

Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

200
1822 - 2022

L4 – Embedded C (ACDA4013)

Generating a Program for a Specified Embedded Requirement in C

David Mathew Lewis

2219439

Electrical and Electronic Engineering

Embedded systems are ubiquitous in our daily lives, powering devices ranging from simple household appliances to sophisticated automotive systems. Mastering the intricacies of Embedded C programming is essential for crafting efficient and reliable software solutions for such embedded systems. In this assignment, we embark on a journey to understand the syntax and semantics of Embedded C programming while adhering to good programming practices.

[illegible]

Control of Motor A (SW1-B): We are tasked with writing a program that controls the direction of Motor A based on the value of RC1. Depending on the value, the motor will rotate clockwise, anti-clockwise, or stop. Additionally, we must display the ADC reading value and rotation direction on the LCD display.

Display of ADC Reading (SW1-A): When SW1-A is selected, the digital output of the ADC reading (RV1) must be displayed on the LEDs, and its value should be shown on the LCD display. Additionally, the value should be echoed onto the virtual terminal.

Button Control for Motor A (SW1-D): Pressing SW1-D should make Motor A rotate clockwise as long as the button is depressed, and it should stop when released. The status of the button press should be displayed on the LCD display and echoed onto the virtual terminal.

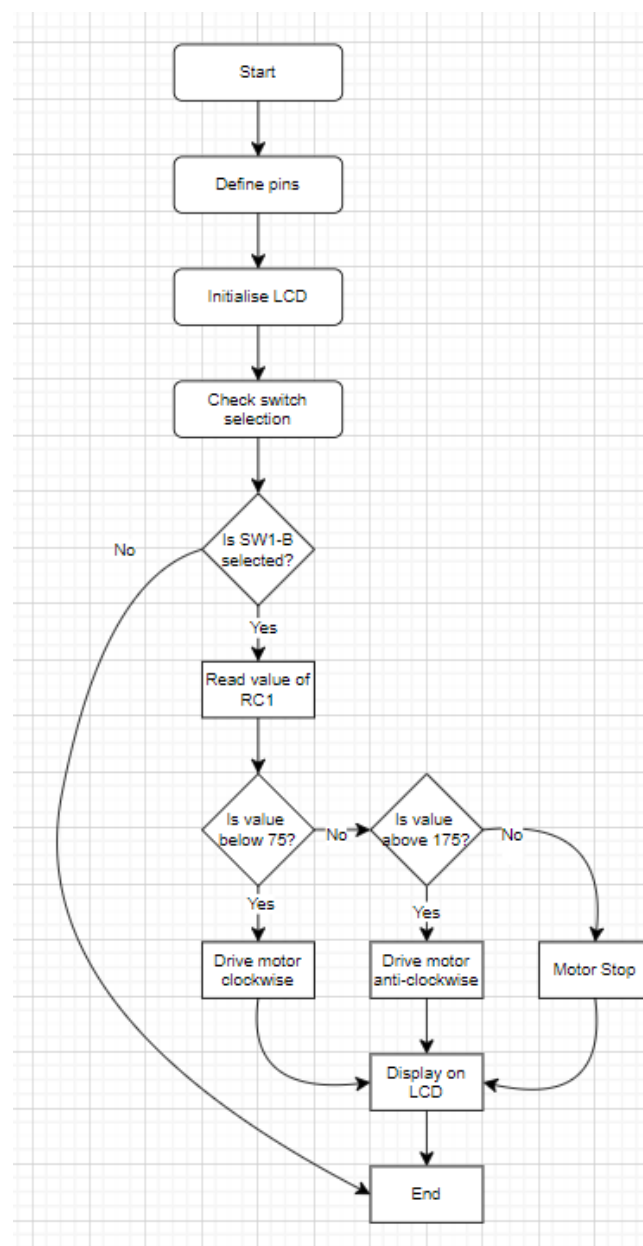
By implementing well-structured code and utilizing functions effectively, we aim to develop a compact and readable program that efficiently addresses each aspect of the assignment brief.

Let's dive into the programming journey, exploring Embedded C to craft solutions that demonstrate our understanding of programming for embedded systems.

Task 1 - Control of Motor A (SW1-B):

Flowchart

This flowchart outlines the steps involved in determining the direction of Motor A based on the value of RC1 when SW1-B is selected. It includes conditions for clockwise rotation, anti-clockwise rotation, and stopping the motor, along with the corresponding actions to be taken and the display requirements on line 2 of the display.



Code

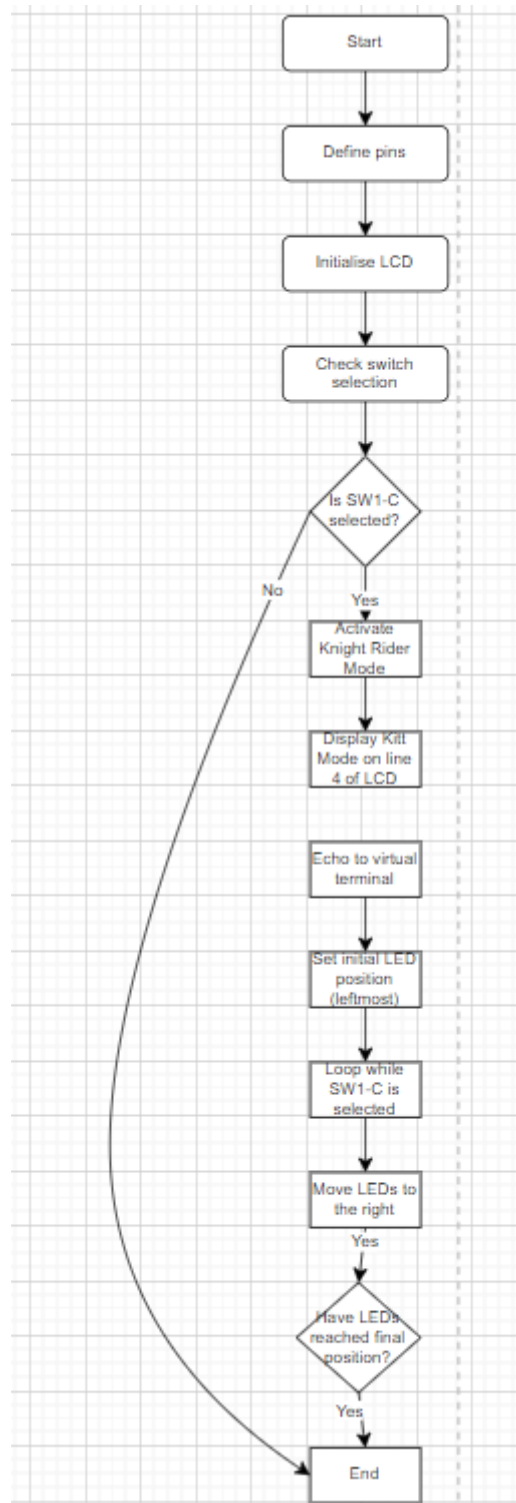
```
1  #include <18F45K50.h>           //Include chip
2  #use delay(internal=31khz)      // Internal oscillator
3
4  #include <lcd.c>                 //Included LCD
5  #define LCD_ENABLE_PIN PIN_D2  //Explains pin-out
6  #define LCD_RS_PIN PIN_D0
7  #define LCD_RW_PIN PIN_D1
8  #define LCD_DATA4 PIN_D4
9  #define LCD_DATA5 PIN_D5
10 #define LCD_DATA6 PIN_D6
11 #define LCD_DATA7 PIN_D7
12
13 #define MOTOR_A_PIN  PIN_C0      //Defines motor pins
14 #define MOTOR_B_PIN PIN_C1
15
16 #define SW1-A PIN_A2             //Defines switch pins
17 #define SW1-B PIN_A3
18 #define SW1-C PIN_A4
19 #define SW1-D PIN_A5
20
21 #define RV1 PIN_A0              //Defines variable resistor
22 #define BUTTON PIN_A1           //Defines push button
23
24 void main(void) {
25
26     TRISA = 0xFF;               // Configure port A as inputs
27     TRISB = 0x00;               // Set port B as all outputs
28     PORTB = 0x00;               // Initialize port B to all LEDs off
29     TRISC &= ~(1 << 0);         // RC0 as output
30     TRISC &= ~(1 << 1);         // RC1 as output
31     LATC &= ~(1 << 0);          //Initialise RCO to 0
32     LATC &= ~(1 << 1);          //Initialise RC1 to 0
33 }
```

```
34 lcd_init(); // Initialize LCD
35
36 while (1) {
37     if (RA3 == 0) { // Switch B is selected
38         int adc_value = ADRESL; // Read ADC value
39
40         lcd_clear();
41         lcd_goto(0, 0);
42         lcd_puts("ADC: ");
43         lcd_putint(adc_value);
44
45         if (adc_value < 75) {
46             RC0 = 1; // Motor A clockwise
47             lcd_goto(1, 0);
48             lcd_puts("CW");
49         } else if (adc_value > 175) {
50             RC1 = 1; // Motor B anti-clockwise
51             lcd_goto(1, 0);
52             lcd_puts("CCW");
53         } else {
54             RC0 = 0; // Motor A stop
55             RC1 = 0; // Motor B stop
56             lcd_goto(1, 0);
57             lcd_puts("Stop");
58         }
59     } else {
60         RA3 = 0; // Switch not selected
61         lcd_clear();
62         lcd_goto(0, 0);
63         lcd_puts("Switch B not selected");
64     }
```

Task 2 - Knight Rider LED Effect (SW1-C):

Flowchart

This flowchart outlines the steps involved in implementing the Knight Rider mode when SW1-C is selected. It includes actions such as displaying "Kitt mode" on line 4 of the LCD, echoing the message to the virtual terminal, and controlling the LED sequence from left to right and back again. The loop continues until SW1-C is no longer selected, at which point the Knight Rider mode is deactivated, and the flowchart ends.



Code

```

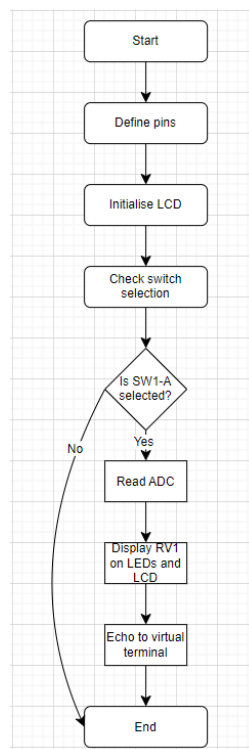
66 □ while (1) {
67     if (RA4 == 0) //Switch C is selected
68         const unsigned char led_pattern[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80};
69         int i;
70 □ for (i = 0; i < sizeof(led_pattern); i++) {
71             LATD = led_pattern[i];
72             __delay_ms(150);
73         }
74 □ for (i = sizeof(led_pattern) - 2; i >= 1; i--) {
75             LATD = led_pattern[i];
76             __delay_ms(150);
77         }
78     }
79
80 □ void displayMessage(void) {
81     LCD_Clear();
82     LCD_Set_Cursor(4, 5); //Display centrally on line 4
83     LCD_Write_String("Kitt mode");
84     printf("Kitt mode\n"); // Echo to virtual terminal
85 }
86 □ unsigned int readADC() {
87     ADCON0bits.GO = 1; // Start conversion
88     while (ADCON0bits.DONE); // Wait for conversion to complete
89     return ((unsigned int)ADRESH << 8) | ADRESL; // Return result

```

Task 3 - Display of ADC Reading (SW1-A):

Flowchart

This flowchart outlines the steps involved in displaying the digital output of the ADC reading RV1 on the LEDs, centred on line 1 of the LCD, and echoed onto the virtual terminal when SW1-A is selected. The flow proceeds linearly from checking the switch state to displaying the value and echoing it, and then ends.



Code

```

92 void displayOnLCDandTerminal(unsigned int adcValue) {
93     char buffer[16];
94     sprintf(buffer, "%4u", adcValue); // Format ADC value
95     Lcd_Set_Cursor(1, 4); // Set cursor to center of line 1
96     Lcd_Write_String(buffer); // Display on LCD
97     printf("%s\n", buffer); // Echo on virtual terminal
98 }
99
100 void main() {
101     init();
102     while (1) {
103         if (SW1_A) {
104             unsigned int adcValue = readADC();
105             LEDS = adcValue >> 4; // Display ADC value on LEDs
106             displayOnLCDandTerminal(adcValue);
107         }
108
109         while (1) {
110             if (SW1-D == 1) { // Button is pressed
111                 MOTOR_A = 1; // Turn motor clockwise
112                 LCD_Set_Cursor(3, 1);
113                 LCD_Write_String("Button Pressed");
114                 printf("Button Pressed\n");
115             } else {
116                 MOTOR_A = 0; // Stop motor
117                 LCD_Set_Cursor(3, 1);
118                 LCD_Write_String("Button Released ");
119                 printf("Button Released\n");
120             }
121             __delay_ms(100); // Debounce delay
122         }
123     }
124 }

```

Task 4 - Button Control for Motor A (SW1-D):

Flowchart

This flowchart outlines the steps involved in controlling Motor A based on the state of SW1-D and the button press. It includes actions such as displaying the status of the button on line 3 of the LCD, echoing the status to the virtual terminal, rotating Motor A clockwise when the button is pressed, and stopping Motor A when the button is released. The loop continues while the button is pressed, ensuring continuous motor rotation, and ends when the button is released. Finally, the flowchart terminates.

Code

```
123 □ void main() {  
124     initPIC18F45K50();  
125     lcd_goto(0x40); // Go to line 3 of LCD  
126  
127 □ while (1) {  
128 □     if (SW1_D == 1) {  
129         MOTOR_A = 1; // Turn on Motor A  
130         lcd_puts("Button Pressed ");  
131         printf("Button Pressed\n"); // Echo on virtual terminal  
132     } else {  
133         MOTOR_A = 0; // Turn off Motor A  
134         lcd_puts("Button Released");  
135         printf("Button Released\n"); // Echo on virtual terminal  
136     }  
137     __delay_ms(100); // Debounce delay  
138     lcd_goto(0x40); // Return cursor to line 3  
139 }
```