```
/********************************************************************************
      * FILE NAME: user_routines.c <EDU VERSION>
*
      * DESCRIPTION:
      * This file contains the default mappings of inputs
      * (like switches, joysticks, and buttons) to outputs on the EDU RC.
*
      * USAGE:
      * You can either modify this file to fit your needs, or remove it from your
      * project and replace it with a modified copy.
*
********************************************************************************
**/


#include "ifi_aliases.h"
#include "ifi_default.h"
#include "ifi_utilities.h"
#include "user_routines.h"
#include "printf_lib.h"



/*** DEFINE USER VARIABLES AND INITIALIZE THEM HERE ***/
/* EXAMPLES: (see MPLAB C18 User's Guide, p.9 for all types)
unsigned char wheel_revolutions = 0; (can vary from 0 to 255)
unsigned int delay_count = 7; (can vary from 0 to 65,535)
int angle_deviation = 142; (can vary from -32,768 to 32,767)
unsigned long very_big_counter = 0; (can vary from 0 to 4,294,967,295)
*/



/*********************************************************************************
***
      * FUNCTION NAME: Setup_Who_Controls_Pwms
      * PURPOSE: Each parameter specifies what processor will control the pwm. *
      * CALLED FROM: User_Initialization
      * Argument Type IO Description * ------------------------
      * pwmSpec1 int I USER/MASTER (defined in ifi_aliases.h)
      * pwmSpec2 int I USER/MASTER
      * pwmSpec3 int I USER/MASTER
      * pwmSpec4 int I USER/MASTER
      * pwmSpec5 int I USER/MASTER
      * pwmSpec6 int I USER/MASTER
      * pwmSpec7 int I USER/MASTER
      * pwmSpec8 int I USER/MASTER
      * RETURNS: void
********************************************************************************* **/

static void Setup_Who_Controls_Pwms(int pwmSpec1,int pwmSpec2,int pwmSpec3, int
pwmSpec4,
                                    int pwmSpec5,int pwmSpec6,int pwmSpec7, int
pwmSpec8) {
  txdata.pwm_mask = 0xFF; /* Default to master controlling all PWMs. */
```

```
  if (pwmSpec1 == USER) /* If User controls PWM1 then clear bit0. */
    txdata.pwm_mask &= 0xFE; /* same as txdata.pwm_mask =
txdata.pwm_mask & 0xFE; */
  if (pwmSpec2 == USER) /* If User controls PWM2 then clear bit1. */
   txdata.pwm_mask &= 0xFD; if (pwmSpec3 == USER) /* If
 User controls PWM3 then clear bit2. */
   txdata.pwm_mask &= 0xFB; if (pwmSpec4 == USER) /* If
 User controls PWM4 then clear bit3. */
   txdata.pwm_mask &= 0xF7; if (pwmSpec5 == USER) /* If
 User controls PWM5 then clear bit4. */
   txdata.pwm_mask &= 0xEF; if (pwmSpec6 == USER) /* If
 User controls PWM6 then clear bit5. */
   txdata.pwm_mask &= 0xDF; if (pwmSpec7 == USER) /* If
 User controls PWM7 then clear bit6. */
   txdata.pwm_mask &= 0xBF; if (pwmSpec8 == USER) /* If
 User controls PWM8 then clear bit7. */
    txdata.pwm_mask &= 0x7F; }


/************************************************************************ ***
      * FUNCTION NAME: User_Initialization
      * PURPOSE: This routine is called first (and only once) in the Main
function.
      * You may modify and add to this function.
      * The primary purpose is to set up the DIGITAL IN/OUT -ANALOG IN
      * pins as analog inputs, digital inputs, and digital outputs.
      * CALLED FROM: main.c
      * ARGUMENTS: none
      * RETURNS: void
************************************************************************ **/

void User_Initialization (void) {
  rom const char *strptr = "IFI User Processor Initialized ...";

/* FIRST: Set up the pins you want to use as analog INPUTs. */
  //none

/* SECOND: Configure the number of analog channels. */
  Set_Number_of_Analog_Channels(NO_ANALOG); /* See ifi_aliases.h */

/* THIRD: Set up any extra digital inputs. */ IO1=INPUT; /*these input
initializations correspond to the inputs in*/
  IO2=INPUT; /*the Eden_2006 function*/
  IO3=INPUT;
  IO4=INPUT;/*NOTE THAT IS NOT A ZERO BUT AN "OH"*/

  IO16=INPUT;

/* FOURTH: Set up the pins you want to use as digital OUTPUTs. */
//none

/* FIFTH: Initialize the values on the digital outputs. */
//none

/* SIXTH: Set your initial PWM values. Neutral is 127. */
  pwm01 = pwm02 = pwm03 = pwm04 = pwm05 = pwm06 = pwm07 = pwm08 = 127;

/* SEVENTH: Choose which processor will control which PWM outputs. */
  Setup_Who_Controls_Pwms(MASTER,MASTER,MASTER,MASTER,MASTER,MASTER,MASTER,
MASTER);
```

```
/* EIGHTH: Set your PWM output type. Only applies if USER controls PWM 1, 2,
3, or 4. */
  /* Choose from these parameters for PWM 1-4
respectively: */
  /* IFI_PWM - Standard IFI PWM output generated with
Generate_Pwms(...) */
  /* USER_CCP - User can use PWM pin as digital I/O or CCP pin. */
  Setup_PWM_Output_Type(IFI_PWM,IFI_PWM,IFI_PWM,IFI_PWM);


/* Add any other user initialization code here. */

  Initialize_Serial_Comms();

  Putdata(&txdata); /* DO NOT CHANGE! */

  printf("%s\n", strptr); /* Optional - Print initialization message. */

  User_Proc_Is_Ready(); /* DO NOT CHANGE! - last line of User_Initialization */ }


/******************************************************************************* ***
      * FUNCTION NAME: Process_Data_From_Master_uP
      * PURPOSE: Executes every 17ms when it gets new data from the master
      * microprocessor.
      * CALLED FROM: main.c
      * ARGUMENTS: none
      * RETURNS: void
******************************************************************************* **/

void Process_Data_From_Master_uP(void)
{ Getdata(&rxdata); /* Get fresh data from the master microprocessor. */

/*****************************Eden_Code_2006******************************** **/
//SOFTWARE USED: MPLab v6.62, MPLabv7.3, MPLab C18 v2.4 //AUTHOR: DAVID
MIKOLAJEWSKI //DATE: INITIAL: MARCH 20 2006 // LATEST: MARCH 22 2006

/*****************************BEGINNING OF
CODE*******************************/

    /*NOTE: the inputs need to be defined in the initialization*/

/**** CODE DEFINITIONS ****/ #define rc_in5 PWM_in5 /*this corresponds to the
  Base lift/raise*/ #define rc_in6 PWM_in6 /*this corresponds to the Base
  chain/claw
drive*/ #define rc_in3 PWM_in3 /**/

  #define m_a pwm01 /*this corresponds to motor A on pwm01*/
  #define m_b pwm02 /*this corresponds to motor B on pwm02*/
  #define m_c pwm03 /*this corresponds to motor C on pwm03*/


//sensors marked with an "i" means they are defined in the initialization #define
  in_1 rc_dig_in01 //"Landing Gear" DOWN sensor -i #define in_2 rc_dig_in02
  //"Landing Gear" UP sensor -i #define in_3 rc_dig_in03 //i #define in_4
  rc_dig_in04 //i

  #define in_16 rc_dig_in16 //Lower Push Button -i /*+++++
[code definitions] +++++*/
```

```
/**** VARIABLES ****/ /*+++++
[variables] +++++*/


/**** TURN ON THE LCD DISPLAY ****/
solenoid1=1;
/*+++++ [turn on the lcd display] +++++*/


/**** [ELECTROMAGNET] ****/
if(in_2 == 1) { solenoid4 = 1; } else { solenoid4=0; }
/*+++++ [electromagnet] +++++*/


/**** BASE LIFT CODE ****/ if ((in_2 == 1)&&(rc_in5 > 127))/*if ls2 is not hit */
/*(meaning if the base is lifted)*/
{

m_c = rc_in5 ;
}
else if ((in_1 == 1)&&(rc_in5 < 127))
{
m_c = rc_in5;
}
else
{
m_c = 127;
}

/*+++++ [base lift code] +++++*/



/**** SLIDER LIFT CODE ****/

if ((rc_in5 > 127)&&(in_3 == 1)) /*if limit open, and base lowered*/
{
m_a = rc_in3; /* put the motor output to red; thus up. used to be 73*/
}
else /*otherwise stop the motor*/
{
m_a = 127;
}

/*+++++ [slider lift code] +++++*/



/**** SLIDER LOWERING MOMENTARY ****/

 if (in_16 == 0) {
 m_a = 200; }
/*+++++ [slider lowering momentary] +++++*/



/**** BERT- CHAIN/CLAW CONTROL ****/
```

```
m_b=rc_in6;

/*+++++ [bert- chain/claw control] +++++*/



/**** PRINT STATEMENTS ****/
//print RC values
/*printf("rc_in5 = %d\n", (int)rc_in5);
printf("rc_in6 = %d\n", (int)rc_in6);


printf("\n");*/

-

//print inputs
/*printf("in_1 = %d\n", (int)in_1);
printf("in_2 = %d\n", (int)in_2);
printf("in_3 = %d\n", (int)in_3);
printf("in_4 = %d\n", (int)in_4);

printf("in_16 = %d\n", (int)in_16);

printf("\n");*/

//print outputs
/*printf("m_a = %d\n", (int)m_a);
printf("m_b = %d\n", (int)m_b);
printf("m_c = %d\n", (int)m_c);


printf("\n");*/
printf("electro=%d\n",solenoid4);
//clear screen so that you can see the values
//printf("\n\n\n");
/*+++++ [print statements] +++++*/


/*******************************END OF
CODE*********************************/
//NOTE: see the top of this code to view authors and information



/*****************************Eden_Code_2006*******************************
**/



  Putdata(&txdata); /* DO NOT CHANGE! */ }
```