ECE 367 Fall 2012 Final Exam Project

Programmable Thermostat

Design a programmable thermostat for a standard home heating system. The thermostat must have the following minimum set of functioning features:

1. 12 Hour AM/PM clock (hr:mn am/pm format) (must have an additional ~100 times speedup control switch for demonstration purpose.) Once programmed and running the clock must run continuously (possibly in the background) while other systems are operating
2. Temperature Sensor System (Use a thermistor connected to a voltage divider into an internal NanoCore12 AD converter – Temperature range 64 – 80 degrees F tuned for voltage range 0.5 to 4.5 V)
3. LCD Display to Show Current Time, Current Temperature, Current time range, and State of the System (Heat ON/OFF)
4. Programmable Functions: i) Clock Set ii.) Temperature Setting for Two Fixed Time of Day Ranges: Day (6 am to 10:59 pm) and Night (11 pm to 5:59 am)

The report must include all of the following items:

The Source code which must include:
Header: Your Name
UIN
ECE 367
Final exam Project
12/10/2012 (or early turn-in date if applies)
Explanation of the purpose of the code.
Explanation of the NanoCore12 pin usage.
Comments: Every line of code including EQU's needs a comment for explanation.

Logic Diagram: Basic I/O between components
Electric Circuit Diagram: Neat, complete electric circuit wiring diagram with component pin labels.
Design Calculations: Show all of your calculations for the temperature sensing system.
User Manual: Carefully written user instructions.
Conclusion: Test your system and discuss how well does your system meet the specifications?

IMPORTANT: You MUST submit an electronic copy of your assembly code file when you do your demo. Bring a thumb (jump) drive with a copy of your code. NO CODE, NO DEMO! You fail!

Correctly completing all of the fundamental features above is worth 180 points.
If your System does not work you MUST still show your circuit and turn in a report no later than the end of the scheduled ECE 367 course final exam time (Monday 12/10/2012 10 A.M.) . LATE WORK WILL NOT BE ACCEPTED FOR ANY REASON!

You may earn extra points if you correctly implement any of the following. If these extra features do not work there will **not** be partial credit. These features must function correctly.

1. Use an interrupt for any purpose except RESET: 20 points each interrupt system that preforms a **relevant** purpose.

2. Implement four **adjustable** (programmable) time ranges (Morning, Day, Evening, Night) with **adjustable** (programmable) temperature settings for each time range. – 30 points

3. Have a complete cooling system (AC) control option that operates under the same time ranges as heat time ranges with unique programmable temperature settings. Display should indicate AC ON/OFF condition. – 20 points.

EARLY TURN-IN EXTRA CREDIT

You MUST make an appointment by email for an early turn-in appointment. No walk-ins!

Extra Credit Schedule for early turn-in by:

Wednesday: 12/05/2012, 5 p.m. 60 points. None accepted during class or after 5 p.m.
Thursday: 12/06/2012, 3 p.m. 50 points. None accepted after 3 p.m.
Friday: 12/07/2012, Demos accepted from noon until noon until 2:45 ONLY!
And during lecture 40 points. None Accepted after 4 P.M. –Firm!
Times earlier in the day, for the day above, can be arranged.

If you do not make an appointment for early turn-in you will not be allowed to do the demo!

Accept for Friday during lecture, all demos for early turn-in extra credit will be done for Prof. Becker in his office. Room 804 SEO. Please let me know by email if and when you intend to do an early turn-in.

Again - **If you do not make an appointment for early turn-in you will not be allowed to do the demo.**

To receive early turn-in credit you MUST have all of the fundamental features working and a completed report. Once you turn in your report and demo your system you will not be allowed to re-submit for any other extra credit. All evaluation will be on a first come first serve basis. Do not show up 2 minutes before closing time. When the deadline arrives, early turn-in times out!

ECE 367: Final Exam Project
Programmable Thermostat

Semester: Fall 2012

Name: Kai Zhao

Signature: _____

UIN: 670720413

Due Date: 2012 December 06

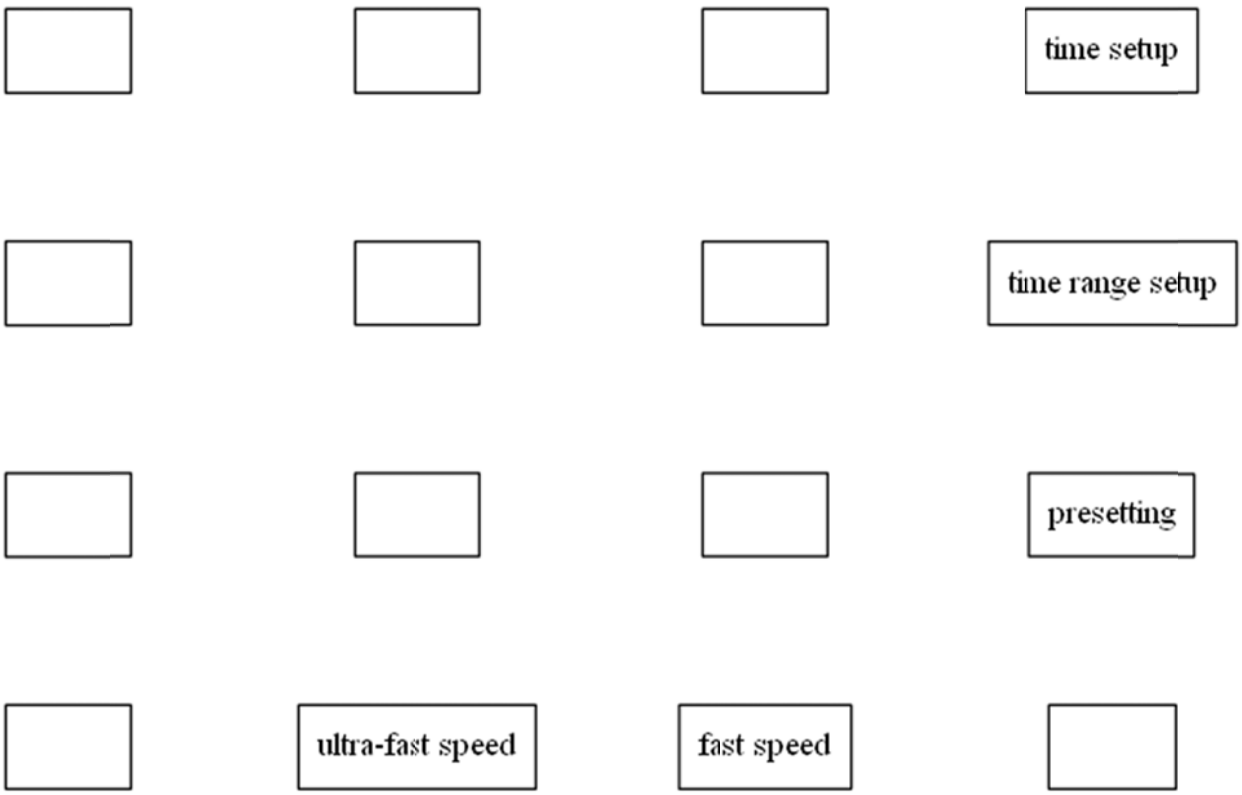
Lab Section: T11

TA: Chenjie Tang

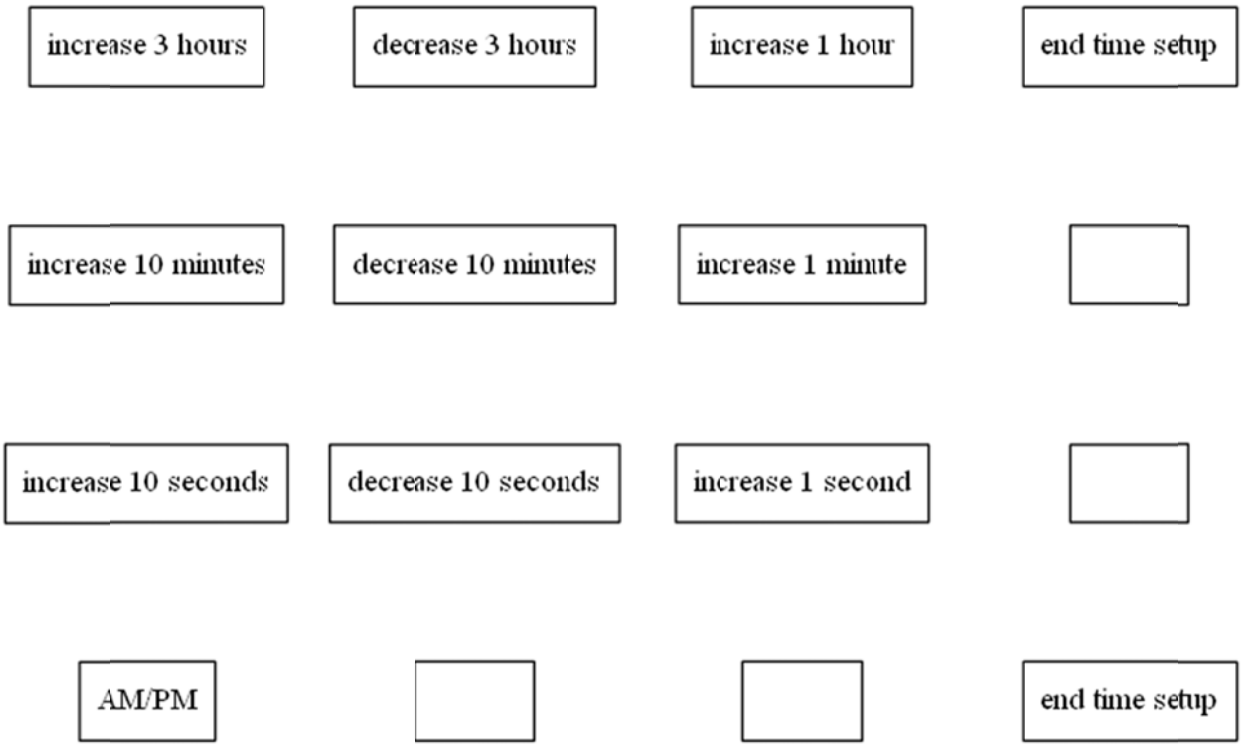
User Manual:

This program is a programmable thermostat. The user should first supply power to the circuit and then press the reset button to start the display. On startup, the time is 12:00 AM. The user has an option of adjusting the time, adjusting the time ranges, adjusting when the heater or AC should turn on for each time range. Furthermore, there are some preset time ranges such morning person mode (there is a longer morning period and the heater/AC is more sensitive during the morning) and money saving mode (the room need to be really cold or really warm for the heater or AC to turn on). The user also has a toggle option for fast timer speed (~100x faster), ultra-fast timer (~1000x faster) speed, and ultra-double fast timer speed (~1100x faster). By using the interrupt buttons, the user also have the option of manually turning on/off the heater/AC.

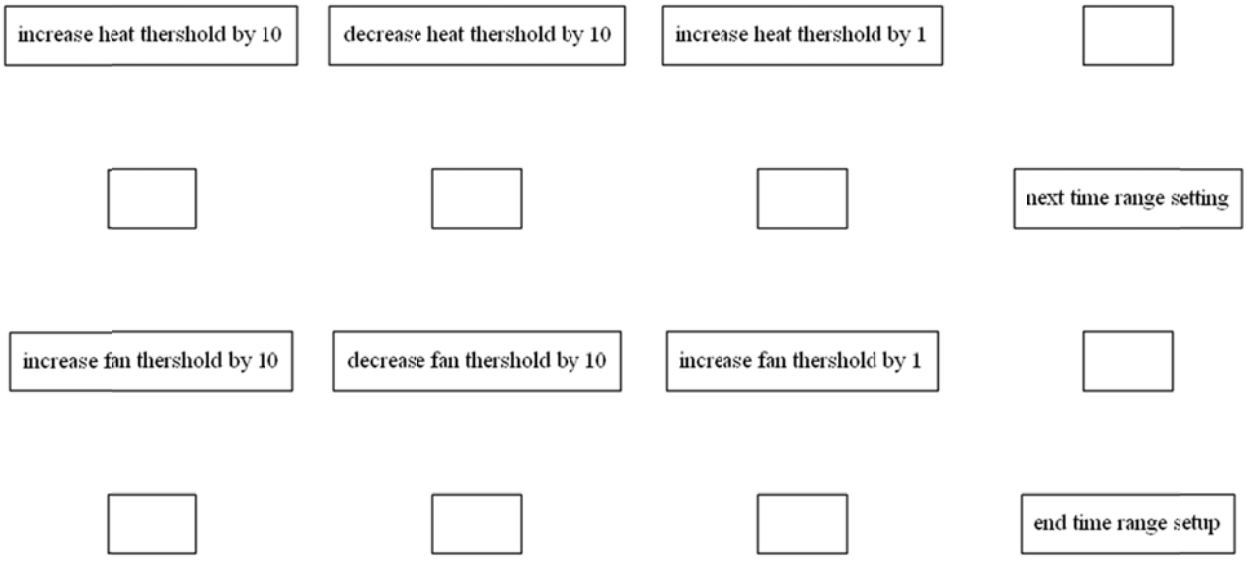
By default, the keys are



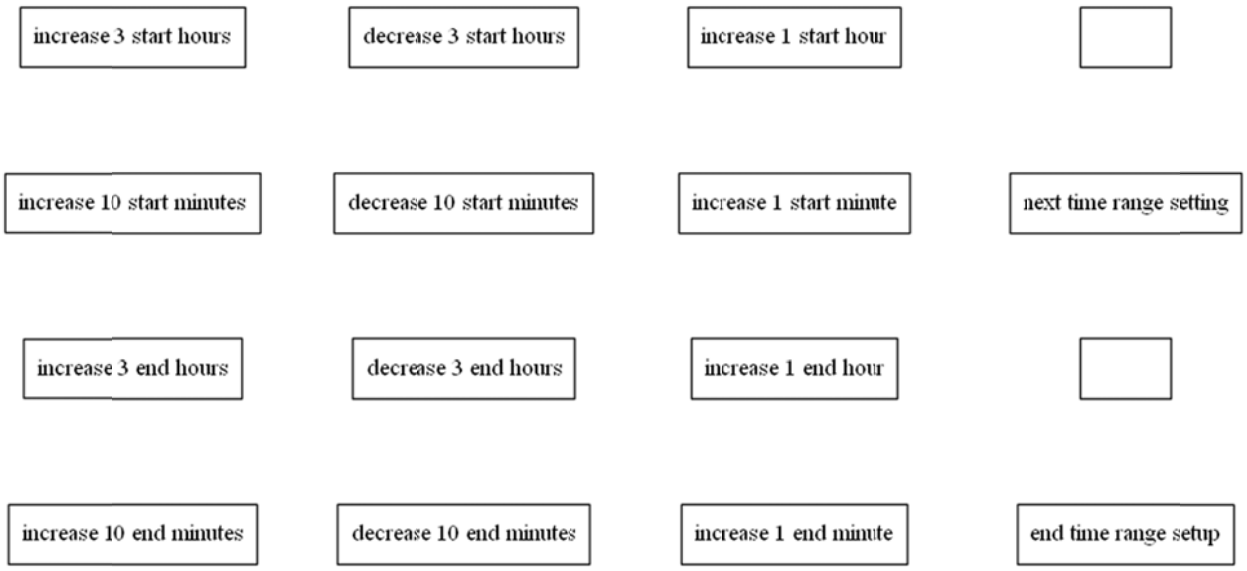
'A' sets the time. After pressing ‘A’, the keyboard becomes the following:



'B' sets the time range. After pressing ‘B’ an odd amount of times, the keyboard becomes the following:



After pressing ‘B’ an odd amount of times, the keyboard becomes the following:



'C' switches to a different presetting.

'D' ends programming mode and displays time.

'E' sets ~100x fast/normal timer speed.

'F' sets ~1000x ultra-fast/normal timer speed.

Toggle ‘E’ and ‘F’ together for ~1100x ultra-double fast timer speed.

‘XIRQ’ (left) toggles the AC to be (1) automatic, (2) manual on, and (3) manual off.

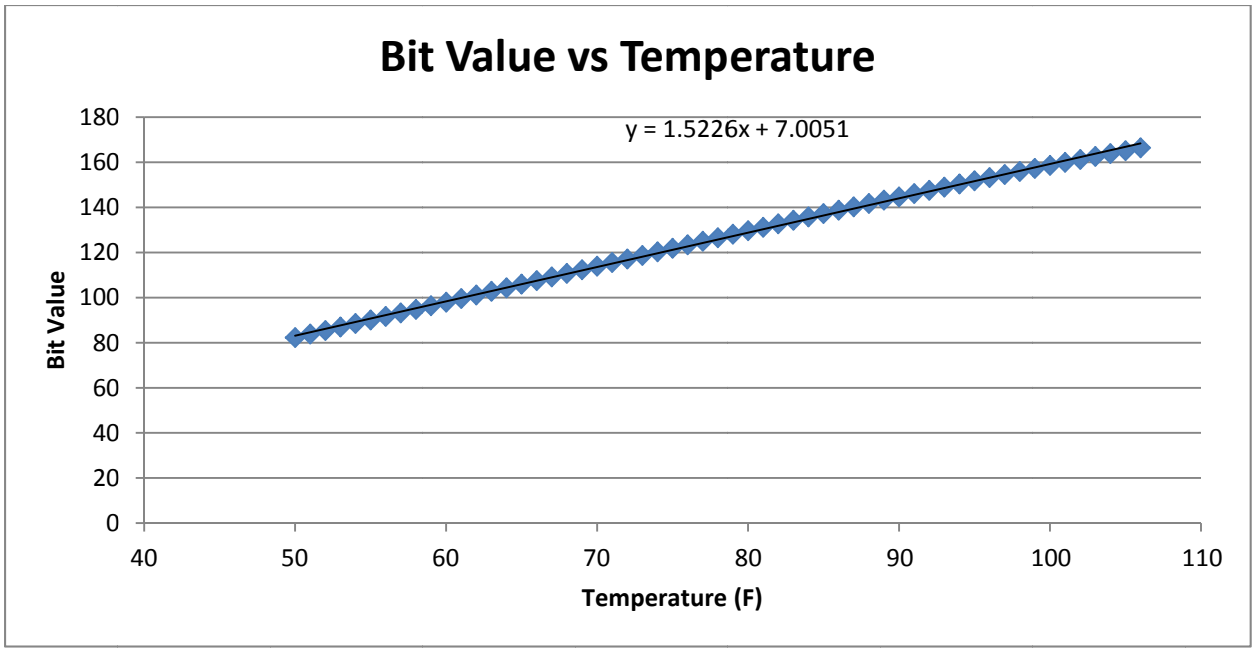
‘IRQ’ (right) toggles the heater to be (1) automatic, (2) manual on, and (3) manual off.

Design Calculations:

For the temperature, a linear approximation as shown below was used. A linear approximation is valid because the thermostat is only valid through a certain range of temperatures. As the range of value increases, then my linear approximation will have more error. However, for room temperatures, the thermostat is pretty accurate.

F	C	K	Rt2	V(t)	bit	bit change
50	10	283	20369.57	1.646385	82	
51	10.55556	283.5556	19813.22	1.677108	84	2
52	11.11111	284.1111	19274.16	1.707991	85	2
53	11.66667	284.6667	18751.78	1.739023	87	2
54	12.22222	285.2222	18245.51	1.770193	89	2
55	12.77778	285.7778	17754.8	1.80149	90	2
56	13.33333	286.3333	17279.12	1.832904	92	2
57	13.88889	286.8889	16817.94	1.864423	93	2
58	14.44444	287.4444	16370.79	1.896037	95	2
59	15	288	15937.19	1.927734	96	2
60	15.55556	288.5556	15516.67	1.959503	98	2
61	16.11111	289.1111	15108.8	1.991334	100	2
62	16.66667	289.6667	14713.16	2.023214	101	2
63	17.22222	290.2222	14329.33	2.055133	103	2
64	17.77778	290.7778	13956.92	2.087079	104	2
65	18.33333	291.3333	13595.56	2.119043	106	2
66	18.88889	291.8889	13244.88	2.151011	108	2
67	19.44444	292.4444	12904.52	2.182975	109	2
68	20	293	12574.15	2.214922	111	2
69	20.55556	293.5556	12253.45	2.246843	112	2
70	21.11111	294.1111	11942.08	2.278726	114	2
71	21.66667	294.6667	11639.76	2.310562	116	2
72	22.22222	295.2222	11346.19	2.342339	117	2
73	22.77778	295.7778	11061.08	2.374047	119	2
74	23.33333	296.3333	10784.17	2.405677	120	2
75	23.88889	296.8889	10515.18	2.437219	122	2
76	24.44444	297.4444	10253.88	2.468663	123	2
77	25	298	10000	2.5	125	2
78	25.55556	298.5556	9753.32	2.53122	127	2
79	26.11111	299.1111	9513.608	2.562314	128	2

80	26.66667	299.6667	9280.644	2.593274	130	2
81	27.22222	300.2222	9054.215	2.624091	131	2
82	27.77778	300.7778	8834.117	2.654757	133	2
83	28.33333	301.3333	8620.151	2.685263	134	2
84	28.88889	301.8889	8412.126	2.715602	136	2
85	29.44444	302.4444	8209.859	2.745765	137	2
86	30	303	8013.17	2.775747	139	1
87	30.55556	303.5556	7821.887	2.805539	140	1
88	31.11111	304.1111	7635.845	2.835135	142	1
89	31.66667	304.6667	7454.882	2.864528	143	1
90	32.22222	305.2222	7278.843	2.893712	145	1
91	32.77778	305.7778	7107.579	2.922681	146	1
92	33.33333	306.3333	6940.943	2.95143	148	1
93	33.88889	306.8889	6778.797	2.979951	149	1
94	34.44444	307.4444	6621.004	3.008242	150	1
95	35	308	6467.433	3.036296	152	1
96	35.55556	308.5556	6317.958	3.064109	153	1
97	36.11111	309.1111	6172.457	3.091676	155	1
98	36.66667	309.6667	6030.811	3.118994	156	1
99	37.22222	310.2222	5892.905	3.146058	157	1
100	37.77778	310.7778	5758.629	3.172865	159	1
101	38.33333	311.3333	5627.876	3.199411	160	1
102	38.88889	311.8889	5500.541	3.225694	161	1
103	39.44444	312.4444	5376.525	3.25171	163	1
104	40	313	5255.731	3.277457	164	1
105	40.55556	313.5556	5138.064	3.302932	165	1
106	41.11111	314.1111	5023.434	3.328134	166	1



The presetting options are as follows:

Mode	Values	Night	Morning	Day	Evening	presetting
Default	Time Range	12:00AM-05:59AM	06:00AM-11:59AM	12:00PM-05:59PM	06:00PM-11:59PM	0
	Heater (F) - AC (F)	70-80	70-80	70-80	70-80	
Demonstration	Time Range	11:00PM-05:59AM	12:00AM-12:00AM	06:00AM-10:59PM	12:00AM-12:00AM	1
	Heater (F) - AC (F)	75-76	75-76	75-76	75-76	
Morning Person	Time Range	10:00PM-03:59AM	04:00AM-11:30AM	11:31AM-05:59PM	06:00PM-09:59PM	2
	Heater (F) - AC (F)	70-80	74-75	70-80	60-90	
Money Saving	Time Range	12:00AM-05:59AM	06:00AM-11:59AM	12:00PM-05:59PM	06:00PM-11:59PM	3
	Heater (F) - AC (F)	50-100	50-100	60-90	50-100	
Evening Person	Time Range	03:00AM-06:59AM	07:00AM-11:59AM	12:00AM-05:59PM	06:00PM-02:59AM	not included
	Heater (F) - AC (F)	70-80	60-90	70-80	74-75	

Conclusion:

My system meets the specifications very well and the timer is quite accurate.

Extra features includes

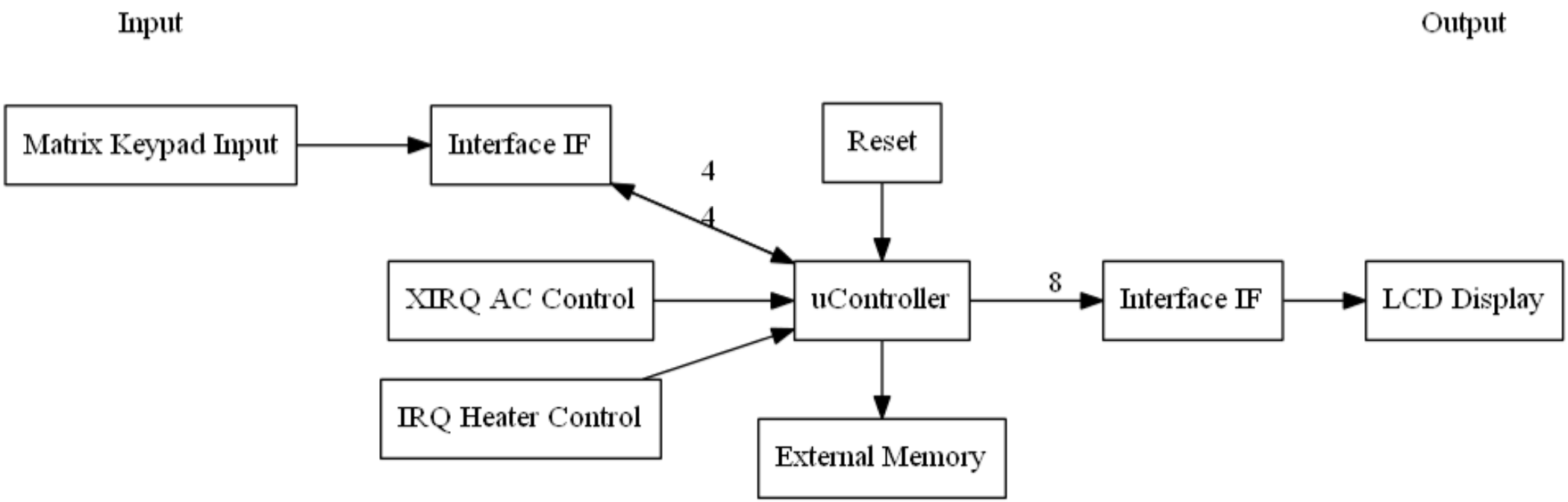
two interrupts (one controls the heater while the other controls the AC),

4 adjustable time ranges with adjustable temperature settings,

a complete AC cooling system, seconds value (as opposed to only minutes and hours),

hard-coded presetting,

and (~1000x) ultra-fast speedup control for the timer.


Input
Output
Matrix Keypad Input
Interface IF
4
4
Reset
XIRQ AC Control
uController
8
Interface IF
LCD Display
IRQ Heater Control
External Memory

```
/*
;;Identifying Information;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; University of Illinois at Chicago, Dept. of Electrical & Computer Engineering
; ECE 367 -Microprocessor-Based Design
; Project Final: Programmable Thermostat in C Language                                ;
; Version 2012 Dec 06
; By: Kai Zhao
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;Program Description;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
This program is a programmable thermostat. The user should first supply power to
  the circuit and then press the reset button to start the display. On startup,
  the time is 12:00 AM. The user has an option of adjusting the time, adjusting
  the time ranges, adjusting when the heater or AC should turn on for each time
  range. Furthermore, there are some preset time ranges such morning person mode
  (there is a longer morning period and the heater/AC is more sensitive during
  the morning) and money saving mode (the room need to be really cold or really
  warm for the heater or AC to turn on). The user also has a toggle option for
  fast timer speed (~100x faster), ultra-fast timer (~1000x faster) speed, and
  ultra-double fast timer speed (~1100x faster). By using the interrupt buttons,
  the user also have the option of manually turning on/off the heater/AC

'A' sets the time.
'B' sets the time range.
'C' switches to a different presetting.
'D' ends programming mode and displays time.
'E' sets ~100x fast/normal timer speed.
'F' sets ~1000x ultra-fast/normal timer speed.
'E' and 'F' together for ~1100x ultra-double fast timer speed.
'XIRQ' (left) toggles the AC to be (1) automatic, (2) on, and (3) off.
'IRQ' (right) toggles the heater to be (1) automatic, (2) on, and (3) off.

PinT0 is used for column 1 input
PinT1 is used for column 2 input
PinT2 is used for column 3 input
PinT3 is used for column 4 input
PinT4 is used for row 1 output
PinT5 is used for row 2 output
PinT6 is used for row 3 output
PinT7 is used for row 4 output
PinM0 is used for RS
PinM1 is used for E (pin 6 of the LCD Display)
PinM2 is not used
PinM3 is used for RCK
PinM4 is used for Ser in
PinM5 is used for SCK
PinA0 is used for A/D voltage input form the thermistor
Pin XIRQ is used for AC control
Pin IRQ is used for Heater Control
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;*/

/* Some include (header) files needed by Codewarrior with machine info for the NanoCore12        */

#include <hidef.h>                    /* common defines and macros */
#include "derivative.h"               /* derivative-specific definitions */

/* We need to define some constants. Similar to EQU's in assembly                               */
/* The commented out defines already exist in one of the above header files. The compiler       */
/* does not like the redundancy. So, they are commented out with the // symbols                 */

#define     IOREGS_BASE  0x0000

#define     _IO8(off)    *(unsigned char  volatile *)(IOREGS_BASE + off) //define form prototype 8-bit
#define     _IO16(off)   *(unsigned short volatile *)(IOREGS_BASE + off) //define form prototype 16-bit


#define   PORTT     _IO8(0x240)     /* portT data register is unsigned 8-bit at address $0240       */
/* because of the form prototype defines above this is the same as                              */
/* #define PORTT *(unsigned char  volatile *) (0x240);    Means PORTT points to address $0240    */
/* the statement PORTT = 0x34; means to store $34 at location $0240                             */
/* if the contents of PORTT is 0xd3 then the assignment x = PORTT; means x is now equal to 0xd3 */
/************************************************************************************************/
#define   PORTTi     _IO8(0x241)      // portT data register
#define   IRQCR      _IO8(0x001E)     // used for interrupts
#define   keypad    PORTT
#define   PORTM     _IO8(0x250)       // portM data register
//#define     TSCR1    _IO8(0x46)     //timer system control register
#define     MCCTL    IO8(0x66)        //modulus down conunter control
```

```
 79: #define     MCFLG   _IO8(0x67)        //down counter flags
 80: #define     MCCNT   _IO16(0x76)       //modulus down counter register
 81: //#define     PTT    _IO8(0x240)     //portt data register
 82: //#define     DDRT     _IO8(0x242)     //portt direction register
 83: //#define     CRGFLG  _IO8(0x37)      //pll flags register
 84: //#define     SYNR    _IO8(0x34)      //synthesizer / multiplier register
 85: //#define     REFDV   _IO8(0x35)      //reference divider register
 86: //#define     CLKSEL  _IO8(0x39)      //clock select register
 87: //#define     PLLCTL  _IO8(0x3a)      //pll control register
 88: #define   SPCR1    _IO8(0xD8)         // used to attempt using the LCD display
 89: #define   SPCR2    _IO8(0xD9)         // used to attempt using the LCD display
 90: #define   SPIB     _IO8(0xDA)         // used to attempt using the LCD display
 91: #define   SPSR     _IO8(0xDB)         // used to attempt using the LCD display
 92: #define   SPDR     _IO8(0xDD)         // used to attempt using the LCD display
 93: #define   ATD0CTL0 _IO8(0x80)         // used for AD converter slot input
 94: #define   ATD0CTL1 _IO8(0x81)         // used for AD converter slot input
 95: #define   ATD0CTL2 _IO8(0x82)         // used for AD converter slot input
 96: #define   ATD0CTL3 _IO8(0x83)         // used for AD converter slot input
 97: #define   ATD0CTL4 _IO8(0x84)         // used for AD converter slot input
 98: #define   ATD0CTL5 _IO8(0x85)         // used for AD converter slot input
 99: #define   ATD0STAT0 _IO8(0x86)        // used for AD converter slot input
100: #define   ATD0DR0  _IO8(0x90)         // used for AD converter slot input
101: #define   ATD0DR1  _IO8(0x92)         // used for AD converter slot input
102: #define   RS       (0x01)             // used for the SIPO to the LCD display
103: #define   ENABLE   (0x02)             // used for the SIPO to the LCD display
104: #define   RCK      (0x08)             // used for the SIPO to the LCD display
105:
106: // Let's define some bit locations for some flags and config bits
107: #define   PLLSEL  0x80                // included in sample code for bit manipulation
108: #define   LOCK    0x08                // included in sample code for bit manipulation
109: #define   TFFCA   0x10                // included in sample code for bit manipulation
110: #define   MCZF    0x80                // included in sample code for bit manipulation
111: #define   BIT0    0x01                // included in sample code for bit manipulation
112: #define   BIT1    0x02                // included in sample code for bit manipulation
113: #define   BIT2    0x04                // included in sample code for bit manipulation
114: #define   BIT3    0x08                // included in sample code for bit manipulation
115:
116: // Prototype for IRQ Interrupt Service Routine function named irqISR
117: // The next five lines are required
118: // Change the name for other interrupts
119: extern void near irqISR(void);         // used to change code
120: extern void near xirqISR(void);        // irqISR() is in a different file
121:
122: #pragma CODE_SEG __NEAR_SEG NON_BANKED// used to change code
123: #pragma CODE_SEG DEFAULT               // change code section to default
124: typedef void (*near tIsrFunc) (void);
125: const tIsrFunc _vect[] @0xFFF2 = {
126:     irqISR,                          // used for heater control
127:       xirqISR                        // used for AC control
128: };                                    // This format is for a single interrupt system
129:
130: // Next we have an array. This is the same as a lookup table in assembler. The "const" (constant) type
131: // means to store into ROM not RAM memory. The array is for 7-seg display patterns of decimal digits
132: // 0 - 9. Note the last value in the array will clear the 7-seg display (it's 0x00)
133:
134: const unsigned char segpat[11] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x67,0x00};
135:
136:
137: // Make Flag1 an 8-bit character variable. Initialize Flag1 to 0x00.
138: char  Flag1 = 0x00;                    // used in sample code for method
139:
140: /* Here we give function prototypes before we start on the main code */
141:
142: void SetClk8(void);                    // prototype to attempt LCD Display
143: int  updateTemperature(void);          // update temperature
144: void delayby1ms(int);                  // prototype to delay
145: char getkey(void);                     // prototype to grab input
146: void keyrelease(void);                 // prototype to check key release
147: void outdata(char);                    // prototype to output data
148: void outdata2(char);                   // prototype to output data
149: void initLCD(void);                    // prototype to attempt LCD Display
150: void command(int);                     // prototype to attempt LCD Display
151: void print(char);                      // prototype to attempt LCD Display
152: void displayLCD(int, int, int, int, int, int, int);
153: /* The main program begins here */
154:
155: unsigned char heatOverride = 0x00;     // used for heater control
156: unsigned char fanOverride = 0x00;      // used for fan control
```

```
157: void main(void)
158: {
159:   /* put your own code here */
160:   /* Below is a simple program example that sets up the PLL to 24mHz,       */
161:   /* gets a value from the keypad and displays it in binary on portM        */
162:   /* pins PM5, PM4, PM3, and PM2. There is some delay required and,         */
163:   /* of course, a key release routine. Does not seem to be any key bounce.  */
164:
165:   char key1;                          // define key1 as 8-bit character
166:   char key2;                          // used to store key grabbed
167:   char ones = 0;                      // counter variable
168:   char tens = 0;                      // counter variable
169:   int doBreak = 0;                    // boolean variable
170:   int doBreak2 = 0;                   // boolean variable
171:   int slowSpeed = 7;                  // used to control normal speed
172:   int ultraSpeed = 0;                 // used to toggle ultra-fast speed
173:   int speed = slowSpeed;              // toggle variable
174:   int direction = 0;                  // used to store time data
175:   int step = 0;                       // used to store time data
176:   int hours = 0;                      // used to store time data
177:   int minutes = 0;                    // used to store time data
178:   int seconds = 0;                    // used to store time data
179:   int loopCounter = 0;                // loop counter
180:   int loopCounter2 = 0;               // loop counter
181:   int timeRange = 0;                  // time data
182:   int temperatureBits = 0;            // used to store temperature data
183:   int temperatureF = 0;               // used to store temperature data
184:   int presettings = 0;                // used to store temperature data
185:   int nightHourS = 0;                 // used to store time range data
186:   int nightMinuteS = 0;               // used to store time range data
187:   int nightHourE = 5;                 // used to store time range data
188:   int nightMinuteE = 59;              // used to store time range data
189:   int nightHeat = 70;                 // used to store time range data
190:   int nightFan = 80;                  // used to store time range data
191:   int morningHourS = 6;               // used to store time range data
192:   int morningMinuteS = 0;             // used to store time range data
193:   int morningHourE = 11;              // used to store time range data
194:   int morningMinuteE = 59;            // used to store time range data
195:   int morningHeat = 70;               // used to store time range data
196:   int morningFan = 80;                // used to store time range data
197:   int dayHourS = 12;                  // used to store time range data
198:   int dayMinuteS = 0;                 // used to store time range data
199:   int dayHourE = 17;                  // used to store time range data
200:   int dayMinuteE = 59;                // used to store time range data
201:   int dayHeat = 70;                   // used to store time range data
202:   int dayFan = 80;                    // used to store time range data
203:   int eveningHourS = 18;              // used to store time range data
204:   int eveningMinuteS = 0;             // used to store time range data
205:   int eveningHourE = 23;              // used to store time range data
206:   int eveningMinuteE = 59;            // used to store time range data
207:   int eveningHeat = 70;               // used to store time range data
208:   int eveningFan = 80;                // used to store time range data
209:   SetClk8();                          // go setup the PLL
210:
211:   DDRT = 0xf0;                        // setup the data direction registers
212:   DDRM = 0x3b;                        // setup the data direction registers
213:
214:   SPIB = 0x22;                        // setup the data direction registers
215:   SPCR1 = 0x50;                       // setup the data direction registers
216:   SPCR2 = 0x00;                       // setup the data direction registers
217:   PORTM = PORTM|RCK;                  // set RCK
218:   PORTM = PORTM&!ENABLE;              // clear PortM
219:
220:   IRQCR = 0xC0;                       // enable IRQ with falling edge trigger
221:   asm(ANDCC #$bf);                    // enable XIRQ interrupt
222:   EnableInterrupts;                   // same as asm(cli)
223:   initLCD();                          // init LCD
224:   while (1)                           // always
225:   {
226:     for (loopCounter = 0; loopCounter<speed; loopCounter++)
227:     {                                 // for speed
228:       command(0x01);                  // clear display
229:       if (hours>12)                   // print out hours
230:       {                               // print out hours
231:         print((hours-12)/10+48);      // print out hours
232:         print((hours-12)%10+48);      // print out hours
233:       }                               // print out hours
234:       else                            // print out hours
```

```
235:       {                                // print out hours
236:         print(hours/10+48);            // print out hours
237:         print(hours%10+48);            // print out hours
238:       }                                // print ':'
239:       print(':');                      // print out minutes
240:       print(minutes/10+48);            // print out minutes
241:       print(minutes%10+48);            // print out minutes
242:       print(':');                      // print out seconds
243:       print(seconds/10+48);            // print out seconds
244:       print(seconds%10+48);            // print out seconds
245:       if (hours<12)                    // print out AM/PM
246:       {                                // print out AM/PM
247:         print('A');                    // print out AM/PM
248:       }                                // print out AM/PM
249:       else                             // print out AM/PM
250:       {                                // print out AM/PM
251:         print('P');                    // print out AM/PM
252:       }                                // print out AM/PM
253:       print('M');                      // print out AM/PM
254:       print(' ');                      // print out AM/PM
255:       temperatureBits =  updateTemperature(); // update temperature
256:       if (temperatureBits<143)         // print out temperature
257:         temperatureF = 50+(temperatureBits-82)/2; // print out temperature
258:       else                             // print out temperature
259:         temperatureF = 79+(temperatureBits-143); // print out temperature
260:       print(temperatureF/100+48);      // print out temperature
261:       print((temperatureF%100)/10+48);// print out temperature
262:       print(temperatureF%10+48);       // print out temperature
263:       print('F');                      // print out temperature
264:       command(0xc0);                   // go to next line
265:       if (hours>=morningHourS&&hours<=morningHourE&&(hours>morningHourS||minutes>=morningMinuteS)&&(hours<
266:       {                                // compute time range
267:         print('M');                    // print out time range
268:         print('o');                    // print out time range
269:         print('r');                    // print out time range
270:         print('n');                    // print out time range
271:         print('H');                    // print out time range
272:         print('e');                    // print out time range
273:         print('a');                    // print out time range
274:         print('t');                    // print out time range
275:         print('O');                    // print out time range
276:         if ((heatOverride==0&&temperatureF<morningHeat)||heatOverride==1)
277:         {                               // print out heat data
278:           print('n');                   // print out heat data
279:           print(' ');                   // print out heat data
280:         }                               // print out heat data
281:         else                            // print out heat data
282:         {                               // print out heat data
283:           print('f');                   // print out heat data
284:           print('f');                   // print out heat data
285:         }                               // print out AC data
286:         print('A');                    // print out AC data
287:         print('c');                    // print out AC data
288:         print('O');                    // print out AC data
289:         if ((fanOverride==0&&temperatureF>morningFan)||fanOverride==1)
290:         {                               // print out AC data
291:           print('n');                   // print out AC data
292:           print(' ');                   // print out AC data
293:         }                               // print out AC data
294:         else                            // print out AC data
295:         {                               // print out AC data
296:           print('f');                   // print out AC data
297:           print('f');                   // print out AC data
298:         }                               // print out AC data
299:       }                                 // print out AC data
300:       else if (hours>=dayHourS&&hours<=dayHourE&&(hours>dayHourS||minutes>=dayMinuteS)&&(hours<dayHourE||m
301:       {                                 // compute time range
302:         print('D');                     // print out time range
303:         print('a');                     // print out time range
304:         print('y');                     // print out time range
305:         print(' ');                     // print out time range
306:         print('H');                     // print out heat data
307:         print('e');                     // print out heat data
308:         print('a');                     // print out heat data
309:         print('t');                     // print out heat data
310:         print('O');                     // print out heat data
311:         if ((heatOverride==0&&temperatureF<dayHeat)||heatOverride==1)
312:         {                               // print out heat data
```

```
313:          print('n');                  // print out heat data
314:          print(' ');                  // print out heat data
315:        }                              // print out heat data
316:        else                           // print out heat data
317:        {                              // print out heat data
318:          print('f');                  // print out heat data
319:          print('f');                  // print out heat data
320:        }                              // print out heat data
321:        print('A');                    // print out AC data
322:        print('c');                    // print out AC data
323:        print('O');                    // print out AC data
324:        if ((fanOverride==0&&temperatureF>dayFan)||fanOverride==1)
325:        {                              // print out AC data
326:          print('n');                  // print out AC data
327:          print(' ');                  // print out AC data
328:        }                              // print out AC data
329:        else                           // print out AC data
330:        {                              // print out AC data
331:          print('f');                  // print out AC data
332:          print('f');                  // print out AC data
333:        }                              // print out AC data
334:      }                                // print out AC data
335:      else if (hours>=eveningHourS&&hours<=eveningHourE&&(hours>eveningHourS||minutes>=eveningMinuteS)&&(h
336:      {                                // compute time range
337:        print('E');                    // print out time range
338:        print('v');                    // print out time range
339:        print('e');                    // print out time range
340:        print('n');                    // print out time range
341:        print('H');                    // print out heat data
342:        print('e');                    // print out heat data
343:        print('a');                    // print out heat data
344:        print('t');                    // print out heat data
345:        print('O');                    // print out heat data
346:        if ((heatOverride==0&&temperatureF<eveningHeat)||heatOverride==1)
347:        {                              // print out heat data
348:          print('n');                  // print out heat data
349:          print(' ');                  // print out heat data
350:        }                              // print out heat data
351:        else                           // print out heat data
352:        {                              // print out heat data
353:          print('f');                  // print out heat data
354:          print('f');                  // print out heat data
355:        }                              // print out heat data
356:        print('A');                    // print out AC data
357:        print('c');                    // print out AC data
358:        print('O');                    // print out AC data
359:        if ((fanOverride==0&&temperatureF>eveningFan)||fanOverride==1)
360:        {                              // print out AC data
361:          print('n');                  // print out AC data
362:          print(' ');                  // print out AC data
363:        }                              // print out AC data
364:        else                           // print out AC data
365:        {                              // print out AC data
366:          print('f');                  // print out AC data
367:          print('f');                  // print out AC data
368:        }                              // print out AC data
369:      }                                // print out AC data
370:      else // if (hours>=nightHourS&&hours<=nightHourE&&(hours>nightHourS||minutes>=nightMinuteS)&&(hours<
371:      {                                // compute time range
372:        print('N');                    // print out time range
373:        print('g');                    // print out time range
374:        print('h');                    // print out time range
375:        print('t');                    // print out time range
376:        print('H');                    // print out heat data
377:        print('e');                    // print out heat data
378:        print('a');                    // print out heat data
379:        print('t');                    // print out heat data
380:        print('O');                    // print out heat data
381:        if ((heatOverride==0&&temperatureF<nightHeat)||heatOverride==1)
382:        {                              // print out heat data
383:          print('n');                  // print out heat data
384:          print(' ');                  // print out heat data
385:        }                              // print out heat data
386:        else                           // print out heat data
387:        {                              // print out heat data
388:          print('f');                  // print out heat data
389:          print('f');                  // print out heat data
390:        }                              // print out heat data
```

```
391:         print('A');                    // print out AC data
392:         print('c');                    // print out AC data
393:         print('O');                    // print out AC data
394:         if ((fanOverride==0&&temperatureF>nightFan)||fanOverride==1)
395:         {                               // print out AC data
396:           print('n');                  // print out AC data
397:           print(' ');                  // print out AC data
398:         }                               // print out AC data
399:         else                            // print out AC data
400:         {                               // print out AC data
401:           print('f');                  // print out AC data
402:           print('f');                  // print out AC data
403:         }                               // print out AC data
404:       }                                 // print out AC data
405: /* The following lines of code are used to print out the temperature in Celsius
406:       print(' ');
407:       if (temperatureBits<154)
408:         temperatureC = 10+(temperatureBits-82)/3;
409:       else
410:         temperatureC = 36+(temperatureBits-154)/2;
411:       print(temperatureC/10+48);
412:       print(temperatureC%10+48);
413:       print('C');
414: */
415:       key1 = getkey();                // grab key
416:       if (key1==10)                   // check key
417:       {                               // do key option
418:         while (1)                     // while setting time
419:         {
420:           key2 = getkey();            // grab another key
421:           if (key2==10||key2==13)     // do key option of updating time
422:             break;                    // do key option of updating time
423:           if (key2==1)                // do key option of updating time
424:             hours += 3;               // do key option of updating time
425:           if (key2==2)                // do key option of updating time
426:             hours -= 3;               // do key option of updating time
427:           if (key2==3)                // do key option of updating time
428:             hours++;                  // do key option of updating time
429:           if (key2==4)                // do key option of updating time
430:             minutes += 10;            // do key option of updating time
431:           if (key2==5)                // do key option of updating time
432:             minutes -= 10;            // do key option of updating time
433:           if (key2==6)                // do key option of updating time
434:             minutes++;                // do key option of updating time
435:           if (key2==7)                // do key option of updating time
436:             seconds += 10;            // do key option of updating time
437:           if (key2==8)                // do key option of updating time
438:             seconds -= 10;            // do key option of updating time
439:           if (key2==9)                // do key option of updating time
440:             seconds++;                // do key option of updating time
441:           if (key2==0)                // do key option of updating time
442:             hours += 12;              // do key option of updating time
443:           if (seconds>59)             // adjust time
444:           {                           // adjust time
445:             seconds -= 60;            // adjust time
446:             minutes++;                // adjust time
447:           }                           // adjust time
448:           if (seconds<0)              // adjust time
449:           {                           // adjust time
450:             seconds += 60;            // adjust time
451:             minutes--;                // adjust time
452:           }                           // adjust time
453:           if (minutes>59)             // adjust time
454:           {                           // adjust time
455:             minutes -= 60;            // adjust time
456:             hours++;                  // adjust time
457:           }                           // adjust time
458:           if (minutes<0)              // adjust time
459:           {                           // adjust time
460:             minutes += 60;            // adjust time
461:             hours--;                  // adjust time
462:           }                           // adjust time
463:           if (hours>23)               // adjust time
464:             hours -=24;               // adjust time
465:           if (hours<0)                // adjust time
466:             hours +=24;               // adjust time
467:           command(0x01);              // print updated time
468:           print('S');                 // print updated time
```

```
469:            print('e');                 // print updated time
470:            print('t');                 // print updated time
471:            print(' ');                 // print updated time
472:            print('t');                 // print updated time
473:            print('i');                 // print updated time
474:            print('m');                 // print updated time
475:            print('e');                 // print updated time
476:            print(':');                 // print updated time
477:            command(0xc0);              // print updated time
478:            if (hours>12)               // print updated time
479:            {                           // print updated time
480:              print((hours-12)/10+48);  // print updated time
481:              print((hours-12)%10+48);  // print updated time
482:            }                           // print updated time
483:            else                        // print updated time
484:            {                           // print updated time
485:              print(hours/10+48);       // print updated time
486:              print(hours%10+48);       // print updated time
487:            }                           // print updated time
488:            print(':');                 // print updated time
489:            print(minutes/10+48);       // print updated time
490:            print(minutes%10+48);       // print updated time
491:            print(':');                 // print updated time
492:            print(seconds/10+48);       // print updated time
493:            print(seconds%10+48);       // print updated time
494:            if (hours<12)               // print updated time
495:            {                           // print updated time
496:              print('A');               // print updated time
497:            }                           // print updated time
498:            else                        // print updated time
499:            {                           // print updated time
500:              print('P');               // print updated time
501:            }                           // print updated time
502:            print('M');                 // print updated time
503:            delayby1ms(60);             // delay
504:          }
505:        }
506:        if (key1==11)                   // check key
507:        {
508:          doBreak = 0;                  // set booleans
509:          doBreak2 = 0;                 // set booleans
510:          while (doBreak2==0)           // while setting time ranges
511:          {
512:            while (doBreak==0&&doBreak2==0) // while setting night
513:            {
514:              key2 = getkey();          // grab another key
515:              // Update the Temperature Ranges for Time Ranges
516:              if (key2==11)             // update time range values
517:                doBreak = 1;            // update time range values
518:              if (key2==13)             // update time range values
519:                doBreak2 = 1;           // update time range values
520:              if (key2==1)              // update time range values
521:                nightHeat += 10;        // update time range values
522:              if (key2==2)              // update time range values
523:                nightHeat -= 10;        // update time range values
524:              if (key2==3)              // update time range values
525:                nightHeat++;            // update time range values
526:              if (key2==7)              // update time range values
527:                nightFan += 10;         // update time range values
528:              if (key2==8)              // update time range values
529:                nightFan -= 10;         // update time range values
530:              if (key2==9)              // update time range values
531:                nightFan++;             // update time range values
532:              // Adjust the Temperature Ranges to be in Bound
533:              if (nightFan>999)         // update time range values
534:                nightFan = 999;         // update time range values
535:              if (nightFan<0)           // update time range values
536:                nightFan = 0;           // update time range values
537:              if (nightHeat>999)        // update time range values
538:                nightHeat = 999;        // update time range values
539:              if (nightHeat<0)          // update time range values
540:                nightHeat = 0;          // update time range values
541:              // display/update LCD
542:              displayLCD(1, nightFan, nightHeat, nightHourS, nightMinuteS, nightHourE, nightMinuteE);
543:            }
544:            doBreak = 0;                // update boolean
545:            while (doBreak==0&&doBreak2==0) // while setting night
546:            {
```

```
547:              key2 = getkey();          // grab key
548:              // Update the Time Ranges
549:              if (key2==11)             // update time range values
550:                doBreak = 1;            // update time range values
551:              if (key2==13)             // update time range values
552:                doBreak2 = 1;           // update time range values
553:              if (key2==1)              // update time range values
554:                nightHourS += 3;        // update time range values
555:              if (key2==2)              // update time range values
556:                nightHourS -= 3;        // update time range values
557:              if (key2==3)              // update time range values
558:                nightHourS++;           // update time range values
559:              if (key2==4)              // update time range values
560:                nightMinuteS += 10;     // update time range values
561:              if (key2==5)              // update time range values
562:                nightMinuteS -= 10;     // update time range values
563:              if (key2==6)              // update time range values
564:                nightMinuteS++;         // update time range values
565:              if (key2==7)              // update time range values
566:                nightHourE += 3;        // update time range values
567:              if (key2==8)              // update time range values
568:                nightHourE -= 3;        // update time range values
569:              if (key2==9)              // update time range values
570:                nightHourE++;           // update time range values
571:              if (key2==0)              // update time range values
572:                nightMinuteE += 10;     // update time range values
573:              if (key2==15)             // update time range values
574:                nightMinuteE -= 10;     // update time range values
575:              if (key2==14)             // update time range values
576:                nightMinuteE++;         // update time range values
577:              // Adjust the Time Range to be in bound
578:              if (nightMinuteS>59)      // update time range values
579:              {                         // update time range values
580:                nightMinuteS -= 60;     // update time range values
581:                nightHourS++;           // update time range values
582:              }                         // update time range values
583:              if (nightMinuteS<0)       // update time range values
584:              {                         // update time range values
585:                nightMinuteS += 60;     // update time range values
586:                nightHourS--;           // update time range values
587:              }                         // update time range values
588:              if (nightHourS>23)        // update time range values
589:                nightHourS -=24;        // update time range values
590:              if (nightHourS<0)         // update time range values
591:                nightHourS +=24;        // update time range values
592:              if (nightMinuteE>59)      // update time range values
593:              {                         // update time range values
594:                nightMinuteE -= 60;     // update time range values
595:                nightHourE++;           // update time range values
596:              }                         // update time range values
597:              if (nightMinuteE<0)       // update time range values
598:              {                         // update time range values
599:                nightMinuteE += 60;     // update time range values
600:                nightHourE--;           // update time range values
601:              }                         // update time range values
602:              if (nightHourE>23)        // update time range values
603:                nightHourE -=24;        // update time range values
604:              if (nightHourE<0)         // update time range values
605:                nightHourE +=24;        // update time range values
606:              // display/update LCD
607:              displayLCD(1, nightFan, nightHeat, nightHourS, nightMinuteS, nightHourE, nightMinuteE);
608:            }
609:            doBreak = 0;                // update boolean
610:            while (doBreak==0&&doBreak2==0) // while setting morning tme range
611:            {
612:              key2 = getkey();          // get key
613:              // Update the Temperature Ranges for Time Ranges
614:              if (key2==11)             // update time range values
615:                doBreak = 1;            // update time range values
616:              if (key2==13)             // update time range values
617:                doBreak2 = 1;           // update time range values
618:              if (key2==1)              // update time range values
619:                morningHeat += 10;      // update time range values
620:              if (key2==2)              // update time range values
621:                morningHeat -= 10;      // update time range values
622:              if (key2==3)              // update time range values
623:                morningHeat++;          // update time range values
624:              if (key2==7)              // update time range values
```

```
625:              morningFan += 10;       // update time range values
626:            if (key2==8)              // update time range values
627:              morningFan -= 10;       // update time range values
628:            if (key2==9)              // update time range values
629:              morningFan++;           // update time range values
630:            // Adjust the Temperature Ranges to be in Bound
631:            if (morningFan>999)       // update time range values
632:              morningFan = 999;       // update time range values
633:            if (morningFan<0)         // update time range values
634:              morningFan = 0;         // update time range values
635:            if (morningHeat>999)      // update time range values
636:              morningHeat = 999;      // update time range values
637:            if (morningHeat<0)        // update time range values
638:              morningHeat = 0;        // update time range values
639:            // display/update LCD
640:            displayLCD(2, morningFan, morningHeat, morningHourS, morningMinuteS, morningHourE, morningMinu
641:          }
642:          doBreak = 0;                // update boolean
643:          while (doBreak==0&&doBreak2==0) // while setting morning time range
644:          {
645:            key2 = getkey();          // get key
646:            // Update the Time Ranges
647:            if (key2==11)             // update time range values
648:              doBreak = 1;            // update time range values
649:            if (key2==13)             // update time range values
650:              doBreak2 = 1;           // update time range values
651:            if (key2==1)              // update time range values
652:              morningHourS += 3;      // update time range values
653:            if (key2==2)              // update time range values
654:              morningHourS -= 3;      // update time range values
655:            if (key2==3)              // update time range values
656:              morningHourS++;         // update time range values
657:            if (key2==4)              // update time range values
658:              morningMinuteS += 10;   // update time range values
659:            if (key2==5)              // update time range values
660:              morningMinuteS -= 10;   // update time range values
661:            if (key2==6)              // update time range values
662:              morningMinuteS++;       // update time range values
663:            if (key2==7)              // update time range values
664:              morningHourE += 3;      // update time range values
665:            if (key2==8)              // update time range values
666:              morningHourE -= 3;      // update time range values
667:            if (key2==9)              // update time range values
668:              morningHourE++;         // update time range values
669:            if (key2==0)              // update time range values
670:              morningMinuteE += 10;   // update time range values
671:            if (key2==15)             // update time range values
672:              morningMinuteE -= 10;   // update time range values
673:            if (key2==14)             // update time range values
674:              morningMinuteE++;       // update time range values
675:            // Adjust the Time Range to be in bound
676:            if (morningMinuteS>59)    // update time range values
677:            {                         // update time range values
678:              morningMinuteS -= 60;   // update time range values
679:              morningHourS++;         // update time range values
680:            }                         // update time range values
681:            if (morningMinuteS<0)     // update time range values
682:            {                         // update time range values
683:              morningMinuteS += 60;   // update time range values
684:              morningHourS--;         // update time range values
685:            }                         // update time range values
686:            if (morningHourS>23)      // update time range values
687:              morningHourS -=24;      // update time range values
688:            if (morningHourS<0)       // update time range values
689:              morningHourS +=24;      // update time range values
690:            if (morningMinuteE>59)    // update time range values
691:            {                         // update time range values
692:              morningMinuteE -= 60;   // update time range values
693:              morningHourE++;         // update time range values
694:            }                         // update time range values
695:            if (morningMinuteE<0)     // update time range values
696:            {                         // update time range values
697:              morningMinuteE += 60;   // update time range values
698:              morningHourE--;         // update time range values
699:            }                         // update time range values
700:            if (morningHourE>23)      // update time range values
701:              morningHourE -=24;      // update time range values
702:            if (morningHourE<0)       // update time range values
```

```
703:              morningHourE +=24;      // update time range values
704:            // display/update LCD
705:            displayLCD(2, morningFan, morningHeat, morningHourS, morningMinuteS, morningHourE, morningMinu
706:          }
707:          doBreak = 0;                // update boolean
708:          while (doBreak==0&&doBreak2==0)
709:          {                           // while setting day temperatures
710:            key2 = getkey();          // get key
711:            // Update the Temperature Ranges for Time Ranges
712:            if (key2==11)             // update time range values
713:              doBreak = 1;            // update time range values
714:            if (key2==13)             // update time range values
715:              doBreak2 = 1;           // update time range values
716:            if (key2==1)              // update time range values
717:              dayHeat += 10;          // update time range values
718:            if (key2==2)              // update time range values
719:              dayHeat -= 10;          // update time range values
720:            if (key2==3)              // update time range values
721:              dayHeat++;              // update time range values
722:            if (key2==7)              // update time range values
723:              dayFan += 10;           // update time range values
724:            if (key2==8)              // update time range values
725:              dayFan -= 10;           // update time range values
726:            if (key2==9)              // update time range values
727:              dayFan++;               // update time range values
728:            // Adjust the Temperature Ranges to be in Bound
729:            if (dayFan>999)           // update time range values
730:              dayFan = 999;           // update time range values
731:            if (dayFan<0)             // update time range values
732:              dayFan = 0;             // update time range values
733:            if (dayHeat>999)          // update time range values
734:              dayHeat = 999;          // update time range values
735:            if (dayHeat<0)            // update time range values
736:              dayHeat = 0;            // update time range values
737:            // display/update LCD
738:            displayLCD(3, dayFan, dayHeat, dayHourS, dayMinuteS, dayHourE, dayMinuteE);
739:          }
740:          doBreak = 0;                // update boolean
741:          while (doBreak==0&&doBreak2==0)
742:          {
743:            key2 = getkey();          // update time range values
744:            // Update the Time Ranges
745:            if (key2==11)             // update time range values
746:              doBreak = 1;            // update time range values
747:            if (key2==13)             // update time range values
748:              doBreak2 = 1;           // update time range values
749:            if (key2==1)              // update time range values
750:              dayHourS += 3;          // update time range values
751:            if (key2==2)              // update time range values
752:              dayHourS -= 3;          // update time range values
753:            if (key2==3)              // update time range values
754:              dayHourS++;             // update time range values
755:            if (key2==4)              // update time range values
756:              dayMinuteS += 10;       // update time range values
757:            if (key2==5)              // update time range values
758:              dayMinuteS -= 10;       // update time range values
759:            if (key2==6)              // update time range values
760:              dayMinuteS++;           // update time range values
761:            if (key2==7)              // update time range values
762:              dayHourE += 3;          // update time range values
763:            if (key2==8)              // update time range values
764:              dayHourE -= 3;          // update time range values
765:            if (key2==9)              // update time range values
766:              dayHourE++;             // update time range values
767:            if (key2==0)              // update time range values
768:              dayMinuteE += 10;       // update time range values
769:            if (key2==15)             // update time range values
770:              dayMinuteE -= 10;       // update time range values
771:            if (key2==14)             // update time range values
772:              dayMinuteE++;           // update time range values
773:            // Adjust the Time Range to be in bound
774:            if (dayMinuteS>59)        // update time range values
775:            {                         // update time range values
776:              dayMinuteS -= 60;       // update time range values
777:              dayHourS++;             // update time range values
778:            }                         // update time range values
779:            if (dayMinuteS<0)         // update time range values
780:            {                         // update time range values
```

```
781:                dayMinuteS += 60;       // update time range values
782:                dayHourS--;             // update time range values
783:              }                         // update time range values
784:              if (dayHourS>23)          // update time range values
785:                dayHourS -=24;          // update time range values
786:              if (dayHourS<0)           // update time range values
787:                dayHourS +=24;          // update time range values
788:              if (dayMinuteE>59)        // update time range values
789:              {                         // update time range values
790:                dayMinuteE -= 60;       // update time range values
791:                dayHourE++;             // update time range values
792:              }                         // update time range values
793:              if (dayMinuteE<0)         // update time range values
794:              {                         // update time range values
795:                dayMinuteE += 60;       // update time range values
796:                dayHourE--;             // update time range values
797:              }                         // update time range values
798:              if (dayHourE>23)          // update time range values
799:                dayHourE -=24;          // update time range values
800:              if (dayHourE<0)           // update time range values
801:                dayHourE +=24;          // update time range values
802:              // display/update LCD
803:              displayLCD(3, dayFan, dayHeat, dayHourS, dayMinuteS, dayHourE, dayMinuteE);
804:            }
805:            doBreak = 0;                // udpate boolean
806:            while (doBreak==0&&doBreak2==0) // while setting evening
807:            {
808:              key2 = getkey();          // grab key
809:              // Update the Temperature Ranges for Time Ranges
810:              if (key2==11)             // update time range values
811:                doBreak = 1;            // update time range values
812:              if (key2==13)             // update time range values
813:                doBreak2 = 1;           // update time range values
814:              if (key2==1)              // update time range values
815:                eveningHeat += 10;      // update time range values
816:              if (key2==2)              // update time range values
817:                eveningHeat -= 10;      // update time range values
818:              if (key2==3)              // update time range values
819:                eveningHeat++;          // update time range values
820:              if (key2==7)              // update time range values
821:                eveningFan += 10;       // update time range values
822:              if (key2==8)              // update time range values
823:                eveningFan -= 10;       // update time range values
824:              if (key2==9)              // update time range values
825:                eveningFan++;           // update time range values
826:              // Adjust the Temperature Ranges to be in Bound
827:              if (eveningFan>999)       // update time range values
828:                eveningFan = 999;       // update time range values
829:              if (eveningFan<0)         // update time range values
830:                eveningFan = 0;         // update time range values
831:              if (eveningHeat>999)      // update time range values
832:                eveningHeat = 999;      // update time range values
833:              if (eveningHeat<0)        // update time range values
834:                eveningHeat = 0;        // update time range values
835:              // display/update LCD
836:              displayLCD(4, eveningFan, eveningHeat, eveningHourS, eveningMinuteS, eveningHourE, eveningMinu
837:            }
838:            doBreak = 0;                // update boolean
839:            while (doBreak==0&&doBreak2==0)
840:            {                           // while setting evening time range
841:              key2 = getkey();          // grab key
842:              // Update the Time Ranges
843:              if (key2==11)             // update time range values
844:                doBreak = 1;            // update time range values
845:              if (key2==13)             // update time range values
846:                doBreak2 = 1;           // update time range values
847:              if (key2==1)              // update time range values
848:                eveningHourS += 3;      // update time range values
849:              if (key2==2)              // update time range values
850:                eveningHourS -= 3;      // update time range values
851:              if (key2==3)              // update time range values
852:                eveningHourS++;         // update time range values
853:              if (key2==4)              // update time range values
854:                eveningMinuteS += 10;   // update time range values
855:              if (key2==5)              // update time range values
856:                eveningMinuteS -= 10;   // update time range values
857:              if (key2==6)              // update time range values
858:                eveningMinuteS++;       // update time range values
```

```
859:             if (key2==7)               // update time range values
860:               eveningHourE += 3;       // update time range values
861:             if (key2==8)               // update time range values
862:               eveningHourE -= 3;       // update time range values
863:             if (key2==9)               // update time range values
864:               eveningHourE++;          // update time range values
865:             if (key2==0)               // update time range values
866:               eveningMinuteE += 10;    // update time range values
867:             if (key2==15)              // update time range values
868:               eveningMinuteE -= 10;    // update time range values
869:             if (key2==14)              // update time range values
870:               eveningMinuteE++;        // update time range values
871:             // Adjust the Time Range to be in bound
872:             if (eveningMinuteS>59)     // update time range values
873:             {                          // update time range values
874:               eveningMinuteS -= 60;    // update time range values
875:               eveningHourS++;          // update time range values
876:             }                          // update time range values
877:             if (eveningMinuteS<0)      // update time range values
878:             {                          // update time range values
879:               eveningMinuteS += 60;    // update time range values
880:               eveningHourS--;          // update time range values
881:             }                          // update time range values
882:             if (eveningHourS>23)       // update time range values
883:               eveningHourS -=24;       // update time range values
884:             if (eveningHourS<0)        // update time range values
885:               eveningHourS +=24;       // update time range values
886:             if (eveningMinuteE>59)     // update time range values
887:             {                          // update time range values
888:               eveningMinuteE -= 60;    // update time range values
889:               eveningHourE++;          // update time range values
890:             }                          // update time range values
891:             if (eveningMinuteE<0)      // update time range values
892:             {                          // update time range values
893:               eveningMinuteE += 60;    // update time range values
894:               eveningHourE--;          // update time range values
895:             }                          // update time range values
896:             if (eveningHourE>23)       // update time range values
897:               eveningHourE -=24;       // update time range values
898:             if (eveningHourE<0)        // update time range values
899:               eveningHourE +=24;       // update time range values
900:             // display/update LCD
901:             displayLCD(4, eveningFan, eveningHeat, eveningHourS, eveningMinuteS, eveningHourE, eveningMinu
902:           }
903:           doBreak = 0;                 // update time range values
904:         }
905:       }
906:       if (key1==12)                    // check key
907:       {
908:         if (presettings<3)             // update presettings
909:           presettings++;               // update presettings
910:         else                           // update presettings
911:           presettings = 0;             // update presettings
912:         command(0x01);                 // clear display
913:         if (presettings==0)            // update time range values
914:         {                              // update time range values
915:           nightHourS = 0;              // update time range values
916:           nightMinuteS = 0;            // update time range values
917:           nightHourE = 5;              // update time range values
918:           nightMinuteE = 59;           // update time range values
919:           nightHeat = 70;              // update time range values
920:           nightFan = 80;               // update time range values
921:           morningHourS = 6;            // update time range values
922:           morningMinuteS = 0;          // update time range values
923:           morningHourE = 11;           // update time range values
924:           morningMinuteE = 59;         // update time range values
925:           morningHeat = 70;            // update time range values
926:           morningFan = 80;             // update time range values
927:           dayHourS = 12;               // update time range values
928:           dayMinuteS = 0;              // update time range values
929:           dayHourE = 17;               // update time range values
930:           dayMinuteE = 59;             // update time range values
931:           dayHeat = 70;                // update time range values
932:           dayFan = 80;                 // update time range values
933:           eveningHourS = 18;           // update time range values
934:           eveningMinuteS = 0;          // update time range values
935:           eveningHourE = 23;           // update time range values
936:           eveningMinuteE = 59;         // update time range values
```

```
  937:           eveningHeat = 70;           // update time range values
  938:           eveningFan = 80;            // update time range values
  939:           print('D');                 // display the presetting mode
  940:           print('e');                 // display the presetting mode
  941:           print('f');                 // display the presetting mode
  942:           print('a');                 // display the presetting mode
  943:           print('u');                 // display the presetting mode
  944:           print('l');                 // display the presetting mode
  945:           print('t');                 // display the presetting mode
  946:         }
  947:         if (presettings==1)           // update time range values
  948:         {                             // update time range values
  949:           nightHourS = 23;            // update time range values
  950:           nightMinuteS = 0;           // update time range values
  951:           nightHourE = 5;             // update time range values
  952:           nightMinuteE = 59;          // update time range values
  953:           nightHeat = 75;             // update time range values
  954:           nightFan = 76;              // update time range values
  955:           morningHourS = 0;           // update time range values
  956:           morningMinuteS = 0;         // update time range values
  957:           morningHourE = 0;           // update time range values
  958:           morningMinuteE = 0;         // update time range values
  959:           morningHeat = 75;           // update time range values
  960:           morningFan = 76;            // update time range values
  961:           dayHourS = 6;               // update time range values
  962:           dayMinuteS = 0;             // update time range values
  963:           dayHourE = 22;              // update time range values
  964:           dayMinuteE = 59;            // update time range values
  965:           dayHeat = 75;               // update time range values
  966:           dayFan = 76;                // update time range values
  967:           eveningHourS = 0;           // update time range values
  968:           eveningMinuteS = 0;         // update time range values
  969:           eveningHourE = 0;           // update time range values
  970:           eveningMinuteE = 0;         // update time range values
  971:           eveningHeat = 75;           // update time range values
  972:           eveningFan = 76;            // update time range values
  973:           print('D');                 // display the presetting mode
  974:           print('e');                 // display the presetting mode
  975:           print('m');                 // display the presetting mode
  976:           print('o');                 // display the presetting mode
  977:           print('n');                 // display the presetting mode
  978:           print('s');                 // display the presetting mode
  979:           print('t');                 // display the presetting mode
  980:           print('r');                 // display the presetting mode
  981:           print('a');                 // display the presetting mode
  982:           print('t');                 // display the presetting mode
  983:           print('i');                 // display the presetting mode
  984:           print('o');                 // display the presetting mode
  985:           print('n');                 // display the presetting mode
  986:         }
  987:         if (presettings==2)           // update time range values
  988:         {                             // update time range values
  989:           nightHourS = 22;            // update time range values
  990:           nightMinuteS = 0;           // update time range values
  991:           nightHourE = 3;             // update time range values
  992:           nightMinuteE = 59;          // update time range values
  993:           nightHeat = 70;             // update time range values
  994:           nightFan = 80;              // update time range values
  995:           morningHourS = 4;           // update time range values
  996:           morningMinuteS = 0;         // update time range values
  997:           morningHourE = 11;          // update time range values
  998:           morningMinuteE = 30;        // update time range values
  999:           morningHeat = 74;           // update time range values
 1000:           morningFan = 75;            // update time range values
 1001:           dayHourS = 11;              // update time range values
 1002:           dayMinuteS = 31;            // update time range values
 1003:           dayHourE = 17;              // update time range values
 1004:           dayMinuteE = 59;            // update time range values
 1005:           dayHeat = 70;               // update time range values
 1006:           dayFan = 80;                // update time range values
 1007:           eveningHourS = 18;          // update time range values
 1008:           eveningMinuteS = 0;         // update time range values
 1009:           eveningHourE = 21;          // update time range values
 1010:           eveningMinuteE = 59;        // update time range values
 1011:           eveningHeat = 60;           // update time range values
 1012:           eveningFan = 90;            // update time range values
 1013:           print('M');                 // display the presetting mode
 1014:           print('o');                 // display the presetting mode
```

```
1015:           print('r');                // display the presetting mode
1016:           print('n');                // display the presetting mode
1017:           print('i');                // display the presetting mode
1018:           print('n');                // display the presetting mode
1019:           print('g');                // display the presetting mode
1020:         }
1021:         if (presettings==3)           // update time range values
1022:         {                             // update time range values
1023:           nightHourS = 0;             // update time range values
1024:           nightMinuteS = 0;           // update time range values
1025:           nightHourE = 5;             // update time range values
1026:           nightMinuteE = 59;          // update time range values
1027:           nightHeat = 50;             // update time range values
1028:           nightFan = 100;             // update time range values
1029:           morningHourS = 6;           // update time range values
1030:           morningMinuteS = 0;         // update time range values
1031:           morningHourE = 11;          // update time range values
1032:           morningMinuteE = 59;        // update time range values
1033:           morningHeat = 50;           // update time range values
1034:           morningFan = 100;           // update time range values
1035:           dayHourS = 12;              // update time range values
1036:           dayMinuteS = 0;             // update time range values
1037:           dayHourE = 17;              // update time range values
1038:           dayMinuteE = 59;            // update time range values
1039:           dayHeat = 60;               // update time range values
1040:           dayFan = 90;                // update time range values
1041:           eveningHourS = 18;          // update time range values
1042:           eveningMinuteS = 0;         // update time range values
1043:           eveningHourE = 23;          // update time range values
1044:           eveningMinuteE = 59;        // update time range values
1045:           eveningHeat = 50;           // update time range values
1046:           eveningFan = 100;           // update time range values
1047:           print('M');                // display the presetting mode
1048:           print('o');                // display the presetting mode
1049:           print('n');                // display the presetting mode
1050:           print('e');                // display the presetting mode
1051:           print('y');                // display the presetting mode
1052:           print(' ');                // display the presetting mode
1053:           print('s');                // display the presetting mode
1054:           print('a');                // display the presetting mode
1055:           print('v');                // display the presetting mode
1056:           print('i');                // display the presetting mode
1057:           print('n');                // display the presetting mode
1058:           print('g');                // display the presetting mode
1059:         }
1060:         delayby1ms(500);              // delay to display
1061:       }
1062:       if (key1==14)                   // check key
1063:       {
1064:         if (speed==slowSpeed)         // toggle fast speed
1065:           speed = 1;                  // toggle fast speed
1066:         else                          // toggle fast speed
1067:           speed = slowSpeed;          // toggle fast speed
1068:       }
1069:       if (key1==15)                   // check key
1070:       {
1071:         if (ultraSpeed==0)            // toggle ultra-fast speed
1072:           ultraSpeed = 1;             // toggle ultra-fast speed
1073:         else                          // toggle ultra-fast speed
1074:           ultraSpeed = 0;             // toggle ultra-fast speed
1075:       }
1076:       delayby1ms(60);                 // delay to display time
1077:     }
1078:     seconds++;                        // update timer
1079:     if (speed==1)                     // update timer
1080:       seconds += 100;                 // update timer
1081:     if (ultraSpeed==1)                // update timer
1082:       minutes += 14;                  // update timer
1083:     while (seconds>59)                // update timer
1084:     {                                 // update timer
1085:       seconds -= 60;                  // update timer
1086:       minutes++;                      // update timer
1087:     }                                 // update timer
1088:     if (minutes>59)                   // update timer
1089:     {                                 // update timer
1090:       minutes -= 60;                  // update timer
1091:       hours++;                        // update timer
1092:     }                                 // update timer
```

```
1093:     if (hours>23)                    // update timer
1094:       hours -=24;                    // update timer
1095:   }
1096:
1097:
1098:   EnableInterrupts;    // same as asm(cli)
1099:
1100:
1101:   for(;;) {
1102:     _FEED_COP(); /* feeds the dog */
1103:   }              /* loop forever */
1104:                  /* please make sure that you never leave main */
1105: }                /* else bad things may happen.*/
1106:
1107: //** Update Tempterature ************************************************************
1108: int updateTemperature(void)
1109: {
1110:   int returnTemperature;             // read voltage from thermistor input
1111:   ATDCTL2 = 0x80;                    // read voltage from thermistor input
1112:   delayby1ms(1);                     // read voltage from thermistor input
1113:   ATDCTL3 = 0x08;                    // read voltage from thermistor input
1114:   ATDCTL4 = 0x85;                    // read voltage from thermistor input
1115:   ATDCTL5 = 0x80;                    // read voltage from thermistor input
1116:   while(!(ATDSTAT0&0x80));           // read voltage from thermistor input
1117:   returnTemperature = ATDDR0;        // read voltage from thermistor input
1118:   ATDCTL2 = 0x00;                    // read voltage from thermistor input
1119:   return returnTemperature;          // read voltage from thermistor input
1120: }
1121:
1122: //** Display LCD ********************************************************************
1123: void displayLCD(int k, int Fan, int Heat, int HourS, int MinuteS, int HourE, int MinuteE)
1124: {
1125:   command(0x01);                     // clear LCD display
1126:   if (k==1)                          // print the time range
1127:   {                                  // print the time range
1128:     print('N');                     // print the time range
1129:     print('g');                     // print the time range
1130:     print('h');                     // print the time range
1131:     print('t');                     // print the time range
1132:   }                                  // print the time range
1133:   if (k==2)                          // print the time range
1134:   {                                  // print the time range
1135:     print('M');                     // print the time range
1136:     print('o');                     // print the time range
1137:     print('r');                     // print the time range
1138:     print('n');                     // print the time range
1139:   }                                  // print the time range
1140:   if (k==3)                          // print the time range
1141:   {                                  // print the time range
1142:     print('D');                     // print the time range
1143:     print('a');                     // print the time range
1144:     print('y');                     // print the time range
1145:     print(' ');                     // print the time range
1146:   }                                  // print the time range
1147:   if (k==4)                          // print the time range
1148:   {                                  // print the time range
1149:     print('E');                     // print the time range
1150:     print('v');                     // print the time range
1151:     print('e');                     // print the time range
1152:     print('n');                     // print the time range
1153:   }
1154:   print('H');                       // print the time range
1155:   print(':');                       // print the time range values
1156:   print(Heat/100+48);               // print the time range values
1157:   print((Heat%100)/10+48);          // print the time range values
1158:   print(Heat%10+48);                // print the time range values
1159:   print('F');                       // print the time range values
1160:   print('A');                       // print the time range values
1161:   print(':');                       // print the time range values
1162:   print(Fan/100+48);                // print the time range values
1163:   print((Fan%100)/10+48);           // print the time range values
1164:   print(Fan%10+48);                 // print the time range values
1165:   print('F');                       // print the time range values
1166:   command(0xc0);                    // print the time range values
1167:   if (HourS>12)                     // print the time range values
1168:   {                                 // print the time range values
1169:     print((HourS-12)/10+48);        // print the time range values
1170:     print((HourS-12)%10+48);        // print the time range values
```

```
1171:   }                                   // print the time range values
1172:   else                                // print the time range values
1173:   {                                   // print the time range values
1174:     print(HourS/10+48);               // print the time range values
1175:     print(HourS%10+48);               // print the time range values
1176:   }                                   // print the time range values
1177:   print(':');                         // print the time range values
1178:   print(MinuteS/10+48);               // print the time range values
1179:   print(MinuteS%10+48);               // print the time range values
1180:   if (HourS<12)                       // print the time range values
1181:   {                                   // print the time range values
1182:     print('A');                       // print the time range values
1183:   }                                   // print the time range values
1184:   else                                // print the time range values
1185:   {                                   // print the time range values
1186:     print('P');                       // print the time range values
1187:   }                                   // print the time range values
1188:   print('M');                         // print the time range values
1189:   print('-');                         // print the time range values
1190:   if (HourE>12)                       // print the time range values
1191:   {                                   // print the time range values
1192:     print((HourE-12)/10+48);          // print the time range values
1193:     print((HourE-12)%10+48);          // print the time range values
1194:   }                                   // print the time range values
1195:   else                                // print the time range values
1196:   {                                   // print the time range values
1197:     print(HourE/10+48);               // print the time range values
1198:     print(HourE%10+48);               // print the time range values
1199:   }                                   // print the time range values
1200:   print(':');                         // print the time range values
1201:   print(MinuteE/10+48);               // print the time range values
1202:   print(MinuteE%10+48);               // print the time range values
1203:   if (HourE<12)                       // print the time range values
1204:   {                                   // print the time range values
1205:     print('A');                       // print the time range values
1206:   }                                   // print the time range values
1207:   else                                // print the time range values
1208:   {                                   // print the time range values
1209:     print('P');                       // print the time range values
1210:   }                                   // print the time range values
1211:   print('M');                         // print the time range values
1212:   delayby1ms(60);                     // delay to display
1213: }
1214:
1215:
1216: //** Init LCD ************************************************************************
1217: void initLCD()
1218: {
1219:   delayby1ms(100);                    // delay
1220:   command(0x30);                      // init LCD
1221:   delayby1ms(100);                    // delay
1222:   command(0x30);                      // init LCD
1223:   delayby1ms(100);                    // delay
1224:   command(0x30);                      // init LCD
1225:   command(0x38);                      // init LCD
1226:   command(0x0C);                      // turn on display
1227:   command(0x01);                      // clear display
1228:   delayby1ms(2);                      // delay
1229: }
1230:
1231: //** Execute LCD Command ***************************************************************
1232: void command(int input)
1233: {
1234:   while(!(SPSR&0x20));                // Wait for register empty flag
1235:   SPDR = input;                       // output command cia SPI to SIPO
1236:   while(!(SPSR&0x80));                // wait
1237:   SPSR = SPSR|0x80;                   // load spdr
1238:   input = SPDR;                       // load spdr
1239:   asm(NOP);                           // wait
1240:   PORTM = PORTM&!RCK;                 // pulse RCK
1241:   asm(NOP);                           // wait
1242:   asm(NOP);                           // wait
1243:   PORTM = PORTM|RCK;                  // pulse RCK
1244:   PORTM = PORTM&!RS;                  // pulse RS
1245:   asm(NOP);                           // wait
1246:   asm(NOP);                           // wait
1247:   asm(NOP);                           // wait
1248:   PORTM = PORTM|ENABLE;               // pulse enable
```

```
1249:   asm(NOP);                       // wait
1250:   asm(NOP);                       // wait
1251:   PORTM = PORTM&!ENABLE;           // pulse enable
1252:   delayby1ms(5);                  // delay
1253: }
1254:
1255: //** Print to LCD ************************************************************
1256: void print(char charToPrint)
1257: {
1258:   while(!(SPSR&0x20));              // Wait for register empty flag
1259:   SPDR = charToPrint;               // output command cia SPI to SIPO
1260:   while(!(SPSR&0x80));              // Wait for register empty flag
1261:   charToPrint = SPDR;               // load spdr
1262:   asm(NOP);                         // wait
1263:   PORTM = PORTM&!RCK;               // pulse rck
1264:   asm(NOP);                         // wait
1265:   asm(NOP);                         // wait
1266:   PORTM = PORTM|RCK;                // pulse rck
1267:   PORTM = PORTM|RS;                 // pulse rs
1268:   asm(NOP);                         // wait
1269:   asm(NOP);                         // wait
1270:   PORTM = PORTM|ENABLE;             // pulse enable
1271:   asm(NOP);                         // wait
1272:   asm(NOP);                         // wait
1273:   PORTM = PORTM&!ENABLE;            // pulse enable
1274:   delayby1ms(2);                    // wait
1275: }
1276:
1277: /*****************************************************************************/
1278: /*                                                                           */
1279: /* The getkey functions gets the key value from a 4X4 matrix keypad connected    */
1280: /* PortT. Rows (0,1,2,3) = P4,P5,P6,P7                                       */
1281: /*     Columns (0,1,2,3) = P0,P1,P2,P3                                       */
1282: /* The strategy used here is nessted if -else statements and is similar to what  */
1283: /* we did in assembly language. There are more efficient and elegant strategies. */
1284: /*                                                                           */
1285: /*****************************************************************************/
1286: //** Grab Key ***************************************************************
1287: char getkey(void)
1288: {                       // We test the keys in sequence - row 0 columns 0,1,2,3
1289:                         // row 2 columns 0,1,2,3 etc. until we have checked
1290:    char keyX;           // all of the keys. If a key is pressed then we save the
1291:                         // value in keyX and jump down to return without
1292:                         // checking any more keys.  Note that there many
1293:                         // more ways to do this.
1294:
1295:    PORTT = 0x00;        // clear portT
1296:    asm(NOP);            // short wait times with assembler NOP
1297:    PORTT |= 0x10;       // PORTT = PORTT | 0x10; OR PORTT with $10. ie. set row 0 (PT4) High
1298:    asm(nop);
1299:    asm(nop);
1300:
1301:    if(PORTT & BIT0)     // AND PORT with 0x01 and check if ans is 1 (TRUE). ie. Check column 0 for HIGH. If
1302:            keyX = 1;    // then set keyX to 1 and jump to return.
1303:    else if(PORTT & BIT1)  // Check column 1
1304:            keyX = 2;
1305:    else if(PORTT & BIT2)  // Check column 2
1306:            keyX = 3;
1307:    else if(PORTT & BIT3)  // Check column 3
1308:            keyX = 10;
1309:    else {
1310:           PORTT = 0x00;     // Clear PortT and start on row 1
1311:           PORTT |= 0x20;    //  Set row 1 High
1312:           asm(nop);
1313:           asm(nop);
1314:
1315:           if(PORTT & BIT0) // Check column 0 etc., etc.
1316:            keyX = 4;                   // etc., etc.
1317:           else if(PORTT & BIT1)        // etc., etc.
1318:            keyX = 5;                   // etc., etc.
1319:           else if(PORTT & BIT2)        // etc., etc.
1320:            keyX = 6;                   // etc., etc.
1321:           else if(PORTT & BIT3)        // etc., etc.
1322:            keyX = 11;                  // etc., etc.
1323:           else {                       // etc., etc.
1324:               PORTT = 0x00;            // etc., etc.
1325:               PORTT |= 0x40; // row 2 High
1326:               asm(nop);                // Check columns etc., etc.
```

```
1327:               asm(nop);               // etc., etc.
1328:                                       // etc., etc.
1329:               if(PORTT & BIT0)        // etc., etc.
1330:                 keyX = 7;             // etc., etc.
1331:               else if(PORTT & BIT1)   // etc., etc.
1332:                 keyX = 8;             // etc., etc.
1333:               else if(PORTT & BIT2)   // etc., etc.
1334:                 keyX = 9;             // etc., etc.
1335:               else if(PORTT & BIT3)   // etc., etc.
1336:                 keyX = 12;            // etc., etc.
1337:               else {                  // etc., etc.
1338:                 PORTT = 0x00;         // etc., etc.
1339:                 PORTT |= 0x80; // row 3 High
1340:                 asm(nop);             // Check columns etc., etc.
1341:                 asm(nop);             // etc., etc.
1342:                                       // etc., etc.
1343:                 if(PORTT & BIT0)      // etc., etc.
1344:                   keyX = 0 ;          // etc., etc.
1345:                 else if(PORTT & BIT1) // etc., etc.
1346:                   keyX = 15;          // etc., etc.
1347:                 else if(PORTT & BIT2) // etc., etc.
1348:                   keyX = 14;          // etc., etc.
1349:                 else if(PORTT & BIT3) // etc., etc.
1350:                   keyX = 13;          // etc., etc.
1351:                 else            // if we get to here ==> no key pressed
1352:                   keyX = 0x1f;  // nokey signal
1353:               }
1354:           }
1355:     }
1356:     if (keyX<16)
1357:     {
1358:        keyrelease();              // wait for key release
1359:        delayby1ms(100);           // wait a bit - 100 X 1ms = 0.1 sec
1360:     }
1361:     return (keyX);                // return the key value
1362: }
1363:
1364: /*************************************************************/
1365: /*                                                           */
1366: /* Key release routine. Check each coulmn bit. If HIGH wait */
1367: /* until it goes LOW to break out of the while statement.   */
1368: /* Note that we are reading the input register of PortT     */
1369: /* which is at address $0241 and is called (here) PORTTi    */
1370: /*                                                           */
1371: /*************************************************************/
1372: void keyrelease(void)
1373: {           //PORTT = 0xf0             // Set all rows high (not needed here. Why?)
1374:             while((PORTTi & 0x01));    // if column 0 is HIGH wait here until LOW
1375:             while((PORTTi & 0x02));    // if column 1 is HIGH wait here until LOW
1376:             while((PORTTi & 0x04));    // if column 2 is HIGH wait here until LOW
1377:             while((PORTTi & 0x08));    // if column 3 is HIGH wait here until LOW
1378: //            return(0);               // return needs a value so use 0
1379: }
1380:
1381: /*********************************************************************/
1382: /* The following function creates a time delay which is equal to the */
1383: /* multiple of 1ms. The value passed in k specifies the number of    */
1384: /* milliseconds to be delayed.                                       */
1385: /*********************************************************************/
1386: void delayby1ms(int k)
1387: {
1388:   int ix;
1389:   TSCR1 = 0x90;       /* enable TCNT and fast timer flag clear */
1390:   TSCR2 = 0x06;       /* disable timer interrupt, set prescaler to 64 */
1391:   TIOS  |= BIT0;      /* enable OC0 */
1392:   TFLG1 &= BIT0;      /* clear timer flag OC0F*/
1393:   TC0 = TCNT + 375;
1394:
1395:   for(ix = 0; ix < k; ix++)
1396:   {
1397:     while(!(TFLG1 & BIT0));   // wait for timer
1398:     TC0 += 375;               // wait for time
1399:   }
1400:
1401:   TIOS  &= (~BIT0);  /* disable OC0 */
1402: }
1403:
1404: /*********************************************************************/
```

```
1405: /* This function enables PLL and use an 8-MHz crystal oscillator to */
1406: /* generate 24-MHz E clock. Same as done in assembler.              */
1407: /*******************************************************************/
1408: void SetClk8(void)
1409: {
1410:   asm(sei);          // turn of interrupts
1411:   CLKSEL  &= PLLSEL; // disengage PLL from system
1412:   SYNR    = 0x02;    // set SYSCLK to 24 MHz from a 4-MHz oscillator
1413:   REFDV   = 0;       //
1414:   PLLCTL  = 0x40;    // turn on PLL, set automatic
1415:   while(!(CRGFLG&LOCK)); // wait for HIGN on LOCK bit at address CRGFLG
1416:   asm(nop);          // very short delays
1417:   asm(nop);
1418:   CLKSEL  |= PLLSEL; // clock derived from PLL
1419:   asm(cli);          // turn on interrups
1420: }
1421:
1422: //** Interrupts ***************************************************************
1423: interrupt void irqISR(void)
1424: {
1425:   if (heatOverride<2)                // toggle heat settings
1426:     heatOverride++;                  // toggle heat settings
1427:   else heatOverride = 0;             // toggle heat settings
1428: }
1429:
1430: interrupt void xirqISR(void)
1431: {
1432:   if (fanOverride<2)                 // toggle AC settings
1433:     fanOverride++;                   // toggle AC settings
1434:   else fanOverride = 0;              // toggle AC settings
1435: }
```