Final Project:
Secure Science Lab Simulation

Cooper Zuranski

Fall 2021

Table of Contents:

List of Figures and Tables

Introduction

Purpose:

The goal of this project was to create an intricate system utilizing the vast majority of the knowledge learned this semester both in lecture and lab of ECE 362: Microprocessing Systems and Interfacing. This required mastery of the range of available commands for the 68HC12 microcontroller, coding design practices suited for a microcontroller program, multi-file programming, ability to plan a design for a sophisticated system from a list of requirements, and the ability to troubleshoot both hardware and software (IDE) behaviors. The "Secure Science Lab Simulation" (hereinafter referred to as the "lab") has a pin-pad secured system, door access/ automatic doors, emergency shutdown button, adjustable HVAC/thermostat fan system, running date/time clock (which can be set), fire suppression system & fire alarm pull, speaker system for sounds/alarms, LCD message display, three rows of lights, and an auto-off system for them. This was programmed almost entirely in assembly language for the Motorola 68HC12 using the Freescale CodeWarrior IDE with extremely little coding in C. It is approximately 10,068 lines of code (not including libraries) and took about 84 total hours.

Assumptions:

Basic assumptions used throughout this project include the precision of timed events. All items which are intentionally supposed to replicate a certain amount of time are rounded to be as close as the math of the dividing RTI control frequency would allow without using any complex correction algorithms.

All variables are declared in main.asm, grouped by association of their usage with respective subroutines and states. Initialization of the strings which are altered in the usage of the program follow, then device initialization such as DDRs, and RTI period of 0.128 ms.

Other design-based assumptions include: Typo of the example of LEDs representing half-full C02 canister levels - the example was given as $F0 though $0F makes more sense for a halfway marker. Requirements surrounding statements of "any time" were clarified against contradictory system states which should prevent operation – these trump the stated requirements. It would realistically make sense for password entry to be replaced with '*', and thermostat to only vary between 65-75 degrees. The sound for the door closing should occur even during the fire suppression sequence. Instructions use both 8~1 and 7~0 bit notation without specifying, so choices were made based on what did not interfere with DDR and Port T switches that complete the circuit for the speaker and DC motor. Leap years are not considered, nor are dates outside this century for design simplicity, as this is only a simulation, and not the emphasis of the project. Scaling of the potentiometer value is based on my regular board which ranged [0,138].

Design

Peripherals:

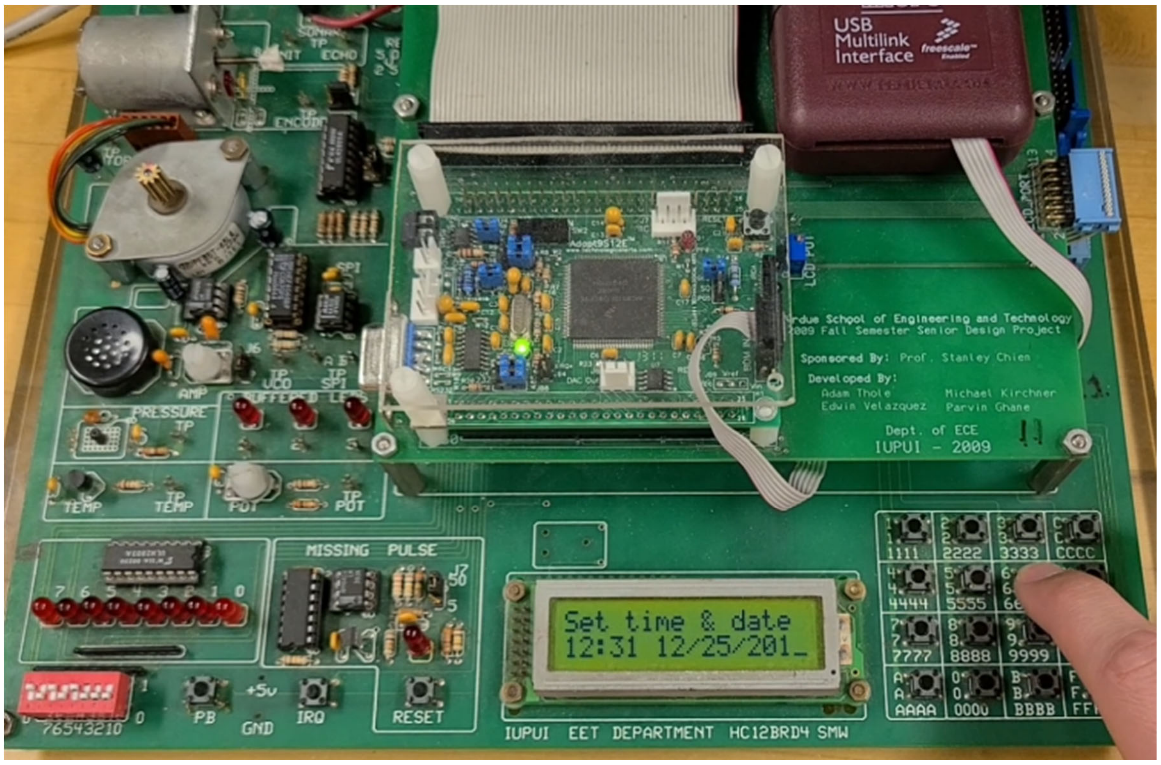


Figure 1: Lab Board with Microcontroller/ All Peripherals

Centered under the acrylic is the Motorola 68HC12 microcontroller. In the operation mode utilized, it has two 8-bit accumulators, A:B, which cascade into D, which is 16-bit. Additionally, there are 16-bit accumulators X and Y. A and B are used mainly for calculations, and separating items into 8-bit pieces, while X and Y are used for pointer arithmetic or extended 16 and 32-bit instructions. Not all peripherals available were utilized in the project. Those that were are listed, along with their purpose in the simulation, below.

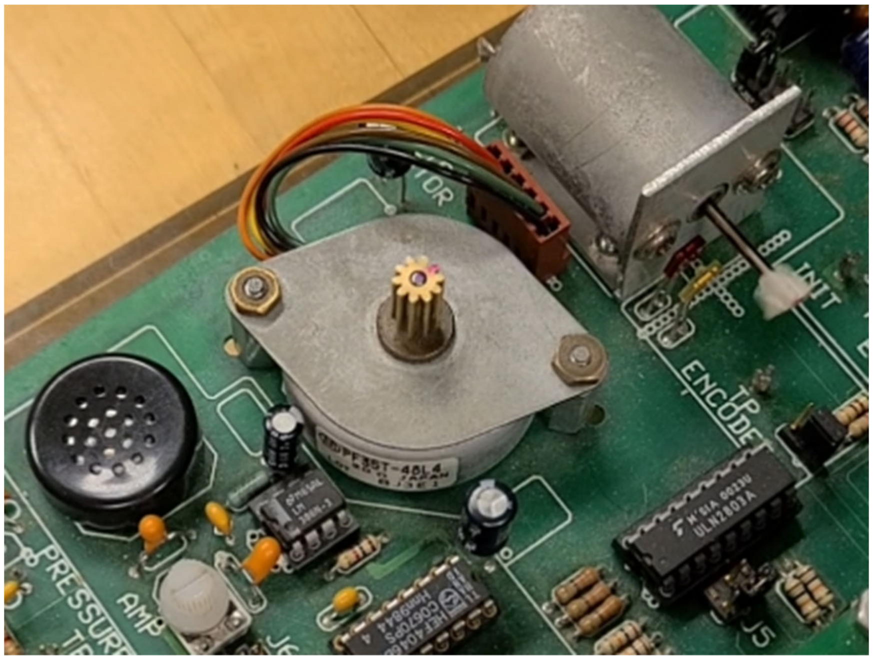


Figure 2: Speaker / Stepper Motor / DC Motor

1. Stepper Motor: Serves as the C02 pump of the fire suppression system. It rotates clockwise one rotation, then half a rotation counterclockwise to represent the pump priming. It then accelerates to a constant speed, held for 15 seconds to empty the C02 tank, before decelerating. It lastly performs the opposite of the initial routine as the cool down.

2. DC-Motor: Acts as the HVAC system fan. Its duty cycle and therefore speed is correlated to set temperature, which is proportional to the value of the potentiometer. It will spin at 100% duty in the case of a fire.

3. Speaker: Plays chimes and sirens for events such as emergency shutdown, fire alarm, or an active door.

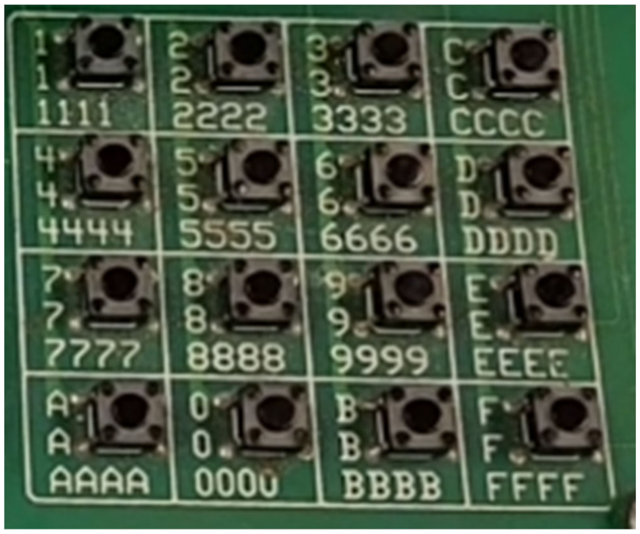


Figure 3: Hex Keypad

4. Keypad: Serves as the user interface to the lab control system. In applicable scenarios, A scrolls up, B down, and F selects. Elsewise, values are taken from entry as appropriate, often through a scrutinous input validation system.

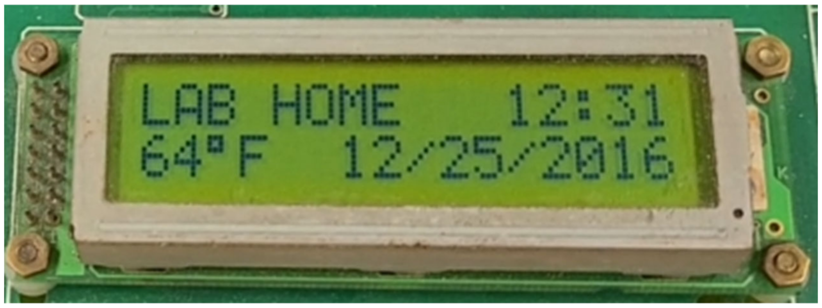


Figure 4: LCD Screen

5. LCD Screen: Serves as the monitor for the admin to be able to navigate the sub-menus and settings of the lab control system.

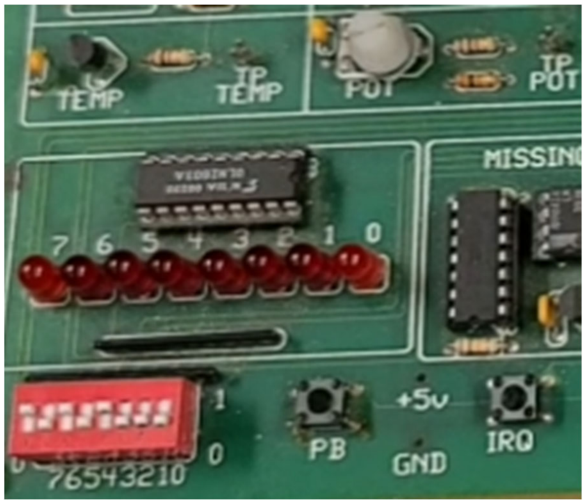


Figure 5: Potentiometer / Push Button / Switches / LEDs / IRQ

6. Potentiometer: Serves as the speed control HVAC fan and thermostat. More clockwise increases the duty cycle and lowers the temperature and vice versa.

7. Push Button: Serves as the door access button. The user is prompter for a password, then the door opens, holds, and closes.

8. Switches: (7 ~ 0)
    Switch 7: Hi to lo movement initiates the fire alarm sequence
    Switch 6: Lo to hi enters light control menu, hi to lo to exit
    Switch 2: Row one light switch. Lo to hi on
    Switch 1: Sim., row 2
    Switch 0: Sim., row 1

9. LEDs: In an emergency shutoff state, all LEDs strobe in an alarm-like manner. In a fire state, they represent the fill level of the C02 tank. All lit is full, empty is none. Decreases from left $FF to right $0. In normal state, serve as rows of lights. One on indicates its entire row.
    LED 2: Row three lights
    LED 1: Row two lights
    LED 0: Row one lights

10. IRQ: Serves as the emergency shutoff button for the laboratory.

Software Implementation:
Discussion:

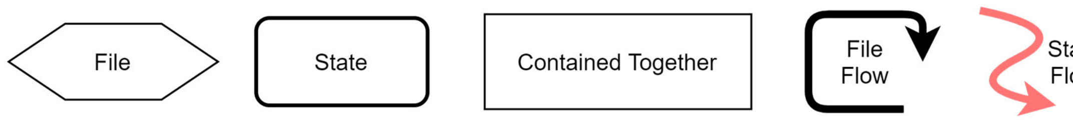
The software implementation was laid out in the fashion shown below in the system flow chart. My approach was non-modular, and not entirely consistent. This is because the requirements required a robust system that could branch back, forth, around, and through multiple states, all while maintaining others. Which when done via code, if I were to reuse as much code as possible for a robust design, meant an open-faced state machine that could weave in-and-out of states in a non-modular fashion.

The system is primarily a state machine with states based mostly around the LCD, mostly contained within main.asm. The RTI controls almost all state maintenance (aside from when keypad input is required, as that takes too long to occur in the RTI) such as peripheral action and counter progression.

The "main.asm" contains declaration of all variables but little initialization, as states are to be able to be used multiple times, with different entry points. Therefore, most states have an initial entry section (often notated as init-..) where related variables needed are reset, including dynamic strings used in LCD display.

States are global flags containing 1/0 such that they can utilize bit-branch instructions, and be accessed in constantly occurring things (such as the HVAC fan) that vary operation depending on system state. The global flag approach is also done to cut down on execution time (and the key on how so much seemingly fits into a .128ms RTI), as state-deemed unnecessary portions of code can be skipped, whilst keeping them in a logical/convenient location. Lastly, by the same benefit previously stated, this also allows states to prevent the user from actions/changing states when the design intent implies such.

System Flow Chart:



1) Bi, monodirectional, state arrows show flow... NOT guaranteed that path will be taken every time, ie conditional
2) Colored files are only accessed when program is in that color's associated state
3) The rounded rectangle signifies a state, and where it is most maintained/initialized
4) Arrows into the top of a file signify entry, arrows leaving from the sides or bottom of a file show calling / re-entry
5) Black arrows show actual file flow, while colored arrows show state flow
6) States which are simple and/or only entered through means of another state are not color-coded
7) States often flow through others' code even without said other being set

Figure 6: Flowchart Key & Rules

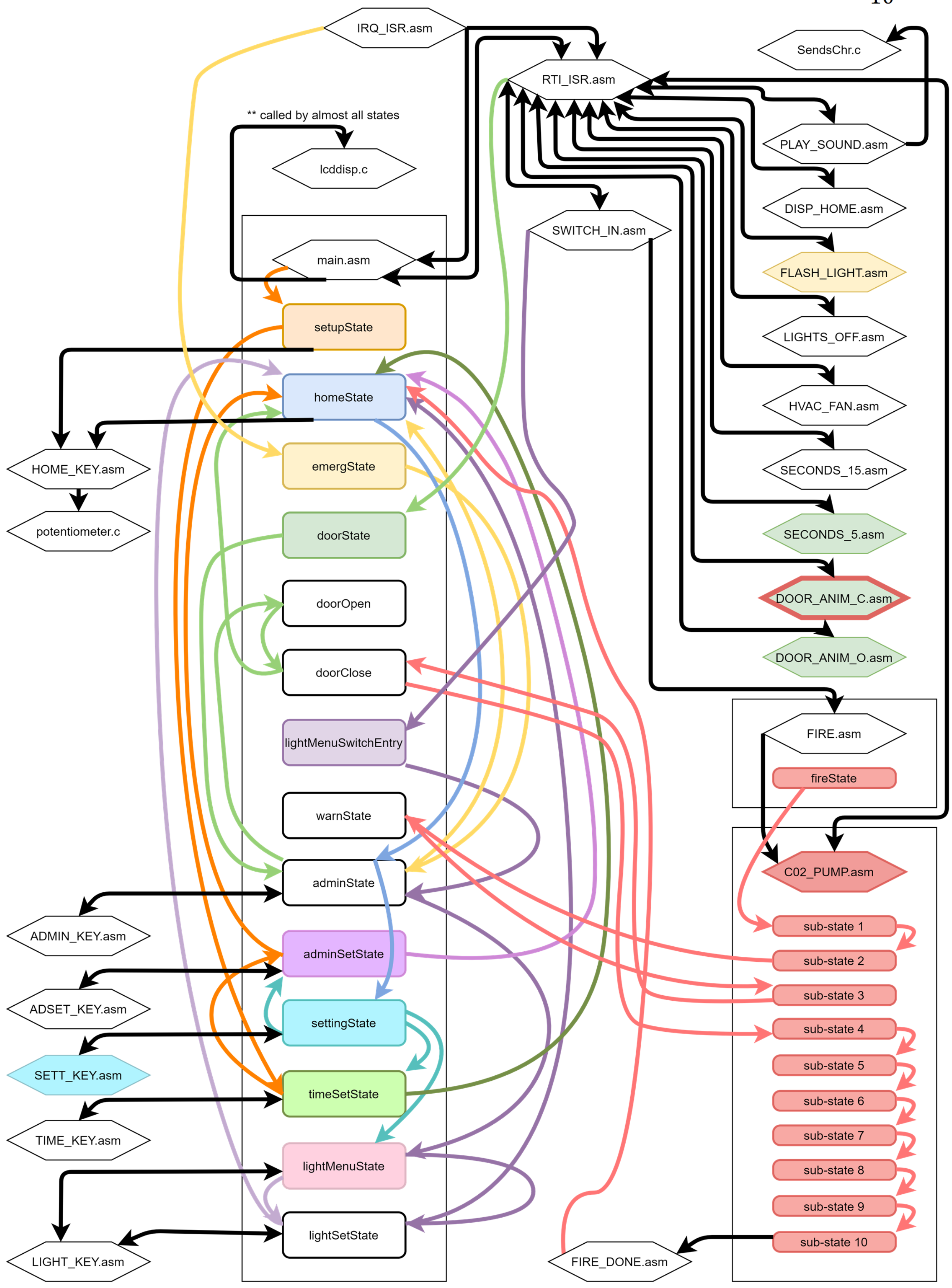


Figure 7: System & State Flowchart

Error Handling and Failsafe Techniques

Failsafe and error-handling techniques implemented into the program include:

- No menus may be accessed without admin pin past initial set up.
- Initial set up must be completed, nothing else can break this state.
- Once started inputting any values, little can interrupt this.
- Adaptive key-entry blocking (described next) prevents need for back-end handling.
- When in emergency/fire state, large amounts of peripheral controlling code regularly used are skipped so they cannot interfere with states' peripheral usages.
- Even when not displayed, the time/date displaying code is still ran, just executed differently due to the current active state.
- All input peripherals are checked at much larger multiples of the RTI period, so that even when not debounced, values are more stable.
- At duty cycles lower than ~ 35%, the HVAC fan jitters instead of rotating. Therefore values under this are simply considered "off" for more elegant use.
- Provided code for lcd display mysteriously alters register values. So when execution of this is occurring in/around the longer keypad input scanning process, values become corrupted – always out of $0-$F range. Therefore any key press determined to be above $F occurring is simply thrown out.
- The potentiometers on all boards read differently. The one on the board I used most was from 0-138, so I based my scaling algorithm around this. On other boards, even if a value is out of range, the duty cycle will still be administered as 0% or 100%.

Design Changes:

There were few design changes made. One was based around the assumption (which was confirmed) stated earlier of a typo in the order in which LEDs indicating the C02 tank fill were to empty. I used left-to-right. Another was as opposed to the thermostat being adjustable from 0-100 °F, I imposed a different scaling algorithm to scale the temperature between 64-75 °F as the prior is entirely unrealistic for an HVAC system. All other changes were additions listed below.

Project Additions:

Beyond the required design of the lab control system, I incorporated the following additions:

- LCD evacuation countdown
    - After the C02 pump completes its priming sequence, people in the lab have 10 seconds to evacuate before the doors seal. I added a warning message with active visual countdown.

- Adaptive key-entry blocking
    - Regular required input validation includes not allowing hex digits to be entered into times/dates. However certain times/dates are invalid, even with valid characters, such as a 13th month, 60th day, or 36th hour.
    - Instead of throwing an error, my system dynamically checks key entries as they occur, to not allow inputs of these at all. For example, if the digit '3' is entered as the first day-digit, the system wouldn't input a press of '7'.
    - Accounts for months with 30 vs 31 days (though not leap years, as the benefit of accounting for that in a 5-minute simulation is well beyond the point of diminishing returns).
- Live auto-off set time display
    - Instead of needing to select a row of lights in order to see its current set shut-off time, I have incorporated it into the selection menu of the rows itself. This allows for more ease-of-use.
- Sub-menu state time-out
    - The requirements state to time-out of the settings menu after 15 seconds of no further selection. Similarly to how wiggling one's mouse keeps a computer monitor on, scrolling through the menu is also considered to keep the system from timing-out back to the home screen. Additionally, this functionality is implemented similarly within the light overview menu.

Functionality of Project

Working:

All parts of the project function correctly, with virtually no issue. Hardware issues plagued parts of the project, though I created failsafes surrounding them to interfere with the user as little as possible.

Not Working:

The hex keypad keys seem to not always make contact, some need pressed very hard. The potentiometers on all boards read non-linearly and inconsistently in range. The dipswitches are not securely electrically continuous/isolated – without a very large reading period, flipping one switch will affect the readings of many of the others. The DC motor undergoes significant resistance and doesn't have enough inertia for continuous rotation until ~ %35 duty cycle. Remedies for these listed issues are mentioned under "Error Handling and Failsafe Techniques."

Conclusion

In conclusion, this project encompassed almost everything covered in ECE 362 and more. Completing a complex system design all in assembly was a massive undertaking, yet it is astounding what you can accomplish with a handful of registers, ~ 150 instructions, and interrupts.

If I were to do this again, I would certainly change the overall layout of the system. I was mostly consistent and considerate about re-using code dynamically, but not entirely. For instance, many states have their own keypad routine, merely so the LCD and few other things can be updated whilst looping, waiting for input. Seeing how my system ended up, I could've consolidated these into one file, utilizing the state-flag system I already had. I would also either move things related to the fire state with the others in main.asm in the main "state machine," or modularize each of them. Most likely the prior, as jumping past/partway through a state's code was invaluable in my efforts to reuse code, as well as simplified flow between states.

Future Improvements

Future improvements (hypothetically) considered during this project include:

- Spanish Mode
    - Merely an extra selection in the setup stage, Spanish mode would replace the strings used throughout the system with ones in Spanish. This has applicability, as this is a common feature in most consumer electronics.
- Error Audiation
    - Though incorrect input is blocked by the system as is, an inexperienced user could mistake this as a broken button. A single beep error tone would prevent this.
- A More Elegant Fix to Issues with the Built-In LCD Subroutines
    - As described before, the given LCD subroutines have odd side-effects. These are remedied but could be remedied more elegantly with more time.

Appendix I: User Manual

Contents:

1) Lighting

i) Switches

- Switches 0-2 (7~0) control light rows 1,2,&3.
- When flipped up, the lights turn on for that row.
- To turn the lights off manually, flip the switch from high to low. The lights will blink twice as they shut off.

ii) Auto Off Menu

- Use A to scroll up, B to scroll down through the list of light rows.
- Press F to select the one the cursor is currently on.
- Enter the time in 24-hour format you wish for the lights in that row to shutoff at.
- After a light is shut off via auto-off, the switch must cycle to hi before being on again.

iii) Switch Access Menu

- One of two ways to access the “Auto Off Menu” is moving switch 6 (7~0) lo->hi.
- Enter the admin pin.
- Complete all steps in “Auto Off Menu”.
- When done, move switch 6 hi->lo..

2) Settings

i) Access

- From the home screen, press F to access settings .
- Must enter admin pin to gain access to any settings.
- Scroll settings options with A up, B down.

- F to select the sub-menu highlighted by the cursor.

ii) Auto Off Menu
- This is the ulterior way of accessing the "Auto Off Menu".
- See "Auto Off Menu".

iii) Change Date/Time
- Input each individual value for the date and time 1 by 1.
- Any unacceptable values will be blocked from input.
- When finished, menu will exit.

iv) Change Pin
- Enter new pin value.
- When finished, menu will exit.

3) Door Access
- Press pushbutton for door access.
- You will be prompted for admin pin.
- Once entered, the doors will open, stay for 5 seconds, then close.

4) Fire Alarm

i) Switch
- Switch 7 (7~0) is the fire alarm.
- Pull from hi->lo to activate fire suppression system

ii) Notification System
- A fire alarm will sound.
- The screen will display a warning with 10 second countdown.
- You must evacuate the lab in this period.
- After this, the doors will seal.

iii) C02 Pump
- Will prime, run for 15 seconds to empty the tank, then cool down.
- No intervention is required.
- The state of the system will return to normal after this.

5) Emergency Shutoff
- The IRQ button is the emergency shutoff.
- When pressed, the system will prevent any further action.
- Lights will strobe as well.
- Admin pin entry will return the system to normal

6) HVAC

i) Potentiometer
- The potentiometer acts as the thermostat.
- To cool the room, turn it clockwise (& v.v.).

ii) Fan Speed
- The fan speed will change to correspond with your set temperature.
- A cooler temperature will have a faster fan speed.

Appendix II: Code

Note:

All variables are global and treated as such. This is due to the fact that at any point, a handful of subroutines can be checking the state of another part of the system. There is almost no time where passing a value/ receiving a value has any benefit greater than that of the global variables. It only complicates things and takes up processing time. Looking back, there are a few opportunities where passed parameters would be more elegant for less lines of code, however as it stands, there are none.

---

main.asm

This file contains almost all states of the state machine surrounding the system, and all items too long to fit within the RTI. I chose not to isolate them as the ability to literally flow from one state directly into another is handy for commonly occurring things. This file also has all variables and constants used in the entire project, declared by their association with other files or states. Only some are initialized here, most redundantly.

```
            INCLUDE 'derivative.inc'
;********************** EXPORT REFERENCES
*****************************************************
            XDEF Entry, _Startup
            ; we use export 'Entry' as symbol. This allows us to
            ; reference 'Entry' either in the linker .prm file
            ; or from C/C++ later on

            XREF __SEG_END_SSTACK      ; symbol defined by the linker
for the end of the stack

            XREF
RTI_ISR,PLAY_SOUND,SendsChr,PlayTone,display_string,init_LCD,HOME_KEY,A
DMIN_KEY,SETT_KEY,TIME_KEY,ADSET_KEY,LIGHT_KEY,SECONDS_5
            XDEF    Port_S,Port_T,Port_P,Port_U
            XDEF    RTIFLG,RTIENA,RTICTL,songCount,fireState
            XDEF    fireSnd,doorSnd,emrgSnd,autoSnd,curSong,curNote
            XDEF    active15,done15,out15Count,in15Count,reset15
            XDEF    curSwitch,priorSwitch,liSwState
            XDEF    blink1Count,blink2Count,blink3Count
            XDEF    S_curSong,S_curNote
            XDEF
step,c02State,stepLast,count30ms,stepNum,count15ms,in10Count,out10Count
,gradCt,gradCtLast,in2143msCt,out2143msCt
            XDEF
doorAnimCCur,doorAnimCDne,doorClose,doorStateC,doorAnimOCur,doorAnimODn
e,doorOpen,doorStateO,doorOpen,count250ms
            XDEF    fireDispCur,fireDispDne,warnMsg1
```

Appendix III: Project and Report Requirements

Project Requirements:

Secure Science Lab

Objective:

Write an assembly program to simulate a controls system of a science laboratory.

Peripheral Simulations:

1. *Stepper-Motor:*
   C02 pump. Has priming, running, and cooldown stages: Rotate CW one rotation, ½ rotation CCW. Stop momentarily before accelerating to speed. Run constantly for 15s. Decelerate, rotate CCW once, then ½ a rotation CW.
2. *DC-Motor:*
   HVAC fan. Speed proportional to set temperature (cooler = faster). Full speed during fire.
3. *Keypad:*
   Scroll, select, and input values for:
   a) *Set the time, date, and admin pin.*
   b) *'F' from home state to enter settings (after successful admin check).*
   c) *Scroll settings menu (up 'A', down 'B') and lighting rows in lighting menu.*
   d) *Change time, date, and admin pin.*
   e) *Select a row and set a time for auto-shutoff to occur.*
   f) *Enter admin pin for security purposes.*
4. *Potentiometer:*
   Controls thermostat and subsequently HVAC fan.
5. *LCD:*
   a) *Door open and close animation via moving blocks in and out from screen edges.*
   b) *Home screen with date, time, and set temperature.*
   c) *Warning message when fire occurs.*
   d) *Welcome message upon first turn-on.*
   e) *Prompt admin pin entry.*
   f) *Menu of sub-menu options.*
   g) *Lighting menu to alter/set auto shutoff times.*
   h) *Menus to set time/date and admin pin.*
6. *Push Button:*
   Door access button. Press to prompt for admin pin and subsequent door opening on correct. Done via LCD open animation, 5s pause, door close animation (with paired chime sound).
7. *Switches [7~0]:*
   7-fire alarm (hi->lo), 6-light menu alternate entry (lo->hi), exit (hi->lo), 2~0 -rows 1-3 light switches
8. *Speaker:*
   Fire alarm, emergency alarm, door chime, and light chime.
9. *LEDs [7~0]:*
   2~0 correspond with light rows 1-3 (must blink twice whenever shut off, no matter the means). Strobe in emergency shutdown. Show fill of C02 during fire, empties over 15s.
10. *IRQ:*
    Emergency shutdown button. Disable everything. Cause light strobe, admin pin to clear.
11. *RTI:*
    Controls timing.

Requirements:

1. *The user must be able to turn on/off any row of lights at any time. Any changes to the switches must be indicated by the appropriate blinking LED.*

2. *The user must be able to adjust the overall temperature of the science lab at any time and the DC motor should automatically adjust accordingly.*

3. *The user must be able to set an auto-shutoff timer for the lights.*

4. *The emergency shutoff and fire alarm should have the ability to be activated at any time.*

5. *The overall layout of your system should be easy to understand and make sense. The user should be able to operate the system with little to no training or explanation. If you are unsure if the layout of your system makes sense, ask one of your TAs or fellow students to try to move through your system.*

6. *No delay loops are allowed, you must utilize the Real Time Interrupt. DELAY LOOPS ARE ONLY ALLOWED OR HEX KEYPAD DE-BOUNCE.*

Report Requirements:

Final Project Report Requirement

The final project should include the following sections:

1. *Cover letter:*
    a. *The project title*
    b. *The name of the team members*
    c. *Project date (Fall 2013)*
    d. *Should include a picture of the equipment and/or a picture relevant to the project*

2. *Table of contents*

3. *List of figures and tables*

4. *Introduction:*
    a. *Describe the goal and the purpose of the project.*
    b. *Assumptions made*

5. *Design:*
    a. *Show the peripherals that are used in the project and what they are used for*
    b. *Software implementation of the project:*
        i. *Give a high level description and discussion*
        ii. *General system flow chart*
        iii. *Flow charts for each ~~module~~*
        iv. *Error handling and fail safe techniques*
    c. *Changes made in the design*
    d. *Additions to the project*

~~*6. Description of the division of work between team members*~~

7. *Description of which parts of the proposed project is working and which part is not working*

8. *Conclusion*

9. *Discussion and suggestions for future improvements on your project*

10. *Appendixes:*
    a. *User Manual*
    b. *Code:*
        i. *The code should be commented (useful and meaningful comments)*
        ii. *Description of each file, subroutine and procedure:*
            - *Name*
            - *Inputs/Outputs and method of passing the parameters*
            - *General Description*

11. *References*