# 中国科学技术大学信息科学技术学院

# 2023-2024 学年第 2 学期考试【题型示意卷】

■样卷　　□A 卷　　□B 卷

半开卷考试要求：

（1） 可携带一张大小不超过 A4（即宽度小于 21cm，长度小于 29.7cm）的纸张。

（2） 所携带纸张上需要写上学号和姓名。

（3） 考试结束时，考生所携带纸张需要随答卷一起上交。

题型示意卷说明：

（1） 各题型占比非最终考题占比，仅用于题型示意。

（2） 最终考题题型不超越本卷范围。

（3） 各章节知识点分值占比非最终考题分值占比。

（4） 各小题分值非最终考题分值。

（5） 样卷不提供参考答案。

课程名称：计算机原理与嵌入式系统　　课程编号：EE2003

开课院系：信息科学技术学院　　考试形式：半开卷

姓名：＿＿＿＿＿＿　学号：＿＿＿＿＿＿　班级：＿＿＿＿＿＿

| 题号 | 一 | 二 | 三 | 四（　） | 四（　） | 四（　） | 五（　） | 五（　） | 五（　） | 五（　） | 总分 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 得分 | | | | | | | | | | | |

*第四题、第五题中选做的题号请填入上表

第二题（选择题）答题卡

| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| 21 | 22 | 23 | 24 | 25 | | | | | |
| | | | | | | | | | |

第三题（判断题）答题卡

| 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| 36 | | | | | | | | | |
| | | | | | | | | | |

（装订线内不要答题）

## 一、 填空题 (10 分，请直接在试卷上答题)

**请将学号前 2 个字符替换为“PB”，然后把中间 8 个字符左右反转后作为 16 进制的数位。如学号 PB22012345，得到 16 进制数 0x4321022B；再如，学号 IJ23061374，则得到 16 进制数 0x7316032B。回答问题 1~5。**

1、 若该 16 进制数为原码表示的数值。把这个数表示为补码后的二进制比特串为 _____。

2、 若该 16 进制数为补码表示的数值。则这个数原码对应的二进制比特串为 _____。

3、 若该 16 进制数表示的是定点小数，且小数点在最低比特的左边。则这种定点小数表示法可以表示的数值范围为_____。

4、 若该 16 进制数对应的是 C 程序中 float 类型变量 *x*，则 *x* 所表达的数值为 _____（给出表达式即可）。

5、 现将该 16 进制数存储在计算机的主存储器中，存放地址为 0x0001 0100 ~ 0x0001 0104。在下面 2 张表中分别填写大端格式和小端格式时各地址对应存储单元的内容（填写 16 进制格式）。

| 大端格式 | |
|---|---|
| 0x0001 0103 | |
| 0x0001 0102 | |
| 0x0001 0101 | |
| 0x0001 0100 | |

| 小端格式 | |
|---|---|
| 0x0001 0103 | |
| 0x0001 0102 | |
| 0x0001 0101 | |
| 0x0001 0100 | |

6、 _____、_____和控制相关冲突会导致指令流水线出现断流。

7、 常见存储器类型有_____、_____和光存储器等。

8、 主存块到 Cache 块的地址映射包括直接相联映射、_____、_____三种地址映射方案。

9、 Cortex-M3/M4 中断向量表中保存的是_____。

10、 Cortex-M3 内部的 NVIC 支持_____个外部中断的管理。

## 二、 单项选择题 (30 分，请试卷首页答题卡上答题)

11、 已知带符号整数用补码表示。变量 X，Y，Z 的机器数分别为 FFFDH，FFDFH，7FFCH，下列结论中，正确的是（ ）

A) 若 X，Y，Z 为无符号整数，则 Z<X<Y
B) 若 X，Y，Z 为无符号整数，则 X<Y<Z
C) 若 X，Y，Z 为带符号整数，则 X<Y<Z
D) 若 X，Y，Z 为带符号整数，则 Y<X<Z

12、 理想情况下下列处理器类型中，CPI(Cycle Per Instruction)为 1 的是（　　）

I. 单周期 CPU（指令周期等于 T 周期）
II. 多周期 CPU（指令周期对应多个 T 周期）
III. 基本流水线 CPU
IV. 超标量流水线 CPU

A) 仅 I、II
B) 仅 I、III
C) 仅 II、IV
D) 仅 III、IV

13、 2016 年 6 月，由国家并行计算机工程技术研究中心研制的“神威・太湖之光”在全球超级计算机 TOP500 榜单上排名第一。其浮点运算速度为 93.0146 PFLOPS，这说明该计算机每秒完成的浮点数操作次数为（　　）

A) $9.3\times10^{13}$ 次
B) $9.3\times10^{15}$ 次
C) 9.3 千万亿次
D) 9.3 亿亿次

14、 以下关于微程序控制器中机器指令、微指令、微操作描述正确的是（　　）

I. 每一条机器指令分解为若干条微指令
II. 每一条机器指令对应一条微指令来执行
III. 每一条微指令对应一个微操作
IV. 每一条微指令对应若干个微操作
V. 每一条微指令对应若条机器指令

A) I、II
B) I、III
C) II、III
D) III、IV
E) IV、V
F) A~E 均不合题意

15、 以下关于典型 ROM 说法正确的是（　　）

I. Mask ROM（掩模 ROM）中的数据用户不能修改
II. ROM 芯片中字线用来选中存储单元，而位线用于数据的输入和输出
III. PROM (Programmable ROM)可以利用外围电路实现数据的多次写入
IV. EPROM (Erasable PROM)可以利用外围电路实现数据的多次修改
V. EEPROM (Electrically EPROM)可以利用外围电路实现数据的多次修改
VI. Flash 和 $E^2$PROM 都基于浮置栅中是否存有电荷来实现“0”或“1”的逻辑状态存储
VII. NAND Flash 比 NOR Flash 存储密度更高是因为少一个选通管

A) I、II、III
B) II、III、IV
C) I、II、V
D) V、VI、VII
E) IV、V、VII
F) A~E 均不合题意

16、 以下关于 Cache 说法正确的是（　　）

A) 贯穿读出（Look Through）方式下，CPU 得到的所有数据均从 Cache 中读出的
B) 旁路读出（Look Aside）方式下，CPU 得到的所有数据均从 Cache 中读出的
C) 写通方式（Write Through）方式下，CPU 写入存储器的数据可能不经过 Cache
D) 写回方式（Write Back）方式下，CPU 写入存储器的数据可能不经过 Cache

17、 以下关于总线时序说法正确的是（　　）

I. AHB Lite 支持多主控设备
II. AHB 的突发传输不仅支持单个数据的传输，也支持数据块的传输
III. AHB 的突发传输在传输开始前必须指定传输数据的长度
IV. AHB 所传输的数据本身和数据的地址需要在同一个时钟周期推送到总线上

V. AHB 所传输的数据本身和数据的地址必须在不同的时钟周期推送到总线上
VI. AHB 的 Split 功能提供了灵活性，但不利于提高总线利用率

A) I、IV　C) II、V　E) II、VI
B) II、IV　D) III、V　F) A~E 均不合题意

18、 以下关于 AHB 说法正确的是（　　）

I. AHB Lite 支持多主控设备
II. AHB 的突发传输不仅支持单个数据的传输，也支持数据块的传输
III. AHB 的突发传输在传输开始前必须指定传输数据的长度
IV. AHB 所传输的数据本身和数据的地址需要在同一个时钟周期推送到总线上
V. AHB 所传输的数据本身和数据的地址必须在不同的时钟周期推送到总线上
VI. AHB 的 Split 功能提供了灵活性，但不利于提高总线利用率

A) I、IV　C) II、V　E) II、VI
B) II、IV　D) III、V　F) A~E 均不合题意

19、 以下关于“I/O 端口”说法正确的是（　　）

I. 一个 I/O 接口只能有一个 I/O 端口
II. I/O 端口本质上是具备存储能力的电路单元
III. 一般的 I/O 接口电路中常包括控制端口、数据端口、状态端口
IV. I/O 端口的地址可以和主存中存储单元地址统一编址
V. I/O 端口的地址必须位于独立于主存的地址空间
VI. I/O 端口统一编址情形，I/O 端口的地址必须位于高位地址段

A) I、II、III　D) IV、V、VI
B) II、III、IV　E) A~E 均不合题意
C) III、IV、V

20、 以下关于 SPI 总线说法正确的是（　　）

I. SPI 是全双工通信总线，即主设备读从设备、主设备写从设备可同时进行
II. 采用 SPI 通信时，主设备和从设备是共用时钟信号的
III. SPI 主设备通过 CS#信号指示从设备数据传输的开始与结束
IV. SPI 是同步通信总线
V. SPI 总线是串行通信总线
VI. 从设备向主设备发送数据既可以在时钟上升沿，也可以在时钟的下降沿
VII. 接收设备采样数据线的时间总是滞后发送设备推送数据半个周期

A) 仅 I、II、III、IV　D) 仅 IV、V、VI、VII
B) 仅 II、III、IV、V　E) A~D 均不合题意
C) 仅 III、IV、V、VI

21、 以下关于 Cortex-M3/M4 操作状态与操作模式法正确的是（　　）

I. 特权线程模式下可以通过置位 CONTROL 寄存器 nPRIV 位进入非特权模式
II. 非特权线程模式下可以通过置零 CONTROL 寄存器 nPRIV 位进入特权模式
III. 处理模式下可以通过置零 CONTROL 寄存器 nPRIV 位进入特权线程模式
IV. 非特权模式下访问 CONTROL 寄存器会触发 Usage Fault 异常
V. 非特权线程模式下访问存储器可能会触发 MemManage Fault 异常
VI. 处理模式下可以通过置零 CONTROL 寄存器 nPRIV 位进入特权线程模式
VII. 处理模式下只能使用 MSP 而不能使用 PSP
VIII. 线程模式下只能使用 PSP 而不能使用 MSP

A) I、II、III、IV、V
B) II、III、IV、V、VI
C) III、IV、V、VI、VII
D) IV、V、VI、VII、VIII
E) A~E 均不合题意

22、 Cortex-M3 采用的是以下哪种堆栈（ ）

A) Full Ascending Stack
B) Full Descending Stack
C) Empty Ascending Stack
D) Empty Descending Stack

23、 以下关于 Cortex-M3/M4 存储器子系统说法正确的是（ ）

I. 4GB 的存储器映射关系是固定不变的
II. 既可以配置为大端模式，也可以配置为小端模式
III. 所有 Cortex-M3/M4 的存储器访问指令都支持非对齐的数据传输
IV. 4GB 空间所有区域均可以采用位段（Bit-Band）操作方式来访问
V. 位段操作能保障“读→修改→写”的流程不被其他操作打断
VI. 处理器读存储器时，存储器系统会检查所访问区域的存储器访问属性
VII. 非特权线程访问内核私有区域可能会触发 MemManage Fault 异常
VIII. 处理器不会改变代码的执行顺序，因而不需要存储器屏障指令

A) I、II、III、IV
B) II、III、IV、V
C) III、IV、V、VI
D) IV、V、VI、VII
E) V、VI、VII、VIII
F) A~E 均不合题意

24、 以下关于 Cortex-M3/M4 中断/异常机制说法正确的是（ ）

I. 特殊寄存器 PRIMASK 和 FAULTMASK 均用于中断屏蔽
II. 与 PRIMASK 不同的是，FAULTMASK 无需清理，当负责错误处理的异常处理程序返回时，会自动复位 FAULTMASK
III. 特殊寄存器 BASEPRI 采用了可伸缩的设计
IV. 每个异常都会处于激活、非激活、挂起、激活并挂起之一的某一个状态
V. 每个异常都有一个独立的优先级寄存器
VI. 异常的优先级寄存器只能由特权访问等级代码访问
VII. 中断优先级寄存器中数值约小，表示对应中断源的优先级约高
VIII. 多个相同优先级的异常同时发生时，处理器会先处理异常类型号低的异常
IX. 中断使能、中断禁止采用了 2 个独立的寄存器
X. 特定中断源的中断使能或禁止用寄存器中的 1 位来实现

A) I、II、III、IV、V
B) II、III、IV、V、VI
C) III、IV、V、VI、VII
D) IV、V、VI、VII、VIII
E) V、VI、VII、VIII、IX
F) VI、VII、VIII、IX、X
G) A~F 均不合题意

25、 以下关于 STM32F103 说法正确的是（ ）

I. 最小硬件系统至少应包含电源、时钟、复位电路模块
II. 其中断向量表存放的起始地址可以由用户随意指定
III. 其 SysTick 计数时钟来源可以配置
IV. 各个 GPIO、TIM 等模块可以由用户指定连接在 APB1 或 APB2 上
V. 既支持大端模式也支持小端模式
VI. 其多数引脚支持引脚功能复用

VII. 其各定时器模块只能对 SysTick 进行计数

A) I、II、III　　C) III、V、VI　　E) V、VI、VII

B) II、III、IV　　D) IV、V、VI　　F) A~E 均不合题意

## 三、 判断题 (11 分，打√或×，请试卷首页答题卡上答题)

26、 Cortex-M3 处理器内核采用了冯诺依曼结构的三级流水线。（　　）

27、 Cortex-M3 处理器工作在线程模式下时，代码一定是非特权的。（　　）

28、 Cortex-M3 中某些特殊寄存器只有特权访问等级才能执行。（　　）

29、 在单主控设备、多从控设备的情形下是不需要总线仲裁的。（　　）

30、 AHB 所传输的数据及其地址必须在同一个时钟周期推送到总线上。（　　）

31、 冯·诺依曼结构和哈佛结构是两种不同的 ISA。（　　）

32、 Cortex-M3 系列处理器支持 Thumb-2 指令集。（　　）

33、 Cortex-M3 中 MOV 指令可将数据从一个寄存器送到存储器的特定位置。（　）

34、 Cortex-M3 中异常的类型号越小，该异常所对应的优先级就越高。（　　）

35、 存储器字扩展时，各芯片不同位数据线分别与数据总线 $D_7$~$D_0$ 位并联。（　）

36、 非特权线程试图访问内核私有区域，将导致 MemUsage 类异常。（　　）

## 四、 简答题(9 分，每题 3 分，**5 题选做 3 题**，在稿纸上答题)

37、 简述 MIPS 作为计算机性能评价指标的优缺点。

38、 简要对比 SRAM 和 DRAM。

39、 简述 TCM 与高速缓存 Cache 的异同。

40、 简述 Cortex-M3 与 Cortex-M4 使用两个堆栈的优点。

41、 简述 Cortex-M 处理器的中断优先级分组机制。

## 五、 设计及综合题 (40 分，每题 10 分，**6 题选做 4 题**，在稿纸上答题)

42、 某计算机寄存器宽度是 32 位，主存地址为 32 位，按字节编址（即每存储单元 1B），使用大端模式（低字节存放在地址最大的单元），未使用虚拟存储器技术。数据 Cache 与主存之间均采用组相联映射方式，Cache line 中数据字段大小为 16B，数据 Cache 容量为 16KB，代码运行前数据 Cache 为空。有如下 C 语言程序段：

```
for(int k=8191; k>=0; k=k-1)    s[k]=0x0800 0000 + k;
```

若数组 s 及变量 k 均为 int 型，int 型数据占 4B；变量 k 分配在寄存器中，数组 s 在主存中的起始地址为 0800 0000H。请回答下列向题。

（1） 已知该计算机采用补码表示法，写出 s[6]的真值（写出表达式即可）。

（2） 列出 s[6]所占用的 4 个存储单元的地址。

（3） 写出 s[6]所占用的 4B 中最高地址所对应字节的二进制值。

（4） 列出 s[6]载入 Cache 后，其 4B 各自对应的 Cache 地址。

（5） 代码执行过程中，访问数组 s 的数据 Cache 缺失次数是多少？

43、 某系统采用 STM32F407 处理器（Cortex-M4，32 根地址线，32 根数据线），存储器按字节编址。根据图 1 信息，回答下列问题。

（1） 该系统（芯）片内 SRAM 容量为多少 KB？

（2） 该系统（芯）片内 Flash 容量为多少 KB？

（3） 该系统（芯）片外存储的容量和有效地址范围是？

（4） 若采用规格为 128K × 8-bit 的 SRAM 芯片实现（3）中的片外存储，给出片外扩展 SRAM 存储器子系统与 CPU 的连接示意图（要求体现地址线连接关系）。

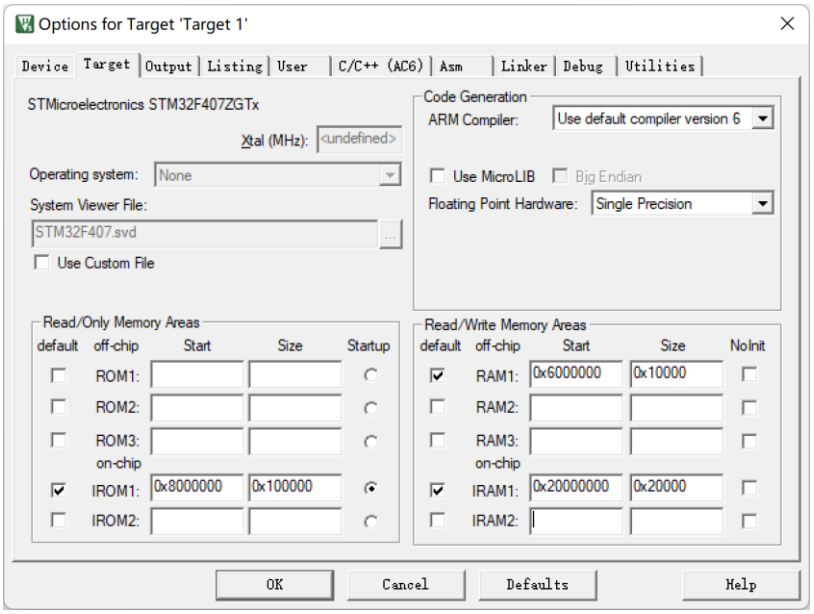


图 1 某 STM32 系统的片内存储器

44、 参考附录 3，分析图 2 中总线操作时序。回答下列问题。

（1） HTRANS[1:0]信号“???”应该为什么取值？

（2） HADDR[31:0]信号“地址 A”和“地址 B”应该为什么取值？

（3） HRDATA[31:0]中数据 1、数据 2、数据 3、数据 4 对应的地址分别是什么？

（4） HRDATA[31:0]中数据 1、数据 2、数据 3、数据 4 的含义分别是什么？

（5） 若数据 1 的传输是由指令“LDR R1, [R0]”引起的，该指令执行后 R0 和 R1 中保存的值是什么？

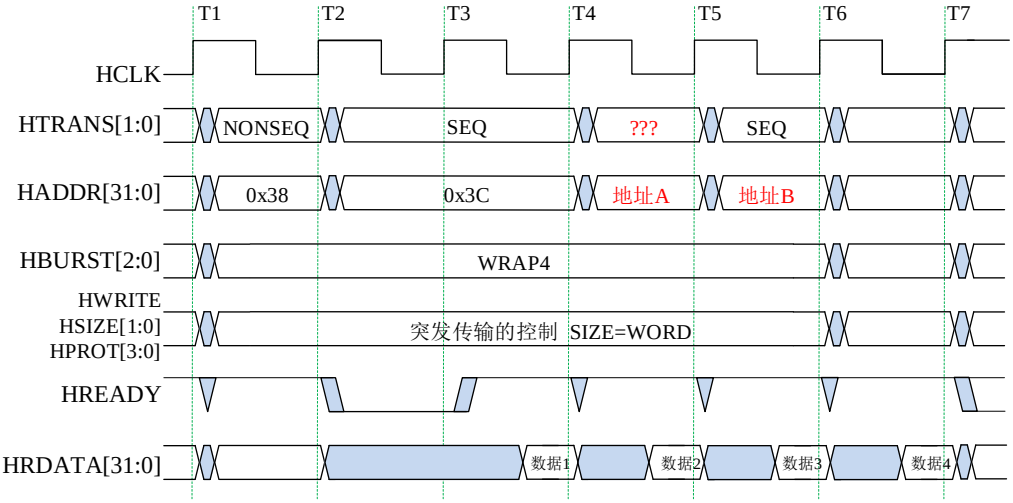


图 2 某 STM32 系统的总线操作示意图

45、 在一个 STM32 点亮 LED 的应用中，部分程序代码如附录 1 所示。请回答以下问题。

（1） 简述 GPIO_Configuration 函数对 IO 接口配置的步骤。

（2） 分析该程序，LED 分别连接在哪些 IO 引脚上？请画出一个简明硬件连线图；根据此连线图，当引脚输出高电平时，是点亮还是熄灭 LED?

（3） 分析循环点亮 LED 代码，补充相应的注释①②。

（4） 可以用定时器实现精确循环（点亮）功能。已知系统时钟为 72MHz，采用定时器 TIM2 产生周期为 500ms 的定时时间间隔（控制 LED 的亮灭）。请简述定时器配置的主要步骤，并补充完整定时函数的空余部分③④。

46、 在实验所用 STM32F407 实验箱上用拨动开关 DIP0/DIP1 控制 LED0/LED1 亮灭。硬件连接如图 3 所示。拟设计的程序功能为：拨动开关 DIP0/DIP1 由不接地（PE4==1/PE5==1）拨动至接地（PE4==0/PE5==0）时，触发中断来实现 LED0 和 LED1 亮灭切换。部分代码如附录 2 所示。请结合课程知识，分析附录 1 代码，回答以下问题。

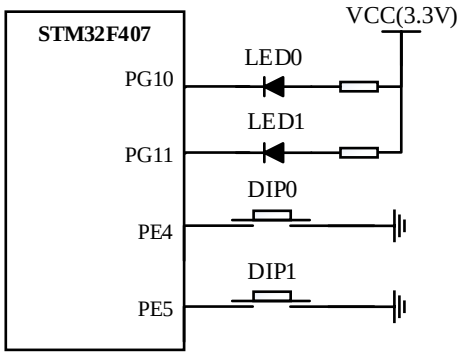


图 3 实验箱的 I/O 引脚连接示意图

（1） 根据附录 3 表 1 关于 GPIO 寄存器的描述，解释代码行①和②的含义。

（2） 简要说明 main()函数所实现的功能。

（3） 根据附录 3 信息，写出 EXTI9_5_IRQHandler()函数的入口地址。

（4） 修改 EXTI9_5_IRQHandler()函数代码，实现如下功能：在 DIP0 接地（PE4==0）时保持原有功能，而 DIP0 不接地时点亮 LED0 并熄灭 LED1。

# 附录 1：STM32 点亮 LED 的主程序代码片段

```
void GPIO_Configuration(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2|GPIO_Pin_3;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
}
//…
while (1)
  {
  /*循环点亮 LED*/
  GPIO_WriteBit(GPIOA, GPIO_Pin_2, (BitAction)0x01);   //①
  Delay(0XFFFFF);
  GPIO_WriteBit(GPIOA, GPIO_Pin_2, (BitAction)0x00);   //②
  Delay(0XFFFFF);
  GPIO_WriteBit(GPIOA, GPIO_Pin_3, (BitAction)0x01);
  Delay(0XFFFFF);
  GPIO_WriteBit(GPIOA, GPIO_Pin_3, (BitAction)0x00);
  Delay(0XFFFFF);
  }
//…
void TIM2_Delay500MS ( )
{
  TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure ;
  RCC_APB1PeriphClockCmd (RCC_APB1Periph_TIM2, ENABLE) ;
  TIM_TimeBaseStructure.TIM_Prescaler=   ③ ;
  TIM_TimeBaseStructure.TIM_Period=   ④ ;
  TIM_TimeBaseStructure.TIM_CounterMode= TIM_CounterMode_Up ;
  TIM_TimeBaseInit (TIM2, &TIM_TimeBaseStructure) ;
  TIM_ClearFlag ( TIM2 , TIM_FLAG_Update ) ;
  TIM_Cmd (TIM2, ENABLE) ;
  while (TIM_GetFlagStatus (TIM2, TIM_FLAG_Update) ==RESET) ;
}
```

# 附录 2：中断相关代码（main.c）

```
//此处略去头文件信息
#define LED0_IsOn()    GPIO_ReadInputDataBit (GPIOG,GPIO_Pin_10)//读取 PG10

void LED0_Config (void) ;
void KEY0_Config(void) ;
void EXTI_Config (void) ;
void NVIC_Config (void) ;

int main (void)
{
      LED0_Config ( ) ;//
      KEY0_Config ( ) ;//
      EXTI_Config ( ) ;//
      NVIC_Config ( ) ;//
      while (1) ;
}
```

```
void EXTI_Config (void)
{
    EXTI_InitTypeDef EXTI_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);
    SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOE, GPIO_PinSource5);
    EXTI_InitStructure.EXTI_Line= EXTI_Line5;
    EXTI_InitStructure.EXTI_Mode= EXTI_Mode_Interrupt;
    EXTI_InitStructure.EXTI_Trigger=EXTI_Trigger_Falling;
    EXTI_InitStructure.EXTI_LineCmd= ENABLE;
    EXTI_Init(&EXTI_InitStructure) ;
}

void NVIC_Config (void)
{
    NVIC_InitTypeDef NVIC_InitStructure;
    NVIC_PriorityGroupConfig (NVIC_PriorityGroup_1) ;
    NVIC_InitStructure.NVIC_IRQChannel= EXTI9_5_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority= 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority= 1;
    NVIC_InitStructure.NVIC_IRQChannelCmd= ENABLE;
    NVIC_Init ( &NVIC_InitStructure) ;
}
void LED0_Config (void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOE | RCC_AHB1Periph_GPIOG, ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11 | GPIO_Pin_10;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
    GPIO_Init(GPIOG, &GPIO_InitStructure);
}
void KEY0_Config (void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOE, ENABLE);
    GPIO_InitStructure.GPIO_Pin    = GPIO_Pin_4 | GPIO_Pin_5;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
    GPIO_Init (GPIOE, &GPIO_InitStructure) ;
}

void EXTI9_5_IRQHandler (void)
{
    if (EXTI_GetITStatus (EXTI_Line5) != RESET)
        {
            if (LED0_IsOn())
                GPIOG->BSRR = 0x0C000000     //①
            else
                GPIOG->BSRR = 0x00000C00;    //②
            EXTI_ClearITPendingBit (EXTI_Line5) ;
        }
}

void EXTI4_IRQHandler (void) { }
```

# 附录 3：STM32F407 系列 MCU 相关信息

表 1　STM32F407 的 GPIO_BSRR 寄存器位定义

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BR15 | BR14 | BR13 | BR12 | BR11 | BR10 | BR9 | BR8 | BR7 | BR6 | BR5 | BR4 | BR3 | BR2 | BR1 | BR0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BS15 | BS14 | BS13 | BS12 | BS11 | BS10 | BS9 | BS8 | BS7 | BS6 | BS5 | BS4 | BS3 | BS2 | BS1 | BS0 |

GPIO_BSRR[31:16]　BRy: Port x reset bit y (y=0..15)

BRy 为 1，表示 Port x 的输出数据寄存器（ODRx）的 y 位置 0，即 Port x 的 y 位输出 0

GPIO_BSRR[15:0] BSy: Port x set bit y (y=0..15)

BSy 为 1，表示 Port x 的输出数据寄存器（ODRx）的 y 位置 1，即 Port x 的 y 位输出 1

表 2　STM32F407 系列 MCU 的中断向量表（部分）

| **Position** | **Priority** | **Type of priority** | **Acronym** | **Description** | **Address** |
|---|---|---|---|---|---|
| - | - | - | - | Reserved | 0x0000_0000 |
| - | -3 | fixed | Reset | Reset | 0x0000_0004 |
| | | | | | |
| - | 4 | settable | Debug Monitor | Debug Monitor | 0x0000 0030 |
| - | - | - | - | Reserved | 0x0000 0034 |
| - | 5 | settable | PendSV | Pendable request for system service | 0x0000 0038 |
| - | 6 | settable | SysTick | System tick timer | 0x0000_003C |
| 0 | 7 | settable | WWDG | Window Watchdog interrupt | 0x0000_0040 |
| 1 | 8 | settable | PVG | PVD through EXTI line detection interrupt | 0x0000_0040 |
| 10 | 17 | settable | EXTI4 | EXTI Line0 interrupt | 0x0000_0068 |
| 23 | 30 | settable | EXTI9_5 | EXTI Line[9:5] interrupts | 0x0000_009C |
| 28 | 35 | settable | TIM2 | TIM2 global interrupt | 0x0000_00B0 |
| 37 | 44 | settable | USART1 | USART1 global interrupt | 0x0000_00D4 |
| 38 | 45 | settable | USART2 | USART2 global interrupt | 0x0000_00D8 |
| 39 | 46 | settable | USART3 | USART3 global interrupt | 0x0000_00DC |

# 附录 4：ARM 汇编指令相关信息

**ADD 指令**

加法指令，典型语法是“**ADD Rd, Rn, Rm**”，该指令将 Rn+Rm 的结果存入 Rd。

**LDR 指令**

从存储器读取字数据至寄存器。典型语法为“**LDR Rd, [Rn, #offset]**”，该指令从存储器位置 Rn+offset 读取一个字存入寄存器 Rd。若指定操作数选址方式为前变址寻址（从位置 Rn+offset 读取一个字，更新 Rn 为 Rn+offset），其典型语法为“**LDR Rd, [Rn, #offset]!**”。若指定操作数选址方式为后变址寻址（从位置 Rn 读取一个字，更新 Rn 为 Rn+offset），其典型语法为“**LDR Rd, [Rn], #offset**”。