```
//---------------------------------------------------------------------
//
//                           Software License Agreement
//
// The software supplied herewith by Microchip Technology Incorporated
// (the “Company”) for its PICmicro® Microcontroller is intended and
// supplied to you, the Company’s customer, for use solely and
// exclusively on Microchip PICmicro Microcontroller products. The
// software is owned by the Company and/or its supplier, and is
// protected under applicable copyright laws. All rights are reserved.
//  Any use in violation of the foregoing restrictions may subject the
// user to criminal sanctions under applicable laws, as well as to
// civil liability for the breach of the terms and conditions of this
// license.
//
// THIS SOFTWARE IS PROVIDED IN AN “AS IS” CONDITION. NO WARRANTIES,
// WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED
// TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
// PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT,
// IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR
// CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
//
//---------------------------------------------------------------------
//  File:       Init.c
//
//
//
// The following files should be included in the MPLAB project:
//
//      SensoredBLDC.c      -- Main source code file
//      Interrupts.c
//      Init.c
//      SensoredBLDC.h      -- Header file
//      p33FJ32MC204.gld    -- Linker script file
//
//
//--------------------------------------------------------------------

#include "p33FJ32MC204.h"
#include "SensoredBLDC.h"

/*******************************************************************
        Below is the code required to setup the ADC registers for :
        1. 1 channel conversion (in this case AN8)
        2. PWM trigger starts conversion
        3. Pot is connected to CH0 and AN8
        4. Manual Stop Sampling and start converting
        5. Manual check of Conversion complete

********************************************************************/
void InitADC10(void)
{
    AD1PCFGL = 0xFFFF;
    AD1PCFGLbits.PCFG8 = 0;         // ensure AN8 pot

    /* set channel scanning here, auto sampling and convert,
       with default read-format mode */
    AD1CON1 = 0x006C;
    /* select 10-bit, 1 channel ADC operation */
    AD1CON1bits.AD12B = 0;

    /* No channel scan for CH0+, Use MUX A,
       SMPI = 1 per interrupt, Vref = AVdd/AVss */
    AD1CON2 = 0x0000;

    /* Set Samples and bit conversion time */
    AD1CON3 = 0x032F;


    AD1CSSL = 0x0000;

    /* channel select AN8 */
    AD1CHS0 = 0x0008;

    IFS0bits.AD1IF = 0;
    IEC0bits.AD1IE = 1;
```

```
        AD1CON1bits.ADON = 1;
    }

    /*******************************************************************
    InitMCPWM, intializes the PWM as follows:
    1. FPWM = 39000 hz
    2. Independant PWMs
    3. Control outputs using OVDCON
    4. Set Duty Cycle with the ADC value read from pot
    5. Set ADC to be triggered by PWM special trigger
    *******************************************************************/

    void InitMCPWM(void)
    {
        PTPER = FCY/FPWM - 1;

        PWMCON1 = 0x0700;           // disable PWMs
        OVDCON = 0x0000;            // allow control using OVD
        PDC1 = 100;                 // init PWM 1, 2 and 3 to 100
        PDC2 = 100;
        PDC3 = 100;
        SEVTCMP = PTPER;
        PWMCON2 = 0x0F00;           // 16 postscale values
        PTCON = 0x8000;             // start PWM

    }
    /*******************************************************************
      Function:        void InitICandCN(void)

      Overview:        Configure Hall sensor inputs, one change notification and
                       two input captures. on IC7 the actual capture value is used
                       for further period calculation

      Note:            None.
    *******************************************************************/

    void InitIC(void)
    {
        //Hall A -> IC1. Hall A is used for Speed measurement and commutation.
        //Hall B -> IC2. Hall B is only used for commutation.
        //Hall C -> IC3. Hall C is only used for commutation.

        TRISB |= 0x000E;    // Ensure that hall connections are inputs

        IC1CON = 1;         // Init all 3 Hall Effect capture inputs:
        IC2CON = 1;         // Timer 3, every capture event, rise & fall edges
        IC7CON = 1;

        IFS0bits.IC1IF = 0; // Clear interrupt flag
        IFS0bits.IC2IF = 0;
        IFS1bits.IC7IF = 0;

        IEC0bits.IC1IE = 1; // Enable interrupt
        IEC0bits.IC2IE = 1;
        IEC1bits.IC7IE = 1;
    }

    /***********************************************************************
    Tmr3 is used to determine the rotor speed so it is set to count using Tcy/256

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

    void InitTMR3(void)
    {
        T3CON = 0x0030;         // internal Tcy/256 clock
        TMR3 = 0;
        PR3 = 0xFFFF;
    }

    /***********************************************************************
    Initialize the UART2 for BAUD = 9600, no parity, 1 stop

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

    void InitTMR1(void)
    {
        // MIPs / (Scale * Hz interrupt)
        // For this design, 8MHz * 2.5PLL, = 10 MIP
        // 10MIP/(64 * 1000) = 156, => PR1 = 156 for 1000Hz interrupt

        T1CON = 0x8020;         // internal Tcy/64 clock
        TMR1 = 0;
        PR1 = T1PR1;
        T1CONbits.TON = 1;      // turn on timer 1
        IFS0bits.T1IF = 0;
        IEC0bits.T1IE = 1;
        return;
    }
```