Meenakshi Shankar
Sr. No 22400
M.Tech EPD

EMBEDDED SYSTEM DESIGN (E3-257)

LAB ASSIGNMENT – 8

→Controlling the Motor speed and Blink rate using PWM

1. The Configuration of registers is set such that PE5 will be the PWM output for motor driving and PE3 is the input for ADC.

```
// Enable Peripheral Clocks
              SYSCTL_RCGCPWM_R |= 1;       // Enable clock to PWM0
              SYSCTL_RCGCGPIO_R |= 0x00000010;        // Enable clock at GPIOE (for ADC0)
              SYSCTL_RCC_R |= 0x001E0000;  // Enable divisor for PWM clock (clk/64)
              // Enable port PE5 for PWM0 M0PWM5
              GPIO_PORTE_AFSEL_R = 0x20;   // PE5 uses alternate function
              GPIO_PORTE_PCTL_R &= ~0x00F00000; // Make PE5 PWM output pin
              GPIO_PORTE_PCTL_R |= 0x00400000;  // Function 4 of pin PE5 -> M0PWM5
              GPIO_PORTE_DEN_R |= 0x20;    // Digital enable for PE5
              PWM0_2_CTL_R = 0;            // Stop counter
              PWM0_2_GENB_R = 0x0000008C;  // M0PWM5 output set when reload
                                           // Clear when match PWMCMPA
              PWM0_2_LOAD_R = 5000;        // Set load value for 50Hz : ((16MHz/64) / 50 Hz) = 5000
              PWM0_2_CMPA_R = 4400;        // set duty cycle to 180 degree for servo
              PWM0_2_CTL_R = 1;            // start timer
              PWM0_ENABLE_R = 0x20;        // Start PWM0 ch5
//
              SYSCTL_RCGCGPIO_R |= 0x01;
              GPIO_PORTA_DIR_R |= (1<<3)|(1<<2);
              GPIO_PORTA_DEN_R |= (1<<3)|(1<<2);
              GPIO_PORTA_DR8R_R |= (1<<2);

              GPIO_PORTA_DATA_R |= (1<<3);
              GPIO_PORTA_DATA_R &= ~(1<<2);
              GPIO_PORTA_DATA_R &= ~(1<<3);
              GPIO_PORTA_DATA_R |= (1<<2);
```

```
ADC0_ACTSS_R |= 0x00000008; /* enable ADC0 sequencer 3 */
ADC0_EMUX_R &= ~0xF000; /* software trigger conversion */
ADC0_SSMUX3_R = 0; /* get input from channel 0 */
/* initialize PE3 for AIN0 input */
GPIO_PORTE_AFSEL_R |= 8; /* enable alternate function */
GPIO_PORTE_DEN_R &= ~8; /* disable digital function */
GPIO_PORTE_AMSEL_R |= 8; /* enable analog function */
ADC0_SSCTL3_R |= 6; /* take one sample at a time, set flag at 1st sample */
ADC0_ACTSS_R |= 8; /* enable ADC0 sequencer 3 */
```

2. We are capturing the new updated adc value and displaying the that change on the SSD for 5 seconds.
   This is done by configuring a timer_sec value which reads any new adc value for the difference with respect to current time ; if its > 5 sec then the flag is set back to zero for recognizing the next 5 secs.
   An update is only considered if the difference in values is atleast 80.

Meenakshi Shankar
Sr. No 22400
M.Tech EPD

```
262 if(abs(temp_upd - result) > 80)
263 {
264     adc_change  = 1;
265     TIMER1_TAILR_R =  map(result, 10,4050,62500,3125);
266     pre_tim = time_sec;
267
268
269
270 }
271
272 if(time_sec-pre_tim >50)
273 {
274     adc_change = 0;
275 }
```

3. Blink rate is also adjusted based on a Timer.
   The timer is configured as below:

```
188 //timer setup
189 SYSCTL_RCGCTIMER_R |= 1<<1;
190 TIMER1_CTL_R &= (~(1<<0));
191 TIMER1_CFG_R = 0X4;
192 TIMER1_TAMR_R = 0X2;
193 TIMER1_TAPR_R |= 0XFF;
194 TIMER1_TAILR_R = 6250;
195 TIMER1_ICR_R |= 1<<0;
196 TIMER1_CTL_R |= 1<<0;
197 TIMER1_IMR_R |= 1<<0;
198 NVIC_EN0_R |= 0X200000; // INTERRUPT 21
199
200
```

In the handler for the timer two flags named start and stop are checked and toggle the GPIO_PORTF_DATA register to control whether blinking should take place or not.

If the ADC value is >4050 then:
We set the start flag to 1;
Blink is not allowed and it's a constant glowing LED.

If the ADC value is < 50 then:
We set the stop flag to 0;
Blink is not allowed and it's a constant off LED.

If the ADC value is in between then:
We set the stop flag to 0 as well as start =0;
Blinking is allowed.

Meenakshi Shankar
Sr. No 22400
M.Tech EPD

```
711 void Timer1_Handler(void)
712 {
713     if(start)
714         GPIO_PORTF_DATA_R |= 1<<1;
715     else if(stop)
716         GPIO_PORTF_DATA_R &= ~(1<<1);
717     else
718         GPIO_PORTF_DATA_R ^= 1<<1;
719     TIMER1_ICR_R  |= 1<<0;
720 }
 721
```

Mapping to the rate is done as below

```
265     TIMER1_TAILR_R =  map(result, 10,4050,62500,3125);
```

4. Configuring UART commands
   All previous commands Timer start/stop/pause/resume are available.
   And can also be operated via switches.

   “Set to xxx”
   Command is used to allow down counting in seconds.
   The down counting operation is taken care of in the Systick Handler using a set_dwn_flag.

```
446 else if((strcmp(cmnd.type, "setto")==0))
447 {
448
449     set_val = atoi(cmnd.data);
450
451     m_sec = 0;
452     sec = set_val;
453
454
455     flag = 1;
456
457     NVIC_ST_CTRL_R |= 7;
458
459     set_dwn_flag = 1;
460
461    // pause_stat = 1;
462
463     printstring("Time Set");
464
465     printstring("\n\r");
466
467
468     printstring("Valid Entry\n\r");
469     check= 1;
470
471 }
```

Meenakshi Shankar
Sr. No 22400
M.Tech EPD

```
724 void SysTick_Handler(void)
725 {
726
727
728     if(set_dwn_flag == 0)
729
730      {
731          time_sec++;
732          if(m_sec==9)
733              {
734                  sec++;
735              }
736
737          m_sec>8?(m_sec = 0 ):(m_sec++);
738
739
740      }
741
742
743
744     else if(set_dwn_flag == 1)
745      {
746
747         if(sec > 1)
748         {
749             sec--;
750             delayMs(1000);
751         }
752
753         if(sec == 1)
754              {
755                  //m_sec++;
756                  set_dwn_flag = 0;
757
758              }
759
760
```

For the switches operation the GPIOF_Handler is set and configured as in previous assignments.