC33063A 1.5 A Peak Boost/Buck/Inverting Switching Regulator datasheet," *Texas Instruments*, Oct. 2005. [Online]. Available: http://focus.ti.com/lit/ds/symlink/mc33063a.pdf. [Accessed: Feb. 2006].

[16] Epson Electronics America, "SMD High Frequency 25 MHz Crystal Unit datasheet," *Epson Electronics America*, [Online]. Available: http://www.eea.epson.com/go/Prod_Admin/Categories/EEA/QD/Crystals/mhzSMD_Crystals/go/Resources/TestC2/MA406. [Accessed: Mar. 2006].

[17] George Novacek, "Designing for Reliability, Maintainability, and Safety," *Circuit Cellar*, Dec. 2000. [Online]. Available: http://shay.ecn.purdue.edu/~dsml/ece477/Notes/PDF/4-Mod13_ref.pdf. [Accessed: Apr. 2006].

[18] Mark Roberti, "Big Brother's Enemy," *RFID Journal*, July 21, 2003, [Online]. Available: http://www.rfidjournal.com/article/articleview/509/1/1/ . . [Accessed: Apr. 2006].

[19] Mark Baard, "RFID: Sign of the (End) Times?," *Wired News*, Mar 02, 2006, [Online]. Available: http://www.wired.com/news/technology/0,70308-0.html. [Accessed: Apr. 2006].

[20] Virginia Waste Minimization Program, "Fact Sheet: Printed Circuit Board Manufacturers," *Virginia Waste Minimization Program,* Vol. 1 Issue 9, Oct 24, 1995, [Online]. Available: http://es.epa.gov/techinfo/facts/vdwm/va-fs6.html. [Accessed: Apr. 2006].

[21] FCC: Office of Engineering and Technology, "Radio Frequency Safety FAQs," *FCC: Office of Engineering and Technology*, Jan 09, 2006, [Online]. Available: http://www.fcc.gov/oet/rfsafety/rf-faqs.html. [Accessed: Apr. 2006].

[22] Cisco Systems, "Wireless Systems and RF Safety Issues," *Cisco Systems*, 2005, [Online]. Available: http://www.cisco.com/en/US/products/hw/wireless/ps4570/products_white_paper09186a0080088791.shtml. [Accessed: Apr. 2006].

[23] Joint Service Pollution Prevention Opportunity Handbook, "Printer Circuit Board Recycling," *Joint Service Pollution Prevention Opportunity Handbook*, May 2003, [Online]. Available: http://p2library.nfesc.navy.mil/P2_Opportunity_Handbook/2_II_8.html. [Accessed: Apr. 2006].

[24] Tutorial-Reports.com, "Impact of RFID Tags on Recycling," *Tutorial-Reports.com*, 2006, [Online]. Available: http://www.tutorial-reports.com/wireless/rfid/environment/recycling.php?PHPSESSID=958c9a20c3affc755e9e1a12f2550aa9. [Accessed: Apr. 2006].

[25] Association for Automatic Identification and Mobility, "eWaste: Environmental & Recycling Issues," *Association for Automatic Identification and Mobility*, Slide 26, Sep 28, 2005, [Online]. Available: http://www.autoid.org/metatraffic2/track.asp?mtr=/presentations/2005/eWaste_REG_Frontline_20050928.ppt. [Accessed: Apr. 2006].

[26] NCR, "FastLane-Mini advertisement brochure," *NCR EB-2061-0904*, 2004, [Online]. Available: http://www.ncr.com/en/products/pdf/hardware/sa_fastlane_mini.pdf. [Accessed: Mar. 2006].

[27] Fujitsu, "USCAN1 Self-Checkout information brochure," *Fujitsu USCAN1-DS-0804*, [Online]. Available: http://www.fujitsu.com/downloads/SOL/ftxs/datasheet/U-SCAN1.pdf. [Accessed: Mar. 2006].

[28] Star Micronics, "NP-211 Thermal Kiosk Receipt Printer," *Star Micronics*, July 30, 2004, [Online]. Available: http://www.starmicronics.com/printers/printers_pages/support/manuals/NP_manuals/NP211SM.pdf. [Accessed: Mar. 2006].

[29] Molex, "HyperJack Modular RJ45 Jack with Integrated Magnetics and LED," *Molex*, August 8, 2005, [Online]. Available: http://www.molex.com/pdm_docs/sd/480250002_sd.pdf. [Accessed: Feb. 2006].

[30] Freescale Semiconductor, "Implementing an Ethernet Interface with the MC9S12NE64," *Freescale Semiconductor Application Note AN2759*, Rev 0.2, Sep. 2004, [Online]. Available: http://www.freescale.com/files/microcontrollers/doc/app_note/AN2759.pdf. [Accessed: Feb. 2006].

[31] Freescale Semiconductor, "System Design and Layout Techniques for Noise Reduction in MCU-Based Systems," *Freescale Semiconductor Application Note AN1259*, 1995, [Online]. Available: http://www.freescale.com/files/microcontrollers/doc/app_note/AN1259.pdf. [Accessed: Feb. 2006].

[32] Freescale Semiconductor, "CodeWarrior Development Studio for Freescale HC9S12/XGATE Microcontrollers," *Freescale Semiconductor*, Rev A, 2005, [Online]. Available: http://www.freescale.com/files/soft_dev_tools/doc/data_sheet/950-00081.pdf. [Accessed: Feb. 2006].

[33] Freescale Semiconductor, "DEMO9S12NE64 evaluation kit," *Freescale Semiconductor*, Rev 0.8, Sep. 2005, [Online]. Available: http://www.freescale.com/files/microcontrollers/doc/user_guide/DEMO9S12NE64UM.pdf. [Accessed: Feb. 2006].

Appendix A: Individual Contributions

A.1 Contributions of Jennifer Tietz:

I was designated the team leader at the beginning of the project, and therefore I was generally responsible for scheduling team meetings, maintaining our design schedule, and ensuring that assignments were completed and edited by all members of the team. I also prepared most of the TCSPs throughout the semester and assisted other team members in their documentation assignments as needed.

I had a significant part in the component selection and hardware design of the system as I completed the Circuit Design and Theory of Operations paper. I assisted in parts and sample acquisition and was solely responsible for generating the original schematic of our design. Throughout the semester, I continually updated the schematic to maintain changes that were made to the design. I also created and updated block diagrams for our project as the system development progressed. I populated nearly all of the PCB with the exception of the microcontroller, power supply, and some minor components, and performed initial hardware testing on the RFID reader, printer, keypad, and power supply to ensure proper operation.

Prior to writing any software, I developed the flowchart and code module hierarchy for the overall design of the system. As the team leader, I delegated specific coding tasks to members of the team according to what former knowledge and experience they had, as well as what they were comfortable with learning. I wrote detailed pseudocode for our main program and organized the state machine flow of the code. I also included pseudocode for supporting functions to be written by other team members. Jared and I worked together to modify the pseudocode into our actual main program, which included a state machine execution to control the shopping session, simulated RFID interrupts, and keypad polling and interpretation. We also wrote the software for the RFID reader together while simultaneously debugging the hardware in lab, and later implemented SCI interrupts on an RFID data receive.

I completed the code to properly initialize the thermal receipt printer, as well as the function to read the shopping cart data from SRAM and generate a formatted printed receipt. I wrote the pseudocode for the function that updates the LCD based on the state variable in the main program, and made modifications to the code once Jonathan completed it to properly format the data on the screen. My experience with generating a printed receipt resulted in my

completing the Java server function of generating an e-mail receipt with the current date, time, user information, and shopping cart data. After the software was fully completed, I commented all of the code.

I was involved in nearly all of the hardware and software debugging with the exception of the Ethernet communication software. I aided in determining solutions to several problems we encountered throughout the design process, including the inability to trigger timer and SCI interrupts in the system, the improper functioning of the thermal printer when connected to the PCB, and repeated testing and debugging of the keypad.

Finally, I completed the Reliability and Safety Analysis assignment, which included determining the MTTF of the most fragile components and a FMECA of each module of the schematic in Appendix C. I also wrote the ECE Senior Design Report and most of the User Manual.

A.2 Contributions of Jared Suttles:

I was heavily involved in almost every aspect of this project. My main responsibilities were part research and acquisition, power supply design, PCB design, software design, considerable hardware assembly, and hardware and software debugging. Also, I was the main webmaster of our team webpage.

My first contribution to the project was many hours of parts research and acquisition. Before meetings, I would research many alternatives to consider for the major components in our design. Josh and I found many different choices that we then discussed at team meetings. Jennifer and I then purchased the items or requested samples for the team. I was also in charge of purchasing the discrete components from DigiKey.

The power supply was a significant challenge for me, since I am a Computer Engineer rather than an Electrical Engineer. I first had to choose a main input voltage, and then had to design circuits to convert that value to the necessary 3.3, 5.0, and 12.0 V values for our design. This also meant that I needed to have a significant understanding of the current consumption of our major components. The 3.3 and 5.0 V values were simply LDO Voltage Regulator circuits, but the 12.0 V circuit required a lot of research. I eventually found a TI switching regulator and used the Typical Application Circuit found in the component's datasheet to design the step-up regulator. When the circuits were first tested, they outputted clean and precise values.

I had no experience with Orcad Layout when I began this course. The first time I created the PCB, it was large (27.4 sq. in.) and had 108 vias. It also violated many of the PCB design considerations discussed in Section 9 of this report. Jonathan was a great help in routing much of the actual PCB in layout. In the second iteration, I took great care in placing the parts and began the difficult routing jobs. Jonathan finished routing the circuits and I spent the rest of our allotted time optimizing the traces and compacting the board. In the end, the board was a lot better, only 16.0 sq. in. and had only 61 vias. It also finally took all of our design considerations into account.

The software was a very segmented effort. The whole team worked together to develop a general algorithm. Jennifer and I worked on the main polling loop code, the RFID handling code, and the keypad handling code. The main code was drawn out multiple times in pseudocode before being coded. I wrote code to test the RFID handling in both a polling and interrupt mode before I settled on the interrupt method. The keypad code was also challenge,

since we chose not to use an encoder. After much testing, I finally found that strobing the columns in a timer interrupt and checking the rows in the main polling loop was the optimal solution. I wrote the timer interrupt code, shopping cart maintenance code, and I integrated all of the modules into the final project. I modified the data structures Jonathan had designed so that it worked with my code. I also worked with him on the LCD methods used in our design, helping him port his existing assembly over to C. I integrated Josh's UDP code into the main code, which involved some slight modifications. I also made slight modifications to the Java server code, adding a few new command handling functions.

No one on our team had much experience with PCB population or soldering. When we all tested out our skills on the test boards, it turned out that Jennifer and I were the best at soldering. That, combined with the fact that I had more free time to devote to the project, allowed me to take hardware assembly as one of my responsibilities. I did most of the power supply circuitry, and I soldered the microcontroller. I also occasionally helped Jennifer finish populating the rest of the board.

One of my biggest responsibilities was debugging the hardware and the software. I tested many circuits and code modules prior to integrating them into the final project. I was also the unfortunate one to discover our two major mistakes. The first was the RS232 circuit, in which I mistakenly routed a pin to low instead of high. This was the lone flywire we had to place on our PCB. The second mistake was in the routing of the Rx and Tx lines in the Ethernet circuit, though we solved that problem with the use of a proprietary CAT5 Ethernet cable in which we switched the paths of the twisted pairs. Software debugging was also obviously a huge part of the project, and I made most of the modifications to the code to fix the problems we found.

The webpage was my last responsibility. I wrote most of the HTML for the site and also developed a javascript to display our code in a viewer friendly fashion on the webpage. I also chose our photo album generation software and wrote a python script to format the auto-generated HTML files to match our webpage. This was a fun and perhaps the most superfluous part of this project, but I was glad I put in the effort I did when our webpage looked so good in the end.

I also was responsible for the PCB Considerations and Ethical and Environmental Concerns papers.

A.3 Contributions of Joshua D Chapman:

I was responsible primarily for the project packaging and for the design and development of the communications code. I also helped in the debugging of the main code and in the research of a suitable RFID system. I also was responsible for the Packaging Specifications and Design paper and the Patent Liability Analysis paper.

I was in charge of the packaging for this project. I wrote the packaging homework and acquired a case from OKW enclosures. The case required quite a few modifications to be right for the project. I drilled holes in the faceplate for the keypad pins and screws. I made a CAD drawing of the faceplate and sent it to the machine shop to get a hole drilled for the LCD screen. I also cut out holes in the back of the case to accommodate the printer as well as the RJ-45, DB9 and AC adapter connectors. I also had to modify the insides of the case in certain areas to allow a comfortable amount of room for the components as well as a place for the receipt to print out.

The majority of my time on this project was devoted to the Ethernet code so that the project could use an external database to look up items and customer info. The other reason for the Ethernet was so that the customers could receive an e-mail receipt of the purchase. The OpenTCP stack was used to implement the Ethernet functionality on the microcontroller. Most of the initial time was spent trying to figure out exactly how the different functions worked in the example applications and how they related to each other. At first I tried implementing all of the communications using TCP/IP. There were several issues with this approach. It seemed as if the ECN network did not like to communicate with the project even when the IP address and MAC address of the desktop PC were programmed onto it. Due to this and a few other difficulties, I decided that it would be best to use a UDP protocol. This is something that had already been demonstrated with one of the example applications. It would need heavy modifications to work with our needs, but at least it was already known that UDP packets could be sent and received. I got code working that could send a UDP packet from the project to the external server. This packet consisted of a command and a serial number.

I also used Eclipse, a java development kit, to make a UDP server. This server would accept a packet and look up the serial number sent in that packet. If the command was for user data it would return the name, e-mail and PIN of the customer. It would return a packet with item info if it was an item command. I also implemented the e-mail functionality on this side of

the server. If the command was to send the e-mail, the server would look up the e-mail address and send out an e-mail through its internet connection.

I also helped in several other areas including debugging of software and hardware. The majority of my time was spent in the areas discussed in this section but a lot of time was devoted to helping with debugging, papers and other areas in support of the team.

A.4 Contributions of Jonathan J Chen:

I had two major responsibilities for the project. Firstly, I had the responsibility to create and test the LCD code. Secondly I was responsible for much of the PCB routing. I was also responsible for the Design Constraint Analysis and the Software Design Narrative.

The LCD code was a very important part of the project, as the display is the most vital means of communication between the system and the customer. Since I had previous experience with the LCD, most of the time spent on the LCD software was used for porting over assembly code to embedded C. After successful interfacing, I wrote a collection of subroutines to easily print pixels and/or characters to the screen. This way my teammates had a much easier time debugging the integrated code since the usage of subroutines was very intuitive. In addition, I also created the different state specific screens for each state machine transition that requires a different display.

I had the opportunity to research and select a thermal receipt printer that perfectly suited our needs.

I was also responsible for the Design Constraint Analysis. I had to familiarize myself with areas of the design which I had little knowledge about, for example, the power circuit and the integrated Ethernet module.

On the other hand, I had to write the Software Design Narrative, and therefore I needed a very thorough understanding of the software implementation as a whole, including the keypad interface, serial communication with the RFID reader and the thermal printer, as well as the overall flow of the main program (interrupt-flag-driven polling state machine).

I also received plenty of practice in routing the PCB layout, in both the preliminary layout and the finalized version. Since none of us had any experiences with routing, I invested a lot of time in learning the tricks and shortcuts of the layout software. In the finalized version I aided Jared in routing the board to lower the area of the PCB by approximately 41.6%.

In addition, I also helped with debugging the server code and building miscellaneous components (buses, shrink wrap crimped wires, etc.).

Appendix B: Packaging

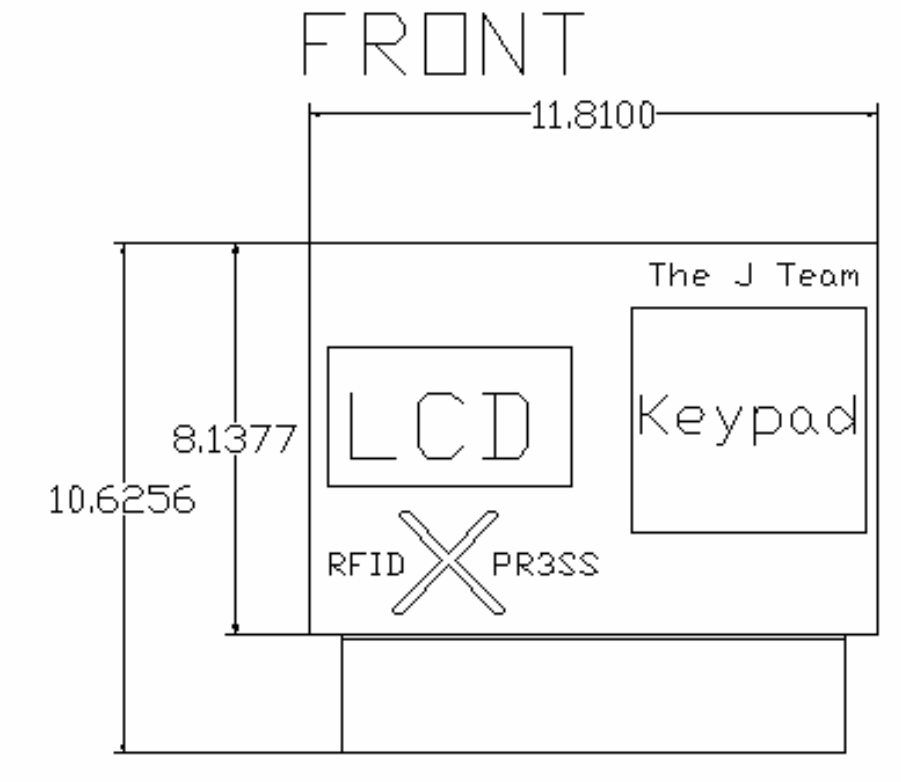


Figure B-1. Front View of Main Unit

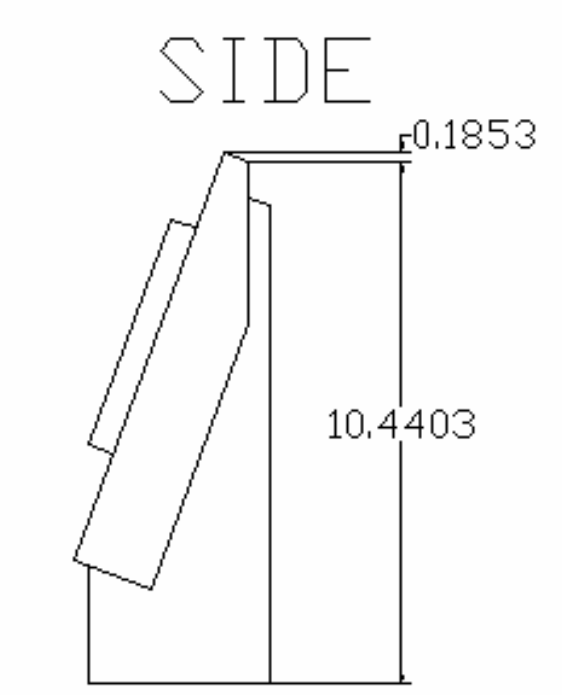


Figure B-2. Side View of Main Unit

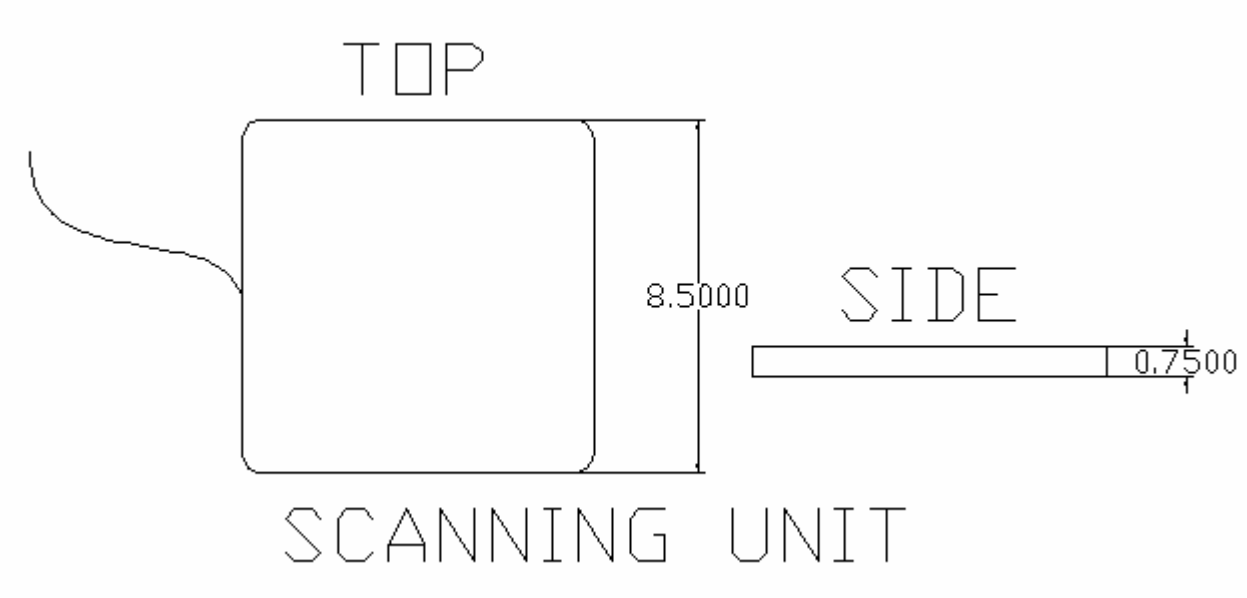


Figure B-3. Top and Side View of Scanning Unit

Appendix C: Schematic

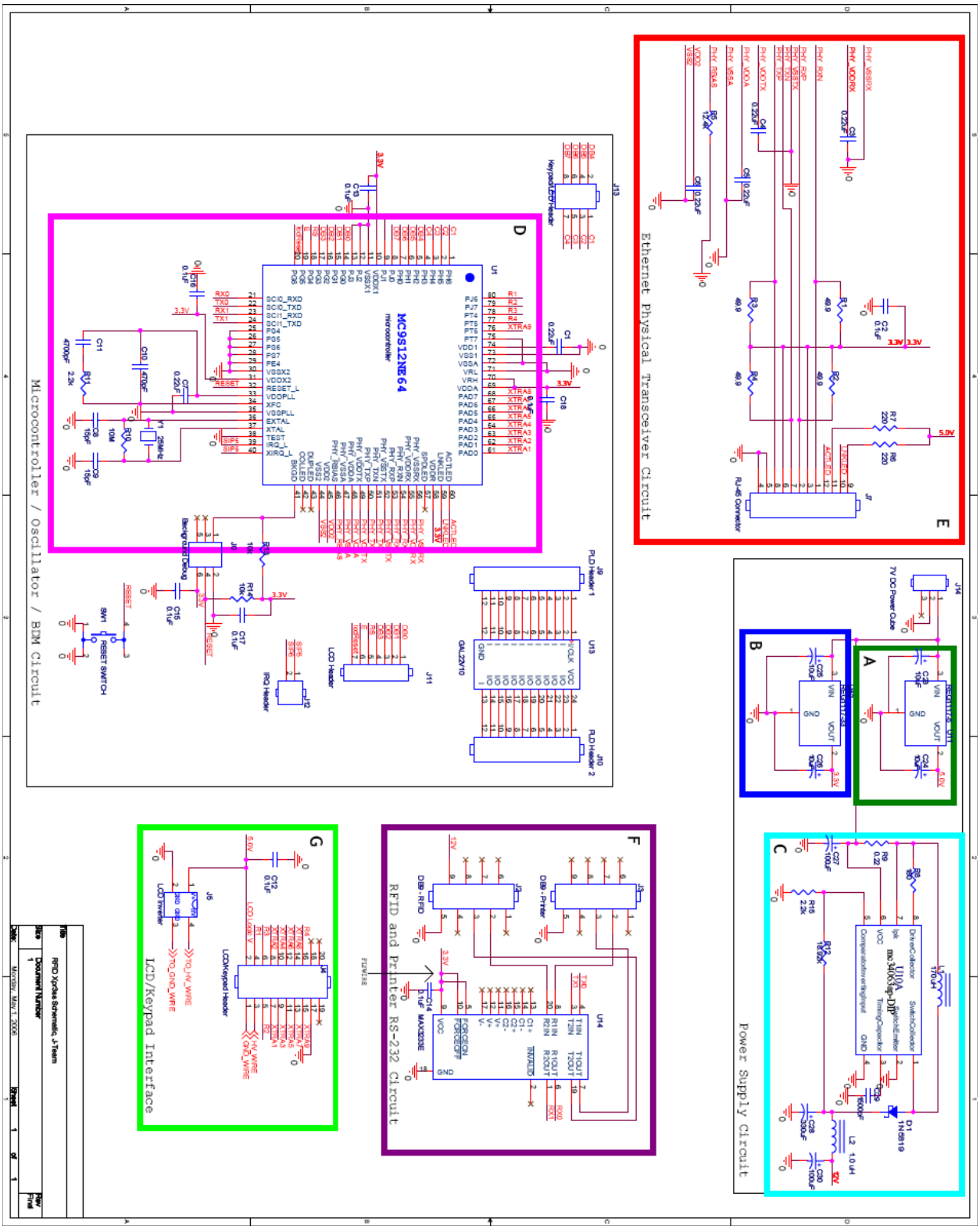


Figure C-1. Final Schematic (Blocked for FMECA Worksheet)

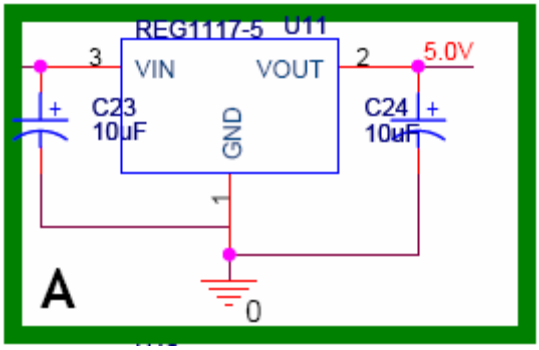


Figure C-2. 5V Voltage Regulator

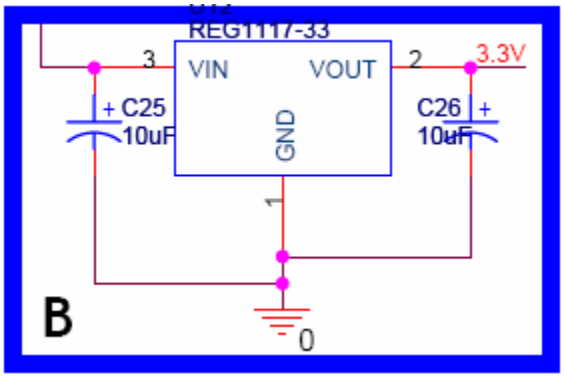


Figure C-3. 3.3V Voltage Regulator

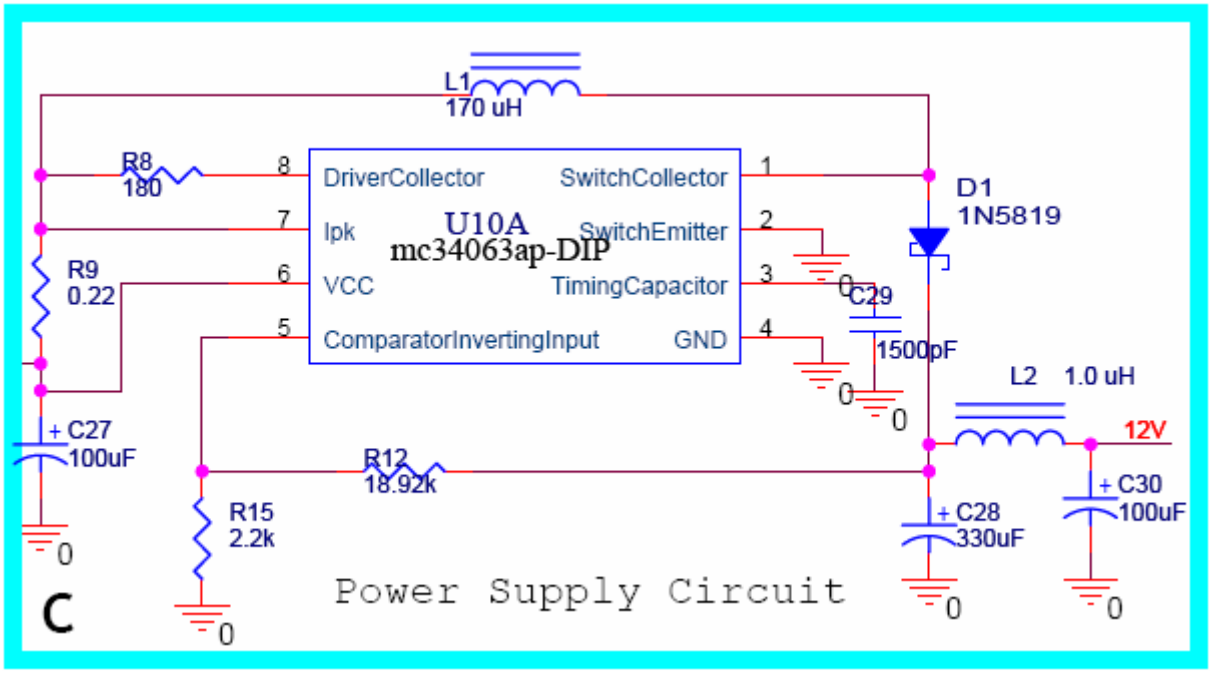


Figure C-4. 12V Switching Voltage Regulator

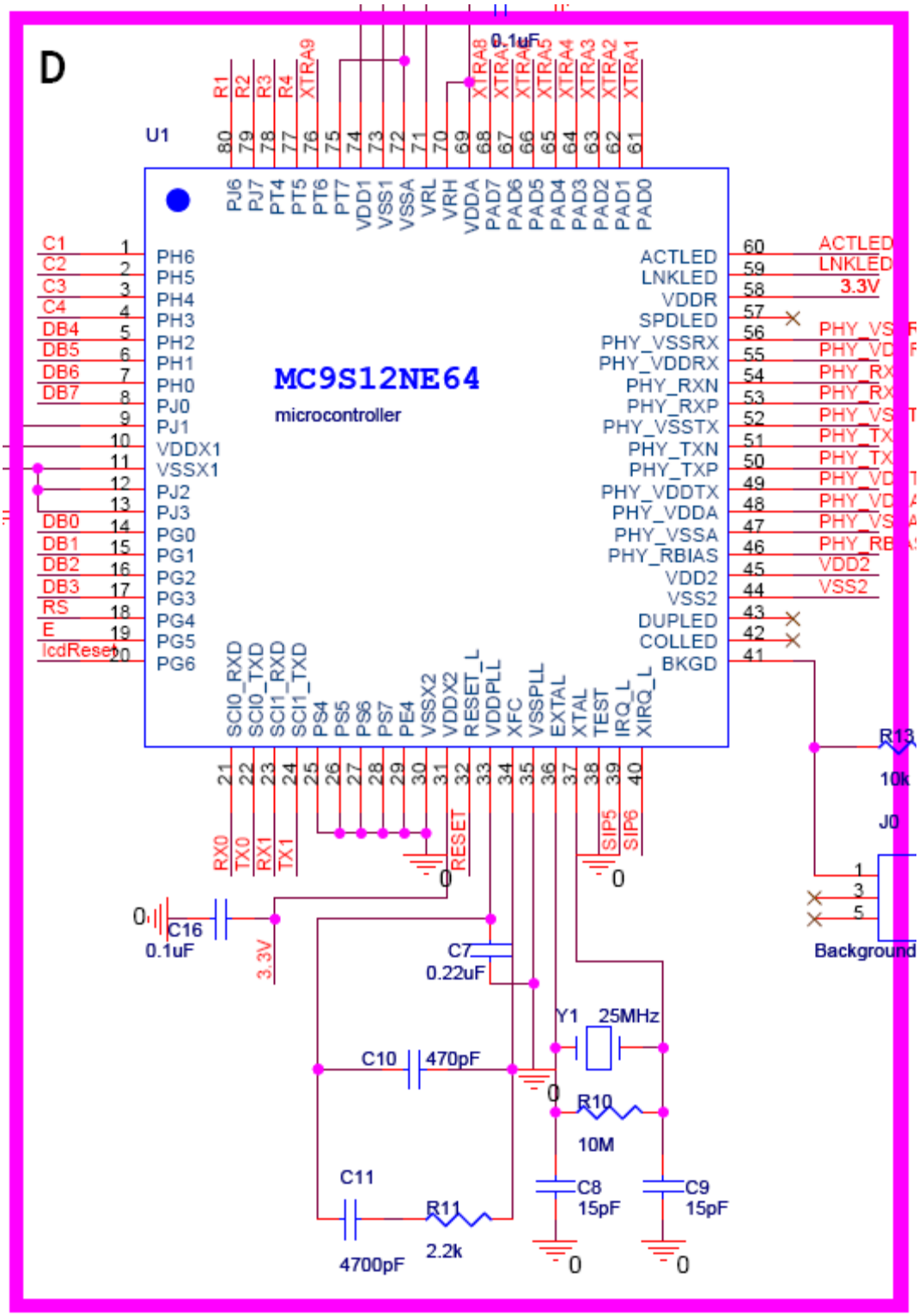


Figure C-5. Microcontroller Circuit

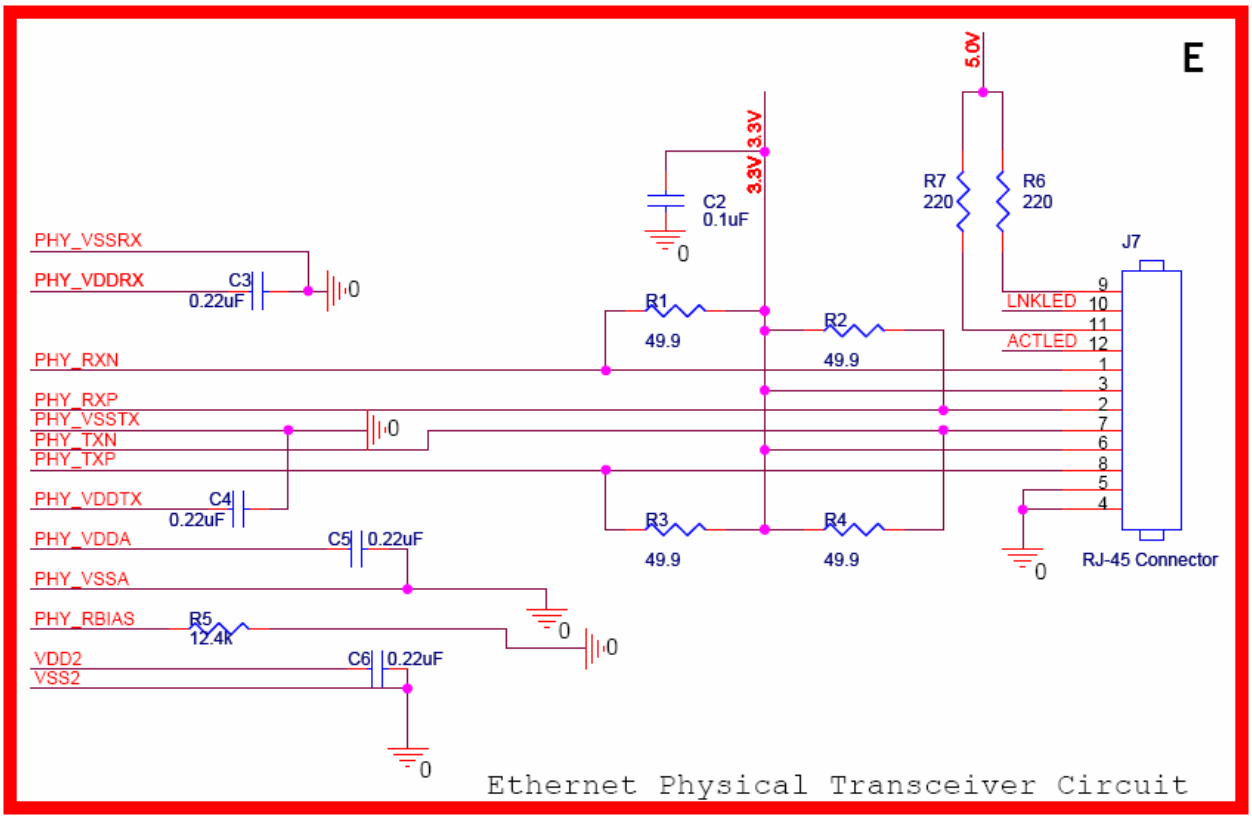


Figure C-6. Ethernet Physical Transceiver Circuit

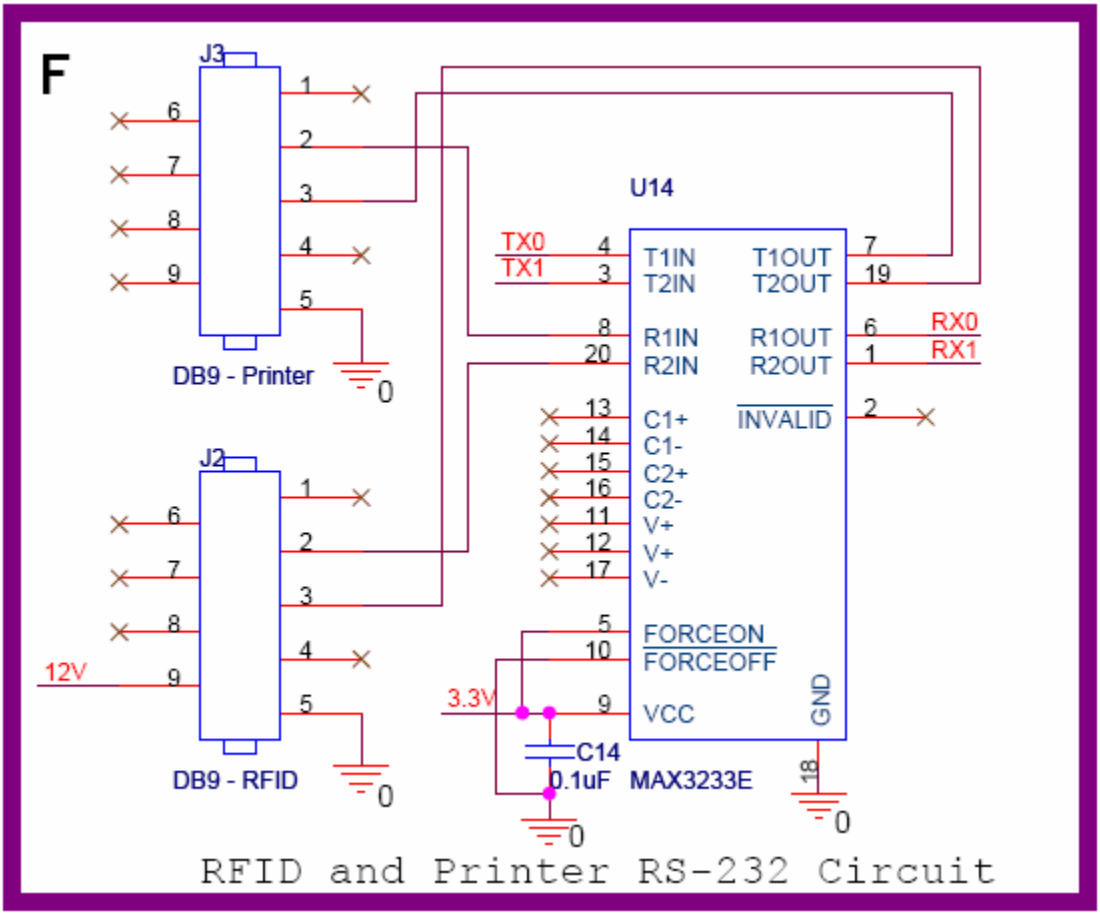


Figure C-7. RS-232 Circuit

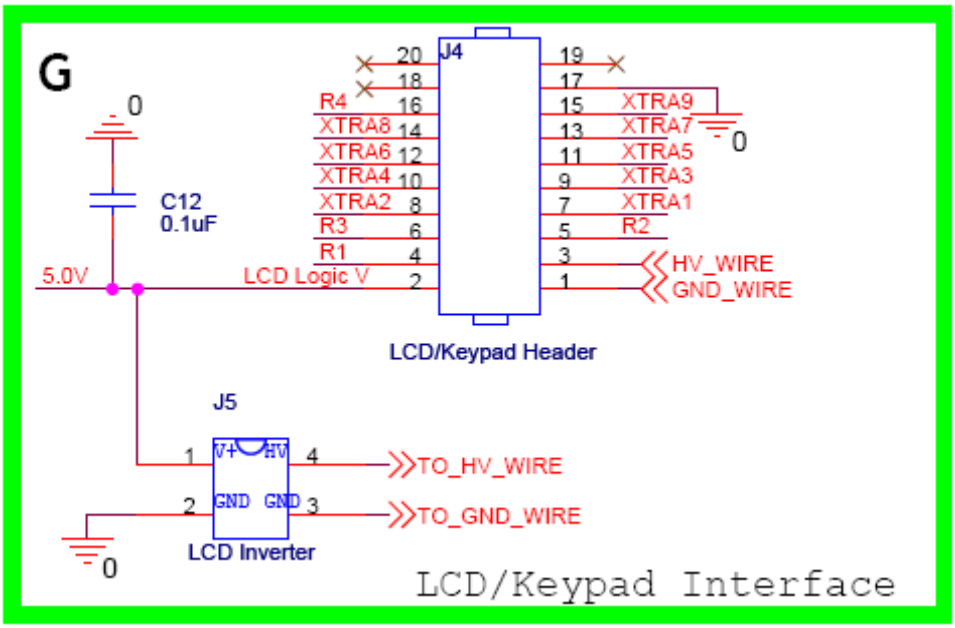


Figure C-8. 5V Voltage Regulator

Appendix D: PCB Layout Top and Bottom Copper

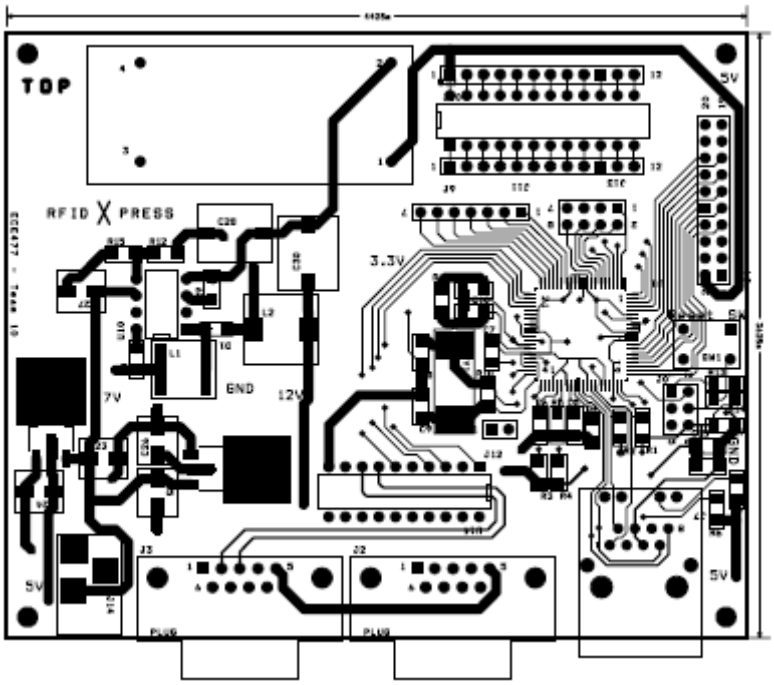


Figure D-1. PCB Top Copper

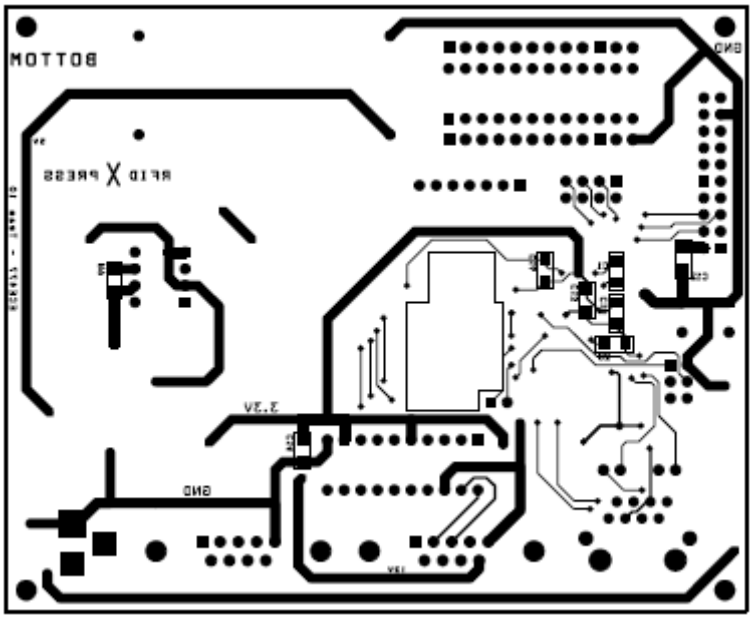


Figure D-2. PCB Bottom Copper

Appendix E: Parts List Spreadsheet

Vendor	*Manufacturer*	*Part No.*	*Description*	*Unit Cost*	**Qty**	*Total Cost*
Freescale	Freescale	MC9S12NE64	16-bit microcontroller (80 pin)	7.92	1	7.92
Allied Electronics	Storm Interface	6000-210023	16 Key PIN Entry Keypad	112.71	1	112.71
CrystalFontz	CrystalFontz	CFAG240128D	240x128 Graphic LCD	112.63	1	112.63
CrystalFontz	CrystalFontz	CFAICCFL1	Inverter for backlit graphic modules	5.47	1	5.47
Intersoft Corp.	Intersoft Corp.	WM-RO-MR2	Medium Range RFID Reader	130.00	1	130.00
Maxim/Dallas Semiconductor	Maxim IC	MAX3233E	RS232 Level Translator	7.80	1	7.80
Current Components	Star Micronics	NP-211	Thermal kiosk printer with semi-automatic cutter	236.38	1	236.38
				TOTAL		**$612.91**

Appendix F: Software Listing

Embedded Microcontroller Code

```
/**********************************************************************
 *
 *       Copyright (C) 2006 J Team, Inc.
 *       All Rights Reserved
 *
 * File Name      : Main.c
 * Project Name   : RFIDXpress.mcp
 * Description    : This file contains main() and most functionality of the RFID
 *                  Xpress system.  The main() function begins by initializing
 *                  the Ethernet.  The infinite while loop consists of a state
 *                  machine that governs the flow of control.  It also polls
 *                  the Ethernet module to determine if a packet was received.
 *                  The RFIDInterrupt() function handles an SCI interrupt and
 *                  extracts the serial number data.
 *                  The HandleTimerInterrupt() function strobes the columns of
 *                  the keypad and updates the LCD clock.
 *                  The ProcInit() function initializes the peripherals and
 *                  clock variables.
 *                  The LCDInit() function clears the LCD RAM.
 *                  The HandlePacket() function interprets packets received
 *                  from the Java server.
 *                  The UpdateLCD() function uses the state determined in the
 *                  main() function to set the cursor and display new information
 *                  to the LCD.
 *                  The PollKeypad() function checks each row to see if a key
 *                  press occurred in that row, while the HandleTimerInterrupt()
 *                  function asserts one column at a time.
 *                  The PrintReceipt() function generates a formatted receipt with
 *                  the user, date, and shopping cart information.
 *
 * Authors: Jared Suttles
 *          Jennifer Tietz
 *          Jonathan Chen
 *          Joshua Chapman
 *
 * Version : 1.2
 * Date    : 05/01/06
 *
 **********************************************************************/

/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "Events.h"
#include "database.h"

/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
#include "LCD_methods.h"
#include "bitmaps.h"
#include "UDP_Client_app.h"
#include "ne64api.h"
#include "ne64config.h"
#include "ne64debug.h"
#include "ne64driver.h"
```

```
#include "mBuf.h"
#include "debug.h"
#include "datatypes.h"
#include "timers.h"
#include "system.h"
#include "ethernet.h"
#include "arp.h"
#include "icmp.h"
#include "ip.h"
#include "udp.h"
#include "address.h"

/*************************/
/*       Constants       */
/*************************/

/* States */
#define IDLE                    1
#define USER_QUERY_DB           2
#define INVALID_USER            3
#define CAPTURE_PIN             4
#define SESSION_START           5
#define SESSION_RUNNING         6
#define INVALID_KEYPRESS        7
#define ITEM_QUERY_DB           8
#define CONFIRM_REMOVAL         9
#define ITEM_NOT_MATCHED        10
#define CANCEL_SESSION          11
#define REMOVE_LAST_ITEM        12
#define SESSION_END             13
#define PRINT                   14
#define EMAIL                   15
#define FINISH_SESSION          16
#define TIME_SYNCH              17

/* Masks and Flags */
#define RFID_FLAG_MASK          0x01
#define KEYPAD_FLAG_MASK        0x02
#define CHK_KEYPAD_FLAG_MASK     0x04
#define VALID_QUERY_FLAG_MASK    0x08
#define CANCEL_ONCE_MASK      0x10
#define ONLY_ONCE_MASK          0x20
#define CHK_RFID_MASK         0x40
#define MASK_CANCEL_MASK      0x80
#define RFID_FLAG             0
#define KEYPAD_FLAG           1
#define CHK_KEYPAD_FLAG          2
#define VALID_QUERY_FLAG      3
#define CANCEL_ONCE           4
#define ONLY_ONCE             5
#define CHK_RFID_FLAG         6
#define MASK_CANCEL_FLAG      7

/* RFID serial number type */
#define USER 1
#define ITEM 0

/* Keypad buttons */
#define KEY_ONE     1
#define KEY_TWO     2
#define KEY_THREE   3
#define KEY_FOUR    4
#define KEY_FIVE    5
```

```
#define KEY_SIX     6
#define KEY_SEVEN   7
#define KEY_EIGHT   8
#define KEY_NINE    9
#define KEY_ZERO    0
#define KEY_STAR    11
#define KEY_POUND   12
#define KEY_ENTER   13
#define KEY_CLEAR   14
#define KEY_HELP    15
#define KEY_CANCEL  16

/* Time */
#define AM 1
#define PM 2


/*************************/
/*         Macros        */
/*************************/
#define SetBit(bit_ID, varID)        (varID |= (byte)(1<<bit_ID))
#define ClearBit(bit_ID, varID)      (varID &= ~(byte)(1<<bit_ID))
#define clrReg8Bits(RegName, ClrMask)(RegName &= ~(byte)(ClrMask))


/*************************/
/*    Data Structures    */
/*************************/
typedef struct{
    char itemSerial[14];      // Item name, fixed 25 chars
    char itemName[26];    // Item name, fixed 25 chars
    int  nameLength;      // Item name length
    char itemPrice[6];    // Item price, format "XXXXX", fixed 5 chars
} itemStruct;

typedef struct{
    char userName[21];    // Full user name with spaces
    int  nameLength;      // Full user name length
    char userEmail[31];       // Full e-mail address (append "/" if > 30)
    int  emailLength;     // E-mail address length
    int cartSizeI;        // Number of items in cart
    long cartTotalI;      // Cart total
    int timeHrI;          // Hours of day
    int timeMinI;         // Minutes of day
    int timeSecsI;        // Seconds of day
    int timeHalfI;        // AM/PM
    int dateMonthI;           // Current month
    int dateDayI;         // Current day
    int dateYearI;        // Current year
    itemStruct items[20];     // Array of purchased items
} sessionStruct;


/*************************/
/*        Globals        */
/*************************/
byte state;
byte returnState;
byte flags;
int keyPressed;
int  pinCounter;
char currentPin[5];
char userPin[5];
```

```
char lastSerial[14];
char userSerial[14];
char serialNum[14];
long counter = 0;
int timecounter = 0;
int printcounter = 1;
int testcounter = 0;
int waitcounter = 0;
int pollCounter2 = 0;
sessionStruct currSession;
const int calendar[] = {
    31,31,31,30,31,30,31,31,30,31,30,31
};

INT8 SendData;             //1 if data needs sent. 0 if not.
INT8 RcvData;              //1 if data needs sent. 0 if not.
char toSend[15] = "I:0413AAC34F3A";
char rcvPacket[55];

struct netif localmachine;

extern  tU08   gotlink;


/*************************/
/*  Function Prototypes   */
/*************************/
void PrintReceipt(void);
void PollKeypad(void);
void ProcInit(void);
void LCDInit(void);
//void EthInit(void);
void UpdateLCD(void);
void SetupSession(void);
void WaitFor(int);
int HandlePacket(void);
extern void RTI_Enable (void);
void mystrcpy(char *, char*, int, int);


/*************************/
/*         main()         */
/*************************/
void main()
{
    INT16 len;
    INT32 i;
    INT32 j = 0;
    INT32 k = 0;

    state = TIME_SYNCH;
    flags = 0x00;
    SendData = 0;

    ProcInit();
    LCDInit();
    SetupSession();

    _INIT_DEBUG();

    /* Set our network information. This is for static configuration.
        if using BOOTP or DHCP this will be a bit different. */
```

```
  /* IP address */
  localmachine.localip = *((UINT32 *)ip_address);

  /* Default gateway */
  localmachine.defgw   = *((UINT32 *)ip_gateway);

  /* Subnet mask */
  localmachine.netmask = *((UINT32 *)ip_netmask);

  /* Ethernet (MAC) address */
  localmachine.localHW[0] = hard_addr[0];
  localmachine.localHW[1] = hard_addr[1];
  localmachine.localHW[2] = hard_addr[2];
  localmachine.localHW[3] = hard_addr[3];
  localmachine.localHW[4] = hard_addr[4];
  localmachine.localHW[5] = hard_addr[5];

  /* Init system services   */
  timer_pool_init();

  /* Initialize all buffer descriptors */
  mBufInit ();

  /* Interrupts can be enabled AFTER timer pool has been initialized */
  /* Initialize all network layers */
  EtherInit();

  asm cli;

/* Initialize required network protocols */
  arp_init();
  (void) udp_init();
  udp_demo_init();

  /* Enable RTI */
  RTI_Enable ();

  while (1)
  {
      if (gotlink) {
          /* Try to receive Ethernet Frame */
          if( NETWORK_CHECK_IF_RECEIVED() == TRUE ) {
              switch( received_frame.protocol)
              {
              case PROTOCOL_ARP:
                      process_arp (&received_frame);
                      break;
                  case PROTOCOL_IP:
                      len = process_ip_in(&received_frame);
                  if(len < 0)
                      break;
                  switch (received_ip_packet.protocol)
                      {
                          case IP_ICMP:
                              process_icmp_in (&received_ip_packet, len);
                              break;
                      case IP_UDP:
                              process_udp_in (&received_ip_packet,len);
                              break;
                      default:
                          break;
                  }
                  break;
```

```
            default:
                break;
        }
        /* discard received frame */
        NETWORK_RECEIVE_END();
    }
    arp_manage();
    udp_demo_run();
}

if (state == TIME_SYNCH) {
    if (!(flags & ONLY_ONCE_MASK)) {
        UpdateLCD();
        toSend[0] = 'T';
        SetBit(ONLY_ONCE, flags);
        SetBit(MASK_CANCEL_FLAG, flags);
    }

    SendData = 1;
    if(RcvData) {
        if (HandlePacket()) {
            flags = 0x00;
            state = IDLE;
        } else {
            flags = 0x00;
            state = TIME_SYNCH;
        }
    }
}

else if (state == IDLE) {
    if (!(flags & ONLY_ONCE_MASK)) {
        UpdateLCD();
        SetBit(ONLY_ONCE, flags);
        SetBit(CHK_RFID_FLAG, flags);
        SetBit(MASK_CANCEL_FLAG, flags);
    }

    if (flags & RFID_FLAG_MASK)        // RFID interrupt occurred
    {
        mystrcpy(userSerial,serialNum,13,13);
        state = USER_QUERY_DB;
        flags = 0x00;
    }
}

else if (state == USER_QUERY_DB) {
    if (!(flags & ONLY_ONCE_MASK)) {
        toSend[0] = 'U';
        mystrcpy(&toSend[1],userSerial,13,13);
        SetBit(ONLY_ONCE, flags);
    }

    SendData = 1;
    if(RcvData) {
        if (HandlePacket()) {
            pinCounter = 0;
            flags = 0x00;
            state = CAPTURE_PIN;
        } else {
            flags = 0x00;
            state = INVALID_USER;
        }
```

```
        }
    }

    else if (state == INVALID_USER) {
        UpdateLCD();
        WaitFor(4);
        asm sei;
        SetupSession();
        asm cli;
        state = IDLE;
    }

    else if (state == CAPTURE_PIN) {
        if (!(flags & ONLY_ONCE_MASK)) {
            UpdateLCD();
            SetBit(CHK_KEYPAD_FLAG, flags);
            SetBit(ONLY_ONCE, flags);
        }
        if (flags & KEYPAD_FLAG_MASK) {
            if (pinCounter < 4) {
                if ((keyPressed >= 0) && (keyPressed <= 9)) {
                    currentPin[pinCounter] = keyPressed + '0';
                    pinCounter++;
                    UpdateLCD();
                } else if (keyPressed == KEY_CLEAR) {
                    if (pinCounter > 0) {
                        pinCounter--;
                        UpdateLCD();
                    }
                }
        } else {
                if (keyPressed == KEY_CLEAR) {
                    pinCounter--;
                    UpdateLCD();
                } else if (keyPressed == KEY_ENTER) {
                    for (j = 0; j < 4; j++) {
                        if (currentPin[j] != userPin[j]) {
                            state = INVALID_USER;
                            break;
                        }
                    }
                    if (j == 4) {
                        state = SESSION_START;
                    }
                    flags = 0x00;
                }
            }
            ClearBit(KEYPAD_FLAG, flags);
        }
    }

    else if (state == SESSION_START) {
        if (!(flags & ONLY_ONCE_MASK)) {
            UpdateLCD();
            SetBit(ONLY_ONCE, flags);
            SetBit(CHK_RFID_FLAG, flags);
        }
        if (flags & RFID_FLAG_MASK) {
            state = ITEM_QUERY_DB;           // First item RFID was swiped
            flags = 0x00;
        }
    }
```

```
else if (state == SESSION_RUNNING) {
    if (!(flags & ONLY_ONCE_MASK)) {
        UpdateLCD();
        SetBit(ONLY_ONCE, flags);
        SetBit(CHK_KEYPAD_FLAG, flags);
        SetBit(CHK_RFID_FLAG, flags);
    }
    if (flags & RFID_FLAG_MASK) {     // If next item RFID was swiped
        state = ITEM_QUERY_DB;
        flags = 0x00;
    }
    if (flags & KEYPAD_FLAG_MASK) {   // If key was pressed
        if (keyPressed == KEY_CLEAR) {
            state = CONFIRM_REMOVAL;
        } else if (keyPressed == KEY_ENTER) {
            state = SESSION_END;
        } else {                       // Digit, #, *, or ? key pressed
            state = INVALID_KEYPRESS;
        }
        flags = 0x00;
    }
}

else if (state == INVALID_KEYPRESS) {
    UpdateLCD();
    WaitFor(4);
    state = SESSION_RUNNING;
}

else if (state == ITEM_QUERY_DB) {
    if (!(flags & ONLY_ONCE_MASK)) {
        for (i = 0; i < 13; i++) {
            if (lastSerial[i] != serialNum[i]) {
                mystrcpy(lastSerial,serialNum,13,13);
                break;
            } else if (i == 12) {
                flags = 0x00;
                SetBit(ONLY_ONCE, flags);
                SetBit(CHK_KEYPAD_FLAG, flags);
                SetBit(CHK_RFID_FLAG, flags);
                state = SESSION_RUNNING;
            }
        }
        if (state != SESSION_RUNNING) {
            toSend[0] = 'I';
            mystrcpy(&toSend[1],serialNum,13,13);
            SetBit(ONLY_ONCE, flags);
        }
    }

    if (state != SESSION_RUNNING) {
        SendData = 1;
    }

    if(RcvData) {
        if (HandlePacket()) {
            state = SESSION_RUNNING;
        } else {
            state = ITEM_NOT_MATCHED;
        }
        flags = 0x00;
    }
}
```

```
else if (state == ITEM_NOT_MATCHED) {
    UpdateLCD();
    WaitFor(4);
    state = SESSION_RUNNING;
}

else if (state == CANCEL_SESSION) {
    if (!(flags & CANCEL_ONCE_MASK)) {
        UpdateLCD();
        ClearBit(ONLY_ONCE, flags);
        SetBit(CANCEL_ONCE, flags);
        SetBit(CHK_KEYPAD_FLAG, flags);
        SetBit(MASK_CANCEL_FLAG, flags);
    }
    if (flags & KEYPAD_FLAG_MASK) {
        // Check for confirmation of cancel
        if (keyPressed == KEY_ENTER) {
            // User confirmed cancel
            flags = 0x00;
            state = FINISH_SESSION;
        } else if (keyPressed == KEY_CANCEL) {
            // User changed mind - no cancel
            flags = 0x00;
            state = returnState;
        }
        ClearBit(KEYPAD_FLAG, flags);
    }
}

else if (state == CONFIRM_REMOVAL) {
    if (!(flags & ONLY_ONCE_MASK)) {
        UpdateLCD();
        SetBit(ONLY_ONCE, flags);
        SetBit(CHK_KEYPAD_FLAG, flags);
        SetBit(MASK_CANCEL_FLAG, flags);
    }
    if (flags & KEYPAD_FLAG_MASK) {
        // Check for confirmation of item removal from cart
        if (keyPressed == KEY_ENTER) {
            // User confirmed item removal from cart
            flags = 0x00;
            state = REMOVE_LAST_ITEM;
        } else if (keyPressed == KEY_CANCEL) {
            // User changed mind - do not remove item, return to cart
            flags = 0x00;
            state = SESSION_RUNNING;
        }
        ClearBit(KEYPAD_FLAG, flags);
    }
}

else if (state == REMOVE_LAST_ITEM) {
    if (!(flags & ONLY_ONCE_MASK)) {
        toSend[0] = 'R';
        mystrcpy(&toSend[1],
            currSession.items[currSession.cartSizeI-  1].itemSerial, 13, 13);
        UpdateLCD();
        SetBit(ONLY_ONCE, flags);
        SetBit(MASK_CANCEL_FLAG, flags);
    }
    SendData = 1;
    if(RcvData) {
```

```
            HandlePacket();
            state = SESSION_RUNNING;
            flags = 0x00;
        }
    }

    else if (state == SESSION_END) {
        if (!(flags & ONLY_ONCE_MASK)) {
            UpdateLCD();
            SetBit(CHK_KEYPAD_FLAG, flags);
            SetBit(MASK_CANCEL_FLAG, flags);
            SetBit(ONLY_ONCE, flags);
        }
        if (flags & KEYPAD_FLAG_MASK) {
            // Ask user for print and/or email receipt
            if (keyPressed == KEY_ENTER) {
                // User selected print receipt
                flags = 0x00;
                state = PRINT;
            } else if (keyPressed == KEY_CANCEL) {
                // User refused print receipt - only email
                flags = 0x00;
                state = EMAIL;
            }
            ClearBit(KEYPAD_FLAG, flags);
        }
    }

    else if (state == PRINT) {
        UpdateLCD();
        asm sei;
        PrintReceipt();
        asm cli;
        WaitFor(3);
        state = EMAIL;  // Email receipt by default
    }

    else if (state == EMAIL) {
        if (!(flags & ONLY_ONCE_MASK)) {
            toSend[0] = 'M';
            mystrcpy(&toSend[1],userSerial,13,13);
            UpdateLCD();
            SetBit(ONLY_ONCE, flags);
        }
        SendData = 1;
        if(RcvData) {
            HandlePacket();
            WaitFor(3);
            state = FINISH_SESSION;
            flags = 0x00;
        }
    }

    else if (state == FINISH_SESSION) {
        UpdateLCD();
        SetupSession();
        WaitFor(8);
        asm cli;
        state = IDLE;
    }

    PollKeypad();
} /* End Infinite Main Loop */
```

```
}


/*************************/
/*       mystrcpy()      */
/*************************/

void mystrcpy(char *dst, char*src, int dstlen, int srclen) {
    int i = 0;

    for (i = 0; i < dstlen; i++) {
        if ( i < srclen) {
            dst[i] = src[i];
        } else {
            dst[i] = ' ';
        }
    }
}


/*************************/
/*     SetupSession()    */
/*************************/

void SetupSession(void) {
    currSession.cartSizeI = 0;
    currSession.cartTotalI = 0;
    currSession.userName[20] = NULL;
    currSession.userEmail[30] = NULL;
    pinCounter = 0;
    lastSerial[0] = '?';
}


/*************************/
/*     RFIDInterrupt()   */
/*************************/

interrupt void RFIDInterrupt(void) {
    int startFlag = 0;
    int mycounter = 0;
    char holder = ' ';

    SCI1SR1;
    holder = SCI1DRL;

    if (!(CHK_RFID_MASK & flags)) {
        return;
    }

    serialNum[0] = holder;

    if (SCI1DRL == 0x3A) {
        startFlag = 1;
        mycounter++;
    } else {
        return;
    }

    while (mycounter < 14) {
        while ((SCI1SR1 & SCI1SR1_RDRF_MASK) == 0);
        if (SCI1DRL == 0x3A) startFlag = 1; // Received serial start character (:)
        if (startFlag == 1) {
```

```
            serialNum[mycounter] = SCI1DRL;
            mycounter++;
        }
    }

    SetBit(RFID_FLAG, flags);
}


/*************************/
/* HandleTimerInterrupt() */
/*************************/

interrupt void HandleTimerInterrupt(void) {
    if (pollCounter2 == 0) {
        PTJ_PTJ6 = 1;                       // Set C1 to high
        PTJ_PTJ7 = 0;
        PTT_PTT4 = 0;
        PTT_PTT5 = 0;
    } else if (pollCounter2 == 1) {
        PTJ_PTJ6 = 0;
        PTJ_PTJ7 = 1;                       // Set C2 to high
        PTT_PTT4 = 0;
        PTT_PTT5 = 0;
    } else if (pollCounter2 == 2) {
        PTJ_PTJ6 = 0;
        PTJ_PTJ7 = 0;
        PTT_PTT4 = 1;                        // Set C3 to high
        PTT_PTT5 = 0;
    } else if (pollCounter2 == 3) {
        PTJ_PTJ6 = 0;
        PTJ_PTJ7 = 0;
        PTT_PTT4 = 0;
        PTT_PTT5 = 1;                        // Set C4 to high
    }
    pollCounter2 = (pollCounter2 + 1) % 4;

    TFLG1_C6F = 1;
    timecounter++;
    if (timecounter == 25) {
        currSession.timeSecsI++;
        if (currSession.timeSecsI > 59) {
            currSession.timeSecsI = 0;
            currSession.timeMinI++;
            if (currSession.timeMinI > 59) {
                currSession.timeMinI = 0;
                currSession.timeHrI++;
                if (currSession.timeHrI == 12) {
                    if (currSession.timeHalfI == AM) {
                        currSession.timeHalfI = PM;
                    } else {
                        currSession.timeHalfI = AM;
                        currSession.dateDayI++;
                        if (currSession.dateDayI > calendar[currSession.dateMonthI]) {
                            currSession.dateDayI = 1;
                            currSession.dateMonthI++;
                            if (currSession.dateMonthI > 12) {
                                currSession.dateMonthI = 1;
                                currSession.dateYearI++;
                                if (currSession.dateYearI > 99) {
                                    currSession.dateYearI = 0;
                                }
                            }
```

```
                    }
                }
            } else if (currSession.timeHrI > 12) {
                currSession.timeHrI = 1;
            }
        }
        if ((state == SESSION_RUNNING)) {
            UpdateLCD();
        }
    }
    waitcounter++;
    timecounter = 0;
  }
}


/************************/
/*      ProcInit()      */
/************************/

void ProcInit(void) {
    SCI0BD = 163;               // Printer port - 9600 baud
    SCI0CR2 = 0x08;             // Printer port - transmit only
    SCI1BD = 163;               // RFID reader port - 9600 baud
    SCI1CR2 = 0x24;             // RFID reader port - receive interrupt mode
    DDRG = 0xFF;
    DDRH = 0x07;
    PERH = 0x78;
    PPSH = 0x78;
    DDRJ_DDRJ0 = 1;
    DDRJ_DDRJ6 = 1;
    DDRJ_DDRJ7 = 1;
    DDRT_DDRT4 = 1;
    DDRT_DDRT5 = 1;

    TSCR1 = 0x80;
    TCTL1 = 0x00;
    TIOS = 0xF0;
    TIE = 0x40;
    TSCR2 = 0x0E;
    TC7 = 0x3D0A;
    TC6 = 0x3D09;

    CLKSEL=0;
    CLKSEL_PLLSEL = 0;          // Select clock source from XTAL
    PLLCTL_PLLON = 0;           // Disable the PLL
    SYNR = 0;                   // Set the multiplier register
    REFDV = 0;                  // Set the divider register
    PLLCTL = 192;
    PLLCTL_PLLON = 1;           // Enable the PLL
    while(!CRGFLG_LOCK);        // Wait
    CLKSEL_PLLSEL = 1;          // Select clock source from PLL

    INTCR_IRQEN = 0;  // Disable the IRQ interrupts, enable after CPU reset by default
}


/************************/
/*       LCDInit()      */
/************************/

void LCDInit(void) {
    LCD_reset();
```

```
}


/*************************/
/*     HandlePacket()     */
/*************************/

int HandlePacket(void) {
    char mynum[3] = "00";
    int startpos = 0;
    int i = 0;

    SendData = 0;
    RcvData = 0;
    if (rcvPacket[0] == '~') {
        return 0;
    } else if (state == TIME_SYNCH) {

        /* Grab Month */
        if (rcvPacket[startpos+1] == '/') {
            currSession.dateMonthI = rcvPacket[startpos] - '0';
            startpos += 2;
        } else {
            currSession.dateMonthI = 10 + (rcvPacket[startpos+1] - '0');
            startpos += 3;
        }

        /* Grab Day */
        if (rcvPacket[startpos+1] == '/') {
            currSession.dateDayI = rcvPacket[startpos] - '0';
            startpos += 2;
        } else {
            currSession.dateDayI = (10 * (rcvPacket[startpos] - '0')) +
                (rcvPacket[startpos+1] - '0');
            startpos += 3;
        }

        /* Grab Year */
        currSession.dateYearI = (10 * (rcvPacket[startpos] - '0')) +
            (rcvPacket[startpos+1] - '0');
        startpos += 3;

        /* Grab Hours */
        if (rcvPacket[startpos+1] == ':') {
            currSession.timeHrI = rcvPacket[startpos] - '0';
            startpos += 2;
        } else {
            currSession.timeHrI = 10 + (rcvPacket[startpos+1] - '0');
            startpos += 3;
        }

        /* Grab Minutes */
        currSession.timeMinI = (10 * (rcvPacket[startpos] - '0')) +
            (rcvPacket[startpos+1] - '0');
        startpos += 3;

        /* Grab Seconds */
        currSession.timeSecsI = (10 * (rcvPacket[startpos] - '0')) +
            (rcvPacket[startpos+1] - '0');
        startpos += 3;

        if (rcvPacket[startpos] == 'A') {
            currSession.timeHalfI = AM;
```

```
        } else {
            currSession.timeHalfI = PM;
        }

    } else if (state == USER_QUERY_DB) {
        mystrcpy(currSession.userName, rcvPacket, 20, 20);
        i = 19;
        while(rcvPacket[i] == ' ') {
            i--;
        }
        currSession.nameLength = i + 1;
        mystrcpy(currSession.userEmail, &rcvPacket[20], 30, 30);
        i = 49;
        while(rcvPacket[i] == ' ') {
            i--;
        }
        currSession.emailLength = i + 1 - 20;
        mystrcpy(userPin, &rcvPacket[50], 4, 4);
    } else if (state == REMOVE_LAST_ITEM) {

        /* Recalculate cart total by subtracting last item's price */
        if (currSession.items[currSession.cartSizeI-1].itemPrice[0] != ' ') {
            currSession.cartTotalI -= 10000 *
                (currSession.items[currSession.cartSizeI-1].itemPrice[0] - '0');
        }
        if (currSession.items[currSession.cartSizeI-1].itemPrice[1] != ' ') {
            currSession.cartTotalI -= 1000 *
                (currSession.items[currSession.cartSizeI-1].itemPrice[1] - '0');
        }
        if (currSession.items[currSession.cartSizeI-1].itemPrice[2] != ' ') {
            currSession.cartTotalI -= 100 *
                (currSession.items[currSession.cartSizeI-1].itemPrice[2] - '0');
        }
        if (currSession.items[currSession.cartSizeI-1].itemPrice[3] != ' ') {
            currSession.cartTotalI -= 10 *
                (currSession.items[currSession.cartSizeI-1].itemPrice[3] - '0');
        }
        if (currSession.items[currSession.cartSizeI-1].itemPrice[4] != ' ') {
            currSession.cartTotalI -= 1 *
                (currSession.items[currSession.cartSizeI-1].itemPrice[4] - '0');
        }
        currSession.cartSizeI--;
        mystrcpy(lastSerial,
            currSession.items[currSession.cartSizeI-1].itemSerial, 13, 13);
    } else if (state == ITEM_QUERY_DB) {
        mystrcpy(currSession.items[currSession.cartSizeI].itemSerial, serialNum, 13,
            13);
        mystrcpy(currSession.items[currSession.cartSizeI].itemName, rcvPacket, 25, 25);
        i = 24;
        while(rcvPacket[i] == ' ') {
            i--;
        }
        currSession.items[currSession.cartSizeI].nameLength = i + 1;
        mystrcpy(currSession.items[currSession.cartSizeI].itemPrice, &rcvPacket[25], 5,
            5);

        /* Calculate new cart total by adding new item price */
        if (currSession.items[currSession.cartSizeI].itemPrice[0] != ' ') {
            currSession.cartTotalI += 10000 *
                (currSession.items[currSession.cartSizeI].itemPrice[0] - '0');
        }
        if (currSession.items[currSession.cartSizeI].itemPrice[1] != ' ') {
```

```
            currSession.cartTotalI += 1000 *
                (currSession.items[currSession.cartSizeI].itemPrice[1] - '0');
        }
        if (currSession.items[currSession.cartSizeI].itemPrice[2] != ' ') {
            currSession.cartTotalI += 100 *
                (currSession.items[currSession.cartSizeI].itemPrice[2] - '0');
        }
        if (currSession.items[currSession.cartSizeI].itemPrice[3] != ' ') {
            currSession.cartTotalI += 10 *
                (currSession.items[currSession.cartSizeI].itemPrice[3] - '0');
        }
        if (currSession.items[currSession.cartSizeI].itemPrice[4] != ' ') {
            currSession.cartTotalI += 1 *
                (currSession.items[currSession.cartSizeI].itemPrice[4] - '0');
        }
        currSession.cartSizeI++;
    }

    return 1;
}


/*************************/
/*      UpdateLCD()      */
/*************************/

void UpdateLCD(void) {
    int item_counter, e, f, flag = 0;
    char indexStr[3] = " 0";
    char datetime[18] = "11:11AM  11/11/11";
    char datetime2[21] = "11:11:11AM  11/11/11";
    char mynumber[3] = "00";
    char myprice[8] = "$000.00";
    char temp = ' ';

    LCD_clr_scr_char(0);
    if (state == IDLE) {
        LCD_graphic_ini();

        /* Write 4 blank rows */
        for (e = 0; e < 4; e++) {
            for (f = 0; f < 30; f++) {
                 LCD_write_data(0x00);
            }
        }

        /* Write RFID Xpress 109 row logo */
        for (e = 0; e < LOGO_SIZE; e++) {
            LCD_write_data(logo_bitmap[e]);
        }

        /* Write 'please scan your keyfob' 12 row logo */
        for (e = 0; e < FOOTER_SIZE; e++) {
            LCD_write_data(please_scan_bitmap[e]);
        }

        /* Write 3 blank rows */
        for (e = 0; e < 3; e++) {
            for (f = 0; f < 30; f++) {
                 LCD_write_data(0x00);
            }
        }
    } else if (state == TIME_SYNCH) {
```

```
        LCD_graphic_ini();

        /* Write 4 blank rows */
        for (e = 0; e < 4; e++) {
            for (f = 0; f < 30; f++) {
                LCD_write_data(0x00);
            }
        }

        /* Write RFID Xpress 109 row logo */
        for (e = 0; e < LOGO_SIZE; e++) {
            LCD_write_data(logo_bitmap[e]);
        }

        /* Write 3 rows blank */
        for (e = 0; e < 15; e++) {
            for (f = 0; f < 30; f++) {
                LCD_write_data(0x00);
            }
        }
    } else if (state == INVALID_USER) {      // Invalid keyfob scanned
        LCD_char_ini();
        LCD_set_char_add(5,4);
        LCD_print_string("Sorry, customer not recognized.");
    } else if (state == CAPTURE_PIN) {
        LCD_char_ini();
                // Prompt user for PIN
        LCD_set_char_add(5,9);
        LCD_print_string("Welcome, ");
        LCD_print_string(currSession.userName);
        LCD_set_char_add(6,5);
        LCD_print_string("Please enter your 4-digit PIN:");
        LCD_set_char_add(8,17);
        if (pinCounter == 0)
            LCD_print_string(" ");
        else if (pinCounter == 1)
            LCD_print_string("*");
        else if (pinCounter == 2)
            LCD_print_string("* *");
        else if (pinCounter == 3)
            LCD_print_string("* * *");
        else if (pinCounter == 4)
            LCD_print_string("* * * *");
        else
            LCD_print_string("FATAL ERROR");
    } else if (state == SESSION_START) {        // Instruct user to scan first item
        LCD_char_ini();
        LCD_set_char_add(5,6);
        LCD_print_string("Please scan your first item");
    } else if (state == SESSION_RUNNING) {          // Shopping cart display
        LCD_char_ini();
        LCD_set_char_add(0,0);
        LCD_print_string("RFID Xpr3ss");
        LCD_set_char_add(0,21);
        mynumber[0] = (currSession.timeHrI / 10) + '0';
        mynumber[1] = (currSession.timeHrI % 10) + '0';
        mystrcpy(&datetime[0],mynumber,2,2);
        mynumber[0] = (currSession.timeMinI / 10) + '0';
        mynumber[1] = (currSession.timeMinI % 10) + '0';
        mystrcpy(&datetime[3],mynumber,2,2);
        mynumber[0] = (currSession.dateMonthI / 10) + '0';
        mynumber[1] = (currSession.dateMonthI % 10) + '0';
        mystrcpy(&datetime[9],mynumber,2,2);
```

```
mynumber[0] = (currSession.dateDayI / 10) + '0';
mynumber[1] = (currSession.dateDayI % 10) + '0';
mystrcpy(&datetime[12],mynumber,2,2);
mynumber[0] = (currSession.dateYearI / 10) + '0';
mynumber[1] = (currSession.dateYearI % 10) + '0';
mystrcpy(&datetime[15],mynumber,2,2);
mynumber[1] = 'M';
if (currSession.timeHalfI == AM) {
    mynumber[0] = 'A';
} else {
    mynumber[0] = 'P';
}
mystrcpy(&datetime[5],mynumber,2,2);
LCD_print_string(datetime);
LCD_set_char_add(1,0);
LCD_print_string("ITEM                                PRICE ");

/* Print last 7 scanned items in shopping cart to allow for automatic
 * scrolling
 */
for (item_counter=currSession.cartSizeI-7; item_counter<currSession.cartSizeI;
    item_counter++)
{
    if (currSession.cartSizeI == 0){break;}
    if (item_counter<0){item_counter=0;}
    if (item_counter < 9) {
        indexStr[0] = ' ';
    } else {
        indexStr[0] = (char)(((item_counter+1) / 10)+'0');
    }
    indexStr[1] = (char)(((item_counter+1) % 10) + '0');
    LCD_print_string(indexStr);
    LCD_print_string(" ");
    LCD_print_string(currSession.items[item_counter].itemName);
    LCD_print_string("     ");
    mystrcpy(&myprice[1], currSession.items[item_counter].itemPrice,3,3);
    mystrcpy(&myprice[5], &currSession.items[item_counter].itemPrice[3],2,2);
    LCD_print_string(myprice);
}

LCD_set_char_add(9,33);
LCD_print_string("-------");

temp = (currSession.cartSizeI / 10) + '0';
if (temp == '0') {
    mynumber[0] = ' ';
} else {
    mynumber[0] = temp;
}

mynumber[1] = (currSession.cartSizeI % 10) + '0';

LCD_set_char_add(10,0);
LCD_print_string(mynumber);
LCD_print_string(" items in your cart");
LCD_set_char_add(10,3);
LCD_print_string("items in your cart    TOTAL: $");

/* Decode integer cart total into its character representation */
temp = (currSession.cartTotalI / 100000) + '0';
if ((temp == '0') && (flag == 0)) {
    myprice[0] = ' ';
} else {
```

```
            myprice[0] = temp;
            flag = 1;
        }

        temp = ((currSession.cartTotalI % 100000) / 10000) + '0';
        if ((temp == '0') && (flag == 0)) {
            myprice[1] = ' ';
        } else {
            myprice[1] = temp;
            flag = 1;
        }

        temp = ((currSession.cartTotalI % 10000) / 1000) + '0';
        if ((temp == '0') && (flag == 0)) {
            myprice[2] = ' ';
        } else {
            myprice[2] = temp;
            flag = 1;
        }

        temp = ((currSession.cartTotalI % 1000) / 100) + '0';
        if ((temp == '0') && (flag == 0)) {
            myprice[3] = ' ';
        } else {
            myprice[3] = temp;
            flag = 1;
        }

        temp = ((currSession.cartTotalI % 100) / 10) + '0';
        if ((temp == '0') && (flag == 0)) {
            myprice[5] = ' ';
        } else {
            myprice[5] = temp;
            flag = 1;
        }

        temp = ((currSession.cartTotalI % 10) / 1) + '0';
        if ((temp == '0') && (flag == 0)) {
            myprice[6] = ' ';
        } else {
            myprice[6] = temp;
            flag = 1;
        }
        LCD_print_string(myprice);
        LCD_set_char_add(11,1);
        LCD_print_string("[ PRESS ENTER TO FINISH ]");
    } else if (state == INVALID_KEYPRESS) {  // Keypress not recognized
        LCD_char_ini();
        LCD_set_char_add(5,10);
        LCD_print_string("Key not recognized");
    } else if (state == ITEM_NOT_MATCHED) {  // Item RFID not found
        LCD_char_ini();
        LCD_set_char_add(5,10);
        LCD_print_string("Item not recognized");
    } else if (state == CANCEL_SESSION) {
        // Confirm user wants to cancel shopping session
        LCD_char_ini();
        LCD_set_char_add(4,0);
        LCD_print_string(" Are you sure you want to quit shopping?");
        LCD_set_char_add(6,0);
        if (returnState == CAPTURE_PIN) {
            LCD_print_string("   Press ENTER  to confirm,             ");
            LCD_print_string("         CANCEL to return to pin entry  ");
```

```
        } else {
            LCD_print_string("     Press ENTER  to confirm,           ");
            LCD_print_string("           CANCEL to return to cart");
        }
    } else if ((state == REMOVE_LAST_ITEM) || (state == CONFIRM_REMOVAL)) {
        // Confirm user wants to remove last item
        LCD_char_ini();
        LCD_set_char_add(2,5);
        LCD_print_string("Are you sure you want to remove         ");
        LCD_set_char_add(4,(20 - currSession.items[currSession.cartSizeI-1].nameLength
            / 2));
        LCD_print_string(currSession.items[currSession.cartSizeI-1].itemName);
        LCD_set_char_add(6,13);
        LCD_print_string("from your cart?");
        LCD_set_char_add(8,0);
        LCD_print_string("      Press ENTER  to remove item,      ");
        LCD_print_string("            CANCEL to return to cart");
    } else if (state == SESSION_END) {    // Prompt user for printed receipt
        LCD_char_ini();
        LCD_set_char_add(5,5);
        LCD_print_string("Do you want a printed receipt?");
        LCD_set_char_add(7,0);
        LCD_print_string("         Press ENTER  for yes,          ");
        LCD_print_string("               CANCEL to skip");
    } else if (state == PRINT) {          // Print receipt
        LCD_char_ini();
        LCD_set_char_add(4,15);
        LCD_print_string("Printing...");
        LCD_set_char_add(6,7);
        LCD_print_string("Please take your receipt...");
    } else if (state == EMAIL) {          // Email receipt
        LCD_char_ini();
        LCD_set_char_add(4,11);
        LCD_print_string("Emailing receipt to");
        LCD_set_char_add(6,(20 - currSession.emailLength / 2));
        LCD_print_string(currSession.userEmail);
    } else if (state == FINISH_SESSION) { // Thank user for shopping
        LCD_graphic_ini();

        /* Write 4 blank rows */
        for (e = 0; e < 4; e++) {
            for (f = 0; f < 30; f++) {
                LCD_write_data(0x00);
            }
        }

        /* Write RFID Xpress 109 row logo */
        for (e = 0; e < LOGO_SIZE; e++) {
            LCD_write_data(logo_bitmap[e]);
        }

        /* Write 'thank you for shopping' 12 row logo */
        for (e = 0; e < FOOTER_SIZE; e++) {
            LCD_write_data(thank_you_bitmap[e]);
        }

        /* Write 3 blank rows */
        for (e = 0; e < 3; e++) {
            for (f = 0; f < 30; f++) {
                LCD_write_data(0x00);
            }
        }
    } else {
```

```
        LCD_set_char_add(5,10);
        LCD_print_string("UNKNOWN ERROR");
    }
}


/*************************/
/*       WaitFor()       */
/*************************/

void WaitFor(int time) {
    waitcounter = 0;
    while (waitcounter < time);
}


/*************************/
/*     PollKeypad()      */
/*************************/

void PollKeypad(void)
{
    asm sei;
    if (PTH_PTH6 == 1) {                  // If row 1 is pressed
        if (pollCounter2 == 1) {          // If col 1 is pressed
            keyPressed = KEY_ONE;
            SetBit(KEYPAD_FLAG, flags);
        } else if (pollCounter2 == 2) {   // If col 2 is pressed
            keyPressed = KEY_TWO;
            SetBit(KEYPAD_FLAG, flags);
        } else if (pollCounter2 == 3) {   // If col 3 is pressed
            keyPressed = KEY_THREE;
            SetBit(KEYPAD_FLAG, flags);
        } else {                          // If col 4 is pressed
            keyPressed = KEY_ENTER;
            SetBit(KEYPAD_FLAG, flags);
        }
    } else if (PTH_PTH5 == 1) {           // If row 2 is pressed
        if (pollCounter2 == 1) {          // If col 1 is pressed
            keyPressed = KEY_FOUR;
            SetBit(KEYPAD_FLAG, flags);
        } else if (pollCounter2 == 2) {   // If col 2 is pressed
            keyPressed = KEY_FIVE;
            SetBit(KEYPAD_FLAG, flags);
        } else if (pollCounter2 == 3) {   // If col 3 is pressed
            keyPressed = KEY_SIX;
            SetBit(KEYPAD_FLAG, flags);
        } else {                          // If col 4 is pressed
            keyPressed = KEY_CLEAR;
            SetBit(KEYPAD_FLAG, flags);
        }
    } else if (PTH_PTH4 == 1) {           // If row 3 is pressed
        if (pollCounter2 == 1) {          // If col 1 is pressed
            keyPressed = KEY_SEVEN;
            SetBit(KEYPAD_FLAG, flags);
        } else if (pollCounter2 == 2) {   // If col 2 is pressed
            keyPressed = KEY_EIGHT;
            SetBit(KEYPAD_FLAG, flags);
        } else if (pollCounter2 == 3) {   // If col 3 is pressed
            keyPressed = KEY_NINE;
            SetBit(KEYPAD_FLAG, flags);
        } else {                          // If col 4 is pressed
            keyPressed = KEY_HELP;
```

```
            SetBit(KEYPAD_FLAG, flags);
        }
    } else if (PTH_PTH3 == 1) {           // If row 4 is pressed
        if (pollCounter2 == 1) {          // If col 1 is pressed
            keyPressed = KEY_STAR;
            SetBit(KEYPAD_FLAG, flags);
        } else if (pollCounter2 == 2) {   // If col 2 is pressed
            keyPressed = KEY_ZERO;
            SetBit(KEYPAD_FLAG, flags);
        } else if (pollCounter2 == 3) {   // If col 3 is pressed
            keyPressed = KEY_POUND;
            SetBit(KEYPAD_FLAG, flags);
        } else {                          // If col 4 is pressed
            keyPressed = KEY_CANCEL;
            SetBit(KEYPAD_FLAG, flags);
        }
    }

    /* Wait until key is released */
    while (PTH_PTH6 || PTH_PTH5 || PTH_PTH4 || PTH_PTH3);

    asm cli;

    if (flags & KEYPAD_FLAG_MASK) {        // Cancel button pressed
        if ((keyPressed == KEY_CANCEL) && (!(flags & MASK_CANCEL_MASK))) {
            returnState = state;
            state = CANCEL_SESSION;
            ClearBit(KEYPAD_FLAG, flags);
        } else if (!(flags & CHK_KEYPAD_FLAG_MASK)) {
            ClearBit(KEYPAD_FLAG, flags);
        }
    }
}


/*************************/
/*     PrintReceipt()     */
/*************************/

void PrintReceipt(void)
{
    int printcounter = 0;
    const char Logo[] = "*********************************            \\\\//
**         RFID // PRESS        **             //\\\\
*********************************";
    char DashDash[32] =  "                          -------";
    char Total[32] =     "                 Total: $1111.11";
    char ThankYou[32] =  " Thank you for shopping with us.";
    char ComeAgain[32] = "       Please come again!       ";
    char datetime[18] = "11:11AM  11/11/11";
    int i, j, itemLen, priceLen, flag = 0;
    char datetime2[21] = "11:11:11AM  11/11/11";
    char mynumber[3] = "00";
    char myprice[9] = "$0000.00";
    char temp = ' ';

    /* Transmit CR and LF */
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = 0x0D;
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = 0x0A;
```

```
/* Print RFID Xpress Logo */
while (printcounter < 160) {
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = Logo[printcounter];
    printcounter++;
}

/* Transmit CR and LF */
while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
SCI0DRH = 0;
SCI0DRL = 0x0D;
while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
SCI0DRH = 0;
SCI0DRL = 0x0A;
while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
SCI0DRH = 0;
SCI0DRL = 0x0A;

mynumber[0] = (currSession.timeHrI / 10) + '0';
mynumber[1] = (currSession.timeHrI % 10) + '0';
mystrcpy(&datetime2[0],mynumber,2,2);
mynumber[0] = (currSession.timeMinI / 10) + '0';
mynumber[1] = (currSession.timeMinI % 10) + '0';
mystrcpy(&datetime2[3],mynumber,2,2);
mynumber[0] = (currSession.timeSecsI / 10) + '0';
mynumber[1] = (currSession.timeSecsI % 10) + '0';
mystrcpy(&datetime2[6],mynumber,2,2);
mynumber[0] = (currSession.dateMonthI / 10) + '0';
mynumber[1] = (currSession.dateMonthI % 10) + '0';
mystrcpy(&datetime2[12],mynumber,2,2);
mynumber[0] = (currSession.dateDayI / 10) + '0';
mynumber[1] = (currSession.dateDayI % 10) + '0';
mystrcpy(&datetime2[15],mynumber,2,2);
mynumber[0] = (currSession.dateYearI / 10) + '0';
mynumber[1] = (currSession.dateYearI % 10) + '0';
mystrcpy(&datetime2[18],mynumber,2,2);
mynumber[1] = 'M';
if (currSession.timeHalfI == AM) {
    mynumber[0] = 'A';
} else {
    mynumber[0] = 'P';
}
mystrcpy(&datetime2[8],mynumber,2,2);

for (j = 0; j < 12; j++) {
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = ' ';
}

printcounter = 0;

while (printcounter < 20) {
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = datetime2[printcounter];
    printcounter++;
}

/* Transmit CR and LF */
while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
```

```
    SCI0DRH = 0;
    SCI0DRL = 0x0D;
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = 0x0A;
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = 0x0A;

    /* Print items with price in shopping cart */
    for (i = 0; i < currSession.cartSizeI; i++)
    {
        itemLen = 0;
        priceLen = 0;

        /* Print item */
        while (itemLen < 20){
            while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
            SCI0DRH = 0;
            SCI0DRL = currSession.items[i].itemName[itemLen];
            itemLen++;
        }

        /* Print whitespace between item and price */
        for (j = 0; j < 5; j++) {
            while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
            SCI0DRH = 0;
            SCI0DRL = ' ';
        }

        while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
        SCI0DRH = 0;
        SCI0DRL = '$';

        /* Print price */
        while (priceLen < 3) {
            while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
            SCI0DRH = 0;
            SCI0DRL = currSession.items[i].itemPrice[priceLen];
            priceLen++;
        }

        while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
        SCI0DRH = 0;
        SCI0DRL = '.';

        while (priceLen < 5) {
            while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
            SCI0DRH = 0;
            SCI0DRL = currSession.items[i].itemPrice[priceLen];
            priceLen++;
        }

        /* Print CR and LF */
        while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
        SCI0DRH = 0;
        SCI0DRL = 0x0D;
        while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
        SCI0DRH = 0;
        SCI0DRL = 0x0A;
    }

    printcounter = 0;
```

```
while (printcounter < 32) {
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = DashDash[printcounter];
    printcounter++;
}

/* Print CR and LF */
while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
SCI0DRH = 0;
SCI0DRL = 0x0D;
while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
SCI0DRH = 0;
SCI0DRL = 0x0A;

/* Convert integer cart total to character representation */
temp = (currSession.cartTotalI / 100000) + '0';
if ((temp == '0') && (flag == 0)) {
    myprice[1] = ' ';
} else {
    myprice[1] = temp;
    flag = 1;
}

temp = ((currSession.cartTotalI % 100000) / 10000) + '0';
if ((temp == '0') && (flag == 0)) {
    myprice[2] = ' ';
} else {
    myprice[2] = temp;
    flag = 1;
}

temp = ((currSession.cartTotalI % 10000) / 1000) + '0';
if ((temp == '0') && (flag == 0)) {
    myprice[3] = ' ';
} else {
    myprice[3] = temp;
    flag = 1;
}

temp = ((currSession.cartTotalI % 1000) / 100) + '0';
if ((temp == '0') && (flag == 0)) {
    myprice[4] = ' ';
} else {
    myprice[4] = temp;
    flag = 1;
}

temp = ((currSession.cartTotalI % 100) / 10) + '0';
if ((temp == '0') && (flag == 0)) {
    myprice[6] = ' ';
} else {
    myprice[6] = temp;
    flag = 1;
}

temp = ((currSession.cartTotalI % 10) / 1) + '0';
if ((temp == '0') && (flag == 0)) {
    myprice[7] = ' ';
} else {
    myprice[7] = temp;
    flag = 1;
```

```
    }

    /* Print Total */
    mystrcpy(&Total[24],myprice,8,8);
    printcounter = 0;

    while (printcounter < 32) {
        while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
        SCI0DRH = 0;
        SCI0DRL = Total[printcounter];
        printcounter++;
    }

    /* Print CR and LF */
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = 0x0D;
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = 0x0A;
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = 0x0A;

    /* Print last line of receipt */
    printcounter = 0;
    while (printcounter < 32) {
        while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
        SCI0DRH = 0;
        SCI0DRL = ThankYou[printcounter];
        printcounter++;
    }

    /* Print CR and LF */
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = 0x0D;
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = 0x0A;

    printcounter = 0;
    while (printcounter < 32) {
        while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
        SCI0DRH = 0;
        SCI0DRL = ComeAgain[printcounter];
        printcounter++;
    }

    /* Print CR and LF */
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = 0x0D;
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = 0x0A;
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = 0x0A;
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = 0x0A;
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
```

```
    SCI0DRH = 0;
    SCI0DRL = 0x0A;
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = 0x0A;
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = 0x0A;
    while ((SCI0SR1 & SCI0SR1_TDRE_MASK) == 0);
    SCI0DRH = 0;
    SCI0DRL = 0x0A;
}
```

```
/*****************************************************************************
 *
 *       Copyright (C) 2006 J Team, Inc.
 *       All Rights Reserved
 *
 * File Name      : Vectors.c
 * Project Name   : RFIDXpress.mcp
 * Description    : This file contains the ISR vectors for the project
 *
 * Authors: Jared Suttles
 *          Jennifer Tietz
 *          Jonathan Chen
 *          Joshua Chapman
 *
 * Version : 2.0
 * Date    : 05/1/06
 *
 *****************************************************************************/

#include "Cpu.h"

extern void near _EntryPoint(void);    /* Startup routine */
extern void near RFIDInterrupt(void);
extern void near HandleTimerInterrupt(void);
extern void near IRQInterrupt(void);
extern void near RealTimeInterrupt(void);
extern void near emac_ec_isr(void);
extern void near emac_lc_isr(void);
extern void near emac_b_rx_error_isr(void);
extern void near emac_rx_b_b_o_isr(void);
extern void near emac_rx_b_a_o_isr(void);
extern void near emac_rx_error_isr(void);
extern void near emac_mii_mtc_isr(void);
extern void near emac_rx_fc_isr(void);
extern void near emac_f_tx_c_isr(void);
extern void near emac_rx_b_b_c_isr(void);
extern void near emac_rx_b_a_c_isr(void);
extern void near ephy_isr(void);

typedef void (*near tIsrFunc)(void);

const tIsrFunc _vect[] @0xFF80 = {     /* Interrupt table */
    Cpu_Interrupt,      /* 0 Default (unused) interrupt */
    Cpu_Interrupt,      /* 1 Default (unused) interrupt */
    Cpu_Interrupt,      /* 2 Default (unused) interrupt */
    Cpu_Interrupt,      /* 3 Default (unused) interrupt */
    Cpu_Interrupt,      /* 4 Default (unused) interrupt */
    Cpu_Interrupt,      /* 5 Default (unused) interrupt */
    Cpu_Interrupt,      /* 6 Default (unused) interrupt */
    Cpu_Interrupt,      /* 7 Default (unused) interrupt */
    Cpu_Interrupt,      /* 8 Default (unused) interrupt */
    Cpu_Interrupt,      /* 9 Default (unused) interrupt */
    Cpu_Interrupt,      /* 10 Default (unused) interrupt */
    Cpu_Interrupt,      /* 11 Default (unused) interrupt */
    Cpu_Interrupt,      /* 12 Default (unused) interrupt */
    Cpu_Interrupt,      /* 13 Default (unused) interrupt */
    Cpu_Interrupt,      /* 14 Default (unused) interrupt */
    Cpu_Interrupt,      /* 15 Default (unused) interrupt */
    emac_ec_isr,
    emac_lc_isr,
    emac_b_rx_error_isr,
    emac_rx_b_b_o_isr,
    emac_rx_b_a_o_isr,
```

```
    emac_rx_error_isr,
    emac_mii_mtc_isr,
    emac_rx_fc_isr,
    emac_f_tx_c_isr,
    emac_rx_b_b_c_isr,
    emac_rx_b_a_c_isr,
    ephy_isr,
    Cpu_Interrupt,      /* 28 Default (unused) interrupt */
    Cpu_Interrupt,      /* 29 Default (unused) interrupt */
    Cpu_Interrupt,      /* 30 Default (unused) interrupt */
    Cpu_Interrupt,      /* 31 Default (unused) interrupt */
    Cpu_Interrupt,      /* 32 Default (unused) interrupt */
    Cpu_Interrupt,      /* 33 Default (unused) interrupt */
    Cpu_Interrupt,      /* 34 Default (unused) interrupt */
    Cpu_Interrupt,      /* 35 Default (unused) interrupt */
    Cpu_Interrupt,      /* 36 Default (unused) interrupt */
    Cpu_Interrupt,      /* 37 Default (unused) interrupt */
    Cpu_Interrupt,      /* 38 Default (unused) interrupt */
    Cpu_Interrupt,      /* 39 Default (unused) interrupt */
    Cpu_Interrupt,      /* 40 Default (unused) interrupt */
    Cpu_Interrupt,      /* 41 Default (unused) interrupt */
    RFIDInterrupt,      /* 40 Default (unused) interrupt */
    Cpu_Interrupt,      /* 41 Default (unused) interrupt */
    Cpu_Interrupt,      /* 44 Default (unused) interrupt */
    Cpu_Interrupt,      /* 45 Default (unused) interrupt */
    Cpu_Interrupt,      /* 46 Default (unused) interrupt */
    Cpu_Interrupt,      /* 47 Default (unused) interrupt */
    Cpu_Interrupt,      /* 48 Default (unused) interrupt */
    HandleTimerInterrupt,      /* 49 Default (unused) interrupt */
    Cpu_Interrupt,      /* 50 Default (unused) interrupt */
    Cpu_Interrupt,      /* 51 Default (unused) interrupt */
    Cpu_Interrupt,      /* 52 Default (unused) interrupt */
    Cpu_Interrupt,      /* 53 Default (unused) interrupt */
    Cpu_Interrupt,      /* 54 Default (unused) interrupt */
    Cpu_Interrupt,      /* 55 Default (unused) interrupt */
    RealTimeInterrupt,     /* 56 Default (unused) interrupt */
    Cpu_Interrupt,      /* 57 Default (unused) interrupt */
    Cpu_Interrupt,      /* 58 Default (unused) interrupt */
    Cpu_Interrupt,      /* 59 Default (unused) interrupt */
    Cpu_Interrupt,      /* 60 Default (unused) interrupt */
    Cpu_Interrupt,      /* 61 Default (unused) interrupt */
    Cpu_Interrupt,      /* 62 Default (unused) interrupt */
    _EntryPoint        /* Reset vector */
};
```

```
/*****************************************************************************
 *
 *       Copyright (C) 2006 J Team, Inc.
 *       All Rights Reserved
 *
 * File Name      : address.c
 * Project Name   : RFIDXpress.mcp
 * Description    : This file contains the static IP data for our system
 *
 * Authors: Jared Suttles
 *          Jennifer Tietz
 *          Jonathan Chen
 *          Joshua Chapman
 *
 * Version : 2.0
 * Date    : 05/1/06
 *
 *****************************************************************************/

#include "MOTTYPES.h"

const tU08 hard_addr [6] = { 0x01, 0x23, 0x45, 0x56, 0x78, 0x9a };

const tU08 prot_addr [4] = { 192, 168,   2,   3 };
const tU08 netw_mask [4] = { 255, 255, 255,   0 };
const tU08 dfgw_addr [4] = { 192, 168,   2,   1 };
const tU08 brcs_addr [4] = { 192, 168,   2, 255 };
```

```
/*****************************************************************************
 *
 *       Copyright (C) 2006 J Team, Inc.
 *       All Rights Reserved
 *
 * File Name      : UDP_Client_app.h
 * Project Name   : RFIDXpress.mcp
 * Description    : This file contains the function prototypes for the UDP code.
 *
 * Authors: Jared Suttles
 *          Jennifer Tietz
 *          Jonathan Chen
 *          Joshua Chapman
 *
 * Version : 2.0
 * Date    : 05/1/06
 *
 *****************************************************************************/

/* Prototypes (i.e. Function Definitions) */
void udp_demo_run(void);
void udp_demo_init(void);
```

```
/*****************************************************************************
 *
 *       Copyright (C) 2006 J Team, Inc.
 *       All Rights Reserved
 *
 * File Name      : UDP_Client_app.c
 * Project Name   : RFIDXpress.mcp
 * Description    : This file contains the actual Ethernet transmission and
 *          Receiving functionality for the RFID Xpress system.  The
 *          SendSerial and EventListener methods are the transmission
 *          and receiving methods, respectively.
 *
 * Authors: Jared Suttles
 *          Jennifer Tietz
 *          Jonathan Chen
 *          Joshua Chapman
 *
 * Version : 2.0
 * Date    : 05/1/06
 *
 *****************************************************************************/


#include "debug.h"
#include "datatypes.h"
#include "globalvariables.h"
#include "system.h"
#include "tcp_ip.h"

#include "string.h"
#include "MOTTYPES.h"
#include "IO_Map.h"


/* The applications that use UDP must implement following function stubs         */
/* void application_name_init (void) - call once when processor starts           */
/* void application_name_run (void) - call periodically on main loop             */
/* INT32 application_name_eventlistener (INT8, UINT8, UINT32, UINT16, UINT16,
    UINT16) */
/* - called by TCP input process to inform arriving data, errors etc             */

void udp_demo_run(void);

INT32 udp_demo_eventlistener(INT8 , UINT8 , UINT32 , UINT16 , UINT16 , UINT16 );
INT8 udp_demo_soch;

#define UDP_DEMO_PORT 2001              /* Port to be used on MICRO */
#define UDP_DEMO_RMTHOST_IP   0xC0A80201 /* IP of Server*/
#define UDP_DEMO_RMTHOST_PRT  2000       /* Port of Server */

extern INT8 SendData;           //1 if data needs sent. 0 if not.
extern INT8 RcvData;           //1 if data needs sent. 0 if not.
extern char toSend[15];
extern char rcvPacket[31];




/*
 * Sends the data in the toSend array over the ethernet connection
 */
```

```
UINT16 SendSerial() {
    UINT16  i, datalen;

    if (SendData == 1) {

        for (i = 0; i< 14; i++) {
            net_buf[UDP_APP_OFFSET+i]= toSend[i]; //'U';
        }

        datalen = i+1;

        i = udp_send(udp_demo_soch, UDP_DEMO_RMTHOST_IP, UDP_DEMO_RMTHOST_PRT, net_buf
            + UDP_APP_OFFSET, NETWORK_TX_BUFFER_SIZE-UDP_APP_OFFSET, datalen);

        SendData = 0;
    }

    return 0;
}


/*
 * Initialize resources needed for the UDP socket application
 */

void udp_demo_init(void){

    DEBUGOUT("Initializing UDP demo client\r\n");

    /* Get socket:
     *  0 - for now not type of service implemented in UDP
     *  udp_echo_eventlistener - pointer to listener function
     *  UDP_OPT_SEND_CS|UDP_OPT_CHECK_CS - checksum options. Calculate
     *      checksum for outgoing packets and check checksum for
     *      received packets.
     */
    udp_demo_soch=udp_getsocket(0 , udp_demo_eventlistener , UDP_OPT_SEND_CS |
        UDP_OPT_CHECK_CS);

    if(udp_demo_soch == -1){
        DEBUGOUT("No free UDP sockets!! \r\n");
        RESET_SYSTEM();
    }

    /* open socket for receiving/sending of the data on defined UDP_DEMO_RMTHOST_PRT*/
    (void)udp_open(udp_demo_soch,UDP_DEMO_PORT);

    /* for now no data sending */
    SendData = 0;
}


/* UDP Demo app main loop that is periodically invoked from the
 * main loop (see main_con.c)
 */

void udp_demo_run(void){
    void)SendSerial();
}
/*
 * Event listener invoked when TCP/IP stack receives UDP datagram for
 * a given socket. Parameters:
 * - cbhandle - handle of the socket this packet is intended for. Check it
```

```
 *  just to be sure, but in general case not needed
 * - event - event that is notified. For UDP, only UDP_EVENT_DATA
 * - UDP_DEMO_RMTHOST_IP - IP address of remote host who sent the UDP datagram
 * - UDP_DEMO_RMTHOST_PRT - UDP_DEMO_RMTHOST_PRT number of remote host who sent the
UDP datagram
 * - buffindex - buffer index in RTL8019AS allowing you to read
 *  received data more than once from Ethernet controller by
 *  invoking NETWORK_RECEIVE_INITIALIZE(buffindex) and then start
 *  reading the bytes all over again
 */

extern byte state;

INT32 udp_demo_eventlistener (INT8 cbhandle, UINT8 event, UINT32 ipaddr, UINT16 port,
    UINT16 buffindex, UINT16 datalen)
{
    int i,j,k;
    ipaddr = 0;
    buffindex = 0;
    port = 0;

    if(cbhandle!=udp_demo_soch){
        DEBUGOUT("Not my handle!!!!");
        return (-1);
    }

    switch(event){
        case UDP_EVENT_DATA:

            /* read data to transmit buffer and send response*/
            if(datalen>(NETWORK_TX_BUFFER_SIZE-UDP_APP_OFFSET))
                datalen=NETWORK_TX_BUFFER_SIZE-UDP_APP_OFFSET;

            RECEIVE_NETWORK_BUF(net_buf+UDP_APP_OFFSET,datalen);

            for (i = 0; i <= datalen; i++){
                rcvPacket[i] = net_buf[UDP_APP_OFFSET+i];
            }

            break;

        default:
            /* should never get here */
            DEBUGOUT("Unknown UDP event :-(");
            break;
    }

    if (SendData == 1) {
        RcvData = 1;
    }

    return 0;
}
```

```
/*****************************************************************************
 *
 *       Copyright (C) 2006 J Team, Inc.
 *       All Rights Reserved
 *
 * File Name      : LCD_methods.h
 * Project Name   : RFIDXpress.mcp
 * Description    : This file contains the methods for interfacing with the LCD.
 *
 * Authors: Jared Suttles
 *          Jennifer Tietz
 *          Jonathan Chen
 *          Joshua Chapman
 *
 * Version : 2.0
 * Date    : 05/1/06
 *
 *****************************************************************************/


#define lcdEnable PTG_PTG5
#define lcdRS     PTG_PTG4
#define lcdReset  PTG_PTG6
#define LCD_BUS_0 PTG_PTG0
#define LCD_BUS_1 PTG_PTG1
#define LCD_BUS_2 PTG_PTG2
#define LCD_BUS_3 PTG_PTG3
#define LCD_BUS_4 PTH_PTH2
#define LCD_BUS_5 PTH_PTH1
#define LCD_BUS_6 PTH_PTH0
#define LCD_BUS_7 PTJ_PTJ0
#define NULL      0

void LCD_strobe() {
    lcdEnable = 1;
    lcdEnable = 0;
}

void LCD_reset() {
    lcdReset=0;
    lcdReset=1;
}

void LCD_set_reg(int func, int val) {
    lcdRS=1;
    LCD_BUS_0 = (func & 0x01); // set register
    LCD_BUS_1 = (func & 0x02) >> 1; // set register
    LCD_BUS_2 = (func & 0x04) >> 2; // set register
    LCD_BUS_3 = (func & 0x08) >> 3; // set register
    LCD_BUS_4 = (func & 0x10) >> 4; // set register
    LCD_BUS_5 = (func & 0x20) >> 5; // set register
    LCD_BUS_6 = (func & 0x40) >> 6; // set register
    LCD_BUS_7 = (func & 0x80) >> 7; // set register
    LCD_strobe();

    lcdRS=0;
    LCD_BUS_0 = (val & 0x01); // set register
    LCD_BUS_1 = (val & 0x02) >> 1; // set register
    LCD_BUS_2 = (val & 0x04) >> 2; // set register
    LCD_BUS_3 = (val & 0x08) >> 3; // set register
    LCD_BUS_4 = (val & 0x10) >> 4; // set register
    LCD_BUS_5 = (val & 0x20) >> 5; // set register
    LCD_BUS_6 = (val & 0x40) >> 6; // set register
```

```
    LCD_BUS_7 = (val & 0x80) >> 7; // set register
    LCD_strobe();
}

// 0x32 for Graphic mode Master Display on
void LCD_set_mode(int mode) {LCD_set_reg(0,mode);}

// please use 0x07. set char pitch register
void LCD_char_pitch(int cpitch) {LCD_set_reg(1,cpitch);}

// use 29 for g mode, set no. of char/bytes in row
void LCD_set_nchar(int nchar) {LCD_set_reg(2,nchar);}

// use 0x7F as time division
void LCD_set_time_div(int Nx) {LCD_set_reg(3,Nx);}

// Np = number of pixels the cursor is from top of char
void LCD_set_cursor(int Np) {LCD_set_reg(4,Np);}

// set lower display address. Always set lower first!
void LCD_set_daddl(int daddl) {LCD_set_reg(8,daddl);}

// set upper dispaly address. Always set lower first!
void LCD_set_daddu(int daddu) {LCD_set_reg(9,daddu);}

// set lower cursor address. Always set lower first!
void LCD_set_caddl(int caddl) {LCD_set_reg(10,caddl);}

// set upper cursor address.Always set lower first!
void LCD_set_caddu(int caddu) {LCD_set_reg(11,caddu);}

// Writes data
void LCD_write_data(int data) {LCD_set_reg(12,data);}

void LCD_set_char_add(int row, int col) { // for char mode only
    int abs, lower, upper;
    abs=row*40+col;
    lower=abs%256;
    upper=abs-lower;
    upper=upper >> 8;
    LCD_set_caddl(lower);
    LCD_set_caddu(upper);
}

void LCD_print_string(char *str) {
    while (*str != NULL) {
        LCD_write_data(*str);
        str++;
    }
}

void LCD_putchar(char toput) {
    LCD_write_data(toput);
}

void LCD_graphic_ini() {
    int counter;
    LCD_set_time_div(127);
    LCD_set_mode(50);
    LCD_char_pitch(7);
    LCD_set_nchar(29);
    LCD_set_daddl(0);
    LCD_set_daddu(0);
```

```
    LCD_set_caddl(0);
    LCD_set_caddu(0);
    for (counter = 0; counter < 4096; counter++) {
        LCD_write_data(0x00);
    }
    LCD_set_caddl(0x00);
    LCD_set_caddu(0x00);
}

void LCD_char_ini() {
    int counter;
    LCD_set_time_div(127);
    LCD_set_cursor(167);
    LCD_char_pitch(165);
    LCD_set_nchar(38);
    LCD_set_mode(48);
    LCD_set_daddl(0);
    LCD_set_daddu(0);
    LCD_set_caddl(0);
    LCD_set_caddu(0);
    for (counter = 0; counter < 4096; counter++) {
        LCD_write_data(0x00);
    }
    LCD_set_caddl(0);
    LCD_set_caddu(0);
}

void LCD_clr_scr_char(int screen) {
    int row=screen*12;
    int i;
    LCD_set_char_add(row,0);
    for (i=0; i<12; i++) {
        LCD_print_string("                                          ");
    }
}
```

Java Server Code

```
/*****************************************************************************
 *
 *       Copyright (C) 2006 J Team, Inc.
 *       All Rights Reserved
 *
 * File Name      : RFIDudpServer1.java
 * Project Name   : UDPserv
 * Description    : This file contains the main program that instantiates a
 *                  thread of the RFIDudpServer1Thread class.
 *
 * Authors: Jared Suttles
 *          Jennifer Tietz
 *          Jonathan Chen
 *          Joshua Chapman
 *
 * Version : 1.2
 * Date    : 05/01/06
 *
 *****************************************************************************/

import java.io.*;

public class RFIDudpServer1 {
    public static void main(String[] args) throws IOException {
        new RFIDudpServer1Thread().start();
    }
}
```

```
/*****************************************************************************
 *
 *       Copyright (C) 2006 J Team, Inc.
 *       All Rights Reserved
 *
 * File Name      : RFIDudpServer1Thread.java
 * Project Name   : UDPserv
 * Description    : This code was adapted from SunMicrosystem's example code.
 *                  Datagram packets recceived by this code are in the following
 *                  format:
 *
 *                  CMD:SERIALNUMBER
 *                  where CMD is:
 *                  U => Get User Data
 *                  I => Get Item Data, Add to Shopping Cart
 *                  R => Remove Item from Shopping Cart
 *                  M => End + E-Mail
 *
 * Authors: Jared Suttles
 *          Jennifer Tietz
 *          Jonathan Chen
 *          Joshua Chapman
 *
 * Version : 1.2
 * Date    : 05/01/06
 *
 *****************************************************************************/

import java.io.*;
import java.net.*;
import java.text.DateFormat;
import java.text.NumberFormat;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.HashMap;
import java.util.Locale;

public class RFIDudpServer1Thread extends Thread {

    public boolean mailedFlag = true;
    protected DatagramSocket socket = null;
    protected BufferedReader in = null;
    protected HashMap ShoppingCart = null;

    public RFIDudpServer1Thread() throws IOException {
        this("RFIDudpServer1Thread");
    }

    public RFIDudpServer1Thread(String name) throws IOException {
        super(name);
        int i = 0;
        i = 2000;
        boolean flag = true;
        while (flag) {
            try
            {
                socket = new DatagramSocket(i);
                flag = false;
            } catch (SocketException e) {
                System.out.println("Got an error on " + i);
            }
        }
        System.out.println("Got Socket " + i);
```

```
        ShoppingCart = new HashMap();

    }

    public void run() {
        while (true) {
            try {
                byte[] buf = new byte[14];

                // receive request
                DatagramPacket packet = new DatagramPacket(buf, buf.length);
                socket.receive(packet);

                // figure out response
                String FileLine = null;
                String GotString = new String(packet.getData());
                System.out.println("GOT PACKET WITH: " + GotString + " As its data");
                FileLine = getData(GotString);
                buf = FileLine.getBytes();

                // send the response to the client at "address" and "port"
                InetAddress address = packet.getAddress();
                int port = packet.getPort();
                System.out.println("To port " + port);
                packet = new DatagramPacket(buf, buf.length, address, port);
                socket.send(packet);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }


    // Right now it will just get the next line of the file and return it.. later,
    // it should find the corresponding Serial number.
    protected String getData(String FromPacket) {
        String[] CmdSerial = null;
        String[] LineArray = null;
            String returnValue = null;
        String checkSerial = null;
        String SerialCMP = null;
        char CMD = 'V';
        boolean found = false;

        System.out.println("\nReceived Packet: "+FromPacket);

        try {
            // This should open the file and then read a line from it.
            in = new BufferedReader(new FileReader("RFIDXpr3ss.txt"));
        } catch (Exception e) {
            System.err.println("Could not open RFID Xpr3ss DB");
            System.exit(0);
        }
        //Split the packet input string to CMD and Serial#
        CmdSerial = FromPacket.split(":");
        System.out.println("After Split: " + CmdSerial[0]+ " - " + CmdSerial[1]);
        //Get the Serial number from the packet
        SerialCMP = CmdSerial[1];
        //Get the Command from the packet
        CMD = CmdSerial[0].charAt(0);
```

```
        try {
            // Set the file pointer
            if ((CMD == 'U') || (CMD == 'M')) {
                while ((checkSerial = in.readLine()).compareTo("//USERS") != 0);
            } else if ((CMD == 'I') || (CMD == 'R')) {
                while ((checkSerial = in.readLine()).compareTo("//ITEMS") != 0);
            } else if (CMD == 'T'){
                System.out.println("Received Time Synch Request");
                GregorianCalendar myDate = new GregorianCalendar();
                Date d = myDate.getTime();
                DateFormat df1 = DateFormat.getDateTimeInstance(DateFormat.SHORT,
                    DateFormat.MEDIUM);
                returnValue = df1.format(d);
                System.out.println("Returning: " + returnValue);
                found = true;
            }

            while(found == false)
            {
                if ((checkSerial = in.readLine()) == null)
                {
                    returnValue = "~ERROR";
                    System.out.println("Invalid Serial Number");
                    break;
                } else if ((checkSerial.charAt(0) == '/') && (checkSerial.charAt(1) ==
                            '/'))
                {
                    returnValue = "~ERROR";
                    System.out.println("Invalid Serial Number");
                    break;
                }
                //System.out.println("FROM File GOT:" + checkSerial);
                LineArray = checkSerial.split(",");

                if (SerialCMP.compareTo(LineArray[0]) == 0)
                {
                    if (CMD == 'I'){
                        returnValue = LineArray[1]+LineArray[2];
                        System.out.println("Returning: " + returnValue);
                        AddToCart(LineArray);
                        found = true;
                    }
                    else if (CMD == 'R'){
                        returnValue = "ACK";
                        System.out.println("Returning: " + returnValue);
                        RmFromCart(LineArray);
                        found = true;
                    }
                    else if(CMD == 'U') {
                        returnValue = LineArray[3]+LineArray[2]+LineArray[1];
                        System.out.println("Returning User Info:" + returnValue);
                        mailedFlag = false;
                        found = true;
                    }
                    else if(CMD == 'M'){
                        returnValue = "SENT";
                        sendMail("smtp.purdue.edu", LineArray[2], LineArray[3]);
                        System.out.println("Returning ACK:" + returnValue);
                        found = true;
                    }
                    else {
                        returnValue = "~ERROR";
                        System.out.println("Returning ERR no packet");
```

```
                    break;
                }
            }
        }
    } catch (IOException e) {
        returnValue = "IOException occurred in server.";
    }
    return returnValue;
}

public void sendMail(String mailServer, String recipient, String username) {
    if (!mailedFlag) {
        mailedFlag = true;
        try {
            Socket s = new Socket(mailServer, 587);
            BufferedReader in = new BufferedReader
                (new InputStreamReader(s.getInputStream(), "8859_1"));
            BufferedWriter out = new BufferedWriter
                (new OutputStreamWriter(s.getOutputStream(), "8859_1"));
            String messageBody = MakeReceiptBody(username);
            send(in, out, "EHLO rfidXpr3ss");
            send(in, out, "AUTH LOGIN");
            send(in, out, "anN1dHRsZXM=");
            send(in, out, "amp0dXR0bGVzMTIxNw==");
            send(in, out, "MAIL FROM: <rfid_Xpr3ss@purdue.edu>");
            send(in, out, "RCPT TO: " + recipient);
            send(in, out, "DATA");
            send(out, "Subject: Receipt of Purchase");
            send(out, "From: rfid Xpr3ss <rfid_Xpr3ss@purdue.edu>");
            send(out, "\n");
            // message body
            send(out, messageBody);
            //send(out, "Thanks for your purchase\n");
            //send(out, "Hey, look it works!");
            send(out, "\n.\n");
            send(in, out, "QUIT");
            s.close();
            ShoppingCart = new HashMap();

        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}

private String MakeReceiptBody(String username) {
    String messageBody = "";
    String myuser;
    float Total = 0;
    ItemStruct temp;
    int i = 19;

    // Get time and date for email
    GregorianCalendar myDate = new GregorianCalendar();
    Date d = myDate.getTime();
    DateFormat df1 = DateFormat.getDateTimeInstance(DateFormat.MEDIUM,
        DateFormat.MEDIUM);
    String s1 = df1.format(d);
    messageBody += s1 + "\n\n";

    while (username.charAt(i) == ' ') i--;
    myuser = username.substring(0,i+1);
```

```
        // Get user name
        messageBody += myuser + ",\n\nPlease find your receipt below, courtesy of RFID
            Xpr3ss.\n\n\n";

        // Initialize message body
        messageBody += "ITEM                            PRICE\n------
                       -------\n";

        // Get array of keys from hash map
        Object keys[] = ShoppingCart.keySet().toArray();

        for (int Index = 0; Index < keys.length; Index++) {
            // Get current item from hash map
            temp = (ItemStruct) ShoppingCart.get((String)keys[Index]);

            // Add item and price to message body
            messageBody += temp.Name + "    " + GetCurrencyString(temp.Price,7) + "\n";

            // Add item price to total
            Total += temp.Price;
        }
        // Append Total
        messageBody += "                              -------\n" +
                       "                     TOTAL:  " + GetCurrencyString(Total,8) +
                       "\n";

        // Conclude message body
        messageBody += "\nThank you for using RFID Xpr3ss!\n";

        return messageBody;
    }

    private String GetCurrencyString(float total, int fieldwidth) {
        Locale locale = Locale.US;
        String currency = NumberFormat.getCurrencyInstance(locale).format(total);
        while(currency.length() < fieldwidth){
            currency = currency.substring(0,1)+" "+currency.substring(1);
        }
        return currency;
    }

    public void send(BufferedReader in, BufferedWriter out, String s) {
        try {
            out.write(s + "\n");
            out.flush();
            System.out.println(s);
            s = in.readLine();
            //System.out.println(s);
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void send(BufferedWriter out, String s) {
        try {
            out.write(s + "\n");
            out.flush();
            System.out.println(s);
        }
        catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }

    private void AddToCart(String LineArray[]) {
        ItemStruct temp = new ItemStruct(LineArray[1],getFloatVal(LineArray[2]));
        ShoppingCart.put(LineArray[0],temp);
    }

    private void RmFromCart(String LineArray[]) {
        ShoppingCart.remove(LineArray[0]);
    }

    private float getFloatVal(String str) {
        float val = 0;
        for (int strIndex = str.length()-1; strIndex >= 0; strIndex--) {
            if (str.charAt(strIndex) <= '9' && str.charAt(strIndex) >= '0') {
                val += (str.charAt(strIndex) - '0') * Math.pow(10,(str.length()-
                    strIndex-3));
            }
        }
        return val;
    }
}
```

```
/*****************************************************************************
 *
 *       Copyright (C) 2006 J Team, Inc.
 *       All Rights Reserved
 *
 * File Name      : ItemStruct.java
 * Project Name   : UDPserv
 * Description    : This file contains the structure defining an item name
 *                  and price.
 *
 * Authors: Jared Suttles
 *          Jennifer Tietz
 *          Jonathan Chen
 *          Joshua Chapman
 *
 * Version : 1.2
 * Date    : 05/01/06
 *
 *****************************************************************************/

public class ItemStruct {
    public String Name;
    public float Price;

    public ItemStruct(String Name, float Price) {
        this.Name = Name;
        this.Price = Price;
    }
}
```

Example of the database file

```
//ITEMS
0413AADF243A,Toilet Paper 6pk         , 1630
0413AAC71B37,Hamburger                ,  129
0413AACDC145,Captain Crunch           ,  634
0413AAD7422B,Ritz Crackers            ,  315
0413AAC34F3A,The Fast and Furious DVD , 1570
0413AADB1132,Digital Systems Book     ,34999
0413AAB7D63D,GI JOE Action Figure     , 1562
0413AAD3EB48,3-Ring Binder            ,  275
0413AABD923B,Counter-Strike           , 1995
0413AADBB446,12-pk Diet Coke Bacon    ,  499
//USERS
0413A08D4821,5233,jjchen@purdue.edu             ,Jonathan Chen
04118A1C8B2D,9839,jdchapma@purdue.edu           ,Josh Chapman
041112BE8D2C,1464,jtietz@purdue.edu             ,Jennifer Tietz
```

Appendix G: FMECA Worksheet

Failure No.	Failure Mode	Possible Causes	Failure Effects	Method of Detection	Criticality	Remarks
A1	Output = 0 V	Failure of any component in A or an external short to ground	LCD ceases to function, short could potentially cause overheating	Observation	High	Customer could potentially be injured if component overheating leads to fire
A2	Output > 5.0 V	Failure of U11	Potential damage to LCD	Observation	Low	Regulator should have automatic shutdown if voltage or current get too high
A3	Output out of tolerance	Failure of C23, C24, U11	Out-of-spec operating voltage; unpredictable	Observation	Low	
B1	Output = 0 V	Failure of any component in B or an external short to ground	µC and RS232 line driver cease to function, potential overheating	Observation	High	Customer could potentially be injured if component overheating leads to fire
B2	Output > 3.3 V	Failure of U12	Potential damage to µC and/or RS232 line driver	Observation	Low	Regulator should have automatic shutdown if voltage or current get too high
B3	Output out of tolerance	Failure of C25, C26, U12	Out-of-spec operating voltage; unpredictable	Observation	Low	
C1	Output = 0 V	Failure of any component in C or an external short to ground	RFID reader ceases to function, short could potentially cause overheating	Observation	High	Customer could potentially be injured if component overheating leads to fire

C2	Output > 12 V	Failure of U10	Potential damage to RFID reader	Observation	Low	Regulator should have automatic shutdown if voltage or current get too high
C3	Output out of tolerance	Failure of L1, L2, D1, R8, R9, R12, R15, C27, C28, C29, C30, U10	Out-of-spec operating voltage; unpredictable	Observation	Low	
D1	Output continuously 0	U1, Y1, C1, C7, C8, C9, C10, C11, C13, C16, C17, C18, R10, R13, R14, reset circuitry, software	LCD, printer may display erratic output, keypad, Ethernet may not function properly	Observation	Low	
D2	Output continuously 1	U1, Y1, C1, C7, C8, C9, C10, C11, C13, C16, C17, C18, R10, R13, R14, software	LCD, printer may display erratic output, keypad, Ethernet may not function properly	Observation	Low	
D3	User information not validated correctly	Failure of U1, software	Incorrect PIN accepted, wrong user authenticated	Observation	High	Could compromise user's identity, allow purchases on others' accounts
E1	Ethernet doesn't work	Failure of any component in E, U1, software	Email, database lookups, time synchronization unsuccessful	Observation	Low	ACTLED and LNKLED provide Ethernet status notification
F1	Receipt fails to print	Failure of U14, J3, U1, software	Receipt printing unsuccessful, default to email receipt	Observation	Low	Could also be caused by failure of printer itself

F2	RFID reader fails to detect tags	Failure of U14, J2, U1, software	RFID tag not acknowledged, unable to scan items or user key fobs	Observation	Low	Could also be caused by failure of RFID reader itself
G1	LCD display functioning improperly	Failure of U1, J5, C12, software	LCD displays incorrect or no information, unable to view cart or status	Observation	Low	Could also be caused by failure of LCD or inverter module itself
G2	LCD backlight too bright or off	Failure of J5	LCD backlight failing, possible problem with inverter	Observation	High	Customer could potentially be injured if inverter malfunction leads to fire
G3	Keypad entries not detected	Failure of U1	Key presses not recognized, unable to enter PIN or make menu selections	Observation	Low	Could also be caused by failure of keypad itself