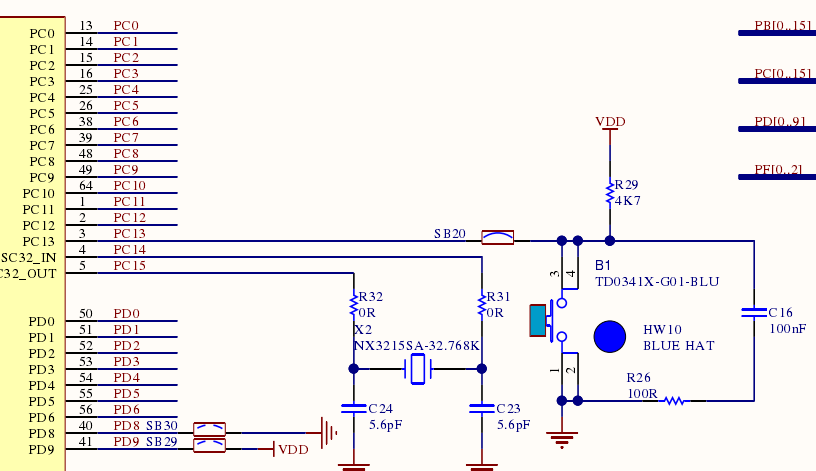
**实现中断计数器在LCD1602显示**

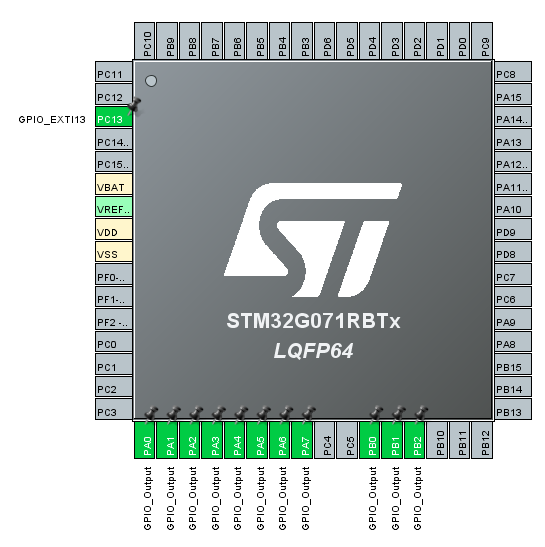
**第一步**：查阅硬件手册找到中断，设计硬件电路连接



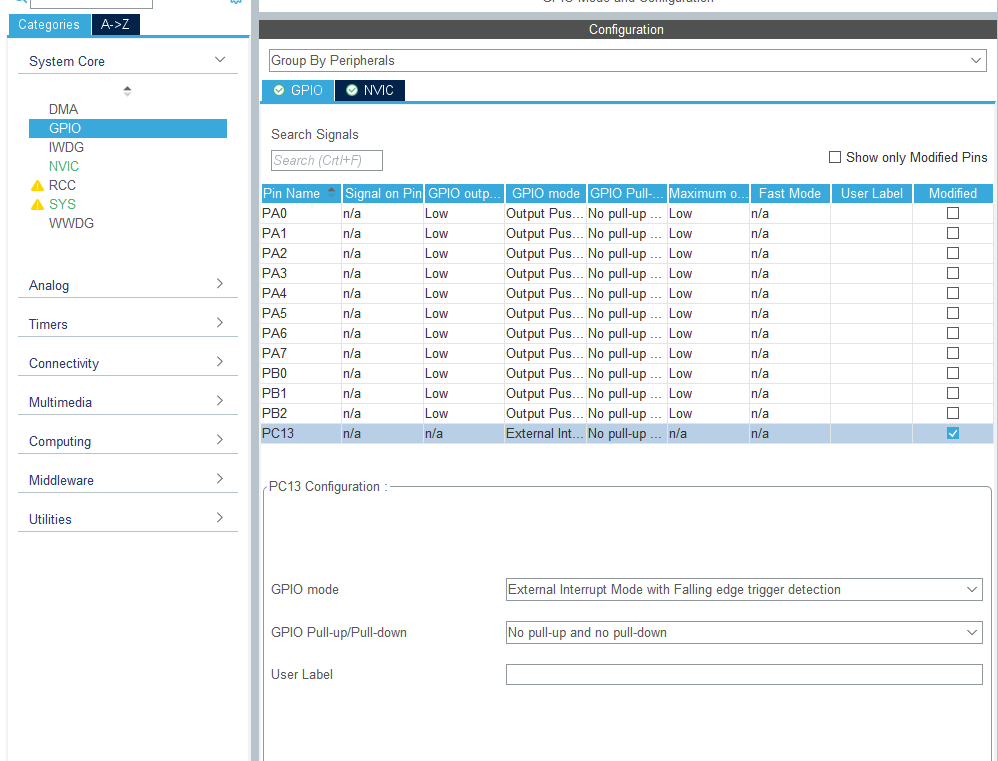
由上图可知中断按键连接在PC13管脚。

LCD1602电路连接：八位数据线连接在GPIOA[7:0],E、RW、RS分别连接在GPIOB[2:0]。

**第二步**：根据电路连接创建工程文件



如上图所示：将PC13设置为中断，将PA[7:0]以及PB[2:0]均设置为输出端口。



将该中断设置为下降沿触发。



使能中断。

**第三步**：添加设计代码

①在main.c文件开头添加：

#define u8 unsigned char

#define u16 unsigned int

void delay(void);

void lcdwritecmd(unsigned char cmd);

void lcdwritedata(unsigned char dat);

void lcdinit(void);

void lcdsetcursor(unsigned char x, unsigned char y);

void lcdshowstr(unsigned char x, unsigned char y, unsigned char \*str);

int location=9;

int counter=0;

unsigned char tstr[5];

②在主函数的GPIO初始化和while(1)循环之间添加：

lcdinit();

delay();

lcdshowstr(0,0,"Interrupt Counter");

lcdshowstr(11,1,"times");

sprintf(tstr,"%d",counter);

lcdshowstr(location,1,tstr);

③在主函数之后添加子函数

void delay ()

{

for(int i=0;i<99;i++)

for(int j=0;j<99;j++)

{}

}

void lcdwritecmd(unsigned char cmd)

{

// lcdwait();

delay();

GPIOB->ODR=0x00; //E=0\RW=0\RS=0

GPIOA->ODR=cmd;

GPIOB->ODR=0x04; //E=1\RW=0\RS=0

delay();

GPIOB->ODR=0x00; //E=0\RW=0\RS=0

}

void lcdwritedata(unsigned char dat)

{

// lcdwait();

delay();

GPIOB->ODR=0x01; //E=0\RW=0\RS=1

GPIOA->ODR=dat;

GPIOB->ODR=0x05; //E=1\RW=0\RS=1

delay();

GPIOB->ODR=0x01; //E=0\RW=0\RS=1

}

void lcdinit()

{

lcdwritecmd(0x38);

lcdwritecmd(0x0c);

lcdwritecmd(0x06);

lcdwritecmd(0x01);

}

void lcdsetcursor(unsigned char x, unsigned char y)

{

unsigned char address;

if(y==0)

address=0x00+x;

else

address=0x40+x;

lcdwritecmd(address|0x80);

}

void lcdshowstr(unsigned char x, unsigned char y, unsigned char \*str)

{

lcdsetcursor(x,y);

while((\*str)!='\0')

{

lcdwritedata(\*str);

str++;

}

}

void HAL\_GPIO\_EXTI\_Falling\_Callback(uint16\_t GPIO\_Pin)

{

counter++;

if(counter>99999)

{

lcdshowstr(0,1," !!!ERROR!!! ");

}

else if(counter>9999) location=5;

else if(counter>999) location=6;

else if(counter>99) location=7;

else if(counter>9) location=8;

sprintf(tstr,"%d",counter);

lcdshowstr(location,1,tstr);

}

**第四步**：总结

全局变量：

int location=9; 辅助LCD1602输出数字的定位（数字字符串的起始位置）

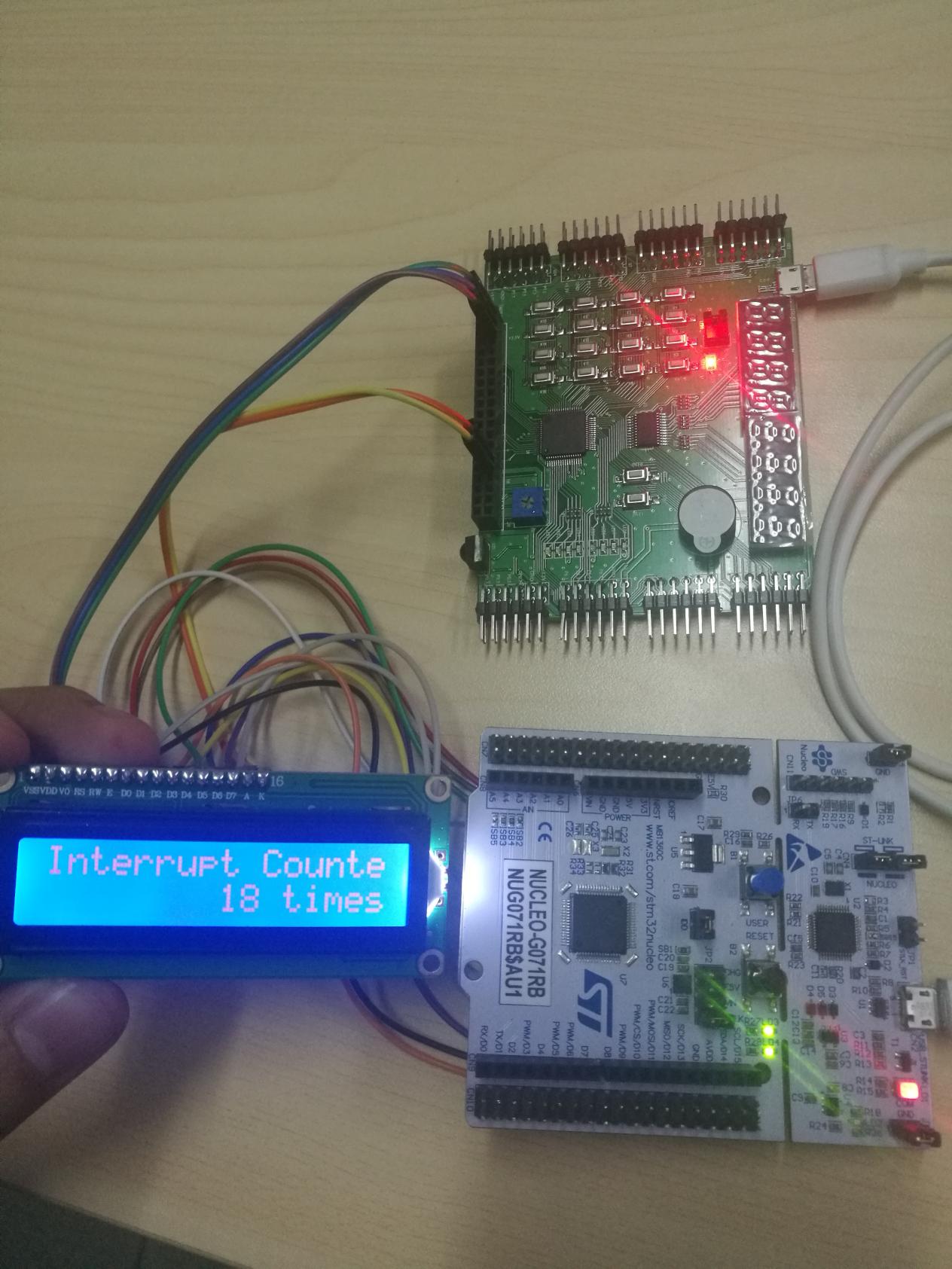
int counter=0; 计数器的值（数字）

unsigned char tstr[5]; 计数器的值转换为字符串才能输出到LCD1602

程序运行：

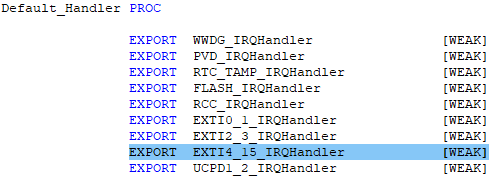
在主函数进入while(1)循环之前初始化LCD1602，并显示初始状态。在进入循环之后，当有中断触发时计数器加一，并重新根据输出数字的位数定位起始位置。【注】：当计数器值超过99999时，超出显示范围输出ERROR，否则输出新的数值。执行完中断程序后，跳转回主函数的while(1)循环，因为该循环内是空操作，所以LCD1602会继续保持原来的输出状态不变，直到下一次中断来临。

**第五步**：效果展示

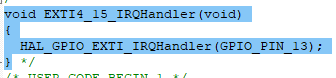


**第六步**：分析上述实验HAL库的应用

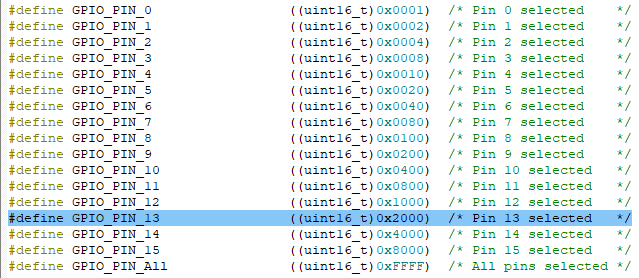
①在stsrtup\_stm32g071xx.s文件下的启动引导代码中先声明了4号--15号外部中断共用一个入口

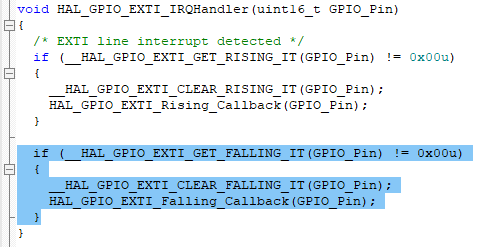


②其中断程序的入口是stm32g0xx\_it.c文件下的函数EXTI4\_15\_IRQHandler(void)



③由上图可知，进入中断后调用了HAL库函数HAL\_GPIO\_EXTI\_IRQHandler(GPIO\_PIN)。管脚GPIO\_PIN的定义在文件stm32g0xx\_hal\_gpio.h文件下。被调用的HAL库函数在stm32g0xx\_hal\_gpio.c文件下。





因为我们设置的中断是下降沿触发所以该函数前半部分关于上升沿触发函数并无作用。

④函数stm32g0xx\_hal\_gpio.c的功能如下：

先通过函数\_HAL\_GPIO\_EXTI\_GET\_FALLING\_IT(GPIO\_PIN)判断中断是否为外部13号管脚产生，如果是4号-15其他管脚产生则不执行任何程序。如果是13号管脚产生，则通过函数\_HAL\_GPIO\_EXTI\_CLEAR\_FALLING\_IT(GPIO\_PIN)清除中断标志位，然后调用函数HAL\_GPIO\_EXTI\_Falling\_callback(GPIO\_PIN)执行中断服务程序。

<用户在main.c文件编写HAL\_GPIO\_EXTI\_Falling\_callback(GPIO\_PIN)中断程序代码>

**第七步**：避开HAL库直接写中断思路。

①将中断程序入口改到main.c文件下：注释掉stm32g0xx\_it.c文件下的函数EXTI4\_15\_IRQHandler(void)。在main.c文件下添加同名函数。

②不再调用其他函数，根据上述中断执行的逻辑在中断程序入口函数下添加内容。自定义中断来源判断函数，自定义清除中断标志位函数。在其之后，直接续写中断服务程序逻辑。

**第八步**：创建工程并添加代码

①创建工程并生成初始代码文件同上。

②在注释掉stm32g0xx\_it.c文件下的函数EXTI4\_15\_IRQHandler(void)；

③在main.c文件下的开头添加变量定义以及函数声明

#define u8 unsigned char

#define u16 unsigned int

void delay(void);

void lcdwritecmd(unsigned char cmd);

void lcdwritedata(unsigned char dat);

void lcdinit(void);

void lcdsetcursor(unsigned char x, unsigned char y);

void lcdshowstr(unsigned char x, unsigned char y, unsigned char \*str);

int location=9;

int counter=0;

unsigned char tstr[5];

④在其之后自定义中断来源判断函数，自定义清除中断标志位函数。

ITStatus EXTI\_GetITStatus(uint32\_t EXTI\_Line)

{

ITStatus bitstatus = RESET;

uint32\_t enablestatus = 0;

/\* Check the parameters \*/

assert\_param(IS\_GET\_EXTI\_LINE(EXTI\_Line));

enablestatus = EXTI->IMR1 & EXTI\_Line;

if (((EXTI->FPR1 & EXTI\_Line) != (uint32\_t)RESET) && (enablestatus != (uint32\_t)RESET))

bitstatus = SET;

else

bitstatus = RESET;

return bitstatus;

}

void EXTI\_ClearITPendingBit(uint32\_t EXTI\_Line)

{

/\* Check the parameters \*/

assert\_param(IS\_EXTI\_LINE(EXTI\_Line));

EXTI->FPR1 = EXTI\_Line;

}

⑤在主函数while(1)循环之前添加代码：

lcdinit();

delay();

lcdshowstr(0,0,"Interrupt Counter");

lcdshowstr(11,1,"times");

sprintf(tstr,"%d",counter);

lcdshowstr(location,1,tstr);

⑥添加LCD1602驱动函数

void delay ()

{

for(int i=0;i<99;i++)

for(int j=0;j<99;j++)

{}

}

void lcdwritecmd(unsigned char cmd)

{

delay();

GPIOB->ODR=0x00; //E=0\RW=0\RS=0

GPIOA->ODR=cmd;

GPIOB->ODR=0x04; //E=1\RW=0\RS=0

delay();

GPIOB->ODR=0x00; //E=0\RW=0\RS=0

}

void lcdwritedata(unsigned char dat)

{

delay();

GPIOB->ODR=0x01; //E=0\RW=0\RS=1

GPIOA->ODR=dat;

GPIOB->ODR=0x05; //E=1\RW=0\RS=1

delay();

GPIOB->ODR=0x01; //E=0\RW=0\RS=1

}

void lcdinit()

{

lcdwritecmd(0x38);

lcdwritecmd(0x0c);

lcdwritecmd(0x06);

lcdwritecmd(0x01);

}

void lcdsetcursor(unsigned char x, unsigned char y)

{

unsigned char address;

if(y==0)

address=0x00+x;

else

address=0x40+x;

lcdwritecmd(address|0x80);

}

void lcdshowstr(unsigned char x, unsigned char y, unsigned char \*str)

{

lcdsetcursor(x,y);

while((\*str)!='\0')

{

lcdwritedata(\*str);

str++;

}

}

⑦添加中断服务程序

void EXTI4\_15\_IRQHandler(void)

{

if (EXTI\_GetITStatus(0x2000) != 0x00) // 0x2000==EXTI\_PIN\_13

{

EXTI\_ClearITPendingBit(0x2000); //Clear interrupt flag bit

counter++; //interrupt users code

if(counter>99999)

{

lcdshowstr(0,1," !!!ERROR!!! ");

}

else if(counter>9999) location=5;

else if(counter>999) location=6;

else if(counter>99) location=7;

else if(counter>9) location=8;

sprintf(tstr,"%d",counter);

lcdshowstr(location,1,tstr);

}

}

**第九步**：备注

在其管脚定义中的unit16\_t只是变量类型的说明，对代码运行结果并无实际影响。例如：(unit16\_t)0x2000=0x2000。相同的0x00u，这里的u也是数据类型说明，代表unsigned long无符号长整型数，对代码运行结果也无实际影响。例如：0x00u=0x00。