

EE308 MICROPROCESSORS LABORATORY

LAB -6-

Safe Lock

ZAID ALBU-AZAM 17050241001

MERVE DELİBAŞ 17050211032

The main body

```
LAB6.c startup_TM4C123.s
   1 #include <stdint.h>
   2 #include <stdlib.h>
   3 #include <tm4cl23gh6pm.h> /*By adding the header file we don't need to define the registers*/
   5 /* Declaring functions we may use */
   6 void SysClock 40MHz ();
   7 void SysTick init();
   8 void SysTick Wait (unsigned long reload);
   9 void SysTick Wait n ms (unsigned long n);
  10 void Port A init ();
  11 void Port B init ();
  12 void Port F init ();
  13 unsigned long resistanceRead();
  14 void ADCO PE3 init(void);
  15
  16
  17 int main (void)
  18 - SysClock 40MHz ();
  19
          SysTick init();
  20
          Port A init ();
          Port B init ();
  21
  22
          Port F init ();
  23
          ADCO PE3 init();
  24
  25
                                       // Variables to save the 1) set botton read 2) open botton read 3) The phases
          int set, open, phase=0;
  26
          int L[3], S[3];
                                       // Arrays to hold POT variables (L[3] for setting S[3] for opening
  27
  28
          while (1)
  29 -
            set = GPIO PORTF DATA R & 0x10; // Read and hold SW1 value (set botton)
  30
  31
            open = GPIO PORTF DATA R & 0x01; // Read and hold SW2 value (open botton)
  32
  33
            if (GPIO PORTA DATA R == 0x00)
                                              // This line is to check the locks state. If they are locked
                                              // and set botton is pressed it will not enter the setting phase
  34
```

```
LAB6.c
           startup_TM4C123.s
    30
               set = GPIO PORTF DATA R & 0x10;
                                                 // Read and hold SWl value (set botton)
               open = GPIO PORTF DATA R & 0x01; // Read and hold SW2 value (open botton)
    31
    32
    33
               if (GPIO PORTA DATA R == 0x00)
                                                 // This line is to check the locks state. If they are locked
    34
                                                 // and set botton is pressed it will not enter the setting phase
    35 -
    36
                 while (set==0)
                                                 // If set botton(negative logic) has been pressed enter the set phase
    37 -
    38
                   switch (phase)
                                                    // Setting the value of the 1st lock
    39 -
                   { case 0:
     40
                       GPIO PORTA DATA R ^= 0x80; // Toggle only the first LED-PA7 (Lock 1)
                       SysTick Wait n ms(250);
                                                    // Toggling with 1/4 sec intervals
    41
     42
                     if((GPIO PORTB DATA R & 0x10) ==0x10) // Reading the next botton (PB4). If it pressed:
    43
                       { L[0]= (resistanceRead()/100);
                                                           // Hold the POT value divided by 100 in the first elment of L array.
     44
                                                            // Lock 1 value is set
    45
                         GPIO PORTA DATA R |= 0x80;
                                                            // Stop toggling (Lit) first LED (Lock 1 is locked)
     46
                         phase++;
                                                            // Go to the next phase
     47
    48
                     break:
    49
                     case 1:
                                                     // Setting the value of the 2nd lock
                                                     // Toggle only the 2nd LED-PA6 (Lock 2). The 1st LED has been lit
    51
                       GPIO PORTA DATA R ^= 0x40;
    52
                                                     // in previous phase and it will not chamge because we XORing it with 0
                                                     // Toggling with 1/4 sec intervals
    53
                       SysTick Wait n ms(250);
    54
                       if((GPIO PORTB DATA R & 0x10) == 0x10) // Reading the next botton (PB4). If it pressed:
                       { L[1] = (resistanceRead()/100);
                                                           // Hold the POT value divided by 100 in the 2nd elment of L array.
    55 -
    56
                                                            // Lock 2 value is set
    57
                         GPIO PORTA DATA R |= 0x40;
                                                            // Stop toggling(Lit) 2nd LED (Lock 2 is locked)
    58
                         phase++;
                                                            // Go to the next phase
    59
    60
                       break:
     61
     62
                     case 2:
                                                     // Setting the value of the 3rd lock
                                                     // Toggle only the 3nd LED-PA5(Lock 3). The 1st&2nd LEDs have been lit
     63
                       GPIO PORTA DATA R ^= 0x20:
<
```

```
startup TM4C123.s
LAB6.c
  62
                                                   // Setting the value of the 3rd lock
                  case 2:
  63
                    GPIO PORTA DATA R ^= 0x20;
                                                   // Toggle only the 3nd LED-PA5(Lock 3). The 1st&2nd LEDs have been lit
  64
                                                   // in previous phase and they will not chamge because we XORing them with 0
  65
                    SysTick Wait n ms(250);
                                                   // Toggling with 1/4 sec intervals
                    if((GPIO PORTB DATA R & 0x10) ==0x10) // Reading the next botton (PB4). If it pressed:
  66
  67
                    { L[2]= (resistanceRead()/100);
                                                           // Hold the POT value divided by 100 in the 3nd elment of L array.
  68
                                                           // Lock 3 value is set
  69
                                                           // Stop toggling(Lit) 3nd LED (Lock 3 is lcoked)
                      GPIO PORTA DATA R |= 0x20;
  70
                                                           // Go to the next phase
                      phase++;
  71
                    }
  72
                  break;
  73
  74
                  case 3:
                                                   // Ending setting phase
  75
                    set=1;phase=0;
                                                   // Clear set value (negative logic) and clear phase value
  76
                  break;
  77
  78
  79
            }
  80
  81
            if (GPIO PORTA DATA R == 0xE0)
                                                   // This line is to check the locks state. If they are open
  82
                                                   // and open botton is pressed it will not enter the opening phase
  83 -
  84
                                                   // If open botton(negative logic) has been pressed enter the open phase
              while (open==0)
  85
  86
                switch (phase)
  87 日
                { case 0:
                                                   // Opening the 1st lock
  88
                                                   // Toggle only the first LED-PA7 (Lock 1)
                    GPIO PORTA DATA R ^= 0x80;
  89
                    SysTick Wait n ms (250);
                                                   // Toggling with 1/4 sec intervals
  90
                    if((GPIO PORTB DATA R & 0x10) ==0x10) // Reading the next botton (PB4). If it pressed:
                                                           // Hold the POT value divided by 100 in the first elment of S array.
  91
                    { S[0]= (resistanceRead()/100);
  92
                                                           // someone entered value to open 1st lock
  93
                      if(L[0]==S[0])
                                                           // Copmare the set value of 1st lock (previously set) with the value
  94
                                                           // recently entered. If they are equal:
  95 |
                      { GPIO PORTA DATA R &= ~0x80:
                                                           // Stop toggling(Turn OFF) first LED (Lock 1 is opend)
```

```
LAB6.c
          startup TM4C123.s
              while (open==0)
                                                   // If open botton(negative logic) has been pressed enter the open phase
  84
  85
  86
                switch (phase)
  87 -
                 { case 0:
                                                   // Opening the 1st lock
  88
                    GPIO PORTA DATA R ^= 0x80;
                                                   // Toggle only the first LED-PA7 (Lock 1)
  89
                     SysTick Wait n ms(250);
                                                   // Toggling with 1/4 sec intervals
                     if((GPIO PORTB DATA R & 0x10) == 0x10) // Reading the next botton (PB4). If it pressed:
  90
                     { S[0]= (resistanceRead()/100);
  91
                                                           // Hold the POT value divided by 100 in the first elment of S array.
  92
                                                           // someone entered value to open 1st lock
  93
                      if(L[0]==S[0])
                                                           // Copmare the set value of 1st lock (previously set) with the value
  94
                                                           // recently entered. If they are equal:
  95
                       { GPIO PORTA DATA R &= ~0x80;
                                                           // Stop toggling (Turn OFF) first LED (Lock 1 is opend)
  96
                        phase++;
                                                           // Go to the next phase
  97
                      }
  98
  99
                    break:
 100
 101
                  case 1:
                                                           // Opening the 2nd lock
                                                           // Toggle only the 2nd LED-PA6 (Lock 2)
 102
                     GPIO PORTA DATA R ^= 0x40;
 103
                    SysTick Wait n ms(250);
                                                           // Toggling with 1/4 sec intervals
 104
                    if((GPIO PORTB DATA R & 0x10) == 0x10) // Reading the next botton (PB4). If it pressed:
 105 -
                     { S[1] = (resistanceRead()/100);
                                                           // Hold the POT value divided by 100 in the 2nd elment of S array.
                                                           // someone entered value to open 2nd lock
 106
 107
                                                           // Copmare the set value of 2nd lock (previously set) with the value
                      if(L[1]==S[1])
 108
                                                           // recently entered. If they are equal:
 109 -
                                                           // Stop toggling (Turn OFF) 2nd LED (Lock 2 is opend)
                       { GPIO PORTA DATA R &= ~0x40;
                                                           // Go to the next phase
 110
                        phase++;
 111
 112
 113
                      break:
 114
 115
                     case 2:
                                                           // Opening the 3rd lock
 116
                       GPIO PORTA DATA R ^= 0x20;
                                                           // Toggle only the 3rd LED-PA5 (Lock 3)
 117
                      SysTick Wait n ms(250):
                                                           // Toggling with 1/4 sec intervals
```

```
LAB6.c
         startup_TM4C123.s
 114
 115
                    case 2:
                                                           // Opening the 3rd lock
                                                           // Toggle only the 3rd LED-PA5 (Lock 3)
 116
                      GPIO PORTA DATA R ^= 0x20;
 117
                      SysTick Wait n ms(250);
                                                           // Toggling with 1/4 sec intervals
 118
                      if((GPIO PORTB DATA R & 0x10) ==0x10) // Reading the next botton (PB4). If it pressed:
 119
                      { S[2]= (resistanceRead()/100);
                                                            // Hold the POT value divided by 100 in the 3nd elment of S array.
 120
                                                            // someone entered value to open 3rd lock
 121
                                                             // Copmare the set value of 3rd lock (previously set) with the value
                        if(L[2]==S[2])
 122
                                                             // recently entered. If they are equal:
 123
                                                            // Stop toggling (Turn OFF) 3rd LED (Lock 3 is opend)
                        { GPIO PORTA DATA R &= ~0x20;
 124
                          phase++;
                                                            // Go to the next phase
 125
 126
 127
                      break:
 128
 129
                    case 3:
                                                           // Ending opening phase
 130
                      open=1;phase=0;
                                                          // Clear open value(negative logic) and clear phase value
 131
                      break;
 132
 133
 134
 135
          }
 136
 137
        return 0;
 138
 139
 140
 141
 142
 143 /* Function to set system clock for 40MHz */
 144 void SysClock 40MHz ()
 145 - { /* 1- Enableing USERCC2 in RCC2 */
      SYSCTL RCC2 R |= 0x800000000;
                                                                    //RCC2{31}=1
 147
       /* 2- Bypass PLL while initializing */
```

The functions we used

```
LAB6.c
        startup_TM4C123.s
140
141
142
143 /* Function to set system clock for 40MHz */
144 void SysClock 40MHz ()
145 ⊟{ /* 1- Enableing USERCC2 in RCC2 */
      SYSCTL RCC2 R |= 0x800000000;
                                                                  //RCC2{31}=1
147
       /* 2- Bypass PLL while initializing */
148
       SYSCTL RCC2 R |= 0x000000800;
                                                                  //RCC2[11]=1
149
       /* 3- Setting external crystal value and oscillator source for PLL */
150
       SYSCTL RCC R = (SYSCTL RCC R &~ 0X000007C0)
                                                                  //Clearing RCC[10:6] then setting XTAL=RCC[10:6]=15 (16MHz ext. OSC.
151
                       + (0x15<<6);
152
       /* 4- Configuration for main oscillator source */
153
       SYSCTL RCC2 R &= ~0x000000070;
                                                                  //Clearing RCC2[6:4] OSCSRRC2 =000=MOSC
154
       /* 5- Clearing PWRDN to active PLL */
155
      SYSCTL RCC2 R &= ~0x00002000;
                                                                  //Clearing RCC2[13]=PERDN=0
156
       /* 6- Setting system clock */
157
       SYSCTL RCC2 R |= 0x400000000;
                                                                  //Setting RCC2[30]=1 (selecting DIV400=400MHz
158
       /* 7- Setting system clock divider RCC[28:22] */
159
      SYSCTL RCC2 R = (SYSCTL RCC2 R &~ 0x1FC00000)+(0x09<<22); //Gives N=9
160
       /* 8- Wait for PLL to lock by polling PLLRIS */
161
       while ((SYSCTL RIS R &0x00000040) == 0) {};
       /* 9- Setting BYPASS to 0, select PLL as the source of system clock */
162
       SYSCTL RCC2 R &= ~0x00000800;
163
                                                                  //RCC2[11]=1
164 }
165
166 /* Function for initializing SysTick counter */
167 void SysTick init (void)
168 -{ NVIC ST CTRL R =0;
                                             // Disabling
      NVIC ST RELOAD R =0x00FFFFFF;
                                             // Max reload value
      NVIC ST CURRENT R =0;
170
                                             // Any write clears it
      NVIC ST CTRL R =0x000000005;
171
                                             // Enabling SysTick with core clock
172 }
173
```

```
LAB6.c startup TM4C123.s
   172 }
   173
   174 /* Function to determine reload value (how many clock ceycles to wait).40MHz (25ns period) clock */
   175 void SysTick Wait (unsigned long reload)
   176 - { NVIC ST RELOAD R = reload;
   177
        NVIC ST CURRENT R = 0;
   178
         while((NVIC ST CTRL R & 0x00010000) == 0) {} //Waiting for flag
   179 }
   180
   181 /* Function for wait n ms (considering 40MHz clock) */
   182 void SysTick Wait n ms (unsigned long n)
   183 -{ unsigned long i;
   184 for (i=0 ; i<n ; i++)
   185
        {SysTick Wait (39999);}
   186 }
   187 -
   188 /* Function to initilaize Port B */
   189 void Port B init ()
   190 ⊟{ SYSCTL RCGCGPIO R |= 0x02 ;
                                                        // GPIO Port B Run Mode Clock Gating Control
   191
         while ((SYSCTL PRGPIO R & 0x02) == 0) {};
                                                      // Waiting for preipherals to be ready
   192
         GPIO PORTB DIR R |= ~0x10;
                                                        // Specifie PB4 input (Next botton)
   193
            GPIO PORTB DEN R |= 0x10;
                                                       // Enabling pin as digital
   194
            GPIO PORTB DATA R = 0x00;
                                                       // Intialize data to 0
   195 }
   196 -
   197 /* Function to initilaize Port A */
   198 void Port A init ()
   199 -{ SYSCTL RCGCGPIO R |= 0x01;
                                                     // GPIO Port A Run Mode Clock Gating Control
   200
            while((SYSCTL PRGPIO R & 0x01) == 0){}; // Waiting for preipherals to be ready
   201
            GPIO PORTA DIR R = 0xE0;
                                                    // Specifie PA7-5 output (The three Locks(LEDs) respectively)
   202
            GPIO PORTA DEN R = 0xE0;
                                                     // Enabling pins as digital
   203
            GPIO PORTA DATA R = 0;
                                                     // Intialize data to 0
   204 }
   205
<
```

```
LAB6.c
         startup_TM4C123.s
 206 /* Function to initilaize Port F */
 207 void Port F init ()
 208 ⊟{ SYSCTL RCGCGPIO R |= 0x20 ;
                                                    // GPIO Port F Run Mode Clock Gating Control
 209
          while ((SYSCTL PRGPIO R & 0x20) == 0) {};
                                                   // Waiting for preipherals to be ready
          GPIO PORTF LOCK R = 0x4c4f434b;
                                                    // Unlocking Port F
 210
 211
          GPIO PORTF CR R = 0xFF;
                                                    // Allow changing to the pins
 212
          GPIO PORTF PUR R |= 0x11;
                                                    // Enabling Pull-Up resistors for PF4 and PF0
 213
          GPIO PORTF DIR R = ~0x11;
                                                    // Specifie PF4 and PF0 inputs (Set locks/open locks bottons respectively)
 214
          GPIO PORTF DEN R = 0x11;
                                                   // Enabling digital functionality
          GPIO PORTF DATA R = 0;
                                                    // Intialize data to 0
 215
 216 }
 217
 218
 219 - void ADCO PE3 init (void) {
 220
 221
 222
                                               // Activate ADCO
        SYSCTL RCGCADC R |= 0x01;
 223
        SYSCTL RCGCGPIO R |= 0x10;
                                               // Activate clock for Port E
 224
        while((SYSCTL PRGPIO R&Ox10) == 0){}; // Wait for stabilization
        GPIO PORTE DIR R &= ~0x08;
 225
                                               // Make PE3 input
        GPIO PORTE AFSEL R |= 0x08;
 226
                                               // Enable alternate function on PE3
 227
        GPIO PORTE DEN R &= ~0x08;
                                               // Disable digital I/O on PE3
 228
        GPIO PORTE AMSEL R |= 0x08;
                                               // Enable analog functionality on PE3
 229
        ADCO PC R &= ~0xF;
                                               // Clear sampling rate
                                               // Configure for 125K samples/sec
 230
        ADCO PC R |= 0x1;
                                               // Sequencer 3 is highest priority
 231
        ADCO SSPRI R = 0 \times 0123;
        ADCO ACTSS R &= ~0x0008;
                                               // Disable sample sequencer 3
 232
 233
        ADCO EMUX R &= ~0xF000;
                                               // Seg3 is software trigger
 234
        ADCO SSMUX3 R &= ~0xF;
                                              // Clear channel selection
 235
        ADCO SSMUX3 R += 0;
                                              // Set channel AINO
236
        ADCO SSCTL3 R = 0x6:
                                              // No TSO DO, Yes IEO ENDO
        ADCO ACTSS R |= 0x8;
                                              // Enable SS3
 237
 238
 239
```

```
LAB6.c
           startup TM4C123.s
   218
   219 -void ADCO PE3 init (void) {
   221
   222
           SYSCTL RCGCADC R |= 0x01;
                                                  // Activate ADCO
   223
          SYSCTL RCGCGPIO R |= 0x10;
                                                  // Activate clock for Port E
   224
           while((SYSCTL PRGPIO R&0x10) == 0){}; // Wait for stabilization
          GPIO PORTE DIR R &= ~0x08;
   225
                                                  // Make PE3 input
   226
           GPIO PORTE AFSEL R |= 0x08;
                                                  // Enable alternate function on PE3
   227
           GPIO PORTE DEN R &= ~0x08;
                                                  // Disable digital I/O on PE3
   228
          GPIO PORTE AMSEL R |= 0x08;
                                                  // Enable analog functionality on PE3
   229
          ADCO PC R &= ~0xF;
                                                  // Clear sampling rate
                                                  // Configure for 125K samples/sec
   230
          ADCO PC R |= 0x1;
   231
          ADCO SSPRI R = 0x0123;
                                                  // Sequencer 3 is highest priority
   232
          ADCO ACTSS R &= ~0x0008;
                                                  // Disable sample sequencer 3
   233
          ADCO EMUX R &= ~0xF000;
                                                  // Seg3 is software trigger
   234
          ADCO SSMUX3 R &= ~0xF;
                                                 // Clear channel selection
   235
          ADCO SSMUX3 R += 0;
                                                 // Set channel AINO
   236
          ADCO SSCTL3 R = 0x6;
                                                 // No TSO DO, Yes IEO ENDO
          ADCO ACTSS R |= 0x8;
                                                  // Enable SS3
   237
   238
   239
   240 /* Function convert from ADC value to readale voltage */
   241 unsigned long resistanceRead ()
   242 -{ unsigned long ADC result, R;
   243
   244
   245
            ADCO PSSI R = 0x0008;
                                                      // Initiate SS3
                                                      // Wait for A/D conversion to finish
   246
            while ((ADCO RIS R&0x08) == 0) {};
   247
            ADC result = ADCO SSFIFO3 R&OxFFF;
                                                      // Get ADC result
            ADCO ISC R = 0x00008;
   248
                                                      // Clear interrupt status for SS3 to be able to start sampling again
   249
            return R = (ADC result)*(10000/4095.0); // Convert ADC value to resistance by quantizing
   250 -1
<
```

Video Link

https://youtu.be/vYyftYnwnLE

AIKEN questions

Digital Sample = (Analog Input x 4095) / max analog value.

A) TRUE

B) FALSE

ANSWER: A

Which of the following register is used to read the ADC0 result?

A) ADC0_SSPRI_R

B) ADC0_SSMUX3_R

C) ADC0_SSCTL3_R

D) ADC0_SSFIFO3

ANSWER: D

Which of the following is not one of the four limitations exist when sampling data?

- A) Amplitude resolution
- B) Amplitude range
- C) Time quantization
- D) ADC pins number

ANSWER: D

Which of the following enables sample sequencer 1?

- A) ADC0_ACTSS_R = 0x0008;
- B) ADC0_ACTSS_R = 0x0004;
- C) ADC0_ACTSS_R = 0x0002;
- D) ADC0_ACTSS_R = 0x0001;

ANSWER: C

Which of the following enables alternate function on PE3?

- A) GPIO PORTE AMSEL R = 0x08;
- B) GPIO_PORTE_AFSEL_R $\mid = -0x08$;
- C) GPIO_PORTE_AMSEL_R = 0xF7;
- D) GPIO_PORTE_AFSEL_R $\mid = -0xF7$;

ANSWER: D

When using "switch" in C, we put "break;" after each case so the program go into the next case.

A) TRUE

B) FALSE

ANSWER: B

ADC0_PC_R register is used to specify the sampling rate.

A) TRUE

B) FALSE

ANSWER: A