```
//#define TEST

/****************************************************************************
 Description
   This module initializes the PWM capabilities on ports U0, U1, and U4 of the E128.

 Notes
   PWM_CHANNEL0 & PWM_CHANNEL1 driven in locked anti-phase for two wheel drive, change
   direction easily... easy to implement in hardware and software
   PWM_CHANNEL4 driven with PWM duty 0-100 --> corresponds from 0 - 18V


 Author
   Debbie Li 02/01/2013

 Revisions
 When           Who     What/Why
 -------------- ---     --------
 02/01/2013     BMJ     Changed code to initialize on 4 ports to drive 2 motors
                        in Anti-Phase Locked Drive mode. Conversation with Jina
                        determined that this would be easiest from an electronics
                        standpoint. Can revisit, and modify if we want to, no
                        problem
****************************************************************************/

/*----------------------------- 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 PERIOD  100 //7500 KHz Signal
#define FW_PERIOD 100//flywheel period

//These are our new parameters for setting full and half speed
#define FULL_FORWARD 79
#define FULL_REVERSE 21
#define HALF_FORWARD 68
#define HALF_REVERSE 32
#define FULL_ROTATE_1 27//68        //Remember to adjust both rotates 1 and 2, keep orientation
#define FULL_ROTATE_2 73
#define HALF_ROTATE_1 37   //37 is too slow
#define HALF_ROTATE_2 63   //63 is too slow
#define CLOSE_SHOT 19     //3ft //19
#define FAR_SHOT 21       //5ft
#define PP_SHOT 23        //long ways
#define LEVIATHAN_PWM 50
/*----------------------------- Module Variables --------------------------*/

/*static unsigned int Wheel0Speed = 0;
```

```
static unsigned int Wheel1Speed = 0;
static unsigned int Wheel0ThreeEdgeWidth = 1;
static unsigned int Wheel1ThreeEdgeWidth = 1;
*/

/*----------------------------- Module Prototypes -------------------------*/
//void interrupt _Vec_tim0ch4 Wheel0EncoderIR( void );
//void interrupt _Vec_tim0ch5 Wheel1EncoderIR( void );

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

void InitPWMHardware( void )
{
    //DRIVETRAIN
    MODRR |= _S12_MODRR0;   //map PWM0 to PORT U0 --> wheel 0
    MODRR |= _S12_MODRR1;   //map PWM1 to port U1 --> wheel 1
    //FLYWHEEL
    MODRR |= _S12_MODRR4;   //map PWM2 to PORT U4 --> flywheel

    //set PWM polarity output low
    //PWMPOL |= _S12_PPOL0;    //set output initially low
    //PWMPOL |= _S12_PPOL1;    //set output initially low
    //PWMPOL |= _S12_PPOL4;    //set output initially low - Hex inverter on FLY board


    //WANT 8KHz signal (based on Motor Characterization)
    //numTicks = (24MHz/7.5KHz)/PRESCALE_16/SCALE*2_1 = 100 Ticks for periodLength

    PWMCLK = _S12_PCLK0;  //use scaled clock, in addition to pre-scaling
    PWMCLK |= _S12_PCLK1; //use scaled clock, in addition to pre-scaling
    PWMPRCLK = PSCALE16;  //divide by 16
    PWMSCLA = 1;         //divide by SCALE*2

    PWMPER0 = PERIOD; //Set PWM Period Length
    PWMPER1 = PERIOD; //Set PWM Period Length
    PWMPER4 = FW_PERIOD; //PWM Period Length for flywheel

    PWMDTY0 = 50;            //initialize duty cycle to 50 --> off
    PWMDTY1 = 50;            //initialize duty cycle to 50 --> off
    PWMDTY4 = 0;             //initialize duty cycle to 0 --> off

    PWME |= _S12_PWME0;     //enable the PWM on port U0
    PWME |= _S12_PWME1;     //enable the PWM on port U1
    PWME |= _S12_PWME4;     //enable the PWM on port U4
}

void InitControlTimerHardware( void )
{
    TIM0_TSCR1 = _S12_TEN;  //Enable timer 0 system
    TIM0_TSCR2 = PSCALE128; //Divide by 128; 1 tick = 5.33 us

    //Setup the port as Output Compare (Timer0 Ch4)
    TIM0_TIOS |= (_S12_IOS4);

    //TIM0_TCTL1 |= (_S12_OM4 | _S12_OL4); //Defaults to pin disconnected

    //Initial setting of Compare Register
    TIM0_TC4 = 18750; //18750; //100ms

    //Initial Clear of the Timer Interrupt Flag
    TIM0_TFLG1 = _S12_C4F; //TIM0_TFLG1_C4F;


    //Enable Interrupts
    //TIM0_TIE |= _S12_C4I;   //Enabled after every Forward/Reverse FULL
    //Disabled after every non-Forward/Reverse FULL
```

```
    EnableInterrupts;
}

/*void InitEncoderHardware( void )
{
   TIM0_TSCR1 = _S12_TEN;  //Enable timer 0 system
   TIM0_TSCR2 = PSCALE128; //Divide by 128; 1 tick = 5.33 us

   //Setup the ports as Input Capture (PT0 and PT1)
   //TIM0_TIOS &= ~(_S12_IOS0 | _S12_IOS1 ); //Redundant, resets to zero

   // Set to interrupt on Rising Edges
   TIM0_TCTL3 |= _S12_EDG4A;
   //TIM0_TCTL3 |= _S12_EDG4B;
   TIM0_TCTL3 |= _S12_EDG5A;
   //TIM0_TCTL3 |= _S12_EDG5B;

   //Initial Clear of the Timer Interrupt Flag
   TIM0_TFLG1 = _S12_C4F; //TIM0_TFLG1_C4F;
   TIM0_TFLG1 = _S12_C5F; //TIM0_TFLG1_C5F;

   //Enable Interrupts
   TIM0_TIE |= _S12_C4I; //(TIM0_TIE_C4I);// | TIM0_TIE_C5I);
   TIM0_TIE |= _S12_C5I;
   EnableInterrupts;

   DDRT_DDRT2 = 1;

}
*/
/*****************************************************************************
 Function
    SetDutyCycle

 Parameters
   char Channel - can be a number from 0 to 3, represents the channel of PWM
   on Port U to set duty cylce

   char Duty - a percentage, can be a number from 0 to 100

 Returns
   Nothing

 Description
   This function sets the duty cycle of the channel selected

 Notes

 Author
   Brandon Jennings 02/01/2013
*****************************************************************************/
void SetDutyCycle( char Channel, char Duty )
{
    switch ( Channel )
        {
        case PWM_CHANNEL0 :
            if ((Duty <= 100) && (Duty >= 0))
                {
                    PWMDTY0 = (Duty*PERIOD)/100;
                }
            else
                {
                    //printf("An error occured. You entered an incorrect Duty Cycle value into
SetDutyCycle. \r\n");
                }
            break;
```

```
        case PWM_CHANNEL1 :
            if (Duty <= 100 && Duty >= 0)
                {
                    PWMDTY1 = (Duty*PERIOD)/100;
                }
            else
                {
                    //printf("\nAn error occured. You entered an incorrect Duty Cycle value into
SetDutyCycle. \r\n");
                }
            break;

        case PWM_CHANNEL4 :
            if (Duty <= 100 && Duty >= 0)
                {
                    PWMDTY4 = (Duty*FW_PERIOD)/100;
                }
            else
                {
                    //printf("\nAn error occured. You entered an incorrect Duty Cycle value into
SetDutyCycle. \r\n");
                }
            break;

        default :
            //No case was hit, throw an error!
            //printf("\nAn error occured. You entered an incorrect channel to SetDutyCycle. \r
\n");

            break;

            return;
        }
}

/*****************************************************************************
 Function
    DriveForwardFull

 Parameters
   None

 Returns
   Nothing

 Description
   When called, this function drives both motors in the forward direction

 Notes

 Author
   Brandon Jennings 02/01/2013
*****************************************************************************/
void DriveForwardFull( signed int deltaPWM )
{
    SetDutyCycle(PWM_CHANNEL0, FULL_FORWARD + deltaPWM);
    SetDutyCycle(PWM_CHANNEL1, FULL_FORWARD);
    //printf("F deltaPWM = %i, %i\r\n",deltaPWM, QueryTheta(SelfNum));
}

/*****************************************************************************
 Function
    DriveForwardFullHalf

 Parameters
   None
```

```
 Returns
   Nothing

 Description
   When called, this function drives both motors in the forward direction

 Notes

 Author
   Ben Sagan 03/01/2013
*****************************************************************************/
void DriveForwardHalf( void )
{
    SetDutyCycle(PWM_CHANNEL0, HALF_FORWARD);
    SetDutyCycle(PWM_CHANNEL1, HALF_FORWARD);

}

/*****************************************************************************
 Function
    DriveReverseFull

 Parameters
   None

 Returns
   Nothing

 Description
   When called, this function drives both motors in the reverse direction

 Notes

 Author
   Brandon Jennings 02/01/2013
*****************************************************************************/
void DriveReverseFull( signed int deltaPWM )
{
    SetDutyCycle(PWM_CHANNEL0, FULL_REVERSE + deltaPWM);
    SetDutyCycle(PWM_CHANNEL1, FULL_REVERSE);
    //printf("R deltaPWM = %i, %i\r\n",deltaPWM, QueryTheta(SelfNum));

}

/*****************************************************************************
 Function
    DriveReverseHalf

 Parameters
   None

 Returns
   Nothing

 Description
   When called, this function drives both motors in the reverse direction

 Notes

 Author
   Brandon Jennings 02/01/2013
*****************************************************************************/
void DriveReverseHalf( void )
{
    SetDutyCycle(PWM_CHANNEL0, HALF_REVERSE);
    //SetWheel0Speed( 47 );
    SetDutyCycle(PWM_CHANNEL1, HALF_REVERSE);
```

```
}

/*****************************************************************************
 Function
    RotateClockwise

 Parameters
   None

 Returns
   Nothing

 Description
   When called, this function drives both motors in the opposite directions

 Notes

 Author
   Brandon Jennings 02/01/2013
*****************************************************************************/
void RotateClockwise( void )
{
    SetDutyCycle(PWM_CHANNEL0, FULL_ROTATE_1);
    //SetWheel0Speed( 47 );
    SetDutyCycle(PWM_CHANNEL1, FULL_ROTATE_2);
    //puts("rotate clockwise\r\n");

}

/*****************************************************************************
 Function
    RotateClockwiseHalf

 Parameters
   None

 Returns
   Nothing

 Description
   When called, this function drives both motors in the opposite directions

 Notes

 Author
   Ben Sagan 03/01/2013
*****************************************************************************/
void RotateClockwiseHalf( void )
{
    SetDutyCycle(PWM_CHANNEL0, HALF_ROTATE_1);
    //SetWheel0Speed( 47 );
    SetDutyCycle(PWM_CHANNEL1, HALF_ROTATE_2);
    //puts("rotate halfcw \r\n");


}

/*****************************************************************************
 Function
    RotateCounterClockwise

 Parameters
   None

 Returns
   Nothing
```

```
 Description
   When called, this function drives both motors in the reverse direction

 Notes

 Author
   Brandon Jennings 02/01/2013
*******************************************************************************/
void RotateCounterClockwise( void )
{
    SetDutyCycle(PWM_CHANNEL0, FULL_ROTATE_2);
    //SetWheel0Speed( 47 );
    SetDutyCycle(PWM_CHANNEL1, FULL_ROTATE_1);
    //puts("rotate ccw\r\n");
}

/*******************************************************************************
 Function
    RotateCounterClockwiseHalf

 Parameters
   None

 Returns
   Nothing

 Description
   When called, this function drives both motors in the reverse direction

 Notes

 Author
   Ben Sagan 03/01/2013
*******************************************************************************/
void RotateCounterClockwiseHalf( void )
{
    SetDutyCycle(PWM_CHANNEL0, HALF_ROTATE_2);
    //SetWheel0Speed( 47 );
    SetDutyCycle(PWM_CHANNEL1, HALF_ROTATE_1);
}

/*******************************************************************************
 Function
    StopMotor

 Parameters
   None

 Returns
   Nothing

 Description
   When called, this function stops both motors

 Notes

 Author
   Jina Wang 02/02/2013
*******************************************************************************/
void StopMotor( void )
{
    SetDutyCycle(PWM_CHANNEL0, 50);
    SetDutyCycle(PWM_CHANNEL1, 50);
    //puts("stopmotor\r\n");
}
/*******************************************************************************/
```

```
/*******************************************************************************
 Function
    RampUpFlywheel

 Parameters
   None

 Returns
   Nothing

 Description
   When called, this function stops both motors

 Notes

 Author
   Debbie Li 02/25/2013
*******************************************************************************/
void RampUpFlywheel( char ShootingDistance )
{
    //PWM 20 shoots 5 feet
    //PWM 18 shoots 3 feet
    if (ShootingDistance == LEVIATHAN)
        {
            SetDutyCycle(PWM_CHANNEL4, LEVIATHAN_PWM);
        }
    else if (ShootingDistance == 3)
        {
            SetDutyCycle(PWM_CHANNEL4, CLOSE_SHOT);
        }
    else if (ShootingDistance == 5)
        {
            SetDutyCycle(PWM_CHANNEL4, FAR_SHOT);
        }
    else
        {
            SetDutyCycle(PWM_CHANNEL4, PP_SHOT);
        }


}
/*******************************************************************************/

/*******************************************************************************
 Function
    RampUp

 Parameters
   None

 Returns
   Nothing

 Description
   When called, this function stops both motors

 Notes

 Author
   Debbie Li 02/25/2013
*******************************************************************************/
void RampDownFlywheel( void )
{
    SetDutyCycle(PWM_CHANNEL4, 12); //ramp down flywheel
}
```

```
void FlywheelOff( void )
{
    SetDutyCycle(PWM_CHANNEL4, 0); //turn off flywheel
}



/*****************************************************************************
TEST HARNESSS
*****************************************************************************/
#ifdef TEST

void main ( void )
{
    int x = 0;

    InitPWMHardware();
    InitServoHardware();



    SetDutyCycle(PWM_CHANNEL4, LEVIATHAN_PWM);
    SetAngle(135);

    for (x = 0; x< 20000; x++)
        {
        }

    SetAngle(55);

    while (1);

}


#endif
```