遥控通信总结

一、概述

在逛论坛的时候发现还没有同学对遥控部分内容做过比较详细的整理，因此在这里做一个遥控器&键鼠的通讯和控制上的总结，主要包括硬件、通讯格式、解码和一些小 tips，便于以后的同学学习，本人也是第一年参加比赛，欢迎各路大神在下方留言补充纠正~

二、硬件&通信介绍

RM 比赛中各个参赛队伍使用的都是大疆官方提供的遥控器套装，包括遥控器和接收机，接收机上共三个引脚：VCC，GND，DBUS（数据通道），首次使用需要进行遥控器和接收机配对，在两者都上电的情况下短按接收机上的对频按键即可（位于指示灯旁边）。在官方更新了遥控器的固件版本之后，遥控器上手轮也已经开放。

遥控器通讯采用的是 2.5GHz 频段的 DBUS 通讯协议（这里不对 DBUS 进行过多阐释了，有兴趣的同学可以自行学习），接收机工作电压为 4-8.4V，输出信号满足 TTL 电平，但是为负逻辑，因此接收机和单片机之间需要加反相器来获得正确的数据（官方开发板的 USART1 接口已经集成）。

接收机的数据发送周期为 14ms，每次发送 18 字节数据，和单片机通过串口通信，通信参数如下：

DBUS通信参数如下表所示：

DBUS 参数	数值
波特率	100Kbps
单元数据长度	8
奇偶校验位	偶校验
结束位	1
流控	无

每次发送的数据如下：

域	**通道 0**	**通道 1**	**通道 2**	**通道 3**	**S1**	**S2**
偏移	0	11	22	33	46	44
长度(bit)	11	11	11	11	2	2
符号位	无	无	无	无	无	无
范围	最大值 1684 中间值 1024 最小值 364	最大值 1684 中间值 1024 最小值 364	最大值 1684 中间值 1024 最小值 364	最大值 1684 中间值 1024 最小值 364	最大值 3 最小值 1	最大值 3 最小值 1
功能	无符号类型遥控器通道 0 控制信息	无符号类型遥控器通道 1 控制信息	无符号类型遥控器通道 2 控制信息	无符号类型遥控器通道 3 控制信息	遥控器发射机 S1 开关位置 1：上 2：下 3：中	遥控器发射机 S2 开关位置 1：上 2：下 3：中
域	**鼠标 X 轴**	**鼠标 Y 轴**	**鼠标 Z 轴**	**鼠标左键**	**鼠标右键**	**按键**
偏移	48	64	80	96	104	112
长度 (bit)	16	16	16	8	8	16
符号位	有	有	有	无	无	无
范围	最大值 32767 最小值 -32768 静止值 0	最大值 32767 最小值 -32768 静止值 0	最大值 32767 最小值 -32768 静止值 0	最大值 1 最小值 0	最大值 1 最小值 0	位值标识
功能	鼠标在 X 轴的移动速度 负值标识往左移动 正值标识往右移动	鼠标在 Y 轴的移动速度 负值标识往左移动 正值标识往右移动	鼠标在 Z 轴的移动速度 负值标识往左移动 正值标识往右移动	鼠标左键是否按下 0 --没按下 1 --按下	鼠标右键是否按下 0 --没按下 1 --按下	每个按键对应一个位，共 16 个按键，按下该位为 1，未按下该位为 0

键盘键位对应键值（实测得出）：

键位	W	S	A	D	Shift	Ctrl	Q	E
键值	0x0001	0x0002	0x0004	0x0008	0x0010	0x0020	0x0040	0x0080
键位	R	F	G	Z	X	C	V	B
键值	0x0100	0x0200	0x0400	0x0800	0x1000	0x2000	0x4000	0x8000

三、解码

遥控器的数据每一帧为 18 个字节，采用串口接收，每个字节都用 CPU 处理效率较低，因此考虑使用 DMA，串口中断使能 IDLE（线路空闲），当一帧数据接收完毕之后触发串口中断，调用 DMA 进行接收。官方手册提供了基于标准库的串口&DMA 配置以及解码函数，在这里只贴出中断服务函数（HAL 库）以及解码函数，串口和 DMA

略去。

代码实现（如果要使用请提前定义相关变量）：

1.中断服务函数

```
void RemotreCtl_Data_Receive(void)
{
    uint32_t rx_data_len = 0;
    if((__HAL_UART_GET_FLAG(&huart1,UART_FLAG_IDLE)!=RESET))        //判断一帧数据是否接收完成
    {
      __HAL_UART_CLEAR_IDLEFLAG(&huart1);                           //清空 IDLE 标志位
      (void)USART1->SR;                                             //清空 SR 寄存器
      (void)USART1->DR;                                             //清空 DR 寄存器
      __HAL_DMA_CLEAR_FLAG(&hdma_usart1_rx,DMA_FLAG_TCIF2_6);   //清除 DMA 传输完成标注
      HAL_UART_DMAStop(&huart1);                                   //停止 DMA 接收
      rx_data_len=BSP_USART1_DMA_RX_BUF_LEN-__HAL_DMA_GET_COUNTER(&hdma_usart1_rx); //数据量
      HAL_UART_Receive_DMA(&huart1, USART1_DMA_RX_BUF, BSP_USART1_DMA_RX_BUF_LEN); //读数据
      if(rx_data_len == 18)                                         //如果数据量正确，则进行解码
      {
            RC_DataHandle(USART1_DMA_RX_BUF);
      }
    }
}
```

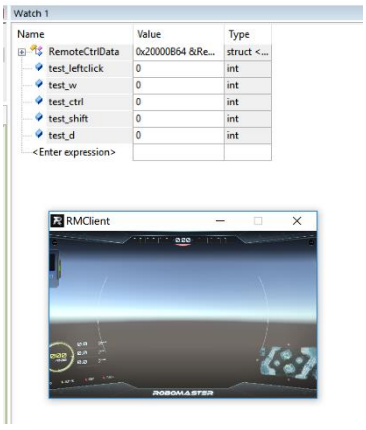
2.解码函数

```
void RC_DataHandle(uint8_t *pData)
{
      if(pData == NULL)
    {
        return;
    }
      /*pData[0]为 ch0 的低 8 位，Data[1]的第三位为 ch0 的高三位*/
    RemoteCtrlData.remote.ch0 = ((uint16_t)pData[0] | (uint16_t)pData[1] << 8) & 0x07FF;
      /*pData[1]的高 5 位为 ch1 的低五位，pData[2]的低 6 位为 ch1 的高 6 位*/
      RemoteCtrlData.remote.ch1 = ((uint16_t)pData[1] >> 3 | (uint16_t)pData[2] << 5) & 0x07FF;
      /*pData[2]的高 2 位为 ch3 的低二位， pData[3]为 ch3 的 3~10 位，pData[4]的最低位为 ch3 的最高位*/
      RemoteCtrlData.remote.ch2 = ((uint16_t)pData[2] >> 6 | (uint16_t)pData[3] << 2 | (uint16_t)pData[4]
<< 10) & 0x07FF;
      /*pData[4]的高 7 位为 ch4 的低 7 位，pData[5]的低 4 位为 ch4 的高 4 位*/
      RemoteCtrlData.remote.ch3 = ((uint16_t)pData[4] >> 1 | (uint16_t)pData[5] << 7) & 0x07FF
      /*pData[5]的高 8 位为 S1*/
      RemoteCtrlData.remote.s1 = ((pData[5] >> 6) & 0x03);
      /*pData[5]的 6、7 位为 s2*/
      RemoteCtrlData.remote.s2 = ((pData[5] >> 4) & 0x03);
      /*pData[6]和 pData[7]鼠标 x 方向*/
      RemoteCtrlData.mouse.x = ((int16_t)pData[6] | (int16_t)pData[7] << 8);
      /*pData[8]和 pData[9]为鼠标 Y 方向*/
```

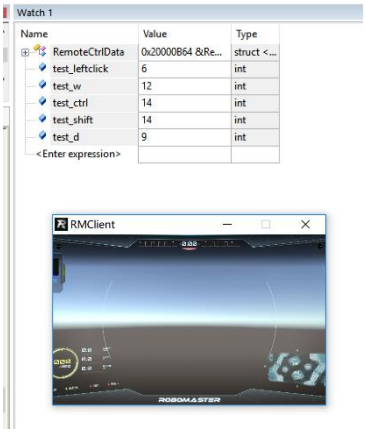
```
    RemoteCtrlData.mouse.y   = ((int16_t)pData[8] | (int16_t)pData[9] << 8);
    /*pData[10]和 pData[11]为鼠标 Z 方向*/
    RemoteCtrlData.mouse.z      = ((int16_t)pData[10] | (int16_t)pData[11] << 8);
    /*pData[12]为鼠标左键*/
    RemoteCtrlData.mouse.press_l = pData[12];
    /*pData[13]  为鼠标右键*/
    RemoteCtrlData.mouse.press_r = pData[13];
    /*pData[14]和 pData[15]为键盘*/
    RemoteCtrlData.key.v = ((int16_t)pData[14]) | ((int16_t)pData[15] << 8);
}
```

四、键鼠控制的小 tips

键鼠控制和遥控器有所不同，接收机发送频率较高，按键存在延时，按下一个键该键值会被置高一段时间，并且在中断中不能通过延时来解决该问题。



按键之前



短暂按下按键之后

基本上短按都会持续 7~14 次接收周期， 所以如果想要用按键做模式切换的话，至少需要延时 20 个接收周期，也就是在第一次接收到这个键之后 20 次接收中断中不再对这个键进行解码，因为模式切换通常来说不会在短时间内进行，所以我建议 1S 之内都不对这个按键进行解码，从而实现可靠模式切换。

2019 年 1 月 30 日

北京理工大学

RM 战队电控组整理