UNIVERSITY OF TORONTO

INSTITUTE FOR AEROSPACE STUDIES

4925 Dufferin Street, Toronto, Ontario, Canada, M3H 5T6

FINAL REPORT

THE MULTIPURPOSE STORAGE SYSTEM [1]

AER201: Engineering Design
Professor M. R. Emami

Team 40	Duluxan Sritharan	996272901
PRA0101	Fangzhou Su	996225106
TA: Damien Frost	David Wang	996218285

DIVISION OF ENGINEERING SCIENCE
UNIVERSITY OF TORONTO

April 14, 2009

FINAL REPORT

THE MULTIPURPOSE STORAGE SYSTEM [1]

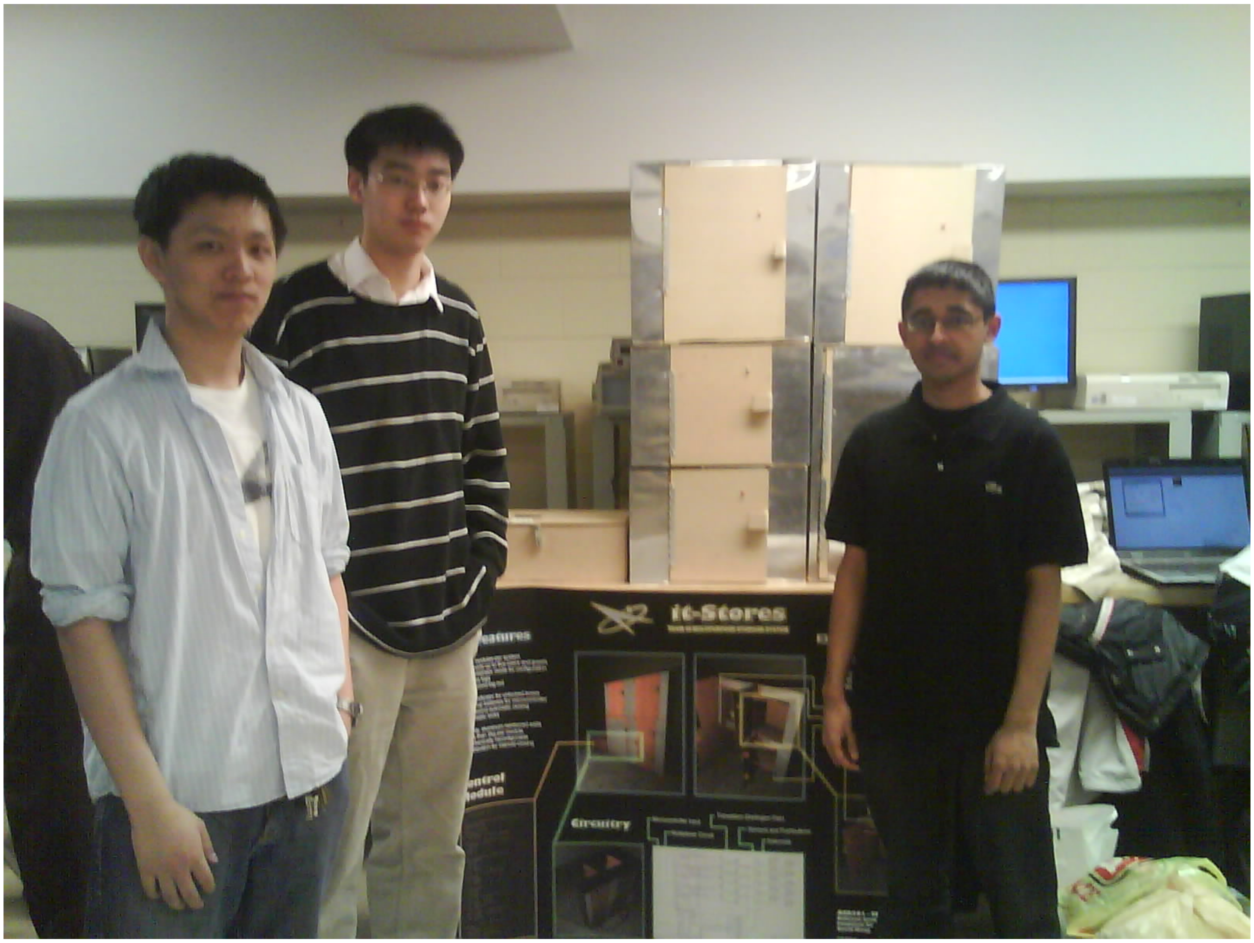
AER201: Engineering Design
Professor M. R. Emami

Team 40	Duluxan Sritharan	996272901
PRA0101	Fangzhou Su	996225106
TA: Damien Frost	David Wang	996218285

DIVISION OF ENGINEERING SCIENCE
UNIVERSITY OF TORONTO

April 14, 2009

TEAM/PROJECT PHOTO



Members of Team 40: Fangzhou Su, David Wang and Duluxan Sritharan (from left to right).

ACKNOWLEDGEMENTS

We would like to acknowledge the contributions of several groups and individuals that helped make this project a reality.

First and foremost, we would like to thank Professor Emami for his invaluable guidance, both in terms of this project and engineering design in general. He has spent countless hours preparing this course, organizing the laboratory, preparing teaching material and assisting groups in every phase of this project from conception to final implementation.

Secondly, we would like to thank our TA, Damien Frost, for his constructive criticism and feedback at every step of this process. He kept us focused and made sure we thought of the big picture. His technical expertise was also very helpful.

We would also like to thank our various suppliers for providing suggestions about which materials would be most appropriate. In particular, we would like to thank Lawrence (Hak-Wa) Chan from Creatron for his input.

Finally, we would like to acknowledge our colleagues in AER201 for many inspirations and ideas, and providing input and positive suggestions. Our final project was largely shaped by the mutual collaboration of our many colleagues.

ABSTRACT

This report is a compilation of the design process undertaken to complete a beta prototype of an autonomous storage system. Existing self-storage systems are difficult to customize to the personal needs of a user. Most existing solutions tend to be very homogenous in terms of the amount of storage space they provide, are intended for use in a fixed configuration and due to their non-automated nature, can be difficult to administer.

The proposed solution is an electronic modular storage system that allows compartments of three different sizes to be assembled in random configurations. The system is operated by a microcontroller, which confers advantages over traditional manual systems including flexibility in module assignment, and improved security in terms of tracking system activity.

All members were responsible for the conception and final integration of the prototype, but Duluxan Sritharan was responsible for software development and the user interface, Fangzhou Su designed circuits for the actuators and sensors, and David Wang was responsible for the construction of the storage modules.

The development cost for the prototype, including experimentation and spare parts was $1045 while the final material cost of the prototype itself is $194.76. These funds were raised wholly by the team members.

The beta prototype meets all constraints and fares well in the criteria, providing valid evidence that the proposed solution would be effective in a range of consumer and industrial applications.

TABLE OF CONTENTS

REPORT PREAMBLE

TECHNICAL BODY

APPENDICES AND SUPPLEMENTARY INFORMATION

NOTATION

Symbol	**Designation**
I	Moment of Inertia (g*cm^2)
r	Radius of rotation (cm)
dm	Differential mass (g)
ρ	Density (g/cm^3)
t	Thickness (cm)
h	Height (cm)
w	Width (cm)
τ	Torque (N*m)
α	Angular acceleration (rad/s^2)
F	Net force (N)
P	Power (W)
V	Voltage (V)
R	Resistance (Ω)
I	Current (A)

ABBREVIATIONS

Abbreviation	Designation
AC	Alternating Current
AS/RS	Automatic Storage and Retrieval System
BCD	Binary-Coded Decimal
CCT	Circuit
CDN	Canadian
DC	Direct Current
D-Sub	D-Subminiature
EEPROM	Electrically Erasable Programmable Read-Only Memory
EM	Electromechanical
I2C	Inter-Integrated Circuit
IC	Integrated Circuit
LCD	Liquid Crystal Display
LED	Light Emitting Diode
MC	Microcontroller
PC	Personal Computer
PCB	Printed Circuit Board
PIC	Peripheral Interface Controller
RAM	Random Access Memory
RFP	Request for Proposal
ROM	Read-Only Memory
RTC	Real-Time Clock
US	United States
USART	Universal Synchronous/Asynchronous Receiver/Transmitter

1. INTRODUCTION

The self-storage industry is booming in North America, with annual sales in excess of $20 billion US [2]. In fact, the average US household rents 20 square feet of storage space [2], in addition to free storage in the form of mailboxes, lockers, and garage organizers. Given the prevalence of self-storage, and the diverse array of uses that exist for it, there is a clear demand for storage modules that are secure, easy to use, and specialized according to the nature of the user's need. Any such functional system would be a boon for consumers and industry alike.

Current systems tend to be very homogeneous in terms of module size which makes it difficult to customize systems to the needs of specific individuals. In addition, most systems tend to have fixed configurations, which limits their versatility in handling a variety of users with different needs over a long period of time. Considering that different modules inevitably tend to be re-assigned to different users over time, an electronic system would be advantageous because they would provide a means of maintaining a record of system activity.

Current patents and products that would serve appropriately in required context tend to be highly specialized for particular industries, which is contrary to the multi-purpose nature of the desired system.

2. BACKGROUND INFORMATION AND PERSPECTIVE

2.1 IDEA SURVEY

Existing ideas for both individual components and the entire system were surveyed for relevance to the RFP. For the door opening/closing mechanism, motor-driven devices, hydraulic arms, electronic doors such as those used for wheelchair accessible doors, obstructive devices such as door stops, and even pulling arms such as those on pedal-operated trash cans were considered. Relevant locking mechanisms include deadbolts, physical jigs and magnetic adhesion. Existing solutions for maintaining the rigidity of assembled systems include both material solutions (e.g Velcro), structural forms (e.g. jigsaw molds), and physical restraints (e.g. cord bike locks). Inspiration for modularity was drawn from IKEA furniture, which is always designed to work both *in situ* and as a part of a larger system.

Mailbox rooms in apartment complexes are very similar to the required product, albeit for the lack of automation. Mailboxes of different sizes can be inserted and locked in place from a back room by the superintendent, while users can only access their assigned compartments from the front. Locker rooms in swimming pools also have similar operational characteristics since they are comprised of lockers of different sizes. Here, instead of an automated interface, the user signs in with the clerk, who gives him/her a key to access a certain module. These systems are relevant since the product outlined in this proposal is intended to be used in settings like these.

2.2 MARKET SURVEY

Automated, modular storage systems are not readily available for consumer use. While electronic safes are prevalent in both residential and commercial applications, they are unsuitable for this particular problem, because they are designed to be standalone devices. Low-cost modular storage systems such as tool organizers or communal mailboxes still rely on keys, combination locks or padlocks, requiring the user to manually operate each storage compartment. Warehouse storage lockers do exist but are intended mostly for industrial applications with capacities exceeding levels appropriate for everyday use in applications like mailboxes. One product that is similar to the one required is the Hanel Multi-Space produced by Industore, which has “variable container widths, different payload capacities” and the ability to add modules later as required [3]. However, this product not only stores items but also transports them out of reach to minimize floor space, which is not desired in this case.

2.3 LITERATURE SURVEY

There is little research or documentation available. Most scientific articles pertain to AS/RS (automatic storage and retrieval systems) that are used in manufacturing to transport items to different levels of a factory. The U.S. Patent Office has a patent (20080208389) for “an automated storage system comprising: a) a plurality of storage locations; b) at least one access location; c) at least one storage container provided on at least one storage location; d) a control system and at least one user interface, the control system further comprising a retrieval mode and a storage mode” [4]. While the overall system configuration is similar to what is required by the RFP, it is intended for loading/unloading modules directly onto vehicles.

2.4 LIMITATIONS

The main limitation on any design solution is striking a balance between various criteria. The system must be very secure yet light and portable, modular and reconfigurable yet difficult to take apart and of course satisfy strict cost constraints. It is easy to maximize reliability, security, ease of use or robustness individually but striking a feasible and effective balance of these factors is much for difficult. For example, a safe is a very special subset of possible solutions which maximizes only the security factor, but already violates constraints since reasonably secure safes cost upwards of $2000.

3. OBJECTIVES, CONSTRAINTS AND DESIGN PARAMETERS

The objective is to design a proof-of-concept prototype of an automated storage system consisting of different-sized storage modules by manufacturing five storage modules successfully, such that the configurable, modular, and automated nature of the device is illustrated. The functionality of this system includes set-up and interaction through an LCD/keypad interface. More specifically, the prototype must achieve several physical, functional and security objectives.

The prototype must contain 2 small modules, 2 medium modules and 1 large module with nominal interior dimensions (H mm x W mm x D mm) of 200x250x200, 350x300x200 and 500x400x200 respectively within a 10mm tolerance for each dimension. The outer dimensions for all the modules must not exceed 700mm (H) x 600mm (W) x 400mm (D). Each module must not weigh more 2 kg, and must be able to support another 1 kg. The modules should be designed such that they can be easily and quickly configured and assembled. The system must be powered by a standard AC 110 V-60 Hz- 3 pin outlet. In the case of a power outage, a back-up rechargeable DC power supply must ensure uninterrupted operation. Lastly, the total material cost for the prototype must not exceed $200 CDN before taxes.

The system must be controlled by an on-board processor, which facilitates opening/closing and locking/unlocking the storage modules. The system should be able to automatically close module doors 15 seconds after opening, although the user should be able to close the door before this period, or keep it open after, if he/she desires. Once closed, the door must lock within 3 seconds. The operator must be able to interact with the system using an LCD/keypad interface that allows various functionalities to be performed according to the operator’s status as administrator, user or guest, including gaining access to specific storage modules.

In particular, the administrator must be able to regulate all accounts by setting validity periods, monitoring system activity via the weekly logs and having the capability to access all modules. It should be difficult to violate the integrity of the system from the front faces of the modules (i.e. disassemble the modules, tamper with electronic components or gain unauthorized entry).

4. ACCEPTANCE CRITERIA IN DECISION MAKING

All decisions regarding the proposal including specific subsystem decisions were made as a team. When the decision influenced all subsystems, we aimed for consensus but in the case of a majority without consensus, we expected constructive participation on a going-forward basis from the third member. In decisions that were particularly pertinent to a specific subsystem, or when a certain member had more expertise, the other members contributed feedback but ultimately deferred to the judgment of the expert member. In cases where all three members had different opinions or there wasn't enough immediate information to make a decision, it was expected that each member would perform individual research, so that a decision could be made by the next meeting scheduled within 2 days. In all considerations, the value of practicality was emphasized and simple designs that could be implemented easily were always preferred over more impressive but complicated designs.

There was no one set of criteria used for decision-making, but rather criteria were derived for each component based on the most relevant set of parameters. Candidate solutions were then evaluated according to these criteria and the best solution was chosen. Included below is a set of major design considerations followed by list of ranked criteria (most to least important) used to inform decisions.

Table 1: Acceptance Criteria for Design Considerations

Design Consideration	Ranked Criteria
Exterior Module Dimensions	Configurability, Material Cost, Bulkiness
Module Design	Strength, Simplicity, Cost
Module Attachment Mechanism	Manufacturability, Configurability
Locking Mechanism	Manufacturability, Complexity
Door Mechanism	Cost, Manufacturability, Complexity, Power Consumption
Solenoid Signal Circuit	Complexity, Implementation Time
Solenoid Power Circuit	Cost, Safety, Complexity
Voltage for Door Jamming Solenoids	Safety, Strength
Voltage for Locking Solenoids	Safety, Strength, Complexity
Operator's Door Closing Mechanism	Security, Functionality, Complexity
Choice of Microcontroller Board	Cost, Timeline, Functionality, Usability

5. BUDGET AND FINANCES

Item	Quantity	Unit Cost	Total Cost
Electromechanical Components			
Pushing Solenoid (Door Jam)	5	$2.99	$14.95
Pulling Solenoid (Lock)	5	$2.49	$12.45
Hardboard	2 – 24” x 48” sheets	$2.48	$4.96
Aluminum Sheet	20 sq. ft	$0.80/sq. ft	$16.00
Hinges	6	$0.49	$2.97
Microswitches	5	$1.49	$7.45
Pushbuttons	5	$0.99	$4.95
Retractor Device	5	$0.99	$4.95
Velcro	5 cm x 2 cm	-	$0.99
Rivets	20	$6.00/100	$1.20
Glue	3 sticks	$6.00/12 sticks	$1.50
Magnets	10	$0.50	$5.00
Steel Tabs	22	$0.30	$6.60
		Electromechanical Total:	*$83.97*
Circuit Components			
Power Supply (salvage from computer)	1	$5.00	$5.00
PCB Board	1	$4.50	$4.50
Rechargeable Battery	8	$1.25	$10.00
Signal Transistors	10	$1.00	$10.00
Logic Gate Chips	3	$0.50	$1.50
Circuit Diodes	10	$0.10	$1.00
Power Supply Diodes	2	$0.50	$1.00
Wires and Cables	-	-	$10.00
D-Sub Connectors	10	$0.49	$4.90
		Circuit Total:	*$47.90*
Microcontroller Components			
PIC DevBugger Board	1	$50.00	$50.00
Real-Time Clock Chip	1	$5.00	$5.00
3 V Coin Batter	1	$1.89	$1.89
Keypad/LCD	-	-	$6.00
		Microcontroller Total:	*$62.89*
TOTAL			**$194.76**

The prototype cost is less than the constraint of $200. Development costs for the prototype are indicated on a task-by-task basis on the GANTT Charts (Appendix E), and total $1045. Costs of specialty parts were derived by contacting suppliers (see Appendix C).

6. DIVISION OF PROBLEM

The team consists of three members who are responsible for both administrative, conceptual and implementation tasks. Conceptual and administrative tasks must be performed as a group because they require a united vision of all members to ensure success. Implementation tasks were subdivided into three components, each spearheaded by a team member. For these subsystems, members were still expected to act as the resource person in their field of expertise after the timeline expired, but the focus was more on integration.

Table 2: Timeline and division of tasks for project.

Task Category	Members	General Description	Timeline
Administrative	All	All members will attend meetings, plan schedules, engage in correspondence with the customer, help in the preparation of deliverables and procure supplies	Jan. 7 – Apr. 14
Conceptual	All	All members will contribute ideas and feedback regarding both the prototype and the implementation plan.	Jan. 7 – Jan. 25
Electromechanical (EM)	David Wang	Design, analysis, fabrication, assembly and integration of storage modules, and actuation mechanisms	Jan. 9 – Mar. 11
Circuits (CCT)	Fangzhou Su	Acquisition of power supplies and construction of solenoid driver circuits and power circuits. At the end of this subset timeline, circuit member will assist the Microcontroller member in completing and testing code.	Jan. 22 – Mar 2
Microcontroller (MC)	Duluxan Sritharan	Design of program algorithm and development of all software for user interface, equipment interface, and data storage and retrieval	Jan 7 – Mar 3
Integration	All	Integrating all subsystems into a single unit, testing for functionality and debugging any issues that may arise.	Mar 4 – Apr 8

A complete list of tasks by subsystem is included in Appendix J.

7. ELECTROMECHANICAL SUBSYSTEM

7.1 ELECTROMECHANICAL OVERVIEW

The electromechanical system includes the solenoid set-up and module construction for all 5 modules (2 small, 2 medium, 1 large). Each module must weigh less than 2 kg, lock/unlock automatically, close automatically, hold the door open for at least 15 seconds, be impenetrable from the front and the sides, and can be attached and detached quickly in different configurations.

7.2 ASSESSMENT OF PROBLEM

Several mechanisms need to created that are easily replicable including locking and closing systems. A method of construction needs to chosen that allows similar fabrication for all modules and allow all modules to be configurable as part of a larger system. It is necessary to consider competing factors such as cost, robustness and weight in designing the modules. The main problems identified are:

- determining optimal dimensions and materials for storage modules
- design of a central hub for circuits and microcontroller
- creation of a robust locking mechanism
- creation of a reliable jamming mechanism
- creation of a reliable closing mechanism

7.3 DIMENSIONS AND MATERIALS OF STORAGE MODULES

7.3.1 Analysis of Problem

The inner dimensions of the small, medium and large modules (H mm x W mm x D mm) must be 200x250x200, 350x300x200 and 500x400x200 respectively within a 10mm tolerance for each dimension. The outer dimensions must be no bigger than (H mm x W mm x D mm) 700x600x400. In addition to the 2kg/module weight limit, the modules must be structurally sound and support 1 kg without failing. In addition, the modules must also be modular in order to accommodate various configurations.

7.3.1 Solution

The outer dimensions of the small, medium and large modules (H mm x W mm x D mm) are 300x450x225, 450x450x225 and 600x450x225 respectively, with a 5mm tolerance for each dimension (see Table E.1 for a full set of dimensions and attributes).

These outer dimensions guarantee that the modules can be assembled in various configurations without having unsightly gaps that reduce the modularity of the system. See Figure E.1 for plan and front views of the modules in different configurations.

One casing was constructed for the inner compartment and another for the outer compartment of each module. On the front face, a plate was fashioned so as to hide the space between the two compartments from the operator. The door for each module is a typical vertical hinge design, with the door having the same width as the inner compartment and the same height as the outer compartment. See Figure E.2 ad E.3 for drawings of the small and medium modules.

The outer walls and three of the four inner walls for the modules are constructed out of cardboard, because it is cheap, lightweight, structurally sound, and allows the concept to be effectively conveyed. The cardboard is paneled with aluminum to give the impression of sturdiness and provide extra rigidity. The door and the wall attached to the door hinge are made of hardboard panels, a very thin, but sturdy type of plywood. The walls are connected using wood glue. Hardboard allows the prototype to withstand both the 1 kg load, and the loads from supporting other storage modules, without being overweight. The weight of the large module was designed to be lighter than the medium module due to the more prevalent usage of aluminum. However, as aluminum is nearly 3 times as expensive as hardboard, this scheme was only applied for the large module. See Table E.2 for a list of weights for each component and refer to Table E.3 for the weight breakdown of each module.

The modules also have metal L-brackets protruding from the rear to aid in modular attachment. For a fixed configuration, bolts can be placed between the L-brackets to firmly affix the modules to each other. Since all modules have the same depth, this allows the modules to be attached together easily. However, for a more arbitrary configuration, a cord can be run through the holes, linking all of the modules together. This cord is then locked down to provide security.

7.4 CONTROL MODULE

7.4.1 Analysis of Problem

A hub must be created to house the circuitry and the microcontroller. Only the LCD display and the keypad must be accessible to the user; all other components are to be hidden inside the module. Cables connecting the modules to the hub must be inaccessible to the user.

7.4.2 Solution

The control module measures (H mm x W mm x D mm) 150x450x225 (see Figure E.4 for a complete hub design with dimensions). The control module has the same width as the other modules allowing for optimum configurability. The control module also has the capability to be mounted on a nearby wall unit as the administrator sees fit. The control module is completely made of hardboard, as it trades weight for protection of the internal circuits. Its space is partitioned to house the power supply, back-up batteries, circuit and PIC DevBugger board without causing interference (see Figure E.5 for partition of control module). On the back, there is space for the power supply unit cable connections and ports for each of the modules. For this project, five D-Sub ports for the modules were created; however, the large size of the control module's back wall can accommodate for as many as 10-12 ports. There is also an additional port provided for the optional serial interface to the PIC DevBugger board. The control module also comes equipped with a small lock to secure the unit.

7.5 LOCKING MECHANISM

7.5.1 Analysis of Problem

The lock for each module must be electronically triggered. When the door is closed, the module should be locked, and cannot be unlocked by any physical method. However, when the user requires a module to be opened, that module must have its lock raised for 3 seconds. If the user does not open the door after 3 seconds, the lock must fall back into its rest position. After the user opens and closes the door, the module must lock itself immediately upon closing, and must remain difficult to tamper with.

7.5.2 Solution

The locking mechanism is a simple deadbolt device. A curved protruding lock seat was attached to the inner face of the door, with a slot for a locking pin (see Figure E.6). A solenoid was mounted vertically above the inner roof of the module, such that when the system is inactive, the pin will rest in the lock seat, and prevent the door from opening.

When the operator unlocks the door from the interface, power is supplied to the solenoid, raising the pin, and allowing the door to be opened. After 3 seconds, power is cut to the solenoid, causing the pin to drop down to its rest position. If the door is not opened during this time, the pin will fall back into the hole, and the door will be locked again. If the door is opened, the pin will still drop. When the door eventually closes, the pin will ride up on the rounded edge of the lock seat, fall into the pin slot and thereby lock the door.

The locking solenoid selected for this project was a Ledex 12V Tubular Pull Solenoid (see Appendix I for data sheet). This was chosen due to the strong pulling power of the solenoid and the long pull distance, thereby allowing the solenoid pin to rest firmly in the lock seat. While overheating is a concern for this solenoid due to its large number of coil turns, it is suitable for the lock because the opening signal is given to the lock for a maximum of 3 seconds.

The pin retracts over a distance of approximately 10mm, and was modified from the original plunger by cutting off the extended cast iron rod, retaining only the steel pin. Grooves were cut into the top of the pin, and a wire was pushed through a hole pre-drilled through the pin. This wire extended through another pre-existing hole at the rear of the solenoid, effectively suspending the pin at a certain height. The wire is soldered into a closed loop of a larger diameter than the solenoid hole, preventing the pin from dropping out of the solenoid during its rest state.

7.6 JAMMING MECHANISM

7.6.1 Analysis of Problem

In user or guest mode, the door must stay open for 15 seconds, while in administrator mode, the door must stay open indefinitely. This system must counter-act the closing mechanism (see section 7.7) during this period. As such, the door must be physically jammed open in an elegant, non-intrusive manner. However, after the allotted time is up (or when the operator wants to close the door at their own leisure), the jamming system must be deactivated in order for the closing system to perform its task.

7.6.2 Solution

The jamming device consists of the push solenoid mounted vertically and a jamming arm on the door (see Figures E.7 and E.8). When the user opens the door, the curved outer face of the arm gently pushes up against the solenoid pin and slides past, allowing the door to open. However, should the user release the door, the flat inner face of the arm prevents the solenoid from pushing upwards and retracting. This effectively jams the door at a 90-degree angle. Should the user desire to open the door at greater than 90 degrees, the user must hold it at such an angle. However, if the user releases the door, it will still remain jammed at the 90-degree angle.

After 15 seconds are up (for the user or guest) or when the closing button on the outside of the door is pushed (admin and user), the solenoid pin pushes upward, extracting itself from the path of the jamming arm. Coupled with the passive device, the door closes automatically, landing on the inner sensor. This sensor opens the circuit to the jamming solenoid, causing it to drop back down to its rest position.

The jamming solenoid is a miniature 12V Guardian A420-067074 push solenoid (see Appendix I for datasheet). This solenoid was selected due to its small size and mass, as well as its low operating temperature and relatively low power consumption. In addition, its rod protrudes from the rear of the housing; thus, when in an inverted orientation, it effectively acts as a pull solenoid, dropping down due to gravity in its rest state.

7.7 Closing Mechanism

7.7.1 Assessment of Problem

The closing mechanism must automatically close the door within 3 seconds without failure. It must retract the door quickly and with enough force to allow the locking mechanism to work effectively. However, it must not close with such severity as to cause physical harm to the operator.

7.7.2 Solution

The closing device simply consists of a prefabricated in-situ coiled spring mechanism. These devices are used in retractable pen keychains, and are very good for this project due to the strength of the spring and the discreet aesthetic of the extendable wire. The metal tips are attached to the door via a single heavy-duty staple, which provides a firm attachment to the wire while minimally impacting the exterior.

Please refer to Item E.1 for calculations for the maximum tension experienced in the spring, based on the moment of inertia of the doors about their hinge. Based on this calculation, the force required is 0.07 N, which can be handled by the spring, which exerts a uniform force of approximately 0.35N. This allows the door to close within 0.6 seconds. This force is strong enough to force the lock pin up and into the lock seat, while the extended time of closing gives the user enough warning that the door is closing.

To ensure that the doors do not "bounce" when closed, magnets are located along the vertical edge of the door and inner wall. This ensures a smooth closing action. It also provides additional force to force the lock pin into the lock seat, and also hinders foreign intrusion.

7.8 Suggestions for Subsystem Improvements

As a first-generation prototype, improvements can be made in terms of reliability & robustness, weight, and cost.

7.8.1 Reliability & Robustness

The locking system is generally reliable for all of the modules. However, to improve the reliability, the design can be refined so that the pin head is a perfect hemisphere so it rides up the lock seat in a smoother manner. In addition, the lock seat should be made of metal with a non-stick coating in order to reduce friction.

The jamming system can also be refined so that the jamming arm does not flex. Currently in some of the modules, the jamming arm can be perturbed when there is a vibration, causing the solenoid to slip and allowing the door to swing closed. This can be remedied by using a thicker jamming arm or a solenoid with a longer pin.

The system can be made more impenetrable by using stronger magnets to resist users from simply pulling on and opening the door without permission. Better build quality also ensures fewer panel edges, which could be exploited by malicious users.

7.8.2 Weight

An alternative to cardboard would have been foamboard or corrugated plastic, which are both lighter (albeit more expensive). In particular, foamboard plus aluminum is a very structurally sound combination that resists both the effects of loading and impact. For our design, a foamboard + aluminum door design for all of our modules would have likely improved the weight without adding considerable cost to our design.

7.8.3 Cost

Because this prototype is a one-off model, parts would have to be made or bought in discrete units, thereby increasing the price. In mass production however, many of the parts can be ordered in bulk (i.e. hinges, aluminum sheet, rivets, etc), hence decreasing the per unit price.

8. CIRCUIT SUBSYSTEM

8.1 Circuitry Overview

The circuit subsystem consists of the power source, back-up battery circuit, PIC board, 7 to 10 decoder, transistor circuit, solenoids, sensors and switches (as shown in Figure F.1). The multiplexer circuit, transistor circuit, solenoid circuits, and sensor circuits are mounted on the driver/sensor circuit board. The board is connected to a 12V and 5V power source, and it is connected to the PIC via a 40-pin ribbon cable (see Figure F.2 for complete schematic).

8.2 Assessment of Problem

The circuit subsystem needs to provide power for the PIC board and the actuators. It also needs to connect the PIC board's output signals to the actuators and deliver input signals from the sensors and switches to the PIC board. Four main tasks were identified for the circuit:

- Transmit PIC output signals to power the solenoids
- Transmit sensor signals to the PIC
- Deliver power to the entire system
- Switch between DC power and back-up batteries

8.3 Transmitting PIC Output Signals to the Solenoids

8.3.1 Analysis of Problem

Ten solenoids are to be controlled by the PIC microcontroller, and each solenoid is to be independent from each other. Thus, it is imperative that the PIC can control each solenoid individually. In order to drive the solenoids, enough power must be provided for the actuation to occur. It is also important for the circuit that the PIC is not affected by the inductance of the solenoid. Hence, the PIC board also must be protected from voltage spikes from the solenoids.

8.3.2 Solution

To conserve pin usage of the PIC, the data sent from the PIC is coded with two of the pins choosing between the "jam" solenoid and the "unlock" solenoid and five pins choosing the box on which the solenoids will be activated. A multiplexer circuit built from and-gates was used to decode the information, and the decoded signals are then fed through a transistor, which powers the solenoids. The multiplexer circuit acts as a 7-to-10 decoder by decoding the 7-bit information from the output pins of the PIC into 10-bits to pass on to the transistors. A schematic of the transistor circuit is shown in Figure F.3. The AND-Gates in the multiplexer circuit can provide a maximum of 20mA. In order to drive the solenoids, 1A of current is needed. To amplify the current, 10 transistor circuits are used. Each transistor circuit consists of a Darlington TIP-122 transistor (see Appendix I for datasheet) and a 1kΩ resistor (see Figure F.4 for transistor circuit). To protect the transistor circuit from voltage spikes, a 1N4001 diode is placed across each solenoid to drain the induced currents. An LED is attached in parallel with each lock solenoid with a 1kΩ resistor to indicate when the door is unlocked (see Figure F.5 for solenoid circuit).

To ensure the proper functioning of the multiplexer circuit, the circuit itself was simulated using Altera's Quartus II 8.0SP1 software. See Item F.1 for simulation results depicting the correct, desired behavior. The resultant waveform from of the outputs from the AND-gate circuit is the desired waveform, so the circuit design was cleared for development on the PCB board on this basis.

8.4 TRANSMITTING SENSOR SIGNALS TO THE PIC

8.4.1 Analysis of Problem

Each box has a signal to the PIC (via a pushbutton) to indicate when the operator wants to close the door, and also has a signal to the PIC (via a microswitch) to indicate when the box is closed. A conventional system would call for two inputs from each box, which would require a total of 10 input pins on the PIC. The PIC input pins can receive up to 20mA per pin at 5V.

8.4.2 Solution

To conserve the number of pins used, the sensor microswitch and the close-door pushbutton is connected in parallel to generate 1 signal per box (see Figure F.6 for sensor circuit). This requires only 5 pins from the PIC. The microswitch and the pushbutton were set up so that each state is known to the PIC (refer to Section 9.6 for more information).

8.5 POWERING THE SYSTEM

8.5.1 Analysis of Problem

The system requires a stable 12V and 5V DC power source. The 12V source must be able to drive the PIC board (voltage regulator rated for 1A) two of the solenoids (44Ω and 9Ω, 1.8A in total) at once and the 5V source must be able to sustain the current for the multiplexer circuit and for the sensors.

8.5.2 Solution

To meet the power requirements, a computer power supply was used in the system. See Table F.1 for a breakdown of power requirements by component and refer to Item F.2 for calculations used to derive the power requirements. The power supply can provide 15A at 12 V and 30A at 5V, which exceeds the required amount of 35 W. It is important for the proper functioning of the driver/sensor board that all circuit components have a common ground. Using a computer power supply from a salvaged computer also provides a benefit in terms of cost and reliability, since computer power supplies undergo extensive testing before being released to market.

8.6 SWITCHING BETWEEN DC POWER AND BACK-UP BATTERIES

8.6.1 Analysis of Problem

The PIC board must be powered and remain operational during a power outage. It needs to do so by switching to a set of back-up batteries when there is no DC power. The PIC is very sensitive to voltage variations, thus the power switching must occur very fast. The solenoids do not need to be powered when the DC power is disconnected.

8.6.2 Solution

The back-up battery switching circuit (see Figure F.7) accomplishes this task and provides the minimum required voltage to the PIC board. The output from the back-up battery circuit is divided into 12V and 5V. The 12V output is used to power the PIC board, and the 5V is used to power the sensors and the multiplexer circuit. In this circuit, NTE586 Schottky Diodes (see Appendix I for datasheet) are used to ensure fast switching. The power for the solenoids is completely separated from this circuit to further protect the PIC and the sensors from induced voltage spikes.

8.7 SUGGESTIONS FOR SUBSYSTEM IMPROVEMENT

The design of the circuit subsystem could be improved in the following three areas to further improve usability and cost.

8.7.1 Improved Encoding of Signals

The output was coded from 10 bits to 7 bits. However, it can be further improved to only 4 bits, which conserves the power usage from the PIC, and it allows the other pins for more functionality. The input from the sensors and the push buttons are connected in parallel, which caused the PIC to receive the same signal whether the door is open or when the user has pushed the button. This input later has to be deciphered using heuristics. A more efficient approach is to have separate inputs for each and then encode the 10 pins to 4 bits. By encoding them, rather than putting the sensors and pushbuttons in parallel, it is possible for the PIC to receive exact signals of whether the door is open and whether the user has pushed the button.

8.7.2 Stepping Down Voltage using Voltage Regulators

The current back-up battery circuit uses a set of resistors to step down the voltage for the sensors and the multiplexer circuit. This is not an ideal solution since the internal resistance of the sensor circuit varies depending on the state of the modules. A better solution would be to use a 5V voltage regulator to step down the voltage instead of using a set of resistors.

8.7.3 Battery Recharging

The circuit subsystem can be further improved by having the ability to charge the back-up batteries when DC power is connected.

9. MICROCONTROLLER SUBSYSTEM

9.1 MICROCONTROLLER OVERVIEW

The microcontroller unit is at the heart of the system, driving actuators and providing a way for the user to begin interacting with the machine. The microcontroller that is used is the PIC16F877 from MicroChip. The PIC DevBugger board is used so the PIC's connection to ground, power and the oscillator are already made. The hardware required for the user interface is a 16 character, 2 line 5x8 pixel LCD display controlled by Hitachi's HD44780 Driver IC, a 4x4 matrix keypad and a DS1307 RTC Chip (see Appendix I for datasheets). Data Memory RAM is used to store variables during runtime. The code itself is downloaded on to Flash ROM while the activity logs and account information are stored in EEPROM. An overview of the code is presented in Item G.1 and the complete compendium required to program the PIC is presented in Item G.2.

9.2 ASSESSMENT OF PROBLEM

The role of the PIC is to process all input from the user and the machine and produce the correct output. When dealing with the operator, this implies using keypad input to display appropriate prompts on the LCD screen. When dealing with the machine, this entails using sensor and switch data, to run appropriate algorithms to send the correct output signals to the circuit. The main problems identified in the creation of this subsystem are:

- Providing an easy-to-use interface.
- Developing functions that could be inherited based on the operator's authorization.
- Maintaining the security and reliability of the system.
- Storing data efficiently
- Interacting with peripheral hardware.

9.3 PROVIDING AN EASY-TO-USE INTERFACE

9.3.1 Analysis of Problem

A 16x2 LCD screen is the only method of conveying information to the operator. It is thus essential to transmit as much information as possible on this limited amount of space. The information displayed needs to not only indicate to the operator's possible options at a particular state of the interface, but also explain how to navigate to other screens. In the case of the logs, it is also essential to transmit information about dates, times, user names, and module names in a very limited amount of space. The keypad, in addition to having 10 digits, has four letters and two rudimentary symbols. It is then inevitable that keypad prompts need to be designed with a level of redundancy in order to facilitate the input of alphanumeric data, while also providing enough utility keys for navigation.

9.3.2 Solution

The solution that was developed is a tree-based finite state machine, in which each level has a series of sister menus. Within each of these menus, there are more options laid out in a similar fashion (see Figure G.1 for a layout of the interface). In order to make navigation easy, the option presented at each screen was spelled out on the top line. On the bottom line, the keys that are required to perform the current option, as well as navigate to the next and previous options are indicated. Each collection of sister screens, also has one option to go back, which allows the operator to traverse back up the option tree. The pound and asterisk keys are relabeled as the left and right keys, while the number 0 is used as the 'OK' key. In order to improve user friendliness, certain keys are disabled when not required. For example, the letter keys are disabled when setting an expiry time. A backspace key is also provided so that the operator can go back and fix input.

9.4 DEVELOPING INHERITABLE FUNCTIONS

9.4.1 Analysis of Problem

The total list of tasks that can be performed by the all the operators is substantive, and writing individual and specific functions for each is not only onerous but difficult to manage (see Table G.1 for break-up of tasks). Considering that only 8192 words of program memory are available, there is also a risk of overflowing the program memory. Even if this risk is avoided, writing code in this manner is also difficult to debug, and makes inheritance of member functions virtually impossible. It is also important to centralize common functions due to paging issues, which could render the code clumsy, problematic and slow.

9.4.2 Solution

Many of the required functions are common to all three types of operators, with the scope of the function varying based on the operator's authorization. As such, an object oriented paradigm is adopted so that member functions can be inherited based on authorization. Functions were developed that would take in certain registers as input, and based on the encoded values of these registers, would provide certain capabilities. The functions for assigning modules, adding users/guests, opening modules, and changing passwords were developed in this way. For example, all three levels of operators have the capability to open modules, but which module specifically depends on the authorization. Similarly, both administrators and users have the capability to set passwords, assign modules, and set expiry times (for the user and the guest respectively) so a common function was created for these three tasks. Finally, there are certain tasks that do not need to discriminate by operator type such as changing passwords, so here again a common function was developed.

9.5 Maintaining the Security and Reliability of the System

9.5.1 Analysis of Problem

The utmost priority of this prototype is the maintenance of security both physically and electronically. Thus the partition of functions is paramount, in ensuring that the operator is unable to 'hack' the interface by performing unexpected input sequences that may result in breaches of security. The main methods by which security may be breached are during log-in, due to stack overflows, and while performing inherited or common functions.

9.5.2 Solution

In order to prevent security breaches during log-in, four-letter alphanumeric identification tags and passwords are facilitated, permitting enough combinations to prevent easy guessing. Also, any time the log-in function is accessed, a small helper function is called immediately to delete any users that may have expired. This eliminates the possibility of operators accessing the system after the validity period. Another obstacle to security is stack overflows, which can be created by performing 'call' statements until the microcontroller is no longer able to discern its return address and is forced to jump to the next instruction. This is avoided by ensuring that the stack was never more than 6 layers deep, which provides an effective margin against the 8 calls required for the stack to overflow. The third method of hacking is while performing inherited functions. This is avoided by accessing all inherited or common functions using call statements, so that at the end of the function, the program returns to the point-of-entry, which is within the realm of the operator's authorization. Similarly, for functions specific to a certain type of operator, 'goto' statements are used both to enter and exit, to prevent unnecessary use of the stack and prevent access by other operators.

Other security features were also developed such as a 60-second timeout, so that the interface automatically logs out users after 60 seconds if no input is received. This prevents unauthorized use by third parties if a user forgets to logout.

9.6 Interacting with Peripheral Hardware

9.6.1 Analysis of Problem

Despite the versatility of the PIC on the DevBugger Board, an additional clock chip (DS1307) is required to autonomously keep and update the time. Interacting with this hardware is vital in displaying date/time information in standby mode as well as in the generation of system activity logs. In addition, the requirement that the system time is accurate even without power, necessitates the use of this chip which is independently powered by a standby coin battery. The PIC must also receive both input in the form of sensors from each of five modules, and deliver output signals to drive the solenoids. Considering the fairly limited availability of pins, it is required that signals be sent in an effective manner so that each module can be configured effectively.

9.6.2 Solution

The interaction between the PIC and the RTC chip was facilitated via an internal I2C bus. The code that was provided was in the form of macros was converted to functions in order to improve the modularity of these functions and reduce the affected size of program memory. The most important aspect of the clock chip is in the development of logs. Here, the time was obtained before and after the modules are closed in order to store the time of entry as well as the elapsed time.

The code for interacting with the actuators and sensors was fairly straight-forward since only one module needs to be dealt with at a time. The pin assignments for interacting with the machine are included in Table G.2. These assignments are ideal because it allows all input to occur in one port (Port A), and all output to be designated to another port (Port C). This makes debugging and wiring very easy. When interacting with the machine, there is a very specific series of time or event-driven steps that is followed, in accordance with the RFP (see Figure G.2 for flowchart of algorithm). For example, the door must be unlocked for exactly three seconds and then locked again, if it is not opened. Also, in user or guest mode, once opened the door must close after 15 seconds unless the user keeps the door open. The specific waveforms that are expected and handled by the machine are shown in Figure G.3.

9.7 STORING DATA EFFICIENTLY

9.7.1 Analysis of Problem

Vital statistics such as account information, logs and system configuration must be saved for later retrieval by the system. However, the only form of non-volatile memory that is readily accessible is EEPROM, which allows stored data to be accessed even after a power outage. However the EEPROM only has 256 bytes of memory so only vital pieces of data can be stored. Furthermore, the data must be encoded to maximize the value of the stored information.

9.7.2 Solution

The EEPROM is partitioned into three sections (see Table G.3 for a full partition scheme of EEPROM). The first section has reserved space for configuration information including configured modules, active users, active guests, administrator name, administrator password and a control byte indicate whether it is the first restart. See Table G.4 for a summary of special purpose EEPROM bytes. The second section has reserved space for account information (user names, passwords, module assignments, expiry dates and times) for users and guests. The third section has reserved space for logs partitioned by user. Data derived from the keypad (such as usernames) are stored in the raw format from the key (i.e. 4-bit nibble) allowing two characters to be stored per byte. All configuration bytes are encoded bitwise with each bit corresponding to a particular module, guest, or user. Numerical data from the clock is stored in the form binary-coded decimal (BCD) allowing two decimal digits to be stored per byte. As a result of efficient encoding, there is extra space left on EEPROM, allowing the administrator to create an extra user account if desired.

9.8 SUGGESTIONS FOR SUBSYSTEM IMPROVEMENT

The three main areas for improvement are timing functions, pin assignments and the PC interface.

9.8.1 Timing Functions

Currently timing functions, for machine interfacing as well as the 60-second logout are performed using dummy loops that were configured using trial and error to last an appropriate window of time. This could be more efficiently performed using the PIC's built-in timer interrupt, which would make the code less hassle-free and improve event handling capabilities. Using the timer interrupt with pre-set constants would also the administrator to modify system capabilities (such as allowing the door to stay open for 30 seconds) easily.

9.8.2 Pin Assignments

The second area of improvement is the pin assignments. The I2C bus is wired internally on the DevBugger on pins 3 and 4 of Port C. However, the only full unused port (disregarding the clock) is also Port C. Since the only time Port C needs to be used as output, is during the open module function, the current implementation, has the I2C as the default on Port C and actually disengages it during interaction with the solenoids. This means that it is currently impossible to use both the solenoids and the RTC chip. Although this poses no hindrance in satisfying all the required functions, in the future if further functionality is required, this issue needs to be addressed by connecting the I2C bus to other available pins such as pins 1 and 2 on Port E.

9.8.3 PC Interface

A PC interface should definitely be a considered to bolster the functionality of this product. This option could be facilitated by the DevBugger board's inbuilt USART module. This would enable the administrator to perform all of the current functions using the comfort and versatility of a PC, which is ultimately more user-friendly than even the best-designed LCD and keypad interface.

10. INTEGRATION

10.1 OVERVIEW

The integration of the project involved the following objectives:

- Successful output of signals from the microcontroller pins to the circuit board
- Successful output of signals from the circuit board to the storage modules
- Completion of the storage modules, including the installation of self-closing mechanism, the lock mechanism, and the door jammer mechanism

During the first integration stage, the following tasks were accomplished:

- Completion of control module, which houses the DevBugger board, the power supply, the battery supply, and the circuit board
- Creating the circuit which controls the power source of the DevBugger board (AC normally, DC back-up)

The following was accomplished during the second stage of integration:

- Installation of sensors
- Internal wiring of the modules
- Fabrication of extension cables from circuit board to modules

The final integration stage consisted of testing the follow operations:

- Sensor response
- Pushbutton response
- Proper unlocking, closing and locking under a variety of circumstances

10.2 PHASE I: COMPLETION OF CONTROL MODULE

During this stage, the control module was fabricated. Space for the internal components was provisioned. Upon completion of the module, the circuit board was glued onto the floor of the box. The DevBugger board was placed on Velcro pads, and placed atop the support pillars to allow for easy removal and servicing. The power supply was screwed into the back wall, and the battery pack was attached to the side wall via a Velcro backing. Cables were run from the circuit board to the rear of the control module, where the plugs were glued in place.

10.3 PHASE II: DEVELOPING SYSTEM MODULARITY

At this stage, each box was outfitted with a sensor and a pushbutton. The sensor was necessary for the microcontroller to know when to stop sending signals to the jamming solenoid. The pushbutton was necessary for the administrator to close the module (and for the user to close the module before 15 seconds were up). Initially, the sensor was another pushbutton that the door pushed against. However, it was discovered after testing that the pushbutton sensor was not very effective due to its poor sensitivity. It also tended to obstruct the door slightly, which dislodged the lock pin. Thus, it was replaced with a microswitch with a thinner interface, and moved closer to the door face to allow for optimum sensor contact without adversely affecting the locking mechanism.

Small holes were drilled into the walls, through which wires were fed and soldered onto the electromechanical systems. Serial ports were soldered onto the external end, which would connect to the control module either directly or through an extension cable. The extension cables were made from multi-strand copper core wire, which were twisted using the power drill.

Single strand wire was attached from the door button back to the serial port. This required that the wire wind itself along the door against the hinge. However, the natural elasticity of the wire meant that the door would have difficulty closing perfectly. This resulted in the sensor not working routinely due to the gap between the door and the sensor face. To eliminate this problem, the sensor wires were re-fabricated using multi-strand wires, which allowed for more leeway and flexibility.

10.3 Phase III: Testing

The third stage of integration required that the system be tested to see if there were any errors in the logic of the code. This included the team members opening and closing the modules in various ways, pressing the buttons and other actions in various combinations and sequences to see if the code could be affected in any way.

In addition, by testing the modules repeatedly, the reliability of the components could be better understood. Problems such as locking, jamming, and retraction occurred occasionally during the testing process, some which required modifications to the design itself.

One major issue was the pushbutton’s binary input, which essentially acted to trigger the jamming solenoid on and off after each push, regardless of the state of the module. This problem was solved by reading the signal from the pushbutton just once, and deactivating the pushbutton until the user confirmed on the keypad that the module was secured.

11. OVERALL SYSTEM IMPROVEMENT SUGGESTIONS

This section addresses overall improvements to the system that are applicable in addition to the specific subsystem improvements mentioned earlier. Though the product operates effectively and satisfies the requirements of this project, there is room for improvement in many aspects of the project. As a first-generation prototype conceived, designed, and built within 3 months, there are inevitably shortcomings that prevent the prototype from becoming market-worthy. To move this project into the beta phase of design, and eventually to market, a method of construction should be refined and simplified.

In the beginning of the project, fabrication of the modules and the circuits underwent several stages of experimentation. In order for a product to be easily mass-produced, the important steps during construction must be isolated, and then simplified so that it can be repeated many times over without fault. Tolerances must be also reduced in order to have a more reliable product. This can be improved through partly through production techniques, but mainly through the design of the machine itself. For instance, the use of adhesives can be improved so that the pieces do not flex due to environmental factors. Parts such as the lock seat can be moulded and reproduced to create predictable results, rather than having them bespoke for each module set-up. More specifically, the system can be improved in terms of building materials, attachment mechanisms and central hub connection.

As an alpha prototype, the modules consist of a cardboard frame encased in aluminum sheeting. Although this serves satisfactorily to illustrate the concept, if put into production, it would be necessary to use more robust materials such as wood panelling and aluminum frames.

Similarly, the door closing device relies on a nylon string to provide tension, but in consumer or industrial applications, this string could easily be cut during day-to-day use. It is therefore necessary to use a retractor device with a chain spool, for example, instead of nylon to improve the robustness of the design. The door jammer is also made out of slender wood which may break after repeated use. It would be worthwhile to invest in a metallic version instead to improve durability.

The attachment mechanism currently used is a bolt system that locks the modules together from the back. However, since the walls are as rigid as was desired, it is possible to pry open the modules from the front face. Although the design should work in concept, in order to realize its effect in the product, it is necessary to reinforce the steel tabs at the back and ensure that they are better lined up, so that when two modules are screwed together, they essentially behave as one. The system could be improved in this regard, by having longer tabs to increase the binding surface and employing more tabs along the edge of the walls. The tabs should also be permanently affixed to the modules using screws instead of glue.

In addition, more testing is necessary to work out the various cases and conditions from the signals coming in from the sensors that could cause unwanted effects. The option for a PC interface that manages the accounts and downloads logs is another improvement which could simplify the administrator’s duties. The PC interface could also have the option of remotely unlocking the modules as well.

The final consideration for improvement pertains to connecting the modules with the central module. Currently, the wires protruding from the module are approximately 4 feet long, which is adequate to connect to the hub. In industrial applications, where hundreds of modules may need to be connected however, it is clumsy to use extension cables to connect each module. It is therefore a worthwhile, to equip each module with a self-retracting spool of extendable wire that could be unwound to the appropriate length to connect to the central hub.

12. ACCOMPLISHED SCHEDULE

A chronological list of milestones accomplished, along with the nature of each milestone is listed in the following table. All the milestones outlined in the proposal were met, although the duration of certain tasks was modified. GANTT Charts outlining the proposed and accomplished schedule in detailing are included in Appendix K.

Table 3: List of Milestones in Chronological Order

No.	Date	Members	Milestone Description
1	Jan 9	All	*Deliverable:* Team finalized and subsystem responsibilities assigned.
2	Jan 15	EM	All drawings should be completed and overall concept of actuator system to be used should be decided.
3	Jan 15	MC	The microcontroller should have a solid idea of the program structure, and a general understanding of how to interface with the user and the system. The team will be informed of progress to date to garner any feedback.
4	Jan 26	All	*Deliverable:* Design proposal outlining conceptual design phase and selected solution should be complete and submitted.
5	Jan 26	CCT	Overall circuit design complete.
6	Jan 29	EM	Material selection finalized and solenoids purchased and installed in mule prototype.
7	Jan 29	MC	The user interface will be complete allowing for complete menu traversal and interaction. However, user prompts are not expected to produce any mechanical response and are symbolic only.
8	Jan 29	CCT	Specific circuit designs and component calculations complete
9	Feb 4	EM	*Individual Evaluation 1:* Completion of small modules including fabrication, installation of solenoids, and testing
10	Feb 4	MC	*Individual Evaluation 1:* The code for running the keypad and LCD along with the first version of the machine interface will be complete and functional.
11	Feb 4	CCT	*Individual Evaluation 1:* All prototyping done, circuits designs finalized and ready for soldering. Calculations of power complete, all components acquired
12	Feb 11	All	*Deliverable:* Submit notebooks containing all project and design activities.
13	Feb 16	EM	Full completion of medium modules including fabrication, installation of solenoids and testing.
14	Feb 16	MC	All data structures must be implemented and the administrator must be able to access all logs from EEPROM.
15	Feb 25	EM	*Individual Evaluation 2:* Completion of large module including fabrication, installation of solenoids, and testing

16	Feb 25	MC	*Individual Evaluation 2*: The Microcontroller member will have completed the final assembly code and downloaded it onto the PIC to demonstrate its functionality.
17	Feb 25	CCT	*Individual Evaluation 2*: Circuit soldering complete, all sub-circuits functional and debugged.
18	Mar 11	All	*Team Evaluation 1:* The system should be integrated and demonstrate some basic functionalities.
19	Mar 25	All	*Team Evaluation 2:* The system is expected to be completely functional except for very minor bugs.
20	Apr. 8	All	*Public Demonstration:* The prototype will be presented to the public and the team will field any questions.
21	Apr 14	All	*Deliverable:* The final report outlining the team’s process and prototype in detail will be completed and submitted.
22	Apr 14	All	*Deliverable:* Each member of the team will submit his design notebook with all design and project activities performed over the semester.

13. CONCLUSION

A prototype of an autonomous storage system with five storage modules was developed. The prototype satisfies all constraints, meets all requirements and performs well in the criteria outlined in the RFP. The total prototype cost is $194.76 and took 98 days to conceive, design and test to satisfaction. The prototype employs solenoids to lock and jam the door and a retractor to close the door. The door can be closed by the use of a pushbutton and a microswitch is used to detect if the door is closed. Extensions to the prototype including constructing the walls of the module out of more robust materials before releasing it to the market, and providing the operator with a larger LCD display to increase usability.

The prototype is meant to be a proof-of-concept of a system capable of handling many more storage modules. As such the system could be improved by considering the issue of scalability. In order to manufacture a system with many more modules, issues such as cabling need to considered. A microcontroller with more input/output pins should also be considered as well as the deployment of more control modules for each set of storage modules. The system could be simply improved by incorporating a burglar alarm and a PC interface to allow for easy managing and maintenance of the system.

Team 40 is satisfied with the delivery of a fully functional proof-of-concept prototype that has been developed on time and on budget.

APPENDIX A: REFERENCES

[1] M. R. Emami. *AER201 Engineering Design Course Notes.* Toronto: McGraw-Hill Ryerson, 2009, pp 18-30.

[2] Self Storage Association. “Self Storage Association Factsheet” Accessed January 20, 2009. Available HTTP: http://www.selfstorage.org/SSA/Home/AM/ContentManagerNet/ContentDisplay.aspx?Section=Home&ContentID=3900

[3] Industore. “Hanel: Multi-Space” Accessed January 14, 2009. Available HTTP: http://www.industore.co.uk/multi-space.php

[4] US Patent Office. “Automated Storage System (US20080208389)”. Accessed January 16, 2009. Available HTTP: http://www.patents.com/AUTOMATED-STORAGE-SYSTEM/US20080208389/en-US/

[5] Akers, L. E. (1966). *Particle Board and Hardboard.* Oxford: Pergamon Press, p. 76.

APPENDIX B: BIBLIOGRAPHY

Bates, Martin (2004). *PIC Microcontrollers: An Introduction to Microelectronics*, 2nd Edition, Elsevier.

Huang, Han-Way (2005). *PIC Microcontroller: An introduction to software and hardware interfacing*. Cengage Learning.

Morton, John (1995). *The PIC Microcontroller*. Elsevier.

Sedra, A, Smith, K. C. (1999). *Microelectronic Circuits*, 2nd Edition. Oxford University Press.

Verle, Milan (2008). *PIC Microcontrollers*, 1st Edition. Mikroelectronika

Vranesic, Zvonko (2008). *ECE253 – Digital and Computer Systems*. University of Toronto.

APPENDIX C: SUPPLIERS

Active Surplus
347 Queen Street West, 2nd Floor
Toronto, ON
M5V 2A4
(416) 593-0909
www.activesurplus.com

AER201 Design Store
Sandford Fleming Building
10 King's College Road, Room 3302
Toronto, ON
M5S 3G4

Brafasco
50 Milner Ave.
Scarborough, ON
M1S 3P8
(416) 298-0095

Creatron
255 College St.
Toronto, ON
http://www.creatroninc.com/contact_us.php

Home Depot
428 Ellesmere Road
Scarborough, ON
M1R 4E6
(416) 609-1800

Home Hardware
306 College St.
St. Toronto, ON, Canada

Office Depot
32 Steeles Avenue West
Thornhill, ON
L4J7Y1

Paper Mart
5361 Alexander St.
Los Angeles, CA
90040

Sayal
3791 Victoria Park Ave.
Units 1-5
Toronto, Ontario
Canada M1W 3K6
http://www.sayal.com/

APPENDIX D: DESCRIPTION OF OVERALL MACHINE

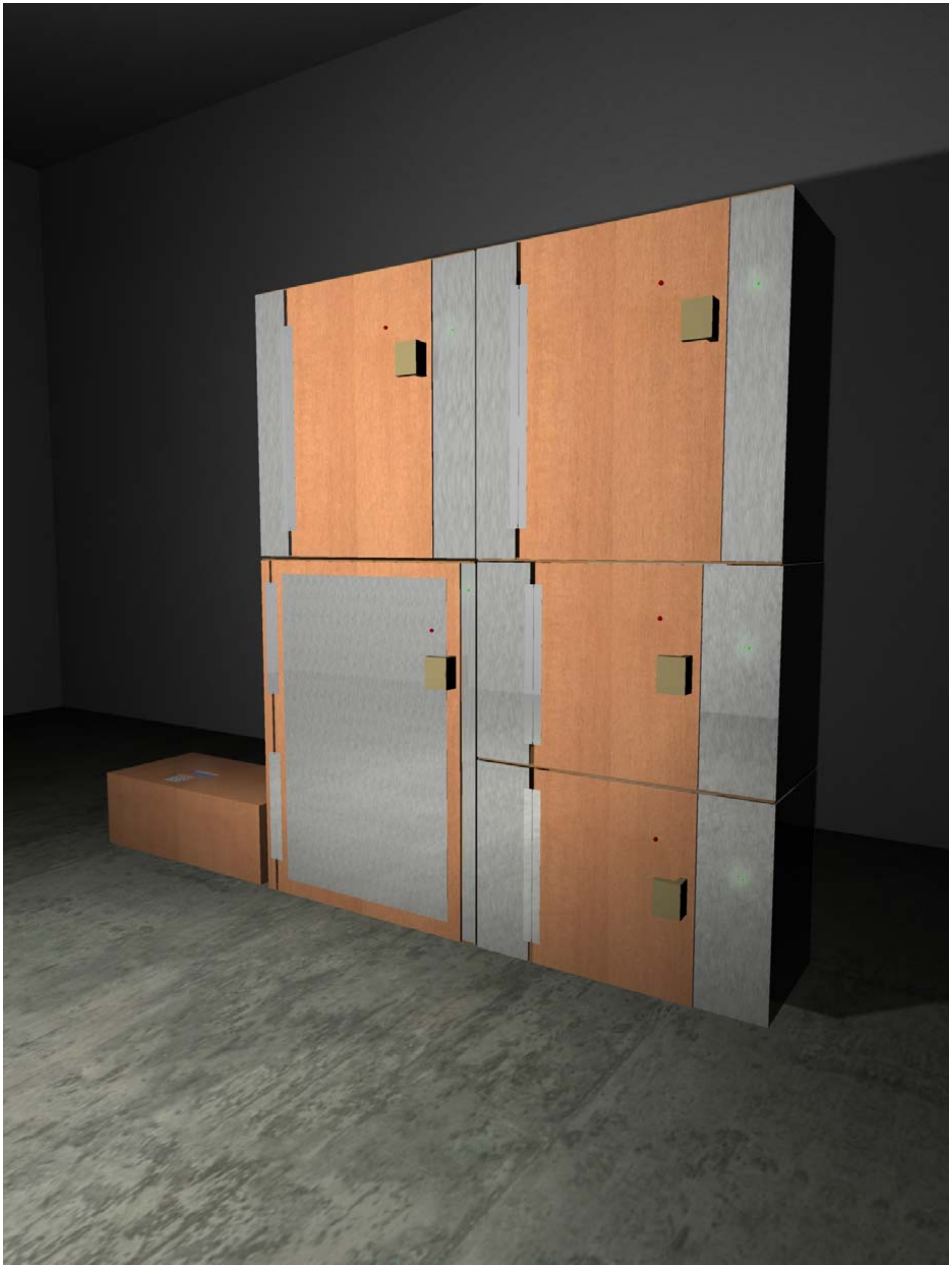


Figure D.1: Rendering of prototype with five storage modules (right) and control module (left).

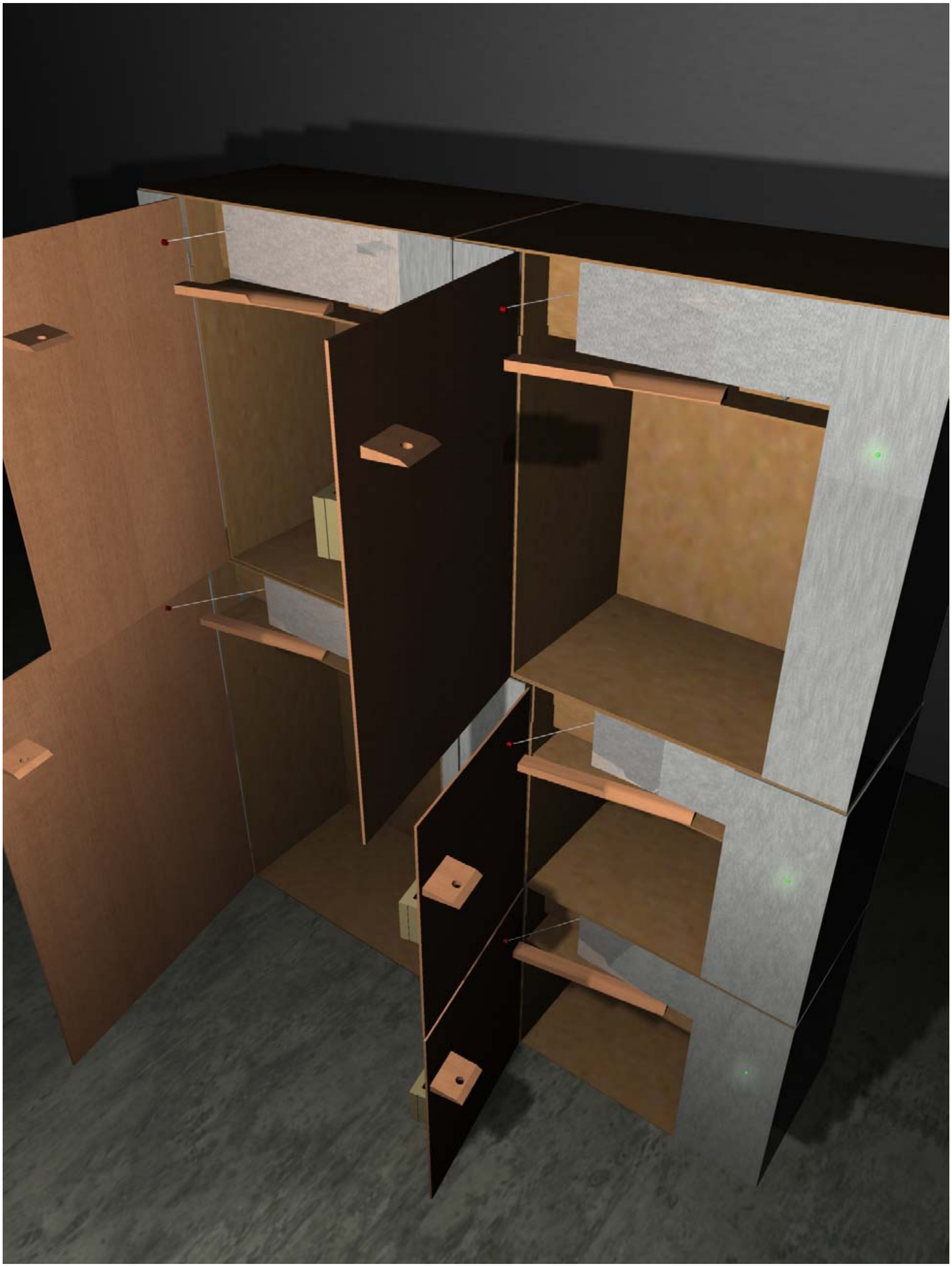


Figure D.2: Rendering of the interior of the prototype with the storage modules open.

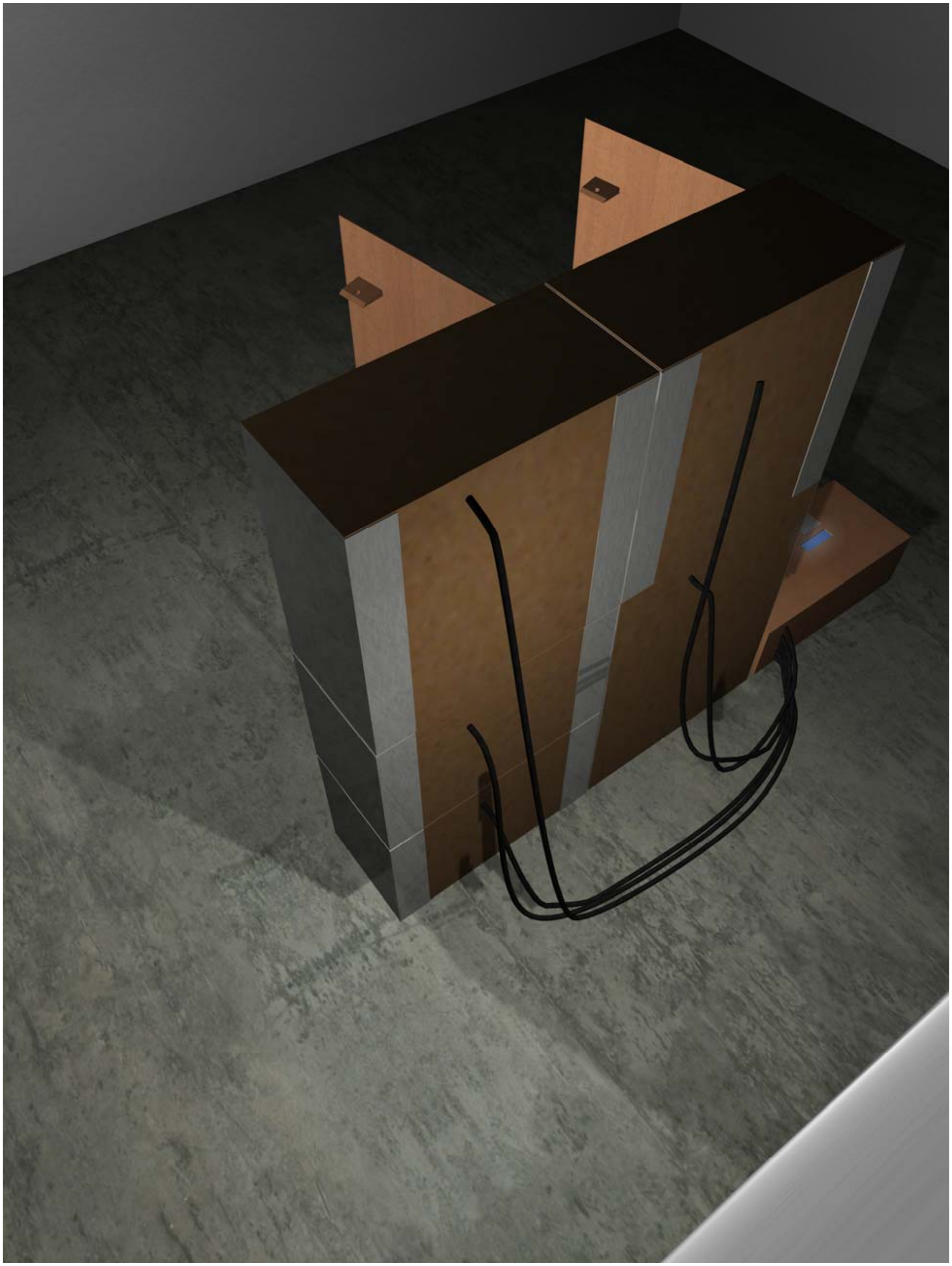


Figure D.3: Rear view rendering of the system illustrating module-to-hub connection.

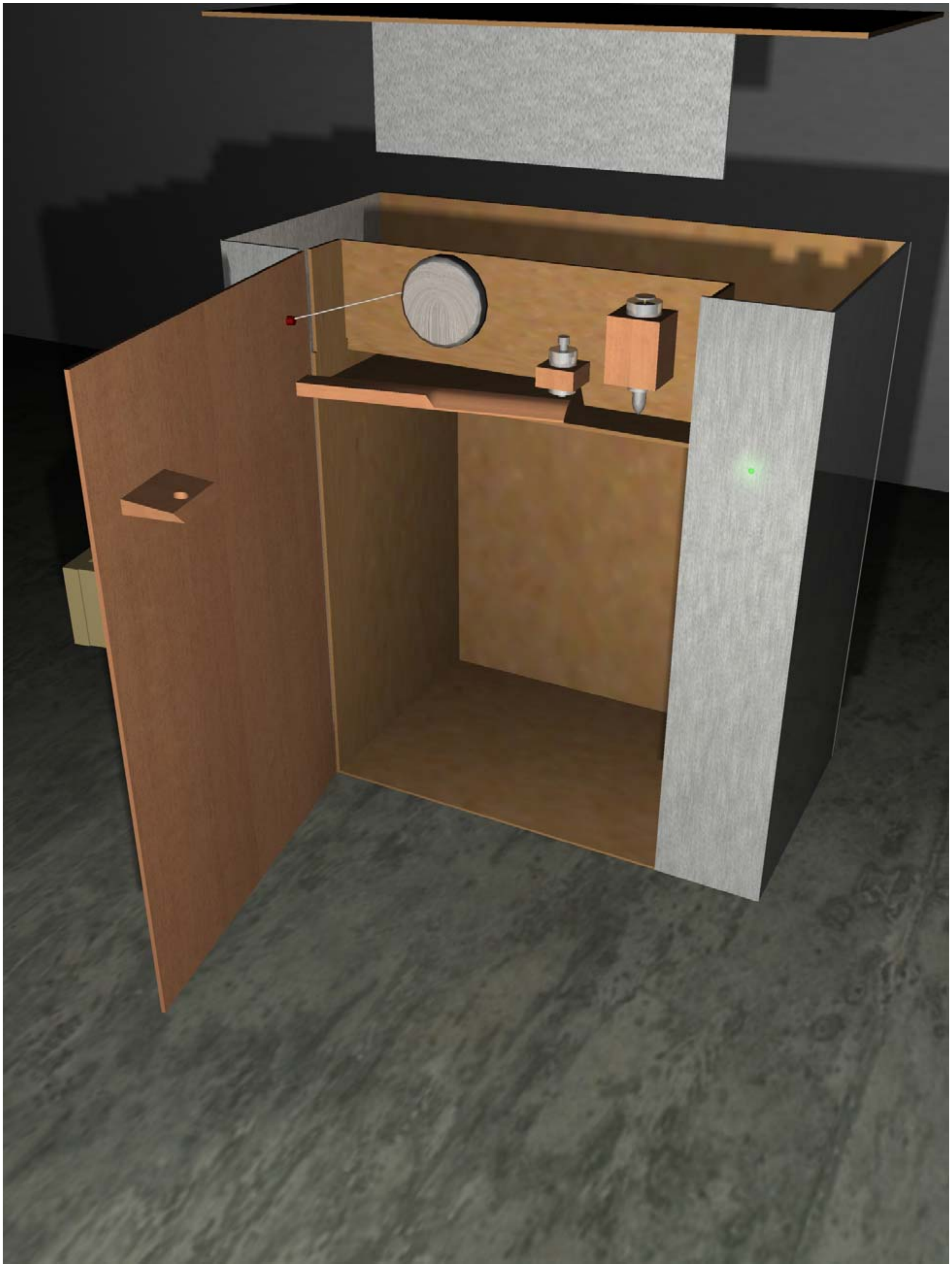


Figure D.4: Cutaway rendering of a medium module.

Table D.1: Overall Specifications of System

Attribute	Rating
Prototype Version	Beta
Prototype Cost	$194.76
Prototype Application	Consumer and Industrial
Power Consumption	35 W
Main Power Supply	110 V – 60 Hz 3-pin outlet
Backup Power Supply	8 AA 1.2 V Batteries
Small Module Weight	1632 g
Medium Module Weight	1988 g
Large Module Weight	1985 g
Module Support Capability	5
Microcontroller	PIC16F877

APPENDIX E: ELECTROMECHANICAL SUPPLEMENT

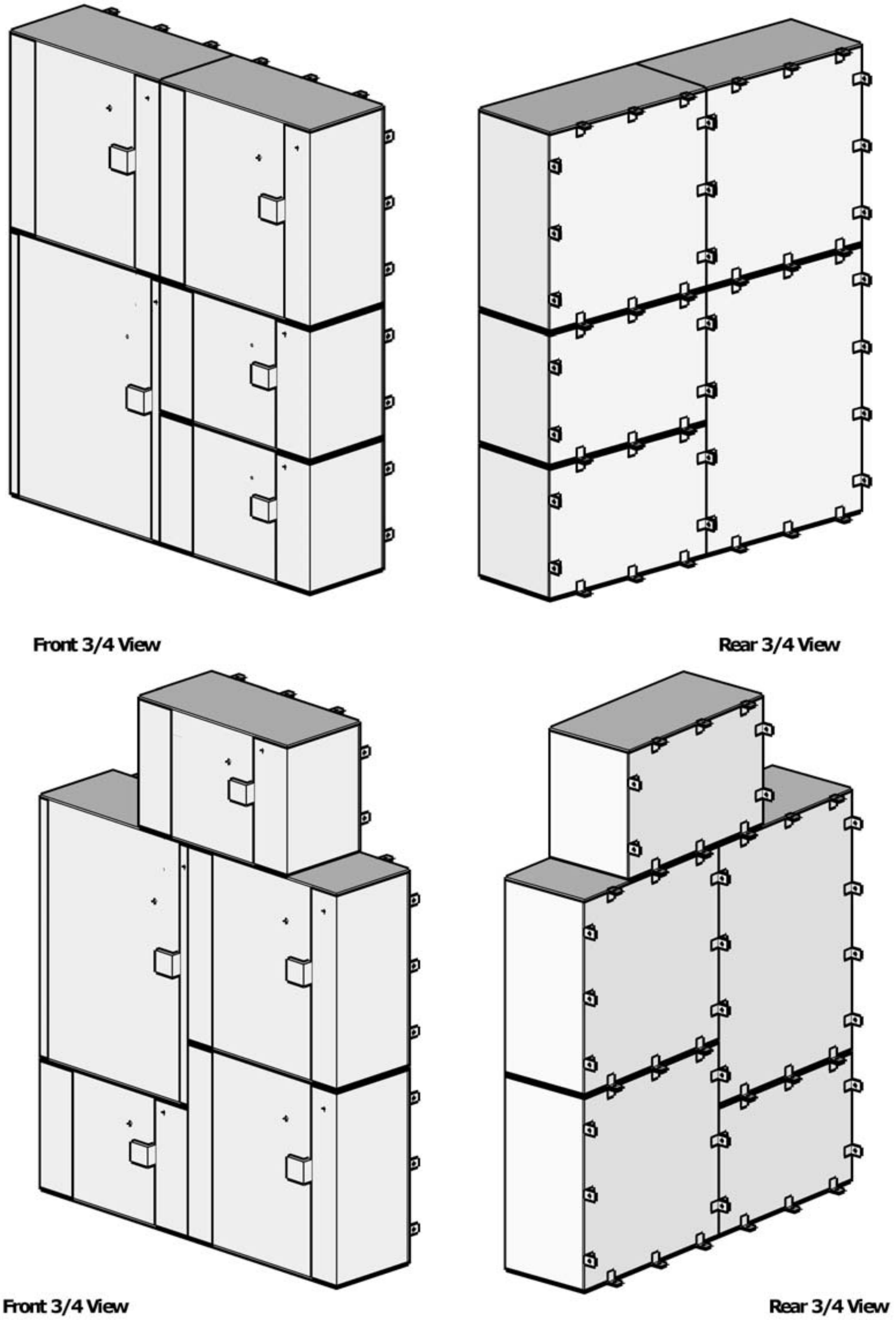


Figure E.1: Different configurations of the storage modules.

Small Module - Isometric Plan Scale: 1:5

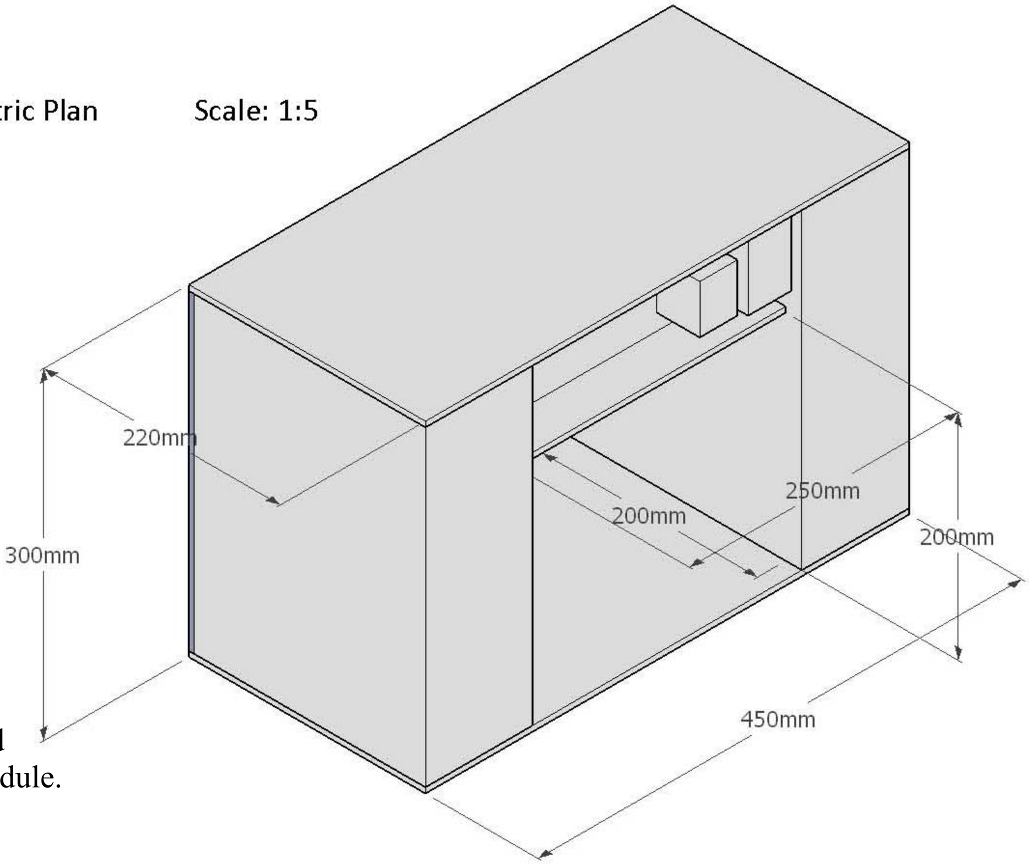


Figure E.2: Detailed drawing of small module.

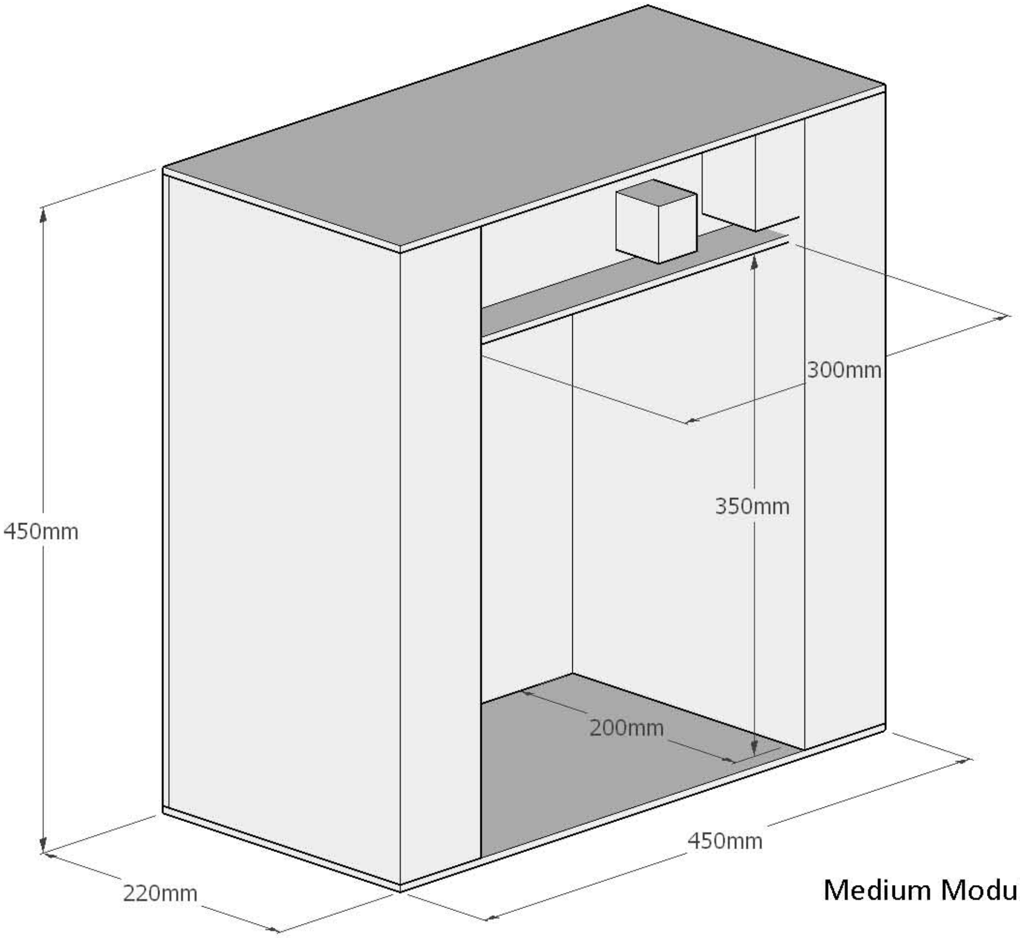


Figure E.3: Detailed drawing of medium module.

Medium Module - Isometric Plan Scale: 1:5

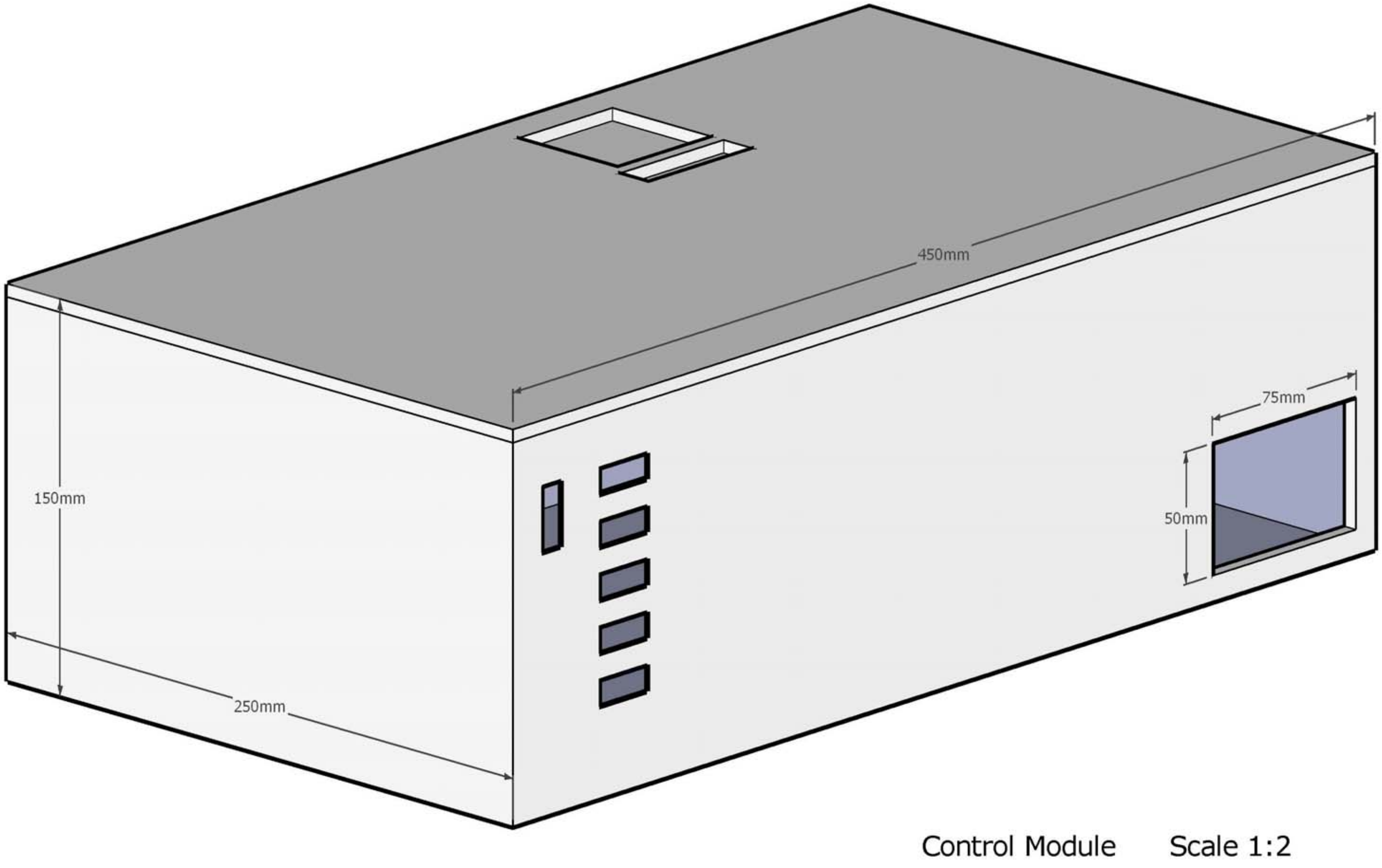


Figure E.4: Detailed drawing of control module.

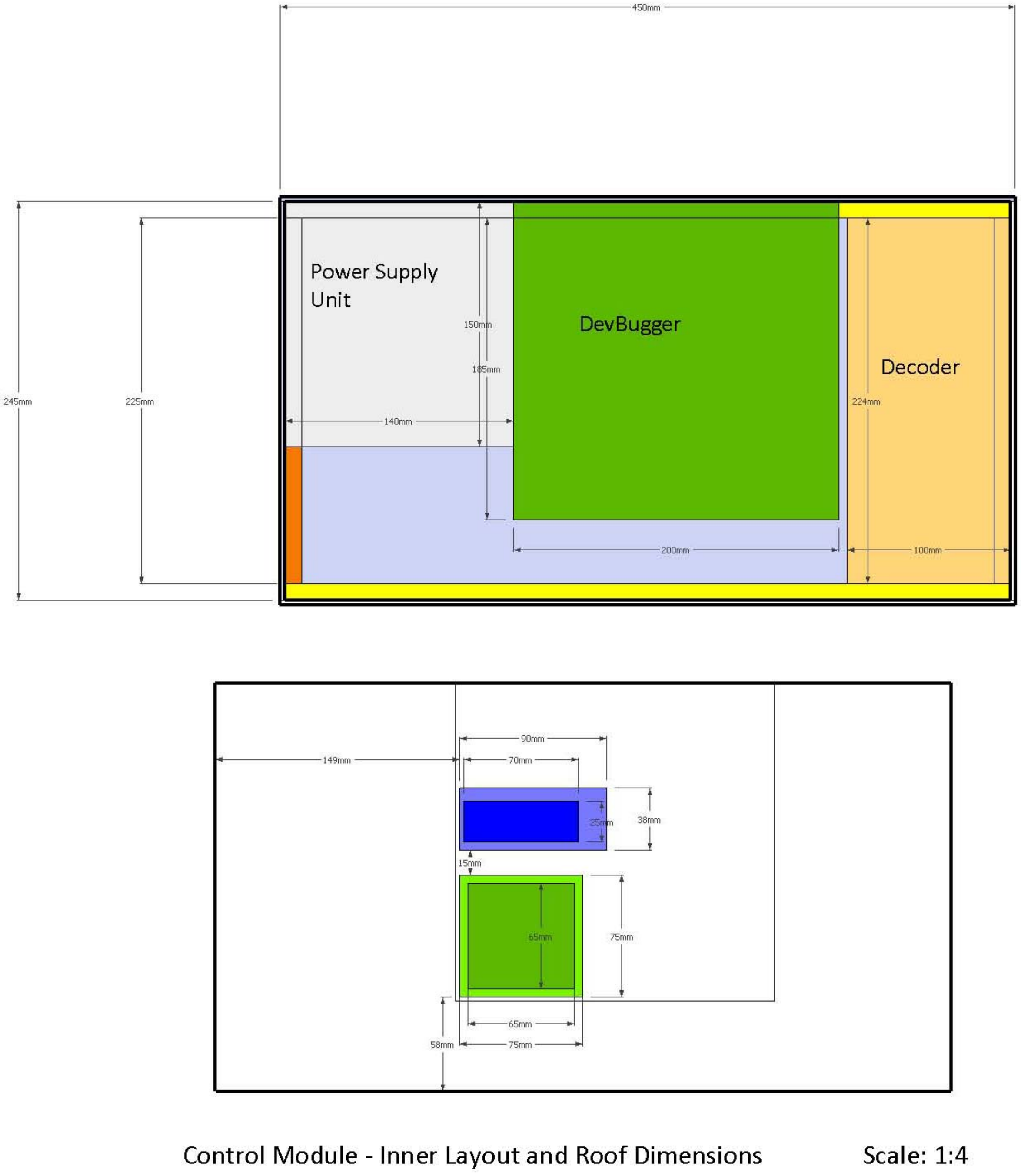


Figure E.5: Detailed overhead view of the interior (above) and exterior (below) of control module.

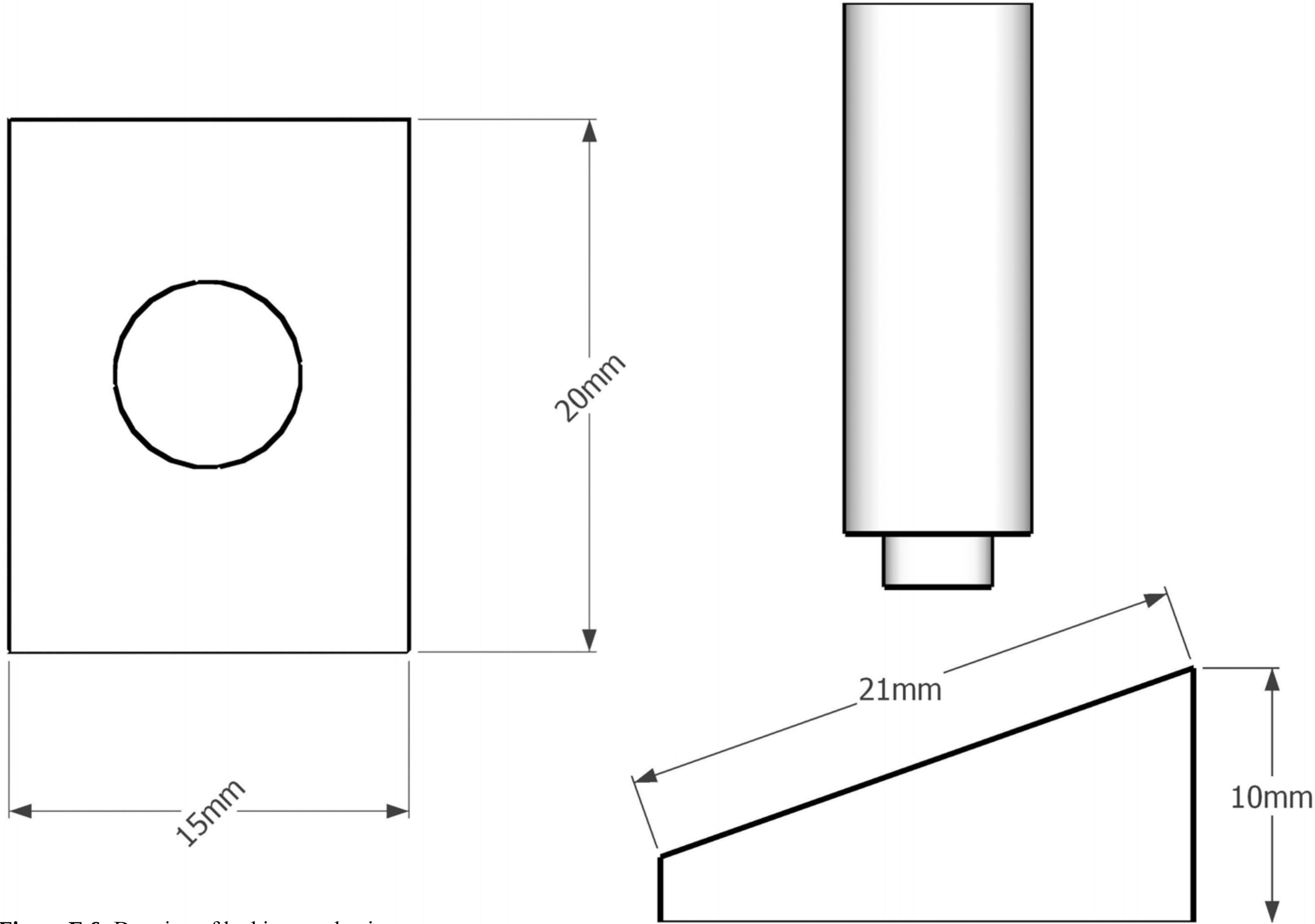


Figure E.6: Drawing of locking mechanism.

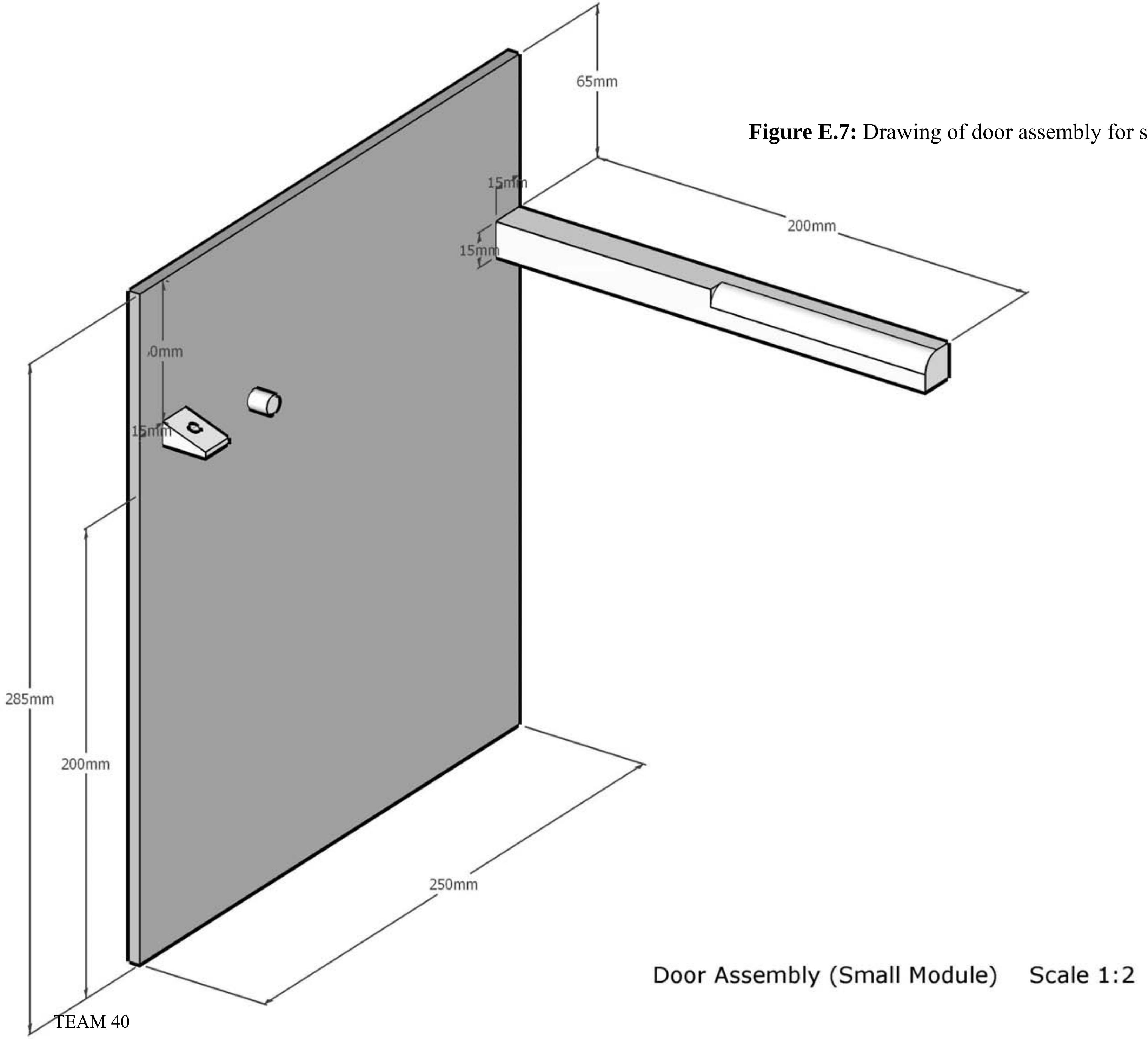


Figure E.7: Drawing of door assembly for small module.

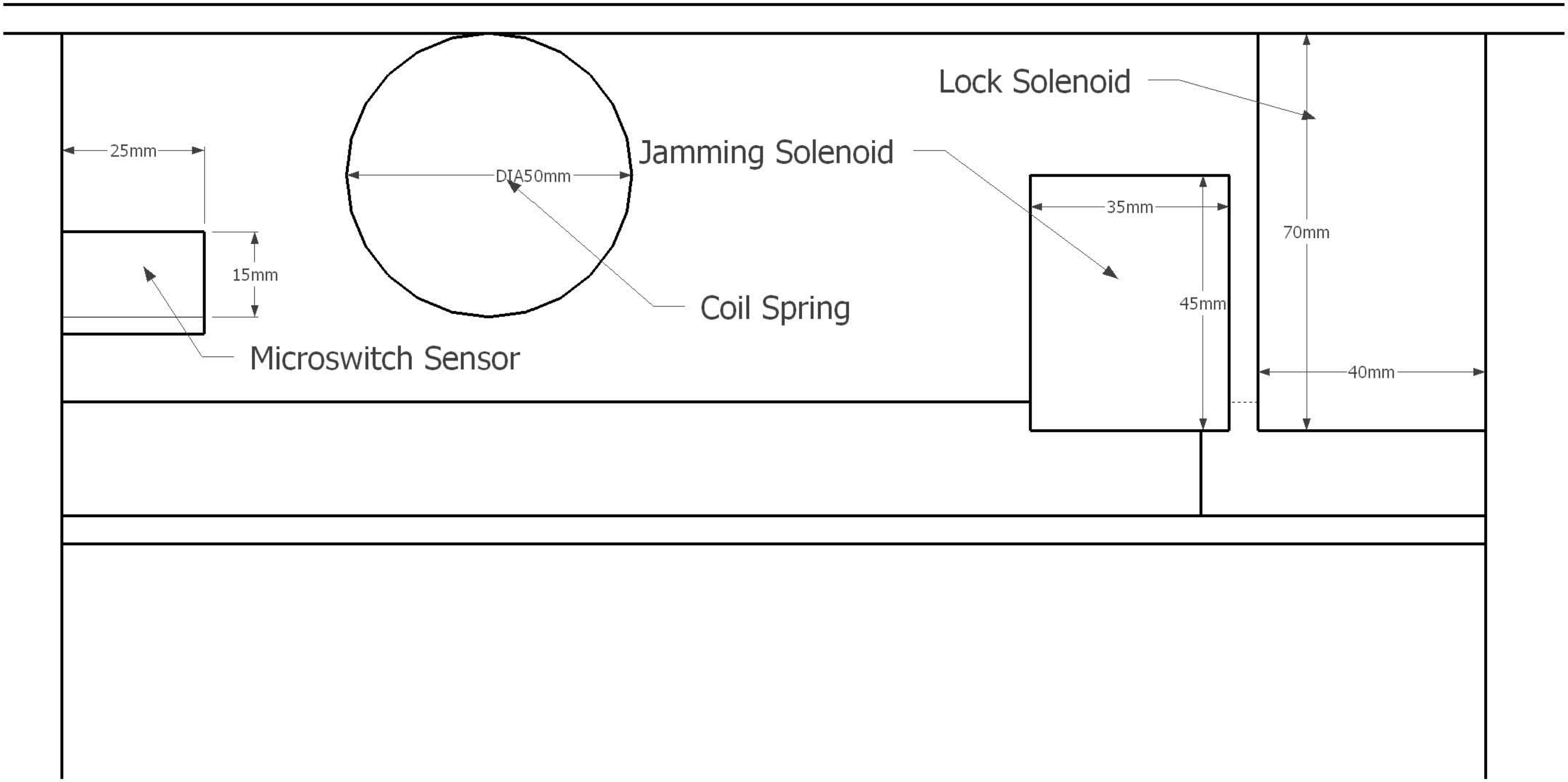


Figure E.8: Cutaway view of mechanisms behind aluminum face (from small module).

Table E.1: Dimensional and weight attributes of modules.

Module	Weight (g)	Inner Dimensions (mm)			Outer Dimensions (mm)		
		Height	Width	Depth	Height	Width	Depth
Small	1632	200	250	200	300	450	225
Medium	1988	350	300	200	450	450	225
Large	1985	500	400	200	600	450	225

Table E.2: Weight of Individual Components

Component	Mass/Density
Hardboard	$2.5x10^{-3}g/mm^2$
Cardboard	$1.0x10^{-3}g/mm^2$
Aluminum	$0.8x10^{-3}g/mm^2$
Lock solenoid (after pin modification)	100 g
Jamming solenoid	36 g
Jamming arm	15 g
Spring	25 g
Door handle	15 g
Hinge	50 g (small), 100g (large)
Screws and rivets	15g (small), 25g (medium), 40g (large)
Tabs	20g (small), 30g (large)
Magnets	20g (small), 30g (large)

Table E.3: Breakdown of Modules by Weight

Component	**Surface Area (mm^2)**	**Material**	**Unit Mass (g)**	**Total Mass (g)**
SMALL MODULE				
Outer Wall (x2)	64500	Cardboard Aluminum	116.1 (x2)	232.2
Inner Wall (x2)	64500	Cardboard	64.5 (x2)	129
Back Wall	135000	Cardboard	135	135
Outer Roof	86000	Cardboard	86	86
Front Panel	37500 (x2)	Cardboard Aluminum	60.8 (x2)	121.6
Front Door	75000	Hardboard	187.5	187.5
Inner Roof	50000	Cardboard	50	50
Floor	86000	Hardboard	215	215
Overall Mass (without hardware):				**1156.3 g**
MEDIUM MODULE				
Outer Wall (x2)	96750	Cardboard Aluminum	174.5 (x2)	348.3
Inner Wall (x2)	96750	Cardboard	96.8 (x2)	193.6
Back Wall	202500	Cardboard	202.5	202.5
Outer Roof	86000	Cardboard	86	86
Front Panel	33750 (x2)	Cardboard Aluminum	60.8 (x2)	121.6
Front Door	157500	Hardboard	392.5	392.5
Inner Roof	70000	Cardboard	70	70
Floor	86000	Hardboard	215	215
Overall Mass (without hardware):				**1602.5 g**
LARGE MODULE				
Outer Wall (x2)	129000	Aluminum	103.2 (x2)	206.4
Inner Wall (x2)	129000	Cardboard	129 (x2)	258
Back Wall	270000	Cardboard	270	270
Outer Roof	86000	Cardboard	86	86
Front Panel	15000 (x2)	Aluminum	12 (x2)	24
Front Door	70000	Hardboard (frame)	175	175
	240000	Aluminum (covering)	192	192
Inner Roof	80000	Cardboard	80	80
Floor	86000	Hardboard	215	215
Overall Mass (without hardware):				**1506.4 g**

Item E.1: Force Calculation for Door Retractor

The calculation for the maximum allowable tension in the spring is based on the moment of inertia of the doors about their hinge. This is calculated as follows:

(1) $$I = \int r^2\, dm$$

(2) $$I = \int \rho t h x^2 dx \qquad \text{(evaluated between x = 0 and x = w)}$$

where ρ is the density of the door, t is the thickness, w is the width, h is the height and x is the distance from the hinge.. With these values, we are then able to find the torque required from the spring to close the door within three seconds (at least π/6 radians per second in three seconds, or π/12 rad/s^2). Hence, we can find the force required from the rubber ties (and the force required by the user to keep the door open) by:

(3) $$\tau = I\alpha = Fr$$

Given that the density of hardboard is approximately 0.5g/cm^3 [5], the thickness is 2 mm, the dimensions of the biggest door are 60cm (H) x 40cm (W) x 30 cm (D), and the distance of spring attachment is approximately 5 cm from the hinge, we can estimate the force required to be the following:

$$\begin{aligned} I &= \tfrac{1}{3}(\rho t) * hw^3 \\ &= \tfrac{1}{3}(0.5*0.2)*60*40^3 \\ &= 128000 \text{ g*cm}^2 \end{aligned}$$

$$\begin{aligned} F &= I\alpha/r = 128000*(\pi/12)/5 \\ &= 6702 \text{ g*cm/s}^2 \\ &= 0.07 \text{ N} \end{aligned}$$

Therefore, the force required is 0.07 N, which can be handled by the spring, which exerts a uniform force of approximately 0.35N. This allows the door to close within 0.6 seconds.

APPENDIX F: CIRCUITS SUPPLEMENT

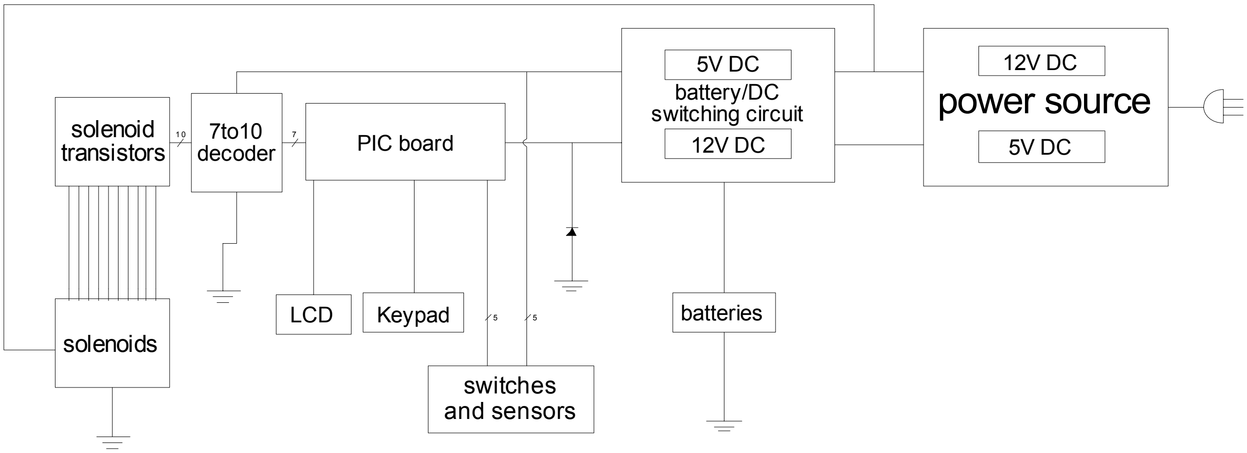


Figure F.1: Overview of Circuit Subsystem

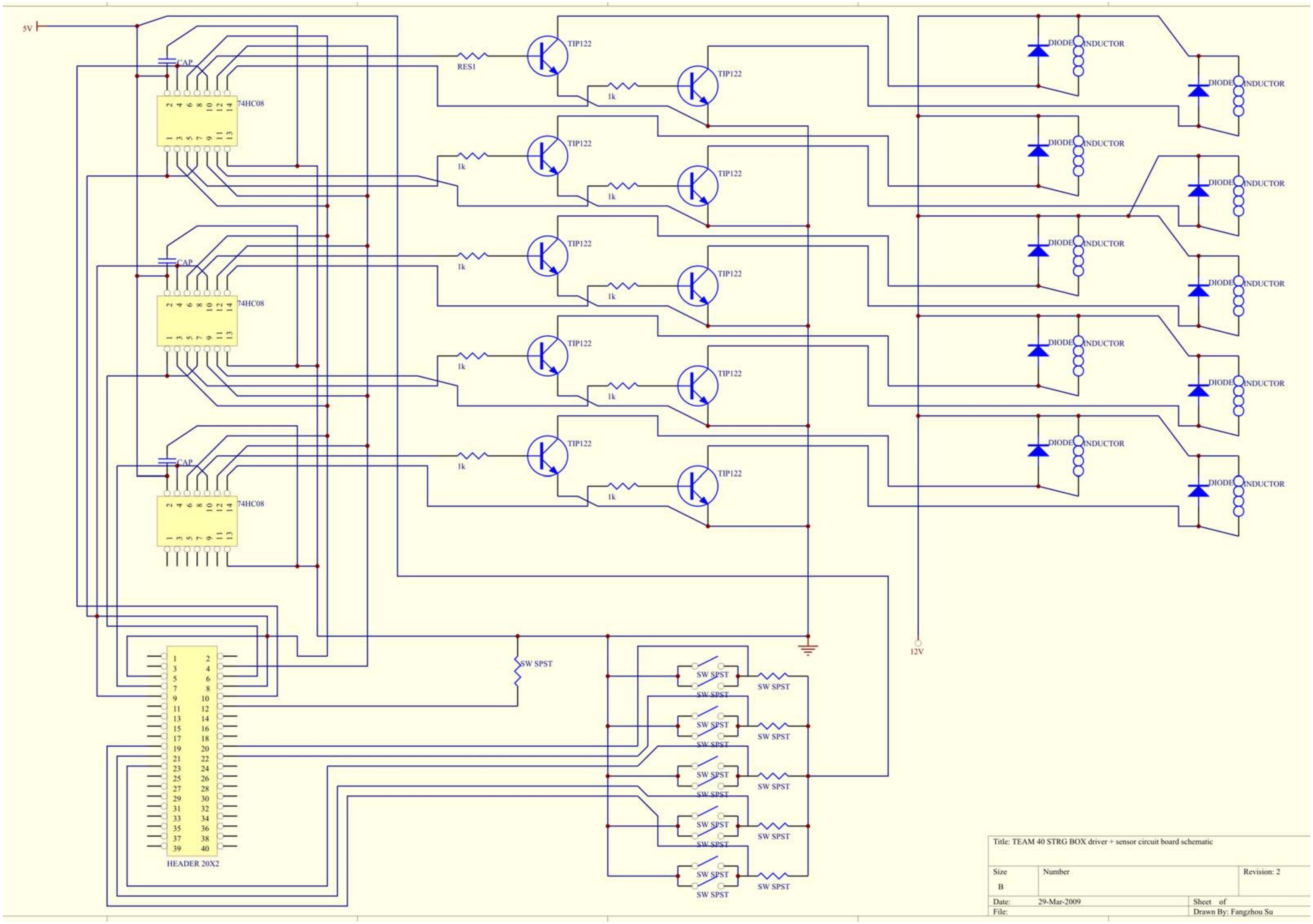


Figure F.2: Detailed schematic of circuit subsystem

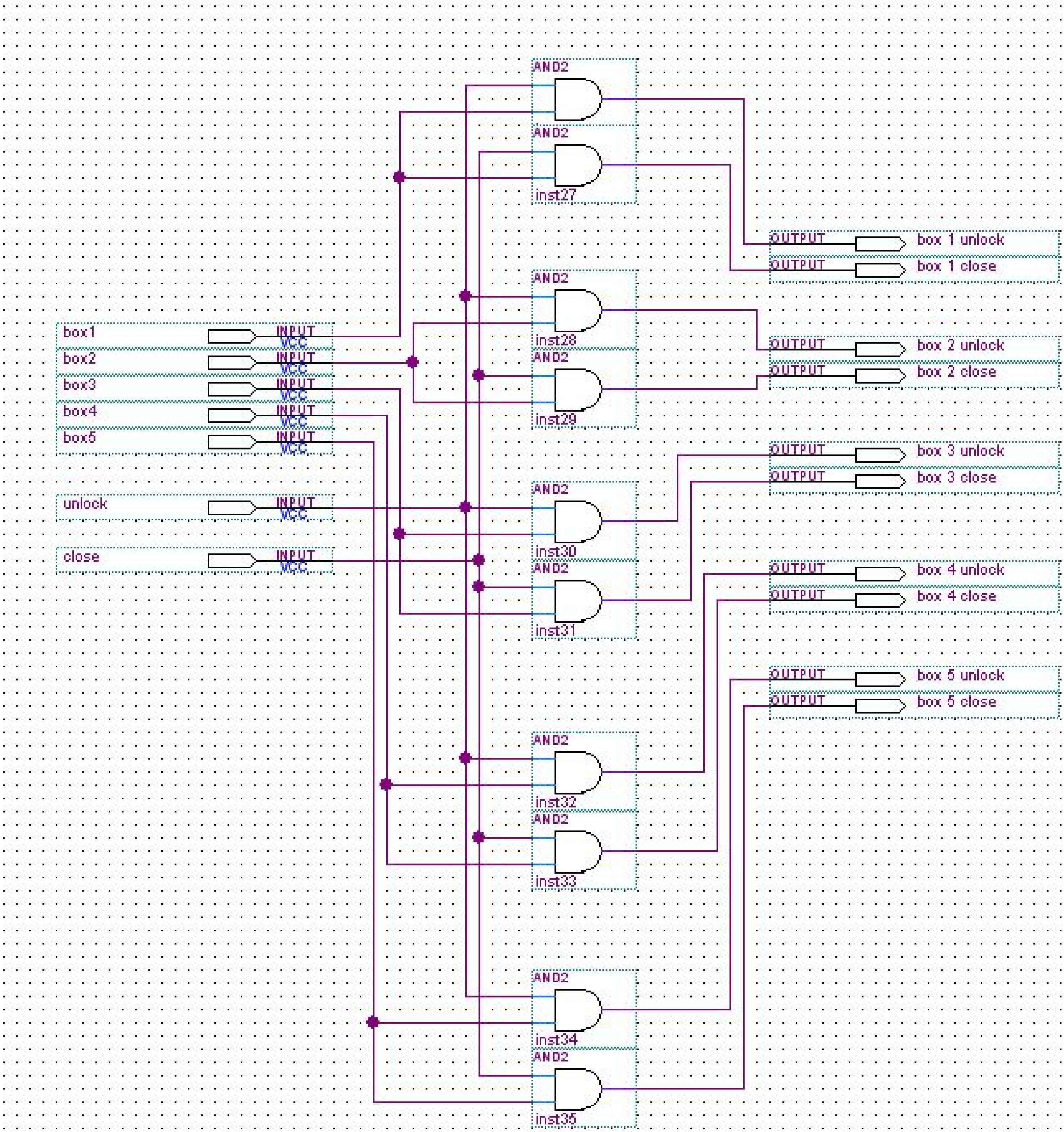


Figure F.3: Circuit schematic of multiplexer circuit.

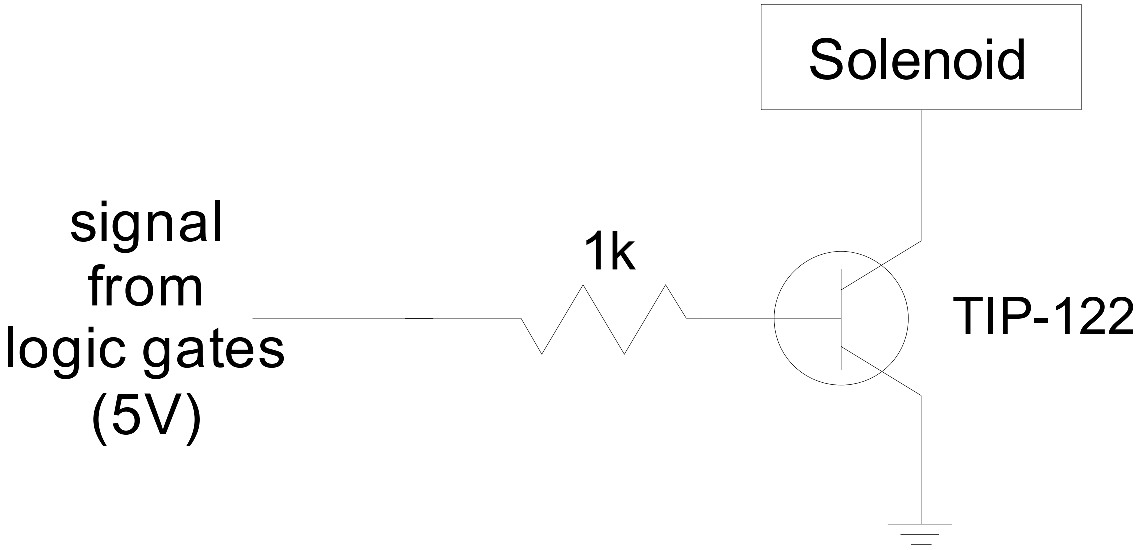


Figure F.4: Circuit schematic of transistor circuit.

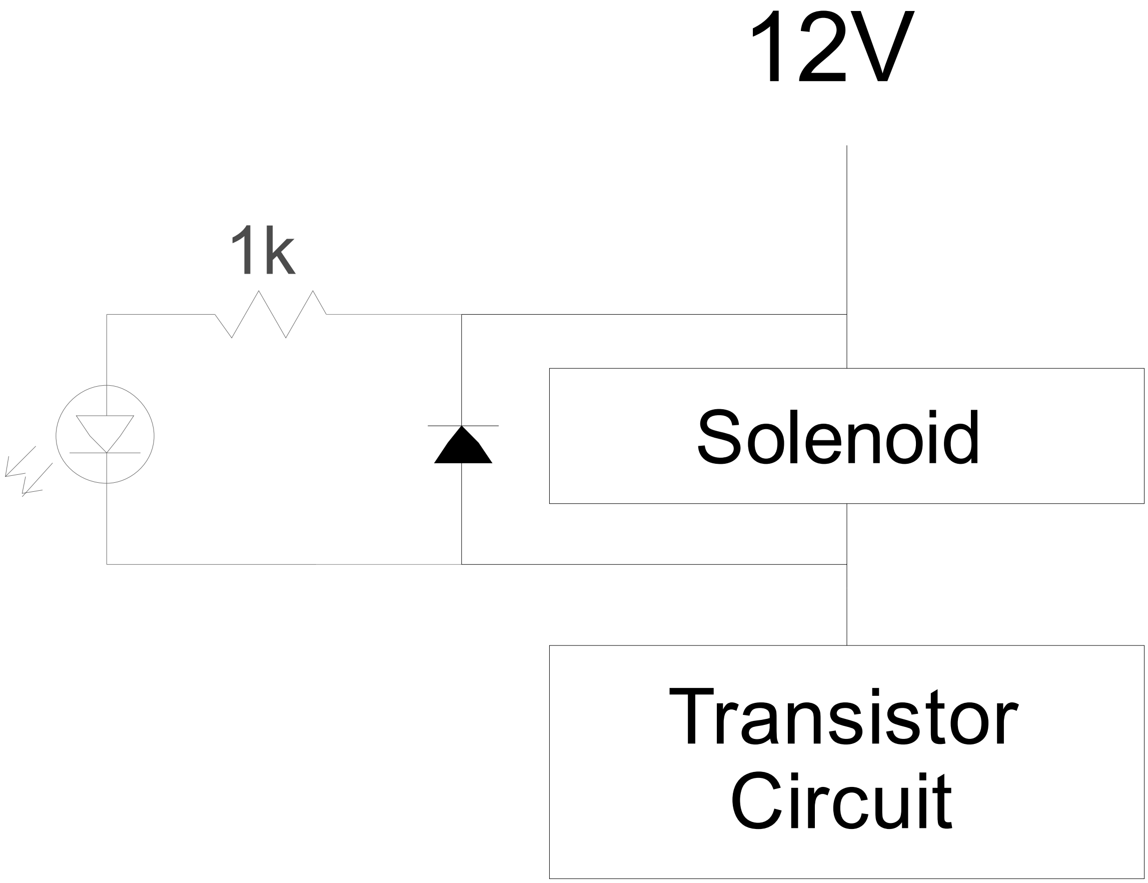


Figure F.5: Circuit schematic of solenoid circuit.

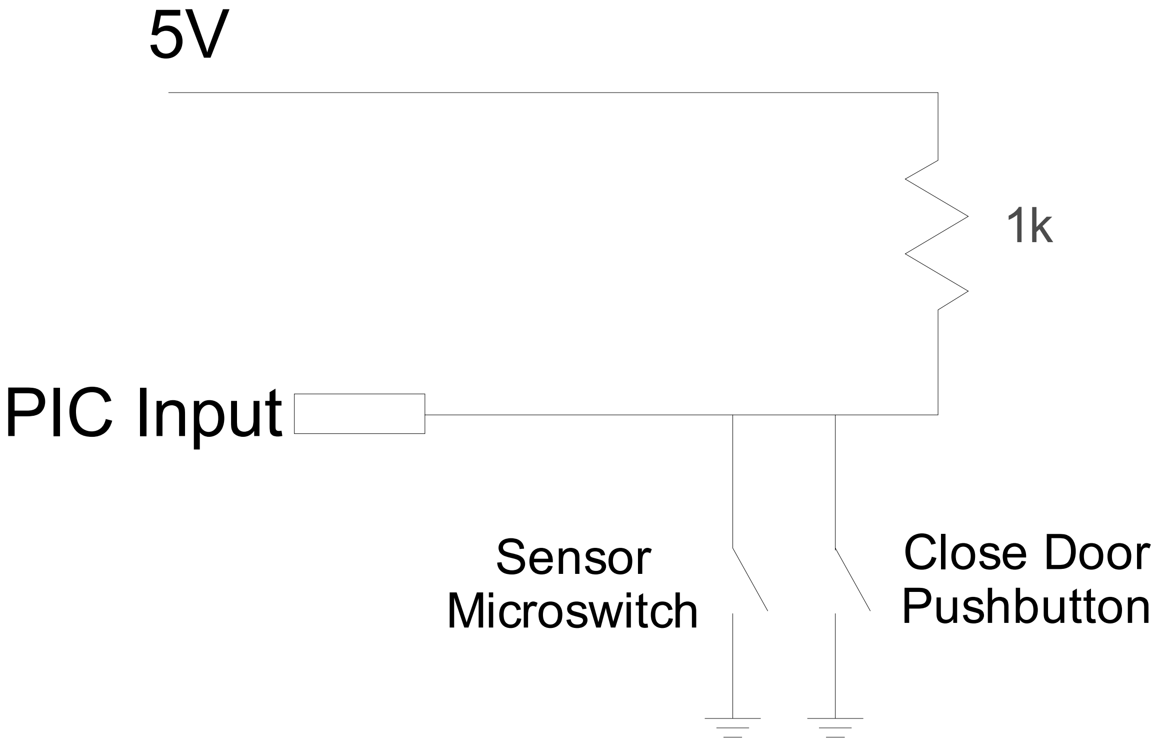


Figure F.6: Circuit schematic of sensor circuit.

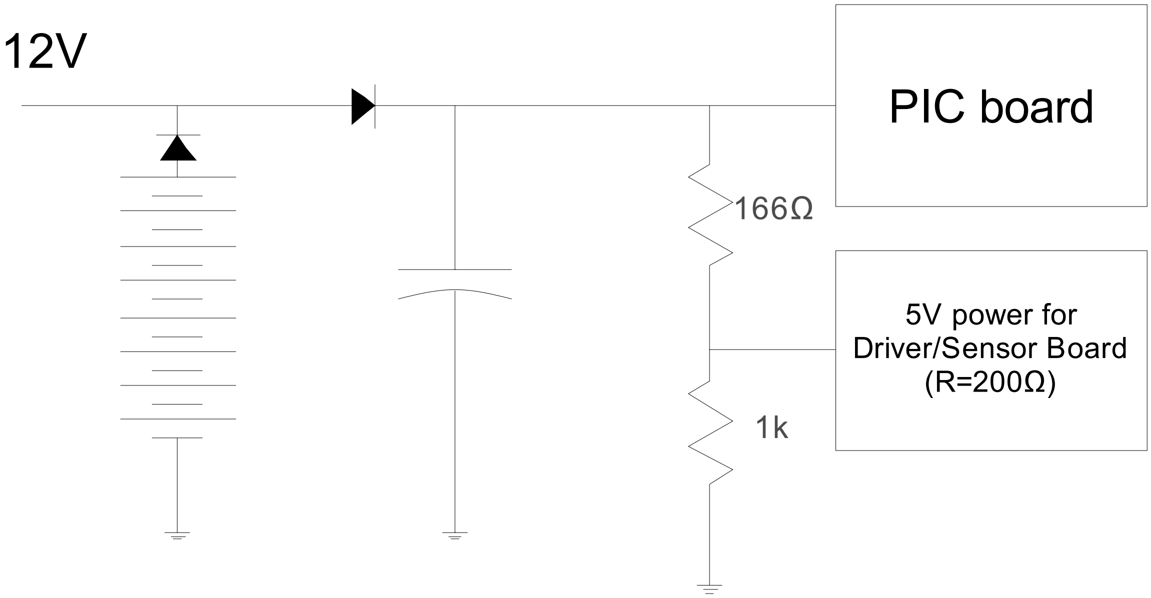
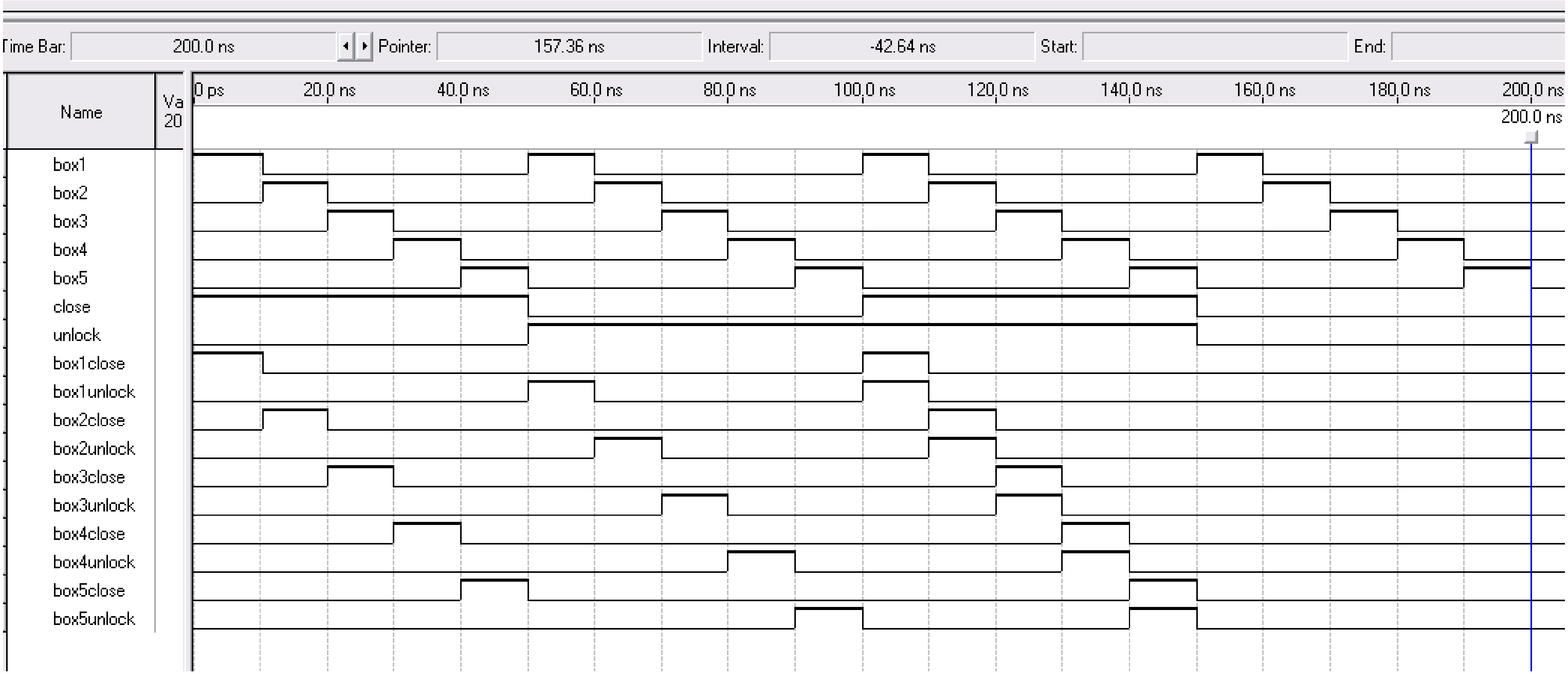


Figure F.7: Back-Up Battery Switching Circuit.



Time Bar: 200.0 ns

Pointer: 157.36 ns

Interval: -42.64 ns

Start:

End:

Item F.1: Simulated Results from Transistor Circuit

Item F.2: Power Consumption of System

The power consumption of the entire system is the sum of the power consumption of each component.

Lock Solenoids - The internal resistance of the solenoid was measured with a multimeter. Its value is 8Ω.

$$P = \frac{V^2}{R} = \frac{12V^2}{8\Omega} = 18W$$

Jam Solenoids - The internal resistance of the solenoid was measured with a multimeter. Its value is 44Ω.

$$P = \frac{V^2}{R} = \frac{12V^2}{44\Omega} = 3.3W$$

PIC DevBugger Board - The power consumption rating is taken from the power rating for the voltage regulator that was used on the board. It is rated for 1A.

$$P = VI = 12V \times 1A = 12W$$

AND-Gate chips - The 74HC08 quad 2-input AND gate chip takes 0.02A

$$P = VI = 5V \times 0.02A = 0.1W$$

Sensors - The sensor circuit has an internal resistance of 1kΩ, and is subjected to 5V at all times

$$P = \frac{V^2}{R} = \frac{5V^2}{1000} = 0.025W$$

LED Indicators - The LED's are connected to a 12V supply with a 1kΩ resistor in series.

$$P = \frac{V^2}{R} = \frac{12V^2}{1000} = 0.144W$$

Under normal circumstances, only 1 lock and 1 jam solenoid will be activated, so only 1 solenoid will be taken into consideration when calculating power consumption.

$$total\ power\ consumption = \sum Power\ for\ each\ component$$

$$total\ power\ consumption = 12 + 18 + 3.3 + 0.1 + 0.125 + 0.144$$

$$total\ power\ consumption = 35\ W$$

Table F.1: Power Consumption of Circuit Components

Component	Voltage (V)	Current (A)	Power (W)
Lock Solenoids	12	1.5	18
Jam Solenoids	12	0.3	3.3
PIC DevBugger board	12	1	12
AND-Gate chips x 3	5	0.02	0.1 x 3
Sensors x 5	5	0.005	0.025 x 5
LED x 5	12	0.012	0.144

APPENDIX G: MICROCONTROLLER SUPPLEMENT

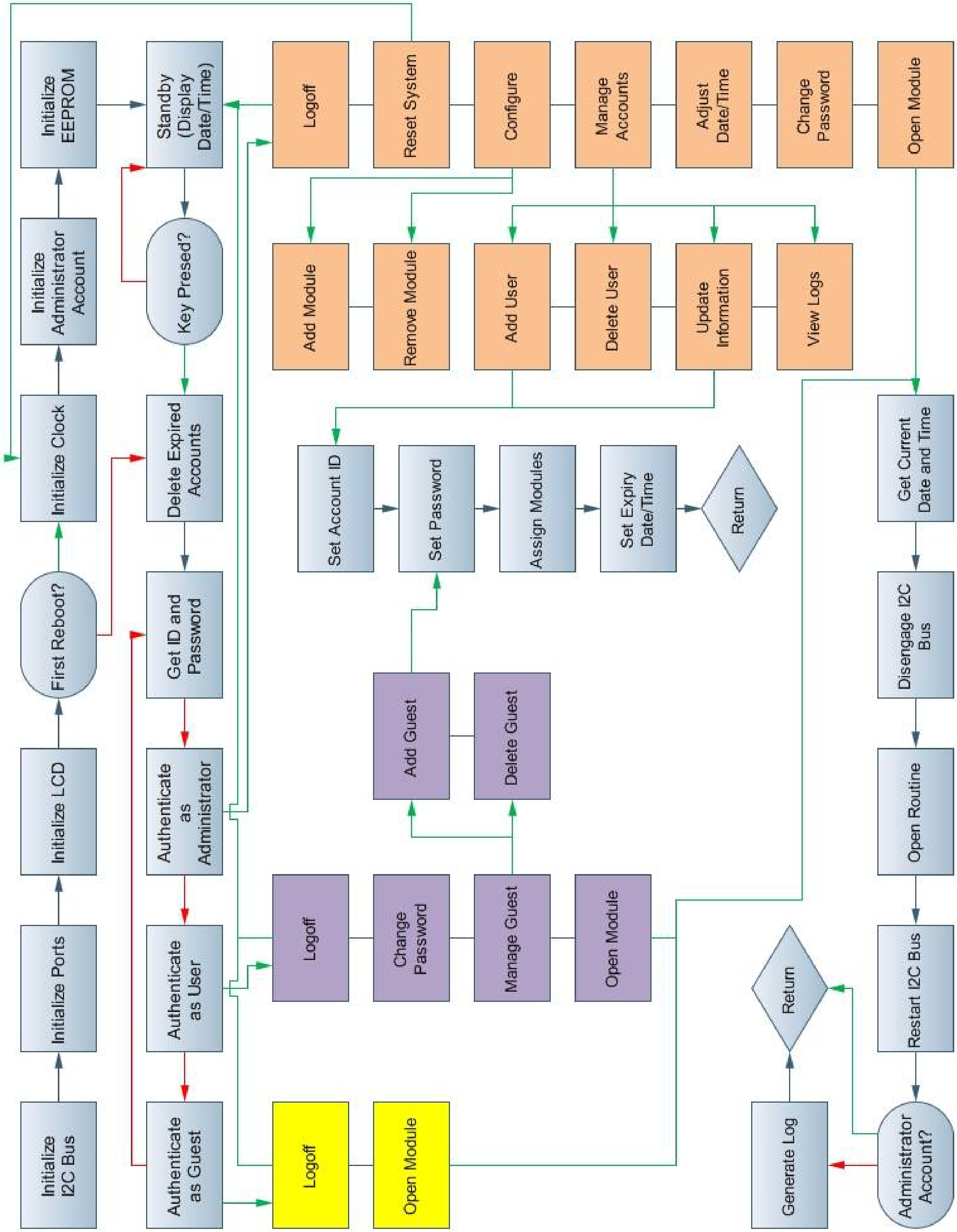


Figure G.1: Program algorithm and operator interface.

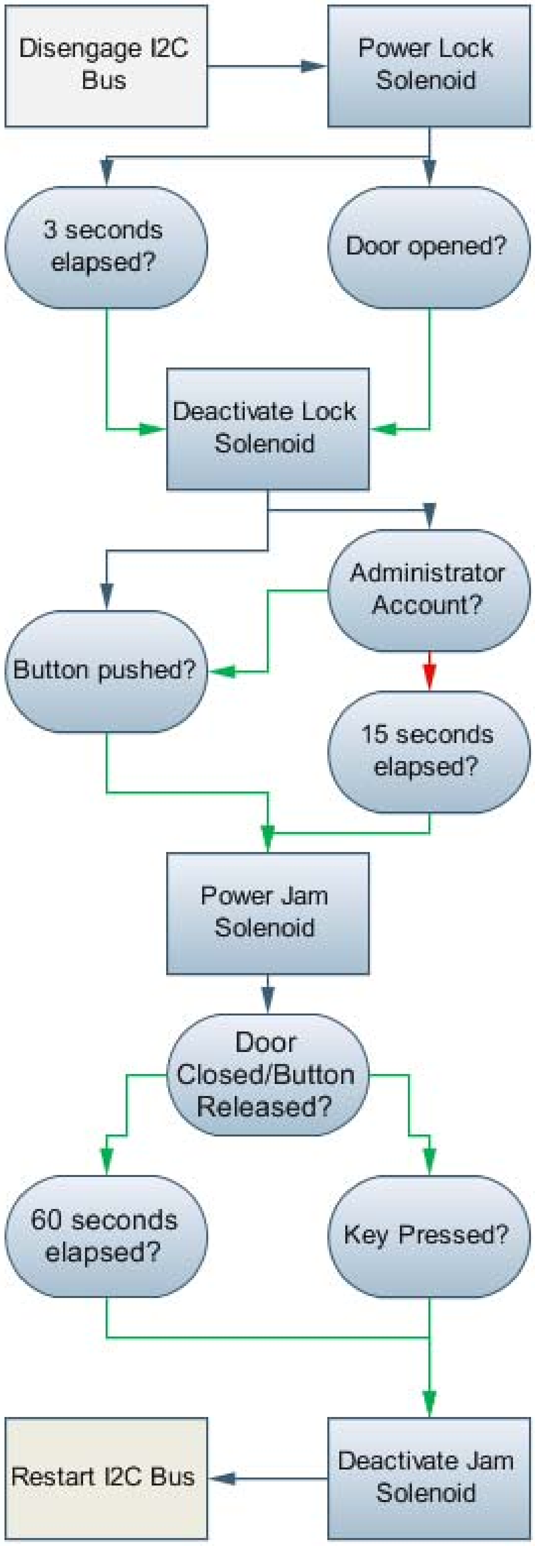
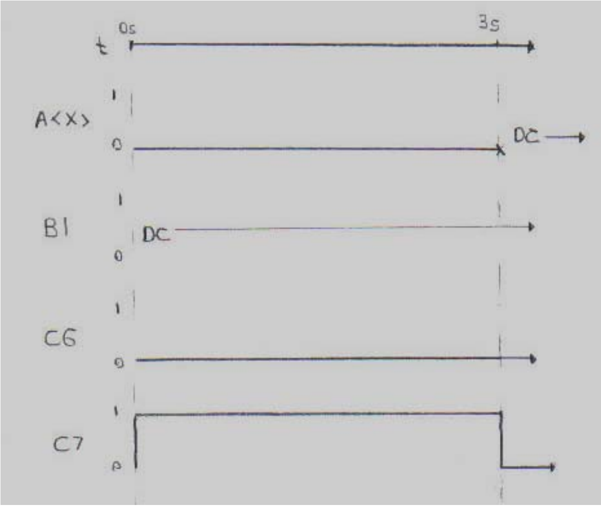


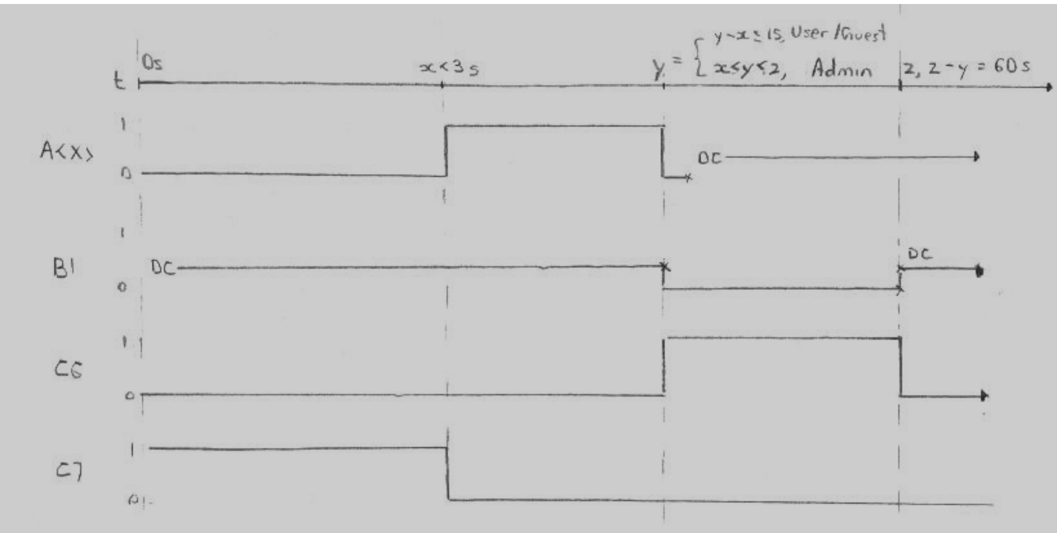
Figure G.2: Flowchart of algorithm for machine interface.

Figure G.3: Expected waveforms from input and corresponding output signals.



Case 1 (top): Operator doesn't open door within three second interval

Case 2 (middle): Operator opens door within 3 seconds, closes door within 15 seconds (user/guest), or pushes button (user/guest or admin) and acknowledges door has closed on keypad within 60 seconds.



Case 3 (bottom): Operator opens door within 3 seconds, closes door within 15 seconds (user/guest), or pushes button (user/guest or admin) but doesn't acknowledge door has closed on keypad within 60 seconds.

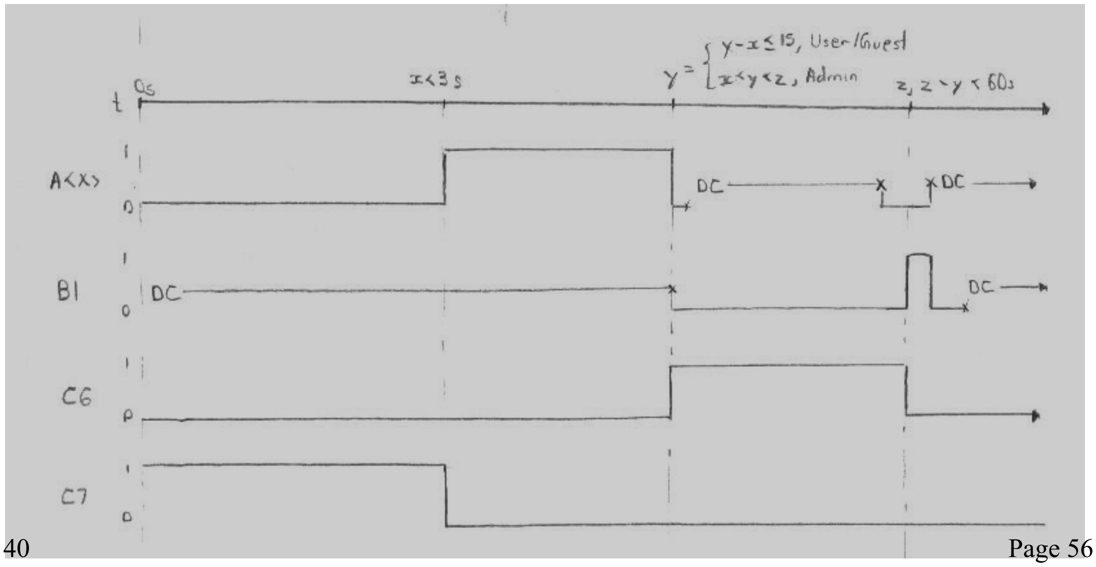


Table G.1: Break-up of functionalities by operator type.

Operator Type	Capabilities	
Administrator	1. Configure Modules 2. Add User 3. Edit User Information 4. Delete Users 5. View Logs	6. Open Modules 7. Adjust Date/Time 8. Change Password 9. Reset System
User	1. Open Modules 2. Add Guest	3. Delete Guest
Guest	1. Open Module	

Table G.2: Pin Assignments on PIC Microcontroller

Pin	I/O	Application
A<0>	I	Detects AC or battery supplied power 0 - Battery Power 1 - 12 V DC from AC power supply
A<1:5>	I	A<X> detects microswitch and pushbutton input for Module X 0 – Button pushed or microswitch closed 1 – Button not pushed and microswitch open
B<0>	-	Unused
B<1>	I	Detects keypad interrupt 0 – Interrupt cleared 1 – Interrupt enabled
B<2:3>	-	Unused
B<7:4>	I	4-bit input based on MM74C922 hex encoder for keypad (see datasheet)
C<0>	-	Unused
C<1:5>	O	C<X> signals module X is being used 0 – Module X not being used 1 – Module X being used
C<6>	O	Signal for door jamming solenoid 0 – Door jamming solenoid powered (lifted) 1 – Door jamming solenoid not powered (jamming)
C<7>	O	Signal for lock solenoid 0 – Lock solenoid powered (lifted, not locked) 1 – Lock solenoid not powered (dropped)
D<0:1>	-	Unused
D<2>	O	Register Select (RS) for HD44780 based LCD display 0 – Instruction mode 1 – Read/write mode
D<3>	O	Enable bit for HD44780 LCD display (pulse low for transmission) 0 – Low 1 – High
D<7:4>	O	4-bit output based on HD44780 8x5 pixel encoder (see datasheet)
E<0:2>	-	Unused

Table G.3: EEPROM Partition

Address	Data	Address	Data
0	Admin ID (first 2 letters)	128	User 1 Log 5 Year
1	Admin ID (last 2 letters)	129	User 1 Log 5 Hour
2	Admin PW (first 2 letters)	130	User 1 Log 5 Minute
3	Admin PW (last 2 letters)	131	User 1 Log 5 Module Address
4	Admin Access Byte	132	User 1 Log 5 Elapsed Time
5	Module Configuration Byte	133	User 2 Log 1 Month
6	Module 1 ID (first 2 letters)	134	User 2 Log 1 Date
7	Module 1 ID (last 2 letters)	135	User 2 Log 1 Year
8	Module 2 ID (first 2 letters)	136	User 2 Log 1 Hour
9	Module 2 ID (last 2 letters)	137	User 2 Log 1 Minute
10	Module 3 ID (first 2 letters)	138	User 2 Log 1 Module Address
11	Module 3 ID (last 2 letters)	139	User 2 Log 1 Elapsed Time
12	Module 4 ID (first 2 letters)	140	User 2 Log 2 Month
13	Module 4 ID (last 2 letters)	141	User 2 Log 2 Date
14	Module 5 ID (first 2 letters)	142	User 2 Log 2 Year
15	Module 5 ID (last 2 letters)	143	User 2 Log 2 Hour
16	Active User Configuration Byte	144	User 2 Log 2 Minute
17	Active Guest Configuration Byte	145	User 2 Log 2 Module Address
18	User 1 ID (first 2 letters)	146	User 2 Log 2 Elapsed Time
19	User 1 ID (last 2 letters)	147	User 2 Log 3 Month
20	User 1 PW (first 2 letters)	148	User 2 Log 3 Date
21	User 1 PW (last 2 letters)	149	User 2 Log 3 Year
22	User 1 Access Byte	150	User 2 Log 3 Hour
23	User 1 Expiry Month	151	User 2 Log 3 Minute
24	User 1 Expiry Date	152	User 2 Log 3 Module Address
25	User 1 Expiry Year	153	User 2 Log 3 Elapsed Time
26	User 1 Expiry Hour	154	User 2 Log 4 Month
27	User 1 Expiry Minute	155	User 2 Log 4 Date
28	User 1 Expiry Second	156	User 2 Log 4 Year
29	Guest 1 PW (first 2 letters)	157	User 2 Log 4 Hour
30	Guest 1 PW (last 2 letters)	158	User 2 Log 4 Minute
31	Guest 1 Access Byte	159	User 2 Log 4 Module Address
32	Guest 1 Expiry Month	160	User 2 Log 4 Elapsed Time
33	Guest 1 Expiry Date	161	User 2 Log 5 Month
34	Guest 1 Expiry Year	162	User 2 Log 5 Date
35	Guest 1 Expiry Hour	163	User 2 Log 5 Year
36	Guest 1 Expiry Minute	164	User 2 Log 5 Hour
37	Guest 1 Expiry Second	165	User 2 Log 5 Minute
38	User 2 ID (first 2 letters)	166	User 2 Log 5 Module Address
39	User 2 ID (last 2 letters)	167	User 2 Log 5 Elapsed Time
40	User 2 PW (first 2 letters)	168	User 3 Log 1 Month
41	User 2 PW (last 2 letters)	169	User 3 Log 1 Date

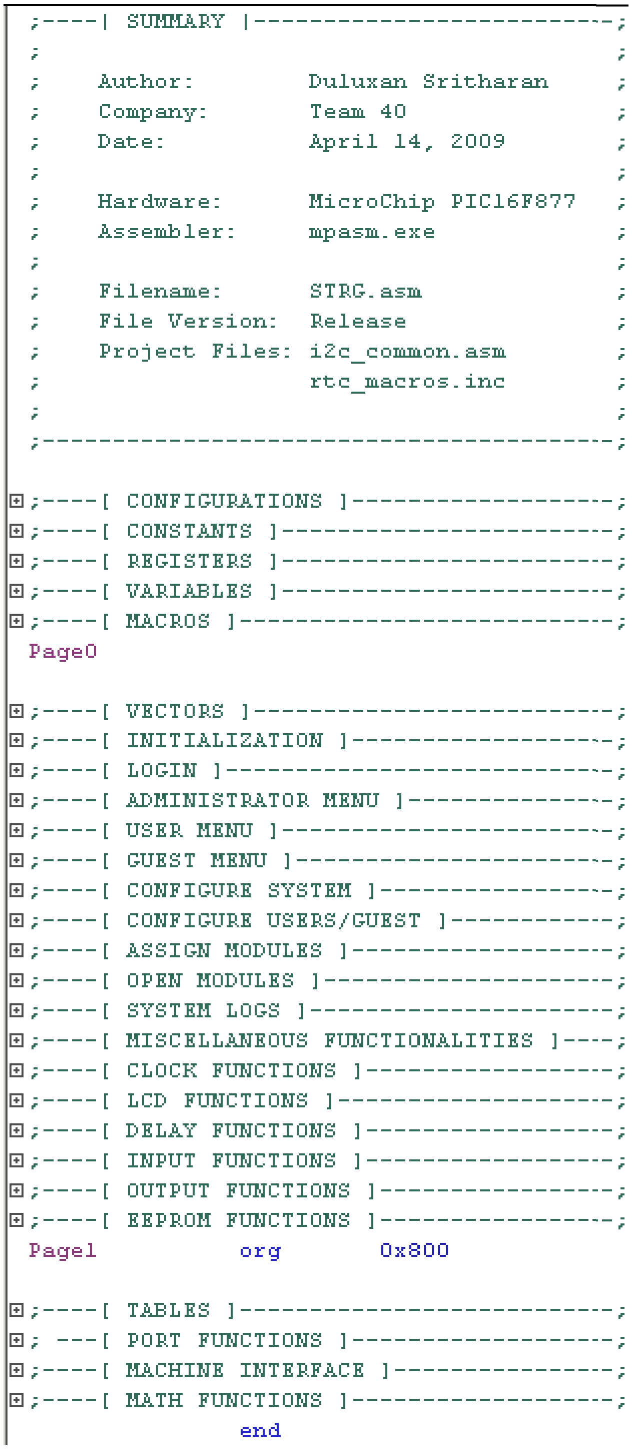
42	User 2 Access Byte	170	User 3 Log 1 Year
43	User 2 Expiry Month	171	User 3 Log 1 Hour
44	User 2 Expiry Date	172	User 3 Log 1 Minute
45	User 2 Expiry Year	173	User 3 Log 1 Module Address
46	User 2 Expiry Hour	174	User 3 Log 1 Elapsed Time
47	User 2 Expiry Minute	175	User 3 Log 2 Month
48	User 2 Expiry Second	176	User 3 Log 2 Date
49	Guest 2 PW (first 2 letters)	177	User 3 Log 2 Year
50	Guest 2 PW (last 2 letters)	178	User 3 Log 2 Hour
51	Guest 2 Access Byte	179	User 3 Log 2 Minute
52	Guest 2 Expiry Month	180	User 3 Log 2 Module Address
53	Guest 2 Expiry Date	181	User 3 Log 2 Elapsed Time
54	Guest 2 Expiry Year	182	User 3 Log 3 Month
55	Guest 2 Expiry Hour	183	User 3 Log 3 Date
56	Guest 2 Expiry Minute	184	User 3 Log 3 Year
57	Guest 2 Expiry Second	185	User 3 Log 3 Hour
58	User 3 ID (first 2 letters)	186	User 3 Log 3 Minute
59	User 3 ID (last 2 letters)	187	User 3 Log 3 Module Address
60	User 3 PW (first 2 letters)	188	User 3 Log 3 Elapsed Time
61	User 3 PW (last 2 letters)	189	User 3 Log 4 Month
62	User 3 Access Byte	190	User 3 Log 4 Date
63	User 3 Expiry Month	191	User 3 Log 4 Year
64	User 3 Expiry Date	192	User 3 Log 4 Hour
65	User 3 Expiry Year	193	User 3 Log 4 Minute
66	User 3 Expiry Hour	194	User 3 Log 4 Module Address
67	User 3 Expiry Minute	195	User 3 Log 4 Elapsed Time
68	User 3 Expiry Second	196	User 3 Log 5 Month
69	Guest 3 PW (first 2 letters)	197	User 3 Log 5 Date
70	Guest 3 PW (last 2 letters)	198	User 3 Log 5 Year
71	Guest 3 Access Byte	199	User 3 Log 5 Hour
72	Guest 3 Expiry Month	200	User 3 Log 5 Minute
73	Guest 3 Expiry Date	201	User 3 Log 5 Module Address
74	Guest 3 Expiry Year	202	User 3 Log 5 Elapsed Time
75	Guest 3 Expiry Hour	203	User 4 Log 1 Month
76	Guest 3 Expiry Minute	204	User 4 Log 1 Date
77	Guest 3 Expiry Second	205	User 4 Log 1 Year
78	User 4 ID (first 2 letters)	206	User 4 Log 1 Hour
79	User 4 ID (last 2 letters)	207	User 4 Log 1 Minute
80	User 4 PW (first 2 letters)	208	User 4 Log 1 Module Address
81	User 4 PW (last 2 letters)	209	User 4 Log 1 Elapsed Time
82	User 4 Access Byte	210	User 4 Log 2 Month
83	User 4 Expiry Month	211	User 4 Log 2 Date
84	User 4 Expiry Date	212	User 4 Log 2 Year
85	User 4 Expiry Year	213	User 4 Log 2 Hour
86	User 4 Expiry Hour	214	User 4 Log 2 Minute

87	User 4 Expiry Minute	215	User 4 Log 2 Module Address
88	User 4 Expiry Second	216	User 4 Log 2 Elapsed Time
89	Guest 4 PW (first 2 letters)	217	User 4 Log 3 Month
90	Guest 4 PW (last 2 letters)	218	User 4 Log 3 Date
91	Guest 4 Access Byte	219	User 4 Log 3 Year
92	Guest 4 Expiry Month	220	User 4 Log 3 Hour
93	Guest 4 Expiry Date	221	User 4 Log 3 Minute
94	Guest 4 Expiry Year	222	User 4 Log 3 Module Address
95	Guest 4 Expiry Hour	223	User 4 Log 3 Elapsed Time
96	Guest 4 Expiry Minute	224	User 4 Log 4 Month
97	Guest 4 Expiry Second	225	User 4 Log 4 Date
98	User 1 Log 1 Month	226	User 4 Log 4 Year
99	User 1 Log 1 Date	227	User 4 Log 4 Hour
100	User 1 Log 1 Year	228	User 4 Log 4 Minute
101	User 1 Log 1 Hour	229	User 4 Log 4 Module Address
102	User 1 Log 1 Minute	230	User 4 Log 4 Elapsed Time
103	User 1 Log 1 Module Address	231	User 4 Log 5 Month
104	User 1 Log 1 Elapsed Time	232	User 4 Log 5 Date
105	User 1 Log 2 Month	233	User 4 Log 5 Year
106	User 1 Log 2 Date	234	User 4 Log 5 Hour
107	User 1 Log 2 Year	235	User 4 Log 5 Minute
108	User 1 Log 2 Hour	236	User 4 Log 5 Module Address
109	User 1 Log 2 Minute	237	User 4 Log 5 Elapsed Time
110	User 1 Log 2 Module Address	238	-
111	User 1 Log 2 Elapsed Time	239	-
112	User 1 Log 3 Month	240	-
113	User 1 Log 3 Date	241	-
114	User 1 Log 3 Year	242	-
115	User 1 Log 3 Hour	243	-
116	User 1 Log 3 Minute	244	-
117	User 1 Log 3 Module Address	245	-
118	User 1 Log 3 Elapsed Time	246	-
119	User 1 Log 4 Month	247	-
120	User 1 Log 4 Date	248	-
121	User 1 Log 4 Year	249	-
122	User 1 Log 4 Hour	250	-
123	User 1 Log 4 Minute	251	-
124	User 1 Log 4 Module Address	252	-
125	User 1 Log 4 Elapsed Time	253	-
126	User 1 Log 5 Month	254	-
127	User 1 Log 5 Date	255	System Restart Byte

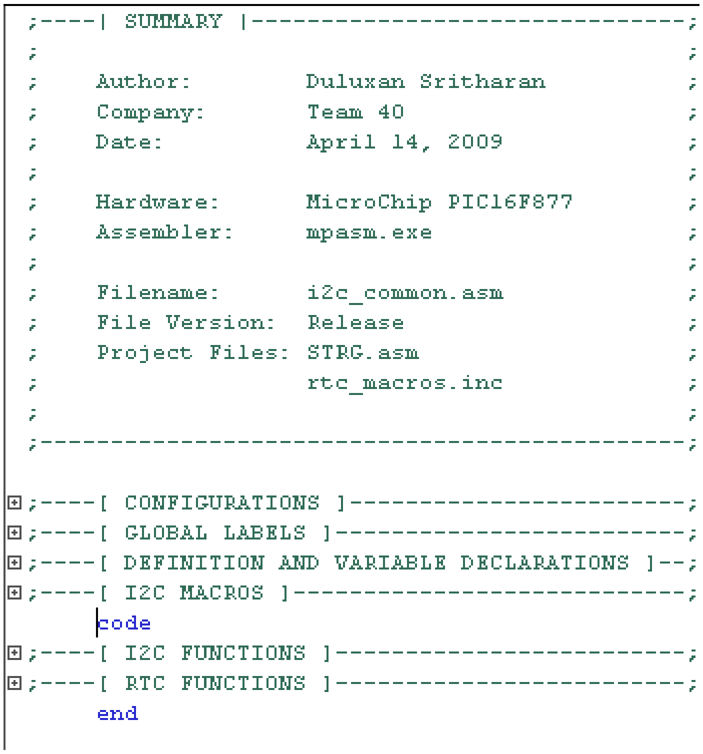
Table G.4: Special Purpose EEPROM Bytes

<table>
<tr><th>Address</th><th>Name</th><th colspan="8">Bits <7:0></th></tr>
<tr><td>4</td><td>xAdmAccess</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td colspan="10">xAdmAccess is a control register for the generation of logs and allowing the administrator to open all modules. This value should always be 255.</td></tr>
<tr><td>5</td><td>xModConfig</td><td>DC</td><td>DC</td><td>0/1</td><td>0/1</td><td>0/1</td><td>0/1</td><td>0/1</td><td>DC</td></tr>
<tr><td colspan="10">xModConfig is control register for storing which modules are configured. Module X is configured if bit X is 1 or else it is not if bit X is 0.</td></tr>
<tr><td>16</td><td>xUActive</td><td>DC</td><td>DC</td><td>DC</td><td>0/1</td><td>0/1</td><td>0/1</td><td>0/1</td><td>DC</td></tr>
<tr><td colspan="10">xUActive is a control register for storing which users are active. User X is active if bit X is 1 or else is inactive if bit X is 0.</td></tr>
<tr><td>17</td><td>xGActive</td><td>DC</td><td>Dc</td><td>DC</td><td>0/1</td><td>0/1</td><td>0/1</td><td>0/1</td><td>DC</td></tr>
<tr><td colspan="10">xGActive is a control register for storing which guest are active. Guest X is active if bit X is 1 or else is inactive if bit X is 0.</td></tr>
<tr><td>255</td><td>xRestart</td><td>0/1</td><td>0/1</td><td>0/1</td><td>0/1</td><td>0/1</td><td>0/1</td><td>0/1</td><td>0/1</td></tr>
<tr><td colspan="10">xRestart is a control register for system reboot. The default value is 255 during restart and is cleared to 0 during the first initialization.</td></tr>
</table>

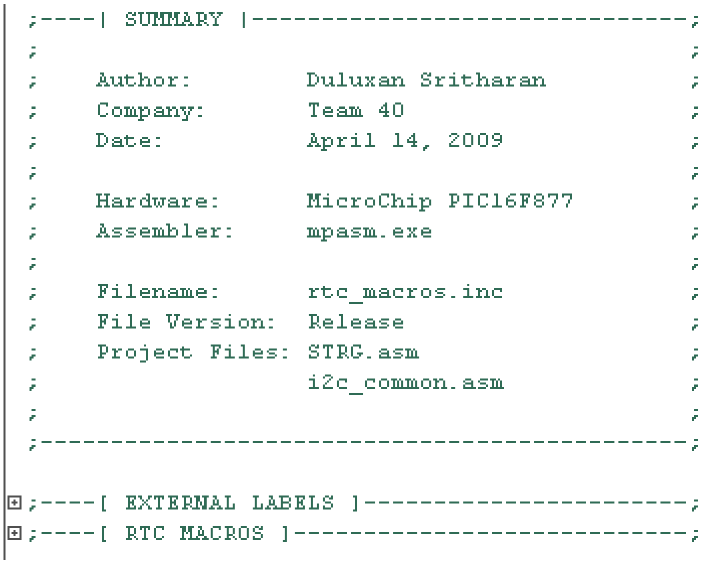
Item G.1: Overview of Code for System



```
 ;----| SUMMARY |---------------------------;
 ;                                          ;
 ;    Author:        Duluxan Sritharan      ;
 ;    Company:       Team 40                ;
 ;    Date:          April 14, 2009         ;
 ;                                          ;
 ;    Hardware:      MicroChip PIC16F877    ;
 ;    Assembler:     mpasm.exe              ;
 ;                                          ;
 ;    Filename:      STRG.asm               ;
 ;    File Version:  Release                ;
 ;    Project Files: i2c_common.asm         ;
 ;                   rtc_macros.inc         ;
 ;                                          ;
 ;------------------------------------------;

⊞;----[ CONFIGURATIONS ]--------------------;
⊞;----[ CONSTANTS ]-------------------------;
⊞;----[ REGISTERS ]-------------------------;
⊞;----[ VARIABLES ]-------------------------;
⊞;----[ MACROS ]----------------------------;
 Page0

⊞;----[ VECTORS ]---------------------------;
⊞;----[ INITIALIZATION ]--------------------;
⊞;----[ LOGIN ]-----------------------------;
⊞;----[ ADMINISTRATOR MENU ]----------------;
⊞;----[ USER MENU ]-------------------------;
⊞;----[ GUEST MENU ]------------------------;
⊞;----[ CONFIGURE SYSTEM ]------------------;
⊞;----[ CONFIGURE USERS/GUEST ]-------------;
⊞;----[ ASSIGN MODULES ]--------------------;
⊞;----[ OPEN MODULES ]----------------------;
⊞;----[ SYSTEM LOGS ]-----------------------;
⊞;----[ MISCELLANEOUS FUNCTIONALITIES ]-----;
⊞;----[ CLOCK FUNCTIONS ]-------------------;
⊞;----[ LCD FUNCTIONS ]---------------------;
⊞;----[ DELAY FUNCTIONS ]-------------------;
⊞;----[ INPUT FUNCTIONS ]-------------------;
⊞;----[ OUTPUT FUNCTIONS ]------------------;
⊞;----[ EEPROM FUNCTIONS ]------------------;
 Page1          org       0x800

⊞;----[ TABLES ]----------------------------;
⊞; ---[ PORT FUNCTIONS ]--------------------;
⊞;----[ MACHINE INTERFACE ]-----------------;
⊞;----[ MATH FUNCTIONS ]--------------------;
                end
```



```
 ;----| SUMMARY |------------------------------------;
 ;                                                  ;
 ;    Author:        Duluxan Sritharan              ;
 ;    Company:       Team 40                        ;
 ;    Date:          April 14, 2009                 ;
 ;                                                  ;
 ;    Hardware:      MicroChip PIC16F877            ;
 ;    Assembler:     mpasm.exe                      ;
 ;                                                  ;
 ;    Filename:      i2c_common.asm                 ;
 ;    File Version:  Release                        ;
 ;    Project Files: STRG.asm                       ;
 ;                   rtc_macros.inc                 ;
 ;                                                  ;
 ;--------------------------------------------------;

⊞;----[ CONFIGURATIONS ]----------------------------;
⊞;----[ GLOBAL LABELS ]-----------------------------;
⊞;----[ DEFINITION AND VARIABLE DECLARATIONS ]--;
⊞;----[ I2C MACROS ]--------------------------------;
        code
⊞;----[ I2C FUNCTIONS ]-----------------------------;
⊞;----[ RTC FUNCTIONS ]-----------------------------;
        end
```



```
 ;----| SUMMARY |------------------------------------;
 ;                                                  ;
 ;    Author:        Duluxan Sritharan              ;
 ;    Company:       Team 40                        ;
 ;    Date:          April 14, 2009                 ;
 ;                                                  ;
 ;    Hardware:      MicroChip PIC16F877            ;
 ;    Assembler:     mpasm.exe                      ;
 ;                                                  ;
 ;    Filename:      rtc_macros.inc                 ;
 ;    File Version:  Release                        ;
 ;    Project Files: STRG.asm                       ;
 ;                   i2c_common.asm                 ;
 ;                                                  ;
 ;--------------------------------------------------;

⊞;----[ EXTERNAL LABELS ]---------------------------;
⊞;----[ RTC MACROS ]--------------------------------;
```

Item G.2: Compendium of Code for Microcontroller Subsystem (see following pages).

```
;----| SUMMARY |--------------------------------------------------------------------------------;
;                                                                                               ;
;    Author:        Duluxan Sritharan                                                           ;
;    Company:       Team 40                                                                     ;
;    Date:          April 14, 2009                                                              ;
;                                                                                               ;
;    Hardware:      MicroChip PIC16F877                                                         ;
;    Assembler:     mpasm.exe                                                                   ;
;                                                                                               ;
;    Filename:      STRG.asm                                                                    ;
;    File Version:  Release                                                                     ;
;    Project Files: i2c_common.asm                                                              ;
;                   rtc_macros.inc                                                              ;
;                                                                                               ;
;-----------------------------------------------------------------------------------------------;

;----[ CONFIGURATIONS ]-------------------------------------------------------------------------;

;{
     __CONFIG  (_CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON & _HS_OSC & _WRT_ENABLE_ON & _CPD_OFF
               & _LVP_OFF & _DEBUG_OFF)
                                   ; set configuration register
     errorlevel -302               ; ignore bank switch warning
     list p=16f877, r = DEC        ; list directive to define processor

     #include <p16f877.inc>        ; processor specific variable definitions
     #include <rtc_macros.inc>     ; macros for real-time clock
;}


;----[ CONSTANTS ]------------------------------------------------------------------------------;

;{

; mnemonics for LCD bits (PORTD)

RS             equ            2
E              equ            3

; mnemonics for RTC addresses

RTC_Second     equ            0
RTC_Minute     equ            1
RTC_Hour       equ            2
RTC_Date       equ            4
RTC_Month      equ            5
RTC_Year       equ            6

; mnemonics for EEPROM addresses

xAdmID         equ            0
xAdmPW         equ            2
xAdmAccess     equ            4
xModConfig     equ            5
xMod1ID        equ            6
xMod2ID        equ            8
xMod3ID        equ            10
xMod4ID        equ            12
xMod5ID        equ            14
xUActive       equ            16
xGActive       equ            17

xUser1ID       equ            18
xUser1PW       equ            20
xUser1Access   equ            22
xUser1Valid    equ            23
xGuest1PW      equ            29
xGuest1Access  equ            31
xGuest1Valid   equ            32

xUser4ID       equ            78
xUser4PW       equ            80
```

```
xUser4Access   equ            82
xUser4Valid    equ            83
xGuest4PW      equ            89
xGuest4Access  equ            91
xGuest4Valid   equ            92

xUser1Log      equ            98
xUser4Log      equ            206

xRestart       equ            255

;}

;----[ REGISTERS ]------------------------------------------------------------------------------;

;{

clock          equ            0x75      ; address of stored binary clock value
dig10          equ            0x77      ; address of parsed ten's digit
dig1           equ            0x78      ; address of parsed one's digit
rtcAdr         equ            0x79      ; address of register for field
rtcVal         equ            0x7A      ; address of register for value

;}


;----[ VARIABLES ]------------------------------------------------------------------------------;

;{

     cblock H'20'

          ; variables for taking input from keypad

          num_check
          num_test
          key_no

          ; variables for RTC read/write

          field
          clockvalue
          RTC_value
          RTC_addr

          ; variables used to store date/time information

          hour
          minute
          second
          date
          month
          year

          ehour
          eminute
          esecond
          edate
          emonth
          eyear

          ; variables used to store elapsed time in log generation

          hundred
          ten
          one
          duration

          ; variables used in delay functions

          delay1
          delay2
```

```
delay3

; variables for EEPROM read/write

value
addr

; variables for recording login input

IDchar1
IDchar2
PWchar1
PWchar2

; variables for login verification

check
check_bit
wrong

; variables for printing messages

table_index
str_size

; variables for inheritance of functions in user interface

IDAddr
PWAddr
AccAddr
LogAddr

modaddr
modconfig
uconfig
gconfig
curlog
access
parent_access
child_access
child_addr
mod_bit

; variables for selecting screens and menus in interface hierarchy

screen_sel
module_sel
user_sel
guest_sel
orig_order
order

; variables for general purpose computation

count
comp

; temporary variables

temp
tempaddr
lcd_tmp
kp_ret
kp_tmp
kp_tmp1
kp_tmp2
kp_tmp3
kp_tmp4
char1
char2
long
```

```
     endc

;}


;----[ MACROS ]---------------------------------------------------------------------------------;

;{

;    Writes string at 'str' label to LCD

WRT_STR        macro     str
               local     loop, prep

               movwf     str_size       ; Counter for character offset

loop           movfw     str_size
               pagesel   str
               call      str            ; Goto 'str' label + str_size offset
               pagesel   Main
               movwf     temp           ; Temp holds the character
               incf      temp, f        ; Check if it's 0 (end of string)
               decfsz    temp, f
               goto      prep           ; If not....goto to prep
               return                   ; Else...return
prep           incf      str_size, f    ; Increase offset
               call      WrtLCD         ; Print character
               goto      loop           ; Repeat

               endm

;    Ensures that PCL is on same page as 'table' label

PCLSwitch      macro     table
               movwf     table_index    ; Save current index
               movlw     HIGH table     ; Get the page table is on
               movwf     PCLATH         ; Move PCLATH to that page
               movfw     table_index    ; Move index back into the working reg
               addlw     LOW  table     ; Offset label
               btfsc     STATUS,C       ; Check carry bit
               incf      PCLATH,f       ; If in next page, increment PCLATH
               movwf     PCL            ; Write the correct address to PCL
               endm

;}

Page0

;----[ VECTORS ]--------------------------------------------------------------------------------;

;{

               org       0x0000              ;    Standard reset
               goto      Main                ;    Goto main code.

               org       0x0004              ;    Interrupt reset
               goto      Main                ;    No interrupts

;}


;----[ INITIALIZATION ]-------------------------------------------------------------------------;

;{

;DESCRIPTION:       Initializes peripherals, ports and system
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

Main           call      LongDelay
               pagesel   i2c_common_setup
               call      i2c_common_setup    ;    set-up I2C bus
               pagesel   InitPort
               call      InitPort            ;    set-up ports
               pagesel   Main

               call      InitLCD             ;    set-up LCD

;DESCRIPTION:       Initializes storage system
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

InitSystem     movlw     xRestart
               movwf     addr
               call      ReadROM
               movwf     value
               incfsz    value, f            ;    value = 255 -> system reset

               goto      IDMenu

               call      InitClock
               call      InitAdmin
               call      InitROM

               goto      AdmMenu

;DESCRIPTION:       Initializes the clock by clearing all fields
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

InitClock      movlw     0
               movwf     value

               movlw     RTC_Second
               movwf     field

               call      WriteRTC
               incf      field, f
               call      WriteRTC
               incf      field, f
               call      WriteRTC

               movlw     RTC_Date
               movwf     field

               call      WriteRTC
               incf      field, f
               call      WriteRTC
               incf      field, f
               call      WriteRTC

               return

;DESCRIPTION:       Initializes administrator account
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

InitAdmin      call      CursorOn

               call      ClrLCD              ; Set administrator ID
               call      AdminID
               call      Line2LCD

               movlw     xAdmID
               movwf     IDAddr
               movwf     addr

               call      GetFour
               call      StoreFour

               call      ClrLCD              ; Set administrator PW
               call      AdminPW
               call      Line2LCD
               movlw     xAdmPW
               movwf     PWAddr
               movwf     addr

               call      GetFour
               call      StoreFour

               movlw     xAdmAccess          ; Retrieve administrator access (all)
               movwf     addr
               call      ReadROM
               movwf     access

               return

;DESCRIPTION:       Initializes EEPROM on PIC
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

InitROM        movlw     0
               movwf     value

               movlw     xModConfig          ; no modules configured
               movwf     addr
               call      WrtROM

               movlw     xUActive            ; no users active
               movwf     addr
               call      WrtROM

               movlw     xGActive            ; no guests active
               movwf     addr
               call      WrtROM

               movlw     xRestart            ; system initialized
               movwf     addr
               call      WrtROM


               movlw     4
               movwf     count

               movlw     xUser1ID
               movwf     addr

               call      ResetAccount        ; reset account data
               movlw     20
               addwf     addr, f
               decfsz    count, f
               goto      $-4

               movlw     4
               movwf     count

               movlw     xUser1Log           ; reset log data
               movwf     addr

               call      ResetLog
               movlw     36
               addwf     addr, f
               decfsz    count, f
               goto      $-4

               return

;DESCRIPTION:       Resets/initializes account data in EEPROM
;INPUT REGISTERS:   addr
;OUTPUT REGISTERS:  None

ResetAccount

               movlw     0
               movwf     value
```

```
                movlw     4                   ; user has access to no modules
                addwf     addr, f
                call      WrtROM

                movlw     9                   ; guest has access to no modules
                addwf     addr, f
                call      WrtROM

                movlw     13
                subwf     addr, f

                return

;DESCRIPTION:       Resets/initializes log data in EEPROM
;INPUT REGISTERS:   addr
;OUTPUT REGISTERS:  None

ResetLog        movfw     addr                ; reset pointer to next log
                movwf     value
                call      WrtROM

                movlw     0xFF                ; mark all log slots as empty (d'255)
                movwf     value

                incf      addr, f
                movlw     5
                movwf     temp
                movlw     7

                call      WrtROM
                movlw     7
                addwf     addr, f
                decfsz    temp, f
                goto      $-4

                movlw     36
                subwf     addr, f

                return

;}


;----[ LOGIN ]-------------------------------------------------------------------------------;

;{

;DESCRIPTION:       Displays date/time and waits for input
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

Standby         bcf       PORTC, 7            ; make sure no solenoids are powered
                bcf       PORTC, 6
                call      ClrLCD
                call      CursorOff
                call      Line1LCD
                call      PrintDate           ; display date
                call      PrintSpace
                call      PrintSpace
                call      Press
                call      Line2LCD
                call      PrintTime           ; display time
                call      PrintSpace
                call      AnyKey

StandbyLoop     call      Line2LCD
                call      PrintTime           ; update time

                btfss     PORTB, 1
                goto      StandbyLoop         ; loop if no input
                btfsc     PORTB, 1
                goto      $-1
                goto      InitSystem          ; else set-up system

;DESCRIPTION:       Checks if current users have expired and shows log-in menu
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  IDchar1, IDchar2, PWchar1, PWchar2

IDMenu          call      ValidVerify         ; checks user expiry

                call      ClrLCD

                call      CursorOn            ; prompts for ID
                call      UserID
                call      Line2LCD

                call      GetFour
                movfw     char1
                movwf     IDchar1
                movfw     char2
                movwf     IDchar2

                call      ClrLCD              ; prompts for password
                call      Password
                call      Line2LCD

                call      GetFour
                movfw     char1
                movwf     PWchar1
                movfw     char2
                movwf     PWchar2

;DESCRIPTION:       Checks if operator is administrator
;INPUT REGISTERS:   IDchar1, IDchar2, PWchar1, PWchar2
;OUTPUT REGISTERS:  None

AdminVerify     movlw     xAdmID              ; load admin location to check
                movwf     IDAddr
                movlw     xAdmPW
                movwf     PWAddr
                movlw     xAdmAccess
                movwf     AccAddr
                movlw     0
                movwf     LogAddr

                call      IDCompare           ; compares characters

                btfss     wrong, 2            ; if not an admin, check if a user
                goto      UserVerify

                goto      AdmMenu             ; else proceed to admin menu

;DESCRIPTION:       Checks if operator is a typical user
;INPUT REGISTERS:   IDchar1, IDchar2, PWchar1, PWchar2
;OUTPUT REGISTERS:  None

UserVerify      movlw     xUser1ID            ; load user location to check
                movwf     IDAddr
                movlw     xUser1PW
                movwf     PWAddr
                movlw     xUser1Access
                movwf     AccAddr
                movlw     xUser1Log
                movwf     LogAddr

                movlw     16                  ; xth user has xth bit = 1
                movwf     check_bit

UVerifyLoop     movfw     uconfig             ; check if user exists
                andwf     check_bit, w
                movwf     check

                call      IDCompare           ; compares characters
```

```
               btfss     wrong, 2
               goto      UNext
               incf      check,f
               decfsz    check, f
               goto      UserMenu

UNext          movlw     20                   ; next user (i+20) in EEPROM
               addwf     IDAddr, f
               addwf     PWAddr, f
               addwf     AccAddr, f
               movlw     36                   ; next log (i+36) in EEPROM
               addwf     LogAddr, f

               rrf       check_bit, f

               btfss     check_bit, 0
               goto      UVerifyLoop

;DESCRIPTION:       Checks if operator is a guest
;INPUT REGISTERS:   IDchar1, IDchar2, PWchar1, PWchar2
;OUTPUT REGISTERS:  None

GuestVerify    movlw     xUser1ID             ; load guest location to check
               movwf     IDAddr
               movlw     xGuest1PW
               movwf     PWAddr
               movlw     xGuest1Access
               movwf     AccAddr
               movlw     xUser1Log
               movwf     LogAddr

               movlw     16                   ; xth guest has xth bit = 1
               movwf     check_bit

GVerifyLoop    movfw     gconfig              ; check if guest exists
               andwf     check_bit, w
               movwf     check

               call      IDCompare            ; compares characters

               btfss     wrong, 2
               goto      GNext
               incf      check,f
               decfsz    check, f
               goto      GuestMenu            ; guest exists and log-in correct

GNext          movlw     20                   ; next guest (i+20) in EEPROM
               addwf     IDAddr, f
               addwf     PWAddr, f
               addwf     AccAddr, f
               movlw     36                   ; next log (i+36) in EEPROM
               addwf     LogAddr, f

               rrf       check_bit, f

               btfss     check_bit, 0
               goto      GVerifyLoop

               call      ClrLCD               ; login info is wrong
               call      CursorOff
               call      Denied

               call      HumanDelay

               goto      IDMenu               ; try again

;DESCRIPTION:       Checks if any user or guest accounts have expired
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  uconfig, gconfig

ValidVerify    call      GetTime              ; get the current time
```

```
               movlw     xUActive            ; get configuration for active users
               movwf     addr
               call      ReadROM
               movwf     uconfig

               movlw     16                  ; check if user x is active
               movwf     check_bit
               movfw     uconfig
               movwf     check

               movlw     xUser1Valid
               movwf     addr

               call      CompareTime         ; cycle through users

               movlw     xUActive            ; update active users
               movwf     addr
               movfw     check
               movwf     value
               movwf     uconfig
               call      WrtROM


               movlw     xGActive            ; get configuration for active guests
               movwf     addr
               call      ReadROM
               movwf     gconfig

               movlw     16                  ; check if guest x is active
               movwf     check_bit
               movfw     gconfig
               movwf     check

               movlw     xGuest1Valid
               movwf     addr

               call      CompareTime         ; cycle through guests

               movfw     uconfig
               andwf     check, w
               movwf     gconfig             ; ensure guests don't outlast users
               movwf     value

               movlw     xGActive            ; update active guests
               movwf     addr
               call      WrtROM

               return

;DESCRIPTION:       Checks if any User/Guest X has expired
;INPUT REGISTERS:   check, check_bit, addr
;OUTPUT REGISTERS:  check

CompareTime
               movfw     check_bit           ; check if current user exists
               andwf     check, w
               btfss     STATUS, Z
               goto      $+4
               movlw     5
               addwf     addr, f
               goto      CheckNext

               call      ClrLCD

               call      ReadROM             ; get expiry date/time
               movwf     emonth

               incf      addr, f
               call      ReadROM
               movwf     edate
               incf      addr, f
               call      ReadROM
               movwf     eyear

               incf      addr, f
               call      ReadROM
               movwf     ehour

               incf      addr, f
               call      ReadROM
               movwf     eminute

               incf      addr, f
               call      ReadROM
               movwf     esecond

               movfw     year
               subwf     eyear, w
               btfss     STATUS, C
               goto      AccExpired          ; check if year expired
               btfss     STATUS, Z
               goto      CheckNext

               movfw     month
               subwf     emonth, w
               btfss     STATUS, C
               goto      AccExpired          ; check if month expired
               btfss     STATUS, Z
               goto      CheckNext

               movfw     date
               subwf     edate, w
               btfss     STATUS, C
               goto      AccExpired          ; check if day expired
               btfss     STATUS, Z
               goto      CheckNext

               movfw     hour
               subwf     ehour, w
               btfss     STATUS, C
               goto      AccExpired          ; check if hour expired
               btfss     STATUS, Z
               goto      CheckNext

               movfw     minute
               subwf     eminute, w
               btfss     STATUS, C
               goto      AccExpired          ; check if minute expired
               btfss     STATUS, Z
               goto      CheckNext

               movfw     second
               subwf     esecond, w          ; check if second expired
               btfsc     STATUS, C
               goto      CheckNext

AccExpired     comf      check_bit, w        ; remove active status
               andwf     check, f

CheckNext      movlw     15                  ; goto next user/guest
               addwf     addr, f
               rrf       check_bit, f

               btfss     check_bit, 0
               goto      CompareTime

               return

;DESCRIPTION:       Checks if login info matches user/guest X
;INPUT REGISTERS:   IDAddr, PWAddr, LogAddr, AccAddr
;OUTPUT REGISTERS:  wrong, access, curlog
```

```
IDCompare      movlw     0                   ; wrong = # right characters
               movwf     wrong

               movfw     IDAddr              ; check if ID characters are same
               movwf     addr
               movfw     IDchar1
               movwf     value
               call      CheckROM
               addwf     wrong, f

               incf      addr, f
               movfw     IDchar2
               movwf     value
               call      CheckROM
               addwf     wrong, f

               movfw     PWAddr              ; check if PW characters are same
               movwf     addr
               movfw     PWchar1
               movwf     value
               call      CheckROM
               addwf     wrong, f

               incf      addr, f
               movfw     PWchar2
               movwf     value
               call      CheckROM
               addwf     wrong, f

               movfw     AccAddr             ; retrieve access configuration
               movwf     addr
               call      ReadROM
               movwf     access

               movfw     LogAddr             ; retrieve pointer to next log
               movwf     addr
               call      ReadROM
               movwf     curlog

               return

;DESCRIPTION:       Prints welcome information for correct login
;INPUT REGISTERS:   IDAddr
;OUTPUT REGISTERS:  None

Greeting       call      ClrLCD
               call      CursorOff

               call      Welcome             ; print "Welcome"
               call      PrintSpace

               movfw     IDAddr              ; print user name
               movwf     addr
               call      PrintName

               movlw     "!"
               call      WrtLCD

               call      HumanDelay

               call      CursorOn

               return

;}


;----[ ADMINISTRATOR MENU ]-----------------------------------------------------------------;

;{

;DESCRIPTION:       Generates administrator main menu
```

```
;INPUT REGISTERS:   IDAddr, PWaddr, Accaddr, access
;OUTPUT REGISTERS:  None

AdmMenu        call      Greeting            ; display greeting

               movlw     64
               movwf     screen_sel          ; register for choosing screen

AdmLoop        call      ClrLCD

               btfsc     screen_sel, 6       ; display options
               call      Configure
               btfsc     screen_sel, 5
               call      ManageAcc
               btfsc     screen_sel, 4
               call      OpenMod
               btfsc     screen_sel, 3
               call      AdjDT
               btfsc     screen_sel, 2
               call      ChangePW
               btfsc     screen_sel, 1
               call      ResetSystem
               btfsc     screen_sel, 0
               call      Logoff

               call      Line2LCD
               call      YesOpt

Adm_Input      call      KPScroll            ; poll for input
               movwf     key_no
               btfsc     key_no, 0
               goto      ARightCirc          ; next option
               btfsc     key_no, 1
               goto      ADo_Opt             ; do current option
               goto      ALeftCirc           ; previous option

ADo_Opt        btfsc     screen_sel, 6       ; branch to sub-menu
               goto      Do_Configure        ; configure modules
               btfsc     screen_sel, 5
               goto      Do_Manage           ; manage accounts
               btfsc     screen_sel, 4
               call      Do_Open;            ; open modules
               btfsc     screen_sel, 4
               goto      AdmLoop
               btfsc     screen_sel, 3
               goto      Do_AdjDT            ; adjust date and time
               btfsc     screen_sel, 2
               call      Do_ChangePW         ; change admin password
               btfsc     screen_sel, 2
               goto      AdmLoop
               btfsc     screen_sel, 1
               goto      Do_ResetSystem      ; reset system
               btfsc     screen_sel, 0
               goto      Standby             ; logoff

ARightCirc     bcf       STATUS, C           ; next screen
               rrf       screen_sel, f
               btfss     STATUS, C
               goto      AdmLoop
               movlw     B'01000000'
               movwf     screen_sel
               goto      AdmLoop

ALeftCirc      bcf       STATUS, C           ; previous screen
               rlf       screen_sel, f
               btfss     screen_sel, 7
               goto      AdmLoop
               movlw     B'00000001'
               movwf     screen_sel
               goto      AdmLoop

;}
```

```
;----[ USER MENU ]------------------------------------------------------------------------------;

;{

;DESCRIPTION:       Generates user main menu
;INPUT REGISTERS:   IDAddr, PWaddr, Accaddr, LogAddr, access, curlog
;OUTPUT REGISTERS:  None

UserMenu       call      Greeting            ; display greeting

               movlw     8                   ; register for choosing screen
               movwf     screen_sel

               movlw     xGActive
               movwf     addr
               call      ReadROM
               movwf     gconfig

UserLoop       call      ClrLCD

               btfsc     screen_sel, 3       ; display options
               call      OpenMod
               btfsc     screen_sel, 2
               call      ChangePW
               btfsc     screen_sel, 1
               call      GuestAcc
               btfsc     screen_sel, 0
               call      Logoff

               call      Line2LCD

               btfsc     screen_sel, 1       ; display action item if guest screen
               goto      $+3
               call      YesOpt
               call      User_Input
               movfw     gconfig
               andwf     check_bit, w
               movwf     check
               btfsc     STATUS, Z
               goto      $+3
               call      DelOpt
               goto      User_Input
               call      AddOpt

User_Input     call      KPScroll            ; poll for input
               movwf     key_no
               btfsc     key_no, 0
               goto      URightCirc          ; next option
               btfsc     key_no, 1
               goto      UDo_Opt             ; do option
               goto      ULeftCirc           ; previous option


UDo_Opt        btfsc     screen_sel, 3       ; branch to sub_menu
               call      Do_Open;            ; open module
               btfsc     screen_sel, 3
               goto      UserLoop
               btfsc     screen_sel, 2
               call      Do_ChangePW         ; change password
               btfsc     screen_sel, 2
               goto      UserLoop
               btfsc     screen_sel, 1
               goto      Do_ManageGuest      ; create guest account
               btfsc     screen_sel, 1
               goto      UserLoop
               goto      UserSave            ; logoff

URightCirc     bcf       STATUS, C           ; next screen
               rrf       screen_sel, f
               btfss     STATUS, C
               goto      UserLoop
               movlw     B'00001000'
               movwf     screen_sel
               goto      UserLoop

ULeftCirc      bcf       STATUS, C           ; previous screen
               rlf       screen_sel, f
               btfss     screen_sel, 4
               goto      UserLoop
               movlw     B'00000001'
               movwf     screen_sel
               goto      UserLoop


Do_ManageGuest incf      check, f            ; add/delete guest option
               decfsz    check, f
               goto      GuestDel


GuestAdd       movfw     PWAddr              ; delete outdated guest access
               addlw     11
               movwf     addr
               movlw     0
               movwf     value
               call      WrtROM
               movlw     2
               subwf     addr, f

               call      AddGuest            ; create guest account
               movfw     check_bit
               iorwf     gconfig, f
               goto      UserLoop

GuestDel       comf      check_bit, w        ; delete guest active status
               andwf     gconfig, f
               goto      UserLoop

UserSave       movfw     gconfig             ; save any changes to guest
               movwf     value
               movlw     xGActive
               movwf     addr
               call      WrtROM
               goto      Standby             ; logoff
;}


;----[ GUEST MENU ]-----------------------------------------------------------------------------;

;{

;DESCRIPTION:       Generates guest main menu
;INPUT REGISTERS:   IDAddr, PWaddr, Accaddr, LogAddr, access, curlog
;OUTPUT REGISTERS:  None

GuestMenu      call      Greeting            ; display greeting

               movlw     1
               movwf     screen_sel          ; register for choosing screen

GuestLoop      call      ClrLCD

               btfss     screen_sel, 0       ; display options
               call      Logoff
               btfsc     screen_sel, 0
               call      OpenMod

               call      Line2LCD
               call      YesOpt

Guest_Input    call      KPScroll            ; poll for input
               movwf     key_no
               btfsc     key_no, 1
```

```
                goto      GDo_Opt             ; do option
                goto      GCirc               ; next/prev screen


GDo_Opt         btfsc     screen_sel, 0       ; branch to sub-menus
                call      Do_Open;            ; open modules
                btfsc     screen_sel, 0
                goto      GuestLoop
                goto      Standby             ; log off

GCirc           movlw     B'00000001'         ; display next/prev screen
                xorwf     screen_sel, f
                goto      GuestLoop

;}


;----[ CONFIGURE SYSTEM ]-----------------------------------------------------------------;

;{

;DESCRIPTION:       Generates menu for configuring which modules are active
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

Do_Configure    movlw     xModConfig
                movwf     addr
                call      ReadROM
                movwf     modconfig           ; get current system configuration

                movlw     2
                movwf     module_sel          ; mod X -> xth bit = 1 of module_sel
                movlw     xMod1ID
                movwf     addr
                movlw     1
                movwf     order


ConfigureLoop   btfss     module_sel, 0       ; module or back screen
                goto      ConfigureMod
                call      ClrLCD
                call      Back
                call      Line2LCD
                call      YesOpt
                goto      Config_Input

ConfigureMod    call      ClrLCD              ; print module number
                call      Module
                call      PrintSpace
                call      Enumerate
                call      PrintSpace

                movfw     modconfig
                andwf     module_sel, w
                movwf     check
                incf      check, f
                decfsz    check, f
                goto      OldMod              ; check if slot is occupied

                call      Free                ; new module
                call      Line2LCD
                call      AddOpt              ; display Add Option
                goto      Config_Input

OldMod          call      PrintName           ; module already configured
                call      Line2LCD
                call      RemoveOpt           ; display remove option
                goto      Config_Input


Config_Input    call      KPScroll            ; poll for input
                movwf     key_no
```

```
               btfsc     key_no, 0
               goto      CRightCirc          ; next module screen
               btfsc     key_no, 1
               goto      CDo_Opt             ; branch to sub-menu
               goto      CLeftCirc           ; previous module screen

CDo_Opt        btfsc     module_sel, 0
               goto      ConfigSave          ; back screen - save changes
               incf      check, f
               decfsz    check, f
               goto      ConfigDel           ; slot taken - delete module
               goto      ConfigAdd           ; slot free - add module
ConfigAdd      call      AddModule           ; prompt for module name
               movfw     module_sel
               iorwf     modconfig, f        ; update active modules
               goto      ConfigureLoop
ConfigDel      comf      module_sel, w       ; remove active status
               andwf     modconfig, f
               goto      ConfigureLoop

ConfigSave     movfw     modconfig           ; save settings
               movwf     value
               movlw     xModConfig
               movwf     addr
               call      WrtROM
               goto      AdmLoop

CRightCirc     incf      order, f            ; change screen to next module
               movlw     2
               addwf     addr, f

               bcf       STATUS, C
               rlf       module_sel, f
               btfss     module_sel, 6
               goto      ConfigureLoop
               movlw     B'00000001'
               movwf     module_sel

               movlw     0
               movwf     order
               movlw     xMod1ID
               movwf     addr
               movlw     2
               subwf     addr, f
               goto      ConfigureLoop

CLeftCirc      decf      order, f            ; change screen to previous module
               movlw     2
               subwf     addr, f

               bcf       STATUS, C
               rrf       module_sel, f
               btfss     STATUS, C
               goto      ConfigureLoop
               movlw     B'00100000'
               movwf     module_sel

               movlw     5
               movwf     order
               movlw     xMod5ID
               movwf     addr
               goto      ConfigureLoop

;DESCRIPTION:       Prompts and saves name of Module X
;INPUT REGISTERS:   addr
;OUTPUT REGISTERS:  None

AddModule      call      ClrLCD              ; prompt for module name
               call      ModID
               call      Line2LCD

               call      GetFour
               call      StoreFour

               decf      addr, f

               return
;}


;----[ CONFIGURE USERS/GUEST ]------------------------------------------------------------------;

;{

;DESCRIPTION:       Generates menus for managing/creating/deleting users
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

Do_Manage      movlw     xUActive            ; get current status of users
               movwf     addr
               call      ReadROM
               movwf     uconfig

               movlw     16                  ; User X = 5-Xth bit of uconfig
               movwf     user_sel
               movlw     1
               movwf     order
               movlw     xUser1ID
               movwf     addr
               movlw     xUser1Log
               movwf     LogAddr


ManageLoop     btfss     user_sel, 0         ; check if back screen
               goto      ManageUser
               call      ClrLCD
               call      Back
               call      Line2LCD
               call      YesOpt
               goto      Manage_Input

ManageUser     call      ClrLCD              ; screen for managing user X

               call      User                ; print user number
               call      PrintSpace

               call      Enumerate
               call      PrintSpace

               movfw     uconfig             ; check if slot is free
               andwf     user_sel, w
               movwf     check
               incf      check, f
               decfsz    check, f
               goto      OldUser

               call      Free                ; slot is free - allow adding
               call      Line2LCD
               call      AddOpt
               goto      Manage_Input

OldUser        call      PrintName           ; slot is full - allowing managing
               call      Line2LCD
               call      ManageOpt
               goto      Manage_Input


Manage_Input   call      KPScroll            ; poll for input
               movwf     key_no
               btfsc     key_no, 0
               goto      MRightCirc          ; next screen
               btfsc     key_no, 1
               goto      MDo_Opt             ; branch to sub-menu
               goto      MLeftCirc           ; previous screen
```

```
MDo_Opt         btfsc     user_sel, 0          ; save changes if back screen
                goto      ManageSave
                incf      check, f
                decfsz    check, f
                goto      UserManage           ; manage users if slot is full

UserAdd         call      UserDelete           ; delete current settings

                call      AddUser              ; propagate new settings
                movfw     user_sel             ; update changes to active setting
                iorwf     uconfig, f
                goto      ManageLoop

UserManage      movfw     order                ; save current state in previous menu
                movwf     orig_order

                movlw     1                    ; display options
                movwf     order

                call      ClrLCD

                call      Enumerate
                call      Edit                 ; 1. Edit
                call      PrintSpace
                call      PrintSpace
                call      PrintSpace
                call      PrintSpace

                incf      order, f
                call      Enumerate            ; 2. Logs
                call      Log

                call      Line2LCD

                incf      order, f
                call      Enumerate
                call      Delete               ; 3. Delete
                call      PrintSpace
                call      PrintSpace

                incf      order, f
                call      Enumerate            ; 4. Back
                call      Back

                movfw     orig_order           ; restore parent menu settings
                movwf     order


Action_Input    call      KPGetChar            ; get number input
                call      KPHexToChar
                movwf     key_no
                movlw     48
                subwf     key_no, f

                decfsz    key_no, f
                goto      $+3
                call      AddUser              ; change settings
                goto      UserManage
                decfsz    key_no, f
                goto      $+2
                goto      Do_AccessLog         ; view logs
                decfsz    key_no, f
                goto      $+3
                call      UserDelete           ; delete users
                goto      ManageLoop
                decfsz    key_no, f
                goto      Action_Input         ; invalid input
                goto      ManageLoop           ; go back

UserDelete      comf      user_sel, w          ; delete active status
                andwf     uconfig, f
```

```
               movfw     addr
               movwf     tempaddr

               call      ResetAccount        ; delete module assignments

               movfw     LogAddr
               movwf     addr

               call      ResetLog            ; delete saved logs

               movfw     tempaddr
               movwf     addr

               return

ManageSave     movfw     uconfig             ; save changes to user active status
               movwf     value
               movlw     xUActive
               movwf     addr
               call      WrtROM
               goto      AdmLoop

MRightCirc     incf      order, f            ; next user
               movlw     20
               addwf     addr, f
               movlw     36
               addwf     LogAddr, f
               bcf       STATUS, C
               rrf       user_sel, f
               btfss     STATUS, C
               goto      ManageLoop
               movlw     B'00010000'
               movwf     user_sel
               movlw     1
               movwf     order
               movlw     xUser1ID
               movwf     addr
               movlw     xUser1Log
               movwf     LogAddr
               goto      ManageLoop

MLeftCirc      decf      order, f            ; previous user
               movlw     20
               subwf     addr, f
               movlw     36
               subwf     LogAddr, f
               bcf       STATUS, C
               rlf       user_sel, f
               btfss     user_sel, 5
               goto      ManageLoop
               movlw     B'00000001'
               movwf     user_sel
               movlw     5
               movwf     order
               movlw     xUser4ID
               addlw     20
               movwf     addr
               movlw     xUser4Log
               addlw     36
               movwf     LogAddr
               goto      ManageLoop

               return

;DESCRIPTION:       Prompts for user name and inherits function from AddGuest
;INPUT REGISTERS:   addr, access
;OUTPUT REGISTERS:  None

AddUser        movfw     order
               movwf     orig_order
               call      ClrLCD
               call      UserID
               call      Line2LCD

               movfw     check
               btfsc     STATUS, Z
               goto      $+3
               call      PrintName
               call      Line2LCD

               call      GetFour
               call      StoreFour

               incf      addr, f

               call      AddGuest            ; remaining changes are same as guest

               movlw     4
               subwf     child_addr,w
               movwf     addr

               movfw     orig_order
               movwf     order

               return

;DESCRIPTION:       Prompts for password, module assignment and expiry
;INPUT REGISTERS:   addr, access
;OUTPUT REGISTERS:  None

AddGuest       call      ClrLCD              ; prompt for password
               call      Password
               call      Line2LCD

               incf      check, f            ; check if old or new user
               decfsz    check, w
               call      PrintName
               decfsz    check, f
               call      Line2LCD

               call      GetFour
               call      StoreFour

               movfw     addr
               addlw     2
               movwf     child_addr

               call      ClrLCD              ; set expiry time
               call      CursorOff
               call      ExpiryPrompt
               call      HumanDelay
               call      CursorOn


               call      Expiry

               decf      child_addr, f

               movfw     child_addr
               movwf     addr
               call      ReadROM
               movwf     child_access

               movfw     access
               movwf     parent_access

               call      ClrLCD              ; assign modules
               call      CursorOff
               call      AssignModules
               call      HumanDelay
               call      CursorOn
```

```
                call      AssignModule

                movfw     PWAddr
                movwf     addr

                movfw     parent_access
                movwf     access

                return

;DESCRIPTION:       Updates/creates expiry times for users/guests
;INPUT REGISTERS:   child_addr
;OUTPUT REGISTERS:  None

Expiry          movfw     child_addr
                movwf     addr

                call      ClrLCD
                incf      check, f
                decfsz    check, w            ; has expiry time been set already
                goto      ShowExpiry          ; if so show stats

                call      ClrLCD
                call      DatePrompt

                call      Line2LCD
                call      TimePrompt

                goto      SetExpiry           ; else prompt for new stats

ShowExpiry      call      ReadROM             ; display current month expiry
                call      PrintBCD
                incf      addr, f

                movlw     "/"
                call      WrtLCD

                call      ReadROM             ; display current date expiry
                call      PrintBCD
                incf      addr, f

                movlw     "/"
                call      WrtLCD

                call      ReadROM             ; display current year expiry
                call      PrintBCD
                incf      addr, f

                call      Line2LCD

                call      ReadROM             ; display current hour expiry
                call      PrintBCD
                incf      addr, f

                movlw     ":"
                call      WrtLCD

                call      ReadROM             ; display current minute expiry
                call      PrintBCD
                incf      addr, f

                movlw     ":"
                call      WrtLCD

                call      ReadROM             ; display current second expiry
                call      PrintBCD
                incf      addr, f

SetExpiry       movfw     child_addr
                movwf     addr

                call      Line1LCD
```

```
               call      GetNum              ; get month
               call      WrtROM
               incf      addr, f

               movlw     "/"
               call      WrtLCD

               call      GetNum              ; get date
               call      WrtROM
               incf      addr, f

               movlw     "/"
               call      WrtLCD

               call      GetNum              ; get year
               call      WrtROM
               incf      addr, f

               call      Line2LCD

               call      GetNum              ; get hour
               call      WrtROM
               incf      addr, f

               movlw     ":"
               call      WrtLCD

               call      GetNum              ; get minute
               call      WrtROM
               incf      addr, f

               movlw     ":"
               call      WrtLCD

               call      GetNum              ; get seond
               call      WrtROM
               incf      addr, f

               return

;}


;----[ ASSIGN MODULES ]-------------------------------------------------------------------------;

;{

;DESCRIPTION:       Assigns modules from admin->users or users->guests
;INPUT REGISTERS:   parent_access, child_access
;OUTPUT REGISTERS:  None

AssignModule   movlw     xModConfig          ; get current active modules
               movwf     addr
               call      ReadROM
               movwf     modconfig

               movlw     2                   ; module x = xth bit of modconfig
               movwf     module_sel
               movlw     xMod1ID
               movwf     addr
               movlw     1
               movwf     order


AssignLoop     btfss     module_sel, 0       ; check if back screen
               goto      AssignMod
               call      ClrLCD
               call      Done
               call      Line2LCD
               call      YesOpt
               movlw     1
```

```
                movwf    check
                goto     Assign_Input

AssignMod       call     ClrLCD              ; print module number

                call     Module
                call     PrintSpace
                call     Enumerate
                call     PrintSpace

                movfw    modconfig
                andwf    module_sel, w
                movwf    check
                incf     check, f
                decfsz   check, f
                goto     $+2
                goto     AssignDeny          ; module not setup

                movfw    parent_access
                andwf    module_sel, w
                movwf    check
                incf     check, f
                decfsz   check, f
                goto     $+2
                goto     AssignDeny          ; parent does not have module access

                call     PrintName
                call     Line2LCD

                movfw    child_access        ; see if child already has access
                andwf    module_sel, w
                movwf    mod_bit
                incf     mod_bit, f
                decfsz   mod_bit, f
                goto     OldAssign

                call     AssignOpt
                goto     Assign_Input

OldAssign       call     RemoveOpt           ; display remove option
                goto     Assign_Input

AssignDeny      call     Denied              ; display denied message
                call     Line2LCD
                call     NullOpt


Assign_Input    call     KPScroll            ; poll for input
                movwf    key_no
                btfsc    key_no, 0
                goto     AIRightCirc
                btfsc    key_no, 1
                goto     AIDo_Opt
                goto     AILeftCirc

AIDo_Opt        btfsc    module_sel, 0       ; back screen? save changes
                goto     AISave
                incf     check, f
                decfsz   check, f
                goto     AICheck             ; ok to assign/remove modules
                goto     Assign_Input        ; access was denied - no changes

AICheck         incf     mod_bit, f
                decfsz   mod_bit, f
                goto     AIDel               ; already assigned - delete module
                goto     AIAdd               ; add module

AIAdd           movfw    module_sel          ; update child_access
                iorwf    child_access, f
                goto     AssignLoop

AIDel           comf     module_sel, w       ; update child_access
```

```
               andwf     child_access, f
               goto      AssignLoop

AISave         movfw     child_access        ; save assignment settings
               movwf     value
               movfw     child_addr
               movwf     addr
               call      WrtROM
               return

AIRightCirc    incf      order, f            ; next assign module screen
               movlw     2
               addwf     addr, f

               bcf       STATUS, C
               rlf       module_sel, f
               btfss     module_sel, 6
               goto      AssignLoop
               movlw     B'00000001'
               movwf     module_sel

               movlw     0
               movwf     order
               movlw     xMod1ID
               movwf     addr
               movlw     2
               subwf     addr, f
               goto      AssignLoop

AILeftCirc     decf      order, f            ; previous assign module screen
               movlw     2
               subwf     addr, f

               bcf       STATUS, C
               rrf       module_sel, f
               btfss     STATUS, C
               goto      AssignLoop
               movlw     B'00100000'
               movwf     module_sel

               movlw     5
               movwf     order
               movlw     xMod5ID
               movwf     addr
               goto      AssignLoop

;}


;----[ OPEN MODULES ]----------------------------------------------------------------------------;

;{

;DESCRIPTION:       Open module menu for admin, users and guests
;INPUT REGISTERS:   access, curlog, LogAddr
;OUTPUT REGISTERS:  None

Do_Open        movlw     xModConfig          ; get current system status
               movwf     addr
               call      ReadROM
               movwf     modconfig

               movlw     2                   ; xth module = xth bith of modconfig
               movwf     module_sel
               movlw     xMod1ID
               movwf     addr
               movlw     1
               movwf     order

OpenLoop       btfss     module_sel, 0       ; back screen or module screen
               goto      ModList
               call      ClrLCD
               call      Back
               call      Line2LCD
               call      YesOpt
               movlw     1
               movwf     check
               goto      Open_Input

ModList        call      ClrLCD              ; print module number

               call      Module
               call      PrintSpace

               call      Enumerate
               call      PrintSpace

               movfw     modconfig
               andwf     module_sel, w
               movwf     check
               incf      check, f
               decfsz    check, f
               goto      $+2
               goto      OpenDeny            ; module not set-up - deny access

               movfw     access
               andwf     module_sel, w
               movwf     check
               incf      check, f
               decfsz    check, f
               goto      $+2
               goto      OpenDeny            ; unauthorized - deny access

               call      PrintName           ; access granted - print name
               call      Line2LCD
               call      OpenOpt
               goto      Open_Input          ; print open option

OpenDeny       call      Denied              ; print denied option
               call      Line2LCD
               call      NullOpt

Open_Input     call      KPScroll            ; poll for input
               movwf     key_no
               btfsc     key_no, 0
               goto      ORightCirc          ; next open module screen
               btfsc     key_no, 1
               goto      ODo_Opt             ; branch to sub-menu
               goto      OLeftCirc           ; previous open module screen

ODo_Opt        btfss     module_sel, 0       ; back screen? save settings
               goto      ODo_Open
               movfw     PWAddr
               movwf     addr
               return

ODo_Open       incf      check, f
               decfsz    check, f
               call      OpenModule          ; access ok - open module
               goto      OpenLoop            ; access denied- invalid input

ORightCirc     incf      order, f            ; next open module screen
               movlw     2
               addwf     addr, f

               bcf       STATUS, C
               rlf       module_sel, f
               btfss     module_sel, 6
               goto      OpenLoop
               movlw     B'00000001'
               movwf     module_sel

               movlw     0
               movwf     order
               movlw     xMod1ID
               movwf     addr
               movlw     2
               subwf     addr, f
               goto      OpenLoop

OLeftCirc      decf      order, f            ; previous open module screen
               movlw     2
               subwf     addr, f

               bcf       STATUS, C
               rrf       module_sel, f
               btfss     STATUS, C
               goto      OpenLoop
               movlw     B'00100000'
               movwf     module_sel

               movlw     5
               movwf     order
               movlw     xMod5ID
               movwf     addr
               goto      OpenLoop


OpenModule

               btfsc     PORTA, 0
               goto      PowerOn

               call      ClrLCD
               call      CursorOff
               call      Denied
               call      Line2LCD
               call      LowPower

               call      HumanDelay
               call      CursorOn
               return

PowerOn        incfsz    access, w           ; get current if not admin
               call      GetTime

               pagesel   OpenRoutine
               call      StopSlave           ; stop I2C (using Port C)
               call      OpenRoutine         ; interact with maching
               call      StartSlave          ; restart I2C
               pagesel   OpenModule

               incfsz    access, w           ; generate log if not admin
               call      GenLog

               decfsz    long, f
               return
               goto      Standby

;}


;----[ SYSTEM LOGS ]----------------------------------------------------------------------------;

;{

;DESCRIPTION:       Generates logs for users/guests
;INPUT REGISTERS:   LogAddr, curlog, hour, minute, second, month, date, year
;OUTPUT REGISTERS:  None

GenLog         movlw     RTC_Hour            ;get current hour
               movwf     RTC_addr
               call      ReadRTC
               call      ClockEncode
               movfw     clockvalue
               movwf     ehour
```

```
movlw     RTC_Minute          ;get current minute
movwf     RTC_addr
call      ReadRTC
call      ClockEncode
movfw     clockvalue
movwf     eminute

movlw     RTC_Second          ; get current second
movwf     RTC_addr
call      ReadRTC
call      ClockEncode
movfw     clockvalue
movwf     esecond

movfw     addr
movwf     modaddr

movfw     curlog              ; no logs yet - initialize pointer
subwf     LogAddr, w

btfsc     STATUS, Z
incf      curlog, f

movfw     curlog
movwf     addr

movfw     month               ; save month
movwf     value
call      WrtROM
incf      addr, f

movfw     date                ; save date
movwf     value
call      WrtROM
incf      addr, f

movfw     year                ; save year
movwf     value
call      WrtROM
incf      addr, f

movfw     hour                ; save hour
movwf     value
call      WrtROM
incf      addr, f

movfw     minute              ; save minute
movwf     value
call      WrtROM
incf      addr, f

movfw     IDAddr
addlw     2
subwf     PWAddr, w
btfss     STATUS, Z
goto      $+3
clrw
goto      $+2
movlw     128

iorwf     modaddr, w          ; save name address of module opened
movwf     value
call      WrtROM
incf      addr, f

pagesel   Elapsed
call      Elapsed             ; get elapsed time
pagesel   GenLog

movfw     duration            ; save elapsed time
movwf     value
```

```
               call      WrtROM
               incf      addr, f

               movlw     7
               addwf     curlog, w
               movwf     curlog
               movwf     temp

               movfw     LogAddr
               addlw     35
               subwf     temp, f             ; end of log list - cycle back
               decfsz    temp, f
               goto      $+4
               movfw     LogAddr
               addlw     1
               movwf     curlog

               movfw     curlog              ; save pointer to next log
               movwf     value
               movfw     LogAddr
               movwf     addr

               call      WrtROM

               movfw     modaddr
               movwf     addr

               return

;DESCRIPTION:       Allows admin to view log for User X
;INPUT REGISTERS:   LogAddr, curlog
;OUTPUT REGISTERS:  None

Do_AccessLog   movfw     order
               movwf     orig_order

               movlw     1
               movwf     order

               movfw     addr
               movwf     child_addr

               movfw     LogAddr             ; get current point
               movwf     addr
               call      ReadROM
               movwf     addr

               subwf     LogAddr, w
               btfss     STATUS, Z
               goto      $+2
               goto      NoLog               ; is log emptry
               movlw     7
               subwf     addr, f

               movfw     LogAddr             ; if not go to latest entry
               subwf     addr, w
               btfsc     STATUS, C
               goto      LogLoop
               movfw     LogAddr             ; if we're at the start, go to end
               addlw     29
               movwf     addr
               goto      LogLoop

NoLog          incf      LogAddr, w          ; dummy pointer
               movwf     addr

LogLoop        call      ClrLCD              ; list log number
               call      Enumerate

               movfw     addr
```

```
                call      ReadROM             ; is entry empty
                movwf     temp
                incfsz    temp, f             ; if so skip the stats
                goto      PrintStats

                call      PrintSpace          ; print empty, and goto input
                call      Empty
                call      Line2LCD
                call      NullOpt
                movlw     7
                addwf     addr, f
                goto      Log_Input

PrintStats      call      ReadROM             ; print month
                call      PrintBCD
                incf      addr, f

                movlw     "/"
                call      WrtLCD

                call      ReadROM             ; print date
                call      PrintBCD
                incf      addr, f

                movlw     "/"
                call      WrtLCD

                call      ReadROM             ; print year
                call      PrintBCD
                incf      addr, f

                call      PrintSpace

                call      ReadROM             ; print hour
                call      PrintBCD
                incf      addr, f

                movlw     ":"
                call      WrtLCD

                call      ReadROM             ; print minute
                call      PrintBCD
                incf      addr, f

                call      Line2LCD

                movlw     127
                call      WrtLCD
                call      PrintSpace

                movfw     addr
                movwf     tempaddr

                call      ReadROM             ; print name of opened module
                movwf     addr
                btfss     addr, 7
                movlw     "U"
                btfsc     addr, 7
                movlw     "G"
                call      WrtLCD
                call      PrintSpace
                movlw     b'01111111'
                andwf     addr, f
                call      PrintName
                call      PrintSpace

                movfw     tempaddr
                movwf     addr
                incf      addr, f

                call      ReadROM
                movwf     duration
```

```
               movlw     16                  ; get duration
               addwf     duration, w
               btfsc     STATUS, C
               goto      TooLong             ; check if > 4 minutes

               pagesel   GetElapsed
               call      GetElapsed          ; get elpased time (bin to dec)
               pagesel   PrintStats


PrintElapsed   call      PrintSpace

               movfw     hundred
               addlw     48
               call      WrtLCD

               movfw     ten
               addlw     48
               call      WrtLCD

               movfw     one
               addlw     48
               call      WrtLCD

               goto      LogPrint

TooLong        movlw     ">"                 ; too long? print > than 240 seconds
               call      WrtLCD
               movlw     "2"
               call      WrtLCD
               movlw     "4"
               call      WrtLCD
               movlw     "0"
               call      WrtLCD


LogPrint       movlw     "s"
               call      WrtLCD
               call      PrintSpace
               movlw     126
               call      WrtLCD

               incf      addr, f

Log_Input      call      KPScroll            ; poll for input
               movwf     key_no
               btfsc     key_no, 0
               goto      LRightCirc          ; older log
               btfsc     key_no, 1
               goto      LDo_Opt             ; branch to sub-menu
               goto      LLeftCirc           ; newer log

LDo_Opt        movfw     child_addr          ; go back if at back screen
               movwf     addr
               movfw     orig_order
               movwf     order
               goto      UserManage

LRightCirc     movlw     5                   ; goto older log
               subwf     order, w
               btfsc     STATUS, Z
               clrf      order
               incf      order, f

               movlw     14
               subwf     addr, f

               movfw     LogAddr
               subwf     addr, w
               btfsc     STATUS, C
               goto      LogLoop
```

```
                movfw    LogAddr
                addlw    29
                movwf    addr
                goto     LogLoop

LLeftCirc       movlw    5                   ; goto newer log
                decf     order, f
                btfsc    STATUS, Z
                movwf    order

                movfw    LogAddr
                addlw    36
                subwf    addr, w
                btfss    STATUS, Z
                goto     LogLoop
                movfw    LogAddr
                movwf    addr
                incf     addr, f
                goto     LogLoop

;}


;----[ MISCELLANEOUS FUNCTIONALITIES ]----------------------------------------------------------;

;{

;DESCRIPTION:       Allows user/administrator to change password
;INPUT REGISTERS:   PWAddr
;OUTPUT REGISTERS:  None

Do_ChangePW     movfw    PWAddr              ; get address to save password
                movwf    addr

                call     ClrLCD              ; prompt for password
                call     Password
                call     Line2LCD

                call     GetFour
                call     StoreFour

                return

;DESCRIPTION:       Allows administrator to reset system (logs, account, modules)
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

Do_ResetSystem movlw     xRestart
                movwf    addr
                movlw    255
                movwf    value
                call     WrtROM

                goto     InitSystem

;DESCRIPTION:       Allows administrator to adjust date/time display
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

Do_AdjDT        call     ClrLCD
                call     DatePrompt

                call     Line2LCD
                call     TimePrompt

                call     Line1LCD

                movlw    RTC_Month           ; get month
                movwf    field
                call     GetNum
                call     WriteRTC
```

```
               movlw     "/"
               call      WrtLCD

               movlw     RTC_Date            ; get date
               movwf     field
               call      GetNum
               call      WriteRTC

               movlw     "/"
               call      WrtLCD

               movlw     RTC_Year            ; get year
               movwf     field
               call      GetNum
               call      WriteRTC

               call      Line2LCD

               movlw     RTC_Hour            ; get hour
               movwf     field
               call      GetNum
               call      WriteRTC

               movlw     ":"
               call      WrtLCD

               movlw     RTC_Minute          ; get minute
               movwf     field
               call      GetNum
               call      WriteRTC

               movlw     ":"
               call      WrtLCD

               movlw     RTC_Second          ; get second
               movwf     field
               call      GetNum
               call      WriteRTC

               goto      AdmLoop

;}


;----[ CLOCK FUNCTIONS ]-----------------------------------------------------------------;

;{

;DESCRIPTION:       Tranmit data through I2C bus
;INPUT REGISTERS:   field, value
;OUTPUT REGISTERS:  none

WriteRTC       rtc_set   field, value
               banksel 0x00
               return

;DESCRIPTION:       Upload data through I2C bus
;INPUT REGISTERS:   RTC_addr
;OUTPUT REGISTERS:  dig10, dig1

ReadRTC        rtc_read  RTC_addr
               banksel   0x00
               return

;DESCRIPTION:       Converts 2-byte ASCII value to 1-byte binary
;INPUT REGISTERS:   dig10, dig1
;OUTPUT REGISTERS:  clockvalue

ClockEncode    movlw     48                  ; tens digit = upper nibble
               subwf     dig10, w
               andlw     0x0F
               movwf     clockvalue
               swapf     clockvalue, f

               movlw     48                  ; ones digit = lower nibble
               subwf     dig1, w
               andlw     0x0F
               addwf     clockvalue, f
               return

;DESCRIPTION:       Get date and time from RTC chip
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  hour, minute, second, month, date, year

GetTime
               movlw     RTC_Hour            ;get current hour
               movwf     RTC_addr
               call      ReadRTC
               call      ClockEncode
               movfw     clockvalue
               movwf     hour

               movlw     RTC_Minute          ;get current minute
               movwf     RTC_addr
               call      ReadRTC
               call      ClockEncode
               movfw     clockvalue
               movwf     minute

               movlw     RTC_Second          ; get current second
               movwf     RTC_addr
               call      ReadRTC
               call      ClockEncode
               movfw     clockvalue
               movwf     second

               movlw     RTC_Month           ; get current month
               movwf     RTC_addr
               call      ReadRTC
               call      ClockEncode
               movfw     clockvalue
               movwf     month

               movlw     RTC_Date            ; get current date
               movwf     RTC_addr
               call      ReadRTC
               call      ClockEncode
               movfw     clockvalue
               movwf     date

               movlw     RTC_Year            ; get current year
               movwf     RTC_addr
               call      ReadRTC
               call      ClockEncode
               movfw     clockvalue
               movwf     year

               return

;}


;----[ LCD FUNCTIONS ]----------------------------------------------------------------------------;

;{

;DESCRIPTION:       Initialize the LCD
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

InitLCD
               banksel   PORTD

               call      LongDelay           ; wait for LCD POR to finish (~15ms)
               call      LongDelay
               call      LongDelay

               movlw     B'00110011'         ; ensure 8-bit mode first
               call      WrtIns
               call      LongDelay

               movlw     B'00110010'
               call      WrtIns
               call      LongDelay

               movlw     B'00101000'         ; 4 bits, 2 lines,5X8 dot
               call      WrtIns
               call      LongDelay

               call      CursorOn            ; turn on cursor

               movlw     B'00000110'         ; increment cursor without shifting screen
               call      WrtIns
               call      LongDelay

               call      ClrLCD              ; clear screen

               return

;DESCRIPTION:       Clears the LCD
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

ClrLCD         movlw     B'00000001'         ; command for clearing LCD RAM
               call      WrtIns
               call      LongDelay

               return

;DESCRIPTION:       Writes literal characters to the LCD
;INPUT REGISTERS:   w
;OUTPUT REGISTERS:  None

WrtLCD         movwf     lcd_tmp             ; store character to be printed
               call      MovMSB              ; move MSB to PORTD
               call      E_Pulse             ; pulse enable
               swapf     lcd_tmp,w           ; move LSB to PORTD
               call      MovMSB
               call      E_Pulse             ; pulse clock
               return

;DESCRIPTION:       Pulses line low and high to transmit data
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

E_Pulse        call      ShortDelay
               bcf       PORTD,E             ; set enable low
               call      ShortDelay
               bsf       PORTD,E             ; set enable high
               return

;DESCRIPTION:       Transmits upper nibble then lower nibble
;INPUT REGISTERS:   w
;OUTPUT REGISTERS:  None

MovMSB         andlw     0xF0                ; clear 4 LSBs
               iorwf     PORTD,f             ; move into PORTD
               iorlw     0x0F                ; clear 4 MSBs
               andwf     PORTD,f             ; move into PORTD
               return

;DESCRIPTION:       Move cursor to Line 1
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

Line1LCD       movlw     B'10000000'         ; command for moving to line 1
```

```
               call      WrtIns
               call      LongDelay
               return

;DESCRIPTION:       Move cursor to Line 2
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

Line2LCD       movlw     B'10101000'          ; command for moving to line 2
               call      WrtIns
               call      LongDelay
               return

;DESCRIPTION:       Turn cursor on
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

CursorOn       movlw     B'00001111'          ; display on, cursor on, blink on
               call      WrtIns
               call      LongDelay

               return

;DESCRIPTION:       Turn cursor off
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

CursorOff      movlw     B'00001100'          ; display on, cursor off, blink off
               call      WrtIns
               call      LongDelay
               return

;DESCRIPTION:       Sends command to LCD
;INPUT REGISTERS:   w
;OUTPUT REGISTERS:  None

WrtIns         bcf       PORTD, RS            ; instruction mode
               call      WrtLCD               ; write instruction
               bsf       PORTD, RS            ; data mode

               return

;}


;----[ DELAY FUNCTIONS ]-------------------------------------------------------------------------;

;{

;DESCRIPTION:       Delay for 750 ms
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

HumanDelay     movlw     150
               movwf     delay3

HD_Loop        call      LongDelay
               decfsz    delay3, f
               goto      HD_Loop

               return

;DESCRIPTION:       Delay for 5 ms
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

LongDelay      movlw     20
               movwf     delay2

LD_Loop        call      ShortDelay
               decfsz    delay2,f
               goto      LD_Loop
               return

;DESCRIPTION:       Delay for 160 us
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

ShortDelay     movlw     0xFF
               movwf     delay1
               decfsz    delay1,f
               goto      $-1

               return

;}


;----[ INPUT FUNCTIONS ]------------------------------------------------------------------------;

;{

;DESCRIPTION:       Poll for input from keypad and log out if more than 60 secods
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  w

KPGetChar
               movlw     RTC_Second          ; get starting time second
               movwf     RTC_addr
               call      ReadRTC
               call      ClockEncode
               decf      clockvalue, w       ; subtract 1
               movwf     second
               incf      second, w
               btfss     STATUS, Z
               goto      Polling
               movlw     59                  ; if 0 then make it 59
               movwf     second

Polling        movlw     RTC_Second          ; get current second
               movwf     RTC_addr
               call      ReadRTC
               call      ClockEncode
               movfw     clockvalue
               subwf     second, w           ; check if its same as starting
               btfsc     STATUS, Z
               goto      Standby             ; if so logout
               clrw
               btfss     PORTB,1             ; wait until data from keypad input
               goto      Polling             ; keep updating elapsed time
               swapf     PORTB,W             ; read PortB<7:4> into W<3:0>
               andlw     0x0F                ; clear W<7:4>
               btfsc     PORTB,1             ; wait until key is released
               goto      $-1

               return

;DESCRIPTION:       Converts binary keypad values to ASCII
;INPUT REGISTERS:   w
;OUTPUT REGISTERS:  w

KPHexToChar    PCLSwitch AlphaNum
AlphaNum       dt        "123A456B789C*0#D", 0

;DESCRIPTION:       Gets input for menu scrolling ('#', '0', '*')
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  w

KPScroll       call      KPGetChar           ; get input
               movwf     key_no
               incf      key_no, f
               btfsc     key_no, 4
               goto      KPScroll
               incf      key_no, f
               btfsc     key_no, 4
               retlw     1                   ; is it '#' (next)

               incf      key_no, f
               btfsc     key_no, 4
               retlw     2                   ; is it '0' (do)

               incf      key_no, f
               btfsc     key_no, 4
               retlw     4                   ; is it '*' (prev)

               goto      KPScroll            ; invalid input

;DESCRIPTION:       Gets alphanumeric input (everything except *, #)
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  w

KPGetAlphaNum  call      KPGetChar           ; get input
               movwf     kp_tmp
               movlw     0x0E
               xorwf     kp_tmp, w
               btfsc     STATUS, Z
               goto      KPGetAlphaNum       ; try again if '#'

               clrf      kp_ret
               movlw     0x0C
               xorwf     kp_tmp, w
               btfsc     STATUS, Z
               return

               incf      kp_ret, f
               movfw     kp_tmp
               call      KPHexToChar
               call      WrtLCD
               return

;DESCRIPTION:       Gets and prints two characters and returns one binary byte
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  value

GetFour        call      KPGetAlphaNum            ; get input
               movfw     kp_ret
               btfsc     STATUS, Z
               goto      GetFour

               movfw     kp_tmp
               movwf     kp_tmp1

Char2          call      KPGetAlphaNum            ; get input
               movfw     kp_ret
               btfss     STATUS, Z
               goto      $+4
               movlw     b'00010000'
               call      WrtIns
               goto      GetFour

               movfw     kp_tmp
               movwf     kp_tmp2

Char3          call      KPGetAlphaNum            ; get input
               movfw     kp_ret
               btfss     STATUS, Z
               goto      $+4
               movlw     b'00010000'
               call      WrtIns
               goto      Char2

               movfw     kp_tmp
               movwf     kp_tmp3
               call      KPGetAlphaNum            ; get input
               movfw     kp_ret
               btfss     STATUS, Z
               goto      $+4
               movlw     b'00010000'
               call      WrtIns
               goto      Char3

               movfw     kp_tmp
               movwf     kp_tmp4

               swapf     kp_tmp1, w
               iorwf     kp_tmp2, w
               movwf     char1

               swapf     kp_tmp3, w
               iorwf     kp_tmp4, w
               movwf     char2

               return

;DESCRIPTION:       Gets one digit and displays it on LCD
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  comp

GetDigit       call      KPGetChar           ; get input
               call      KPHexToChar         ; convert to ASCII

               movwf     temp
               btfss     temp, 4
               goto      GetDigit            ; '#' or '*' - try again

               movwf     comp
               movlw     0x3A
               subwf     comp, f
               btfss     comp,7
               goto      GetDigit            ; a letter not number - try again

               movfw     temp                ; print number
               call      WrtLCD
               movfw     comp
               addlw     0x0A
               movwf     comp

               return

;DESCRIPTION:       Gets two digit number and packs it in one binary byte
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  value

GetNum         call      GetDigit
               swapf     comp, w
               movwf     value
               call      GetDigit
               addwf     value, f
               return

;}


;----[ OUTPUT FUNCTIONS ]------------------------------------------------------------------------;

;{

;DESCRIPTION:       Prints "X: " for given X
;INPUT REGISTERS:   order
;OUTPUT REGISTERS:  None

Enumerate      movfw     order
               addlw     48
               call      WrtLCD
               movlw     ":"
               call      WrtLCD

               return

;DESCRIPTION:       Prints a space
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

PrintSpace     movlw     " "
               call      WrtLCD
               return

;DESCRIPTION:       Prints one keypad encoded byte as two ASCII characters
;INPUT REGISTERS:   w
;OUTPUT REGISTERS:  None

PrintASCII     movwf     value               ; print first character
               swapf     value, w
               andlw     0x0F
               call      KPHexToChar
               call      WrtLCD

               movfw     value               ; print second character
               andlw     0x0F
               call      KPHexToChar
               call      WrtLCD

               return

;DESCRIPTION:       Prints one binary byte as two ASCII numerals
;INPUT REGISTERS:   w
;OUTPUT REGISTERS:  None

PrintBCD
               movwf     temp                ; print first numeral
               swapf     temp, w
               andlw     0x0F
               addlw     48
               call      WrtLCD

               movfw     temp                ; print second numeral
               andlw     0x0F
               addlw     48
               call      WrtLCD

               return

;DESCRIPTION:       Prints four character name (ID/PW/module)
;INPUT REGISTERS:   addr
;OUTPUT REGISTERS:  None

PrintName      call      ReadROM             ; get first encoded byte
               call      PrintASCII
               incf      addr, f
               call      ReadROM             ; get second encoded byte
               call      PrintASCII
               decf      addr, f

               return

;DESCRIPTION:       Retrieves and displays date on LCD
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

PrintDate      movlw     RTC_Month           ; print month
               movwf     RTC_addr
               call      ReadRTC
               call      DisplayRTC

               movlw     "/"
               call      WrtLCD
               movlw     RTC_Date            ; print date
               movwf     RTC_addr
               call      ReadRTC
               call      DisplayRTC

               movlw     "/"
               call      WrtLCD

               movlw     RTC_Year            ; print year
               movwf     RTC_addr
               call      ReadRTC
               call      DisplayRTC

               return

;DESCRIPTION:       Retrieves and displays time on LCD
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

PrintTime      movlw     RTC_Hour            ; print hour
               movwf     RTC_addr
               call      ReadRTC
               call      DisplayRTC

               movlw     ":"
               call      WrtLCD

               movlw     RTC_Minute          ; print minute
               movwf     RTC_addr
               call      ReadRTC
               call      DisplayRTC

               movlw     ":"
               call      WrtLCD

               movlw     RTC_Second          ; print second
               movwf     RTC_addr
               call      ReadRTC
               call      DisplayRTC

               return

;DESCRIPTION:       Displays 10 and 1 digit from clock to LCD
;INPUT REGISTERS:   dig10, dig1
;OUTPUT REGISTERS:  None

DisplayRTC     movfw     dig10               ; display tens digit
               call      WrtLCD
               movfw     dig1                ; display ones digit
               call      WrtLCD

               return

;DESCRIPTION:       Moves address of message into w and goes to right table
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

AdminID        movlw     0
               goto      Message1Disp
AdminPW        movlw     14
               goto      Message1Disp
UserID         movlw     28
               goto      Message1Disp
Password       movlw     38
               goto      Message1Disp
ModID          movlw     54
               goto      Message1Disp
PCInterface    movlw     71
               goto      Message1Disp
Configure      movlw     84
               goto      Message1Disp
ManageAcc      movlw     94
```

```
              goto      Message1Disp
AssignModules movlw     110
              goto      Message1Disp
ExpiryPrompt  movlw     125
              goto      Message1Disp
OpenMod       movlw     136
              goto      Message1Disp
AdjDT         movlw     148
              goto      Message1Disp
DatePrompt    movlw     165
              goto      Message1Disp
TimePrompt    movlw     174
              goto      Message1Disp
ChangePW      movlw     183
              goto      Message1Disp
ResetSystem   movlw     199
              goto      Message1Disp
Edit          movlw     212
              goto      Message1Disp
Log           movlw     217
              goto      Message1Disp
GuestActive   movlw     221
              goto      Message1Disp
GuestAcc      movlw     234
              goto      Message1Disp
Welcome       movlw     248
              goto      Message1Disp

Denied        movlw     0
              goto      Message2Disp
Empty         movlw     7
              goto      Message2Disp
Free          movlw     13
              goto      Message2Disp
Delete        movlw     20
              goto      Message2Disp
Module        movlw     27
              goto      Message2Disp
User          movlw     34
              goto      Message2Disp
Press         movlw     39
              goto      Message2Disp
AnyKey        movlw     45
              goto      Message2Disp
LowPower      movlw     55
              goto      Message2Disp
Unlocked      movlw     65
              goto      Message2Disp
ModuleOpened  movlw     76
              goto      Message2Disp
Obstructed    movlw     93
              goto      Message2Disp
YesOpt        movlw     110
              goto      Message2Disp
AddOpt        movlw     127
              goto      Message2Disp
OpenOpt       movlw     144
              goto      Message2Disp
DelOpt        movlw     161
              goto      Message2Disp
RemoveOpt     movlw     178
              goto      Message2Disp
AssignOpt     movlw     195
              goto      Message2Disp
ManageOpt     movlw     212
              goto      Message2Disp

Back          movlw     229
              goto      Message2Disp
Done          movlw     235
              goto      Message2Disp
Logoff        movlw     241
```

```
                goto      Message2Disp
Skip            movlw     249
                goto      Message2Disp


NullOpt         movlw     127
                call      WrtLCD
                movlw     14
                movwf     count
                call      PrintSpace
                decfsz    count, f
                goto      $-2
                movlw     126
                call      WrtLCD

                return

;DESCRIPTION:       Offsets w from correct table
;INPUT REGISTERS:   w
;OUTPUT REGISTERS:  None

Message1Disp    WRT_STR   Message1       ; print from "Messages1" table
                return

Message2Disp    WRT_STR   Message2       ; print from "Messages2" table
                return

;}


;----[ EEPROM FUNCTIONS ]------------------------------------------------------------------;

;{

;DESCRIPTION:       Writes data to EEPROM
;INPUT REGISTERS:   addr, value
;OUTPUT REGISTERS:  None

WrtROM          movfw     addr

                banksel   EEADR               ; set address
                movwf     EEADR

                banksel   value
                movfw     value

                banksel   EEDATA              ; set value
                movwf     EEDATA

                banksel   EECON1              ; standard write sequence
                bcf       EECON1, EEPGD
                bsf       EECON1, WREN
                movlw     0x55
                movwf     EECON2
                movlw     0xAA
                movwf     EECON2
                bsf       EECON1, WR
                bcf       EECON1, WREN
                btfsc     EECON1, WR
                goto      $-1

                banksel   addr

                return

;DESCRIPTION:       Reads data from EEPROM
;INPUT REGISTERS:   addr
;OUTPUT REGISTERS:  w

ReadROM         movfw     addr

                banksel   EEADR               ; set address
```

```
               movwf     EEADR

               banksel   EECON1              ; standard read sequence
               bcf       EECON1, EEPGD
               bsf       EECON1, RD

               banksel   EEDATA              ; get data
               movfw     EEDATA

               banksel   addr

               return

;DESCRIPTION:       Compares if contents of value are same as in EEPROM addr
;INPUT REGISTERS:   addr
;OUTPUT REGISTERS:  w

CheckROM       call      ReadROM             ; get data

               subwf     value, f            ; get difference
               incf      value, f
               decfsz    value, f
               retlw     0                   ; different values
               retlw     1                   ; same value

StoreFour      movfw     char1
               movwf     value
               call      WrtROM

               incf      addr, f
               movfw     char2
               movwf     value
               call      WrtROM

               return

;}


Page1          org       0x800

;----[ TABLES ]---------------------------------------------------------------------------------;

;{

;DESCRIPTION:       Table of messages
;INPUT REGISTERS:   N/A
;OUTPUT REGISTERS:  N/A

Message1       PCLSwitch Table1              ; change pages if 256 byte boundary
               ;         "Message", end of str         ;start length
Table1         dt        "Set Admin ID:", 0            ;0        14
               dt        "Set Admin PW:", 0            ;14       14
               dt        "Enter ID:", 0                ;28       10
               dt        "Enter Password:", 0          ;38       16
               dt        "Enter Module ID:", 0         ;54       17
               dt        "PC Interface", 0             ;71       13
               dt        "Configure", 0                ;84       10
               dt        "Manage Accounts", 0          ;94       16
               dt        "Assign Modules", 0           ;110      15
               dt        "Set Expiry", 0               ;125      11
               dt        "Open Module", 0              ;136      12
               dt        "Adjust Date/Time", 0         ;148      17
               dt        "MM/DD/YY", 0                 ;165      9
               dt        "HH:MM:SS", 0                 ;174      9
               dt        "Change Password", 0          ;183      16
               dt        "Reset System", 0             ;199      13
               dt        "Edit", 0                     ;212      5
               dt        "Log", 0                      ;217      4
               dt        "Guest Active", 0             ;221      13
               dt        "Guest Account", 0            ;234      14
               dt        "Welcome", 0                  ;248      8
```

```
Message2       PCLSwitch Table2              ; change pages if 256 byte boundary
               ;         "Message", end of str         ;start length
Table2         dt        "Denied", 0                   ;0        7
               dt        "Empty", 0                    ;7        6
               dt        "(Free)", 0                   ;13       7
               dt        "Delete", 0                   ;20       7
               dt        "Module", 0                   ;27       7
               dt        "User", 0                     ;34       5
               dt        "Press", 0                    ;39       6
               dt        "Any Key..", 0                ;45       10
               dt        "Low Power", 0                ;55       10
               dt        "Unlocked..", 0               ;65       11
               dt        "Module Opened...", 0         ;76       17
               dt        "Door Obstructed!", 0         ;93       17
               dt        127, "    0--Yes    ", 126, 0 ;110      17
               dt        127, "    0--Add    ", 126, 0 ;127      17
               dt        127, "    0-Open    ", 126, 0 ;144      17
               dt        127, "   0-Delete   ", 126, 0 ;161      17
               dt        127, "   0-Remove   ", 126, 0 ;178      17
               dt        127, "   0-Assign   ", 126, 0 ;195      17
               dt        127, "   0-Manage   ", 126, 0 ;212      17
               dt        "Back?", 0                    ;229      6
               dt        "Done?", 0                    ;235      6
               dt        "Logoff?", 0                  ;241      8
               dt        "Skip", 126, 0                ;249      6

;}


; ---[ PORT FUNCTIONS ]-------------------------------------------------------------------------;

;{

;DESCRIPTION:       Initializes ports
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

InitPort       clrf      INTCON              ; no interrupts
               banksel   PORTA               ; clear all data latches
               clrf      PORTA
               clrf      PORTB
               clrf      PORTC
               clrf      PORTD
               clrf      PORTE


               banksel   ADCON1              ; set port A as digital
               movlw     6
               movwf     ADCON1

               banksel   TRISA               ; set port A as input
               movlw     b'00111111'
               movwf     TRISA
               movlw     b'11110010'         ; 4-bit keypad input
               movwf     TRISB
               movlw     b'00011000'         ; C<3:4> used by clock
               movwf     TRISC
               clrf      TRISD               ; all port D is output
               clrf      TRISE               ; don't need port E

               banksel   0x00

               return

;DESCRIPTION:       Ensure no data transmission occur on I2C bus
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

I2C_Idle       btfsc     SSPSTAT, R_W        ; transmitting?
               goto      $-1
               movfw     SSPCON2
               andlw     0x1F                 ; mask ACKEN, RCEN, PEN, RSEN, SEN
               btfss     STATUS, Z
               goto      $-3

               return

;DESCRIPTION:       Restarts I2C bus
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

StartSlave
               banksel   TRISC               ; initialize port c
               movlw     b'00011000'
               movwf     TRISC

               call      I2C_Idle            ; make sure no data transmission

               movlw     b'00001000'         ; config SSP for Master Mode I2C
               banksel   SSPCON
               movwf     SSPCON
               bsf       SSPCON,SSPEN        ; enable SSP module

               banksel   SSPCON2             ; enable repeated start bit
               bsf       SSPCON2,RSEN
               btfsc     SSPCON2,RSEN
               goto      $-1

               banksel   0

               return

;DESCRIPTION:       Disengages I2C bus
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

StopSlave      call      I2C_Idle            ; make sure no data transmission

               banksel   SSPCON2             ; pause write enable
               bsf       SSPCON2,PEN
               btfsc     SSPCON2,PEN
               goto      $-1

               banksel   SSPCON              ; disable SSP module
               clrf      SSPCON

               banksel   TRISC               ; set up PORTC for output
               clrf      TRISC
               banksel   0

               return

;}


;----[ MACHINE INTERFACE ]----------------------------------------------------------------------;

;{

;DESCRIPTION:       Sends output signals to solenoids and gets input from sensors
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

OpenRoutine
               banksel   PORTC
               movfw     module_sel
               movwf     PORTC
               bsf       PORTC, 7            ; unlock module

               pagesel   ClrLCD              ; display unlocked message
               call      ClrLCD
               call      Unlocked
               pagesel   OpenRoutine

               clrf      long

               movlw     15                  ; wait three seconds
               movwf     delay1
Open1          movlw     255
               movwf     delay2
Open2          movlw     255
               movwf     delay3
Open3          movfw     module_sel
               andwf     PORTA, w            ; microswitch opened
               btfss     STATUS, Z
               goto      DoorOpened
               decfsz    delay3,f
               goto      Open3
               decfsz    delay2,f
               goto      Open2
               decfsz    delay1, f
               goto      Open1

               goto      DoneInteract        ; door never opened - relock

DoorOpened     bcf       PORTC, 7            ; door opened, relax lock

               pagesel   ClrLCD              ; display opened message
               call      ClrLCD
               call      ModuleOpened
               pagesel   OpenRoutine

               incfsz    access, w           ; if admin, keep open until button
               goto      JammedOpen

ForeverOpen    movfw     module_sel
               andwf     PORTA, w            ; has button been pushed?
               btfsc     STATUS, Z
               goto      Confirm             ; if so branch to confirm
               goto      ForeverOpen


JammedOpen     movlw     85                  ; wait for 15 seconds
               movwf     delay1
Button1        movlw     255
               movwf     delay2
Button2        movlw     255
               movwf     delay3
Button3        movfw     module_sel
               andwf     PORTA, w
               btfsc     STATUS, Z           ; has button been pushed?
               goto      Confirm             ; if so goto Confirm
               decfsz    delay3,f
               goto      Button3
               decfsz    delay2,f
               goto      Button2
               decfsz    delay1, f
               goto      Button1


Confirm        bsf       PORTC, 6            ; release door jammer
               pagesel   HumanDelay
               call      HumanDelay
               pagesel   OpenRoutine
               movfw     module_sel
               andwf     PORTA, w
               btfsc     STATUS, Z           ; is door still open/button pushed
               goto      Proceed

DoorStuck
               pagesel   ClrLCD              ; if so, idle until user rectifies
               call      ClrLCD
               call      Obstructed
               pagesel   OpenRoutine
               movfw     module_sel
               andwf     PORTA, w
               btfss     STATUS, Z
               goto      $-3

Proceed
               pagesel   ClrLCD              ; system ready, wait for user to go
               call      ClrLCD
               call      Done
               call      Line2LCD
               call      Press
               call      PrintSpace
               call      AnyKey
               pagesel   OpenRoutine

               movlw     225                 ; wait for 60 seconds
               movwf     delay1
KP1            movlw     255
               movwf     delay2
KP2            movlw     255
               movwf     delay3
KP3            movfw     module_sel
               andwf     PORTA, w
               btfss     STATUS, Z
               goto      DoorStuck           ; button pushed again or door opened
               btfss     PORTB, 1
               goto      TestDoor
               btfsc     PORTB, 1
               goto      $-1                 ; user acknowledges completion
               goto      DoneInteract
TestDoor       decfsz    delay3,f
               goto      KP3
               decfsz    delay2,f
               goto      KP2
               decfsz    delay1, f
               goto      KP1

               incf      long, f

DoneInteract
               banksel   PORTC               ; clear all solenoid output
               clrf      PORTC

               return

;}


;----[ MATH FUNCTIONS ]-------------------------------------------------------------------------;

;{

;DESCRIPTION:       Converts a two-digit binary coded decimal number to binary
;INPUT REGISTERS:   comp
;OUTPUT REGISTERS:  comp

BCDToBinary    swapf     comp, w
               andlw     0x0F
               movwf     temp                ; temp holds LSB of comb
               movlw     0x0F
               andwf     comp, f             ; comp holds MSB

               bcf       STATUS, C
               rlf       temp, f             ; XY = 2(X)+8(X)+Y
               movfw     temp
               addwf     comp, f
               bcf       STATUS, C
               rlf       temp, f
               bcf       STATUS, C
               rlf       temp, f
               movfw     temp
```

```
                addwf       comp, f

                return

;DESCRIPTION:        Calculates the elapsed time of module opening and clsoing
;INPUT REGISTERS:    second, esecond, minute, eminute, hour, ehour
;OUTPUT REGISTERS:   duration

Elapsed         clrf        duration

                movfw       second                  ; convert starting seconds to binary
                movwf       comp
                call        BCDToBinary
                movfw       comp
                movwf       second

                movfw       esecond                 ; convert ending seconds to binary
                movwf       comp
                call        BCDToBinary
                movfw       comp
                movwf       esecond

                movfw       minute                  ; convert starting minutes to binary
                movwf       comp
                call        BCDToBinary
                movfw       comp
                movwf       minute

                movfw       eminute                 ; convert ending minutes to binary
                movwf       comp
                call        BCDToBinary
                movfw       comp
                movwf       eminute

                movfw       hour                    ; convert starting hours to binary
                movwf       comp
                call        BCDToBinary
                movfw       comp
                movwf       hour

                movfw       ehour                   ; convert ending hours to binary
                movwf       comp
                call        BCDToBinary
                movfw       comp
                movwf       ehour

                movfw       second                  ; get difference between seconds
                subwf       esecond, f

                btfss       STATUS, C
                goto        $+2                     ; carry
                goto        $+9                     ; no carry
                movfw       eminute
                btfsc       STATUS, Z               ; if subtrahend is 0, inc minuend
                goto        $+3
                decf        eminute, f              ; else dec subtrahend
                goto        $+2
                incf        minute, f
                movlw       60                      ; add after carry
                addwf       esecond, f

                movfw       minute                  ; get difference between minutes
                subwf       eminute, f

                btfss       STATUS, C
                goto        $+2                     ; carry
                goto        $+4                     ; no carry
                decf        ehour, f                ; decrease elapsed hours
                movlw       60
                addwf       eminute, f              ; add after carry

                movlw       4
                movwf     temp
                incf      eminute, f

FindMinutes     decfsz    eminute, f          ; add 60 while elapsed time < 4 min
                goto      $+2
                goto      FindSeconds         ; less than 4 minutes
                movlw     60
                addwf     duration, f
                decfsz    temp, f
                goto      FindMinutes
                return                        ; greater than 4 minutes

FindSeconds     movfw     esecond             ; duration = 60*min + sec
                addwf     duration, f

                return

;DESCRIPTION:       Converts the elapsed time from binary to decimal values
;INPUT REGISTERS:   duration
;OUTPUT REGISTERS:  hundred, ten, one

GetElapsed      clrf      hundred
                clrf      ten
                clrf      one

                movlw     2
                movwf     temp

HundredLoop     movlw     100                 ; count # of hundreds
                subwf     duration, w
                btfss     STATUS, C
                goto      TenLoop             ; remaining less than 100 seconds
                movwf     duration
                incf      hundred, f
                goto      HundredLoop

TenLoop         movlw     10                  ; count # of tends
                subwf     duration, w
                btfss     STATUS, C
                goto      OneLoop             ; remaining less than 10 seconds
                movwf     duration
                incf      ten, f
                goto      TenLoop

OneLoop         movfw     duration            ; duration - 100*H - 10*T = One
                movwf     one

                return

;}


                end
```

```
;----| SUMMARY |--------------------------------------------------------------------------------;
;                                                                                               ;
;    Author:        Duluxan Sritharan                                                           ;
;    Company:       Team 40                                                                     ;
;    Date:          April 14, 2009                                                              ;
;                                                                                               ;
;    Hardware:      MicroChip PIC16F877                                                         ;
;    Assembler:     mpasm.exe                                                                   ;
;                                                                                               ;
;    Filename:      i2c_common.asm                                                              ;
;    File Version:  Release                                                                     ;
;    Project Files: STRG.asm                                                                    ;
;                   rtc_macros.inc                                                              ;
;                                                                                               ;
;-----------------------------------------------------------------------------------------------;

;----[ CONFIGURATIONS ]-------------------------------------------------------------------------;

;{

     include <p16f877.inc>
     errorlevel     -302
     errorlevel     -305

;}


;----[ GLOBAL LABELS ]--------------------------------------------------------------------------;

;{

     global    write_rtc,read_rtc,rtc_convert,i2c_common_setup

;}


;----[ DEFINITION AND VARIABLE DECLARATIONS ]---------------------------------------------------;

;{

     cblock 0x71                   ;these variable names are for reference only. The following
          dt1            ;0x71      addresses are used for the RTC module
          dt2            ;0x72
          ADD            ;0x73
          DAT            ;0x74
          DOUT           ;0x75
          B1             ;0x76
          dig10          ;0x77
          dig1           ;0x78
     endc

;}


;----[ I2C MACROS ]-----------------------------------------------------------------------------;

;{

;DESCRIPTION:       If bad ACK bit received, goto err_address
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

i2c_common_check_ack     macro     err_address

               banksel   SSPCON2
               btfsc     SSPCON2,ACKSTAT
               goto      err_address

               endm

;DESCRIPTION:       Initiate start condition on the bus
;INPUT REGISTERS:   None
```

```
;OUTPUT REGISTERS:  None

i2c_common_start         macro

               banksel   SSPCON2
               bsf       SSPCON2,SEN
               btfsc     SSPCON2,SEN
               goto      $-1

               endm

;DESCRIPTION:       Initiate stop condition on the bus
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

i2c_common_stop          macro

               banksel   SSPCON2
               bsf       SSPCON2,PEN
               btfsc     SSPCON2,PEN
               goto      $-1

               endm

;DESCRIPTION:       Initiate repeated start on the bus for changing direction of SDA without stop
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

i2c_common_repeatedstart macro

               banksel   SSPCON2
               bsf       SSPCON2,RSEN
               btfsc     SSPCON2,RSEN
               goto      $-1

               endm

;DESCRIPTION:       Send an acknowledge to slave device
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

i2c_common_ack           macro

               banksel   SSPCON2
               bcf       SSPCON2,ACKDT
               bsf       SSPCON2,ACKEN
               btfsc     SSPCON2,ACKEN
               goto      $-1

               endm

;DESCRIPTION:       Send a not acknowledge to slave device
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

i2c_common_nack          macro

               banksel   SSPCON2
               bsf       SSPCON2,ACKDT
               bsf       SSPCON2,ACKEN
               btfsc     SSPCON2,ACKEN
               goto      $-1

               endm

;DESCRIPTION:       Writes W to SSPBUF and send to slave device
;INPUT REGISTERS:   w
;OUTPUT REGISTERS:  SSPBUF

i2c_common_write         macro

               banksel   SSPBUF
```

```
                         SSPBUF
               banksel   SSPSTAT
               btfsc     SSPSTAT,R_W         ; While transmit is in progress, wait
               goto      $-1
               banksel   SSPCON2

               endm

;DESCRIPTION:       Reads data from slave and saves it in W.
;INPUT REGISTERS:   SSPBUF
;OUTPUT REGISTERS:  w

i2c_common_read          macro

               banksel   SSPCON2
               bsf       SSPCON2,RCEN        ; Begin receiving byte from
               btfsc     SSPCON2,RCEN
               goto      $-1
               banksel   SSPBUF
               movf      SSPBUF,w

               endm

;}


     code

;----[ I2C FUNCTIONS ]------------------------------------------------------------------;

;{

;DESCRIPTION:       Sets up I2C as master device with 100kHz baud rate
;INPUT REGISTERS:   None
;OUTPUT REGISTERS:  None

i2c_common_setup

               banksel   SSPSTAT
               clrf      SSPSTAT             ; I2C line levels, and clear all flags
               movlw     d'24'               ; 100kHz baud rate: 10MHz osc / [4*(24+1)]
               banksel   SSPADD
               movwf     SSPADD              ; RTC only supports 100kHz

               movlw     b'00001000'         ; Config SSP for Master Mode I2C
               banksel   SSPCON
               movwf     SSPCON
               bsf       SSPCON,SSPEN        ; Enable SSP module

               i2c_common_stop               ; Ensure the bus is free

               bcf       PCLATH,3
               bcf       PCLATH,4

               return

;}

;----[ RTC FUNCTIONS ]------------------------------------------------------------------;

;{

;DESCRIPTION:       Handles writing data to RTC
;INPUT REGISTERS:   0x73, 0x74
;OUTPUT REGISTERS:  None

write_rtc
               i2c_common_start              ;Select the DS1307 on the bus, in WRITE mode

               movlw     0xD0                ;DS1307 address | WRITE bit

               i2c_common_write
               i2c_common_check_ack   WR_ERR

               banksel   0x73                ;Write data to I2C bus (Register Address in RTC)
               movf      0x73,w              ;Set register pointer in RTC

               i2c_common_write
               i2c_common_check_ack   WR_ERR

               banksel   0x74                ;Write RTC data to I2C bus
               movf      0x74,w              ;Write data to register in RTC

               i2c_common_write
               i2c_common_check_ack   WR_ERR

               goto      WR_END

               WR_ERR

               nop

               WR_END
               i2c_common_stop               ;Release the I2C bus

               bcf       PCLATH,3
               bcf       PCLATH,4

               return

;DESCRIPTION:        This reads from the RTC and saves it into DOUT or address 0x75
;INPUT REGISTERS:    0x73
;OUTPUT REGISTERS:   0x75

read_rtc
               i2c_common_start              ;Select the DS1307 on the bus, in WRITE mode

               movlw     0xD0                ;DS1307 address | WRITE bit

               i2c_common_write
               i2c_common_check_ack   RD_ERR

               banksel   0x73                ;Write data to I2C bus (Register Address in RTC)
               movf      0x73,w              ;Set register pointer in RTC

               i2c_common_write
               i2c_common_check_ack   RD_ERR

               i2c_common_repeatedstart      ;Re-Select the DS1307 on the bus, in READ mode

               movlw     0xD1                ;DS1307 address | READ bit

               i2c_common_write
               i2c_common_check_ack   RD_ERR
               i2c_common_read               ;Read data from I2C bus (Contents of Register in RTC)

               banksel   0x75
               movwf     0x75

               i2c_common_nack               ;Send acknowledgement of data reception

               goto      RD_END

               RD_ERR

               nop

               RD_END    i2c_common_stop     ;Release the I2C bus

               bcf       PCLATH,3
               bcf       PCLATH,4

               return
```

```
;DESCRIPTION:       Converts a binary number into two digit ASCII numbers
;INPUT REGISTERS:   w
;OUTPUT REGISTERS:  0x77, 0x78

rtc_convert
               banksel   0x76
               movwf     0x76                ; B1 = HHHH LLLL
               swapf     0x76,w              ; W  = LLLL HHHH
               andlw     0x0f                ; Mask upper four bits 0000 HHHH
               addlw     0x30                ; convert to ASCII
               movwf     0x77                ;saves into 10ths digit

               banksel   0x76
               movf      0x76,w
               andlw     0x0f                ; w  = 0000 LLLL
               addlw     0x30                ; convert to ASCII
               movwf     0x78                ; saves into 1s digit

               bcf       PCLATH,3
               bcf       PCLATH,4

               return

;}


     end
```

```
;----| SUMMARY |------------------------------------------------------------------;
;                                                                                ;
;    Author:        Duluxan Sritharan                                            ;
;    Company:       Team 40                                                      ;
;    Date:          April 14, 2009                                               ;
;                                                                                ;
;    Hardware:      MicroChip PIC16F877                                          ;
;    Assembler:     mpasm.exe                                                    ;
;                                                                                ;
;    Filename:      rtc_macros.inc                                               ;
;    File Version:  Release                                                      ;
;    Project Files: STRG.asm                                                     ;
;                   i2c_common.asm                                               ;
;                                                                                ;
;--------------------------------------------------------------------------------;

;----[ EXTERNAL LABELS ]----------------------------------------------------------;

;{

    extern    write_rtc,read_rtc,rtc_convert,i2c_common_setup

;}


;----[ RTC MACROS ]---------------------------------------------------------------;

;{

;DESCRIPTION:       Loads the data in datliteral into the address of addliteral in the RTC
;INPUT REGISTERS:   addliteral, datliteral
;OUTPUT REGISTERS:  None

rtc_set        macro     addliteral,datliteral

               banksel   0x73
               movfw     addliteral
               movwf     0x73
               banksel   0x74
               movfw     datliteral
               movwf     0x74
               pagesel   write_rtc
               call      write_rtc

               endm

;DESCRIPTION:       Read RTC at addliteral and convert into both binary and two-digit ASCII
;INPUT REGISTERS:   addliteral
;OUTPUT REGISTERS:  0x75, 0x77, 0x78

rtc_read       macro     addliteral

               movfw     addliteral
               banksel   0x73
               movwf     0x73
               pagesel   read_rtc
               call      read_rtc
               banksel   0x75
               movf      0x75,w
               pagesel   rtc_convert
               call      rtc_convert

               endm

;}
```

APPENDIX H: STANDARD OPERATING PROCEDURES

H.1 ADJUSTMENT TOOLS

These tools are required only by the administrator when configuring and maintaining the system. Alternate tools may be used if so desired.

1. Drill – To fasten the modules together using the tabs or to mount the control module in a desired location.

2. Pocket Screwdriver – To reduce the brightness or contrast on the LCD. Please see datasheet on the PIC DevBugger board for details on how to adjust the potentiometer.

3. Lock – The administrator is expected to provide his/her own lock for securing the control module

4. Batteries/Battery Recharger – The administrator is expected to possess his/her own battery recharger for recharging the given set of batteries at periodic intervals, or to periodically buy new batteries.

5. Lubrication Spray – The hinges of the modules and solenoid pins should periodically lubricated to ensure smooth operation.

H.2 ADMINISTRATOR PROCEDURE

H.2.1 Setting Up the Modules

1. Arrange modules in preferred configuration. Ensure that the surface is flat and that the modules are flush against each other.
2. If using bolts to secure modules, insert a 3/8” diameter screw through the aligned tabs. Tighten a 3/8” nut onto the screw to firmly secure modules in place.
3. If using rope to secure modules, thread rope through the tabs of all modules which must be secured. For best result, attempt to thread rope through all tabs. Secure rope by locking.
4. Unlock the control module. Install batteries in back-up power pack. Ensure that the batteries have been allowed to charge for 3 hours before the first operation.
5. Using the plugs from the rear of each module, connect each module to the ports at the back of the control module. If the plug cannot reach the control module, use an extension cable to complete the connection.
6. Once all of the modules are connected, plug in the power adapter and switch on the power supply.
7. Switch on the DevBugger board. All of the boxes should have their green LEDs flash for a fraction of a second to indicate that they are operational.
8. Lock the control module

H.2.2 Configuring the System

1. For the first boot-up of the user interface, enter the desired administrator ID and password.
2. Adjust the date and time so that system logs created in the future will be accurate
3. Configure the connected modules by selecting OK on the configure screen and following the prompts.

H.2.3 Managing Accounts

1. Select the "Manage Accounts" screen.
2. For each user slot, the administrator may either create or manage an account depending on if the slot is free.
3. To create an account, select Add. The administrator will be prompted for the user's ID, password, expiry and module assignment. The modules that can be assigned are restricted to be a subset of those configured using the "Configure" screen.
4. To manage an account, select Manage. In this menu, you may delete a user, edit information or view system logs.
5. The system logs display the following information in the specified order – the date of module entry, the time of module entry, user or guest access, name of module accessed, and the duration of time for which the module was unlocked.

H.2.4 Changing Password

1. To change administrator password, select '0' at this menu screen.
2. Enter a four-digit alphanumeric code when prompted. Ensure that the administrator ID and password combination does not conflict with that of any user.

H.2.5 Resetting the System

1. To completely reset the system, including deleting system configuration settings, resetting the clock, deleting all accounts and clearing logs, press '0' at this screen.
2. Reset the administrator ID and password as prompted.
3. Reconfigure the system as specified in the 'Configuring the System' section.

H.2.6 Opening/Closing the Module

1. Select the module to be opened in the Open Module menu
2. Press 0 to unlock the module
3. Open the module with the flashing LED within 3 seconds
4. To keep the door ajar, swing the door 90 degrees until the door is held open by the jamming arm
5. When closing the module, press the red button on the outside of the door. Ensure that no obstacles prevent the door from closing.
6. When the door has successfully locked, press any key on the keypad to return to the Open Module menu.

H.3 USER PROCEDURE

H.3.1 Signing In

1. Enter your 4-digit user ID and password at the stand-by menu. The ID and password must be given to you by the administrator

H.3.2 Managing Guest Accounts

1. Enter a four-digit for the guest to use. Set an expiry date and time for the guest. If the expiry date and time for the guest occurs after the expiry time for the user account, both the user and guest will be deleted at the user's expiry.
2. Assign the modules that the guest is intended to have access to. The assigned modules for the guest can only be subset of the modules for which the user has access.

H.3.3 Opening/Closing the Module

1. Select the module to be opened in the Open Module menu
2. Press 0 to unlock the module
3. Open the module with the flashing LED within 3 seconds.
4. To keep the door ajar, swing the door 90 degrees until the door is held open by the jamming arm
5. After 15 seconds, the door will automatically try to close. To prevent the door from closing after this, hold the door open for the necessary amount of time.
6. To close the door before 15 seconds, press the red button on the outside of the door. Ensure that no obstacles prevent the door from closing.
7. When the door has successfully locked, press any key on the keypad to return to the Open Module menu.

H.3.4 Changing Password

1. To change user password, select '0' at this menu screen.
2. Enter a four-digit alphanumeric code when prompted. Ensure that the password does not conflict with any guest password that may be set.

H.4 GUEST PROCEDURE

H.4.1 Signing In

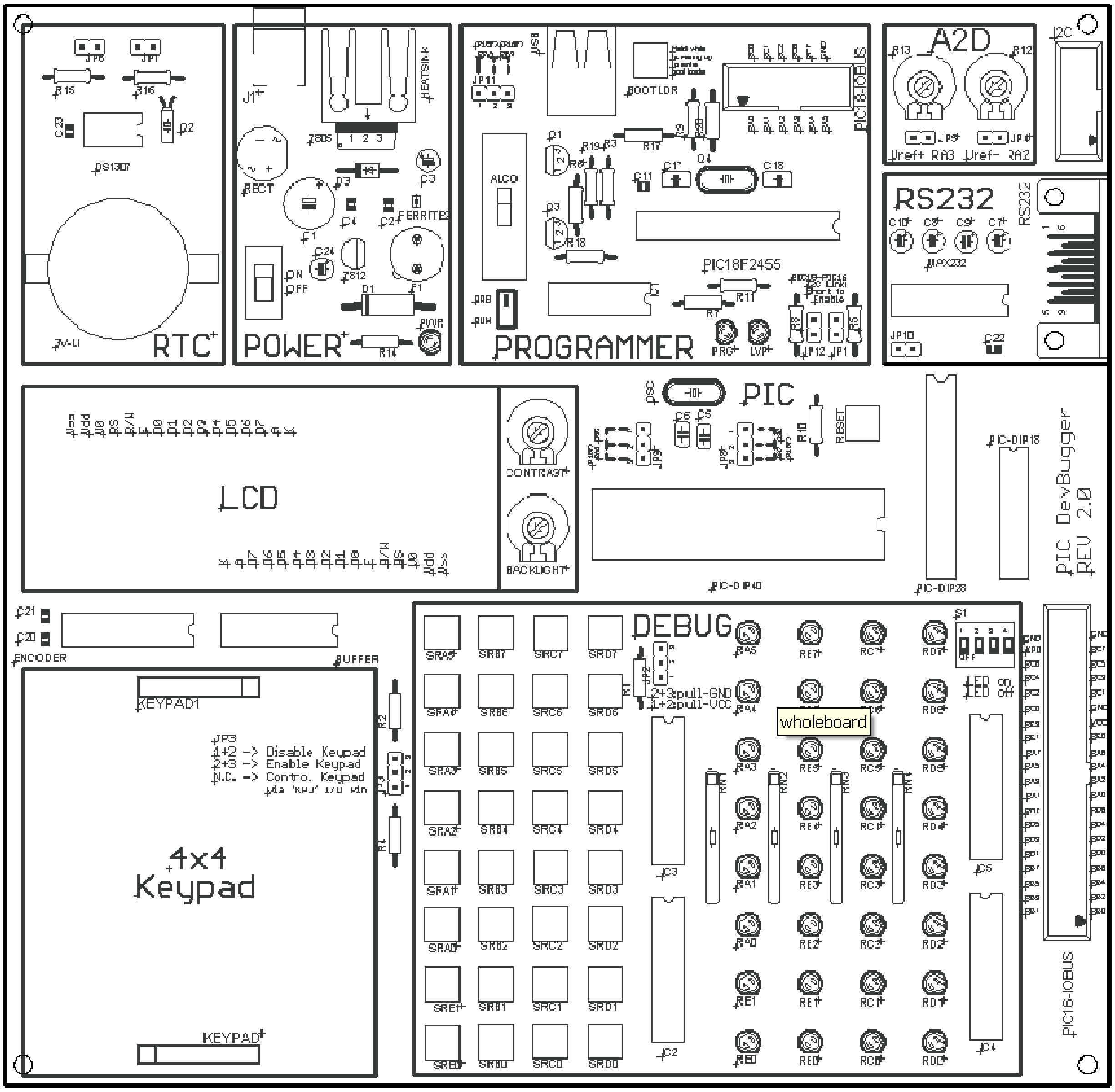
1. Enter your 4-digit user ID and password at the stand-by menu. The ID and password must be given to you by the user
2. If you have entered the correct ID and password, but are still denied access, please contact the user to determine your allotted access time and/or your number of access trials.

H.4.2 Opening/Closing the Module

1. Select the module to be opened in the Open Module menu
2. Press 0 to unlock the module
3. Open the module with the flashing LED within 3 seconds.
4. To keep the door ajar, swing the door 90 degrees until the door is held open by the jamming arm
5. After 15 seconds, the door will automatically try to close. To prevent the door from closing after this, hold the door open for the necessary amount of time.
6. To close the door before 15 seconds, press the red button on the outside of the door. Ensure that no obstacles prevent the door from closing.
7. When the door has successfully locked, press any key on the keypad to return to the Open Module menu.

APPENDIX I: DATASHEETS

PIC DevBugger Manual



MICROCHIP

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

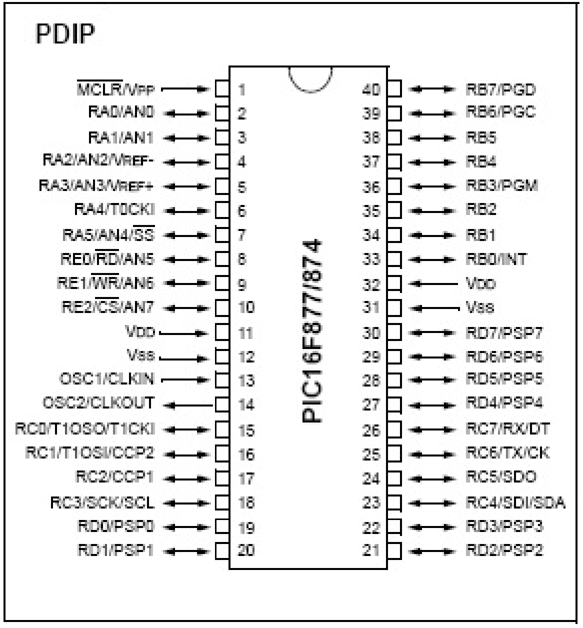
Devices Included in this Data Sheet:

- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
  Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

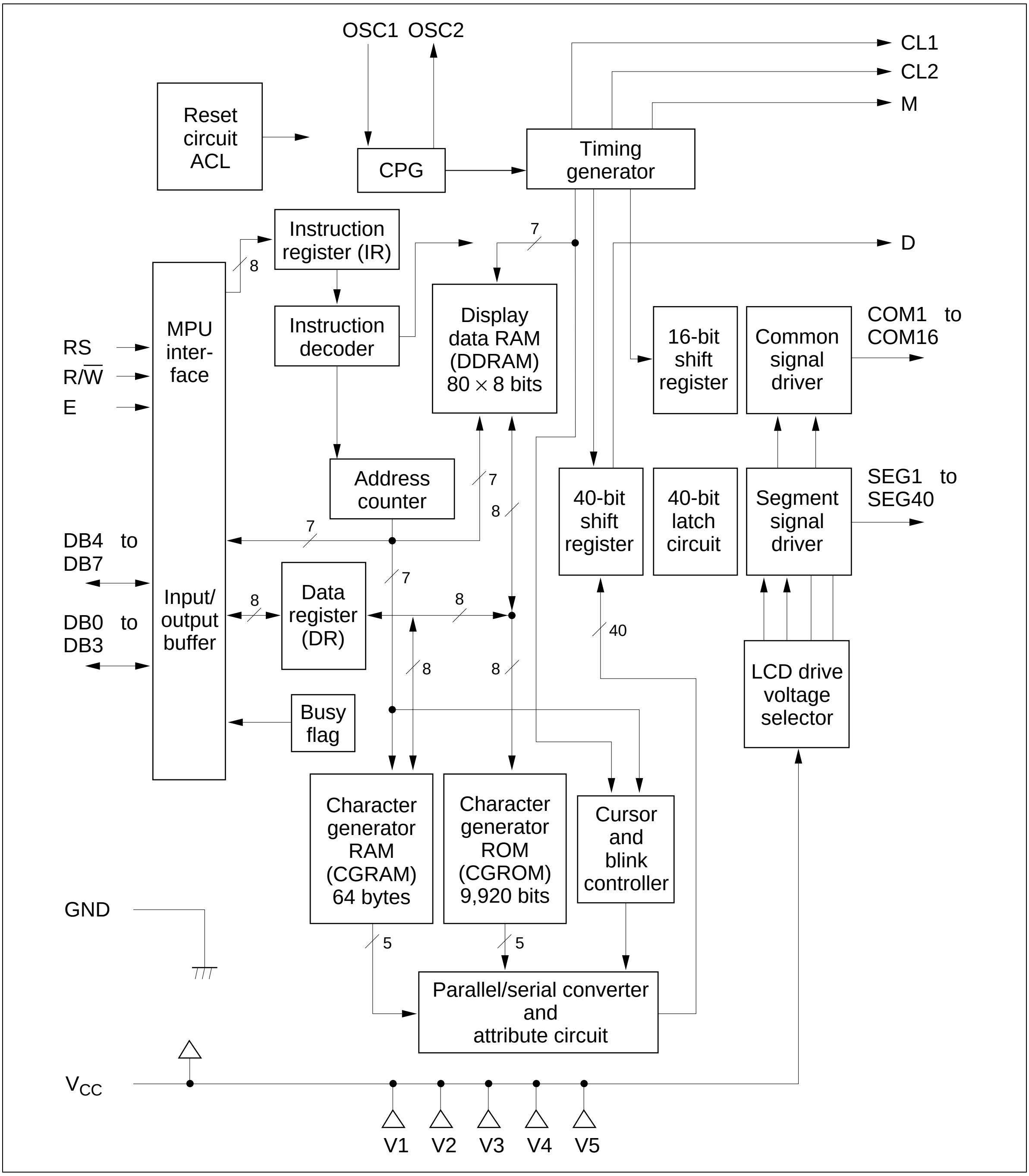
Pin Diagram



Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

HD44780U Block Diagram



National Semiconductor

July 1993

MM54C922/MM74C922 16-Key Encoder MM54C923/MM74C923 20-Key Encoder

General Description

These CMOS key encoders provide all the necessary logic to fully encode an array of SPST switches. The keyboard scan can be implemented by either an external clock or external capacitor. These encoders also have on-chip pull-up devices which permit switches with up to 50 kΩ on resistance to be used. No diodes in the switch array are needed to eliminate ghost switches. The internal debounce circuit needs only a single external capacitor and can be defeated by omitting the capacitor. A Data Available output goes to a high level when a valid keyboard entry has been made. The Data Available output returns to a low level when the entered key is released, even if another key is depressed. The Data Available will return high to indicate acceptance of the new key after a normal debounce period; this two-key roll-over is provided between any two switches.

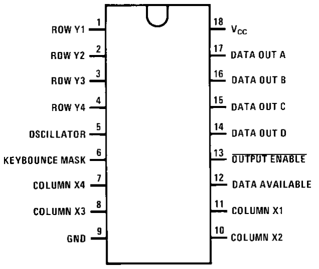
An internal register remembers the last key pressed even after the key is released. The TRI-STATE® outputs provide for easy expansion and bus operation and are LPTTL compatible.

Features

- 50 kΩ maximum switch on resistance
- On or off chip clock
- On-chip row pull-up devices
- 2 key roll-over
- Keybounce elimination with single capacitor
- Last key register at outputs
- TRI-STATE outpust LPTTL compatible
- Wide supply range 3V to 15V
- Low power consumption

Connection Diagrams

Pin Assignment for Dual-In-Line Package

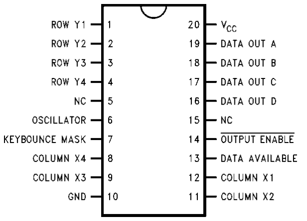


TL/F/6037–1

Top View

Order Number MM54C922 or MM74C922

Pin Assignment for SOIC

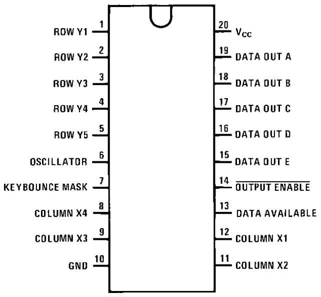


TL/F/6037–14

Top View

Order Number MM74C922

Pin Assignment for DIP and SOIC Package



TL/F/6037–2

Top View

Order Number MM54C923 or MM74C923

TRI-STATE® is a registered trademark of National Semiconductor Corporation.

MM54C922/MM74C922 16-Key Encoder, MM54C923/MM74C923 20-Key Encoder

©1995 National Semiconductor Corporation TL/F/6037

RRD-B30M105/Printed in U. S. A.

DALLAS SEMICONDUCTOR

DS1307
64 x 8 Serial Real-Time Clock

www.maxim-ic.com

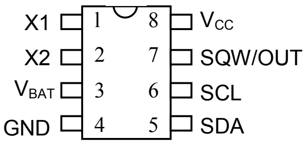
FEATURES

- Real-time clock (RTC) counts seconds, minutes, hours, date of the month, month, day of the week, and year with leap-year compensation valid up to 2100
- 56-byte, battery-backed, nonvolatile (NV) RAM for data storage
- Two-wire serial interface
- Programmable squarewave output signal
- Automatic power-fail detect and switch circuitry
- Consumes less than 500nA in battery backup mode with oscillator running
- Optional industrial temperature range: -40°C to +85°C
- Available in 8-pin DIP or SOIC
- Underwriters Laboratory (UL) recognized

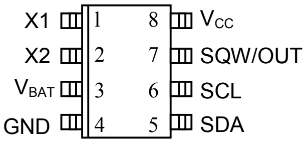
ORDERING INFORMATION

DS1307	8-Pin DIP (300-mil)
DS1307Z	8-Pin SOIC (150-mil)
DS1307N	8-Pin DIP (Industrial)
DS1307ZN	8-Pin SOIC (Industrial)

PIN ASSIGNMENT



DS1307 8-Pin DIP (300-mil)



DS1307 8-Pin SOIC (150-mil)

PIN DESCRIPTION

V_{CC}	- Primary Power Supply
X1, X2	- 32.768kHz Crystal Connection
V_{BAT}	- +3V Battery Input
GND	- Ground
SDA	- Serial Data
SCL	- Serial Clock
SQW/OUT	- Square Wave/Output Driver

DESCRIPTION

The DS1307 Serial Real-Time Clock is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially via a 2-wire, bi-directional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power sense circuit that detects power failures and automatically switches to the battery supply.

ELECTRONICS, INC.
44 FARRAND STREET
BLOOMFIELD, NJ 07003
(973) 748–5089

NTE586
Silicon Rectifier Diode
Schottky Barrier, Fast Switching

Features:

- Low Switching Noise
- Low Forward Voltage Drop
- High Current Capability
- High Reliability
- High Surge Capability

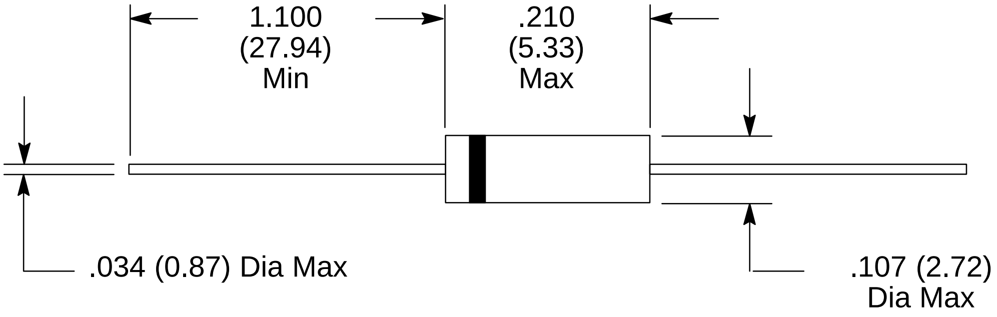
Maximum Ratings and Electrical Characteristics: (T_A = +25°C unless otherwise specified. Single phase, half wave, 60Hz, resistive or inductive load. For capacitive load, derate current by 20%.

Maximum Recurrent Peak Reverse Current	40V
Maximum RMS Voltage	28V
Maximum DC Blocking Voltage	40V
Maximum Average Forward Rectified Current (375” . (9.5mm) lead length at T_L = +95°C).	3.0A
Peak Forward Surge Current (8.3ms single half sine–wave superimposed on rated load T_L = +75°C)	80A
Maximum Instantaneous Forward Voltage at 3A DC (Note 1)	.525V
Maximum Average Reverse Current at Rated DC Blocking Voltage	
T_A = +25°C	1.0mA
T_A = +100°C	10mA
Typical Thermal Resistance, Junction–to–Ambient (Note 2), R_{thJA}	80°C/W
Typical Junction Capacitance (Note 3)	110pF
Operating Junction Temperature Range T_J	–65° to +125°C
Storage Temperature Range T_{STG}	–65° to +125°C

Note 1. measured at Pulse Width 300µs, Duty Cycle 2%.

Note 2. Thermal Resistance Junction to Ambient Vertical PC Board Mounting, 0.5” (12.7mm) Lead Length.

Note 3. Measured at 1MHz and applied reverse voltage of 4.0 Volts.



Color Band Denotes Cathode

SPICE MODELS: 1N4001 1N4002 1N4003 1N4004 1N4005 1N4006 1N4007

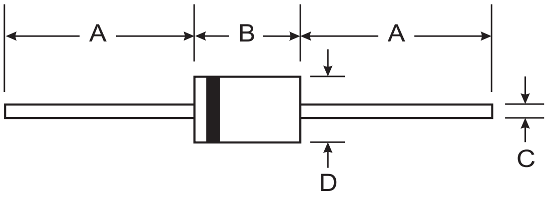
DIODES INCORPORATED

1N4001/L - 1N4007/L

1.0A RECTIFIER

Features

- Diffused Junction
- High Current Capability and Low Forward Voltage Drop
- Surge Overload Rating to 30A Peak
- Low Reverse Leakage Current
- **Lead Free Finish, RoHS Compliant (Note 4)**



Dim	DO-41 Plastic		A-405	
	Min	Max	Min	Max
A	25.40	—	25.40	—
B	4.06	5.21	4.10	5.20
C	0.71	0.864	0.53	0.64
D	2.00	2.72	2.00	2.70
All Dimensions in mm				

"L" Suffix Designates A-405 Package
No Suffix Designates DO-41 Package

Mechanical Data

- Case: DO-41, A-405
- Case Material: Molded Plastic. UL Flammability Classification Rating 94V-0
- Moisture Sensitivity: Level 1 per J-STD-020C
- Terminals: Finish - Bright Tin. Plated Leads Solderable per MIL-STD-202, Method 208
- Polarity: Cathode Band
- Mounting Position: Any
- Ordering Information: See Last Page
- Marking: Type Number
- Weight: DO-41 0.30 grams (approximate)
  A-405 0.20 grams (approximate)

Maximum Ratings and Electrical Characteristics

@ T_A = 25°C unless otherwise specified

Single phase, half wave, 60Hz, resistive or inductive load.
For capacitive load, derate current by 20%.

Characteristic	Symbol	1N 4001/L	1N 4002/L	1N 4003/L	1N 4004/L	1N 4005/L	1N 4006/L	1N 4007/L	Unit
Peak Repetitive Reverse Voltage Working Peak Reverse Voltage DC Blocking Voltage	V_{RRM} V_{RWM} V_R	50	100	200	400	600	800	1000	V
RMS Reverse Voltage	$V_{R(RMS)}$	35	70	140	280	420	560	700	V
Average Rectified Output Current (Note 1) @ T_A = 75°C	I_O	1.0							A
Non-Repetitive Peak Forward Surge Current 8.3ms single half sine-wave superimposed on rated load (JEDEC Method)	I_{FSM}	30							A
Forward Voltage @ I_F = 1.0A	V_{FM}	1.0							V
Peak Reverse Current @ T_A = 25°C at Rated DC Blocking Voltage @ T_A = 100°C	I_{RM}	5.0 50							µA
Typical Junction Capacitance (Note 2)	C_j	15				8			pF
Typical Thermal Resistance Junction to Ambient	$R_{\theta JA}$	100							K/W
Maximum DC Blocking Voltage Temperature	T_A	+150							°C
Operating and Storage Temperature Range (Note 3)	T_j, T_{STG}	-65 to +150							°C

Notes:
1. Leads maintained at ambient temperature at a distance of 9.5mm from the case.
2. Measured at 1. MHz and applied reverse voltage of 4.0V DC.
3. JEDEC Value.
4. RoHS revision 13.2.2003. Glass and High Temperature Solder Exemptions Applied, see *EU Directive Annex Notes 5 and 7.*

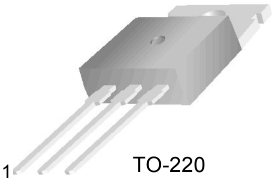
FAIRCHILD
SEMICONDUCTOR®

October 2008

TIP120/TIP121/TIP122

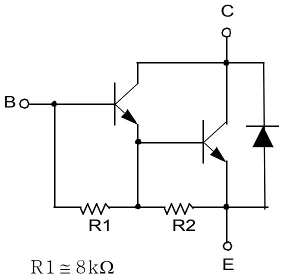
NPN Epitaxial Darlington Transistor

- Medium Power Linear Switching Applications
- Complementary to TIP125/126/127



1.Base 2.Collector 3.Emitter

Equivalent Circuit



R1 ≅ 8kΩ
R2 ≅ 0.12kΩ

Absolute Maximum Ratings* T_a = 25°C unless otherwise noted

Symbol	Parameter	Ratings	Units
V_{CBO}	Collector-Base Voltage : TIP120 : TIP121 : TIP122	60 80 100	V V V
V_{CEO}	Collector-Emitter Voltage : TIP120 : TIP121 : TIP122	60 80 100	V V V
V_{EBO}	Emitter-Base Voltage	5	V
I_C	Collector Current (DC)	5	A
I_{CP}	Collector Current (Pulse)	8	A
I_B	Base Current (DC)	120	mA
P_C	Collector Dissipation (T_a=25°C)	2	W
	Collector Dissipation (T_C=25°C)	65	W
T_J	Junction Temperature	150	°C
T_{STG}	Storage Temperature	- 65 ~ 150	°C

* These ratings are limiting values above which the serviceability of any semiconductor device may be impaired.

© 2007 Fairchild Semiconductor Corporation

Quad 2-input AND gate 74HC/HCT08

PIN DESCRIPTION

PIN NO.	SYMBOL	NAME AND FUNCTION
1, 4, 9, 12	1A to 4A	data inputs
2, 5, 10, 13	1B to 4B	data inputs
3, 6, 8, 11	1Y to 4Y	data outputs
7	GND	ground (0 V)
14	V_{CC}	positive supply voltage

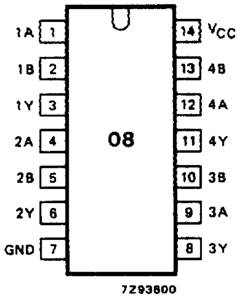


Fig.1 Pin configuration.

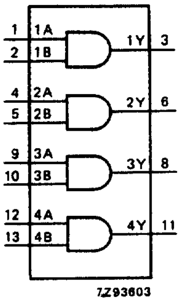


Fig.2 Logic symbol.

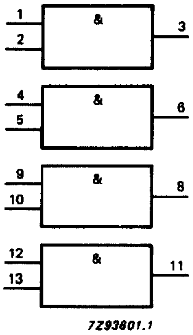


Fig.3 IEC logic symbol.

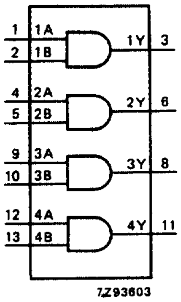


Fig.4 Functional diagram.

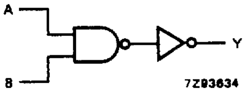


Fig.5 HC logic diagram (one gate).

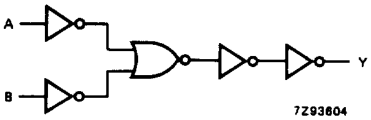


Fig.6 HCT logic diagram (one gate).

FUNCTION TABLE

INPUTS		OUTPUT
nA	nB	nY
L	L	L
L	H	L
H	L	L
H	H	H

Note

1. H = HIGH voltage level
   L = LOW voltage level

STA® Pull Tubular Solenoids — 20 mm Dia. x 40 mm

Part Number: 195224 - X XX

All catalogue products manufactured after April 1, 2006 are RoHS Compliant

XX — Coil AWG Number
(from performance chart below)

X — Plunger Configurations and anti-rotation flat on mounting
1 Flat face plunger without anti-rotation flat
2 60° plunger without anti-rotation flat
5 Flat face plunger with anti-rotation flat
6 60° plunger with anti-rotation flat

- Well-suited for battery operation.

See the "Battery Operated Solenoids" section for complete information.

Performance

Maximum Duty Cycle			100%	50%	25%	10%
Maximum ON Time (sec) when pulsed continuously[1]			∞	230	25	6
Maximum ON Time (sec) for single pulse[2]			∞	265	63	15
Watts (@ 20°C)			7	14	28	70
Ampere Turns (@ 20°C)			855	1200	1700	2700
Coil Data						
awg (0XX)[3]	Resistance (@20°C)	# Turns[4]	VDC (Nom)	VDC (Nom)	VDC (Nom)	VDC (Nom)
24	1.10	330	2.7	3.8	5.6	8.8
25	2.13	488	3.9	5.5	7.7	12.2
26	2.90	544	4.5	6.4	9.0	14.2
27	5.27	760	6.1	8.6	12.1	19.2
28	9.15	1026	8.0	11.3	16.0	25.0
29	12.50	1146	9.4	13.2	18.7	30.0
30	20.70	1491	12.0	17.0	24.0	38.0
31	33.60	1904	15.0	22.0	31.0	48.0
32	53.50	2394	19.4	27.0	39.0	61.0
33	83.50	2970	24.0	34.0	48.0	76.0

[1] Continuously pulsed at stated watts and duty cycle

[2] Single pulse at stated watts (with coil at ambient room temperature 20°C)

[3] Other coil awg sizes available — please consult factory

[4] Reference number of turns

Specifications

Dielectric Strength	1000 VRMS
Recommended Minimum Heat Sink	Maximum watts dissipated by solenoid are based on an unrestricted flow of air at 20°C, with solenoid mounted on the equivalent of an aluminium plate measuring 76 mm square by 3.2 mm thick
Coil Resistance	±5% tolerance
Holding Force	Flat Face: 23.3 N @ 20°C 60°: 12.8 N @ 20°C
Weight	83.6 g
Plunger Weight	20.1 g
Dimensions	See page F29

How to Order

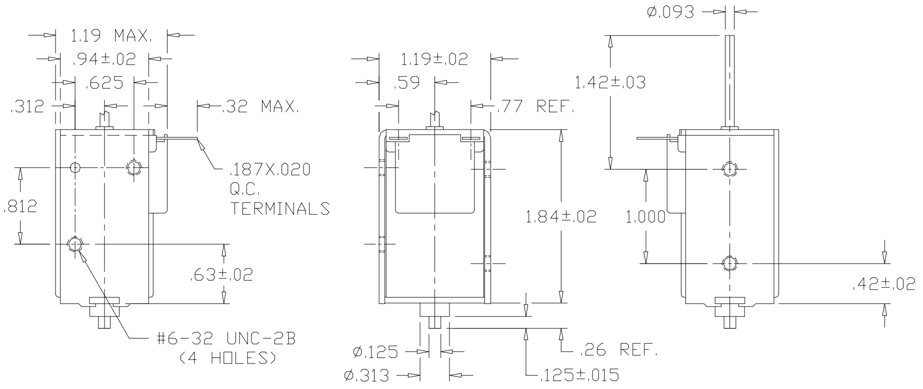
Add the plunger number and the coil awg number to the part number (for example: to order a unit with a 60° plunger configuration without an anti-rotation flat rated for 12 VDC at 25% duty cycle, specify 195224-227.

Please see www.ledex.com (click on Stock Products tab) for our list of stock products available through our distributors.

GUARDIAN
Electric Manufacturing

ISO 9001 / AS9100 Certified Company
Woodstock, IL 60098 Ph. (815)334-3600 Fax (815)337-0377
Or visit us at our website www.guardian-electric.com

Model 11P Push AC Frame Solenoid



Solenoid shown energized and fully seated.

Total Weight:	5.0 oz.
Plunger Weight:	.6 oz.

Model No.	Part No.	Duty Cycle	Volts	Res. (Ω)	Power (VA)	Current, Seated (mA)
11P-I-120A	A420-065707-00	*Intermittent	120	85	40	333
11P-C-120A	A420-065706-00	Continuous	120	225	12	100
11P-I-240A	A420-065709-00	*Intermittent	240	345	40	167
11P-C-240A	A420-065708-00	Continuous	240	920	12	50

When ordering, please refer to Part No., as listed above.
Consult factory for custom configurations.

Push Force (oz.)								Holding Force (oz.)
Stroke (in.)	0.125	0.250	0.375	0.500	0.625	0.750	1.000	-
Continuous Duty	13	11	10	9	8	7	4	12
*Intermittent Duty	24	20	19	18	17	16	8	20

UL Recognition

Recognized under the Component Recognition Program of **Underwriters Laboratories, Inc.**

Continuous Duty

100% 'On' Time

*Intermittent Duty

15% 'On' Time, (240 Seconds 'On' Max. Followed By 1360 Seconds 'Off' Min.)

RoHS

These parts comply with ***RoHS*** Directive 2002/95/EC

QT
OPTOELECTRONICS

CLEAR LENS T-1¾ SOLID STATE LAMPS

ELECTRO-OPTICAL CHARACTERISTICS (25°C Free Air Temeperature)

PARAMETER	TEST COND.	UNITS	MV6152 MV5152	MV6352 MV5352	MV64520 MV5452	MV64521	MV6752 MV5752
Forward voltage (V_F)							
typ.	I_F=20 mA	V	2.0	2.1	2.2	2.2	2.0
max.	I_F=20 mA	V	3.0	3.0	3.0	3.0	3.0
Luminous Intensity							
min.	I_F=20 mA	mcd	17.0	10.0	12.0	30.0	17.0
typ.	I_F=20 mA	mcd	100.0	90.0	25.0	100.0	100.0
Peak wavelength	I_F=20 mA	nm	635	585	562	562	635
Spectral line half width	I_F=20 mA	nm	45	35	30	30	45
Capacitance							
typ.	V=0, f=1 MHz	pF	45	45	20	20	45
Reverse voltage (V_R)							
min.	I_R=100 μA	V	5	5	5	5	5
Reverse current (I_R)							
max.	V_R=5.0 V	μ A	100	100	100	100	100
Viewing angle (total)	See Fig. 4	degrees	28	28	35	35	28

ABSOLUTE MAXIMUM RATINGS (T_A=25°C Unless Otherwise Specified)

	YELLOW	RED AND H. E. RED	GREEN
Power dissipation	85 mW	120 mW	120 mW
Derate linearly from 25°C (MVX452/4A from 50°C)	1.6 mW/°C	1.6 mW/°C	1.6 mW/°C
Storage and operating temperatures	−55°C to +100°C	−55°C to +100°C	−55°C to +100°C
Lead soldering time at 260° C (See Note 2)	5 sec.	5 sec.	5 sec.
Continuous forward current	20 mA	35 mA	30 mA
Peak forward current (1μsec pulse, 0.3% duty cycle)	60 mA	1.0 A	90 mA
Reverse voltage	5.0 V	5.0 V	5.0 V

NOTES

1. *The axis of spatial distribution are typically within a 10° cone within reference to the central axis of the device.*
2. *The leads of the device were immersed in molten solder, at 260°C, to a point 1/16 inch (1.6 mm) from the body of the device per MIl-S-750, with a dwell time of 5 seconds.*

APPENDIX J: BREAK-UP OF TASKS

J.1 TASKS FOR ELECTROMECHANICAL MEMBER

J.1.1 Pre-production

- Outline functionality of each moving component
- Characterize the performance of various actuators
    - Force
    - Power requirement
    - Size
    - Price
- Revise technical drawings based on dimensions of selected motors/solenoids
- Test the pushing/pulling power of solenoids without attachment to circuitry
- Complete “mule” prototype
    - Attach solenoids and springs to module
    - Power with batteries
    - Finalize fabrication techniques
    - Solve geometric constraint issues

J.1.2 Production

- Obtain moving parts and structural components
- Fabricate module one at a time
    - Frame construction
    - Hinge and spring attachment
    - Solenoid attachment
    - Wiring
- Test functionality of each without attaching to circuitry

J.1.3 Post-production

- Module integration and testing
    - Connect to circuit board and microcontroller
- Final troubleshooting

J.2 TASKS FOR CIRCUIT MEMBER

J.2.1 Prototyping

- Obtain required voltage from electromechanical member
- Design overall circuitry
- Explore possibilities of interference
- Design specific circuit diagrams
    - Driver circuit
    - Solenoid circuit

    - Power-battery circuit
    - Manual switch circuit
- Obtain data on voltage, current, power rating for components
- Create circuit on protoboard for testing
    - Testing with MC output signals for driver circuit
    - Testing power supply with multimeter
- Finalize overall voltage, current, power requirement
- Obtain components and parts for soldering

J.2.2 Soldering

- Driver circuit
    - Logical gates
    - connections
- Solenoid circuit
    - Transistors
    - Relays
    - Manual switches
- Testing with MC output signals
- Power-battery circuit
    - Testing power supply with PIC board
- Overall connections

J.2.3 Subsystem Integration and debugging

- Connecting circuits
- Convergence with microcontroller
    - Set up functions to debug and test circuits
    - Assist in creating timing functions to ensure constraints are met
    - Test to see if all signals are amplified correctly
    - Test to see if pushbuttons are functioning correctly
    - Testing power supply
- Interfacing with actuators
    - Test to see if all actuators are driven properly
- Final troubleshooting

J.3 TASKS FOR MICROCONTROLLER MEMBER

J.3.1 Preparation

- Familiarization with PIC and peripheral interfacing
- Problem definition
- Flowchart creation
- Familiarization with MPLAB IDE
- Creation of pseudo-code

J.3.2 User Interface

- Coding main template with basic definitions
- Coding functions for LCD interface
    - Code function to display arbitrary strings
    - Code function to move cursor
- Debugging and integrating LCD interface
- Coding functions for keypad interface
    - Code function to read string
    - Implement process for entering all alphanumeric characters on 4x4 keypad
- Debugging and integrating keypad interface
- Coding functions for menu traversal
    - Code functions to travel up and down menu hierarchy
    - Code functions to scroll up and down on the screen
- Debugging and integrating menu traversal
- Integrating all user interface functions
    - Ensure that what the user types appears appropriately on the LCD
    - Ensure that scrolling and menu traversal works appropriately

J.3.3 Mechanism Interface

- Code for Solenoids
    - Write function to set appropriate pins for solenoids high and low
- Code for Pushbuttons
    - Write function that detect which pushbutton was detected
- Debugging and integrating mechanical interface

J.3.4 Data Structures and Storage

- Coding functions for EEPROM storage
    - Devise hash algorithm for storing account data
    - Write function to traverse and retrieve data from EEPROM
- Coding data structures for account information
    - Devise data structures to store account IDs, passwords and module assignment
    - Write functions to store this data efficiently
- Integrating data structures and data storage
    - Write functions to ensure data structures are stored in RAM properly

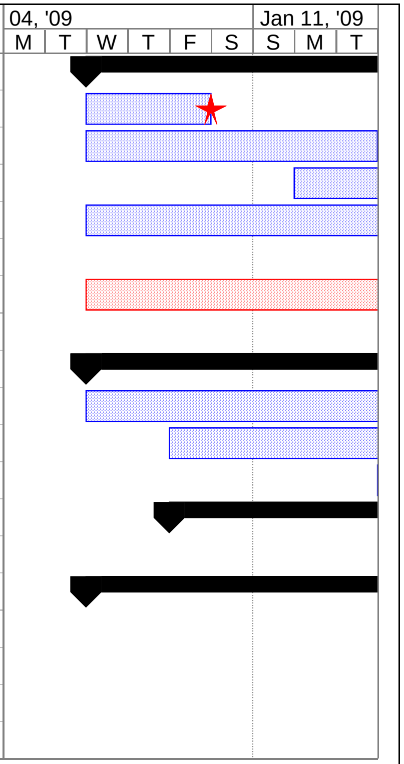
J.3.5 Subsystem Integration and Testing

- Combine user and actuator interface (with Circuit member – MC specific roles listed)
    - Set up functions to convey user input to actuator code
    - Implement procedural logic shown in pseudocode
- Subsystem integration and debugging
    - Test to see if commands on keypad correspond to what is displayed on LCD
    - Test if open module command produces high voltage on correct pins

APPENDIX K: GANTT CHARTS

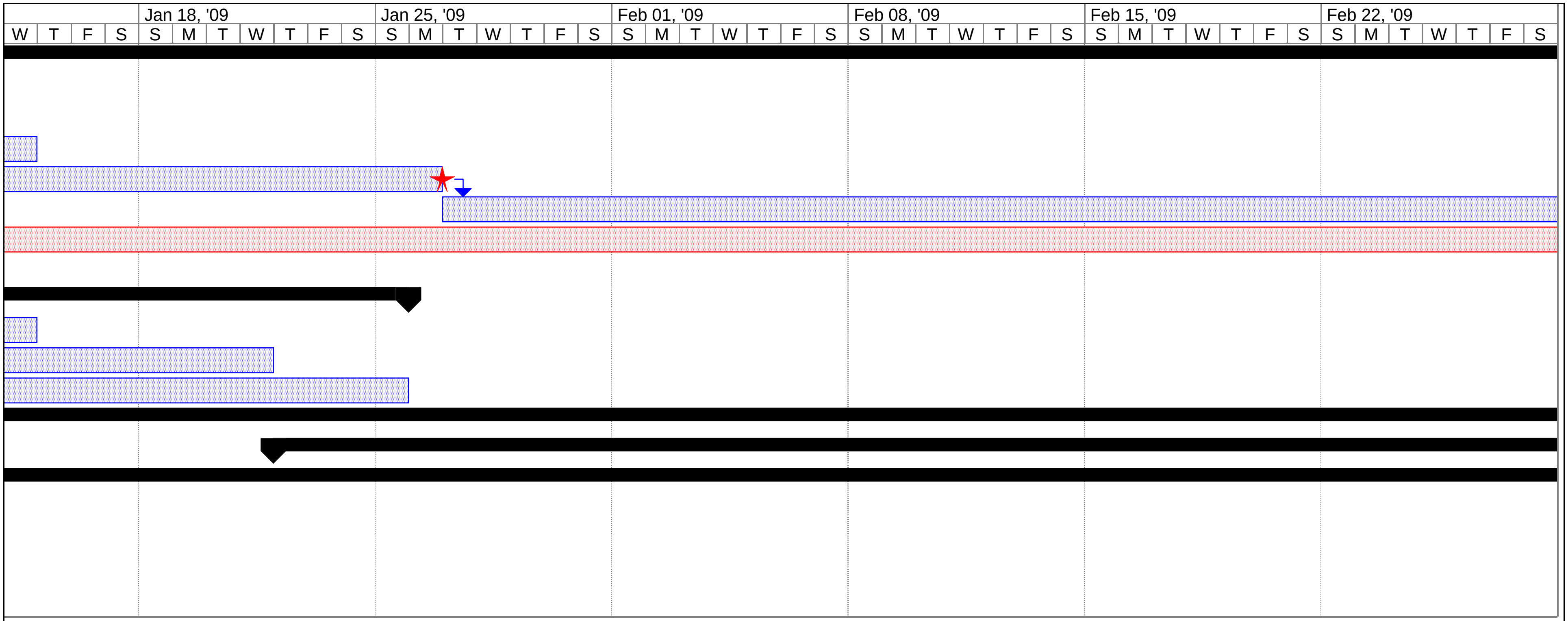
Milestones are indicated as red stars. Where milestones occur in the middle of a prolonged task, the whole task is shown in red. Please consult Section 12 for specific dates. The first GANTT chart is for administrative, conceptual and integration tasks, followed by specific GANTT charts for the electromechanical, circuits, and microcontroller subsystems respectively.

ID	ℹ	Task Name	Duration	Start	Finish	Cost
1		**Administrative**	**98 days**	**Wed 1/07/09**	**Tue 4/14/09**	**$715.00**
2		Finalize teams and assign subsystem responsibilities	3 days	Wed 1/07/09	Fri 1/09/09	$0.00
3		Purchase notebooks and course notes	7 days	Wed 1/07/09	Tue 1/13/09	$165.00
4	▦	Obtain funds for design stores accounts and kits	3 days	Mon 1/12/09	Wed 1/14/09	$550.00
5	▦	Work on design proposal	20 days	Wed 1/07/09	Mon 1/26/09	$0.00
6	▦	Work on final report	78 days	Tue 1/27/09	Tue 4/14/09	$0.00
7		Update and submit notebooks	98 days	Wed 1/07/09	Tue 4/14/09	$0.00
8	▦	Return kits and close store accounts	1 day	Mon 4/13/09	Mon 4/13/09	$0.00
9		**Conceptual**	**19 days**	**Wed 1/07/09**	**Sun 1/25/09**	**$0.00**
10	▦	Understand RFP	8 days	Wed 1/07/09	Wed 1/14/09	$0.00
11	▦	Brainstorm ideas and perform surveys	13 days	Fri 1/09/09	Wed 1/21/09	$0.00
12	▦	Ask for clarification	12 days	Wed 1/14/09	Sun 1/25/09	$0.00
13		**Electromechanical**	**62 days**	**Fri 1/09/09**	**Wed 3/11/09**	**$125.00**
27		**Circuits**	**40 days**	**Thu 1/22/09**	**Mon 3/02/09**	**$140.00**
47		**Microcontroller**	**56 days**	**Wed 1/07/09**	**Tue 3/03/09**	**$20.00**
78		**Integration**	**36 days**	**Wed 3/04/09**	**Wed 4/08/09**	**$45.00**
79	▦	System Integration and Testing - Basic Functionality	8 days	Wed 3/04/09	Wed 3/11/09	$15.00
80		System Integration and Testing - Required Functionality	14 days	Thu 3/12/09	Wed 3/25/09	$15.00
81	▦	Final Debugging and Preparation for Presentation	14 days	Thu 3/26/09	Wed 4/08/09	$15.00



Project: Team 40 - Overall Schedule
Date: Sun 4/12/09

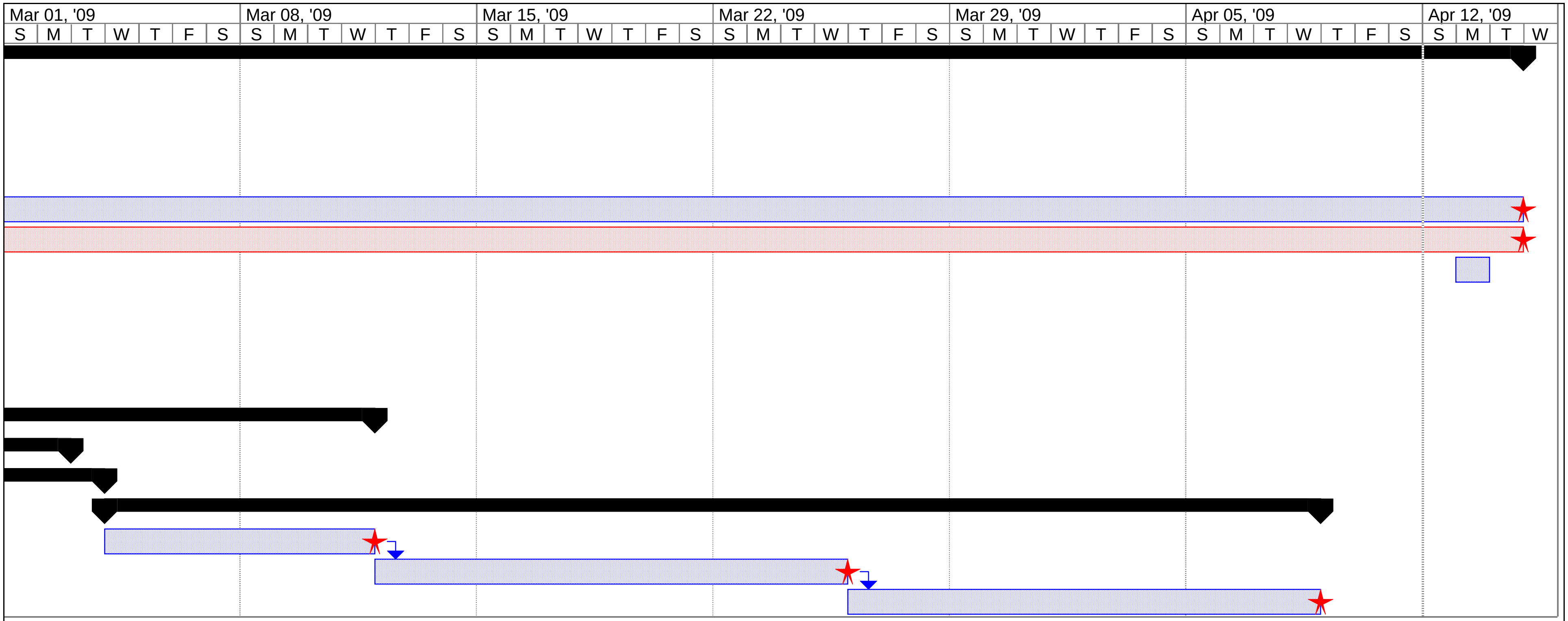
Task	▨	Rolled Up Task	▨	External Tasks	▬
Progress	▬	Rolled Up Milestone	◇	Project Summary	▬
Milestone	◆	Rolled Up Progress	▬	Group By Summary	▬
Summary	▬	Split	┈	Deadline	⇩



Project: Team 40 - Overall Schedule
Date: Sun 4/12/09

Task	▨	Rolled Up Task	▨	External Tasks	▨
Progress	■	Rolled Up Milestone	◇	Project Summary	▬
Milestone	◆	Rolled Up Progress	■	Group By Summary	▬
Summary	▬	Split	┈	Deadline	⇩

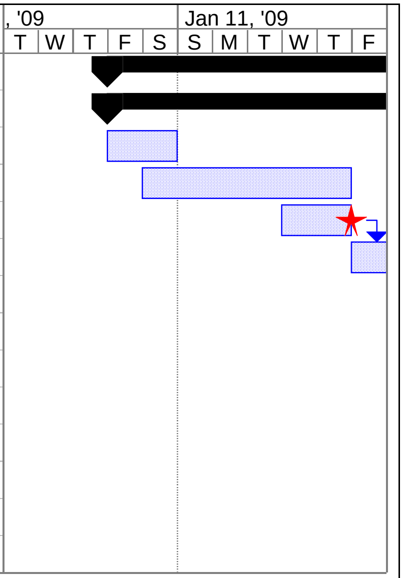
Page 2



Project: Team 40 - Overall Schedule
Date: Sun 4/12/09

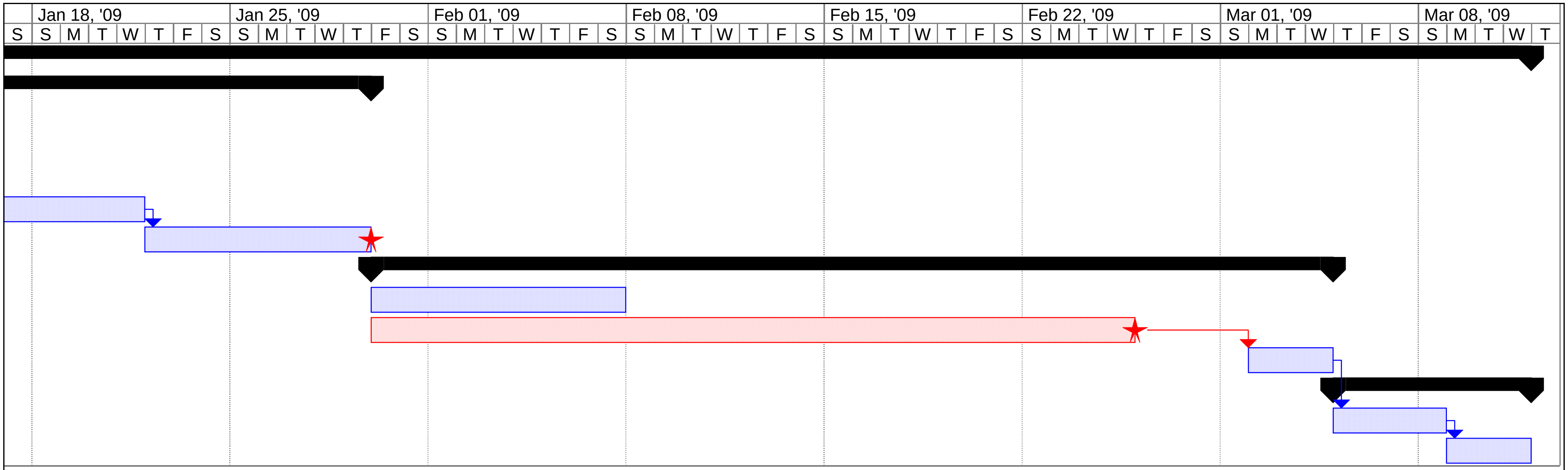
Task	▨	Rolled Up Task	▨	External Tasks	▨
Progress	▬	Rolled Up Milestone	◇	Project Summary	▬
Milestone	◆	Rolled Up Progress	▬	Group By Summary	▬
Summary	▬	Split	┆	Deadline	⇩

ID	ℹ	Task Name	Duration	Start	Finish	Cost
1		**Electromechanical**	**62 days**	**Fri 1/09/09**	**Wed 3/11/09**	**$125.00**
2		**Pre-Production**	**21 days**	**Fri 1/09/09**	**Thu 1/29/09**	**$50.00**
3		Outline functionality of each moving component	2 days	Fri 1/09/09	Sat 1/10/09	$0.00
4		Characterize the performance of various actuators	6 days	Sat 1/10/09	Thu 1/15/09	$30.00
5		Revise technical drawings based on dimensions of selected actu	2 days	Wed 1/14/09	Thu 1/15/09	$0.00
6		Test the power of solenoids without attachment to circuitry	6 days	Fri 1/16/09	Wed 1/21/09	$0.00
7		Complete "mule prototype	8 days	Thu 1/22/09	Thu 1/29/09	$20.00
8		**Production**	**34 days**	**Fri 1/30/09**	**Wed 3/04/09**	**$60.00**
9		Obtain moving parts and structural components	9 days	Fri 1/30/09	Sat 2/07/09	$60.00
10		Fabricate module one at a time	27 days	Fri 1/30/09	Wed 2/25/09	$0.00
11		Test functionality of each without attaching to circuitry	3 days	Mon 3/02/09	Wed 3/04/09	$0.00
12		**Post-Production**	**7 days**	**Thu 3/05/09**	**Wed 3/11/09**	**$15.00**
13		Module integration and testing	4 days	Thu 3/05/09	Sun 3/08/09	$0.00
14		Final troubleshooting	3 days	Mon 3/09/09	Wed 3/11/09	$15.00



Project: Team 40 - Electromech Schec
Date: Sun 4/12/09

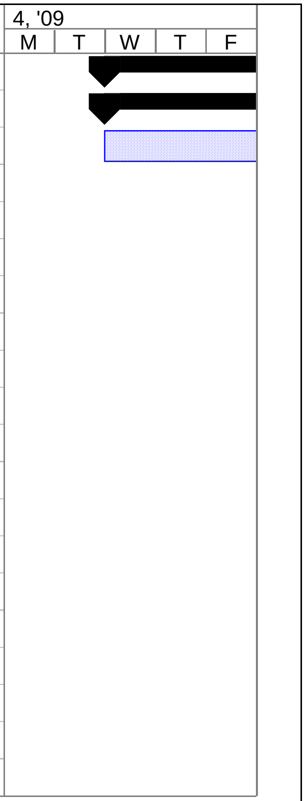
Task	▨	Rolled Up Task	▨	External Tasks	▨
Progress	━	Rolled Up Milestone	◇	Project Summary	▼━▼
Milestone	◆	Rolled Up Progress	━	Group By Summary	▼━▼
Summary	▼━▼	Split	┈	Deadline	⇩



Project: Team 40 - Electromech Schec
Date: Sun 4/12/09

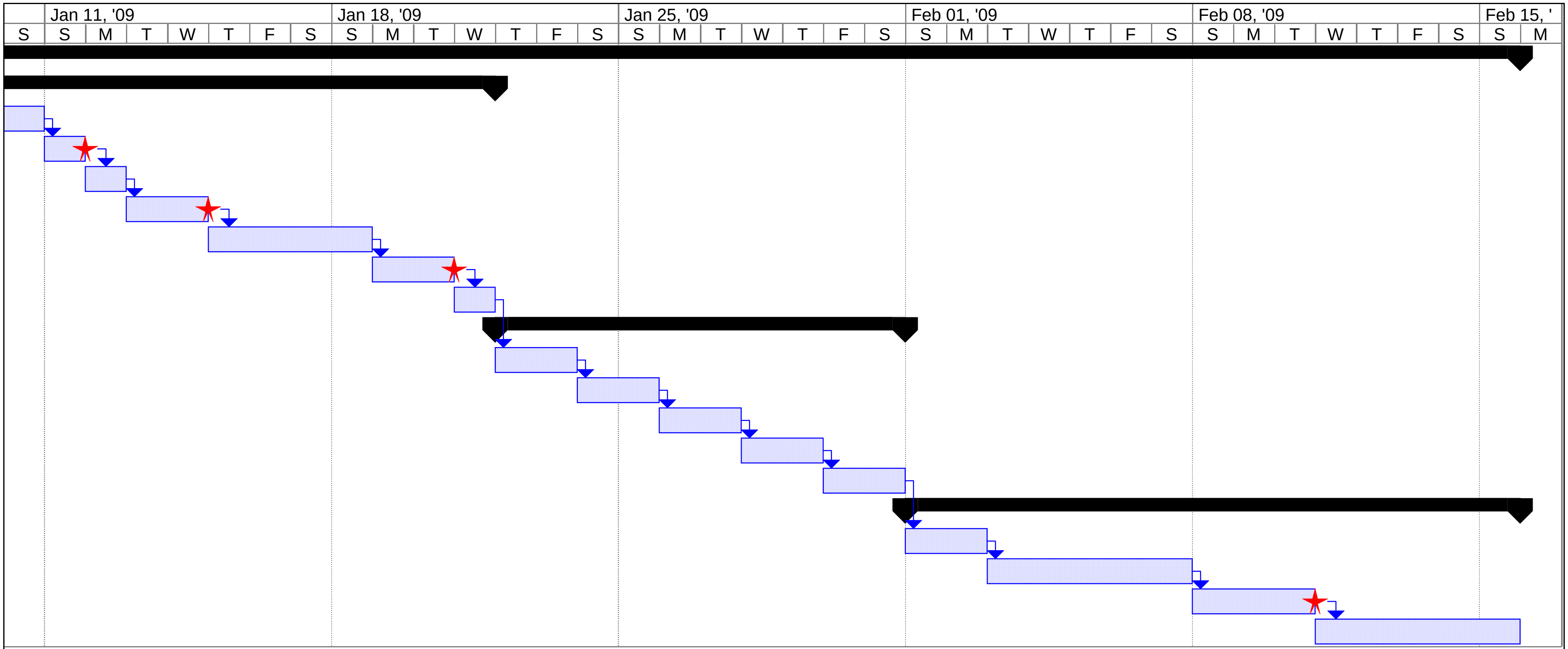
Task	▨	Rolled Up Task	▨	External Tasks	▨
Progress	▬	Rolled Up Milestone	◇	Project Summary	▬
Milestone	◆	Rolled Up Progress	▬	Group By Summary	▬
Summary	▬	Split	┈	Deadline	⇩

ID	Task Name	Duration	Start	Finish	Cost	Predecessors
1	**Circuits**	**40 days**	**Wed 1/07/09**	**Sun 2/15/09**	**$140.00**	
2	**Prototyping**	**15 days**	**Wed 1/07/09**	**Wed 1/21/09**	**$80.00**	
3	Obtain required voltage from electromechanical member	4 days	Wed 1/07/09	Sat 1/10/09	$0.00	
4	Design overall circuitry	1 day	Sun 1/11/09	Sun 1/11/09	$0.00	3
5	Explore possibilities of interference	1 day	Mon 1/12/09	Mon 1/12/09	$0.00	4
6	Design specific circuit diagrams	2 days	Tue 1/13/09	Wed 1/14/09	$0.00	5
7	Create circuit on protoboard for testing	4 days	Thu 1/15/09	Sun 1/18/09	$40.00	6
8	Finalize overall voltage, current and power requirements	2 days	Mon 1/19/09	Tue 1/20/09	$0.00	7
9	Obtain components and parts for soldering	1 day	Wed 1/21/09	Wed 1/21/09	$40.00	8
10	**Soldering**	**10 days**	**Thu 1/22/09**	**Sat 1/31/09**	**$60.00**	
11	Driver Circuit	2 days	Thu 1/22/09	Fri 1/23/09	$5.00	9
12	Solenoid Circuit	2 days	Sat 1/24/09	Sun 1/25/09	$5.00	11
13	Testing with MC output signals	2 days	Mon 1/26/09	Tue 1/27/09	$25.00	12
14	Power-battery circuit	2 days	Wed 1/28/09	Thu 1/29/09	$25.00	13
15	Overall connections	2 days	Fri 1/30/09	Sat 1/31/09	$0.00	14
16	**Subsystem Integration and Debugging**	**15 days**	**Sun 2/01/09**	**Sun 2/15/09**	**$0.00**	
17	Connecting circuits	2 days	Sun 2/01/09	Mon 2/02/09	$0.00	15
18	Convergence with Microcontroller	5 days	Tue 2/03/09	Sat 2/07/09	$0.00	17
19	Interfacing with Actuators	3 days	Sun 2/08/09	Tue 2/10/09	$0.00	18
20	Final Troubleshooting	5 days	Wed 2/11/09	Sun 2/15/09	$0.00	19



Project: Team 40 - Circuit Schedule
Date: Sun 4/12/09

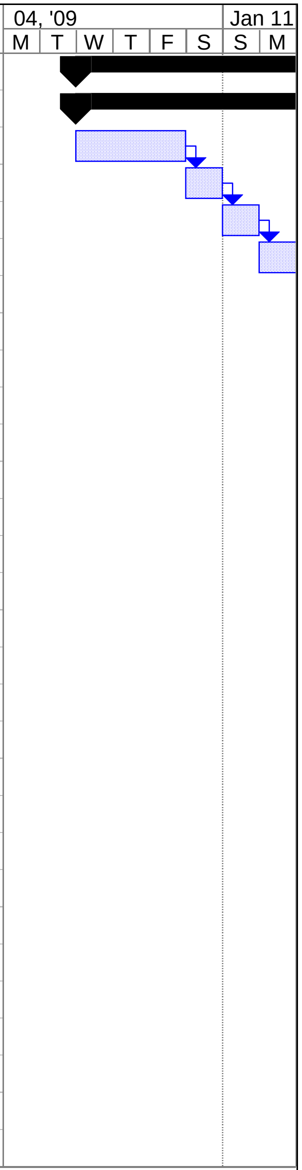
Task	▨	Rolled Up Task	▨	External Tasks	▨
Progress	■	Rolled Up Milestone	◇	Project Summary	▬
Milestone	◆	Rolled Up Progress	■	Group By Summary	▬
Summary	▬	Split	┈	Deadline	⇩



Project: Team 40 - Circuit Schedule
Date: Sun 4/12/09

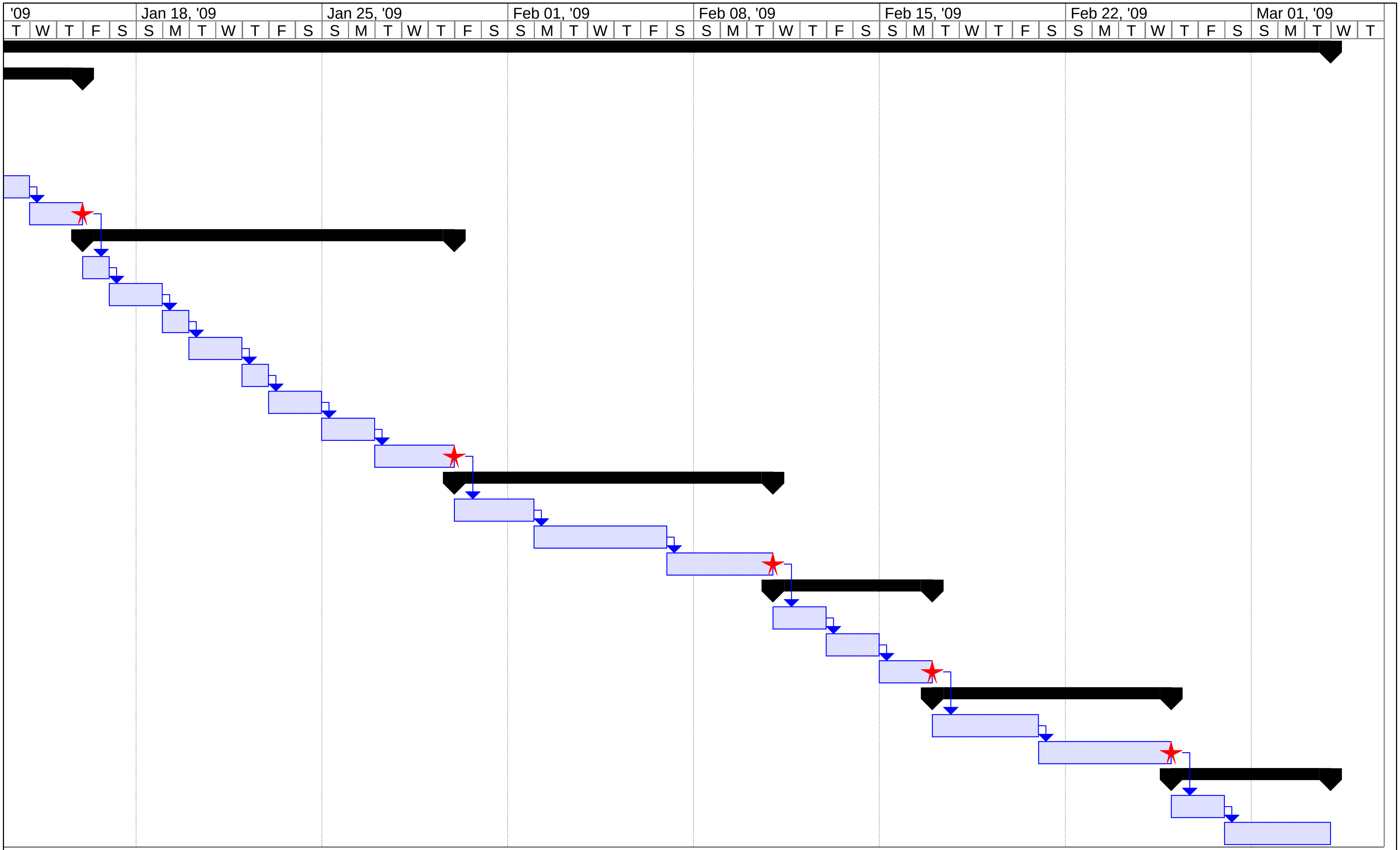
Task	▨	Rolled Up Task	▨	External Tasks	▨
Progress	▬	Rolled Up Milestone	◇	Project Summary	▬
Milestone	◆	Rolled Up Progress	▬	Group By Summary	▬
Summary	▬	Split	┄	Deadline	⇩

ID	Task Name	Duration	Start	Finish	Cost	Predecessors	Resource Names
1	**Microcontroller**	**56 days**	**Wed 1/07/09**	**Tue 3/03/09**	**$20.00**		
2	**Preparation**	**9 days**	**Wed 1/07/09**	**Thu 1/15/09**	**$0.00**		
3	Familiarization with PIC and peripheral interfacing	3 days	Wed 1/07/09	Fri 1/09/09	$0.00		MC
4	Problem definition	1 day	Sat 1/10/09	Sat 1/10/09	$0.00	3	All
5	Flowchart creation	1 day	Sun 1/11/09	Sun 1/11/09	$0.00	4	MC
6	Familiarization with MPLAB IDE	2 days	Mon 1/12/09	Tue 1/13/09	$0.00	5	MC
7	Creation of pseudo-code	2 days	Wed 1/14/09	Thu 1/15/09	$0.00	6	MC
8	**User Interface**	**14 days**	**Fri 1/16/09**	**Thu 1/29/09**	**$0.00**		
9	Coding main template with basic definitions	1 day	Fri 1/16/09	Fri 1/16/09	$0.00	7	MC
10	Coding functions for LCD interface	2 days	Sat 1/17/09	Sun 1/18/09	$0.00	9	MC
11	Debugging and integrating LCD interface	1 day	Mon 1/19/09	Mon 1/19/09	$0.00	10	MC
12	Coding functions for keypad interface	2 days	Tue 1/20/09	Wed 1/21/09	$0.00	11	MC
13	Debugging and integrating keypad interface	1 day	Thu 1/22/09	Thu 1/22/09	$0.00	12	MC
14	Coding functions for menu traversal	2 days	Fri 1/23/09	Sat 1/24/09	$0.00	13	MC
15	Debugging and integrating menu traversal	2 days	Sun 1/25/09	Mon 1/26/09	$0.00	14	MC
16	Integrating all user interface functions	3 days	Tue 1/27/09	Thu 1/29/09	$0.00	15	MC
17	**Data Structures and Storage**	**12 days**	**Fri 1/30/09**	**Tue 2/10/09**	**$0.00**		
18	Coding functions for EEPROM storage	3 days	Fri 1/30/09	Sun 2/01/09	$0.00	16	MC
19	Coding data structures and account information	5 days	Mon 2/02/09	Fri 2/06/09	$0.00	18	MC
20	Integrating data structures and data storage	4 days	Sat 2/07/09	Tue 2/10/09	$0.00	19	MC
21	**Mechanism Interface**	**6 days**	**Wed 2/11/09**	**Mon 2/16/09**	**$0.00**		
22	Code for Solenoids	2 days	Wed 2/11/09	Thu 2/12/09	$0.00	20	MC
23	Code for pushbuttons	2 days	Fri 2/13/09	Sat 2/14/09	$0.00	22	MC
24	Debugging and integrating mechanical interface	2 days	Sun 2/15/09	Mon 2/16/09	$0.00	23	MC
25	**Subsystem Integration and Testing**	**9 days**	**Tue 2/17/09**	**Wed 2/25/09**	**$20.00**		
26	Combine user and actuator interface	4 days	Tue 2/17/09	Fri 2/20/09	$10.00	24	MC,CCT
27	Subsystem Integration and Debugging	5 days	Sat 2/21/09	Wed 2/25/09	$10.00	26	MC,CCT
28	**Bonus Features**	**6 days**	**Thu 2/26/09**	**Tue 3/03/09**	**$0.00**		
29	Date/Time	2 days	Thu 2/26/09	Fri 2/27/09	$0.00	27	
30	PC Interface 1	4 days	Sat 2/28/09	Tue 3/03/09	$0.00	29	



Project: Team 40 - Microcontroller Sch
Date: Sun 4/12/09

Task	▨	Rolled Up Task	▨	External Tasks	▨
Progress	▬	Rolled Up Milestone	◇	Project Summary	▬
Milestone	◆	Rolled Up Progress	▬	Group By Summary	▬
Summary	▬	Split	┅	Deadline	⇩



Project: Team 40 - Microcontroller Sch
Date: Sun 4/12/09

Task	▨	Rolled Up Task	▨	External Tasks	▭
Progress	▬	Rolled Up Milestone	◇	Project Summary	▬
Milestone	◆	Rolled Up Progress	▬	Group By Summary	▬
Summary	▬	Split	┈	Deadline	⇩