# 1. 准备工作

硬件准备

首先需要准备一个开发板，这里我准备的是STM32L4的开发板（BearPi）：

mark

# 2.生成MDK工程

选择芯片型号

打开STM32CubeMX，打开MCU选择器：

mark

搜索并选中芯片STM32L431RCT6:

mark

配置时钟源

如果选择使用外部高速时钟（HSE），则需要在System Core中配置RCC；

如果使用默认内部时钟（HSI），这一步可以略过；

这里我都使用外部时钟：

mark

配置串口

小熊派开发板板载ST-Link并且虚拟了一个串口，原理图如下：

mark

这里我将开关拨到AT-MCU模式，使PC的串口与USART1之间连接。

接下来开始配置USART1：

mark

NVIC配置

在NVIC中配置USART中断优先级：

mark

配置时钟树

STM32L4的最高主频到80M，所以配置PLL，最后使HCLK = 80Mhz即可：

mark

mark

生成工程设置

mark

代码生成设置

最后设置生成独立的初始化文件：

mark

生成代码

点击GENERATE CODE即可生成MDK-V5工程：

mark

# 3. 在MDK中编写、编译、下载用户代码

定义发送和接收缓冲区

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

uint8\_t hello[] = "quot;USART1 is ready...\n";

uint8\_t recv\_buf;

/\* USER CODE END 0 \*/

## 重新实现中断回调函数 在NVIC一讲中我们探索了HAL库的中断处理机制，HAL中弱定义了一个中断回调函数 `HAL\_UART\_RxCpltCallback`， 我们需要在用户文件中重新定义该函数，放在哪都可以，这里我放在 `main.c` 中：

/\* USER CODE BEGIN 4 \*/

/\* 中断回调函数 \*/

void HAL\_UART\_RxCpltCallback(UART\_HandleTypeDef \*huart)

{

/\* 判断是哪个串口触发的中断 \*/

if(huart ->Instance == USART1)

{

//将接收到的数据发送

HAL\_UART\_Transmit(huart, &recv\_buf, 1,0);

//重新使能串口接收中断

HAL\_UART\_Receive\_IT(huart, &recv\_buf, 1);

}

}

/\* USER CODE END 4 \*/

修改main函数

在main函数中首先开启串口中断接收，然后发送提示信息：

int main(void)

{

HAL\_Init();

SystemClock\_Config();

MX\_GPIO\_Init();

MX\_USART1\_UART\_Init();

/\* USER CODE BEGIN 2 \*/

//使能串口中断接收

HAL\_UART\_Receive\_IT(& huart1, & recv\_buf, 1);

//发送提示信息

HAL\_UART\_Transmit\_IT(&huart1, (uint8\_t\*)hello, sizeof(hello));

/\* USER CODE END 2 \*/

while (1)

{

}

}

编译代码

点击如图所示的按钮编译工程

image.png

编译成功

mark

设置下载器

点击如图所示按钮打开设置页面

image.png

进行下载设置，选择“ST-Link Debugger”，并点击“Settings”。

mark

在“Flash Download”菜单下勾选“Reset and Run”选项，已达到烧录程序后单片机自动复位并运行程序的目的。

mark

下载运行

点击"LOAD"按钮即可烧录代码到单片机中。

image.png

烧录成功

mark

实验现象

下载运行后，实验现象如下：

mark

至此，我们已经学会了如何配置USART使用中断模式发送和接收数据，下一节将讨论如何配置USART使用DMA模式发送数据和接收数据。