一、

（10分）在实验课中，使用的芯片型号是STM32F429IGT6，请通过查阅硬件手册了解GPIOB组的10个寄存器的地址。

寄存器名称	偏移地址	寄存器地址（计算值）	功能描述
`GPIOB_MODER`	0x00	**0x4002 0400**	配置引脚模式（输入、输出、复用、模拟）
`GPIOB_OTYPER`	0x04	**0x4002 0404**	配置引脚输出类型（推挽或开漏）
`GPIOB_OSPEEDR`	0x08	**0x4002 0408**	配置引脚输出速度
`GPIOB_PUPDR`	0x0C	**0x4002 040C**	配置引脚上拉/下拉电阻
`GPIOB_IDR`	0x10	**0x4002 0410**	输入数据寄存器，读取引脚电平状态
`GPIOB_ODR`	0x14	**0x4002 0414**	输出数据寄存器，写入以控制引脚电平
`GPIOB_BSRR`	0x18	**0x4002 0418**	位设置/复位寄存器，写入1以设置或复位对应引脚
`GPIOB_LCKR`	0x1C	**0x4002 041C**	配置寄存器锁存
`GPIOB_AFR[0]`	0x20	**0x4002 0420**	复用功能选择寄存器低半部分（引脚0-7）
`GPIOB_AFR[1]`	0x24	**0x4002 0424**	复用功能选择寄存器高半部分（引脚8-15）

二、

（20分）举例说明在你的某个嵌入式设备中如何体现嵌入式微处理器的特点的。

举例：智能手环（列出一种设备给4`）

嵌入式微处理器的特点：实时性强、低功耗、接口丰富、高可靠性、生命周期长、产品系列化。4`

下面的具体说明 2`

实时性强：智能手环需要实时响应用户的操作（如触摸、按键）和传感器数据（如心率变化）。嵌入式微处理器采用实时操作系统（RTOS），能够快速处理中断请求，保证系统的实时性。

低功耗：智能手环需要长时间佩戴，功耗是关键指标。嵌入式微处理器采用低功耗设计，如动态电压调节（DVS）、睡眠模式等技术，降低功耗。

接口丰富：智能手环集成了多种传感器（如心率传感器、加速度计、陀螺仪等），嵌入式微处理器将传感器接口、数据处理、存储等功能集成在一起，减少了外部电路的复杂性。

高可靠性：嵌入式微处理器采用冗余设计、错误检测和纠正（EDAC）等技术，提高了系统的可靠性。 例如，微处理器内置看门狗定时器，当系统出现异常时，能够自动复位，恢复正常运行。

生命周期长：智能手环需要长时间稳定运行。

产品系列化：一种产品通常都有系列化。

三、

（10分）根据你做的实验解释“宿主机/目标机”开发模式

解释

3`**宿主机（Host Machine）**：开发者使用的开发平台，通常是性能强大的通用计算机（如PC、服务器），安装有完整的开发工具链（编译器、调试器、IDE等）。

3`**目标机（Target Machine）**：嵌入式系统实际运行的硬件平台，资源有限（如低功耗微控制器、小型SoC），无法直接运行开发工具。

4`**开发模式**：采用交叉编译，通过宿主机生成目标机的可执行代码，并通过调试接口（如JTAG、SWD）将代码下载到目标机运行，实现跨平台开发。

四、

（20分）请给出求最小公倍数的C语言代码和汇编代码

C 10'

```c
// 求最大公约数（GCD）函数，使用欧几里得算法
int gcd(int a, int b)
    { while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

// 求最小公倍数（LCM）函数
int lcm(int a, int b) {
    return (a * b) / gcd(a, b); // LCM = (a * b) / GCD(a, b)
}
```

汇编 10' 汇编代码只要有核心代码就可以给满分

```
.section .data
prompt:     .asciz "请输入两个正整数："
format_in: .asciz "%d %d"
result_msg: .asciz "最小公倍数

    .section .bss
num1:       .skip 4
num2:       .skip 4
result:     .skip 4

    .section .text
```

```
    .global main
    .extern printf, scanf

main:
    ; 打印提示信息
    ldr r0, =prompt          ; 加载提示信息地址
    bl printf                ; 调用printf输出提示

    ; 读取用户输入
    ldr r0, =format_in       ; 加载输入格式地址
    ldr r1, =num1            ; 加载第一个数的地址
    ldr r2, =num2            ; 加载第二个数的地址
    bl scanf                 ; 调用scanf读取输入

    ; 加载输入的两个数
    ldr r0, =num1            ; r0 = &num1
    ldr r0, [r0]             ; r0 = num1
    ldr r1, =num2            ; r1 = &num2
    ldr r1, [r1]             ; r1 = num2

    ; 计算GCD(r0, r1)
    bl gcd                   ;调用gcd函数， 结果保存在r0

    ; 计算LCM = (num1 * num2) / GCD
    ldr r1, =num1            ; r1 = &num1
    ldr r1, [r1]             ; r1 = num1
    ldr r2, =num2            ; r2 = &num2
    ldr r2, [r2]             ; r2 = num2
    mul r1, r1, r2           ; r1 = num1 * num2
    udiv r2, r1, r0          ; r2 = r1 / r0 (LCM)

    ; 保存结果
    ldr r1, =result          ; r1 = &result
    str r2, [r1]             ; result = LCM

    ; 打印结果
    ldr r0, =result_msg      ; 加载结果消息地址
    ldr r1, =result          ; 加载结果地址
    ldr r1, [r1]             ; r1 = result
    bl printf                ; 调用printf输出结果

    ; 返回
    mov r0, #0               ; 返回0
    bx lr                    ; 返回调用者

; 计算最大公约数 (GCD) 的函数
gcd:
    cmp r1, #0               ; 比较r1是否为0
    beq gcd_done             ; 如果r1 == 0， 跳转到gcd_done
    mov r2, r1               ; r2 = r1
    udiv r1, r0, r1          ; r1 = r0 / r1
```

```
63      mls r0, r1, r2, r0      ; r0 = r0 - r1 * r2 (r0 = r0 % r1)
64      b gcd                   ; 递归调用gcd
65  gcd_done:
66      bx lr                   ; 返回调用者
```

五、

（10分）解释ADC的工作原理。

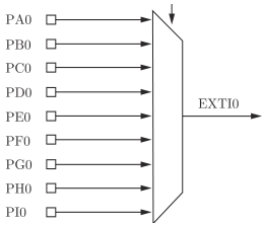
工作流程 （取样、保持、量化和编码）

1. 信号输入 2.5‘ ：模拟信号（如传感器输出）通过输入端进入ADC。
2. 采样保持（Sample and Hold）2.5‘： 在采样瞬间，ADC会“冻结”信号值，以便在量化过程中保持稳定。
3. 量化与编码 2.5‘：将采样值映射到最近的量化级别，并转换为数字代码。
4. 数字输出 2.5‘： 最终的数字信号通过总线输出，供后续处理（如微控制器、DSP等）。

六、

（10分）解释GPIO作为外部中断是如何映射到外部中断/事件线上的。

GPIO一个端口的引脚GPIOx.0~GPIOx.15分别对应外部中断/事件线0~15，比如对应到EXTI0线上的引脚如下图所示。



外部中断控制器（EXTI）用于管理微处理器中的外部中断线，Cortex-M4中的EXTI可支持23个外部中断，其中EXTI线0~15对应GPIO的输入中断。

GPIO作为外部中断的入口，它的端口引脚$GPIO_x0$~$GPIO_x15$（其中x=A/B/C/D/E/F/G/H/I）分别对应外部中断0~15，因此每个中断线对应9个端口，以中断事件线0为例，它对应了$GPIO_A0$~$GPIO_I0$

然而中断/事件线每次只能连接到1个引脚上，因此需要配置来决定中断/事件线具体连接到哪个引脚上。

七、

（10分）请解释I2C是如何竞争总线的?

采用“线与”的总裁方式，下面再简单介绍一下线与的基本含义，也可以通过例子介绍。

线与：只要有一个设备输出低电平（0），信号线就会被拉低，表现为低电平；只有当所有设备都释放总线（输出高阻态），信号线才会被上拉电阻拉高，表现为高电平。

在I2C总线中，多个主机设备通过“线与”逻辑竞争总线控制权。当多个主机同时尝试发送数据时，总线上的电平由所有主机的输出共同决定，实现仲裁。

例：

假设场景：

两个主机设备（Host A和Host B）同时尝试向从设备发送数据。

仲裁过程：

起始条件：

Host A和Host B同时发送起始条件（START condition），即SDA线从高电平变为低电平，同时SCL线保持高电平。

数据位发送：

每个主机在SCL高电平期间发送数据位，并同时读取SDA线的电平。

冲突检测：

如果Host A发送的位为0，而Host B发送的位为1，则SDA线会被Host A拉低为0。

Host B检测到SDA线电平与自己发送的位不一致，立即退出仲裁。

仲裁结果：

Host A获得总线控制权，继续完成数据传输。

八、

（10分）通过这一个的学习，你对“嵌入式系统设计”课程有什么感受？你对后期的授课方式和学习方式有什么建议？（字数低于150字的将扣分）

只要超过150字就给满分，我都看过，应该每人都超过了，这样都可以给满分。