2016

E10 Microwave

DESIGN MANUAL

ANNIE ZHANG (Z3459054) AND STEPHEN CHUNG (Z3463108)

Contents

System Flow Control

The microwave system is structured by four modes: entry mode, running mode, paused mode and finished mode. Figure 1 illustrates the flow of the entire system. Entry mode allows the user to configure time and power settings. The food is cooked in running mode according to the time input and power level. The user is able to adjust the time whilst in running mode as well as enter paused mode. The food is finished cooking when the time reaches zero and enters finished mode prompting the user to remove the food by opening the door. After closing the door, the system will return back to entry mode. If the door is opened at any time, the system will refuse any input until the door is closed. The top-most LED will be lit when the door is opened. The door is closed by default. The * button on the keypad is regarded as the start button and the # button is regarded as the stop button. In each mode, specified input buttons are unique to each mode and any unspecified input buttons will do nothing. Keypad button presses will have a 250ms beep to indicate it being pressed. When no keypad buttons have been pressed for 10 seconds whilst in entry mode, running mode or paused mode, the display backlight will fade off over 500ms. The display backlight will be turned back on when a keypad button is pressed again, and will always be on in finished mode.

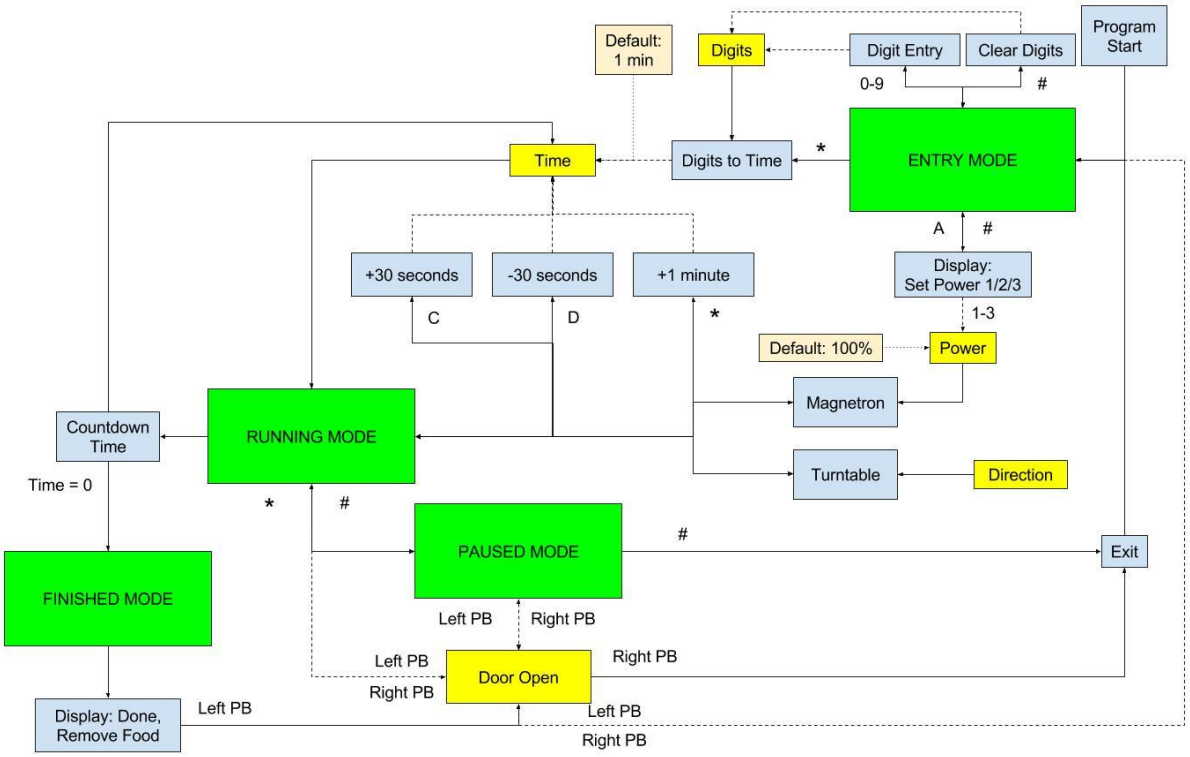

Figure 1: System Flow Diagram

Data Structures

The data structures used in the project comprise of a number of variables stored in data memory. All variables are 1 byte in size.

Firstly, it uses variables to obtain input from the keypad, namely row, col, rmask and cmask. Row and col allow specify which specific key is being pressed, whereas rmask and cmask act as masks for extracting the value of the pin of the key that is being pressed. A currentInput variable keeps track of the latest key being read from the keypad, to be handled later. The reading of the keypad and handling its input occurs in the main program loop.

A variable doorIsOpen is used to specify whether the door to the microwave is currently open or closed. Interrupts and timers will ignore any input if this variable is set to 1, meaning that the door is open. Note that an open door state is not interpreted as a separate mode, but rather just as a state where no input can be made. A mode variable keeps track of which mode the device is currently operating in. Entry mode refers to 1, power selection is 2, running mode is 3, paused mode is 4 and finished mode is 5. The variable power specifies which power level the device is currently set to. It is set to 100 for 100%, 50 for 50% and 25 for 25%.

In entry mode, four variables, a, b, c and d, are used to keep track of the digits currently entered into the program, corresponding to the thousands, hundreds, tens and ones digit respectively. As digits are entered in entry mode, these variables will be populated accordingly. These digits are reset every time the program re-enters entry mode from paused or finished mode.

When the program is entering running mode from entry mode, these four variables a, b, c and d will have their representative time converted into minutes and seconds, to be stored into two registers, mins and secs. Since mins and secs are used regularly in running mode for displaying the time, they are assigned registers rather than a space in data memory. Running mode uses these registers rather than the four variables.

Once the program enters running mode, a variable functionsRunning will be set to be true. This variable refers to the fact that functions, such as the magnetron and turntable are on. The variable resets every time operation is stopped completely, such as when going to finish or back to entry mode. A number of counters are used to store the number of counts of a specific time period. The counter counts the raw signal from the prescaled clock frequency. The counter250 specifies how many 250 ms has passed from the last time it was reset. The other 3 timers have a similar purpose. Two variables, magnetronCounter and magnetronRunning, are

used to determine the frequency and behavior of the magnetron. The variable magnetronCounter determines at which time intervals it should turn on or off, while magnetronRunning determines whether it should turn on or off. The index variable specifies which character will be displayed as the turntable on the LCD screen. Further, the CCWrotation variable determines which direction the turntable is currently rotating. This direction alternates every time it re-enters running mode, from any other mode. When CCWrotation is 1, the turntable is rotating counter-clockwise, and vice versa.

The playFinishSound, finishSoundCounter, nFinishSounds and finishSoundIsOn variables implement the feature where a sound will play for 1 second long, 3 times when the microwave has finished cooking. playFinishSound specifies whether the sound is to be playing or not. finishSoundCounter is a timer for 1 second. nFinishSounds specifies how many seconds have passed already. finishSoundIsOn specifies whether the sound is currently playing or not. The beepCounter and keyPressed variables implement the 250ms on-key-press beep that will be played at all times. When keyPressed is set to be 1, after a key is pressed, timer 2 will play a sound for 250ms. BeepCounter is used to count for 250ms. keyPressed is subsequently set to 0 after 250ms has passed.

Two macros, getVar and setVar, allow access of these variables easily.

```
; getVar <VARIABLE LABEL> <REGISTER>
.macro getVar
     ldi XH, high(@0)
     ldi XL, low(@0)
     ld @1, X
.endmacro

; setVar <VARIABLE LABEL> <REGISTER>
.macro setVar
     ldi XH, high(@0)
     ldi XL, low(@0)
     st X, @1
.endmacro
```

Algorithms

The program initializes at RESET, where various ports, timers and LCD are initialized, variables are set to their initial value and mode is placed in entry mode. The main program loop reads keypad input and handles the button being pressed depending on the mode and whether the door is open or not.

The program comprises of the following interrupts and timers. INT0 is used as an interrupt to trigger the event where the door was closed, while INT1 is used to trigger the door being opened. TIMER0 is used to handle running mode. It decrements the time shown on the display, updates the turntable, and handles the magnetron.

```
.cseg
.org 0x0000
     jmp RESET
.org INT0addr
     jmp CLOSE_DOOR
.org INT1addr
     jmp OPEN_DOOR
.org OVF0addr
     jmp TIMER
```

Handling keypad input

At convert_end, the keypad input has been interpreted and the input must now be handled. Firstly, the program checks to see if the door is open. If it is, the input is not handled as it jumps to end and restarts the main loop.

```
; Logic:
     ; if open == true
     ;    restart
```

Otherwise, the program will clear the display and check which mode it is currently running in. Then it will branch to that section of code that handles input according to the mode.

```
; Branch
     ; if mode == entry
```

```
; elsif mode == powerAdjustment
; elsif mode == running
; elsif mode == paused
; elsif mode == finished
```

At the end of each mode, the program will check for debouncing, and, once the key does not register as being pressed anymore, it will jump back to the start of the loop and start scanning for a keypad press again.

Entry mode

If the mode is entry mode, then the program will allow input into variables a, b, c and d, the time variables, if a digit was pressed, and it will clear the time if a hash was pressed. If A was pressed, it will switch the mode over to the power adjustment mode, while if an asterisk was pressed, the running mode will start. Appropriate text will be displayed on the LCD when the button is pressed.

```
; Entry mode
    ; if 0-9
    ;    insert digit to the time
    ;    display updated time
    ; if #
    ;    clear time
    ;    display cleared time
    ; if A
    ;    mode = powerAdjustment
    ;    display "Set Power 1/2/3"
    ; if *
    ;    if time entered
    ;         mode = running
    ;         display time
    ;    else
    ;         mode = running
```

```
;          set time = 1 min
;          display 1 min of time
```

Power selection mode

In power selection mode, the keys 1, 2 and 3 update the power variable accordingly. The # button will cause mode to be shifted back to entry, with the digits a, b, c and d re-displayed on the LCD.

```
; Power Selection Mode
     ; elsif mode == powerAdjustment
     ;    if 1
     ;         power = 100
     ;    if 2
     ;         power = 50
     ;    if 3
     ;         power = 25
     ;    if #
     ;         mode = entry
     ;         display entryStuff
```

Running Mode

If the mode is running, C and D will cause 30 seconds to be added to and subtracted from the mins and secs registers respectively. If the asterisk was pressed, 1 minute will be added instead, and if hash is pressed, the mode will be placed in paused mode.

```
; Running Mode
     ; elsif mode == running
     ;    if C
     ;         add 30s to time
     ;    if D
     ;         minus 30s to time
     ;    if *
     ;         add 1 min to time
```

```
;     if #
;          mode = paused
```

Paused mode

If the mode is paused instead, then asterisk will switch the mode to running, and hash will switch the mode to entry.

```
; Paused mode
     ; elsif mode == paused
     ;     if *
     ;          mode = running
     ;     if #
     ;          mode = entry
```

Finished mode

In finished mode, if the hash was pressed, then mode will be set to entry. Note that pressing the door in finished mode will also cause the mode to be set to entry, however this is not handled here, but rather in the open door interrupt.

```
; Finished mode
     ; elsif mode == finished
     ;     if #
     ;          mode = entry
```

Functions

The printDigitsToLCD function will, in entry mode, display the correct time, turntable, group number and door state. On the other hand, the printTimeToLCD function will, in running mode, display the correct time, turntable, group number and door state.

The display_numbers macro in actuality calls a checkTensDigit function which splits the current time held in the mins and secs registers into digits to be displayed on the screen. The clearTimeAndDigits function will clear the a, b, c and d time variables used in entry mode as well as the mins and secs registers used during running mode.

The turntable function will, according to the current character it should be displaying, specified by the index variable, display that character on the display. The updateIndex function will, when called, either increment or decrement the index variable between the bounds of 0 and 3 inclusive according to whether the CCWrotation variable is set or not. The turntable will turn counter-clockwise if the CCWrotation variable is set to 1. The toggleRotationDirection will toggle the value of CCWrotation by using an exclusive OR.

The updateTime function will decrement 1 second from the time represented by the mins and secs registers and print this on the display.

The updateMagnetron function, when called, will check which segment of the second has been reached, and then therefore will decide whether to turn the magnetron on or off. It will then set the magnetron to be on or off. The writePowerLabels function will display “Set Power 1/2/3” as well as the turntable, group number and door state.

The writeFinishedText function will display “Done\nRemove Food” on the display. The writeSecondLineToDisplay function will display the group number and door state to the display. This is used in the other functions that require writing the group number and door state. The writeDoorStateToDisplay will write either a “C” or “O” depending on whether the door is open or not, effective immediately. The setDoorLEDClosed and setDoorLEDOpen functions will cause the top LED to be on or off.

The setMotorOn, setMotorOff, resumeMotor and pauseMotor functions will turn the motor on or off. The difference between the first two and last two are how they handle the behavior of the motor when it is turned on again. When the motor is set to off using setMotorOff, it will begin again from the start of its cycles rather than where it left off when it was paused, which can be achieved by using pauseMotor.

The finished function will set the mode to finished, reset all counters, and turn off the magnetron and turntable.

Interrupts

The interrupts used in this program are for when the door is opened and when the door is closed. It is implemented in this manner because the door can be opened or closed at any point in the program.

Open Door

The variables temp1, temp2, temp3 and v, together with the X pointer (XH and XL) and the status register are all saved to the stack as these variables will be used in this interrupt but may also be used elsewhere in the program.

```
; -- save

push temp1
push temp2
push temp3
push v
push XL
push XH
in temp1, SREG
push temp1
```

The first step of instructions for this interrupt is setting the doorIsOpen flag to 1 to signify that the door has been opened. The functions setDoorLEDOpen and pauseMotor are also called to set the top most LED on and to stop the motor if it was running, which all indicate that the door has been opened. Next, the mode that as interrupted is checked. Normally it would return back to its original mode but if the microwave was in running mode, it would go into paused mode after the door has opened. Also, the microwave in finished mode would go into entry mode after the door has opened as per the specifications. This would involve clearing the display and clearing the time and digits variables for a new instance of the program.

```
getVar mode, v
cpi v, 3
breq runningToPausedSinceDoorOpened
cpi v, 5
breq finishedToEntrySinceDoorOpened
rjmp OPEN_DOOR_end

runningToPausedSinceDoorOpened:
     ldi v, 4
     setVar mode, v
     rjmp OPEN_DOOR_end

finishedToEntrySinceDoorOpened:
     ldi v, 1
     setVar mode, v

     lcd_write_com LCD_DISP_CLR ;clear the display
     lcd_wait_busy ;take yo time buddy
```

```
        rcall clearTimeAndDigits
        rcall printDigitsToLCD

        rjmp OPEN_DOOR_end
```

The interrupt concludes by restoring the values stored to the stack back to their original variables for correct usage when it resumes its position in the program. Additionally the state of the door on the display is overwritten to its current state which is O for opened.

```
OPEN_DOOR_end:
        ; -- restore
        lcd_write_com LCD_SHIFT_LEFT
        call writeDoorStateToDisplay

        pop temp1
        out SREG, temp1
        pop XH
        pop XL
        pop v
        pop temp3
        pop temp2
        pop temp1

        reti ;return from interrupt
```

Close Door

The close_door interrupt is similar in structure as the open_door interrupt with the exception that close door does not alter the mode. The variables temp1, temp2, temp3 and v, together with the X pointer (XH and XL) and the status register are all also saved to the stack as these variables will be used in this interrupt but may also be used elsewhere in the program.

```
; -- save
push temp1
push temp2
push temp3
push v
push XL
push XH
in temp1, SREG
push temp1
```

Again, the first step of instructions for the close door interrupt is to check the current state of the door. If the door is currently opened, it will set the doorIsOpen flag to 0 to signify the door has closed. It will also call the function setDoorLEDClosed to set the top most LED off to indicate the door is no longer opened. Additionally the state of the door on the display is overwritten to its current state which is now C for closed.

```
; -- instructions
getVar doorIsOpen, v
cpi v, 0
brne CLOSE_DOOR_cont
jmp CLOSE_DOOR_end

CLOSE_DOOR_cont:

     ldi v, 0
     setVar doorIsOpen, v

     call setDoorLEDClosed

     lcd_write_com LCD_SHIFT_LEFT
     call writeDoorStateToDisplay
```

The interrupt concludes by restoring the values stored to the stack back to their original variables for correct usage when it resumes its position in the program.

```
     ; -- restore
CLOSE_DOOR_end:
     pop temp1
     out SREG, temp1
     pop XH
     pop XL
     pop v
     pop temp3
     pop temp2
     pop temp1
     reti
```

Timer 0

Timer 0 will only function if the mode is running.

If functionsRunning is equal to 0, it means the magnetron, turntable and timer are not running, therefore the timers, magnetron variables and turntable variables are reset. The magnetron is turned on. functionsRunning is set to 1.

In normal operation, when functionsRunning is equal to 1, the timer will start to count intervals of 250ms, 500ms, 1 second and 5 seconds using counter250, counter500, counter1000 and counter5000. When 250ms has passed, counter250 will increment by 1. When 500ms has passed, counter500 will increment by 1. When 1 second has passed, counter1000 will increment by 1 and similarly for counter5000.
A function, updateMagnetron, is called every 250ms, updateTime is called every 1 second, and updateIndex is called every 5 seconds.
The updateIndex function will, when called, either increment or decrement the index variable between the bounds of 0 and 3 inclusive according to whether the CCWrotation variable is set or not. The turntable will turn counter-clockwise if the CCWrotation variable is set to 1.

Firstly, updateIndex checks to see whether the CCWrotation variable is set or not. If it is, then it decrements the index, otherwise it increments it. Then it checks whether the index is equal to 4 or -1, and if it is, then it correspondingly sets it back into the bound of 0 and 3, inclusive.

The updateTime function will decrement 1 second from the time represented by the mins and secs registers and print this on the display.

Firstly, it checks to see whether secs is equal to 0, and if so, then it checks to see whether mins is equal to 0. If both are 0, then operation stops and the program jumps to finished mode. If secs is equal to 0 and mins is not 0, it will put 59 in the secs register and decrement 1 from the mins register. Otherwise, it will decrement 1 from the secs register.

The updateMagnetron function, when called, will check which segment of the second has been reached, and then therefore will decide whether to turn the magnetron on or off. It will then set the magnetron to be on or off.

If the power is 100, the function returns. If the power is 50, and the magnetronCounter is 2 or 4, this is indicating that it is time to toggle the magnetron running mode, from off to on or on to off. However, for a power level of 25, the function will toggle the running mode only if the magnetronCounter is 1 or 4, at 250ms or a full second.

Timer2

Timer 2 handles two of the advanced features, namely the Finish Sounds and Key Sounds. When the microwave is finished, it will beep 3 times, of length 1 second each. It will also beep for 250ms when a key press is registered.

Finish Sounds

The implementation of finish sounds requires the use of four variables. `playFinishSound` specifies whether the finish sound should actually be playing or not. It will be playing if the program entered finished mode from running mode, and it will stop playing if the program entered entry mode when the open door button was pushed or when the hash key was pressed. The variable `finishSoundCounter` records the number of counts that has passed, and counts up to a second. When a second has passed, the `nFinishSounds` variable will increase by 1 and the `finishSoundIsOn` variable will toggle. The variable `nFinishSounds` records the number of times that a second has passed. When this variable reaches 6, `playFinishSound` is turned off. The variable `finishSoundIsOn` will specify if the sound is to be currently outputted or not.

```
playFinishSound: .byte 1
finishSoundCounter: .byte 1
nFinishSounds: .byte 1
finishSoundIsOn: .byte 1


; if playFinishSound == 1
;    if number of seconds passed == 6
;         playFinishSound = 0
;    out 0 to PORTB
;    if on
;         delay
;         out 0xFF to PORTB
;    if 1 sec has passed
;         clear counter
;         inc number of seconds passed
;         toggle on/off
```

Key Sounds

The key sounds are triggered by a key press on the keypad. When this occurs, the variable `keyPressed` is set to 1 and the `beepCounter` will be reset to 0. The timer will detect this as an indication to start playing sound, and the `beepCounter` will start counting up for 250ms. When 250ms has passed, `keyPressed` will be set to 0, and therefore the sound will stop playing.

```
; if keyPressed == 1
;    play sound
;         when 250ms is up
;              keyPressed = 0
```

Modules

Entry Mode

Upon the start of the program, the system will begin with entry mode. The conclusion of the program will also return back to entry mode. In this mode, the display will show the time being entered, the developer group name, the position of the turntable and the state of the door.

Whilst in entry mode, the system will take inputs from the keypad. An input of 0-9 will be interpreted as the time input and will be stored as individual digits. The stop or # button will clear these digits to all zeroes. Pressing A will take the user to a different display that reads "Set Power 1/2/3" which prompts the user to enter 1-3 on the keypad. This will be interpreted as 25%, 50% or 100% power respectively and is stored accordingly. This is reflected by the LED display. The stop # button will return to entry mode.

If the user has finished configuring time and power and is ready to cook food, pressing the start or * button will exit entry mode and enter running mode. The digits stored will be converted into time as minutes and seconds which will be used in running mode. If no time is entered then the default time is 1 minute and if no power is entered then the default power is 100%.

If the left push button is pressed at any time, this indicates that the door has opened and any input will be ignored with the exception of the right push button. This is reflected in the display which includes the door state. The user must shut the door to continue, which is achieved by pressing the right push button and will allow the user to return to entry mode.

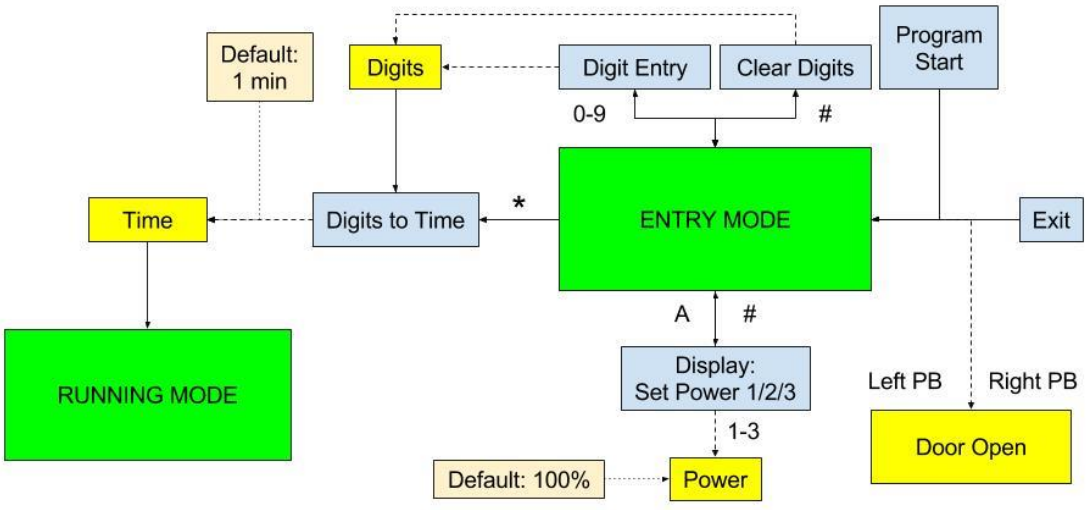

Figure 2: Entry Mode Flow Diagram

Running Mode

After configuring the time and power settings in entry mode, running mode allows the food to be cooked. In this mode, the display will show the current time left, the developer group name, the position of the turntable and the state of the door.

Whilst in cooking mode, the system will countdown the time by decrementing one second. Pressing the C button on the keypad will add 30 seconds to the time and pressing the D button will subtract 30 seconds from the time. Pressing the start or * button will add 1 minute to the time. The time on the display will be continually updated to reflect the current time left.

The magnetron will be turned on and will only be on whilst in running mode in order to cook the food. The magnetron power can be configured in entry mode and the magnetron running is represented by the spinning of the motor. Since a magnetron cannot adjust its magnitude, the power level is adjusted by the portion of the second it is turned on. For example a power level of 25% means it is turned on for only the first 250ms of the second, a power level of 50% means it is turned on for only the first 500ms of the second and a power level of 100% means it is always on. The motor operates in a similar manner at 100 revolutions per second.

Likewise the turntable will only rotate when in running mode for even heat distribution. This is reflected by the display which will rotate through the characters ‘-‘, ‘\’, ‘|’, ‘`’ at 3 revolutions per minute to represent a spinning turntable. Every time the microwave is running, the turntable will rotate in the opposite direction of its previous run.

The user is able to pause the cooking of food by pressing the stop or # button in which the system will enter pause mode. When the food has finished cooking, that is when the time has decremented to 0 minutes and 0 seconds, the system will enter finished mode. If the left push button is pressed at any time, this indicates that the door has opened and any input will be ignored with the exception of the right push button. This is reflected in the display which includes the door state. The user must shut the door to continue, which is achieved by pressing the right push button and will allow the user to return to running mode.

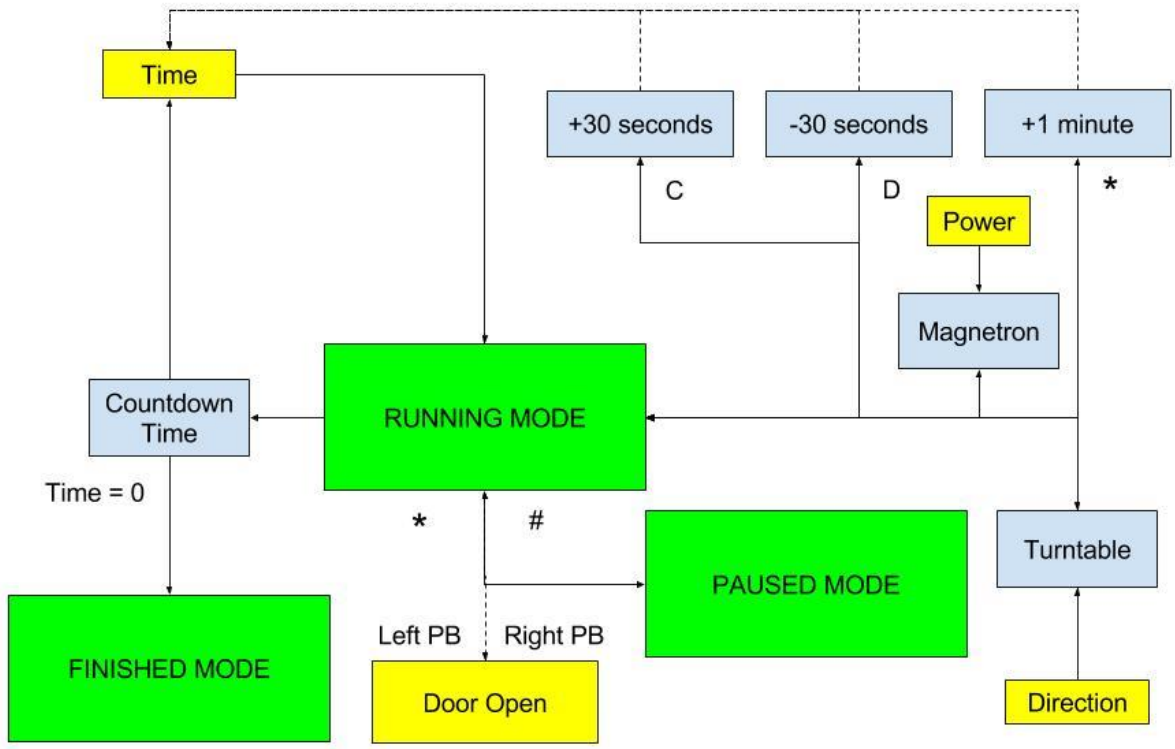

Figure 3: Running Mode Flow Diagram

Paused Mode

If the user wishes to temporarily stop the cooking of food whilst in running mode, they can enter pause mode by pressing the stop or # button. In this mode, the time will stop decrementing temporarily and the display will show the current time left, the developer group name, the position of the turntable and the state of the door. The user may resume cooking and re-enter running mode by pressing the start or * button, or may decide to finish cooking and press the stop or # button whilst in paused mode to exit and return to entry mode.

If the left push button is pressed at any time, this indicates that the door has opened and any input will be ignored with the exception of the right push button. This is reflected in the display which includes the door state. The user must shut the door to continue, which is achieved by pressing the right push button and will allow the user to return to paused mode.

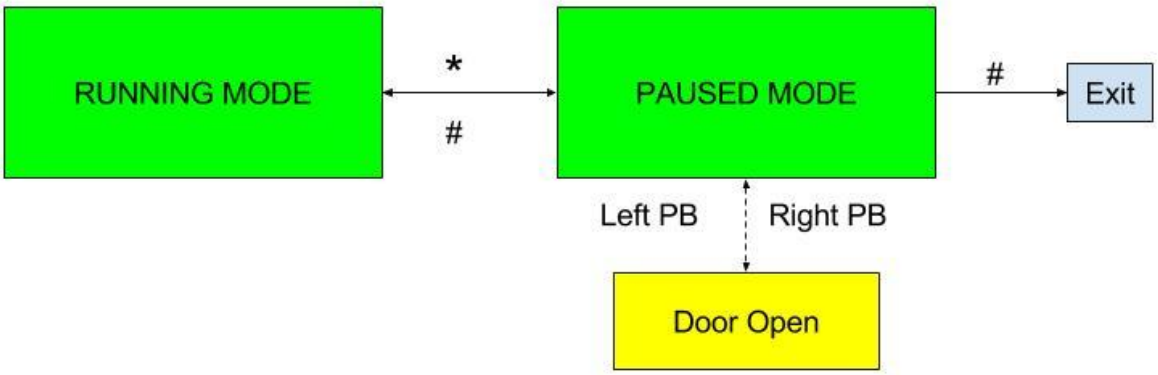

Figure 4: Paused Mode Flow Diagram

Finished Mode

When the food has finished cooking in running mode, this means that the time has decremented down to 0 minutes and 0 seconds. In that instance, the system will enter finished mode and beep 3 times for 1 second each separated by 1 second of silence. The system will prompt the user to open the door and remove the food with the display “Done, Remove food” which will require the user to press the left push button. This signifies the opening of the door to remove food and any input will be ignored with the exception of the right push button. If the door is opened before the beeps are finished, it will interrupt the beeps. The user must shut the door and is achieved by pressing the right push button which will conclude the system program and return back to entry mode.

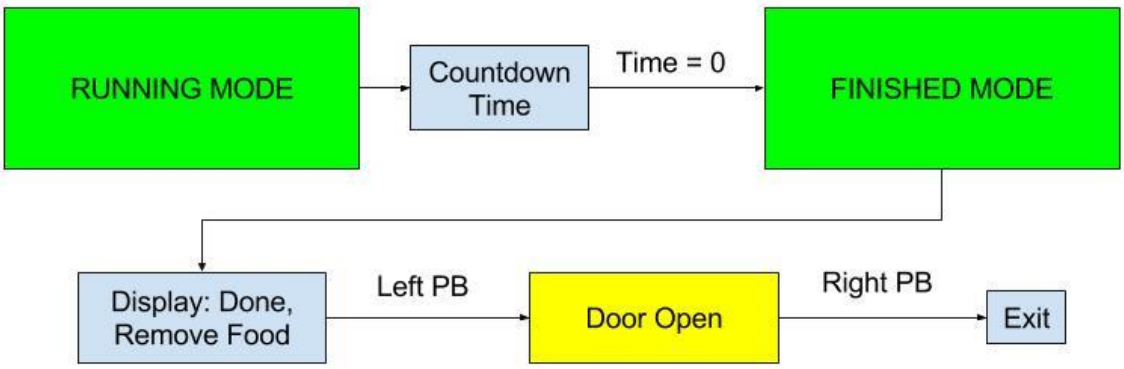

Figure 5: Finished Mode Flow Diagram