Functions defined in c6xdskdigio.c

Function: **void Init_C6713DSK(void)**
Parameters: None
Return value: None
Description: Initializes the DSP's external memory interface for the C6713 daughter card.

_Needs to be the first function you call in your main() function._
Example:

```c
void main(void) {
    Init_C6713DSK();
}
```

Function: **void Init_encoders(int chip)**
Parameters: chip 1 or 2 indicating which LS7266(encoder) chip to initialize on the daughter card
Return value: None
Description: Call this function before using the read_encoder functions. Init_encoders enables both optical encoder channels on the specified chip. Sets up the chip in X4 quadrature decode mode. Zeros the encoder count register.
Example:

```c
void main(void) {
    Init_encoders(1);  // Chip 1
    Init_encoders(2);  // Chip 2
}
```

Function: **void read_encoders(int chip,float *enc1,float *enc2,struct encoder_parameters encpars)**
Parameters:
- chip 1 or 2 indicating which LS7266 chip to read on the daughter card.
- *enc1 A pointer to a variable that will receive the chip's Enc. 1 value in radians.
- *enc2 A pointer to a variable that will receive the chip's Enc. 2 value in radians.
- encpars A structure of type encoder_parameters which contains the needed parameters to convert the channel's encoder count to radians. See c6xdskdigio.h for a description of encoder_parameters. Possible choices for you to use in GE 420 and GE 423 are: g_standard_furuta, g_standard_DCMotor, g_standard_GearMotor.
Return value: None
Description: This function reads both encoder channels of the specified chip and returns the values in radians with the pointers enc1 and enc2.
Example:

```c
// Define global variables (or they could be local)
float enc1 = 0;
float enc2 = 0;
float enc3 = 0;
float enc4 = 0;

//Then inside your control code call
// chip 1 Also notice that enc1 and enc2 are passed as reference
read_encoders(1,&enc1,&enc2,g_standard_DCMotor);
// chip 2
read_encoders(2,&enc3,&enc4,g_standard_DCMotor);
```

Function: **void Init_pwm(int chip)**
Parameters: chip 1 or 2 indicating which CTS82C54(PWM Timer) chip to setup on the daughter card
Return value: None
Description: Call this function before calling any of the other PWM functions. You will need to do this for each chip you are using.
Example:
void main(void) {
    Init_pwm(1); // Chip 1
    Init_pwm(2); // Chip 2
}

Function: void out_PWM(int chip, int channel, float u)
Parameters:
    chip  1 or 2 indicating which CTS82C54 (PWM Timer) chip to command on the
daughter card.
    channel 1 or 2 indicating which PWM channel to command.
    u   The control effort as a value between -10 and +10.
Return value: None
Description: This function commands the duty cycle of a 20KHz carrier frequency
PWM output. You will be using this PWM signal to command a 3952
PWM AMP chip to drive your DC motor.
Example:
    // define global variables (or they could be local)
    float u1=0;
    float u2=0;
    float u3=0;
    float u4=0;

    // in your control code assign u1 through u4 to the desired output value
    // this assignment will be related to a control equation or an open loop output

    // then in your code output the values
    out_PWM(1,1,u1); //PWM channel 1
    out_PWM(1,2,u2); //PWM channel 2
    out_PWM(2,1,u3); //PWM channel 3
    out_PWM(2,2,u4); //PWM channel 4
Functions defined in dac2815.c

Function: **void Init_DAC2815(void)**
Parameters: None
Return value: None
Description: Use this function to Enable the DAC channels. Must be called before using writeDAC2815.
Example:
```c
void main(void) {
    Init_DAC2815();
}
```

Function: **void writeDAC2815(float dac1,float dac2)**
Parameters:
  - `dac1` The desired voltage value to be output on DAC channel 1.
  - `dac2` The desired voltage value to be output on DAC channel 2.
Return value: None
Description: Use this function to command the DAC channels with a -10Volt to +10Volt output signal.
Example:
```c
// define global variables (or they could be local)
float dac1=0;
float dac2=0;

// in your control code assign dac1 and dac2 to the desired output voltage

// then in your code output the values
writeDAC2815(dac1,dac2);
```
Functions defined in max3100uart.c

Function: `void Init_UART1_19200(void)`
Parameters: None
Return value: None
Description: Initialize UART1 to a baud rate of 19200 bits/sec.
Example:
```c
void main(void) {
    Init_UART1_19200();
}
```

Function: `void Init_UART_LCD(void)`
Parameters: None
Return value: None
Description: Initialize LCD Screen. See this source and the LCD manual to determine starting conditions for the LCD screen.
Example:
```c
void main(void) {
    Init_UART1_19200();
    Init_UART_LCD();
}
```

Function: `void UARTPrintfLine1(char *format,...)`
Parameters:
- `format` A formatting string for ANSI C, i.e. "foo=%f"
- ... The values to be substituted into your string.
Description: Starts a task to send a string of characters to the LCD or other device. The formatted string of characters will be sent to the top line of the LCD screen.
Example:
```c
// this example prints out encoder channels 1 and 2 to the LCD screen inside a period function "myprd"
float enc1=0;
float enc2=0;
void main(void) {
    Init_UART1_19200();
    Init_UART1_LCD();
}
void myprd(void) {
    read_encoders(1,&enc1,&enc2,g_standardDCMotor);
    // Print to 20 character of Line 1
    UARTPrintfLine1("e1=%.2f,e2=%.2f",enc1,enc2);
}
```

Function: `void UARTPrintfLine2(char *format,...)`
Parameters:
- `format` A formatting string for ANSI C, i.e. "foo=%f"
- ... The values to be substituted into your string.
See any C book that discusses the "printf" instruction to understand these parameters better.

Return value: None

Description: Starts a task to send a string of characters to the LCD or other device. The formatted string of characters will be sent to the bottom line of the LCD screen.

Example:

```c
// this example prints out encoder channels 1 and 2 to the LCD screen inside a period function "myprd"
float enc1=0;
float enc2=0;

void main(void) {
    Init_UART1_19200();
    Init_UART1_LCD();
}

void myprd(void) {
    read_encoders(1,&enc1,&enc2,g_standardDCMotor);
    // Print to 20 character of Line 1
    UARTPrintfLine1("e1=%.2f,e2=%.2f",enc1,enc2);
}
```

Function: `int SmallSprintf(char *dest, char *format,...)`

Description: Less functionality version of the ANSI C standard sprintf. Has all the functionality we will need on the DSP and uses much less memory. Always use this function instead of sprintf.

Example:

```c
// this example relates to the robots in Mechatronics
// this example sends a formatted string over the wireless card inside a period function "myprd"
float enc1=0;
float enc2=0;
char sendbuffer[40];

void myprd(void) {
    read_encoders(1,&enc1,&enc2,g_standardDCMotor);
    // First create the formatted string
    SmallSprintf(sendbuffer, "%e1=%.2f,e2=%.2f",enc1,enc2);
    // then send the data over the wireless channel
    WirelessSend(sendbuffer,sizeof(sendbuffer));
}
```

**Functions defined in switch_led.c**

Function: `unsigned int get_switchstate(void)`

Parameters: None

Return value: An integer value between 0 and 15 representing the state of the switches.

Description: This function reads the state of the four DIP switches on the board.

Function: `void set_LEDstate(unsigned int leds)`

Parameters:

- `leds` An integer between 0 and 15 representing the state of the four LEDs.
Return value: None
Description: This function sets the state of the four LEDs on the DSK board next to the four DIP switches.
**Functions defined in ADC7862.c**

Function: **void ADC7862_Start()**
Parameters: None :void
Return value: None.
Description: This function initiates a conversion on the ADC7862 chip.

Function: **void ADC7862_Read(float *adc1, float *adc2)**
Parameters: *adc1 and *adc2, Return the voltage value sampled at the two ADC inputs of the ADC7862 chip. The range of the sampled voltage is -10V to 10V. Note that these variables are pointers so you will need to pass the address (use the ‘&’) of two float variables to this function.

Return value: None

Example:
```
// Define global variables (or they could be local)
float adc1 = 0;
float adc2 = 0;

// Then inside a PRD process call
void myPRD(void) {
    ADC7862_Start();
}

Then inside HWI_INT7’s function call

void myINT7 (void) {
    ADC7862_Read(&adc1,&adc2);
    Then the rest of your control code
}