

第二次任务

嵌软II：中断，串口通信

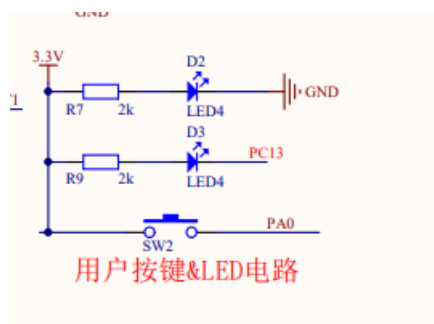
中断任务

知识学习

- 中断概念：STM32中CPU正在运行处理某个事件时，有另一个任务需要芯片马上去处理，STM32中CPU停下正在处理的事件，去处理这个更加紧急的任务，在处理完这个紧急任务后，重新回到原来停止工作的地方继续运行原事件。
- 为什么要有中断：中断有以下几个作用：首先可以处理好高速的CPU与低速外设的数据传输的问题，然后可以让CPU与多个外设连接同时工作，其次可以处理一些紧急事件，包括一些突发情况，比如断电的情况，让CPU运行更稳定。
- 中断的处理流程：
 1. 中断源发出中断的申请
 2. 判断处理器是否同意该中断，同时判断该中断源是否被屏蔽
 3. 中断优先级进行排序
 4. 记录断点地址，从中断向量表中调取中断
 5. 执行中断
 6. 回到记录的断点地址，继续执行原任务
- HAL库中中断的调用流程：
 1. 当中断触发条件触发时，调用该引脚的中断服务程序EXTI0_IRQHandler 函数
 2. 接着调用外部函数通用处理函数HAL_CPIO_EXTI_IRQHandler
 3. 再调用外部中断回调函数HAL_GPIO_EXTI_Callback,运行中断的内容

实验

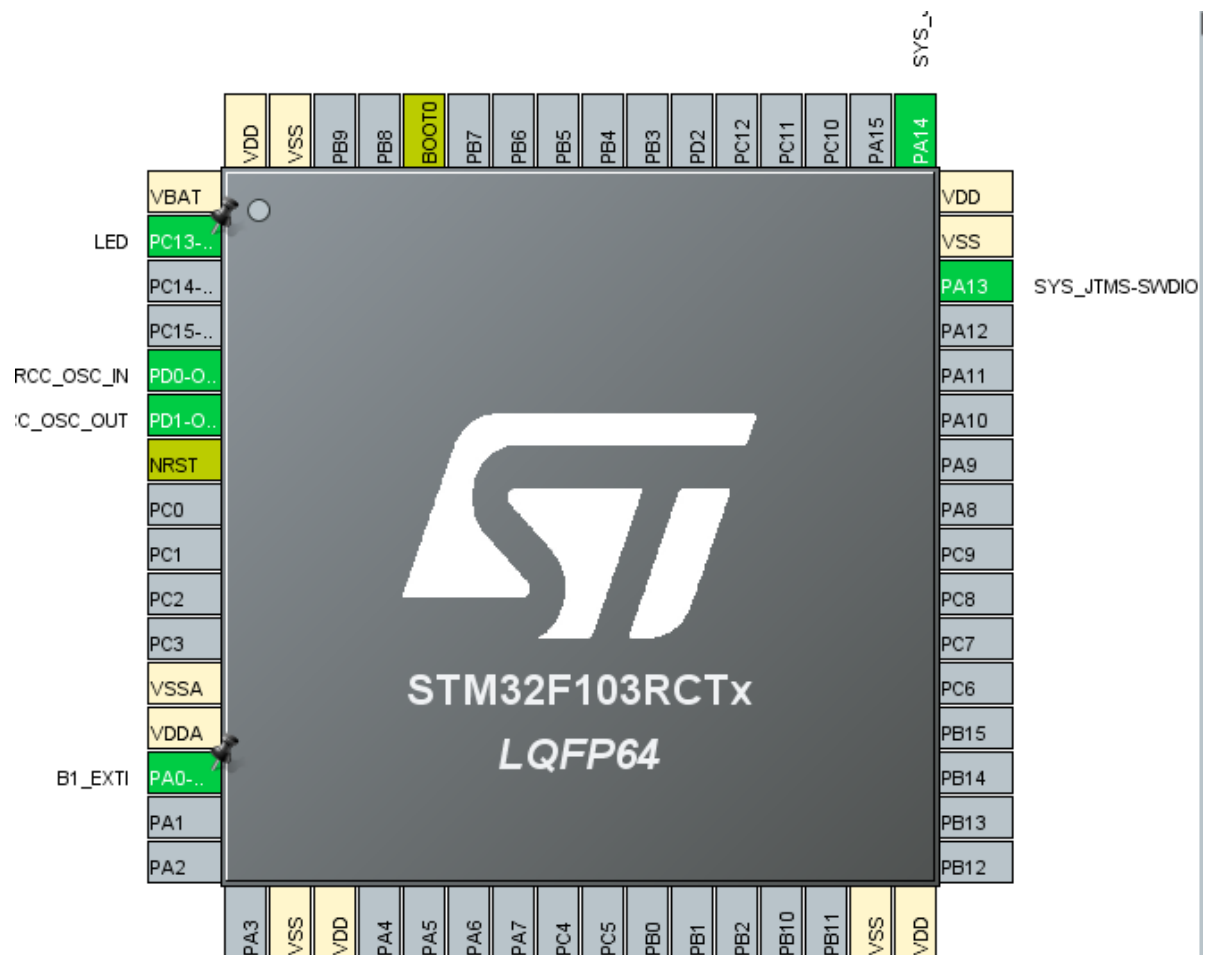
1.电路图分析



当PC13输出高电位时，LED4灯灭；输出低电位时，灯亮

当开关SW2按下时，PA0为低电位；松开时为高电位

2.配置GPIO



Configuration

☐ Group By Peripherals

☒ GPIO

☒ NVIC

Search Signals

☐ Show only Modified Pins

Pin Name	Signal on Pin	GPIO output ...	GPIO mode	GPIO Pull-up...	Maximum ou...	User Label	Modified
PA0-WKUP	n/a	n/a	External Inte...	No pull-up an...	n/a		<input type="checkbox"/>
PC13-TAMP...	n/a	Low	Output Push...	No pull-up an...	Low	LED	<input checked="" type="checkbox"/>

PC13-TAMPER-RTC Configuration :

GPIO output level

Low

GPIO mode

Output Push Pull

GPIO Pull-up/Pull-down

No pull-up and no pull-down

Maximum output speed

Low

User Label

LED

设置好PC13的相关初始参数

Configuration

☐ Group By Peripherals

GPIO

NVIC

Search Signals

Search (Ctrl+F)

☐ Show only Modified Pins

Pin Name	Signal on Pin	GPIO output ...	GPIO mode	GPIO Pull-up...	Maximum ou...	User Label	Modified
PA0-WKUP	n/a	n/a	External Inte...	No pull-up an...	n/a	B1_EXTI	<input checked="" type="checkbox"/>
PC13-TAMP...	n/a	Low	Output Push...	No pull-up an...	Low	LED	<input checked="" type="checkbox"/>

PA0-WKUP Configuration :

GPIO mode

External Interrupt Mode with Falling edge trigger detection

GPIO Pull-up/Pull-down

No pull-up and no pull-down

User Label

B1_EXTI

设置下降沿触发中断

NVIC Mode and Configuration

Configuration

✓ NVIC
✓ Code generation

Priority Group 4 bits for pre-emption priority 0 bits for subpriority Sort by Preemption Priority and Sub Priority

Search
Show only enabled interrupts
Force DMA channels Interrupts

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	✓	0	0
Hard fault interrupt	✓	0	0
Memory management fault	✓	0	0
Prefetch fault, memory access fault	✓	0	0
Undefined instruction or illegal state	✓	0	0
System service call via SWI instruction	✓	0	0
Debug monitor	✓	0	0
Pendable request for system service	✓	0	0
Time base: System tick timer	✓	0	0
PVD interrupt through EXTI line 16	□	0	0
Flash global interrupt	□	0	0
RCC global interrupt	□	0	0
EXTI line0 interrupt	✓	1	0

设置中断优先级为1，子优先级为0

3.代码

```
volatile uint8_t flag = 0;
```

定义flag用于判断按键按下第几次

```
void HAL_GPIO_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == B1_EXTI_Pin)
    {
        flag = (++flag)%3;
    }
}
```

回调函数中每次按下按键对flag进行++运算，同时在取除以3的余数，以此来确定此时所处的状态。

```
if(flag == 0)
{
    HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);
    HAL_Delay(1000);
    HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);
    HAL_Delay(1000);
}
else if(flag == 1)
{
    HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);
    HAL_Delay(50);
    HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);
}
```

```

        HAL_Delay(1000);
    }
    else
    {
        HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);
    }
}

```

LED灯的三种模式以及判断内容，以此实现每按下一次按键，LED就会转向下一个状态。

串口通信

知识学习

- 什么是通信协议？为什么要有通信协议

通信协议是指进行信息交换的两个设备在通信过程所需要遵守的共同规则，其中包括三要素：语法、语义和定时规则。通信协议存在的目的是为了使要进行信息交换的两个设备能顺利接受、识别与处理信息，从而顺利完成整个通信过程。

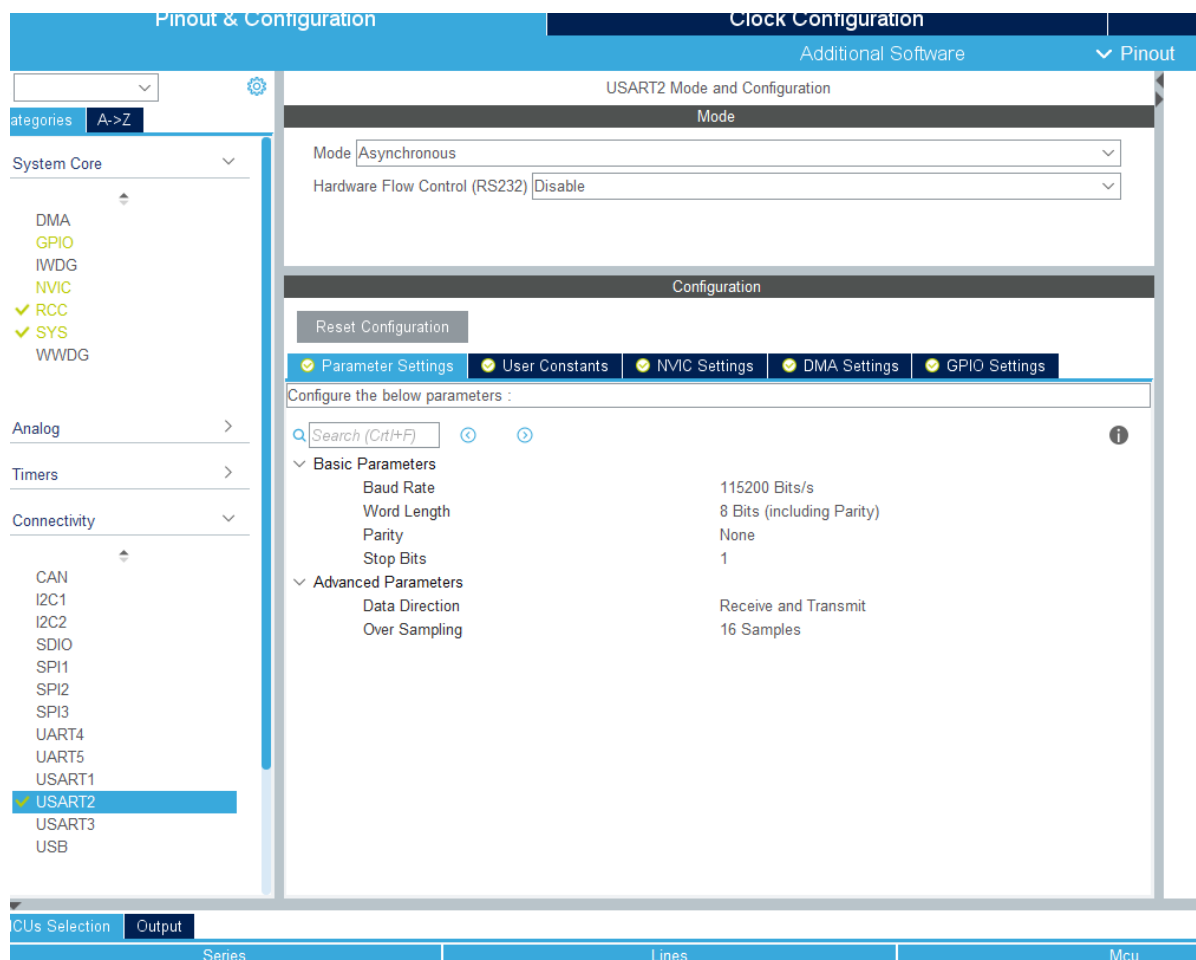
- 通信协议的物理层：物理层是指通过物理传输介质以及相关的通信协议和标准建立起来的物理线路。主要功能有：构建数据通路、透明传输、传输数据、数据编码以及数据传输管理。同时具有机械特性、电气特性、功能特性以及规程特性。
- 通信协议的协议层：协议层指网络七层协议，从上到下分别是应用层、表示层、会话层、传输层、网络层、数据链路层和物理层。通过分成结构能更好的创建互联环境、减低复杂度。
- 数据帧：数据帧是指数据链路层的协议数据单元，包括帧头、数据部分和帧尾。
- 校验位：校验位分为奇偶校验位，作用是判断数据在传输过程中是否有误。奇校验位表示传输数据加上奇校验位中1的个数为奇数个，偶校验位则为偶数个。
- 波特率：表示通信传输数据的效率，指一秒内传输的二进制位数。
- 串口有哪几中断？哪些事件可以触发串口中断

串口中断有4种，RX中断、THRE中断以及RBR中断，其中RBR中断包括可用RDA中断和接收超时CTI中断。按下按键或访问非法地址会触发串口中断。

实验

轮询方式

1.配置引脚



设置了USART2为串口通道

2.代码

```
if(HAL_UART_Receive(&huart2,RecBuf,5,100) == HAL_OK)
{
    HAL_UART_Transmit(&huart2,RecBuf,5,100);
}
```

在PC成功传输5个数据给STM32后，STM32再传输回PC

中断串口通信

1.引脚配置

Additional Software

Pinout

USART2 Mode and Configuration

Mode

Mode Asynchronous

Hardware Flow Control (RS232) Disable

Configuration

Reset Configuration

Parameter Settings

User Constants

NVIC Settings

DMA Settings

GPIO Settings

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
USART2 global interrupt	<input checked="" type="checkbox"/>	0	0

Pinout & Configuration

Clock Configuration

Additional Software

Pinout

Categories

A-Z

System Core

DMA

GPIO

IWDG

NVIC

RCC

SYS

WWDG

analog

mers

connectivity

CAN

I2C1

I2C2

SDIO

SPI1

SPI2

SPI3

UART4

UART5

USART1

USART2

USART3

USB

ultimedia

NVIC Mode and Configuration

Configuration

NVIC

Code generation

Priority Group

4 bits for pre-emption priority 0 bits for subpriority

Sort by Preemption Priority and Sub Priority

Search

Search (Cr...

Show only enabled interrupts

Force DMA channels Interrupts

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Prefetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
USART2 global interrupt	<input checked="" type="checkbox"/>	2	0

Enabled

Preemption Priority

2

Sub Priority

0

Us Selection

Output

Series	Lines	Mcu
M32F1	STM32F103	STM32F103RCTx
M32F1	STM32F103	STM32F103RCYx

在轮询的配置的基础上加上一个中断即可

2.代码

```

void HAL_UART_RxCpltCallback(UART_HandleTypeDef* huart)
{
    if( huart ->Instance == USART2)
    {
        RxFlag = 1;
        HAL_UART_Receive_IT(&huart2, (uint8_t*)RxBuffer, LENGTH);
    }
}

while (1)
{
    /* USER CODE END WHILE */

    if(RxFlag==1)
    {
        RxFlag = 0;
        printf("Recevie Success\n");
        HAL_UART_Transmit_IT(&huart2, (uint8_t*)RxBuffer, LENGTH);
    }
}

```

```
/* USER CODE BEGIN 3 */
```

```
}
```

整个串口处理过程为先进进行串口中断方式接受函数，然后进入串口2的中断服务程序，来调用串口中断通用处理函数，该函数调用串口中断接收函数，当接收完数据后，最后调用串口接收回调函数，即完成了整个过程。