RoboMasters 遥控器控制协议

---V1.4

RoboMasters 比赛中，参赛队员需使用 DJI 官方提供的 2.4GHz 遥控系统，具体型号为 DT7 和 DR16
详细信息如下所示：

官方产品主页	http://www.dji.com/cn/product/dt7-dr16-rc-system
遥控器驱动	http://download.dji-innovations.com/downloads/dt7/cn/DT7&DR16_RC_System_User_Manual_v1.00_cn.pdf
使用手册	http://download.dji-innovations.com/downloads/driver/DJI_WIN_Driver_Installer.exe

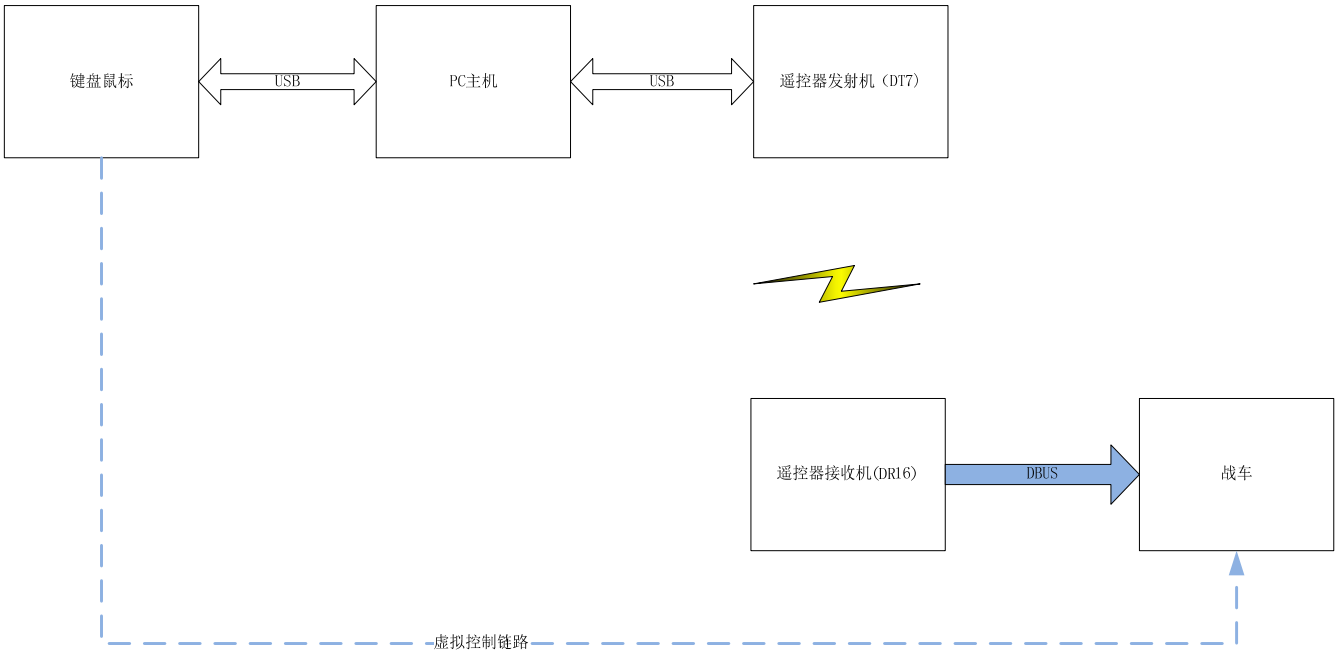
特别注意：

1：仅仅装 DT7 驱动软件，不要安装 DT7&DR16 调参软件

2：RoboMasters 提供的遥控系统是 DJI 特别定制的，不要随意升级遥控器固件，否则会造成遥控器无法使用

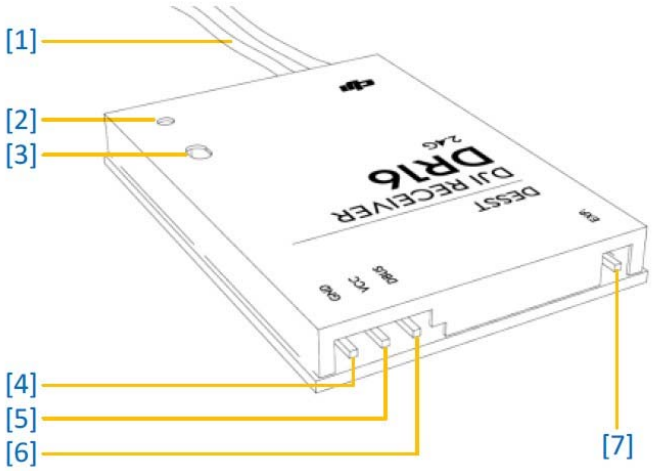
遥控器接收机输出信号为标准 DBUS 协议数据，同时 RoboMasters 官方为了便于 PC 控制战车，在原有遥控器的工程

上，增加了额外的键盘鼠标控制信息，控制链路如下所示



接下来，本文重点介绍图中的蓝色单向 DBUS 控制帧

遥控器接收机



[1] 天线
[2] LED 指示灯
[3] 对频按键
[4] GND
[5] VCC
[6] DBUS
[7] EXP.

这里重点关注 4/5/6 三根控制线，4 是地线，5 是 VCC 用于为接收机供电，电压范围为 4-8.4V，6 是 DBUS 数据输出线

当接收机和发射机建立连接后，接收机会每隔 7ms 通过 DBUS 发送一帧数据（18 字节），DBUS 的通信参数如下

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

注：DBUS 信号控制电平符合 TTL，但是和普通 UART 信号是相反的，所以需要在 MCU 端需要增加三极管取反电路，MCU 才能正常识别出 UART 信号

控制帧的结构如下所示

表 1　遥控器信息

域	**通道 0**	**通道 1**	**通道 2**	**通道 3**	**S1**	**S2**
偏移	0	11	22	33	44	46
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

遥控器的通道和控制开关如下所示

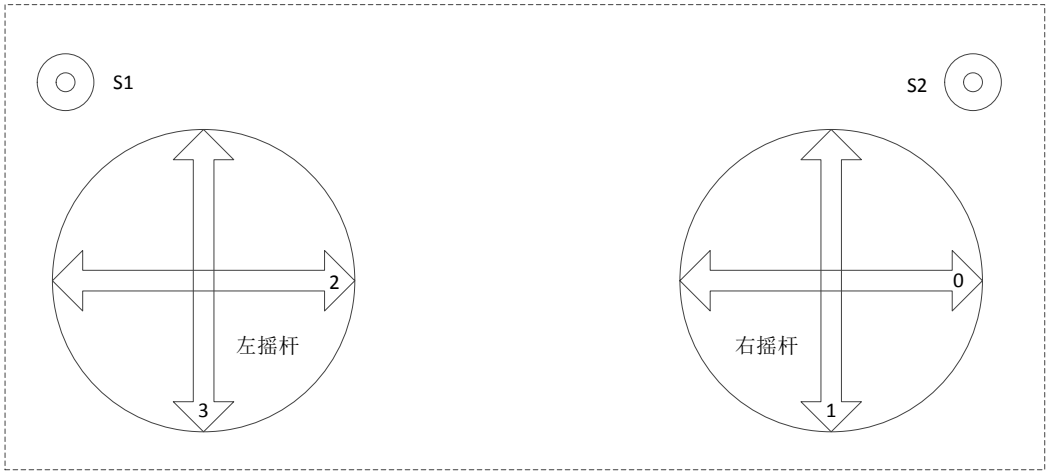


表 2，鼠标信息

域	**鼠标 X 轴**	**鼠标 Y 轴**	**鼠标 Z 轴**	**鼠标左键**	**鼠标右键**
偏移	48	64	80	86	94
长度	16	16	16	8	8

符号位	有	有	有	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
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

表 3，键盘信息

域	**按键**	**保留字段**
偏移	102	
长度	16	
符号位	无	
范围	位值标识	
功能	每个按键对应一个 bit，如下所示 Bit0 ------- W 键 Bit1 ------- S 键 Bit2 ------- A 键 Bit3 ------- D 键 Bit4 ------- Q 键 Bit5 ------- E 键 Bit6 ------- Shift 键 Bit7 ------- Ctrl 键	

对应的数据帧结构如下所示

```
#pragma pack(1)
typedef union
{
    struct
    {
        struct
        {
            uint8_t  ch0_h:8;             //!< Byte 0

            uint8_t  ch0_l:3;             //!< Byte 1
            uint8_t  ch1_h:5;

            uint8_t  ch1_l:6;             //!< Byte 2
            uint8_t  ch2_h:2;

            uint8_t  ch2_m:8;             //!< Byte 3

            uint8_t  ch2_l:1;             //!< Byte 4
            uint8_t  ch3_h:7;

            uint8_t  ch3_l:4;             //!< Byte 5
```

```
            uint8_t  s1:2;
            uint8_t  s2:2;
        }rc;

        struct
        {
            int16_t x;                    //!< Byte 6-7
            int16_t y;                    //!< Byte 8-9
            int16_t z;                    //!< Byte 10-11
            uint8_t press_l;              //!< Byte 12
            uint8_t press_r;              //!< Byte 13
        }mouse;

        struct
        {
            uint16_t v;                   //!< Byte 14-15
        }key;

        uint16_t resv;                    //!< Byte 16-17
    };
    uint8_t buf[18];                      //!< Union --> Byte<0-17>
}RC_Ctl_Define_t;
```

附录： 参考代码（STM32F4 平台）

```
// #pragma pack(1)
// typedef union
// {
//     struct
//     {
//         struct
//         {
//             uint8_t  ch0_h:8;             //!< Byte 0
//
//             uint8_t  ch0_l:3;             //!< Byte 1
//             uint8_t  ch1_h:5;
//
//             uint8_t  ch1_l:6;             //!< Byte 2
//             uint8_t  ch2_h:2;
//
//             uint8_t  ch2_m:8;             //!< Byte 3
//
//             uint8_t  ch2_l:1;             //!< Byte 4
//             uint8_t  ch3_h:7;
//
//             uint8_t  ch3_l:4;             //!< Byte 5
//             uint8_t  s1:2;
//             uint8_t  s2:2;
//         }rc;
//
//         struct
//         {
//             int16_t x;                    //!< Byte 6-7
//             int16_t y;                    //!< Byte 8-9
//             int16_t z;                    //!< Byte 10-11
//             uint8_t press_l;              //!< Byte 12
//             uint8_t press_r;              //!< Byte 13
//         }mouse;
//
//         struct
//         {
//             uint16_t v;                   //!< Byte 14-15
//         }key;
//
//         uint16_t resv;                    //!< Byte 16-17
//     };
//     uint8_t buf[18];                      //!< Union --> Byte<0-17>
// }RC_Ctl_Define_t;

/* ----------------------- RC Channel Definition---------------------------- */
#define RC_CH_VALUE_MIN              ((uint16_t)364 )
#define RC_CH_VALUE_OFFSET           ((uint16_t)1024)
#define RC_CH_VALUE_MAX              ((uint16_t)1684)

/* ----------------------- RC Switch Definition---------------------------- */
#define RC_SW_UP                     ((uint16_t)1)
#define RC_SW_MID                    ((uint16_t)3)
#define RC_SW_DOWN                   ((uint16_t)2)

/* ----------------------- PC Key Definition-------------------------------- */
#define KEY_PRESSED_OFFSET_W         ((uint16_t)0x01<<0)
#define KEY_PRESSED_OFFSET_S         ((uint16_t)0x01<<1)
#define KEY_PRESSED_OFFSET_A         ((uint16_t)0x01<<2)
#define KEY_PRESSED_OFFSET_D         ((uint16_t)0x01<<3)
#define KEY_PRESSED_OFFSET_Q         ((uint16_t)0x01<<4)
#define KEY_PRESSED_OFFSET_E         ((uint16_t)0x01<<5)
#define KEY_PRESSED_OFFSET_SHIFT     ((uint16_t)0x01<<6)
#define KEY_PRESSED_OFFSET_CTRL      ((uint16_t)0x01<<7)

/* ----------------------- Data Struct ------------------------------------- */
typedef struct
{
```

```
    struct
    {
        uint16_t ch0;
        uint16_t ch1;
        uint16_t ch2;
        uint16_t ch3;
        uint8_t  s1;
        uint8_t  s2;
    }rc;

    struct
    {
        int16_t x;
        int16_t y;
        int16_t z;
        uint8_t press_l;
        uint8_t press_r;
    }mouse;

    struct
    {
        uint16_t v;
    }key;
}RC_Ctl_t;

/* ----------------------- Internal Data ----------------------------------- */
volatile unsigned char sbus_rx_buffer[25];
static RC_Ctl_t RC_Ctl;

/* ----------------------- Function Implements  ---------------------------- */

/**************************************************************************
 * @fn      RC_Init
 *
 * @brief   configure stm32 usart2 port
 *          -   USART Parameters
 *              -   100Kbps
 *              -   8-N-1
 *          -   DMA Mode
 *
 * @return  None.
 *
 * @note    This code is fully tested on STM32F405RGT6 Platform, You can port it
 *          to the other platform.
 */
void RC_Init(void)
{
    /* -------------- Enable Module Clock Source ----------------------------*/
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA | RCC_AHB1Periph_DMA1, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);
    GPIO_PinAFConfig(GPIOA,GPIO_PinSource3, GPIO_AF_USART2);

    /* -------------- Configure GPIO ---------------------------------------*/
    {
        GPIO_InitTypeDef  gpio;
        USART_InitTypeDef usart2;

        gpio.GPIO_Pin   = GPIO_Pin_3 ;
        gpio.GPIO_Mode  = GPIO_Mode_AF;
        gpio.GPIO_OType = GPIO_OType_PP;
        gpio.GPIO_Speed = GPIO_Speed_100MHz;
        gpio.GPIO_PuPd  = GPIO_PuPd_NOPULL;
        GPIO_Init(GPIOA, &gpio);

        USART_DeInit(USART2);
        usart2.USART_BaudRate            = 100000;
        usart2.USART_WordLength          = USART_WordLength_8b;
        usart2.USART_StopBits            = USART_StopBits_1;
        usart2.USART_Parity              = USART_Parity_Even;
        usart2.USART_Mode                = USART_Mode_Rx;
```

```
        usart2.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
        USART_Init(USART2,&usart2);

        USART_Cmd(USART2,ENABLE);
        USART_DMACmd(USART2,USART_DMAReq_Rx,ENABLE);
    }

    /* -------------- Configure NVIC  ----------------------------------------*/
    {
        NVIC_InitTypeDef  nvic;

        nvic.NVIC_IRQChannel                    = DMA1_Stream5_IRQn;
        nvic.NVIC_IRQChannelPreemptionPriority = 1;
        nvic.NVIC_IRQChannelSubPriority        = 1;
        nvic.NVIC_IRQChannelCmd                 = ENABLE;
        NVIC_Init(&nvic);
    }

    /* -------------- Configure DMA ----------------------------------------*/
    {
        DMA_InitTypeDef   dma;

        DMA_DeInit(DMA1_Stream5);
        dma.DMA_Channel              = DMA_Channel_4;
        dma.DMA_PeripheralBaseAddr   = (uint32_t)&(USART2->DR);
        dma.DMA_Memory0BaseAddr      = (uint32_t)sbus_rx_buffer;
        dma.DMA_DIR                  = DMA_DIR_PeripheralToMemory;
        dma.DMA_BufferSize           = 18;
        dma.DMA_PeripheralInc        = DMA_PeripheralInc_Disable;
        dma.DMA_MemoryInc            = DMA_MemoryInc_Enable;
        dma.DMA_PeripheralDataSize   = DMA_PeripheralDataSize_Byte;
        dma.DMA_MemoryDataSize       = DMA_MemoryDataSize_Byte;
        dma.DMA_Mode                 = DMA_Mode_Circular;
        dma.DMA_Priority             = DMA_Priority_VeryHigh;
        dma.DMA_FIFOMode             = DMA_FIFOMode_Disable;
        dma.DMA_FIFOThreshold        = DMA_FIFOThreshold_1QuarterFull;
        dma.DMA_MemoryBurst          = DMA_Mode_Normal;
        dma.DMA_PeripheralBurst      = DMA_PeripheralBurst_Single;
        DMA_Init(DMA1_Stream5,&dma);

        DMA_ITConfig(DMA1_Stream5,DMA_IT_TC,ENABLE);
        DMA_Cmd(DMA1_Stream5,ENABLE);
    }
}

/*****************************************************************************
 * @fn      DMA1_Stream5_IRQHandler
 *
 * @brief   USART2 DMA ISR
 *
 * @return  None.
 *
 * @note    This code is fully tested on STM32F405RGT6 Platform, You can port it
 *          to the other platform.
 */
void DMA1_Stream5_IRQHandler(void)
{
    if(DMA_GetITStatus(DMA1_Stream5, DMA_IT_TCIF5))
    {
        DMA_ClearFlag(DMA1_Stream5, DMA_FLAG_TCIF5);
        DMA_ClearITPendingBit(DMA1_Stream5, DMA_IT_TCIF5);

        RC_Ctl.rc.ch0 = (sbus_rx_buffer[0]| (sbus_rx_buffer[1] << 8)) & 0x07ff;          //!< Channel 0
        RC_Ctl.rc.ch1 = ((sbus_rx_buffer[1] >> 3) | (sbus_rx_buffer[2] << 5)) & 0x07ff;  //!< Channel 1
        RC_Ctl.rc.ch2 = ((sbus_rx_buffer[2] >> 6) | (sbus_rx_buffer[3] << 2) |           //!< Channel 2
                         (sbus_rx_buffer[4] << 10)) & 0x07ff;
        RC_Ctl.rc.ch3 = ((sbus_rx_buffer[4] >> 1) | (sbus_rx_buffer[5] << 7)) & 0x07ff;  //!< Channel 3
        RC_Ctl.rc.s1  = ((sbus_rx_buffer[5] >> 4)& 0x000C) >> 2;                         //!< Switch left
        RC_Ctl.rc.s2  = ((sbus_rx_buffer[5] >> 4)& 0x0003);                              //!< Switch right
```

```
        RC_Ctl.mouse.x = sbus_rx_buffer[6] | (sbus_rx_buffer[7] << 8);                  //!< Mouse X axis
        RC_Ctl.mouse.y = sbus_rx_buffer[8] | (sbus_rx_buffer[9] << 8);                  //!< Mouse Y axis
        RC_Ctl.mouse.z = sbus_rx_buffer[10] | (sbus_rx_buffer[11] << 8);                //!< Mouse Z axis
        RC_Ctl.mouse.press_l = sbus_rx_buffer[12];                                      //!< Mouse Left Is Press ?
        RC_Ctl.mouse.press_r = sbus_rx_buffer[13];                                      //!< Mouse Right Is Press ?

        RC_Ctl.key.v = sbus_rx_buffer[14] | (sbus_rx_buffer[15] << 8);                  //!< KeyBoard value
    }
}
```