

PUPPY を EyER で制御をするための各プログラムソースの変更追加点

使用ボード : BB64E3687F

1. sci.c 追加変更部分

①Get_EyER_data 関数の追加

```
/*
*****
EyER コマンド判定
*****
*/
unsigned char Get_EyER_data(void)
{
    unsigned char EyER_Re;          /* PUPPY への命令データを格納する変数 */
    unsigned char code_EyER;       /* EyER からの受信データを格納する変数 */

    code_EyER = 0;                 /* 初期化 */

    /* 受信したときにのみデータを code_EyER へ格納 */
    if( ( SCI3.SSR.BYTE & 0x40 ) == 0x40)    code_EyER = SCI3.RDR;

    /* 通信エラーが発生した場合に各エラーフラグをクリアする */
    if( !( SCI3.SSR.BYTE & 0x38 ) == 0)      SCI3.SSR.BYTE = 0x00;

    /* 受信したデータ(EyER から送られてきたデータ)の判定 */
    switch(code_EyER)               /* 一文字のみの判定 */
    {
        case '1':                   /* 受信データが「1」であるときの処理 */
            sci_puts("IRRECV:001 ¥r¥n"); /* 確認のためのシリアル出力 */
            EyER_Re = 0;              /* 「0」という命令データを EyER_Re に格納 */
            break;

        case '2':                   /* 受信データが「2」であるときの処理 */
            sci_puts("IRRECV:002 ¥r¥n"); /* 確認のためのシリアル出力 */
            EyER_Re = 1;              /* 「1」という命令データを EyER_Re に格納 */
            break;

        default:                    /* 上記以外の処理 */
            EyER_Re = 9;              /* 「9」という命令データを EyER_Re に格納 */
            break;
    }

    return EyER_Re;
}
```

②シリアル通信レートを 9600bps に変更

```
/******  
SCI 初期化処理  
*****/  
void init_sci()  
{  
    SCI3.BRR = BRR_9600;    /*ビットレート 9600bps*/  
    wait(1);  
    wait(1);  
    wait(1);  
    SCI3.SCR3.BYTE = 0X30; /*TE,RE=1*/  
    IO.PMR1.BIT.TXD = 1;    /*P22→TXD 端子*/  
}
```

2. sci.h

③Get_EyER_data 関数のプロトタイプ宣言の追加

```
#ifndef _SCI_H_  
#define _SCI_H_  
  
void init_sci();                /*SCI 初期化関数*/  
void sci_write( unsigned char data); /*1 バイトライト*/  
unsigned char sci_read();        /*1 バイトリード*/  
void sci_puts( const char* str);    /*ストリングライト*/  
void sci_dataout(long data , unsigned char len);/*数値データ出力*/  
unsigned char Get_EyER_data(void);    /* EyER コマンド判定関数 */  
#endif
```

3. acrobat.c

④acrobat 関数の変更追加

```
void acrobat(unsigned char type ,long time , float value)  
{  
  
    float                omegaD=0.0f;        /*車輪回転角速度の目標値*/  
    float                omega_err = 0.0f;    /*車輪回転角速度の制御偏差*/  
    float                omega_sum = 0.0f;    /*車輪回転角速度の制御偏差の積算値*/  
    float                tau = 0.0f;         /*トルク指令値*/  
    float                tau_1 =0.0f;        /*前制御周期のトルク指令値*/  
    float                count_err;         /*位置制御用の制御偏差*/  
    float                count_sum = 0.0f;   /*位置制御用の制御偏差の積算値*/  
}
```

```

long          l = 1;          /*ループカウンタ*/
float         countD = C_C * value; /*ロータリエンコーダカウント値の目標値*/

unsigned char get_EyER;      /* EyER からの受信データを格納する変数 */
long         count = 0;      /* カウントに使用する変数 */

sci_puts("TEST BB64E3687F ㊄r㊄n");          /* シリアル通信確認用のシリアル出力 */
if(type==0)    sci_puts("車輪速度制御㊄r㊄n");
else          sci_puts("位置制御㊄r㊄n");      /* 位置制御のときのみ EyER での動作可能 */

time /= 10;    /*ループ回数決定*/
set_active(1); /*モータ制御開始*/
while(1){
    get_EyER = Get_EyER_data();          /* get_EyER に命令コマンドを格納する */
    if(get_cflag()){                    /*制御周期到来(10ms 毎)*/
        /* =====倒立走行制御計算*/
        /*---モータに流す電流の目標値を設定*/
        Id = tau/K_MOTOR;

        /*--- ↓ 次の制御入力を計算 ↓ */
        current_status();                /*状態量を観測 */
        omega_err = omegaD-dFai;         /*制御偏差算出*/
        tau = F1*ThetaG+F2*dTheta+F3*dFai+F4*omega_sum+F5*tau_1;

        omega_sum += omega_err;          /*制御偏差を積算*/
        tau_1 = tau;                     /*制御入力(トルク)をセーブ*/
        /*--- ↑ 計算終了 ↑ */

        /* =====付加機能*/
        /*---位置制御モード*/
        if(type)
        {
            if(get_EyER == 9)
            {
                /* 通常の倒立制御 */
                count++;                  /* 一定時間をつくるためのカウント値 */
                if(count<10)
                {
                    /* ロータリエンコーダの初期カウント値を格納 */
                    countD=get_count();
                }

                if(count>=50) /* 一定時間経過したときの処理 */

```

```

    {
        count_err = (float)get_count()-countD;
        /*ロータリエンコーダのカウンタ値から車輪角速度目標値を計算*/
        omegaD = KP_POS*count_err+KI_POS*count_sum;

        /*車輪角速度目標値を設定*/
        count_sum+=count_err;
        /*カウンタエラーを積算*/
        if(omegaD < -15.0f) omegaD = -15.0f; /*出力制限*/
        if(omegaD > 15.0f) omegaD = 15.0f; /*出力制限*/
        /*---LED 点灯 点灯パターン*/
        if (Id<-0.5f) set_led(0,0X03); /*○○●●*/
        else if (Id< 0.0f) set_led(0,0X02); /*○○●○*/
        else if (Id==0.0f) set_led(0,0X0F); /*●●●●*/
        else if (Id< 0.5f) set_led(0,0X04); /*○●○○*/
        else set_led(0,0X0C); /*●●○○*/
    }
    if(count>1000) count=200; /* 上限値を超えると 0 にクリアする */
}
else if(get_EyER == 0) /* EyER からの受信データが「1」のときの処理 */
{
    count=0;
    omegaD = 10.0f; /* モータへの制御量を設定 モータ正回転 */
    set_led(0,0X03); /*○○●●*/
}
else if(get_EyER == 1) /* EyER からの受信データが「2」のときの処理 */
{
    count=0;
    omegaD = -8.0f; /* モータへの制御量を設定 モータ逆回転 */
    set_led(0,0X0C); /*●●○○*/
}
}
/*---速度制御モード*/
else
    omegaD = value; /*ユーザ指令値をそのまま車輪速度目標値に設定*/

if( duty_check() || (1 == time) )break;
l = (l & 0x7fffffff)+1;

/*---LED 点灯 点灯パターン*/
/*if (Id<-0.5f) set_led(0,0X03); ○○●●*/
/*else if (Id< 0.0f) set_led(0,0X02); ○○●○*/
/*else if (Id< 0.5f) set_led(0,0X04); ○●○○*/

```

```
                /*else                set_led(0,0X0C);                ●●○○*/  
  
            }  
        }  
        set_led(0,0x00);        /*LED 全消灯*/  
        set_active(0);        /*モータ制御終了*/  
        output_wheel(0);        /*モータ停止*/  
    }  
}
```

各種設定

- ・ PUPPY 側のディップ SW 設定
3 を ON 側、4 を数字側 （ 位置制御モード時のみ動作 ）s
- ・ BB64E3867F 側の J6_RXD ジャンパ設定
TTL 側をショート
- ・ EyER の使用するチャンネル
チャンネル 1、2 を使用
チャンネル 1、2 に登録されている赤外線を受信した場合に、前進、後退をいたします。