```
/*******************************************************************************
 Module
     E128_Servo.c

 Description
     servo output for testing

 Notes
     sets rollover period to 21.85mS ased on 24MHz clock.
     Starts low, goes high at [period (0xffff) minus desired pulse length]
     DYL edited code 02/26/13
     Because servo line goes thru an inverting Schmitt trigger, line now starts high (maybe)
     and is set low on OC and then toggles on overflow

 History
 When           Who     What/Why
 -------------- ---     --------
 02/25/13 22:00 BLS     adapted Ed's servolib for our purposes
 02/26/13 01:30 DYL     mapped to port U4 --> Timer2 Channel6

*******************************************************************************/

/*----------------------------- Include Files -----------------------------*/
/* include header files for this state machine as well as any machines at the
   next lower level in the hierarchy that are sub-machines to this machine*/

#include <stdio.h>
#include <stdlib.h>
#include <mc9s12e128.h>
#include <S12e128bits.h>
#include <Bin_Const.h>
#include <termio.h>
#include <hidef.h>
#include "S12eVec.h"

#include "E128_PWM.h"         //has all prescale definitions
#include "E128_SPI.h"
#include "E128_Servo.h"
#include "FAC_FSM.h"
#include "NavigationFSM.h"
#include "AlignPPService.h"
#include "DriveTrainService.h"
#include "ArtilleryFSM.h"


/*----------------------------- Module Defines ----------------------------*/
#define TicksPerMicroSec   3

//Physical min and max limits of servo
#define MIN_POSITION       600             //-90 degrees
#define MAX_POSITION       2400            //+90 degrees
#define NEUTRAL_POSITION   1500            //0 degrees

//These need to be updated once the servo is mounted in position
#define LOADING_ANGLE   60//60            //This is our loading position
#define SHOOTING_ANGLE  150//150            //This is our shooting position

/*----------------------------- Module Code -----------------------------*/

//Init the servo timer on Timer2
void InitServoHardware(void)
{
    TIM2_TSCR1 = _S12_TEN;     //enable timer system
    TIM2_TSCR2 = PSCALE8;      //Divide by 8; 1 tick = .333 uS

    //Set up OC4 to time the PWM pulse to the servo
    TIM2_TIOS |= _S12_IOS6;    //set Timer2, Channel6 to output compare
```

```
    /* NO INTERRUPTS
    TIM2_TFLG1 = _S12_C6F;      //clear any existing flag coming out of reset
    TIM2_TIE |= _S12_C6I;       //enable Timer2 Channel6 local interrupt
    EnableInterrupts;           //this seems redudent, but why not
    */

    TIM2_TCTL1 |= (_S12_OM6);  //sets line low
    //TIM2_TCTL1 |= (_S12_OM6 | _S12_OL6); //sets line high
    TIM2_TTOV |= _S12_TOV6;    //set overflow toggle for Channel6

    ArtilleryServoLoad();      //initialize in "closed" servo gate

    //printf("Servo Hardware Initialized. \r\n");

}


//Shooting --> moves servo to shooting position
void ArtilleryServoShoot(void)
{
    SetAngle(SHOOTING_ANGLE);
}

//Loading --> moves servo to loading position
void ArtilleryServoLoad(void)
{
    SetAngle(LOADING_ANGLE);
}

void SetAngle(unsigned int InputAngle)
//need to convert an input angle to a time in uS
{
    static unsigned int ServoPosition;
    if ((InputAngle <= 180) && (InputAngle >= 0))
        {
            ServoPosition = 600 + 10*InputAngle; //scales InputAngle from 0-180 to 600-2400
            TIM2_TC6 = (0xffff - (ServoPosition * TicksPerMicroSec));
        }
    else
        {
            //printf("\n Invalid Servo PWM angle requested");
        }
}
```