Code Files

\*\*Red bars across represents a new file of code

---------------------------------------------

$NOLIST

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

; Buzzer

;

; Derek Chan

;

; Function: Interface for the Buzzer component

;

; Subroutines:

; Buzzer\_Init: Initlizes P1MOD, TMOD, \*\*\*DOES NOT ENABLE INTERRUPTS\*\*\*

; Buzzer\_ISR: Makes the buzzer beep, \*prioritized\*

; Buzzer\_Start\_Beep: Starts the buzzer

; Buzzer\_Stop\_Beep: Stops the buzzer

; Buzzer\_Beep\_Multiple(int numBeeps): Beeps the buzzer numBeeps times

;

; Interrupt:

; org 000BH

; ljmp Buzzer\_ISR

; Registers:

; P1.5 - Buzzer

; Constants:

; BUZZER\_CLK EQU 33333333

; BUZZER\_FREQ EQU 2000

; BUZZER\_T0\_RELOAD EQU 65536-(BUZZER\_CLK/(12\*2\*BUZZER\_FREQ))

; Variables:

; Buzzer\_Beep\_Count : ds 1

; Buzzer\_Beep\_Num : ds 1

;

; Buzzer\_Beep\_Active : dbit 1

; Buzzer\_Continuous\_Tone : dbit 1

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

CSEG

Buzzer\_Init:

orl P1MOD, #00100000B ; P1.5 is an output

orl TMOD, #00000001B ; GATE=0, C/T\*=0, M1=0, M0=1: 16-bit timer

clr TR0 ; Disable timer 0

clr TF0

mov TH0, #high(BUZZER\_T0\_RELOAD)

mov TL0, #low(BUZZER\_T0\_RELOAD)

setb ET0 ; Enable T0 interrupt

setb PT0 ; Set T0 priority

setb Buzzer\_Beep\_Active

clr Buzzer\_Continuous\_Tone

mov Buzzer\_Beep\_Count, #0

ret

--------------------------------------------

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;Buzzer\_ISR

;;

;;Interrupt service routine for the buzzer. Causes

;;the buzzer to beep according to set parameters.

;;

;;@param - Buzzer\_Continuous\_Tone:

;; Buzzer goes BEEEEEEEEEEEP if set to 1,

;; Buzzer goes beep-beep-beep... if set to 0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Buzzer\_ISR:

push acc

push psw

jb Buzzer\_Continuous\_Tone, Buzzer\_ISR\_Make\_Beep

clr C

mov A, Buzzer\_Beep\_Count

add A, #1

mov Buzzer\_Beep\_Count, A

jnc Buzzer\_ISR\_Check\_Beep

cpl Buzzer\_Beep\_Active

mov A, Buzzer\_Beep\_Num

inc A

mov Buzzer\_Beep\_Num, A

Buzzer\_ISR\_Check\_Beep:

jnb Buzzer\_Beep\_Active, Buzzer\_ISR\_end

Buzzer\_ISR\_Make\_Beep:

clr A

cpl P1.5

Buzzer\_ISR\_end:

mov TH0, #high(BUZZER\_T0\_RELOAD)

mov TL0, #low(BUZZER\_T0\_RELOAD)

pop psw

pop acc

reti

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;Buzzer\_Start\_Beep

;;

;;Starts the buzzer

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Buzzer\_Start\_Beep:

setb TR0

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;Buzzer\_Stop\_Beep

;;

;;Stops the buzzer

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Buzzer\_Stop\_Beep:

clr TR0

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;Buzzer\_Beep\_Multiple(int numBeeps)

;;

;;Macro function that beeps the buzzer 'numBeeps' times

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Buzzer\_Beep\_Multiple MAC

push ACC

mov A, R7

push ACC

mov R7, #(%0)

lcall Buzzer\_Beep\_Multiple\_helper

pop ACC

mov R7, A

pop ACC

ENDMAC

Buzzer\_Beep\_Multiple\_helper:

mov Buzzer\_Beep\_Num, #0

mov A, R7

rl A

mov R7, A

setb TR0

Buzzer\_Beep\_Multiple\_Loop:

mov A, Buzzer\_Beep\_Num

clr c

subb A, R7

jc Buzzer\_Beep\_Multiple\_Loop

clr TR0

ret

$LIST

---------------------------------------------

$NOLIST

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

; Door

;

; Derek Chan

;

; Function: Controls the SSR Relay box

; Pins: P1.1 - Door input

; Constants and Variables to be declared: Door\_Open: 1 if door is open, 0 if door is closed

; Functions:

; SSR\_Enable - turns on the SSR

; SSR\_Disable - turns off the SSR

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

CSEG

Door\_Init:

anl P1MOD, #11111101B ;set P1MOD as input

ret

---------------------------------------------

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;Door\_Check

;;

;;Updates Door\_Open bit to indicate if the door

;;is open (1) or closed (0)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Door\_Check:

jb P1.1, Door\_Check\_closed

;door is open

setb Door\_Open

ret

Door\_Check\_closed:

clr Door\_Open

ret

$LIST

$NOLIST

;Function: 1)To initialize the LCD Display

; 2)To be able to write text to the LCD

;

;note: functions taken from spi.logger.asm

;

; Functions added:

; 1)

;

Wait40us:

mov R0, #149

Wait40us\_L0:

nop

nop

nop

nop

nop

nop

djnz R0, Wait40us\_L0 ; 9 machine cycles-> 9\*30ns\*149=40us

ret

LCD\_command:

mov LCD\_DATA, A

clr LCD\_RS

nop

nop

setb LCD\_EN ; Enable pulse should be at least 230 ns

nop

nop

nop

nop

nop

nop

clr LCD\_EN

ljmp Wait40us

LCD\_put:

mov LCD\_DATA, A

setb LCD\_RS

nop

nop

setb LCD\_EN ; Enable pulse should be at least 230 ns

nop

nop

nop

nop

nop

nop

clr LCD\_EN

ljmp Wait40us

LCD\_Init:

; Turn LCD on, and wait a bit.

setb LCD\_ON

clr LCD\_EN ; Default state of enable must be zero

lcall Wait40us

mov LCD\_MOD, #0xff ; Use LCD\_DATA as output port

clr LCD\_RW ; Only writing to the LCD in this code.

mov a, #0ch ; Display on command

lcall LCD\_command

mov a, #38H ; 8-bits interface, 2 lines, 5x7 characters

lcall LCD\_command

mov a, #01H ; Clear screen (Warning, very slow command!)

lcall LCD\_command

; Delay loop needed for 'clear screen' command above (1.6ms at least!)

mov R1, #40

Clr\_loop:

lcall Wait40us

djnz R1, Clr\_loop

ret

Display\_welcome\_message:

; Display the first row

mov a, #80H

lcall LCD\_command

mov a, #'R'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'f'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'w'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'O'

lcall LCD\_put

mov a, #'v'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

;display the second row

mov a, #0c0H

lcall LCD\_command

mov a, #'C'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'I'

lcall LCD\_put

mov a, #'s'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'O'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

ret

Display\_open\_door:

; Display the first row

mov a, #80H

lcall LCD\_command

mov a, #'P'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'s'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'p'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

;display the second row

mov a, #0c0H

lcall LCD\_command

mov a, #'o'

lcall LCD\_put

mov a, #'v'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'d'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

ret

Display\_close\_door:

; Display the first row

mov a, #80H

lcall LCD\_command

mov a, #'P'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'s'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'c'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'s'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

;display the second row

mov a, #0c0H

lcall LCD\_command

mov a, #'o'

lcall LCD\_put

mov a, #'v'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'d'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

ret

Display\_fatal\_error:

; Display the first row

mov a, #80H

lcall LCD\_command

mov a, #'F'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'E'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

;display the second row

mov a, #0c0H

lcall LCD\_command

mov a, #'P'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'s'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'R'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'s'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

ret

Display\_soak\_temp\_set:

; Display the first row

mov a, #80H

lcall LCD\_command

mov a, #'P'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'s'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'E'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

;display the second row

mov a, #0c0H

lcall LCD\_command

mov a, #'S'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'k'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'T'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'m'

lcall LCD\_put

mov a, #'p'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

ret

Display\_soak\_time\_set:

; Display the first row

mov a, #80H

lcall LCD\_command

mov a, #'P'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'s'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'E'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

;display the second row

mov a, #0c0H

lcall LCD\_command

mov a, #'S'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'k'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'T'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #'m'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

ret

Display\_reflow\_temp\_set:

; Display the first row

mov a, #80H

lcall LCD\_command

mov a, #'P'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'s'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'E'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

;display the second row

mov a, #0c0H

lcall LCD\_command

mov a, #'R'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'f'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'w'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'T'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'m'

lcall LCD\_put

mov a, #'p'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

ret

Display\_reflow\_time\_set:

; Display the first row

mov a, #80H

lcall LCD\_command

mov a, #'P'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'s'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'E'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

;display the second row

mov a, #0c0H

lcall LCD\_command

mov a, #'R'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'f'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'w'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'T'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #'m'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

ret

Display\_Confirmation\_message:

; Display the first row

mov a, #80H

lcall LCD\_command

mov a, #'C'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #'f'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #'m'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #'g'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

;display the second row

mov a, #0c0H

lcall LCD\_command

mov a, #'V'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'u'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'s'

lcall LCD\_put

mov a, #'.'

lcall LCD\_put

mov a, #'.'

lcall LCD\_put

mov a, #'.'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

lcall waitHalfSec

lcall waitHalfSec

lcall WaitHalfSec

lcall WaitHalfSec

; Display the first row

mov a, #80H

lcall LCD\_command

mov a, #'S'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'k'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'T'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'m'

lcall LCD\_put

mov a, #'p'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov bcd+0, soak\_temperature+0

mov bcd+1, soak\_temperature+1

mov bcd+2, soak\_temperature+2

mov a, bcd+1

anl a, #0fH

orl a, #30H

lcall LCD\_put

mov a, bcd+0

swap a

anl a, #0fH

orl a, #30H

lcall LCD\_put

mov a, bcd+0

anl a, #0fH

orl a, #30h

lcall LCD\_put

;lower value

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

;display the second row

mov a, #0c0H

lcall LCD\_command

mov a, #'S'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'k'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'T'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #'m'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov bcd+0, soak\_time+0

mov bcd+1, soak\_time+1

mov bcd+2, soak\_time+2

mov a, bcd+1

anl a, #0fH

orl a, #30H

lcall LCD\_put

mov a, bcd+0

swap a

anl a, #0fH

orl a, #30H

lcall LCD\_put

mov a, bcd+0

anl a, #0fH

orl a, #30h

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

lcall Wait\_for\_Confirmation

; Display the first row

mov a, #80H

lcall LCD\_command

mov a, #'R'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'f'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'w'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'T'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'m'

lcall LCD\_put

mov a, #'p'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov bcd+0, reflow\_temperature+0

mov bcd+1, reflow\_temperature+1

mov bcd+2, reflow\_temperature+2

mov a, bcd+1

anl a, #0fH

orl a, #30H

lcall LCD\_put

mov a, bcd+0

swap a

anl a, #0fH

orl a, #30H

lcall LCD\_put

mov a, bcd+0

anl a, #0fH

orl a, #30h

lcall LCD\_put

mov a, #' '

lcall LCD\_put

;display the second row

mov a, #0c0H

lcall LCD\_command

mov a, #'R'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'f'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'w'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'T'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #'m'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov bcd+0, reflow\_time+0

mov bcd+1, reflow\_time+1

mov bcd+2, reflow\_time+2

mov a, bcd+1

anl a, #0fH

orl a, #30H

lcall LCD\_put

mov a, bcd+0

swap a

anl a, #0fH

orl a, #30H

lcall LCD\_put

mov a, bcd+0

anl a, #0fH

orl a, #30h

lcall LCD\_put

mov a, #' '

lcall LCD\_put

lcall waitHalfSec

lcall Wait\_for\_Confirmation

ret

Display\_preset\_or\_manual:

mov a, #80h

lcall LCD\_command

mov a, #'C'

lcall LCD\_put

mov a, #'h'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'s'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'P'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'s'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #0c0H

lcall LCD\_command

mov a, #'I'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #'p'

lcall LCD\_put

mov a, #'u'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'B'

lcall LCD\_put

mov a, #'y'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'H'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #'d'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

ret

Display\_Status:

mov a, #80H

lcall LCD\_command

mov a, #'T'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

;displaying current temperature

mov x+0, temperature\_measured+0

mov x+1, temperature\_measured+1

lcall hex2bcd

mov a, bcd+1

anl a, #00001111B

orl a, #30H

lcall LCD\_put

mov a, bcd+0

swap a

anl a, #00001111B

orl a, #30H

lcall LCD\_put

mov a, bcd+0

anl a, #00001111B

orl a, #30H

lcall LCD\_put

mov a, #'C'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'T'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov x+0, target\_temperature+0

mov x+1, target\_temperature+1

lcall hex2bcd

mov a, bcd+1

anl a, #00001111B

orl a, #30H

lcall LCD\_put

mov a, bcd+0

swap a

anl a, #00001111B

orl a, #30H

lcall LCD\_put

mov a, bcd+0

anl a, #00001111B

orl a, #30H

lcall LCD\_put

mov a, #'C'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

;display the second row

mov a, #0c0H

lcall LCD\_command

mov a, #'S'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

;need to have logic to test which state we're in

mov a, state

cjne a, #STATE\_STANDBY, G1

mov a, #'S'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #'d'

lcall LCD\_put

mov a, #'b'

lcall LCD\_put

mov a, #'y'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

G1: mov a, state

cjne a, #STATE\_HEATING1, G2

mov a, #'H'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #'g'

lcall LCD\_put

mov a, #'1'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

G2: mov a, state

cjne a, #STATE\_SOAK, G3

mov a, #'S'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'k'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

G3: mov a, state

cjne a, #STATE\_HEATING2, G4

mov a, #'H'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #'g'

lcall LCD\_put

mov a, #'2'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

G4: mov a, state

cjne a, #STATE\_REFLOW, G5

mov a, #'R'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'f'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'w'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

G5: mov a, state

cjne a, #STATE\_COOLDOWN, G6

mov a, #'C'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'d'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'w'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

G6: mov a, state

cjne a, #STATE\_OPEN\_DOOR, G7

mov a, #'O'

lcall LCD\_put

mov a, #'p'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'D'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

G7:

ret

Display\_Options:

mov a, #80H

lcall LCD\_command

mov a, #1H

orl a, #30H

lcall LCD\_put

mov a, #')'

lcall LCD\_put

mov a, #'S'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov a, #01H

orl a, #30H

lcall LCD\_put

mov a, #03H

orl a, #30H

lcall LCD\_put

mov a, #00H

orl a, #30H

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'S'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov a, #0H

orl a, #30H

lcall LCD\_put

mov a, #06H

orl a, #30H

lcall LCD\_put

mov a, #00H

orl a, #30H

lcall LCD\_put

mov a, #0c0H

lcall LCD\_command

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'R'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov a, #02H

orl a, #30H

lcall LCD\_put

mov a, #01H

orl a, #30H

lcall LCD\_put

mov a, #00H

orl a, #30H

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'R'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov a, #0H

orl a, #30H

lcall LCD\_put

mov a, #03H

orl a, #30H

lcall LCD\_put

mov a, #0H

orl a, #30H

lcall LCD\_put

lcall waithalfsec

lcall waithalfsec

lcall waithalfsec

lcall waithalfsec

lcall waithalfsec

lcall waithalfsec

lcall waithalfsec

lcall waithalfsec

mov a, #80H

lcall LCD\_command

mov a, #2H

orl a, #30H

lcall LCD\_put

mov a, #')'

lcall LCD\_put

mov a, #'S'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov a, #01H

orl a, #30H

lcall LCD\_put

mov a, #05H

orl a, #30H

lcall LCD\_put

mov a, #00H

orl a, #30H

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'S'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov a, #0H

orl a, #30H

lcall LCD\_put

mov a, #9H

orl a, #30H

lcall LCD\_put

mov a, #0H

orl a, #30H

lcall LCD\_put

mov a, #0c0H

lcall LCD\_command

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'R'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov a, #02H

orl a, #30H

lcall LCD\_put

mov a, #02H

orl a, #30H

lcall LCD\_put

mov a, #00H

orl a, #30H

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'R'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov a, #0H

orl a, #30H

lcall LCD\_put

mov a, #4H

orl a, #30H

lcall LCD\_put

mov a, #0H

orl a, #30H

lcall LCD\_put

lcall waithalfsec

lcall waithalfsec

lcall waithalfsec

lcall waithalfsec

lcall waithalfsec

lcall waithalfsec

lcall waithalfsec

lcall waithalfsec

mov a, #80H

lcall LCD\_command

mov a, #3H

orl a, #30H

lcall LCD\_put

mov a, #')'

lcall LCD\_put

mov a, #'S'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov a, #01H

orl a, #30H

lcall LCD\_put

mov a, #07H

orl a, #30H

lcall LCD\_put

mov a, #00H

orl a, #30H

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'S'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov a, #1H

orl a, #30H

lcall LCD\_put

mov a, #2H

orl a, #30H

lcall LCD\_put

mov a, #0H

orl a, #30H

lcall LCD\_put

mov a, #0c0H

lcall LCD\_command

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'R'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov a, #02H

orl a, #30H

lcall LCD\_put

mov a, #03H

orl a, #30H

lcall LCD\_put

mov a, #00H

orl a, #30H

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'R'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov a, #0H

orl a, #30H

lcall LCD\_put

mov a, #4H

orl a, #30H

lcall LCD\_put

mov a, #5H

orl a, #30H

lcall LCD\_put

lcall waithalfsec

lcall waithalfsec

lcall waithalfsec

lcall waithalfsec

lcall waithalfsec

lcall waithalfsec

lcall waithalfsec

lcall waithalfsec

mov a, #80H

lcall LCD\_command

mov a, #'P'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'s'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'c'

lcall LCD\_put

mov a, #'h'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'s'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #0c0H

lcall LCD\_command

mov a, #'O'

lcall LCD\_put

mov a, #'p'

lcall LCD\_put

mov a, #'t'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #01H

orl a, #30H

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #02H

orl a, #30H

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #03H

orl a, #30H

lcall LCD\_put

mov a, #' '

lcall LCD\_put

ret

display\_soak\_temperature\_high:

mov a, #80h

lcall LCD\_command

mov a, #'E'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'S'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'k'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'T'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'m'

lcall LCD\_put

mov a, #'p'

lcall LCD\_put

mov a, #0c0H

lcall LCD\_command

mov a, #'E'

lcall LCD\_put

mov a, #'x'

lcall LCD\_put

mov a, #'c'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'d'

lcall LCD\_put

mov a, #'s'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'M'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'x'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'T'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'m'

lcall LCD\_put

mov a, #'p'

lcall LCD\_put

ret

display\_reflow\_temperature\_high:

mov a, #80h

lcall LCD\_command

mov a, #'E'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'R'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'f'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'T'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'m'

lcall LCD\_put

mov a, #'p'

lcall LCD\_put

mov a, #0c0H

lcall LCD\_command

mov a, #'E'

lcall LCD\_put

mov a, #'x'

lcall LCD\_put

mov a, #'c'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'d'

lcall LCD\_put

mov a, #'s'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'M'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'x'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'T'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'m'

lcall LCD\_put

mov a, #'p'

lcall LCD\_put

ret

display\_reflow\_time\_high:

mov a, #80h

lcall LCD\_command

mov a, #'E'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #':'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'R'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'f'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'T'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #'m'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #0c0H

lcall LCD\_command

mov a, #'E'

lcall LCD\_put

mov a, #'x'

lcall LCD\_put

mov a, #'c'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'d'

lcall LCD\_put

mov a, #'s'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'M'

lcall LCD\_put

mov a, #'a'

lcall LCD\_put

mov a, #'x'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'T'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #'m'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

ret

display\_finished:

mov a, #80h

lcall LCD\_command

mov a, #'S'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'l'

lcall LCD\_put

mov a, #'d'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'r'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #'g'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'D'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'n'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #0c0H

lcall LCD\_command

mov a, #'R'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #'m'

lcall LCD\_put

mov a, #'o'

lcall LCD\_put

mov a, #'v'

lcall LCD\_put

mov a, #'e'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #'C'

lcall LCD\_put

mov a, #'h'

lcall LCD\_put

mov a, #'i'

lcall LCD\_put

mov a, #'p'

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

mov a, #' '

lcall LCD\_put

ret

end

---------------------------------------------

$modde2

org 0000H

ljmp main\_init

org 000BH

ljmp Buzzer\_ISR

org 001BH

ljmp ISR\_Timer

;Thermocouple Constants

FREQ EQU 33333333

BAUD EQU 115200

T2LOAD EQU 65536-(FREQ/(32\*BAUD))

THERMO\_TEMP\_ADJ EQU 5 ;negative offset because LM335 reports too high

OVERSHOOT\_COMPENSATE EQU 25

;Serial Port Constants

MISO EQU P0.0

MOSI EQU P0.1

SCLK EQU P0.2

CE\_ADC EQU P0.3

CE\_EE EQU P0.4

CE\_RTC EQU P0.5

;Buzzer Constants

BUZZER\_CLK EQU 33333333

BUZZER\_FREQ EQU 2000

BUZZER\_T0\_RELOAD EQU 65536-(BUZZER\_CLK/(12\*2\*BUZZER\_FREQ))

;Timer Constants

Timer\_XTAL EQU 33333333

Timer\_FREQ EQU 100

TIMER1\_RELOAD EQU 65538-(Timer\_XTAL/(12\*Timer\_FREQ))

;State Constants

STATE\_STANDBY EQU 0

STATE\_HEATING1 EQU 1

STATE\_SOAK EQU 2

STATE\_HEATING2 EQU 3

STATE\_REFLOW EQU 4

STATE\_COOLDOWN EQU 5

STATE\_OPEN\_DOOR EQU 6

STATE\_FAILURE EQU 7

STATE\_DONE EQU 8

DSEG at 30H

;User\_Interface Variables

soak\_temperature : ds 2

soak\_time : ds 2

reflow\_temperature : ds 2

reflow\_time : ds 2

target\_temperature : ds 2

;Thermocouple Variables

Temperature\_Measured : ds 2

Outside\_Temperature\_Measured: ds 2

;Buzzer Variables

Buzzer\_Beep\_Count : ds 1

Buzzer\_Beep\_Num : ds 1

;Timer

Timer\_count10ms: ds 1

Timer\_Total\_Time\_Seconds: ds 1 ;incrementing every second

Timer\_Total\_Time\_Minutes: ds 1 ;incrementing every minute

Timer\_Elapsed\_Time: ds 2 ;incrementing every second

;Math16/32 Variables

x : ds 2

y : ds 2

bcd : ds 3

op : ds 1

;State

state : ds 1

BSEG

;Math16

mf : dbit 1

;Door

Door\_Open : dbit 1

;Buzzer bit variables

Buzzer\_Beep\_Active : dbit 1

Buzzer\_Continuous\_Tone : dbit 1

;Thermo2 Variables

Temperature\_Measured\_Sign : dbit 1

;UI Variables

UI\_Input\_Error : dbit 1

CSEG

$include(math16.asm)

$include(SSR.asm)

$include(Serial\_Port.asm)

$include(Buzzer.asm)

$include(Thermo2.asm)

$include(User\_Interface.asm)

$include(LCD\_Display.asm)

$include(Door.asm)

$include(Timer.asm)

$include(Read\_sw5.asm)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;main\_Maintain\_Temperature(var temp)

;;

;;Maintains the desired temperature given in var temp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main\_Maintain\_Temperature MAC

lcall Thermocouple\_Update

mov x+0, Temperature\_Measured+0

mov x+1, Temperature\_Measured+1

mov y+0, %0+0

mov y+1, %0+1

lcall x\_lt\_y

lcall main\_Maintain\_Temperature\_helper

ENDMAC

main\_Maintain\_Temperature\_helper:

jb mf, main\_Maintain\_Temperature\_tooCold

;too hot:

lcall SSR\_Disable

ret

main\_Maintain\_Temperature\_tooCold:

lcall SSR\_Enable

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;main\_Alert\_Open\_Door

;;

;;Stalling code that stops all functions until the

;;door is closed

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main\_Alert\_Open\_Door:

lcall Door\_Check

jnb Door\_Open, main\_Alert\_Open\_Door\_done

lcall Display\_Close\_Door

setb Buzzer\_Continuous\_Tone

clr ET1

lcall Buzzer\_Start\_Beep

mov A, #0FFH

mov LEDRA, A

mov LEDRB, A

mov LEDRC, A

mov LEDG, A

main\_Alert\_Open\_Door\_loop:

lcall Door\_Check

jb Door\_Open, main\_Alert\_Open\_Door\_loop

setb ET1

clr Buzzer\_Continuous\_Tone

lcall Buzzer\_Stop\_Beep

mov A, #0

mov LEDRA, A

mov LEDRB, A

mov LEDRC, A

mov LEDG, A

main\_Alert\_Open\_Door\_done:

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;main\_state\_standby

;;

;;Function:

;; \*Checks all inputs

;; \*Turns all outputs off

;; \*Displays message "Please specify parameters: "

;; "Soak: (temp/time) | Reflow (temp/time) "

;;

;;State Change:

;; STATE\_HEATING1:

;; \*On button pressed

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main\_state\_standby:

;mov LEDRA, #10000000B

lcall UI\_Set\_Up\_Parameters

lcall test\_proper\_values

jb UI\_Input\_Error, main\_state\_standby

;;

;;TODO: remove this override

;;;

;mov soak\_temperature, #low(150)

;mov soak\_temperature+1, #high(150)

;mov soak\_time, #low(90)

;mov soak\_time+1, #high(90)

;mov reflow\_temperature, #low(217)

;mov reflow\_temperature+1, #high(217)

;mov reflow\_time, #low(55)

;mov reflow\_time+1, #high(55)

mov state, #STATE\_HEATING1

lcall Timer\_Reset

mov target\_temperature, soak\_temperature

mov target\_temperature+1, soak\_temperature+1

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;main\_state\_heating1

;;

;;Function:

;; \*Heats the oven to the soak temperature

;;

;;State Change:

;; STATE\_SOAK:

;; \*Temperature\_Measured == soak\_temperature

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main\_state\_heating1:

mov LEDRA, #00100000B

lcall main\_Alert\_Open\_Door

lcall Timer\_Display

lcall waitHalfSec ;delay to make the LCD not glitch up

lcall Thermocouple\_Update

lcall Serial\_Port\_Send\_String

lcall Display\_Status ;this is UI\_Update

mov x+0, Temperature\_Measured+0

mov x+1, Temperature\_Measured+1

mov y+0, soak\_temperature+0

mov y+1, soak\_temperature+1

lcall x\_lt\_y

jb mf, main\_state\_heating1\_close

mov state, #STATE\_SOAK

Buzzer\_Beep\_Multiple(4)

lcall Timer\_Reset\_Elapsed\_Time

main\_state\_heating1\_close:

load\_y(OVERSHOOT\_COMPENSATE)

lcall add16

mov y+0, soak\_temperature+0

mov y+1, soak\_temperature+1

lcall x\_lt\_y

jb mf, main\_state\_heating1\_else

lcall SSR\_Disable

sjmp main\_state\_heating1\_done

main\_state\_heating1\_else:

lcall SSR\_Enable

main\_state\_heating1\_done:

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;main\_state\_soak

;;

;;Function:

;; \*Maintains oven temperature for the time specified

;; in soak\_time

;;

;;State Change:

;; STATE\_HEATING2:

;; \*elapsed\_time == soak\_time

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main\_state\_soak:

mov LEDRA, #00110000B

lcall main\_Alert\_Open\_Door

lcall Display\_Status ;this is UI\_Update

lcall Timer\_Display

lcall waitHalfSec ;delay to make the LCD not glitch up

main\_Maintain\_Temperature(soak\_temperature)

lcall Serial\_Port\_Send\_String

mov x+0, Timer\_elapsed\_time+0

mov x+1, Timer\_elapsed\_time+1

mov y+0, soak\_time+0

mov y+1, soak\_time+1

lcall x\_lt\_y

jb mf, main\_state\_soak\_done

mov state, #STATE\_HEATING2

mov target\_temperature, reflow\_temperature

mov target\_temperature+1, reflow\_temperature+1

Buzzer\_Beep\_Multiple(4)

lcall Timer\_Reset\_Elapsed\_Time

mov target\_temperature, reflow\_temperature

main\_state\_soak\_done:

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;main\_state\_heating2

;;

;;Function:

;; \*Heats the oven to the reflow temperature

;;

;;State Change:

;; STATE\_REFLOW:

;; \*Temperature\_Measured == reflow\_temperature

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main\_state\_heating2:

mov LEDRA, #00111000B

lcall main\_Alert\_Open\_Door

lcall Display\_Status ;this is UI\_Update

lcall Timer\_Display

lcall waitHalfSec ;delay to make the LCD not glitch up

lcall Thermocouple\_Update

lcall Serial\_Port\_Send\_String

mov x+0, Temperature\_Measured+0

mov x+1, Temperature\_Measured+1

mov y+0, reflow\_temperature+0

mov y+1, reflow\_temperature+1

lcall x\_lt\_y

jb mf, main\_state\_heating2\_close

mov state, #STATE\_REFLOW

Buzzer\_Beep\_Multiple(4)

lcall Timer\_Reset\_Elapsed\_Time

main\_state\_heating2\_close:

load\_y(5)

lcall add16

mov y+0, reflow\_temperature+0

mov y+1, reflow\_temperature+1

lcall x\_lt\_y

jb mf, main\_state\_heating2\_else

lcall SSR\_Disable

sjmp main\_state\_heating2\_done

main\_state\_heating2\_else:

lcall SSR\_Enable

main\_state\_heating2\_done:

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;main\_state\_reflow

;;

;;Function:

;; \*Maintains oven temperature at reflow\_temperature

;; for the time specified in reflow\_time

;;

;;State Change:

;; STATE\_COOLDOWN:

;; \*elapsed\_time == reflow\_time

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main\_state\_reflow:

mov LEDRA, #00111100B

lcall main\_Alert\_Open\_Door

lcall Display\_Status ;this is UI\_Update

lcall Timer\_Display

lcall waitHalfSec ;delay to make the LCD not glitch up

main\_Maintain\_Temperature(reflow\_temperature)

lcall Serial\_Port\_Send\_String

mov x+0, Timer\_elapsed\_time+0

mov x+1, Timer\_elapsed\_time+1

mov y+0, reflow\_time+0

mov y+1, reflow\_time+1

lcall x\_lt\_y

jb mf, main\_state\_reflow\_done

mov state, #STATE\_OPEN\_DOOR

Buzzer\_Beep\_Multiple(4)

lcall Timer\_Reset\_Elapsed\_Time

mov target\_temperature, #low(40)

mov target\_temperature+1, #high(40)

main\_state\_reflow\_done:

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;main\_state\_cooldown

;;

;;Function:

;; \*Beep until the door opens

;;

;;State Change:

;; STATE\_OPEN\_DOOR:

;; \*Door\_Open == true

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main\_state\_cooldown:

mov LEDRA, #00111110B

lcall Display\_Status ;this is UI\_Update

lcall Timer\_Display

lcall waitHalfSec ;delay to make the LCD not glitch up

lcall Thermocouple\_Update

lcall Serial\_Port\_Send\_String

lcall SSR\_Disable

setb Buzzer\_Continuous\_Tone

lcall Buzzer\_Start\_Beep

lcall Door\_Check

jnb Door\_Open, main\_state\_cooldown\_done

mov state, #STATE\_OPEN\_DOOR

lcall Buzzer\_Stop\_Beep

clr Buzzer\_Continuous\_Tone

lcall Timer\_Reset\_Elapsed\_Time

main\_state\_cooldown\_done:

ret

---

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;main\_state\_open\_door

;;

;;Function:

;; \*Waits until component is cool

;;

;;State Change:

;; STATE\_STANDBY:

;; \*Temperature\_Measured < 40 degrees C

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main\_state\_open\_door:

;check if the door got re-closed

lcall Door\_Check

jb Door\_Open, main\_state\_open\_door\_next

lcall Display\_Open\_Door

setb Buzzer\_Continuous\_Tone

clr ET1

lcall Buzzer\_Start\_Beep

mov A, #0FFH

mov LEDRA, A

mov LEDRB, A

mov LEDRC, A

mov LEDG, A

main\_state\_Open\_Door\_loop:

lcall Door\_Check

jnb Door\_Open, main\_state\_Open\_Door\_loop

setb ET1

clr Buzzer\_Continuous\_Tone

lcall Buzzer\_Stop\_Beep

mov A, #0

mov LEDRA, A

mov LEDRB, A

mov LEDRC, A

mov LEDG, A

main\_state\_open\_door\_next:

mov LEDRA, #00111111B

lcall Display\_Status ;this is UI\_Update

lcall Timer\_Display

lcall waitHalfSec ;delay to make the LCD not glitch up

lcall Thermocouple\_Update

lcall Serial\_Port\_Send\_String

mov x+0, Temperature\_Measured+0

mov x+1, Temperature\_Measured+1

mov y+0, #low(40)

mov y+1, #high(40)

lcall x\_gt\_y

jb mf, main\_state\_open\_door\_done

mov state, #STATE\_DONE

Buzzer\_Beep\_Multiple(3)

lcall Timer\_Reset

main\_state\_open\_door\_done:

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;main\_state\_done

;;

;;Finished!

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main\_state\_done:

mov LEDRA, #00111111B

mov LEDG, #10000000B

lcall Display\_Finished

mov x+0, Timer\_elapsed\_time+0

mov x+1, Timer\_elapsed\_time+1

mov y+0, #low(5)

mov y+1, #high(5)

lcall x\_lt\_y

jb mf, main\_state\_done\_done

mov state, #STATE\_STANDBY

lcall Timer\_Reset

lcall Timer\_Clear

clr A

mov LEDRA, A

mov LEDG, A

main\_state\_done\_done:

ret

---------------------------------------------

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;main\_state\_failure

;;

;;Something went horribly wrong with the process

;;Wait for user to open door, then reset machine

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main\_state\_failure:

lcall Timer\_Clear

setb Buzzer\_Continuous\_Tone

lcall Buzzer\_Start\_Beep

;lcall LCD\_Please\_Close\_Door

main\_state\_failure\_loop:

lcall Door\_check

jnb Door\_Open, main\_state\_failure\_loop

;lcall LCD\_Critical\_Error

main\_state\_failure\_forever:

sjmp main\_state\_failure\_forever

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;init

;;

;;Starting point for the program. Initializes

;;all of the components for the controller.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main\_init:

mov SP, #7FH

mov A, #0FFH

mov HEX0, A

mov HEX1, A

mov HEX2, A

mov HEX3, A

mov HEX4, A

mov HEX5, A

mov HEX6, A

mov HEX7, A

clr A

clr C

mov LEDG, A

mov LEDRA, A

mov LEDRB, A

mov LEDRC, A

mov state, #STATE\_STANDBY ;initialize state

mov soak\_temperature, A

mov soak\_time, A

mov reflow\_temperature, A

mov reflow\_time, A

mov target\_temperature, A

mov temperature\_measured, A

mov outside\_temperature\_measured, A

mov soak\_temperature+1, A

mov soak\_time+1, A

mov reflow\_temperature+1, A

mov reflow\_time+1, A

mov target\_temperature+1, A

mov temperature\_measured+1, A

lcall SSR\_init

lcall Serial\_Port\_init

lcall Thermocouple\_Input\_init

lcall Init\_Timer

lcall Door\_init

lcall LCD\_init

lcall Buzzer\_init

setb EA

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;main:

;

;State-based outputs are generated according to

;the inputs received.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main:

mov A, state

cjne A, #STATE\_STANDBY, main\_checkEmergencyStop

lcall main\_state\_standby

sjmp main

;check emergency stop button

main\_checkEmergencyStop:

mov A, SWC

jnb ACC.1, main\_heating1

mov A, #0FFH

mov LEDRA, A

mov LEDRB, A

mov LEDRC, A

mov LEDG, A

die:

mov A, SWC

jb ACC.1, die

mov state, #STATE\_STANDBY

ljmp main\_init

;lcall check\_emergency\_stop

main\_heating1:

mov A, state

cjne A, #STATE\_HEATING1, main\_soak

lcall main\_state\_heating1

sjmp main

main\_soak:

cjne A, #STATE\_SOAK, main\_heating2

lcall main\_state\_soak

sjmp main

main\_heating2:

cjne A, #STATE\_HEATING2, main\_reflow

lcall main\_state\_heating2

sjmp main

main\_reflow:

cjne A, #STATE\_REFLOW, main\_cooldown

lcall main\_state\_reflow

sjmp main

main\_cooldown:

cjne A, #STATE\_COOLDOWN, main\_open\_door

lcall main\_state\_cooldown

sjmp main

main\_open\_door:

cjne A, #STATE\_OPEN\_DOOR, main\_done

lcall main\_state\_open\_door

sjmp main

main\_done:

cjne A, #STATE\_DONE, main\_failure

lcall main\_state\_done

sjmp main

main\_failure:

cjne A, #STATE\_FAILURE, main\_error

ljmp main\_failure

;if for some reason, our state is an incorrect value,

; reset the device for safety

main\_error:

ljmp main\_init

END

---------------------------------------------

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Read\_Switch5

;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

$NOLIST

myLUT:

DB 0C0H, 0F9H, 0A4H, 0B0H, 099H ; 0 TO 4

DB 092H, 082H, 0F8H, 080H, 090H ; 4 TO 9

DB 088H, 083H, 0C6H, 0A1H, 086H, 08EH ; A to F

Display:

mov dptr, #myLUT

; Display Digit 0

mov A, bcd+0

anl a, #0fh

movc A, @A+dptr

mov HEX0, A

; Display Digit 1

mov A, bcd+0

swap a

anl a, #0fh

movc A, @A+dptr

mov HEX1, A

; Display Digit 2

mov A, bcd+1

anl a, #0fh

movc A, @A+dptr

mov HEX2, A

; Display Digit 3

;mov A, bcd+1

;swap a

;anl a, #0fh

;movc A, @A+dptr

;mov HEX3, A

; Display Digit 4

;mov A, bcd+2

;anl a, #0fh

;movc A, @A+dptr

;mov HEX4, A

ret

Shift\_Digits:

mov R0, #4 ; shift left four bits

Shift\_Digits\_L0:

clr c

mov a, bcd+0

rlc a

mov bcd+0, a

mov a, bcd+1

rlc a

mov bcd+1, a

mov a, bcd+2

rlc a

mov bcd+2, a

djnz R0, Shift\_Digits\_L0

; R7 has the new bcd digit

mov a, R7

orl a, bcd+0

mov bcd+0, a

; make the four most significant bits of bcd+2 zero

mov a, bcd+2

anl a, #0fH

mov bcd+2, a

ret

$NOLIST

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;Motor

;;

;;Driver for the motor

;;

;;Constants:

;; ORANGE EQU P1.3

;; YELLOW EQU P1.4

;; PINK EQU P1.6

;; BLUE EQU P1.7

;;Ports:

;; P1.2 - Red wire

;; P1.3 - Orange wire

;; P1.4 - Yellow wire

;; P1.6 - Pink wire

;; P1.7 - Blue wire

;;Variables:

;; Motor\_Phase: ds 1

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Motor\_ISR:

push ACC

mov A, Motor\_Phase

cjne A, #0, Motor\_ISR\_phase1

mov LEDRB, A

setb ORANGE

clr YELLOW

clr PINK

clr BLUE

add A, #1

mov Motor\_Phase, A

pop ACC

ret

Motor\_ISR\_phase1:

cjne A, #1, Motor\_ISR\_phase2

mov LEDRB, A

setb ORANGE

setb YELLOW

clr PINK

clr BLUE

add A, #1

mov Motor\_Phase, A

pop ACC

ret

Motor\_ISR\_phase2:

cjne A, #2, Motor\_ISR\_phase3

mov LEDRB, A

clr ORANGE

setb YELLOW

clr PINK

clr BLUE

add A, #1

mov Motor\_Phase, A

pop ACC

ret

Motor\_ISR\_phase3:

cjne A, #3, Motor\_ISR\_phase4

mov LEDRB, A

clr ORANGE

setb YELLOW

setb PINK

clr BLUE

inc A

mov Motor\_Phase, A

pop ACC

ret

Motor\_ISR\_phase4:

cjne A, #4, Motor\_ISR\_phase5

mov LEDRB, A

clr ORANGE

clr YELLOW

setb PINK

clr BLUE

inc A

mov Motor\_Phase, A

pop ACC

ret

Motor\_ISR\_phase5:

cjne A, #5, Motor\_ISR\_phase6

mov LEDRB, A

clr ORANGE

clr YELLOW

setb PINK

setb BLUE

inc A

mov Motor\_Phase, A

pop ACC

ret

Motor\_ISR\_phase6:

cjne A, #6, Motor\_ISR\_phase7

mov LEDRB, A

clr ORANGE

clr YELLOW

clr PINK

setb BLUE

inc A

mov Motor\_Phase, A

pop ACC

ret

Motor\_ISR\_phase7:

mov LEDRB, A

setb ORANGE

clr YELLOW

clr PINK

setb BLUE

mov Motor\_Phase, #0

pop ACC

ret

Motor\_Init:

orl P1MOD, #11011100B

setb P1.2

ret

Motor\_WaitHalfSec:

mov R2, #90

N3: mov R1, #250

N2: mov R0, #250

N1: djnz R0, N1 ; 3 machine cycles-> 3\*30ns\*250=22.5us

djnz R1, N2 ; 22.5us\*250=5.625ms

djnz R2, N3 ; 5.625ms\*90=0.5s (approximately)

ret

---------------------------------------------

$LIST

Wait50ms:

;33.33MHz, 1 clk per cycle: 0.03us

mov R0, #30

L3: mov R1, #74

L2: mov R2, #250

L1: djnz R2, L1 ;3\*250\*0.03us=22.5us

djnz R1, L2 ;74\*22.5us=1.665ms

djnz R0, L3 ;1.665ms\*30=50ms

ret

; Check if SW0 to SW15 are toggled up. Returns the toggled switch in

; R7. If the carry is not set, no toggling switches were detected.

ReadNumber:

mov r4, SWA ; Read switches 0 to 7

mov r5, SWB ; Read switches 8 to 15

mov a, r4

orl a, r5

jz ReadNumber\_no\_number

lcall Wait50ms ; debounce

mov a, SWA

clr c

subb a, r4

jnz ReadNumber\_no\_number ; it was a bounce

mov a, SWB

clr c

subb a, r5

jnz ReadNumber\_no\_number ; it was a bounce

mov r7, #16 ; Loop counter

ReadNumber\_L0:

clr c

mov a, r4

rlc a

mov r4, a

mov a, r5

rlc a

mov r5, a

jc ReadNumber\_decode

djnz r7, ReadNumber\_L0

sjmp ReadNumber\_no\_number

ReadNumber\_decode:

dec r7

setb c

ReadNumber\_L1:

mov a, SWA

jnz ReadNumber\_L1

ReadNumber\_L2:

mov a, SWB

jnz ReadNumber\_L2

ret

ReadNumber\_no\_number:

clr c

ret

;mycode:

; mov SP, #7FH

; clr a

; mov LEDRA, a

; mov LEDRB, a

; mov LEDRC, a

; mov LEDG, a

; mov bcd+0, a

; mov bcd+1, a

; mov bcd+2, a

; lcall Display

;forever:

; lcall ReadNumber

; jnc forever

; lcall Shift\_Digits

; lcall Display

; ljmp forever

End

---------------------------------------------

$NOLIST

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; Serial Port Interface

;;

;; Nina Dacanay

;;

;; Function: 1.)Takes and Reads the temperature from

;; the K-Type thermocouple connected to

;; the MCP3004 ADC Converter.

;;

;; 2.)Displays the temperature to a Python

;; Strip Chart through the Serial Port.

;;

;; Inputs: 1.)Temperature

;; Outputs: 2.)Temperature Strip Chart

;;

;; Timers used:

;; 1.)Timer 2 (To configure the srial port and baud rate)

;;

;; Look-up tables used:

;; 1.)Serial\_Port\_My\_Lut\_ASCII

;;

;; Constants to be initialized:

;; 1.)T2LOAD EQU 65536-(FREQ/(32\*BAUD))

;; 2.)FREQ EQU 33333333

;; 3.)BAUD EQU 115200

;; Variables and Registers:

;; 1.)Accumulator

;; 2.)Temperature\_Measured

;; 3.)SBUF

;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

CSEG

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Serial\_Port\_My\_Lut\_ASCII

;

;Look-up table for the Python Strip Chart

;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Serial\_Port\_My\_Lut\_ASCII:

DB 030H, 031H, 032H, 033H, 034H ; 0 TO 4

DB 035H, 036H, 037H, 038H, 039H ; 4 TO 9

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Serial\_Port\_Init

;

;Initializes Timer 2 to be used for Serial Port Communication

;Initializes SCON to enable the use of the Serial Port

;

;@modifies Timer 2

; SCON

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Serial\_Port\_Init:

clr TR2 ; Disable timer 2

mov T2CON, #30H ; RCLK=1, TCLK=1

mov RCAP2H, #high(T2LOAD)

mov RCAP2L, #low(T2LOAD)

setb TR2 ; Enable timer 2

mov SCON, #52H

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Serial\_Port\_Putchar

;

;Send a character to register SBUF that is output from the serial port

;

;@modifies SBUF

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Serial\_Port\_Putchar:

JNB TI, Serial\_Port\_Putchar

CLR TI

MOV SBUF, a

RET

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Serial\_Port\_Send\_String

;

;Send a constant string that is terminated with an '\n' through the serial port

;

;I kept the code to display the temperature in the hex displays for testing purposes

;

;@modifies SBUF

; BCD+0, BCD+1, BCD+2

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Serial\_Port\_Send\_String:

mov x+0, Temperature\_Measured

mov x+1, Temperature\_Measured+1

lcall hex2bcd

mov dptr, #Serial\_Port\_My\_Lut\_ASCII

;jb Temperature\_Measured\_Sign, Serial\_Port\_Send\_String\_positive

;mov A, #'-'

;lcall Serial\_Port\_Putchar

Serial\_Port\_Send\_String\_positive:

; Display Digit 5

mov A, bcd+2

swap a

anl a, #0fh

movc A, @A+dptr

lcall Serial\_Port\_Putchar

; Display Digit 4

mov A, bcd+2

anl a, #0fh

movc A, @A+dptr

lcall Serial\_Port\_Putchar

; Display Digit 3

mov A, bcd+1

swap a

anl a, #0fh

movc A, @A+dptr

lcall Serial\_Port\_Putchar

; Display Digit 2

mov A, bcd+1

anl a, #0fh

movc A, @A+dptr

lcall Serial\_Port\_Putchar

; Display Digit 1

mov A, bcd+0

swap a

anl a, #0fh

movc A, @A+dptr

lcall Serial\_Port\_Putchar

; Display Digit 0

mov A, bcd+0

anl a, #0fh

movc A, @A+dptr

lcall Serial\_Port\_Putchar

mov A, #'\r'

lcall Serial\_Port\_Putchar

mov A, #'\n'

lcall Serial\_Port\_Putchar

ret

$LIST

---------------------------------------------

$NOLIST

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; SSR

;;

;; Derek Chan

;;

;; Function: Controls the SSR Relay box

;; Constants and Variables to be declared: n/a

;; Subroutines:

;; SSR\_Enable - turns on the SSR

;; SSR\_Disable - turns off the SSR

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

CSEG

SSR\_Init:

orl P1MOD, #00000001B

clr P1.0

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;SSR\_Enable

;;

;;Enables the SSR, thus turning on the oven

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

SSR\_Enable:

setb P1.0

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;SSR\_Disable

;;

;;Disables the SSR, thus turning off the oven

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

SSR\_Disable:

clr P1.0

ret

$LIST

---------------------------------------------

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;Test\_Buzzer

;;

;;Test program for the buzzer module.

;;

;;SW0 - Turns buzzer on/off

;;SW1 - Sets buzzer to beep or long-tone

;;

;;P1.5 is used for the buzzer

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

$modde2

BUZZER\_CLK EQU 33333333

BUZZER\_FREQ EQU 2000

BUZZER\_T0\_RELOAD EQU 65536-(BUZZER\_CLK/(12\*2\*BUZZER\_FREQ))

org 0000H

ljmp init

org 000BH

ljmp Buzzer\_ISR

DSEG at 30H

Buzzer\_Beep\_Count : ds 1

Buzzer\_Beep\_Num : ds 1

BSEG

Buzzer\_Beep\_Active : dbit 1

Buzzer\_Continuous\_Tone : dbit 1

CSEG

$include(Buzzer.asm)

init:

clr A

mov LEDRA, A

mov LEDRB, A

mov LEDRC, A

mov LEDG, A

lcall Buzzer\_Init

setb EA

forever:

mov A, SWA

jnb ACC.0, forever\_stopbeep

mov LEDG, #10000000B

lcall Buzzer\_Start\_Beep

sjmp forever\_checksw1

forever\_stopbeep:

mov LEDG, #00000000B

lcall Buzzer\_Stop\_Beep

forever\_checksw1:

jnb ACC.1, forever\_longtone

mov LEDRA, #00000001B

setb Buzzer\_Continuous\_Tone

sjmp forever\_checkKey3

forever\_longtone:

mov LEDRA, #00000000B

clr Buzzer\_Continuous\_Tone

forever\_checkKey3:

jb KEY.3, forever\_checkKey2

mov LEDRC, #00000010B

Buzzer\_Beep\_Multiple(4)

mov LEDRC, #0

jnb KEY.3, $

forever\_checkKey2:

jb KEY.2, forever\_checkKey1

mov LEDRC, #00000010B

Buzzer\_Beep\_Multiple(2)

mov LEDRC, #0

jnb KEY.2, $

forever\_checkKey1:

jb KEY.1, forever

mov LEDRC, #00000010B

Buzzer\_Beep\_Multiple(8)

mov LEDRC, #0

jnb KEY.1, $

sjmp forever

END

---------------------------------------------

$modde2

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Test file for Door

;

;P1.1 is for one arm of the door, 5V is for the other

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

org 0000h

lcall init

DSEG at 30H

BSEG

Door\_Open : dbit 1

CSEG

$include(Door.asm)

init:

lcall Door\_init

clr A

mov LEDRA, A

mov LEDRB, A

mov LEDRC, A

mov LEDG, A

forever:

lcall Door\_Check

jb Door\_Open, forever\_open

;door is closed

mov LEDG, #0

sjmp forever

forever\_open:

;door is open

mov LEDG, #1

sjmp forever

END

---------------------------------------------

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Test file for Motor

;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

$modde2

ORANGE EQU P1.3

YELLOW EQU P1.4

PINK EQU P1.6

BLUE EQU P1.7

TIMER\_XTAL EQU 33333333

TIMER\_FREQ EQU 100

TIMER1\_RELOAD EQU 65538-(TIMER\_XTAL/(12\*TIMER\_FREQ))

org 0000H

ljmp init

DSEG at 30H

Motor\_Phase: ds 1

CSEG

$include(Motor.asm)

Wait50ms:

;33.33MHz, 1 clk per cycle: 0.03us

mov R0, #2

L3: mov R1, #74

L2: mov R2, #250

L1: djnz R2, L1 ;3\*250\*0.03us=22.5us

djnz R1, L2 ;74\*22.5us=1.665ms

djnz R0, L3 ;1.665ms\*30=50ms

ret

init:

clr A

mov LEDRA, A

mov LEDRB, A

mov LEDRC, A

mov LEDG, A

mov Motor\_Phase, A

lcall Motor\_Init

forever:

lcall Motor\_ISR

mov LEDRA, Motor\_Phase

lcall Wait50ms

sjmp forever

---------------------------------------------

$modde2

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Test file for SSR

;

;Use SW0 to check enable and disable the SSR

;P1.0 is the output of the SSR Enable/Disable

;You'll use this output to activate/deactivate a BJT

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

org 0000h

lcall init

CSEG

$include(SSR.asm)

init:

lcall SSR\_init

clr A

mov LEDRA, A

mov LEDRB, A

mov LEDRC, A

mov LEDG, A

forever:

mov A, SWA

jb ACC.0, on

;turn ssr off

lcall SSR\_disable

mov LEDRA, #0

sjmp forever

on:

;turn ssr on

lcall SSR\_enable

mov LEDRA, #1

sjmp forever

END

---------------------------------------------

;Test For Status Display

$modde2

org 000H

ljmp main

STATE\_STANDBY EQU 0

STATE\_HEATING1 EQU 1

STATE\_SOAK EQU 2

STATE\_HEATING2 EQU 3

STATE\_REFLOW EQU 4

STATE\_COOLING EQU 5

STATE\_OPEN\_DOOR EQU 6

DSEG at 30h

;Math16/32 Variables

x : ds 2

y : ds 2

bcd : ds 3

;LCD\_DATA : ds 1

BSEG

;LCD\_ON : dbit 1

;LCD\_EN : dbit 1

;LCD\_MOD : dbit 1

;LCD\_RW : dbit 1

BSEG

;Math16

mf : dbit 1

CSEG

myLUT:

DB 0C0H, 0F9H, 0A4H, 0B0H, 099H ; 0 TO 4

DB 092H, 082H, 0F8H, 080H, 090H ; 4 TO 9

DB 088H, 083H, 0C6H, 0A1H, 086H, 08EH ; A to F

$include (LCD\_Display.asm)

main:

clr A

clr C

mov LEDG, A

mov LEDRA, A

mov LEDRB, A

mov LEDRC, A

lcall LCD\_init

forever:

mov state, STATE\_STANDBY

lcall Display\_Status

lcall waitHalfSec

sjmp forever

end

---------------------------------------------

$modde2

;

;Test file for Thermo\_LCD

;

org 0000h

lcall init

MISO EQU P0.0

MOSI EQU P0.1

SCLK EQU P0.2

CE\_ADC EQU P0.3

DSEG at 30H

x: ds 2

y: ds 2

bcd: ds 3

Temperature\_Measured: ds 2

Outside\_Temperature\_Measured: ds 1

BSEG

mf: dbit 1

ChannelSelect: dbit 1

Temperature\_Measured\_Sign: dbit 1

CSEG

$include(Thermo2.asm)

$include(math16.asm)

;Delay half a second

WaitHalfSec:

mov R2, #90

L3: mov R1, #250

L2: mov R0, #250

L1: djnz R0, L1

djnz R1, L2

djnz R2, L3

ret

; Look-up table for 7-seg displays

BCD\_LUT:

DB 0C0H, 0F9H, 0A4H, 0B0H, 099H ; 0 TO 4

DB 092H, 082H, 0F8H, 080H, 090H ; 4 TO 9

; Look-up table for the Python Temperature Strip Chart

Serial\_Port\_My\_Lut\_ASCII:

DB 030H, 031H, 032H, 033H, 034H ; 0 TO 4

DB 035H, 036H, 037H, 038H, 039H ; 4 TO 9

; Display the value on HEX display

Display:

mov dptr, #BCD\_LUT

; Display Digit 0

mov A, bcd+0

anl a, #0fh

movc A, @A+dptr

mov HEX0, A

; Display Digit 1

mov A, bcd+0

swap a

anl a, #0fh

movc A, @A+dptr

mov HEX1, A

; Display Digit 2

mov A, bcd+1

anl a, #0fh

movc A, @A+dptr

mov HEX2, A

; Display Digit 3

mov A, bcd+1

swap a

anl a, #0fh

movc A, @A+dptr

mov HEX3, A

; Display Digit 4

mov A, bcd+2

anl a, #0fh

movc A, @A+dptr

mov HEX4, A

; Display Digit 5

mov A, bcd+2

swap a

anl a, #0fh

movc A, @A+dptr

mov HEX5, A

ret

init:

lcall Thermocouple\_Input\_Init

clr A

mov LEDRA, A

mov LEDRB, A

mov LEDRC, A

mov LEDG, A

forever:

lcall Thermocouple\_ReadCH1

mov LEDRA, R7

mov LEDG, R6

mov x+0, Outside\_Temperature\_Measured+0

mov x+1, #0

lcall hex2bcd

lcall Display

sjmp forever

END

---------------------------------------------

$modde2

;

;Test file for T

;

org 0000h

lcall init

MISO EQU P0.0

MOSI EQU P0.1

SCLK EQU P0.2

CE\_ADC EQU P0.3

DSEG at 30H

x: ds 2

y: ds 2

bcd: ds 3

Temperature\_Measured: ds 2

Outside\_Temperature\_Measured: ds 1

BSEG

mf: dbit 1

ChannelSelect: dbit 1

Temperature\_Measured\_Sign: dbit 1

CSEG

$include(Thermo2.asm)

$include(math16.asm)

;Delay half a second

WaitHalfSec:

mov R2, #90

L3: mov R1, #250

L2: mov R0, #250

L1: djnz R0, L1

djnz R1, L2

djnz R2, L3

ret

; Look-up table for 7-seg displays

BCD\_LUT:

DB 0C0H, 0F9H, 0A4H, 0B0H, 099H ; 0 TO 4

DB 092H, 082H, 0F8H, 080H, 090H ; 4 TO 9

; Look-up table for the Python Temperature Strip Chart

Serial\_Port\_My\_Lut\_ASCII:

DB 030H, 031H, 032H, 033H, 034H ; 0 TO 4

DB 035H, 036H, 037H, 038H, 039H ; 4 TO 9

; Display the value on HEX display

Display:

mov dptr, #BCD\_LUT

; Display Digit 0

mov A, bcd+0

anl a, #0fh

movc A, @A+dptr

mov HEX0, A

; Display Digit 1

mov A, bcd+0

swap a

anl a, #0fh

movc A, @A+dptr

mov HEX1, A

; Display Digit 2

mov A, bcd+1

anl a, #0fh

movc A, @A+dptr

mov HEX2, A

; Display Digit 3

mov A, bcd+1

swap a

anl a, #0fh

movc A, @A+dptr

mov HEX3, A

; Display Digit 4

mov A, bcd+2

anl a, #0fh

movc A, @A+dptr

mov HEX4, A

; Display Digit 5

mov A, bcd+2

swap a

anl a, #0fh

movc A, @A+dptr

mov HEX5, A

ret

init:

lcall Thermocouple\_Input\_Init

clr A

mov LEDRA, A

mov LEDRB, A

mov LEDRC, A

mov LEDG, A

forever:

lcall Thermocouple\_ReadCH0

mov LEDRA, R7

mov LEDG, R6

mov x+0, Temperature\_Measured+0

mov x+1, Temperature\_Measured+1

lcall hex2bcd

lcall Display

lcall waitHalfSec

lcall waitHalfSec

sjmp forever

END

---------------------------------------------

$modde2

;

;Test file for T

;

org 0000h

lcall init

MISO EQU P0.0

MOSI EQU P0.1

SCLK EQU P0.2

CE\_ADC EQU P0.3

THERMO\_TEMP\_ADJ EQU 10

T2LOAD EQU 65536-(FREQ/(32\*BAUD))

FREQ EQU 33333333

BAUD EQU 115200

DSEG at 30H

x: ds 2

y: ds 2

bcd: ds 3

Temperature\_Measured: ds 2

Outside\_Temperature\_Measured: ds 2

BSEG

mf: dbit 1

ChannelSelect: dbit 1

Temperature\_Measured\_Sign: dbit 1

CSEG

$include(Thermo2.asm)

$include(math16.asm)

$include(Serial\_Port.asm)

;Delay half a second

WaitHalfSec:

mov R2, #90

L3: mov R1, #250

L2: mov R0, #250

L1: djnz R0, L1

djnz R1, L2

djnz R2, L3

ret

; Look-up table for 7-seg displays

BCD\_LUT:

DB 0C0H, 0F9H, 0A4H, 0B0H, 099H ; 0 TO 4

DB 092H, 082H, 0F8H, 080H, 090H ; 4 TO 9

; Display the value on HEX display

Display:

mov dptr, #BCD\_LUT

; Display Digit 0

mov A, bcd+0

anl a, #0fh

movc A, @A+dptr

mov HEX0, A

; Display Digit 1

mov A, bcd+0

swap a

anl a, #0fh

movc A, @A+dptr

mov HEX1, A

; Display Digit 2

mov A, bcd+1

anl a, #0fh

movc A, @A+dptr

mov HEX2, A

; Display Digit 3

mov A, bcd+1

swap a

anl a, #0fh

movc A, @A+dptr

mov HEX3, A

; Display Digit 4

mov A, bcd+2

anl a, #0fh

movc A, @A+dptr

mov HEX4, A

; Display Digit 5

mov A, bcd+2

swap a

anl a, #0fh

movc A, @A+dptr

mov HEX5, A

ret

init:

lcall Thermocouple\_Input\_Init

lcall Serial\_Port\_Init

clr A

mov LEDRA, A

mov LEDRB, A

mov LEDRC, A

mov LEDG, A

forever:

;lcall Thermocouple\_ReadCH0

;lcall Thermocouple\_ReadCH1

;mov x+0, Temperature\_Measured+0

;mov x+1, Temperature\_Measured+1

;mov y+0, Outside\_Temperature\_Measured+0

;mov y+1, #0

lcall Thermocouple\_Update

mov x+0, Temperature\_Measured+0

mov x+1, Temperature\_Measured+1

;clr mf

;lcall add16

sjmp forever\_display

forever\_display:

lcall hex2bcd

lcall Display

lcall Serial\_Port\_Send\_String

lcall waithalfsec

lcall waithalfsec

sjmp forever

END

---------------------------------------------

;;;;;;;;;;;;;;

;;Timer

;;

;;Nina Dacanay, Derek Chan

;;

;;Keeps track of elapsed time for the system

;;

;;;;;;;;;;;;;;;

$MODDE2

XTAL EQU 33333333

FREQ EQU 100

TIMER1\_RELOAD EQU 65538-(XTAL/(12\*20\*FREQ))

org 0000H

ljmp My\_Program

org 00B3H

ljmp ISR\_Timer

DSEG at 30H

Timer\_count10ms: ds 1

Timer\_Total\_Time\_Seconds: ds 1 ;incrementing every second

Timer\_Total\_Time\_Minutes: ds 1 ;incrementing every minute

Timer\_Elapsed\_Time: ds 2 ;incrementing every second

x : ds 2

bcd : ds 3

y : ds 2

BSEG

mf : dbit 1

CSEG

$include(math16.asm)

$include(Timer.asm)

My\_Program:

mov SP, #7FH

mov LEDRA,#0

mov LEDRB,#0

mov LEDRC,#0

mov LEDG,#0

mov Timer\_Total\_Time\_Seconds, #000H

mov Timer\_Total\_Time\_Minutes, #0H

mov Timer\_Elapsed\_Time, #000H

mov Timer\_Elapsed\_Time+1, #0H

lcall Init\_Timer

setb EA ; Enable all interrupts

Timer\_Forever:

lcall Timer\_Display

mov LEDG, Timer\_Elapsed\_Time

jb SWA.1, Timer\_Reset\_TimeElapsed

jb SWA.2, Timer\_Reset\_Everything

sjmp Timer\_Forever

Timer\_Reset\_TimeElapsed:

lcall Timer\_Reset\_Elapsed\_Time

sjmp Timer\_Forever

Timer\_Reset\_Everything:

lcall Timer\_Reset

sjmp Timer\_Forever

END

---------------------------------------------

$modde2

org 000H

ljmp main

STATE\_STANDBY EQU 0

STATE\_HEATING1 EQU 1

STATE\_SOAK EQU 2

STATE\_HEATING2 EQU 3

STATE\_REFLOW EQU 4

STATE\_COOLING EQU 5

STATE\_OPEN\_DOOR EQU 6

DSEG at 30h

soak\_temperature : ds 2

soak\_time : ds 2

reflow\_temperature: ds 2

reflow\_time : ds 2

state : ds 2

;Math16/32 Variables

x : ds 2

y : ds 2

bcd : ds 3

;LCD\_DATA : ds 1

BSEG

;LCD\_ON : dbit 1

;LCD\_EN : dbit 1

;LCD\_MOD : dbit 1

;LCD\_RW : dbit 1

BSEG

;Math16

mf : dbit 1

CSEG

myLUT:

DB 0C0H, 0F9H, 0A4H, 0B0H, 099H ; 0 TO 4

DB 092H, 082H, 0F8H, 080H, 090H ; 4 TO 9

DB 088H, 083H, 0C6H, 0A1H, 086H, 08EH ; A to F

$include (LCD\_Display.asm)

$include (Read\_sw5.asm)

$include (User\_Interface.asm)

$include (math16.asm)

main:

clr A

clr C

mov LEDG, A

mov LEDRA, A

mov LEDRB, A

mov LEDRC, A

lcall LCD\_init

forever:

lcall UI\_Set\_Up\_Parameters

sjmp forever

end

---------------------------------------------

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Spyder-Python Strip Chart Code

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.animation as animation

import sys, time, math

import time

import serial

xsize=100

# configure the serial port

ser = serial.Serial(

port='COM3',

baudrate=115200,

parity=serial.PARITY\_NONE,

stopbits=serial.STOPBITS\_TWO,

bytesize=serial.EIGHTBITS

)

ser.isOpen()

def data\_gen():

t = data\_gen.t

while True:

t+=1

strin=ser.readline()

val=strin

yield t, val

def run(data):

# update the data

t,y = data

if t>-1:

xdata.append(t)

ydata.append(y)

if t>xsize: # Scroll to the left.

ax.set\_xlim(t-xsize, t)

line.set\_data(xdata, ydata)

return line,

def on\_close\_figure(event):

sys.exit(0)

data\_gen.t = -1

fig = plt.figure()

fig.canvas.mpl\_connect('close\_event', on\_close\_figure)

ax = fig.add\_subplot(111)

line, = ax.plot([], [], lw=2)

ax.set\_ylim(0, 250)

ax.set\_xlim(0, xsize)

ax.grid()

xdata, ydata = [], []

# Important: Although blit=True makes graphing faster, we need blit=False to prevent

# spurious lines to appear when resizing the stripchart.

ani = animation.FuncAnimation(fig, run, data\_gen, blit=False, interval=50, repeat=False)

plt.show()

---------------------------------------------

$NOLIST

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;Thermocouple Input

;;

;;Kyujin Park, Nina Dacanay, Glyn Han, Derek Chan

;;

;;Function: Loads the variable Temperature\_Measured with the correct

;; temperature in the oven, calibrated with the cold junction

;; outside of the box.

;;

;;Constants:

;; MISO EQU P0.0

;; MOSI EQU P0.1

;; SCLK EQU P0.2

;; CE\_ADC EQU P0.3

;;

;;Variables:

;; Temperatre\_Measured: ds 2

;; Outside\_Temperature\_Measured: ds 2

;;

;;Functions:

;; Thermocouple\_Update

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Thermocouple\_Input\_Init

;Initializes Port 0 to be used for SPI communication

;

;@modifies P0MOD, CE\_ADC, SCLK

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Thermocouple\_Input\_Init:

orl P0MOD, #00001000b ; make CE\_ADC\* output

setb CE\_ADC ; disables ADC initially - we aren't using it yet

lcall Thermocouple\_Input\_INIT\_SPI

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Thermocouple\_Update

;

;Updates the temperature variables.

;Reads from Thermocouple and LM335, adds the temperatures,

;and stores the resulting temperature in Temperature\_Measured

;

;@modifies Temperature\_Measured+0 - "True" temperature LSBs

; Temperature\_Measured+1 - MSBs

; Outside\_Temperature\_Measured - LM335 variable

; R7, R6, R1, R0, ACC, PSW, CE\_ADC

; x

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Thermocouple\_Update:

lcall Thermocouple\_ReadCH0

lcall Thermocouple\_ReadCH1

mov A, Temperature\_Measured+0

add A, Outside\_Temperature\_Measured

clr c

subb A, #THERMO\_TEMP\_ADJ

mov Temperature\_Measured+0, A

mov A, Temperature\_Measured+1

addc A, #0

mov Temperature\_Measured+1, A

ret

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

;Helper subroutines - not used in the main program

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

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Oven

;

;@returns Temperature\_Measured

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Thermocouple\_ReadCH0:

mov B, #0

lcall Thermocouple\_Input\_Read\_ADC

lcall Thermocouple\_Input\_Convert\_Binary\_To\_Temperature

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Outside

;

;@returns Temperature\_Measured\_Outside

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Thermocouple\_ReadCH1:

mov B, #1

lcall Thermocouple\_Input\_Read\_ADC

lcall Thermocouple\_Input\_Convert\_Binary\_To\_Outside\_Temperature

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Thermocouple\_Input\_Read\_ADC

;

;Reads the 10-bit value from the ADC and stores it in

;the registers. Automatically follows the reading with a

;conversion of the temperature to be stored

;

;@returns R7 - Most significant bits (XXXXXX98 binary)

; R6 - Least significant bits (76543210 binary)

; Temperature\_Measured+0 - LSBs

; Temperature\_Measured+1 - MSBs

;@modifies A, R0, R1, R6, R7, PSW

; CE\_ADC

; x, Temperature\_Measured

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Thermocouple\_Input\_Read\_ADC:

clr CE\_ADC

mov R0, #00000001B ; Start bit:1

lcall Thermocouple\_Input\_Do\_SPI

Thermocouple\_Input\_Read\_ADC\_Subroutine:

mov a, b

swap a

anl a, #0F0H

setb acc.7 ; Single mode (bit 7).

mov R0, a ; Select channel

lcall Thermocouple\_Input\_Do\_SPI

mov a, R1 ; R1 contains bits 8 and 9

anl a, #03H

mov R7, a

mov R0, #55H ; Send them trash. Ye boi.

lcall Thermocouple\_Input\_Do\_SPI

mov a, R1 ; R1 contains bits 0 to 7

mov R6, a

setb CE\_ADC

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Thermocouple\_Input\_BinaryToTemperature

;Lookup table used for the conversion of binary values

;from the ADC into temperature values. Note that 2-bytes

;are needed to store the values

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Thermocouple\_Input\_BinaryToTemperature:

DW 000, 000, 001, 001, 001, 001, 002, 002, 002, 003, 003, 003, 003, 004, 004, 004, 004, 005, 005, 005, 006, 006, 006, 006, 007, 007, 007, 008, 008, 008, 008, 009, 009, 009, 010, 010, 010, 010, 011, 011, 011, 011, 012, 012, 012, 013, 013, 013, 013, 014, 014, 014, 015, 015, 015, 015, 016, 016, 016, 017, 017, 017, 017, 018, 018, 018, 018, 019, 019, 019, 020, 020, 020, 020, 021, 021, 021, 022, 022, 022, 022, 023, 023, 023, 024, 024, 024, 024, 025, 025, 025, 025, 026, 026, 026, 027, 027, 027, 027, 028, 028, 028, 029, 029, 029, 029, 030, 030, 030, 031, 031, 031, 031, 032, 032, 032, 032, 033, 033, 033, 034, 034, 034, 034, 035, 035, 035, 036, 036, 036, 036, 037, 037, 037, 038, 038, 038, 038, 039, 039, 039, 040, 040, 040, 040, 041, 041, 041, 041, 042, 042, 042, 043, 043, 043, 043, 044, 044, 044, 045, 045, 045, 045, 046, 046, 046, 047, 047, 047, 047, 048, 048, 048, 048, 049, 049, 049, 050, 050, 050, 050, 051, 051, 051, 052, 052, 052, 052, 053, 053, 053, 054, 054, 054, 054, 055, 055, 055, 055, 056

DW 056, 056, 057, 057, 057, 057, 058, 058, 058, 059, 059, 059, 059, 060, 060, 060, 061, 061, 061, 061, 062, 062, 062, 062, 063, 063, 063, 064, 064, 064, 064, 065, 065, 065, 066, 066, 066, 066, 067, 067, 067, 068, 068, 068, 068, 069, 069, 069, 069, 070, 070, 070, 071, 071, 071, 071, 072, 072, 072, 073, 073, 073, 073, 074, 074, 074, 075, 075, 075, 075, 076, 076, 076, 076, 077, 077, 077, 078, 078, 078, 078, 079, 079, 079, 080, 080, 080, 080, 081, 081, 081, 082, 082, 082, 082, 083, 083, 083, 083, 084, 084, 084, 085, 085, 085, 085, 086, 086, 086, 087, 087, 087, 087, 088, 088, 088, 089, 089, 089, 089, 090, 090, 090, 090, 091, 091, 091, 092, 092, 092, 092, 093, 093, 093, 094, 094, 094, 094, 095, 095, 095, 096, 096, 096, 096, 097, 097, 097, 097, 098, 098, 098, 099, 099, 099, 099, 100, 100, 100, 101, 101, 101, 101, 102, 102, 102, 103, 103, 103, 103, 104, 104, 104, 104, 105, 105, 105, 106, 106, 106, 106, 107, 107, 107, 108, 108, 108, 108, 109, 109, 109, 110, 110, 110, 110, 111, 111, 111, 111, 112

DW 112, 112, 113, 113, 113, 113, 114, 114, 114, 115, 115, 115, 115, 116, 116, 116, 117, 117, 117, 117, 118, 118, 118, 119, 119, 119, 119, 120, 120, 120, 120, 121, 121, 121, 122, 122, 122, 122, 123, 123, 123, 124, 124, 124, 124, 125, 125, 125, 126, 126, 126, 126, 127, 127, 127, 127, 128, 128, 128, 129, 129, 129, 129, 130, 130, 130, 131, 131, 131, 131, 132, 132, 132, 133, 133, 133, 133, 134, 134, 134, 134, 135, 135, 135, 136, 136, 136, 136, 137, 137, 137, 138, 138, 138, 138, 139, 139, 139, 140, 140, 140, 140, 141, 141, 141, 141, 142, 142, 142, 143, 143, 143, 143, 144, 144, 144, 145, 145, 145, 145, 146, 146, 146, 147, 147, 147, 147, 148, 148, 148, 148, 149, 149, 149, 150, 150, 150, 150, 151, 151, 151, 152, 152, 152, 152, 153, 153, 153, 154, 154, 154, 154, 155, 155, 155, 155, 156, 156, 156, 157, 157, 157, 157, 158, 158, 158, 159, 159, 159, 159, 160, 160, 160, 161, 161, 161, 161, 162, 162, 162, 162, 163, 163, 163, 164, 164, 164, 164, 165, 165, 165, 166, 166, 166, 166, 167, 167, 167, 168, 168

DW 168, 168, 169, 169, 169, 169, 170, 170, 170, 171, 171, 171, 171, 172, 172, 172, 173, 173, 173, 173, 174, 174, 174, 175, 175, 175, 175, 176, 176, 176, 176, 177, 177, 177, 178, 178, 178, 178, 179, 179, 179, 180, 180, 180, 180, 181, 181, 181, 182, 182, 182, 182, 183, 183, 183, 183, 184, 184, 184, 185, 185, 185, 185, 186, 186, 186, 187, 187, 187, 187, 188, 188, 188, 189, 189, 189, 189, 190, 190, 190, 190, 191, 191, 191, 192, 192, 192, 192, 193, 193, 193, 194, 194, 194, 194, 195, 195, 195, 196, 196, 196, 196, 197, 197, 197, 198, 198, 198, 198, 199, 199, 199, 199, 200, 200, 200, 201, 201, 201, 201, 202, 202, 202, 203, 203, 203, 203, 204, 204, 204, 205, 205, 205, 205, 206, 206, 206, 206, 207, 207, 207, 208, 208, 208, 208, 209, 209, 209, 210, 210, 210, 210, 211, 211, 211, 212, 212, 212, 212, 213, 213, 213, 213, 214, 214, 214, 215, 215, 215, 215, 216, 216, 216, 217, 217, 217, 217, 218, 218, 218, 219, 219, 219, 219, 220, 220, 220, 220, 221, 221, 221, 222, 222, 222, 222, 223, 223, 223, 224, 224

DW 224, 224, 225, 225, 225, 226, 226, 226, 226, 227, 227, 227, 227, 228, 228, 228, 229, 229, 229, 229, 230, 230, 230, 231, 231, 231, 231, 232, 232, 232, 233, 233, 233, 233, 234, 234, 234, 234, 235, 235, 235, 236, 236, 236, 236, 237, 237, 237, 238, 238, 238, 238, 239, 239, 239, 240, 240, 240, 240, 241, 241, 241, 241, 242, 242, 242, 243, 243, 243, 243, 244, 244, 244, 245, 245, 245, 245, 246, 246, 246, 247, 247, 247, 247, 248, 248, 248, 248, 249, 249, 249, 250, 250, 250, 250, 251, 251, 251, 252, 252, 252, 252, 253, 253, 253, 254, 254, 254, 254, 255, 255, 255, 255, 256, 256, 256, 257, 257, 257, 257, 258, 258, 258, 259, 259, 259, 259, 260, 260, 260, 261, 261, 261, 261, 262, 262, 262, 262, 263, 263, 263, 264, 264, 264, 264, 265, 265, 265, 266, 266, 266, 266, 267, 267, 267, 268, 268, 268, 268, 269, 269, 269, 269, 270, 270, 270, 271, 271, 271, 271, 272, 272, 272, 273, 273, 273, 273, 274, 274, 274, 275, 275, 275, 275, 276, 276, 276, 277, 277, 277, 277, 278, 278, 278, 278, 279, 279, 279, 280, 280

DW 280, 280, 281, 281, 281, 282, 282, 282, 282, 283, 283, 283, 284, 284, 284, 284, 285, 285, 285, 285, 286, 286, 286, 287

;;;;;;;;;;;;;;;;;;;;;

;Helper delay loop

;Uses R3

;;;;;;;;;;;;;;;;;;;;;

Thermocouple\_Input\_Delay:

mov R3, #20

Thermocouple\_Input\_Delay\_loop:

djnz R3, Thermocouple\_Input\_Delay\_loop

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Thermocouple\_Input\_INIT\_SPI

;

;Helper subroutine to initialize SPI

;

;@requires: SCLK as P0.2

; MISO as P0.0

; MOSI as P0.1

; CE\_ADC\* as P0.3

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Thermocouple\_Input\_INIT\_SPI:

orl P0MOD, #00000110b ; Set SCLK, MOSI as outputs

anl P0MOD, #11111110b ; Set MISO as input

clr SCLK ; For mode (0,0) SCLK is zero

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Thermocouple\_Input\_Do\_SPI

;

;Helper subroutine

;Runs the SPI routine to read/write

;

;@param R0 - the byte to write out

;@returns R1 - the byte read by the routine

;@modifies R0, R1, R2,

; MISO, MOSI, SCLK

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Thermocouple\_Input\_Do\_SPI:

push acc

push psw

mov R1, #0 ; Received byte stored in R1

mov R2, #8 ; Loop counter (8-bits)

Thermocouple\_Input\_Do\_SPI\_Loop:

mov a, R0 ; Byte to write is in R0

rlc a ; Carry flag has bit to write

mov R0, a

mov MOSI, c

setb SCLK ; Transmit

mov c, MISO ; Read received bit

mov a, R1 ; Save received bit in R1

rlc a

mov R1, a

clr SCLK

djnz R2, Thermocouple\_Input\_Do\_SPI\_Loop

pop psw

pop acc

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Thermocouple\_Input\_Convert\_Binary\_To\_Temperature

;

;Helper subroutine

;

;Converts the binary voltage reading from the ADC into

;a 2-byte temperature value in binary

;

;@param R7 - MSBs of the binary voltage reading

; R6 - LSBs of the binary voltage reading

;@returns Temperature\_Measured - 2-byte temperature value

; T\_M+0 - Least significant byte

; T\_M+1 - Most significant byte

;@modifies A, DPTR, PSW, x, x+1

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Thermocouple\_Input\_Convert\_Binary\_To\_Temperature:

;dptr = LUT + (2 \* binary value)

mov dptr, #Thermocouple\_Input\_BinaryToTemperature

clr C

mov A, dpl

add A, R6

mov dpl, A

mov A, dph

addc A, R7

mov dph, A

clr C

mov A, dpl

add A, R6

mov dpl, A

mov A, dph

addc A, R7

mov dph, A

clr A

movc A, @A+dptr

mov Temperature\_Measured+1, A ;high part is stored first

inc dptr

clr A

movc A, @A+dptr

mov Temperature\_Measured+0, A ;then low part

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Thermocouple\_Input\_Binary\_To\_Outside\_Temperature

;

;Lookup table used for the conversion of binary values

;from the ADC into temperature values. Note that 2-bytes

;are needed to store the values

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Thermocouple\_LM335\_LUT:

DB 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 254, 253, 253, 253, 252, 251, 251, 251, 250, 249, 249, 249, 248, 247, 247, 247, 246, 245, 245, 245, 244, 243, 243, 243, 243, 242, 241, 241, 241, 240, 239, 239, 239, 238, 237, 237, 237, 236, 235, 235, 235, 234, 233, 233, 233, 232, 231, 231, 231, 230, 229, 229, 229, 228, 227, 227, 227, 226, 226

DB 225, 225, 224, 224, 223, 223, 222, 222, 221, 221, 220, 220, 219, 219, 218, 218, 217, 217, 216, 216, 215, 215, 214, 214, 213, 213, 212, 212, 211, 211, 211, 210, 210, 209, 209, 208, 208, 207, 207, 206, 206, 205, 205, 204, 204, 203, 203, 202, 202, 201, 201, 200, 200, 199, 199, 198, 198, 197, 197, 196, 196, 196, 195, 194, 194, 194, 193, 192, 192, 192, 191, 190, 190, 190, 189, 188, 188, 188, 187, 186, 186, 186, 185, 184, 184, 184, 183, 182, 182, 182, 181, 180, 180, 180, 180, 179, 178, 178, 178, 177

DB 176, 176, 176, 175, 174, 174, 174, 173, 172, 172, 172, 171, 170, 170, 170, 169, 168, 168, 168, 167, 166, 166, 166, 165, 164, 164, 164, 164, 162, 162, 162, 162, 160, 160, 160, 160, 158, 158, 158, 158, 156, 156, 156, 156, 154, 154, 154, 154, 152, 152, 152, 152, 150, 150, 150, 150, 148, 148, 148, 148, 148, 146, 146, 146, 146, 144, 144, 144, 144, 142, 142, 142, 142, 140, 140, 140, 140, 138, 138, 138, 138, 136, 136, 136, 136, 134, 134, 134, 134, 133, 132, 132, 132, 131, 130, 130, 130, 129, 128, 128

DB 128, 127, 126, 126, 126, 125, 124, 124, 124, 123, 122, 122, 122, 121, 120, 120, 120, 119, 118, 118, 118, 118, 117, 116, 116, 116, 115, 114, 114, 114, 113, 112, 112, 112, 111, 110, 110, 110, 109, 108, 108, 108, 107, 106, 106, 106, 105, 104, 104, 104, 103, 102, 102, 102, 101, 101, 100, 100, 99, 99, 98, 98, 97, 97, 96, 96, 95, 95, 94, 94, 93, 93, 92, 92, 91, 91, 90, 90, 89, 89, 88, 88, 87, 87, 86, 86, 86, 85, 85, 84, 84, 83, 83, 82, 82, 81, 81, 80, 80, 79

DB 79, 78, 78, 77, 77, 76, 76, 75, 75, 74, 74, 73, 73, 72, 72, 71, 71, 71, 70, 69, 69, 69, 68, 67, 67, 67, 66, 65, 65, 65, 64, 63, 63, 63, 62, 61, 61, 61, 60, 59, 59, 59, 58, 57, 57, 57, 56, 55, 55, 55, 55, 54, 53, 53, 53, 52, 51, 51, 51, 50, 49, 49, 49, 48, 47, 47, 47, 46, 45, 45, 45, 44, 43, 43, 43, 42, 41, 41, 41, 40, 39, 39, 39, 39, 37, 37, 37, 37, 35, 35, 35, 35, 33, 33, 33, 33, 31, 31, 31, 31

DB 29, 29, 29, 29, 27, 27, 27, 27, 25, 25, 25, 25, 23, 23, 23, 23, 23, 21, 21, 21, 21, 19, 19, 19, 19, 17, 17, 17, 17, 15, 15, 15, 15, 13, 13, 13, 13, 11, 11, 11, 11, 9, 9, 9, 9, 8, 7, 7, 7, 6, 5, 5, 5, 4, 3, 3, 3, 2, 1, 1, 1, 0, 1, 1, 1, 2, 3, 3, 3, 4, 5, 5, 5, 6, 7, 7, 7, 7, 8, 9, 9, 9, 10, 11, 11, 11, 12, 13, 13, 13, 14, 15, 15, 15, 16, 17, 17, 17, 18, 19

DB 19, 19, 20, 21, 21, 21, 22, 23, 23, 23, 24, 24, 25, 25, 26, 26, 27, 27, 28, 28, 29, 29, 30, 30, 31, 31, 32, 32, 33, 33, 34, 34, 35, 35, 36, 36, 37, 37, 38, 38, 39, 39, 39, 40, 40, 41, 41, 42, 42, 43, 43, 44, 44, 45, 45, 46, 46, 47, 47, 48, 48, 49, 49, 50, 50, 51, 51, 52, 52, 53, 53, 54, 54, 54, 55, 56, 56, 56, 57, 58, 58, 58, 59, 60, 60, 60, 61, 62, 62, 62, 63, 64, 64, 64, 65, 66, 66, 66, 67, 68

DB 68, 68, 69, 70, 70, 70, 70, 71, 72, 72, 72, 73, 74, 74, 74, 75, 76, 76, 76, 77, 78, 78, 78, 79, 80, 80, 80, 81, 82, 82, 82, 83, 84, 84, 84, 85, 86, 86, 86, 86, 88, 88, 88, 88, 90, 90, 90, 90, 92, 92, 92, 92, 94, 94, 94, 94, 96, 96, 96, 96, 98, 98, 98, 98, 100, 100, 100, 100, 102, 102, 102, 102, 102, 104, 104, 104, 104, 106, 106, 106, 106, 108, 108, 108, 108, 110, 110, 110, 110, 112, 112, 112, 112, 114, 114, 114, 114, 116, 116, 116

DB 116, 117, 118, 118, 118, 119, 120, 120, 120, 121, 122, 122, 122, 123, 124, 124, 124, 125, 126, 126, 126, 127, 128, 128, 128, 129, 130, 130, 130, 131, 132, 132, 132, 132, 133, 134, 134, 134, 135, 136, 136, 136, 137, 138, 138, 138, 139, 140, 140, 140, 141, 142, 142, 142, 143, 144, 144, 144, 145, 146, 146, 146, 147, 148, 148, 148, 149, 149, 150, 150, 151, 151, 152, 152, 153, 153, 154, 154, 155, 155, 156, 156, 157, 157, 158, 158, 159, 159, 160, 160, 161, 161, 162, 162, 163, 163, 164, 164, 164, 165

DB 165, 166, 166, 167, 167, 168, 168, 169, 169, 170, 170, 171, 171, 172, 172, 173, 173, 174, 174, 175, 175, 176, 176, 177, 177, 178, 178, 179, 179, 179, 180, 181, 181, 181, 182, 183, 183, 183, 184, 185, 185, 185, 186, 187, 187, 187, 188, 189, 189, 189, 190, 191, 191, 191, 192, 193, 193, 193, 194, 195, 195, 195, 195, 196, 197, 197, 197, 198, 199, 199, 199, 200, 201, 201, 201, 202, 203, 203, 203, 204, 205, 205, 205, 206, 207, 207, 207, 208, 209, 209, 209, 210, 211, 211, 211, 211, 213, 213, 213, 213

DB 215, 215, 215, 215, 217, 217, 217, 217, 219, 219, 219, 219, 221, 221, 221, 221, 223, 223, 223, 223, 225, 225, 225, 225

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Thermocouple\_Input\_Convert\_Binary\_To\_Outside\_Temperature

;

;Helper subroutine

;

;Converts the binary voltage reading from the ADC into

;a 2-byte temperature value in binary

;

;@param R7 - MSBs of the binary voltage reading

; R6 - LSBs of the binary voltage reading

;@returns Temperature\_Measured - 2-byte temperature value

; T\_M+0 - Least significant byte

; T\_M+1 - Most significant byte

; Temperature\_Measured\_Sign: db 1

;@modifies A, DPTR, PSW, x, x+1

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Thermocouple\_Input\_Convert\_Binary\_To\_Outside\_Temperature:

mov x+0, R6

mov x+1, R7

mov y+1, #high(560)

mov y+0, #low(560)

lcall x\_gt\_y

mov Temperature\_Measured\_Sign, mf

;dptr = LUT + (2 \* binary value)

mov dptr, #Thermocouple\_LM335\_LUT

clr C

mov A, dpl

add A, R6

mov dpl, A

mov A, dph

addc A, R7

mov dph, A

clr A

movc A, @A+dptr

mov Outside\_Temperature\_Measured+0, A

ret

$LIST

---------------------------------------------

$NOLIST

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

; Thermocouple/Serial Input Interface

; (THIS IS A DRAFT, NOT THE FINAL FILE)

;

; Kyujin Park, Nina Dacanay, Glyn Han

;

; Function: 1) Reads the voltage from the K-Type Thermocouple connected to

; the MCP 3004 ADC Converter

; 2) The temperature will be calculated

; => (ADC\*62/256)+(ADC\*63/256)-273 (include math16.asm, maybe math32.asm)

;

; Possible Labels need to be declared

; My\_Lut, Display, Init\_Serial\_Port, Wait\_Half\_Sec,

; Init\_SPI, Do\_SPI\_G, Do\_SPI\_G\_Loop, Delay, Delay\_Loop,

; Read\_ADC\_Channel, MISO, MOSI, SCLK, CE\_ADC, CE\_EE, CE\_RTC

;

; Most Difficult Part : Displaying Temperature on LED Screen

; Conversion Calculation

;

; Functions : Loop\_Up\_Table; Display\_Seven\_Seg(from 0-9), Wait\_Half\_Sec, Timer\_Two

; Timer\_Two, My\_Program

;

; Equation(voltage to temp) : The equation above 0 °C is of the form

; E = sum(i=0 to n) c\_i t^i + a0 exp(a1 (t - a2)^2)

;

; Extra Files need for calculation: math16.asm, math32.asm (depending on the overflow of bit operation)

;

; Registers/Variables :

; thermocouple\_temp: db 2

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

;Thermocouple\_Input\_Init:

;ret

; Look-up table for 7-seg displays

myLUT:

DB 0C0H, 0F9H, 0A4H, 0B0H, 099H ; 0 TO 4

DB 092H, 082H, 0F8H, 080H, 090H ; 4 TO 9

; Display the value on HEX display

Display:

mov dptr, #myLUT

; Display Digit 0

mov A, bcd+0

anl a, #0fh

movc A, @A+dptr

mov HEX0, A

; Display Digit 1

mov A, bcd+0

swap a

anl a, #0fh

movc A, @A+dptr

mov HEX1, A

ret

;Initializes the serial port through timer 2

InitSerialPort:

clr TR2 ; Disable timer 2

mov T2CON, #30H ; RCLK=1, TCLK=1

mov RCAP2H, #high(T2LOAD)

mov RCAP2L, #low(T2LOAD)

setb TR2 ; Enable timer 2

mov SCON, #52H

ret

;Stores the value taken from the serial port into the serial buffer (SBUF)

Thermocouple\_Putchar:

JNB TI, Thermocouple\_Putchar

CLR TI

MOV SBUF, a

RET

;Converts the value from numerical to string

;Allows Python to read and graph a strip chart of the

;Temperature inside the oven

Thermocouple\_Send\_Number:

mov dptr, #String\_Number

; Display Digit 0

mov A, bcd+0

swap a

anl a, #0fh

movc A, @A+dptr

lcall Thermocouple\_Putchar

mov A, bcd+0

anl a, #0fh

movc A, @A+dptr

lcall Thermocouple\_Putchar

; Display Digit 1

mov A, #0AH

movc A, @A+dptr

lcall Thermocouple\_Putchar

ret

;look up chart table for Thermocouple\_Send\_Number

String\_Number:

DB '0','1','2','3','4','5','6','7','8','9','\n'

;Delay half a second

WaitHalfSec:

mov R2, #90

L3: mov R1, #250

L2: mov R0, #250

L1: djnz R0, L1

djnz R1, L2

djnz R2, L3

ret

;set outputs and inputs for the serial

INIT\_SPI:

orl P0MOD, #00000110b ; Set SCLK, MOSI as outputs

anl P0MOD, #11111110b ; Set MISO as input

clr SCLK ; For mode (0,0) SCLK is zero

ret

;

DO\_SPI\_G:

push acc

mov R1, #0 ; Received byte stored in R1

mov R2, #8 ; Loop counter (8-bits)

DO\_SPI\_G\_LOOP:

mov a, R0 ; Byte to write is in R0

rlc a ; Carry flag has bit to write

mov R0, a

mov MOSI, c

setb SCLK ; Transmit

mov c, MISO ; Read received bit

mov a, R1 ; Save received bit in R1

rlc a

mov R1, a

clr SCLK

djnz R2, DO\_SPI\_G\_LOOP

pop acc

ret

Delay:

mov R3, #20

Delay\_loop:

djnz R3, Delay\_loop

ret

; Channel to read passed in register b

Read\_ADC\_Channel:

clr CE\_ADC

mov R0, #00000001B ; Start bit:1

lcall DO\_SPI\_G

mov a, b

swap a

anl a, #0F0H

setb acc.7 ; Single mode (bit 7).

mov R0, a ; Select channel

lcall DO\_SPI\_G

mov a, R1 ; R1 contains bits 8 and 9

anl a, #03H

mov R7, a

mov R0, #55H ; It doesn't matter what we transmit...

lcall DO\_SPI\_G

mov a, R1 ; R1 contains bits 0 to 7

mov R6, a

setb CE\_ADC

ret

Thermocouple\_Program:

mov sp, #07FH

clr a

mov LEDG, a

mov LEDRA, a

mov LEDRB, a

mov LEDRC, a

orl P0MOD, #00111000b ; make all CEs outputs

setb CE\_ADC

setb CE\_EE

clr CE\_RTC ; RTC CE is active high

lcall INIT\_SPI

lcall initserialport

Thermocouple\_Forever:

mov b, #0 ; Read channel 0

lcall Read\_ADC\_Channel

mov x+1, R7

mov x+0, R6

; The temperature can be calculated as (ADC\*500/1024)-273 (may overflow 16 bit operations)

; or (ADC\*250/512)-273 (may overflow 16 bit operations)

; or (ADC\*125/256)-273 (may overflow 16 bit operations)

; or (ADC\*62/256)+(ADC\*63/256)-273 (Does not overflow 16 bit operations!)

Load\_y(62)

lcall mul16

mov R4, x+1

mov x+1, R7

mov x+0, R6

Load\_y(63)

lcall mul16

mov R5, x+1

mov x+0, R4

mov x+1, #0

mov y+0, R5

mov y+1, #0

lcall add16

clr mf

Load\_y(273)

lcall x\_lt\_y

jnb mf, Thermocouple\_Positive\_Temperature

Thermocouple\_Negative\_Temperature:

lcall xchg\_xy

lcall sub16

mov a, #'-'

lcall Thermocouple\_Putchar

mov hex2, #3FH

sjmp Thermocouple\_Result

Thermocouple\_Positive\_Temperature:

Load\_y(273)

lcall sub16

mov hex2, #7FH

Thermocouple\_Result:

lcall hex2bcd

lcall Display

lcall Thermocouple\_Send\_Number

lcall WaitHalfSec

lcall WaitHalfSec

ljmp Thermocouple\_Forever

END

$LIST

---------------------------------------------

;;;;;;;;;;;;;;

;;Timer

;;

;;Nina Dacanay, Derek Chan

;;

;;Keeps track of elapsed time for the system

;;

;;Constants:

;; TIMER\_XTAL EQU 33333333

;; TIMER\_FREQ EQU 100

;; TIMER1\_RELOAD EQU 65538-(TIMER\_XTAL/(12\*TIMER\_FREQ))

;;

;;Variables:

;; Timer\_count10ms: ds 1

;; Timer\_Total\_Time\_Seconds: ds 1 ;incrementing every second

;; Timer\_Total\_Time\_Minutes: ds 1 ;incrementing every minute

;; Timer\_Elapsed\_Time: ds 2 ;incrementing every second

;;

;;Interrupt Service Routine:

;; org 0AB8H

;; ljmp ISR\_Timer

;;

;;;;;;;;;;;;;;;

$NOLIST

ISR\_Timer:

; Reload the timer

mov TH1, #high(TIMER1\_RELOAD)

mov TL1, #low(TIMER1\_RELOAD)

; Save used register into the stack

push psw

push acc

push dph

push dpl

; Increment the counter and check if a second has passed

inc Timer\_count10ms

mov a, Timer\_count10ms

cjne A, #100, ISR\_Timer1\_L0

mov Timer\_count10ms, #0

mov a, Timer\_Elapsed\_Time

add a, #1

mov Timer\_Elapsed\_Time, A

mov A, Timer\_Elapsed\_Time+1

addc a, #0

mov Timer\_Elapsed\_Time+1, A

mov a, Timer\_Total\_Time\_Seconds

add a, #1

mov Timer\_Total\_Time\_Seconds, a

cjne A, #60, ISR\_Timer1\_L0

mov Timer\_Total\_Time\_Seconds, #0

mov a, Timer\_Total\_Time\_Minutes

add a, #1

mov Timer\_Total\_Time\_Minutes, a

cjne A, #60, ISR\_Timer1\_L0

mov Timer\_Total\_Time\_Minutes, #0

ISR\_Timer1\_L0:

; Restore used registers

pop dpl

pop dph

pop acc

pop psw

reti

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;Init\_Timer

;;

;;Initializes the timer

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Init\_Timer:

orl TMOD, #00010000B ; GATE=0, C/T\*=0, M1=0, M0=1: 16-bit timer

clr TR1 ; Disable timer 1

clr TF1

mov TH1, #high(TIMER1\_RELOAD) ;PLEASE EXPLAIN WHAT THIS IS

mov TL1, #low(TIMER1\_RELOAD) ;PLEASE EXPLAIN WHAT THIS IS

setb TR1 ; Enable timer 1

setb ET1 ; Enable timer 1 interrupt

mov Timer\_Total\_Time\_Seconds, #000H

mov Timer\_Total\_Time\_Minutes, #0H

mov Timer\_Elapsed\_Time, #000H

mov Timer\_Elapsed\_Time+1, #0H

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;Look-up table for 7-segment displays

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Timer\_LUT:

DB 0C0H, 0F9H, 0A4H, 0B0H, 099H

DB 092H, 082H, 0F8H, 080H, 090H

DB 0FFH ; All segments off

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;Resets the elapsed time

;;

;;@modifies ACC, Timer\_Elapsed\_Time

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Timer\_Reset\_Elapsed\_Time:

clr A

mov Timer\_Elapsed\_Time+0, A

mov Timer\_Elapsed\_Time+1, A

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;Resets both elapsed and total time

;;

;;@modifies ACC, Timer\_Elapsed\_Time

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Timer\_Reset:

clr A

mov Timer\_Elapsed\_Time+0, A

mov Timer\_Elapsed\_Time+1, A

mov Timer\_Total\_Time\_Seconds, A

mov Timer\_Total\_Time\_Minutes, A

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;Clears HEX Displays

;;

;;@modifies ACC, Timer\_Elapsed\_Time

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Timer\_Clear:

mov A, #0FFH

mov HEX0, A

mov HEX1, A

mov HEX2, A

mov HEX3, A

mov HEX4, A

mov HEX5, A

mov HEX6, A

mov HEX7, A

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Display Timer

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Timer\_Display:

lcall Timer\_Display\_Elapsed

lcall Timer\_Display\_Total

ret

Timer\_Display\_Elapsed:

mov dptr, #Timer\_LUT

mov x+0, Timer\_Elapsed\_Time+0

mov x+1, Timer\_Elapsed\_Time+1

lcall hex2bcd

mov a, bcd+0

anl a, #0fH

movc a, @a+dptr

mov HEX0, a

mov a, bcd+0

swap a

anl a, #0fH

movc a, @a+dptr

mov HEX1, a

mov a, bcd+1

anl a, #0fH

movc a, @a+dptr

mov HEX2, a

mov a, bcd+1

swap a

anl a, #0fH

movc a, @a+dptr

mov HEX3, a

ret

Timer\_Display\_Total:

mov dptr, #Timer\_LUT

mov x+0, Timer\_Total\_Time\_Seconds

mov x+1, #0H

lcall hex2bcd

mov a, bcd+0

anl a, #0fH

movc a, @a+dptr

mov HEX4, a

mov a, bcd+0

swap a

anl a, #0fH

movc a, @a+dptr

mov HEX5, a

mov x+0, Timer\_Total\_Time\_Minutes

mov x+1, #0H

lcall hex2bcd

mov a, bcd+0

anl a, #0fH

movc a, @a+dptr

mov HEX6, a

mov a, bcd+0

swap a

anl a, #0fH

movc a, @a+dptr

mov HEX7, a

ret

$LIST

---------------------------------------------

$NOLIST

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

; User Interface

;

; Jessica Hua and Sasha Dordzijev

;

; Function: 1)Settings Initialization (Settings\_Initialization)

; Called at the beginning of the code, this is the user interface

; function used to get the temperature settings and stores them

; into registers.

; 1.1)Displays Welcome\_message

; 1.2)Displays Manual or Preset

; - checks to see if the user wants preset(KEY3)or manual (KEY2)

; 1.3)Manual Setting

; 1.3.1)Displays Enter\_Soak\_temp

; 1.3.2)Waits\_for\_Values

; -waits for the value from the switches

; and waits for Key1 for continue

; 1.3.3)Displays Enter\_Soak\_time

; 1.3.4)Wait\_for\_Values

; 1.3.5)Displays Enter\_Reflow\_temp

; 1.3.6)Wait\_for\_values

; 1.3.7)Displays Enter\_Reflow\_time

; 1.3.8)Wait\_for\_Values

; 1.4)Preset Setting

; 1.4.1)Displays Display\_options

; 1.4.2)Wait\_for\_preset\_values

; -waits for the values from

; switch 1-3 and KEY1 for continue

; 1.5)Displays Confirmation\_Message

;

; 3)Status Display / Value Display (Display\_board)

; function: Displays Current Temp / Target Temp to top line

; Displays Status on bottom line

;

; Inputs: 1) Switches 0-9 for inputing digits

; 2) KEY1 for Pressing Continue

; 3) KEY2 for manual

; 4) KEY3 for preset

;

; Outputs: 1)HEX Display (writes the correct value to x+0 and x+1, and calls hex2bcd and display

; 2) LCD Display depending on status (Derek's portion) displays the

; current state that it is in (Set Up/Heating/Cooldown etc) and

; also displays what you want to enter in the initialization portion

;

; Memory: 1) Soak Temperature - soak\_temperature

; 2) Soak Time - soak\_time

; 3) Reflow Temperature - reflow\_temperature

; 4) Reflow Time - relow\_time

; note: time is given is seconds, and temperature is given; indegrees C. both take two registers to store

;

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

;CSEG

;Function: Gets the correct parameters for over control from the user

UI\_Set\_Up\_Parameters:

;Settings\_Initializations:

;Displays that the reflow oven controller is on

lcall Display\_welcome\_message

;waits 2 seconds

lcall WaitHalfSec

lcall WaitHalfSec

lcall WaitHalfSec

lcall WaitHalfSec

lcall Display\_preset\_or\_manual

;waits for the user to choose preset or manual

ljmp Wait\_for\_preset\_or\_manual

preset:

;Displays the options

lcall Display\_options

lcall turnoff\_7seg

;waits for the user to choose an option

ljmp Wait\_for\_preset\_values

lcall turnoff\_7seg

Settings\_Initialization\_nonwelcome:

;Setting up Soak\_temp

lcall Display\_soak\_temp\_set

lcall Wait\_for\_Values

mov soak\_temperature+0, bcd+0

mov soak\_temperature+1, bcd+1

mov soak\_temperature+2, bcd+2

mov bcd+2, #0

mov bcd+0, #0

mov bcd+1, #0

;Setting up Soak Time

lcall Display\_soak\_time\_set

lcall Wait\_for\_Values

mov soak\_time+0, bcd+0

mov soak\_time+1, bcd+1

mov soak\_time+2, bcd+2

mov bcd+2, #0

mov bcd+0, #0

mov bcd+1, #0

; Setting Up Reflow Temp

lcall Display\_reflow\_temp\_set

lcall Wait\_for\_Values

mov reflow\_temperature+0, bcd+0

mov reflow\_temperature+1, bcd+1

mov reflow\_temperature+2, bcd+2

mov bcd+2, #0

mov bcd+0, #0

mov bcd+1, #0

;Setting Up Reflow Time

lcall Display\_reflow\_time\_set

lcall Wait\_for\_Values

mov reflow\_time+0, bcd+0

mov reflow\_time+1, bcd+1

mov reflow\_time+2, bcd+2

mov bcd+2, #0

mov bcd+0, #0

mov bcd+1, #0

lcall Display

;Displays the confirmation message

Confirmation\_message:

lcall turnoff\_7seg

lcall Display\_Confirmation\_message

;converts all values from BCD into hex for all

;the stored parameters

lcall convertbcd2hex

ret

;Waits for the user to choose an option, or KEY3 for back

wait\_for\_preset\_values:

jnb KEY.3, preset

jb SWA.1, option1

jb SWA.2, option2

jnb SWA.3, wait\_for\_preset\_values\_op3

ljmp option3

wait\_for\_preset\_values\_op3:

jmp wait\_for\_preset\_values

;Waits for the user to choose preset or manual

Wait\_for\_preset\_or\_manual:

jnb KEY.3, jump\_Settings\_initialization\_nonwelcome ;key 3 is preset values

jnb KEY.2, wait\_key2 ;key 2 is manual

jmp wait\_for\_preset\_or\_manual

;because jumps suck

jump\_settings\_Initialization\_nonwelcome:

ljmp settings\_initialization\_nonwelcome

;wait for KEY.2 to be unpressed

wait\_key2:

jb KEY.2, jump\_Preset

jmp wait\_key2

;because jumps suck

jump\_preset:

ljmp preset

Wait\_for\_Confirmation:

jnb KEY.2, jump\_Settings\_Initialization\_nonwelcome

jnb KEY.1, Return\_function

jmp Wait\_for\_confirmation

;Function: Waits for the user to enter a value, and leaves the loop if KEY1 is pressed

Wait\_for\_Values:

;Wait\_for\_Values\_loop:

lcall Display

lcall ReadNumber

jnb KEY.1, wait\_key1

jnc Wait\_for\_Values

lcall Shift\_Digits

lcall Display

ljmp Wait\_for\_Values

;waits for KEY1 to be unpressed

wait\_key1:

jb KEY.1, Return\_function

jmp wait\_key1

;because I suck at coding

Return\_function:

ret

WaitHalfSec:

mov R2, #90

N3: mov R1, #250

N2: mov R0, #250

N1: djnz R0, N1 ; 3 machine cycles-> 3\*30ns\*250=22.5us

djnz R1, N2 ; 22.5us\*250=5.625ms

djnz R2, N3 ; 5.625ms\*90=0.5s (approximately)

ret

;Writes the correct values to the parameters depending

;on what option was chosen

option1:

;move values into the correct registers

mov soak\_temperature+0, #00110000B

mov soak\_temperature+1, #00000001B

mov soak\_time+0, #01100000B

mov soak\_time+1, #00000000B

mov reflow\_temperature+0, #00010000B

mov reflow\_temperature+1, #00000010B

mov reflow\_time+0, #00110000B

mov reflow\_time+1, #00000000B

ljmp confirmation\_message

option2:

mov soak\_temperature+0, #01010000B

mov soak\_temperature+1, #00000001B

mov soak\_time+0, #10010000B

mov soak\_time+1, #00000000B

mov reflow\_temperature+0, #00100000B

mov reflow\_temperature+1, #00000010B

mov reflow\_time+0, #01000000B

mov reflow\_time+1, #00000000B

ljmp confirmation\_message

option3:

mov soak\_temperature+0, #01110000B

mov soak\_temperature+1, #00000001B

mov soak\_time+0, #00100000B

mov soak\_time+1, #00000001B

mov reflow\_temperature+0, #00110000B

mov reflow\_temperature+1, #00000010B

mov reflow\_time+0, #01000101B

mov reflow\_time+1, #00000000B

ljmp confirmation\_message

;converts values from bcd2hex and stores it into the parameters

convertbcd2hex:

mov bcd+0, soak\_temperature+0

mov bcd+1, soak\_temperature+1

lcall bcd2hex

mov soak\_temperature+0, x+0

mov soak\_temperature+1, x+1

mov bcd+0, soak\_time+0

mov bcd+1, soak\_time+1

lcall bcd2hex

mov soak\_time+0, x+0

mov soak\_time+1, x+1

mov bcd+0, reflow\_temperature+0

mov bcd+1, reflow\_temperature+1

lcall bcd2hex

mov reflow\_temperature+0, x+0

mov reflow\_temperature+1, x+1

mov bcd+0, reflow\_time+0

mov bcd+1, reflow\_time+1

lcall bcd2hex

mov reflow\_time+0, x+0

mov reflow\_time+1, x+1

ret

;turning off all of the 7-seg displays

turnoff\_7seg:

mov HEX0, #11111111B

mov HEX1, #11111111B

mov HEX2, #11111111B

mov HEX3, #11111111B

mov HEX4, #11111111B

ret

test\_proper\_values:

clr UI\_Input\_Error

;Compares all temperatures and sees if the soak temp and reflow temp is less than 235

;if it is too high, display error, then jump to setting up parameters again

load\_y(235)

mov x+0, soak\_temperature

mov x+1, soak\_temperature+1

lcall x\_lt\_y

jb mf, test\_proper\_values\_checkReflow

lcall display\_soak\_temperature\_high

lcall waitHalfSec

lcall waitHalfSec

lcall waitHalfSec

lcall waitHalfSec

setb UI\_Input\_Error

test\_proper\_values\_checkReflow:

;if it is too high, display error, then jump to setting up parameters again

load\_y(235)

mov x+0, reflow\_temperature

mov x+1, reflow\_temperature+1

lcall x\_lt\_y

jb mf, test\_proper\_values\_checkReflowTime

lcall display\_reflow\_temperature\_high

lcall WaitHalfSec

lcall waitHalfSec

lcall waitHalfSec

lcall waitHalfSec

setb UI\_Input\_Error

;Compares to see if the reflow time is less or equal to 45

;if it is too high, display error, then jump to setting up parameters again

test\_proper\_values\_checkReflowTime:

load\_y(45)

mov x+0, reflow\_time

mov x+1, reflow\_time+1

lcall x\_lt\_y

jb mf, test\_proper\_values\_done

lcall display\_reflow\_time\_high

lcall waitHalfSec

lcall waitHalfSec

lcall waitHalfSec

setb UI\_Input\_Error

;if everything is fine, continue with the process

test\_proper\_values\_done:

ret

$LIST

---------------------------------------------