Stm32 接收报文格式

采用 modbus 协议作为参考，方便于后续对哨兵以及自定义机器人的通信

自瞄帧格式

自瞄数据功能码为：0x16

Stm32(云台控制器)ID：0x01　　帧类型：标准帧

大小：72bit

数据	DATA[0]	DATA[1]	DATA[2]	DATA[3]	DATA[4]	DATA[5]	DATA[6]	DATA[7]
内容	Stm32ID	功能码	是否存在敌人	Yaw 高	Yaw 低	Pitch 高	Pitch 低	CRC 校验
位数	8 位	8 位	8 位	8 位	8 位	8 位	8 位	16 位

回传速率：10ms 每帧 (100 帧)

Yaw 和 Pitch 范围：-180~180

注：无需回传

注：0 为中心角度，采用角度制，单位°

注：yaw 和 pitch 为 float 格式

注：实际传感器分辨帧率不到 100 帧

注：小端格式发送

说明

从上往下看顺时针旋转 yaw 轴，yaw 值增大

从后向前看向上旋转 pitch 轴，pitch 值增大

要求发送时间不超过 1ms，假设 1ms 发送 72 位，1 秒就是 72000 位，考虑到后续回传或其他数据的发送，约定波特率为 115200

附录：

CRC 校验：

```
/* Private define------------------------------------------------------------*/

/* Private variables---------------------------------------------------------*/

/* Private function prototypes-----------------------------------------------*/
static uint16_t CRC_Check(uint8_t*,uint8_t);   //CRC 校验

/* Public variables----------------------------------------------------------*/
CRC_16_t    CRC_16 = {0,0,0,CRC_Check};
/*****************************************************
```

说明：CRC 添加到消息中时，低字节先加入，然后高字

CRC 计算方法：

1.预置 1 个 16 位的寄存器为十六进制 FFFF(即全为 1);称此寄存器为 CRC 寄存器;

2.把第一个 8 位二进制数据(既通讯信息帧的第一个字节)与 16 位的 CRC 寄存器的低 8 位相异或，把结果放于 CRC 寄存器;

3.把 CRC 寄存器的内容右移一位(朝低位)用 0 填补最高位，并检查右移后的移出位;

4.如果移出位为 0:重复第 3 步(再次右移一位);

如果移出位为 1:CRC 寄存器与多项式 A001(1010 0000 0000 0001)进行异或;

5.重复步骤 3 和 4，直到右移 8 次，这样整个 8 位数据全部进行了处理;

6.重复步骤 2 到步骤 5，进行通讯信息帧下一个字节的处理;

7.将该通讯信息帧所有字节按上述步骤计算完成后，得到的 16 位 CRC 寄存器的高、低字节进行交换;

```
/*
    * @name     CRC_Check
    * @brief    CRC 校验
    * @param    CRC_Ptr->数组指针，LEN->长度
    * @retval CRC 校验值
*/
uint16_t CRC_Check(uint8_t *CRC_Ptr,uint8_t LEN)
{
    uint16_t CRC_Value = 0;
    uint8_t   i              = 0;
    uint8_t   j              = 0;
```

```
    CRC_Value = 0xffff;
    for(i=0;i<LEN;i++)
    {
        CRC_Value ^= *(CRC_Ptr+i);
        for(j=0;j<8;j++)
        {
            if(CRC_Value & 0x00001)
                CRC_Value = (CRC_Value >> 1) ^ 0xA001;
            else
                CRC_Value = (CRC_Value >> 1);
        }
    }
    CRC_Value = ((CRC_Value >> 8) +   (CRC_Value << 8)); //交换高低字节
    return CRC_Value;
}
/*************************************************************
  End Of File
*************************************************************/
```

原文链接：https://blog.csdn.net/qq_28576837/article/details/127981068

发送方软件设计

```
#pragma pack(push, 1) //  指定  1  字节对齐
typedef struct
{
  __uint8_t stm32_id;
  __uint8_t function_code;
  bool enemy;
  __uint8_t yaw_h;
  __uint8_t yaw_l;
  __uint8_t pitch_h;
  __uint8_t pitch_l;
  __uint16_t CRC;
} YunTai;
#pragma pack(pop) //  恢复默认对齐
```

```
int modbus_write_registers_noreply(unsigned char slave_id, unsigned char function_code,
float yaw, float pitch)
{
    switch (function_code)
    {
    case MB_WRITE_AUTOMATIC_AIMING_REGISTERS:
    {
        uint16_t *data_byte_yaw = (uint16_t *)&yaw;
        uint16_t *data_byte_pitch = (uint16_t *)&pitch;
        YunTai YunTai2stm32;
        YunTai2stm32.stm32_id = MB_STM32_YUNTAI_ID;
        YunTai2stm32.function_code = MB_WRITE_AUTOMATIC_AIMING_REGISTERS;
        YunTai2stm32.enemy = 0;
        YunTai2stm32.yaw_h = data_byte_yaw[1];
        YunTai2stm32.yaw_l = data_byte_yaw[0];
        YunTai2stm32.pitch_h = data_byte_pitch[1];
        YunTai2stm32.pitch_l = data_byte_pitch[0];
        YunTai2stm32.CRC = CRC_Check((uint8_t *)&YunTai2stm32, 56);
        SerialPortWriteBuffer(Uart_inf.UID2STM32, &YunTai2stm32, sizeof(YunTai2stm32));
        break;
    }
    default:
    {
        printf("没有实现发送格式\n");
        break;
    }
    }
    return MB_OK;
}
```

数据示例：

[17:15:06.436] 01 16 00 A7 DD AF 50 A0 05

[17:15:06.446] 01 16 00 A1 82 AF 28 CC DD

[17:15:06.455] 01 16 00 92 01 B0 4F 58 7B

[17:15:06.465] 01 16 00 90 44 AB EE 1D D7

[17:15:06.475] 01 16 00 5B 4F AD EE 38 9B

[17:15:06.485] 01 16 00 4C 19 9A EC 72 9F

[17:15:06.495] 01 16 00 8A 52 2E 86 3C F3

[17:15:06.505] 01 16 00 32 E0 6D FD FB A1

[17:15:06.515] 01 16 00 32 79 72 BB 35 CF

[17:15:06.526] 01 16 00 9D EE 9B 02 4A 86

[17:15:06.535] 01 16 00 9E AA B0 A2 82 01

[17:15:06.545] 01 16 00 00 9A 5F 10 3C 4C

软件设计建议参考

urands/stModbus：用于 Cortex-M 的 Modbus RTU（STM32 系列：STM32F103、STM32F3xx）

alejoseb/Modbus-STM32-HAL-FreeRTOS: Modbus TCP and RTU, Master and Slave for STM32 using Cube HAL and FreeRTOS