GRID INDEPENDENT FUEL CELL OPERATED SMART HOME (GIFCOSH)

Final Report
For period June 15, 2002 – December 15, 2003

Dr. M.S. Alam

University of South Alabama,
Mobile, Al 36688

December, 2003

Prepared for

THE U.S. DEPARTMENT OF ENERGY

AWARD NO. DE-FG02-02ER63376

NOTICE

UNIVERSITY OF SOUTH ALABAMA

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

GRID INDEPENDENT FUEL CELL OPERATED SMART HOME (GIFCOSH)

Submitted by:

Dr. Mohammad S. Alam (PI/PD)
University of South Alabama
College of Engineering
ECE Department, EEB 75
Mobile, AL 36688-0002

Phone: 251-460-6117
Fax: 251-460-6028

Email: malam@usouthal.edu

December 7, 2003

CONTENTS

Project Title: Grid Independent Fuel Cell Powered Home Using Smart Energy Management Control System

PI/PD: Dr. M. S. Alam

Period: June 15, 2002 – September 15, 2003

Recipient Organization:
University of South Alabama, College of Engineering,
ECE Department, EEB #75, Mobile, AL 36688-0002

DOE Award Number: DE-FG02-02ER63376

Unexpended Funds: $197.52

YEAR I PROJECT FINAL REPORT

Year I funding supports the design, development and demonstration of the efficiency, utility and reliability of a fuel cell power plant (FCPP) supplying the power needs of a laboratory based 'home' that utilizes a Smart Energy Management Control (SEMaC) system.

1. System Design Study

Requirements Definition:
The following requirements were defined: vision of the demonstration, safety, reliability and redundancy, the economic target, lab connectivity, the technology time target, the data needs, and the performance metrics. The report has been submitted.

1.1. Energy Supply System Options have been determined. Natural gas has been chosen as the energy supply. The Plug Power FCPP can be run on either natural gas or propane. Natural gas was chosen since it is available on campus.

1.2. Control System Options have been determined. See Appendix A (SEMaC) and Appendix B (System Hardware Controller).

1.3. A Diagnostic Suite has been defined. See Appendix C (DAQ).

2. Laboratory Development

A 10,000 sq. ft. laboratory has been constructed in the existing Laboratory Building of the College of Engineering at the University of South Alabama. Inside the laboratory a 500 sq. ft. test home, consisting of two rooms, has been constructed (Fig. 1). One room is fitted with appliances normally found in the kitchen area of a home, while the other room has an entertainment system, a computer and a plasma display. Power is supplied by a natural gas FCPP, located outside the Laboratory Building, which is connected in parallel with the local grid. All appliances and outlets will be under the control of the SEMaC system.

3. Physical Home/Site Design

This task has been cancelled due to insufficient funding received for Year II of the project.

4. Systems Development

4.1. An appliance suite was chosen based on**:**

- Consumer Reports
- Internet surveys

The suite consists of a washer, dryer, stove, refrigerator, dishwasher, water heater, microwave, iron, vacuum cleaner, blow dryer, an entertainment system, computer equipment and a plasma monitor.

4.1.1. Load Profile Data

In order to obtain representative user power consumption data, load measuring equipment was purchased. A dedicated weather-proof cabinet has been constructed to hold the load measuring equipment. Load profiles of a number of all electric homes have been recorded over a two-day period at fifteen second intervals. The average power consumed has been less than 2 kW in most cases. Peak power consumption greater than 5kW has been observed in all cases. The data collection exercise is ongoing.

4.1.2. System Transient Analysis

Transient data related to air conditioning start-up have been obtained from the Co-operative Rural Network (CRN) Residential Fuel Cell Users Group. These results show a start-up transient current of about 7.5 times the steady-state or run current. There are negligible changes in the grid supply voltage and hence the power surge is also about 7.5 times the steady-state or run power. This extra power is required for about 0.75 seconds and can be supplied by the grid in the grid-parallel arrangement. This reduces the stress on the intermediate energy storage system, with the grid effectively acting as an infinite storage device.

4.2. Energy Supply System

This is an extremely important area of the project. Considerable investigative effort has been directed towards understanding and planning of the energy supply system. The energy supply system can be broadly categorized as:

- Primary energy source.
- Intermediate energy source.
- Backup energy source.

4.2.1. Fuel Cell/Reformer System Selection

The fuel cell/reformer system, namely the FCPP, is the primary energy source of the project. There are currently six different types of fuel cells in use in industry. Each type of fuel cell has its advantages and disadvantages. For instance, some are less efficient but operate at reasonably low temperatures (50 – 80^{o}C), (25% electrical efficiency), while others are more efficient but operate at moderate temperatures (200^{o}C), (50% electrical efficiency), while others are even more efficient but operate at extremely high temperatures (800 – 1000^{o}C), (70% electrical efficiency). After careful analysis of the marketplace a natural gas driven 5kW unit with waste heat utilization capability from Plug Power has been chosen. There are three fixed outputs available, namely, 2.5 kW, 4.0 kW and 5.0 kW, with electrical efficiencies of 26%, 25% and 23.5%, respectively. With waste heat utilization, the efficiency of the unit increases to 60%, 65% and 55%, at set points of 2.5 kW, 4.0 kW and 5kW, respectively.

The FCPP is grid parallel but is unable to load follow, except in the standalone mode. Hence, the local grid must meet all demands, both transient and steady state, above the set level. Based on available data, this unit should easily meet the needs of the laboratory 'home'.

An agreement has been being entered into between the University of South Alabama and Plug Power Inc to share operational data on the FCPP.

4.2.2. Intermediate Storage System

Intermediate storage is supplied by a battery bank, consisting of four 12 V deep cycle batteries. Their main use is to stabilize the 48 V DC bus and to supply energy for transients when the local grid is down. Their energy storage capacity, when fully charged, is about 4,000 Wh or 4 kW for 1 hour.

4.2.3. Backup Energy Source

A back-up system using a natural gas generator may be employed in the event that the local grid is down.

4.3 System Modeling and Simulation

A comprehensive literature search was undertaken and based on the results of this search, a Proton Exchange Membrane (PEM) fuel cell stack was modeled using MATLAB. Two models have been developed to study operational performance under steady-state and transient conditions. Using characteristic curves obtained from the **steady-state model**, a theoretical 5 kW FCPP was designed. It consists of 100 individual stacks, connected in series, each with a membrane area of 100 cm^2. Each stack supplies 100 A at 0.55 V. Hence, 100 stacks in series will nominally supply 100 A at 55 V or 5.5kW of power. Losses of 0.5 kW in the DC/DC converter and the inverter

are assumed. Then, the efficiency and consumption of hydrogen and natural gas versus power were obtained from the model, using the design parameters chosen. A consumption rate of 1.3 ft^3/min of natural gas was obtained from the theoretical design, operating at a nominal 5 kW load output. The **transient model** includes the methanol reformer, the PEM stack and the power conditioning unit. The model is then used to predict the output voltage and study the transient response of a PEM power plant when subjected to rapid changes in a residential load connected to it. The results show the fast response capabilities of the PEM power plant in following changes in the load. A third model has been developed to investigate **strategies for active and reactive power control of a PEM fuel cell.** The model gives a scenario for controlling both active and reactive power output from the fuel cell power plant. The model is then used to predict the output voltage, the active power and the current when the fuel cell power plant is subjected to rapid changes in a load connected to it. The results show quick response and effectiveness of the proposed scheme to control the active and reactive output power.

4.4 Control System Hardware

4.4.1 Control Architecture Design, Hardware and Software Selection

Hardware was developed to monitor the current in each load and to switch off such loads if the need arises (for example, if the grid is down or a room is unoccupied). The monitoring hardware consists of current sensing coils attached to each load circuit, a multiplexer and an ultra-fast analog-to-digital (A/D) converter board. The A/D board resides inside a host computer. The controller hardware consists of a Motorola ColdFire processor reading switch positions, switching loads on or off, and sending and receiving information over an Ethernet connection to the host computer. Over-ride and reset switches are available at each load, which allows the user to cancel the computer command and return power to the load or appliance. The controller has the ability to phase control a load, such as incandescent lights, so as to dim them to the extent desired.

Also, hardware has been developed to monitor occupancy of each room, so that power can be cut off to this room, depending on the length of time the room remains unoccupied. Temperature and humidity sensors, connected to the host computer, are also present in each room.

4.4.2. Control Algorithm Development

Algorithms have been developed that utilize the data being collected from, firstly, the ‘house’ and, secondly, from the energy supply system, and when coupled with the predicted usage data, results in the most efficient use of the energy supply system. These algorithms are critical to the efficient use of the FCPP, maintaining the batteries in good condition, as well as providing reliable energy to serve the needs of the house dwellers.

4.4.3. Software Testing

Software testing is being carried out as each new module is developed. All possible scenarios are currently being investigated.

4.4.4. Diagnostic System Design

The Plug Power unit comes with built in diagnostic software that allows test personnel to safely start and shutdown the system.

4.4.5. Data Recording, Data Mining and Reporting System

FCPP data is recorded on hard disc. Since large streams of data are generated, data mining techniques are being employed to extract trends and statistical information.

4.5 Control System Software

4.5.1 Architecture Design, Platform and Language Selection

The Motorola ColdFire microprocessor's and host computer's software have been developed using C++. Decisions made by the host computer are displayed, by means of a plasma display, using a Graphical User Interface (GUI). The GUI software is written in C++. The GUI also allows control of appliances from the host computer. See Appendix D (GUI).

4.5.2 Models

4.5.2.1 Load Profiles

Load profiles, available from existing databases, were judged inadequate because of their coarse granularity. Hardware and software have been purchased and power surveys have been conducted over a number of days, on a number of homes, using a 15 second time stamp.

4.5.2.2 Energy Supply System

A grid connection and/or backup generator is present to supply power to the instrumentation. This enables continuous data acquisition and activation of safety shut down procedures in the event of a problem developing in the system.

5. Meetings with Strategic Players

In any development project, it is critical to understand the market for the end product. The CRN and the Residential Fuel Cell Owner's Group were involved from the outset of

the project. Meetings were also held during the course of Year 1 with the Houston Area Research Center (HARC) and the Fuel Cell Testing Center, (FCTC), Johnstown. We gratefully acknowledge their support and help.

6. Academic Impact

Six faculty members, three post-docs and six graduate students are being trained and supported through this project. Currently, six Masters theses are in progress and one Masters project is completed.

PUBLICATIONS

1. M. El-Sharkh, A. Rahman, M. Alam, P. Byrne, A. Sakla, T. Thomas, “Proton Exchange Membrane Fuel Cell Dynamic Model for Residential Use," submitted to IEEE Transactions on Energy Conversion.
2. M. Y. El-Sharkh. A. Rahman, M. S. Alam, A. A. Sakla, P. C. Byrne and T. Thomas, “Strategies for Active and Reactive Power Control for PEM Fuel Cell Power Plants,” submitted to Journal of Power Sources.
3. P. Byrne, T. Thomas, M. Alam, A. Rahman, M. El-Shark, A. Sakla, “Proton Exchange Membrane Fuel Cell Steady-State Model for Residential Use," submitted to the Journal of Power Sources.

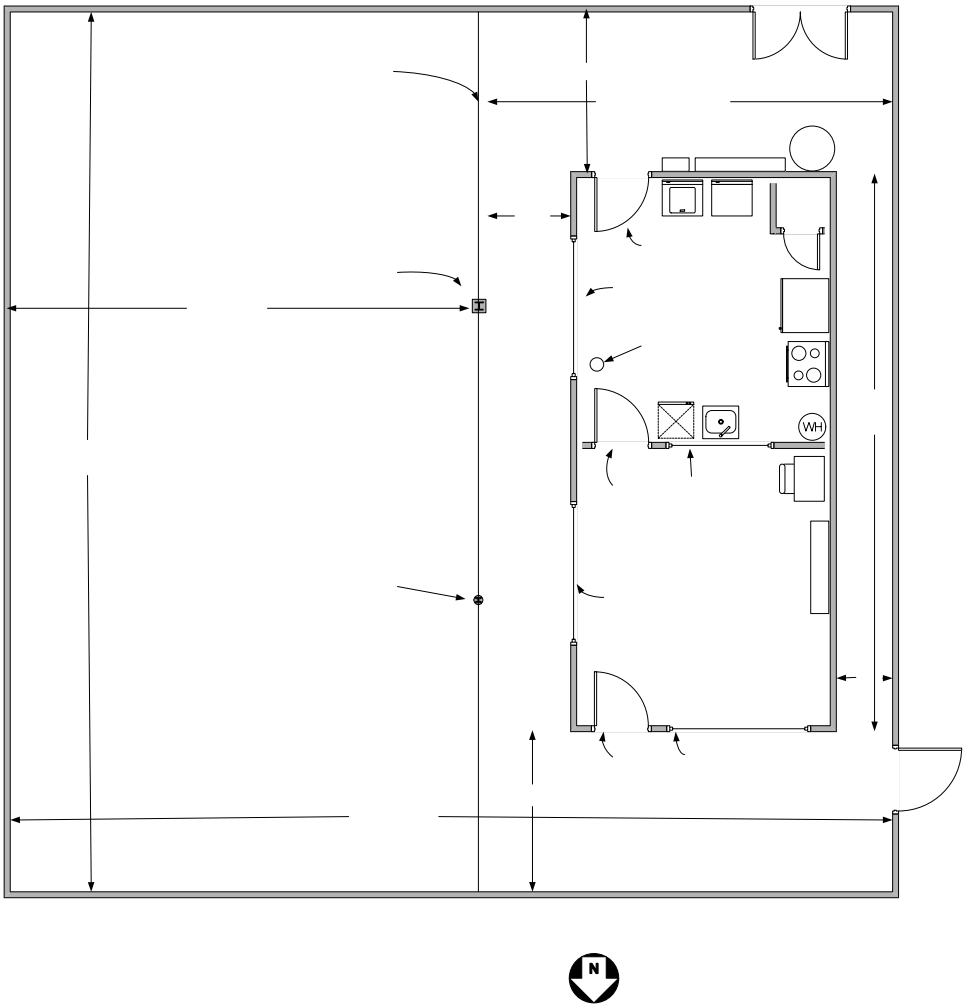


Fig 1. Floor Plan of the Laboratory 'Test Home'.

APPENDIX A

SEMaC

SEMaC Overview

Ali Mehrabi

1. SEMaC

SEMaC refers to the Host PC and the Energy Management Control System Smart Software residing inside the Host PC. The following components make up the entire SEMaC system:

- High-Speed Pentium IV PC
- High-Speed 12-Bit A/D board
- Compiled Smart Code residing inside the Host PC

SEMaC is designed as a modular system with the following modules built in it:

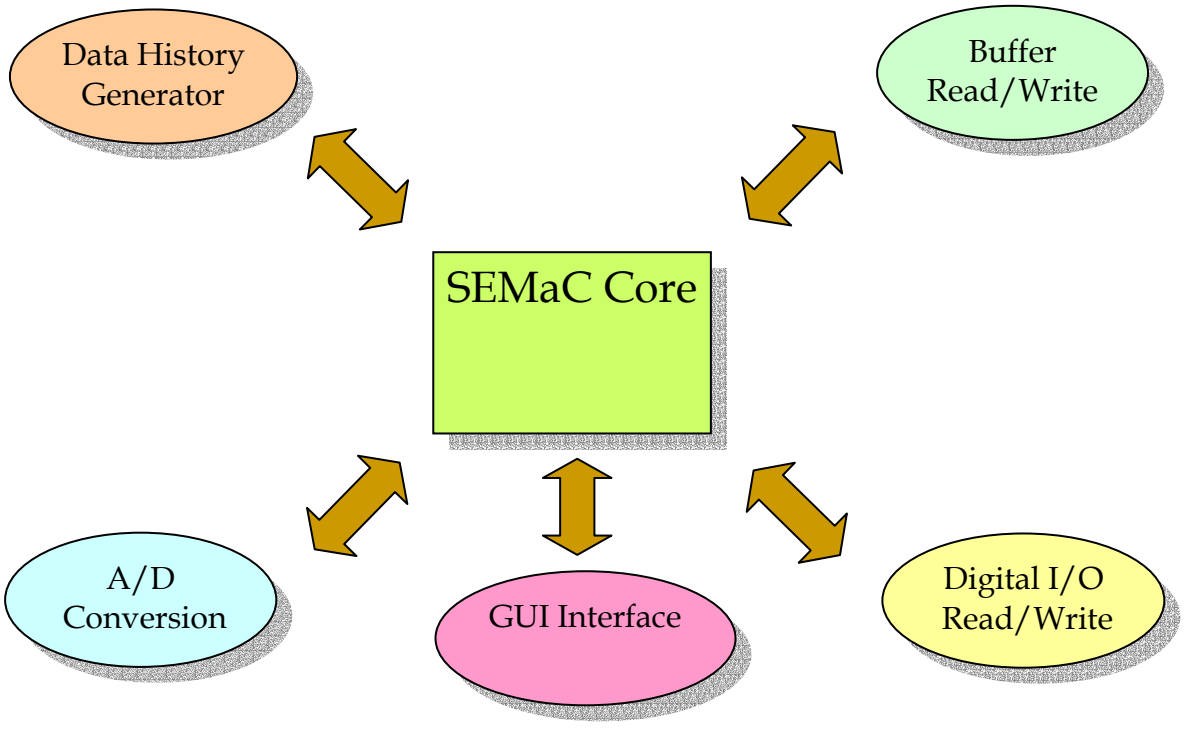


Fig. A2 SEMaC Modules

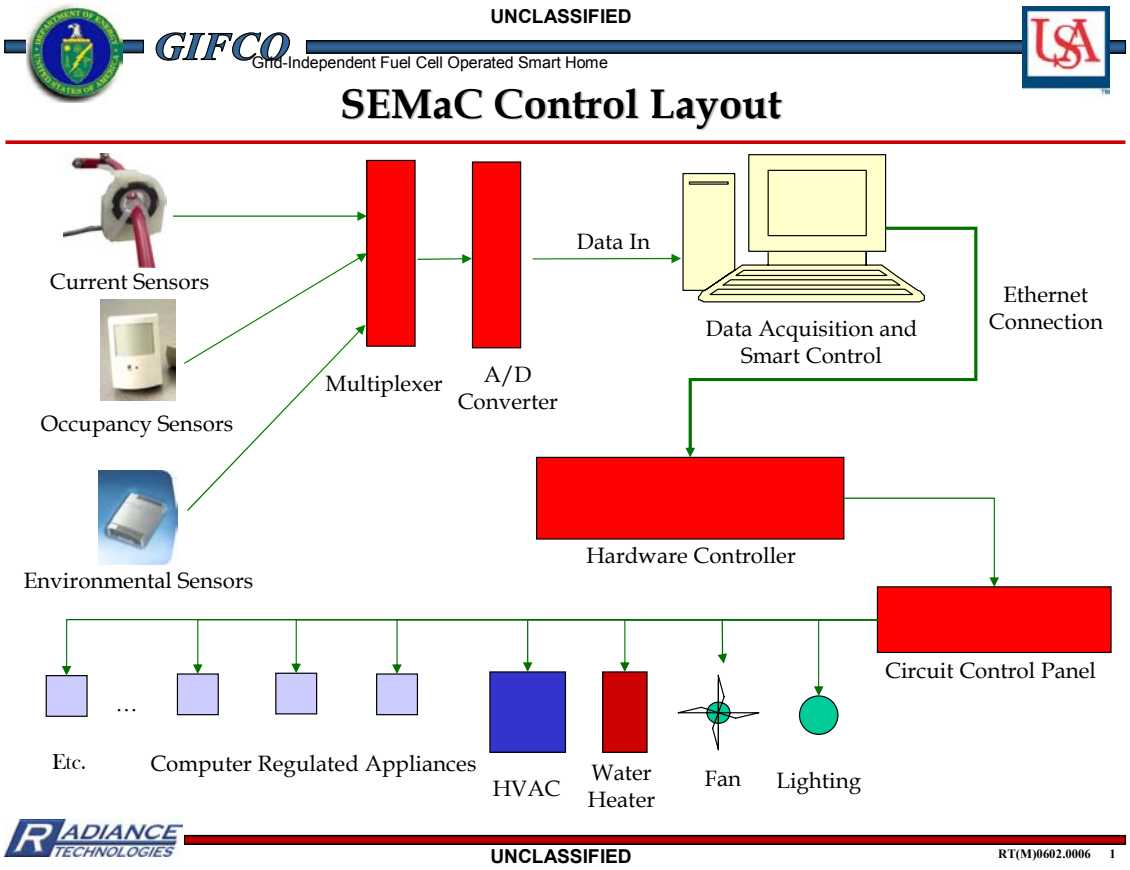


Fig. A1 SEMaC Control Layout

2. A Typical SEMaC Sequence

1. Read device priorities.
2. Read Power Source Specifications.
3. Read device override status.
4. Read the status of the auxiliary batteries.
5. Monitor the battery charge condition and charge batteries when needed.
6. Input instantaneous load profile (from all sensors).
7. Add up instantaneous power consumption and compare it with the total available power.
8. Is it less? Perform efficient energy management.
9. Is it more? Turn the lowest priority devices off, place them on a schedule, and go back to step 3.

3. System Hardware Controller (SHC)

System Hardware Controller (SHC) is referred to a smart hardware controller powered by a Motorola ColdFire MCF5272 microcontroller. The job of the SHC is to receive and execute smart decisions made by the SEMaC and report the status of the loads back to the SEMaC.

System Design Notes

For

Smart Energy Management Control (SEMaC) Software

ACRONYM LIST

DAQ	Data Acquisition
GIFCO	Grid Independent Fuel Cell Operated Smart Home
GUI	Graphical User Interface
SEMaC	Smart Energy Management Control
SHC	System Hardware Controller
USA	University of South Alabama

1 MISSION NEED

For the purposes of the GIFCO project, we will need a smart controller that is capable of making energy management decisions based upon various factors relating to current electrical demand and the historical pattern[i] of load usage by a particular occupant or set of occupants. The controller must be able to interpret the data acquired from the DAQ and then be able to issue commands to the SHC to actually manage the loads. It must also be able to show the user what management decisions have been made by sending a message to any GUI clients that are connected and also process commands that the GUI has issued to the controller.

2 OVERALL MISSION AREA

The mission area of the SEMaC covers all decisions relating to energy management in a home. The DAQ will provide energy usage data to the SEMaC and the SEMaC's responsibility will be to decide what, if anything, to do. The SEMaC will house any intelligent algorithms required for the energy management and all relevant historical data[i] that has been gathered by the DAQ for use in future management decisions. If management is required, the SEMaC will send commands to the SHC.

3 THE PROPOSED SYSTEM

The SEMaC will consist of the appropriate hardware and software needed to receive data from the DAQ and commands from the GUI, make smart energy management decisions, issue those decisions to the SHC, and display the results of those decisions to the user via the GUI.

4 SEMAC OVERVIEW.

The SEMaC software will fulfill the following missions. For the purposes of the first year demonstration, some features will not be available yet, or are being implemented by alternate methods. These will be footnoted in the text.

4.1 Scan DAQ

The SEMaC software will scan the DAQ for all load, environmental, and override/reset[ii] data.

4.2 Run Energy Management Algorithms

The SEMaC software will compare the data of the current scan with that of the last scan and decisions shall be made based on the current loads, as well as any environmental or override/reset status changes. The SEMaC will use various algorithms to determine if management is necessary, based on the target load and if energy saving algorithms are to be also applied to the home. These core algorithms may be based on historical load profile data, predictive analysis, fuzzy logic and/or neural networks, and user entered management routines.[iii]

4.3 Issue Commands to SHC

Once decisions have been made that management is necessary, a command will be sent to the SHC[iv] to turn off a load or loads, depending on how much power is needed.

4.4 Send Messages to GUI

Once any decisions have been made and commands have been issued, the SEMaC will send a message to any GUI clients that may be connected.

4.5 Take Commands from GUI

During normal operations, GUI clients may also issue commands to turn on or off appliances. The SEMaC will considered these commands as overrides, and treat them just as if they came from a physical override switch.

5 SOFTWARE

5.1 General Design

5.1.1 The SEMaC software uses a modular approach and contains the following modules: DAQ, Core, Control, and GUI.

5.1.2 The SEMaC software is a multi-threaded application, due to low-latency hardware interface requirements.

5.1.3 The SEMaC software is being developed on the Microsoft .NET platform. The .NET platform offers many features that will provide several benefits over other development environments. Web services, Multilanguage development, and XML support are three .NET features that will form the basis for several SEMaC features.

5.2 DAQ module

5.2.1 The DAQ module interfaces to the DAQ multiplexer hardware to receive information from the various sensors in the home.

5.2.2 The DAQ module will run in a separate thread to avoid any latency that could occur while processing data that is collected.

5.2.3 The communication interface to the DAQ multiplexer is via I/O ports available on the A/D card installed in the SEMaC PC.

5.2.4 The DAQ module will receive sensor data and reset/override[v] switch data via an A/D card. The SEMaC will scan the DAQ multiplexer for sensor data. This data is converted to digital and received via the SEMaC PC's A/D card.

5.2.5 The received DAQ data is then placed in an array indexed according to the number of DAQ channels, and provided to the main program thread and the Core Module, described in the next section.

5.3 Core module

5.3.1 The core module provides all decision making logic based on current sensor data, analysis of past data[i], and user input data. Algorithms developed as a result of preliminary data gathering and analysis shall be implemented in the core module.

5.4 Control module

5.4.1 The Control module interfaces to the System Hardware Controller (SHC) via a TCP connection over Ethernet[vi].

5.4.2 Based on input from the Core Module, the SHC module shall issue commands and receive acknowledgements from the SHC, using a simple predefined control language.

5.4.3 Interface - The SHC communicates with the SEMaC software via the following defined message set.

5.4.3.1 Load Management Message - contains the following fields: Message ID, Task ID, and Load Command (0-1024, indicating desired load level. 0=OFF, 512=%50, 1024=%100, etc.)

5.4.3.2 Task Complete Message - contains the following fields: Message ID and Task ID. The receipt of this message confirms to SEMaC that the requested Management Task has been accomplished.

5.5 GUI Module

5.5.1 The GUI module connects to the GUI application via a TCP network connection.

5.5.2 The GUI module implements a TCP Server, listening on TCP port 8989. The GUI module accepts multiple GUI client connections, allowing multiple clients to run at once.

5.5.3 Upon completing any control decisions and issuing SHC commands, a network packet is constructed, showing the status of all loads, environment sensors, and override/reset switch status.

5.5.4 The GUI module also receives any requests from the GUI for control of loads. These commands are treated like overrides in the Core module.

5.5.5 GUI Network Packet Definitions. The following items define the network packets between the SEMaC and the GUI client.

5.5.5.1 Load Profile Packet. The Load Profile Packet is sent to the GUI on each DAQ scan, after any management decisions have been made. The packet contains load/environmental and override switch data for all DAQ channels:

(SEMaC to GUI)

```
<STX>
   Field:Packet Type
   Type:Integer
   Use: Defines the type of message as 01=Load Profile or
   02=Management Message
<FS>
   Field:Device ID
   Type:Integer
   Use:Defines the ID of the device being reported
```

```
<FS>
   Field:Status
   Type:Integer
   Use: The Status of the load.  0=Off, 1=On
   Note:if the device has no status, as in an environment sensor,
   this will be 0
<FS>
   Field:Load
   Type:Single
   Use: Specifies the load (in amps) of the appliance
   Note: if the device does not represent a load, this will be 0
<FS>
   Field:Override
   Type:Integer
   Use: Reports the override state of the load. 1=device is in an
   override state 0=device not in override state
<FS>
   Field:Temperature
   Type:Single
   Use: If the Device ID represents a Temperature sensor, this is
   the temperature in Degrees Fahrenheit.
<FS>
   Field: Relative Humidity
   Type:Integer
   Use: If the device ID represents a humidity sensor, this is the
   %RH of the space being monitored
<FS>
   Field:Occupancy
   Type:Integer
   Use:If the Device ID represents an Occupancy sensor, the
   occupancy state of the room being monitored by the Device ID
   0=unoccupied 1=occupied

<RS>
   Same packet as above, starting at Device ID
<RS>
   …
<RS>
   …

<ETX>
```

5.5.5.2 Request Packet. A request packet is sent from the GUI to SEMaC in response to a user requesting that a load be turned on or off. Note: This will only turn off the supply to the load, not the device or appliance attached to the supply:

(GUI to SEMAC)

```
<STX>
   Field:Packet Type
   Type:Integer
   Use:The type of message.  1=Request Packet
<FS>
   Field:Device ID
   Type:Integer
```

```
   Use:The device ID of the control request.
<FS>
   Field:Command
   Type:Integer
   Use:The requested action  0=Turn Off Load  1=Turn On Load
<ETX>
```

5.5.5.3 Management Message. A Management Message is sent to the GUI each time a SEMaC management decision is made.

(SEMAC to GUI)

```
<STX>
   Field:Packet Type
   Type:Integer
   USE:  The type of message.  0=Management Message
<FS>
   Field:Message
   Type:String (review the data type for the .NET System.String
   class to determine the data type in C++)
   Use:  A 0-512 byte message for the GUI to display.
<ETX>
```

5.6 Performance

5.6.1 System performance will be determined by the number of DAQ channels that can be processed per second. The system performance target is to scan and make decisions on ~32 channels per second[vii].

6 MANAGEMENT ALGORITHMS

This section describes the energy management algorithms that will be used for the demo.

6.1 Occupancy/Priority

6.1.1 When the SEMaC Control module determines that the level of power demand is above the pre-defined management threshold[viii], the active loads in all un-occupied and occupied spaces are collected and sorted into a list based on un-occupied and occupied states, respectively.

6.1.2 The list of loads is then evaluated and sorted according to the load's management priority. This priority is defined when the load is entered into the system[ix]. A lower priority means the load is more susceptible to management than a higher priority load.

6.1.3 The list of loads is then evaluated to determine how many loads must be managed to bring the load of the system back under the management threshold.

6.1.4 The command to turn off the list of loads is then sent to the SHC.

6.2 Override

6.2.1 During the course of normal operations, it is possible that the number of loads in un-occupied spaces is insufficient to bring the power usage below the management threshold. The result is that low priority loads in occupied spaces,

and higher priority loads in un-occupied spaces are subject to being managed by the SEMaC.

6.2.2 Override is a mechanism by which the priority of a load may be temporarily increased by a home occupant by means of hardware switches near the load's power source (outlet or near the appliance), and via the GUI interface.

6.2.3 During normal DAQ scans, if a load currently being managed by SEMaC is overridden, the load's priority is temporarily increased. During the Occupancy/Priority algorithm, managed loads with higher priorities than un-managed loads are evaluated and resorted. This results in other loads with lower priorities than the overridden now being subject to management by SEMaC.

6.2.4 The period by which a load may be overridden is configurable, but typically is considered overridden until the load becomes un-used (current consumption reduces to a level defined as "off" for that load).

---

[i] Historical Data will not be used to determine management needs in the demo.

[ii] In the demonstration, device override and reset data is acquired from the Centralite Elegance controller.

[iii] For the demonstration, much simpler algorithms are to be used based on a load/priority/occupancy basis.

[iv] For the demonstration, the commands are sent as commands to the Centralite Elegance controller.

[v] In the demonstration, device override and reset data is acquired from the Centralite Elegance controller in a polled fashion, after each DAQ scan.

[vi] For the demo, the SHC is the Centralite Elegance Controller. The connection to this device is via a 19kbps serial connection.

[vii] Factors limiting performance include the bottleneck of a serial connection to the Centralite Elegance board at 19.2 kbps. The SHC design calls for an Ethernet connection at 100 Mbps. Also, the DAQ spec includes switch/override status as part of the DAQ scan, eliminating this polling interval from the DAQ scan application thread.

[viii] For the demo, the management threshold will be pre-defined at 5kW.

[ix] For the demo, the list of loads is built into the SEMaC, based on the model home design. A typical system would store these in a database table for ease of configuration.

APPENDIX B

SYSTEM HARDWARE CONTROLLER

Switch Concentrator (STARS) boards along with their wall switches, several Relay Driver (RLYDRVR) boards along with their solid-state relays, and a Fan Speed Controller board.

The control system can be connected to a host PC through an Ethernet, a USB, or an RS232 interface. The host PC can send commands to the System Controller to program different parameters, inquire about status of loads and switches, or control power to loads. In addition, the System Controller can be connected to a phone line through a modem for remote access.

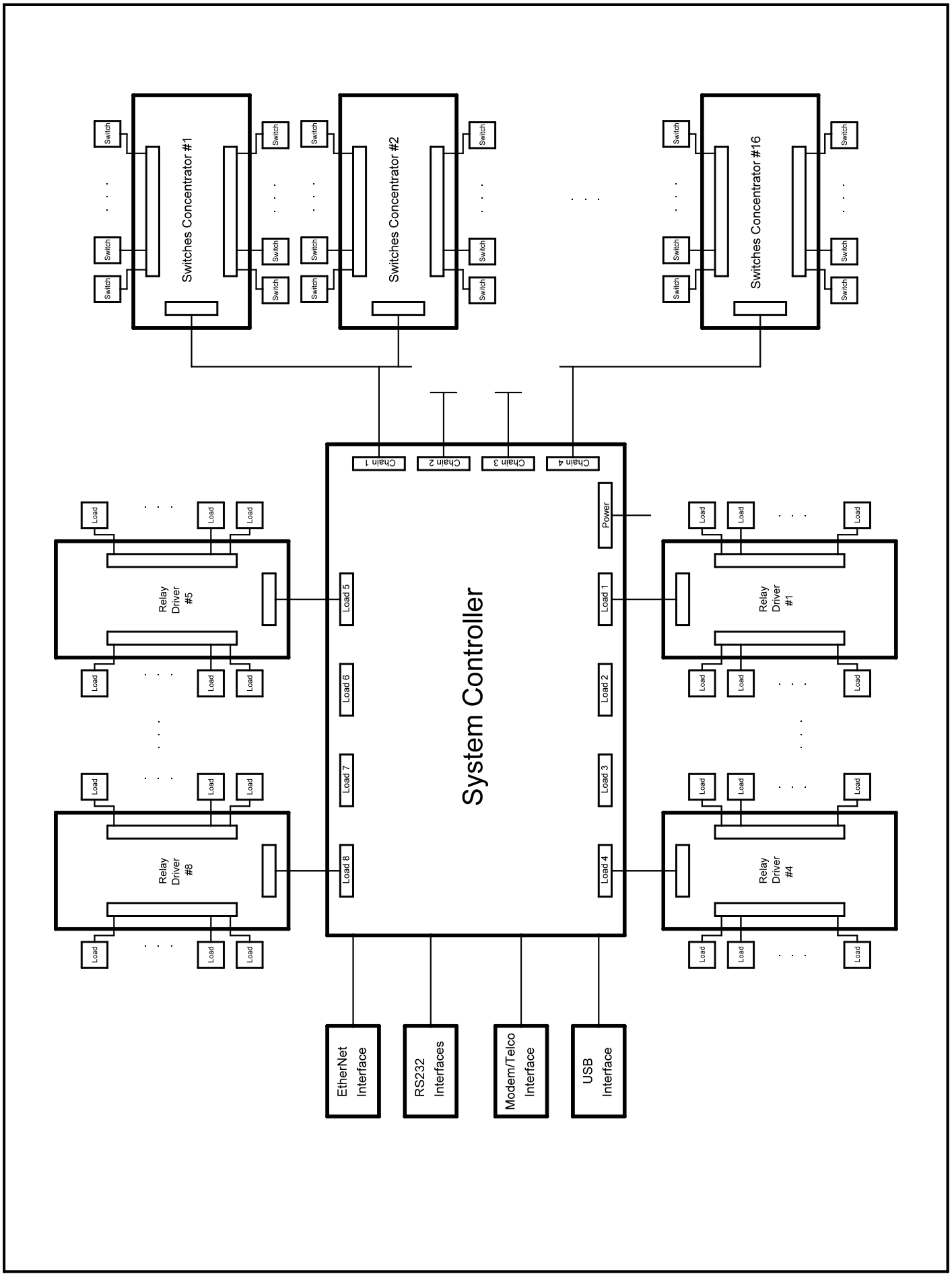
The System Controller samples switch states and control power to loads as programmed. Load power control can be either a simple on/off, or phase-control to allow any power level between 0 and 100%.

Switch states are sensed through STARS, and information is sent serially to the System Controller. In addition, each switch has an associated LED that provides a feedback to the user regarding the state of the corresponding load. LED information is also sent serially by the System Controller to the STARS.

The System Controller sends commands serially to the RLYDRVR to control power delivered to loads. The RLYDRVR controls power delivered to loads through solid-state relays. The RLYDRVR utilizes phase-control (through an FPGA) to control amount of power delivered to dimmable loads.

The System Controller sends commands serially to the Fan Speed Controller to control speed of ceiling fans connected to it. The Fan Speed Controller controls the speed of fans attached to it through voltage amplitude control.

All the above mentioned boards are microprocessor-based systems. The System Controller uses the Motorola ColdFire MCF5272 micro-controller along with two Microchip PIC (16C622A) micro-controllers. The STARS uses a Microchip PIC (16C622A) micro-controller. The RLYDRVR and the Fan Speed Controller use the Motorola MC68HC11 micro-controller.


Switches Concentrator #1
Switches Concentrator #2
Switches Concentrator #16
Switch
Chain 1
Chain 2
Chain 3
Chain 4
System Controller
Power
Load 1
Load 2
Load 3
Load 4
Load 5
Load 6
Load 7
Load 8
Relay Driver #1
Relay Driver #4
Relay Driver #5
Relay Driver #8
Load
EtherNet Interface
RS232 Interfaces
Modem/Telco Interface
USB Interface

FC CONTROLLER

Introduction:

The FC CONTROLLER (or SYSTEM CONTROLLER) board is the main controller (see associated schematics) that interfaces to up to sixteen STARS boards and up to eight RLYDRVR/LVRB boards. Thus, it is capable of interfacing to as much as 384 switches and controlling as much as 192 load relays. This board is based on the ColdFire MCF5272 micro-controller (providing an EtherNet, USB, an RS232, and an RS232 debug ports) along with Flash, SDRAM and battery-backed SRAM with real-time clock (RTC), one DUARTs (one RS232 interface and a modem), and some I/O ports. It also contains two PIC micro-controllers (PIC16C622A). One PIC micro-controller (Chain PIC) communicates with 16 STARS boards serially using 16 bi-directional lines (one line for each STARS board). The other PIC micro-controller (Load PIC) communicates with 8 RELAY DRIVER (RLYDRVR) and/or LOW VOLTAGE RELAY (LVRB) boards serially using 8 unidirectional lines (one line for each RLYDRVR/LVRB board). In addition, this board provides 8 reset lines to and receives 8 error lines from the RLYDRVR/LVRB boards. The Chain PIC communicates with the ColdFire MCF5272 (CPU) using an 8-bit bi-directional one-level deep FIFO with handshaking. The Load PIC communicates with the CPU using an 8-bit unidirectional one-level deep FIFO with handshaking. This board receives AC power, which is rectified and regulated to provide power. In addition, separate AC power is rectified and fed unregulated to the STARS boards.

Hardware Description:

The FC CONTROLLER board is based on the ColdFire MCF5272 micro-controller operating at 66 MHz. One serial port (URT0) of the CPU is used as an RS232 debug port. Either the EtherNet port, the USB port, or the other serial port (URT1) of the CPU can be used to communicate with the host. The hardware provides for a 2MB of Flash for code and configuration storage, a 4MB of SDRAM, and an 8KB of battery-backed SRAM with real time clock (Time Keeper). There is one DUART (16552) that provides one RS232 serial port and a modem (SF224ATFH1 or CH1786) connection to a phone line. Eight DIP switches provide configuration settings.

In addition to the main CPU, there are two PIC micro-controllers (PIC16C622A) along with their circuitry. One PIC micro-controller (Chain PIC) interfaces to the STARS boards through four Chain connectors, where each Chain connector can connect to four STARS boards. This Chain PIC communicates with the CPU through two 8-bit registers (one for each direction) along with two simple handshake lines. The other PIC micro-controller (Load PIC) interfaces to the RLYDRVR/LVRB boards through eight Load connectors, where each Load connector connects to one RLYDRVR/LVRB board. This Load PIC communicates with the CPU through one 8-bit register (one direction from CPU to PIC) along with one simple handshake line. The CPU provides 8 latched reset lines, one for each Load connector, and receives 8 error lines, one from each Load connector.

The following describes the memory map in terms of used chip selects, and offsets relative to those used chip selects. Software should initialize the different chip select registers properly to conform to this mapping.

Address Range	Device
$0000 0000 - $003F FFFF	SDRAM (4MB, 32 bits wide, using -CS7)
$0040 0000 - $0040 0000	CPU_to_Chain_PIC_Data_Register Write (1B, 8 bits wide, using -CS1)
$0040 0100 - $0040 0100	CPU_to_Load_PIC_data_Register Write (1B, 8 bits wide, using -CS1)
$0040 0200 - $0040 0200	Load_Reset_Register Write (1B, 8 bits wide, using -CS1)
$0040 0300 - $0040 0300	Chain_PIC_to_CPU_Data_Register Read (1B, 8 bits wide, using -CS1)
$0040 0400 - $0040 0400	Load_Error_Register Read (1B, 8 bits wide, using -CS1)
$0040 0500 - $0040 0500	IO_Register1 Read (1B, 8 bits wide, using -CS1)
$0040 0600 - $0040 0600	IO_Register2 Read (1B, 8 bits wide, using -CS1)
$0040 0700 - $0040 073C	External DUART Registers (16B, 8 bits wide, each byte is placed on a 32-bit long word boundary, using -CS1)
$0080 0000 - $0080 1FFF	External SRAM (8KB, 8 bits wide, using -CS2)
$0100 0000 - $0100 0000	CPU Error LED Write (1B, 8 bits wide, write $01 to turn LED ON, and $00 to turn LED OFF, using -CS4)
$FFE00 0000 - $FFFF FFFF	Flash (2MB, using -CS0)

The IO_Register1 and IO_Register2 are as follows.
(1) IO_Register1 is used as input connected to 8 DIP switches for configuration settings.

(2) IO_Regiser2 is used as follows.
Bit #0: signal CH2CP_FF.
Bit #1: signal CP2CH_FF.
Bit #2: signal CP2LD_FF.
Bit3 #7-3: not used.

Either the Ethernet interface or the USB interface of the CPU may be used to connect to a host. In addition, URT0 serial interface of the CPU can be used as a debug port, while URT1 serial interface of the CPU can be used to connect to a host or to a third-party device.

Channel A of the external DUART provides RS232-1 port, which may be used to connect to a host or to a third-party device. Channel B of the external DUART provides a serial port for the modem. Channel A interrupt output signal (INTRA) is connected to –INT1 of the CPU through an inverter, and channel B interrupt output signal (INTRB) is connected to -INT3 of the CPU through an inverter.

The Chain PIC micro-controller has 5 RA I/O lines (RA[4:0]) and 8 RB I/O lines (RB[7:0]). These lines are assigned different signal according to the following table.

Line	Signal Name	Description
RA[1:0]	Address Bits [1:0]	Address bits to select 1 out of 4 registers (output).
RA[2]	CHPICWD*	Chain PIC write data signal (output).
RA[3]	CHPICRD*	Chain PIC read data signal (output).
RA[4]	Not used	
RB[7:0]	CHPICRB[7:0]	Chain PIC data bus (input/output).

The Chain PIC circuitry contains the following registers/buffers.

Register	Description
Chain_12_Data_Out_Register	Contains 8 data bits to drive Chains 1 and 2 data lines, write only.
Chain_34_Data_Out_Register	Contains 8 data bits to drive Chains 3 and 4 data lines, write only.
Chain_PIC_to_CPU_Data_Register	Contains 8 data bits from the Chain PIC to the ColdFire MCF5272 CPU, write only.
Chain_12_Data_In_Buffer	Contains 8 data bits from Chains 1 and 2 data lines, read only.
Chain_34_Data_In_Buffer	Contains 8 data bits from Chains 3 and 4 data lines, read only.
CPU_to_Chain_PIC_Data_Register	Contains 8 data bits from the ColdFire MCF5272 CPU to the Chain PIC, read only.
Chain_PIC_Status_Register bit1	Contains 2 data bits, bits 0 provides signal CH2CP_FF and provides signal CP2CH_FF, read only.

The Load PIC micro-controller has 5 RA I/O lines (RA[4:0]) and 8 RB I/O lines (RB[7:0]). These lines are assigned different signal according to the following table.

Line	Signal Name	Description
RA[0]	CP2LD_FF	CPU_to_Load_PIC_data_Register Full Flag (input).
RA[1]	RCP2LDD*	CPU to Load PIC read data signal (output).
RA[2]	WLDOD*	Load PIC to load relay write data signal (output).
RA[4:3]	Not used	
RB[7:0]	LDPICRB[7:0]	Load PIC data bus (input/output).

The Load PIC circuitry contains the following registers/buffers.

Register	Description
Load_PIC_Data_Out_Register	Contains 8 data bits to drive load relay data lines, write only.
CPU_to_Load_PIC_data_Register the	Contains 8 data bits from the ColdFire MCF5272 CPU to Load PIC, read only.

Initialization:

- For the ColdFire MCF5272 CPU:
(1) All internal CPU registers should be initialized properly.
(2) The CPU interfaces should be initialized.
(3) The external DUART should be enabled, and initialized to the proper parameters.
(4) IO_Register1 should be read to determine configuration DIP switch settings.
(5) The Chain_PIC_to_CPU_Data_Register should be read to clear the CH2CP_FF flag.
(6) CPU_to_Chain_PIC_Reset_Register may be cleared by writing "0" to it (it is also cleared during reset).
(7) CPU Error LED flip-flop Register may be cleared by writing "0" to it (it is also cleared during reset).

- For the Chain PIC:
(1) RA[3:2] lines should be initialized to "1" level. RA[3:0] are outputs.
(2) The CPU_to_Chain_PIC_Data_Register should be read to clear the CP2CH_FF flag.
(3) The Chain_12_Data_Out_Register and Chain_34_Data_Out_Register may be cleared by writing "0"" to them (they are also cleared during reset).

- For the Load PIC:
(1) All RA[2:1] lines should be initialized to "1" level. RA[0] is input.
(2) The CPU_to_Load_PIC_data_Register should be read to clear the CP2LD_FF flag.
(3) The Load_PIC_Data_Out_Register may be cleared writing "0" to it (it is also cleared during reset).

Operation:

The CPU activities include the following.
(1) Getting switch states from the Chain PIC, and providing load states to the Chain PIC.
(2) Processing switch states information, and providing necessary action(s) to the Load PIC by sending proper commands.
(3) Monitoring all interfaces from the CPU and taking proper actions.
(4) Monitoring all serial interfaces from the external DUART and taking proper actions.
(5) Monitoring any error condition(s) from the RLYDRVR/LVRB boards and taking proper action(s).

Jumper Selections:

TBD.

Communications between the FC CONTROLLER and the STARS:

The Chain PIC micro-controller communicates with 16 STARS boards, simultaneously, through 16 bi-directional lines, one line for each STARS board. Each four lines are connected to a Chain connector. Accordingly, there are four chains (Chain1, Chain2, Chain3 and Chain4) along with their connectors. The baud rate is 10,000 bits/s. The data format is one start bit ("1"), and 24 data bits (least significant bit first). The start bit is always sent by the FC CONTROLLER board as a means of synchronization for the STARS boards. The protocol is as follows.

(1) The Chain PIC sends "0" (to all 16 STARS) by writing $00 to the Chain_12_Data_Out_Register and Chain_34_Data_Out_Register. It waits for at least 5000 us. This is used as a means of synchronization prior to every data exchange.
(2) The Chain PIC sends "1" (to all 16 STARS) by writing $FF to the Chain_12_Data_Out_Register and Chain_34_Data_Out_Register for one bit period (100 us).
(3) The Chain PIC sends all 24 bits of load states to all STARS simultaneously, one bit at a time (100 us/bit) and starting with least significant bit, by writing proper data to both registers.
(4) The Chain PIC sends another start bit, similar to what was done in step (2). It, then, removes it by sending a "0".
(5) The Chain PIC should sample all 16 STARS at ½ bit time (50 us) past the high to low transition of the start bit sent in previous step by reading the Chain_12_Data_In_Buffer and the Chain_34_Data_In_Buffer. The Chain PIC should sample the rest of the 24 data bits for each STARS board at the rate of 100 us/bit. Necessary bit unpacking, shifting and packing operations should be performed to yield 48 bytes of information (3 bytes for each STARS board). This information represents the switch states of each STARS board, and should be saved in memory for later transfer to the ColdFire MCF5272 CPU.
(6) The Chain PIC repeats steps (1) – (5) continuously.

The Chain PIC can write to the Chain_12_Data_Out_Register as follows.
- Set RA[1:0] to "00", and RB[7:0] to desired data byte.
- Set RA[2] to "0", then set it to "1".

The Chain PIC can write to the Chain_34_Data_Out_Register as follows.
- Set RA[1:0] to "01", and RB[7:0] to desired data byte.
- Set RA[2] to "0", then set it to "1".

The Chain PIC can read from the Chain_12_Data_In_Buffer as follows.
- Set RA[1:0] to "00".
- Set RA[3] to "0".
- Read desired data from RB[7:0].
- Set RA[3] to "1'.

The Chain PIC can read from the Chain_34_Data_In_Buffer as follows.
- Set RA[1:0] to "01".
- Set RA[3] to "0".
- Read desired data from RB[7:0].
- Set RA[3] to "1'.

Communications between the ColdFire MCF5272 CPU and the Chain PIC:

The ColdFire MCF5272 CPU communicates with the Chain PIC through the CPU_to_Chain_PIC_Data_Register and the Chain_PIC_to_CPU_Data_Register along with their full flag status bits CP2CH_FF and CH2CP_FF. The CPU always initiates the data transfer, and whenever it needs to get switch states/send load states. The process has three phases, as follows.

(1) In the first phase, the CPU sends a command ($01 for bit transfer mode, or $02 for byte transfer mode for switch states data).
(2) In the second phase, the CPU receives either 384 bytes (in bit transfer mode, only least significant bit is pertinent), or 48 bytes (byte transfer mode) from the STARS board containing the switch states.
(3) In the third phase, the CPU sends out 48 bytes to the STARS containing the relay states.

Whenever the CPU needs to send a byte to the Chain PIC, the following protocol should be followed.

(1) The CPU waits for the CP2CH_FF signal to be "0", then it writes that byte to the CPU_to_Chain_PIC_Data_Register. This write action sets the CP2CH_FF to "1".
(2) When the Chain PIC detects that the CP2CH_FF is "1" by reading the Chain_PIC_Status_Register (bit 1), it reads the CPU_to_Chain_PIC_Data_Register to get the sent byte. This read action clears the CP2CH_FF flag.

Whenever, the Chain PIC needs to send a byte to the CPU, the following protocol should be followed.

(1) The Chain PIC waits for the CH2CP_FF to be "0" (bit 0 of the Chain_PIC_Status_Register), then it writes that byte to the Chain_PIC_to_CPU_Data_Register. This write action sets the CH2CP_FF to "1".
(2) When the CPU detects that the CH2CP_FF signal is "1", it reads the Chain_PIC_to_CPU_Data_Register to get the sent byte. This read action clears the CH2CP_FF flag.

Additional commands may be added later, such as a command to get a particular switch state/send corresponding load state, etc., if needed.

The Chain PIC can write to the Chain_PIC_to_CPU_Data_Register as follows.
- Set RA[1:0] to "10", and RB[7:0] to desired data byte.
- Set RA[2] to "0", then set it to "1".

The Chain PIC can read from the CPU_to_Chain_PIC_Data_Register as follows.
- Set RA[1:0] to "10".
- Set RA[3] to "0".
- Read desired data from RB[7:0].
- Set RA[3] to "1'.

The Chain PIC can read from the Chain_PIC_Status_Register as follows.
- Set RA[1:0] to "11".

- Set RA[3] to “0”.
- Read desired data from RB[7:0].
- Set RA[3] to “1’.

Communications between the ColdFire MCF5272 CPU and the Load PIC:

The ColdFire MCF5272 CPU communicates with the Load PIC through the CPU_to_Load_PIC_Data_Register along with its full flag status bits CP2LD_FF. The CPU always initiates the data transfer, and whenever it needs to send commands to the Load PIC. The CPU sends a command string, which consists of 5 bytes. The first byte is a mask byte, where a 1-bit indicates that the command is to be sent to the corresponding RLYDRVR/LVRB board. The second byte is a synchronization byte ($FC, $FD, or $FE) to be sent to the assigned RLYDRVR/LVRB board(s) as specified in the mask byte. The other three bytes are the command bytes to be sent following the synchronization byte. The Load PIC always receives commands from the CPU. The protocol for sending one byte is as follows.

(1) The CPU waits for the CP2LD_FF signal to be “0”, then it sends a byte to the Load PIC, by writing that byte to the CPU_to_Load_PIC_Data_Register. This write action sets the CP2LD_FF to “1”.
(2) When the Load PIC detects that the CP2LD_FF (RA[0]) is “1, it reads the CPU_to_Load_PIC_Data_Register . This read action clears the CP2LD_FF flag.
(3) Steps (1) and (2) are repeated every time the CPU needs to send a byte to the Load PIC.

The Load PIC can read from the CPU_to_Load_PIC_Data_Register as follows.
- Set RA[1] to “0”.
- Read desired data from RB[7:0].
- Set RA[1] to “1’.

Communications between the FC CONTROLLER and the RLYDRVR/LVRB:

The Load PIC micro-controller communicates with 8 RLYDRVR/LVRB boards, simultaneously, through 8 unidirectional lines, one line for each RLYDRVR/LVRB board. Each line is connected to a Load connector. Accordingly, there are eight load connectors (Load1, Load2, Load3, Load4, Load5, Load6, Load7 and Load8). The baud rate is 19.6 K bits/s. The data format is one start (“1”), 8 data bits (least significant bit first), no parity bit, and one stop bit (“0”). The Load PIC can send a command (4 bytes) to either one RLYDRVR/LVRB board or several boards simultaneously as indicated by the mask byte. When the Load PIC needs to send a byte a RLYDRVR/LVRB board, it follows the following steps.

(1) The Load PIC sends a start bit (“1”) to that RLYDRVR/LVRB board by writing the proper byte value to the Load_PIC_Data_Out_Register for one bit time (51 us).
(2) The Load PIC sends the bits of the desired byte (least significant bit first) one bit at a time to that RLYDRVR/LVRB (each bit lasts for 51 us) by writing the proper byte value to the Load_PIC_Data_Out_Register.
(3) The Load PIC sends a stop bit (“0”) to that RLYDRVR/LVRB board by writing the proper byte value to the Load_PIC_Data_Out_Register for one bit time (51 us).

The above steps can be repeated whenever the Load PIC needs to send a byte to a RLYDRVR/LVRB board.

Each command sent to the RLYDRVR board has 4 bytes. The first byte is a synchronization byte ($FC, $FD, or $FE). The other 3 bytes are as follows.
- The first byte has a format of "cccr rrrr".
- The second byte has format of "llll lxxx".
- The third byte has a format of "yyyy yyyy".

Where "ccc" (most significant 3 bits of the first byte) indicates the command code.
The "rrrrr" (least significant 5 bits of the first byte) represents a particular transition rate (maximum 32 different rates), if any.
The "lllll" (most significant 5 bits of the second byte) represents a load relay number between "00000" and 10111", inclusively.
The "xxx yyyy yyyy" (least significant 3 bits of the second byte concatenated with the third byte) represents a particular dim level. A binary value of "000 0000 0000" corresponds to always/fully ON, a binary value of "111 1111 1111" corresponds to always/fully OFF, a binary value in between corresponds to that dimming level.
The command codes are as follows.
- "000" = Not used.
- "001" = Single load transition.
- "010" = Not used.
- "011" = Not used.
- "100" = Not used.
- "101" = All loads transition.
- "110" = Not used.
- "111" = Not used.

The following table provides the values of the transition rate.

"rrrrr"	Value	"rrrrr"	Value	"rrrrr"	Value	"rrrrr"	Value
00	Immediate	08	9 seconds	16	41 seconds	24	210 seconds
01	1 second	09	11 seconds	17	49 seconds	25	250 seconds
02	2 seconds	10	13 seconds	18	60 seconds	26	300seconds
03	3 seconds	11	16 seconds	19	75 seconds	27	380 seconds
04	4 seconds	12	19 seconds	20	90 seconds	28	450 seconds
05	5 seconds	13	23 seconds	21	110 seconds	29	550 seconds
06	6 seconds	14	28 seconds	22	140 seconds	30	675 seconds
07	7 seconds	15	34 seconds	23	175 seconds	31	800 seconds

Each command sent to the LVRB board has 4 bytes. The first byte is a synchronization byte ($FC, $FD, or $FE). The other 3 bytes are as follows.
- The first byte has a format of "cccr rrrr".
- The second byte has format of "llll lxxx".
- The third byte has a format of "yyyy yyyy".

Where "ccc" (most significant 3 bits of the first byte) indicates the command code.

The “rrrrr” (least significant 5 bits of the first byte) represents a particular transition time period code (maximum 32 different rates), if any.
The “lllll” (most significant 5 bits of the second byte) represents a load relay number between “00000” and 10111”, inclusively.
The “xxx yyyy yyyy” (least significant 3 bits of the second byte concatenated with the third byte) represents the desired state of the relay (only binary values “000 0000 0000”, i.e., fully ON, and “111 1111 1111”, i.e., fully OFF, are allowed).
The command codes are as follows.
- “000” = Not used.
- “001” = Single transition of a single relay.
- “010” = Not used.
- “011” = Not used.
- “100” = Not used.
- “101” = Single transition of all relays.
- “110” = Not used.
- “111” = Not used.

The following table provides the values of the transition time period.

“rrrrr”	Value	“rrrrr”	Value	“rrrrr”	Value	“rrrrr”	Value
00	Indefinite	08	3.00 seconds	16	14.0 seconds	24	90.0 seconds
01	0.25 seconds	09	4.00 seconds	17	16.0 seconds	25	120 seconds
02	0.50 seconds	10	5.00 seconds	18	18.0 seconds	26	300 seconds
03	0.75 second	11	6.00 seconds	19	20.0 seconds	27	600 seconds
04	1.00 seconds	12	7.00 seconds	20	25.0 seconds	28	900 seconds
05	1.50 seconds	13	8.00 seconds	21	30.0 seconds	29	1200 seconds
06	2.00 seconds	14	10.0 seconds	22	45.0 seconds	30	1800 seconds
07	2.50 seconds	15	12.0 seconds	23	60.0 seconds	31	2700 seconds

The Load PIC can write to the Load_PIC_Data_Out_Register as follows.
- Set RB[7:0] to desired data byte.
- Set RA[2] to “0”, then set it to “1”.

DIP Switches Configuration Settings:

There are 8 DIP switches for configuration settings, as follows.
Switch 1 : ON: Load default switches/loads/scenes configurations in EEPROM.
OFF: Do not load default configurations.
Switch 2 : ON: Load refresh is enabled.
OFF: Load refresh is disabled.
Switch 3 : ON: Third-party communication RS232-1 baud rate is 9.6 K.
OFF: Third-party communication RS232-1 baud rate is 19.2 K.

Switch 4 : ON: Third-party communication RS232-2 baud rate is 9.6 K.

OFF: Third-party communication RS232-2 baud rate is 19.2 K.

Switch 5 : Not used
Switch 6 : Not used
Switch 7 : Not used
Switch 8 : Not used

Communications between a PC (Personal Computer) and the FC CONTROLLER:

The FC CONTROLLER may communicate with a PC through either Ethernet port of CPU, USB port of CPU, or an RS232 (channel A of DUART or URT1 of CPU) port. RS232 communications parameters are 19200 baud rate, 8 data bits, 1 stop bit and no parity. This communications link may be used by a PC to either program the FC CONTROLLER with different configurations, retrieve status information from the FC CONTROLLER, or command the FC CONTROLLER to perform a certain task. The FC CONTROLLER always acts as a slave to a PC, i.e., it always responds to commands from a PC.

Communications from a PC to the FC CONTROLLER is always using ASCII code, while communications from the FC CONTROLLER to a PC is always using binary code. All commands are started with ASCII "^" (ASCII code $5E), followed by a 2-byte command code (ASCII "00" - "FF"), and a 4-byte number. The 2-byte command code may have a value from the following table. The 4-byte number following the command is only meaningful for certain commands, as indicated in the following table. For other commands, this 4-byte number is not meaningful, but must be sent (e.g. zeros may be sent). The table below lists the supported commands.

Command Code and Format	**Meaning**	**Number meaningful or not**
^00xxxx	Send code version	No
^01xxxx	Send customer number	No
^02xxxx	Send real-time clock settings	No
^03xxxx	Send all switch configurations (parameters)	No
^04xxxx	Send all load configurations (parameters)	No
^05xxxx	Send all scene configurations (parameters)	No
^06xxxx	Send all switch status (states)	No
^07xxxx	Send all load status (states)	No
^08xxxx	Send all scene status (states)	No
^09xxxx	Send customer options	No
^0Axxxx	Send all instant switch values (ON/OFF)	No

^0Bxxxx	Send all instant load values (ON/OFF)	No
^0Cxxxx	Send miscellaneous parameters	No
^0Dxxxx	Send error status of all RLYDRVR/LVRB boards	No
^10xxxx	Send sunrise/sunset table	No
^23nnnn	Send single switch configurations (parameters)	Yes
^24nnnn	Send single load configurations (parameters)	Yes
^81xxxx	Receive customer number	No
^82xxxx	Receive real-time clock settings	No
^83xxxx	Receive all switch configurations (parameters)	No
^84xxxx	Receive all load configurations (parameters)	No
^85xxxx	Receive all scene configurations (parameters)	No
^89xxxx	Receive customer options	No
^90xxxx	Receive sunrise/sunset table	No
^A3nnnn	Receive single switch configurations (parameters)	Yes
^A4nnnn	Receive single load configurations (parameters)	Yes
^E0llrr	Set temporary load/relay level and rate/pulse width	Yes
^E1nnnn	Activate single load/relay using temporary level and rate	Yes
^E4nnnn	Activate single load/relay	Yes
^E5nnnn	Activate single scene	Yes
^F4nnnn	De-activate single load/relay	Yes
^F5nnnn	De-activate single scene	Yes

---

xxxx is a 4-digit ASCII number that is not meaningful, but must be sent (e.g. zeros may be sent, ASCII "0000").
nnnn is a 4-digit ASCII number specifying either a switch number (ASCII "0000" - "017F"), a load number (ASCII "0000" - "00BF"), or a scene number (ASCII "0000" - "00FF").
ll is a 2-digit ASCII number specifying load level value (from 00 to 99, where 00 is fully OFF, and 99 is fully ON).
rr is a 2-digit number specifying a code for rate (RLYDRVR) (from 00 to 31) at which a load is activated to the specified level, according to the following table.

"**rr**"	Value	"**rr**"	Value	"**rr**"	Value	"**rr**"	Value
00	Immediate	08	9 seconds	16	41 seconds	24	210 seconds

01	1 second	09	11 seconds	17	49 seconds	25	250 seconds
02	2 seconds	10	13 seconds	18	60 seconds	26	300seconds
03	3 seconds	11	16 seconds	19	75 seconds	27	380 seconds
04	4 seconds	12	19 seconds	20	90 seconds	28	450 seconds
05	5 seconds	13	23 seconds	21	110 seconds	29	550 seconds
06	6 seconds	14	28 seconds	22	140 seconds	30	675 seconds
07	7 seconds	15	34 seconds	23	175 seconds	31	800 seconds

rr is a 2-digit number specifying a code for rate (LVRB) (from 00 to 31) at which a relay is activated for a specified pulse width, according to the following table.

"rrrrr"	Value	"rrrrr"	Value	"rrrrr"	Value	"rrrrr"	Value
00	Indefinite	08	3.00 seconds	16	14.0 seconds	24	90.0 seconds
01	0.25 seconds	09	4.00 seconds	17	16.0 seconds	25	120 seconds
02	0.50 seconds	10	5.00 seconds	18	18.0 seconds	26	300 seconds
03	0.75 second	11	6.00 seconds	19	20.0 seconds	27	600 seconds
04	1.00 seconds	12	7.00 seconds	20	25.0 seconds	28	900 seconds
05	1.50 seconds	13	8.00 seconds	21	30.0 seconds	29	1200 seconds
06	2.00 seconds	14	10.0 seconds	22	45.0 seconds	30	1800 seconds
07	2.50 seconds	15	12.0 seconds	23	60.0 seconds	31	2700 seconds

After the FC CONTROLLER receives a command and a number, it should reply with either an ASCII ACK (ASCII code $06) if it supports that command, or an ASCII NAK (ACSII code $15) if it does not support that command.

If the command is a "send" command, the FC CONTROLLER should send a binary 2-byte block length (upper byte first), followed by the block data bytes (binary form). The PC should respond by sending an ACK indicating that it received the entire data block.

If the command is a "receive" command, the PC should send an ASCII 4-byte block length (upper byte first), after which the FC CONTROLLER will send an ACK. After receiving an ACK, the PC should send the block data bytes (ASCII form). The block of data should be sent in packets of 64 ASCII bytes long, if its length is more than 64. The last packet should contain the remaining bytes. After each packet, the FC CONTROLLER will send and ACK to the PC indicating that it has received and processed that packet. The PC should not attempt to send the next packet prior to receiving the ACK of the current packet. This regulation mechanism is needed due to buffer size limitations in the memory subsystem of the FC CONTROLLER board.

If the command is an "activate/de-activate", no further action is necessary. The ACK sent by the FC CONTROLLER indicates that the task is executed.

The following describes the details of the code version, customer number, real-time clock settings, switch configurations (parameters), load configurations (parameters), scene configurations (parameters), switch status (states), load status (states), scene status (states), and sunrise/sunset table.

Code Version:

Code version has 4 digits and occupies 2 bytes (abcd), and refers to version Vab.cd of the current code.

Customer Number:

Customer number has 8 digits and occupies 4 bytes (abcdefgh), and refers to a unique number assigned by a dealer to a customer.

Real-time Clock Settings:

Real-time settings consist of 7 bytes (BCD format) according to the following order.

Byte #	Contents	BCD Format
1	Seconds	00 - 59
2	Minutes	00 - 59
3	Hours (24 hour format)	00 - 23
4	Day of Week (1:Sunday, 2:Monday, .. , 7:Saturday)	01 - 07
5	Date (Day of Month)	01 - 31
6	Month	01 - 12
7	Year	00 - 99

Switch Configurations (Parameters):

There are 384 entries, each entry has 2 bytes, as follows.

1st byte:
- bit 7 — Load (0) or a scene (1) switch
- bit 6 — Normal (0) or simple (1) switch if (Non-Contact and Non-Ramp), Normally Open (0) or Normally Closed (1) (if Contact and Non-Ramp), or Up/Raise (0) or Down/Lower (1) (if Ramp)
- bit 5 — Non-Contact (0) or Contact (1)
- bit 4 — Non-Ramp (0) or Ramp (1)

Bit #6	Bit #5	Bit #4	Function
0	0	0	Normal, Non-Contact, Non-Ramp
1	0	0	Simple, Non-Contact, Non-Ramp
0	1	0	Normally-Open, Contact, Non-Ramp
1	1	0	Normally-Closed, Contact, Non-Ramp
0	0	1	Up/Raise, Ramp
1	0	1	Down/Lower, Ramp
0	1	1	Not used
1	1	1	Not used

	bit 3	Don't send switch action (0), or send switch action (1) to third-party RS232 channels
	bit 2	Switch controls load/scene on same (0) or different (1) HOME CONTROLLER board
	bits 1:0	FC CONTROLLER board number to control (0 to 3) if bit 2 is 1
2nd byte:	bits 7:0	Load number (0 to 191) or Scene number (0 to 255)

The order of the switch entries is as follows:
SW1 for STARS 1A,1B,1C,1D, 2A,2B,2C,2D, 3A,3B,3C,3D, 4A,4B,4C,4D (16 entries, 32 bytes)
SW2 for STARS 1A,1B,1C,1D, 2A,2B,2C,2D, 3A,3B,3C,3D, 4A,4B,4C,4D (16 entries, 32 bytes)
...
SW24 for STARS 1A,1B,1C,1D, 2A,2B,2C,2D, 3A,3B,3C,3D, 4A,4B,4C,4D (16 entries, 32 bytes)

Load Configurations (Parameters):

There are 192 entries, each entry has 5 bytes, as follows.

1st byte:	bit 7	Dim (1) or not (0) for RLYDRVR, Push-Hold (1) or Tap (0) for LVRB
	bit 6	All ON (1) or not (0) for RLYDRVR and LVRB
	bit 5	All OFF (1) or not (0) for RLYDRVR and LVRB
	bit 4	Vacation mode (1) or not (0) for RLYDRVR and LVRB
	bit 3	Soft ON (1) or not (0) for RLYDRVR only, ignored for LVRB (should be 0)
	bit 2	Soft OFF (1) or not (0) for RLYDRVR only, ignored for LVRB (should be 0)
	bit 1	Alarm Flash (1) or not (0) for RLYDRVR only, ignored for LVRB (should be 0)
	bit 0	LVRB (1) or RLYDRVR (0)
2nd byte:	bit 7	Not used for RLYDRVR, interlocked (1) or separate (0) for LVRB Note that relays 8 & 9, 10 & 11, 12 & 13, and 14 & 15 may be interlocked.
	bit 6	Not used for RLYDRVR, Flip-flop if interlocked (1) or not (0) for LVRB. Note that Flip-flop has no effect if Push-hold type (only Tap type).
	bits 5	Fan Speed Control (1) or not (0) for RLYDRVR only, ignored for LVRB (should be 0)
	bits 4:0	Dimming (ramp) rate for RLYDRVR (0 to 31, should be 0 for non-dimmer), pulse width ticks code for LVRB (must be 0 for Push-hold type, indefinite)
3rd byte:	bits 7:6	Phase Number for 3-phase system for RLYDRVR and LVRB ("00" for phase 1, "01" for phase 2, and "10" for phase 3).
	bits 5	Not used

	bits 4:0	Soft (ON/OFF) rate (0 to 31, should be 0 for non-dimmer) for RLYDRVR only, ignored for LVRB (should be zeros)
4th & 5th bytes:	bits 15:11	Relay number in its RLYDRVR/LVRB Board (0 to 23) for RLYDRVR, (0 to 15) for LVRB as follows: 0 to 7 are for separate relays, 8 to 15 are for separate/interlocked relays
	bits 10:0	Soft ON level ("000 0000 0000" always ON, ... , "111 1111 1111" always OFF, should be "000 0000 0000" for non-dimmer) for RLYDRVR only, ignored (should be zeros) for LVRB. For a fan speed control, use the following: "000 xxxx xxxx" for 100% speed, "010 xxxx xxxx" for 75% speed, "100 xxxx xxxx" for 50% speed, "110 xxxx xxxx" for 25% speed, and "111 xxxx xxxx" for 0% speed.)

The order of the load entries is as follows:

LD1, LD2, LD3, ... , LD24 for RLYDRVR/LVRB 1 (24 entries, 96 bytes)
LD1, LD2, LD3, ... , LD24 for RLYDRVR/LVRB 2 (24 entries, 96 bytes)
...
LD1, LD2, LD3, ... , LD24 for RLYDRVR/LVRB 8 (24 entries, 96 bytes)

Scene Configurations (Parameters):

Scene configurations contain 3 blocks. Note that some scenes may be either global only, local only or global/local.
The first block has 2 bytes, and it contains a pointer for the first available location to add a new scene.
The second block has 256 entries. Each entry has 2 bytes, and contains a pointer to scene that scene's parameters.
The third block has up to 8,192 bytes, and contains scene parameters. Each scene will contain a variable number of bytes (minimum 1 byte) as follows.

1st byte:	bits 7:0	Action code as follows:	
		$00	No action (no scene) (always global)
		$01	Transition load(s) (global or local)
		$02	All ON/OFF (always global)
		$03	Alarm flash (always global)
		$04	Vacation mode (always global)
		$05	Protected transition load(s) (global or local)
		$06	Power-up (global or local)
		$07	Timed (always local)
		$08	ASCII string (always local)
		$09-$FF	Not used

If action code is $00: no more bytes following (always global).

If action code is $01: 4 more bytes as follows:

1st byte:	bit 7	Last load in scene (1) or not last load in scene (0)
	bit 6	Scene data following last load (1) or not (0) Note that this bit may be 1 only if bit #7 is 1.
	bit 5	Global (1) or local (0)
	bits 4:0	Transition rate (if load is a dimmer) for RLYDRVR, pulse width ticks code for LVRB (0 for indefinite)
2nd byte:	bits 7:0	Load number (0 to 191)
3rd & 4th bytes:	bits 15:11	Relay number in its RLYDRVR/LVRB Board (0 to 23)
	bits 10:0	Transition level ("000 0000 0000" fully ON, ... , "111 1111 1111" fully OFF for RLYDRVR, "0xx xxxx xxxx" for ON, or "1xx xxxx xxxx" for OFF for LVRB. For a fan speed control, use the following: "000 xxxx xxxx" for 100% speed, "010 xxxx xxxx" for 75% speed, "100 xxxx xxxx" for 50% speed, "110 xxxx xxxx" for 25% speed, and "111 xxxx xxxx" for 0% speed.)

If scene data following last load (bit 6 of 1st byte is 1): more scene bytes as follows:

1st byte:	bit 7	Last scene data (1) or not last scene data (0)
	bit 6	Turn scene ON (1) or OFF (0)
	bits 5:0	Not used
2nd byte:	bits 7:0	Scene number to trigger (0 to 255)
3rd & 4th bytes:	bits 15:0	Delay time to trigger scene in seconds

If action code is $02: no more bytes following (always global).

If action code is $03: no more bytes following (always global).

If action code is $04: no more bytes following (always global). Vacation mode has a fixed time window of operation between 5:00 p.m. and midnight.

If action code is $05: 4 more bytes as follows (similar to action code $01 for Transition loads):

1st byte:	bit 7	Last load in scene (1) or not last load in scene (0)
	bit 6	Scene data following last load (1) or not (0) Note that this bit may be 1 only if bit #7 is 1.
	bit 5	Global (1) or local (0)
	bits 4:0	Transition rate (if load is a dimmer) for RLYDRVR, pulse width ticks code for LVRB (0 for indefinite)
2nd byte:	bits 7:0	Load number (0 to 191)
3rd & 4th bytes:	bits 15:11	Relay number in its RLYDRVR/LVRB Board (0 to 23)
	bits 10:0	Transition level ("000 0000 0000" fully ON, ... , "111 1111 1111" fully OFF for RLYDRVR, "0xx xxxx xxxx" for ON, or "1xx xxxx xxxx" for OFF for LVRB. For a fan speed control, use the following: "000 xxxx xxxx" for 100% speed, "010 xxxx xxxx" for 75% speed, "100 xxxx xxxx" for 50% speed, "110 xxxx xxxx" for 25% speed, and "111 xxxx xxxx" for 0% speed.)

If scene data following last load (bit 6 of 1st byte is 1): more scene bytes as follows:

1st byte:	bit 7	Last scene data (1) or not last scene data (0)
	bit 6	Turn scene ON (1) or OFF (0)
	bits 5:0	Not used
2nd byte:	bits 7:0	Scene number to trigger (0 to 255)
3rd & 4th bytes:	bits 15:0	Delay time to trigger scene in seconds

If action code is $06: 4 more bytes as follows (similar to action code $01 for Transition loads):

1st byte:	bit 7	Last load in scene (1) or not last load in scene (0)
	bit 6	Scene data following last load (1) or not (0) Note that this bit may be 1 only if bit #7 is 1.
	bit 5	Global (1) or local (0)
	bits 4:0	Transition rate (if load is a dimmer) for RLYDRVR, pulse width ticks code for LVRB (0 for indefinite)
2nd byte:	bits 7:0	Load number (0 to 191)
3rd & 4th bytes:	bits 15:11	Relay number in its RLYDRVR/LVRB Board (0 to 23)
	bits 10:0	Transition level ("000 0000 0000" fully ON, ... , "111 1111 1111" fully OFF for RLYDRVR, "0xx xxxx xxxx" for ON, or "1xx xxxx xxxx" for OFF for LVRB. For a fan speed control, use the following: "000 xxxx xxxx" for 100% speed, "010 xxxx xxxx" for 75% speed, "100 xxxx xxxx" for 50% speed, "110 xxxx xxxx" for 25% speed, and "111 xxxx xxxx" for 0% speed.)

If scene data following last load (bit 6 of 1st byte is 1): more scene bytes as follows:

1st byte:	bit 7	Last scene data (1) or not last scene data (0)
	bit 6	Turn scene ON (1) or OFF (0)
	bits 5:0	Not used
2nd byte:	bits 7:0	Scene number to trigger (0 to 255)
3rd & 4th bytes:	bits 15:0	Delay time to trigger scene in seconds

If action code is $07: 5 more bytes as follows (always local).

1st byte:	bit 7	Absolute time (0) or relative time (1)
	bit 6	Sunrise (0) or sunset (1) (only for relative time)
	bit 5	Before (0) or after (1) (only for relative time)
	bit 4	Turn scene OFF (0) or ON (1)
	bit 3	Month data not available (0) or available (1)
	bits 2:0	Not used
2nd byte:	bits 7:0	Scene number (0 to 255) to be triggered
3rd byte:	bit 7	Not used
	bits 6:0	Days of week to trigger scene (bit 0 for Sunday, bit 1 for Monday, bit 2 for Tuesday, bit 3 for Wednesday, bit 4 for Thursday, bit 5 for Friday, and bit 6 for Saturday)
4th byte:	bits 7:6	Not used
	bits 5:4	10 Hours (Absolute or relative time)

	bits	3:0	Hours (24 Hour Format) (Absolute or relative time)
5th byte:	bit	7	Not used
	bits	6:4	10 Minutes (Absolute or relative time)
	bits	3:0	Minutes (Absolute or relative time)

If month data is available (bit 3 of 1st byte is 1): two more bytes as follows:

6th & 7th bytes:	bits	15:12	Not used
	bits	11:0	Months of the year to trigger scene (bit 0 for Jan, bit 1 for Feb, bit 2 for Mar, bit 3 for Apr, bit 4 for May, bit 5 for Jun, bit 6 for Jul, bit 7 for Aug, bit 8 for Sep, bit 9 for Oct, bit 10 for Nov, and bit 11 for Dec)

If action code is $08: more bytes as follows (always local).

1st byte:	bit	7	Send ASCII string to third-party interface RS232-3, channel A of UART2 (1) or not (0)
	bit	6	Send ASCII string to third-party interface RS232-2, channel B of UART1 (1) or not (0)
	bits	5:0	Not used
Next bytes:	contain the ASCII string ended with $00		

Switch Status (States):

There are 384 entries, each has 3 bytes, as follows.

1st byte:	bit	7	Not used
	bit	6	Short ON Flag (1: ON Counter reached max Short ON value)
	bit	5	Long ON Flag (1: ON Counter reached max Long ON value)
	bit	4	Stuck ON Flag (1: ON Counter reached max Stuck ON value)
	bits	3:0	De-bounce Counter
2nd & 3rd bytes:			ON Counter

The order of the switch entries is as follows:

SW1 for STARS 1A,1B,1C,1D, 2A,2B,2C,2D, 3A,3B,3C,3D, 4A,4B,4C,4D (16 entries, 48 bytes)

SW2 for STARS 1A,1B,1C,1D, 2A,2B,2C,2D, 3A,3B,3C,3D, 4A,4B,4C,4D (16 entries, 48 bytes)

...

SW24 for STARS 1A,1B,1C,1D, 2A,2B,2C,2D, 3A,3B,3C,3D, 4A,4B,4C,4D (16 entries, 48 bytes)

Load Status (States):

There are 192 entries, each has 3 bytes, as follows.

1st & 2nd bytes:	bit	15	Load state (0:OFF , 1:ON)
	bit	14	Ramp direction (0:from OFF to ON, 1:from ON to OFF)
	bit	13	Protected (0:not protected, 1:protected)
	bits	12:11	Not used
	bits	10:0	11-bit integer part of current level for RLYDRVR

			("000 0000 0000" always ON, ... , "111 1111 1111" always OFF), 3-bit current level ("000" ON and "111" OFF) + upper 8 bits of pulse width ticks for LVRB
3rd byte:	bits	7:0	8-bit fractional part of current level for RLYDRVR, lower 8 bits of pulse width ticks for LVRB

The order of the load entries is as follows:

LD1, LD2, LD3, ... , LD24 for RLYDRVR/LVRB 1 (24 entries, 72 bytes)
LD1, LD2, LD3, ... , LD24 for RLYDRVR/LVRB 2 (24 entries, 72 bytes)
...
LD1, LD2, LD3, ... , LD24 for RLYDRVR/LVRB 8 (24 entries, 72 bytes)

Scene Status (States):

There are 256 entries, each has 5 bytes, as follows.

1st byte:	bit	7	Scene ON or OFF activated (1:ON/OFF invoked, 0:ON/OFF not invoked)
	bit	6	Scene ON activated (1:ON invoked, 0:ON not invoked)
	bit	5	Scene OFF activated (1:OFF invoked, 0:OFF not invoked)
	bit	4	Scene progress state (0:done, 1:in progress)
	bit	3	Protected (0:not protected, 1:protected)
	bit	2	Delayed (0:not delayed, 1:delayed)
	bit	1	Last action done to scene (0:OFF, 1:ON)
	bit	0	Ramp direction (0:from OFF to ON, 1:from ON to OFF)
2nd byte:	bits	7:0	Scene load counter
3rd & 4th bytes:	bits	15:0	Scene trigger time in minutes past mid-night (if timed scenes), or delay time to trigger scene in seconds (if delayed).
5th byte:	bits	7:0	Display (LED) delay counter

Sunrise/Sunset Table:

There are 12 entries, one entry for each month (January, February, march, April, May, June, July, August, September, October, November and December). Each entry has 4 bytes. The first two bytes represent sunrise time (24 hour format) for the first day of that month, and the next two bytes represent sunset time (24 hour format) for the same day, as follows.

1st byte:	bits	7:6	Not used
	bits	5:4	10 Hours (Absolute time for sunrise)
	bits	3:0	Hours (24 Hour Format) (Absolute time for sunrise)
2nd byte:	bit	7	Not used
	bits	6:4	10 Minutes (Absolute time for sunrise)
	bits	3:0	Minutes (Absolute time for sunrise)
3rd byte:	bits	7:6	Not used
	bits	5:4	10 Hours (Absolute time for sunset)
	bits	3:0	Hours (24 Hour Format) (Absolute time for sunset)
4th byte:	bit	7	Not used

	bits	6:4	10 Minutes (Absolute time for sunset)
	bits	3:0	Minutes (Absolute time for sunset)

Customer Options:

There are two bytes that contain customer options, as follows.

1st byte:	bit	7	Disable daylight saving/standard time adjustment (1:disable, 0:enable)
	bit	6	Enable sending a CR after data for third-party get commands (0: disable, 1: enable), or after sending switch action
	bits	5:0	Not used
2nd byte:	bits	7:0	Number of rings required before modem automatically answers a call (0: disables auto-answer mode, 01-99: range of rings, BCD format)

All Instant Switch Values (ON/OFF):

There are 24 entries, each has 2 bytes. Each two-byte entry holds the ON/OFF state of 16 switches (0 for OFF and 1 for ON), as follows.

1st byte	bit	0	STARS 1A
	bit	1	STARS 1B
	bit	2	STARS 1C
	bit	3	STARS 1D
	bit	4	STARS 2A
	bit	5	STARS 2B
	bit	6	STARS 2C
	bit	7	STARS 2D
2nd byte	bit	0	STARS 3A
	bit	1	STARS 3B
	bit	2	STARS 3C
	bit	3	STARS 3D
	bit	4	STARS 4A
	bit	5	STARS 4B
	bit	6	STARS 4C
	bit	7	STARS 4D

The order of the switch entries is as follows:

SW1 for STARS 1A,1B,1C,1D, 2A,2B,2C,2D, 3A,3B,3C,3D, 4A,4B,4C,4D (2 bytes)
SW2 for STARS 1A,1B,1C,1D, 2A,2B,2C,2D, 3A,3B,3C,3D, 4A,4B,4C,4D (2 bytes)
...
SW24 for STARS 1A,1B,1C,1D, 2A,2B,2C,2D, 3A,3B,3C,3D, 4A,4B,4C,4D (2 bytes)

All Instant Load Values (ON/OFF):

There are 8 entries, each has 3 bytes. Each 3-byte entry holds the ON/OFF state of 24 loads (0 for OFF and 1 for ON), as follows.

1st byte	bit	0	Load 1

	bit	1	Load 2
	…		
	bit	7	Load 8
2nd byte	bit	0	Load 9
	bit	1	Load 10
	…		
	bit	7	Load 16
3rd byte	bit	0	Load 17
	bit	1	Load 18
	…		
	bit	7	Load 24

The order of the load entries is as follows:

LD1, LD2, LD3, ... , LD24 for RLYDRVR/LVRB 1 (3 bytes)
LD1, LD2, LD3, ... , LD24 for RLYDRVR/LVRB 2 (3 bytes)
...
LD1, LD2, LD3, ... , LD24 for RLYDRVR/LVRB 8 (3 bytes)

Miscellaneous Parameters:

There are two bytes that contain miscellaneous parameters, as follows.

1st byte:	bits	7:0	Not used
2nd byte:	bits	7:4	Not used
	bits	3:0	Size of buffer for that channel as follows: 0001: 64 bytes, 0010: 128 bytes, 0011: 192 bytes, 0100: 256 bytes, 0101: 320 bytes, 0110: 384 bytes, 0111: 448 bytes, 1000: 512 bytes, 1001: 576 bytes, 1010: 640 bytes, 1011: 704 bytes, 1100: 768 bytes, 1101: 832 bytes, 1110: 896 bytes, 1111: 960 bytes, 0000: 1024 bytes

Error Status of all RLYDRVR/LVRB Boards:

There is one byte that contains instant error status for all 8 RLYDRVR/LVRB boards, one bit for each board, as follows.

1st byte:	bit	0	For RLYDRVR/LVRB board #0 (0:no error, 1:error)
	bit	1	For RLYDRVR/LVRB board #1 (0:no error, 1:error)
	bit	2	For RLYDRVR/LVRB board #2 (0:no error, 1:error)
	bit	3	For RLYDRVR/LVRB board #3 (0:no error, 1:error)
	bit	4	For RLYDRVR/LVRB board #4 (0:no error, 1:error)
	bit	5	For RLYDRVR/LVRB board #5 (0:no error, 1:error)
	bit	6	For RLYDRVR/LVRB board #6 (0:no error, 1:error)
	bit	7	For RLYDRVR/LVRB board #7 (0:no error, 1:error)

Communications between a Remote PC (Personal Computer) and the FC CONTROLLER:

The FC CONTROLLER can communicate with a remote PC over the phone lines through a modem and an RS232 (channel B of DUART). Communications parameters are 2400 baud, 8 data bits, 1 stop bit and no parity. This communications link may be used by a remote PC to either program the FC CONTROLLER with different configurations, retrieve status information from the FC CONTROLLER, or command the FC CONTROLLER to perform a certain task. The FC CONTROLLER always acts as a slave to a remote PC, i.e., it always responds to commands from that remote PC. Upon power up, the modem is initialized by software and configured to answer automatically incoming calls after the number of rings programmed in the 2nd byte of Customer Options (0: disables auto-answer mode, 01-99: range of rings, BCD format). The modems answers an incoming call after the programmed number of rings (if enabled), and establishes communication with the above parameters. A remote PC can then proceed to communicate with the FC CONTROLLER in exactly the same manner as described under "Communications between a PC (Personal Computer) and the FC CONTROLLER". When the calling PC is done, it should hang up the phone line, after which the FC CONTROLLER will terminate its answer session.

Communications between a Third-Party Device and the FC CONTROLLER:

The FC CONTROLLER can communicate with a third-party device through two RS232 interfaces (channel A of DUART using connector RS232-1, and channel URT1 of the CPU using connector RS232-2). Communications parameters for connector RS232-1 are either 19.2 K baud (if DIP Switch 3 is OFF), or 9.6 K baud (if DIP Switch 3 is ON), 8 data bits, 1 stop bit and no parity. Communications parameters for connector RS232-2 are either 19.2 K baud (if DIP Switch 4 is OFF), or 9.6 K baud (if DIP Switch 4 is ON), 8 data bits, 1 stop bit and no parity. This communications link may be used by a third-party device to either send some control commands or receive some status information regarding both loads and scenes. The FC CONTROLLER always acts as a slave to a third-party device, i.e., the FC CONTROLLER always responds to commands from that third-party device. In addition, the FC CONTROLLER can send an ASCII string to both third-party channels indicating when a switch is pressed and when that switch is released (if that switch is programmed to do so). The ASCII string format is "**Psnnn**" for a pressed switch, and "**Rsnnn**" for a released switch, where **s** is a 1-digit ASCII number specifying FC CONTROLLER board number (0 for a single-system), and **nnn** is a 3-digit ASCII number specifying that switch number (from 001 to 384). In a single-system configuration (one FC CONTROLLER board), up to two third-party devices can be connected through connectors RS232-1 and RS232-2 on that FC CONTROLLER board.

Communications between a third-party device and the FC CONTROLLER in a single-system (one FC CONTROLLER board) configuration is always using ASCII code (both directions). All commands are started with ASCII "^" (ASCII code $5E), followed by a 1-byte command code. For a single-system configuration, command codes are upper-case ASCII "A" - "L" only. The command may be followed by a number of ASCII digits that varies according to the command used. The following table shows the currently supported commands in a single-system

configuration, their formats as well as response expected if any. Note that each response may be followed (if enabled in Customer Options) by an ASCII carriage return byte (ASCII code $0D).

Command Code and Format	Meaning	Response
^Annn	Activate load/relay number **nnn**.	None
^Bnnn	De-activate load/relay number **nnn**.	None
^Cnnn	Activate scene number **nnn**.	None
^Dnnn	De-activate scene number **nnn**.	None
^Ennnllrr	Activate load/relay number **nnn** to level **ll** at rate/pulse width **rr**.	None
^Fnnn	Get level of load/relay number **nnn**.	**ll**
^G	Get instant ON/OFF status of all loads/relays.	**ddd … d**
^H	Get instant ON/OFF status of all switches.	**sss … s**
^Innn	Press switch number **nnn**.	None
^Jnnn	Release switch number **nnn**.	None
^K	Get instant real-time clock settings.	**ssmmhhwwddmmyy**
^Lssmmhhwwddmmyy	Set real-time clock settings.	None

If enable sending a CR after data for third-party get commands option is chosen (bit #6 of 1st byte in the Customer Options is set), then a CR will be sent after data sent as a response to every get command (e.g., ^F, ^G, ^H, and ^K).

nnn is a 3-digit ASCII number specifying either a load/relay number (from 001 to 192), a scene number (from 001 to 256), or a switch number (from 001 to 384). Note that, since an LVRB has only 16 relays, relay numbers range from 001 to 016 (1st LVRB), 025 to 040 (2nd LVRB), 049 to 064 (3rd LVRB), 073 to 088 (4th LVRB), 097 to 112 (5th LVRB), 121 to 136 (6th LVRB), 145 to 160 (7th LVRB), and 169 to 184 (8th LVRB).
ll is a 2-digit ASCII number specifying load/relay level value (from 00 to 99, where 00 is fully OFF, and 99 is fully ON). For a relay in an LVRB, only levels allowed are 00 (OFF) or 99 (ON).
rr is a 2-digit number specifying a code for rate (from 00 to 31) at which a load in an RLYDRVR board is activated to the specified level, according to the following table.

"**rr**"	Value	"**rr**"	Value	"**rr**"	Value	"**rr**"	Value
00	Immediate	08	9 seconds	16	41 seconds	24	210 seconds
01	1 second	09	11 seconds	17	49 seconds	25	250 seconds
02	2 seconds	10	13 seconds	18	60 seconds	26	300seconds
03	3 seconds	11	16 seconds	19	75 seconds	27	380 seconds
04	4 seconds	12	19 seconds	20	90 seconds	28	450 seconds
05	5 seconds	13	23 seconds	21	110 seconds	29	550 seconds
06	6 seconds	14	28 seconds	22	140 seconds	30	675 seconds
07	7 seconds	15	34 seconds	23	175 seconds	31	800 seconds

rr is a 2-digit number specifying a code for rate (from 00 to 31) at which a relay in an LVRB board is activated for a specified pulse width, according to the following table.

"**rr**"	Value	"**rr**"	Value	"**rr**"	Value	"**rr**"	Value
7	**00 INDEFINITE**	**08**	**3.00 SECONDS**	**16**	**14.0 SECONDS**	**24**	**45.0 SECONDS**
01	0.25 seconds	09	4.00 seconds	17	16.0 seconds	25	50.0 seconds
02	0.50 seconds	10	5.00 seconds	18	18.0 seconds	26	60.0 seconds
03	0.75 second	11	6.00 seconds	19	20.0 seconds	27	70.0 seconds
04	1.00 seconds	12	7.00 seconds	20	25.0 seconds	28	80.0 seconds
05	1.50 seconds	13	8.00 seconds	21	30.0 seconds	29	90.0 seconds
06	2.00 seconds	14	10.0 seconds	22	35.0 seconds	30	100 seconds
07	2.50 seconds	15	12.0 seconds	23	40.0 seconds	31	120 seconds

ddd … d is a 48-digit ASCII hex number, where every 6-digit entry holds the ON/OFF state of 24 loads or 16 relays (0 for OFF and 1 for ON), as follows.

1st 2 digits (least significant)	bit	0	Load 1
	bit	1	Load 2
	…		
	bit	7	Load 8
2nd 2 digits (middle significant)	bit	0	Load 9
	bit	1	Load 10
	…		
	bit	7	Load 16
3rd 2 digits (most significant)	bit	0	Load 17
	bit	1	Load 18
	…		
	bit	7	Load 24

The order of the 8 load entries is as follows:

LD1, LD2, LD3, ... , LD24 for RLYDRVR/LVRB 1 (6 digits)
LD1, LD2, LD3, ... , LD24 for RLYDRVR/LVRB 2 (6 digits)
...
LD1, LD2, LD3, ... , LD24 for RLYDRVR/LVRB 8 (6 digits)

Note that for an LVRB, the last 8 bits (most significant 2 digits) are meaningless, since an LVRB board has only 16 relays.

sss … s is a 96-digit ASCII hex number, where every 4-digit entry holds the ON/OFF state of 16 switches (0 for OFF and 1 for ON), as follows.

1st 2 digits (least significant)	bit	0	STARS 1A
	bit	1	STARS 1B
	bit	2	STARS 1C
	bit	3	STARS 1D
	bit	4	STARS 2A
	bit	5	STARS 2B
	bit	6	STARS 2C
	bit	7	STARS 2D
2nd 2 digits (most significant)	bit	0	STARS 3A
	bit	1	STARS 3B

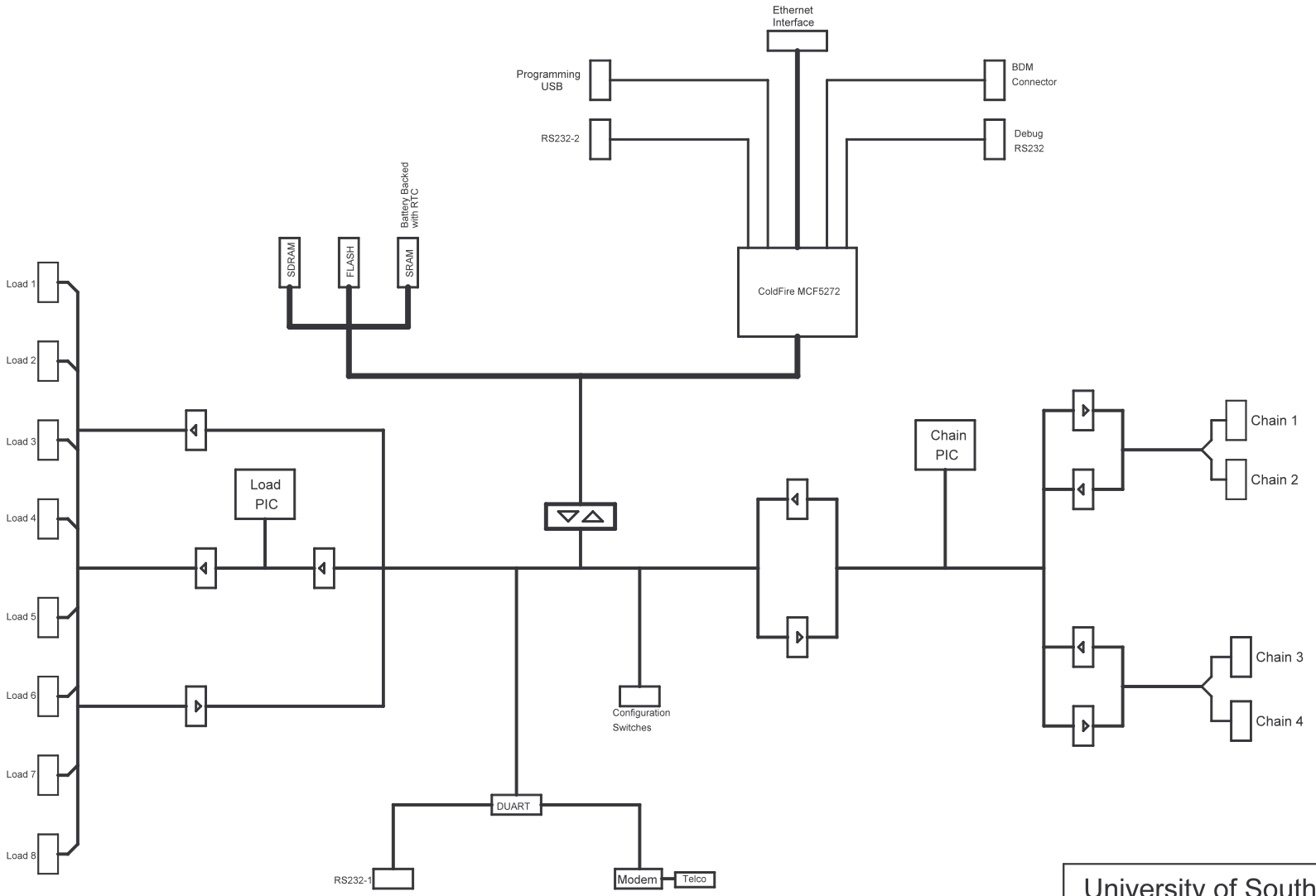
bit	2	STARS 3C
bit	3	STARS 3D
bit	4	STARS 4A
bit	5	STARS 4B
bit	6	STARS 4C
bit	7	STARS 4D

The order of the 24 switch entries is as follows:
SW1 for STARS 1A,1B,1C,1D, 2A,2B,2C,2D, 3A,3B,3C,3D, 4A,4B,4C,4D (4 digits)
SW2 for STARS 1A,1B,1C,1D, 2A,2B,2C,2D, 3A,3B,3C,3D, 4A,4B,4C,4D (4 digits)
...
SW24 for STARS 1A,1B,1C,1D, 2A,2B,2C,2D, 3A,3B,3C,3D, 4A,4B,4C,4D (4 digits)

Note that any combination for **nnn**, **ll** and **rr** outside the specified ranges will cause the command to be ignored. In addition, command **^Ennnllrr** will be ignored if it specifies a level other than 00 or 99, or a transition rate other than 00 for non-dimmer loads.

ssmmhhwwddmmyy are the real-time clock settings consisting of 14 bytes (BCD format) as follows.

Two Bytes	Contents	BCD Format
ss	Seconds	00 - 59
mm	Minutes	00 - 59
hh	Hours (24 hour format)	00 - 23
ww	Day of Week (1:Sunday, 2:Monday, .. , 7:Saturday)	01 - 07
dd	Date (Day of Month)	01 - 31
mm	Month	01 - 12
yy	Year	00 - 99

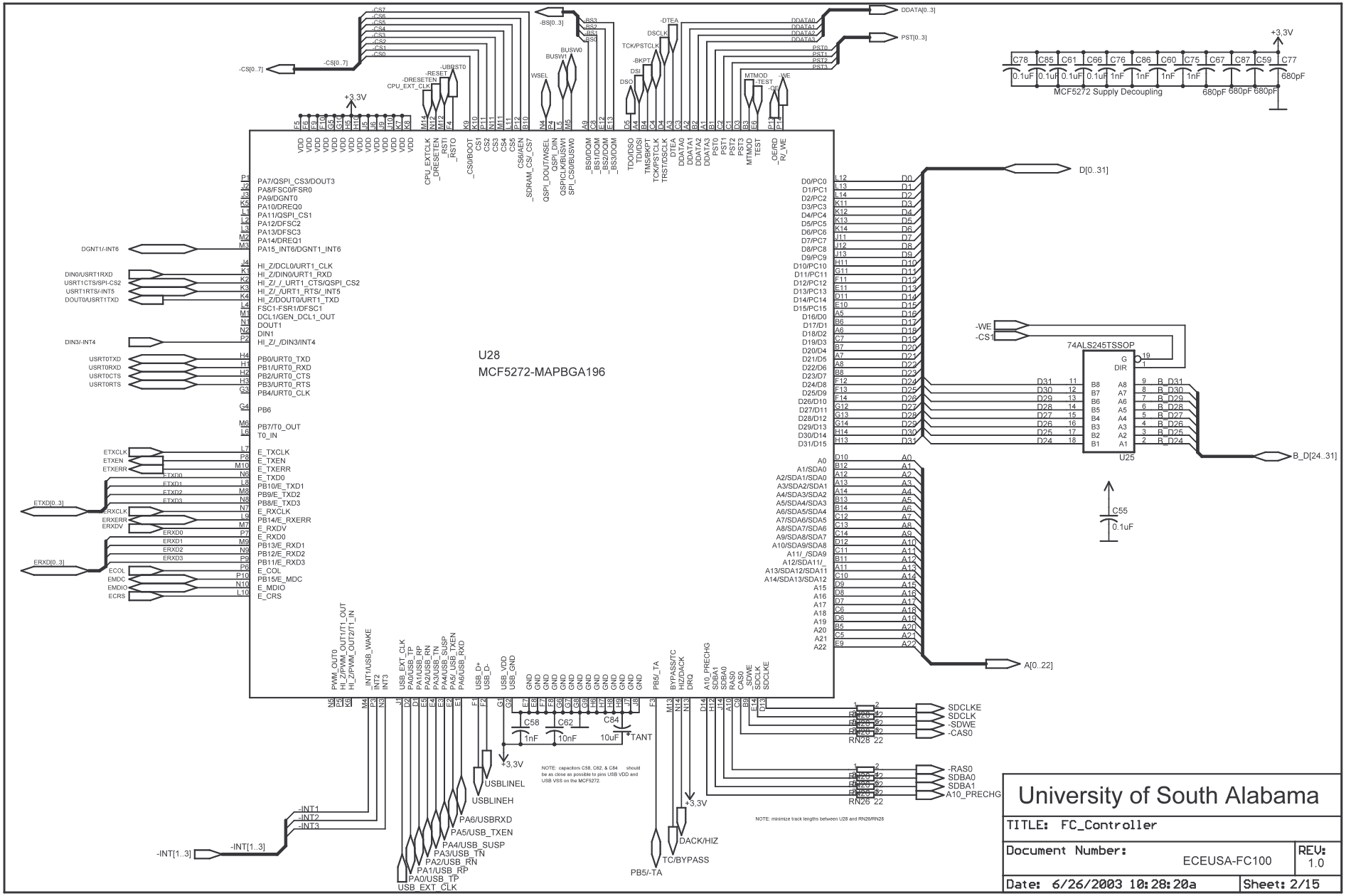


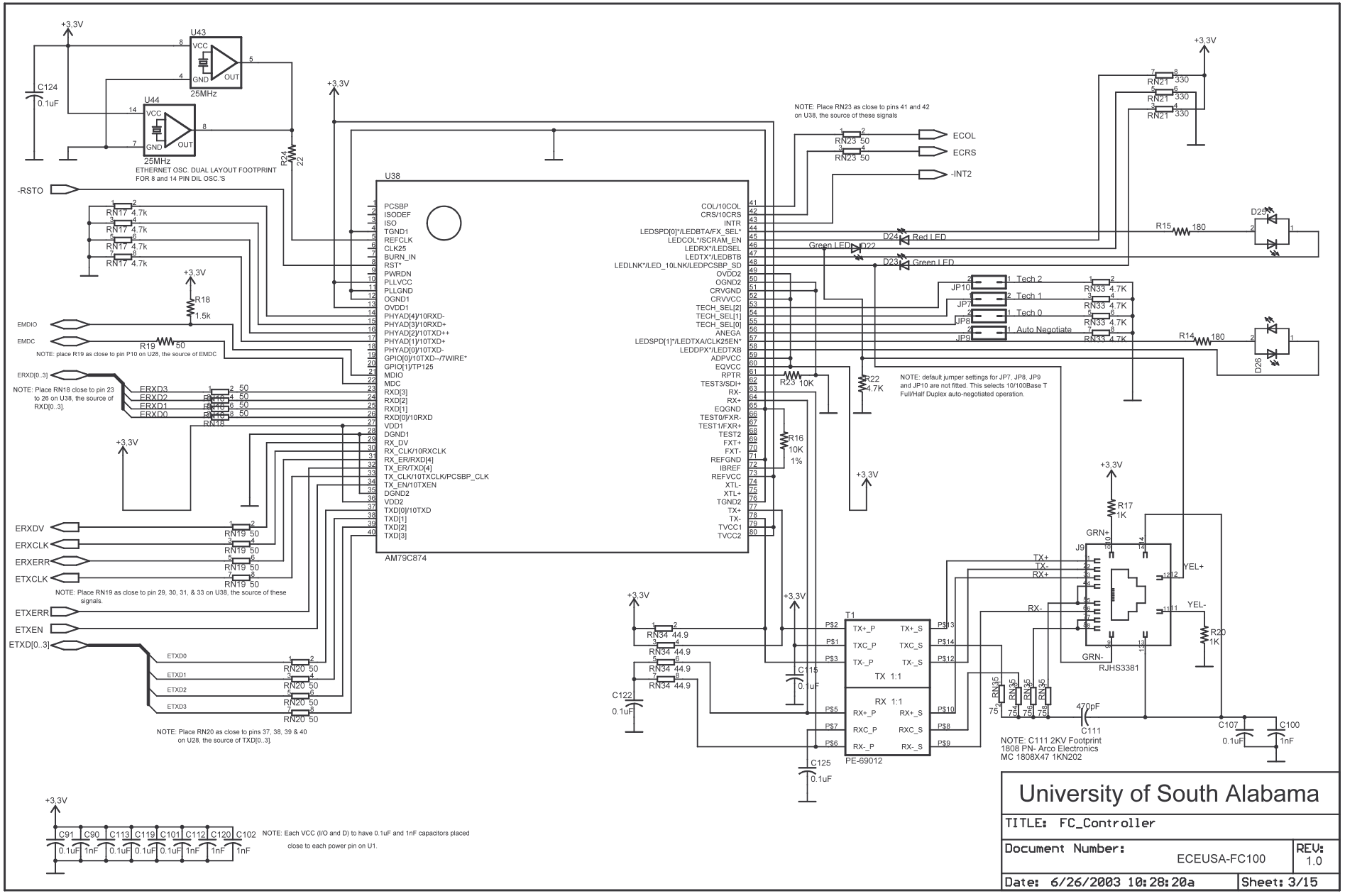
University of South Alabama

TITLE: FC_Controller

Document Number: ECEUSA-FC100 REV: 1.0

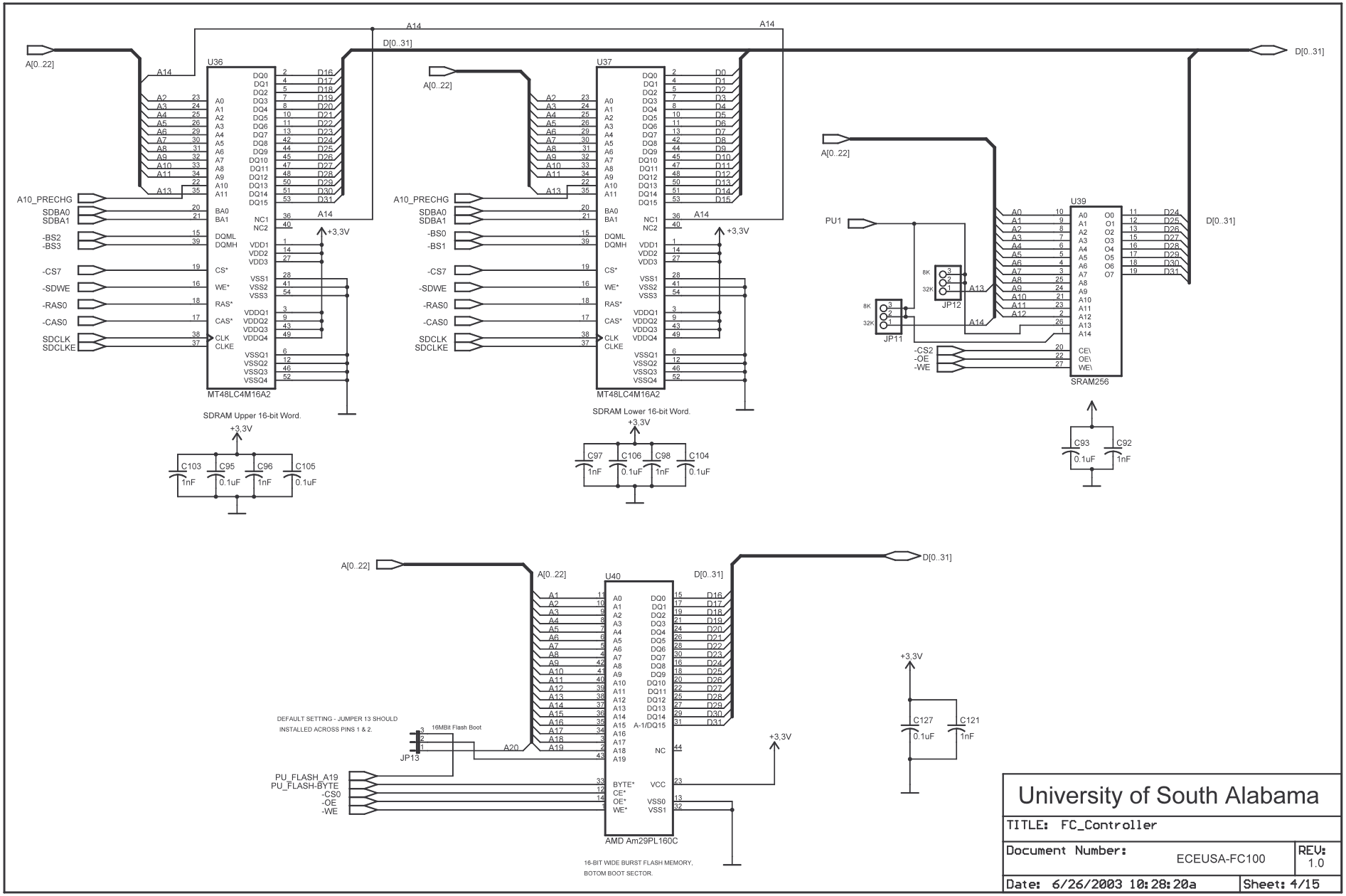
Date: 6/26/2003 10:28:20a Sheet: 1/15


U28
MCF5272-MAPBGA196
MCF5272 Supply Decoupling
74ALS245TSSOP
U25
USBLINEL
USBLINEH
NOTE: capacitors C58, C62, & C84 should be as close as possible to pins USB VDD and USB VSS on the MCF5272.
NOTE: minimize track lengths between U28 and RN26/RN28
University of South Alabama
TITLE: FC_Controller
Document Number: ECEUSA-FC100
REV: 1.0
Date: 6/26/2003 10:28:20a
Sheet: 2/15



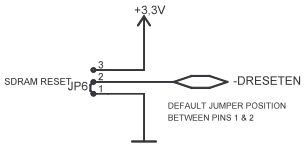
University of South Alabama

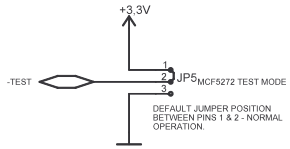
TITLE: FC_Controller	
Document Number: ECEUSA-FC100	REV: 1.0
Date: 6/26/2003 10:28:20a	Sheet: 3/15

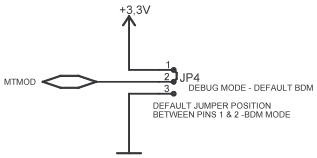


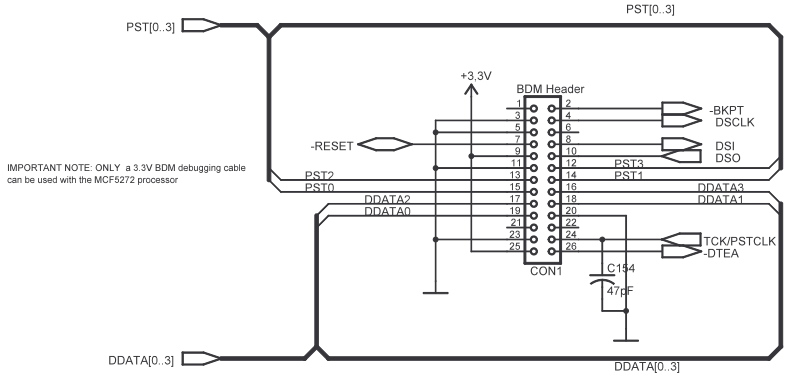
University of South Alabama

TITLE: FC_Controller		
Document Number:	ECEUSA-FC100	REV: 1.0
Date: 6/26/2003 10:28:20a		Sheet: 4/15

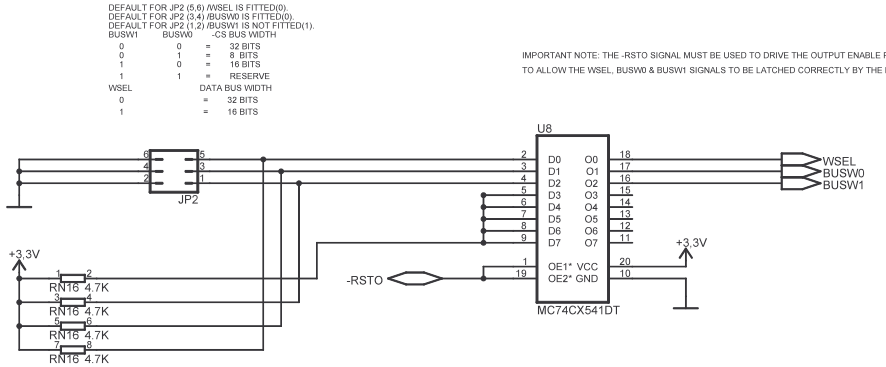




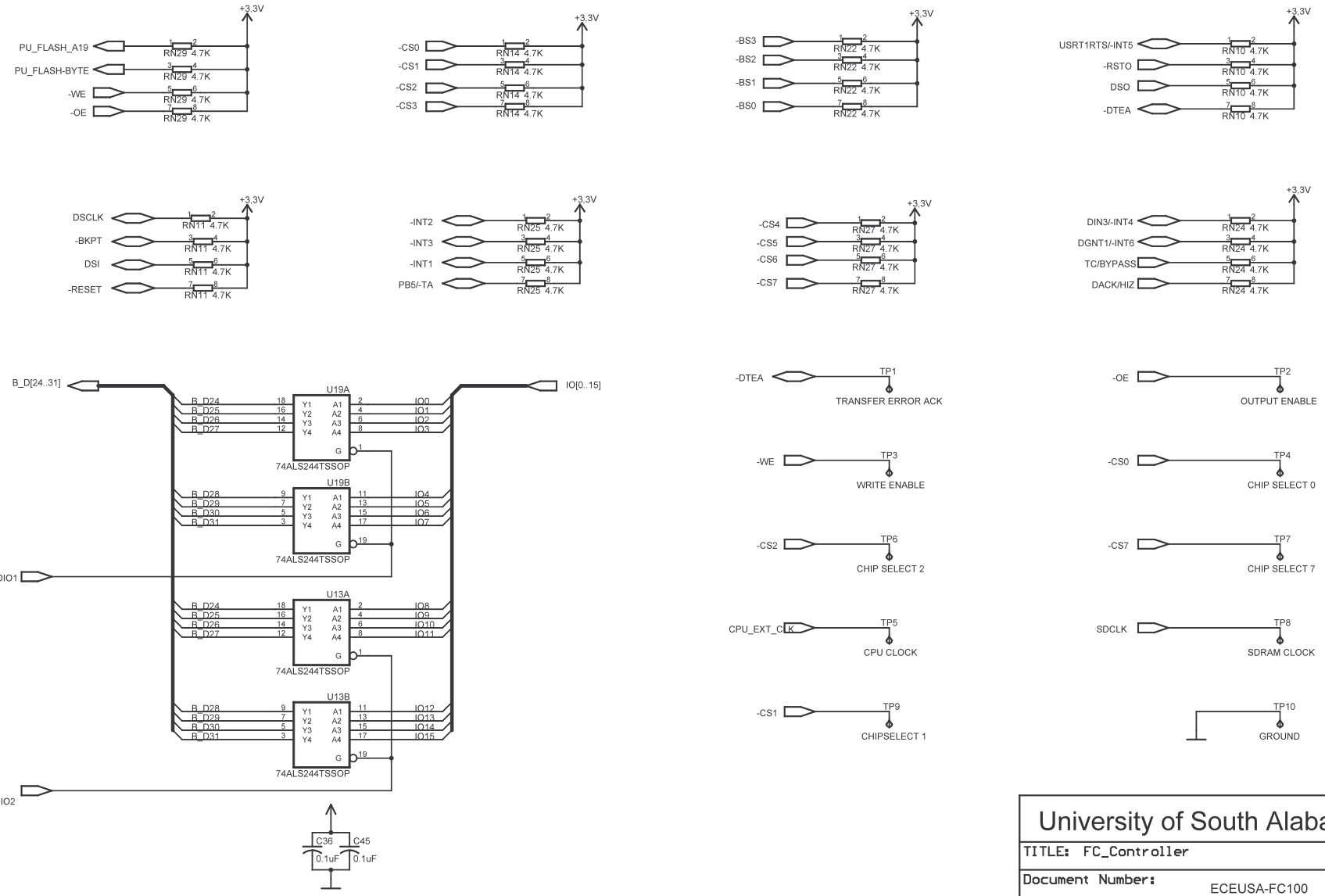




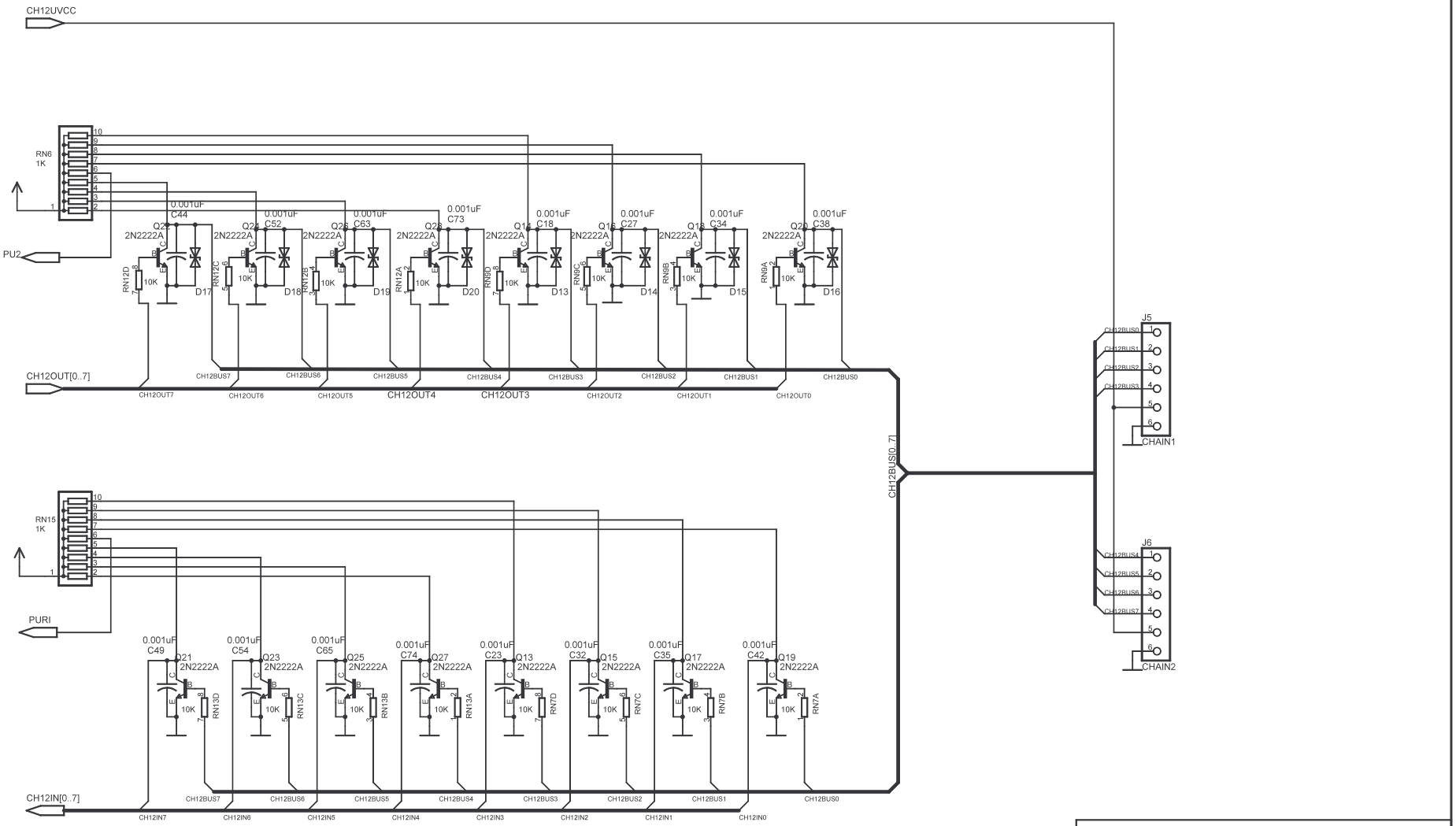
NOTE: 4.7K pull up resistors are used on signals -BKPT, DSCLK
DSI, DSO, -RESET & -DTEA.



University of South Alabama		
TITLE: FC_Controller		
Document Number: ECEUSA-FC100		REV: 1.0
Date: 6/26/2003 10:28:20a	Sheet: 5/15	

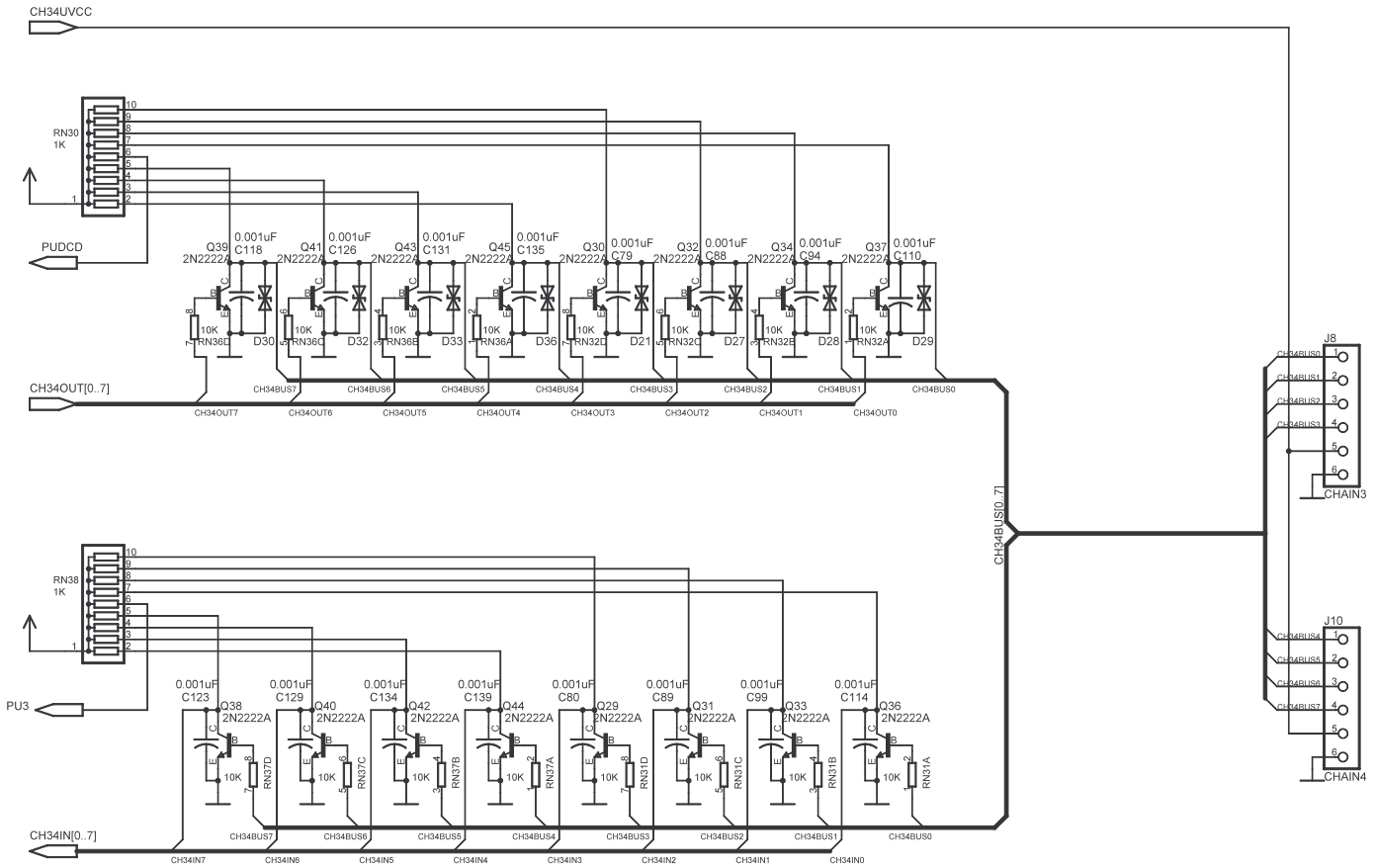


University of South Alabama		
TITLE: FC_Controller		
Document Number:	ECEUSA-FC100	REV: 1.0
Date: 6/26/2003 10:28:20a		Sheet: 6/15

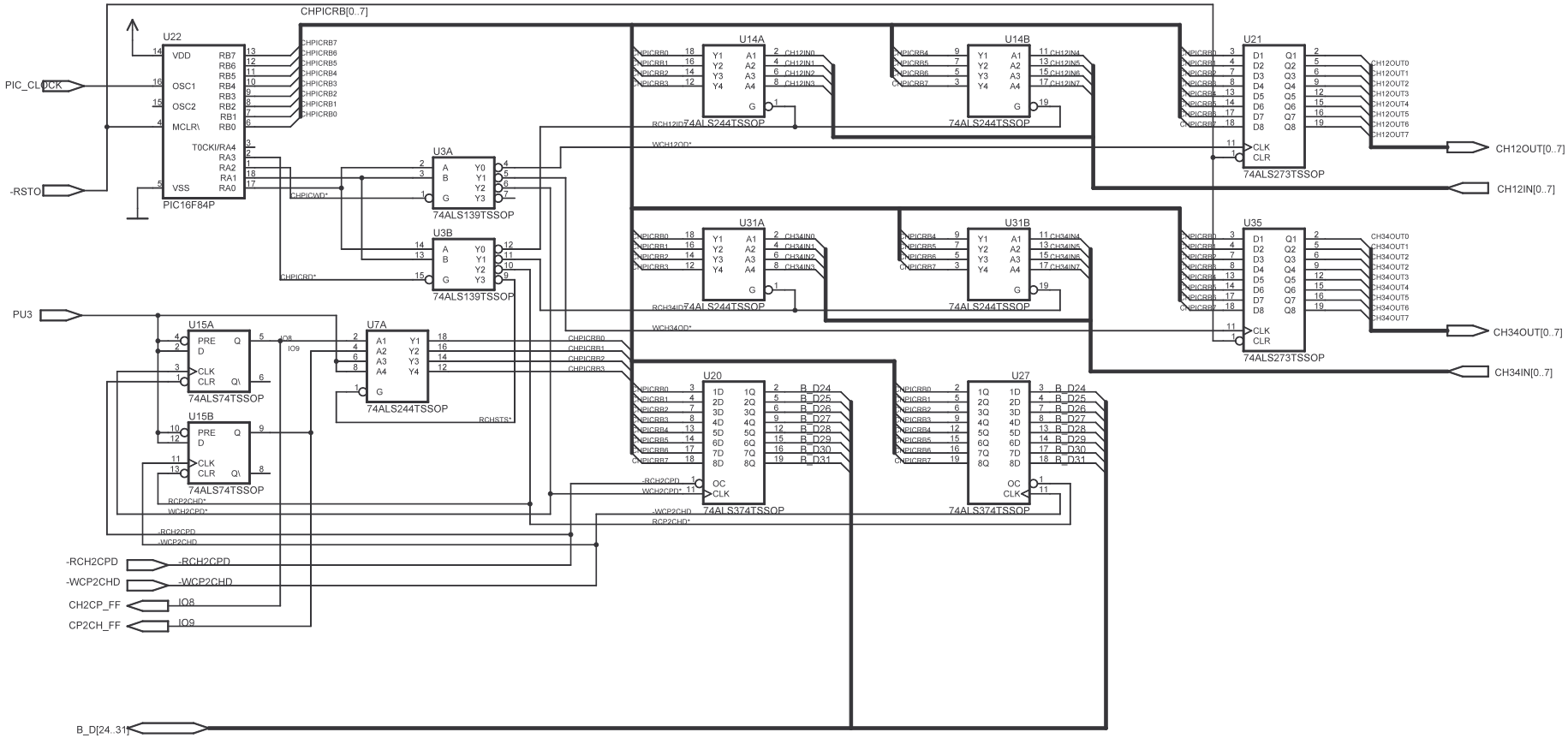


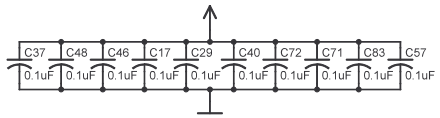
University of South Alabama

TITLE: FC_Controller		
Document Number:	ECEUSA-FC100	REV: 1.0
Date: 6/26/2003 10:28:20a		Sheet: 7/15

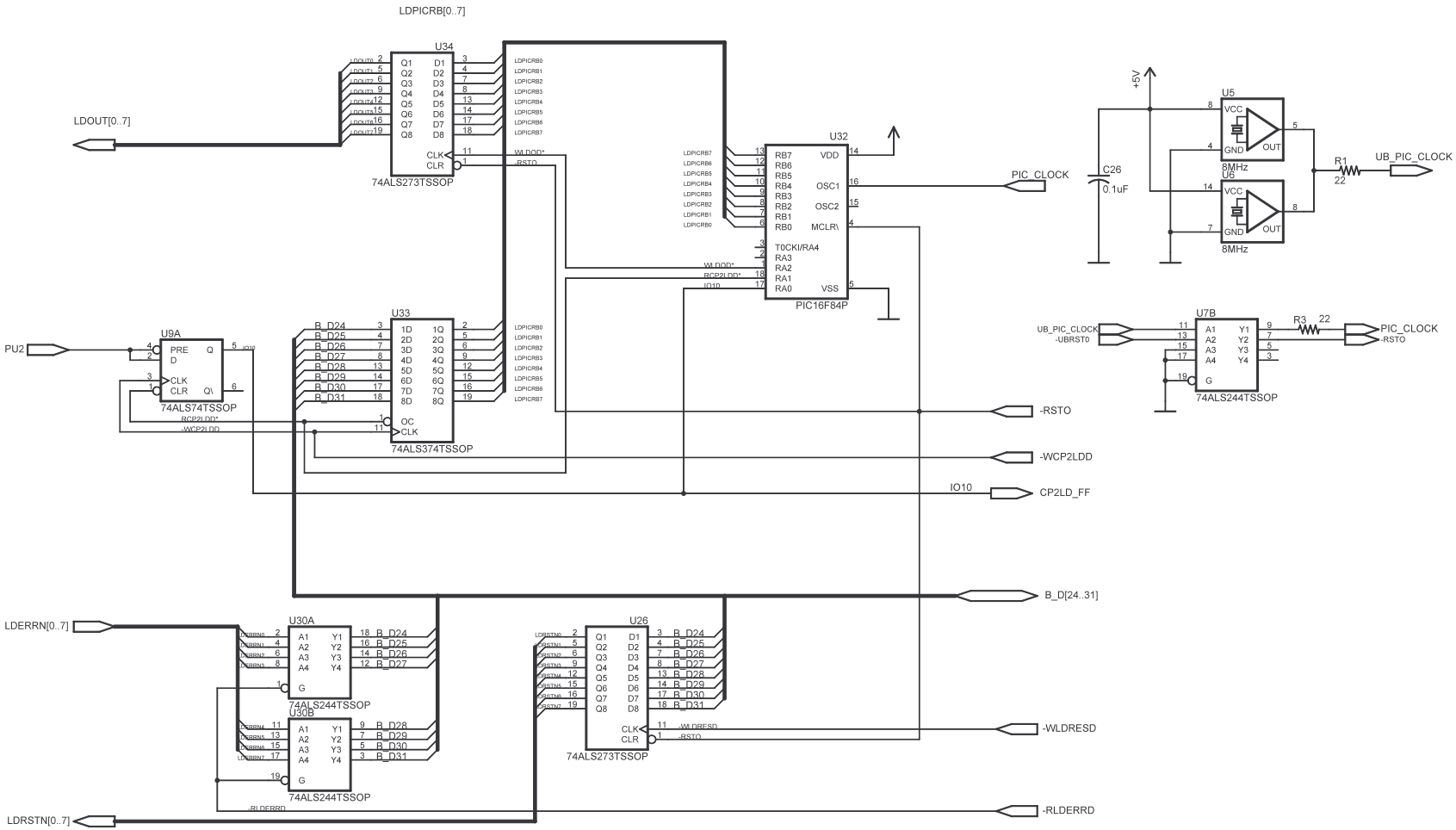


University of South Alabama		
TITLE: FC_Controller		
Document Number:	ECEUSA-FC100	REV: 1.0
Date: 6/26/2003 10:28:20a		Sheet: 8/15

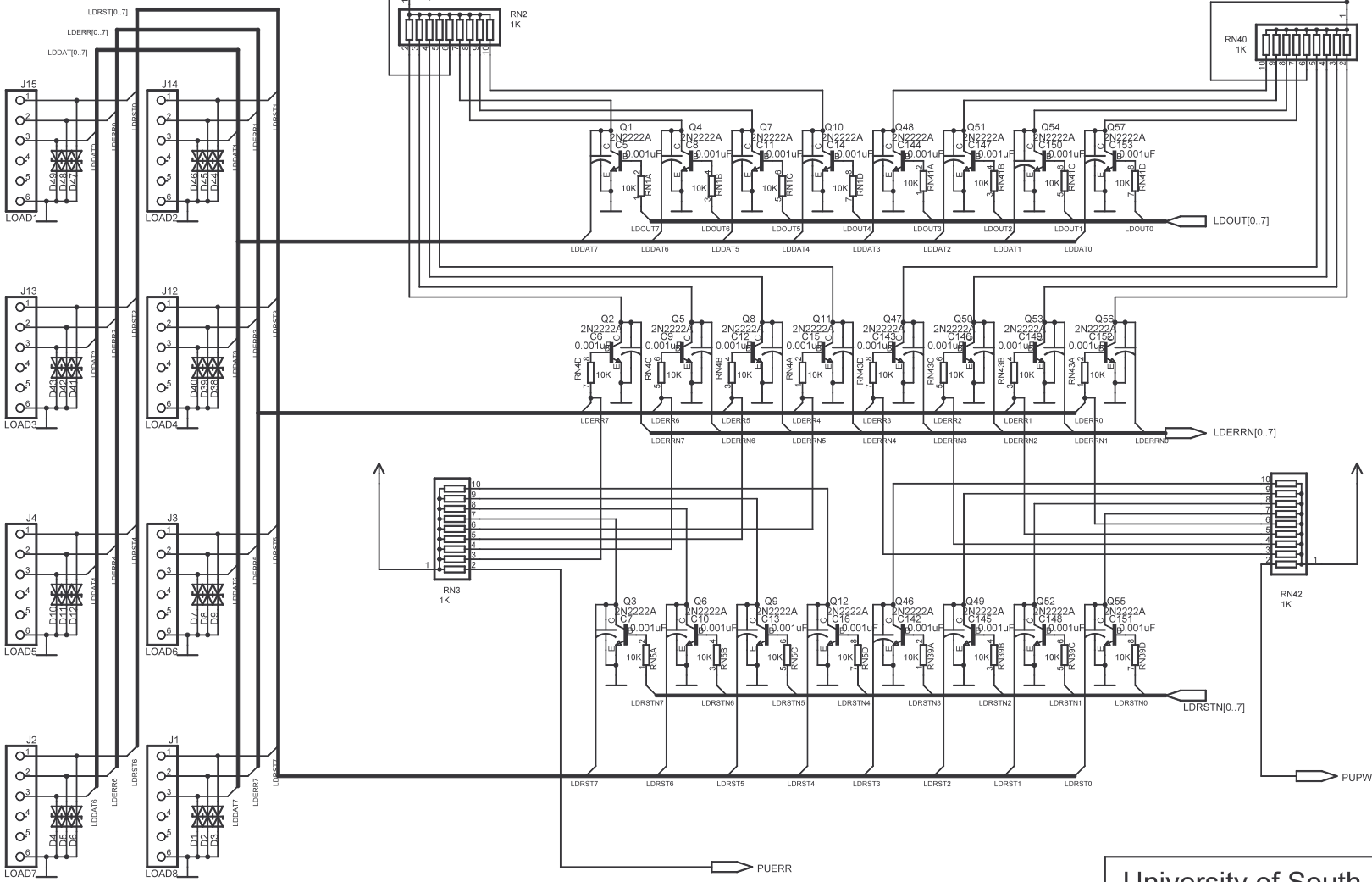




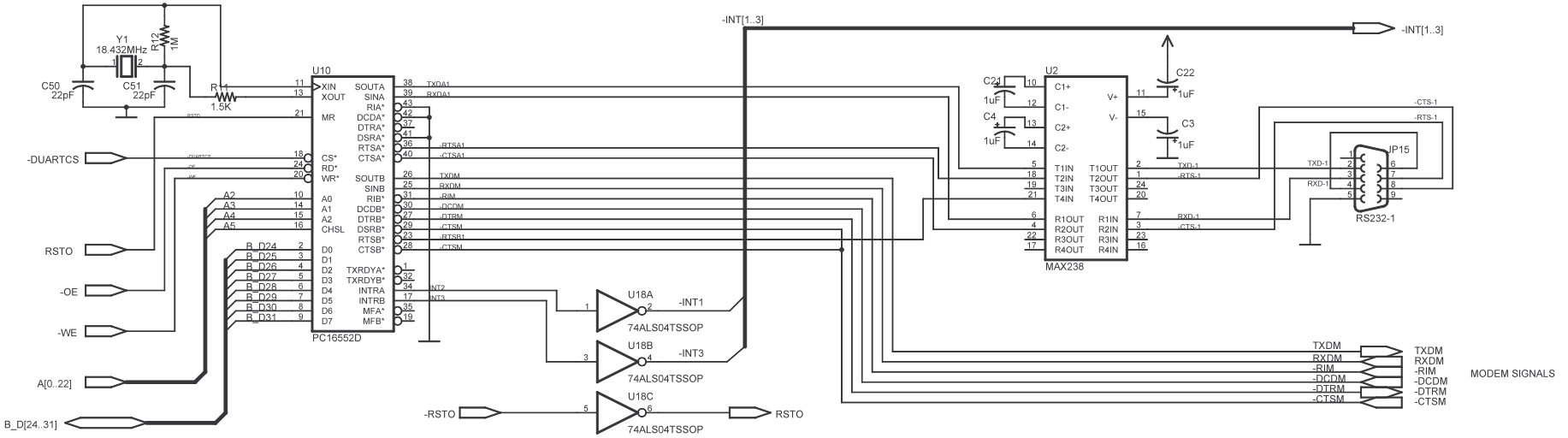
University of South Alabama		
TITLE: FC_Controller		
Document Number:	ECEUSA-FC100	REV: 1.0
Date: 6/26/2003 10:28:20a		Sheet: 9/15

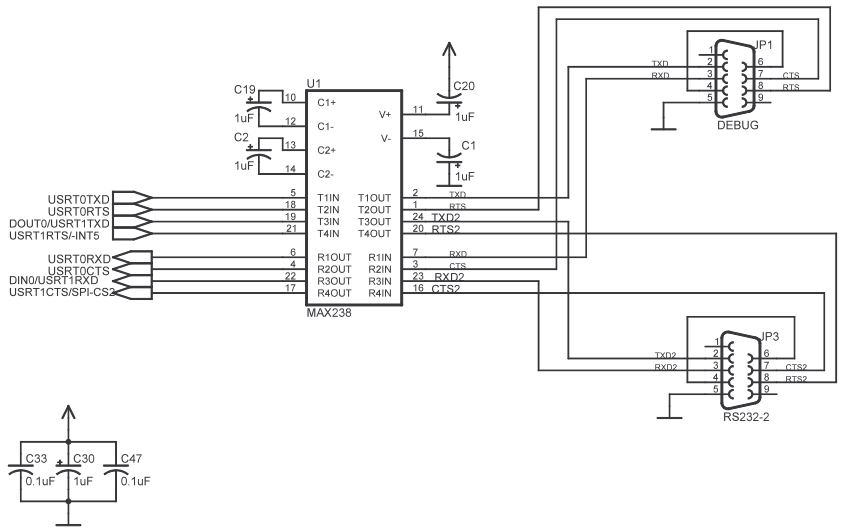


University of South Alabama		
TITLE: FC_Controller		
Document Number: ECEUSA-FC100		REV: 1.0
Date: 6/26/2003 10:28:20a	Sheet: 10/15	



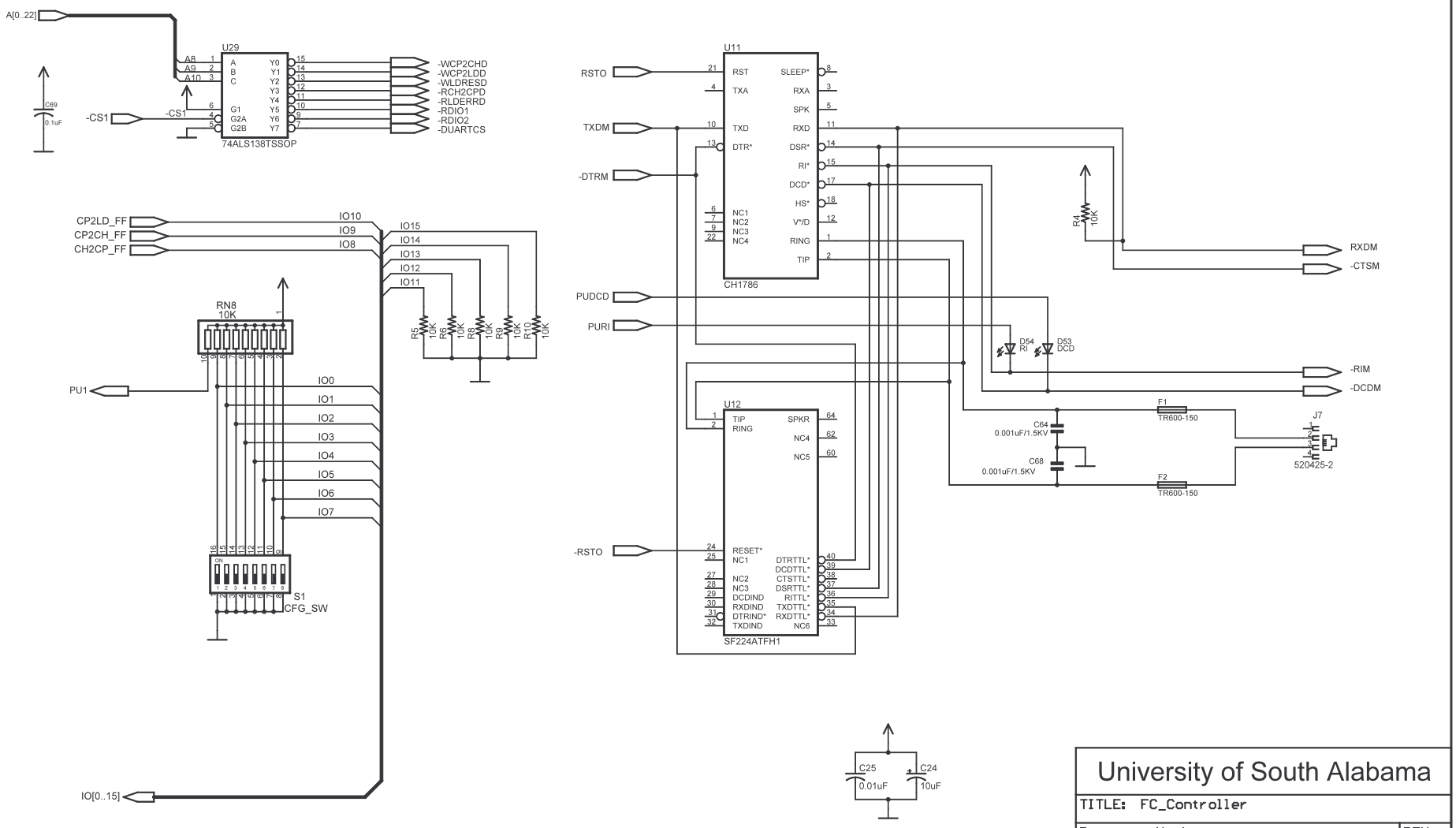
University of South Alabama		
TITLE: FC_Controller		
Document Number: ECEUSA-FC100		REV: 1.0
Date: 6/26/2003 10:28:20a	Sheet: 11/15	



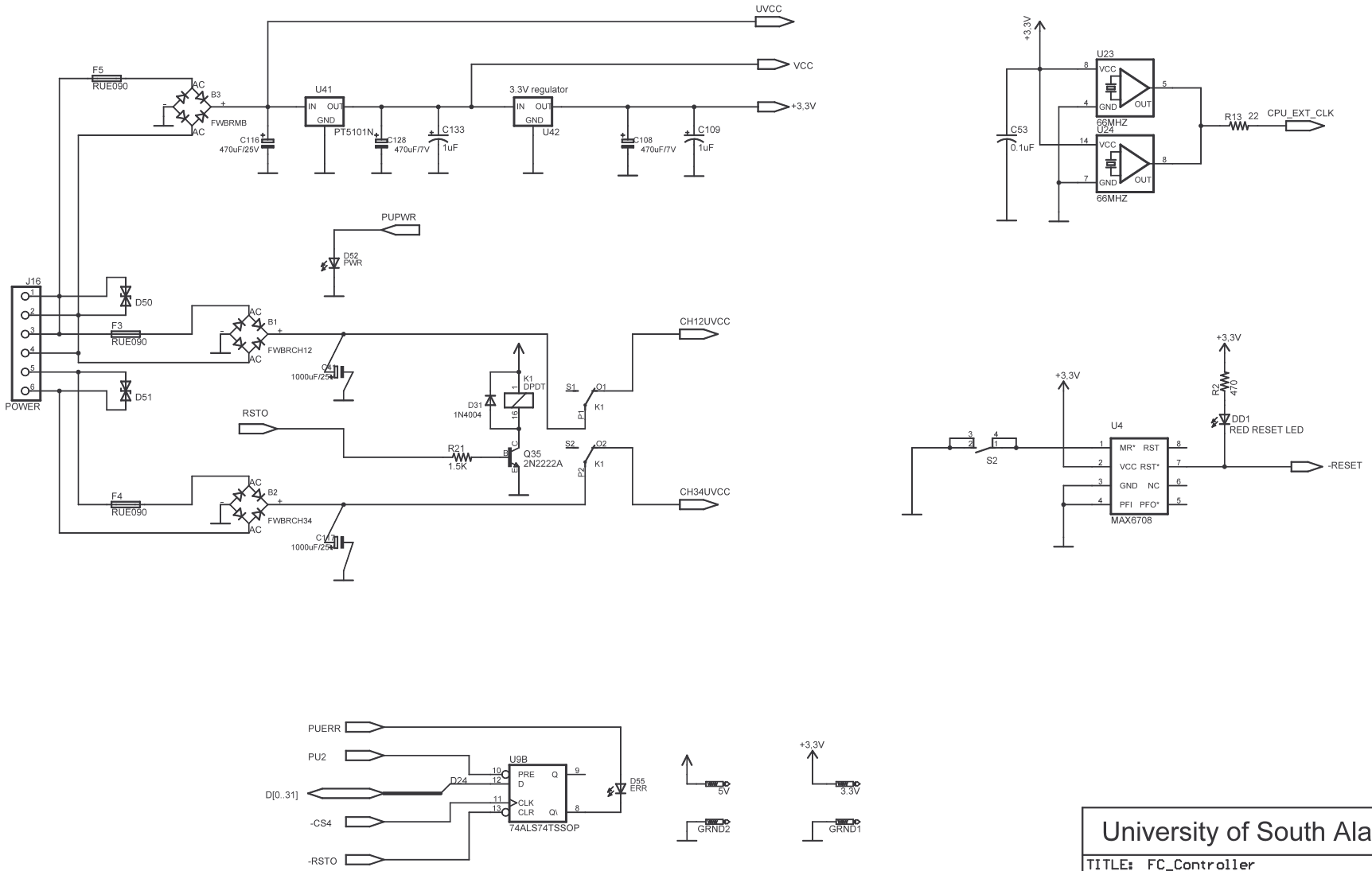


University of South Alabama

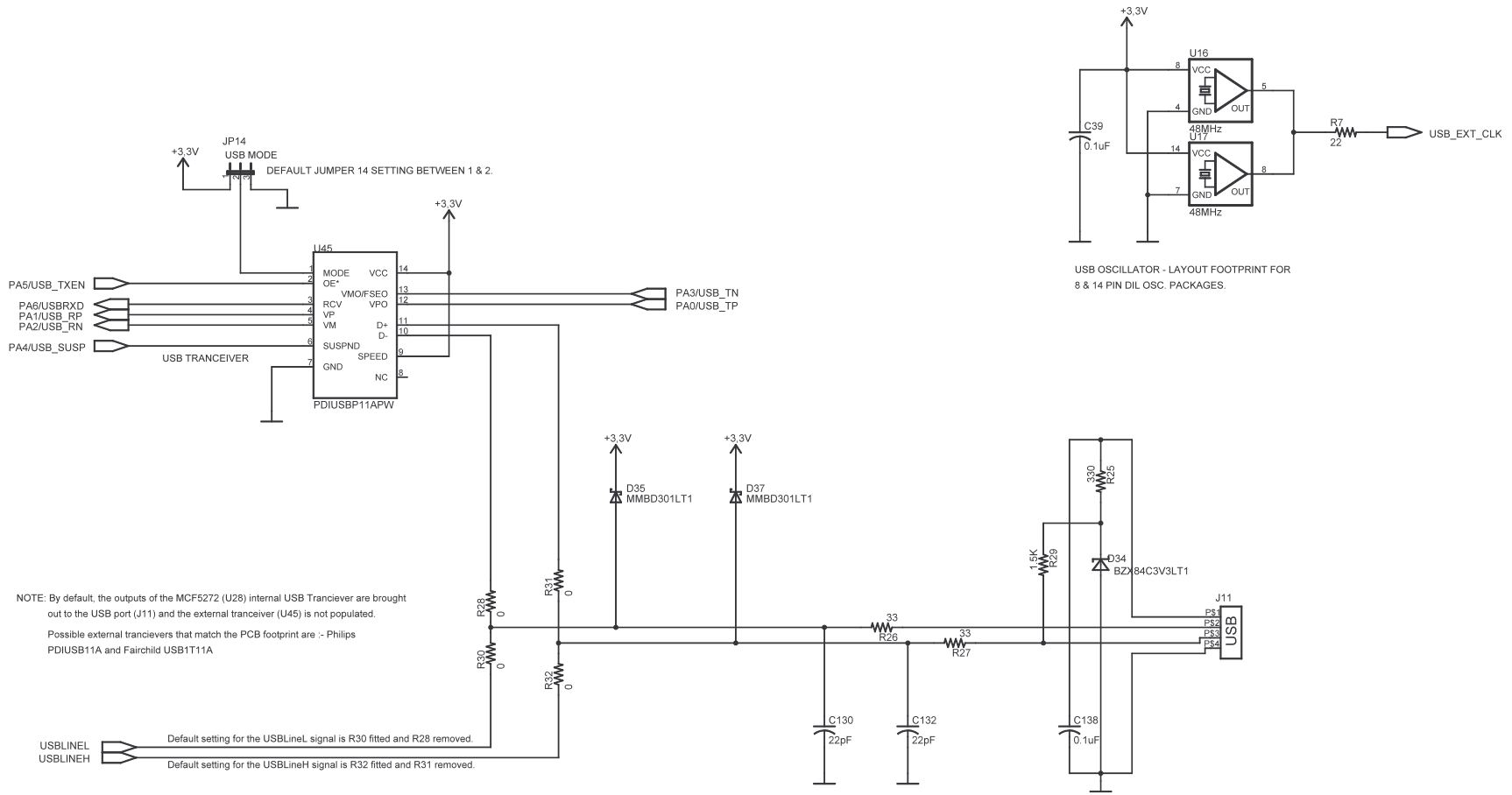
TITLE: FC_Controller		
Document Number:	ECEUSA-FC100	REV: 1.0
Date: 6/26/2003 10:28:20a		Sheet: 12/15



University of South Alabama		
TITLE: FC_Controller		
Document Number:	ECEUSA-FC100	REV: 1.0
Date: 6/26/2003 10:28:20a		Sheet: 13/15



University of South Alabama		
TITLE: FC_Controller		
Document Number:	ECEUSA-FC100	REV: 1.0
Date: 6/26/2003 10:28:20a		Sheet: 14/15



University of South Alabama		
TITLE: FC_Controller		
Document Number:	ECEUSA-FC100	REV: 1.0
Date: 6/26/2003 10:28:20a		Sheet: 15/15

APPENDIX C

Data Acquisition

10/27/2003

GIFCO DAQ Module Description

Ali Mehrabi and Dean Li

1. Overview

The Grid Independent Fuel Cell Operated Smart Home Project (GIFCO) is a study being conducted by the University of South Alabama (USA) and Radiance Technologies (Radiance). GIFCO involves learning human living patterns in a house in order to manage the peaks and valleys of typical home device usage. Electrical load management will be decided by the Smart Energy Management Control or SEMaC. The GIFCO team plans to track living patterns in several ways with a Data Acquisition Module (DAQ). The DAQ will monitor home device electrical usage on a per device level, and also monitor room environmental data as stated in the GIFCO Requirements Document in subsections 3.3.1.1 and 3.3.1.2. The environmental sensors consist of two packages including the Environmental Sensor Box and the Occupancy Sensor. Electrical usage will be monitored by current sensors placed in the relay box. All sensor data will be sent to the multiplexer, and then to the SEMaC.

2. Environmental Sensor

The Environmental Sensor Box will house the temperature and humidity sensors while also receiving the input from the Occupancy Sensor. Incoming power to the box will be +/- 12V. The +12V supply will be fed to a LM7805 regulator which will output +5V. The +5V will be used to power the Humirel HM1500-ND humidity sensor and the LM335 temperature sensor. Their outputs will be connected to TL081 op-amps which use +/- 12V to supply power. The output of the op-amps will then be sent to the SEMaC.

The data from the motion sensor is inputted to a 74HC123 retriggerable monostable multivibrator. The input data is expected to be a normally high +5V which will trigger the multivibrator when it goes low. Once triggered, the multivibrator will output a signal indicating the room is occupied. The room will stay “occupied” for an amount of time that is adjustable by means of a potentiometer up to a period of several minutes. Every time the multivibrator is triggered by the motion sensor, it will reset the timer and continue to show occupancy. If no more motion is detected within the set time period, then the multivibrator will shut its output and the SEMaC will see the room as unoccupied.

3. Occupancy Sensor Logic

The occupancy detector itself is described in the Occupancy Sensor Study. From the study, we concluded that the sensor needed some modifications in order for SEMaC to track room occupancy. The unmodified detector has two different types of motion sensors. It sends the signals from the two sensors to an AND gate and outputs the signal from the AND gate. The modified sensor now sends the two signals to an OR gate. To do this, we took the signals from

two legs of a tri-state LED that indicated which sensor was being set off by its output color. Each signal was sent to the positive input of its own comparator. If either legs of the LED went high, then the corresponding comparator would output a signal. The two comparator's outputs are connected to a OR gate and then inverted with an inverter. The additional logic resides on a small pc board that fits inside the housing of the current motion detector.

4. Current Sensors

Current sensors are used to determine the power draw on each device or circuit. 20A sensors (Bicron EX9LC200) are used for the smaller loads such as lighting and 50A sensors (Bicron EX9LC500) are used for the larger loads such as the oven and HVAC system. Twenty-four of the 20A sensors reside in the low current relay panel box and two are in the high current box. Ten of the 50A sensors reside in the high current box measuring 5 loads; each sensor measuring one leg of the 240V input. The outputs from these current sensors will be connected to a multiplexer board and they will be read by the SEMaC.

5. Multiplexor

There are thirty six current sensing transformers monitoring the current draw on every electrical circuit in the Model House. The corresponding output voltages (using Linear Technology LT1966 RMS-to-DC converters) of these transformers are sent to the SEMaC computer's analog-to-digital converter circuit (Measurement Computing PCI-DAS4020/12 analog-to-digital converter board). The information is then converted to digital format and used by the SEMaC intelligent software for power management. A total of forty eight channels are used to receive power draw, environmental, and occupancy information. The information from the forty eight channels will be sent to a multiplexer circuit (using 74HC4067 analog multiplexers). The job of the multiplexer is to route the information from each channel to the analog-to-digital converter in the Host PC one channel at a time.

A block diagram below shows the basic components of both SEMaC and SHC and their communication links. Further below are the schematics for the environmental sensor board, motion detector logic and multiplexer.

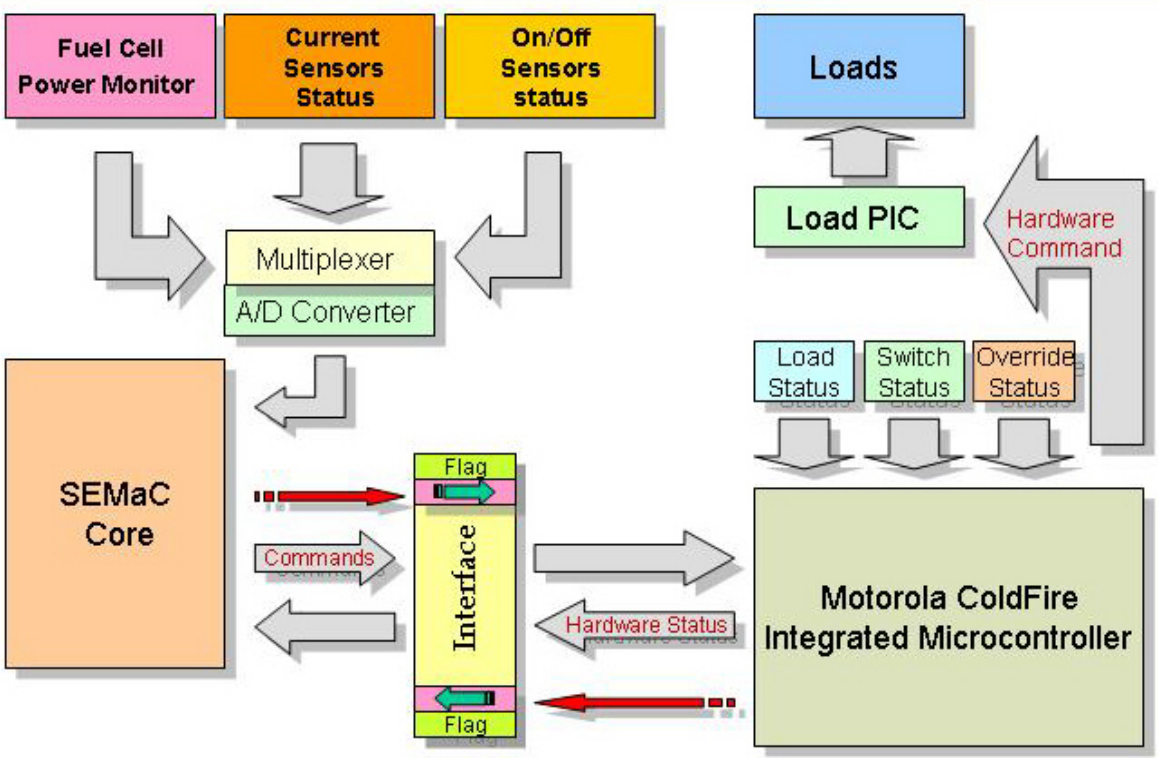
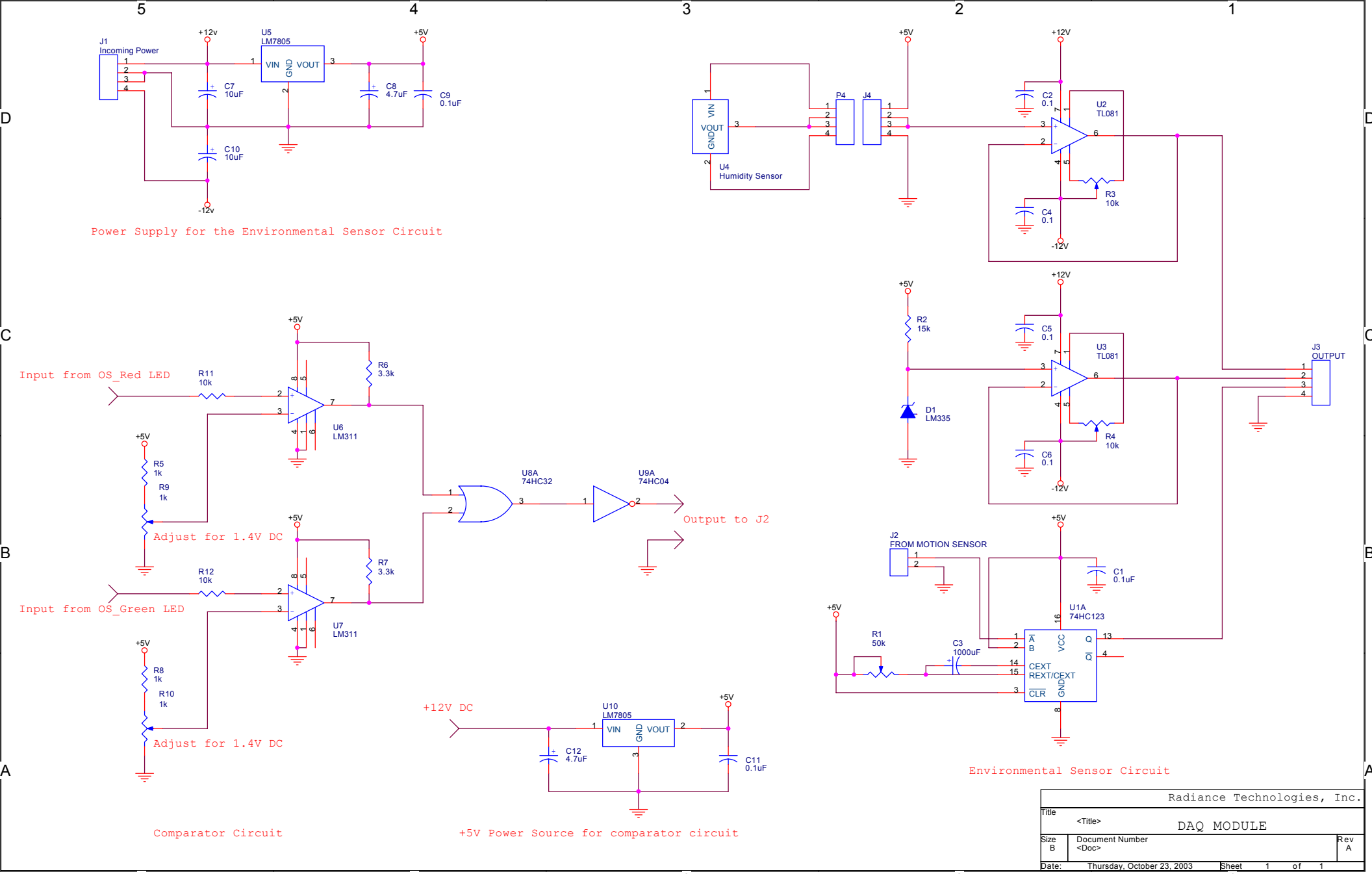
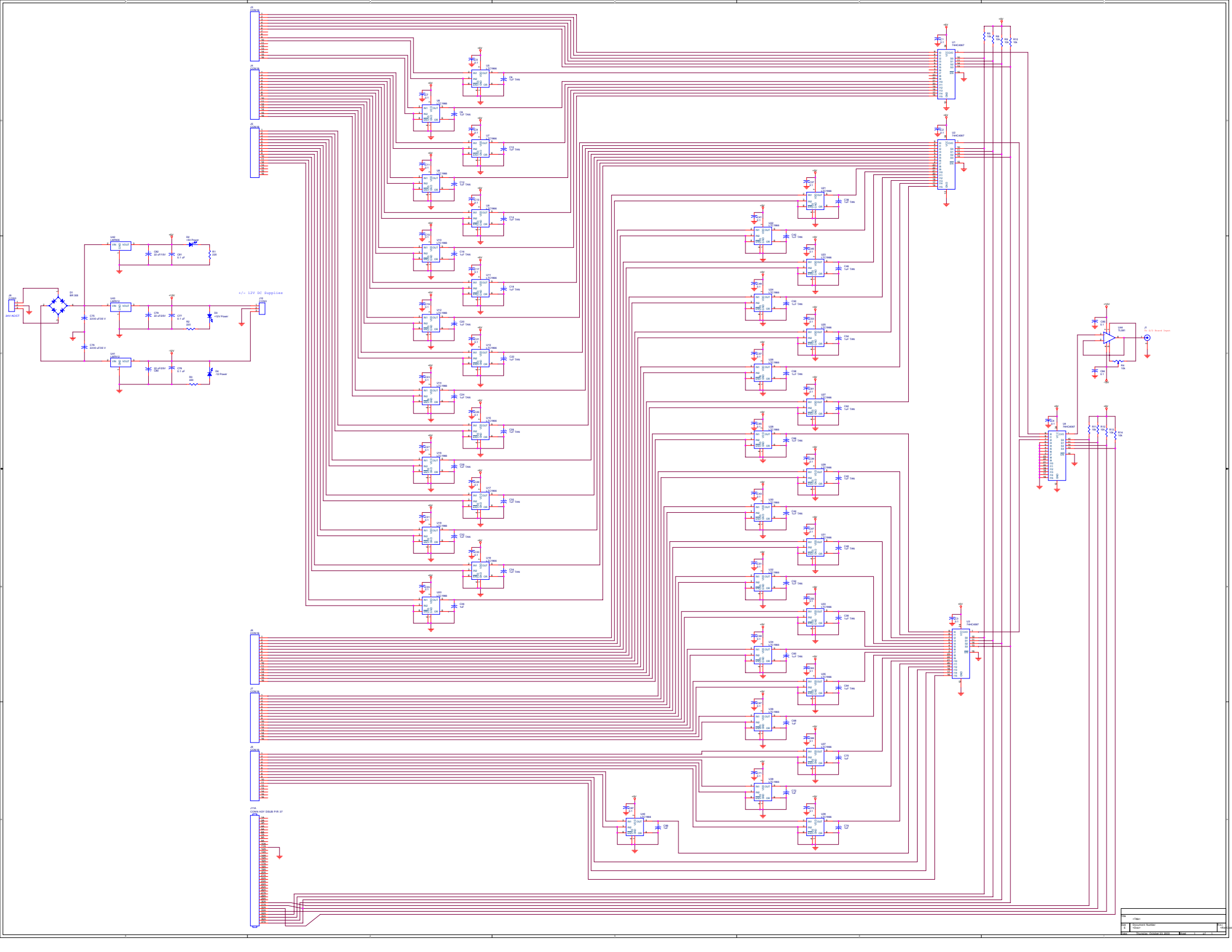


Fig. C1 SEMaC and SHC Communication Links


J1
Incoming Power
+12v
U5
LM7805
VIN
GND
VOUT
+5V
C7
10uF
C8
4.7uF
C9
0.1uF
C10
10uF
-12v
Power Supply for the Environmental Sensor Circuit
U4
Humidity Sensor
P4
J4
C2
0.1
U2
TL081
R3
10k
C4
0.1
R2
15k
D1
LM335
C5
0.1
U3
TL081
J3
OUTPUT
R4
10k
C6
0.1
J2
FROM MOTION SENSOR
C1
0.1uF
U1A
74HC123
R1
50k
C3
1000uF
Environmental Sensor Circuit
Input from OS_Red LED
R11
10k
R6
3.3k
U6
LM311
R5
1k
R9
1k
Adjust for 1.4V DC
U8A
74HC32
U9A
74HC04
Output to J2
Input from OS_Green LED
R12
10k
R7
3.3k
U7
LM311
R8
1k
R10
1k
Adjust for 1.4V DC
Comparator Circuit
+12V DC
U10
LM7805
C12
4.7uF
C11
0.1uF
+5V Power Source for comparator circuit
Radiance Technologies, Inc.
Title
<Title>
DAQ MODULE
Size
B
Document Number
<Doc>
Rev
A
Date:
Thursday, October 23, 2003
Sheet
1
of
1


+/- 12V DC Supplies
24V AC/CT
+12V Power
+12V Power
-12 Power
+5V
+12V
-12V
Title
<Title>
Size
B
Document Number
<Doc>
Rev
A
Date: Thursday, October 23, 2014
Sheet 1 of 1

APPENDIX D

Graphical User Interface

The Graphical User Interface

The graphical user interface (GUI) is written in C++ using Microsoft Visual Studio C++ version 6.0
and Microsoft Foundation Classes (MFC). The program is Windows based, it provides the user with a graphical illustration of the target home. The view of the home is a scale drawing of the home being controlled indicating the locations of all doors, windows, closets, etc. The full view of the home shows all controllable appliances including lights and electrical outlets. Each of these devices has its own associated icon. Occupancy status for each room is also indicated on this view. A mapping of the various icon types to the individual appliances in the home is shown to the right of the home view. A bar graph illustrating the instantaneous home power consumption is placed immediately below the home view and, immediately below that, a message box indicating communications from the SEMAC. The SEMAC contains the intelligent system that maintains total home power consumption at or below a predetermined maximum.

The icons representing the various appliances and devices within the home are color coded to indicate their status. An appliance that is off and drawing no power is indicated in blue. A device that is on, and drawing its maximum power is indicated in bright red. The GUI has the ability to illustrate appliances that have been dimmed or are not drawing their maximum power level with shades of red of varying intensity. Lighter shades of red indicate lower power consumption with deeper shades of red indicating progressively higher power draws. This permits the user, with a single glance to determine the state of every appliance within the home.

The GUI has been implemented to permit the user to examine the status of an individual room more closely if needed. The user can click the mouse on any room. This causes an expanded view of the selected room to replace the view of the home. All appliances are still visible. In addition, sensor data indicating the temperature and humidity in that particular room are added to the display. In the expanded state, the user has the ability to control the appliances in that room. By clicking the mouse on the icon for an individual appliance, the user can send a request to the SEMAC for that appliance state to be changed. Clicking the mouse on an appliance that is presently on, sends a request to the SEMAC for that device to be turned off. If the appliance is presently off, clicking on the icon sends a request for the device to be turned on. These actions using the GUI are only requests. The SEMAC does not have to honor them. Again the SEMAC is charged with keeping overall power consumption below prescribed maximums, so if a user request to turn on an appliance would cause power consumption to exceed this maximum, the SEMAC may ignore the request.

When an individual room view is maximized, all information usually visible on the main view is still present except the overall home view has been replaced by the individual room view. In addition to the bar graph indicating total power consumption, another bar graph indicating total power consumption for this particular room is also displayed.

As indicated above, the GUI is implemented using MFC. The classes defined in this program are illustrated in Figure One below. The relationship among the various methods defined in the program are illustrated in Figure Two, Figure Three and Figure Four below. There are three independent threads in the program. The first, illustrated in Figure Two handles drawing and maintenance of the GUI. The second, illustrated in Figure Three, handles communications with the SEMAC. The third, illustrated in Figure Four, handles appliance manipulation using the mouse. A complete listing of the program is attached.

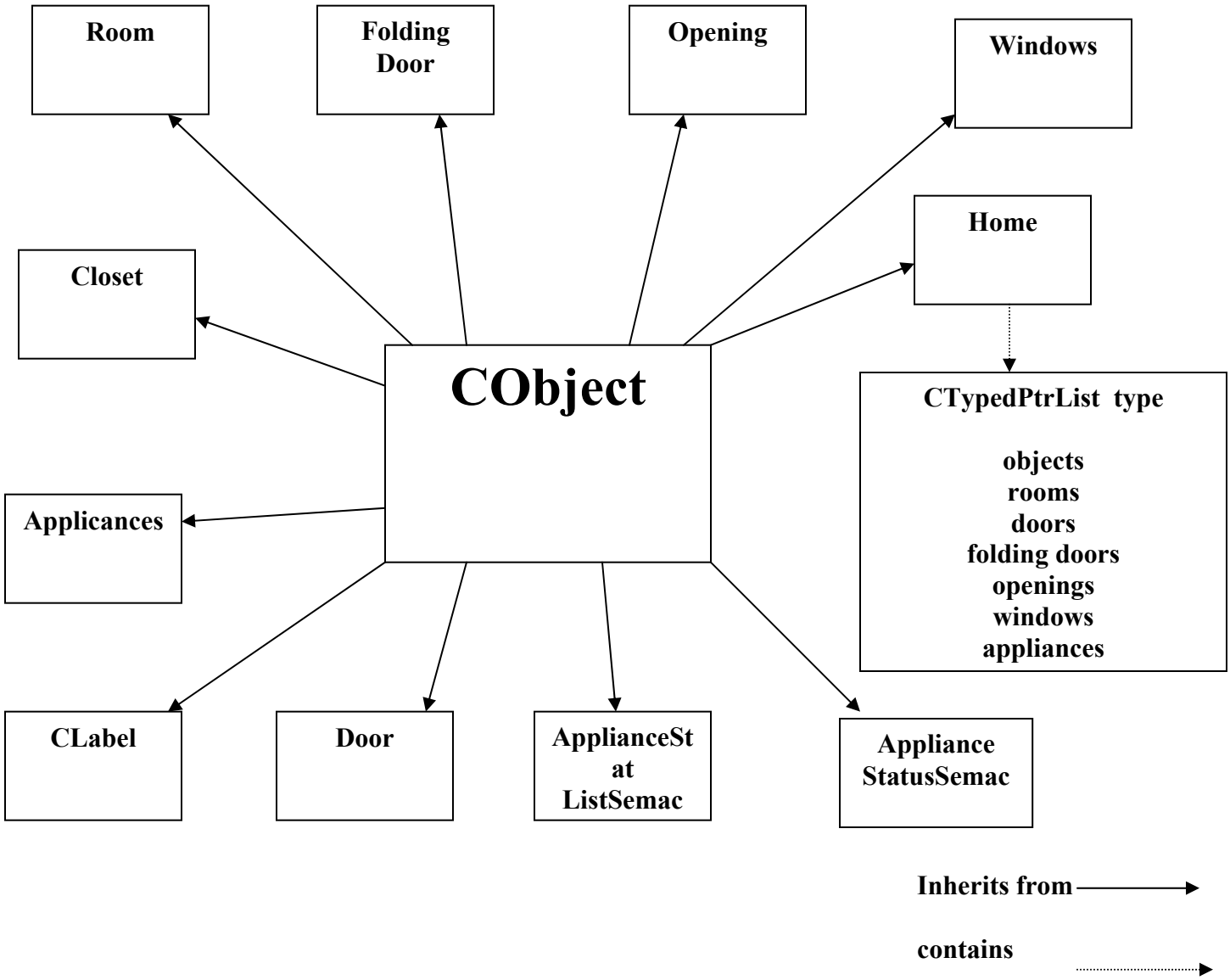

Room
Folding Door
Opening
Windows
Home
Closet
CObject
CTypedPtrList type
objects
rooms
doors
folding doors
openings
windows
appliances
Applicances
CLabel
Door
ApplianceStat ListSemac
Appliance StatusSemac
Inherits from
contains

Figure D1

Figure D2

invokes →

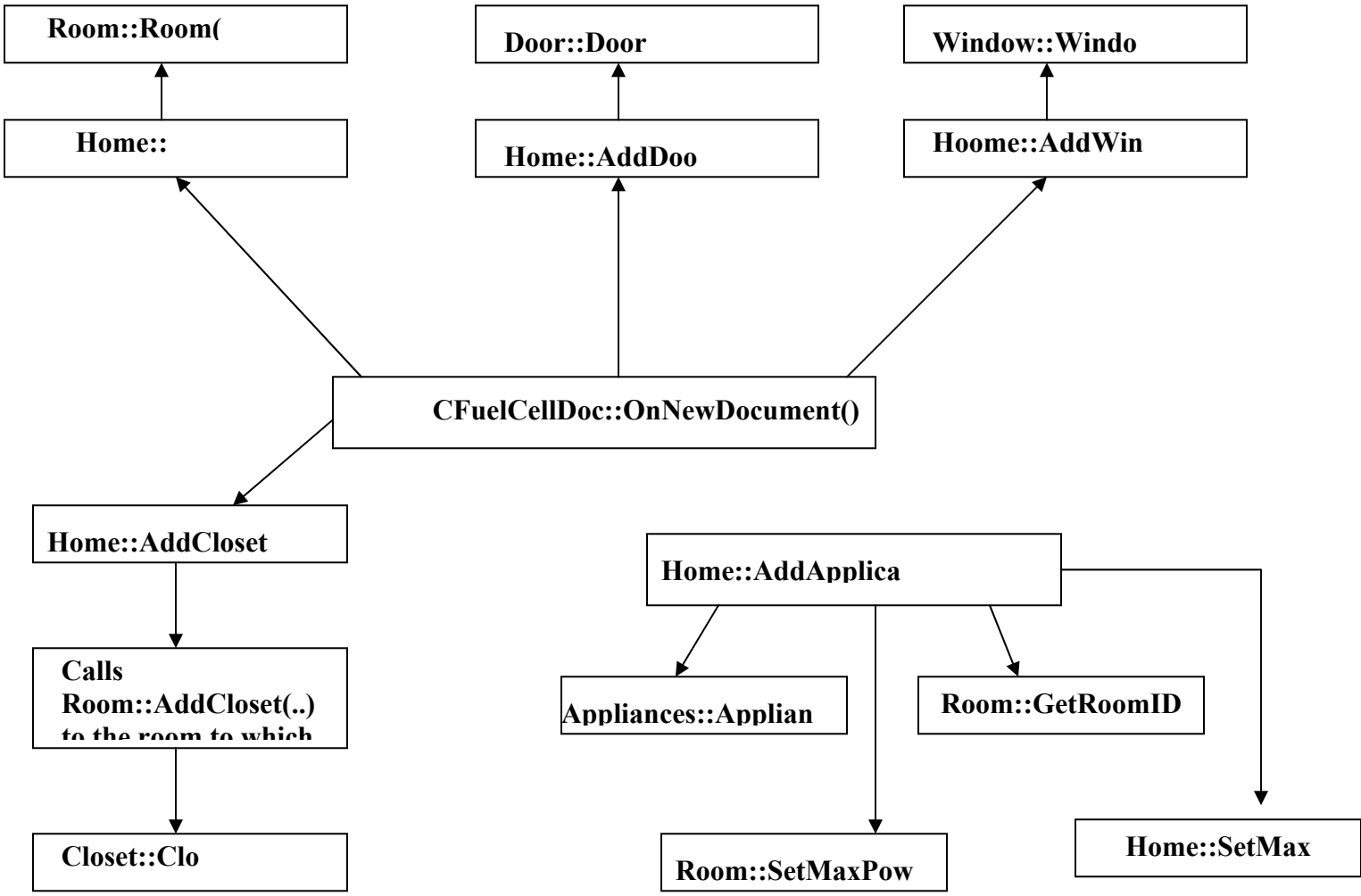


Figure D3
invokes →

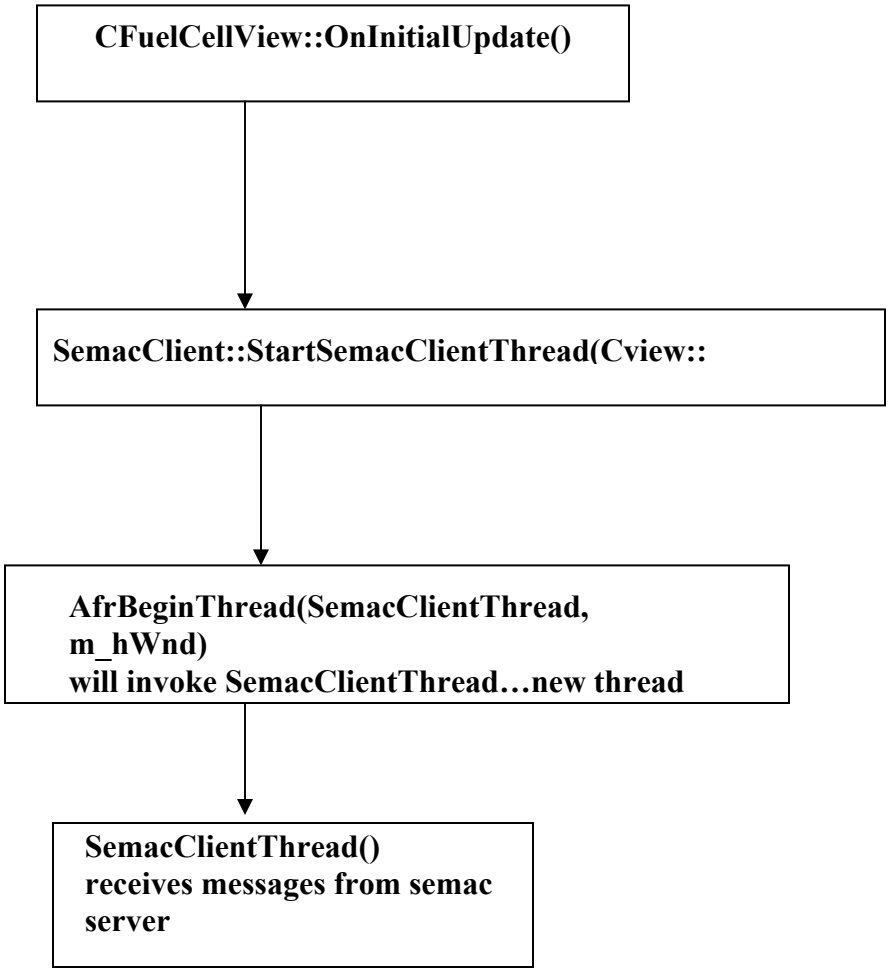
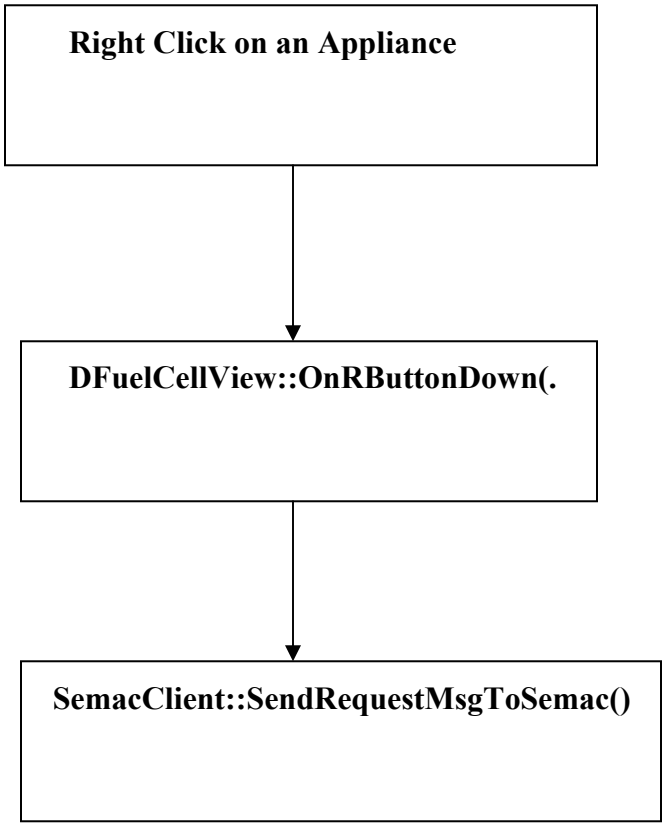


Figure Four

invokes →



```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
THE GUI INTERFACE PROGRAM.  THIS CONSISTS OF MULTIPLE C++
////  WRITTEN IN MICROSOFT VISUAL STUDIO C++ VERSION 6.0 USING
//// MICROSOFT FOUNDATION CLASSES.
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// FuelCellDoc.cpp : implementation of the CFuelCellDoc class
//

#include "stdafx.h"
#include "FuelCell.h"

#include "FuelCellDoc.h"
#include <iostream>
#include <fstream>
using namespace std;

/*Creating COLORREF objects that encapsulate RGB color objects
*/
COLORREF HOME_CLR(RGB(192,192,255)),
ROOM_CLR(RGB(255,255,255)),DOOR_CLR(RGB(255,255,255)),
FOLDING_DOOR_CLR(RGB(0,0,0)), APPLIANCE_ON_CLR(RGB(255,0,0)),
APPLIANCE_OFF_CLR(RGB(100,100,255)),PORCH_CLR(RGB(192,192,192)),
WINDOW_CLR(RGB(255,255,255)),OPENING_CLR(RGB(255,255,255));


#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;  //???????
#endif
/////////////////////////////////////////////////////////////////////////////
// CFuelCellDoc

IMPLEMENT_DYNCREATE(CFuelCellDoc, CDocument)

BEGIN_MESSAGE_MAP(CFuelCellDoc, CDocument)
        //{{AFX_MSG_MAP(CFuelCellDoc)
                // NOTE - the ClassWizard will add and remove mapping macros here.

//    DO NOT EDIT what you see in these blocks of generated code!
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
```

```
// CFuelCellDoc construction/destruction
CFuelCellDoc::CFuelCellDoc()
{
	// TODO: add one-time construction code here

}

CFuelCellDoc::~CFuelCellDoc()
{
	POSITION pos = homes.GetHeadPosition();
	while(pos != NULL)
		delete homes.GetNext(pos);
}
BOOL CFuelCellDoc::OnNewDocument()
{
	if (!CDocument::OnNewDocument())
		return FALSE;
	// TODO: add reinitialization code here
	// (SDI documents will reuse this document)
	homeCount = 0;
   homes.AddTail( new Home(950,650,HOME_CLR));
	homeCount++;
	Appliances::setOffColor(APPLIANCE_OFF_CLR);

   //cornners of room
	int array_r0[40] = {25,50,400,50,400,400,25,400};
	int array_r1[40] = {400,50,775,50,775,400,400,400};



	//closet
	int array_r2[40] = {25,50,100,50,100,125,25,125};

	// rectangles for appliances
		CRect rectFKB(200,210,250,260),rectFGR(570,205,620,255),
		rectW(35,235,85,295),rectDW(345,235,395,295),rectD(35,165,85,225),
		rectS(245,55,310,115),rectR(160,55,235,120),rectWH(350,65,395,110),
		rectHVAC(35,60,90,115);
//rectangles for lights
		CRect rectLGT1(210,150,240,180), rectLGT2(210,330,240,360),
			rectLGT3(580,150,610,180), rectLGT4(580,330,610,360),
			rectLGT5(25,405,45,425), rectLGT6(750,405,770,425);
//rectangles for outlets
		CRect rectHWO1(330,55,350,75), rectHWO2(200,375,220,395),
			rectHWO3(30,130,50,150), rectHWO4(375,150,395,170),
			rectHWO5(405,120,425,140), rectHWO6(750,150,770,170),
```

```
                    rectHWO7(480,55,500,75), rectHWO8(480,375,500,395),
                    rectHWO9(360,375,380,395),rectHWO10(405,230,425,250),
                    rectHWO11(700,375,720,395),rectHWO12(700,55,720,75),
                    rectHWO13(750,250,770,270);




        POSITION pos = homes.GetHeadPosition();
        Home * h = homes.GetNext(pos);

        // adding rooms
                h->addRoom(array_r0,4,ROOM_CLR,"KIT/EAT",CPoint(200,250));
                h->addRoom(array_r1,4,ROOM_CLR,"LIVING R", CPoint(560,250));
        //      h->addRoom(array_r2,4,ROOM_CLR,"AIR-RT",CPoint(30,70));


        //adding closets
        h->addCloset(1,array_r2,4,ROOM_CLR);



    // adding doors
                h->addDoor(1,-1,25,388,25,313,-90,DOOR_CLR);
                h->addDoor(2,-1,775,388,775,313,270,DOOR_CLR);
                h->addDoor(1,2,400,388,400,313,270,DOOR_CLR);
                h->addDoor(1,-1,100,60,100,105,90,DOOR_CLR);




        //adding foldingDoors



        //add opening
//      h->addOpening(13,7,475,345,510,345,90,OPENING_CLR);


    //adding windows
                h->addWindow(1,100,400,300,400,90,WINDOW_CLR);
                h->addWindow(2,475,400,675,400,90,WINDOW_CLR);
                h->addWindow(2,775,75,775,275,180,WINDOW_CLR);
```

```
h->addWindow(1,400,125,400,275,180,WINDOW_CLR);


        //adding appliances
        h->addApplainces(1,rectW,APPLIANCE_ON_CLR,10.f, Appliances::WASHER);
        h->addApplainces(1,rectD,APPLIANCE_ON_CLR,10.f, Appliances::DRIER);
        h->addApplainces(1,rectHVAC,APPLIANCE_ON_CLR,10.f, Appliances::HVAC);
        h->addApplainces(1,rectWH,APPLIANCE_ON_CLR,10.f,
Appliances::WATER_HEATER);
        h->addApplainces(1,rectS,APPLIANCE_ON_CLR,10.f, Appliances::STOVE);
        h->addApplainces(1,rectDW,APPLIANCE_ON_CLR,10.f,
Appliances::DISH_WASHER);
        h->addApplainces(1,rectR,APPLIANCE_ON_CLR,10.f, Appliances::REFRIGERATOR);
        //------------------fans----------------------------------------//
        h->addApplainces(1,rectFKB,APPLIANCE_ON_CLR,10.f, Appliances::FAN);
        h->addApplainces(2,rectFGR,APPLIANCE_ON_CLR,10.f, Appliances::FAN);

        //------------------bulbs----------------------------------------//

        h->addApplainces(1,rectLGT1,APPLIANCE_ON_CLR,10.f, Appliances::LIGHT);
        h->addApplainces(1,rectLGT2,APPLIANCE_ON_CLR,10.f, Appliances::LIGHT);
        h->addApplainces(2,rectLGT3,APPLIANCE_ON_CLR,10.f, Appliances::LIGHT);
        h->addApplainces(2,rectLGT4,APPLIANCE_ON_CLR,10.f, Appliances::LIGHT);
//      h->addApplainces(1,rectLGT5,APPLIANCE_ON_CLR,10.f, Appliances::LIGHT);
        h->addApplainces(2,rectLGT6,APPLIANCE_ON_CLR,10.f, Appliances::LIGHT);
//      ----------outlets-------------------------------------------------------//
        h-
>addApplainces(1,rectHWO1,APPLIANCE_ON_CLR,10.f,Appliances::HIGH_WATTAGE_OU
TLET);
        h->addApplainces(1,rectHWO2,APPLIANCE_ON_CLR,10.f,
Appliances::HIGH_WATTAGE_OUTLET);
        h->addApplainces(1,rectHWO3,APPLIANCE_ON_CLR,10.f,
Appliances::HIGH_WATTAGE_OUTLET);
        h->addApplainces(1,rectHWO4,APPLIANCE_ON_CLR,10.f,
Appliances::HIGH_WATTAGE_OUTLET);
        h->addApplainces(2,rectHWO5,APPLIANCE_ON_CLR,10.f,
Appliances::HIGH_WATTAGE_OUTLET);
        h->addApplainces(2,rectHWO6,APPLIANCE_ON_CLR,10.f,
Appliances::HIGH_WATTAGE_OUTLET);
        h->addApplainces(2,rectHWO7,APPLIANCE_ON_CLR,10.f,
Appliances::HIGH_WATTAGE_OUTLET);
        h->addApplainces(2,rectHWO8,APPLIANCE_ON_CLR,10.f,
Appliances::HIGH_WATTAGE_OUTLET);
        h->addApplainces(1,rectHWO9,APPLIANCE_ON_CLR,10.f,
Appliances::HIGH_WATTAGE_OUTLET);
```

```
		h->addApplainces(2,rectHWO10,APPLIANCE_ON_CLR,10.f,
Appliances::HIGH_WATTAGE_OUTLET);
		h->addApplainces(2,rectHWO11,APPLIANCE_ON_CLR,10.f,
Appliances::HIGH_WATTAGE_OUTLET);
		h->addApplainces(2,rectHWO12,APPLIANCE_ON_CLR,10.f,
Appliances::HIGH_WATTAGE_OUTLET);
		h->addApplainces(2,rectHWO13,APPLIANCE_ON_CLR,10.f,
Appliances::HIGH_WATTAGE_OUTLET);
		//---------------------------ExhaustFan---------------------------------------//
		//----------------------------------------------------------------------------//
		///READING DATA FROM FILE  HAPPENS ONLY ONCE
	arrayPointer = 0;
	ifstream input("AppliancesDoc.txt");
		if(input){

			int homeIdFromFile ;
			input >> homeIdFromFile;
			lastInUse =0;
			while(input>> applIDBuff[lastInUse] >> applStatusBuff[lastInUse] >>
applPowerBuff[lastInUse])
				lastInUse++;
		}
		input.close();
		return TRUE;
}

BOOL CFuelCellDoc::OnOpenDocument(LPCTSTR lpszPathName){
		if (!CDocument::OnOpenDocument(lpszPathName))
			return FALSE;
		return TRUE;

}



void CFuelCellDoc::dataFromFile( CDC* dc,CRect *rect){
		if (arrayPointer >= lastInUse )
			arrayPointer = 0;
		POSITION pos = homes.GetHeadPosition();
		Home * home;
		while(pos != NULL){
			home = homes.GetNext(pos);
			home->setCurPower(applIDBuff,applStatusBuff,applPowerBuff, arrayPointer);
		}
		SetModifiedFlag(TRUE);
		//UpdateAllViews(NULL);     commented out by shahrukh to remove the flicker
```

```
		home->setDC(dc,rect);		//added in process of removing flicker
		home->draw (dc);		//added in process of removing flicker

}


/////////////////////////////////////////////////////////////////////////////
// CFuelCellDoc serialization

void CFuelCellDoc::Serialize(CArchive& ar)
{
		if (ar.IsStoring())
		{
			// TODO: add storing code here
			ar << homeCount;
			POSITION pos = homes.GetHeadPosition();
			while(pos != NULL)
				ar << homes.GetNext(pos);
		}
		else
		{
			// TODO: add loading code here
			ar >> homeCount;
			Home * h;
			for (int i = 0; i < homeCount ; i++){
				ar >> h;
				homes.AddTail(h);
			}
		}
}

/////////////////////////////////////////////////////////////////////////////
// CFuelCellDoc diagnostics

#ifdef _DEBUG
void CFuelCellDoc::AssertValid() const
{
		CDocument::AssertValid();
}

void CFuelCellDoc::Dump(CDumpContext& dc) const
{
		CDocument::Dump(dc);
}
#endif //_DEBUG
```

```
////////////////////////////////////////////////////////////////////////////
// CFuelCellDoc commands



#include "stdafx.h"
#include "Appliances.h"
#include <Math.h>

int Appliances::count;
COLORREF Appliances::appliancesClrOff;

//Definition of class Appliances
IMPLEMENT_SERIAL (Appliances, CObject, 1);
Appliances::Appliances(){}
Appliances::~Appliances(){
	--count;
}

Appliances::Appliances(int theHomeId,int theAttachedToId, CRect theRect,COLORREF
theAppliancesClr,float maxPower,enum Appliances::Shape theShape){
	appliancesId = ++count;
	homeId = theHomeId;
	attachedToId = theAttachedToId;
	appliancesClr = theAppliancesClr;
	onOff = 0;
	rect.operator =(theRect);
	setColor(appliancesClr);
	setMaxPower(maxPower);
	shape = theShape;
	powerConsumed = 0.f;
	curPower = 0.f;
	changeInPower = 0.f;
	//appliancesClrOff = RGB(255,255,255);

}
Appliances::Appliances(const Appliances & a){
	appliancesId = a.appliancesId;
	homeId = a.homeId;
	attachedToId = a.attachedToId;
	appliancesClr = a.appliancesClr;
	onOff = a.onOff;
	rect.operator =(a.rect);
	appliancesClr = a.appliancesClr;
	maxPower = a.maxPower;
	shape = a.shape;
```

```
	powerConsumed = a.powerConsumed;
	curPower = a.curPower;
	changeInPower = a.changeInPower;
}
/*Appliances * Appliances::operator =(Appliances *a){
	return a;
}*/

Appliances  Appliances::operator =(Appliances a){
	return a;
}

float Appliances::getCurPower(){
	return curPower;
}
void Appliances::draw(CDC* dc){
	//to get the shading effect when light is on the following formula
	//can be used 255 +(getR-255)(curPower/maxPower) for all RGB
	float powerRatio = curPower/maxPower;
	CBrush * oldBrush,
		cbrush1(appliancesClrOff),
		cbrush2(RGB((255 + (int)((GetRValue(appliancesClr)-255)*powerRatio)),
			(255 + (int)((GetGValue(appliancesClr)-255)*powerRatio)),
				(255 + (int)((GetBValue(appliancesClr)-
255)*powerRatio))));
	onOff == 0 ?  oldBrush  = dc->SelectObject(&cbrush1): oldBrush = dc-
>SelectObject(&cbrush2);
	CPen pen(PS_SOLID,1,RGB(0,0,0));
	CPen * oldPen =  dc->SelectObject(&pen);
	dc->SetBkMode(TRANSPARENT );
	switch(shape){
	case WASHER : dc->Rectangle(rect); dc->DrawText(_T("W"),-
1,&rect,(DT_SINGLELINE|DT_CENTER|DT_VCENTER)); break;
	case DRIER : dc->Rectangle(rect); dc->DrawText(_T("D"),-
1,&rect,(DT_SINGLELINE|DT_CENTER|DT_VCENTER));break;
	case HVAC : dc->Rectangle(rect); dc->DrawText(_T("AC"),-
1,&rect,(DT_SINGLELINE|DT_CENTER|DT_VCENTER));break;
	case WATER_HEATER : dc->Ellipse(rect); dc->DrawText(_T("H"),-
1,&rect,(DT_SINGLELINE|DT_CENTER|DT_VCENTER)); break;
	case FAN : drawFan(dc); break;
	case LIGHT : drawLight(dc); break;
	case LOW_WATTAGE_OUTLET : drawLWOutlet(dc); break;
	case HIGH_WATTAGE_OUTLET : drawHWOutlet(dc); break;
	case EXHAUST_FAN : drawExhaustFan(dc); break;
	case OVEN : dc->Rectangle(rect); dc->DrawText(_T("O"),-
1,&rect,(DT_SINGLELINE|DT_CENTER|DT_VCENTER)); break;
```

```
        case STOVE : dc->Rectangle(rect); dc->DrawText(_T("S"),-
1,&rect,(DT_SINGLELINE|DT_CENTER|DT_VCENTER)); break;
        case DISH_WASHER : dc->Rectangle(rect); dc->DrawText(_T("DW"),-
1,&rect,(DT_SINGLELINE|DT_CENTER|DT_VCENTER)); break;
        case REFRIGERATOR : dc->Rectangle(rect); dc->DrawText(_T("R"),-
1,&rect,(DT_SINGLELINE|DT_CENTER|DT_VCENTER)); break;
        }
        dc->SetBkMode(OPAQUE );
   dc->SelectObject(oldBrush);
        dc->SelectObject(oldPen);
}
//this function is invoked when appliances are clicked on the screan
int Appliances::setOnOff(CPoint point){
        if (rect.PtInRect(point)){
                onOff = 1-onOff;
                if (onOff){
                        changeInPower = maxPower - curPower;
                        curPower = maxPower;
                }
                else{
                        changeInPower = 0.f - curPower;
                   curPower = 0.f;
                }
                return 1;
        }
        return 0;
}
//this function is invoked when data is read from the file
int Appliances::setOnOff(int status, float thePower){
        changeInPower = thePower - curPower;
        curPower = thePower;
        if(status == onOff) {
                if (onOff)
                        return 1;
                else
                        return 0;
        }
        else {
                onOff = 1-onOff;
                return 1;
        }
}

void Appliances::Serialize (CArchive& ar)
{
   CObject::Serialize (ar);
```

```
        if (ar.IsStoring ()){
                ar << count;
                ar << appliancesClrOff;
                ar << homeId;
                ar << appliancesId;
           ar << attachedToId;
                ar << appliancesClr;
                ar << rect;
                ar << maxPower;
                ar << powerConsumed;
                ar << shape;
                ar << onOff;
                ar << curPower;
                ar << changeInPower;
        }
   else {// Loading, not storing
                ar >> count;
                ar >> appliancesClrOff;
                ar >> homeId;
                ar >> appliancesId;
           ar >> attachedToId;
                ar >> appliancesClr;
                ar >> rect;
                ar >> maxPower;
                ar >> powerConsumed;
                int theShape;
                ar >> theShape;
                shape = (Shape)theShape;
                ar >> onOff;
                ar >> curPower;
                ar >> changeInPower;
        }
}

float Appliances::getPowerChange()
{
        return changeInPower;
}

void Appliances::drawFan(CDC *dc)
{
        CPoint center = rect.CenterPoint();
        int reducedW = rect.Width()/6, reducedH = rect.Height()/6;
        int reducedW1 = rect.Width()/12, reducedH1 = rect.Height()/12;
        CRect centerRect((center.x - reducedW),(center.y - reducedH),(center.x +
reducedW),(center.y + reducedH));
```

```
        CRect wing1(rect.left,(center.y - reducedH1),center.x,(center.y + reducedH1)),
                wing2(center.x,(center.y - reducedH1),rect.right,(center.y + reducedH1)),
                wing3((center.x - reducedW1),rect.top,(center.x + reducedW1),center.y),
                wing4((center.x - reducedW1),center.y,(center.x + reducedW1),rect.bottom);
        dc->Ellipse(centerRect);
        dc->Ellipse(wing1);
        dc->Ellipse(wing2);
        dc->Ellipse(wing3);
        dc->Ellipse(wing4);
}
void Appliances::drawExhaustFan(CDC *dc)
{
        CPoint center = rect.CenterPoint();
        dc->Ellipse(rect);
        int reducedW = rect.Width()/6, reducedH = rect.Height()/6;
        int reducedW1 = rect.Width()/12, reducedH1 = rect.Height()/12;
        CRect centerRect((center.x - reducedW),(center.y - reducedH),(center.x +
reducedW),(center.y + reducedH));
        CRect wing1(rect.left,(center.y - reducedH1),center.x,(center.y + reducedH1)),
                wing2(center.x,(center.y - reducedH1),rect.right,(center.y + reducedH1)),
                wing3((center.x - reducedW1),rect.top,(center.x + reducedW1),center.y),
                wing4((center.x - reducedW1),center.y,(center.x + reducedW1),rect.bottom);
        dc->Ellipse(centerRect);
        dc->Ellipse(wing1);
        dc->Ellipse(wing2);
        dc->Ellipse(wing3);
        dc->Ellipse(wing4);
}

void Appliances::drawLight(CDC *dc)
{
        int reducedW = rect.Width()/7, reducedH = rect.Height()/7 ,
                yPosOfHLine = rect.top + rect.Height()/2,
                xPosOfVLine = rect.left + rect.Width()/2;
        CRect smallRect(rect.left+reducedW, rect.top+reducedH,rect.right-
reducedW,rect.bottom-reducedH);
        dc->Ellipse(smallRect);
        dc->MoveTo(rect.left, yPosOfHLine);
        dc->LineTo(rect.right,yPosOfHLine);
        dc->MoveTo(xPosOfVLine, rect.top);
        dc->LineTo(xPosOfVLine, rect.bottom);
}

void Appliances::drawLWOutlet(CDC *dc)
{
        int reducedW = rect.Width()/7, reducedH = rect.Height()/7 ,
```

```
                yPosOfHLine = rect.top + rect.Height()/2;
        CRect smallRect(rect.left+reducedW, rect.top+reducedH,rect.right-
reducedW,rect.bottom-reducedH);
        dc->Ellipse(smallRect);
        dc->MoveTo(rect.left, yPosOfHLine);
        dc->LineTo(rect.right,yPosOfHLine);
}

void Appliances::drawHWOutlet(CDC *dc)
{
        int reducedW = rect.Width()/7, reducedH = rect.Height()/7 ,
                yPosOfHLine1 = rect.top + rect.Height()/3,
                yPosOfHLine2 = rect.bottom - rect.Height()/3;
        CRect smallRect(rect.left+reducedW, rect.top+reducedH,rect.right-
reducedW,rect.bottom-reducedH);
        dc->Ellipse(smallRect);
        dc->MoveTo(rect.left, yPosOfHLine1);
        dc->LineTo(rect.right,yPosOfHLine1);
        dc->MoveTo(rect.left, yPosOfHLine2);
        dc->LineTo(rect.right,yPosOfHLine2);
}

void Appliances::drawList(CDC *dc, int x, int y)
{
        CBrush * oldBrush,
                cbrush(RGB(255,255,255));
        oldBrush  = dc->SelectObject(&cbrush);
        CPen pen(PS_SOLID,1,RGB(0,0,0));
        CPen * oldPen =  dc->SelectObject(&pen);
        dc->SetBkMode(TRANSPARENT );
        for (int shape = 0 ; shape <= 12 ; shape++){
                CRect rect(x,y,x+30,y+30);
                CRect rectText(x+32,y,x+150,y+30);y+=34;
                setRect(rect);
                switch(shape){
                case WASHER :
                        dc->Rectangle(rect);
                        dc->DrawText(_T("W"),-
1,&rect,(DT_SINGLELINE|DT_CENTER|DT_VCENTER));
                        dc->DrawText(_T("Washer"),-
1,&rectText,(DT_SINGLELINE|DT_VCENTER));
                        break;
                case DRIER :
                        dc->Rectangle(rect);
                        dc->DrawText(_T("D"),-
1,&rect,(DT_SINGLELINE|DT_CENTER|DT_VCENTER));
```

```
                        dc->DrawText(_T("Drier"),-
1,&rectText,(DT_SINGLELINE|DT_VCENTER));
                        break;
                case HVAC :
                        dc->Rectangle(rect);
                        dc->DrawText(_T("AC"),-
1,&rect,(DT_SINGLELINE|DT_CENTER|DT_VCENTER));
                        dc->DrawText(_T("HVAC"),-
1,&rectText,(DT_SINGLELINE|DT_VCENTER));
                        break;
                case WATER_HEATER :
                        dc->Ellipse(rect);
                        dc->DrawText(_T("H"),-
1,&rect,(DT_SINGLELINE|DT_CENTER|DT_VCENTER));
                        dc->DrawText(_T("Water Heater"),-
1,&rectText,(DT_SINGLELINE|DT_VCENTER));
                        break;
                case FAN :
                        drawFan(dc);
                        dc->DrawText(_T("Fan"),-
1,&rectText,(DT_SINGLELINE|DT_VCENTER));
                        break;
                case LIGHT :
                        drawLight(dc);
                        dc->DrawText(_T("Light"),-
1,&rectText,(DT_SINGLELINE|DT_VCENTER));
                        break;
                case LOW_WATTAGE_OUTLET :
                        drawLWOutlet(dc);
                        dc->DrawText(_T("Low Wattg o/l"),-
1,&rectText,(DT_SINGLELINE|DT_VCENTER));
                        break;
                case HIGH_WATTAGE_OUTLET :
                        drawHWOutlet(dc);
                        dc->DrawText(_T("High Wattg o/l"),-
1,&rectText,(DT_SINGLELINE|DT_VCENTER));
                        break;
                case EXHAUST_FAN :
                       drawExhaustFan(dc);
                       dc->DrawText(_T("Exhaust Fan"),-
1,&rectText,(DT_SINGLELINE|DT_VCENTER));
                       break;
                case OVEN :
                        dc->Rectangle(rect);
                        dc->DrawText(_T("O"),-
1,&rect,(DT_SINGLELINE|DT_CENTER|DT_VCENTER));
```

```
                        dc->DrawText(_T("Oven"),-
1,&rectText,(DT_SINGLELINE|DT_VCENTER));
                        break;
                case STOVE :
                        dc->Rectangle(rect);
                        dc->DrawText(_T("S"),-
1,&rect,(DT_SINGLELINE|DT_CENTER|DT_VCENTER));
                        dc->DrawText(_T("Stove"),-
1,&rectText,(DT_SINGLELINE|DT_VCENTER));
                        break;
                case DISH_WASHER :
                        dc->Rectangle(rect);
                        dc->DrawText(_T("DW"),-
1,&rect,(DT_SINGLELINE|DT_CENTER|DT_VCENTER));
                        dc->DrawText(_T("Dish Washer"),-
1,&rectText,(DT_SINGLELINE|DT_VCENTER));
                        break;
                case REFRIGERATOR :
                        dc->Rectangle(rect);
                        dc->DrawText(_T("R"),-
1,&rect,(DT_SINGLELINE|DT_CENTER|DT_VCENTER));
                        dc->DrawText(_T("Refrigerator"),-
1,&rectText,(DT_SINGLELINE|DT_VCENTER));
                        break;
                }
        }
        dc->SetBkMode(OPAQUE );
   dc->SelectObject(oldBrush);
        dc->SelectObject(oldPen);
}

void Appliances::setRect(CRect theRect)
{
        rect.operator =(theRect);
}

#include "stdafx.h"
#include "Closet.h"
#include <iostream>
using namespace std;
int Closet::count;
//Definition of class Closet
IMPLEMENT_SERIAL (Closet, CObject, 1);
Closet::Closet(){}
Closet::~Closet(){
        --count;
```

```
        delete [] corners;
}
Closet::Closet(int theHomeId, int theArray[],int theNumberOfCorners, COLORREF theClr){
        homeId = theHomeId;
        closetId = ++count;
        numberOfCorners = theNumberOfCorners;
        initializeCorners(theArray);
        closetClr = theClr;
}
Closet::Closet(const Closet & c){
        numberOfCorners = c.numberOfCorners;
        homeId = c.homeId;
        int index = 0;
   while(index < numberOfCorners){
           corners[index] = c.corners[index];index++;
        }
        closetId = c.closetId;
        closetClr = c.closetClr;
}
Closet * Closet::operator =(Closet *c){
        return c;
}
Closet  Closet::operator =(Closet c){
        return c;
}
void Closet::draw(CDC* dc){
        CPen pen(PS_SOLID,5,RGB(0,0,0)),* oldPen;
        //CBrush cbrush(RGB(248,248,248));
        //CBrush cbrush(RGB(200,200,255));
        //CBrush cbrush(RGB(255,253,235));
        CBrush cbrush(closetClr), * oldBrush;
        oldPen  = dc->SelectObject(&pen);
        oldBrush = dc->SelectObject(&cbrush);
        dc->Polygon(corners,numberOfCorners);
        dc->SelectObject(oldBrush);
        dc->SelectObject(oldPen);
}


void Closet::initializeCorners(int theArray[]){
        try {

                corners = new CPoint[numberOfCorners];
                int counter = 0;
                while( counter < numberOfCorners){
```

```
                        corners[counter] =
CPoint(theArray[counter<<1],theArray[(counter<<1)+1]);
                        counter++;
                }
        }
        catch(bad_alloc bac){
                cout <<"\nFailure to allocate";
        }
}


void Closet::Serialize (CArchive& ar)
{
   CObject::Serialize (ar);
        if (ar.IsStoring ()){
                ar << count;
                ar << homeId;
                ar << numberOfCorners;
                for (int i = 0 ; i < numberOfCorners ; i++)
                        ar << corners[i];
           ar << closetId;
           ar << closetClr;
        }
   else {// Loading, not storing
                ar >> count;
                ar >> homeId;
                ar >> numberOfCorners;
                try {
                   corners = new CPoint[numberOfCorners];
                        for (int i = 0 ; i < numberOfCorners ; i++)
                                ar >> corners[i];
                }
                catch(bad_alloc bac){
                        cout <<"\nFailure to allocate";
                }
                ar >> closetId;
           ar >> closetClr;
        }
}

#include "stdafx.h"
#include "Door.h"
#include <Math.h>

int Door::count;
//Definition of class Door
```

```
IMPLEMENT_SERIAL (Door, CObject, 1);
Door::Door(){}
Door::~Door(){
        --count;
}

Door::Door(int theHomeId, int theAttachedToId1,int theAttachedToId2, int X1,int Y1,int X2, int
Y2, int theRotation,COLORREF theDoorClr){
        doorId = ++count;
        homeId = theHomeId;
        attachedToId1 = theAttachedToId1;
        attachedToId2 = theAttachedToId2;
        hinge.x = X1;
        hinge.y = Y1;
        lock.x = X2;
        lock.y = Y2;
        rotation = theRotation;
        width = distBtPoints();
        doorClr = theDoorClr;
}
Door::Door(const Door & d){
        doorId = d.doorId;
        homeId = d.homeId;
        attachedToId1 = d.attachedToId2;
        attachedToId1 = d.attachedToId2;
        hinge.x = d.hinge.x ;
        hinge.y = d.hinge.y;
        lock.x = d.lock.x;
        lock.y = d.lock.y;
        rotation = d.rotation;
   width = d.width;
        doorClr = d.doorClr;
}
/*Door * Door::operator =(Door *d){
        return d;
}*/

Door  Door::operator =(Door d){
        return d;
}

void Door::draw(CDC* dc){
        CPen pen(PS_SOLID,1,RGB(0,0,0)), * oldPen;
        //CBrush cbrush(RGB(100,100,255));
        //CBrush cbrush(RGB(148,148,148));
        //CBrush cbrush(RGB(225,223,205));
```

```
        CBrush cbrush(doorClr), * oldBrush;

   oldPen = dc->SelectObject(&pen);
        oldBrush = dc->SelectObject(&cbrush);
        CPoint cp((int)(hinge.x+width*sin(abs(rotation)*3.14/180)),(int)(hinge.y + width *
cos(abs(rotation)*3.14/180)));
        if(rotation >= 0)
           dc->Pie(hinge.x-width,hinge.y-
width,hinge.x+width,hinge.y+width,lock.x,lock.y,cp.x,cp.y);
        else
                dc->Pie(hinge.x-width,hinge.y-
width,hinge.x+width,hinge.y+width,cp.x,cp.y,lock.x,lock.y);
        dc->SelectObject(oldBrush);
        dc->SelectObject(oldPen);
}

int Door::distBtPoints(){

        return (int)sqrt(pow((hinge.x-lock.x),2)+pow((hinge.y-lock.y),2));
}

void Door::Serialize (CArchive& ar)
{
   CObject::Serialize (ar);
        if (ar.IsStoring ()){
                ar << count;
                ar << homeId;
                ar << doorId;
           ar << attachedToId1;
                ar << attachedToId2;
           ar << hinge;
           ar << lock;
                ar << doorClr;
                ar << rotation;
                ar << width;
        }
   else {// Loading, not storing
                ar >> count;
                ar >> homeId;
                ar >> doorId;
           ar >> attachedToId1;
                ar >> attachedToId2;
           ar >> hinge;
           ar >> lock;
                ar >> doorClr;
                ar >> rotation;
```

```
                ar >> width;
        }
}

#include "stdafx.h"
#include "FoldingDoor.h"
#include <Math.h>

int FoldingDoor::count;
//Definition of class FoldingDoor
IMPLEMENT_SERIAL (FoldingDoor, CObject, 1);
FoldingDoor::FoldingDoor(){}
FoldingDoor::~FoldingDoor(){
        --count;
}

FoldingDoor::FoldingDoor(int theHomeId,int theAttachedToId1,int theAttachedToId2, int X1,int
Y1,int X2, int Y2,
                                int theRotation,int theSlidingAngleFromVertical, COLORREF
theFoldingDoorClr){
        homeId = theHomeId;
        foldingDoorId = ++count;
        attachedToId1 = theAttachedToId1;
        attachedToId2 = theAttachedToId2;
        hinge.x = X1;
        hinge.y = Y1;
        lock.x = X2;
        lock.y = Y2;
        rotation = theRotation;
        s_a_f_v = theSlidingAngleFromVertical;
        width = distBtPoints();
        foldingDoorClr = theFoldingDoorClr;
}
FoldingDoor::FoldingDoor(const FoldingDoor & fd){
        foldingDoorId = fd.foldingDoorId;
        homeId = fd.homeId;
        attachedToId1 = fd.attachedToId2;
        attachedToId1 = fd.attachedToId2;
        hinge.x = fd.hinge.x ;
        hinge.y = fd.hinge.y;
        lock.x = fd.lock.x;
        lock.y = fd.lock.y;
        rotation = fd.rotation;
        s_a_f_v = fd.s_a_f_v;
   width = fd.width;
        foldingDoorClr = fd.foldingDoorClr;
```

```
}
/*FoldingDoor * FoldingDoor::operator =(FoldingDoor *fd){
	return fd;
}*/

FoldingDoor  FoldingDoor::operator =(FoldingDoor fd){
	return fd;
}

void FoldingDoor::draw(CDC* dc){
	CPen pen1(PS_SOLID,5,RGB(255,255,255)), *oldPen;
   oldPen = dc->SelectObject(&pen1);
	dc->MoveTo(lock);
	dc->LineTo(hinge);
	CPen pen(PS_SOLID,2,foldingDoorClr);
	dc->SelectObject(&pen);
	CPoint cp((int)(hinge.x+(width/2.)*sin(abs(rotation)*3.14/180)),(int)(hinge.y + (width/2.)
* cos(abs(rotation)*3.14/180)));
	dc->LineTo(cp);
	if(hinge.x <= lock.x && hinge.y <= lock.y)
		dc-
>LineTo((int)(hinge.x+width*cos(abs(rotation)*3.14/180)*sin(s_a_f_v*3.14/180)),(int)(hinge.y +
width * cos(abs(rotation)*3.14/180)*cos(s_a_f_v*3.14/180)));
	else
		dc->LineTo((int)(hinge.x-
width*cos(abs(rotation)*3.14/180)*sin(s_a_f_v*3.14/180)),(int)(hinge.y - width *
cos(abs(rotation)*3.14/180)*cos(s_a_f_v*3.14/180)));
	dc->SelectObject(oldPen);
}

int FoldingDoor::distBtPoints(){

	return (int)sqrt(pow((hinge.x-lock.x),2)+pow((hinge.y-lock.y),2));
}

void FoldingDoor::Serialize (CArchive& ar)
{
   CObject::Serialize (ar);
	if (ar.IsStoring ()){
		ar << count;
		ar << homeId;
		ar << foldingDoorId;
	   ar << attachedToId1;
		ar << attachedToId2;
	   ar << hinge;
	   ar << lock;
```

```
                ar << foldingDoorClr;
                ar << rotation;
                ar << s_a_f_v;
                ar << width;
        }
    else {// Loading, not storing
                ar >> count;
                ar >> homeId;
                ar >> foldingDoorId;
            ar >> attachedToId1;
                ar >> attachedToId2;
            ar >> hinge;
            ar >> lock;
                ar >> foldingDoorClr;
                ar >> rotation;
                ar >> s_a_f_v;
                ar >> width;
        }
}

// FuelCell.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "FuelCell.h"

#include "MainFrm.h"
#include "FuelCellDoc.h"
#include "FuelCellView.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CFuelCellApp

BEGIN_MESSAGE_MAP(CFuelCellApp, CWinApp)
        //{{AFX_MSG_MAP(CFuelCellApp)
        ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
                // NOTE - the ClassWizard will add and remove mapping macros here.
                //    DO NOT EDIT what you see in these blocks of generated code!
        //}}AFX_MSG_MAP
        // Standard file based document commands
        ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
```

```
	ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CFuelCellApp construction

CFuelCellApp::CFuelCellApp()
{

	// TODO: add construction code here,
	// Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CFuelCellApp object

CFuelCellApp theApp;
/////////////////////////////////////////////////////////////////////////////
// CFuelCellApp initialization

BOOL CFuelCellApp::InitInstance()
{
	// Standard initialization
	// If you are not using these features and wish to reduce the size
	//  of your final executable, you should remove from the following
	//  the specific initialization routines you do not need.

	// Change the registry key under which our settings are stored.
	// TODO: You should modify this string to be something appropriate
	// such as the name of your company or organization.
	SetRegistryKey(_T("Local AppWizard-Generated Applications"));

	LoadStdProfileSettings();  // Load standard INI file options (including MRU)

	// Register the application's document templates.  Document templates
	//  serve as the connection between documents, frame windows and views.

	CSingleDocTemplate* pDocTemplate;
	pDocTemplate = new CSingleDocTemplate(
		IDR_MAINFRAME,
		RUNTIME_CLASS(CFuelCellDoc),
		RUNTIME_CLASS(CMainFrame),       // main SDI frame window
		RUNTIME_CLASS(CFuelCellView));
	AddDocTemplate(pDocTemplate);

	// Enable DDE Execute open
```

```
	EnableShellOpen();
	RegisterShellFileTypes(TRUE);

	// Parse command line for standard shell commands, DDE, file open
	CCommandLineInfo cmdInfo;
	ParseCommandLine(cmdInfo);

	// Dispatch commands specified on the command line
	if (!ProcessShellCommand(cmdInfo))
		return FALSE;

	// The one and only window has been initialized, so show and update it.
	m_pMainWnd->ShowWindow(SW_SHOW);
	m_pMainWnd->UpdateWindow();

	// Enable drag/drop open
	m_pMainWnd->DragAcceptFiles();
   return TRUE;
}


/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
		// No message handlers
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
```

```
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog
void CFuelCellApp::OnAppAbout()
{
	CAboutDlg aboutDlg;
	aboutDlg.DoModal();
}

/////////////////////////////////////////////////////////////////////////////
// CFuelCellApp message handlers


// FuelCellDoc.cpp : implementation of the CFuelCellDoc class
//

#include "stdafx.h"
#include "FuelCell.h"

#include "FuelCellDoc.h"
#include <iostream>
#include <fstream>
using namespace std;

/*Creating COLORREF objects that encapsulate RGB color objects
*/
```

```
COLORREF HOME_CLR(RGB(192,192,255)),
ROOM_CLR(RGB(255,255,255)),DOOR_CLR(RGB(255,255,255)),
      FOLDING_DOOR_CLR(RGB(0,0,0)), APPLIANCE_ON_CLR(RGB(255,0,0)),

APPLIANCE_OFF_CLR(RGB(100,100,255)),PORCH_CLR(RGB(192,192,192)),
                  WINDOW_CLR(RGB(255,255,255)),OPENING_CLR(RGB(255,255,255));
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;  //???????
#endif
/////////////////////////////////////////////////////////////////////////////
// CFuelCellDoc

IMPLEMENT_DYNCREATE(CFuelCellDoc, CDocument)

BEGIN_MESSAGE_MAP(CFuelCellDoc, CDocument)
        //{{AFX_MSG_MAP(CFuelCellDoc)
                // NOTE - the ClassWizard will add and remove mapping macros here.
                //    DO NOT EDIT what you see in these blocks of generated code!
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CFuelCellDoc construction/destruction

CFuelCellDoc::CFuelCellDoc()
{
        // TODO: add one-time construction code here

}

CFuelCellDoc::~CFuelCellDoc()
{
        POSITION pos = homes.GetHeadPosition();
        while(pos != NULL)
                delete homes.GetNext(pos);
}
BOOL CFuelCellDoc::OnNewDocument()
{
        if (!CDocument::OnNewDocument())
                return FALSE;
        // TODO: add reinitialization code here
        // (SDI documents will reuse this document)
        homeCount = 0;
   homes.AddTail( new Home(950,650,HOME_CLR));
```

```
        homeCount++;
        Appliances::setOffColor(APPLIANCE_OFF_CLR);

    //cornners of room
        int array_r0[40] = {25,50,400,50,400,400,25,400};
        int array_r1[40] = {400,50,775,50,775,400,400,400};



        //closet
        int array_r2[40] = {25,50,100,50,100,125,25,125};
        /*
        int array_r16[40] = {0,200,90,200,90,255,0,255};
        int array_r17[40] = {130,200,220,200,220,255,130,255};
        int array_r18[40] = {540,145,600,145,600,195,540,195};
        int array_r19[40] = {600,145,670,145,670,195,600,195};
        int array_r20[40] = {600,195,670,195,670,245,600,245};
        int array_r21[40] = {630,365,680,365,680,425,630,425};
        int array_r22[40] = {630,425,680,425,680,485,630,485};
        int array_r23[40] = {600,245,670,245,670,285,600,285};
        int array_r24[40] = {630,345,680,345,680,365,630,365};
        int array_r25[40] = {720,285,740,285,740,345,720,345};
        int array_r26[40] = {450,365,475,365,475,405,450,405};

        */
        // rectangles for appliances
                CRect rectFKB(200,210,250,260),rectFGR(570,205,620,255),
                rectW(35,235,85,295),rectDW(345,235,395,295),rectD(35,165,85,225),
                rectS(245,55,310,115),rectR(160,55,235,120),rectWH(350,65,395,110),
                rectHVAC(35,60,90,115);
//rectangles for lights
                CRect rectLGT1(210,150,240,180), rectLGT2(210,330,240,360),
                        rectLGT3(580,150,610,180), rectLGT4(580,330,610,360),
                        rectLGT5(25,405,45,425), rectLGT6(750,405,770,425);
//rectangles for outlets
                CRect rectHWO1(330,55,350,75), rectHWO2(200,375,220,395),
                        rectHWO3(30,130,50,150), rectHWO4(375,150,395,170),
                        rectHWO5(405,120,425,140), rectHWO6(750,150,770,170),
                        rectHWO7(480,55,500,75), rectHWO8(480,375,500,395),
                        rectHWO9(360,375,380,395),rectHWO10(405,230,425,250),
                        rectHWO11(700,375,720,395),rectHWO12(700,55,720,75),
                        rectHWO13(750,250,770,270);
```

```
	POSITION pos = homes.GetHeadPosition();
	Home * h = homes.GetNext(pos);

	// adding rooms
		h->addRoom(array_r0,4,ROOM_CLR,"KIT/EAT",CPoint(200,250));
		h->addRoom(array_r1,4,ROOM_CLR,"LIVING R",CPoint(560,250));
	//	h->addRoom(array_r2,4,ROOM_CLR,"AIR-RT",CPoint(30,70));


	//adding closets
	h->addCloset(1,array_r2,4,ROOM_CLR);



   // adding doors
		h->addDoor(1,-1,25,388,25,313,-90,DOOR_CLR);
		h->addDoor(2,-1,775,388,775,313,270,DOOR_CLR);
		h->addDoor(1,2,400,388,400,313,270,DOOR_CLR);
		h->addDoor(1,-1,100,60,100,105,90,DOOR_CLR);



	//adding foldingDoors



	//add opening
//	h->addOpening(13,7,475,345,510,345,90,OPENING_CLR);


   //adding windows
//	h->addWindow(4,770,145,790,145,90,WINDOW_CLR);
		h->addWindow(1,100,400,300,400,90,WINDOW_CLR);
		h->addWindow(2,475,400,675,400,90,WINDOW_CLR);
		h->addWindow(2,775,75,775,275,180,WINDOW_CLR);
		h->addWindow(1,400,125,400,275,180,WINDOW_CLR);



	//adding appliances
	h->addApplainces(1,rectW,APPLIANCE_ON_CLR,10.f,Appliances::WASHER);
	h->addApplainces(1,rectD,APPLIANCE_ON_CLR,10.f,Appliances::DRIER);
```

```
        h->addApplainces(1,rectHVAC,APPLIANCE_ON_CLR,10.f,Appliances::HVAC);
        h-
>addApplainces(1,rectWH,APPLIANCE_ON_CLR,10.f,Appliances::WATER_HEATER);
//      h->addApplainces(1,rectO,APPLIANCE_ON_CLR,10.f,Appliances::OVEN);
        h->addApplainces(1,rectS,APPLIANCE_ON_CLR,10.f,Appliances::STOVE);
        h-
>addApplainces(1,rectDW,APPLIANCE_ON_CLR,10.f,Appliances::DISH_WASHER);
        h->addApplainces(1,rectR,APPLIANCE_ON_CLR,10.f,Appliances::REFRIGERATOR);
        //-----------------fans-----------------------------------------//
        h->addApplainces(1,rectFKB,APPLIANCE_ON_CLR,10.f,Appliances::FAN);
        h->addApplainces(2,rectFGR,APPLIANCE_ON_CLR,10.f,Appliances::FAN);

        //-----------------bulbs----------------------------------------//

        h->addApplainces(1,rectLGT1,APPLIANCE_ON_CLR,10.f,Appliances::LIGHT);
        h->addApplainces(1,rectLGT2,APPLIANCE_ON_CLR,10.f,Appliances::LIGHT);
        h->addApplainces(2,rectLGT3,APPLIANCE_ON_CLR,10.f,Appliances::LIGHT);
        h->addApplainces(2,rectLGT4,APPLIANCE_ON_CLR,10.f,Appliances::LIGHT);
//      h->addApplainces(1,rectLGT5,APPLIANCE_ON_CLR,10.f,Appliances::LIGHT);
        h->addApplainces(2,rectLGT6,APPLIANCE_ON_CLR,10.f,Appliances::LIGHT);
/*      h->addApplainces(10,rectLGT10,APPLIANCE_ON_CLR,10.f,Appliances::LIGHT);
        h->addApplainces(11,rectLGT11,APPLIANCE_ON_CLR,10.f,Appliances::LIGHT);
        h->addApplainces(12,rectLGT12,APPLIANCE_ON_CLR,10.f,Appliances::LIGHT);
        h->addApplainces(13,rectLGT13,APPLIANCE_ON_CLR,10.f,Appliances::LIGHT);
        h->addApplainces(14,rectLGT14,APPLIANCE_ON_CLR,10.f,Appliances::LIGHT);
        h->addApplainces(4,rectLGT4MB,APPLIANCE_ON_CLR,10.f,Appliances::LIGHT);
        */
//      ----------outlets---------------------------------------------------------//
        h-
>addApplainces(1,rectHWO1,APPLIANCE_ON_CLR,10.f,Appliances::HIGH_WATTAGE_OU
TLET);
        h-
>addApplainces(1,rectHWO2,APPLIANCE_ON_CLR,10.f,Appliances::HIGH_WATTAGE_OU
TLET);
        h-
>addApplainces(1,rectHWO3,APPLIANCE_ON_CLR,10.f,Appliances::HIGH_WATTAGE_OU
TLET);
        h-
>addApplainces(1,rectHWO4,APPLIANCE_ON_CLR,10.f,Appliances::HIGH_WATTAGE_OU
TLET);
        h-
>addApplainces(2,rectHWO5,APPLIANCE_ON_CLR,10.f,Appliances::HIGH_WATTAGE_OU
TLET);
        h-
>addApplainces(2,rectHWO6,APPLIANCE_ON_CLR,10.f,Appliances::HIGH_WATTAGE_OU
TLET);
```

```
        h-
>addApplainces(2,rectHWO7,APPLIANCE_ON_CLR,10.f,Appliances::HIGH_WATTAGE_OU
TLET);
        h-
>addApplainces(2,rectHWO8,APPLIANCE_ON_CLR,10.f,Appliances::HIGH_WATTAGE_OU
TLET);
        h-
>addApplainces(1,rectHWO9,APPLIANCE_ON_CLR,10.f,Appliances::HIGH_WATTAGE_OU
TLET);
        h-
>addApplainces(2,rectHWO10,APPLIANCE_ON_CLR,10.f,Appliances::HIGH_WATTAGE_O
UTLET);
//      h-
>addApplainces(9,rectLWO9,APPLIANCE_ON_CLR,10.f,Appliances::LOW_WATTAGE_OU
TLET);
        h-
>addApplainces(2,rectHWO11,APPLIANCE_ON_CLR,10.f,Appliances::HIGH_WATTAGE_O
UTLET);
//      h-
>addApplainces(10,rectLWO10,APPLIANCE_ON_CLR,10.f,Appliances::LOW_WATTAGE_O
UTLET);
        h-
>addApplainces(2,rectHWO12,APPLIANCE_ON_CLR,10.f,Appliances::HIGH_WATTAGE_O
UTLET);
//      h-
>addApplainces(11,rectLWO11,APPLIANCE_ON_CLR,10.f,Appliances::LOW_WATTAGE_O
UTLET);
        h-
>addApplainces(2,rectHWO13,APPLIANCE_ON_CLR,10.f,Appliances::HIGH_WATTAGE_O
UTLET);
/*      h-
>addApplainces(12,rectLWO12,APPLIANCE_ON_CLR,10.f,Appliances::LOW_WATTAGE_O
UTLET);
        h-
>addApplainces(12,rectHWO12,APPLIANCE_ON_CLR,10.f,Appliances::HIGH_WATTAGE_
OUTLET);
        h-
>addApplainces(13,rectLWO13,APPLIANCE_ON_CLR,10.f,Appliances::LOW_WATTAGE_O
UTLET);
        h-
>addApplainces(13,rectHWO13,APPLIANCE_ON_CLR,10.f,Appliances::HIGH_WATTAGE_
OUTLET);
        h-
>addApplainces(14,rectLWO14,APPLIANCE_ON_CLR,10.f,Appliances::LOW_WATTAGE_O
UTLET);
```

```
	h-
>addApplainces(14,rectHWO14,APPLIANCE_ON_CLR,10.f,Appliances::HIGH_WATTAGE_
OUTLET);
	h-
>addApplainces(4,rectLWO4MB,APPLIANCE_ON_CLR,10.f,Appliances::LOW_WATTAGE_
OUTLET);
	h-
>addApplainces(4,rectHWO4MB,APPLIANCE_ON_CLR,10.f,Appliances::HIGH_WATTAGE_
OUTLET);
	//---------------------------ExhaustFan---------------------------------------//
	h-
>addApplainces(6,rectEFKB,APPLIANCE_ON_CLR,10.f,Appliances::EXHAUST_FAN);
	h-
>addApplainces(12,rectEFB,APPLIANCE_ON_CLR,10.f,Appliances::EXHAUST_FAN);
	h-
>addApplainces(4,rectEFMB,APPLIANCE_ON_CLR,10.f,Appliances::EXHAUST_FAN);
	*/
	//-----------------------------------------------------------------------------//
	///READING DATA FROM FILE  HAPPENS ONLY ONCE
   arrayPointer = 0;
   ifstream input("AppliancesDoc.txt");
	if(input){

		int homeIdFromFile ;
		input >> homeIdFromFile;
		lastInUse =0;
		while(input>> applIDBuff[lastInUse] >> applStatusBuff[lastInUse] >>
applPowerBuff[lastInUse])
			lastInUse++;
	}
	input.close();
	return TRUE;
}

BOOL CFuelCellDoc::OnOpenDocument(LPCTSTR lpszPathName){
	if (!CDocument::OnOpenDocument(lpszPathName))
		return FALSE;
	return TRUE;

}

/*void CFuelCellDoc::dataFromFile(){
	if (arrayPointer >= lastInUse )
		arrayPointer = 0;
	POSITION pos = homes.GetHeadPosition();
	Home * home;
```

```
        while(pos != NULL){
                home = homes.GetNext(pos);
                home->setCurPower(applIDBuff,applStatusBuff,applPowerBuff, arrayPointer);
        }
        SetModifiedFlag(TRUE);
        //UpdateAllViews(NULL);
        home->draw (NULL);
}*/

void CFuelCellDoc::dataFromFile( CDC* dc,CRect *rect){
        if (arrayPointer >= lastInUse )
                arrayPointer = 0;
        POSITION pos = homes.GetHeadPosition();
        Home * home;
        while(pos != NULL){
                home = homes.GetNext(pos);
                home->setCurPower(applIDBuff,applStatusBuff,applPowerBuff, arrayPointer);
        }
        SetModifiedFlag(TRUE);
        //UpdateAllViews(NULL);     commented out by shahrukh to remove the flicker
        home->setDC(dc,rect);       //added in process of removing flicker
        home->draw (dc);            //added in process of removing flicker

}


/////////////////////////////////////////////////////////////////////////////
// CFuelCellDoc serialization

void CFuelCellDoc::Serialize(CArchive& ar)
{
        if (ar.IsStoring())
        {
                // TODO: add storing code here
                ar << homeCount;
                POSITION pos = homes.GetHeadPosition();
                while(pos != NULL)
                        ar << homes.GetNext(pos);
        }
        else
        {
                // TODO: add loading code here
                ar >> homeCount;
                Home * h;
                for (int i = 0; i < homeCount ; i++){
                        ar >> h;
```

```
                        homes.AddTail(h);
                }
        }
}

/////////////////////////////////////////////////////////////////////////////
// CFuelCellDoc diagnostics

#ifdef _DEBUG
void CFuelCellDoc::AssertValid() const
{
        CDocument::AssertValid();
}

void CFuelCellDoc::Dump(CDumpContext& dc) const
{
        CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CFuelCellDoc commands


// FuelCellView.cpp : implementation of the CFuelCellView class

#include "stdafx.h"
#include "FuelCell.h"
#include "FuelCellDoc.h"
#include "FuelCellView.h"
#include "SemacClient.h"

#define ID_TIMER_CLOCK 1
#define ID_TIMER_SIMULATION 2

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


/////////////////////////////////////////////////////////////////////////////
// CFuelCellView

IMPLEMENT_DYNCREATE(CFuelCellView, CView)
```

```
BEGIN_MESSAGE_MAP(CFuelCellView, CView)
	//{{AFX_MSG_MAP(CFuelCellView)
	ON_WM_CREATE()
	ON_WM_TIMER()
	ON_WM_LBUTTONDBLCLK()
	ON_WM_RBUTTONDOWN()
	//}}AFX_MSG_MAP
	ON_MESSAGE ( MESSAGE_RECEIVED_FROM_SEMAC, OnSemacThreadMessage )
	ON_MESSAGE ( DEVICE_STATUS_MSG_FROM_SEMAC,
OnSemacDeviceStatusMsg )
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CFuelCellView construction/destruction



CFuelCellView::CFuelCellView()
{
	//todoo code
}

CFuelCellView::~CFuelCellView()
{
}

LONG CFuelCellView::OnSemacThreadMessage ( WPARAM wParam, LPARAM lParam )
{

	setSemacMessage( (char*)wParam );
	return 0;
}

LONG CFuelCellView::OnSemacDeviceStatusMsg ( WPARAM wParam, LPARAM lParam )
{
	CRect dRect;		//added by shahrukh to remove the flicker
	GetClientRect (&dRect);	//added by shahrukh to remove the flicker
	CClientDC pDC2 (this);	//added by shahrukh to remove the flicker
	pDoc->dataFromFile(&pDC2,&dRect); //changed by shahrukh to remove the flicker
	return 0;

}
BOOL CFuelCellView::PreCreateWindow(CREATESTRUCT& cs)
{
	// TODO: Modify the Window class or styles here by modifying
```

```
	//  the CREATESTRUCT cs

	return CView::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////////////
// CFuelCellView drawing

void CFuelCellView::OnDraw(CDC* pDC)
{
	CRect rect;
	GetClientRect (&rect);
	// TODO: add draw code for native data here
	POSITION pos = pDoc->homes.GetHeadPosition();
	//while (pos != NULL){  //for  multiple homes not needed at this time
       Home * h = pDoc->homes.GetNext(pos);
            h->drawSemacMessage (this);
       CWnd::OnPaint ();
//        }
   pos = pDoc->homes.GetHeadPosition();
	while (pos != NULL){
		Home * h = pDoc->homes.GetNext(pos);
		//SetBkColor(*pDC,h->getColor());
		h->sethomeInDefaultView (true); //to redraw properly on window rezise, this is a
hack fix it
      h->setSelectedRoomEnlargedPaintedForFirstTime (true); //to redraw properly on window
rezise, this is a hack fix it
		h->setDC(pDC,&rect);
		h->draw(pDC);
	}

}


/////////////////////////////////////////////////////////////////////////////
// CFuelCellView diagnostics

#ifdef _DEBUG
void CFuelCellView::AssertValid() const
{
	CView::AssertValid();
}

void CFuelCellView::Dump(CDumpContext& dc) const
{
	CView::Dump(dc);
```

```
}

CFuelCellDoc* CFuelCellView::GetDocument() // non-debug version is inline
{
	ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CFuelCellDoc)));
	return (CFuelCellDoc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CFuelCellView message handlers

int CFuelCellView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
	if (CView::OnCreate(lpCreateStruct) == -1)
		return -1;

   // Set a timer to fire at 1-second intervals.
   //
   if (!SetTimer (ID_TIMER_CLOCK, 1000, NULL)) {
      MessageBox (_T ("SetTimer failed"));
      return -1;
   }
	if (!SetTimer (ID_TIMER_SIMULATION, 5000, NULL)) {
      MessageBox (_T ("SetTimer failed"));
      return -1;
   }
	return 0;
}


void CFuelCellView::OnTimer(UINT nIDEvent)
{
//
   // Do nothing if the window is minimized.
   //
   if (IsIconic ())
      return;
   if (nIDEvent == ID_TIMER_CLOCK){
		//
		// Get the current time and do nothing if it hasn't changed.
		//
		CTime time = CTime::GetCurrentTime ();
		int nSecond = time.GetSecond ();

		 /*if (nSecond == m_nPrevSecond)
```

```
                return;*/

            //
            // If seconds have changed
            //
            CRect rect;
            GetClientRect (&rect);
            CClientDC pDC (this);
            POSITION pos = pDoc->homes.GetHeadPosition();
            while (pos != NULL){
                Home * h = pDoc->homes.GetNext(pos);
                //SetBkColor(pDC,h->getColor());
                h->setDC(&pDC,&rect);
                h->drawClock(&pDC);
            }

            //if (nSecond != m_nPrevSecond) {
            //    drawClock(&pDC);
            //    m_nPrevSecond = nSecond;

            //}
        }
/*      else {
            CRect dRect;            //added by shahrukh to remove the flicker
            GetClientRect (&dRect);     //added by shahrukh to remove the flicker
            CClientDC pDC2 (this);      //added by shahrukh to remove the flicker
            pDoc->dataFromFile(&pDC2,&dRect); //changed by shahrukh to remove the
flicker
        } */
        CView::OnTimer(nIDEvent);
}


/*OnLButtonDblClk - is called when user double clicks on the "Home". the "homes"
CTypedPtrList object
        is traverresed to find which "Home" was clicked on and then
        Home::selectRoom method of the clicked "Home" is called to find what "room"
        was clicked on inorder to enlarge it
*/
void CFuelCellView::OnLButtonDblClk(UINT nFlags, CPoint point)
{

        CClientDC pDC (this);
        CRect rect;
        GetClientRect (&rect);
        POSITION pos = pDoc->homes.GetHeadPosition();
```

```
        while (pos != NULL){
                Home * h = pDoc->homes.GetNext(pos);
                //SetBkColor(pDC,h->getColor());
                h->setDC(&pDC,&rect);
                pDC.DPtoLP(&point);
                if (!(h->getClicked())){

                        if( h->selectRoom(point)){
          h->setClicked(1);
                                h->setSelectedRoomEnlargedPaintedForFirstTime ( true );
                                //h->draw(&pDC);
                                Invalidate();
                        //      CView::OnUpdate( NULL, 1, (CObject*)&rect );
                                h->drawSemacMessage (this);
                        }
                }
                else {
        h->setClicked(0);
                        h->sethomeInDefaultView ( true );
                        h->setSelectedRoomEnlargedPaintedForFirstTime ( true );
                        //h->draw(&pDC);
                        Invalidate();
                        //CView::OnUpdate( NULL, 1, (CObject*)&rect );
                        h->drawSemacMessage (this);
      }
    }

        CView::OnLButtonDblClk(nFlags, point);
}

void CFuelCellView::OnRButtonDown(UINT nFlags, CPoint point)
{
        CClientDC pDC (this);
        CRect rect;
        GetClientRect (&rect);
        POSITION pos = pDoc->homes.GetHeadPosition();
        while (pos != NULL){
                Home * h = pDoc->homes.GetNext(pos);
                //SetBkColor(pDC,h->getColor());
                if (h->getClicked()){
                        h->setDC(&pDC,&rect);
                        h->setCurPower(point,&pDC);
                        h->draw(&pDC);     //added by shahrukh to remove flicker
                        //Invalidate();    removed by shahrukh as it was causing flicker
                }
        }
```

```
        CView::OnRButtonDown(nFlags, point);
}

void CFuelCellView::OnInitialUpdate()
{
        CView::OnInitialUpdate();

        pDoc = GetDocument();
        ASSERT_VALID(pDoc);

   CClientDC pDC (this);
   CRect rect;
   GetClientRect( &rect );

  //code to draw the semac message box and label for the first time in application life cycle
        POSITION pos = pDoc->homes.GetHeadPosition();
        //while (pos != NULL){  //for  multiple homes not needed at this time
      Home * h = pDoc->homes.GetNext(pos);
                //h->setDC(&pDC,&rect);
                h->setCurrentSemacMessage("waiting for a message from Semac Server");
                h->drawSemacMessage ( this );

//      }

   //make a callto SemacClinet's communicateWithSemacServer()
        SemacClient communicator;
   communicator.startSemacClientThread(CView::m_hWnd);

}

void CFuelCellView::setSemacMessage(char * SemacMessage)
{
        pDoc = GetDocument();
        ASSERT_VALID(pDoc);

   CClientDC pDC (this);
   CRect rect;
   GetClientRect( &rect );

   POSITION pos = pDoc->homes.GetHeadPosition();
        //while (pos != NULL){  //for  multiple homes not needed at this time
      Home * h = pDoc->homes.GetNext(pos);
                h->setDC(&pDC,&rect);
                h->setCurrentSemacMessage ( SemacMessage );
                h->setSemacMessage ( SemacMessage );
}
```

```
#include "stdafx.h"
#include "Home.h"

//#include <fstream>  //for debugging remove later
//#include <iostream> //for debugging remove later
//using namespace std; //for debugging remove later


int Home::count;
//Definition of class Home

IMPLEMENT_SERIAL (Home, CObject, 1);
Home::Home(){}
Home::~Home(){
        --count;
        POSITION pos;
        pos = rooms.GetHeadPosition();
        while(pos != NULL){
                delete rooms.GetNext(pos);
        }
        pos = doors.GetHeadPosition();
        while(pos != NULL){
                delete doors.GetNext(pos);
        }
        pos = foldingDoors.GetHeadPosition();
        while(pos != NULL){
                delete foldingDoors.GetNext(pos);
        }
        pos = openings.GetHeadPosition();
        while(pos != NULL){
                delete openings.GetNext(pos);
        }
        pos = windows.GetHeadPosition();
        while(pos != NULL){
                delete windows.GetNext(pos);
        }
        pos = appliances.GetHeadPosition();
        while(pos != NULL){
                delete appliances.GetNext(pos);
        }
   if ( semacMsgBox != NULL)
        {
                delete semacMsgBox ;
        }
```

```
        if( semacLabel != NULL )
        {
                delete semacLabel ;
        }
}

Home::Home(int theMaxLength, int theMaxWidth, COLORREF theClr){
        homeId = ++count;
        homeClr = theClr;
        maxPower =0.f;
        curPower = 0.f;
        powerConsumed = 0.f;
        selectedRoom = -1;
        maxLength = (int)(1.5*theMaxLength);
        maxWidth = (int)(1.5*theMaxWidth);
        timeSlot = 0;
        roomCount = 0;
        doorCount = 0;
        foldingDoorCount = 0;
        openingCount = 0;
        windowCount = 0;
        appliancesCount = 0;
        clicked = 0;
        homeInDefaultView = true;
        lastRoomPowerLabelWidthPos =0;
        semacMsgBox = new CEdit();
        semacLabel = new CEdit();
        currentSemacMessage = new char [512];
}

Home::Home(const Home & h){
        homeId = h.homeId;
        homeClr = h.homeClr;
        maxPower = h.maxPower;
        curPower = h.curPower;
        powerConsumed = h.powerConsumed;
        maxLength = h.maxLength;
        maxWidth = h.maxWidth;
        timeSlot = 0;
        selectedRoom = h.selectedRoom;
        POSITION pos;
        pos = h.rooms.GetHeadPosition();
   while(pos != NULL){
            Room * r = h.rooms.GetNext(pos);
                rooms.AddTail(r);
```

```
		}
		pos = doors.GetHeadPosition();
		while(pos != NULL){
			Door * d =  h.doors.GetNext(pos);
			doors.AddTail(d);
		}
		pos = foldingDoors.GetHeadPosition();
		while(pos != NULL){
			FoldingDoor * fd =  h.foldingDoors.GetNext(pos);
			foldingDoors.AddTail(fd);
		}
		pos = openings.GetHeadPosition();
		while(pos != NULL){
			Opening * o =  h.openings.GetNext(pos);
			openings.AddTail(o);
		}
		pos = windows.GetHeadPosition();
		while(pos != NULL){
			Window * w =  h.windows.GetNext(pos);
			windows.AddTail(w);
		}
		pos = appliances.GetHeadPosition();
		while(pos != NULL){
			Appliances * a =  h.appliances.GetNext(pos);
			appliances.AddTail(a);
		}
		roomCount = h.roomCount;
		doorCount = h.doorCount;
		foldingDoorCount = h.foldingDoorCount;
		openingCount = h.openingCount;
		windowCount = h.windowCount;
		appliancesCount = h.appliancesCount;
		clicked = h.clicked;
		homeInDefaultView = h.homeInDefaultView;
		lastRoomPowerLabelWidthPos = h.lastRoomPowerLabelWidthPos;
		lastPowerLabelWidthPos = h.lastPowerLabelWidthPos;

}
/*Home * Home::operator =(Home *h){
		return h;
}*/
Home  Home::operator =(Home h){
		return h;
}

void Home::setDC(CDC* dc,CRect *rect){
```

```
   CPoint orgV(maxLength/10,maxWidth/10);
        dc->SetMapMode(MM_ISOTROPIC);        //setting the mapping mode
        dc->SetWindowExt(maxLength,maxWidth); //setting the window acording to the size
        dc->SetViewportExt(rect->Width(),rect->Height());
        dc->LPtoDP(&orgV);
        dc->SetViewportOrg(orgV);
}

void Home::draw(CDC* dc){
        CPoint moveOrg(maxLength,maxWidth), resetOrg;
        CGdiObject * pOldPen = dc->SelectStockObject(NULL_PEN);
        CBrush cbrush(homeClr), *oldBrush;
        oldBrush = dc->SelectObject(&cbrush);

        //dc->Rectangle(-maxLength/10,-maxWidth/10,maxLength,maxWidth); changin position
2 lines down
        //to remove flicker, as the home does not need to be erased every time.
        //dc->SelectObject (pOldPen); moving down with previous statement

        if( (!clicked) && ( homeInDefaultView ) ){
                dc->Rectangle(-maxLength/10,-maxWidth/10,maxLength,maxWidth);
                POSITION pos;
                pos = rooms.GetHeadPosition();
                while(pos != NULL){
                        rooms.GetNext(pos)->draw(dc);
                }
                pos = doors.GetHeadPosition();
                while(pos != NULL){
                        doors.GetNext(pos)->draw(dc);
                }
                pos = foldingDoors.GetHeadPosition();
                while(pos != NULL){
                        foldingDoors.GetNext(pos)->draw(dc);
                }
                pos = openings.GetHeadPosition();
                while(pos != NULL){
                        openings.GetNext(pos)->draw(dc);
                }
                pos = windows.GetHeadPosition();
                while(pos != NULL){
                        windows.GetNext(pos)->draw(dc);
                }
                pos = appliances.GetHeadPosition();
                while(pos != NULL){
                        appliances.GetNext(pos)->draw(dc);
```

```
		}
		homeInDefaultView = false; //added by shahrukh to remove flicker
	}
	else if( (!clicked) && ( ! homeInDefaultView ) ){

		POSITION pos;
		dc->SelectObject (pOldPen); //added by shahrukh to remove flicker
		pos = appliances.GetHeadPosition();
		while(pos != NULL){
			appliances.GetNext(pos)->draw(dc);
		}

	}
	else { // code in else draws the enlarged view
		//dc->Rectangle(-maxLength/10,-maxWidth/10,maxLength,maxWidth);
		dc->SelectObject (pOldPen); //added by shahrukh, while removing flicker
		POSITION pos;
		pos = rooms.GetHeadPosition();
		while(pos != NULL){
			Room * room = rooms.GetNext(pos);
			if (room->getRoomId() == selectedRoom){
				room->SelectOrigin(&moveOrg);
				resetOrg = dc->GetWindowOrg();
				float factor = room->magnifyingFactor(maxLength,maxWidth);
				dc-
>SetWindowExt((int)(maxLength/factor),(int)(maxWidth/factor));
				dc->SetWindowOrg(moveOrg);
				if ( room->getRoomEnlargedPaintedforfirstTime () )
	{
	    dc->Rectangle(-maxLength/10,-maxWidth/10,maxLength,maxWidth);
					room->draw(dc);
						room->setRoomEnlargedPaintedforfirstTime ( false );
				}
			}
		}
		pos = doors.GetHeadPosition();
		while(pos != NULL){
			Door * door = doors.GetNext(pos);

			//attachedToId1 or attachedToId2 as common door between two rooms
have roomId's
			//of both Room objects attached to it.
			if (door->attachedToId1 == selectedRoom || door->attachedToId2 ==
selectedRoom)
				door->draw(dc);
		}
```

```
                pos = foldingDoors.GetHeadPosition();
                while(pos != NULL){
                        FoldingDoor * fdoor = foldingDoors.GetNext(pos);
                        if (fdoor->attachedToId1 == selectedRoom || fdoor->attachedToId2 ==
selectedRoom)
                                fdoor->draw(dc);
                }
                pos = openings.GetHeadPosition();
                while(pos != NULL){
                        Opening * opening = openings.GetNext(pos);
                        if (opening->attachedToId1 == selectedRoom || opening->attachedToId2
== selectedRoom)
                                opening->draw(dc);
                }
                pos = windows.GetHeadPosition();
                while(pos != NULL){
                        Window * window = windows.GetNext(pos);
                        if(window->attachedToId == selectedRoom)
                                window->draw(dc);
                }
                pos = appliances.GetHeadPosition();
                while(pos != NULL){
                        Appliances * appliance = appliances.GetNext(pos);
                        if(appliance->attachedToId == selectedRoom)
                                appliance->draw(dc);
                }
                dc->SetWindowExt(maxLength,maxWidth);
                dc->SetWindowOrg(resetOrg);
                Appliances *app = new Appliances();
                app->drawList(dc,(int)(7.7*maxLength/10),2*maxWidth/10);// ?????????what is
7.7 and 2 and 10
                delete app;
        }
        drawClock(dc);
        drawGraph(dc);
        dc->SelectObject(oldBrush);
        dc->SelectObject(pOldPen);
}

void Home::drawClock(CDC *dc){
        CTime time = CTime::GetCurrentTime();
        int nSecond = time.GetSecond();
        int nMinute = time.GetMinute();
        int nHour = time.GetHour();
   CFont font;
        font.CreatePointFont(140,_T("Times New Roman"));
```

```
        CFont * pOldFont = dc->SelectObject(&font);
        //dc->SetTextColor(RGB(0,0,0));
        CString s;
                if(nHour > 12)
           s.Format(_T("%0.2d:%0.2d:%0.2d PM"),nHour%12,nMinute,nSecond);
                else
        s.Format(_T("%0.2d:%0.2d:%0.2d AM"),nHour,nMinute,nSecond);
        COLORREF bkClr =  dc->SetBkColor(homeClr);
        CGdiObject * pOldPen = dc->SelectStockObject(NULL_PEN);
        CBrush cbrush(homeClr), *oldBrush;
        oldBrush = dc->SelectObject(&cbrush);
        dc->Rectangle((int)(7.7*maxLength/10),maxWidth/10,maxLength,2*maxWidth/10);
        dc->SelectObject (pOldPen);
   dc->TextOut(8*maxLength/10,maxWidth/10,s);
        dc->SetBkColor(bkClr);
        dc->SelectObject(pOldFont);
}
void Home::deleteOldSemacMsgBoxAndLabel()
{
        if ( semacMsgBox != NULL)
        {
                delete semacMsgBox ;
        }

        if( semacLabel != NULL )
        {
                delete semacLabel ;
        }
}
void Home::setSemacMessage(char * SemacMessage)
{
   semacMsgBox->SetWindowText ( _T ( SemacMessage  ) );
}
 void Home::drawSemacMessage( CView * cMainFrameRef){
        deleteOldSemacMsgBoxAndLabel ();
        CRect rect;
        cMainFrameRef->GetClientRect(&rect);
        CClientDC dc (cMainFrameRef);
        setDC(&dc,&rect);
        int msgBoxHeight = maxWidth/40;
        //y cordiantes are common for for label and the box
   int startY = 33*msgBoxHeight;
        int endY = startY+ (2*msgBoxHeight);

        //code for constructing label co-ordiantes(logical) in CPoint and converting them to
Physical Co-ordinates
```

```
        int startXLabel = 2;
        int endXLabel = 162;
        CPoint startXLabelPoint ( startXLabel, startY);
        CPoint endXLabelPoint ( endXLabel, endY);
        dc.LPtoDP ( &startXLabelPoint );
   dc.LPtoDP ( &endXLabelPoint );
        CRect enclosingRectSemacLabel (startXLabelPoint, endXLabelPoint);
        //end of code for label co-ord.

        //code for constructing CEdit box co-ordiantes(logical) in CPoint and converting them to
Physical Co-ordinates
        int endX =  ( Home::maxLength - 5*msgBoxHeight );
        CPoint startPoint( endXLabel + msgBoxHeight , startY );
        CPoint endPoint ( endX, endY );
        dc.LPtoDP  ( &startPoint );
        dc.LPtoDP  ( &endPoint );
        CRect enclosingRectEditBox ( startPoint, endPoint);
   //code for CEdit box Co-Ord ends here

// code for drawing semac label
   semacLabel = new CEdit;
        semacLabel->Create (WS_CHILD | WS_VISIBLE | WS_BORDER | ES_READONLY,
enclosingRectSemacLabel, cMainFrameRef, 1233 );
        semacLabel->SetWindowText ( _T ( "Semac Status:") );
// code for drawing semac label ends here

// code for Semac message text box starts
        semacMsgBox = new CEdit;
        semacMsgBox->Create (WS_CHILD | WS_VISIBLE | WS_BORDER |
ES_AUTOHSCROLL | ES_READONLY, enclosingRectEditBox, cMainFrameRef, 1234 );
        semacMsgBox->SetWindowText ( _T ( Home::currentSemacMessage  ) );
        Home::semacMsgBox->SetFocus();
// code for Semac message text box ends

}

 void Home::setCurrentSemacMessage (char * SemacMessage)
 {
         strcpy( Home::currentSemacMessage, SemacMessage );
 }
void Home::drawGraph(CDC *pdc ){

        int barLength = 6*maxLength/10;
        int barHeight = maxWidth/40;
        int curBar = (int)((curPower/maxPower)*barLength);
        int curRoomBar;
```

```
        float curSelectedRoomPower;
        POSITION pos = rooms.GetHeadPosition();

        while(pos != NULL){
                Room * room = rooms.GetNext(pos);
                if (room->getRoomId() == selectedRoom){
                        curSelectedRoomPower = room->getCurPower();
                        curRoomBar = (int)((curSelectedRoomPower/maxPower)*barLength);
                }
        }

        //to erase the old power label by painting with background color

        pdc->SelectStockObject(NULL_PEN);
        CBrush cbrushErase(homeClr), *oldBrush;
   oldBrush = pdc->SelectObject( &cbrushErase );
   pdc->Rectangle( Home::lastPowerLabelWidthPos, 29*maxWidth/40,
(lastPowerLabelWidthPos + (barLength/3)),29*maxWidth/40+barHeight);
        //erasing done

        CFont font,* pOldFont;

        font.CreatePointFont(120,_T("Times New Roman"));
        pOldFont = pdc->SelectObject(&font);
        //pdc->SetTextColor(RGB(0,0,0));
        CString s1,s2;
        s1.Format(_T("Total Power: %0.2f"),curPower);
        s2.Format(_T("Room Power : %0.2f"),curSelectedRoomPower);
   //eraseStr.Format(_T("Total Power       : %0.2f"),curPower)
        CPen pen(PS_SOLID,1,RGB(0,0,0)),*pOldPen;
        pOldPen = pdc->SelectObject(&pen);
        CBrush cbrush(RGB(255,255,255)), *pOldBrush;
        pOldBrush = pdc->SelectObject(&cbrush);
        COLORREF bkClr =  pdc->SetBkColor(homeClr);
        pdc->Rectangle(0,30*maxWidth/40,barLength,30*maxWidth/40+barHeight);
        CBrush cbrush1( RGB(0,138,138) );
        pdc->SelectObject( &cbrush1 );
        pdc->Rectangle(0,30*maxWidth/40,curBar,30*maxWidth/40+barHeight);
        pdc->TextOut(curBar,29*maxWidth/40,s1);
        if (clicked){
                //to erase the old Room power comsumption label by painting with background
color
        pdc->SelectStockObject(NULL_PEN);
             CBrush cbrushErase(homeClr), *oldBrush;
        oldBrush = pdc->SelectObject( &cbrushErase );
```

```
		//pdc->Rectangle( Home::lastRoomPowerLabelWidthPos, 29*maxWidth/40, (
Home::lastRoomPowerLabelWidthPos  + (barLength/4)),30*maxWidth/40+barHeight);
				pdc->Rectangle( 0, 27*maxWidth/40, (
Home::lastRoomPowerLabelWidthPos  + (barLength/2)),27*maxWidth/40+barHeight);
			//erasing done

				CBrush cbrush2(RGB(255,255,255));
			pdc->SelectObject(&cbrush2);
				pdc-
>Rectangle(0,28*maxWidth/40,barLength,28*maxWidth/40+barHeight);
				CBrush cbrush3(RGB(0,138,138));
			pdc->SelectObject(&cbrush3);
				pdc-
>Rectangle(0,28*maxWidth/40,curRoomBar,28*maxWidth/40+barHeight);
				pdc->TextOut(curRoomBar,27*maxWidth/40,s2);
		}
		pdc->SetBkColor(bkClr);
		pdc->SelectObject(pOldBrush);
		pdc->SelectObject(pOldPen);
		pdc->SelectObject(pOldFont);

		//to use when next power change occurs, to erase the last power label
		Home::lastPowerLabelWidthPos = curBar;
		Home::lastPowerLabel = s1;
}

int Home::selectRoom(CPoint point){
		POSITION pos;
		pos = rooms.GetHeadPosition();
		while(pos != NULL){
			Room * r = rooms.GetNext(pos);
			if (r->selectedRoom(point)){
				selectedRoom = r->getRoomId();
				return 1;
			}
		}
		return 0;
}

void Home::setSelectedRoomEnlargedPaintedForFirstTime( bool setValue ){
		POSITION pos;
		pos = rooms.GetHeadPosition();
		while(pos != NULL){
			Room * r = rooms.GetNext(pos);
			if ( r->getRoomId() == selectedRoom ){
			   r->setRoomEnlargedPaintedforfirstTime ( setValue );
```

```
                        break;
                }
        }
}

void Home::setCurPower(CPoint point,CDC *dc){
        CPoint moveOrg, resetOrg;
        POSITION pos = rooms.GetHeadPosition();
        Room * room;
        while(pos != NULL){
                room = rooms.GetNext(pos);
                if (selectedRoom == room->getRoomId())
                        break;
        }
        room->SelectOrigin(&moveOrg);
        resetOrg = dc->GetWindowOrg();
        float factor = room->magnifyingFactor(maxLength,maxWidth);
        dc->SetWindowExt((int)(maxLength/factor),(int)(maxWidth/factor));
        dc->SetWindowOrg(moveOrg);
        pos = appliances.GetHeadPosition();
        dc->DPtoLP(&point);
        while(pos != NULL){
                Appliances * appliance = appliances.GetNext(pos);
                if(appliance->attachedToId == selectedRoom){
                        if(appliance->setOnOff(point)){
                                curPower = curPower + appliance->getPowerChange();
                           room->setCurPower(appliance->getPowerChange());
                        }
                }
        }
        dc->SetWindowExt(maxLength,maxWidth);
        dc->SetWindowOrg(resetOrg);
}

//if the wrong ids of the appliances are sent then nothing will
//be changed. appliance->getAppliancesId() == appID[i] test is only for
//consistency.
void Home::setCurPower(int appID[], int status[], float power[], int& arrayPointer){
        POSITION pos1,pos2;
        Room * room;
        int i =0;
        pos1 = appliances.GetHeadPosition();
        while(pos1 != NULL){
                Appliances * appliance = appliances.GetNext(pos1);
                if(appliance->getAppliancesId() == appID[arrayPointer]){
                        if(appliance->setOnOff(status[arrayPointer],power[arrayPointer])){
```

```
                                curPower = curPower + appliance->getPowerChange();
                                pos2 = rooms.GetHeadPosition();
                                while(pos2 != NULL){
                                        room = rooms.GetNext(pos2);
                                        if (appliance->attachedToId == room->getRoomId()){
                                                room->setCurPower(appliance-
>getPowerChange());
                                                break;
                                        }
                                }
                        }
                }
                arrayPointer++;
        }
}

int Home::addRoom(int theArray[],int theNumberOfCorners, COLORREF theClr,CString
theName, CPoint point){
        rooms.AddTail(new Room(count,theArray,theNumberOfCorners,theClr,theName,point));
        roomCount++;
        return 1;
}

int Home::addCloset(int roomId, int theArray[],int theNumberOfCorners, COLORREF theClr){
   POSITION pos = rooms.GetHeadPosition();
        while(pos != NULL){
                Room * r = rooms.GetNext(pos);
                if (r->getRoomId() == roomId ){
                        r->addCloset(count,theArray,theNumberOfCorners,theClr);
                        return 1;
                }
        }
        return 0;
}

int Home::addDoor (int theAttachedToId1,int theAttachedToId2, int X1,int Y1,int X2, int Y2, int
theRotation,
                                        COLORREF theDoorClr){
        doors.AddTail(new Door(count,theAttachedToId1, theAttachedToId2, X1, Y1, X2, Y2,
theRotation, theDoorClr));
        doorCount++;
        return 1;
}
int Home::addFoldingDoor (int theAttachedToId1,int theAttachedToId2, int X1,int Y1,int X2, int
Y2, int theRotation,
```

```
                                                    int theSlidingAngleFromVertical, COLORREF
theFoldingDoorClr){
        foldingDoors.AddTail(new FoldingDoor(count, theAttachedToId1, theAttachedToId2,
X1, Y1, X2, Y2,
                                                    theRotation, theSlidingAngleFromVertical,
theFoldingDoorClr));
        foldingDoorCount++;
        return 1;
}
int Home::addOpening (int theAttachedToId1,int theAttachedToId2, int X1,int Y1,int X2, int Y2,
                                                int theInclination,COLORREF theOpeningClr){
        openings.AddTail(new Opening(count, theAttachedToId1, theAttachedToId2, X1, Y1,
X2, Y2,
                                                theInclination,theOpeningClr));
        openingCount++;
        return 1;
}
int Home::addWindow (int theAttachedToId, int X1,int Y1,int X2, int Y2,
                                                int theInclination,COLORREF theWindowClr){
        windows.AddTail(new Window(count, theAttachedToId, X1, Y1, X2, Y2,
                                                theInclination, theWindowClr));
        windowCount++;
        return 1;
}
int Home::addApplainces (int theAttachedToId, CRect theRect,COLORREF theAppliancesClr,
                                                    float maxPower,enum Appliances::Shape
theShape){
        appliances.AddTail(new Appliances(count, theAttachedToId, theRect, theAppliancesClr,
                                                    maxPower, theShape));
        POSITION pos = rooms.GetHeadPosition();
        while(pos != NULL){
                Room * r = rooms.GetNext(pos);
                if (r->getRoomId() == theAttachedToId )
                        r->setMaxPower(maxPower);
        }
        setMaxPower(maxPower);
        appliancesCount++;
        return 1;
}

void Home::Serialize (CArchive& ar)
{
   CObject::Serialize (ar);
        if (ar.IsStoring ()){
                ar << count;
           ar << homeId;
```

```
        ar << homeClr;
                ar << maxPower;
        ar << curPower;
                ar << powerConsumed;
                ar << selectedRoom;
                ar << maxLength;
                ar << maxWidth;
                ar << timeSlot;
                ar << roomCount;
                ar << doorCount;
                ar << foldingDoorCount;
                ar << openingCount;
                ar << windowCount;
                ar << appliancesCount;
                POSITION pos;
        pos = rooms.GetHeadPosition();
    while(pos != NULL){
                   ar << rooms.GetNext(pos);
                }
                pos = doors.GetHeadPosition();
                while(pos != NULL){
                        ar << doors.GetNext(pos);
                }
                pos = foldingDoors.GetHeadPosition();
                while(pos != NULL){
                        ar << foldingDoors.GetNext(pos);
                }
                pos = openings.GetHeadPosition();
                while(pos != NULL){
                        ar << openings.GetNext(pos);
                }
                pos = windows.GetHeadPosition();
                while(pos != NULL){
                        ar << windows.GetNext(pos);
                }
                pos = appliances.GetHeadPosition();
                while(pos != NULL){
                        ar << appliances.GetNext(pos);
                }
                ar << clicked;
      }
else {// Loading, not storing
                ar >> count;
        ar >> homeId;
        ar >> homeClr;
                ar >> maxPower;
```

```
        ar >> curPower;
            ar >> powerConsumed;
            ar >> selectedRoom;
            ar >> maxLength;
            ar >> maxWidth;
            ar >> timeSlot;
            ar >> roomCount;
            ar >> doorCount;
            ar >> foldingDoorCount;
            ar >> openingCount;
            ar >> windowCount;
            ar >> appliancesCount;
            Room * r;
        for (int i = 0 ; i < roomCount ; i++){
               ar >> r;
                    rooms.AddTail(r);
            }
            Door * d;
            for ( i = 0 ; i < doorCount ; i++){
               ar >> d;
                    doors.AddTail(d);
            }
            FoldingDoor * fd;
            for ( i = 0 ; i < foldingDoorCount ; i++){
               ar >> fd;
                    foldingDoors.AddTail(fd);
            }
            Opening * o;
            for ( i = 0 ; i < openingCount ; i++){
               ar >> o;
                    openings.AddTail(o);
            }
            Window * w;
            for ( i = 0 ; i < windowCount ; i++){
               ar >> w;
                    windows.AddTail(w);
            }
            Appliances * a;
            for ( i = 0 ; i < appliancesCount ; i++){
               ar >> a;
                    appliances.AddTail(a);
            }
            ar >> clicked;
    }
}
```

```
// Lable.cpp: implementation of the CLable class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "FuelCell.h"
#include "Lable.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////
IMPLEMENT_SERIAL (CLable, CObject, 1);
CLable::CLable()
{

}

CLable::~CLable()
{

}

CLable::CLable(CString theName, CPoint point)
{
        position.x = point.x;
        position.y = point.y;
        name = theName;
}
void CLable::Serialize (CArchive& ar)
{
   CObject::Serialize (ar);
        if (ar.IsStoring ()){
                ar << name;
           ar << position;

        }
   else {// Loading, not storing
                ar >> name;
           ar >> position;
        }
```

```
}

void CLable::setXPos(int x)
{
  position.x = x;
}
void CLable::setYPos(int y)
{
  position.y = y;
}
void CLable::setName(CString theName)
{
  name = theName;
}


// MainFrm.cpp : implementation of the CMainFrame class
//
#include <afxwin.h>
#include "stdafx.h"
#include "FuelCell.h"

#include "MainFrm.h"

#include "FuelCellDoc.h"
#include "FuelCellView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
        //{{AFX_MSG_MAP(CMainFrame)
                // NOTE - the ClassWizard will add and remove mapping macros here.
                //    DO NOT EDIT what you see in these blocks of generated code !
        //        ON_WM_SIZE()
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction


CMainFrame::CMainFrame()
{
}

CMainFrame::~CMainFrame()
{

}

/*void CMainFrame::OnSize ( UINT nType, int cx, int cy)
{
        CWnd::OnSize( nType, cx, cy );
        if (  nType == SIZE_MINIMIZED   )
        {
                return;
        }
        //code to call setDC to adjust to new window size, call to
        //deleteOldSemacMsgBoxAndLabel and make a new semacMsgBox
        CFuelCellView * currentView = (CFuelCellView *)GetActiveView();
        if( currentView == NULL )
   {
                return;
        }
        currentView->redrawingOnResezingWnd();
        //CWnd::Invalidate ();
        /*CClientDC pDC (currentView);
        CFuelCellDoc* pDoc = (CFuelCellDoc*)currentView->GetDocument();
   CRect rect;
   currentView->GetClientRect( &rect);
        POSITION pos =  pDoc->homes.GetHeadPosition();
        //while (pos != NULL){  //for  multiple homes not needed at this time
      Home * h = pDoc->homes.GetNext(pos);
                h->setDC(&pDC,&rect);
                h->draw(&pDC);
           h->drawSemacMessage ( currentView, cx, cy);
//      }

}*/


BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
```

```
{
        CWnd * desktop = GetDesktopWindow();
   RECT desktopRect;
   desktop->GetClientRect(&desktopRect );
        cs.cx = desktopRect.right - desktopRect.left;
        cs.cy = desktopRect.bottom - desktopRect.top;

        if( !CFrameWnd::PreCreateWindow(cs) )
                return FALSE;
        // TODO: Modify the Window class or styles here by modifying
        //  the CREATESTRUCT cs

        return TRUE;
}


/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
        CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
        CFrameWnd::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers


//opening.cpp

#include "stdafx.h"
#include "Opening.h"
#include <Math.h>

int Opening::count;
//Definition of class Opening
IMPLEMENT_SERIAL (Opening, CObject, 1);
Opening::Opening(){}
```

```
Opening::~Opening(){
	--count;
}

Opening::Opening(int theHomeId, int theAttachedToId1,int theAttachedToId2, int X1,int Y1,int
X2, int Y2, int theInclination,COLORREF theOpeningClr){
	openingId = ++count;
	homeId = theHomeId;
	attachedToId1 = theAttachedToId1;
	attachedToId2 = theAttachedToId2;
	end1.x = X1;
	end1.y = Y1;
	end2.x = X2;
	end2.y = Y2;
	inclination = theInclination;
	width = distBtPoints();
	openingClr = theOpeningClr;
}
Opening::Opening(const Opening & o){
	openingId = o.openingId;
	homeId = o.homeId;
	attachedToId1 = o.attachedToId2;
	attachedToId1 = o.attachedToId2;
	end1.x = o.end1.x ;
	end1.y = o.end1.y;
	end2.x = o.end2.x;
	end2.y = o.end2.y;
	inclination = o.inclination;
   width = o.width;
	openingClr = o.openingClr;
}
/*Opening * Opening::operator =(Opening *w){
	return w;
}*/

Opening  Opening::operator =(Opening w){
	return w;
}

void Opening::draw(CDC* dc){
	CPen pen(PS_DOT,1,RGB(0,0,0)),*pOldPen;
	CBrush cbrush(openingClr), * pOldBrush;
	pOldPen = dc->SelectObject(&pen);
	pOldBrush = dc->SelectObject(&cbrush);
	dc->Rectangle((int)(end1.x-3*cos(abs(inclination)*3.14/180)),(int)(end1.y-
3*sin(abs(inclination)*3.14/180)),
```

```
                (int)(end2.x
+3*cos(abs(inclination)*3.14/180)),(int)(end2.y+3*sin(abs(inclination)*3.14/180)));
        dc->SelectObject(pOldPen);
        dc->SelectObject(pOldBrush);
}

int Opening::distBtPoints(){

        return (int)sqrt(pow((end1.x-end2.x),2)+pow((end1.y-end2.y),2));
}

void Opening::Serialize (CArchive& ar)
{
   CObject::Serialize (ar);
        if (ar.IsStoring ()){
                ar << count;
                ar << homeId;
                ar << openingId;
           ar << attachedToId1;
                ar << attachedToId2;
           ar << end1;
           ar << end2;
                ar << openingClr;
                ar << inclination;
                ar << width;
        }
   else {// Loading, not storing
                ar >> count;
                ar >> homeId;
                ar >> openingId;
           ar >> attachedToId1;
                ar >> attachedToId2;
           ar >> end1;
           ar >> end2;
                ar >> openingClr;
                ar >> inclination;
                ar >> width;
        }
}


// room.cpp
#include "stdafx.h"
#include "Room.h"
#include <iostream>
using namespace std;
```

```
int Room::count;
//Definition of class Room
IMPLEMENT_SERIAL (Room, CObject, 1);
Room::Room(){}
Room::~Room(){
        --count;
        delete lable;
        POSITION pos = closets.GetHeadPosition();
        while(pos != NULL){
                delete closets.GetNext(pos);
        }
        delete [] corners;
}
Room::Room(int theHomeId, int theArray[],int theNumberOfCorners, COLORREF
theClr,CString name, CPoint point) {

        homeId = theHomeId;
        roomId = ++count;
        numberOfCorners = theNumberOfCorners;
        initializeCorners(theArray);
        roomClr = theClr;
        maxPower =0.f;
        curPower = 0.f;
        powerConsumed = 0.f;
        closetCount = 0;
   roomEnlargedPaintedforfirstTime = true;
        lable = new CLable(name,point);
}
Room::Room(const Room & r){
        numberOfCorners = r.numberOfCorners;
        homeId = r.homeId;
        int index = 0;
   while(index < numberOfCorners){
            corners[index] = r.corners[index];index++;
        }
        roomId = r.roomId;
        roomClr = r.roomClr;
        maxPower = r.maxPower;
        curPower = r.curPower;
        powerConsumed = r.powerConsumed;
        closetCount = r.closetCount;
        roomEnlargedPaintedforfirstTime = r.roomEnlargedPaintedforfirstTime;
        minXY = r.minXY;
        maxXY = r.maxXY;
        lable = r.lable;
        POSITION pos = r.closets.GetHeadPosition();
```

```
		while(pos != NULL){
			Closet * closet = r.closets.GetNext(pos);
			closets.AddTail(closet);
		}
}
Room * Room::operator =(Room *r){
		return r;
}
Room  Room::operator =(Room r){
		return r;
}
void Room::draw(CDC* dc){
		CPen pen(PS_SOLID,5,RGB(0,0,0)), *pOldPen;
		//CBrush cbrush(RGB(248,248,248));
		//CBrush cbrush(RGB(200,200,255));
		//CBrush cbrush(RGB(255,253,235));
		CBrush cbrush(roomClr),*pOldBrush;
		pOldPen = dc->SelectObject(&pen);
		pOldBrush = dc->SelectObject(&cbrush);
   dc->Polygon(corners,numberOfCorners);
		POSITION pos = closets.GetHeadPosition();
		while(pos != NULL){
			closets.GetNext(pos)->draw(dc);
		}
		dc->SelectObject(pOldPen);
		dc->SelectObject(pOldBrush);
		COLORREF oldBkCol = dc->SetBkColor(roomClr);
		dc->TextOut(lable->getPosition().x,lable->getPosition().y, lable->getName());
		dc->SetBkColor(oldBkCol);
}

bool Room::getRoomEnlargedPaintedforfirstTime (){
			return roomEnlargedPaintedforfirstTime;
		}

void Room::setRoomEnlargedPaintedforfirstTime ( bool setValue ){
		Room::roomEnlargedPaintedforfirstTime = setValue;
}

void Room::initializeCorners(int theArray[]){
		try {

			corners = new CPoint[numberOfCorners];
			int counter = 0;
			while( counter < numberOfCorners){
```

```
                        corners[counter] =
CPoint(theArray[counter<<1],theArray[(counter<<1)+1]);
                        counter++;
                }
                minXY.x = INT_MAX;
                minXY.y = INT_MAX;
                maxXY.x = 0;
                maxXY.y = 0;
                for (int i =0 ; i < numberOfCorners ; i++){
                        if(minXY.x > corners[i].x)
                                minXY.x = corners[i].x;
                        if(minXY.y > corners[i].y)
                                minXY.y = corners[i].y;
                }
                for (i =0 ; i < numberOfCorners ; i++){
                        if(maxXY.x < corners[i].x)
                                maxXY.x = corners[i].x;
                        if(maxXY.y < corners[i].y)
                                maxXY.y = corners[i].y;
                }
        }
        catch(bad_alloc bac){
                cout <<"\nFailure to allocate";
        }
}

//reference http://home.earthlink.net/~bobstein/inpoly/
int Room::selectedRoom(CPoint point){

        int xnew,ynew;
   int xold,yold;
   int x1,y1;
   int x2,y2;
   int i;
   int inside=0;

   xold=corners[numberOfCorners-1].x;
   yold=corners[numberOfCorners-1].y;
   for (i=0 ; i < numberOfCorners ; i++) {
       xnew=corners[i].x;
       ynew=corners[i].y;
       if (xnew > xold) {
           x1=xold;
           x2=xnew;
           y1=yold;
           y2=ynew;
```

```
        }
        else {
            x1=xnew;
            x2=xold;
            y1=ynew;
            y2=yold;
        }
        if ((xnew < point.x) == (point.x <= xold)          /* edge "open" at one end */
         && ((long)point.y-(long)y1)*(long)(x2-x1)
          < ((long)y2-(long)y1)*(long)(point.x-x1)) {
            inside=!inside;
        }
        xold=xnew;
        yold=ynew;
    }
    return(inside);
}
/* returns the minXY.x and minXY.y as set by the initializeCorners method. selectOrigin is
   called by various methods of Home class
*/
void Room::SelectOrigin(CPoint *point){
        point->x = minXY.x;
        point->y = minXY.y;
}

int Room::addCloset(int homeId,int theArray[],int theNumberOfCorners, COLORREF theClr){
        closetCount++;
        closets.AddTail(new Closet(homeId,theArray,theNumberOfCorners,theClr));
        return 1;
}

void Room::Serialize (CArchive& ar)
{
   CObject::Serialize (ar);
        if (ar.IsStoring ()){
                ar << count;
                ar << homeId;
                ar << numberOfCorners;
                for (int i = 0 ; i < numberOfCorners ; i++)
                        ar << corners[i];
            ar << roomId;
            ar << roomClr;
                ar << maxPower;
            ar << curPower;
                ar << powerConsumed;
                ar << minXY;
```

```
                ar << maxXY;
                ar << lable;
                ar << closetCount;
                POSITION pos;
            pos = closets.GetHeadPosition();
        while(pos != NULL){
                    ar << closets.GetNext(pos);
                }
        }
    else {// Loading, not storing
                ar >> count;
                ar >> homeId;
                ar >> numberOfCorners;
                try {
                    corners = new CPoint[numberOfCorners];
                        for (int i = 0 ; i < numberOfCorners ; i++)
                                ar >> corners[i];
                }
                catch(bad_alloc bac){
                        cout <<"\nFailure to allocate";
                }
                ar >> roomId;
            ar >> roomClr;
                ar >> maxPower;
            ar >> curPower;
                ar >> powerConsumed;
                ar >> minXY;
                ar >> maxXY;
                ar >> lable;
                ar >> closetCount;
                Closet * c;
            for (int i = 0 ; i < closetCount ; i++){
                    ar >> c;
                        closets.AddTail(c);
                }
        }
}

/*Used to calculate how much a room should be enlarge so that enlarged size in not more than the
  original size of the Home.

  called by Home::draw and Home::setCurPower

  theMaxLength - length of the drawing area set when Home object was initialzed in
            CFuelCellDoc::OnNewDocument
  theMaxWidth  -  Width of the drawing area set when Home object was initialzed in
```

```
            CFuelCellDoc::OnNewDocument

*/
float Room::magnifyingFactor(int theMaxLength, int theMaxWidth)
{
        float factor = 7.0f;
        int xLimit = (int)(.66*theMaxLength), yLimit = (int)(.66*theMaxWidth);
        while (abs(factor) > 1) {
                if (factor*(maxXY.x-minXY.x) < xLimit && factor*(maxXY.y-minXY.y) <
yLimit)
                        return factor;
                else
                        factor = factor - .25f;
        }
        return factor;

}

//semac client.cpp


#include "stdafx.h"

#include "SemacClient.h"
#include "FuelCellDoc.h"
#include "FuelCellView.h"
#include<winsock.h>
#include<iostream.h>


//******Will perform actual communication with semac server

const short semacServerPort = 8989;
SOCKET semacClientSocHandle;
SOCKADDR_IN semacServerStruct;
char * semacServerAddr = "127.0.0.1";

UINT semacClientThread( LPVOID pParam )
{
   short stx = 2;
        short etx = 3;
        short rs  = 30;
   short fs  = 28;
   int loadProfilePacket = 1;
   int managementMessage = 2;
```

```
        int numReturnFromConnect;
        WORD wVersionRequested;
    WSADATA wsaData;
        int err = WSAStartup( wVersionRequested, &wsaData );
    if ( err != 0 ) {
       // Tell the user that we could not find a usable
       // WinSock DLL.
       return 0;
        }
    semacClientSocHandle = socket ( PF_INET, SOCK_STREAM, 0 );
        if ( semacClientSocHandle == INVALID_SOCKET)
        {
                cout << "could not create the socket \n"
                   << WSAGetLastError();
                return -1;
        }
    semacServerStruct.sin_family = PF_INET;
        semacServerStruct.sin_port = htons(semacServerPort);
    semacServerStruct.sin_addr.S_un.S_addr = inet_addr(semacServerAddr);
    numReturnFromConnect = connect ( semacClientSocHandle,
(LPSOCKADDR)&semacServerStruct, sizeof (SOCKADDR) );
    if ( numReturnFromConnect == SOCKET_ERROR ){
                cout << "Couldn't connect socket."
         <<  WSAGetLastError();
                return -1;
        }
    HWND hWnd = (HWND)pParam;
        int counter = 1;
        while(true)
        {
           char  * semacMsgBuff = new char[80];
      cout << "semacMsg="<<sizeof(semacMsgBuff)<<"\n";


           numReturnFromConnect = recv ( semacClientSocHandle, semacMsgBuff,40,0 );
//trying to read 5 bytes for<stx>
           if( numReturnFromConnect == SOCKET_ERROR)
                {
        cout << "error while receiving message"
                                << WSAGetLastError();
                   return -1;
                }
           else
                {
                   //char  temp4;
              //char temp;
```

```
        //temp4 = semacMsgBuff[3];
                        ::PostMessage ( hWnd, MESSAGE_RECEIVED_FROM_SEMAC,
(WPARAM)semacMsgBuff, 0);
                        if( counter % 5 == 0)
                        {
                           ::PostMessage ( hWnd, DEVICE_STATUS_MSG_FROM_SEMAC,
(WPARAM)semacMsgBuff, 0);
                        }

                        //::Sleep(1000);
                    cout<<"bytes read = " << numReturnFromConnect <<"\n";
                    cout<<"Message received form Semac Server = "
                          << semacMsgBuff<<"\n";

                }
                ++counter;
        }//end of while loop
}

//******
//******this function will run in a separate and lesson to messages from semac server
/*
UINT semacClientThread( LPVOID pParam )
{

        ::Sleep(5000);
        int k;
        for( k=0; k <= 200; ++k )
        {
                char * message = new char[50];
                char * counterString = new char[20];
                message = " Hello,Message from Semac message number :";
                itoa(k, counterString, 10);
     HWND hWnd = (HWND)pParam;
                 //::PostMessage ( hWnd, MESSAGE_RECEIVED_FROM_SEMAC,
(WPARAM)"Hello,Message from Semac", 0);

                 ::PostMessage ( hWnd, MESSAGE_RECEIVED_FROM_SEMAC,
(WPARAM)counterString, 0);

           //::Sleep(1000);

   }
        return 0;
} */
```

```
//end of semac client function****************
/*void SemacClient::startSemacClientThread (CFuelCellView * cMainFrameRef)
{
	CWinThread * pThread = AfxBeginThread ( semacClientThread, &cMainFrameRef);
	SemacClinet::cMainFrameRef = cMainFrameRef
}*/

void SemacClient::startSemacClientThread (HWND m_hWnd)
{
	CWinThread * pThread = AfxBeginThread ( semacClientThread, m_hWnd );

}

/*void SemacClient::startSemacClientThread ()
{
	CWinThread * pThread = AfxBeginThread ( semacClientThread, NULL );

}*/

/*void SemacClient::callDrawSemacMessage(char * message)
{

}*/


// stdafx.cpp : source file that includes just the standard includes
//	FuelCell.pch will be the pre-compiled header
//	stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"


// Window.cpp


#include "stdafx.h"
#include "Window.h"
#include <Math.h>

int Window::count;
//Definition of class Window
IMPLEMENT_SERIAL (Window, CObject, 1);
Window::Window(){}
Window::~Window(){
	--count;
}
```

```
Window::Window(int theHomeId, int theAttachedToId, int X1,int Y1,int X2, int Y2, int
theInclination,COLORREF theWindowClr){
        windowId = ++count;
        homeId = theHomeId;
        attachedToId = theAttachedToId;
        end1.x = X1;
        end1.y = Y1;
        end2.x = X2;
        end2.y = Y2;
        inclination = theInclination;
        width = distBtPoints();
        windowClr = theWindowClr;
}
Window::Window(const Window & d){
        windowId = d.windowId;
        homeId = d.homeId;
        attachedToId = d.attachedToId;
        end1.x = d.end1.x ;
        end1.y = d.end1.y;
        end2.x = d.end2.x;
        end2.y = d.end2.y;
        inclination = d.inclination;
   width = d.width;
        windowClr = d.windowClr;
}
/*Window * Window::operator =(Window *w){
        return w;
}*/

Window  Window::operator =(Window w){
        return w;
}

void Window::draw(CDC* dc){
        CPen pen(PS_SOLID,1,RGB(0,0,0)), *pOldPen;
        CBrush cbrush(HS_VERTICAL,RGB(0,0,0)), *pOldBrush;
        COLORREF bkClr =  dc->SetBkColor(windowClr);
        pOldPen = dc->SelectObject(&pen);
        pOldBrush = dc->SelectObject(&cbrush);

        //??????what are the different factors like 3, 3.14/180 is degree to radians conversion
        //???? what is the whole calculation for
        dc->Rectangle((int)(end1.x-3*cos(abs(inclination)*3.14/180)),(int)(end1.y-
3*sin(abs(inclination)*3.14/180)),
```

```
                (int)(end2.x
+3*cos(abs(inclination)*3.14/180)),(int)(end2.y+3*sin(abs(inclination)*3.14/180)));
        dc->SetBkColor(bkClr);
        dc->SelectObject(pOldPen);
        dc->SelectObject(pOldBrush);
}

int Window::distBtPoints(){

        return (int)sqrt(pow((end1.x-end2.x),2)+pow((end1.y-end2.y),2));
}

void Window::Serialize (CArchive& ar)
{
   CObject::Serialize (ar);
        if (ar.IsStoring ()){
                ar << count;
                ar << homeId;
                ar << windowId;
           ar << attachedToId;
           ar << end1;
           ar << end2;
                ar << windowClr;
                ar << inclination;
                ar << width;
        }
   else {// Loading, not storing
                ar >> count;
                ar >> homeId;
                ar >> windowId;
           ar >> attachedToId;
           ar >> end1;
           ar >> end2;
                ar >> windowClr;
                ar >> inclination;
                ar >> width;
        }
}


//////////////////////////////////////////////////////////////////////////

///          HEADER FILES

//////////////////////////////////////////////////////////////////////////
```

```
// ApplianceList.h

#if
!defined(AFX_APPLIANCELIST_H__8847E021_88B8_4AE5_A0FC_21DE5756FEF1__INCL
UDED_)
#define
AFX_APPLIANCELIST_H__8847E021_88B8_4AE5_A0FC_21DE5756FEF1__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// ApplianceList.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CApplianceList window

class CApplianceList : public CStatic
{
// Construction
public:
	CApplianceList();

// Attributes
public:

// Operations
public:

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CApplianceList)
	public:
	virtual BOOL Create(LPCTSTR lpszClassName, LPCTSTR lpszWindowName, DWORD
dwStyle, const RECT& rect, CWnd* pParentWnd, UINT nID, CCreateContext* pContext =
NULL);
	//}}AFX_VIRTUAL

// Implementation
public:
	virtual ~CApplianceList();

	// Generated message map functions
protected:
	//{{AFX_MSG(CApplianceList)
```

```
		// NOTE - the ClassWizard will add and remove member functions here.
	//}}AFX_MSG

	DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif //
!defined(AFX_APPLIANCELIST_H__8847E021_88B8_4AE5_A0FC_21DE5756FEF1__INCL
UDED_)

//Appliances.h

//Class Appliances
#ifndef Appliances_H
#define Appliances_H

class Appliances :public CObject
  {
   DECLARE_SERIAL (Appliances)
   static int count;
	static COLORREF appliancesClrOff;
	int attachedToId;	//This variable specifies whether it is a closet(the Appliances
number
				//to which it is attached) or Appliances(-1).
	enum Shape{
	WASHER,
	DRIER,
	HVAC,
	WATER_HEATER,
	FAN,
	LIGHT,
	HIGH_WATTAGE_OUTLET,
	LOW_WATTAGE_OUTLET,
	EXHAUST_FAN,
	OVEN,
	STOVE,
	DISH_WASHER,
	REFRIGERATOR
	};

protected :
```

```
        //???????? why are drawFan and drawExhaustFan method protected where as rest of draw
method
        //are private

          //draws the fan Shape
          void drawFan(CDC * dc);

          //draws the exhust fan Shape
          void drawExhaustFan(CDC * dc);

        int homeId;
        int appliancesId;
        CRect rect;
        COLORREF appliancesClr;
        float maxPower;
        float powerConsumed;
        int onOff;
        Shape shape;
        float curPower;
        float changeInPower;
public :
          float getPowerChange();

   //Invoked By : Home::draw
        /*Description : Default constructor.
                    Makes an empty(no attributes populate) instance of an Appliance.
              Used by Home::draw to get an instance to call Appliance::drawList method
        */
        Appliances();

        ~Appliances();

        //Invoked by Home::addApplainces
        Appliances(int homeId,int attachedToId, CRect rect,COLORREF appliancesClr,float
maxPower, Shape theShape);

        Appliances(const Appliances &);
        Appliances operator= (Appliances a);
        //Appliances* operator= (Appliances *a);

        /*Invoked by : Home::draw (invoked twice )
          Arguments  : dc (pointer to the Drawing context, which is used to render all graphics)
          Type       : pointer to CDC object
        */
        void draw(CDC* dc);
```

```
        /*Invoked By : Appliances::Appliances(int theHomeId,int theAttachedToId, CRect
theRect,COLORREF theAppliancesClr,float maxPower,enum Appliances::Shape theShape)
          Arguments  : theClr (this encapsulates the RGB object)
          Type       : COLORREF class
        */
        void setColor(COLORREF theClr){appliancesClr = theClr;}

        //not being invoed anywhere
        COLORREF getColor(){return appliancesClr;}

        // changes the color of the appliances to mark an appliance off
        static void setOffColor(COLORREF theClr){appliancesClrOff = theClr;}

        //not being invoked anywhere
        static COLORREF getOffColor(){return appliancesClrOff;}

        //not being invoked anywhere
        float getCurPower();

        //not being invoked anywhere
        void setCurPower();

        //not being invoked anywhere
        void setMaxPower(float thePower){maxPower = thePower;}

        //not being invoked anywhere
        float getMaxPower(){return maxPower;}

        //not being invoked anywhere
  void setPowerConsumed(int timeSlot){powerConsumed += (timeSlot*maxPower/3600);}

        //not being invoked anywhere
        float getPowerConsumed(){return powerConsumed;}

        //not being invoked anywhere
        int getOnOff(){return onOff;}

        /*Called by : Home::setCurPower
        */
        int getAppliancesId(){return appliancesId;}

        /*Logic for setting an aplliance off/on, this function is invoked when appliances are
          clicked on the screan
        */
        int setOnOff(CPoint point);
```

```
        /*Logic for setting an aplliance off/on, this function is invoked when data is read from
          the file
        */
        int setOnOff(int status, float power);

    /*Most probably it is invoked by the CFuelCellDoc::Serialize but exactly how ???????????
        */
        void Serialize (CArchive& ar);

private:
          //sets the size of the monitor screen available
          void setRect(CRect theRect);

          //draws the ligths ie the lamp
          void drawLight(CDC* dc);

          //draws Low wattage power outlet
          void drawLWOutlet(CDC* dc);

          //draws High wattage power outlet
          void drawHWOutlet(CDC* dc);
  public:
         /*description : Draws the legend for the appliances, in the enlarged view of a room
                  this method is only used for drawing legends by creating a default
                                    appliance in home class as the behavior is not dependent on
a particular
                                    instance of appliance ie the behavior of this method is same
for any
                                    Appliance object hence this method shoud be disassociated
with the instances
                                    of Appliances and attach it class so it can be called like
                                    Appliance::drawList(...) and does not require an object for
its invocation,
                                    which is the case  in the current implementation e.g. as in
Home::draw at line
                                    201 and 202.
          Called by : Home::draw */
          void drawList(CDC *dc,int x,int y);
};
#endif


// closet.h
```

```
#ifndef Closet_H
#define Closet_H
#include <afxtempl.h>

/*Does not contain min or max co-ordinate and functionality of finding the origin is contained
  in the Room class
*/
class Closet :public CObject
  {
   DECLARE_SERIAL (Closet)
   static int count;
        CPoint * corners;
        int numberOfCorners;
protected:
        int homeId;
        int closetId;
        void initializeCorners(int theArray[]);
        COLORREF closetClr;

public:
        Closet();
        ~Closet();
        Closet(int homeId, int theArray[],int theNumberOfCorners, COLORREF theClr);
        Closet (const Closet & c);
        Closet* operator= (Closet *c);
        Closet operator= (Closet c);
        operator== (Closet c) const {return (closetId == c.closetId);}

        /*Name        : draw
          Invoked by  : Home::draw
          description : code for rendering graphics for a door
          Arguments   : dc (a pointer to context for drawing.)
          type        : CDC
        */
        void draw(CDC* dc);

        int getClosetId(){return closetId;}
        void setColor(COLORREF theClr){closetClr = theClr;}
        COLORREF getColor(){return closetClr;}
        int getNumberOfCorners(){return numberOfCorners;}

        void Serialize (CArchive& ar);
};
#endif
```

```
// door.h

//To add a door we have to provide hinge position lock     position /
//angle of the door with the vertical axes and direction of rotaion /
//To add a window we have to give information of both ends and angle/
// with vertical line .                                    /
////////////////////////////////////////////////////////////////////

//Class Door
#ifndef DOOR_H
#define DOOR_H

class Door :public CObject
  {
   DECLARE_SERIAL (Door)
   static int count;
public:
        int attachedToId1;      //This variable specifies whether it is a closet(the Door number
        int attachedToId2;      //to which it is attached) or Door(-1).

protected :
        int homeId;
        int doorId;
        CPoint hinge,lock;
        int rotation;
        int width;
        COLORREF doorClr;
public :
        Door();
        ~Door();
        Door(int homeId,int attachedToId1,int attachedToId2, int X1,int Y1,int X2, int Y2, int
theRotation,COLORREF doorClr);
        Door(const Door &);
        Door operator= (Door d);
        //Door* operator= (Door *r);

   /*Name       : draw
          Invoked by  : Home::draw
          description : code for rendering graphics for a door
          Arguments   : dc (a pointer to context for drawing.)
          type        : CDC
        */
        void draw(CDC* dc);

        /*
```

```
            Invoked by : Door::Door(int theHomeId, int theAttachedToId1,int theAttachedToId2, int
X1,
                                    int Y1,int X2, int Y2, int theRotation,COLORREF theDoorClr)
            description : implementation of pythegoras theoram to find the length of the diagonal
                        of a triangle using the length of other 2 sides. But is being used to set
                                    width of the Door ?????????
          */
          int distBtPoints();
          void Serialize (CArchive& ar);
};
#endif


// FoldingDoor.h

//Class Door
#ifndef FOLDINGDOOR_H
#define FOLDINGDOOR_H

class FoldingDoor :public CObject
  {
   DECLARE_SERIAL (FoldingDoor)
   static int count;
public:
          int attachedToId1;       //This variable specifies whether it is a closet(the Door number
          int attachedToId2;       //to which it is attached) or Door(-1).

protected :
          int homeId;
          int foldingDoorId;
          CPoint hinge,lock;
          int rotation;
          int s_a_f_v;          //sliding Angle From Vertical
          int width;
          COLORREF foldingDoorClr;
public :
          FoldingDoor();
          ~FoldingDoor();
          FoldingDoor(int homeId,int attachedToId1,int attachedToId2, int X1,int Y1,int X2, int
Y2, int theRotation,int s_a_f_v, COLORREF doorClr);
          FoldingDoor(const FoldingDoor &);
          FoldingDoor operator= (FoldingDoor d);
          //FoldingDoor* operator= (FoldingDoor *r);

          /*Name       : draw
            Invoked by  : Home::draw
```

```
            description : code for rendering graphics for a folding door
            Arguments   : dc (a pointer to context for drawing.)
            type        : CDC
          */
    void draw(CDC* dc);

          /*
           Invoked by  : FoldingDoor::FoldingDoor(int theHomeId,int theAttachedToId1,
                                    int theAttachedToId2, int X1,int Y1,int X2, int Y2,
                                                                                    int
theRotation,int theSlidingAngleFromVertical,
                                                                                    COLORREF
theFoldingDoorClr)
           description : implementation of pythegoras theoram to find the length of the diagonal
                    of a triangle using the length of other 2 sides. But is being used to set
                                    width of the foldingDoor ?????????
          */
          int distBtPoints();
          void Serialize (CArchive& ar);
};
#endif


// FuelCell.h : main header file for the FUELCELL application
//

#if
!defined(AFX_FUELCELL_H__4D38EBCB_A531_41C9_A093_72B0BEE62C3A__INCLUDE
D_)
#define
AFX_FUELCELL_H__4D38EBCB_A531_41C9_A093_72B0BEE62C3A__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
          #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols

/////////////////////////////////////////////////////////////////////////////
// CFuelCellApp:
// See FuelCell.cpp for the implementation of this class
//
```

```
class CFuelCellApp : public CWinApp
{
public:
	CFuelCellApp();

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CFuelCellApp)
	public:
	virtual BOOL InitInstance();
	//}}AFX_VIRTUAL

// Implementation
	//{{AFX_MSG(CFuelCellApp)
	afx_msg void OnAppAbout();
		// NOTE - the ClassWizard will add and remove member functions here.
		//    DO NOT EDIT what you see in these blocks of generated code !
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif //
!defined(AFX_FUELCELL_H__4D38EBCB_A531_41C9_A093_72B0BEE62C3A__INCLUDE
D_)



// FuelCellDoc.h : interface of the CFuelCellDoc class
//
/////////////////////////////////////////////////////////////////////////////
#if
!defined(AFX_FUELCELLDOC_H__72C5A8B0_2422_4371_9999_983FC65008A0__INCLUD
ED_)
#define
AFX_FUELCELLDOC_H__72C5A8B0_2422_4371_9999_983FC65008A0__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
```

```
#include "Home.h"
class CFuelCellDoc : public CDocument
{
protected: // create from serialization only
	CFuelCellDoc();
	DECLARE_DYNCREATE(CFuelCellDoc)

// Attributes
public:

// Operations
public:

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CFuelCellDoc)
	public:

	//Invoked by automatically generated MFC code when the main window is instantiated
	//Every time the main window is repainted this method is invoked
	virtual BOOL OnNewDocument();

	virtual BOOL OnOpenDocument(LPCTSTR lpszPathName);
	virtual void Serialize(CArchive& ar);
	//}}AFX_VIRTUAL

// Implementation
public:
	CTypedPtrList<CObList,Home*> homes;
	int homeCount;
	int applIDBuff[200];
	int applStatusBuff[200];
	float applPowerBuff[200];
	int arrayPointer;
	int lastInUse;
	void dataFromFile(CDC* dc,CRect *rect);
	virtual ~CFuelCellDoc();
#ifdef _DEBUG
	virtual void AssertValid() const;
	virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
```

```
protected:
	//{{AFX_MSG(CFuelCellDoc)
		// NOTE - the ClassWizard will add and remove member functions here.
		//    DO NOT EDIT what you see in these blocks of generated code !
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif //
!defined(AFX_FUELCELLDOC_H__72C5A8B0_2422_4371_9999_983FC65008A0__INCLUD
ED_)


// FuelCellView.h : interface of the CFuelCellView class
//		Contains methods that respond to various events
/////////////////////////////////////////////////////////////////////////////

#if
!defined(AFX_FUELCELLVIEW_H__BD1DBF65_2B51_4884_A83B_184E8DCE4A85__INC
LUDED_)
#define
AFX_FUELCELLVIEW_H__BD1DBF65_2B51_4884_A83B_184E8DCE4A85__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define MESSAGE_RECEIVED_FROM_SEMAC 0x100
#define DEVICE_STATUS_MSG_FROM_SEMAC 0x150

class CFuelCellView : public CView
{

protected: // create from serialization only
	CFuelCellView();
	DECLARE_DYNCREATE(CFuelCellView)

// Attributes

public:
	CFuelCellDoc* GetDocument();
```

```
    CFuelCellDoc* pDoc;

// Operations
public:

// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CFuelCellView)
        public:
        virtual void OnDraw(CDC* pDC);  // overridden to draw this view
        virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
        virtual void OnInitialUpdate();
        //}}AFX_VIRTUAL

// Implementation
public:
        virtual ~CFuelCellView();
        void CFuelCellView::setSemacMessage(char * SemacMessage);
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
        //{{AFX_MSG(CFuelCellView)

        //CFuelCellView message handlers
        afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);

        //refreshes the time every second on the main window.
        afx_msg void OnTimer(UINT nIDEvent);

        /*OnLButtonDblClk - is called when user double clicks on the "Home". the "homes"
CTypedPtrList object
                is traverresed to find which "Home" was clicked on and then
                Home::selectRoom method of the clicked "Home" is called to find what "room"
                was clicked on inorder to enlarge it
    */

        afx_msg void OnLButtonDblClk(UINT nFlags, CPoint point);
        afx_msg void OnRButtonDown(UINT nFlags, CPoint point);
    //gets called when window is minimized, maximized and restored
        //}}AFX_MSG
```

```
        afx_msg LONG OnSemacThreadMessage ( WPARAM wParam, LPARAM lParam );
        afx_msg LONG OnSemacDeviceStatusMsg ( WPARAM wParam, LPARAM lParam );
        DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in FuelCellView.cpp
inline CFuelCellDoc* CFuelCellView::GetDocument()
   { return (CFuelCellDoc*)m_pDocument; }
#endif

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif //
!defined(AFX_FUELCELLVIEW_H__BD1DBF65_2B51_4884_A83B_184E8DCE4A85__INC
LUDED_)


//Home.h

//Class Home
#ifndef Home_H
#define Home_H
#include <afxtempl.h>
#include "Room.h"
#include "Door.h"
#include "FoldingDoor.h"
#include "Opening.h"
#include "Window.h"
#include "Appliances.h"

class Home :public CObject
  {
    DECLARE_SERIAL (Home)  //???????

    /*keeps track of number of homes, it is a static variable so retains
            it value even when goes out of scope
          */
    static int count;


protected :
        int homeId;
        int timeSlot;
```

```
        int maxLength;
        int maxWidth;
        COLORREF homeClr;
        float maxPower;
        float curPower;
        float powerConsumed;
        int selectedRoom;       //to store the roomId of the selected room
        int lastPowerLabelWidthPos;
        int lastRoomPowerLabelWidthPos;
        CString lastPowerLabel;
        CEdit * semacMsgBox ;
        CEdit * semacLabel ;

        //Room *room;//

        /*description for next 12 lines:
          rooms, doors, foldingDoors, openings, windows, appliances are the various
CTypedPtrList
          (linked list) objects to hold the different objects of type Room, Door, FoldingDoor,
          Opening, Window and Appliance respectively. roomCount, doorCount,
foldingDoorsCount,
          openingCount, windowCount, appliancesCount are used to keep track of number of
object
          held in the various lists(mentioned above) respectively
        */

        CTypedPtrList<CObList,Room*> rooms;
        int roomCount;
   CTypedPtrList<CObList,Door*> doors;
        int doorCount;
   CTypedPtrList<CObList,FoldingDoor*> foldingDoors;
        int foldingDoorCount;
   CTypedPtrList<CObList,Opening*> openings;
        int openingCount;
   CTypedPtrList<CObList,Window*> windows;
        int windowCount;
   CTypedPtrList<CObList,Appliances*> appliances;
        int appliancesCount;

   char * currentSemacMessage ;//= new char[512];

        int clicked; //set to 1 if user double clicked on any room else 0

   bool homeInDefaultView; //set to true every time home view has to
                           //be rendered for the first time i.e.
                          //when comming back from enlarged room view
```

```
                    //is set to true by
                    //CFuelCellView::OnLButtonDblClk
                    //After the house has been rendered once time
                    // it is set to false so that only applicans
                    // are redrawn on change of power not the
                    //whole house

public :

        //default constructor
        Home();
        /*Destructor: frees memory associated with various CTypedPtrList objects like rooms,
doors,
                foldingDoors, openings, windows, appliances
        */
        ~Home();

        /*Description : initializes the Home::maxLength, Home::maxWidth, Home::homeClr,
with values
                passed to it as arguments. Initializes Home::timeSlot, Home::roomCount,
                                Home::doorCount, Home::foldingDoorCount,
Home::openingCount,
                                Home::windowCount, Home::appliancesCount,
Home::clicked, Home::maxPower,
                                Home::curPower, Home::powerConsumed to 0.
                Initializes Home::selectedRoom to -1.
                                Increments Home::count by one
         Invoked By  :  CFuelCellDoc::OnNewDocument
        */
        Home(int theMaxLength, int theMaxWidth, COLORREF theClr);

        /*Copies the all the attributes of the "Home" argumnet to make a new instance of "Home"
all
    attributes including the rooms, doors, appliances, windows etc lists are copied. The homeId
    is also copied but one instances of Fuel Cell can not have 2 Homes with the same homeId
        */
        Home(const Home &);

        //Overloading the "=" operator
        Home operator= (Home d);

        //Home* operator= (Home *r);
        /*
          Name       : setDC

          paramaters : "rect" is a pointer to a CRect object encapsulating the monitor screen
```

"pdc" is a pointer to a CDC object encapsulating the drawing context, used to actually render the graphics.

Called By : CFuelCellView::OnDraw
CFuelCellView::OnTimer
CFuelCellView::OnLButtonDblClk
CFuelCellView::OnRButtonDown

Description:Sets the mapping mode
Sets the window size as specified by the "Home" object.

*/
void setDC(CDC* pdc,CRect *rect);

/*Name : draw
Arguments : dc (poiter to the drawing context)
Type : CDC
Description : 1)Draw the normal size - if "isClicked" 0 , traverse CTypedPtrList (linked list) objects Home::rooms, Home::doors, Home::windows, Home::foldigDoors, Home::openings etc lists and get pointers to various objects they hold and invoke their draw methods, respectively.
2)Draw enlarged view - if "isClicked" 1. traverse the Home::rooms CTypedPtrList (linked list) object to find the Room that was clicked on, change the magnifying factor and call its draw method.
Traverse CTypedPtrList(linked list) objects Home::doors, Home::windows, Home::foldigDoors, Home::openings, Home::appliances etc lists and get pointers to various objects and invoke their draw methods if they are associated with the selected room.

Invoked By : CFuelCellView::OnDraw, CFuelCellView::OnTimer,
*/

```
        void draw(CDC* dc);

    /*Description : Draw the current system time
      Invoked by  : CFuelCellView::OnTimer
            Arguments   : dc ( is a pointer to a CDC object encapsulating the drawing context, used
to
                                        actually render the time)
            Type        : CDC
    */
        void drawClock(CDC* dc);

    /*Description : 1)Draws the horizontal bar at the bottom of main window representing the
                        power, comsumed by the all the appliances in the entire Home.
                                        2)Draws the horizontal bar at the bottom of the enlarged
view of a Room
                                          representing the power consumed by appliances in a
particular room, in
                                          addition of the 1).
      Invoked By  : Home::draw,

        */

    void deleteOldSemacMsgBoxAndLabel();

    void drawSemacMessage( CView * cMainFrameRef);

        void setSemacMessage(char * SemacMEssage);

        void setCurrentSemacMessage( char * SemacMessage);

        void drawGraph(CDC *dc);

      /*selectRoom - is called by CFuelCellView::OnLButtonDblClk and passed the x and y
                co-ordinate encapsulated in CPoint object. selectRoom traverses
                the "rooms" object, a CTypedPtrList datastructure and calls Room::selectedRoom
                passing it "point" object
      */
        int selectRoom(CPoint point);

        //not being invoked anywhere
        void setTimeSlot(int theTimeSlot){timeSlot = theTimeSlot;}

        void Home::setSelectedRoomEnlargedPaintedForFirstTime( bool setValue );

        //not being invoked anywhere
        int getTimeSlot(){return timeSlot;}
```

```
   /*Decription : instantiates a new Room object using
                    Room::Room(int theHomeId, int theArray[],int theNumberOfCorners,
                                                    COLORREF theClr,CString name, CPoint point)
constructor.
                And adds it to the Home::rooms a CTypedPtrList datastructure holding all the
                                                    "Room" objects.
          */
          int addRoom (int theArray[],int theNumberOfCorners, COLORREF theClr,CString
theName, CPoint point);

          /*
            Name      : addCloset
            Description: traverses the "rooms" a CTypedPtrList datastructure holding all the "Room"
                    objects and finds the one with matching roomId and call Room::addCloset
                                    method to add the closet to the room.
       Called By : CFuelCellDoc::OnNewDocument
          */
          int addCloset (int roomId,int theArray[],int theNumberOfCorners, COLORREF theClr);

          /*
            Name      : addDoor
            Description: Instantiates a new "Door" object with the arguments passed to it, adds the
                    object to Home::doors object, which is a CTypedPtrList datastructure holding
                                    all the "Door" objects.
                                    Increments the Home::doorCount by one.
       Called By : CFuelCellDoc::OnNewDocument
          */
          int addDoor (int theAttachedToId1,int theAttachedToId2, int X1,int Y1,int X2, int Y2, int
theRotation,
                              COLORREF theDoorClr);

          /*Name      : addFoldingDoor
            Description: Instantiates a new "foldingDoor" object with the arguments passed to it,
                    adds the object to Home::FoldingDoor object, which is a CTypedPtrList
datastructure holding
                                    all the "foldingDoor" objects.
                                    Increments the Home::foldingDoorCount by one.
       Called By : CFuelCellDoc::OnNewDocument
          */
          int addFoldingDoor (int theAttachedToId1,int theAttachedToId2, int X1,int Y1,int X2, int
Y2,
                                        int theRotation,int theSlidingAngleFromVertical, COLORREF
theFoldingDoorClr);

          /*Name      : addOpening
```

```
        Description: Instantiates a new "Opening" object with the arguments passed to it,
                adds the object to Home::openings object, which is a CTypedPtrList
datastructure
                                holding all the "Opening" objects.
                                Increments the Home::openingCount by one.
    Called By : CFuelCellDoc::OnNewDocument
        */
        int addOpening (int theAttachedToId1,int theAttachedToId2, int X1,int Y1,int X2, int Y2,
                int theInclination,COLORREF theOpeningClr);

        /*Name      : addWindow
         Description: Instantiates a new "Window" object with the arguments passed to it,
                adds the object to Home::windows object, which is a CTypedPtrList
datastructure
                                holding all the "Window" objects.
                                Increments the Home::windowCount by one.
    Called By : CFuelCellDoc::OnNewDocument
        */
        int addWindow (int theAttachedToId, int X1,int Y1,int X2, int Y2,
                int theInclination,COLORREF theWindowClr);

        /*Description : 1)Instantiates a new Appliance using the
                Appliances::Appliances(int theHomeId,int theAttachedToId, CRect theRect,
                                        COLORREF theAppliancesClr,float
maxPower,
                                                                enum
Appliances::Shape theShape)
                                constructor, adds it to the Home::appliances object, which
is a
                                CTypedPtrList datastructure holding all the "Appliance"
objects.
            2)Invokes Room:setMaxPower of the room the appliance is added to
                                3)Invokes Home::setMaxPower
        */
        int addApplainces (int theAttachedToId, CRect theRect,COLORREF theAppliancesClr,
                float maxPower,enum Appliances::Shape theShape);
        int getClicked(){return clicked;}

        /*
    Description : Sets the clicked to the "value" , CFuelCellView::OnLButtonDblClk will invoke
                this method with "1" as parameter when "Home" is double clicked and is being
                                enlarged but CFuelCellView::OnLButtonDblClk will invoke
                this method with "0" as parameter when the enlarged "Room" is double clicked
on

        Called by : CFuelCellView::OnLButtonDblClk
```

```
        */
        void setClicked(int value){clicked = value;}

    void sethomeInDefaultView( bool homeState){ homeInDefaultView = homeState; }

        void setColor(COLORREF theClr){homeClr = theClr;}
        COLORREF getColor(){return homeClr;}

        //not being invoked anywhere
        void setMaxPower(float thePower){maxPower += thePower;}

        //not being invoked anywhere
        float getMaxPower(){return maxPower;}

        /*Called by : CFuelCellView::OnRButtonDown

        */
        void setCurPower(CPoint point, CDC * dc);

        /* Called by : CFuelCellDoc::dataFromFile
        */
        void setCurPower(int appID[], int status[], float thePower[], int & arrayPointer);

        //not being invoked anywhere
        float getCurPower(){return curPower;}

        //not being invoked anywhere
        void setPowerConsumed(int timeSlot){powerConsumed += (timeSlot*curPower/3600);}

        //not being invoked anywhere
        float getPowerConsumed(){return powerConsumed;}

        int getMaxLength(){return Home::maxLength ;}

        int getMaxWidth(){return Home::maxWidth; }

        void Serialize (CArchive& ar);
};
#endif


// Lable.h: interface for the CLable class.
//
//////////////////////////////////////////////////////////////////////
```

```
#if
!defined(AFX_LABLE_H__91EEB525_B123_49D7_B4BF_15E7737A7C88__INCLUDED_)
#define AFX_LABLE_H__91EEB525_B123_49D7_B4BF_15E7737A7C88__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CLable : public CObject
{
	DECLARE_SERIAL (CLable)
public:
	void setXPos(int x);
	void setYPos(int y);
	void setName(CString theName);
	CPoint getPosition(){return position;}
	CString getName(){return name;}
	CLable(CString theName, CPoint point);
	CLable();
	virtual ~CLable();
	void Serialize (CArchive& ar);

private:
	CString name;
	CPoint position;
};

#endif //
!defined(AFX_LABLE_H__91EEB525_B123_49D7_B4BF_15E7737A7C88__INCLUDED_)


// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////

#if
!defined(AFX_MAINFRM_H__64B58001_3B8A_4E70_8F64_9317B4D22C14__INCLUDED_
)
#define AFX_MAINFRM_H__64B58001_3B8A_4E70_8F64_9317B4D22C14__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

class CMainFrame : public CFrameWnd
{
```

```
protected: // create from serialization only
	CMainFrame();
	DECLARE_DYNCREATE(CMainFrame)

// Attributes
public:

// Operations
public:

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CMainFrame)
	public:
	virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
	protected:
	//}}AFX_VIRTUAL

// Implementation
public:
	virtual ~CMainFrame();

#ifdef _DEBUG
	virtual void AssertValid() const;
	virtual void Dump(CDumpContext& dc) const;
#endif

// Generated message map functions
protected:
	//{{AFX_MSG(CMainFrame)
		// NOTE - the ClassWizard will add and remove member functions here.
		//    DO NOT EDIT what you see in these blocks of generated code!
	//gets called when window is minimized, maximized and restored
//	afx_msg void OnSize ( UINT nType, int cx, int cy);
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif //
!defined(AFX_MAINFRM_H__64B58001_3B8A_4E70_8F64_9317B4D22C14__INCLUDED_
)
```

```
// Opening.h

//Class Opening
#ifndef Opening_H
#define Opening_H

class Opening :public CObject
  {
   DECLARE_SERIAL (Opening)
   static int count;
        int attachedToId1;      //This variable specifies whether it is a closet(the Door number
        int attachedToId2;      //to which it is attached) or Door(-1).

protected :
        int homeId;
        int openingId;
        CPoint end1,end2;
        int width;
        int inclination;//angle from vertical
        COLORREF openingClr;
public :
        Opening();
        ~Opening();
        Opening(int homeId,int attachedToId1,int attachedToId2, int X1,int Y1,int X2, int Y2, int
theInclination,COLORREF openingClr);
        Opening(const Opening &);
        Opening operator= (Opening w);
        //Opening* operator= (Opening *w);

        /*Name       : draw
          Invoked by  : Home::draw
          description : code for rendering graphics for a door
          Arguments   : dc (a pointer to context for drawing.)
          type       : CDC
        */
        void draw(CDC* dc);

        /*
         Invoked by : Opening::Opening(int theHomeId, int theAttachedToId1,int
theAttachedToId2,
                                int X1,int Y1,int X2, int Y2, int theInclination,
                                                        COLORREF theOpeningClr)
         description : implementation of pythegoras theoram to find the length of the diagonal
                  of a triangle using the length of other 2 sides. But is being used to set
```

```
                                        width of the Opening ?????????
        */
        int distBtPoints();

        void Serialize (CArchive& ar);
};
#endif


// Resource.h

//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by FUELCELL.RC
//
#define IDD_ABOUTBOX                    100
#define IDR_MAINFRAME                   128
#define IDR_FUELCETYPE                  129

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        130
#define _APS_NEXT_CONTROL_VALUE         1000
#define _APS_NEXT_SYMED_VALUE           101
#define _APS_NEXT_COMMAND_VALUE                 32771
#endif
#endif


// Room.h

#ifndef ROOM_H
#define ROOM_H
#include <afxtempl.h>
#include "Closet.h"
#include "Lable.h"      // Added by ClassView


class Room :public CObject
  {
   DECLARE_SERIAL (Room) //??????
   static int count;
        CTypedPtrList<CObList,Closet*> closets;
        int closetCount;
```

```
        CPoint * corners;
        int numberOfCorners;
protected:
        int homeId;  //Which home the room is associated with
        int roomId;  //An int to number and represent the rooms

   /*initializeCorners - takes theArray the array of co-ordinates of the room as the input
                        and finds minimum x and minimum y cordinate and set maxXY.x and
                        maxXY.y respectively. maxXY.x and maxXY.y will be used to resize
                        the Home on the drawing area.

   */
        void initializeCorners(int theArray[]);
        COLORREF roomClr;
        float maxPower;
        float curPower;
        float powerConsumed;

public:

          /*Description : minXY.x and minXY.y contain the values of the minimum x and y co-
ordinates
                        which is calculated by and set by initializeCorners method. minXY.x and
                                        minXY.y are used while enlarging the room to ensure the
whole drawing
                                        area is not repainted over.
           */
          CPoint minXY;

        Room();
        ~Room();

        /*Description : Room constructor that takes different argument that specify a room and
                        instantiates a room this constructor is called by Home::addRoom

     Arguments   : homeId (a number to associate a "Room" with a particular "Home" i.e. if
                             more than one instance of "Home" is running in a single fuel cell
                        application or on a grid but this is an old requirement and as of
                                                        05/10/03 only one instance of "Home" per
fuel cell application will
                                                        be running)

                                                theArray ( hold Co-ordinates for the room the size of the
                                                        array = 2*theNumberOfCorners)

                theNumberOfCorners ( Specifies the number of corners a room has)
```

theClr (An object encapsulating the RGB object that represents colors)

theName (CString aobject that encapsulates the name displayed on the drawring area used by CLabel)

point (CPoint object used to encapsulate the position co-ordinates that is used by CLabel object to set its postition on the drawing area.)

Type :int, int, int, COLORREF, CString, CPoint

*/

```
Room(int homeId, int theArray[],int theNumberOfCorners, COLORREF theClr,CString theName, CPoint point);

/*Description : Not being invoked anywhere now but is included for future use
        Instantiates a room object by copying the attribute of argument "r",
                (???but one home can not have 2 rooms with same roomId and "r.roomId" is copied
                to make a new instance of "Room" in this constructor ??????)
  Arguments   : r (room object to be copied)
  Type        : Room
*/
Room (const Room & r);
Room* operator= (Room *r);
Room operator= (Room r);
operator== (Room r) const {return (roomId == r.roomId);}

/*Name        : magnifyingFactor
  Description : Used to calculate how much a room should be enlarge so that enlarged size in not
                more than the original size of the Home.
  Arguments   : theMaxLengt (length of the drawing area set when Home object was initialzed in
                CFuelCellDoc::OnNewDocument)

                theMaxWidth (Width of the drawing area set when Home object was initialzed in
                CFuelCellDoc::OnNewDocument)
  Type        : int ,int

  called by   : Home::draw and Home::setCurPower
*/
```

```
        float magnifyingFactor(int theMaxLength, int theMaxWidth);

        /*Name       : draw
          Arguments  : dc (a pointer to context for drawing.)
          type       : CDC
    Description : 1)All the drawing operations related to a "room" takes place here
                  2)Traverses the Room::closets list, gets pointer to the closet objects
                                        and calls their draw method.
          Called by  : Home::draw method
        */
        void draw(CDC* dc);

        /* ?????? implementaion not in room.cpp neither in any other file as of 05/14/03 */
        void drawClosets(CDC *dc);

        /*Name       : getRoomId
    Arguments  : void
          type       : N/A
          Description : returns the Room::roomId attribute.
          Invoked By  : Home::draw, Home::drawGraph, Home::selectRoom, Home::setCurPower
2 times,
                Home::addCloset, Home::addApplainces
        */
        int getRoomId(){return roomId;}

        //Not being invoked any where, as the "roomClr" attribute is set when the "Room" object
is
        //instantiate. But setColor can be used later to set "roomClr" attribute
        void setColor(COLORREF theClr){roomClr = theClr;}

        //Not being invoked any where but might be used in future.
        COLORREF getColor(){return roomClr;}

        /*Name       : SetMaxPower
          Description : increment the power by "thePower", which is the maximum power
consumption
                : of an apppliance being added.
          Argument   : thePower
          type       : float
          Called by  : Home::addApplainces
        */
        void setMaxPower(float thePower){maxPower += thePower;}

        /*Description : returns the maximum power consumption of a room which, is the sum of
the
              maximun power consumptions of various appliances, which is calculted in
```

```
                                        setMaxPower method.
        Called by    : Not called anywhere as of 05/14/03
        */
        float getMaxPower(){return maxPower;}

        /*Called by:Home::setCurPower

        */
        void setCurPower(float thePower){curPower += thePower;}

        //Called by : Home::drawGraph
        float getCurPower(){return curPower;}

        //not being invoked any where
        void setPowerConsumed(int timeSlot){powerConsumed += (timeSlot*curPower/3600);}

//Not being called anywhere
        float getPowerConsumed(){return powerConsumed;}

        //Not being called anywhere
        int getNumberOfCorners(){return numberOfCorners;}

        bool getRoomEnlargedPaintedforfirstTime ();

        void setRoomEnlargedPaintedforfirstTime ( bool setValue );

/* reference http://home.earthlink.net/~bobstein/inpoly/
   Name        : selectedRoom
          Description : take CPoint(encapsulating x and y co-ordinates) as input and finds
              if it is in current instance of room or not. Called by Home::selectRoom
*/
        int selectedRoom(CPoint point);

        /*Called by: Home::draw, Home::setCurPower
        */
        void SelectOrigin(CPoint *point);

        /*Called by: Home::addCloset

        */
        int addCloset (int homeId, int theArray[],int theNumberOfCorners, COLORREF theClr);

        /*Description : used to serialize ie save to a file "Room" object attributes.
                    used also to re-instantiate a Room object, by reading the attributes form
                                        a file.
         */
```

```
        void Serialize (CArchive& ar);

 private:
          CLable *lable;  //encapsulates the name of the Room
          CPoint maxXY;
          CString name;
          bool roomEnlargedPaintedforfirstTime;
};
#endif

//SemacCEdit.h

#ifdef SemacCEdit_H
#define
class SemacCEditv : CEdit
#endif


//SemacClient.h

#ifndef SEMACCLIENT_H
#define SEMACCLIENT_H

void semacClientThread();
//class CFuelCellView;

class SemacClient
{
        public:
                void startSemacClientThread (HWND m_hWnd);
                //void startSemacClientThread( CFuelCellView * cMainFrameRef );
                //void startSemacClientThread();
                //void callDrawSemacMessage(char * message);
};
#endif


//SemacDeviceStatusPacket.h

#ifndef SemacDeviceStatusPacket_H
#define SemacDeviceStatusPacket_H
class SemacDeviceStatusPacket
{
   private:
      int packetType;
      int deviceID;
```

```
        float deviceLoad; // SemcManagementMsgPacket.h

#ifndef SemacManagementMsgPacket_H
#define SemacManagementMsgPacket_H
class SemacManagementMsgPacket
{
   private:
                        int packetType;
            char semacMessage [512];
};
#endif


// SemacRequestPacket.h

#ifndef SemacRequestPacket_H
#define SemacRequestPacket_H
class SemacRequestPacket
{
   private:
                        int packetType;
                        int deviceId;
                        int command;
};
#endif// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//

#if
!defined(AFX_STDAFX_H__4F1937D3_2E32_4623_80D7_1370DBB0D979__INCLUDED_)
#define AFX_STDAFX_H__4F1937D3_2E32_4623_80D7_1370DBB0D979__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define VC_EXTRALEAN            // Exclude rarely-used stuff from Windows headers

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions
#include <afxdtctl.h>           // MFC support for Internet Explorer 4 Common Controls
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                     // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT
```

```
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif //
!defined(AFX_STDAFX_H__4F1937D3_2E32_4623_80D7_1370DBB0D979__INCLUDED_)
        int deviceOverride;
        float hometemp;
        int homeHumdity;
        int homeOccupany;
};
#endif

// Window.h

//Class Window
#ifndef Window_H
#define Window_H

class Window :public CObject
  {
   DECLARE_SERIAL (Window)
   static int count;
          int attachedToId; /*This variable specifies whether it is a closet(the Window number???
please
                        elaborate ??? to which it is attached) or Window(-1). */
protected :
          int homeId;
          int windowId;
          CPoint end1,end2;
          int width;
          int inclination;//angle from vertical
          COLORREF windowClr;
public :
          Window();
          ~Window();
          Window(int homeId,int attachedToId, int X1,int Y1,int X2, int Y2, int
theInclination,COLORREF windowClr);
          Window(const Window &);
          Window operator= (Window w);
          //Window* operator= (Window *w);

          /*??????what are the different factors like 3, 3.14/180 is degree to radians conversion
            ???? what is the whole calculation for
            Name        : draw
            Invoked by  : Home::draw
```

```
            description : code for rendering graphics for a window
            Arguments   : dc (a pointer to context for drawing.)
            type        : CDC
          */
          void draw(CDC* dc);

          /*
           Invoked by : Window::Window(int theHomeId, int theAttachedToId, int X1,int Y1,int
X2,
                                       int Y2, int theInclination,COLORREF theWindowClr)
           description : implementation of pythegoras theoram to find the length of the diagonal
                         of a triangle using the length of other 2 sides. But is being used to set
                                       width of the window ?????????
          */
          int distBtPoints();
          void Serialize (CArchive& ar);
};
#endif
```

APPENDIX E

GIFCO Occupancy Sensor Study

10/27/2003

GIFCO Occupancy Sensor Study

Brett Megginson and Dean Li

1. Introduction

The Grid Independent Fuel Cell Operated Smart Home Project (GIFCO) is a project being conducted by the Department of Energy, in conjunction with the University of South Alabama (USA) and Radiance Technologies, Inc (Radiance). Part of the GIFCO program involves learning human living patterns in order to manage peaks and valleys of typical home electrical usage. Electrical load will be managed by a Smart Energy Management Control System or SEMaC. In order to accomplish its task, SEMaC must monitor room occupancy as stated in the GIFCO Requirements Document in section 3.3.1.3. The information gathered from tracking occupancy will be used to make energy management decisions. For instance, occupancy data will be used to change the management priority of electrical devices causing devices in unoccupied rooms to be managed before devices in occupied rooms.

2. GIFCO Occupancy Sensor Choices

We considered several options to collect occupancy data, including the following:

2.1. Doorframe Entry/Exit Tracker

One option for occupancy detection entailed using two horizontally opposed infrared emitter/detector pairs installed in the door frame at waist height coupled with a control circuitry (see Fig. E1). The order that the two beams were broken indicates either entry or exit and which causes the room occupant count to be incremented or decremented respectively. A problem is that this solution is not stateless. In other words, if power is lost and room occupancy changes prior to power being restored, then the counter would hold the incorrect count until the system is manually corrected.

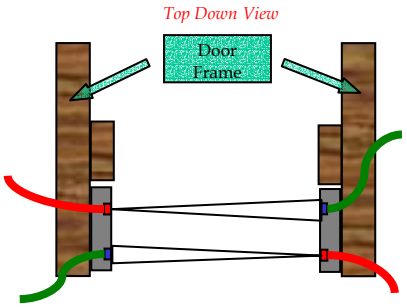


Fig. E1 Proposed Occupancy Detection Unit

2.2. Position Tags

Another option considered was a system using RF transmitting tags, RF receivers and a device to interpret the received signals. The system requires all home occupants to wear the RF tags at all times, and develops room occupancy counts by triangulating the signals from the RF tags. Clearly, one problem with this system is ensuring occupants wear the tags at all times. It is possible that occupants would choose not to wear the tag or simply forget, and then the occupancy data would be inaccurate. Another problem is that the system is both expensive to design and implement.

2.3. Motion Sensor

The third method considered for tracking occupancy was a system using motion detectors placed throughout the house. Motion detectors are inexpensive, stateless and do not require the home user to wear equipment. The problem with using a motion sensor is that significant motion is required for detection. When home occupants are inactive for extended periods of time, motion detectors fail to detect occupancy. We chose to use motion detectors as the basis for detecting occupancy therefore we needed to gain an understanding of its limitations.

3. Motion Detector

The motion detector we chose is a DS9360 TriTech Ceiling Mount PIR/Microwave Intrusion Detector. It is one of the more sensitive units on the market because it uses both infrared and microwave detection technologies. The unit gives visual feedback whenever it senses movement, but both the infrared and microwave sensors must be tripped before the unit alarms. Note that for the purposes of this study, we define “partial alarm” to be movement detection by either the infrared sensor or the microwave sensor, indicated by a green or yellow light. We define a “full alarm” to be movement detection by both sensors indicated by a red light and voltage produced on the motion detector output. We define “at least partial alarm” to be the sum of partial alarms and full alarms.

4. Motion Detector Testing Procedures

We conducted an experiment to gauge the accuracy of the detector. The test was conducted to determine the limitations of the chosen motion sensors. The detector was placed in the middle of a room 13'7" in length, 10'4" in width and 9'6" in height. We separated the room into sections with radial lines spaced 15 degrees and rings at two foot intervals, which produced a total of 65 sections (see Fig. E2).

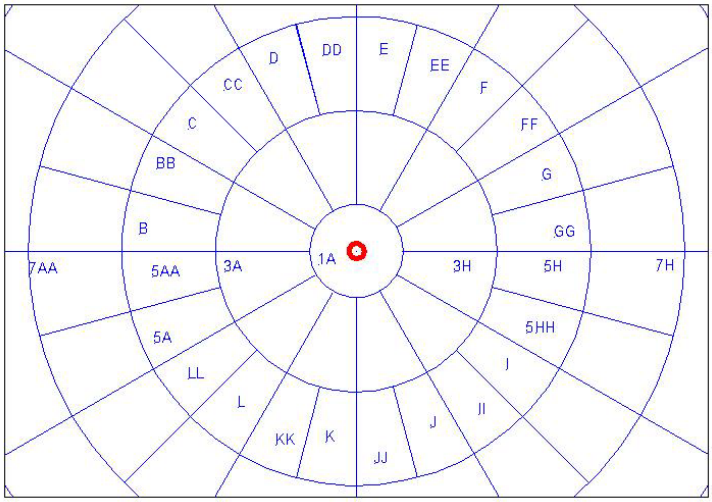


Fig. E2 Floor Layout for Detection Testing

We conducted two separate movement tests in each section. To test small movements, we used hand movement trials and to test larger movements, we used body movement trials. After some informal testing, we decided upon guidelines that would help increase consistency and repeatability. To insure height and distance consistency, we placed a ruler horizontally on the back of a rolling chair, measuring four feet above the floor. We used a stopwatch to insure uniform movement speeds. We moved our hand across the ruler a distance of one foot. The stopwatch was started when movement was initiated and stopped when it ceased. For each trial the chair was placed in the center of the test section. For hand trials we moved only our right hand from right to left at a rate of two feet per second. The body trial consisted of moving our entire body within a section from right to left one foot, at a speed of one foot per second. Any test whose time was two tenths of a second plus or minus the target time was thrown out. We conducted ten trials in each section for each type of movement and recorded whether there was no alarm, partial alarm, or a full alarm.

5. Results

5.1. Results of Detector 1

The test results indicate that the motion detector is significantly more sensitive to movement in certain regions. The raw data are shown in Appendix E1 while the results are grouped in Appendix E2. It is important to note that during the hand motion trials the detector only sensed enough movement to alarm when the movement was located in three closely situated sections as shown in Figure E.1.1[1]. Even with full body movement, the detector only registered a full alarm consistently when the movement was located in a few concentrated areas as seen in Figure E.2.1. The results seem to indicate that this motion detector, when used as designed is not sufficient to indicate occupancy. However, when we look at the partial alarms along with full alarms we see a different picture. Figures E.1.3 and E.2.3, depicting "at least partial alarms," both show a marked increase in detection. Hand movement detection rates increase from 3% to 33% while body movement detection rates increase from 38% to 92%.

The raw data shown in Appendix E1 lists each section's trials in terms of percentage of detection. We calculated each section's area and gave it a corresponding weight. Noting the data for detector 1, the results showed that on average, only about a third of hand movements were indicated by at least one sensor, but 92% of body movements were. The average of the two types of movement is 62%. Full alarms were detected only 3% of the time for hands and 37% for the body. Unweighted and weighted results in each case differed by less than 1%. The unweighted results are close enough to the weighted results that weighting the results does not appear to be important.

5.2. Results of Detector 2

After completing the full round of testing for the first detector, we partially tested the detection rates for our second Bosch sensor of the same model to see if its detection patterns matched our previous result. In testing every other section, the second detector was less accurate with about 20% at least partial detection of hands and about 90% of body, giving it an averaged accuracy of 55%, or 7% less than that of the other sensor. The most sensitive and least sensitive areas did not regularly match; therefore we can not depend on future units to have the same detection patterns. The test results are listed at the end of Appendix E1.

6. Conclusion

After reviewing the results it is clear that relying on the full alarm will not be sufficient for the purposes of detecting occupancy. However, if we use partial alarms, the detection rates will be more acceptable. To do this we added logic circuitry to the detector so that the signal that SEMaC receives will include partial alarms and full alarms. Once the modified detector and the rest of the system have been fully integrated

[1] Figures that have the same number designation were originally meant to be overlapped as transparencies.

into the mock house, we recommend more testing take place to determine if the modifications are sufficient for our purposes of detecting occupancy. We will modify this document to reflect any future findings. Even if the modifications work flawlessly, we realize a 55-63% detection rate misses a significant portion of movement. With the constraints of time, money and complexity, the modified motion sensor is currently the best solution available.

Appendix E1

The following pages are the raw data that was collected during testing.

Trial Record: Hand

Detector 1

Section	% Total Area	% Full Alarm	% Partial Alarm	% At Least Partial	Weighted At Least Partial
1A	2.70	0	100	100	2.70
3A	1.80	0	0	0	0.00
5A	1.80	0	0	0	0.00
7A	2.70	0	0	0	0.00
5AA	1.80	0	30	30	0.54
7AA	2.70	0	20	20	0.54
3B	1.80	0	30	30	0.54
5B	1.80	0	0	0	0.00
7B	2.70	0	0	0	0.00
5BB	1.80	0	30	30	0.54
7BB	2.70	0	40	40	1.08
3C	1.80	0	0	0	0.00
5C	1.80	0	40	40	0.72
7C	2.70	0	20	20	0.54
5CC	1.80	0	30	30	0.54
7CC	n/a	n/a	n/a	n/a	n/a
3D	1.80	0	0	0	0.00
5D	1.80	0	10	10	0.18
7D	n/a	n/a	n/a	n/a	n/a
5DD	1.80	0	10	10	0.18
7DD	n/a	n/a	n/a	n/a	n/a
3E	1.80	0	20	20	0.36
5E	1.80	0	80	80	1.44
7E	n/a	n/a	n/a	n/a	n/a
5EE	1.80	0	50	50	0.90
7EE	n/a	n/a	n/a	n/a	n/a
3F	1.80	0	0	0	0.00
5F	1.80	0	90	90	1.62
7F	n/a	n/a	n/a	n/a	n/a
5FF	1.80	50	40	90	1.62
7FF	2.70	0	80	80	2.16
3G	1.80	0	0	0	0.00
5G	1.80	0	70	70	1.26
7G	2.70	0	10	10	0.27
5GG	1.80	20	70	90	1.62
7GG	2.70	0	0	0	0.00
3H	1.80	0	60	60	1.08
5H	1.80	80	20	100	1.80

Section	% Total Area	% Full Alarm	% Partial Alarm	% At Least Partial	Weighted At Least Partial
7H	2.70	0	90	90	2.43
5HH	1.80	0	10	10	0.18
7HH	2.70	0	100	100	2.70
3I	1.80	0	30	30	0.54
5I	1.80	0	30	30	0.54
7I	2.70	0	30	30	0.81
5II	1.80	0	30	30	0.54
7II	n/a	n/a	n/a	n/a	n/a
3J	1.80	0	10	10	0.18
5J	1.80	0	0	0	0.00
7J	n/a	n/a	n/a	n/a	n/a
5JJ	1.80	0	10	10	0.18
7JJ	n/a	n/a	n/a	n/a	n/a
3K	1.80	0	20	20	0.36
5K	1.80	0	30	30	0.54
7K	n/a	n/a	n/a	n/a	n/a
5KK	1.80	0	50	50	0.90
7KK	n/a	n/a	n/a	n/a	
3L	1.80	0	10	10	0.18
5L	1.80	0	70	70	1.26
7L	n/a	n/a	n/a	n/a	n/a
5LL	1.80	0	0	0	0.00
7LL	2.70	0	0	0	0.00
Totals		150	1470	1620	
Percentages	100.00	3.06	30.00	33.06	33.60

Trial Record: Body

Detector 1

Section	% Total Area	% Full Alarm	% Partial Alarm	% At Least Partial	Weighted At Least Partial
1A	2.70	60	40	100	2.70
3A	1.80	0	90	90	1.62
5A	1.80	50	40	90	1.62
7A	2.70	100	0	100	2.70
5AA	1.80	50	50	100	1.80
7AA	2.70	0	70	70	1.89
3B	1.80	70	30	100	1.80
5B	1.80	70	30	100	1.80
7B	2.70	70	20	90	2.43
5BB	1.80	0	90	90	1.62
7BB	2.70	80	20	100	2.70
3C	1.80	80	20	100	1.80
5C	1.80	60	40	100	1.80
7Q	2.70	30	50	80	2.16
5CC	1.80	30	40	70	1.26
7CC	n/a	n/a	n/a	n/a	n/a
3D	1.80	10	90	100	1.80
5D	1.80	10	50	60	1.08
7D	n/a	n/a	n/a	n/a	n/a
5DD	1.80	0	50	50	0.90
7DD	n/a	n/a	n/a	n/a	n/a
3E	1.80	30	30	60	1.08
5E	1.80	0	100	100	1.80
7E	n/a	n/a	n/a	n/a	n/a
5EE	1.80	50	50	100	1.80
7EE	n/a	n/a	n/a	n/a	n/a
3F	1.80	10	80	90	1.62
5F	1.80	0	100	100	1.80
7F	n/a	n/a	n/a	n/a	n/a
5FF	1.80	100	0	100	1.80
7FF	2.70	100	0	100	2.70
3G	1.80	0	90	90	1.62
5G	1.80	0	100	100	1.80
7G	2.70	60	40	100	2.70
5GG	1.80	30	60	90	1.62
7GG	2.70	0	100	100	2.70
3H	1.80	0	70	70	1.26
5H	1.80	100	0	100	1.80

Section	% Total Area	% Full Alarm	% Partial Alarm	% At Least Partial	Weighted Total
7H	2.70	100	0	100	2.70
5HH	1.80	10	60	70	1.26
7HH	2.70	100	0	100	2.70
3I	1.80	0	100	100	1.80
5I	1.80	20	80	100	1.80
7I	2.70	0	100	100	2.70
5II	1.80	70	30	100	1.80
7II	n/a	n/a	n/a	n/a	n/a
3J	1.80	0	80	80	1.44
5J	1.80	10	90	100	1.80
7J	n/a	n/a	n/a	n/a	n/a
5JJ	1.80	10	70	80	1.44
7JJ	n/a	n/a	n/a	n/a	n/a
3K	1.80	0	100	100	1.80
5K	1.80	0	90	90	1.62
7K	n/a	n/a	n/a	n/a	n/a
5KK	1.80	0	100	100	1.80
7KK	n/a	n/a	n/a	n/a	n/a
3L	1.80	0	90	90	1.62
5L	1.80	80	20	100	1.80
7L	n/a	n/a	n/a	n/a	n/a
5LL	1.80	100	0	100	1.80
7LL	2.70	100	0	100	2.70
Totals		1850	2650	4500	
Percentages	100.00	37.76	54.08	91.84	92.25

Hand and Body Results

Detector 1

Section	% At Least Partial Hand	% At Least Partial Body	% At Least Partial Hand and Body
1A	100	100	100.00
3A	0	90	45.00
5A	0	90	45.00
7A	0	100	50.00
5AA	30	100	65.00
7AA	20	70	45.00
3B	30	100	65.00
5B	0	100	50.00
7B	0	90	45.00
5BB	30	90	60.00
7BB	40	100	70.00
3C	0	100	50.00
5C	40	100	70.00
7Q	20	80	50.00
5CC	30	70	50.00
7CC	n/a	n/a	n/a
3D	0	100	50.00
5D	10	60	35.00
7D	n/a	n/a	n/a
5DD	10	50	30.00
7DD	n/a	n/a	n/a
3E	20	60	40.00
5E	80	100	90.00
7E	n/a	n/a	n/a
5EE	50	100	75.00
7EE	n/a	n/a	n/a
3F	0	90	45.00
5F	90	100	95.00
7F	n/a	n/a	n/a
5FF	90	100	95.00
7FF	80	100	90.00
3G	0	90	45.00
5G	70	100	85.00
7G	10	100	55.00
5GG	90	90	90.00
7GG	0	100	50.00
3H	60	70	65.00
5H	100	100	100.00

Section	% At Least Partial Hand	% At Least Partial Body	% At Least Partial Hand and Body
7H	90	100	95.00
5HH	10	70	40.00
7HH	100	100	100.00
3I	30	100	65.00
5I	30	100	65.00
7I	30	100	65.00
5II	30	100	65.00
7II	n/a	n/a	n/a
3J	10	80	45.00
5J	0	100	50.00
7J	n/a	n/a	n/a
5JJ	10	80	45.00
7JJ	n/a	n/a	n/a
3K	20	100	60.00
5K	30	90	60.00
7K	n/a	n/a	n/a
5KK	50	100	75.00
7KK	n/a	n/a	n/a
3L	10	90	50.00
5L	70	100	85.00
7L	n/a	n/a	n/a
5LL	0	100	50.00
7LL	0	100	50.00
Totals	1620	4500	3060
Percentages	33.06	91.84	62.45

Weighted Hand and Body Results

Section	% Total Area	At Least Partial Hand	At Least Partial Body	At Least Partial Hand and Body
1A	2.70	2.70	2.70	2.70
3A	1.80	0.00	1.62	0.81
5A	1.80	0.00	1.62	0.81
7A	2.70	0.00	2.70	1.35
5AA	1.80	0.54	1.80	1.17
7AA	2.70	0.54	1.89	1.22
3B	1.80	0.54	1.80	1.17
5B	1.80	0.00	1.80	0.90
7B	2.70	0.00	2.43	1.22
5BB	1.80	0.54	1.62	1.08
7BB	2.70	1.08	2.70	1.89
3C	1.80	0.00	1.80	0.90
5C	1.80	0.72	1.80	1.26
7Q	2.70	0.54	2.16	1.35
5CC	1.80	0.54	1.26	0.90
7CC	n/a	n/a	n/a	n/a
3D	1.80	0.00	1.80	0.90
5D	1.80	0.18	1.08	0.63
7D	n/a	n/a	n/a	n/a
5DD	1.80	0.18	0.90	0.54
7DD	n/a	n/a	n/a	n/a
3E	1.80	0.36	1.08	0.72
5E	1.80	1.44	1.80	1.62
7E	n/a	n/a	n/a	n/a
5EE	1.80	0.90	1.80	1.35
7EE	n/a	n/a	n/a	n/a
3F	1.80	0.00	1.62	0.81
5F	1.80	1.62	1.80	1.71
7F	n/a	n/a	n/a	n/a
5FF	1.80	1.62	1.80	1.71
7FF	2.70	2.16	2.70	2.43
3G	1.80	0.00	1.62	0.81
5G	1.80	1.26	1.80	1.53
7G	2.70	0.27	2.70	1.49
5GG	1.80	1.62	1.62	1.62
7GG	2.70	0.00	2.70	1.35
3H	1.80	1.08	1.26	1.17
5H	1.80	1.80	1.80	1.80

Section	% Total Area	At Least Partial Hand	At Least Partial Body	At Least Partial Hand and Body
7H	2.70	2.43	2.70	2.57
5HH	1.80	0.18	1.26	0.72
7HH	2.70	2.70	2.70	2.70
3I	1.80	0.54	1.80	1.17
5I	1.80	0.54	1.80	1.17
7I	2.70	0.81	2.70	1.76
5II	1.80	0.54	1.80	1.17
7II	n/a	n/a	n/a	n/a
3J	1.80	0.18	1.44	0.81
5J	1.80	0.00	1.80	0.90
7J	n/a	n/a	n/a	n/a
5JJ	1.80	0.18	1.44	0.81
7JJ	n/a	n/a	n/a	n/a
3K	1.80	0.36	1.80	1.08
5K	1.80	0.54	1.62	1.08
7K	n/a	n/a	n/a	n/a
5KK	1.80	0.90	1.80	1.35
7KK	n/a	n/a	n/a	n/a
3L	1.80	0.18	1.62	0.90
5L	1.80	1.26	1.80	1.53
7L	n/a	n/a	n/a	n/a
5LL	1.80	0.00	1.80	0.90
Totals				
Percentages	100.00	33.60	92.25	62.93

Trial Record: Hand

Section	% Total Area	% Full Alarm	% Partial Alarm	% At Least Partial	Weighted At Least Partial
1A	5.26	20	80	100	5.26
3A	3.51	0	0	0	0.00
5A	3.51	0	20	20	0.70
7A	n/a	n/a	n/a	n/a	n/a
5AA	n/a	n/a	n/a	n/a	n/a
7AA	5.26	0	0	0	0.00
3B	n/a	n/a	n/a	n/a	n/a
5B	3.51	0	0	0	0.00
7B	n/a	n/a	n/a	n/a	n/a
5BB	n/a	n/a	n/a	n/a	n/a
7BB	5.26	0	10	10	0.53
3C	3.51	0	10	10	0.35
5C	3.51	10	20	30	1.05
7Q	n/a	n/a	n/a	n/a	n/a
5CC	n/a	n/a	n/a	n/a	n/a
7CC	n/a	n/a	n/a	n/a	n/a
3D	n/a	n/a	n/a	n/a	n/a
5D	3.51	0	10	10	0.35
7D	n/a	n/a	n/a	n/a	n/a
5DD	n/a	n/a	n/a	n/a	n/a
7DD	n/a	n/a	n/a	n/a	n/a
3E	3.51	0	0	0	0.00
5E	3.51	0	10	10	0.35
7E	n/a	n/a	n/a	n/a	n/a
5EE	n/a	n/a	n/a	n/a	n/a
7EE	n/a	n/a	n/a	n/a	n/a
3F	n/a	n/a	n/a	n/a	n/a
5F	3.51	0	90	90	3.16
7F	n/a	n/a	n/a	n/a	n/a
5FF	n/a	n/a	n/a	n/a	n/a
7FF	n/a	n/a	n/a	n/a	n/a
3G	3.51	0	20	20	0.70
5G	3.51	0	0	0	0.00
7G	5.26	0	0	0	0.00
5GG	n/a	n/a	n/a	n/a	n/a
7GG	n/a	n/a	n/a	n/a	n/a
3H	n/a	n/a	n/a	n/a	n/a
5H	3.51	0	0	0	0.00

Section	% Total Area	% Full Alarm	% Partial Alarm	% At Least Partial	Weighted At Least Partial
7H	5.26	0	10	10	0.53
5HH	n/a	n/a	n/a	n/a	n/a
7HH	n/a	n/a	n/a	n/a	n/a
3I	3.51	0	10	10	0.35
5I	3.51	0	10	10	0.35
7I	5.26	0	0	0	0.00
5II	n/a	n/a	n/a	n/a	n/a
7II	n/a	n/a	n/a	n/a	n/a
3J	n/a	n/a	n/a	n/a	n/a
5J	3.51	0	0	0	0.00
7J	n/a	n/a	n/a	n/a	n/a
5JJ	n/a	n/a	n/a	n/a	n/a
7JJ	n/a	n/a	n/a	n/a	n/a
3K	3.51	0	0	0	0.00
5K	3.51	0	20	20	0.70
7K	n/a	n/a	n/a	n/a	n/a
5KK	n/a	n/a	n/a	n/a	n/a
7KK	n/a	n/a	n/a	n/a	n/a
3L	n/a	n/a	n/a	n/a	n/a
5L	3.51	0	70	70	2.46
7L	n/a	n/a	n/a	n/a	n/a
5LL	n/a	n/a	n/a	n/a	n/a
7LL	5.26	0	60	60	3.16
Totals		30	450	480	
Percentages	100.00	1.20	18.00	19.20	20.00

Trial Record: Body

Detector 2

Section	% Total Area	% Full Alarm	% Partial Alarm	% At Least Partial	Weighted At Least Partial
1A	5.26	60	40	100	5.26
3A	3.51	0	70	70	2.46
5A	3.51	0	90	90	3.16
7A	n/a	n/a	n/a	n/a	n/a
5AA	n/a	n/a	n/a	n/a	n/a
7AA	5.26	30	40	70	3.68
3B	n/a	n/a	n/a	n/a	n/a
5B	3.51	90	10	100	3.51
7B	n/a	n/a	n/a	n/a	n/a
5BB	n/a	n/a	n/a	n/a	n/a
7BB	5.26	100	0	100	5.26
3C	3.51	0	80	80	2.81
5C	3.51	0	90	90	3.16
7Q	n/a	n/a	n/a	n/a	n/a
5CC	n/a	n/a	n/a	n/a	n/a
7CC	n/a	n/a	n/a	n/a	n/a
3D	n/a	n/a	n/a	n/a	n/a
5D	3.51	30	70	100	3.51
7D	n/a	n/a	n/a	n/a	n/a
5DD	n/a	n/a	n/a	n/a	n/a
7DD	n/a	n/a	n/a	n/a	n/a
3E	3.51	0	80	80	2.81
5E	3.51	50	50	100	3.51
7E	n/a	n/a	n/a	n/a	n/a
5EE	n/a	n/a	n/a	n/a	n/a
7EE	n/a	n/a	n/a	n/a	n/a
3F	n/a	n/a	n/a	n/a	n/a
5F	3.51	0	90	90	3.16
7F	n/a	n/a	n/a	n/a	n/a
5FF	n/a	n/a	n/a	n/a	n/a
7FF	n/a	n/a	n/a	n/a	n/a
3G	3.51	0	60	60	2.11
5G	3.51	0	90	90	3.16
7G	5.26	0	90	90	4.74
5GG	n/a	n/a	n/a	n/a	n/a
7GG	n/a	n/a	n/a	n/a	n/a
3H	n/a	n/a	n/a	n/a	n/a
5H	3.51	20	50	70	2.46

Section	% Total Area	% Full Alarm	% Partial Alarm	% At Least Partial	Weighted At Least Partial
7H	5.26	0	100	100	5.26
5HH	n/a	n/a	n/a	n/a	n/a
7HH	n/a	n/a	n/a	n/a	n/a
3I	3.51	0	90	90	3.16
5I	3.51	40	50	90	3.16
7I	5.26	60	20	80	4.21
5II	n/a	n/a	n/a	n/a	n/a
7II	n/a	n/a	n/a	n/a	n/a
3J	n/a	n/a	n/a	n/a	n/a
5J	3.51	0	100	100	3.51
7J	n/a	n/a	n/a	n/a	n/a
5JJ	n/a	n/a	n/a	n/a	n/a
7JJ	n/a	n/a	n/a	n/a	n/a
3K	3.51	0	100	100	3.51
5K	3.51	80	20	100	3.51
7K	n/a	n/a	n/a	n/a	n/a
5KK	n/a	n/a	n/a	n/a	n/a
7KK	n/a	n/a	n/a	n/a	n/a
3L	n/a	n/a	n/a	n/a	n/a
5L	3.51	100	0	100	3.51
7L	n/a	n/a	n/a	n/a	n/a
5LL	n/a	n/a	n/a	n/a	n/a
7LL	5.26	100	0	100	5.26
Totals		760	1480	2240	
Percentages	100.00	30.40	59.20	89.60	89.82

Hand and Body Results

Detector 2

Section	% At Least Partial Hand	% At Least Partial Body	% At Least Partial Hand and Body
1A	100	100	100.00
3A	0	70	35.00
5A	20	90	55.00
7A	n/a	n/a	n/a
5AA	n/a	n/a	n/a
7AA	0	70	35.00
3B	n/a	n/a	n/a
5B	0	100	50.00
7B	n/a	n/a	n/a
5BB	n/a	n/a	n/a
7BB	10	100	55.00
3C	10	80	45.00
5C	30	90	60.00
7Q	n/a	n/a	n/a
5CC	n/a	n/a	n/a
7CC	n/a	n/a	n/a
3D	n/a	n/a	n/a
5D	10	100	55.00
7D	n/a	n/a	n/a
5DD	n/a	n/a	n/a
7DD	n/a	n/a	n/a
3E	0	80	40.00
5E	10	100	55.00
7E	n/a	n/a	n/a
5EE	n/a	n/a	n/a
7EE	n/a	n/a	n/a
3F	n/a	n/a	n/a
5F	90	90	90.00
7F	n/a	n/a	n/a
5FF	n/a	n/a	n/a
7FF	n/a	n/a	n/a
3G	20	60	40.00
5G	0	90	45.00
7G	0	90	45.00
5GG	n/a	n/a	n/a
7GG	n/a	n/a	n/a
3H	n/a	n/a	n/a
5H	0	70	35.00

Section	% At Least Partial Hand	% At Least Partial Body	% At Least Partial Hand and Body
7H	10	100	55.00
5HH	n/a	n/a	n/a
7HH	n/a	n/a	n/a
3I	10	90	50.00
5I	10	90	50.00
7I	0	80	40.00
5II	n/a	n/a	n/a
7II	n/a	n/a	n/a
3J	n/a	n/a	n/a
5J	0	100	50.00
7J	n/a	n/a	n/a
5JJ	n/a	n/a	n/a
7JJ	n/a	n/a	n/a
3K	0	100	50.00
5K	20	100	60.00
7K	n/a	n/a	n/a
5KK	n/a	n/a	n/a
7KK	n/a	n/a	n/a
3L	n/a	n/a	n/a
5L	70	100	85.00
7L	n/a	n/a	n/a
5LL	n/a	n/a	n/a
7LL	60	100	80.00
Totals	480	2240	1360
Percentages	19.20	89.60	54.40

Hand and Body Results

Detector 2

Section	% Total Area	At least Partial Hand	At Least Partial Body	At Least Partial Hand and Body
1A	5.26	5.26	5.26	5.26
3A	3.51	0.00	2.46	1.23
5A	3.51	0.70	3.16	1.93
7A	n/a	n/a	n/a	n/a
5AA	n/a	n/a	n/a	n/a
7AA	5.26	0.00	3.68	1.84
3B	n/a	n/a	n/a	n/a
5B	3.51	0.00	3.51	1.75
7B	n/a	n/a	n/a	n/a
5BB	n/a	n/a	n/a	n/a
7BB	5.26	0.53	5.26	2.89
3C	3.51	0.35	2.81	1.58
5C	3.51	1.05	3.16	2.11
7Q	n/a	n/a	n/a	n/a
5CC	n/a	n/a	n/a	n/a
7CC	n/a	n/a	n/a	n/a
3D	n/a	n/a	n/a	n/a
5D	3.51	0.35	3.51	1.93
7D	n/a	n/a	n/a	n/a
5DD	n/a	n/a	n/a	n/a
7DD	n/a	n/a	n/a	n/a
3E	3.51	0.00	2.81	1.40
5E	3.51	0.35	3.51	1.93
7E	n/a	n/a	n/a	n/a
5EE	n/a	n/a	n/a	n/a
7EE	n/a	n/a	n/a	n/a
3F	n/a	n/a	n/a	n/a
5F	3.51	3.16	3.16	3.16
7F	n/a	n/a	n/a	n/a
5FF	n/a	n/a	n/a	n/a
7FF	n/a	n/a	n/a	n/a
3G	3.51	0.70	2.11	1.40
5G	3.51	0.00	3.16	1.58
7G	5.26	0.00	4.74	2.37
5GG	n/a	n/a	n/a	n/a
7GG	n/a	n/a	n/a	n/a
3H	n/a	n/a	n/a	n/a
5H	3.51	0.00	2.46	1.23

Section	% Total Area	At least Partial Hand	At Least Partial Body	At Least Partial Hand and Body
7H	5.26	0.53	5.26	2.89
5HH	n/a	n/a	n/a	n/a
7HH	n/a	n/a	n/a	n/a
3I	3.51	0.35	3.16	1.75
5I	3.51	0.35	3.16	1.75
7I	5.26	0.00	4.21	2.11
5II	n/a	n/a	n/a	n/a
7II	n/a	n/a	n/a	n/a
3J	n/a	n/a	n/a	n/a
5J	3.51	0.00	3.51	1.75
7J	n/a	n/a	n/a	n/a
5JJ	n/a	n/a	n/a	n/a
7JJ	n/a	n/a	n/a	n/a
3K	3.51	0.00	3.51	1.75
5K	3.51	0.70	3.51	2.11
7K	n/a	n/a	n/a	n/a
5KK	n/a	n/a	n/a	n/a
7KK	n/a	n/a	n/a	n/a
3L	n/a	n/a	n/a	n/a
5L	3.51	2.46	3.51	2.98
7L	n/a	n/a	n/a	n/a
5LL	n/a	n/a	n/a	n/a
Percentages	100.00	20.00	89.82	54.91

Appendix E2

The following pages show the floor layout of the room where the testing took place. During testing, the floor was taped off so that different sections would have easy visual identification. As noted above, the figures with the same designmation were originally transparencies meant to be layered on top of each other.

Hand: Full Alarm

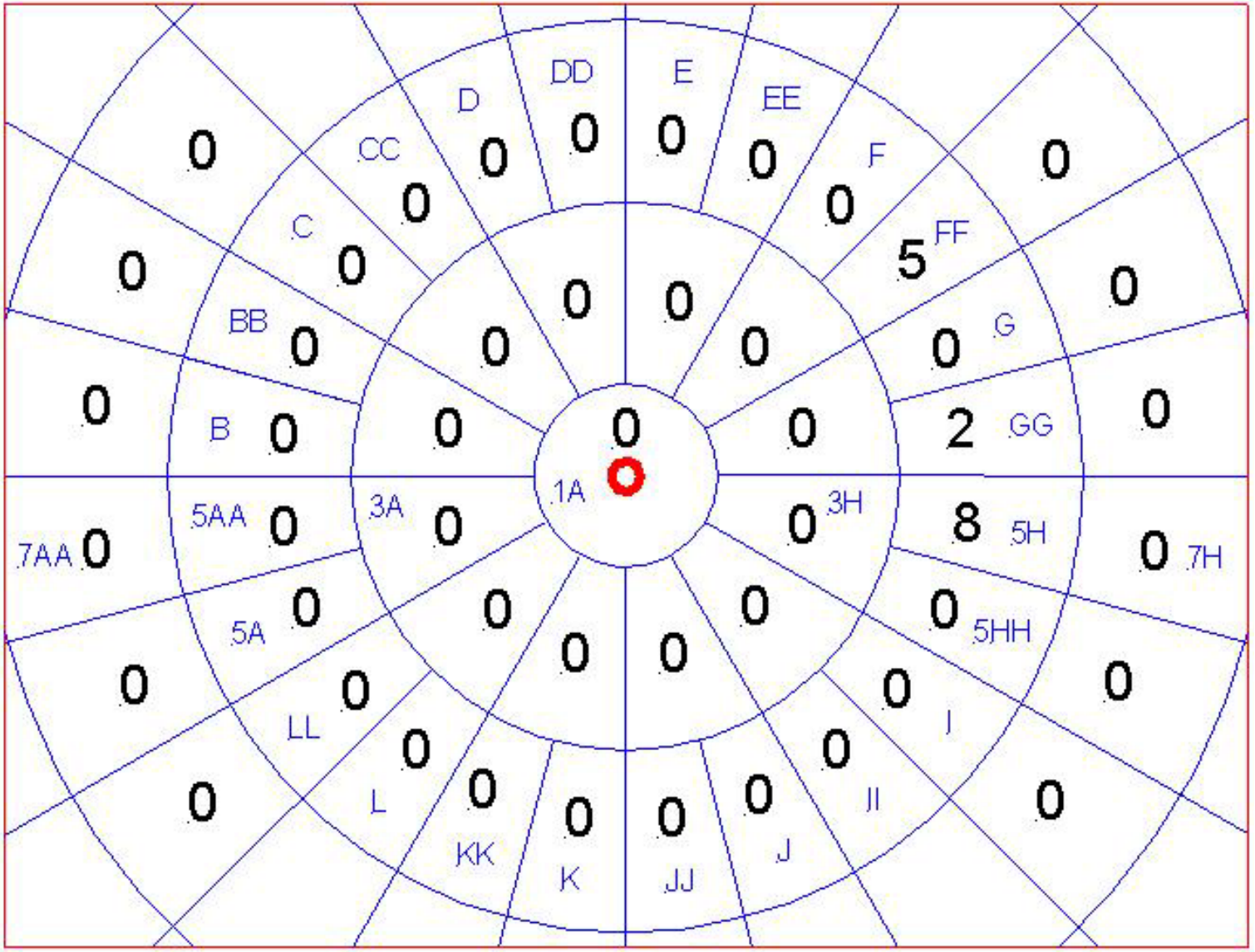


Figure E1.1

Hand: Full Alarm

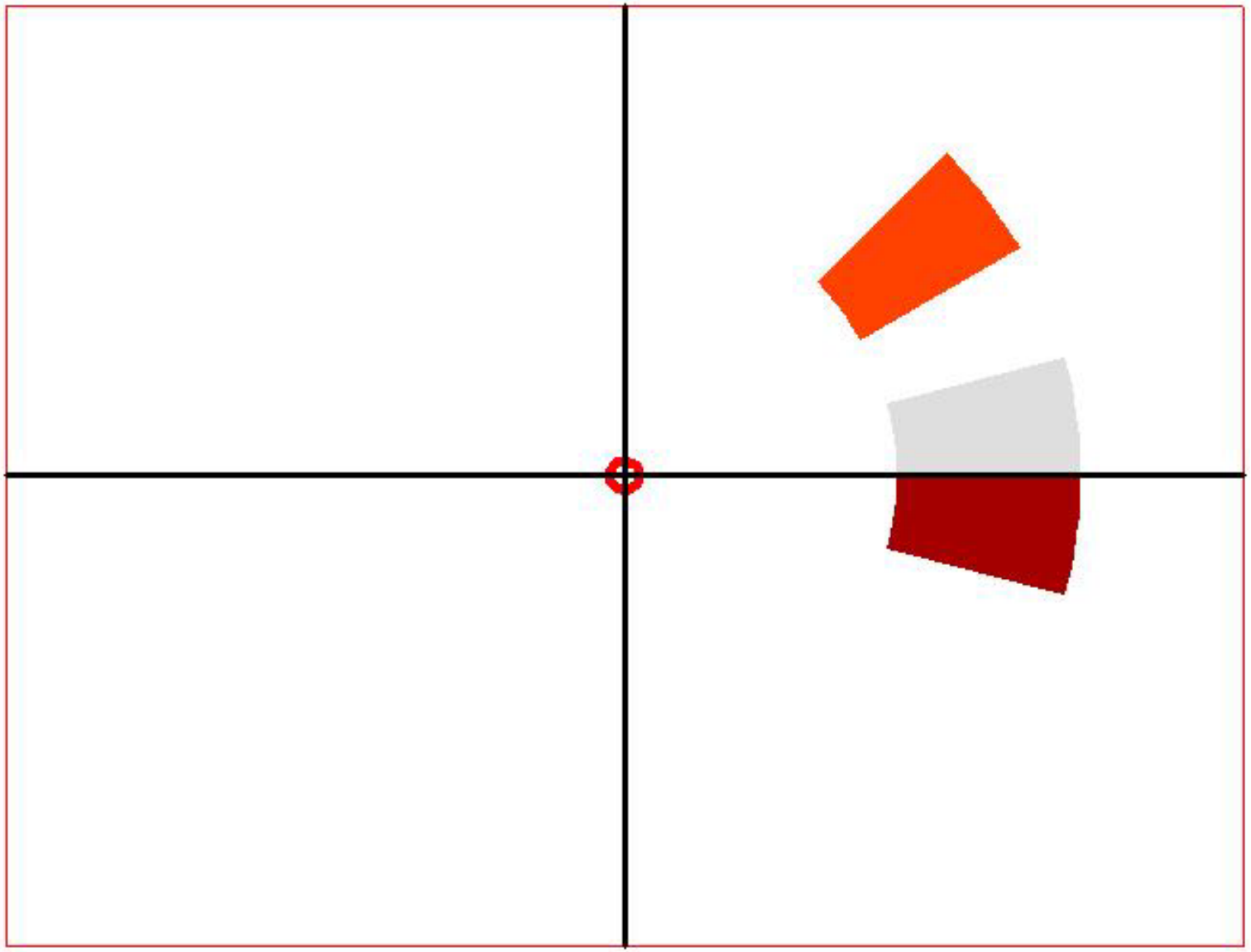


Figure E1.1

Hand: Partial Alarm

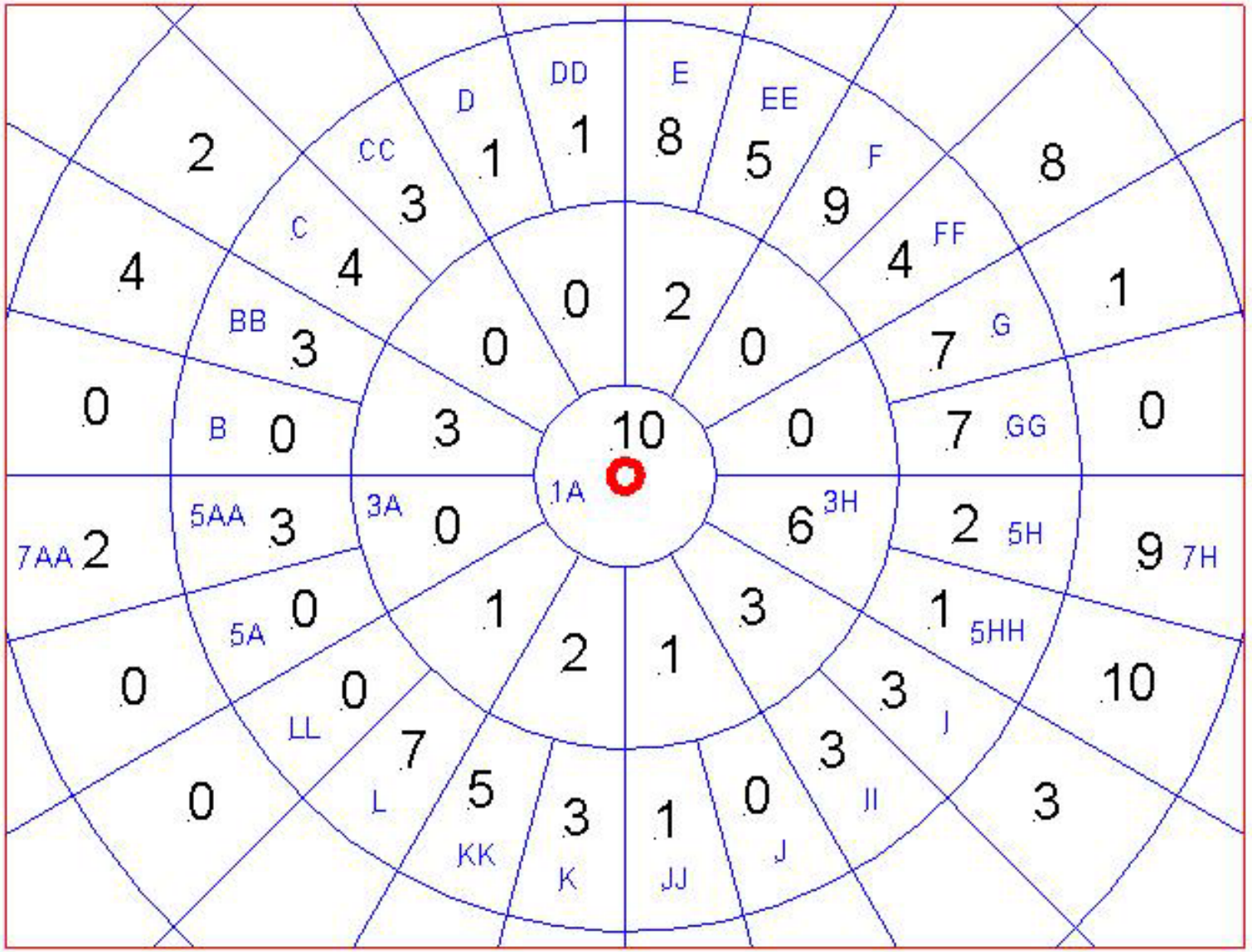


Figure E1.2

Hand: Partial Alarm

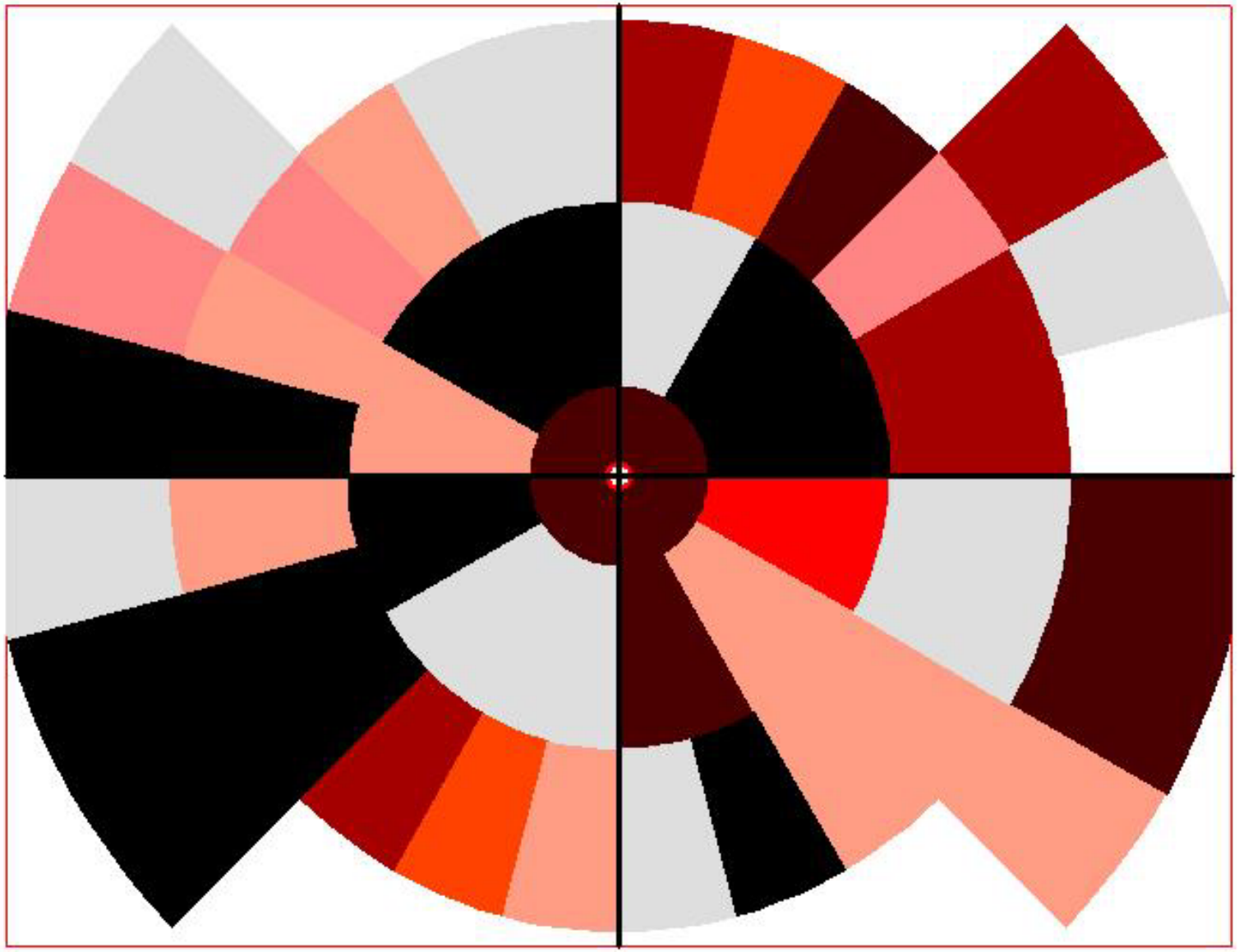


Figure E1.2

Hand: At Least Partial Alarm

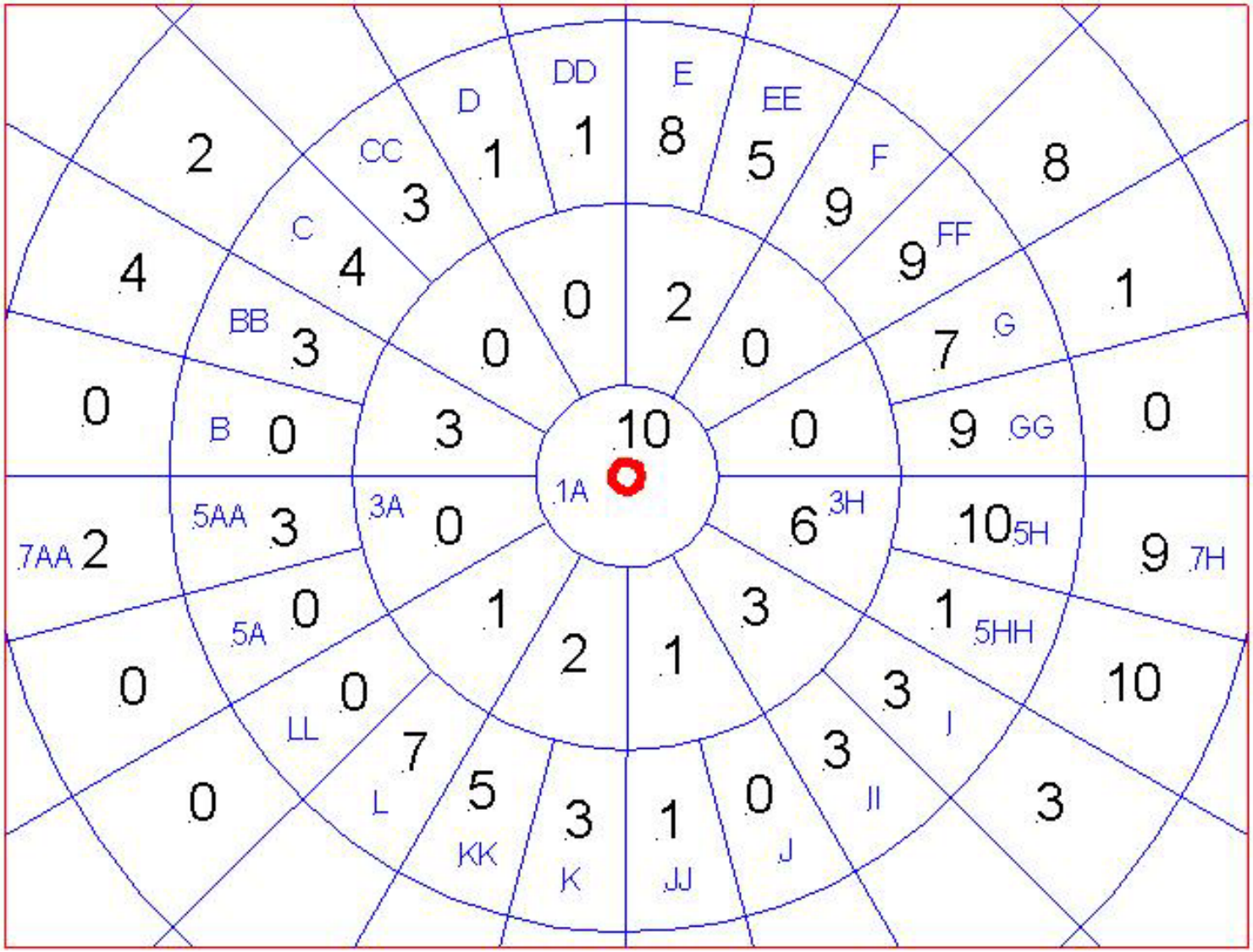


Figure E1.3

Hand: At Least Partial Alarm

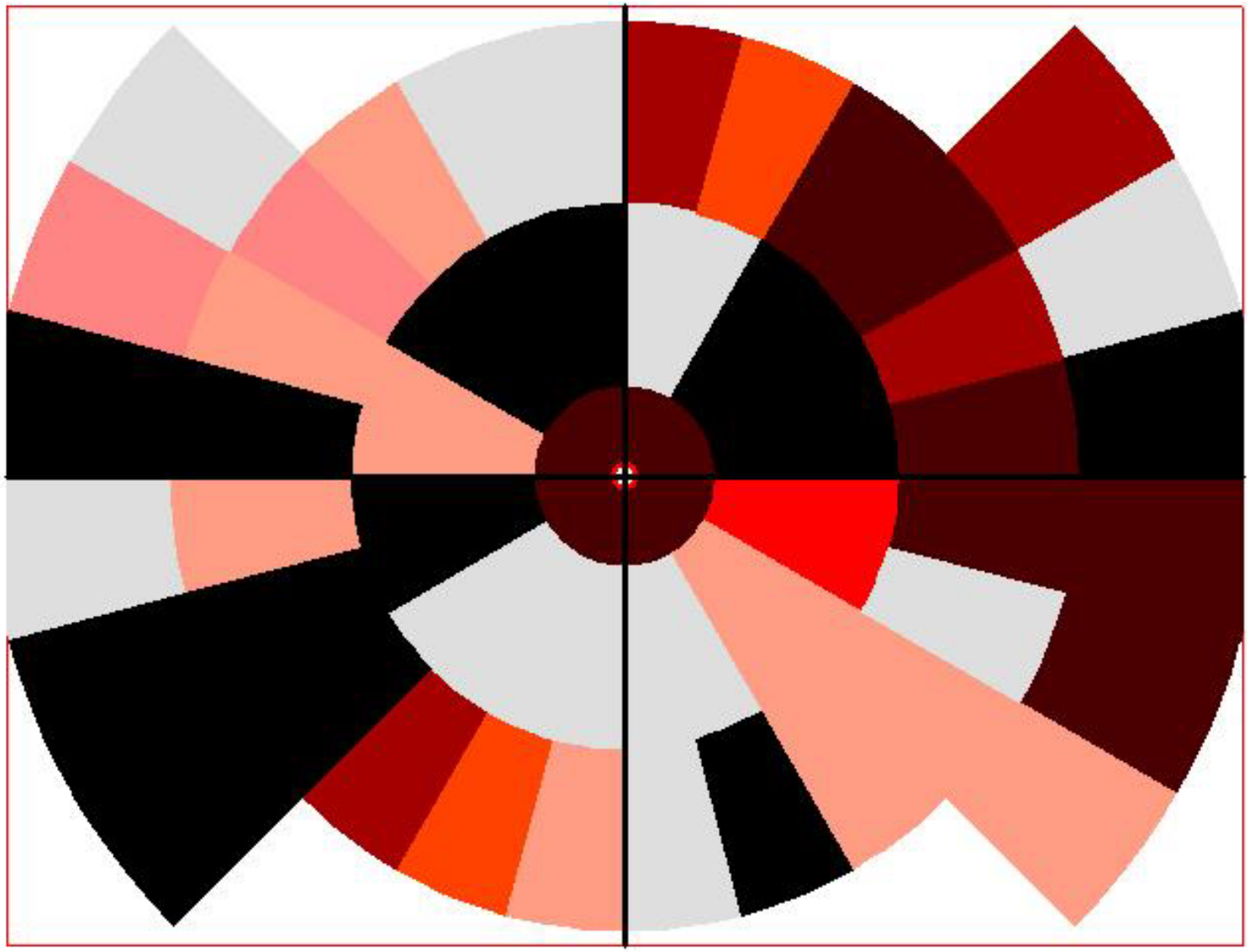


Figure E1.3

Body: Full Alarm

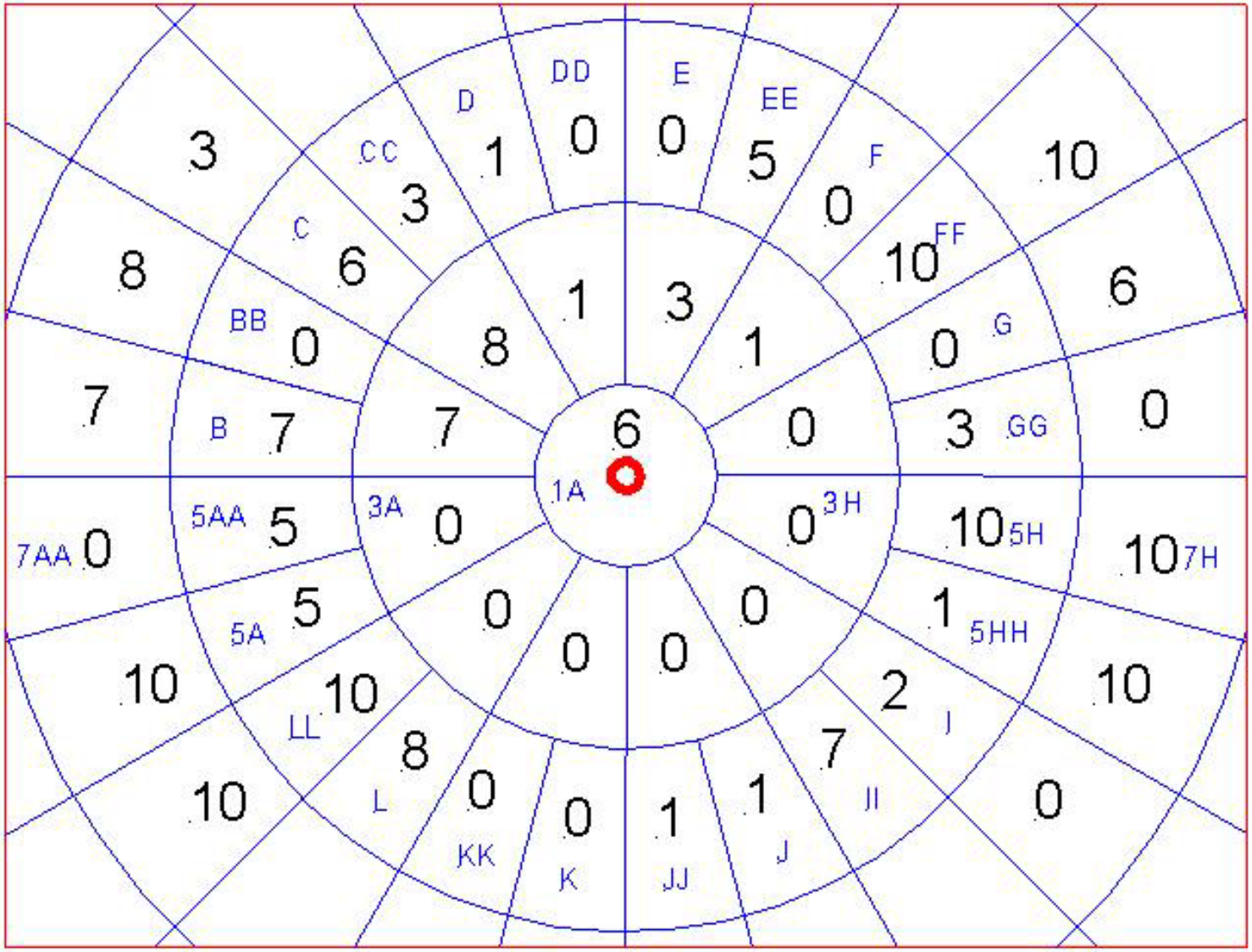


Figure E2.1

Body: Full Alarm

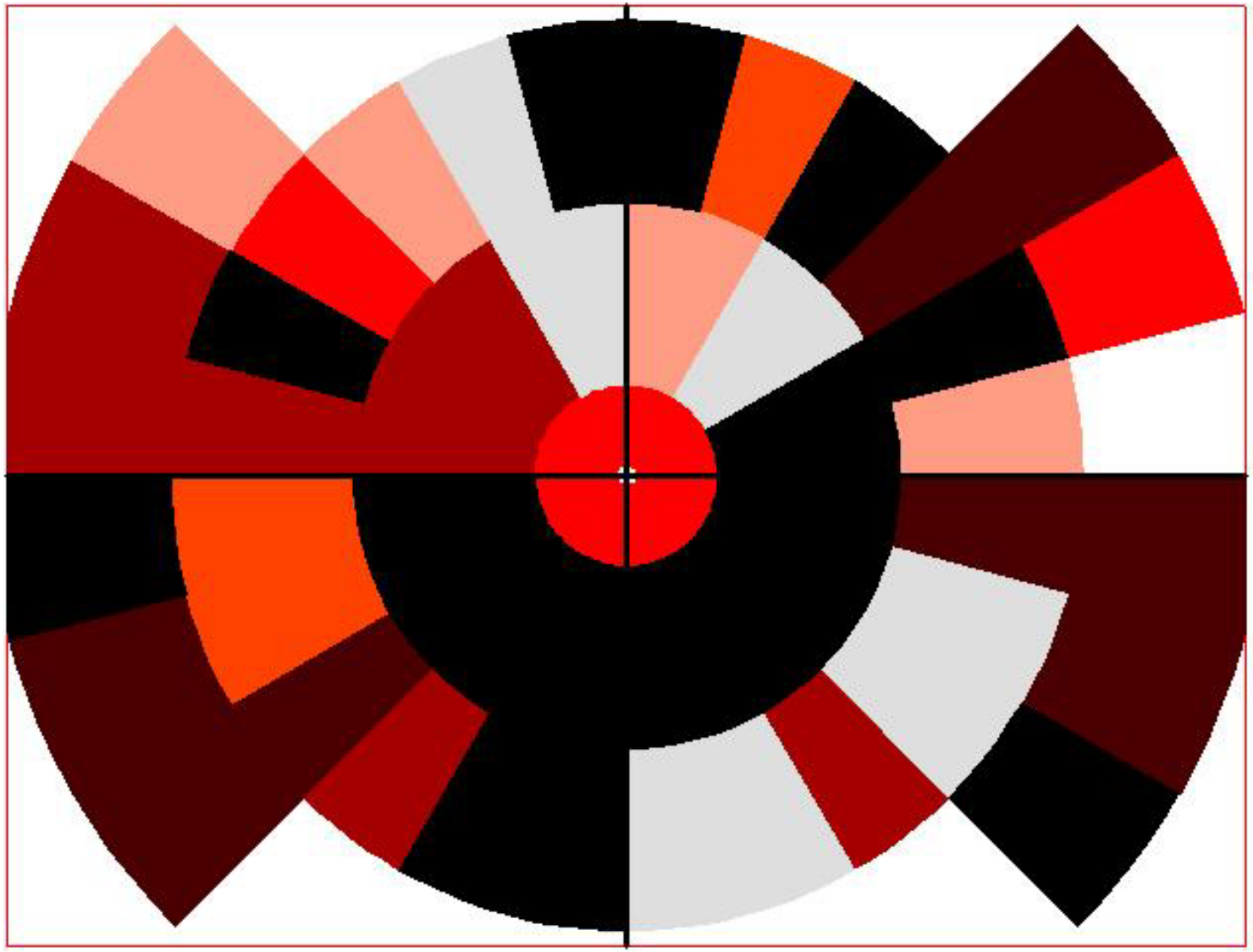


Figure E2.1

Body: Partial Alarm

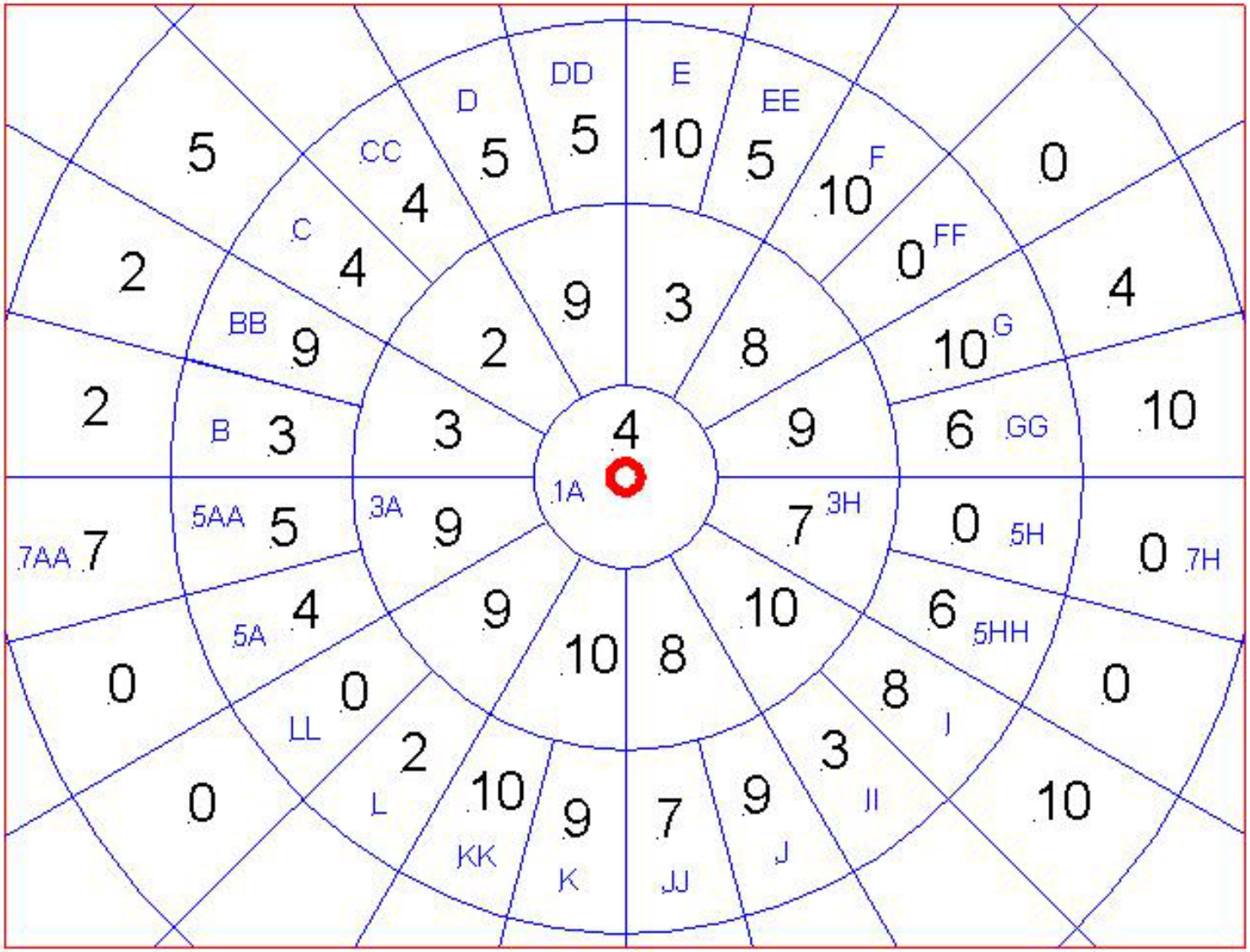


Figure E2.2

Body: Partial Alarm

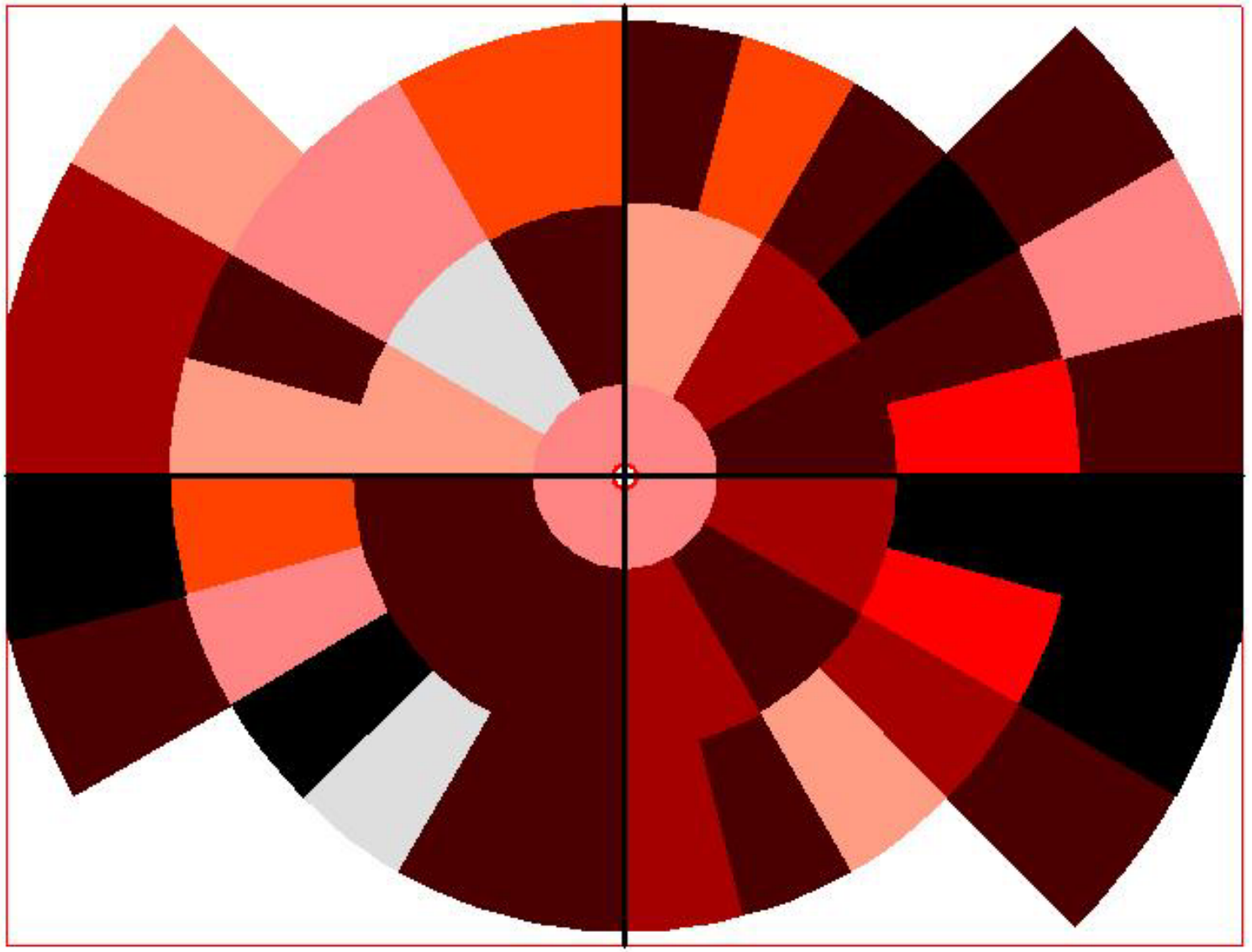


Figure E2.2

Body: At Least Partial Alarm

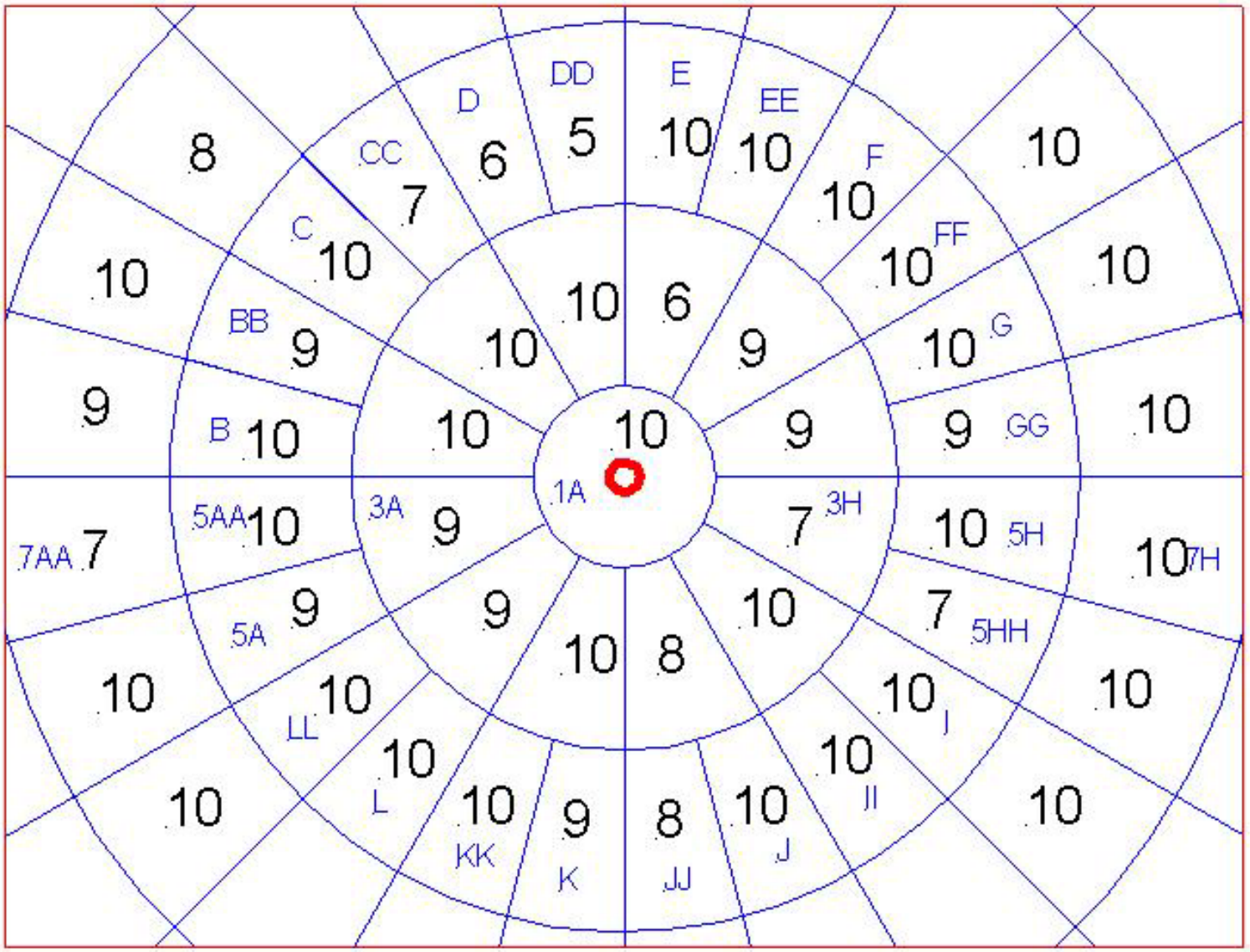


Figure E2.3

Body: At Least Partial Alarm

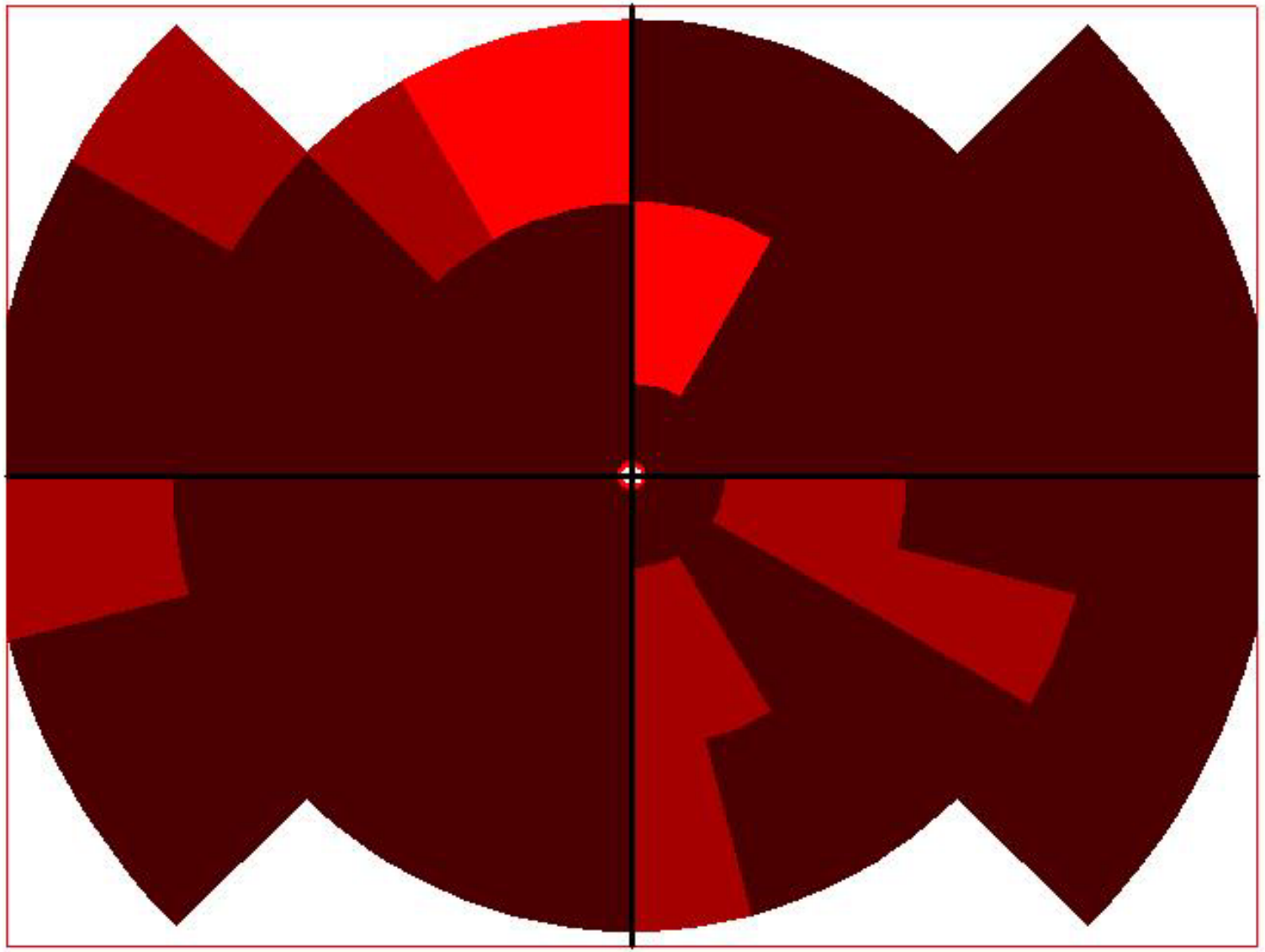


Figure E2.3

Hand and Body: Full Alarm

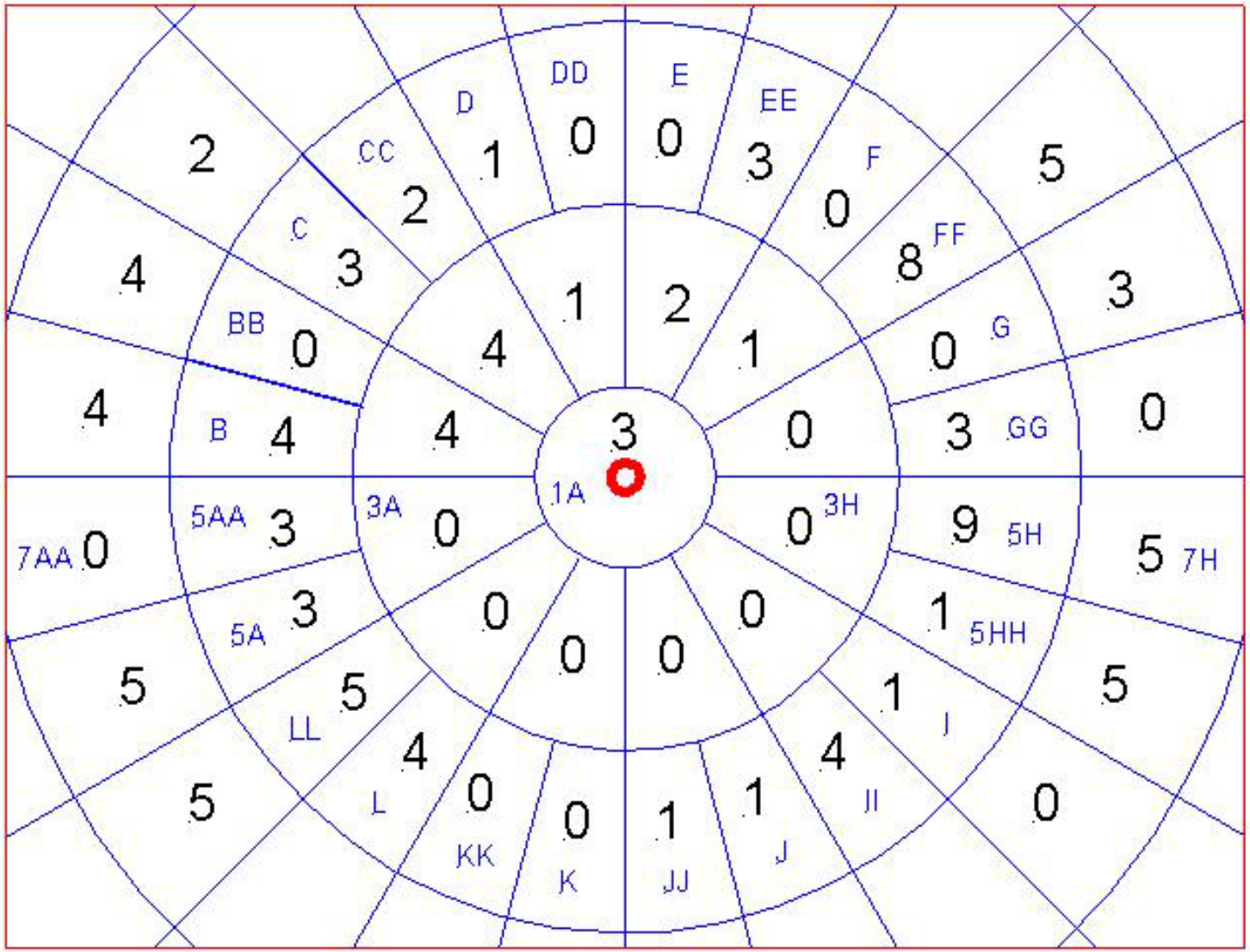


Figure E3.1

Hand and Body: Full Alarm

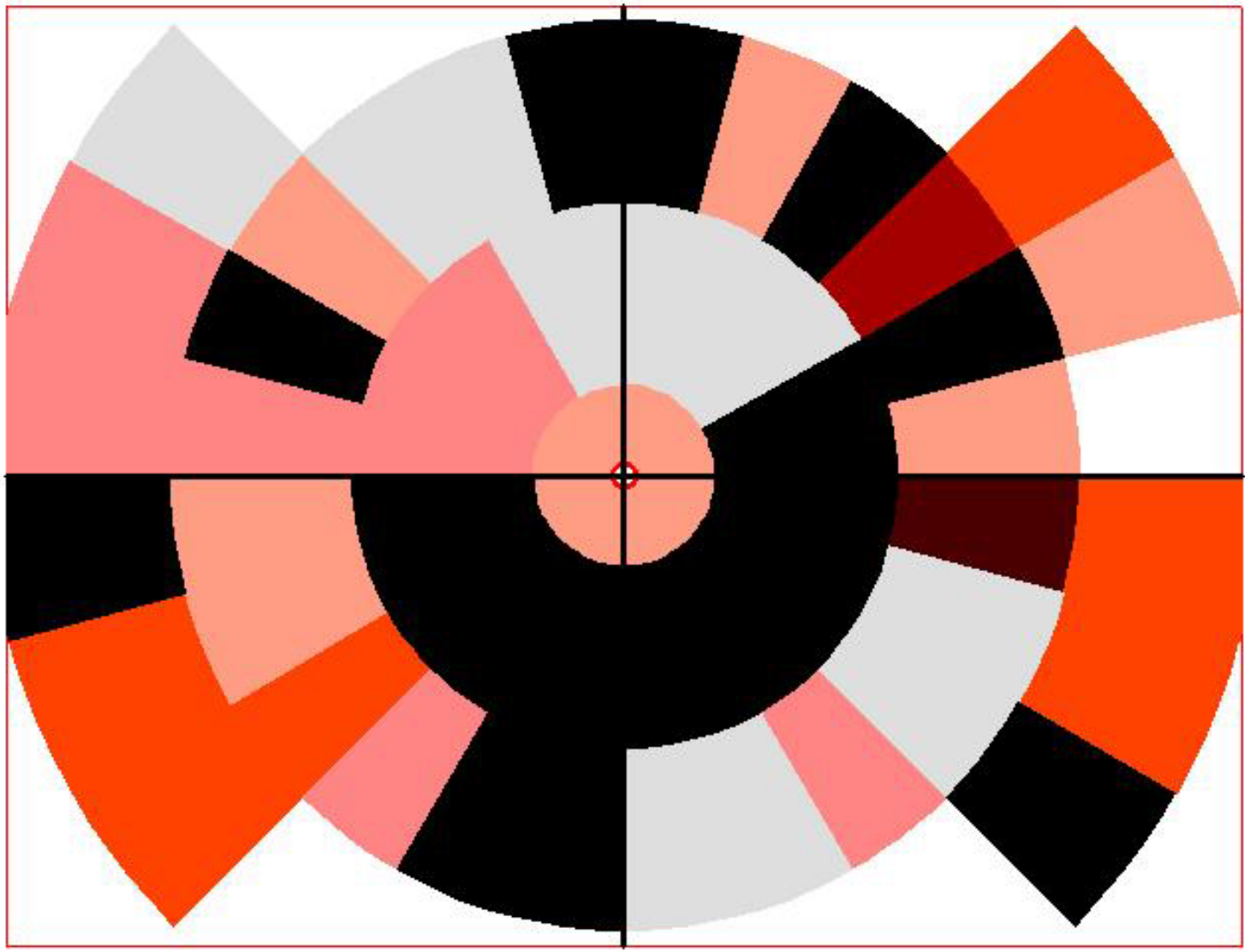


Figure E3.1

Hand and Body: Partial Alarm

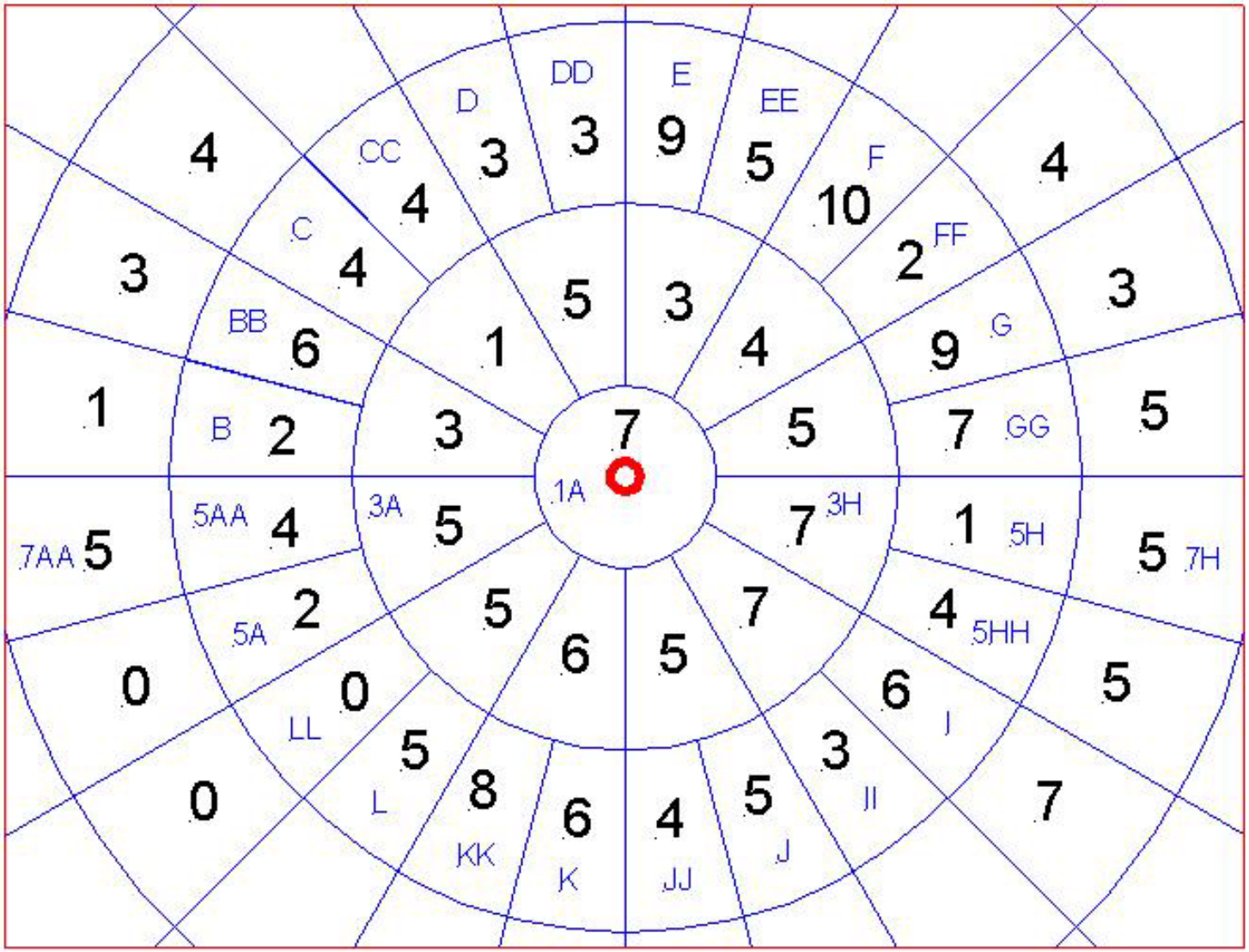


Figure E3.2

Hand and Body: Partial Alarm

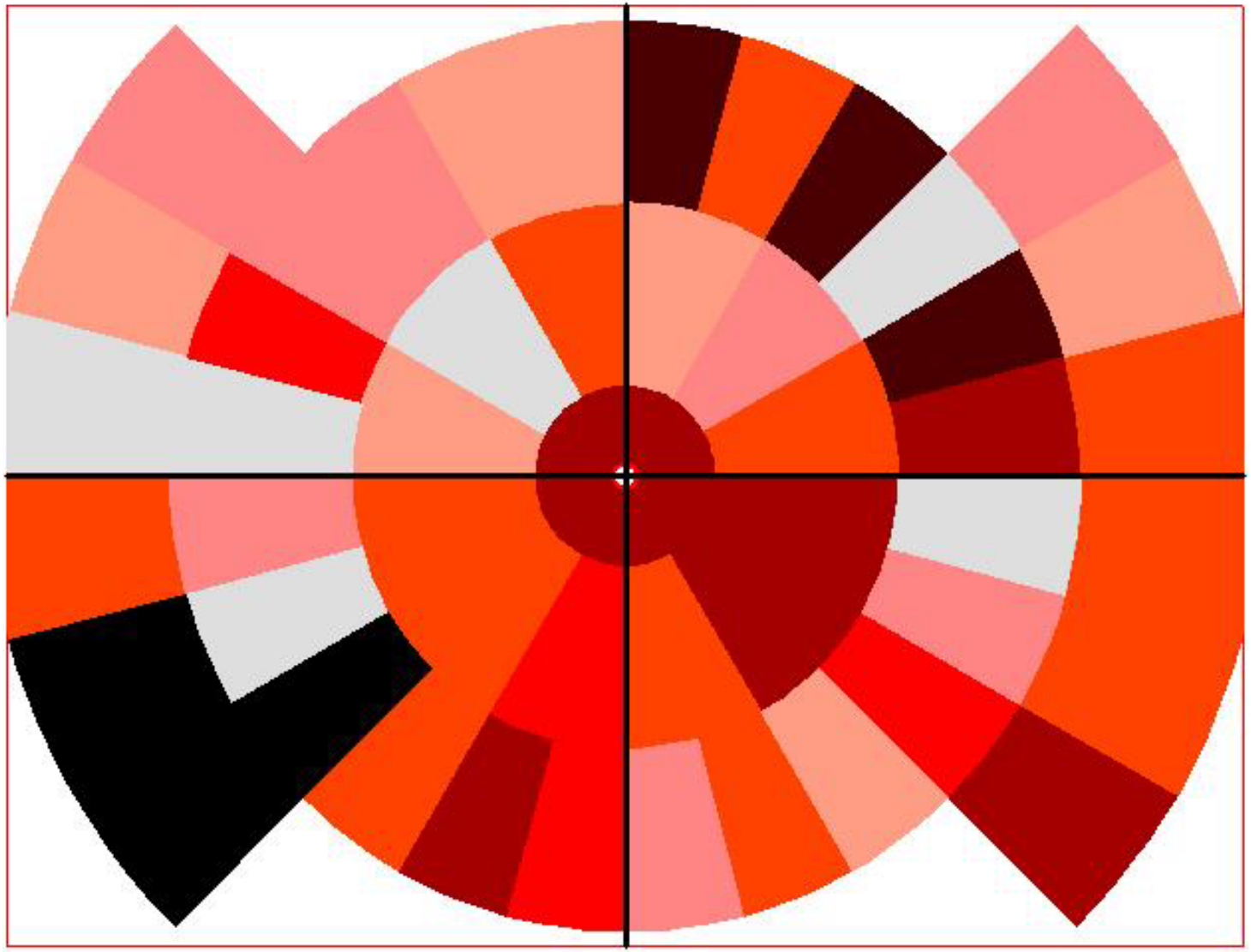


Figure E3.2

Hand and Body: At Least Partial Alarm

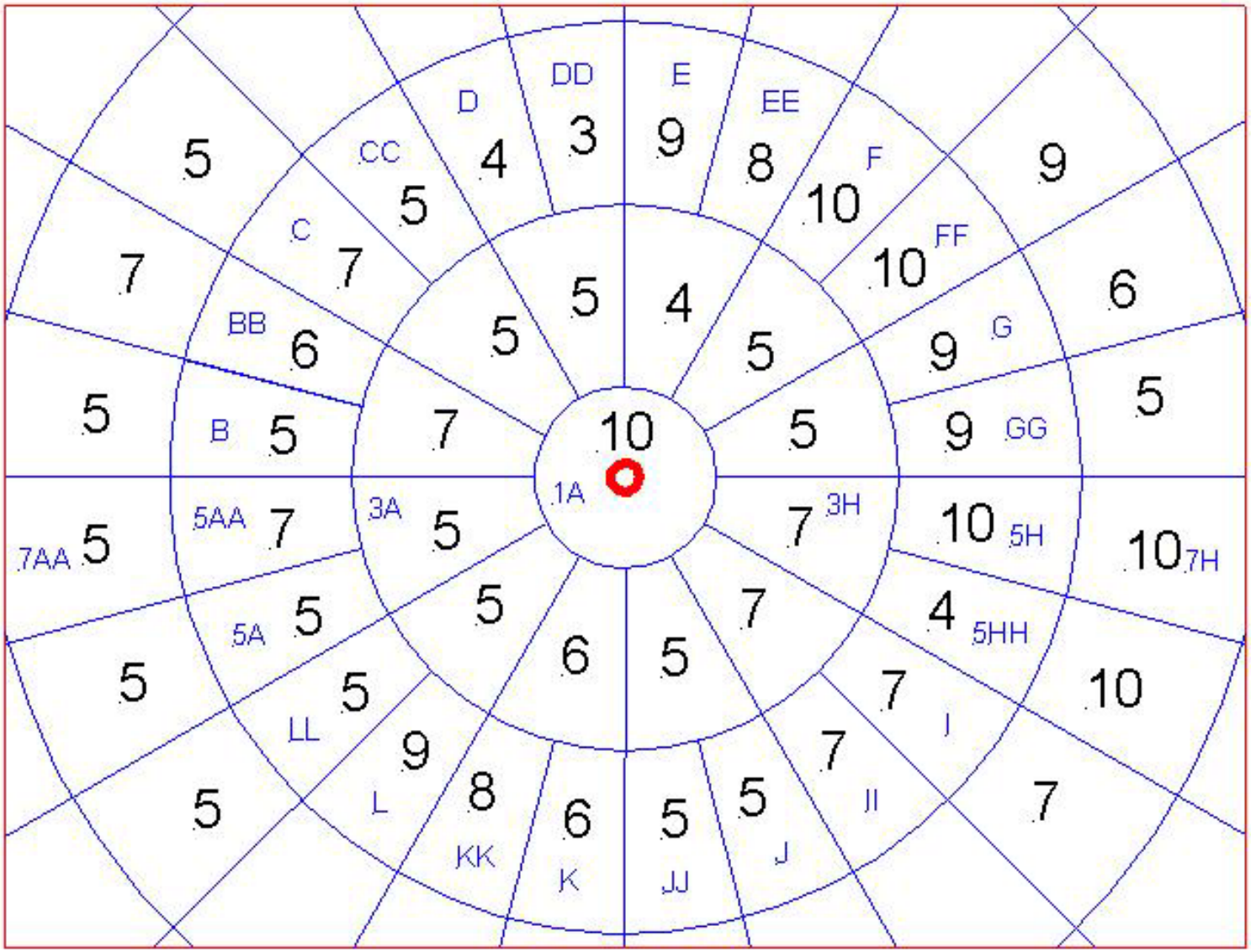


Figure E3.3

Hand and Body: At Least Partial Alarm

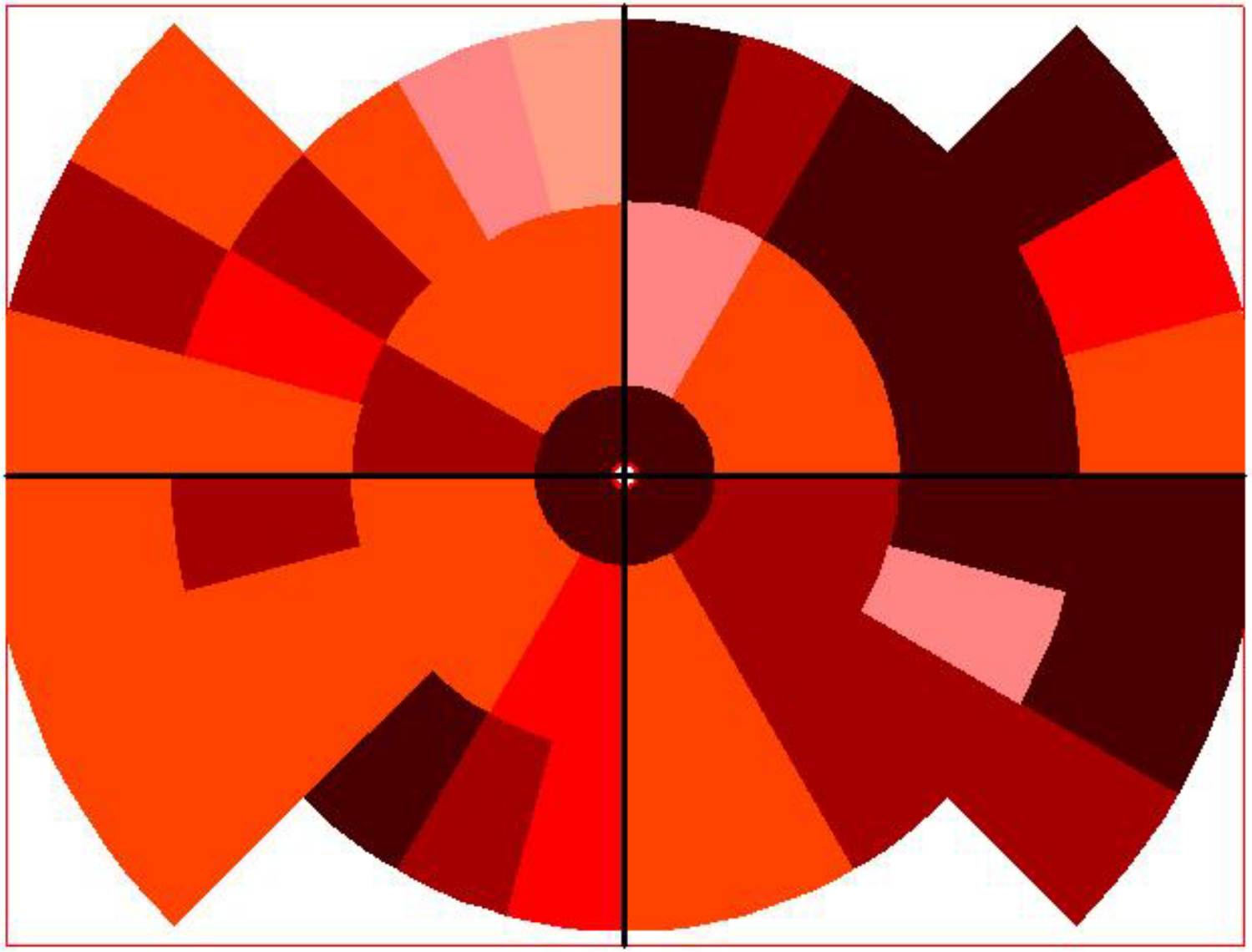


Figure E3.3