```
/************************************************************
*                                       XRF safety controller program                    *
*                                                       5/1/2007                                                 *
*                                               Author Peter Sabin                                       *
*                                                                                                                        *
*       This progam takes input from the XRF hardware and               *
*checks the status of the hardware. It then will send this      *
*status information to the host computer. It will also make     *
*a determination if the XRAY HVPS can be turned on. If all      *
*the hardware is in the non-error condition and the host        *
*computer send a signal to turn on the XRAY HVPS the            *
*controller will turn on the XRAY lamp and the XRAY HVPS.       *
*The controller will constanly monitor the hardware for         *
*errors conditions and shuts off the XRAY HVPS if there is a*
*hardware error condition.                                                                       *
*                                                                                                                        *
*Microcontroller used is the PIC16F886                                           *
*                                                                                                                        *

* The PIC16F886 has the following port assignments                      *
*       Port A                                                                                                   *
*               Bit0    Analog  Monitors HVPS current                            *
*               Bit1    Analog  Monitors XRAY warnig lamp                        *
*               Bit2    Analog  Monitors XRF head ground fault          *
*               Bit3    Analog  Monitors External interlock Sw          *
*               Bit4    Digital Monitors XRAY On/Off signal                      *
*               Bit5    Analog  Monitors on board potentiometer         *
*               Bit6    Digial  Monitors Ext.interlock connection       *
*               Bit7    digital Monitors XRAY lamp connection           *
*                                                                                                                        *
*       Port B                                                                                                   *
*               Bit0    Digital Status output for Microcontroller       *
*               Bit1    Digital Status output for lamp connection       *
*               Bit2    Digital Status output for lamp status           *
*               Bit3    Digital Status output for HVPS current          *
*               Bit4    Digital Status output for XRF head gnd fault*
*               Bit5    Digital Status output for interlock sw fault*
*               Bit6    Digital Status output for interlock sw conn     *
*               Bit7    Digital Status output for interlock sw          *
*                                                                                                                        *
*       Port C                                                                                                   *
*               Bit0    Digital output Control bit for HVPS                      *
*               Bit1    Digital Output Control bit for XRAY lamp        *
*               Bit2    Digital Not used                                                         *
*               Bit3    Digital Not used                                                         *
*               Bit4    Digital Not used                                                         *
*               Bit5    Digital Not used                                                         *
*               Bit6    Digital RS232 TX Output                                          *
*               Bit7    Digital RS232 RX Input                                           *
*                                                                                                                        *
*       Port E                                                                                                   *
*               Bit3    Used for programming                                             *
************************************************************/

/************************************************************
*       Include files                                                                                    *
************************************************************/

#include <16F886.h>
#include <XRF safety controller1.h>

/************************************************************
*       Fuses                                                                                                    *
*               INTRC_IO        Internal RC osc RA6/RA7 set for IO               *
*               NOLVP           No low voltage programming on RB3                *
*               WDT                     Enable watch dog timer                                   *
*               NOIESO          No int/ext oscillator switchover                 *
*               NOPROTECT       Do not protect program memory                    *
*               NOCPD           Do not protect EEprom                                    *
*               PUT                     Use power up timer for restarts                  *
************************************************************/

#fuses INTRC_IO,NOLVP,WDT,NOIESO,NOPROTECT,NOCPD,PUT

/************************************************************
*       Set clock speed to 4Mhz                                                                  *
************************************************************/

#use delay(clock=4Mhz)
```

```
/************************************************************
*       Set RS232 interface used for testing uses RC6/RC7              *
************************************************************/

#use rs232(baud=9600,uart1)

/************************************************************
* setup ports to use TRIS command                                                       *
************************************************************/

#use fast_io(a)
#use fast_io(b)
#use fast_io(c)


/************************************************************
* Subroutines                                                                                           *
************************************************************/

/************************************************************
* PIC_INITIALIZE                                                                                        *
* Routine to ports A,B and C, Status bits, WDT and timer1      *
************************************************************/
void PIC_INITIALIZE()
        {
        printf("Pic initializing\n");
        output_b(0x0fe);                // set PortB bits to 1
                                                        //except bit 0
        output_c(0x00);                 // set PortC bits to 0

        set_tris_a(0x0ff);              //set all PortA to inputs
        set_tris_b(0x00);               //set all PortB to outputs
        set_tris_c(0x0dc);              //set PortC bits 0,1 and 6 output
                                                        //all others as inputs

        setup_wdt(WDT_DIV_8192);//set WDT for about 268ms rate

        setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
                                                        //setup timer1, output is about
                                                        //524ms
        setup_adc_ports(sAN0|sAN1|sAN2|sAN3|sAN4);
                                                        //set up port A bits 0,1,2,3and 5
                                                        //as ADC ports
        setup_adc(ADC_CLOCK_DIV_8);
                                                        // use FOSC/8 adc clock
        }

/************************************************************
* CHECK_EXTERNAL_INTERLOCK_SWITCH                                                       *
* Routine to check external interlock switch for the           *
* following conditions:                                                                         *
* 1. Switch not connected(portA bit 6)                                          *
* 2. Switch is shorted or is open(portA bit 3)                          *
*                                                                                                                       *
* PortB bit 5,6 and 7 will show the status of the interlock    *
* switch as follows:                                                                            *
* bit5 = failure of interlock switch                                            *
* bit6 = connection fault                                                                       *
* bit7 = either short or open interlock switch                          *
************************************************************/

void CHECK_EXTERNAL_INTERLOCK_SWITCH()
        {
        long value;
        printf("Checking external interlock switch\n");
        if((input(INTERLOCK_SWITCH_CONNECT))==0)
                                                                //check for connection
                {
                output_high(INTERLOCK_SWITCH_FAULT_STATUS);
                output_high(INTERLOCK_SWITCH_CONNECT_STATUS);
                }
        else output_low(INTERLOCK_SWITCH_CONNECT_STATUS);

        set_adc_channel(INTERLOCK_SWITCH_SIGNAL);
                                                                //check for short or open
        delay_us(10);                           //read value on ADC3
        value=read_adc();                       //if <.5v or >3.33v then
                                                                //switch is either shorted or open
        printf("Interlock switch =%4Lu\n",value);

        if(value<=100||value>=600)
                {
                output_high(INTERLOCK_SWITCH_STATUS);
                output_high(INTERLOCK_SWITCH_FAULT_STATUS);
                }
        else
                {
                output_low(INTERLOCK_SWITCH_FAULT_STATUS);
                output_low(INTERLOCK_SWITCH_STATUS);
                }
        }

/************************************************************
* CHECK_XRF_HEAD_GND                                                                             *
* Routine that checks the XRF head ground connection.            *
* (PortA bit2,RA2)                                                                                       *
* If XRF head ground signal is less than 2.5v(512) the           *
* system is OK.                                                                                          *
* Above this value and the ground is cut.                                        *
***********************************************************/

void CHECK_XRF_HEAD_GND()
        {
        long value;
        printf("Checking XRF ground fault.\n");

        set_adc_channel(XRF_HEAD_GROUND_FAULT_SIGNAL);
                                                                //check voltage on ADC2
        delay_us(10);                           //this is the ground voltage
        value=read_adc();
        printf("XRF head voltage =%4Lu \n", value);

        if(value<512) output_low(XRF_HEAD_GROUND_FAULT_STATUS);
        else output_high(XRF_HEAD_GROUND_FAULT_STATUS);
        }

/************************************************************
* CHECK_HVPS_CURRENT                                                                             *
* Routine to test the HVPS current                                                       *
* Uses current monitor PortA bit 0 (RA0)                                         *
* If the current is above a value determined by measurement      *
* then a timer starts and to see if the current stays high       *
* for 5 seconds. If the current high for over 5 seconds          *
* then set fault XRAY_HVPS_CURRENT_STATUS (PortB bit4)           *
***********************************************************/

void CHECK_HVPS_CURRENT()
        {
        long value;
        printf("Checking HVPS current.\n");

        set_adc_channel(XRAY_HVPS_CURRENT_SIGNAL);
                                                                        //check AD0 for HVPS current
        delay_us(10);
        value=read_adc();
        printf("HVPS current =%4Lu \n", value);

        if(value>MAX_CURRENT)                   //check if counter timer1 int
                {                                                       //is started if not start.
                if(CURRENT_TIMER>0)                     //0 not started >0 started
                        {
                        if(CURRENT_TIMER>MAX_TIMER_VALUE)
                                {                               // HVPS high current exceeded time
                                output_high(XRAY_HVPS_CURRENT_STATUS);
                                                                //max count done stop timer1
                                disable_interrupts(GLOBAL);
                                disable_interrupts(INT_TIMER1);
                                CURRENT_TIMER=0;
                                }
                        }
                else
                        {
                        enable_interrupts(INT_TIMER1);  //not started so start
                        enable_interrupts(GLOBAL);
                        }
                }
        else
```

```
                {
                disable_interrupts(GLOBAL);
                disable_interrupts(INT_TIMER1);
                CURRENT_TIMER=0;
                output_low(XRAY_HVPS_CURRENT_STATUS);
                }
        }

/************************************************************
* XRAY_ACTIVE_LAMP_FAULT                                                                         *
* Routine to check if the lamp is drawing current and to see*
* if the lamp is connected                                                                       *
* Uses PortA bit1(RA1)  current monitor                                          *
* Uses PortA bit7       lamp connection                                          *
* If lamp not connected then set fault flag                              *
* XRAY_ACTIVE_LAMP_CONNECT_STATUS(PortB bit 1)                           *
* If the current is the nominal value then clear fault           *
* flag XRAY_ACTIVE_LAMP_CURRENT_STATUS(PortB bit2)                       *
************************************************************/

void XRAY_ACTIVE_LAMP_FAULT()
        {
        long value;
        printf("Checking Active Lamp.\n");
        if((input(XRAY_ACTIVE_LAMP_CONNECT))==1)
                {
                output_low(XRAY_ACTIVE_LAMP_CONNECT_STATUS);
                set_adc_channel(XRAY_ACTIVE_LAMP_CURRENT_SIGNAL);
                delay_us(10);
                value=read_adc();
                printf("Lamp current =%4Lu\n", value);

                if(value<LAMP_CURRENT_MIN||value>LAMP_CURRENT_MAX)
                        {
                        output_high(XRAY_ACTIVE_LAMP_STATUS);
                        }
                else output_low(XRAY_ACTIVE_LAMP_STATUS);
                }
        else
                {
                output_high(XRAY_ACTIVE_LAMP_CONNECT_STATUS);
                output_high(XRAY_ACTIVE_LAMP_STATUS);
                }

        }

/************************************************************
* CHECK_ON/OFF_XRAY_CONTOL_CMD                                                           *
* Routine used during error conditiond to make sure the          *
* user turns off the XRF and then we can continue                        *
* Uses PortA bit4                                                                                        *
************************************************************/

void CHECK_ON_OFF_XRAY_CONTROL_CMD()
        {
        while((input(XRAY_CONTROL_CMD))==1)
                {
                printf("waiting\n");
                }
        }
/************************************************************
* TIMER1_ISR                                                                                             *
* Routine for the timer1 ISR                                                             *
* Add one to counter when timer1 interrupt occurs                        *
************************************************************/
#INT_TIMER1
void TIMER1_ISR()
        {
        CURRENT_TIMER++;
        }

/************************************************************
* Main subroutine                                                                                        *
* 1. Check to see if boot was caused by the watchdog.            *
* If the watchdog caused the boot then set PortB bit0            *
* and stop the program.                                                                          *
* 2. Initialize the PIC                                                                          *
* 3. Check the fault hardware                                                            *
* 4. Check to see if the user wants to turn on the XRF.          *
* 5. If true then turn on the XRF.                                                      *
* 6. Continue checking until a fault occurrs or the user        *
* off the XRF.                                                                                          *
************************************************************/
void main()
        {
        int flag_status;
        if(restart_cause()==WDT_TIMEOUT)
                {
                while(1)
                        {
                        set_tris_b(0x0);
                        output_high(XRAY_CONTROLLER_FAIL_STATUS);
                        }
                }

        restart_wdt();
        PIC_INITIALIZE();
        restart_wdt();
        while(1)
                {
                restart_wdt();
                CHECK_EXTERNAL_INTERLOCK_SWITCH();
                CHECK_XRF_HEAD_GND();
                CHECK_HVPS_CURRENT();
                if((input(XRAY_CONTROL_CMD))==1)
                        {
                        output_high(XRAY_ACTIVE_LAMP_CONTROL);
                        delay_ms(10);
                        XRAY_ACTIVE_LAMP_FAULT();
                        flag_status=input_b();
                        if(flag_status==0)
                                {
                                output_high(XRAY_HVPS_CONTROL);
                                }
                        else CHECK_ON_OFF_XRAY_CONTROL_CMD();
                        }
                else
                        {
                        output_low(XRAY_HVPS_CONTROL);
                        output_low(XRAY_ACTIVE_LAMP_CONTROL);
                        }
                }
        }
```