Claudia Tu
Tom Bremner

MICROWAVE EMULATOR

COMP2121 PROJECT

USER MANUAL

SETTING UP YOUR MICROWAVE

Connect each component to the appropriate port, as described in the table below.

COMPONENT	USAGE	PORT CONNECTION
LCD	Displays the status of the microwave.	D7–0 → PF7–0 RS → PA7 E → PA6 RW → PA5 BE → PA4
LED	Displays the power level on bottom 8 LEDs and the door status on top LED.	LED9–2 → PC7–0 LED0 → PH8 (LSB)
Motor	Spins at 75 revolutions per second when the microwave is running. The power level determines at what percentage of each second the motor spins.	OpO → JP91, right pin OpE → +5V Motor → PE2 JP91, right pin → TDX2
Buttons	Opens and closes the microwave door.	PB0 → RDX4 PB1 → RDX3
Keypad	Used to input times and activate various features.	C3–C0 → PL7–4 R3–R0 → PL3–0
Speaker	Used to provide aural feedback for the keypad and to alert the user when the microwave has finished.	Speaker contact → PB2 Speaker contact ⟶ GND
Backlight	Used to improve text visibility in environments with dim ambient light.	BL ⟶ PE6

OPERATION INSTRUCTIONS

- To activate the backlight, press any key. A beep will sound when keypad input is registered.

1. Place food in microwave.

   a. To **open the door** of the microwave, press the left green button. This will change the door indicator on the bottom right corner of the LCD to ‘O’ and turn on the top red LED.
   b. To **close the door** of the microwave, press the right green button. This will change the door indicator on the bottom right corner of the LCD to ‘C’ and turn off the top red LED.

2. If you would like to **change the power level** you want to cook the food at, press the “A” button. The LCD will display “Set Power 1/2/3”. You can then press 1 for 100% power, 2 for 50% power and 3 for 25% power. If the current power level is 100%, all the green LEDs will be lit up. If the current power level is 50%, half of the green LEDs will be lit up. If the current power level is 25%, two of the green LEDs will be lit up.
3. Enter your desired cooking time using the keypad. You may enter any time between 00:01 and 99:99. If you make a mistake, you can **clear the time** by pressing the “#” button.
   a. If you would like to cook the food for 1 minute, you can simply press “*” to start cooking without entering a time.
4. Press “*” to **start cooking** the food. This will turn on the turntable and the motor.
   a. You can increase the cooking time by 1 minute by pressing “*”.
   b. You can increase the cooking time by 30 seconds by pressing “C”. Note that this will increase the seconds only unless the number of seconds exceeds 99.
   c. You can decrease the cooking time by 30 seconds by pressing “D”.
5. Press “#” or the left green button to **pause** the operation. This will stop the turntable and the motor.
   a. Press “#” again to **cancel** the operation.
   b. Press “*” to **restart** the operation. Note that this will only function if the door is closed.
6. When the timer reaches 00:00, the display will display “Done” on the first line and “Remove Food” on the second line. There will be three 1 second beeps, separated by 1 second of silence to alert you that your food is ready.
   a. Press “#” and or the left button to **return** to the beginning.

INTERPRETING THE DISPLAYS

LED

Top	🟥	Door indicator. Lit when door is open, and unlit when door is closed.
	🟧	Unused.
	🟩	Power level 1 (100%) lights up all LEDs from the bottom up to this LED. At this setting, the motor spins at 75 revolutions per second.
	🟩	
	🟩	Power level 2 (50%) lights up half of the LEDs. At this setting, the motor to spins at 75 revolutions per second for the first half of each second.
	🟩	
	🟩	
	🟩	Power level 3 (25%) lights up the bottom two LEDs. At this setting, the motor spins 75 revolutions per second for the first quarter of each second.
Bottom	🟩	

LCD

The LCD always indicates the position of the turntable and the status of the door.

The **turntable** cycles through 4 different characters: `- / | \`. The backslash character is built using the character construction function in the LCD. This indicator is located in the top right corner of the screen.

The **door** status is indicated by C for closed or O for open. This indicator is located in the bottom right corner of the screen.

ENTRY, RUNNING AND PAUSED

3	5	:	2	1											\
															C

When you enter a time in the keypad, this will be displayed on the top left corner in the format `mm:ss`, i.e. the two digits left of the colon indicate the number of minutes remaining, whereas the two digits on the right indicate the number of seconds remaining.

The display also indicates the time remaining on the top left corner in the same format when the microwave is *running* or *paused*.

POWER

S	e	t		P	o	w	e	r		1	/	2	/	3	\
															C

When the LCD displays "Set Power 1/2/3", you can set the *power* by pressing the buttons 1, 2 or 3 on the keypad. This sets the power level to 100%, 50% and 25% respectively. The current power level is indicated by the LEDs (see previous section).

FINISHED

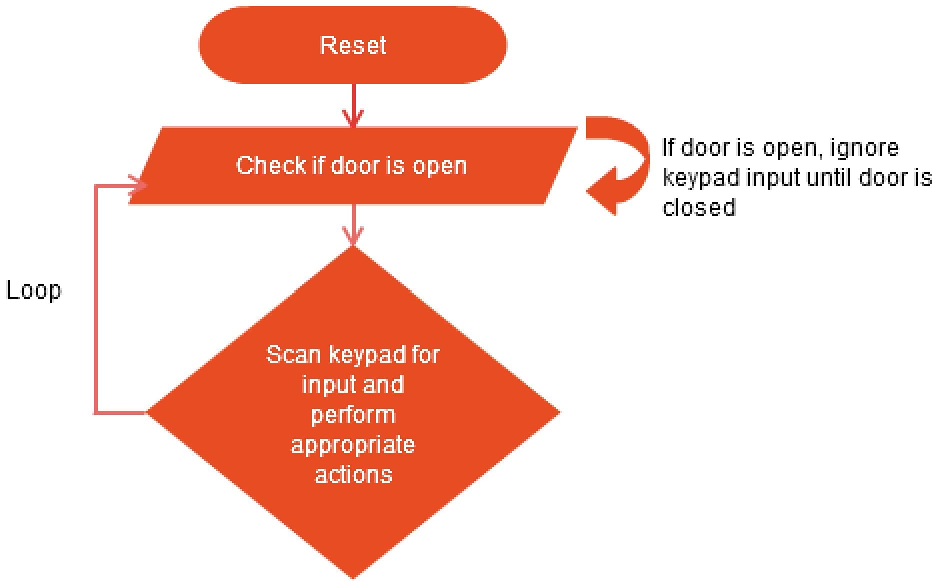
D	o	n	e												\
R	e	m	o	v	e		F	o	o	d					C

When the microwave has *finished* cooking the food, the LCD will display "Done" on the first line and "Remove Food" on the second line.

Claudia Tu
Tom Bremner

DESIGN MANUAL

SYSTEM FLOW CONTROL


Reset
Check if door is open
If door is open, ignore keypad input until door is closed
Loop
Scan keypad for input and perform appropriate actions

1. Our program starts off in RESET, which initialises the stack pointer, data segment variables, ports and port directions, the LCD and LCD backlight, the interrupts associated with the buttons and the settings for various timers.
    - The interrupts that are enabled at this point are the interrupts associated with the buttons and the timer associated with the backlight. All other interrupts are enabled based on keypad and button input.
2. The program then checks if the door is open. If the door is open, the program will ignore keypad input until the door is closed. The door is controlled by buttons that trigger external interrupts.
3. If the door isn't open, the program scans the keypad for input. The program then decides what to do with this input based on the current mode. We record the current mode in a data segment variable that is initialised to *entry* mode.

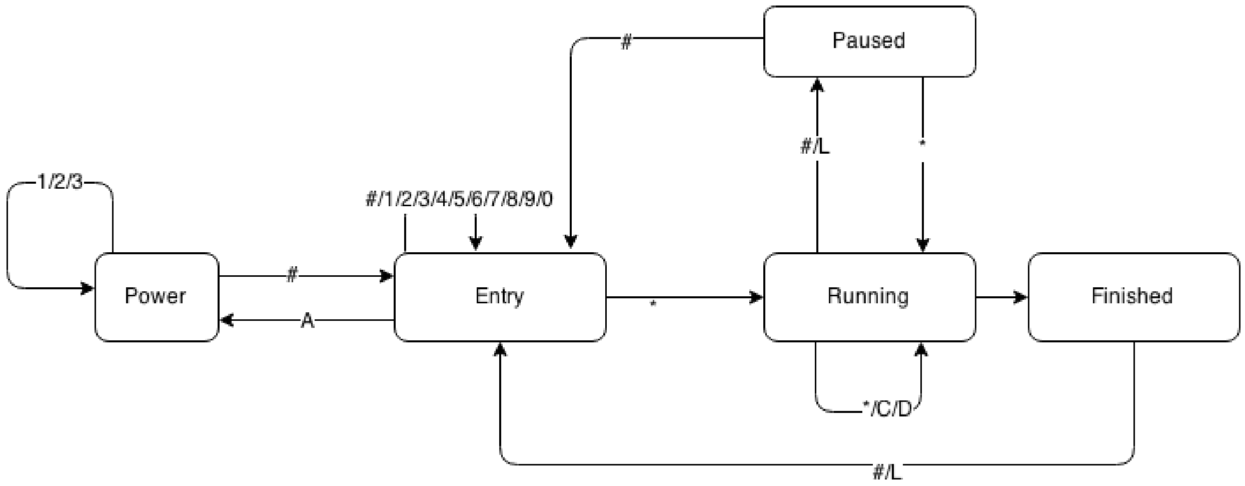
The inputs that the keypad accepts are described in the figure below. The arrows from each mode are labelled with the keypad inputs that are accepted by that mode. For example, the arrow from *paused* to *running* is labelled with an asterisk (*), so if the program is in *paused* mode, the keypad will accept * as an input. Note that changing the mode from Running to Finished is triggered by a timer-based interrupt, so that arrow is unlabelled.

In addition, when a key press is registered, the backlight is turned on and a sound is generated.

The actions that are taken based on a given keypad and mode input are performed by calling functions in other modules. For more detail on how the actions are performed, see the module specification section below.

Claudia Tu
Tom Bremner

Mode State Machine



MODE	KEYPAD INPUT	ACTION	RESULTING MODE
Entry	#	Clear digits entered in timer	Entry
	*	See Running function in Mode helper function	Running
	A	Display *power* mode text	Power
	0–9	Enter cooking time	Entry
Power	#		Entry
	1–3	Set the power level and LEDs	Power
Running	#	Pauses countdown	Paused
	L	Pauses countdown and opens door	Paused
	*	Adds 1 minute to the cooking time	Running
	C	Adds 30 seconds to the cooking time	Running
	D	Subtracts 30 seconds from the cooking time	Running
	Timer	Beep 3 times and display *finished* mode text	Finished
Paused	#	Cancels cooking operation and clears LCD	Entry
	*	Resumes cooking operation	Running
Finished	#	Clears LCD	Entry
	L	Clears LCD	Entry

MODULE SPECIFICATIONS

- **Bold output** indicates that the function or interrupt calls another function.
- Modules correspond to source files in our project. Each source file generally contains a few functions and some relevant constants.

MODULE	FUNCTIONS	INPUT	OUTPUT
Buttons	Initialise buttons		Trigger interrupt on falling edges Enable external interrupts
Delay	Sleep 1ms		
	Sleep 5ms		
	Sleep 20ms		
LCD	Initialise LCD		Initialise LCD Clear digits variable
	Display entered digits	Number of entered digits Entered digits	Entered digits are written to LCD in the format `mm:ss` with leading zeroes
	Display minutes and seconds	Minutes Seconds	Minutes and seconds are written to LCD in the format `mm:ss` with leading zeroes
	Display power text		Write “Set Power 1/2/3” to LCD
	Clear timer text		Clear first line of LCD without overwriting turntable and door indicator
	Clear finished text		Clear first and second line of LCD without overwriting turntable and door indicator
	Write 8-bit integer to LCD	Integer in temp1	Writes integer in temp1 to LCD
LCD Backlight	Initialise backlight and timer		Clears related variables Initialise PWM in Timer 4 Initialise OVF in Timer 2 Enable interrupt in Timer 2

MODULE	FUNCTIONS	INPUT	OUTPUT
	Enable backlight fade in		Clears related variables Sets backlight fade state to fade in
	Enable backlight fade out		Sets backlight fade state to fade out
Mode helper	Running	Entered digits Number of entered digits Turntable direction	Convert entered digits into minutes and seconds Change turntable direction **Start turntable and motor**
	Finished		**Stop turntable and motor** **Play finished sound** Clear entered digits and the number of entered digits Write "Done" and "Remove Food" to LCD Change mode to *finished*
Speaker	Play key press sound		250ms beep
	Initialise finish sound timer		Initialise CTC in Timer 1
	Play finished sound		Enable interrupt in Timer 1
	Check hash button	Keypad input	Change mode to *entry* if hash button (#) is pressed **Clear finished text**
Timer Arithmetic	Add seconds	Integer in temp1	Adds integer in temp1 to seconds
	Subtract seconds	Integer in temp1	Subtracts integer in temp1 from seconds
	Add minutes	Integer in temp2	Adds integer in temp2 to minutes
Turntable and Motor	Build backslash		Makes a custom backslash character for LCD
	Initialise turntable		Initialise related variables

MODULE	FUNCTIONS	INPUT	OUTPUT
			Write first turntable frame to LCD Initialise OVF in Timer 0
	Initialise motor		Initialise PWM in Timer 3 Initialise DDRE Initialise related variables Set LEDs to default power level
	Start turntable and motor		Enable interrupt for Timer 0 Enable motor
	Stop turntable and motor		Disable interrupt for Timer 0 Disable motor
	Load next turntable frame	Called every 0.5 seconds by Timer 0 Local counter stored in data segment	Increments local counter When local counter reaches 5 (2.5s), write next turntable frame to LCD
	Adjust RPS	RPS OCR3A	Checks if the motor is on If the motor is on, adjusts OCRA until RPS is equal to the target RPS Clears RPS

INTERRUPTS	INPUT	OUTPUT
Reset		Initialises stack pointer Initialises data direction registers and clears port registers **Initialises LCD** **Initialises turntable** **Initialises motor** **Initialises buttons** **Initialises backlight timer** **Initialises finished sounds** Initialises a number of variables

		Writes “C” to the LCD door indicator
Right button		Change door state to closed Turn on LED indicator Write “C” to LCD door indicator
Left button	Mode	Change mode from *running* to *paused* or from *finished* to *entry* Change door state to open Turn on LED door indicator Write “O” to LCD door indicator
Holes	RPS	Increments RPS
Timer 0 Overflow	Local counter stored in data segment Power level	If local counter reaches 1953 (0.25s) and the power level is 3, turn off motor If local counter reaches 3906 (0.5s) and power level is 2, turn off motor If local counter reaches 3906 (0.5s), **load next turntable frame** and **adjust RPS** If local counter reaches 7812 (1s), decrement cooking time by 1 second, **display minutes and seconds**, **load next turntable frame**, turn the motor back on (on power levels 2 and 3) and clear local counter
Timer 1 Compare Match A	Local counter stored in data segment Mode	If we have generated 3 beeps or if we have left *finished* mode, disable the interrupt for Timer 1 Otherwise, generate a sound for 1 second and **check if the hash button is pressed** every loop Increment the local counter to indicate that we have generated a beep
Timer 2 Overflow	Mode Backlight state Local counter stored in data segment Local seconds counter stored in data segment OCR4A	If local counter reaches 30, backlight state is fade in and we have reached maximum brightness, set backlight state to stable If local counter reaches 30 and backlight state is fade in, increment OCR4A If local counter reaches 30, backlight state is fade out and we have reached minimum brightness, set backlight state to stable If local counter reaches 30 and backlight state is fade out, decrement OCR4A If mode is not *running* and local seconds counter reaches 10 (10s), **enable backlight fade out**

DATA STRUCTURES

REGISTERS

Registers were used for storing values of temporary variables as well as loading, storing and manipulating the values of the data segment variables. Four registers are specifically defined for use as storing the current row and column when scanning for keypad presses, and others for holding the values of the row mask and column mask. Two registers are defined as temporary registers, used for holding temporary values such as when loading and storing data segment variables, as well as for passing values into functions. Another register is used for calculations in macros, and a final register is designated for use in a function to write 8-bit integers to the LCD.

DATA SEGMENT VARIABLES

Data segment variables were used for storing the values of variables such as the mode (running, paused, entry, power, finished), the state of the door (open/closed), the time left on the countdown, the power value, timer counters (countdown timer, turntable animation, backlight fading) and button de-bouncing states among others.

CONSTANTS

Constants were set using `.set` and `.equ` and were using to help make the code more easily readable and maintainable. Constants were used for I/O Port Masks, mode states (entry, running, paused, power and finished), door states (open/closed), power led states (100%, 50%, 25%), LCD backlight states (fading in, fading out, stable) and LCD instructions.

ALGORITHMS

CONVERTING ENTERED DIGITS TO MINUTES AND SECONDS

Digits entered are stored in data segment in 4 bytes (1 byte per digit) & number of digits entered is stored in a single byte (max 4 digits entered, min 0)

```
minutes = 0;
seconds = 0;
switch (numDigits)
        case 0:
                // if no time is entered, set time to 1 minute
                minutes = 1;
                break;
        case 4:
                minutes = (4th digit)*10;
        case 3:
                minutes += (3rd digit);
        case 2:
                seconds = (2nd digit)*10;
        case 1:
                seconds += (1st digit);
```

```
    default:
        break;
```

DISPLAYING MINUTES AND SECONDS

Minutes and seconds are stored in separate single bytes in the data memory.

```
tempMins = minutes;
counter = 0;
while (tempMins >= 10) {
    tempMins -= 10;
    counter++;
}
printToLCD(counter + '0');
printToLCD(tempMins);
printToLCD(':');
counter = 0;
tempSecs = seconds;
while (tempSecs >= 10) {
    tempSecs -= 10;
    counter++;
}
printToLCD(counter + '0');
printToLCD(tempSecs);
```

ADDING TIME TO TIMER WHILE RUNNING

ADDING SECONDS

```
// s : seconds to add
totalSecs = seconds + s;
if (totalSecs > 99) {
    while (totalSecs > 99) {
        addMinutes(1); // uses next function
        totalSecs -= 60;
    }
}
seconds = totalSecs;
```

ADDING MINUTES

```
// m : minutes to add
if (minutes < 99) {
    minutes++;
}
```

SUBTRACTING SECONDS

```
// s : seconds to subtract
tempMin = minutes;
tempSec = seconds;
if (s <= seconds) {
    tempSec -= s;
} else {
    while (s > tempSec && tempMin > 0) {
        if (tempMin > 0) {
            tempMin--;
            tempSec = 60 - (s - tempSec);
        } else {
            tempMin = minutes;
            tempSec = seconds;
        }
    }
}
seconds = tempSec;
minutes = tempMin;
```

BACKLIGHT

Backlight state, pulse width modulation duty cycle, seconds counter and fade counter are stored in separate single bytes in the data memory whereas the backlight counter is stored in 2 bytes. Each time a button is pressed, the backlight state is set to BACKLIGHT_FADEIN and the backlight counter is reset.

```
Timer2OVF:
    blFadeCounter++;
    if (blFadeCounter == 30) {
        blFadeCounter = 0;
        if (blState == BACKLIGHT_FADEIN) {
            if (blPWM < 0xFF) {
                blPWM++;
            } else {
                blState = BACKLIGHT_STABLE;
            }
        } else if (blState == BACKLIGHT_FADEOUT) {
            if (blPWM > 0) {
                blPWM--;
            } else {
                blState = BACKLIGHT_STABLE;
            }
        }
    }
    displayBacklight(blPWM);

    if (mode != RUNNING) {    // light stays on if microwave is running
        blCounter++;
        if (blCounter == 7812) {
            blCounter = 0;
            blSeconds++;
            if (blSeconds == 10) {
                blSeconds = 0;
                blState = BACKLIGHT_FADEOUT;
            }
        }
        }
```

TURNTABLE ANIMATION

The turntable animation stored in 4 bytes in data memory. The turntable frame (0-3), turntable seconds counter and turntable direction are stored in separate single byte data segment variables and the turntable temporary counter is stored in 2 bytes.

```
if (mode == FINISHED) {
        ttDirection = !ttDirection;
}

TurntableInit:
        ttAnimation = {-,/,|,\};
        ttFrame = 0;
        ttSeconds = 0;
        ttCounter = 0;
        ttDirection = CLOCKWISE;

Timer0OVF
        ttCounter++;
        if (ttCounter == 7812) {
                ttCounter = 0;
                ttSeconds++;
                if (ttSeconds == 5) {
                        ttSeconds = 0;
                        if (ttDirection == CLOCKWISE) {
                                if (ttFrame < TT_FRAMES-1) {
                                        ttFrame++;
                                } else {
                                        ttFrame = 0;
                                }
                        } else {
                                if (ttFrame > 0) {
                                        ttFrame--;
                                } else {
                                        ttFrame = TT_FRAMES;
                                }
                        }
                        lcdCommand(LCD_TURNTABLE_POSITION);
                        printToLCD(ttAnimation[ttFrame]);
                }
        }
```