Enrique Saldana

https://github.com/enri10

Date Submitted: 11/01/18

Task 00: Execute the supplied code, no submission required.

LAB06 Task00 : https://youtu.be/JUPFVbQ5wbY

Task 01: Change the PWM duty cycle to make the servo motor to do a loop of a complete sweep from 0 to 180 deg.

LAB06 Task01 : https://youtu.be/SjWgACQkBX0

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "driverlib/debug.h"
#include "driverlib/pwm.h"
#include "driverlib/pin_map.h"
#include "inc/hw_gpio.h"
#include "driverlib/rom.h"

//50Hz base frequency to control servo(20ms period)
#define PWM_FREQUENCY 50

int main(void)
{
    //Variables used to program the PWM
    //20 is the 0 degree position
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    volatile uint8_t ui8Adjust;
    ui8Adjust = 20;

    //Run CPU at 40MHz
    //Run PWM clock at 40MHz/64 = 625kHz

ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);

    //Enable PWMI, GPIOD,and GPIOF modules
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
```

```
    //Configure PDO as a PWM output pin for module 1
    ROM_GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0);
    ROM_GPIOPinConfigure(GPIO_PD0_M1PWM0);

    //Configure module 1 PWM generator 0 as a down-counter and load the count value
    ui32PWMClock = SysCtlClockGet() / 64;
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
    PWMGenConfigure(PWM1_BASE, PWM_GEN_0, PWM_GEN_MODE_DOWN);
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_0, ui32Load);

    //Set pulse-width
    //PWM module 1, generator 0 is enable as an output
    //Enabled to run
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000);
    ROM_PWMOutputState(PWM1_BASE, PWM_OUT_0_BIT, true);
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_0);

    while(1)
    {
        //Sweep of 0 to 180 in increments of 0.1 ms
        //When servo reaches 180, it returns to 0 degree
        if (ui8Adjust > 115)
        {
            ui8Adjust = 20;
        }
        else
        {
            ui8Adjust = ui8Adjust +5;
        }
        // //Load the PWM pulse width register with the new value
        ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui8Adjust * ui32Load / 1000);

        //Speed of the loop
        ROM_SysCtlDelay(3000000);
    }

}
```

Task 02: Change PWM duty cycle from 10% to 90% to control the brightness of the LED at PF1.

LAB06 Task02 : https://youtu.be/axFRau8zu58

Used Generator 2 in order to use PF1 as a PWM output

The values of 10 and 90 will give me duty cycles of 10% and 90% respectively

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
```

```c
#include "driverlib/gpio.h"
#include "driverlib/debug.h"
#include "driverlib/pwm.h"
#include "driverlib/pin_map.h"
#include "inc/hw_gpio.h"
#include "driverlib/rom.h"

//55Hz base frequency to control servo
#define PWM_FREQUENCY 55

int main(void)
{
    //Variables used to program the PWM
    //Start LED at around 50% duty cycle
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    volatile uint8_t ui8Adjust;
    ui8Adjust = 50;

    //Run CPU at 40MHz
    //Run PWM clock at 40MHz/64 = 625kHz

ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);

    //Enable PWMI,and GPIOF modules
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

    //Configure PF1(Red LED) as output
    ROM_GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1);

    //Configure PF1 as a PWM output pin for module 1
    ROM_GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_1);
    ROM_GPIOPinConfigure(GPIO_PF1_M1PWM5);

    //Unlock the GPIO commit control register
    //Configure PF0 and PF4 as inputs
    //Configure the internal pull-up resistors on both pins
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
    HWREG(GPIO_PORTF_BASE + GPIO_O_CR) |= 0x01;
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = 0;
    ROM_GPIODirModeSet(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0, GPIO_DIR_MODE_IN);
    ROM_GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU);

    //Configure module 1 PWM generator 2 as a down-counter and load the count value
    ui32PWMClock = SysCtlClockGet() / 64;
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
    PWMGenConfigure(PWM1_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN);
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, ui32Load);

    //Set pulse-width
    //PWM module 1, generator 2 is enabled as an output
    //Enabled to run
```

```
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load / 1000);
    ROM_PWMOutputState(PWM1_BASE, PWM_OUT_5_BIT, true);
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_2);

    while(1)
    {
        //Read PF4(SW1)
        //ui8Adjust is decremented until it reaches 10% duty cycle
        //Load the PWM pulse width register with the new value
        if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_4)==0x00)
        {
            ui8Adjust--;
            if (ui8Adjust < 10)
            {
                ui8Adjust = 10;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load / 1000);
        }

        //Read PF0(SW2)
        //Pulse width is incremented until it reaches 90% duty cycle
        //Load the PWM pulse width register with the new value
        if(ROM_GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_0)==0x00)
        {
            ui8Adjust++;
            if (ui8Adjust > 90)
            {
                ui8Adjust = 90;
            }
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load / 1000);
        }

        //Speed of the loop
        ROM_SysCtlDelay(50000);
    }

}
```

Task 03: Change PWM duty cycle from 90% to 10% to control the brightness of the all three LED at PF1, PF2, and PF3 using three nested "for loops".

LAB06 Task03 : https://youtu.be/tnsuxeBblYQ

Used generator 3 in order to use PF2 and PF3 as PWM outputs

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
```

```
#include "driverlib/gpio.h"
#include "driverlib/debug.h"
#include "driverlib/pwm.h"
#include "driverlib/pin_map.h"
#include "inc/hw_gpio.h"
#include "driverlib/rom.h"

//55Hz base frequency to control servo
#define PWM_FREQUENCY 55

int main(void)
{
    //Variables used to program the PWM
    //Start LED at around 90% duty cycle
    volatile uint32_t ui32Load;
    volatile uint32_t ui32PWMClock;
    volatile uint8_t ui8Adjust;
    ui8Adjust = 90;

    //Variables to individually control each LED
    volatile uint8_t red;
    volatile uint8_t green;
    volatile uint8_t blue;

    //Run CPU at 40MHz
    //Run PWM clock at 40MHz/64 = 625kHz

ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);

    //Enable PWMI,and GPIOF modules
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

    //Configure LEDs as outputs
    ROM_GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    //Configure PF1,PF2,and PF3 as a PWM output pins for module 1
    ROM_GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
    ROM_GPIOPinConfigure(GPIO_PF1_M1PWM5);
    ROM_GPIOPinConfigure(GPIO_PF2_M1PWM6);
    ROM_GPIOPinConfigure(GPIO_PF3_M1PWM7);


    //Unlock the GPIO commit control register
    //Configure PF0 and PF4 as inputs
    //Configure the internal pull-up resistors on both pins
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
    HWREG(GPIO_PORTF_BASE + GPIO_O_CR) |= 0x01;
    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = 0;
    ROM_GPIODirModeSet(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0, GPIO_DIR_MODE_IN);
    ROM_GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4|GPIO_PIN_0, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU);

    //Configure module 1 PWM generator 2 as a down-counter and load the count value
```

```
    //Configure module 1 PWM generator 3 as a down-counter and load the count value
    ui32PWMClock = SysCtlClockGet() / 64;
    ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
    PWMGenConfigure(PWM1_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN);
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, ui32Load);
    PWMGenConfigure(PWM1_BASE, PWM_GEN_3, PWM_GEN_MODE_DOWN);
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_3, ui32Load);

    //Set pulse-width for the 3 LED's
    //PWM module 1, generator 2 and 3 are enabled as outputs
    //Enabled to run
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load / 1000);
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6, ui8Adjust * ui32Load / 1000);
    ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_7, ui8Adjust * ui32Load / 1000);
    ROM_PWMOutputState(PWM1_BASE, PWM_OUT_5_BIT|PWM_OUT_6_BIT|PWM_OUT_7_BIT, true);
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_2);
    ROM_PWMGenEnable(PWM1_BASE, PWM_GEN_3);

    while(1)
    {
        //Set RGB values to 90% to output a bright white light
        red = 90;
        green =90;
        blue = 90;
        ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, red * ui32Load / 1000);
        ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6, green * ui32Load / 1000);
        ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_7, blue * ui32Load / 1000);
        //Delay to have bright white light visible for a couple seconds
        ROM_SysCtlDelay(50000000);

        //Nested loop that will cycle RGB LEDS from 90% to 10%
        //Delays are there to slow down the change from 90% to 10%
        for(red=90;red>10; red--)
        {
            ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, red * ui32Load / 1000);
            ROM_SysCtlDelay(200);
            for(green=90; green>10; green--)
            {
              ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6, green * ui32Load / 1000);
              ROM_SysCtlDelay(200);
              for(blue=90; blue>10; blue--)
              {
                  ROM_PWMPulseWidthSet(PWM1_BASE, PWM_OUT_7, blue * ui32Load / 1000);
                  ROM_SysCtlDelay(200);
              }
            }
        }
        //Delay to have the dull white light stay for a couple seconds
        ROM_SysCtlDelay(50000000);
    }

}
```