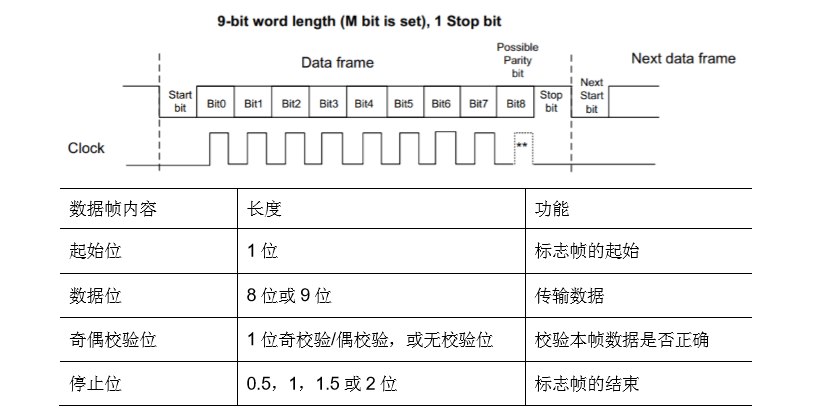
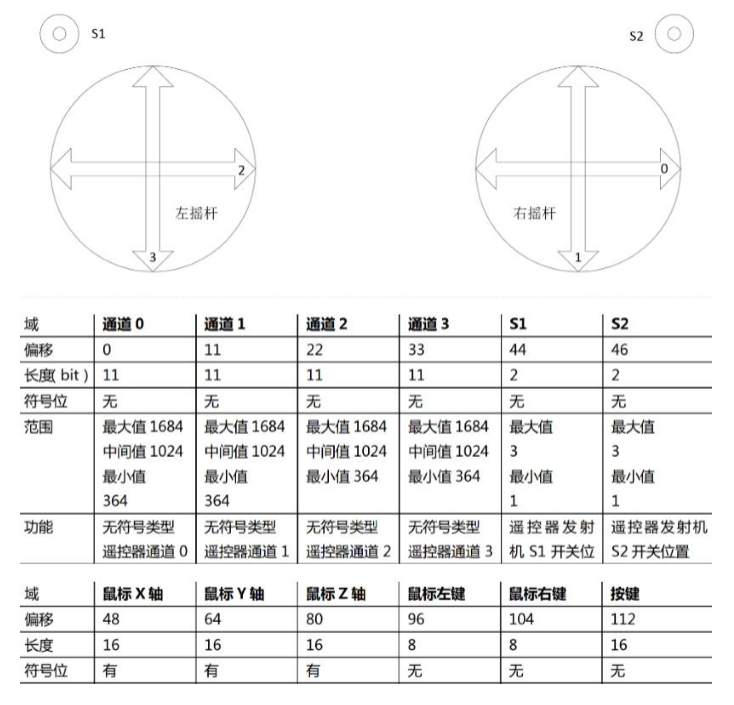
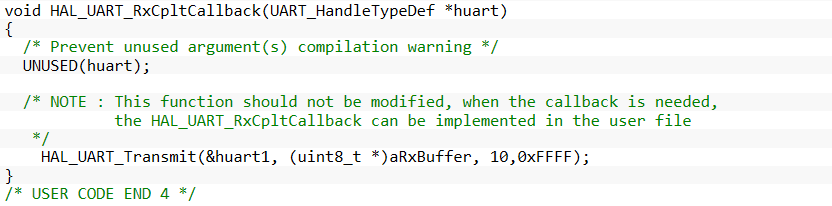
# 串口的概念

1. 串口是一种在单片机，传感器，执行模块等诸多设备上常用的通讯接口，在比赛中，可以通过串口读取遥控器发送来的数据，也可以通过串口读取超声波等传感器的数据，也可以使用串口在单片机和运行计算机视觉的电脑之间进行通讯。
2. 串口全称为通用串行通信接口，是一种非常常用的通信接口。串行即以高低电平表示 1 和0，将数据一位一位顺序发送给接收方。通用串行通信接口有着协议简单，易于控制的优点。
3. 串口的通讯协议由开始位，数据位，校验位，结束位构成。一般以一个低电平作为一帧数据的起始，接着跟随 8 位或者 9 位数据位，之后为校验位，分为奇校验，偶校验和无校验，最 后以一个先高后低的脉冲表示结束位，长度可以设置为 0.5，1，1.5 或 2 位长度。
4. 
5. 一般进行串口通讯时，收发双方要保证遵守同样的协议才能正确的完成收发，除了协议要一致之外，还有一个非常重要的要素要保持一致，那就是通讯的速率，即波特率。波特率是指发送数据的速率,单位为波特每秒，一般串口常用的波特率有 115200，38400，9600 等。串口的波特率和总线时钟周期（clock）成倒数关系，即总线时钟周期越短，单位时间内发送的 码元数量越多，串口波特率就越高。

# 遥控器的原理

1. 遥控器和 stm32 之间采用 DBUS 协议进行通讯。DBUS 通讯协议和串口类似，DBUS 的传 输速率为 100k bit/s，数据长度为 8 位，奇偶校验位为偶校验，结束位 1 位。需要注意的是 DBUS 使用的电平标准和串口是相反的，在 DBUS 协议中高电平表示 0，低电平表示 1，如果使用串口进行接收需要在接收电路上添加一个反相器。
2. 使用 DBUS 接收遥控器的数据，一帧数据的长度为 18 字节，一共 144 位，根据遥控器的说 明书可以查出各段数据的含义，从而进行数据拼接，完成遥控器的解码，如图所示。
3. 
4. 想要和和遥控器的接收机进行串口通信，需要将单片机对应的串口按照说明书的要求设置一下参数
5. 

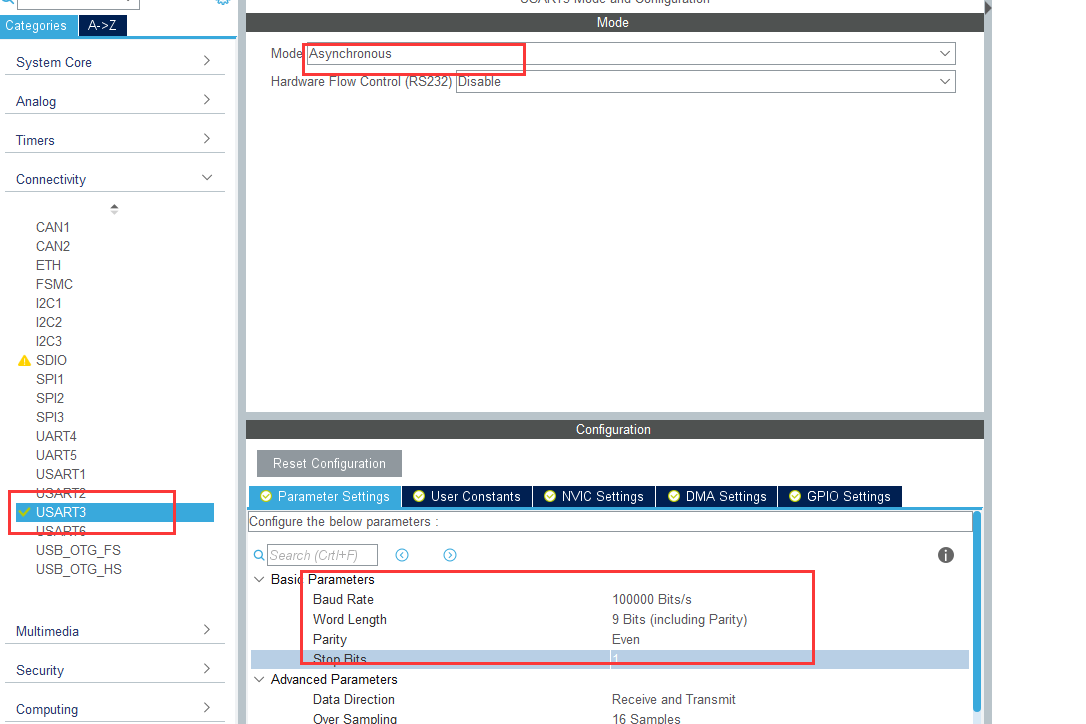
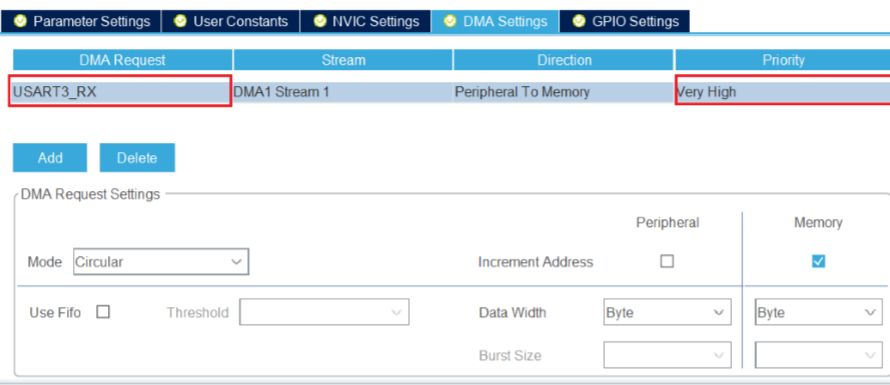
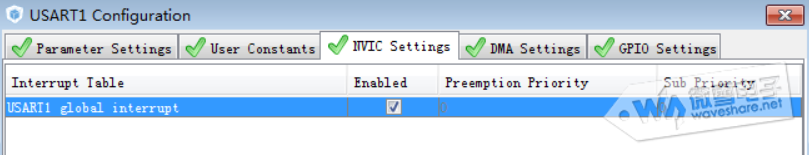
# 串口中断

1. RM赛场上的机器人往往需要处理多种信号的输入，比如遥控器信号，视觉 PC 发送的信号， 各种传感器信号等等，此外还需要进行多种信号的输出，比如控制电机的 CAN 信号，控制舵机的 PWM等等，那么 STM32 是如何有序安排这些任务的呢？就是依赖于中断构成的前 后台机制。
2. 在 STM32 中，对信号的处理可以分为轮询方式和中断方式，轮询方式就是不断去访问一个信号的端口，看看有没有信号进入，有则进行处理，中断方式则是当输入产生的时候，产生一个触发信号告诉 STM32 有输入信号进入，需要进行处理。
3. 例如厨房里烧着开水，主人在客厅里看电视。为了防止开水烧干，他有两种方式，第一种是每隔 10 分钟就去厨房看一眼，另一种是等水壶烧开了之后开始发出响声再去处理。前者是轮询的方式，后者是中断的方式。 每一种中断都有对应的中断函数，当中断发生时，程序会自动跳转到处理函数处运行，而不需要人为进行调用。
4. 使用轮询方法接受串口消息
   1. HAL\_UART\_Receive( UART\_HandleTypeDef \*huart, uint8\_t \*pData, uint16\_t Size, uint32\_t Timeout)
   2. 例如CPU不断查询串口是否传输完成，如传输超过则返回超时错误。轮询方式会占用CPU处理时间，效率较低
5. 使用中断方法接受串口消息
   1. 在函数中要接受的地方添加中断接收模式
   2. 
   3. 当要求接收的数据（这里是10个字节）完成后进入中断
   4. 
   5. 在回调函数中实现其他代码
   6. 注意：如果要连续接收，需要在接收中断回调函数中再次开启接收中断

# DMA的概念

1. DMA 全称为 Direct Memory Access（直接存储器访问），当需要将外部设备发来的数据存 储在存储器中时，如果不使用 DMA 方式则首先需要将外部设备数据先读入 CPU 中，再由CPU 将数据存储到存储器中，如果数据量很大的话，那么将会占用大量的 CPU 时间，而通 过使用 DMA 控制器直接将外部设备数据送入存储器，不需要占用 CPU。 STM32 中的许多通讯如 USART，SPI，IIC 都支持 DMA 方式进行数据的收发。

# CUBEMX的配置

1. 首先查阅开发板手册，找到板载DBUS接口和内部的串口3对应
2. 在 Connectivity 标签页下将 USART3 打开,将其 Mode 设置为 Asynchronous 异步通讯 方式。
3. 将其波特率设置为 100000，数据帧设置为9位数据位（实测8位有错误），单校验位，1 位停止位
4. 
5. 接着开启USART3 的 DMA 功能，在 USART3 下找到 DMA Settings 标签呀，在 USART3 中将 DMA Request 选 为 USART3\_RX，数据从外设流向存储器，Priority 选为 Very High
6. 
7. 如果Mode是Circular，则在回调函数（和串口中断的回调函数是同一个）中不需要再次开启HAL\_UART\_Receive\_DMA(&huart3,sbus\_rx\_buffer,18);
8. DMA的默认中断使能已经开启，实测可以不用再次开启串口中断，如果不用DMA，使用串口接受中断，那么需要在NVIC（中断向量控制器中开启对应的中断）
9. 

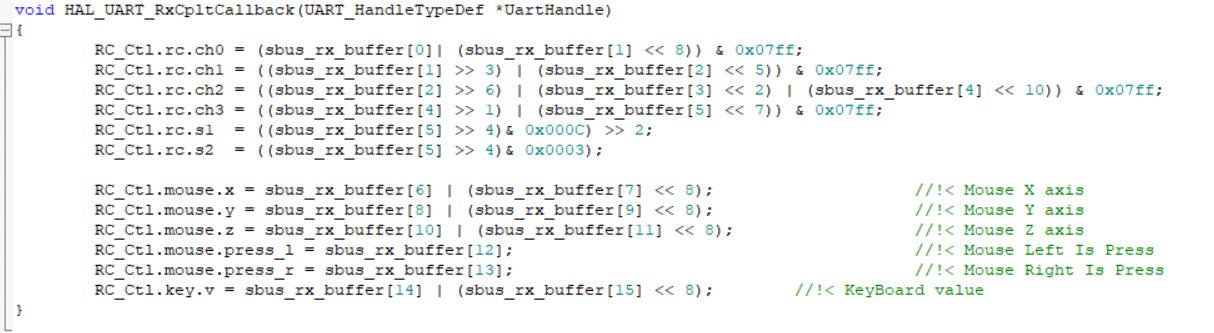
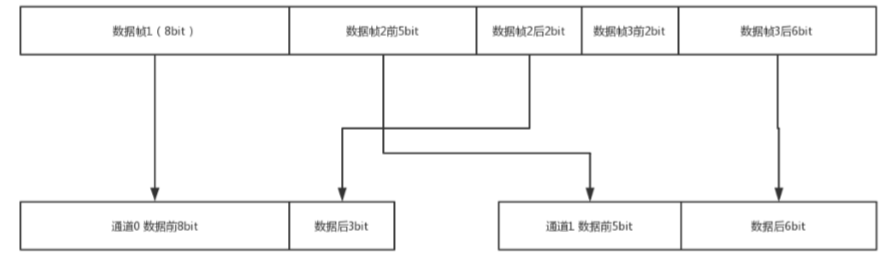
# HAL函数串口DMA接受使用方法

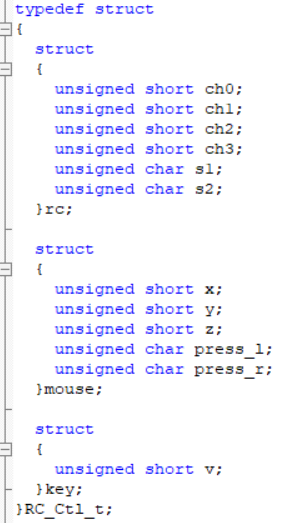
1. 在初始化的时候使用串口DMA接受中断函数带入缓存的地址和接受数据的长度
2. HAL\_UART\_Receive\_DMA(&huart3,sbus\_rx\_buffer,18);
3. 在mian函数的外面定义一个串口接受中断函数
4. void HAL\_UART\_RxCpltCallback(UART\_HandleTypeDef \*UartHandle)

{ 。。。 }

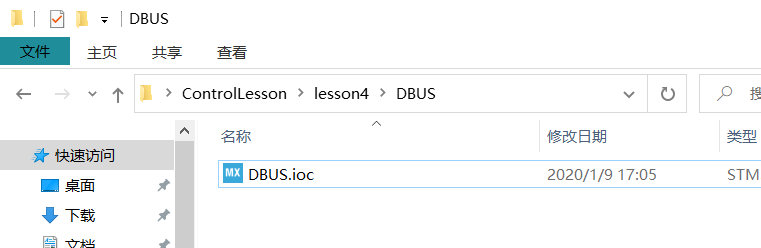
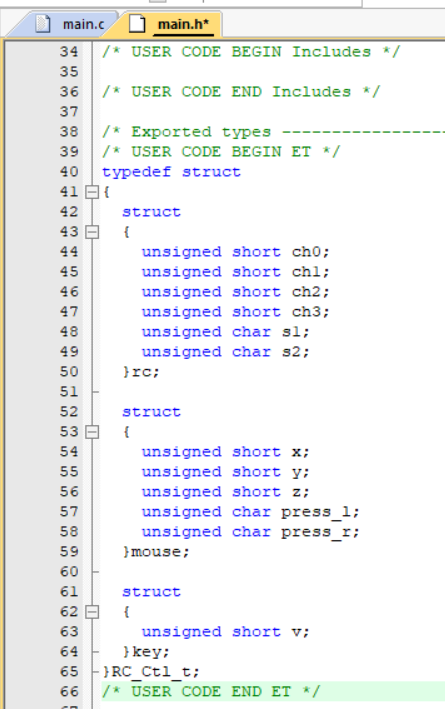
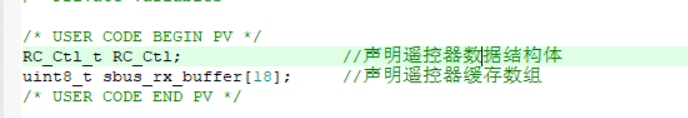
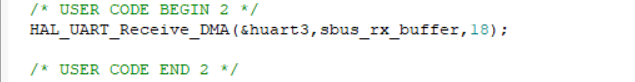
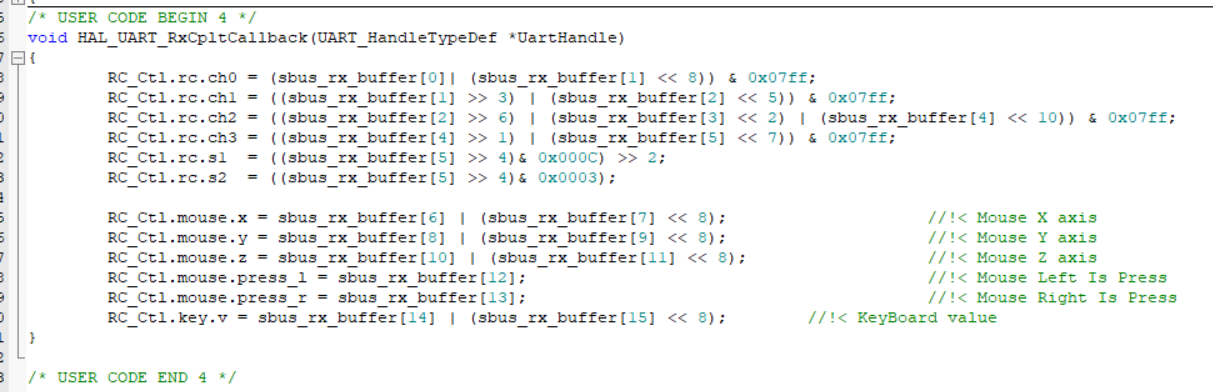
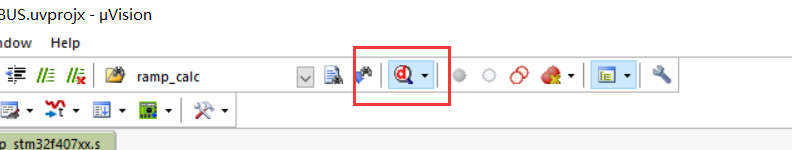
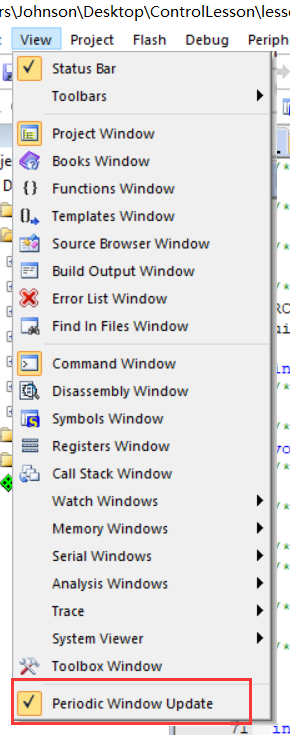
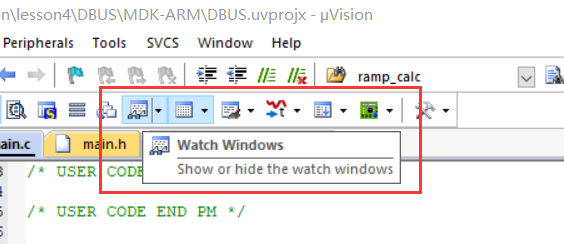
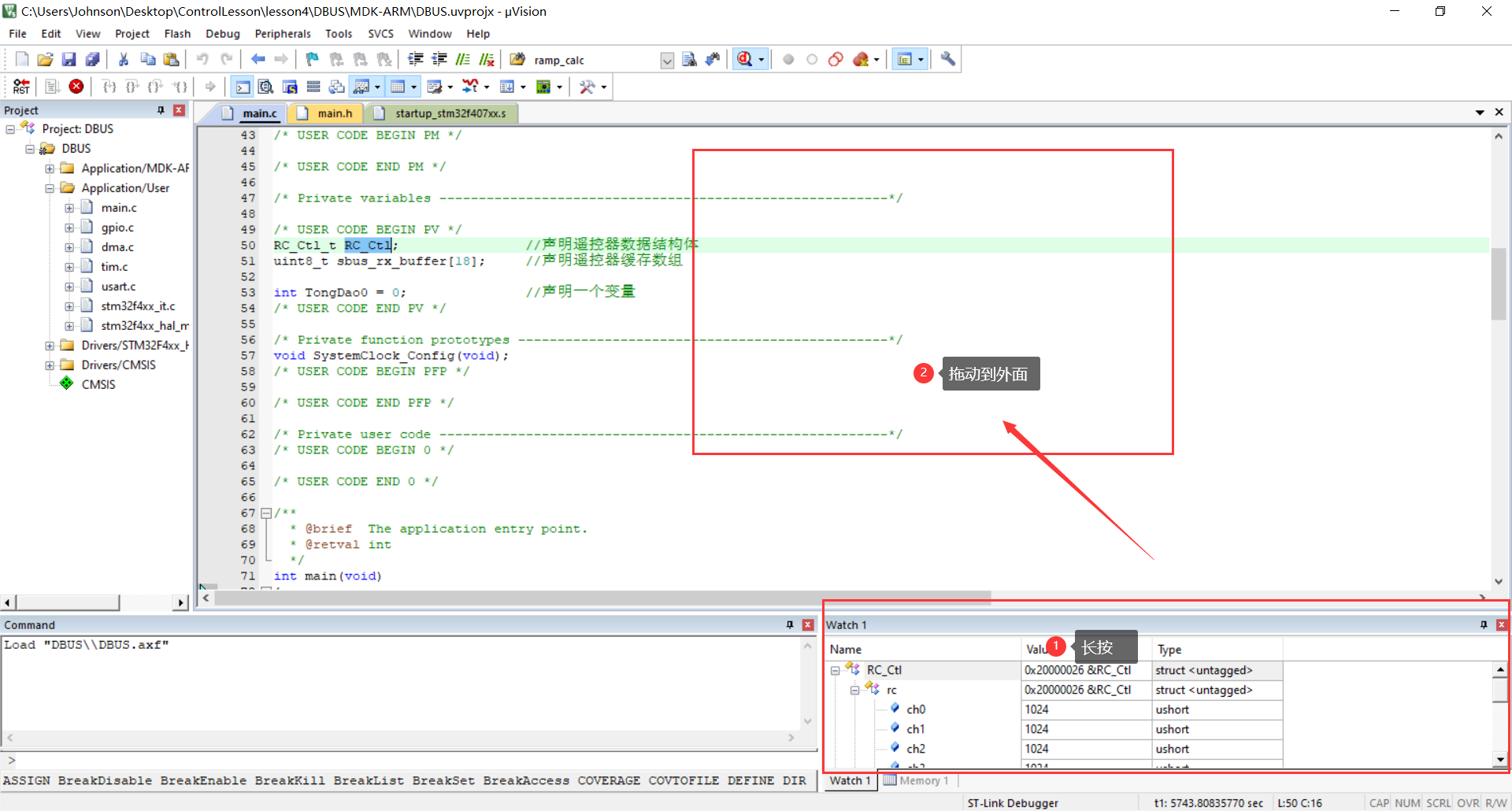
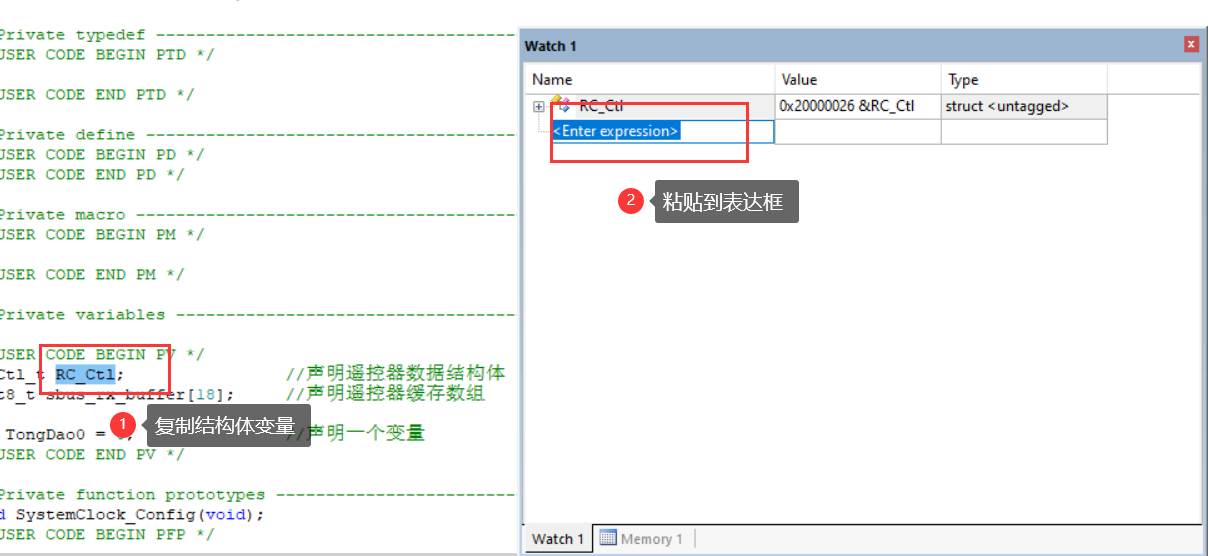
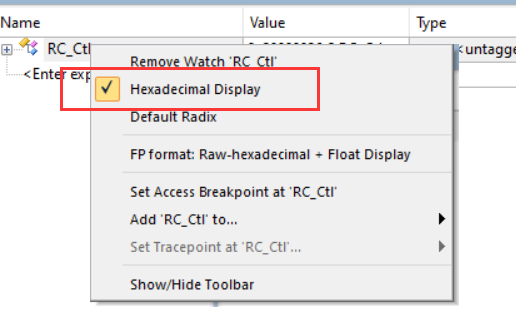
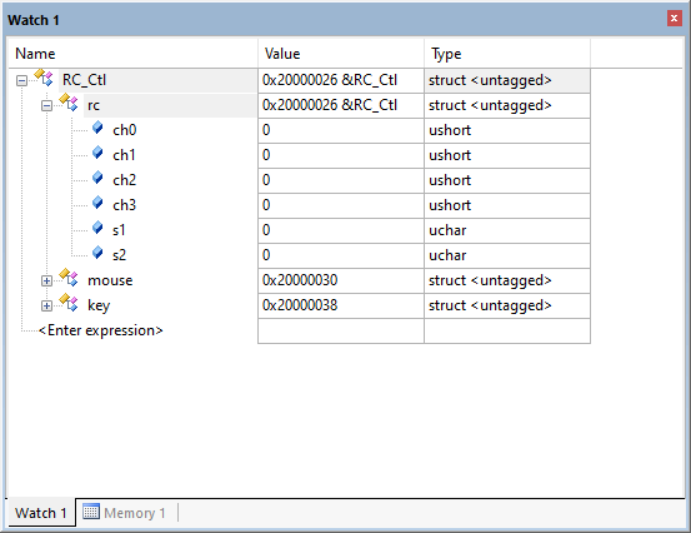
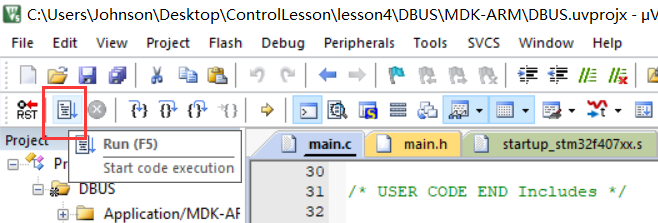
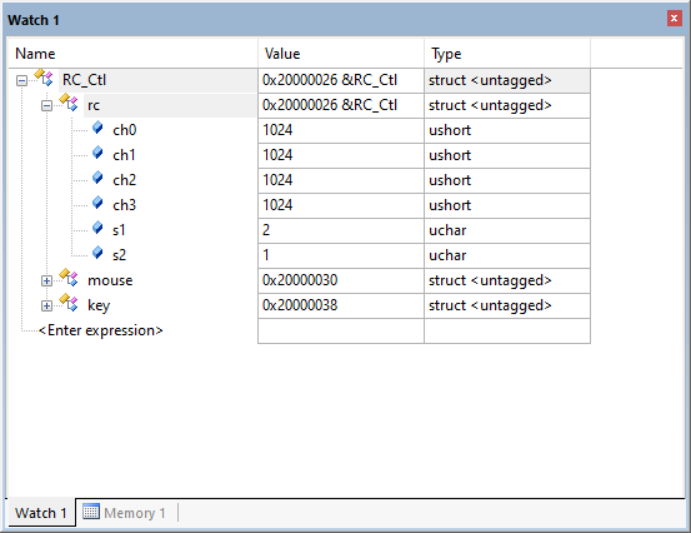
1. 只有当串口接收到了18个字符，才会进入这个中断函数
2. 思考：如果有一次数据被干扰，遥控器的18字节数据只有17个被串口接受，那么在下一次遥控器发送数据时，stm32会把下一次的第一个字节和上一次的17字节拼接在一次保存在缓存里，导致之后的数据全部接收错误

# DBUS数据解析

1. 
2. 在串口中断回调函数中，将缓存中的数据按照遥控器的数据协议拼接成完整的遥控器数据，以通道0的数据为例，从遥控器的手册中查到0的长度为11bit，偏移为0
3. 
4. 这说明如果想要获取通道 0 的数据就需要将第一帧的8bit 数据和第二帧数据的后三bit 数据 拼接，如果想要获取通道 1 的数据就将第二帧数据的前 5bit 和第三帧数据的后 6bit 数据进 行拼接，不断通过拼接就可以获得所有数据帧，拼接过程的示意图如下：
5. 
6. 解码函数通过位运算的方式完成上述的数据拼接工作，十六进制数 0x07ff 的二进制是 0b0000 0111 1111 1111，也就是 11 位的 1，和 0x07ff 进行与运算相当于截取出 11 位的数据。
7. 通道 0 的数据获取：首先将数据帧 1 和左移 8 位的数据帧 2 进行或运算，拼接出 16 位的数据，前 8 位为数据帧 2，后 8 位为数据帧 1，再将其和 0x07ff 相与，截取 11 位，就获得了 由数据帧 2 后 3 位和数据帧 1 拼接成的通道 0 数据。其过程示意图如下：
8. 
9. 同样的方法可以将所有通道的数据获取到结构体中



# 实验1：接受遥控器数据后解析，使用DeBug工具实时显示

1. 创建工程文件，可以将课程3的cubemx工程文件单独复制一份到新的文件夹
2. 
3. 配置好串口后生成代码
4. 在main.h中添加串口数据的结构体
5. 
6. 
7. 在main.c中声明相应的结构体和缓存
8. 
9. 在初始化的时候调用串口DMA中断接受函数，传入定义的缓存和遥控器的数据长度
10. 
11. 在mian函数外面定义中断回调函数，并且输入遥控器解析函数，将缓存中的数据保持到结构体上
12. 编译无误后点击Debug按钮
13. 
14. 
15. 在View菜单下选择变量实时串口更新
16. 
17. 打开Watch Windows变量观察串口
18. 
19. 将串口从底部移出到外面，方面观察
20. 
21. 将需要观察的变量粘贴到表达框中
22. 
23. 右键取消勾选十六进制显示，为十进制显示
24. 
25. 点击+号查看内部成员
26. 
27. 点击Run，开始运行程序
28. 下面是使用KEIL的调试工具显示实时数据
29. 

# 实验2（扩展）

结合课程3，（1）使用遥控器控制舵机的角度，（2）使用遥控器控制舵机速度

思路：

（1）遥控器通道0的输入范围是364~1684，舵机输出的脉宽范围是500~2500，可以使用比例函数，将364~1684均匀映射到500~2500，然后再通过PWM发送值舵机的角度，然后再通过PWM发送值舵机

（2）遥控器通道0的输入范围是364~1684，舵机输出的速度为＋，可以当作顺时针旋转，输出速度为-，当作逆时针旋转，将364~1684减去中值1024，得到-660和+660，通过课程3的斜波函数插值后发送给舵机，注意斜波函数里面需要修改，不然不能实现负速度的功能