程序设计说明

1.使用定时器1产生两路PWM，控制两个电机；
2.定时器2保持和上一版的相同功能；
3.定时器3设置成电机B（左电机的）编码器模式；
4.定时器4设置成电机A（右电机的）编码器模式；
5.定时器5保持和上一版的相同功能，通过usb发送数据；
6.定时器6计算速度和调节转速；
7.定时器6设置如下：

```
TIM_TimeBaseStructure.TIM_Period = 4000;        //定时器初始值 72000000/9/4000 = 2000Hz
TIM_TimeBaseStructure.TIM_Prescaler=(9-1);    //时钟预分频

//由计算可知，定时器6的周期为2000Hz，我们设置成速度200Hz采样频率：
#define SPEED_SAMPLING_TIME        9              // (9+1)*500usec = 5MS   ,200hz
#define SPEED_SAMPLING_FREQ (u16)(2000/(SPEED_SAMPLING_TIME+1))  //200hz 小车速度采样频率
static unsigned short int hSpeedMeas_Timebase_500us = SPEED_SAMPLING_TIME; //电机编码数采集时间间隔

//定时器6中断函数如下：
void TIM6_IRQHandler(void)
{

    //young
    if( TIM_GetITStatus(TIM6 , TIM_IT_Update) != RESET )
    {
        if (hSpeedMeas_Timebase_500us !=0)//电机编码数采集时间间隔未到
        {
                hSpeedMeas_Timebase_500us--;//开始倒数
        }
        else    //电机编码数采集时间间隔到了
        {
                hSpeedMeas_Timebase_500us = SPEED_SAMPLING_TIME;//恢复电机编码数采集时间间隔
        //以下为计算函数
      }

    TIM_ClearITPendingBit(TIM6 , TIM_FLAG_Update);
  }
}
```

端口定时器及连接说明

PWM:
TIM1： PA8 --> OC1 PA11 --> OC4;

编码器：
TIM3： PA6 PA7 （左电机） --> 电机B
TIM4： PB6 PB7 （右电机） --> 电机A

定时器：
TIM5：传输编码器信息，以及位置坐标，方向角，线速度和角速度等；
TIM6：速度计算定时器初始化

驱动板接线：
电机B：IN1 --- PC6 IN2 --- PC7
电机A：IN1 --- PC10 IN2 --- PC11

串口数据打印：
USART1_TX --- PA9 USART_RX --- PA10
可调用printf函数进行打印数据测试程序

通信说明

1.可使用共同体传输float类型数据，测试阶段暂且使用int型数据，添加代码如下：

```
//us_mcu_transfer.h
typedef struct _SJS_MOVE_STATE{
    unsigned long long id;
    int x_position[4];
    int y_position[4];
    int yaw[4];
}SJS_MOVE_STATE;

//us_mcu_transfer.c
//x ---> x坐标
//y ---> y坐标
//yaw ---> 方向角
//此处的cmd，使用此前的encoder的cmd代码：
//#define SJS_ROBOT_ENCODER        0x66
//此处id，还未添加timer计算，后续添加
int sjs_robot_encoder_send(unsigned char cmd, int x, int y, int yaw, unsigned long long id);
```

2.在int robot_mcu_Encoder_decoder中，我们会判断ROBOT_ENCODER_EN标志位，判断语句如下：

```
#define SJS_ROBOT_ENCODER_WRITE            0x63
#define SJS_ROBOT_ENCODER_READ            0x64
#define SJS_ROBOT_ENCODER_CONFIG        0x65
#define SJS_ROBOT_ENCODER                        0x66
```

```
case SJS_ROBOT_ENCODER_WRITE:
            if(data[0] == ENCODER_EN){
                ROBOT_ENCODER_EN = 1; //当此标志位置为1的时候，在定时器5中断中，才会发送数据（通过usb）
            }else if(data[0] == ENCODER_DIS){
                ROBOT_ENCODER_EN = 0;
            }
            break;
        case SJS_ROBOT_ENCODER_READ:

            break;
```

在此我们加一步判断，如果ROBOT_ENCODER_EN == 0时，我们在odometry()函数中，进行如下判断

```
if(0 == ROBOT_ENCODER_EN){
        //当不需要获取数据时，所有数据复位，开始新一轮位置计算，坐标原点位于当前点。
        distance_sum = 0;
        distance_diff = 0;
        orientation = 0;
        orientation_1 = 0;
        position_x = 0;
        position_y = 0;
        velocity_linear = 0;
        velocity_angular = 0;
    }else{
    //计算x,y坐标，方向角，线速度，角速度等数据
  }
```

也就是说，当收到 ROBOT_ENCODER_EN = 1时，即在此时开始计算坐标，当前位置为坐标原点。

文件说明

FUNCTION文件夹：主要功能函数存放处，比如pwm，计算，pid，数据传输等；
HARDWARE文件夹：硬件初始化，定时器，编码器等；