

Joseph Sharp Halpin
CpE 403 Section 1001
10/9/2018

Task 01:

Youtube Link: <https://youtu.be/wzHs0uEpBBw>

Code:

```
main.c
1 #include <stdint.h>
2 #include <stdbool.h>
3 #include "inc/hw_memmap.h"
4 #include "inc/hw_types.h"
5 #include "driverlib/sysctl.h"
6 #include "driverlib/gpio.h"
7 #include "driverlib/debug.h"
8 #include "driverlib/pwm.h"
9 #include "driverlib/pin_map.h"
10 #include "inc/hw_gpio.h"
11 #include "driverlib/rom.h"
12
13
14 #define PWM_FREQUENCY 55
15
16 int main(void)
17 {
18     volatile uint32_t ui32Load;
19     volatile uint32_t ui32PWMClock;
20     volatile uint8_t ui8Adjust;
21     ui8Adjust = 83;
22
23     //set clock rate
24     ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
25     //set PWM rate
26     ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);
27
28     //enable PWM1
29     ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
30     //enable GPIOD
31     ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
32     //enable GPIOF
33     ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
34
35     //set output PWM pin
36     ROM_GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0);
37     ROM_GPIOPinConfigure(GPIO_PD0_M1PWM0);
38
39     //allow SW0 and SW1 to operate
40     HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
41     HWREG(GPIO_PORTF_BASE + GPIO_O_CR) |= 0x01;
42     HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = 0;
43     ROM_GPIODirModeSet(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0, GPIO_DIR_MODE_IN);
44     ROM_GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);
45
46     //get PWM clock to set the period
47     ui32PWMClock = SysCtlClockGet() / 64;
48     ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
49     PWMGenConfigure(PWM1_BASE, PWM_GEN_0, PWM_GEN_MODE_DOWN);
50     PWMGenPeriodSet(PWM1_BASE, PWM_GEN_0, ui32Load);
51 }
```

```

51
52 //initialize the duty cycle and enable the PWM
53 ROM_PWM_PulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000);
54 ROM_PWM_OutputState(PWM1_BASE, PWM_OUT_0_BIT, true);
55 ROM_PWM_GenEnable(PWM1_BASE, PWM_GEN_0);
56
57 while(1)
58 {
59     //check if SW1 is pressed and adjust the servo motor
60     if(ROM_GPIO_PinRead(GPIO_PORTF_BASE, GPIO_PIN_4) == 0x00)
61     {
62         ui8Adjust--;
63         if (ui8Adjust < 1)
64         {
65             ui8Adjust = 1;
66         }
67         ROM_PWM_PulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000);
68     }
69     //check if SW0 is pressed and adjust the servo motor
70     if(ROM_GPIO_PinRead(GPIO_PORTF_BASE, GPIO_PIN_0) == 0x00)
71     {
72         ui8Adjust++;
73         if (ui8Adjust > 120)
74         {
75             ui8Adjust = 120;
76         }
77         ROM_PWM_PulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000);
78     }
79     //delay for 100000 cycles
80     ROM_SysCtlDelay(100000);
81 }
82
83 }

```

Task 02:

Youtube Link: <https://youtu.be/6oyUPCQ3CdA>

Code:

```

1 #include <stdint.h>
2 #include <stdbool.h>
3 #include "inc/hw_memmap.h"
4 #include "inc/hw_types.h"
5 #include "driverlib/sysctl.h"
6 #include "driverlib/gpio.h"
7 #include "driverlib/debug.h"
8 #include "driverlib/pwm.h"
9 #include "driverlib/pin_map.h"
10 #include "inc/hw_gpio.h"
11 #include "driverlib/rom.h"
12
13 #define PWM_FREQUENCY 55
14
15 int main(void)
16 {
17     volatile uint32_t ui32Load;
18     volatile uint32_t ui32PWMClock;
19     int i;
20
21     //set clock rate
22     ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
23
24     //enable PWM1 and GPIOF
25     ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
26     ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
27
28     //set PWM clock rate
29     ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);
30
31     //configure the PWM and PF1
32     HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
33     HWREG(GPIO_PORTF_BASE + GPIO_O_CR) |= 0x01;
34     ROM_GPIOPinConfigure(GPIO_PFI_M1PWM5);
35     ROM_GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_1);
36
37     //configure the PWM
38     PWMGenConfigure(PWM1_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN);
39     PWMGenConfigure(PWM1_BASE, PWM_GEN_3, PWM_GEN_MODE_DOWN);
40
41     //get the period
42     ui32PWMClock = SysCtlClockGet() / 64;
43     ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
44
45     //set the PWM period
46     PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, ui32Load);
47     PWMGenPeriodSet(PWM1_BASE, PWM_GEN_3, ui32Load);
48
49     //set the PWM duty cycle
50     PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui32Load * 0.1);
51
52     //enable PWM1
53     PWMGenEnable(PWM1_BASE, PWM_GEN_2);
54     PWMGenEnable(PWM1_BASE, PWM_GEN_3);
55
56     PWMOutputState(PWM1_BASE, PWM_OUT_5_BIT, true);
57
58     while(1)
59     {
60         //go from 10% duty cycle to 90%
61         for(i = ui32Load * 0.1; i < ui32Load * 0.9; i++)
62         {
63             //reset the duty cycle
64             PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, i);
65             ROM_SysCtlDelay(20000);
66         }
67     }
68 }

```

Task 03:

Youtube Link: <https://youtu.be/-9-JnY1XDV4>

Code:

```
main.c
1 #include <stdint.h>
2 #include <stdbool.h>
3 #include "inc/hw_memmap.h"
4 #include "inc/hw_types.h"
5 #include "driverlib/sysctl.h"
6 #include "driverlib/gpio.h"
7 #include "driverlib/debug.h"
8 #include "driverlib/pwm.h"
9 #include "driverlib/pin_map.h"
10 #include "inc/hw_gpio.h"
11 #include "driverlib/rom.h"
12
13
14 #define PWM_FREQUENCY 55
15
16 int main(void)
17 {
18     volatile uint32_t ui32Load;
19     volatile uint32_t ui32PWMClock;
20     int i, j, k;
21
22     //set clock rate
23     ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
24
25     //enable PWM1 and GPIOF
26     ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
27     ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
28
29     //set PWM clock rate
30     ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);
31
32     //allow PWM to be edited
33     HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
34     HWREG(GPIO_PORTF_BASE + GPIO_O_CR) |= 0x01;
35
36     //configure the LED pins
37     ROM_GPIOPinConfigure(GPIO_PF1_M1PWM5);
38     ROM_GPIOPinConfigure(GPIO_PF2_M1PWM6);
39     ROM_GPIOPinConfigure(GPIO_PF3_M1PWM7);
40     ROM_GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3);
41
42     //configure the PWM
43     PWMGenConfigure(PWM1_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN);
44     PWMGenConfigure(PWM1_BASE, PWM_GEN_3, PWM_GEN_MODE_DOWN);
45
46     //get the period
47     ui32PWMClock = SysCtlClockGet() / 64;
48     ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
49 }
```

```

46 //get the period
47 ui32PWMClock = SysCtlClockGet() / 64;
48 ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
49
50 //set the PWM period
51 PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, ui32Load);
52 PWMGenPeriodSet(PWM1_BASE, PWM_GEN_3, ui32Load);
53
54 //set the duty cycle of the PWM
55 PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui32Load * 0.9);
56 PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6, ui32Load * 0.9);
57 PWMPulseWidthSet(PWM1_BASE, PWM_OUT_7, ui32Load * 0.9);
58
59 //enable the PWM
60 PWMGenEnable(PWM1_BASE, PWM_GEN_2);
61 PWMGenEnable(PWM1_BASE, PWM_GEN_3);
62
63 //set the PWM to output
64 PWMOutputState(PWM1_BASE, PWM_OUT_5_BIT | PWM_OUT_6_BIT | PWM_OUT_7_BIT, true);
65
66 while(1)
67 {
68     //make the red led go from 90% duty cycle to 10%
69     for(i = ui32Load * 0.9; i > ui32Load * 0.1; i--)
70     {
71         PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, i);
72         ROM_SysCtlDelay(10000);
73         //make the green led go from 90% duty cycle to 10%
74         for(j = ui32Load * 0.9; j > ui32Load * 0.1; j--)
75         {
76             PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6, j);
77             ROM_SysCtlDelay(10000);
78             //make the blue led go from 90% duty cycle to 10%
79             for(k = ui32Load * 0.9; k > ui32Load * 0.1; k--)
80             {
81                 PWMPulseWidthSet(PWM1_BASE, PWM_OUT_7, k);
82                 ROM_SysCtlDelay(10000);
83             }
84         }
85     }
86 }
87 }

```