6. SBUS model aircraft remote control

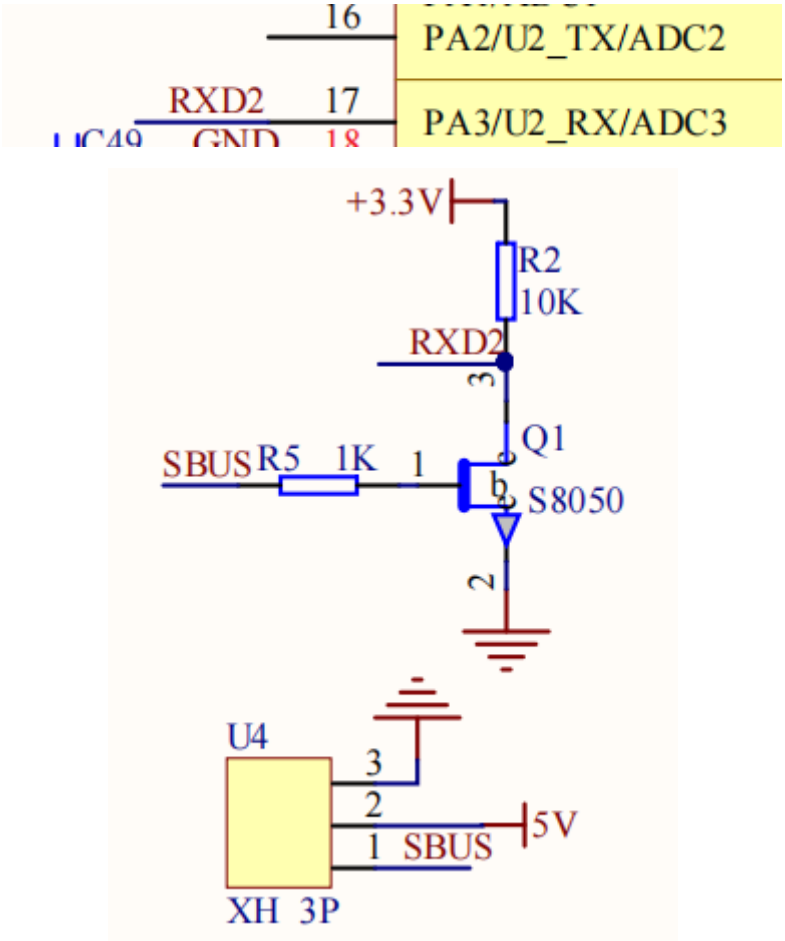
6.1. Purpose of the experiment

Use the serial communication of STM32 to analyze the SBUS protocol data transmitted by the remote control transmitter of the model aircraft, and print the value of each channel.

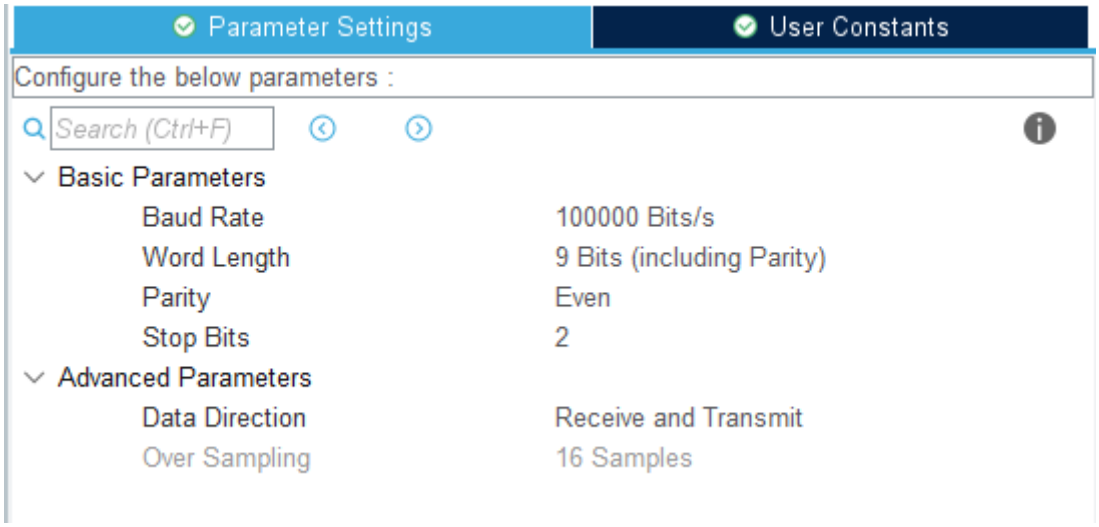
6.2. Configuration pin information

1. Import the ioc file from the Serial project and name it SBUS.

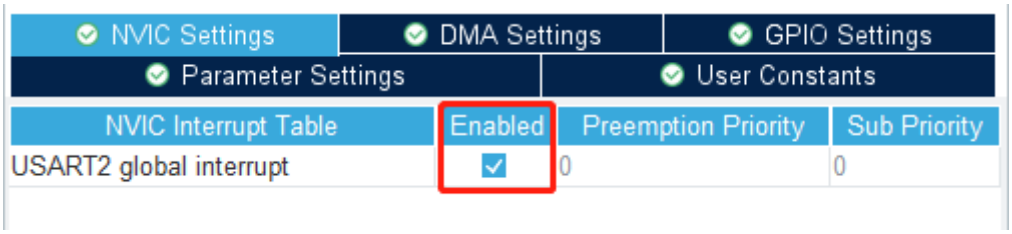
According to the schematic diagram, SBUS is connected to the RX pin of serial port 2, only receiving but not sending.



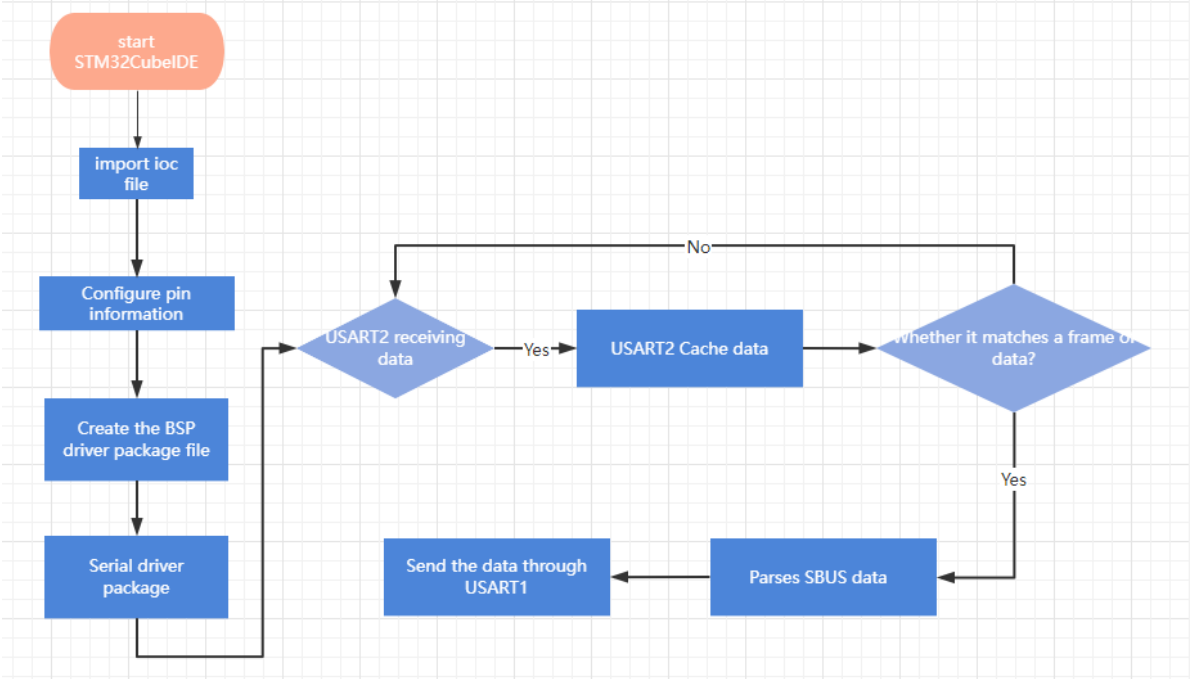
2. Change the mode of serial port 2 to Asynchronous synchronous communication, the baud rate is 100000, the data width: 9 bits, the test: Even, the stop bit: 2 bits. Serial port 2 only uses the receive function, so Data Direction can choose Receive and Transmit or Receive Only.



3. Open the serial port 2 interrupt settings.



6.3. Analysis of the experimental flow chart



6.4. core code explanation

1. Add the following in bsp_uart.c:

USART1_Init(): Initialize the serial port related content, open serial port 1 and serial port 2 to receive 1 data.

```c
// Initialize USART1  初始化串口1
void USART1_Init(void)
{
    HAL_UART_Receive_IT(&huart1, (uint8_t *)&RxTemp, 1);
    HAL_UART_Receive_IT(&huart2, (uint8_t *)&RxTemp_2, 1);

    printf("start serial\n");
}
```

2. In the serial port interrupt callback, judge whether serial port 2 data is received, and at the same time distinguish whether serial port 1 or serial port 2 has received the data.

```c
// The serial port receiving is interrupted. Procedure  串口接收完成中断
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{

    if (huart == &huart1)
    {
        // 测试发送数据，实际应用中不应该在中断中发送数据
        // Test sending data. In practice, data should not be sent during interrupts
        USART1_Send_U8(RxTemp);
        // Continue receiving data  继续接收数据
        HAL_UART_Receive_IT(&huart1, (uint8_t *)&RxTemp, 1);
    }
    if (huart == &huart2)
    {
        SBUS_Reveive(RxTemp_2);
        // printf("a:%d\n", RxTemp_2);
        HAL_UART_Receive_IT(&huart2, (uint8_t *)&RxTemp_2, 1);
    }
}
```

3. Create new bsp_sbus.h and bsp_sbus.c files to manage sbus data analysis content. Create the following in bsp_sbus.h:

```c
#define SBUS_SIGNAL_OK          0x00
#define SBUS_SIGNAL_LOST        0x01
#define SBUS_SIGNAL_FAILSAFE    0x03
#define SBUS_ALL_CHANNELS       0x00


void SBUS_Reveive(uint8_t data);
void SBUS_Handle(void);
```

Among them, SBUS_ALL_CHANNELS controls the number of channels parsed. By default, only eight channels are displayed. If full channel display is required, modify it to 1.

4. SBUS_Reveive(data) receives the data of the serial port as a buffer. If it conforms to the communication protocol of SBUS, it will update a frame of data to the sbus_data array.

```
// Receives SBUS cache data  接收SBUS的缓存数据
void SBUS_Reveive(uint8_t data)
{
    // If the protocol start flag is met, data is received  如果符合协议开始标志，则开始接收数据
    if (sbus_start == 0 && data == SBUS_START)
    {
        sbus_start = 1;
        sbus_new_cmd = 0;
        sbus_buf_index = 0;
        inBuffer[sbus_buf_index] = data;
        inBuffer[SBUS_RECV_MAX - 1] = 0xff;
    }
    else if (sbus_start)
    {
        sbus_buf_index++;
        inBuffer[sbus_buf_index] = data;
    }

    // Finish receiving a frame of data  完成接收一帧数据
    if (sbus_start & (sbus_buf_index >= (SBUS_RECV_MAX - 1)))
    {
        sbus_start = 0;
        if (inBuffer[SBUS_RECV_MAX - 1] == SBUS_END)
        {
            memcpy(sbus_data, inBuffer, SBUS_RECV_MAX);
            sbus_new_cmd = 1;
        }
    }
}
```

5. Analyze the data in sbus_data according to the SBUS communication protocol.

```
// Parses SBUS data into channel values  解析SBUS的数据，转化成通道数值。
static int SBUS_Parse_Data(void)
{
    g_sbus_channels[0]  = ((sbus_data[1] | sbus_data[2] << 8) & 0x07FF);
    g_sbus_channels[1]  = ((sbus_data[2] >> 3 | sbus_data[3] << 5) & 0x07FF);
    g_sbus_channels[2]  = ((sbus_data[3] >> 6 | sbus_data[4] << 2 | sbus_data[5] << 10) & 0x07FF);
    g_sbus_channels[3]  = ((sbus_data[5] >> 1 | sbus_data[6] << 7) & 0x07FF);
    g_sbus_channels[4]  = ((sbus_data[6] >> 4 | sbus_data[7] << 4) & 0x07FF);
    g_sbus_channels[5]  = ((sbus_data[7] >> 7 | sbus_data[8] << 1 | sbus_data[9] << 9) & 0x07FF);
    g_sbus_channels[6]  = ((sbus_data[9] >> 2 | sbus_data[10] << 6) & 0x07FF);
    g_sbus_channels[7]  = ((sbus_data[10] >> 5 | sbus_data[11] << 3) & 0x07FF);
    #ifdef ALL_CHANNELS
    g_sbus_channels[8]  = ((sbus_data[12] | sbus_data[13] << 8) & 0x07FF);
    g_sbus_channels[9]  = ((sbus_data[13] >> 3 | sbus_data[14] << 5) & 0x07FF);
    g_sbus_channels[10] = ((sbus_data[14] >> 6 | sbus_data[15] << 2 | sbus_data[16] << 10) & 0x07FF);
    g_sbus_channels[11] = ((sbus_data[16] >> 1 | sbus_data[17] << 7) & 0x07FF);
    g_sbus_channels[12] = ((sbus_data[17] >> 4 | sbus_data[18] << 4) & 0x07FF);
    g_sbus_channels[13] = ((sbus_data[18] >> 7 | sbus_data[19] << 1 | sbus_data[20] << 9) & 0x07FF);
    g_sbus_channels[14] = ((sbus_data[20] >> 2 | sbus_data[21] << 6) & 0x07FF);
    g_sbus_channels[15] = ((sbus_data[21] >> 5 | sbus_data[22] << 3) & 0x07FF);
    #endif

    // 安全检测，检测是否失联或者数据错误
    // Security detection to check for lost connections or data errors
    failsafe_status = SBUS_SIGNAL_OK;
    if (sbus_data[23] & (1 << 2))
    {
        failsafe_status = SBUS_SIGNAL_LOST;
        printf("SBUS_SIGNAL_LOST\n");
        // lost contact errors  遥控器失联错误
    }
    else if (sbus_data[23] & (1 << 3))
    {
        failsafe_status = SBUS_SIGNAL_FAILSAFE;
        printf("SBUS_SIGNAL_FAILSAFE\n");
        // data loss error  数据丢失错误
    }
    return failsafe_status;
}
```

6. The SBUS_Handle() function is called cyclically in Bsp_Loop(), and the parsed data of each channel is printed out through serial port 1.

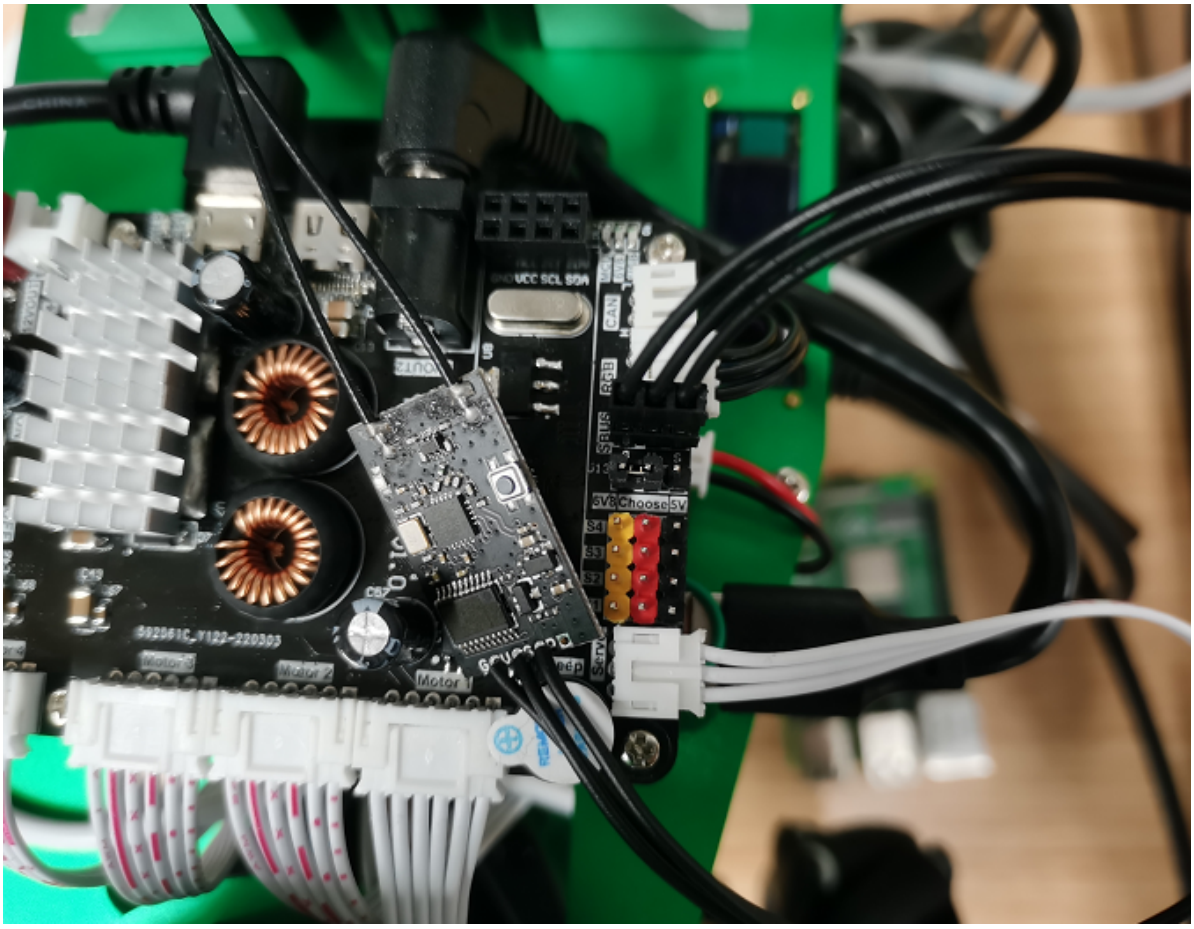
```c
// SBUS receives and processes data handle  SBUS接收处理数据句柄
void SBUS_Handle(void)
{
    if (sbus_new_cmd)
    {
        int res = SBUS_Parse_Data();
        sbus_new_cmd = 0;
        if (res) return;
        #if SBUS_ALL_CHANNELS
        printf("%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d\r\n",
               g_sbus_channels[0], g_sbus_channels[1], g_sbus_channels[2],
               g_sbus_channels[3], g_sbus_channels[4], g_sbus_channels[5],
               g_sbus_channels[6], g_sbus_channels[7], g_sbus_channels[8],
               g_sbus_channels[9], g_sbus_channels[10], g_sbus_channels[11],
               g_sbus_channels[12], g_sbus_channels[13], g_sbus_channels[14],
               g_sbus_channels[15]);
        #else
        printf("%d,%d,%d,%d,%d,%d,%d,%d\r\n",
               g_sbus_channels[0], g_sbus_channels[1], g_sbus_channels[2],
               g_sbus_channels[3], g_sbus_channels[4],g_sbus_channels[5],
               g_sbus_channels[6], g_sbus_channels[7]);
        #endif
    }
}
```

```c
// main.c中循环调用此函数，避免多次修改main.c文件。
// This function is called in a loop in main.c to avoid multiple modifications to the main.c file
void Bsp_Loop(void)
{
    // Detect button down events   检测按键按下事件
    if (Keyl_State(KEY_MODE_ONE_TIME))
    {
        Beep_On_Time(50);
        static int press = 0;
        press++;
        printf("press:%d\n", press);
    }
    SBUS_Handle();

    Bsp_Led_Show_State_Handle();
    // The buzzer automatically shuts down when times out   蜂鸣器超时自动关闭
    Beep_Timeout_Close_Handle();
    HAL_Delay(10);
}
```

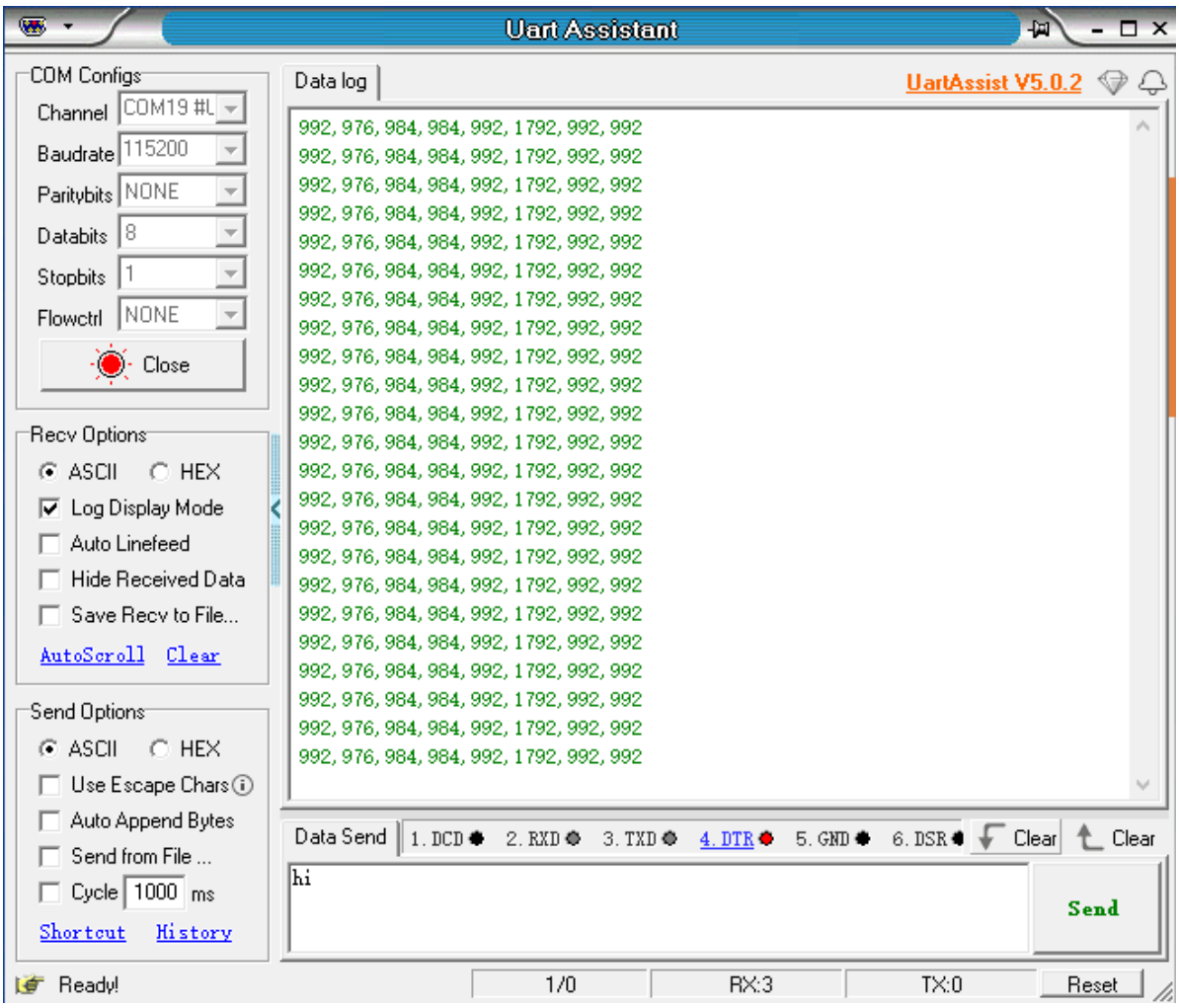
6.5. Hardware connection

Because SBUS communication needs to connect the SBUS receiver to the SBUS interface on the expansion board, S is connected to the signal, V is connected to the positive pole of the power supply, and G is connected to the ground. Therefore, you need to prepare your own model aircraft remote control and SBUS receiver, pair them in advance and turn on the power switch.



6.6. Experimental effect

After programming the program, the LED light flashes every 200 milliseconds. After connecting the expansion board to the computer through the micro-USB data cable, open the serial port assistant (the specific parameters are shown in the figure below), and you can see that the model aircraft remote control has been printed on the serial port assistant. The data of each channel of the controller, when we manually toggle the joystick or button of the model aircraft remote controller, the data will change accordingly.


Uart Assistant
COM Configs
Channel COM19 #L
Baudrate 115200
Paritybits NONE
Databits 8
Stopbits 1
Flowctrl NONE
Close
Recv Options
ASCII HEX
Log Display Mode
Auto Linefeed
Hide Received Data
Save Recv to File...
AutoScroll Clear
Send Options
ASCII HEX
Use Escape Chars
Auto Append Bytes
Send from File ...
Cycle 1000 ms
Shortcut History
Data log
UartAssist V5.0.2
992, 976, 984, 984, 992, 1792, 992, 992
Data Send
1. DCD 2. RXD 3. TXD 4. DTR 5. GND 6. DSR
Clear
Clear
hi
Send
Ready!
1/0
RX:3
TX:0
Reset