**Bank Vault**

Abdelrahman Ramadan

Spring 2022

Diagram, engineering drawing

Description automatically generated

Table of Contents

* List of figures and tables 3
* Introduction 4

1. Goal and purpose
2. Assumptions made

* Design 5

1. Peripherals
2. Software implementation
3. Flowcharts
4. Safe handling techniques
5. Changes made
6. Additions made

* Division of work 18
* Project functionality 18
* Conclusion 18
* Future improvement discussion 19
* Appendixes 20

1. User manual
2. File and Subroutine Description
3. Code

* References 101

List of Figures and Tables:

Figure 1: Stepper motor 5

Figure 2: DC motor 6

Figure 3: Keypad 7

Figure 4: Potentiometer 8

Figure 5: LCD Screen 9

Figure 6: Push Button 10

Figure 7: Dipswitches 11

Figure 8: Speaker 12

Figure 9: LEDS 13

Figure 10: IRQ 14

Figure 11: General System Flowchart 16

Figure 12: Keypad Subroutine Flowchart 17

**Introduction**

**Purpose:**

The purpose of this project is to incorporate all ideas and concepts learnt in this course throughout the semester into a project that simulated a bank vault. The project was implemented on the 68HC12 microcontroller and involved the use of various interfacing techniques to successfully simulate a bank vault. This project also involved interfacing with peripherals present on the microcontroller to provide the user with control over the system. The bank vault system is all connected to each other via flag variables which are controlled all together by the real time interrupt which is present in the program. The system is protected by a password which can be changed upon request by the user. The LCD screen displays information that the user needs in order to control the vault which consists of various menus that display different information. The program was put together using the assembly language through the CodeWarrior IDE.

**Assumptions made:**

A few assumptions were made prior to coming up with this program so that the system boots up directly without bothering the user with the setup of initial conditions instead they can be changed later through the menu. Initially, the system boots up with the clock set to 10:00:00 (10 AM) on a Monday with the current temperature set to 8˚ Celsius. The desired temperature of the vault is set to whatever the potentiometer value is which is scaled from 0 to 75˚ Celsius, this was done to ensure a realistic experience. The vault is also booted up with an amount of $0 present in the bank, so a withdraw is not possible as a first transaction. The maximum amount of money that can be present in the vault is $9,999 and the bank uses base 10 notation, so an input of hexadecimal values is rejected by the program. The clock is in a 24-hour format for the sake of simplicity in the code. There stepper motor operates in two speeds so two different types of delay counter in the RTI are implemented to achieve a visible change of speed to the user.

**Design**

**Peripherals:**

1. Stepper Motor

The stepper motor is used to simulate the action of opening/closing the bank vault. A clockwise and counterclockwise direction is implemented to imitate the opening and closing of the vault respectively.

A picture containing text, electronics

Description automatically generated

Figure 1: Stepper Motor

1. DC-Motor

The DC motor serves as a fan that is controlled by the different in the two types of temperature present in the system. The desired temperature is controlled by the potentiometer while the current temperature of the vault is initially set to 8˚ Celsius but can be changed by the user.

A close-up of a circuit board

Description automatically generated with medium confidence

Figure 2: DC motor

1. Hex Keypad

The keypad serves as the primary control that is manipulated by the user. It is used to surf through the various screens which are displayed on the LCD screen and to take user input for various other operations such as depositing or withdrawing money from the vault.

A picture containing calendar

Description automatically generated

Figure 3: Keypad

1. Potentiometer

The potentiometer acts as the control unit for the desired temperature throughout the program.

Graphical user interface, application

Description automatically generated

Figure 4: Potentiometer

1. LCD screen

The LCD screen is used to display information to the user. It acts as the interface between the user and the system. It is primarily used to display the various menus such as the clock and vault menu, but it is also used to display error messages when the program is faced with rejected input.

A close-up of a computer chip

Description automatically generated with low confidence

Figure 5: LCD screen

1. Push button

The push button is used to declare when a person has entered the vault while it is open. If the vault is closed the push button is not functional. It is also used in series with an LED to display whether a person has entered or left. If a person is locked inside which is indicated by the button, then an emergency vault open will take place.

A picture containing text, electronics, circuit

Description automatically generated

Figure 6: Push Button

1. Dipswitches

The switches are used as a key to the vault, a series of switches will have to be turned on for the program to start the sequence of opening the vault.

A picture containing text

Description automatically generated

Figure 7: Dipswitches

1. Speakers

The speakers are used to alert the user when the vault is in an alarm state.

A picture containing text, electronics

Description automatically generated

Figure 8: Speaker

1. LED’s

The lights are used to alert the user of the state of the vault whether that would be an idle or an alarm state. Certain LEDs are also used to alert the user of an actions such as when a person enters the vault or when the vault is open/closed.

Website

Description automatically generated with medium confidence

Figure 9: LEDS

1. IRQ button

The IRQ is used as a panic button and when pressed will place the system into an alarm state.

A picture containing text, electronics

Description automatically generated

**Software Implementation:**

The project is divided into several files, each of which either controls a single peripheral or takes input from the peripherals and processes it. There are a lot of subroutines but only a few are called in the main, all of subroutines are controlled by certain flags which determine if a subroutine is to be executed or skipped. The Real time interrupt is used to control flags and the system clock since a heavy load in the RTI may cause lag issues in the program. All variables are declared in main.asm file and are referenced in which ever files they are used in.

The program initially boots up with a password screen which is used to set the password and is saved throughout the program unless changed by the user. Once a password has been inputted then the main menu will appear on to the LCD screen which will display the clock and the temperatures. The keypad is used to navigate between the menu’s, there are 3 primary screens which can be scrolled through when logged into the system. The first is the main menu which is mentioned above, the second is the change settings menu which when entered can be used to change the settings of the system, the third and last screen is the change password menu which is used to change the password of the user.

The vault is initially closed and in an idle state, but that can be changed by turning on switches 1 through 3, the program then checks certain conditions and decides whether the vault is ready to be opened or not. If the vault is open, a led will light up and the stepper motor will start spinning in the clockwise direction with a relatively slow rotation speed. A new screen is then displayed which gives the user the option to either deposit or withdraw money, the screen will also display the current amount of money present in the vault. At any time when the vault is open the switches can be used to close the vault and return to the main menu. If an option of either deposit or withdraw is chosen then the amount will have to be inputted by the user through the keypad and once that is done, the vault will close, and the system will return to the main menu.

The fan is controlled by the difference in temperatures and will spin with various speed depending on the difference. The IRQ button can be pressed at any moment to alert the system and ensue an alarm state.

**Flowcharts:**

Diagram

Description automatically generated

Figure 11: General System Flowchart

Discussion:

The system is glued together through flags, most of which are either controlled by the real time interrupt or by processing user input through the keypad or dipswitches. The program does not necessarily call nested subroutines but rather checks flags and decides whether a subroutine should execute or not. Most subroutines are called in main and inside each subroutine is a set of instructions that checks certain flag to see if the subroutine should be executed or not. Take for example the alarm subroutine mentioned in the flowchart above, the alarm subroutine is called in main in an infinite loop along with all the other subroutines, but when the alarm subroutine is called the alarm flag will be checked to see if the system is in a panic state. The alarm flag is set through the IRQ subroutine which when pressed will set the alarm flag and will fire a set of events that is in the alarm subroutine.

Diagram

Description automatically generated

Figure 12: Keypad Subroutine Flowchart

Discussion:

The keypad subroutine is called in main along with all the other subroutines. When a key is pressed the keypad subroutine will pass the button pressed as a parameter to another subroutine whose role is to process keypad input. It will do so by checking certain flags and will decide what to do with the passed parameter. For example, if we are on the main menu and I would like to change settings, to do that I would have to press B on the keypad to scroll to the right. If B is pressed, then the keypad subroutine will call a subroutine to process the input and will check certain flags to see what to do with the given input. Since we are changing screens then my flag controlling the menu state will be manipulated to display the change settings screen.

**Safe Handling Techniques:**

* When depositing or withdrawing money, hex values that are received as inputs are ignored.
* Withdrawing money more than the amount present in the vault will be ignored.
* Potentiometer is scaled to reflect realistic values for temperatures in Celsius.
* Password input is verified
* When changing the clock, only appropriate values are processed. For example, the only range accepted for hours is 0-24.

**Changes made to original project:**

The only change made was the keypad debounce delay which was changed from 10ms to 5ms through trial and error to achieve the smoothest user experience. The rest of the project aligned with the Bank Vault project requirements apart from the non-functional parts of the project which are discussed later.

**Additions made:**

* Password initialization on system boot up.
* Designated LED blinks when the vault is in the process of opening or closing
* Allowed password to contain hexadecimal values for further security of the vault.

**Division of Work:**

The sole contributor to this project is Abdelrahman Ramadan and the work was divided into a little under a month since the initial assignment of the project. The code was the first step implemented followed by the report to ensure accurate descriptions of the software implemented and general information throughout the report.

**Project Functionality:**

Functional:

All peripherals operate the way they should and are functional. Most of the project details mentioned in the requirements were implemented successfully.

Not Functional:

Requirement four that is present in the requirements document was not implemented in my project which was the dollar bills scrolling through the screen when a transaction takes place.Another point that was not implemented was the vault timer which would forcefully close the vault when opened for a certain amount of time. Other than these two requirements, the rest of the project is fully functional.

**Conclusion:**

This project took a lot of effort to put together but with proper planning and effort, I was able to put together a system which is fully functional. This project combined all concepts learned throughout the semester which by themselves were already challenging. Building this system was not only tough but it has taught me essential troubleshooting and planning skills.

**Future Improvements:**

There is a lot to add to this project to make it more user friendly and increase the quality of the system. Below are a few suggestions.

* To increase security, the password should be more than four characters long.
* Change the clock from a 24 to a 12-hour format for better readability.
* A buffer on the screen that would the show the animation of the vault opening and closing.
* Play a variety tone when the panic button is pressed instead of a single tone.
* Switch the functionalities of the potentiometer and dipswitches, so the potentiometer would act as the opening knob of the vault which makes for a more realistic approach.
* Increase the maximum amount of money that can be present in the bank in my case it is $9999.

**Appendix I**

**User Manual:**

Menu:

* There are three menus that can be scrolled through. The main menu should display the date and the clock along with the temperatures.
* ‘A’ is used to scroll to the left while ‘B’ is used to scroll to the right.
* The two other menus are change settings and password both of which if need to be activated by pressing ‘F’ when viewing them on the LCD.

Password:

* On system boot up, the user gets to set their initial password which will be used for the remainder of the program unless changed through the menu.
* Passwords are four characters long and can contain hexadecimal values
* After a wrong try, the system will display ‘incorrect pswd’ and take you back to the main menu.

Bank Vault:

* The vault opening is initiated by turning dipswitches 1 through 3. The program will keep trying to open the vault as long the switches are on.
* Conditions are checked when trying to open the vault, if said conditions are met then the user will be met with the password input screen
* When the vault is open, the money present will be displayed, and the system will give you the option of depositing or withdrawing money.
* The vault can be closed at any time by turning off switches 1 through 3 or by completing a transaction which will then take you back to the main menu.

Stepper motor:

* The stepper motor will turn clockwise when the vault is opened and counterclockwise if the vault is closing.
* The stepper motor has two speeds. One for idle state and the other for emergency close.

DC motor:

* DC motor’s speed is dependent on the difference between current and desired temperature.
* If the difference is zero, then the motor will stop spinning.
* In a way, the motor’s speed is controlled by the potentiometer since the potentiometer controls the desired temperature.

Dipswitches:

* The only dipswitches used in the program is 1 through 3, while switch 4 should always be turned on for the DC motor to operate.

Push Button:

* The push button is only functional when the vault is open and it represents a person entering or leaving the vault, if a person is left inside the vault while its closing then an emergency exit will be initiated which is represented by the fast speed of the stepper motor.

LED:

* LED 8 will flash if the program is in an idle state.
* LED 1 will be lit if the vault is open. If it’s flashing, then the vault is in the process of opening or closing.
* LED 2 reflects the push button presses and will be lit if someone is present in the vault.
* All LED’s will flash when in an emergency state.

IRQ button:

* The IRQ can be pressed at any moment to initiate the vault into a panic state (emergency).

**Appendix II**

**File and subroutine descriptions:**

IRQ\_ISR.asm:

This file contains code that will fire the set of events that take place in an emergency such as flashing the lights and prompting the user with the password screen as well as sound the alarm. No subroutines are present in this file.

RTI\_ISR.asm:

The real time interrupt is in control of all delay flags and counters which are used to control certain peripherals. The RTI also has its own counter which is used when incrementing the real time clock. No subroutines are present in this file.

alarmPanicButton.asm:

This file contains the subroutine which is executed when the system is in an alarm state. The subroutine will prompt the user with the password screen and will check if the vault is open so that it closes.

bankVault.asm:

This file contains a subroutine which is called in main, this subroutine will check if the vault key has been inputted and will then check if the conditions are met to open the vault.

BINConvertBCD.asm:

This file contains two subroutines which are called whenever the program needs to convert a binary number to binary coded decimal or vice versa. Passed parameters are used whenever the subroutines are called since it works for all input.

Keypad.asm:

This file contains the subroutine that will check for keypresses through the keypad, it is very similar to the keypad subroutine written in lab 5 but with small differences.

KeypadControls.asm:

This file contains a subroutine which will process data provided through the keypad. It is called whenever a keypress has been detected and the keypress is passed as a parameter to this subroutine. It also controls which menu is shown on the LCD through flags.

LCDcontrols.asm:

This file contains a subroutine which will manually move characters onto the display depending on the status of the system. The screen is determined through flags set by the KeypadControls.asm subroutine.

LEDcontrol.asm:

This file contains a subroutine which controls the lighting of the system through the LED’s. Certain LED’s stay on or flash depending on system.

PswdControl.asm:

This file contains two subroutines, one of which oversees setting a new password and will store it to the appropriate variable. The other subroutine is used when evaluating the user’s input password with the saved one.

personEmergency.asm:

This file contains the subroutine which oversees detecting when a person has entered the vault while it’s open. It is also initiates the emergency exit when the vault closes on a person. This subroutine also controls the LED which reflects if a person is inside or not.

portConfiguring.asm:

This file contains one subroutine which is only called once before booting up. This subroutine will configure the ports appropriately and will assign the appropriate initial values to some variables and flags, the rest of the variables are initialized in main.asm

Potentiometer.asm:

This file contains one subroutine which will read input from the potentiometer and pass it into another subroutine which will convert it and scaled it appropriately. This subroutine also calculates the duty cycle of the DC motor.

SettingsControl.asm:

This file contains one subroutine which is called when the user chooses to change settings from the menu. It will take user input and assign it to appropriate variables in the program to achieve a change of initial values.

StepperMotor.asm:

This file contains one subroutine which is in control of the stepper motor. Flags are checked before continuing with the subroutine and the program will determine which direction to turn the motor and with what speed.

transactionalLogic.asm:

This file contains a subroutine which is in control of transactions that take place in the vault. It will check whether the amount entered is appropriate and assign to certain variables. It also contains two further subroutines which are executed depending on the choice of the user to withdraw or deposit.

Main.asm:

All variables are declared in main, and some are initialized in this file as well. Main will run an infinite loop which jumps through the various subroutines mentioned above which will then check flags to see if they should be executed or not. Eight subroutines are called in the infinite loop, each of which oversees a peripheral or data taken from peripherals.

**Appendix III**

**Code:**

Main.asm:

INCLUDE 'derivative.inc'

XDEF Entry, \_Startup

XREF \_\_SEG\_END\_SSTACK ; symbol defined by the linker for the end of the stack

; LCD and Pot References

XREF SendsChr,PlayTone,display\_string,init\_LCD,read\_pot

; ISR Refrences

XREF IRQ\_ISR,RTI\_ISR

XDEF RTI\_FLG

; Subroutines Refrences

XREF CONFIG,LCD\_Control,keypad,buttonLogic,Convert\_BIN,checkVaultState,lighting,stepMotor,pCheck,potential

; Global variable for LCD display

XDEF disp

; Timing variables

XDEF hours,minutes,seconds,day,RTI\_counter,seconds\_tens,seconds\_ones,days,dayIndex

XDEF minutes\_tens,minutes\_ones,hours\_tens,hours\_ones,LCDelay,LCDelayFlag

; keypad variables

XDEF menuOption,led,var,keyPress,hexDel,hexDelFlag

; password variables

XDEF password,password\_mask,pswd\_bcd,pswd\_counter,enter\_set,pswd\_bcd\_check

; temperatures values

XDEF des\_temp\_tens,des\_temp\_ones,des\_temp,curr\_temp\_tens,curr\_temp\_ones,currentTemp,des\_temp\_hundreds

XDEF des\_temp\_hundreds\_temp,des\_temp\_tens\_temp,des\_temp\_ones\_temp,tempChange

; flag values

XDEF set\_settings,exitFlag

; Vault variables

XDEF vault\_state,money,money\_bcd,bankCounter,P\_flag

; stepMotor variables

XDEF motorDelay,delayFlag,seq\_ind,ani\_length

; transaction variables

XDEF tmoney,tmoney\_bcd,d\_w

; Alarm variables

XDEF alarmFlag,alarmDelay,alarmDelayFlag

XREF alarmActivate

; Dc motor Variables

XDEF Ton,Toff,DCmotorCounter

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* VARIABLE DECLARATION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MY\_variables: SECTION

disp: ds.b 33 ; LCD display

counter: ds.b 1

hours: ds.b 1 ; 0-24

minutes: ds.b 1 ; 0-60

seconds: ds.b 1 ; 0-60

day: ds.b 3 ; day variable ex: 'FRI'

RTI\_counter ds.b 2

days ds.b 24 ; variable containing days MON-SUN

dayIndex ds.b 1

LCDelay: ds.b 2

LCDelayFlag: ds.b 1

; timing variables ASKII equivalent

seconds\_tens ds.b 1

seconds\_ones: ds.b 1

minutes\_tens ds.b 1

minutes\_ones: ds.b 1

hours\_tens ds.b 1

hours\_ones: ds.b 1

;keypad variables

menuOption: ds.b 1

var: ds.b 1

led: ds.b 2

keyPress: ds.b 1

; Password variables

password: ds.b 2

password\_mask: ds.b 4

pswd\_counter: ds.b 1

pswd\_bcd: ds.b 4 ; BCD password

pswd\_bcd\_check: ds.b 4

enter\_set: ds.b 1

; Temperatures variables

des\_temp\_hundreds: ds.b 1

des\_temp\_tens: ds.b 1

des\_temp\_ones: ds.b 1

des\_temp\_hundreds\_temp: ds.b 2

des\_temp\_tens\_temp: ds.b 2

des\_temp\_ones\_temp: ds.b 2

des\_temp: ds.b 1

curr\_temp\_tens: ds.b 1

curr\_temp\_ones: ds.b 1

currentTemp: ds.b 1

tempChange: ds.b 1

; flag variables

set\_settings: ds.b 1

hexDel: ds.b 2

hexDelFlag: ds.b 1

exitFlag: ds.b 1

; Bank vault variables

vault\_state: ds.b 1 ; 0: Closed 1: Open

bankCounter: ds.b 1

P\_flag: ds.b 1

; Transaction variables

money: ds.b 2 ; max 9999

money\_bcd: ds.b 4 ; \_ \_ \_ \_

d\_w: ds.b 1 ; flag: either deposit or withdraw

tmoney: ds.b 2

tmoney\_bcd ds.b 4

; stepMotor variables

motorDelay: ds.b 2

delayFlag: ds.b 1

seq\_ind: ds.b 1

ani\_length: ds.b 1

; Alarm variables

alarmFlag: ds.b 1

alarmDelay: ds.b 2

alarmDelayFlag: ds.b 1

; DC motor variables

Ton: ds.b 1 ; max = 75

Toff: ds.b 1

DCmotorCounter: ds.b 1

my\_constant: SECTION

RTI\_CTL: equ $003B ;not needed

RTI\_ENA: equ $0038 ;CRGINT

RTI\_FLG: equ $0037 ;CRGFLG

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* INITIALIZATIONS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MyCode: SECTION

Entry:

\_Startup:

;------------------- Device initialization ------------------------

SEI ; Disable interrupts for Configurations

LDS #\_\_SEG\_END\_SSTACK ;initialize the stack pointer

; IRQ CONFIG

movb #$C0,INTCR ;enable IRQ - must have vector

; RTI CONFIG

movb #$80,RTI\_ENA ;enable rti

movb #$10,RTI\_CTL ;set div: .128ms for speaker

JSR CONFIG ; PORTS CONFIG

; Initializing Timing variables

MOVB #10,hours

MOVB #30,minutes

MOVB #0,seconds

MOVW #0,RTI\_counter

MOVB #0,seconds\_tens

MOVB #0,seconds\_ones

MOVB #0,minutes\_tens

MOVB #0,minutes\_ones

MOVB #0,hours\_tens

MOVB #0,hours\_ones

MOVB #0,dayIndex

CLR LCDelay

CLR LCDelayFlag

LDX #days

STX day ; initializing day = 'MON'

; Initializing keypad variables

MOVB #0,keyPress

MOVB #0,menuOption ; initialized to mainMenu

MOVB #0,hexDel

MOVB #0,hexDelFlag

; Inititializing password

CLR password

CLR password+1

MOVB #'\_',password\_mask

MOVB #'\_',password\_mask+1

MOVB #'\_',password\_mask+2

MOVB #'\_',password\_mask+3

MOVB #0,pswd\_counter

MOVB #0,enter\_set

JSR init\_LCD ; initialize LCD screen

; Initialzing Temperature

MOVB #8,currentTemp

MOVB #0,curr\_temp\_tens

MOVB #0,curr\_temp\_ones

MOVB #'0',des\_temp

MOVB #'0',des\_temp\_hundreds

MOVB #'0',des\_temp\_tens

MOVB #'0',des\_temp\_ones

CLR tempChange

; Initializing flags;

CLR set\_settings

CLR d\_w

CLR P\_flag

CLR exitFlag

; Initialzing vault variables

CLR vault\_state

MOVW #0,money

MOVW #0,tmoney

MOVB #'0',money\_bcd

MOVB #'0',money\_bcd+1

MOVB #'0',money\_bcd+2

MOVB #'0',money\_bcd+3

MOVB #'\_',tmoney\_bcd

MOVB #'\_',tmoney\_bcd+1

MOVB #'\_',tmoney\_bcd+2

MOVB #'\_',tmoney\_bcd+3

CLR bankCounter

; Initializing stepMotor variables

MOVW #0,motorDelay

CLR delayFlag

CLR seq\_ind

MOVB #$FF,ani\_length

; Initializing alarm variables

CLR alarmFlag

MOVW #0,alarmDelay

CLR alarmDelayFlag

; Initializing DC motor variables

CLR Ton

CLR Toff

CLR DCmotorCounter

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MAIN CODE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;SEI

CLI ; Enable Interrupts

checkCounter: macro ; macro for implementing delay

LDX RTI\_counter ; uses an RTI counter

CPX \1

BNE \2

endm

loop:

;

JSR LCD\_Control ; updates display every 100ms

hexpad:

JSR keypad

vault: checkCounter #7811,lights

JSR checkVaultState ; 4 under these dont work if alarm is on

lights: JSR lighting

stMotor: JSR stepMotor

pshbutton: JSR pCheck ; psh button subroutine

JSR alarmActivate ; alarm subroutine

JSR potential

next: BRA loop ; infinite loop for code testing

IRQ\_ISR.asm:

XDEF IRQ\_ISR

XREF SendsChr

XREF alarmFlag,ani\_length,PRT\_S,menuOption,vault\_state,motorDelay,delayFlag,seq\_ind

IRQ\_ISR:

LDAA #$FF ; set alarm flag

STAA alarmFlag

LDAA #2

PSHA

LDD #0

JSR SendsChr

LEAS 1,sp

CLR seq\_ind

CLR ani\_length

CLR motorDelay

CLR delayFlag

BCLR PRT\_S,#$FF

RTI

RTI\_ISR.asm:

XDEF RTI\_ISR

XREF LCDelay,LCDelayFlag,PlayTone

XREF RTI\_FLG,PRT\_T,PRT\_S

XREF hours,minutes,seconds,day,RTI\_counter,days

XREF keyPress,menuOption,pswd\_counter

XREF vault\_state,motorDelay,delayFlag,ani\_length,hexDel,hexDelFlag

XREF alarmDelay,alarmDelayFlag,alarmFlag

; Dc motor Variables

XREF Ton,Toff,DCmotorCounter

incrementSecondsBuffer: LBRA incrementSeconds

RTI\_ISR: LDX RTI\_counter

INX

STX RTI\_counter

CPX #7812 ; 7812\*(.128ms)= 1s

BHS incrementSecondsBuffer

continue: LDX alarmDelay

INX

STX alarmDelay

CPX #7811

BLO lcDelay

LDX #0

STX alarmDelay

BSET alarmDelayFlag,#$FF

lcDelay:

LDX LCDelay

INX ; was 2000

STX LCDelay ; might be mistake

CPX #2000 ; set flag for motor delay (slow)

BLO motDelay

LDX #0

STX LCDelay

BSET LCDelayFlag,#$FF

; trial

motDelayQuick:

BRCLR alarmFlag,#$FF, motDelay

LDX motorDelay

INX

STX motorDelay

CPX #469

BLO motDelay

LDX #0

STX motorDelay

BSET delayFlag,#$FF

BRA padDelay

motDelay:

;BRSET alarmFlag,#$FF,padDelay

LDX motorDelay

INX

STX motorDelay

CPX #250 ; set flag for motor delay (slow)

BLO padDelay

LDX #0

STX motorDelay

BSET delayFlag,#$FF

padDelay: LDX hexDel

INX

STX hexDel

CPX #234

BLO inc\_screen

LDX #0

STX hexDel

BSET hexDelFlag,#$FF

inc\_screen: LDAA menuOption

CMPA #4 ; show incorrect screen for a second

BNE checkKey

;CLR RTI\_counter

LDX RTI\_counter

CPX #7811

BLO checkKey

LDAA #1

STAA menuOption

checkKey: BRSET PRT\_T,#$7,openVault ; vault key has been entered

;CLR vault\_state ; comment out and test in lab

checkClosed:

LDAA vault\_state

CMPA #$FF ; if vault is open then check switches for closed state

BNE clrPSWD

BRCLR PRT\_T,#$7,closeVault

clrPSWD: LDAA pswd\_counter

CMPA #5

BLS sMotorP

CLR pswd\_counter

sMotorP: LDAA ani\_length

CMPA #$FF

BEQ alarmLED

CMPA #100 ; 250 rounds of spin

BNE alarmLED

BSET ani\_length,#$FF

alarmLED: BRCLR alarmFlag,#$FF,DCmotor

LDX RTI\_counter

CPX #4000 ; approx 1/2 second

BLO LEDoff

BSET PRT\_S,#$FF

BRA DCmotor

LEDoff: BCLR PRT\_S,#$FF

DCmotor: BRSET alarmFlag,#$FF,motorOffBuffer

INC DCmotorCounter

LDAA DCmotorCounter

CMPA Ton

LBLS motorOn

CMPA #45 ; was 45

LBLS motorOff

CLR DCmotorCounter

exit: LDAA alarmFlag

CMPA #$FF

BNE noTone

JSR PlayTone

noTone

BSET RTI\_FLG, #$80

RTI

motorOffBuffer: LBRA motorOff

openVault: LDAA vault\_state

BNE checkClosed

LDAA #1

STAA vault\_state

BRA checkClosed

closeVault: LDAA #$CC ; state CC means initiate close vault

STAA vault\_state

CLR ani\_length

LDAA #1

STAA menuOption

BRA clrPSWD

incrementSeconds:

CLR RTI\_counter

INC seconds

LDAA seconds

CMPA #60 ; check if a minute passed

BLO continueB

incrementMinute:

CLR seconds

INC minutes

LDAA minutes

CMPA #60 ; check if a hour passed

BLO continueB

incrementHour:

CLR minutes

INC hours ; check if a day passed

LDAA hours

CMPA #24

BLO continueB

incrementDay:

CLR hours

LDAB #3

LDX day

ABX

STX day

LDAA 0,x

CMPA #$30

BEQ resetWeek

LBRA continue

resetWeek: LDX #days

STX day

LBRA continue

continueB: LBRA continue

motorOn: BSET PRT\_T,#8

LBRA exit

motorOff: BCLR PRT\_T,#8

LBRA exit

alarmPanicButton.asm

XDEF alarmActivate

XREF alarmFlag,PRT\_S,menuOption,vault\_state

XREF ani\_length,PRT\_P,seq\_ind,delayFlag

MY\_Constants: SECTION

Qsequence: dc.b $C,$14,$12,$A

alarmActivate:

LDAA alarmFlag

CMPA #$FF

BNE exit

LDAA vault\_state

CMPA #$FF

BNE next

BRSET ani\_length,#$FF,vaultCloseDone ; if ani\_length = FF then motor is disabled

BRCLR delayFlag,#$FF,next ; if delay not reached exit

LDX #Qsequence

LDAB seq\_ind

ABX

LDAA 0,x

STAA PRT\_P

INC seq\_ind

CLR delayFlag

INC ani\_length

LDAA seq\_ind

CMPA #4

BLO next

CLR seq\_ind

next:

LDAA #$2F

STAA menuOption

exit:

RTS

vaultCloseDone:

LDAA #0

STAA vault\_state

RTS

bankVault.asm:

XDEF checkVaultState,checkConditions

XREF vault\_state,currentTemp,hours

XREF password,password\_mask,pswd\_bcd,pswd\_counter,savePassword,enter\_set,enterPassword,pswd\_bcd\_check

XREF buttonLogic,menuOption,PRT\_P,P\_flag

XREF alarmFlag

; vault has 4 states

; 0 not open, has not been attempted

; 1 vault key has been entered but vault not yet open

; 2 conditions met but password not yet entered

; FF conditions met and password entered so vault is open

; CC initiate closing vault, turn motor ACW then turn off LED

checkVaultState:

; If alarm is on dont run this file

LDAA alarmFlag

CMPA #$FF

BEQ exit

LDAA vault\_state

BEQ exit ; if zero then dont open

CMPA #1 ; if one then check conditions

BEQ checkConditions

CMPA #2 ; if 2 then PSWD prompt

BEQ conditionMet

BRA exit

exit: RTS

checkConditions: ; has to be working day 9AM-5PM, current temp within 2-10

LDAA hours

CMPA #9 ; 9AM

BLO conditionNotMet

CMPA #17 ; 5PM

BHI conditionNotMet

LDAA currentTemp

CMPA #2

BLO conditionNotMet

CMPA #10

BHI conditionNotMet

MOVB #2,vault\_state

RTS

conditionNotMet:

MOVB #0,vault\_state

RTS

conditionMet:

LDAA #$2F

STAA menuOption

RTS

BINconvertBCD.asm:

XREF hours,minutes,seconds,day,seconds\_tens,seconds\_ones

XDEF Convert\_BCD,Convert\_BIN

Convert\_BCD:

LDAA #0

LDAB 2,sp ; taking parameter

LDX #10 ; divide by ten to get tens

IDIV

XGDX ; Quotient in D Remainder in X

ADDB #$30

STAB -1,sp

XGDX ; Remainder in D

LDX #1

IDIV ; divide by one to get ones

XGDX ; Quotient in D

ADDB #$30

STAB -2,sp

RTS

Convert\_BIN:

LDAA 3,sp

SUBA #$30

LDAB #10

MUL

LDAA 2,sp

SUBA #$30

ABA

STAA -1,sp

RTS

Keypad.asm:

XREF PRT\_U

XDEF keypad

XREF menuOption,led,var,keyPress,buttonLogic,hexDel,hexDelFlag

my\_constant: SECTION

sequence: dc.b $70,$B0,$D0,$E0

table: dc.b $eb, $77, $7b, $7d, $b7, $bb, $bd, $d7, $db, $dd, $e7, $ed, $7e, $be, $de, $ee

keypad: BRCLR hexDelFlag,#$FF,return

start LDX #sequence ; load sequence for checking rows

LDY #0 ; counter

loop: CPY #4

BEQ return\_clr

LDAA 1,x+

STAA PRT\_U ; store seq into port U

JSR debounce

INY ; increment counter

LDAB PRT\_U

STAB var ; store port U to var for look up table

ANDB #$F ; Mask upper nibble

CMPB #$F ; if F repeat loop

BEQ loop

BRA looktable

looktable: LDY #0000

LDX #table ; load table sequence into X

bigLoop: CPY #16 ; look up table routine from lab 4

BEQ start

LDAA 1,x+

CMPA var

BEQ result

INY

BRA bigLoop

result:

STY led

LDAA led+1

STAA keyPress ; store key Pressed

JSR buttonLogic

;BRA release

release: LDAB PRT\_U

ANDB #$F ; is released

CMPB #$F

BEQ return\_clr

BRA release

debounce: PSHY ; Push Y to keep count

LDY #500 ; of loop counter

delayLoop: DEY ; delay of 1 ms

BNE delayLoop

PULY ; pull y and return

RTS

return\_clr: CLR hexDelFlag

return: RTS ; return from keypad subroutine

keypadControl.asm:

XDEF buttonLogic

XREF menuOption,keyPress

; password variables

XREF password,password\_mask,pswd\_bcd,pswd\_counter,savePassword,enter\_set,enterPassword,pswd\_bcd\_check

XREF settingsControl,RTI\_counter,vault\_state,bankCounter

XREF ani\_length

XREF dep,with,d\_w,alarmFlag,PRT\_S

Backexit: RTS

alarmPrompt:

LDAA #$2F

STAA menuOption

INC pswd\_counter

JSR enterPassword

LDAA pswd\_counter

CMPA #4

BNE Backexit

MOVB #'\_',password\_mask

MOVB #'\_',password\_mask+1

MOVB #'\_',password\_mask+2

MOVB #'\_',password\_mask+3

LDAA pswd\_bcd ; compare input with actual password

CMPA pswd\_bcd\_check

BNE incorrectAlarm ; if not equal show incorrect screen

LDAA pswd\_bcd+1

CMPA pswd\_bcd\_check+1

BNE incorrectAlarm

LDAA pswd\_bcd+2

CMPA pswd\_bcd\_check+2

BNE incorrectAlarm

LDAA pswd\_bcd+3

CMPA pswd\_bcd\_check+3

BNE incorrectAlarm

CLR vault\_state

CLR pswd\_counter

CLR alarmFlag

BCLR PRT\_S,#$FF

LDAA #$1 ; if equal carry on with changing settings

STAA menuOption

LBRA exit

incorrectAlarm:

CLR RTI\_counter ; reset to check for a second

CLR vault\_state ; maybe works test pls

CLR pswd\_counter ; reset counter for next popup

LBRA exit

buttonLogic:

BRA AssignScreen

alarmPromptBuffer: LBRA alarmPrompt

screen1buffer: LBRA screen1

screen2buffer: LBRA screen2

screen3buffer: LBRA screen3

screen4buffer: LBRA screen4

csConfirmbuffer: LBRA csConfirm

pswd\_popupbuffer: LBRA pswd\_popup

vault\_pswd\_promptBuffer: LBRA vault\_pswd\_prompt

transactionBuffer: LBRA transaction

depositBuffer: LBRA deposit

withdrawBuffer: LBRA withdraw

AssignScreen:

BRSET alarmFlag,#$FF,alarmPromptBuffer

LDAA vault\_state

CMPA #2

BEQ vault\_pswd\_promptBuffer

LDAA menuOption

CMPA #0

BEQ pswd\_popupbuffer

CMPA #1

BEQ screen1buffer ; check which screen im currently on

CMPA #2

BEQ screen2buffer

CMPA #3

BEQ screen3buffer

CMPA #$1F

BEQ screen4buffer

CMPA #$2F

BEQ csConfirmbuffer

LDAA d\_w

CMPA #0

BEQ transactionBuffer

CMPA #1

BEQ depositBuffer

CMPA #$FF

BEQ withdrawBuffer

BRA exit

screen1: ; main menu

LDAA keyPress

CMPA #$B

BEQ mmTOcs

LBRA exit

mmTOcs: LDAA #2

STAA menuOption ; scroll from main menu to change settings screen

BRA exit

screen2: ; settings

LDAA keyPress

CMPA #$A

BEQ csTOmm

CMPA #$B

BEQ csTOcp

CMPA #$F

BRA csConfirm

BRA exit

csTOmm: LDAA #1

STAA menuOption

BRA exit

csTOcp: LDAA #3

STAA menuOption

BRA exit

exit: RTS

csConfirm:

LDAA #$2F

STAA menuOption

;alarmPrompt:

JSR enterPassword

INC pswd\_counter

LDAA pswd\_counter

CMPA #5

BNE exit

MOVB #'\_',password\_mask

MOVB #'\_',password\_mask+1

MOVB #'\_',password\_mask+2

MOVB #'\_',password\_mask+3

LDAA pswd\_bcd ; compare input with actual password

CMPA pswd\_bcd\_check

BNE incorrect ; if not equal show incorrect screen

LDAA pswd\_bcd+1

CMPA pswd\_bcd\_check+1

BNE incorrect

LDAA pswd\_bcd+2

CMPA pswd\_bcd\_check+2

BNE incorrect

LDAA pswd\_bcd+3

CMPA pswd\_bcd\_check+3

BNE incorrect

LDAA #$1F ; if equal carry on with changing settings

STAA menuOption

BRA exit

incorrect:

CLR RTI\_counter ; reset to check for a second

CLR vault\_state ; maybe works test pls

CLR pswd\_counter ; reset counter for next popup

LDAA #$4

STAA menuOption ; 4 is incorrect screen

BRA exit

vault\_pswd\_prompt:

LDAA #$2F

STAA menuOption

JSR enterPassword

INC pswd\_counter

LDAA pswd\_counter

CMPA #5

BNE exit

MOVB #'\_',password\_mask

MOVB #'\_',password\_mask+1

MOVB #'\_',password\_mask+2

MOVB #'\_',password\_mask+3

LDAA pswd\_bcd ; compare input with actual password

CMPA pswd\_bcd\_check

BNE incorrect ; if not equal show incorrect screen

LDAA pswd\_bcd+1

CMPA pswd\_bcd\_check+1

BNE incorrect

LDAA pswd\_bcd+2

CMPA pswd\_bcd\_check+2

BNE incorrect

LDAA pswd\_bcd\_check+3

CMPA pswd\_bcd\_check+3

BNE incorrect

LDAA #$FF

STAA vault\_state

LDAA #$5 ; if equal carry on with step motor spin

STAA menuOption

CLR ani\_length

LBRA exit

cpConfirm: LDAA #0

STAA menuOption

LBRA exit ;added

screen3: ; pwsd

LDAA keyPress

CMPA #$A

BEQ cpTOcs

CMPA #$F ; added

BEQ cpConfirm ;added

LBRA exit

cpTOcs: LDAA #2

STAA menuOption

LBRA exit

screen4: JSR settingsControl

pswd\_popup:

JSR savePassword

LDAA pswd\_counter

CMPA #4

BNE exit\_end

CLR pswd\_counter

BSET enter\_set,#1

LDAA #1

STAA menuOption

MOVB #'\_',password\_mask

MOVB #'\_',password\_mask+1

MOVB #'\_',password\_mask+2

MOVB #'\_',password\_mask+3

LBRA exit

transaction:

LDAA keyPress

CMPA #1

BEQ deposit

CMPA #2

BEQ withdraw

BRA exit\_end

deposit:

LDAA #6

STAA menuOption

LDAA #1

STAA d\_w

;BSET d\_w,#1 ; added+3

LDAA keyPress

CMPA #9

BHI exit\_end

;BSET d\_w,#1

JSR dep

BRA exit\_end

withdraw:

LDAA #6

STAA menuOption

LDAA #$FF

STAA d\_w

LDAA keyPress

CMPA #9

BHI exit\_end

;CLR d\_w

JSR dep

BRA exit\_end

exit\_end: RTS

LCDControl.asm:

; LCD and Pot References

XREF SendsChr,PlayTone,display\_string,init\_LCD,disp,read\_pot,pot\_value

XDEF LCD\_Control

; Timing variables

XREF hours,minutes,seconds,day,seconds\_tens,seconds\_ones,minutes\_tens,minutes\_ones,hours\_tens,hours\_ones

; Subroutines

XREF Convert\_BCD

; keypad variables

XREF menuOption,keyPress

; password variables

XREF password,password\_mask,pswd\_bcd

; temperature variables

XREF des\_temp,curr\_temp\_tens,curr\_temp\_ones,currentTemp,des\_temp\_tens,des\_temp\_ones,des\_temp\_hundreds

XREF des\_temp\_hundreds\_temp,des\_temp\_tens\_temp,des\_temp\_ones\_temp,tempChange

; vault variables

XREF vault\_state,money,money\_bcd,tmoney,tmoney\_bcd

XREF LCDelay,LCDelayFlag

backexit: RTS

LCD\_Control:

LDAA menuOption

BRCLR menuOption,#$FF,EPSWDBuffer

CMPA #1

BEQ changeClock

CMPA #2

BEQ settingsBuffer

CMPA #3

BEQ pswdBuffer

CMPA #4

BEQ incPSWDbuffer

CMPA #$1F

BEQ changingValuesBuffer

CMPA #$2F

BEQ EPSWDBuffer

CMPA #$5

BEQ inVaultBuffer

CMPA #6

BEQ DWBuffer

BRA carryOn

EPSWDBuffer:

LBRA EnterPSWD

settingsBuffer:

LBRA changeSettings

pswdBuffer:

LBRA changePswd

changingValuesBuffer:

LBRA changingValues

incPSWDbuffer:

LBRA incPSWD

inVaultBuffer:

LBRA inVault

DWBuffer:

LBRA DW

carryOn:

LBRA EmptyScreen

changeClock:

BRCLR LCDelayFlag,#$FF,backexit

LDAA seconds

PSHA

JSR Convert\_BCD

PULA

LDAA -5,sp

STAA seconds\_ones

LDAA -4,sp

STAA seconds\_tens

LDAA minutes

PSHA

JSR Convert\_BCD

PULA

LDAA -5,sp

STAA minutes\_ones

LDAA -4,sp

STAA minutes\_tens

LDAA hours

PSHA

JSR Convert\_BCD

PULA

LDAA -5,sp

STAA hours\_ones

LDAA -4,sp

STAA hours\_tens

LDAA tempChange

CMPA #0

BNE continueTemp

LDAA currentTemp

PSHA

JSR Convert\_BCD

PULA

LDAA -5,sp

STAA curr\_temp\_ones

LDAA -4,sp

STAA curr\_temp\_tens

LDAA #1

STAA tempChange

continueTemp:

JSR read\_pot

LDD pot\_value

LDAA #0

LDX #100

IDIV

STX des\_temp\_hundreds\_temp

LDX #10

IDIV

STX des\_temp\_tens\_temp

LDX #1

IDIV

STX des\_temp\_ones\_temp

LDD des\_temp\_hundreds\_temp

STAB des\_temp\_hundreds

LDD des\_temp\_tens\_temp

STAB des\_temp\_tens

LDD des\_temp\_ones\_temp

STAB des\_temp\_ones

LDD des\_temp\_hundreds

ADDB #$30

STAB des\_temp\_hundreds

LDD des\_temp\_tens

ADDB #$30

STAB des\_temp\_tens

LDD des\_temp\_ones

ADDB #$30

STAB des\_temp\_ones

mainMenu:

MOVB hours\_tens, disp ; HH

MOVB hours\_ones, disp+1

MOVB #':', disp+2

MOVB minutes\_tens, disp+3 ; MM

MOVB minutes\_ones, disp+4

MOVB #':', disp+5

MOVB seconds\_tens, disp+6 ; SS

MOVB seconds\_ones, disp+7

MOVB #' ', disp+8

LDX day ; DAY

LDAB 0,x

STAB disp+9

LDAB 1,x

STAB disp+10

LDAB 2,x

STAB disp+11

MOVB #' ', disp+12

MOVB #'L', disp+13

MOVB #'A', disp+14

MOVB #'B', disp+15

MOVB curr\_temp\_tens, disp+16 ; 2nd line

MOVB curr\_temp\_ones, disp+17

MOVB #'°', disp+18

MOVB #'C', disp+19

MOVB #' ', disp+20

MOVB #' ', disp+21

MOVB #' ', disp+22

MOVB #'D', disp+23

MOVB #'T', disp+24

MOVB #':', disp+25

MOVB des\_temp\_hundreds, disp+26

MOVB des\_temp\_tens, disp+27

MOVB #'C', disp+28

MOVB #' ', disp+29

MOVB #' ', disp+30

MOVB #'>', disp+31

MOVB #'0', disp+32

LBRA EndFunction

changeSettings:

MOVB #'C', disp

MOVB #'H', disp+1

MOVB #'A', disp+2

MOVB #'N', disp+3

MOVB #'G', disp+4

MOVB #'E', disp+5

MOVB #' ', disp+6

MOVB #'S', disp+7

MOVB #'E', disp+8

MOVB #'T', disp+9

MOVB #'T', disp+10

MOVB #'I', disp+11

MOVB #'N', disp+12

MOVB #'G', disp+13

MOVB #'S', disp+14

MOVB #' ', disp+15

MOVB #'<', disp+16

MOVB #' ', disp+17

MOVB #' ', disp+18

MOVB #' ', disp+19

MOVB #' ', disp+20

MOVB #' ', disp+21

MOVB #' ', disp+22

MOVB #' ', disp+23

MOVB #' ', disp+24

MOVB #' ', disp+25

MOVB #' ', disp+26

MOVB #' ', disp+27

MOVB #' ', disp+28

MOVB #' ', disp+29

MOVB #' ', disp+30

MOVB #'>', disp+31

MOVB #'0', disp+32

LBRA EndFunction

changePswd:

MOVB #' ', disp

MOVB #' ', disp+1

MOVB #'C', disp+2

MOVB #'H', disp+3

MOVB #'A', disp+4

MOVB #'N', disp+5

MOVB #'G', disp+6

MOVB #'E', disp+7

MOVB #' ', disp+8

MOVB #' ', disp+9

MOVB #'P', disp+10

MOVB #'S', disp+11

MOVB #'W', disp+12

MOVB #'D', disp+13

MOVB #' ', disp+14

MOVB #' ', disp+15

MOVB #'<', disp+16

MOVB #' ', disp+17

MOVB #' ', disp+18

MOVB #' ', disp+19

MOVB #' ', disp+20

MOVB #' ', disp+21

MOVB #' ', disp+22

MOVB #' ', disp+23

MOVB #' ', disp+24

MOVB #' ', disp+25

MOVB #' ', disp+26

MOVB #' ', disp+27

MOVB #' ', disp+28

MOVB #' ', disp+29

MOVB #' ', disp+30

MOVB #' ', disp+31

MOVB #'0', disp+32

LBRA EndFunction

EmptyScreen:

MOVB #' ', disp

MOVB #' ', disp+1

MOVB #' ', disp+2

MOVB #' ', disp+3

MOVB #' ', disp+4

MOVB #' ', disp+5

MOVB #' ', disp+6

MOVB #' ', disp+7

MOVB #' ', disp+8

MOVB #' ', disp+9

MOVB #' ', disp+10

MOVB #' ', disp+11

MOVB #' ', disp+12

MOVB #' ', disp+13

MOVB #' ', disp+14

MOVB #' ', disp+15

MOVB #'<', disp+16

MOVB #' ', disp+17

MOVB #' ', disp+18

MOVB #' ', disp+19

MOVB #' ', disp+20

MOVB #' ', disp+21

MOVB #' ', disp+22

MOVB #' ', disp+23

MOVB #' ', disp+24

MOVB #' ', disp+25

MOVB #' ', disp+26

MOVB #' ', disp+27

MOVB #' ', disp+28

MOVB #' ', disp+29

MOVB #' ', disp+30

MOVB #'>', disp+31

MOVB #'0', disp+32

LBRA EndFunction

EnterPSWD:

MOVB #'E', disp

MOVB #'N', disp+1

MOVB #'T', disp+2

MOVB #'E', disp+3

MOVB #'R', disp+4

MOVB #' ', disp+5

MOVB #'P', disp+6

MOVB #'S', disp+7

MOVB #'W', disp+8

MOVB #'D', disp+9

MOVB #':', disp+10

MOVB #' ', disp+11

MOVB #' ', disp+12

MOVB #' ', disp+13

MOVB #' ', disp+14

MOVB #' ', disp+15

MOVB #' ', disp+16

MOVB password\_mask, disp+17

MOVB password\_mask+1, disp+18

MOVB password\_mask+2, disp+19

MOVB password\_mask+3, disp+20

MOVB #' ', disp+21

MOVB #' ', disp+22

MOVB #' ', disp+23

MOVB #' ', disp+24

MOVB #' ', disp+25

MOVB #' ', disp+26

MOVB #' ', disp+27

MOVB #' ', disp+28

MOVB #' ', disp+29

MOVB #' ', disp+30

MOVB #' ', disp+31

MOVB #'0', disp+32

LBRA EndFunction

changingValues:

MOVB hours\_tens, disp ; HH

MOVB hours\_ones, disp+1

MOVB #':', disp+2

MOVB minutes\_tens, disp+3 ; MM

MOVB minutes\_ones, disp+4

MOVB #':', disp+5

MOVB seconds\_tens, disp+6 ; SS

MOVB seconds\_ones, disp+7

MOVB #' ', disp+8

LDX day ; DAY

LDAB 0,x

STAB disp+9

LDAB 1,x

STAB disp+10

LDAB 2,x

STAB disp+11

MOVB #' ', disp+12

MOVB #' ', disp+13

MOVB #' ', disp+14

MOVB #' ', disp+15

MOVB curr\_temp\_tens, disp+16 ; 2nd line

MOVB curr\_temp\_ones, disp+17

MOVB #'°', disp+18

MOVB #'C', disp+19

MOVB #' ', disp+20

MOVB #' ', disp+21

MOVB #' ', disp+22

MOVB #' ', disp+23

MOVB #' ', disp+24

MOVB #' ', disp+25

MOVB #' ', disp+26

MOVB #' ', disp+27

MOVB #' ', disp+28

MOVB #' ', disp+29

MOVB #' ', disp+30

MOVB #' ', disp+31

MOVB #'0', disp+32

LBRA EndFunction

incPSWD:

MOVB #'I', disp

MOVB #'N', disp+1

MOVB #'C', disp+2

MOVB #'O', disp+3

MOVB #'R', disp+4

MOVB #'R', disp+5

MOVB #'E', disp+6

MOVB #'C', disp+7

MOVB #'T', disp+8

MOVB #' ', disp+9

MOVB #'P', disp+10

MOVB #'S', disp+11

MOVB #'W', disp+12

MOVB #'D', disp+13

MOVB #' ', disp+14

MOVB #' ', disp+15

MOVB #' ', disp+16

MOVB #' ', disp+17

MOVB #' ', disp+18

MOVB #' ', disp+19

MOVB #' ', disp+20

MOVB #' ', disp+21

MOVB #' ', disp+22

MOVB #' ', disp+23

MOVB #' ', disp+24

MOVB #' ', disp+25

MOVB #' ', disp+26

MOVB #' ', disp+27

MOVB #' ', disp+28

MOVB #' ', disp+29

MOVB #' ', disp+30

MOVB #' ', disp+31

MOVB #'0', disp+32

LBRA EndFunction

inVault:

MOVB #'1', disp

MOVB #')', disp+1

MOVB #'D', disp+2

MOVB #'E', disp+3

MOVB #'P', disp+4

MOVB #'O', disp+5

MOVB #'S', disp+6

MOVB #'I', disp+7

MOVB #'T', disp+8

MOVB #' ', disp+9

MOVB #'$', disp+10

MOVB money\_bcd, disp+11

MOVB money\_bcd+1, disp+12

MOVB money\_bcd+2, disp+13

MOVB money\_bcd+3, disp+14

MOVB #' ', disp+15

MOVB #'2', disp+16

MOVB #')', disp+17

MOVB #'W', disp+18

MOVB #'I', disp+19

MOVB #'T', disp+20

MOVB #'H', disp+21

MOVB #'D', disp+22

MOVB #'R', disp+23

MOVB #'A', disp+24

MOVB #'W', disp+25

MOVB #' ', disp+26

MOVB #' ', disp+27

MOVB #' ', disp+28

MOVB #' ', disp+29

MOVB #' ', disp+30

MOVB #' ', disp+31

MOVB #'0', disp+32

LBRA EndFunction

DW:

MOVB #'A', disp

MOVB #'M', disp+1

MOVB #'O', disp+2

MOVB #'U', disp+3

MOVB #'N', disp+4

MOVB #'T', disp+5

MOVB #':', disp+6

MOVB #' ', disp+7

MOVB #'$', disp+8

MOVB tmoney\_bcd, disp+9

MOVB tmoney\_bcd+1, disp+10

MOVB tmoney\_bcd+2, disp+11

MOVB tmoney\_bcd+3, disp+12

MOVB #' ', disp+13

MOVB #' ', disp+14

MOVB #' ', disp+15

MOVB #' ', disp+16

MOVB #' ', disp+17

MOVB #' ', disp+18

MOVB #' ', disp+19

MOVB #' ', disp+20

MOVB #' ', disp+21

MOVB #' ', disp+22

MOVB #' ', disp+23

MOVB #' ', disp+24

MOVB #' ', disp+25

MOVB #' ', disp+26

MOVB #' ', disp+27

MOVB #' ', disp+28

MOVB #' ', disp+29

MOVB #' ', disp+30

MOVB #' ', disp+31

MOVB #'0', disp+32

LBRA EndFunction

EndFunction:

LDD #disp ; Load Address of string as parameter

JSR display\_string

CLR LCDelayFlag

RTS

LEDcontrols.asm:

XDEF lighting

XREF vault\_state,menuOption,PRT\_S,RTI\_counter,P\_flag

XREF alarmFlag

exit: RTS

lighting:

LDAA alarmFlag

CMPA #$FF

BEQ exit

LDAA vault\_state

BRSET vault\_state,#$FF,vaultOpen

BRCLR vault\_state,#$FF,vaultClosed

CMPA #$CC

BEQ flashLED

last:

LDX RTI\_counter

CPX #1500

BLO off

BSET PRT\_S,#$80

;LDAA #$80 ; light last led periodically to indicate

;STAA PRT\_S ; idle state

RTS

off:

BCLR PRT\_S,#$80

RTS

flashLED:

LDX RTI\_counter

CPX #1500

BLO flashOff

BSET PRT\_S,#$1

BRA last

flashOff:

BCLR PRT\_S,#$1

BRA last

vaultOpen:

BSET PRT\_S,#$1 ; light first LED to indicate door is open

BRA last

vaultClosed:

BCLR PRT\_S,#$1

BRA last

passwordControl.asm:

XDEF savePassword,enterPassword

XREF menuOption,keyPress

; password variables

XREF password,password\_mask,pswd\_bcd,pswd\_counter,pswd\_bcd\_check

savePassword:

pswd\_popup:

LDAA pswd\_counter

BNE second\_digit

first\_digit: ; counter = 0

INC pswd\_counter

LDAA keyPress

STAA pswd\_bcd

MOVB #'\*',password\_mask

BRA exit

second\_digit: ; counter = 1

LDAA pswd\_counter

CMPA #1

BNE third\_digit

INC pswd\_counter

LDAA keyPress

STAA pswd\_bcd+1

MOVB #'\*',password\_mask+1

BRA exit

third\_digit: ; counter = 2

LDAA pswd\_counter

CMPA #2

BNE fourth\_digit

INC pswd\_counter

LDAA keyPress

STAA pswd\_bcd+2

MOVB #'\*',password\_mask+2

BRA exit

fourth\_digit: ; counter = 3

LDAA pswd\_counter

CMPA #3

BNE exit

INC pswd\_counter

LDAA keyPress

STAA pswd\_bcd+3

MOVB #'\*',password\_mask+3

BRA exit

exit: RTS

enterPassword:

LDAA pswd\_counter

BEQ exit

CMPA #1

BNE second\_digit\_check

first\_digit\_check: ; counter = 1

;INC pswd\_counter

LDAA keyPress

STAA pswd\_bcd\_check

MOVB #'\*',password\_mask

BRA exit

second\_digit\_check: ; counter = 2

LDAA pswd\_counter

CMPA #2

BNE third\_digit\_check

;INC pswd\_counter

LDAA keyPress

STAA pswd\_bcd\_check+1

MOVB #'\*',password\_mask+1

BRA exit

third\_digit\_check: ; counter = 3

LDAA pswd\_counter

CMPA #3

BNE fourth\_digit\_check

;INC pswd\_counter

LDAA keyPress

STAA pswd\_bcd\_check+2

MOVB #'\*',password\_mask+2

BRA exit

fourth\_digit\_check: ; counter = 4

LDAA pswd\_counter

CMPA #4

BNE exit

;INC pswd\_counter

LDAA keyPress

STAA pswd\_bcd\_check+3

MOVB #'\*',password\_mask+3

BRA exit

personEmergency.asm:

XDEF pCheck

XREF vault\_state,PRT\_P,set\_settings,P\_flag,PRT\_S

XREF ani\_length,opening,closing,delayFlag,exitFlag,menuOption

XREF alarmFlag

; if vault is open check for push button clicks

; if vault is closed and bit 1 of PRT is on then initiate

; emergency exit

exit: LBRA leave

pCheck:

; Check if alarm is on

LDAA alarmFlag

CMPA #$FF

BEQ exit

LDAA exitFlag

CMPA #2

BEQ exitClosing

LDAA vault\_state

CMPA #$FF

BEQ pshButton

CMPA #0

BEQ emergencyExit

pshButton:

LDAA PRT\_P

;ANDA #$F

CMPA #$C

BNE leave

relay:

LDAA PRT\_P

CMPA #$2C

BNE relay

INC P\_flag

BRSET P\_flag,#1,odd

BCLR PRT\_S,#2 ; for even clicks

RTS

odd: BSET PRT\_S,#2 ; for odd clicks

RTS

emergencyExit:

BRCLR PRT\_S,#2,leave

LDAA exitFlag

CMPA #1

BEQ exitOpening

LDAA #1

STAA exitFlag

CLR ani\_length

CLR delayFlag

exitOpening:

JSR opening

LDAA ani\_length

CMPA #$FF

BEQ personOut

BRA leave

personOut:

BCLR PRT\_S,#2

CLR ani\_length

LDAA #2

STAA exitFlag

CLR delayFlag

BRA exitClosing

exitClosing:

JSR closing

LDAA ani\_length

CMPA #$FF

BEQ emergencyDone

BRA leave

emergencyDone:

CLR exitFlag

CLR P\_flag

CLR vault\_state

LDAA #1

STAA menuOption

leave: RTS

portConfiguring.asm:

; File used to initialize all ports

XDEF CONFIG,PRT\_U,PRT\_P,PRT\_S,PRT\_T

XREF days

MY\_variables: SECTION

MY\_constants: SECTION

PRT\_P: EQU $258

P\_DDR: EQU $25A

PRT\_S: EQU $248

S\_DDR: EQU $24A

PRT\_T: EQU $240

T\_DDR: EQU $242

PRT\_U: EQU $268

U\_DDR: EQU $26A

U\_ENB: EQU $26C

U\_PLR: EQU $26D

CONFIG: ; Keypad Config

BSET U\_DDR, #$F0

BSET U\_PLR, #$F0

BSET U\_ENB, #$0F

; Port P Config

BSET P\_DDR, #$1E

; Port S Config

BSET S\_DDR, #$FF

; Port T Config

BSET T\_DDR, #$28

MOVB #'M', days

MOVB #'O', days+1

MOVB #'N', days+2

MOVB #'T', days+3

MOVB #'U', days+4

MOVB #'E', days+5

MOVB #'W', days+6

MOVB #'E', days+7

MOVB #'D', days+8

MOVB #'T', days+9

MOVB #'H', days+10

MOVB #'U', days+11

MOVB #'F', days+12

MOVB #'R', days+13

MOVB #'I', days+14

MOVB #'S', days+15

MOVB #'A', days+16

MOVB #'T', days+17

MOVB #'S', days+18

MOVB #'U', days+19

MOVB #'N', days+20

MOVB #'0', days+21

MOVB #'0', days+22

MOVB #'0', days+23

RTS

Potentiometer.asm:

XDEF potential

XREF read\_pot,pot\_value

XREF des\_temp\_ones\_temp,des\_temp\_tens\_temp,des\_temp,currentTemp

XREF alarmFlag

XREF Ton

potential: ; convert desired temp and calculates fan rate

LDAA alarmFlag

CMPA #$FF

BEQ done

LDD des\_temp\_tens\_temp

PSHB

LDD des\_temp\_ones\_temp

PSHB

JSR Convert\_normalBin

PULB

PULB

LDAA -5,sp

STAA des\_temp

LDAA currentTemp

CMPA des\_temp

BHI currGdes

LDAA des\_temp

SUBA currentTemp

STAA Ton

BRA done

currGdes:

LDAA currentTemp

SUBA des\_temp

STAA Ton

done:

RTS

Convert\_normalBin:

LDAA 3,sp

;SUBA #$30

LDAB #10

MUL

LDAA 2,sp

;SUBA #$30

ABA

STAA -1,sp

RTS

settingsControl.asm:

XREF SendsChr,PlayTone,display\_string,init\_LCD,disp,read\_pot,pot\_value,Convert\_BIN

; Timing variables

XREF hours,minutes,seconds,day,days,seconds\_tens,seconds\_ones,minutes\_tens,minutes\_ones,hours\_tens,hours\_ones

XREF menuOption,keyPress,set\_settings,Convert\_BCD,curr\_temp\_tens,curr\_temp\_ones,currentTemp,tempChange

XDEF settingsControl

exitPlus: RTS

settingsControl:

LDAB keyPress

LDAA set\_settings

BEQ HourTens

CMPA #1

BEQ HourOnes

CMPA #2

BEQ MinutesTens

CMPA #3

BEQ MinutesOnes

CMPA #4

BEQ SecondsTens

CMPA #5

BEQ SecondsOnesBuffer

CMPA #6

BEQ dayChangeBuffer

CMPA #7

BEQ temp\_tenBuffer

CMPA #8

BEQ temp\_oneBuffer

RTS

dayChangeBuffer: LBRA dayChange

temp\_tenBuffer: LBRA temp\_ten

temp\_oneBuffer: LBRA temp\_one

SecondsOnesBuffer: LBRA SecondsOnes

HourTens: JSR init\_disp

LDAB keyPress

CMPB #3

BHS exitPlus

ADDB #$30

STAB hours\_tens

;MOVB keyPress,hours\_tens

INC set\_settings

BRA exitPlus

HourOnes:

LDAA hours\_tens

CMPA #2

BEQ HourException

CMPB #9

BHS exitPlus

ADDB #$30

STAB hours\_ones

;MOVB keyPress,hours\_ones

INC set\_settings

BRA exitPlus

HourException:

CMPB #5

BHS exit

ADDB #$30

STAB hours\_ones

;MOVB keyPress,hours\_ones

INC set\_settings

BRA exit

MinutesTens:

CMPB #6

BHS exit

ADDB #$30

STAB minutes\_tens

;MOVB keyPress,minutes\_tens

INC set\_settings

BRA exit

MinutesOnes:

CMPB #10

BHS exit

ADDB #$30

STAB minutes\_ones

;MOVB keyPress,minutes\_ones

INC set\_settings

BRA exit

SecondsTens:

CMPB #6

BHS exit

ADDB #$30

STAB seconds\_tens

;MOVB keyPress,seconds\_tens

INC set\_settings

BRA exit

SecondsOnes:

CMPB #10

BHS exit

ADDB #$30

STAB seconds\_ones

;MOVB keyPress,seconds\_ones

INC set\_settings

BRA exit

dayChange:

CMPB #8

BHS exit

LDAA #3

LDAB keyPress

MUL

LDX #days

ABX

STX day

INC set\_settings

BRA exit

temp\_ten:

CMPB #10

BHS exit

ADDB #$30

STAB curr\_temp\_tens

;MOVB keyPress,curr\_temp\_tens

INC set\_settings

BRA exit

temp\_one:

CMPB #10

BHS exit

ADDB #$30

STAB curr\_temp\_ones

;MOVB keyPress,curr\_temp\_tens

BRA convertResume

exit: RTS

convertResume:

LDAA #1

STAA tempChange

LDAA hours\_tens

PSHA

LDAB hours\_ones

PSHB

JSR Convert\_BIN

PULB

PULA

LDAA -5,sp

STAA hours

LDAA minutes\_tens

PSHA

LDAB minutes\_ones

PSHB

JSR Convert\_BIN

PULB

PULA

LDAA -5,sp

STAA minutes

LDAA seconds\_tens

PSHA

LDAB seconds\_ones

PSHB

JSR Convert\_BIN

PULB

PULA

LDAA -5,sp

STAA seconds

LDAA curr\_temp\_tens

PSHA

LDAB curr\_temp\_ones

PSHB

JSR Convert\_BIN

PULB

PULA

LDAA -5,sp

STAA currentTemp

CLR set\_settings

MOVB #1,menuOption

RTS

init\_disp:

MOVB #'0',hours\_tens

MOVB #'0',hours\_ones

MOVB #'0',minutes\_tens

MOVB #'0',minutes\_ones

MOVB #'0',seconds\_tens

MOVB #'0',seconds\_ones

LDX #days

LDAB #21

ABX

STX day

MOVB #'0',curr\_temp\_tens

MOVB #'0',curr\_temp\_ones

RTS

stepperMotor.asm:

XDEF stepMotor,opening,closing

XREF PRT\_P,vault\_state,seq\_ind,motorDelay,delayFlag,ani\_length

XREF alarmFlag

MY\_Constants: SECTION

sequence: dc.b $A,$12,$14,$C

Csequence: dc.b $C,$14,$12,$A

stepMotor:

LDAA alarmFlag

CMPA #$FF

BEQ exit

LDAA vault\_state

CMPA #$FF ; if vault is not in FF state, continue

BEQ opening

CMPA #$CC ; if vault is closing

BEQ closing

BRA exit

opening:

BRSET ani\_length,#$FF,exit ; if ani\_length = FF then motor is disabled

BRCLR delayFlag,#$FF,exit ; if delay not reached exit

LDX #sequence

LDAB seq\_ind

ABX

LDAA 0,x

STAA PRT\_P

INC seq\_ind

INC ani\_length ; goes up to $20=32 steps

;LDAA ani\_length

;INCA

;CMPA #100

;BEQ exit

;BNE dont\_resetVault

CLR delayFlag

LDAB seq\_ind

CMPB #4

BLO exit

CLR seq\_ind

RTS ; clr sequence indicator if eos reached

closing:

BRSET ani\_length,#$FF,exit ; if ani\_length = FF then motor is disabled

BRCLR delayFlag,#$FF,exit ; if delay not reached exit

LDX #Csequence

LDAB seq\_ind

ABX

LDAA 0,x

STAA PRT\_P

INC seq\_ind

LDAA ani\_length

INCA

STAA ani\_length

CMPA #100

BNE dont\_resetVault

LDAA #0

STAA vault\_state

;INC ani\_length ; goes up to $20=32 steps

dont\_resetVault:

CLR delayFlag

LDAB seq\_ind

CMPB #4

BLO exit

CLR seq\_ind

RTS

exit: RTS

transactionalLogic.asm:

XREF menuOption,money,money\_bcd,tmoney,tmoney\_bcd,bankCounter,keyPress,vault\_state,ani\_length

XDEF dep,with

XREF Convert\_BIN,d\_w,Convert\_BCD

dep:

LDAA #6

STAA menuOption

INC bankCounter

LDAA bankCounter

CMPA #1

BEQ exit ; on 1 put new screen

CMPA #2

BEQ first

CMPA #3

BEQ second

CMPA #4

BEQ third

CMPA #5

BEQ fourth

; start closing vault

JSR c\_i

LDAA #$CC ; state CC means initiate close vault

STAA vault\_state

CLR ani\_length

LDAA #1

STAA menuOption

BRA exit

with:

LDAA #6

STAA menuOption

INC bankCounter

LDAA bankCounter

CMPA #1

BEQ exit ; on 1 put new screen

CMPA #2

BEQ first

CMPA #3

BEQ second

CMPA #4

BEQ third

CMPA #5

BEQ fourth

; start closing vault

JSR c\_i

LDAA #$CC ; state CC means initiate close vault

STAA vault\_state

CLR ani\_length

LDAA #1

STAA menuOption

BRA exit

exit: RTS

first:

LDAA keyPress

ADDA #$30

STAA tmoney\_bcd

RTS

second:

LDAA keyPress

ADDA #$30

STAA tmoney\_bcd+1

RTS

third:

LDAA keyPress

ADDA #$30

STAA tmoney\_bcd+2

RTS

fourth:

LDAA keyPress

ADDA #$30

STAA tmoney\_bcd+3

RTS

c\_i:

CLR bankCounter

LDAA tmoney\_bcd

PSHA

LDAB tmoney\_bcd+1

PSHB

JSR Convert\_BIN

PULB

PULA

LDAA -5,sp

STAA tmoney

LDAA tmoney\_bcd+2

PSHA

LDAB tmoney\_bcd+3

PSHB

JSR Convert\_BIN

PULB

PULA

LDAA -5,sp

STAA tmoney+1

CMPA #1

BEQ addMoney

LDD money ; subtracting from money

SUBD tmoney

STD money

BRA nextIni

addMoney:

LDD money

ADDD tmoney

STD money

nextIni:

LDD money

PSHA

JSR Convert\_BCD ; 0 1 2 3

PULA

LDAA -5,sp ; 2-hundreds

STAA money\_bcd+1

LDAA -4,sp

STAA money\_bcd ; 3-thousands

LDD money

PSHB

JSR Convert\_BCD

PULB

LDAA -5,sp

STAA money\_bcd+3

LDAA -4,sp

STAA money\_bcd+2

CLR d\_w

MOVB #'\_',tmoney\_bcd

MOVB #'\_',tmoney\_bcd+1

MOVB #'\_',tmoney\_bcd+2

MOVB #'\_',tmoney\_bcd+3

RTS

References

Pack, Daniel J., and Steven F. Barrett. *Microcontroller Theory and Applications: HC12 and S12*. Pearson Prentice Hall, 2008.