CODE: Enabling signals for PTX Telemetry

```
//Telemetry_pulses_SPI.c program generates SPI signal at the rate of 19.53kbits per sec.
//the oscillator clock frequency is 20MHz
//we are using the primary oscillator EC mode
#include "p33fxxxx.h"

#define tool_word_count 1
#define COUNT_3 19

int tool_count;
int total_word_count;
int cased_hole;
int tool_id;
int first_word, second_word, third_word;
unsigned char rword_low[2], rword_high;
int data[500];
int jj;
long int sum;
char flag;
int timer_counter;
int SPI_word_index;
int data_received;




//Initialize the UART2 for BAUD = 9600, no parity, 1 stop
void initUART(void)
{
      U2BRG = ((10000000/9600)/16) - 1;   // 9600 baud

      // configure U2MODE
      U2MODEbits.UARTEN = 0;  // Bit15 TX, RX DISABLED, ENABLE at end of func
      U2MODEbits.USIDL = 0;   // Bit13 Continue in Idle
      U2MODEbits.IREN = 0;    // Bit12 No IR translation
      U2MODEbits.RTSMD = 0;   // Bit11 Simplex Mode
      U2MODEbits.UEN = 0;     // Bits8,9 TX,RX enabled, CTS,RTS not
      U2MODEbits.WAKE = 0;    // Bit7 No Wake up (since we don't sleep here)
      U2MODEbits.LPBACK = 0;  // Bit6 No Loop Back
      U2MODEbits.ABAUD = 0;   // Bit5 No Autobaud (would require sending '55')
      U2MODEbits.URXINV = 0;  // Bit4 IdleState = 1  (for dsPIC)
      U2MODEbits.BRGH = 0;    // Bit3 16 clocks per bit period
      U2MODEbits.PDSEL = 0;   // Bits1,2 8bit, No Parity
      U2MODEbits.STSEL = 0;   // Bit0 One Stop Bit


      // Load all values in for U1STA SFR
      U2STAbits.UTXISEL1 = 0; //Bit15 Int when Char is transferred (1/2 config!)
      U2STAbits.UTXINV = 0;   //Bit14 N/A, IRDA config
      U2STAbits.UTXISEL0 = 0; //Bit13 Other half of Bit15
      U2STAbits.UTXBRK = 0;   //Bit11 Disabled
      U2STAbits.UTXEN = 0;    //Bit10 TX pins controlled by periph
      U2STAbits.UTXBF = 0;    //Bit9 *Read Only Bit*
      U2STAbits.TRMT = 0;     //Bit8 *Read Only bit*
      U2STAbits.URXISEL = 0;  //Bits6,7 Int. on character recieved
      U2STAbits.ADDEN = 0;    //Bit5 Address Detect Disabled
      U2STAbits.RIDLE = 0;    //Bit4 *Read Only Bit*
      U2STAbits.PERR = 0;     //Bit3 *Read Only Bit*
      U2STAbits.FERR = 0;     //Bit2 *Read Only Bit*
      U2STAbits.OERR = 0;     //Bit1 *Read Only Bit*
      U2STAbits.URXDA = 0;    //Bit0 *Read Only Bit*
      //IPC7 = 0x4400;  // Mid Range Interrupt Priority level, no urgent reason

      U2MODEbits.UARTEN = 1;  // Enable UART
      //U2STAbits.UTXEN = 1;  // Enable UART Tx


      IFS1bits.U2TXIF = 0;    // Clear the Transmit Interrupt Flag
      IEC1bits.U2TXIE = 0;    // Enable Transmit Interrupts
      IFS1bits.U2RXIF = 0;    // Clear the Receive Interrupt Flag
      IEC1bits.U2RXIE = 1;    // Enable Receive Interrupts

}

void calc_checksum(){

int i = 0;

for(i = 0;i <= jj;i++)
      sum = sum + data[i];
data[++jj] = (0xFFFF & sum);


}


void SPI_TX(int data_word){

SPI1BUF = data_word;

}

void fpga_testing()
{

LATEbits.LATE0 = 0;
LATEbits.LATE1 = 1;

timer_counter = 0;
SPI_TX(data[SPI_word_index++]);

T3CONbits.TON = 1;
IEC0bits.T3IE = 1;
LATEbits.LATE2 = 0;
LATEbits.LATE1 = 0;
LATEbits.LATE2 = 1;
//LATDbits.LATD1 = 1;
LATEbits.LATE3 = 0;


while(1){

if(SPI_word_index == jj)
      break;

//stay in the loop, do nothing until a single operation cycle is complete
if(timer_counter == 79){
//on operation cycle completion wait for sometime and re-initialize all the inputs to
fpga
      LATEbits.LATE3 = 1;
      LATEbits.LATE1 = 0;
      timer_counter = 0;

      __delay32(2000);         //delay added to wait for sometime
      LATEbits.LATE3 = 0;
      LATDbits.LATD1 = 1;      //begin the next operation cycle
      LATEbits.LATE0 = 0;
      T3CONbits.TON = 1;
      IEC0bits.T3IE = 1;
}


}


}




//CLOCK SIGNAL GENERATION (FREQ: 125KHz)
void __attribute__((__interrupt__, no_auto_psv)) _T3Interrupt(void)
{

IFS0bits.T3IF = 0;
T3CONbits.TON = 0;
IEC0bits.T3IE = 0;

++timer_counter;
//begin data tx
if(timer_counter == 1){
      LATEbits.LATE1 = 1;
      LATEbits.LATE2 = 0;
}

if(timer_counter == 3){
      LATDbits.LATD0 = 1;     //signal_control
      SPI1STATbits.SPIEN = 1;
}

if(timer_counter == 5){
      LATEbits.LATE1 = 0;
      LATEbits.LATE2 = 0;
}

//clock generation signal
LATEbits.LATE0 = ~LATEbits.LATE0;

if(timer_counter == 67){
      SPI1STATbits.SPIEN = 0;
      LATDbits.LATD0 = 0;      //signal_control

}

if(timer_counter == 68){
      LATEbits.LATE1 = 1;
      //LATEbits.LATE2 = 0;
}

if(timer_counter == 72){
      LATEbits.LATE1 = 0;
      LATEbits.LATE2 = 1;
}

if(timer_counter == 79)        //operation cycle complete
      SPI_TX(data[SPI_word_index++]);
```

```
else{
      T3CONbits.TON = 1;     //continue the operation cycle execution
      IEC0bits.T3IE = 1;
}


}





//Serial Rx interrupt service routine
//Processing data recieved from LABVIEW
void __attribute__ ((interrupt, no_auto_psv)) _U2RXInterrupt(void){

IFS1bits.U2RXIF = 0;
int rec_word;

static int ii = 0;
rword_low[ii++] = U2RXREG;

if(ii == 2){                              //the third byte
      ii = 0;
      SPI_word_index = 0;

      rec_word = ((rword_low[0] << 8) | rword_low[1]);
      ++tool_id;
      ++total_word_count;
      data[0] = (cased_hole | total_word_count);
      data[--jj] = ((tool_id << 8) | tool_word_count);
      data[++jj] = rec_word;
      calc_checksum();
      ++jj;   //pointing to next element in the array
      data_received =1;
}
IFS1bits.U2RXIF = 0;

}



void add_null(){

int i;
for( i = -1; i < 500; i++, data[i] = 0);
}


void init_hwd()
{

//Initializing SPI module
SPI1STATbits.SPIEN = 0;
SPI1STATbits.SPISIDL = 0;
SPI1STATbits.SPIROV = 0;

SPI1CON1bits.DISSCK = 0;
SPI1CON1bits.DISSDO = 0;
SPI1CON1bits.MODE16 = 1;
SPI1CON1bits.CKE = 0;     //SERIAL DATA TRANSITIONS FROM IDLE STATE TO ACTIVE STATE
SPI1CON1bits.CKP = 0;     //IDLE STATE OF MASTER BIT - CURRENTLY IDLE STATE IS LOW
```

```
SPI1CON1bits.SSEN = 0;    //SLAVE SELECT PIN ENABLE BIT -  CURRENTLY PIN NOT IN USE BY
SPI MODULE
SPI1CON1bits.MSTEN = 0;   //MASTER MODE ENABLE BIT - CURRENTLY SPI MODULE BEAHVES AS A
SLAVE
SPI1CON1bits.SMP = 0;
SPI1CON1bits.SPRE = 3;
SPI1CON1bits.PPRE = 1;

//TRISDbits.TRISD2 = 0;
//TRISDbits.TRISD1 = 0;
TRISDbits.TRISD0 = 0;
TRISEbits.TRISE0 = 0;
TRISEbits.TRISE1 = 0;
TRISEbits.TRISE2 = 0;
TRISEbits.TRISE3 = 0;




}

void init_timer(){

TMR3 = 0;
PR3 = COUNT_3;
T3CONbits.TON = 0;
T3CONbits.TSIDL = 0;
T3CONbits.TGATE = 0;
T3CONbits.TCKPS = 0;
T3CONbits.TCS = 0;
}

void main(){

init_hwd();
initUART();
init_timer();
total_word_count = 0;
cased_hole = 0x4000;

rword_low[0] = 1;
rword_low[1] = 1;

add_null();
jj = 2;
while(1){

      if(data_received == 1){
            fpga_testing();
            data_received = 0;
      }
}

}
```