

```

#include <p33FJ64MC802.h>

_FOSCSEL(FNOSC_FRC);           // Internal FRC oscillator
_FOSC(FCKSM_CSECMD & OSCIOFNC_OFF & POSCMD_NONE);
                                // Clock Switching is enabled and Fail Safe Clock Monitor is
disabled
                                // OSC2 Pin Function: OSC2 is Clock Output
                                // Primary Oscillator Mode: XT Crystanl
_FWDT(FWDTEN_OFF);            // Watchdog Timer Enable: Watchdog timer enabled/disabled by
user software
_FPOR(FPWRT_PWR128) ;          // POR Timer Value: 128ms

_FICD(ICS_PGD1 & JTAGEN_OFF);  // Set FICD: Communicate on PGC1/EMUC1 and PGD1/EMUD1 & JTAG is
Disabled

void configure_i_o_ports();
void initUART();
void initQEI_HEDM5500_1();
void initQEI_HEDM5500_2();
void initQEI_DCmotor();
void init_PWM();
void QEI_change(unsigned char qei_slct);
void UART_tx(unsigned char tx);
unsigned char UART_rx();

int main (void)
{
    //Set Fosc to 40Mhz
    PLLFBD = 65;    // M = 68
    CLKDIVbits.PLLPOST = 0; // N1 = 2
    CLKDIVbits.PLLPRE = 1;  // N2 = 3
    OSCTUN = 0;
    RCONbits.SWDTEN = 0;

    // Clock switch to incorporate PLL
    __builtin_write_OSCCONH(0x01);    // Initiate Clock Switch to
                                        // FRC with PLL (NOSC=0b001)
    __builtin_write_OSCCONL(0x01);    // Start clock switching
    while (OSCCONbits.COSC != 0b001); // Wait for Clock switch to occur

    while(OSCCONbits.LOCK != 1) {};

    int i, z;
    unsigned char pos1_lsbs, pos1_msbs, pos2_lsbs, pos2_msbs;
    unsigned char rx_data, rx_pwm_lsbs, rx_pwm_msbs;
    unsigned char dir, magn1, magn2, led1, led2, qei_select, qei_select_shift;
    unsigned short int rx_pwm_duty;

    qei_select_shift = 0;
    i = 0;

    configure_i_o_ports();
    initUART();
    initQEI_HEDM5500_1();
    initQEI_DCmotor();
    init_PWM();

    while (1)
    {
        rx_pwm_lsbs = UART_rx();
        rx_pwm_msbs = UART_rx();
        rx_data = UART_rx();

        rx_pwm_duty = (rx_pwm_msbs << 8) + rx_pwm_lsbs;
        P1DC1 = rx_pwm_duty;

        led1 = (rx_data & 0b00000001);
        led2 = (rx_data & 0b00000010) >> 1;
        magn1 = (rx_data & 0b00000100) >> 2;
        magn2 = (rx_data & 0b00001000) >> 3;
        dir = (rx_data & 0b00010000) >> 4;
        qei_select = (rx_data & 0b00100000) >> 5;

        LATBbits.LATB0 = led1;
        LATBbits.LATB1= led2;
    }
}

```

```

LATBbits.LATB2 = magn1;
LATBbits.LATB3 = magn2;
LATBbits.LATB15 = dir;

```

```

if(qei_select != qei_select_shift)
{
    QEI_change(qei_select);
    qei_select_shift = qei_select;
}

```

```

/* wait about 11 usec (1/9600) before sending data */
for(z = 0; z < 1000; z++)
{
}

```

```

pos1_lsbs = POS1CNT;
pos1_msbs = POS1CNT >> 8;
pos2_lsbs = POS2CNT;
pos2_msbs = POS2CNT >> 8;

```

```

UART_tx(pos1_lsbs);
UART_tx(pos1_msbs);
UART_tx(pos2_lsbs);
UART_tx(pos2_msbs);
i = 1;

```

```

}

```

```

}

```

```

void configure_i_o_ports()

```

```

{
    ADPCFG = 0xFFFF; //make ADC pins all digital
    TRISBbits.TRISB0 = 0; // Set as a digital output: LED
    TRISBbits.TRISB1 = 0; // Set as a digital output: LED
    TRISBbits.TRISB2 = 0; // Set as a digital output: Magnet 1
    TRISBbits.TRISB3 = 0; // Set as a digital output: Magnet 2
// TRISBbits.TRISB4
// TRISBbits.TRISB5
    TRISBbits.TRISB6 = 1; // Set as a digital input: HEDM-5500/1 QEI_A
    TRISBbits.TRISB7 = 1; // Set as a digital input: HEDM-5500/1 QEI_B
    TRISBbits.TRISB8 = 1; // Set as a digital input: HEDM-5500/2 QEI_A
    TRISBbits.TRISB9 = 1; // Set as a digital input: HEDM-5500/2 QEI_B
    TRISBbits.TRISB10 = 1; // Set as a digital input: DC motor QEI_A
    TRISBbits.TRISB11 = 1; // Set as a digital input: DC motor QEI_B
    TRISBbits.TRISB12 = 1; // Set as a digital input: U1RX
    TRISBbits.TRISB13 = 0; // Set as a digital output: U1TX
    TRISBbits.TRISB14 = 0; // Set as a digital output: PWM
    TRISBbits.TRISB15 = 0; // Set as a digital output: Motor direction

    // Initial states
    LATBbits.LATB0 = 0;
    LATBbits.LATB1 = 0;
    LATBbits.LATB2 = 0;
    LATBbits.LATB3 = 0;
    LATBbits.LATB15 = 0;
}

```

```

void initUART()

```

```

{
    RPINR18bits.U1RXR=12; //Assign UART1 Receive (U1RX) to the RP12 (pin24)
    RPOR6bits.RP13R = 3; //Assign U1T1 (U1TX) To Pin RP13 (pin23)

    // configure U1MODE
    U1MODEbits.UARTEN = 0; // Bit15 TX, RX DISABLED, ENABLE at end of func
    U1MODEbits.USIDL = 0; // Bit13 Continue operation in Idle mode
    U1MODEbits.IREN = 0; // Bit12 IrDA encoder and decoder are disabled
    U1MODEbits.RTSMD = 1; // Bit11 UxRTS is in Simplex mode (but disabled later so
whatever...)
    U1MODEbits.UEN = 0b00; // Bits8,9 UxTX and UxRX pins are enabled and used; UxCTS,
UxRTS and BCLKx pins are controlled by port latches
    U1MODEbits.WAKE = 0; // Bit7 (Enable Wake-up on Start bit Detect During Sleep
Mode bit) No Wake up (since we don't sleep here)
}

```

```

    U1MODEbits.LPBACK = 0;          // Bit6 Loopback mode is disabled
    U1MODEbits.ABAUD = 0;          // Bit5 No Autobaud (would require sending '55') Receive
Polarity Inversion bit
    U1MODEbits.URXINV = 0;          // Bit4 IdleState = 1 (for dsPIC)
    U1MODEbits.BRGH = 1;           // Bit3 Fast-speed mode
    U1MODEbits.PDSEL = 0b00;       // Bits1,2 8bit, No Parity
    U1MODEbits.STSEL = 0;          // Bit0 One Stop Bit

    // Load a value into Baud Rate Generator.
    // See section 19.3.1 of datasheet.
    // U1BRG = (Fcy/(16*BaudRate))-1
    U1BRG = 86; // 40Mhz osc, 115200 baud rate

    // Load all values in for U1STA SFR (EVERY CONFIGURABLE BITS!!!)
    U1STAbits.UTXISEL1 = 1; /*Bit15 (1/2 config!) Interrupt generated when any character is
transferred
    to the Transmit Shift register (which implies at least one location is empty in the
transmit buffer)*/
    U1STAbits.UTXINV = 0; //Bit14 N/A, IRDA config UxTX Idle state is '1'
    U1STAbits.UTXISEL0 = 0; //Bit13 Other half of Bit15
    U1STAbits.UTXBRK = 0; //Bit11 Disabled
    U1STAbits.UTXEN = 0; //Bit10 TX pins controlled by periph (ENABLE at end of func)
    U1STAbits.URXISEL = 0; //Bits6,7 Interrupt flag bit is set when a character is received
    U1STAbits.ADDEN = 0; //Bit5 Address Detect Disabled

// IFS0bits.U1TXIF = 0; // Clear the Transmit Interrupt Flag
// IEC0bits.U1TXIE = 1; // Enable Transmit Interrupts
// IFS0bits.U1RXIF = 0; // Clear the Recieve Interrupt Flag
// IEC0bits.U1RXIE = 1; // Enable Recieve Interrupts

    U1MODEbits.UARTEN = 1; // And turn the peripheral on

    U1STAbits.UTXEN = 1;

    int x;
    /* wait at least 104 usec (1/9600) before sending first char */
    for(x = 0; x < 4160; x++)
    {
    }

    /*
    PC RS232 wires:
    ground:      brown/white
    TxD:         red
    RxD:         narancs/sárga

    max3232 pins
    RS232:  -TxD      pin 14 (T1OUT)
            -RxD      pin 13 (R1IN)
            -Gnd      pin 15 (GND)
    Microchip: -U1TX   pin 11 (T1IN)
              -U1RX   pin 12 (R1OUT)
    */
}

void initQEI_HEDM5500_1()
{
    RPIR14bits.QEA1R = 6;          //pin15
    RPIR14bits.QEB1R = 7;          //pin16

    QEI1CONbits.QEIM = 0;
    QEI1CONbits.CNTERR = 0;
    QEI1CONbits.INDX = 0;
    QEI1CONbits.UPDN = 1;
    QEI1CONbits.QEIM = 0b111;      //4x sebesseg, MAX1CNT-nál reset
    QEI1CONbits.PCDOUT = 0;
    QEI1CONbits.TQCKPS = 0b00;
    QEI1CONbits.POSRES = 0;
    QEI1CONbits.TQCS = 0;
    QEI1CONbits.SWPAB = 1;         //Phase A and Phase B inputs are swapped

    MAX1CNT=3999;

    DFLT1CONbits.CEID = 1;
    DFLT1CONbits.QEOUT = 1;
    DFLT1CONbits.QECK = 5;

```

```
}
```

```
void initQEI_HEDM5500_2()
```

```
{
    RPINR14bits.QEA1R = 8;        //pin17
    RPINR14bits.QEB1R = 9;        //pin18

    QEI1CONbits.QEIM = 0;
    QEI1CONbits.CNTERR = 0;
    QEI1CONbits.INDX = 0;
    QEI1CONbits.UPDN = 1;
    QEI1CONbits.QEIM = 0b111;      //4x sebesseg, MAX1CNT-nal reset
    QEI1CONbits.PCDOUT = 0;
    QEI1CONbits.TQCKPS = 0b00;
    QEI1CONbits.POSRES = 0;
    QEI1CONbits.TQCS = 0;

    MAX1CNT=3999;

    DFLT1CONbits.CEID = 1;
    DFLT1CONbits.QEOUT = 1;
    DFLT1CONbits.QECK = 5;
}
```

```
void initQEI_DCmotor()
```

```
{
    RPINR16bits.QEA2R = 10;       //pin21
    RPINR16bits.QEB2R = 11;       //pin22

    // RPINR16bits.QEA2R = 8;      //pin17
    // RPINR16bits.QEB2R = 9;      //pin18

    QEI2CONbits.QEIM = 0;
    QEI2CONbits.CNTERR = 0;
    QEI2CONbits.INDX = 0;
    QEI2CONbits.UPDN = 1;
    QEI2CONbits.QEIM = 0b111;      //4x sebesseg, MAX1CNT-nal reset
    QEI2CONbits.PCDOUT = 0;
    QEI2CONbits.TQCKPS = 0b00;
    QEI2CONbits.POSRES = 0;
    QEI2CONbits.TQCS = 0;

    MAX2CNT=3071;

    DFLT2CONbits.CEID = 1;
    DFLT2CONbits.QEOUT = 1;
    DFLT2CONbits.QECK = 5;
}
```

```
void init_PWM()
```

```
{
    ACLKCONbits.ASRCSEL = 1;      // Primary Oscillator (POSC) is the clock source
    ACLKCONbits.SELACLK = 0;      // PLL output (FOSC) provides the source clock for the Auxiliary
    Clock Divider

    P1TCONbits.PTMOD = 0b10;      // Center-aligned PWM
    P1TCONbits.PTEN = 1;          // Enabling PWM Pulse Generation

    PWM1CON1bits.PMOD1 = 0;        // PWM I/O pin pair is in the Complementary Output mode
    PWM1CON1bits.PEN1H = 1;        // PWM1H pin is disabled for PWM output
    PWM1CON1bits.PEN1L = 0;        // PWM1L pin disabled; I/O pin becomes general purpose I/O

    P1TPER = 0x1ffe;              // PWM Time Base Register Count Value bits: PWM Frequency =
    2.44kHz
    P1DC1 = 0x0000;              // Initial duty = 0%

    P1DTCN1bits.DTBPS = 0b01;     // Clock period for Dead-Time Unit A is 2 TCY
    P1DTCN2bits.DTS1I = 0;        // PWM1H1: Dead time provided from Unit A
}
```

```
void QEI_change(unsigned char qei_slct)
```

```
{
    if(qei_slct == 0)
        initQEI_HEDM5500_1();
    else
```

```

    initQEI_HEDM5500_2();
}

void UART_tx(unsigned char tx)
{
    if(U1STAbits.UTXBF == 0)
    {
        U1TXREG = tx;
    }
}

unsigned char UART_rx()
{
    unsigned char rx;
    int l = 0;
    while (l == 0)
    {
        // check for receive errors
        if(U1STAbits.FERR = 1)
        {
            continue;
        }
        // must clear the overrun error to keep uart receiving
        if(U1STAbits.OERR = 1)
        {
            U1STAbits.OERR = 0;
            continue;
        }
        // get the data
        if(U1STAbits.URXDA = 1)
        {
            rx = U1RXREG;
            return rx;
            l = 1;
        }
    }
}

```