```c
/*************************************************************************************************************
Project : mtrsp.c
Version : v1.0 for AVR-GCC
Date    : 8/4/2003
Author  : Chris Troutner
Company : MyRobot
Chip    : ATMEGA128
Platform: Model-T Prototype
Comments:   This program sends a 9-bit resolution PWM signal at 32Khz to PB4 and PB6 which are the
            signal lines to the motor driver. The duty cycle of the signal can be set using the 4
            buttons on the board.

            This program uses an button menu system to let the user select the duty cycle of the PWM signal
            that gets sent to each track. The following table shows what each button does:
                          ---------------------------------------------------
                          |     SW1     |   SW2     |   SW3     |   SW4     |
                          ---------------------------------------------------
                          | Left Track  | Left Track|Right Track|Right Track|
                          | + Speed     | - Speed   | + Speed   | - Speed   |
                          ---------------------------------------------------

Clock frequency     : 16.0000 MHz
Memory model        : Small
Internal SRAM size  : 4096
External SRAM size  : 0
Data Stack size     : 1024
*************************************************************************************************************/

//Compiler Derectives

    //Includes
    #include <modelt.h>

    //Defines


//Global Variables
    char t[16];
    uint8_t sw=0;

//Sub-Function Prototypes

//Interrupt Handlers

    INTERRUPT(SIG_OUTPUT_COMPARE1A) {

        PORTB &= 0xEF;      //Clear PB4

    }

    INTERRUPT(SIG_OUTPUT_COMPARE1B) {

        PORTB &= 0xBF;      //Clear PB6

    }

    INTERRUPT(SIG_OUTPUT_COMPARE1C) {
        PORTB |= 0x50;          //Set PB4 and PB6
        TCNT1 = 0;
    }

    //Switch 3
    INTERRUPT(SIG_INTERRUPT0)
    {
        //REMEMBER: SET UP FOR Falling EDGE
        asm volatile ("cli"); //Disable global interrupts while servicing this interrupt.
        ms_spin(2);         //Wait 20ms to let mechanical oscilations to die down.

        if(OCR1B > 0)
            OCR1B--;                //Increment duty cycle for Right Track (Track 2).

        sw = 3;

        EIFR |= 0x0F;           //Clear all the interrupt flags. This instruction makes it so that you
                                //can't trigger an interrupt from another button until the first button's
                                //interrupt routine has been serviced.

        asm volatile ("sei");   //Re-enable global interrupts
    }

    //Switch 4
    INTERRUPT(SIG_INTERRUPT1)
    {
        //REMEMBER: SET UP FOR Falling EDGE
        asm volatile ("cli"); //Disable global interrupts while servicing this interrupt.
        ms_spin(2);             //Wait 20ms to let mechanical oscilations to die down.

        if(OCR1B < 511)
            OCR1B++;            //Increment duty cycle for Right Track (Track 2).

                                //Display new duty cycle.
        sw = 4;

        EIFR |= 0x0F;           //Clear all the interrupt flags. This instruction makes it so that you
                                //can't trigger an interrupt from another button until the first button's
                                //interrupt routine has been serviced.

        asm volatile ("sei");   //Re-enable global interrupts
    }

    //Switch 2
    INTERRUPT(SIG_INTERRUPT2)
    {
        //REMEMBER: SET UP FOR Falling EDGE
        asm volatile ("cli"); //Disable global interrupts while servicing this interrupt.
        ms_spin(2);             //Wait 20ms to let mechanical oscilations to die down.

        if(OCR1A < 511)
```

```
            OCR1A++;            //Increment duty cycle for Right Track (Track 2).

        sw = 2;


        EIFR |= 0x0F;           //Clear all the interrupt flags. This instruction makes it so that you
                                //can't trigger an interrupt from another button until the first button's
                                //interrupt routine has been serviced.

        asm volatile ("sei");   //Re-enable global interrupts
    }

    //Switch 1
    INTERRUPT(SIG_INTERRUPT3)
    {
        //REMEMBER: SET UP FOR Falling EDGE
        asm volatile ("cli"); //Disable global interrupts while servicing this interrupt.
        ms_spin(2);             //Wait 20ms to let mechanical oscilations to die down.

        if(OCR1A > 0)
            OCR1A--;            //Increment duty cycle for Right Track (Track 2).

                                //Display new duty cycle.
        sw = 1;

        EIFR |= 0x0F;           //Clear all the interrupt flags. This instruction makes it so that you
                                //can't trigger an interrupt from another button until the first button's
                                //interrupt routine has been serviced.

        asm volatile ("sei");   //Re-enable global interrupts
    }

//Main
int main(void)
    {
    //Local Variables

    //Initialization
        reset();

        lcd_init();

        //              PERIPHERIAL AND INTERRUPT INITIALIZATION

        //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        //****************************************************************
        //               Timer0/Counter0 Initialization
        //****************************************************************
        TCCR1A=0x00;    // Nothing going on in this register.
        TCCR1B=0x01;    // set clock to 16Mhz
        TCCR1C=0x00;    // Nothing going on in this register.
        TCNT1=0;        // This counter buffer is actually composed of 2 8-bit registers TCNT1H & TCNT1L

        // These registers should have a value between 0 and 511. (1 - OCR1n/511) = Duty Cycle %.
        OCR1A=384;      // PWM Duty Cycle for Track1
        OCR1B=384;      // PWM Duty Cycle for Track2
        OCR1C=511;      // Top of Cycle Window

        // Timer(s)/Counter(s) Interrupt(s) initialization
        TIMSK=0x18;     //Compare A & B interrupts enabled. (pg. 137 of M128 datasheet)
        ETIMSK=0x01;    //Compare C interrupt enabled. (pg. 137 of M128 datasheet)
        //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

        //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        //****************************************************************
        //               External Interrupt 0-3 Initialization
        //****************************************************************
        EICRA=0xAA;     //Set INT0-3 to trigger by a falling edge. (pg. 86 of M128 datasheet)
        EICRB=0x00;     //This controlls INT4-7. (pg. 87 of M128 datasheet)
        EIMSK=0x0F;     //Enable INT0-3 interrupts (Turn them on). (pg. 88 of M128 datasheet)
        //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

        //Send initial Duty Cycle Setting
        strcpy(t, "Change w/ switch");
        line1(t);
        sprintf(t, "L:%3.1f R:%3.1f", 100.0 * ((float) OCR1A/512.0), 100.0 * ((float) OCR1B/512.0));
        line2(t);

    //Enable global interrupts (Keep this instruction at the end of initialization).
        asm volatile ("sei");

        DDRB |= 0xF0;                   //Enable Motor Driver Port
        //PORTB = (0x50|(PORTB&0x0F));  //Initialize Motor Driver
        PORTB |= 0xF0;

    //Main Execution Code
        while (1)
        {
            if(sw != 0) {
                //Display new duty cycle.
                sprintf(t,"Switch %d        ",sw);
                line1(t);

                strcpy(t,"                ");
                line2(t);
                sprintf(t, "L:%4.2f R:%4.2f", 100.0 * ((float) OCR1A/512.0), 100.0 * ((float) OCR1B/512.0));
                line2(t);

                sw = 0;
            }

        };
    }
```