**Microwave (Individual)**

**Emilio Lopez**

**Due: 5/7/2014**

Table of Contents

**Introduction** ……..……………………………………………………………………………… 4

**Design**

Figures (Pictures) ………………………………………………………………………… 3

Peripherals ……………………………………………………………………………….. 5

General Software Implementation ……………………………………………………….. 5

Flow Chart ……………………………………………………………………………….. 6

**Conclusion** ……………………………………………………………………………………… 7

**Appendices**

User Manuel ……………………………………………………………………………... 8

*Code*

main.asm ....……………………………………………………………………… 9

CHECK\_STATUS.asm ………………………………………………………... 13

IRQ\_SER.asm ………………………………………………………………….. 16

MAIN\_MENU.asm …………………………………………………………….. 19

SET\_WATT.asm ………………………………………………………………. 20

SET\_TIME.asm ………………………………………………………………... 23

SET\_MOTOR\_SPEED.asm …………………………………………………… 25

COOK\_MENU.asm ………………………………………………………….… 30

SPIN\_STEPPER\_MOTOR.asm……………………….……………………….. 32

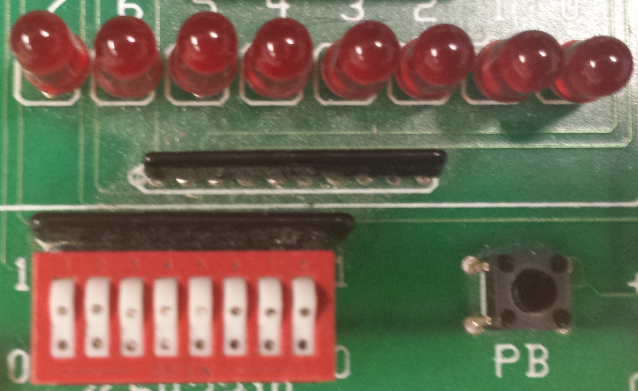
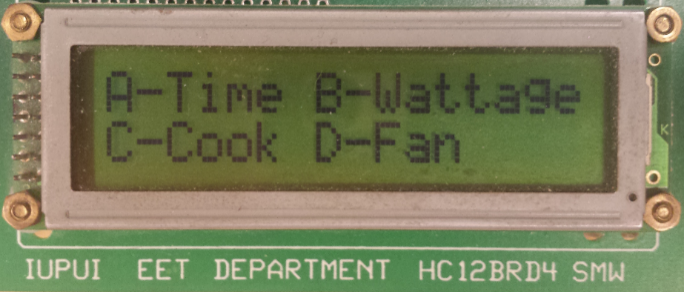
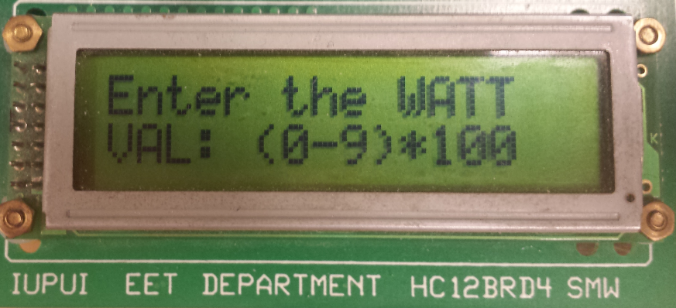
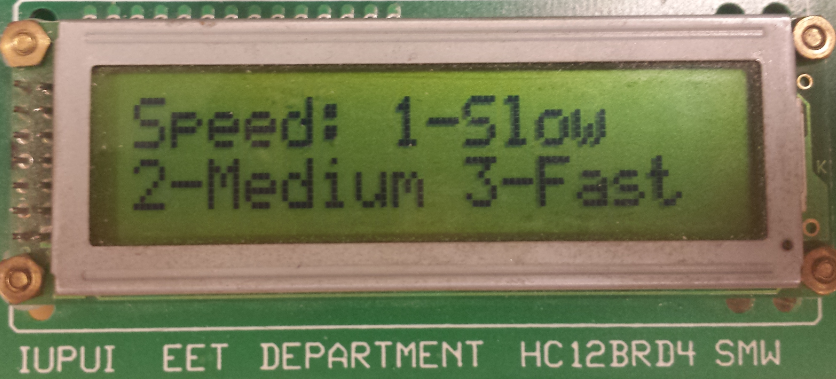
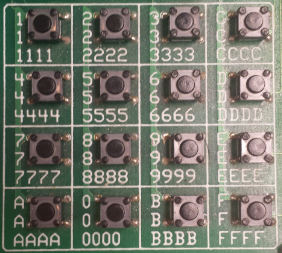
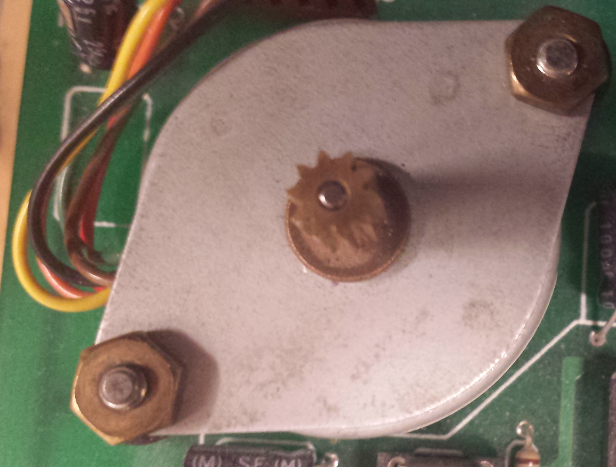
HEX\_READ.asm ………………………………………………………………. 34

DISP\_TIME.asm ……………………………………………………………….. 36

COOL\_DOWN\_DISPLAY.asm………………………………………………... 39

**Table Figures**

Hex Key Pad



Potentiometer

Fan Speed Menu

Wattage Menu

Time Menu

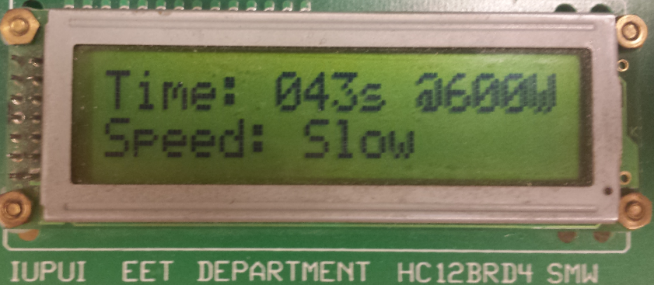
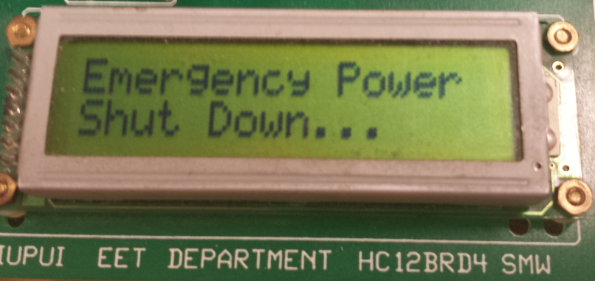
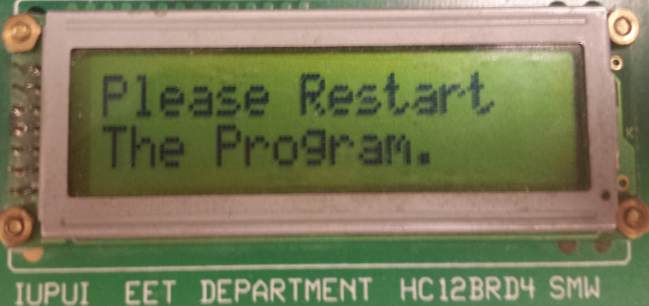
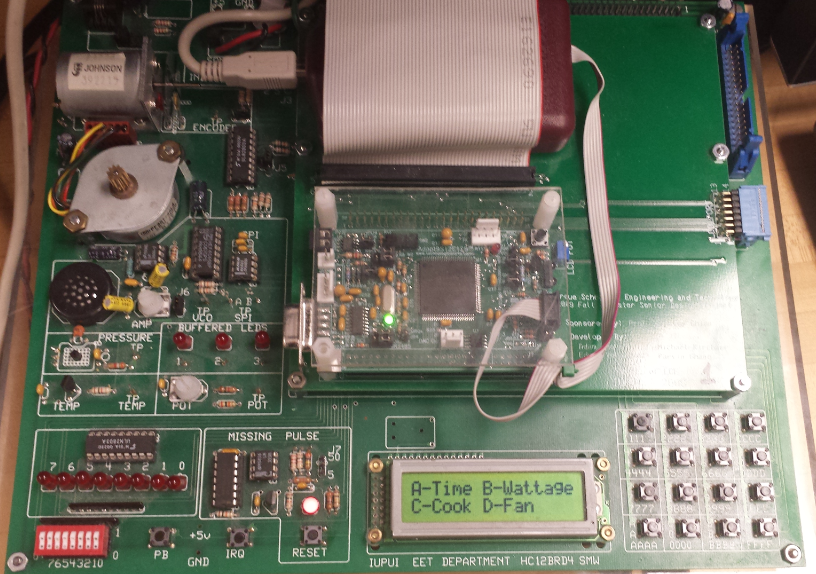
Main Menu

LEDs and Switches

DC Motor (Fan)

IRQ   
(Emergency button)

Stepper Motor



Entire Board

End of Program

Power Shut Down

(After IRQ Pressed)

Specs (Run Time)

Cook Menu

**Introduction**

The purpose of this project is to write an assembly program to simulate a microwave. The MC9S12E128 microcontroller was used to as the simulator. Throughout the course of the entire project, all concepts in both the laboratory and in lecture were to be utilized as tools and to be expanded upon to implement such a complex simulation. The I/O ports are utilized to give a visual simulation of how the microwave behaving at certain instances of time.

**Design**

Peripherals

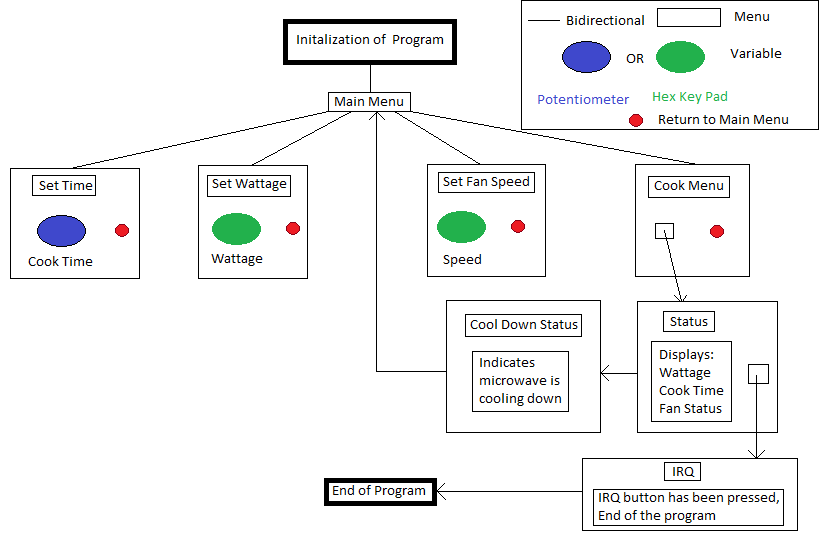
A list of all the peripherals that are used in the project are as follows:

* *Stepper-Motor*: Used to rotate the microwave plate when it is running. The plate runs clockwise during cook time, and counter clockwise during cooling time.
* *DC-Motor:* Used as the microwave’s fan (Fast/Medium/Slow/Off speed options)
* *Keypad:* Used to navigate through the menus, set wattage, fan speed, and start the microwave.
* *Potentiometer:* Used to set the cook time of the microwave
* *LCD:* Used to display the different menus, during runtime shows spin time, wattage, and fan status, and lastly to display the remaining cook time.
* *Switches:* One switch is the status of the microwave door (open/closed), another switch is for the fan (0 off, 1 on)
* *LEDs:* LED 7 displays that the microwave is cooking, LED 0 shows that the microwave is cooling, all LEDs display rapidly when the emergency power down button has been pressed (IRQ)
* *IRQ:* Turns the microwave off immediately and terminates the assembly program.
* *RTI:* Used to control time and check various different statuses of multiple ports.

General Software Implementation

The software implementation of this microwave simulation implements several methods and functions that were constructed in previous labs. The microcontroller first starts off by displaying on the LCD screen the main menu with four different options to select from using the keypad. This main menu was implemented by using the given code files from previous labs (\*.c files). In the main menu, the first loop continuously jumped to the HEX\_READ routine which checked to see whether a button has been press. The options to go into the different menus are A, B, C, and D. When a button has been selected to go to a different menu a different subroutine that corresponds to that selection updates the LCD and different variables (speed, time, and wattage) can be selected. The speed of the fan from the speed menu has three different options; slow medium and fast speed. The speed and wattage is selected using the hex keypad. However, the cook time is selected by using the potentiometer. For the cook menu an RTI is used to check whether switch 7 is on (indicating that the microwave door is closed) and also the RTI checks to see if the button C has been selected from the hex key pad. If switch 7 is turned off (microwave is open) the microwave stops immediately and the cook time stays at the same time it was left at. If switch 7 is closed (microwave door is closed), then the microwave begins to cook at the time where it was last left at. During the time the microwave is in cook mode, four different subroutines are called to perform various tasks. These tasks include; spinning the stepper motor (using a stepper motor routine), spinning the fan at the speed in which the user selected (automatically fast if no speed has been selected), calling a subroutine to refresh the LCD, and lastly to utilize the RTI to count elapsed time in seconds and decrement the cook time that is being displayed to the LCD. During the time the microwave is in cook mode, if the IRQ button has been pushed, an interrupted is called and displays on the LCD “Emergency Power Shut Down…” while the LEDs rapidly flash. After about three seconds, the LCD updates and displays “Please Restart The Program.”. After this display, it terminates the program and the entire program must be restarted to make the microwave function once again. A flow chart below give a visual description of the entire program

Simulation Flow Chart



There were a few changes that I did make to the entire design of the microwave project. To begin with, I have meet all of the requirements however I was unable to implement the time of day into the time menu and was the only part of the project that was not working. I added the IRQ functionality as it was not part of the individual project requirements and made the LEDs flash rapidly when displaying the power shutdown display. Additionally I added the functionality of the opening and closing the door to the microwave which keeps the cook time when opened, and resumes at that time when closed (all done with switch 7 of port T).

Conclusion

In conclusion I believe that this microwave project was overall a successful accomplishment. Though implementation of all parts and integrating different functionalities together was difficult at time. It is imperative to ensure that all of the code that is written has meaningful comment block to help the creator understand what is exactly going on in a subroutine when referred back to at a later time. Utilizing the debugger is an excellent to in identifying what bug/issue may be going on with the program during run time. Additionally it is important to run the program as frequently as possible to ensure when writing a function/subroutine in the program, that it works properly and the program still runs. Lastly I strongly believe that when working on such massive projects, it is crucial to frequently save the program to either an external drive or on the cloud. The reason being is that, if for whatever reason the program crashes or has a very complex to near unidentifiable bug, that the creator can go back to a previous version of the project that is functioning.

**Appendices**

User Manuel

To begin with, please refer to the simulation flow chart above to get a good understanding of how the microwave has been implemented. To turn the microwave on, in the debugger window select the run button to initialize the program. The main menu will appear on the LCD with a list of different menus that can be navigated to. To select these different menus enter either A, B, C, or D from the hex keypad. When setting the wattage and fan speed, please use the hex keypad to enter the value. For selecting a time in the time menu, use the potentiometer to increase or decrease the cook time. All submenus return to the main menu either when a value has been entered or the respective button to go back to the main menu has been selected. To open or close the door turn switch 7 on port T (the furthest switch to the left up (door closed) or down (door open)). Always ensure that fourth switch from the left is always switched upwards to enable the functionality of the DC Motor (the microwave fan). In the cook menu, ensure that switch 7 is switched upwards, and press C to cook. The specifications of the parameters will display on the screen. To turn the fan on during runtime flip the right most switch (switch 0) upwards to turn the fan on. Once the cook time runs to zero, the LCD returns to the main menu. To immediately shut down the microwave for safety issues, press the IRQ button to terminate the program and turn the microwave off. Once this has been pressed the microwave program needs to be restarted.

Code

**Main.asm**

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* This stationery serves as the framework for a \*

;\* user application. For a more comprehensive program that \*

;\* demonstrates the more advanced functionality of this \*

;\* processor, please see the demonstration applications \*

;\* located in the examples subdirectory of the \*

;\* Freescale CodeWarrior for the HC12 Program directory \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Include derivative-specific definitions

INCLUDE 'derivative.inc'

; export symbols

XDEF Entry, \_Startup, main, port\_S, ROW\_VAL, COORD\_TBL, VALUE\_TBL, HEX\_VALUE , port\_T,port\_T\_DDR, ON\_OFF, port\_U,port\_U\_DDR, port\_U\_PSR,PERUu

XDEF COOL\_Y\_N,disp,WATTAGE, START\_COOKING\_Y\_N, RUN\_TIME, SPIN\_SPEED\_OPTION, SLOW\_SPEED\_COUNT, MED\_SPEED\_COUNT, TIME\_COUNTER

; 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, CHECK\_STATUS, HEX\_READ, SPIN\_STEPPER\_MOTOR,SET\_WATT,SET\_TIME,MAIN\_MENU,DISP\_TIME, read\_pot,pot\_value ; symbol defined by the linker for the end of the stack

XREF COOK\_MENU, COOL\_DOWN\_DISPLAY, SET\_MOTOR\_SPEED

; variable/data section

MY\_EXTENDED\_RAM: SECTION

; Insert here your data definition.

WATTAGE: ds.b 1

HEX\_VALUE: ds.b 1

TIME\_SECOND: ds.b 1;runtime of the microwave

ON\_OFF: ds.b 1;power status of the microwave

COOL\_Y\_N: ds.b 1;checks to see if the microwave timer has ran out

LCD\_SECTION: ds.b 1

START\_COOKING\_Y\_N: ds.b 1;checks to see if c has been press from key pad in COOK\_MENU

RUN\_TIME: ds.w 1 ;runtime extracted from the pot

SPIN\_SPEED\_OPTION: ds.b 1;spin speed of the dc motor

SLOW\_SPEED\_COUNT: ds.b 1

MED\_SPEED\_COUNT: ds.b 1

TIME\_COUNTER: ds.b 1

disp: ds.b 33

My\_Constants: SECTION

;====================================================================================================================

port\_T: equ $240

port\_T\_DDR: equ $242

;====================================================================================================================

;LED RELATED

port\_S: equ $248

port\_S\_DDR: equ $24A

;====================================================================================================================

;HEX RELATED

port\_U: equ $268;Port U bits

port\_U\_DDR: equ $26A;Direction Register

port\_U\_PSR: equ $26D;Polarity Select Register

PERUu: equ $26C;Pull up Registers

counter\_1: equ 65000

counter\_2: equ 1000

ROW\_VAL: dc.b $70,$b0,$d0,$e0,$ff; is at begining to compensate for iny at beginning of ROW\_LOOP

COORD\_TBL: dc.b $77,$7b,$7d,$7e,$b7,$bb,$bd,$be,$d7,$db,$dd,$de,$e7,$eb,$ed,$ee,$ff;the ternimator

VALUE\_TBL: dc.b $1,$2,$3,$c,$4,$5,$6,$d,$7,$8,$9,$e,$a,$0,$b,$f

;==========================================================

;RTI RELATED

MYRTIFLG: equ $0037 ; RTI flag register address CRGFLG

MYRTIENA: equ $0038 ; RTI init register address CRGINT

MYRTICTL: equ $003B ; RTI control register address RTICTL

; code section

MyCode: SECTION

main:

\_Startup:

Entry:

lds #\_\_SEG\_END\_SSTACK ;initalize stack

movb #%1000, port\_T\_DDR;for the dc motor to spin

movb #0,COOL\_Y\_N;0 means it is not in a cool down cycle

movb #0,ON\_OFF;initalizes the on off status

movb #0,TIME\_SECOND;value extracted from the potentiometer

;movb #0,TIME\_MINUTE;value extracted from the potentiometer

movb #$ff,port\_S\_DDR

movb #0,HEX\_VALUE

movb #0,TIME\_COUNTER;default timer counter for rti is 0

movb #$17, MYRTICTL ;1MS DELAY

movb #$80, MYRTIENA ;Enable RTI

ldaa #0

staa START\_COOKING\_Y\_N;initailization

staa RUN\_TIME ;initailization

;need to initialize vector table in linker parameter file

cli ;enable interrupt

jsr MAIN\_MENU;init main menu on LCD

;====================================================================================================================

AGAIN:

ldaa ON\_OFF

cmpa #1;if the micro wave is on

beq INIT;start the microwave if it is on

jsr HEX\_READ

cmpa #$a;chose the menu to set the time

beq TIME\_SET

cmpa #$b;chose the menu to set the wattage

beq WATT\_SET

cmpa #$c;chose the menu to set the wattage

beq COOK\_SET

cmpa #$d;chose the menu to set the wattage

beq MOTOR\_SET

bra AGAIN

;====================================================================================================================

TIME\_SET:

jsr SET\_TIME

bra AGAIN

;====================================================================================================================

WATT\_SET: jsr SET\_WATT;go set the wattage routine

bra AGAIN

;====================================================================================================================

COOK\_SET:

jsr COOK\_MENU

bra AGAIN

;====================================================================================================================

MOTOR\_SET:

jsr SET\_MOTOR\_SPEED

bra AGAIN

;====================================================================================================================

INIT:

ldaa #3

jsr DISP\_TIME;DISPLAY STATUS

staa SLOW\_SPEED\_COUNT

ldaa #1

staa MED\_SPEED\_COUNT

movb #%10000000,port\_S;bit 7 is turned on to indicate on time

jsr SPIN\_STEPPER\_MOTOR;spin the motor

;jsr DISP\_LCD;displaying runtime

ldaa ON\_OFF

beq COOL\_DOWN;means it is off;

jsr DISP\_TIME;displays time left on LCD

bra AGAIN

COOL\_DOWN: movb #1,COOL\_Y\_N

jsr COOL\_DOWN\_DISPLAY;display to the lcd that the microwave is cooling down

movb #%00000001,port\_S;cool down phase

jsr SPIN\_STEPPER\_MOTOR;now in the cool down phase

movb #0,port\_S

jsr MAIN\_MENU;reset main menu to lcd

bra AGAIN

end

**CHECK\_STATUS.asm**

xdef CHECK\_STATUS

xref SECOND,port\_S,ON\_OFF, port\_T, port\_T\_DDR,START\_COOKING\_Y\_N,SPIN\_SPEED\_OPTION, SLOW\_SPEED\_COUNT, MED\_SPEED\_COUNT, DISP\_TIME, TIME\_COUNTER

xref RUN\_TIME

My\_Constants: SECTION

RTIE: equ $80

CHECK\_STATUS:

brset port\_T,%10000000,MICRO\_ON;check if bit 0 is on, if so the fan is on

;===================================================================================================================

MICRO\_OFF: movb #0,ON\_OFF;the microwave is off

bclr port\_T,%1000;turn the DC MOTOR OFF

ldaa #RTIE ;re-enable interrupt

staa $37

rti

MICRO\_ON:

ldaa START\_COOKING\_Y\_N;to check if C has been pressed in the cook menu

cmpa #1

bne EXIT

;TIME CHECKER

ldx TIME\_COUNTER

inx ;increase ms counter in memory

stx TIME\_COUNTER

cpx #860 ;equal to 1 second?

bne SWITCH\_ON\_OFF\_CHECK;means it has not been 1s yet

ldx #0 ;reset to 0 if = 1 sec

stx TIME\_COUNTER

jsr DISP\_TIME

ldd RUN\_TIME

cmpb #0

beq EXIT

decb

std RUN\_TIME

SWITCH\_ON\_OFF\_CHECK:

movb #1,ON\_OFF;the microwave is on

ldaa #RTIE ;re-enable interrupt

staa $37

brset port\_T,%00000001,SPIN\_FAN;check if bit 0 is on, if so the fan is on

bclr port\_T,%1000;turn the DC MOTOR OFF

rti

SPIN\_FAN:

ldaa SPIN\_SPEED\_OPTION

cmpa #1;slow spin speed

beq SLOW\_FAN\_SPEED

cmpa #2;medium spin speed

beq MED\_FAN\_SPEED

bset port\_T,%1000;Otherwise it is a fast spin speed

rti

;========================================================================================================

SLOW\_FAN\_SPEED:

ldaa SLOW\_SPEED\_COUNT

beq SLOW\_FAN\_SPEED\_INIT

deca

staa SLOW\_SPEED\_COUNT

bclr port\_T,%1000;turn the DC MOTOR OFF

rti

SLOW\_FAN\_SPEED\_INIT:

ldaa #3

staa SLOW\_SPEED\_COUNT

bset port\_T,%1000;turn the DC MOTOR OFF

rti

;========================================================================================================

MED\_FAN\_SPEED:

ldaa MED\_SPEED\_COUNT

beq MED\_FAN\_SPEED\_INIT

deca

staa MED\_SPEED\_COUNT

bclr port\_T,%1000;turn the DC MOTOR OFF

rti

MED\_FAN\_SPEED\_INIT:

ldaa #2

staa MED\_SPEED\_COUNT

bset port\_T,%1000;turn the DC MOTOR OFF

rti

EXIT:

movb #0,ON\_OFF;the microwave is off

movb #0,START\_COOKING\_Y\_N;resets the c button that has been clicked

bclr port\_T,%1000;turn the DC MOTOR OFF

ldaa #RTIE ;re-enable interrupt

staa $37

rti

**IRQ\_SER.asm**

XDEF IRQ\_SER

XREF init\_LCD,read\_pot,pot\_value,display\_string, HEX\_VALUE, port\_S, HEX\_READ, disp, port\_T

IRQ\_SER:

movb #'E',disp

movb #'m',disp+1

movb #'e',disp+2

movb #'r',disp+3

movb #'g',disp+4

movb #'e',disp+5

movb #'n',disp+6

movb #'c',disp+7

movb #'y',disp+8

movb #' ',disp+9

movb #'P',disp+10

movb #'o',disp+11

movb #'w',disp+12

movb #'e',disp+13

movb #'r',disp+14

movb #' ',disp+15

movb #'S',disp+16

movb #'h',disp+17

movb #'u',disp+18

movb #'t',disp+19

movb #' ',disp+20

movb #'D',disp+21

movb #'o',disp+22

movb #'w',disp+23

movb #'n',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 ;string terminator, acts like '\0'

jsr init\_LCD

ldd #disp

jsr display\_string

bclr port\_T,%1000;turn the DC MOTOR OFF

ldx #8;loop again

FLASH:

pshx;temp save x

ldy #3

KEEP\_DISPLAYING\_1:;DELAY loop to keep displaying the value of wattage

movb #$0,port\_S

ldd #disp

jsr display\_string

movb #0,port\_S

dey

bne KEEP\_DISPLAYING\_1

ldy #3

KEEP\_DISPLAYING\_2:;DELAY loop to keep displaying the value of wattage

movb #$ff,port\_S

ldd #disp

jsr display\_string

movb #0,port\_S

dey

bne KEEP\_DISPLAYING\_2

pulx

dex

bne FLASH

STOP\_COOLING:

bclr port\_T,%1000;turn the DC MOTOR OFF

movb #'P',disp

movb #'l',disp+1

movb #'e',disp+2

movb #'a',disp+3

movb #'s',disp+4

movb #'e',disp+5

movb #' ',disp+6

movb #'R',disp+7

movb #'e',disp+8

movb #'s',disp+9

movb #'t',disp+10

movb #'a',disp+11

movb #'r',disp+12

movb #'t',disp+13

movb #' ',disp+14

movb #' ',disp+15

movb #'T',disp+16

movb #'h',disp+17

movb #'e',disp+18

movb #' ',disp+19

movb #'P',disp+20

movb #'r',disp+21

movb #'o',disp+22

movb #'g',disp+23

movb #'r',disp+24

movb #'a',disp+25

movb #'m',disp+26

jsr init\_LCD

ldd #disp

jsr display\_string

end

**MAIN\_MENU.asm**

XDEF MAIN\_MENU

XREF init\_LCD,read\_pot,pot\_value,display\_string, HEX\_VALUE, port\_S, HEX\_READ, disp

MAIN\_MENU:

movb #'A',disp

movb #'-',disp+1

movb #'T',disp+2

movb #'i',disp+3

movb #'m',disp+4

movb #'e',disp+5

movb #' ',disp+6

movb #'B',disp+7

movb #'-',disp+8

movb #'W',disp+9

movb #'a',disp+10

movb #'t',disp+11

movb #'t',disp+12

movb #'a',disp+13

movb #'g',disp+14

movb #'e',disp+15

movb #'C',disp+16

movb #'-',disp+17

movb #'C',disp+18

movb #'o',disp+19

movb #'o',disp+20

movb #'k',disp+21

movb #' ',disp+22

movb #'D',disp+23

movb #'-',disp+24

movb #'F',disp+25

movb #'a',disp+26

movb #'n',disp+27

movb #' ',disp+28

movb #' ',disp+29

movb #' ',disp+30

movb #' ',disp+31

movb #0,disp+32 ;string terminator, acts like '\0'

jsr init\_LCD

ldd #disp

jsr display\_string

rts

**SET\_WATT.asm**

**XDEF SET\_WATT**

**XREF init\_LCD,read\_pot,pot\_value,display\_string, HEX\_VALUE, port\_S, HEX\_READ, disp, WATTAGE, MAIN\_MENU**

**SET\_WATT:**

**movb #'E',disp**

**movb #'n',disp+1**

**movb #'t',disp+2**

**movb #'e',disp+3**

**movb #'r',disp+4**

**movb #' ',disp+5**

**movb #'t',disp+6**

**movb #'h',disp+7**

**movb #'e',disp+8**

**movb #' ',disp+9**

**movb #'W',disp+10**

**movb #'A',disp+11**

**movb #'T',disp+12**

**movb #'T',disp+13**

**movb #' ',disp+14**

**movb #' ',disp+15**

**movb #'V',disp+16**

**movb #'A',disp+17**

**movb #'L',disp+18**

**movb #':',disp+19**

**movb #' ',disp+20**

**movb #'(',disp+21**

**movb #'0',disp+22**

**movb #'-',disp+23**

**movb #'9',disp+24**

**movb #')',disp+25**

**movb #'\*',disp+26**

**movb #'1',disp+27**

**movb #'0',disp+28**

**movb #'0',disp+29**

**movb #' ',disp+30**

**movb #' ',disp+31**

**movb #0,disp+32 ;string terminator, acts like '\0'**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*string initialization\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**jsr init\_LCD**

**movb #$ff,HEX\_VALUE;init hex value to ensure something is pressed**

**ldy #disp;starting address**

**Loop: jsr HEX\_READ;go to the hex read subroutine**

**ldd #disp**

**jsr display\_string**

**ldaa HEX\_VALUE**

**staa port\_S**

**cmpa #$ff;checks to see if the hex value is the same as the initailzation**

**;this means that the hex keypad has not been changed**

**beq Loop**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**movb #'Y',disp**

**movb #'o',disp+1**

**movb #'u' ,disp+2**

**movb #' ',disp+3**

**movb #'e',disp+4**

**movb #'n',disp+5**

**movb #'t',disp+6**

**movb #'e',disp+7**

**movb #'r',disp+8**

**movb #'e',disp+9**

**movb #'d',disp+10**

**movb #':',disp+11**

**movb #' ',disp+12**

**movb #'0',disp+13**

**movb #'0',disp+14**

**movb #'W',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 ;string terminator, acts like '\0'**

**staa WATTAGE;stores the watage value into the constant**

**adda #48;convert to ascii**

**staa disp+12**

**ldy #30**

**KEEP\_DISPLAYING:;DELAY loop to keep displaying the value of wattage**

**ldd #disp**

**jsr display\_string**

**dey**

**bne KEEP\_DISPLAYING**

**jsr MAIN\_MENU;resets the LCD to main menu**

**rts**

**SET\_TIME.asm**

XDEF SET\_TIME

XREF init\_LCD,read\_pot,pot\_value,display\_string, HEX\_VALUE, HEX\_READ, disp, MAIN\_MENU, RUN\_TIME

My\_variable: section

hundreds\_place: ds.b 1

tens\_place: ds.b 1

ones\_place: ds.b 1

SET\_TIME:

movb #'T',disp

movb #'i',disp+1

movb #'m',disp+2

movb #'e',disp+3

movb #':',disp+4

movb #' ',disp+5

movb #'0',disp+6

movb #'0',disp+7

movb #'0',disp+8

movb #'s',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 #'E',disp+25

movb #'n',disp+26

movb #'t',disp+27

movb #'e',disp+28

movb #'r',disp+29

movb #'-',disp+30

movb #'E',disp+31

movb #0,disp+32 ;string terminator, acts like '\0'

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*string initialization\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

jsr init\_LCD

movb #$ff,HEX\_VALUE;init hex value to ensure something is pressed

ldy #disp;starting address

ldd #disp

jsr display\_string

KEEP\_READING:

jsr read\_pot

ldd pot\_value

ldx #100;hundres place

idiv

tfr x,a

adda #48

staa disp+6

ldaa #0;reset a to its original value which will always be 0

ldx #10 ;tens place

idiv

tfr x,a

adda #48

staa disp+7

ldaa #0;reset a to its original value which will always be 0

addb #48 ;ones place

stab disp+8

ldd #disp

jsr display\_string

jsr HEX\_READ

ldaa HEX\_VALUE

cmpa #$0e;checks to see if the user wants to exit the main menu

bne KEEP\_READING

ldd pot\_value

std RUN\_TIME

jsr MAIN\_MENU;resets the lcd to main menu

rts

**SET\_MOTOR\_SPEED.asm**

XDEF SET\_MOTOR\_SPEED

XREF init\_LCD,read\_pot,pot\_value,display\_string, HEX\_VALUE, port\_S, HEX\_READ, disp, WATTAGE, MAIN\_MENU, SPIN\_SPEED\_OPTION

SET\_MOTOR\_SPEED:

movb #'S',disp

movb #'p',disp+1

movb #'e',disp+2

movb #'e',disp+3

movb #'d',disp+4

movb #':',disp+5

movb #' ',disp+6

movb #'1',disp+7

movb #'-',disp+8

movb #'S',disp+9

movb #'l',disp+10

movb #'o',disp+11

movb #'w',disp+12

movb #' ',disp+13

movb #' ',disp+14

movb #' ',disp+15

movb #'2',disp+16

movb #'-',disp+17

movb #'M',disp+18

movb #'e',disp+19

movb #'d',disp+20

movb #'i',disp+21

movb #'u',disp+22

movb #'m',disp+23

movb #' ',disp+24

movb #'3',disp+25

movb #'-',disp+26

movb #'F',disp+27

movb #'a',disp+28

movb #'s',disp+29

movb #'t',disp+30

movb #' ',disp+31

movb #0,disp+32 ;string terminator, acts like '\0'

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*string initialization\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;read hex key pad

;================================================================================================================

jsr init\_LCD

movb #$ff,HEX\_VALUE;init hex value to ensure something is pressed

ldy #disp;starting address

ldy #1

Loop: jsr HEX\_READ;go to the hex read subroutine

ldd #disp

jsr display\_string

ldaa HEX\_VALUE

cmpa #$01;

beq SLOW\_SPEED

cmpa #$02

beq MED\_JMP;cannot jump to the MED\_SPEED bc out of range

cmpa #$03

beq FAST\_JMP;cannot jump to the FAST\_SPEED bc out of range

bra Loop;nothing has been selected

;================================================================================================================

MED\_JMP:

jmp MED\_SPEED

FAST\_JMP:

jmp FAST\_SPEED

;Slow speed

;================================================================================================================

SLOW\_SPEED:

movb #'S',disp

movb #'L',disp+1

movb #'O' ,disp+2

movb #'W',disp+3

movb #' ',disp+4

movb #'s',disp+5

movb #'p',disp+6

movb #'e',disp+7

movb #'e',disp+8

movb #'d',disp+9

movb #' ',disp+10

movb #'h',disp+11

movb #'a',disp+12

movb #'s',disp+13

movb #' ',disp+14

movb #' ',disp+15

movb #'b',disp+16

movb #'e',disp+17

movb #'e',disp+18

movb #'n',disp+19

movb #' ',disp+20

movb #'s',disp+21

movb #'e',disp+22

movb #'l',disp+23

movb #'e',disp+24

movb #'c',disp+25

movb #'t',disp+26

movb #'e',disp+27

movb #'d',disp+28

movb #'.',disp+29

movb #' ',disp+30

movb #' ',disp+31

movb #0,disp+32 ;string terminator, acts like '\0'

ldaa #1; CHANGE!!!!!!!!!!!!!!!

staa SPIN\_SPEED\_OPTION

adda #48;convert to ascii

;staa disp+12

ldy #45

KEEP\_DISPLAYING\_SLOW:;DELAY loop to keep displaying the value of wattage

ldd #disp

jsr display\_string

dey

bne KEEP\_DISPLAYING\_SLOW

jsr MAIN\_MENU;resets the LCD to main menu

rts

;================================================================================================================

;Medium Speed

;================================================================================================================

MED\_SPEED:

movb #'M',disp

movb #'E',disp+1

movb #'D' ,disp+2

movb #'.',disp+3

movb #' ',disp+4

movb #'s',disp+5

movb #'p',disp+6

movb #'e',disp+7

movb #'e',disp+8

movb #'d',disp+9

movb #' ',disp+10

movb #'h',disp+11

movb #'a',disp+12

movb #'s',disp+13

movb #' ',disp+14

movb #' ',disp+15

movb #'b',disp+16

movb #'e',disp+17

movb #'e',disp+18

movb #'n',disp+19

movb #' ',disp+20

movb #'s',disp+21

movb #'e',disp+22

movb #'l',disp+23

movb #'e',disp+24

movb #'c',disp+25

movb #'t',disp+26

movb #'e',disp+27

movb #'d',disp+28

movb #'.',disp+29

movb #' ',disp+30

movb #' ',disp+31

movb #0,disp+32 ;string terminator, acts like '\0'

ldaa #2; CHANGE!!!!!!!!!!!!!!!

staa SPIN\_SPEED\_OPTION

adda #48;convert to ascii

;staa disp+12

ldy #45

KEEP\_DISPLAYING\_MED:;DELAY loop to keep displaying the value of wattage

ldd #disp

jsr display\_string

dey

bne KEEP\_DISPLAYING\_MED

jsr MAIN\_MENU;resets the LCD to main menu

rts

;================================================================================================================

;Fast Speed

;================================================================================================================

FAST\_SPEED:

movb #'F',disp

movb #'A',disp+1

movb #'S',disp+2

movb #'T',disp+3

movb #' ',disp+4

movb #'s',disp+5

movb #'p',disp+6

movb #'e',disp+7

movb #'e',disp+8

movb #'d',disp+9

movb #' ',disp+10

movb #'h',disp+11

movb #'a',disp+12

movb #'s',disp+13

movb #' ',disp+14

movb #' ',disp+15

movb #'b',disp+16

movb #'e',disp+17

movb #'e',disp+18

movb #'n',disp+19

movb #' ',disp+20

movb #'s',disp+21

movb #'e',disp+22

movb #'l',disp+23

movb #'e',disp+24

movb #'c',disp+25

movb #'t',disp+26

movb #'e',disp+27

movb #'d',disp+28

movb #'.',disp+29

movb #' ',disp+30

movb #' ',disp+31

movb #0,disp+32 ;string terminator, acts like '\0'

ldaa #3; CHANGE!!!!!!!!!!!!!!!

staa SPIN\_SPEED\_OPTION

adda #48;convert to ascii

; staa disp+12

ldy #45

KEEP\_DISPLAYING\_FAST:;DELAY loop to keep displaying the value of wattage

ldd #disp

jsr display\_string

dey

bne KEEP\_DISPLAYING\_FAST

jsr MAIN\_MENU;resets the LCD to main menu

rts

;================================================================================================================

**COOK\_MENU.asm**

XDEF COOK\_MENU

XREF init\_LCD,read\_pot,pot\_value,display\_string, HEX\_VALUE, port\_S, HEX\_READ, disp, WATTAGE, MAIN\_MENU, RUN\_TIME, START\_COOKING\_Y\_N

COOK\_MENU:

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

movb #'T',disp

movb #'i',disp+1

movb #'m' ,disp+2

movb #'e',disp+3

movb #':',disp+4

movb #'0',disp+5

movb #'0',disp+6

movb #'0',disp+7

movb #'s',disp+8

movb #' ',disp+9

movb #' ',disp+10

movb #' ',disp+11

movb #' ',disp+12

movb #' ',disp+13

movb #' ',disp+14

movb #' ',disp+15

movb #'C',disp+16

movb #'o',disp+17

movb #'o',disp+18

movb #'k',disp+19

movb #'-',disp+20

movb #'C',disp+21

movb #' ',disp+22

movb #'E',disp+23

movb #'x',disp+24

movb #'i',disp+25

movb #'t',disp+26

movb #'-',disp+27

movb #'F',disp+28

movb #' ',disp+29

movb #' ',disp+30

movb #' ',disp+31

movb #0,disp+32 ;string terminator, acts like '\0'

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

jsr init\_LCD

movb #$ff,HEX\_VALUE;init hex value to ensure something is pressed

KEEP\_DISPLAYING:;DELAY loop to keep displaying the value of wattage

;placing RUN\_TIME value to array

;----------------------------------------------------------------------------------------

ldd RUN\_TIME

ldx #100;hundres place

idiv

tfr x,a

adda #48

staa disp+5

ldaa #0;reset a to its original value which will always be 0

ldx #10 ;tens place

idiv

tfr x,a

adda #48

staa disp+6

ldaa #0;reset a to its original value which will always be 0

addb #48 ;ones place

stab disp+7

;display array to LCD

;-----------------------------------------------------------------------------------------

ldd #disp

jsr display\_string

;-----------------------------------------------------------------------------------------

;read from the hex keypad

jsr HEX\_READ

ldaa HEX\_VALUE

cmpa #$0f

beq EXIT

cmpa #$0c

beq COOK

;-----------------------------------------------------------------------------------------

bra KEEP\_DISPLAYING

EXIT: jsr MAIN\_MENU;resets the LCD to main menu

rts

COOK:

ldaa #1

staa START\_COOKING\_Y\_N

rts

**SPIN\_STEPPER\_MOTOR.asm**

**XDEF SPIN\_STEPPER\_MOTOR**

**XREF COOL\_Y\_N, ON\_OFF**

**My\_Const: SECTION**

**array: dc.b $ff,$0a,$12,$14,$0c,$ff ;sequence will be sent to the motor**

**port\_P\_DDR: equ $25a**

**port\_P: equ $258**

**port\_T: equ $240**

**; code section**

**MyCode: SECTION**

**SPIN\_STEPPER\_MOTOR:**

**ldaa port\_P\_DDR ;loads port p ddr into acc a**

**oraa #%00011110 ;sets the bits 1-4 in port p ddr**

**staa port\_P\_DDR ;places new value into port p ddr**

**RETRY:**

**ldy #65535**

**SAFE\_DELAY:;safety delay for no sudden changes in rotating**

**dey**

**cpy #0**

**bne SAFE\_DELAY**

**ldaa COOL\_Y\_N**

**beq CLOCK\_W;not in a cool down phase**

**ldaa #7;this is for the cool down phase, it shall cool down for 10 cycles**

**psha**

**bra COUNTER\_CW;it is a cool down phase**

**;====================================================================================================================**

**CLOCK\_W:**

**ldx #array ;loads first element of array into reg. x**

**AGAIN\_CW: ldab #1 ;acc. b will act as a counter**

**LOOP\_CW:**

**ldaa b,x**

**cmpa #$ff ;compares with the terminator $ff**

**beq AGAIN\_CW**

**ldy #25000**

**DELAY\_CW\_1:dey**

**cpy #1**

**bne DELAY\_CW\_1**

**ldy #25000**

**staa port\_P**

**incb ;increments b by one**

**ldaa port\_T**

**ldaa ON\_OFF**

**bne BRANCH\_CW;the microwave has not been turned off**

**rts**

**BRANCH\_CW:bra LOOP\_CW**

**;====================================================================================================================**

**COUNTER\_CW:**

**pula;getting the cool down counter phase**

**deca**

**beq STOP\_COOLING;stop cooling and return to the main program**

**psha;pushes the cool down counter back onto the stack**

**ldx #array ;loads first element of array into reg. x**

**AGAIN\_CCW: ldab #4 ;acc. b will act as a counter**

**LOOP\_CCW: ldaa b,x**

**cmpa #$ff ;compares with the terminator $ff**

**beq COUNTER\_CW**

**ldy #50000 ;120 cycles means 30microseconds of a delay**

**;just a for loop with 120 itterations**

**DELAY\_CCW:dey ;decrements y by 1**

**cpy #0 ;compares y to see if it is the end of the delay loop**

**bne DELAY\_CCW ;branches back to delay when y!=0**

**staa port\_P**

**decb ;increments b by one**

**bra LOOP\_CCW**

**STOP\_COOLING:**

**movb #0,COOL\_Y\_N**

**rts**

**HEX\_READ.asm**

XDEF HEX\_READ

XREF HEX\_VALUE,port\_S,port\_S\_DDR,DELAY,port\_U,port\_U\_DDR,port\_U\_PSR, PERUu, ROW\_VAL, COORD\_TBL, VALUE\_TBL

HEX\_READ:

pshx;save registers on the stack

pshy

ldx #$0000

ldaa #$f0

staa port\_U\_DDR ;initalize

staa port\_U\_PSR

movb #$0f,PERUu

ldx #ROW\_VAL

deca

cmpa #0

beq QUIT

NEXT\_ROW:

ldab #0;b is a counter

ldaa x; a is a pointer of the row value

oraa #$0f

STAA port\_U

ldy #COORD\_TBL;y is a pointer of the corrd table

NEXT\_VALUE:

incb

ldaa y

cmpa #$ff

beq INC\_X

ldaa port\_U

cmpa 1,y+

beq EQUAL

bra NEXT\_VALUE

INC\_X:

inx

ldaa x

cmpa #$ff

beq QUIT;scaned everything, so exit

bra NEXT\_ROW

EQUAL: ldx #VALUE\_TBL

CHECK\_B:

cmpb #1

beq SET\_HEX

inx

decb

bra CHECK\_B

SET\_HEX:

ldaa x;a now has hex keypad value

staa HEX\_VALUE

;staa port\_S

puly

pulx

rts

QUIT:

puly

pulx

rts

**DISP\_TIME.asm**

XDEF DISP\_TIME

XREF init\_LCD,read\_pot,pot\_value,display\_string, HEX\_VALUE, port\_S, HEX\_READ, disp, RUN\_TIME, WATTAGE, SPIN\_SPEED\_OPTION, port\_T

DISP\_TIME:

movb #'T',disp

movb #'i',disp+1

movb #'m',disp+2

movb #'e',disp+3

movb #':',disp+4

movb #' ',disp+5

movb #'0',disp+6

movb #'0',disp+7

movb #'0',disp+8

movb #'s',disp+9

movb #' ',disp+10

movb #'@',disp+11

movb #'0',disp+12

movb #'0',disp+13

movb #'0',disp+14

movb #'W',disp+15

movb #'S',disp+16

movb #'p',disp+17

movb #'e',disp+18

movb #'e',disp+19

movb #'d',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 ;string terminator, acts like '\0'

;WATTAGE DISPLAY

;=============================================================================================

ldaa WATTAGE;stores the watage value into the constant

adda #48;convert to ascii

staa disp+12

;RUN TIME DISPLAY

;=============================================================================================

ldd RUN\_TIME

ldx #100;hundres place

idiv

tfr x,a

adda #48

staa disp+6

ldaa #0;reset a to its original value which will always be 0

ldx #10 ;tens place

idiv

tfr x,a

adda #48

staa disp+7

ldaa #0;reset a to its original value which will always be 0

addb #48 ;ones place

stab disp+8

jsr init\_LCD

;=============================================================================================

brclr port\_T,%00000001,FAN\_OFF;check if bit 0 is off, if so the fan is OFF

;=============================================================================================

ldaa SPIN\_SPEED\_OPTION;checks to see which mode the fan is in

cmpa #1

beq SLOW\_DISPLAY;because out of beq range

cmpa #2

beq MED\_DISPLAY;because out of beq range

bra FAST\_DISPLAY

;=============================================================================================

RETURN:

ldd #disp

jsr display\_string

rts

SLOW\_DISPLAY:

movb #'S',disp+23

movb #'l',disp+24

movb #'o',disp+25

movb #'w',disp+26

bra RETURN

MED\_DISPLAY:

movb #'M',disp+22

movb #'e',disp+23

movb #'d',disp+24

movb #'i',disp+25

movb #'u',disp+26

movb #'m',disp+27

bra RETURN

FAST\_DISPLAY:

movb #'F',disp+23

movb #'a',disp+24

movb #'s',disp+25

movb #'t',disp+26

bra RETURN

FAN\_OFF:

movb #'F',disp+16

movb #'a',disp+17

movb #'n',disp+18

movb #' ',disp+19

movb #'i',disp+20

movb #'s',disp+21

movb #' ',disp+22

movb #'O',disp+23

movb #'F',disp+24

movb #'F',disp+25

jmp RETURN

**COOL\_DOWN\_DISPLAY.asm**

XDEF COOL\_DOWN\_DISPLAY

XREF init\_LCD,read\_pot,pot\_value,display\_string, HEX\_VALUE, port\_S, HEX\_READ, disp

COOL\_DOWN\_DISPLAY:

movb #'C',disp

movb #'o',disp+1

movb #'o',disp+2

movb #'l',disp+3

movb #'i',disp+4

movb #'n',disp+5

movb #'g',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 ;string terminator, acts like '\0'

jsr init\_LCD

ldd #disp

jsr display\_string

rts