C:\Users\abz\Desktop\My FYP\H-Bridge Implementor\Low Level Implementor\Codings\direct_hbridge_control.c

```
/*SUB-implementors thatare to be used in the H-Bridge control implementation.Its an effort
of AS-CORP with
no coprights reserved*/
#include<P18f452.h>
//Port pins assignments for motion control
#define brake1 PORTDbits.RD0
#define brake2 PORTDbits.RD2
#define dir1 PORTDbits.RD1
#define dir2 PORTDbits.RD3
#define TX PORTCbits.RC6
#define RX PORTCbits.RC7
//Functions declarations used in the Command ImplementorProgram--------
//---------------------------------------------------------------------
//--------------------
//Initializing functions
void init_pwm(void);
void delay(unsigned int);
void init_serial(void);
//.....................................
//Decision taker/Router(may be called as!)
void decision_taker(void);
//H control implementor Definition.....................
void motion(unsigned char,unsigned char,unsigned char);
//Motion sub-implementors
void start_m(unsigned char,unsigned char);
void stop_m(void);
void accelerate_m();
void forward_m(unsigned char,unsigned char);
void reverse_m(unsigned char,unsigned char);
void left_turn_m(unsigned char);
void right_turn_m(unsigned char);
void tilt_left_m(unsigned char,unsigned char);
void tilt_right_m(unsigned char,unsigned char);
//Motion Clones(Lowest level task performers)
void set_pwms(unsigned char,unsigned char);
//.....................................
//Constants used in the Functions
//The following constants are used in the motion()to Call relevant subimplementor
const unsigned char start=0x01;
const unsigned char stop=0x02;
const unsigned char accelerate=0x03;
const unsigned char forward=0x04;
const unsigned char reverse=0x05;
const unsigned char left_turn=0x06;
const unsigned char right_turn=0x07;
const unsigned char tilt_left=0x08;
const unsigned char tilt_right=0x09;
//"diff" is the difference of responses of Left and Right motors
const unsigned char diff=0x00;
//---------------------------------
//Universal variables
unsigned int i,j=0; //i&j used in delayfunction
unsigned char pwm_l;
unsigned char pwm_r;
unsigned char pwml_now; //Current values of pwml
unsigned char pwmr_now; //Current values of pwmr
unsigned char tiltl_now; //current tilt_left (pwmr-pwml)
unsigned char tiltr_now; //currenttilt_right(pwml-pwmr)
unsigned char implementor_name; //Name of implementor
unsigned char implementor_task; //Task given to theimplementor
unsigned char com_command; //command byte recived at serial port
unsigned char com_arg1; //first argument recieved at the serial port
unsigned char com_arg2; //second argument recieved at the serial port
//---------------------------------------------------------------------
//---------
//////////////////// -- ISR for Serial Communication Protocol--//////////////////////////
void check_isr(void);
void tx_isr(void);
void rc_isr(void);
#pragma code my_high_prio_int = 0x08
void my_high_prio_int(void)
{
_asm
GOTO check_isr
_endasm
```

```
}
#pragma code
#pragma interrupt check_isr
void check_isr(void)
{
    if(PIR1bits.RCIF==1)
    {
        com_command=RCREG;
        PIR1bits.RCIF=0;
        INTCONbits.GIE=0;
        while(PIR1bits.RCIF==0);
        com_arg1=RCREG;
        PIR1bits.RCIF=0;
        while(PIR1bits.RCIF==0);
        com_arg2=RCREG;
        decision_taker();
        PIR1bits.RCIF=0;
    }
    if(PIR1bits.TXIF==1)
    {
    }
    INTCONbits.GIE=1;
}
//------------------Interuupt Service Routine Ended---------------------------//

void main(void)
{
    init_pwm();
    start_m(1,1);
        forward_m(200,200);
        delay(400);
        stop_m();
        delay(10);
        reverse_m(200,200);
        delay(400);
        stop_m();
        init_serial();
//Checks end here.....................(All subimplementors srechecked and r valid,
//Waiting for interrupt!,,,,,,,,,,,
        while(1);
}

//-----------------------Functions Definitions-----------------------//
void delay(unsigned int wait)
{
    for(i=0;i<=wait;i++)
    {
        for(j=0;j<=400;j++);
    }
}
void init_serial(void)
{
    SPBRG=64;
    TRISCbits.RC7=1;    //rx pin set as input
    TRISCbits.RC6=0;    //tx pin set as output
    TXSTAbits.BRGH=1;   //High Baud rate is selected
    RCSTAbits.SPEN=1;
    RCSTAbits.CREN=1;
    INTCONbits.GIE=1;
    INTCONbits.PEIE=1;
    PIE1bits.RCIE=1;
    PIR1bits.RCIF=0;
    PIR1bits.TXIF=0;

}
void init_pwm(void)
{
    WDTCONbits.SWDTEN=0; //This softwarely disables the WDT feature
    CCP1CON=0x00;
    CCP2CON=0x00;
    PR2=250; //Setting PWM priod
    T2CON=0X01; //timer2 prescalar=1/4;
    TRISCbits.TRISC2=0; //CCP1 pin set as output
    TRISCbits.TRISC1=0; //CCP2 pin set as output
    TRISD=0X00; //PortD set as output
```

```
}
void set_pwms(unsigned char pwml,unsigned char pwmr)
{
    CCPR1L=pwml; //CCP1 used for the left motor
    CCPR2L=pwmr; //CCP2 used for the right motor
    pwml_now=pwml; //Updating the current l-PWM status
    pwmr_now=pwmr; //Updating the current r-PWM status
}
void start_m(unsigned char l_start,unsigned char r_start)
{
    if(l_start!=0)
    {
        dir1=1; //Dir=1 forward direction
        brake1=0;
    }
    if(r_start!=0)
    {
        dir2=1; //Dir=1 forward direction
        brake2=0;
    }
    CCP1CON=0x0C; //Turning the PWMs on.
    CCP2CON=0X0C;
    T2CONbits.TMR2ON=1; //Turning the timer2 on
}
void stop_m(void)
{
    brake1=1; //Applying brakes to the motors
    brake2=1;
}
void forward_m(unsigned char pwmlfwd,unsigned char pwmrfwd)
{
    dir1=1; //Move forward
    dir2=1; //Move forward
    brake1=0; //Ensuring brake1 is removed
    brake2=0; //Ensuring brake2 is removed
    set_pwms(pwmlfwd,pwmrfwd); //Arguments must cause straight Motion
}
void reverse_m(unsigned char pwmlrev,unsigned char pwmrrev)
{
    brake1=0; //Ensuring brake1 is removed
    brake2=0; //Ensuring brake2 is removed
    dir1=0; //Move backward
    dir2=0; //Move backward
    set_pwms(pwmlrev,pwmrrev); //Arguments must cause straight Motion
}
void left_turn_m(unsigned char turn_pwm_right_motor)
{
    brake1=1;
    brake2=0;
    dir1=1;
    dir2=1;
    set_pwms(0,turn_pwm_right_motor);
}
void right_turn_m(unsigned char turn_pwm_left_motor)
{
    brake1=0;
    brake2=1;
    dir1=1;
    dir2=1;
    set_pwms(turn_pwm_left_motor,0);
}
void tilt_left_m(unsigned char pwm_basic_left,unsigned char tilt_diff_left)
{
    brake1=0;
    brake2=0;
    dir1=1;
    dir2=1;
    tiltl_now=tilt_diff_left; //Sets current value of left tilt
    set_pwms(pwm_basic_left,(pwm_basic_left+tilt_diff_left));
}
void tilt_right_m(unsigned char pwm_basic_right,unsigned char tilt_diff_right)
{
    brake1=0;
    brake2=0;
    dir1=1;
```

```
    dir2=1;
    tiltr_now=tilt_diff_right; //Sets current value of right tilt
    set_pwms((pwm_basic_right+tilt_diff_right),pwm_basic_right);
}
//-------------------------------------------------------------------
/*Below is the the implementation of H-Bridge Decision Implementor
function.It will call the Subimplementors corresponding to the motion command
that H-Bridge is required to implement/
*/
void motion(unsigned char m_command,unsigned char arg1,unsigned char arg2)
{
    if(m_command==start) //Command=0x01
    {
        start_m(arg1,arg2); //arg1&arg2 are left and right motors statuses
    }
    if(m_command==stop) //Command=0x02
    {
        stop_m();
    }
    if(m_command==accelerate) //Command=0x03
    {
        return;
    }
    if(m_command==forward) //Command=0x04
    {
        forward_m(arg1,arg2); //arg1&arg2 are left and right motors pwms
    }
    if(m_command==reverse) //Command=0x05
    {
        reverse_m(arg1,arg2); //arg1&arg2 are left and right
    }
    if(m_command==left_turn) //Command=0x06
    {
        left_turn_m(arg1); //arg1 is the pwm of right motor
        /*In case of the single argument funtions
        first of the two arguments sent as pwm value*/
    }
    if(m_command==right_turn) //Command=0x07
    {
        right_turn_m(arg1); //arg1 is the pwm of the right  motor
    }
    if(m_command==tilt_left) //Command=0x08
    {
        tilt_left_m(arg1,arg2); //arg1 is the pwm basic and arg2 is the offset
    }
    if(m_command==tilt_right) //Command=0x09
    {
        tilt_right_m(arg1,arg2); //arg1 is the pwm basic and arg2 is the offset
    }
}
/*This is the code of decision taker taker/Router program that
appoints the
relevant Implementor for the work*/
void decision_taker(void)
{
    //com_command,com_arg1,com_arg2,implementor_task,implementor_name
    implementor_task=com_command&0x0F;
    implementor_name=com_command&0xF0;
    if(implementor_name==0x10)
    {
        motion(implementor_task,com_arg1,com_arg2);
    }
}
```