# 星闪NearLink

## 星闪基础

### 星闪简介

星闪（NearLink），中国原生的新一代无线短距通信技术。面向万物互联时代，星闪引入关键技术和创新理念，赋予智能终端新的连接方式。与传统短距传输技术方案相比，星闪在功耗、速度、覆盖范围和连接性能全面领先，可以在智能终端、智能家居、智能汽车、智能制造等各类细分场景下实现更极致的用户体验。

**星闪Near Link特点**：

1. 低时延：

星闪传输时延是传统无线技术的1/30，同等时间提供30倍的交互信息，由毫秒级迈进微秒级。

星闪技术加持下，无线鼠标刷新率由传统的125-1000Hz提升至4000Hz、手写笔刷新率由传统的120Hz提升至360Hz，丝般流畅，如影随形，极大提高使用体验。

1. 高吞吐：

星闪在设备间信息传输最高速率是传统无线技术（2Mbps）的6倍（12Mbps），能耗只有之前的60%，不仅工作效率提高，质量还大幅提升。

星闪高速率完美支持在无线耳机上的无损音频传输，让录音室级品质音频体验成为现实。

1. 高并发：

与传统技术直连设备只可连接数个相比，星闪最大可达到百量级，避免了传统技术组网能力弱、同步实现难度高导致的跑马灯现象，深度赋能智能家居IOT等领域。

1. 高可靠：

首次将“Polar码”等前沿技术应用于短距无线通信，信息传输可靠性由传统技术的90%提升至99.99%。

1. 抗干扰：

星闪首次将5G Polar码技术用于短距通信，配合干扰避让机制，抗干扰能力比传统无线技术提升7dB。

无论身处人流密集的机场、还是Wi-Fi遍布的商场里，超强的抗干扰能力能让无线音频体验不打折；不再担心周边干扰影响外设稳定性，电竞游戏体验升级。

1. 精定位：

星闪将定位精度由传统无线技术的米级提升到分米级，依托领先的测距算法，有效克服人体遮挡、环境吸收和反射等因素叠加，解决测距结果不稳定、反复解闭锁的痛点。

### 试验所用星闪模块默认参数

#### 串口参数

AT 指令端口：Enhanced com port 波特率：115200bps

烧录端口：Standard com port

最高波特率：921600bps

协议栈极限参数

最大连接数：4 个（连接情况=n\*G+m\*T，n+m=4）

MTU：300

广播参数

广播间隔：100ms

广播类型：可连接可扫描

广播数据：01:01:01:02:01:02

扫描响应：0B:0A:Lierda\_SLE

扫描参数

扫描间隔：12.5ms

占空比：50%

不过滤重复设备

设备名称

Lierda\_SLE

连接参数

链路调度最小间隔：100ms

链路调度最大间隔：110ms

延迟周期：1

超时时间：4 秒

## 实验：Nearlink

### 实验目的

1、了解星闪NearLink；

2、熟悉星闪NearLink的AT指令；

3、掌握配置和使用CH32V307VCT6的USART外设；

4、利用USART外设发送AT指令进行星闪通信。

### 实验内容

1. 两块CH32V307VCT6评估板使用USART外设发送AT指令进行星闪通信。
2. 在一块CH32V307VCT6评估板USART模块发送AT指令让其星闪作为T节点，另一块CH32V307VCT6评估板USART模块发送AT指令让其星闪作为G节点
3. 星闪NearLink之间相互通信
4. 评估版：3.3V\GND\USART2\_TX(PA2)\USART\_RX(PA3)
5. 星闪NearLink：3.3V\GND\TXD\RXD

### 实验环境和要求

CH32V307VCT6评估板和星闪NearLink

开发环境：MounRiver 1.85

星闪T节点和星闪G节点分别在不同的CH32V307VCT6评估板上。

两块评估版分别下载T节点和G节点，T节点和G节点同时上电。

### 硬件引脚连接

|  |  |
| --- | --- |
| 星闪NearLink模块 | 评估版 |
| 3.3V | 3.3V |
| GND | GND |
| RXD | USART2\_TX(PA2) |
| TXD | USART\_RX(PA3) |

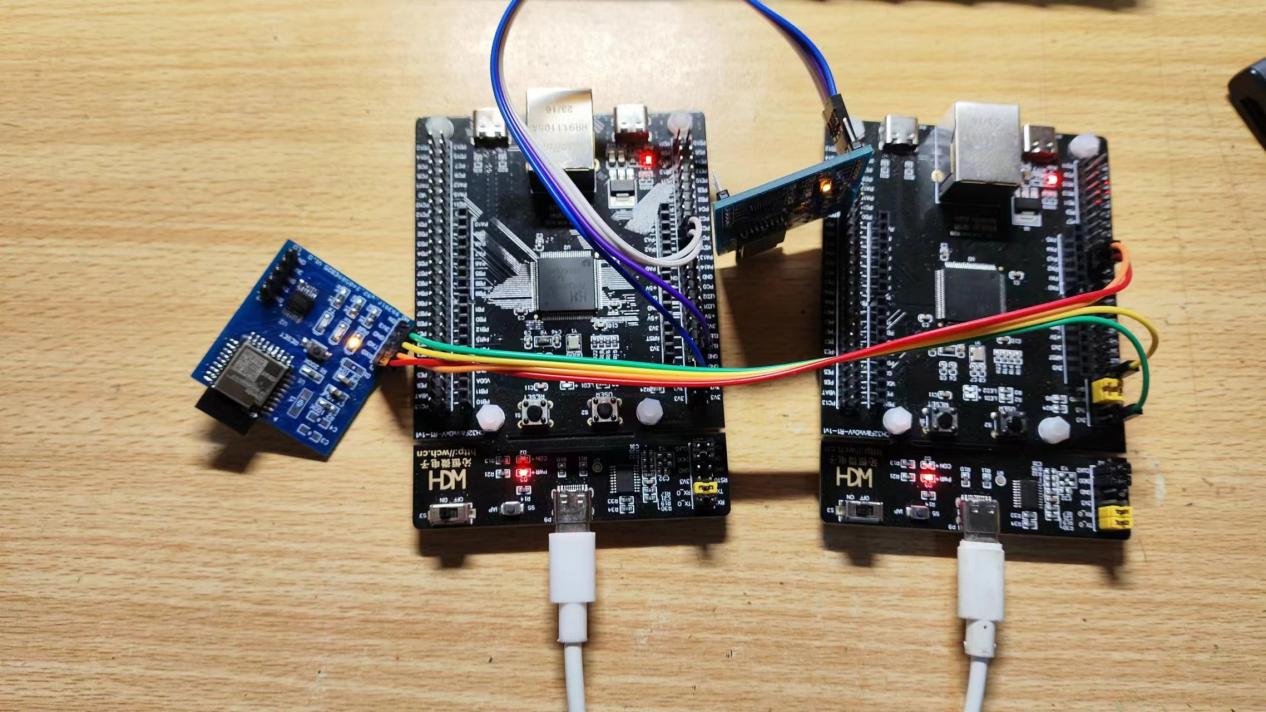


图1-引脚连接

### 实验原理

实验使用USART2收发数据以控制星闪NearLink

* USART2 引脚定义

PA2 引脚作为 USART2 的发送引脚（USART2\_TX），PA3 引脚作为 USART2 的接收引脚（USART2\_RX）。

星闪NearLink的连接并发送数据操作

1. T 节点 通过 AT+ANNOUNCE?指令读取公开状态, 确保公开处于开启状态。如果未开启公开，G 节点将无法扫描此设备并连接，需要发送 AT+ANNOUNCE=1指令。

注意：设置成功后重复设置可能会返回ERROR

1. 节点T 开启扫描，发送AT+SEEK=1。
2. T节点 等待扫描数据事件，事件数据为：+RECEIVEDATA:0,112233445566, xx,xxxx。发现对于G节点的公开信息后通过AT+SEEK=0指令关闭扫描。

（模块地址可使用AT指令“AT+SLEADDR=<type>,<address>\r\n“设置)如：”AT+SLEADDR=0,112233445566\r\n”

1. T节点 通过AT+CONNECT=0,112233445566指令连接T节点。
2. T节点 通过指令令 AT+SENDDATA=0,112233445566,1234发送数据给G节点

### 实验步骤

1. 创建两个项目，一个是星闪节点T项目

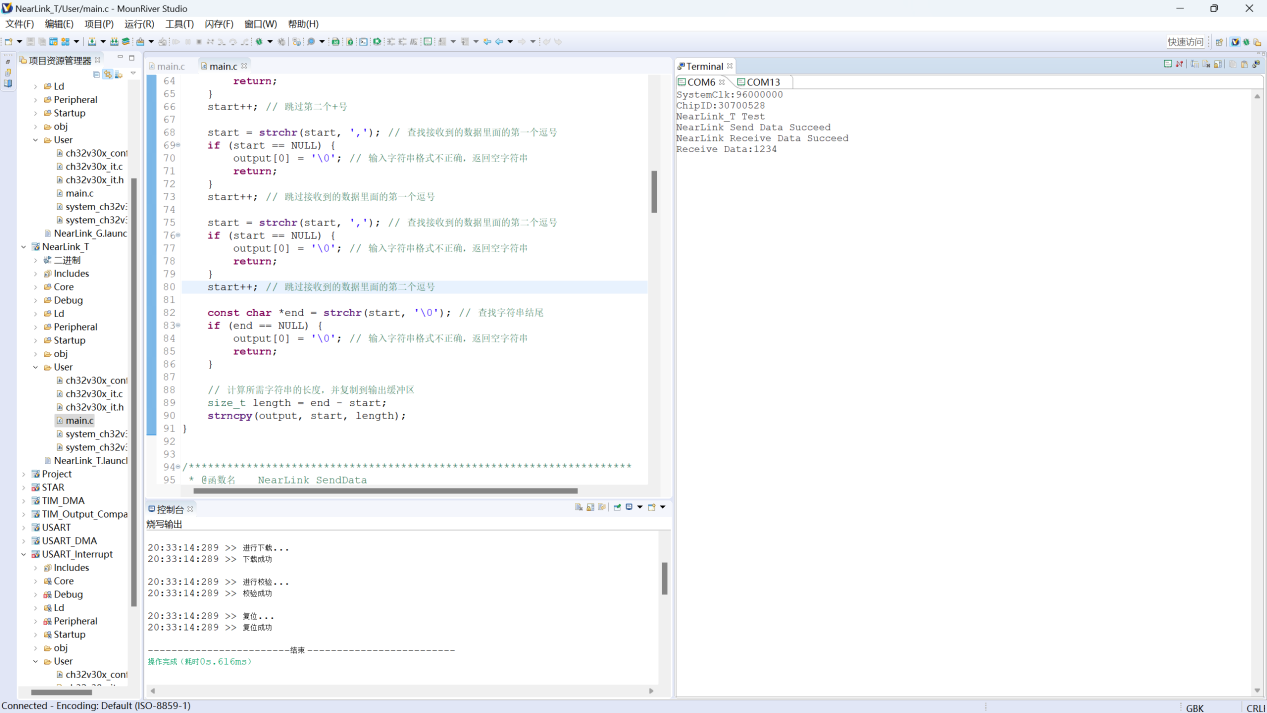


图2-节点T项目

另一个是星闪节点G项目。

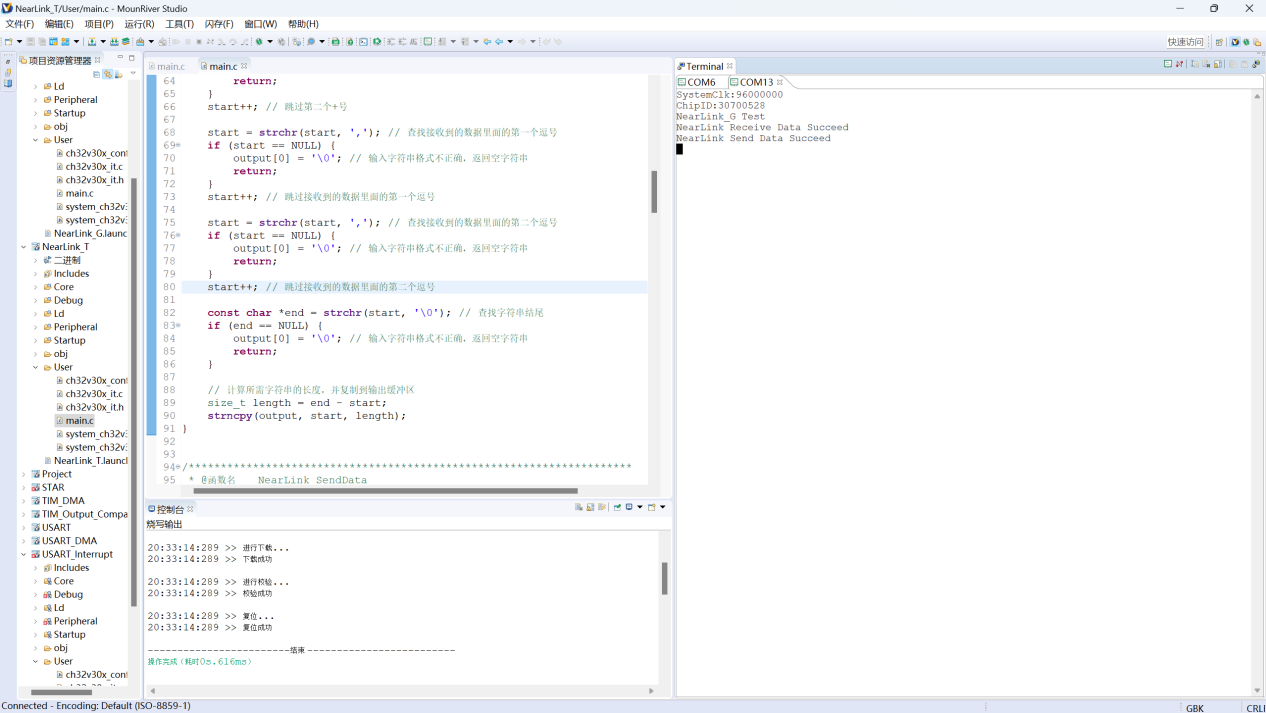


图3-节点G项目

2、配置USART外设的引脚和初始化代码，以便使其可以正常工作。

3、评估版T使用NearLink\_SendData函数发送AT扫描指令，星闪节点T返回扫描结果。评估版处理星闪的返回结果，发送AT发送数据指令。

4、评估版G接收数据并打印数据。

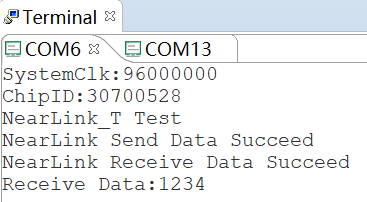
### 实验测试

1、两块板子分别下载NearLink\_T和NearLink\_G程序，两块板子同时上电。

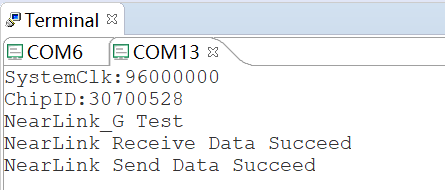
2、在串口监视器中观察评估版T和G接收到的数据。

注意：星闪模块自带超时4s断连，建议开始试验前断开星闪模块供电，烧录程序后打开MountRiver Studio自带串口查看工具，给星闪模块上电后按下评估版RESET键

评估版T的实验运行结果如下图。

图4-节点T运行结果

评估版G的实验运行结果如下图。

图5-节点G运行结果

### 实验总结

通过本实验，使用USART和星闪NearLink实现两块CH32V307VCT6评估板数据通信，其中一块是评估版T，主要连接星闪G和发送数据，另一块是评估版G，主要接收数据。通过该例程我们可以懂得使用AT指令控制星闪模块基本通信的主要过程。

### 实验源码

1. **评估版T连接星闪和发送数据**

**#include "debug.h"**

**#include "string.h"**

**char RxBuffer[8000];**

**//接收缓冲区**

**volatile u8 RxCnt = 0;**

**void USART2\_IRQHandler(void) \_\_attribute\_\_((interrupt("WCH-Interrupt-fast")));**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* @函数名 extract\_address**

**\***

**\* @功能 解包提取星闪地址**

**\***

**\* @参数 input- 源数据**

**\* output - 解包后的数据**

**\***

**\* @返回值 无返回值**

**\*/**

**void extract\_address(const char \*input, char \*output) {**

**const char \*start = strchr(input, ','); // 查找第一个逗号**

**if (start == NULL) {**

**output[0] = '\0'; // 输入字符串格式不正确，返回空字符串**

**return;**

**}**

**start++; // 跳过第一个逗号**

**const char \*end = strchr(start, ','); // 查找第二个逗号**

**if (end == NULL) {**

**output[0] = '\0'; // 输入字符串格式不正确，返回空字符串**

**return;**

**}**

**// 计算所需字符串的长度，并复制到输出缓冲区**

**size\_t length = end - start ;**

**strncpy(output, start, length);**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* @函数名 extract\_data**

**\***

**\* @功能 解包提取数据**

**\***

**\* @参数 input- 源数据**

**\* output - 解包后的数据**

**\***

**\* @返回值 无返回值**

**\*/**

**void extract\_data(const char \*input, char \*output) {**

**const char \*start = strchr(input, ','); // 查找第一个逗号**

**if (start == NULL) {**

**output[0] = '\0'; // 输入字符串格式不正确，返回空字符串**

**return;**

**}**

**start++; // 跳过第一个逗号**

**start = strchr(start, ','); // 查找第二个逗号**

**if (start == NULL) {**

**output[0] = '\0'; // 输入字符串格式不正确，返回空字符串**

**return;**

**}**

**start++; // 跳过第二个逗号**

**const char \*end = strchr(start, '\0'); // 查找字符串结尾**

**if (end == NULL) {**

**output[0] = '\0'; // 输入字符串格式不正确，返回空字符串**

**return;**

**}**

**// 计算所需字符串的长度，并复制到输出缓冲区**

**size\_t length = end - start;**

**strncpy(output, start, length);**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* @函数名 NearLink\_SendData**

**\***

**\* @功能 通过USART发送数据给NearLink星闪**

**\***

**\* @参数 USARTx- 选择USART串口**

**\* Data - 发送的数据**

**\***

**\* @返回值 无返回值**

**\*/**

**void NearLink\_SendData(USART\_TypeDef \*USARTx,const char \*Data)**

**{**

**memset(RxBuffer, 0, sizeof(RxBuffer));**

**RxCnt = 0;**

**while(\*Data)**

**{**

**USART\_SendData(USARTx,\*Data++);**

**while(USART\_GetFlagStatus(USARTx,USART\_FLAG\_TXE)==RESET)**

**{**

**}**

**}**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* @函数名 USARTx\_CFG**

**\***

**\* @功能 初始化USART2外设**

**\***

**\***

**\* @返回值 无返回值**

**\*/**

**void USARTx\_CFG(void)**

**{**

**GPIO\_InitTypeDef GPIO\_InitStructure = {0};**

**//定义GPIO初始化结构体**

**USART\_InitTypeDef USART\_InitStructure = {0};**

**//定义USART初始化结构体**

**NVIC\_InitTypeDef NVIC\_InitStructure = {0};**

**//定义NVIC初始化结构体**

**RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_USART2, ENABLE);**

**//使能USART2的时钟**

**RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE);**

**//使能GPIOA的时钟**

**GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_2;**

**//配置PA2引脚**

**GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;**

**//设置频率**

**GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP;**

**//复用推挽输出模式**

**GPIO\_Init(GPIOA, &GPIO\_InitStructure);**

**//初始化GPIOA的PA2引脚**

**GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_3;**

**//配置PA3引脚**

**GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;**

**//悬空输入模式**

**GPIO\_Init(GPIOA, &GPIO\_InitStructure);**

**//初始化GPIOA的PA3引脚**

**//配置USART参数**

**USART\_InitStructure.USART\_BaudRate = 115200;**

**//设置波特率**

**USART\_InitStructure.USART\_WordLength = USART\_WordLength\_8b;**

**//配置每个数据帧包含8位数据**

**USART\_InitStructure.USART\_StopBits = USART\_StopBits\_1;**

**//配置停止位数**

**USART\_InitStructure.USART\_Parity = USART\_Parity\_No;**

**//配置不使用奇偶校验**

**USART\_InitStructure.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;**

**//配置不使用硬件流控**

**USART\_InitStructure.USART\_Mode = USART\_Mode\_Tx | USART\_Mode\_Rx;**

**//启用发送接收模式**

**USART\_Init(USART2, &USART\_InitStructure);**

**//初始化USART2**

**USART\_ITConfig(USART2, USART\_IT\_RXNE, ENABLE);**

**//启用 USART2的接收数据寄存器非空中断**

**NVIC\_InitStructure.NVIC\_IRQChannel = USART2\_IRQn;**

**//配置 NVIC 为 USART2的中断通道**

**NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 1;**

**//配置 USART2中断的抢占优先级为 1**

**NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 1;**

**//配置 USART2中断的子优先级为 1**

**NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;**

**//使能 USART2中断**

**NVIC\_Init(&NVIC\_InitStructure);**

**//初始化NVIC**

**USART\_Cmd(USART2, ENABLE);**

**//使能USART2**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* 函数名 main**

**\***

**\* 函数功能 主函数**

**\***

**\* 返回值 无返回值**

**\*/**

**int main(void)**

**{**

**NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);**

**//配置NVIC的中断优先级分组**

**SystemCoreClockUpdate();**

**//更新系统核心时钟频率**

**Delay\_Init();**

**//延迟函数初始化**

**USART\_Printf\_Init(115200);**

**//USART初始化并配置波特率为115200**

**//打印系统信息**

**printf("SystemClk:%d\r\n", SystemCoreClock);**

**//打印系统时钟频率**

**printf( "ChipID:%08x\r\n", DBGMCU\_GetCHIPID() );**

**//打印芯片信息**

**printf("NearLink\_G Test\r\n");**

**//打印字符串“USART Interrupt TEST”**

**USARTx\_CFG();**

**//对 USART2外设进行初始化。**

**//STEP1:发送AT等待星闪初始化**

**NearLink\_SendData(USART2, "AT\r\n");**

**Delay\_Ms(100);**

**//STEP2:等待OK**

**while(strstr((char \*)RxBuffer, "OK") == NULL){}**

**//STEP3:接收数据**

**while(USART\_GetFlagStatus(USART2, USART\_FLAG\_RXNE) != SET){}**

**Delay\_Ms(100);**

**while(strstr((char \*)RxBuffer, "RECEIVEDATA") == NULL){}**

**printf("NearLink Receive Data Succeed\r\n");**

**//STEP4:处理数据**

**char Data[128] = {0};**

**char Address[128] = {0};**

**extract\_data(RxBuffer, Data);**

**extract\_address(RxBuffer, Address);**

**//STEP4:转发数据**

**char Nearlink\_Send\_Data[128]={0};**

**sprintf(Nearlink\_Send\_Data,"AT+SENDDATA=0,%s,%s\r\n",Address,Data);**

**NearLink\_SendData(USART2,Nearlink\_Send\_Data);**

**Delay\_Ms(100);**

**while(strstr((char \*)RxBuffer, "OK") == NULL){}**

**printf("NearLink Send Data Succeed\r\n");**

**while(1)**

**{**

**}**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* 函数名 USART2\_IRQHandler**

**\***

**\* 函数功能 此函数处理 USART2 全局中断请求。**

**\***

**\* 返回值 无返回值**

**\*/**

**void USART2\_IRQHandler(void)**

**{**

**if(USART\_GetFlagStatus(USART2, USART\_FLAG\_RXNE) != RESET)**

**//判断 USART2 接收中断是否发生**

**{**

**RxBuffer[RxCnt++] = USART\_ReceiveData(USART2);**

**}**

**}**

1. **评估版G接收数据和转发数据**

#include "debug.h"

#include "string.h"

char RxBuffer[8000]={};

//接收缓冲区

volatile uint8\_t RxCnt = 0;

void USART2\_IRQHandler(void) \_\_attribute\_\_((interrupt("WCH-Interrupt-fast")));

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @函数名 extract\_address

\*

\* @功能 解包提取星闪地址

\*

\* @参数 input- 源数据

\* output - 解包后的数据

\*

\* @返回值 无返回值

\*/

void extract\_address(const char \*input, char \*output) {

const char \*start = strchr(input, ','); // 查找第一个逗号

if (start == NULL) {

output[0] = '\0'; // 输入字符串格式不正确，返回空字符串

return;

}

start++; // 跳过第一个逗号

const char \*end = strchr(start, ','); // 查找第二个逗号

if (end == NULL) {

output[0] = '\0'; // 输入字符串格式不正确，返回空字符串

return;

}

// 计算所需字符串的长度，并复制到输出缓冲区

size\_t length = end - start ;

strncpy(output, start, length);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @函数名 extract\_data

\*

\* @功能 解包提取数据

\*

\* @参数 input- 源数据

\* output - 解包后的数据

\*

\* @返回值 无返回值

\*/

void extract\_data(const char \*input, char \*output) {

const char \*start = strchr(input, '+'); // 查找第一个+号

if (start == NULL) {

output[0] = '\0'; // 输入字符串格式不正确，返回空字符串

return;

}

start++; // 跳过第一个+号

start = strchr(start, '+'); // 查找第二个+号

if (start == NULL) {

output[0] = '\0'; // 输入字符串格式不正确，返回空字符串

return;

}

start++; // 跳过第二个+号

start = strchr(start, ','); // 查找接收到的数据里面的第一个逗号

if (start == NULL) {

output[0] = '\0'; // 输入字符串格式不正确，返回空字符串

return;

}

start++; // 跳过接收到的数据里面的第一个逗号

start = strchr(start, ','); // 查找接收到的数据里面的第二个逗号

if (start == NULL) {

output[0] = '\0'; // 输入字符串格式不正确，返回空字符串

return;

}

start++; // 跳过接收到的数据里面的第二个逗号

const char \*end = strchr(start, '\0'); // 查找字符串结尾

if (end == NULL) {

output[0] = '\0'; // 输入字符串格式不正确，返回空字符串

return;

}

// 计算所需字符串的长度，并复制到输出缓冲区

size\_t length = end - start;

strncpy(output, start, length);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @函数名 NearLink\_SendData

\*

\* @功能 通过USART发送数据给NearLink星闪

\*

\* @参数 USARTx- 选择USART串口

\* Data - 发送的数据

\*

\* @返回值 无返回值

\*/

void NearLink\_SendData(USART\_TypeDef \*USARTx,const char \*Data)

{

memset(RxBuffer, 0, sizeof(RxBuffer));

RxCnt = 0;

while(\*Data)

{

USART\_SendData(USARTx,\*Data++);

while(USART\_GetFlagStatus(USARTx,USART\_FLAG\_TXE)==RESET)

{

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @函数名 USARTx\_CFG

\*

\* @功能 初始化USART2外设

\*

\*

\* @返回值 无返回值

\*/

void USARTx\_CFG(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure = {0};

//定义GPIO初始化结构体

USART\_InitTypeDef USART\_InitStructure = {0};

//定义USART初始化结构体

NVIC\_InitTypeDef NVIC\_InitStructure = {0};

//定义NVIC初始化结构体

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_USART2, ENABLE);

//使能USART2的时钟

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE);

//使能GPIOA的时钟

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_2;

//配置PA2引脚

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

//设置频率

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP;

//复用推挽输出模式

GPIO\_Init(GPIOA, &GPIO\_InitStructure);

//初始化GPIOA的PA2引脚

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_3;

//配置PA3引脚

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;

//悬空输入模式

GPIO\_Init(GPIOA, &GPIO\_InitStructure);

//初始化GPIOA的PA3引脚

//配置USART参数

USART\_InitStructure.USART\_BaudRate = 115200;

//设置波特率

USART\_InitStructure.USART\_WordLength = USART\_WordLength\_8b;

//配置每个数据帧包含8位数据

USART\_InitStructure.USART\_StopBits = USART\_StopBits\_1;

//配置停止位数

USART\_InitStructure.USART\_Parity = USART\_Parity\_No;

//配置不使用奇偶校验

USART\_InitStructure.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;

//配置不使用硬件流控

USART\_InitStructure.USART\_Mode = USART\_Mode\_Tx | USART\_Mode\_Rx;

//启用发送接收模式

USART\_Init(USART2, &USART\_InitStructure);

//初始化USART2

USART\_ITConfig(USART2, USART\_IT\_RXNE, ENABLE);

//启用 USART2的接收数据寄存器非空中断

NVIC\_InitStructure.NVIC\_IRQChannel = USART2\_IRQn;

//配置 NVIC 为 USART2的中断通道

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 1;

//配置 USART2中断的抢占优先级为 1

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 1;

//配置 USART2中断的子优先级为 1

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;

//使能 USART2中断

NVIC\_Init(&NVIC\_InitStructure);

//初始化NVIC

USART\_Cmd(USART2, ENABLE);

//使能USART2

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 main

\*

\* 函数功能 主函数

\*

\* 返回值 无返回值

\*/

int main(void)

{

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);

//配置NVIC的中断优先级分组

SystemCoreClockUpdate();

//更新系统核心时钟频率

Delay\_Init();

//延迟函数初始化

USART\_Printf\_Init(115200);

//USART初始化并配置波特率为115200

//打印系统信息

printf("SystemClk:%d\r\n", SystemCoreClock);

//打印系统时钟频率

printf( "ChipID:%08x\r\n", DBGMCU\_GetCHIPID() );

//打印芯片信息

printf("NearLink\_T Test\r\n");

//打印字符串“NearLink\_T Test”

USARTx\_CFG();

//对 USART2外设进行初始化。

//STEP1:发送AT等待星闪初始化

NearLink\_SendData(USART2, "AT\r\n");

Delay\_Ms(100);

//STEP2:等待OK

while(strstr((char \*)RxBuffer, "OK") == NULL){}

//STEP3:发送AT+SEEK=1等待星闪返回OK

NearLink\_SendData(USART2,"AT+SEEK=1\r\n");

Delay\_Ms(200);

while(strstr((char \*)RxBuffer,"SEEKDATA") == NULL){}

//STEP4:处理返回的数据，解包出地址

char NearLink\_Address[100] = {0};

extract\_address(RxBuffer, NearLink\_Address);

//STEP5:发送AT+SEEK=0等待星闪返回OK

NearLink\_SendData(USART2, "AT+SEEK=0\r\n");

Delay\_Ms(100);

while(strstr((char \*)RxBuffer, "OK") == NULL){}

//STEP6:连接另一块星闪，等待返回OK

char Nearlink\_SetAddress[128]={0};

sprintf(Nearlink\_SetAddress,"AT+CONNECT=0,%s\r\n",NearLink\_Address);

NearLink\_SendData(USART2, Nearlink\_SetAddress);

Delay\_Ms(100);

while(strstr((char \*)RxBuffer, "OK") == NULL){}

//STEP7:发送数据

char Data[128]={"1234"};

char Nearlink\_Send\_Data[128]={0};

sprintf(Nearlink\_Send\_Data,"AT+SENDDATA=0,%s,%s\r\n",NearLink\_Address,Data);

NearLink\_SendData(USART2,Nearlink\_Send\_Data);

Delay\_Ms(100);

while(strstr((char \*)RxBuffer, "OK") == NULL){}

printf("NearLink Send Data Succeed\r\n");

//STEP8:接收返回数据

while(USART\_GetFlagStatus(USART2, USART\_FLAG\_RXNE) != SET){}

Delay\_Ms(100);

while(strstr((char \*)RxBuffer, "RECEIVEDATA") == NULL){}

printf("NearLink Receive Data Succeed\r\n");

//STEP9:处理返回的数据

memset(Data, 0, sizeof(Data));

extract\_data(RxBuffer,Data);

printf("Receive Data:%s\r\n",Data);

while(1)

{

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 USART2\_IRQHandler

\*

\* 函数功能 此函数处理 USART2 全局中断请求。

\*

\* 返回值 无返回值

\*/

void USART2\_IRQHandler(void)

{

if(USART\_GetFlagStatus(USART2, USART\_FLAG\_RXNE) != RESET)

//判断 USART2 接收中断是否发生

{

RxBuffer[RxCnt++] = USART\_ReceiveData(USART2);

}

}