Microwave Simulation

Samuel Tregea

CSC 202-105

Professor George Fazekas

May 16, 2018

Table of Contents

Section 1: Description

This project is a ***simulated*** microwave. It can perform as a regular microwave would (hence the term "simulated"), however since this cannot cook, this project was meant solely for the use of demonstrated purposes and proof of concept.

1.0) The project implements all the following components:

1) Thunderbird
2) LCD display
3) Servo Motor
4) Stepper Motor
5) 7-Segmented Display
6) A/D Converter
    - Thermometer
7) Hex Keypad
8) Active Buzzer
9) Two External LED's
    - One Red LED
    - One Green LED
10) Hardware Interrupt
11) Software Interrupt
12) External Power Supply

Section 1: Wiring

For the Microwave to function properly, all hardware must be wired properly. If not properly wired, components with either overheat, become dysfunctional, or will short circuit the entire board in which all cases will result in an unresponsive output.

1.1) **Thunderbird**

Place the thunderbird anywhere on the bread board, however, it is preferred to have it be placed near the top edge by VCC and GND for the sake of wiring and functional purposes.

1.2) **Liquid Crystal Display (LCD)**

The LCD display will be used to tell the user which mode they are currently in, when the microwave finished cooking, and will display the room temperature when the function lockServo() is called.

- GND (Black Wire) to GND
- VCC (Red Wire) to VCC
- SDA (White Wire) to PJ6
- SCL (Brown Wire) to PJ7

1.3) **Servo Motor**

The Servo Motor is being used as a "lock," since it will both open and close, depending on the nature of the simulation.

- Orange Wire to PP7
- Red Wire to VCC
- Brown Wire to GND

1.4) **Step Motor**

In most microwaves there is a plate that rotates during the entirety of the cook time. The Step Motor will simulate that very plate in the project.

First:

- Connect the Stepper Motor to its circuit

Second, place the following wires accordingly:

- D (Brown Wire) to PT0
- C (White Wire) to T1
- B (Red Wire) to T2
- A (Black Wire) to T3
- GND to GND
- VCC to VCC

Section 1: Wiring (continued)

1.5) **7-Segment Display**
The 7-Segment Display will be implemented as a timer. A user will input a given (or preset) time from the Hex Keypad, and then then will count down to 0.

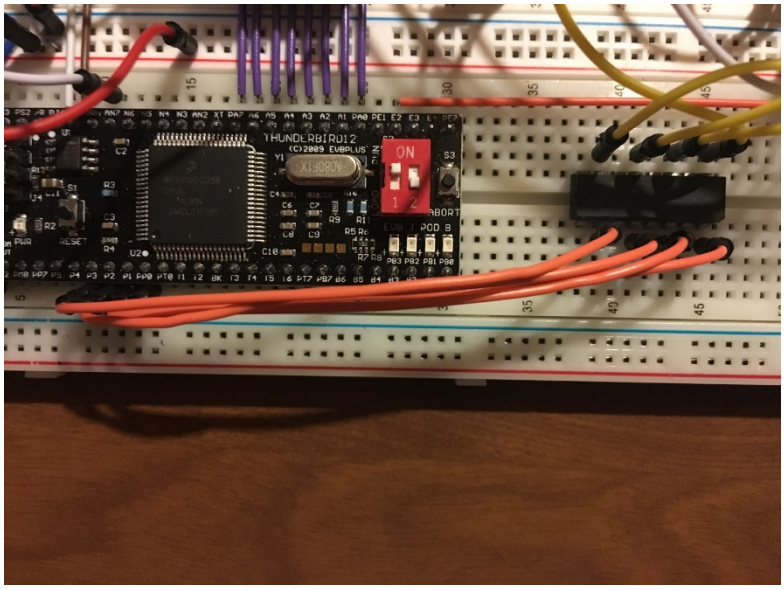
First:

- Place the 74HC595 shift register on the board, preferably over the crevice on the center of the board.

Second:

- Pin wires from P3, P2, P1, and PP0 to the 74HC595, then you will need additional wires to jump to the actual display itself. To place the jump wires, just pin them on the opposite side of the register (in accordance to the previous placement).

Since the wiring can be confusing at the start, provided below is a picture to help getting started. Proper placement should look like this:



Third:

- Jump P3 to Pin 6 of 7 segment display
- Jump P2 to Pin 8
- Jump P1 to Pin 9
- Jump PP0 to Pin 12

Lastly:

- B6 to Pin 5
- B5 to Pin 10
- B4 to Pin 1
- B3 to Pin 2
- B2 to Pin 4
- B1 to Pin 7
- PB0 to Pin 11

Note: Pin 3 is not being used

Section 1: Wiring (continued)

1.6) **A/D Converter: Thermometer**
The thermometer's purpose in this project is to display the current room temperature.
There is a function displayTemp() that will be called whenever lockServo() is called, and will display the current room temperature on the LCD.

First:
- Have flat side of LM35 face towards you.

Second (from left to right):
- VCC to Pin1
- AN7 to Pin2
- GND to Pin3
- 10k Ohm resistor to between GND wire and Pin3 and to VCC

1.7) **Hex Keypad**
The Hex Keypad plays one of the most important roles in the project, since it takes in all of the user's input. The user can enter digits 0-9 to create their own count down time, or they can enter A-D for preset times. To start the microwave, the user must hit the "*" key.

From left to right:
- PA7 to Pin1
- A6 to Pin2
- A5 to Pin3
- A4 to Pin4
- A3 to Pin5
- A2 to Pin6
- A1 to Pin7
- PA0 to Pin8

1.8) **Active Buzzer**
The Active Buzzer plays an important role, but rather in detail. Whenever a user hits a key on the Hex Keypad, the buzzer will beep, also, the buzzer serves as an alarm when the timer reaches 0.

First:
- Place the buzzer to wherever you would like it to be on the board.

Second, pin:
- E3 to Anode (+)
- GND to Cathode (-)

Section 1: Wiring (continued)

1.9.) **External LED's**
The two LED's serve as a display as to the current state of the microwave. The red LED means that the microwave is locked and is cooking, and the green LED means the microwave is open and is ready for input. For proper functionality, two 220k Ohm resistors will be used.

1.9.1) **Red LED**
- M1 to 220k Ohm resistor
- 220k Ohm resistor to Anode (+) of red LED
- GND to Cathode (-)

1.9.2) **Green LED**
- M2 to 220k Ohm resistor
- 220k Ohm resistor to Anode (+) of green LED
- GND to Cathode (-)

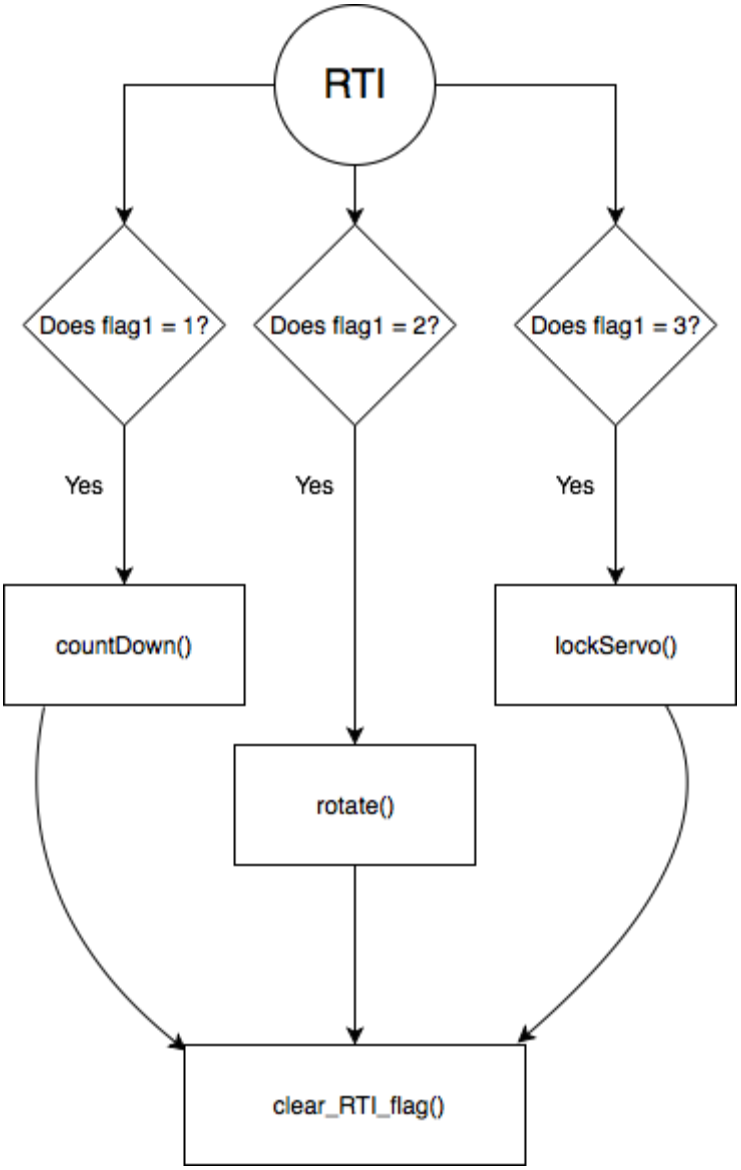
1.10 + 1.11) **Hardware Interrupt + Software Interrupt**
Both the Hardware and Software Interrupts are implemented in the coding aspect of the project, so no wiring is required.

1.12) **External Power Supply**
Since the Servo Motor, Stepper Motor, and the Liquid Crystal Display consume so much power, and external power source is required for the proper functionality of the microwave.

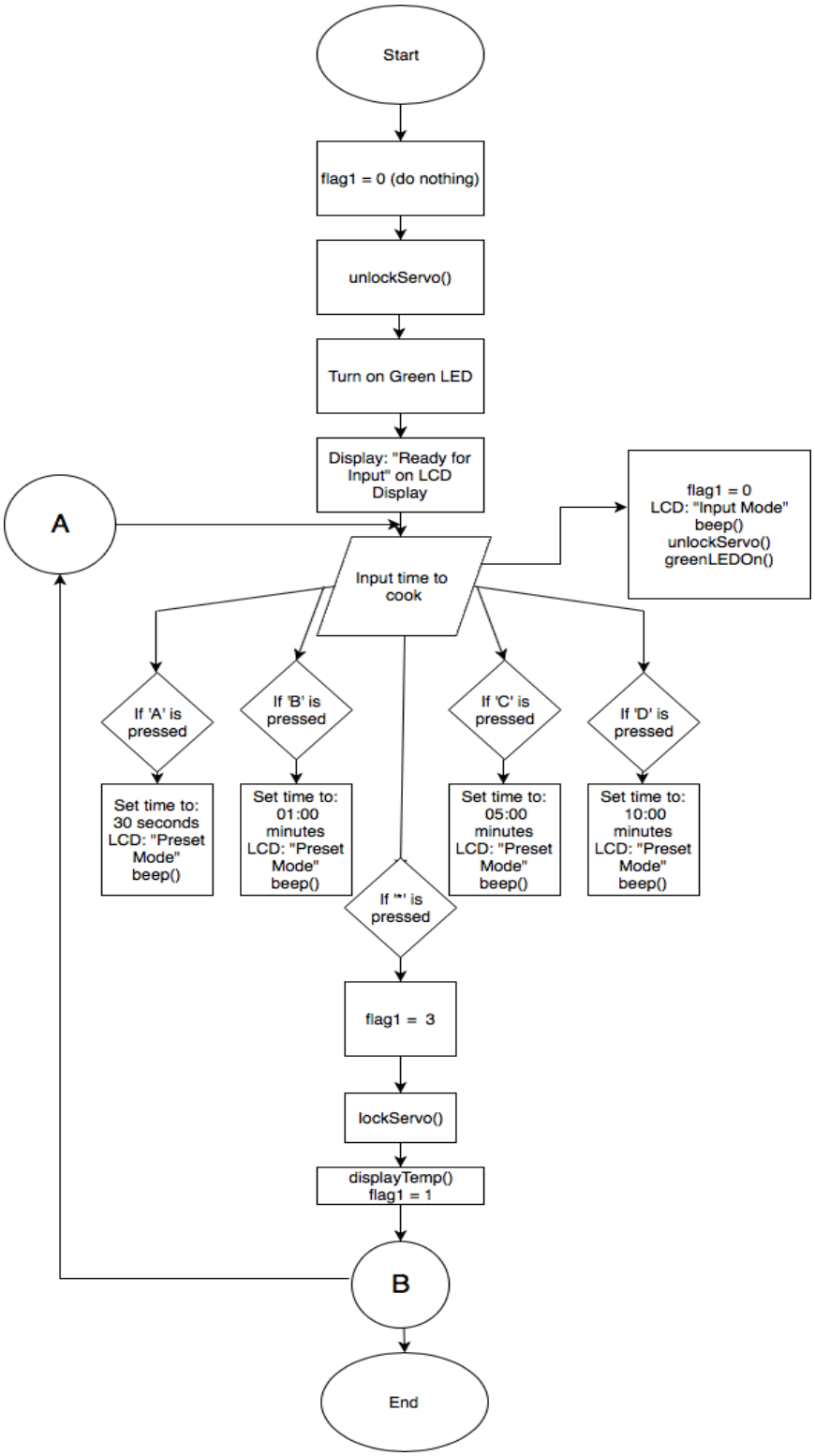
- External Power to VCC
- External Ground to GND

Section 2: Flow Chart

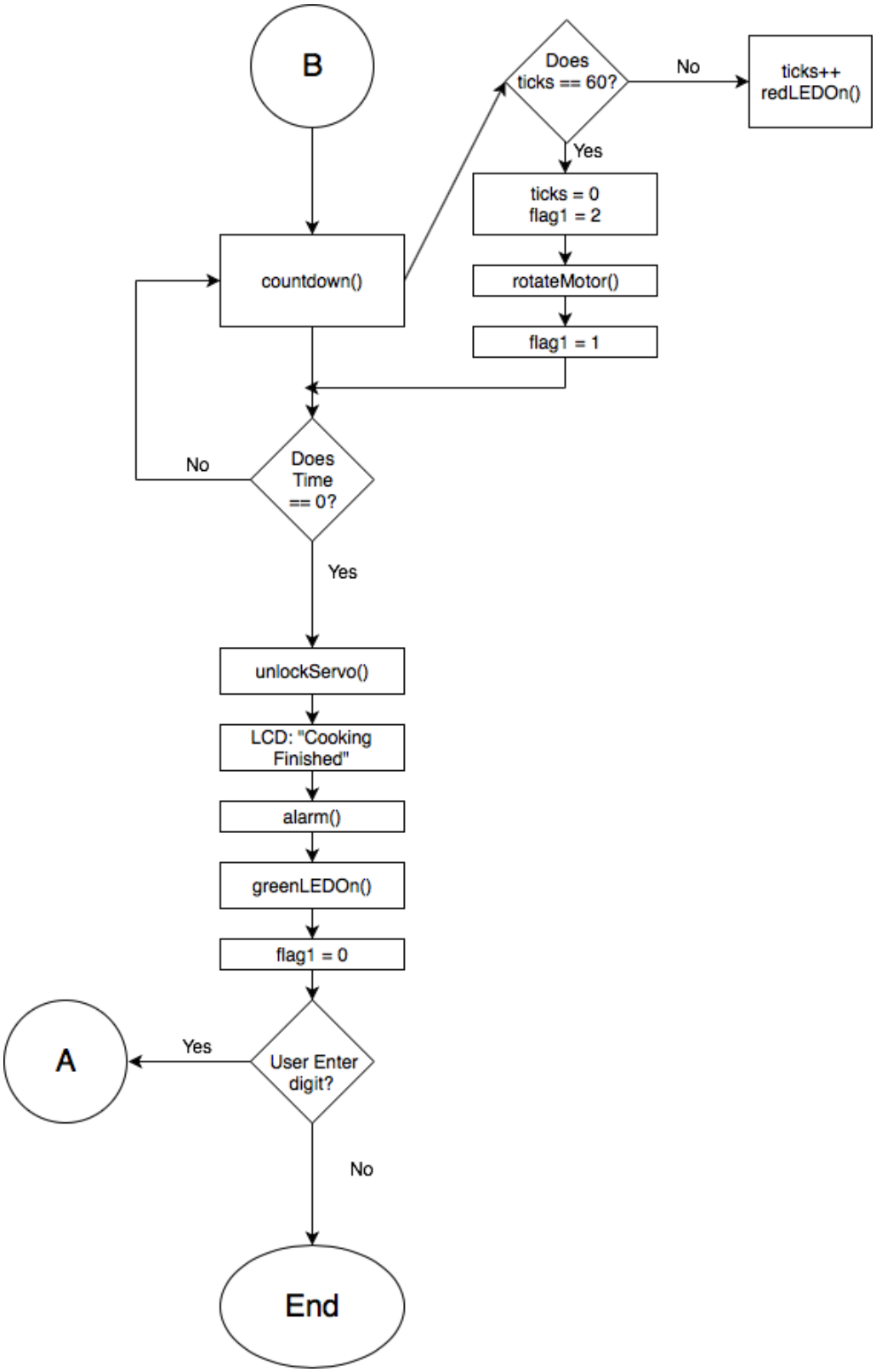
For the proper design of the program, this flowchart was created prior to the actual programming implementation to ensure proper logic flow and to minimize any errors that would ruin the program.



Section 2: Flow Chart (continued)



Section 2: Flow Chart (continued)



Section 3: Code

Following the flow chart, the code is the final implementation of the project. To ensure each component worked, a function was designed to implement the hardware properly and efficiently.

```
1   /*
2    Name: Samuel Tregea
3    Professor: George Fazekas
4    Date: May 16, 2018
5    Description: Final Project for CSC-202
6
7    This project is a simulation of a microwave that implements the following:
8
9    1.) Thunderbird
10   2.) LCD display
11   3.) Servo Motor
12   4.) Step Motor
13   5.) 7-Segmented Display
14   6.) A/D Converter (Thermometer)
15   7.) Hex Keypad
16   8.) Passive Buzzer
17   9.) 2 LED's
18   10.) Hardware Interrupt
19   11.) Software Interrupt
20
21   Most of the code has been placed into void methods which are called in the interrupt when a
22   "flag" is raised, allowing the multiplexing of the 7-Segmented Display to occur without raising any
23  errors.
24
25   Flags:
26   0 = pause
27   1 = countdown()
28   2 = rotate()
29   3 = lockServo()
30
31   LEDS:
32
33   Red - means microwave is locked and cannot be used
34   Green - microwave is open and is ready for input.
35
36   */
37
38  #include <hidef.h>      /* common defines and macros */
39  #include <mc9s12dg256.h>     /* derivative information */
40  #pragma LINK_INFO DERIVATIVE "mc9s12dg256b"
41
42  #include "main_asm.h" /* interface to the assembly module */
43
44  /*
```

```
45      Function Declarations
46     */
47    void countDown(void);
48    void lockServo(void);
49    void unlockServo(void);
50    void displayTemp(void);
51    void beep(void);
52    void alarm(void);
53    void rotate(void);
54    void redLEDOn(void);
55    void redLEDOff(void);
56    void greenLEDOn(void);
57    void greenLEDOff(void);
58
59    //100 ticks ~ 1 second, may change due to several millisecond delays
60     int ticks = 0;
61
62     int i, j; //Used for for-loops
63     int flag1; //Flag that's to be raised and used in the software interrupt
64     int temperature; //Used to display the temperature
65     int width; //used for the Servo methods
66
67     char c;
68     char *message; //String variable
69     int digits_size = 3;
70
71     int digits[] = {
72       0,0,0,0
73     };
74
75     unsigned int arry[] = {0x9,0x3,0x6,0xC};
76
77
78
79    void interrupt 7 handle(){ //real-time interrupt
80
81     if(flag1 == 1){
82       countDown();
83     }
84     if(flag1 == 2){
85       rotate();
86     }
87     if(flag1 == 3){
88       lockServo();
89     }
90
91     clear_RTI_flag();
92    }
93
94
95
```

```
96  void main(void){
97
98    PLL_init();    //set clock frequency to 24 MHz
99    lcd_init();    //enable LCD
100   SCI0_int_init(9600);  //initialize SCI0 at 9600 baud
101   ad0_enable();    //enable a a/d converter 0
102   keypad_enable();//enable keypad
103
104   DDRB = 0xff;  //Enabling PortB for 7 segmented display
105   DDRP = 0xff;  //Enabling Port P for 7 segmented display
106   PTP = 0xff;   //Port P, enabling one segment, use '0' to use all leds (uses negative logic)
107   DDRT = 0xf;   //Enabling Port T for use of alarm
108   RTI_init();
109   servo76_init();      //enable pwm1 for servo
110
111   unlockServo();   //"opening" the microwave, allowing it to be ready for use
112   greenLEDOn();
113   clear_lcd();
114   set_lcd_addr(0x00);    //display on first row LCD
115   message = "Ready for Input";
116   type_lcd(message);
117
118   while(1){
119
120     //Preset time to 30 seconds
121     if(keyscan() == 10){
122       wait_keyup();
123       clear_lcd();
124       set_lcd_addr(0x00);//display on first row LCD
125       message = "Preset Mode";
126       type_lcd(message);
127       beep();
128
129       unlockServo();
130       greenLEDOn();
131
132       digits[0] = 0;
133       digits[1] = 0;
134       digits[2] = 3;
135       digits[3] = 0;    }
136     //Preset time to 01:00 minutes
137     if(keyscan() == 11){
138       wait_keyup();
139       clear_lcd();
140       set_lcd_addr(0x00);//display on first row LCD
141       message = "Preset Mode";
142       type_lcd(message);
143       beep();
144
145       unlockServo();
146       greenLEDOn();
```

```
147
148         digits[0] = 0;
149         digits[1] = 1;
150         digits[2] = 0;
151         digits[3] = 0;
152       }
153       //Preset time to 05:00 minutes
154       if(keyscan() == 12)
155       {
156         wait_keyup();
157         clear_lcd();
158         set_lcd_addr(0x00);//display on first row LCD
159         message = "Preset Mode";
160         type_lcd(message);
161         beep();
162
163         unlockServo();
164         greenLEDOn();
165
166         digits[0] = 0;
167         digits[1] = 5;
168         digits[2] = 0;
169         digits[3] = 0;
170       }
171       //Preset time to 10:00 minutes
172       if(keyscan() == 13)
173       {
174         wait_keyup();
175         clear_lcd();
176         set_lcd_addr(0x00);     //display on first row LCD
177         message = "Preset Mode";
178         type_lcd(message);
179         beep();
180
181         unlockServo();
182         greenLEDOn();
183
184         digits[0] = 1;
185         digits[1] = 0;
186         digits[2] = 0;
187         digits[3] = 0;
188       }
189       //raising a flag for when the user presses"*"
190       //locks servo, then counts down
191       if(keyscan() == 14)
192       {
193          flag1 = 3; //got to lockServo();
194       }
195       //resetting the flag
196       if(keyscan() != 16 && keyscan() != 10 && keyscan() != 11 && keyscan() != 12 && keyscan() != 13
197    && keyscan() != 14 && keyscan() != 15)
```

```
198         {
199           flag1 = 0;//pause
200 
201           c = getkey();
202           wait_keyup();
203           clear_lcd();
204           set_lcd_addr(0x00);//display on first row LCD
205           message = "Input Mode";
206           type_lcd(message);
207           beep();
208 
209           unlockServo();
210           greenLEDOn();
211 
212           //moving the digits to the left
213           //using negative logic
214           for(j = 0; j < 3; j++)
215           {
216             digits[j] = digits[j+1];
217           }
218 
219           //setting the rightmost display to the keyscan
220           digits[3] = c;
221 
222         }
223 
224         /*
225           multiplexing
226         */
227         for(i = 0; i<4; i++)
228         {
229           seg7dec(digits[i],i);
230           ms_delay(1);
231         }
232 
233     }//while
234 
235       asm swi;//returns to the controller and resets it
236 
237     }//main
238 
239 
240     //counts down
241     void countDown(void){
242         ticks++;
243         redLEDOn();
244       if(ticks == 60){
245         ticks = 0;  //resetting ticks
246       if(digits[3] > 0)
247       {
248           digits[3]--;
```

```
249       }//if digits[3]
250       else{
251         if(digits[2] > 0){
252           digits [3] = 9;
253           digits[2]--;
254         }
255         else{
256           if(digits[1] > 0){
257             digits[3] = 9;
258             digits[2] = 5;
259             digits[1]--;
260           } else{
261             if(digits[0] > 0){
262              digits[3] = 9;
263              digits[2] = 5;
264              digits[1] = 9;
265              digits[0]--;
266             } else{
267               digits[0] = 0;
268               digits[1] = 0;
269               digits[2] = 0;
270               digits[3] = 0;
271               unlockServo();//opening the microwave
272               clear_lcd();
273               set_lcd_addr(0x00);//display on first row LCD
274               message = "Cooking";
275               type_lcd(message);
276               set_lcd_addr(0x40);//display on second row LCD
277               message = "Finished";
278               type_lcd(message);
279               alarm();
280               greenLEDOn();//green = microwave is ready to be used, turns off red LED
281               flag1 = 0;//pause flag when all segments are 0
282             }
283
284           }
285         }
286       }
287      }//if ticks
288        else{//else, continue rotating motor
289
290        /*
291         multiplexing here prevents an issue in
292         which the display would continuously blink.
293         */
294        for(i = 0; i<4; i++)
295        {
296           seg7dec(digits[i],i);
297           ms_delay(1);
298        }
299        flag1 = 2; //go to rotate()
```

```
300
301       }
302     }//countDown
303
304     /*
305       Turns the servo motor to simulate a "lock" while the
306       microwave is turned on.
307     */
308     void lockServo(void){
309
310         for(width = 3000; width <= 5500; width = width + 5){
311          set_servo76(width);   //move servo from 3000 to 5500
312         }
313
314         displayTemp();
315         flag1 = 1;//go to countDown()
316     }
317
318     /*
319      "unlocks" the servo motor
320     */
321     void unlockServo(void){
322
323         for(width = 5500; width >= 3000; width = width - 5){
324           set_servo76(width);   //move servo from 5500 to 3000
325         }
326
327          flag1 = 0;//do nothing
328     }
329
330     /*
331       Displays the current temperature of the microwave
332     */
333     void displayTemp(void){
334
335         temperature = ad0conv(7);       //read pot on channel 7
336         set_lcd_addr(0x00);     //display on first row LCD
337         message = "Current Temp:";
338         type_lcd(message);
339         set_lcd_addr(0x40);     //display on 2nd row LCD
340         write_int_lcd(temperature);     //write value in field
341         message = " degrees F";
342         type_lcd(message);
343      }
344
345     /*
346         Used to create a beeping sound when a user hits a
347         button on the hex keypad
348      */
349     void beep(void){
350      DDRE = 0xff;
```

```
351     PORTE = 0xff;
352     ms_delay(5);
353     PORTE = 0x00;
354
355     }
356
357     /*
358        called when countdown() finishes
359
360        Beeps buzzer 3 times along with a flashing external LED
361      */
362     void alarm(void){
363      DDRE = 0xff;
364      PORTE = 0xff;
365      redLEDOn();
366      ms_delay(450);//firstbeep
367      PORTE = 0x00;
368      redLEDOff();
369      ms_delay(450);
370      PORTE = 0xff; //second beep
371      redLEDOn();
372      ms_delay(450);
373      PORTE = 0x00;
374      redLEDOff();
375      ms_delay(450);
376      PORTE = 0xff;
377      redLEDOn();
378      ms_delay(450);//firstbeep
379      PORTE = 0x00;
380      redLEDOff();
381
382     }
383
384     /*
385        Rotates the Step Motor
386      */
387     void rotate(void){
388
389
390        for(i = 0; i<4;i++){
391        PTT = arry[i];
392        ms_delay(2);
393       }
394
395        /*
396         interrupt command
397          */
398       flag1 = 1;//go to countDown()
399
400     }
401
```

```
402  /*
403     Turns on the red LED
404   */
405  void redLEDOn(void){
406   DDRM = 0x02; //m1
407   PTM = 0x02;
408  }
409
410  /*
411     Turns off the red LED
412   */
413  void redLEDOff(void){
414   DDRM = 0x02;
415   PTM = 0x00;
416  }
417
418  /*
419     Turns on the green LED
420   */
421  void greenLEDOn(void){
422   DDRM = 0x04; //m2
423   PTM = 0x04;
424  }
425
426  /*
427     Turns off the green LED
428   */
429  void greenLEDOff(void){
430   DDRM = 0x04;
431   PTM = 0x00;
432  }
```