# 《嵌入式系统基础与实践》习题参考答案

## 第 1 章习题

## 习题 1

1-1 什么是嵌入式系统？

由于嵌入式技术及其应用领域的不断发展，行业内对于嵌入式系统并没有一个统一的定义，国外把嵌入式系统定义为：嵌入式系统是控制、监视或辅助操作设备、机器和车间运行的装置。可以看出，嵌入式系统之所以称为“系统”，是因为该系统中不仅有被控对象，而且包括控制系统所用到的硬件和软件。

国内行业普遍认同的定义为：以应用为中心、以计算机技术为基础，软、硬件可剪裁，适合应用系统对功能、可靠性、成本、体积、功耗严格要求的专用计算机系统，实现对其他设备的控制、监视或管理等功能。也就是说，嵌入式系统的开发是基于功能与性能裁剪的，不需要或用不到的功能可不作考虑，功耗和体积也是嵌入式系统要考虑的主要问题。

1-2 嵌入式系统有哪些主要的部分组成？

嵌入式系统一般由嵌入式微处理器、外围硬件设备、嵌入式操作系统以及应用程序等 4 部分组成，并且分为 4 个层次：硬件层、中间层、软件层和应用层。

1-3 嵌入式系统主要有哪些特点？

嵌入式系统具有嵌入性、专用性以及软硬件可裁剪的特性。专用性主要体现在嵌入式系统的功能是针对特定的应用场合、特殊应用领域，内部采用的也是专用的嵌入式处理器，嵌入式系统对用户是透明的，是用户“看不见”的专用计算机系统。嵌入到应用对象系统中，对体积、环境、功耗等要求苛刻。对软件、硬件进行裁剪的专用计算机系统，实现对象系统的智能化控制。

1-4 简述 MPU、MCU、FPGA 和 DSP 的区别与联系？

1. MCU：Micro-Control Unit，嵌入式微控制器，俗称单片机。它将 CPU、随机存储器 RAM、只读存储器 ROM、I/O、中断系统、定时器/计时器、各种功能外设等资源集成到一个芯片上的微型计算机系统，故称单片机（Single Chip Microcomputer），其只需很少的外围电路或者不需要外围电路，直接供电即可工作，是一个芯片级的计算机。单片机是嵌入式微控制器的典型代表，早期被称为微控制器的代名词，微控制器的最大特点就是单片化，体积大大减小，从而降低了功耗和成本，可靠性高。

2. MPU：Micro Processor Unit，嵌入式微处理器。MPU 是由通用计算机中的 CPU（微处理器）演变而来的，可以理解为增强版的 CPU，即不带外围功能器件，因此嵌入式微处理器系统需要在 MPU 的基础上添加 RAM、ROM、Flash、电源等外围电路，以及 USB、LCD、键盘等外部设备构成高性能的计算机系统。特征是具有 32 位以上的处理器，性能高，可靠性高，价格相对较高。

3. FPGA（Field-Programmable Gate Array，现场可编程门阵列）的内部包含了大量的逻辑单元、丰富的触发器资源和 I/O 引脚，借助于硬件描述语言或其他方式，用户可以根据设计需求修改其内部硬件

结构，从而实现系统功能。区别于 MCU、DSP 等硬件固定，只能通过修改软件实现功能的方式，FPGA 是用硬件来实现算法或控制的。

4. DSP 处理器是专门用于信号处理方面的处理器，其在系统结构和指令算法方面进行了特殊设计，具有较高的编译效率和执行速度，广泛应用于数字滤波，FFT、谱分析等领域。

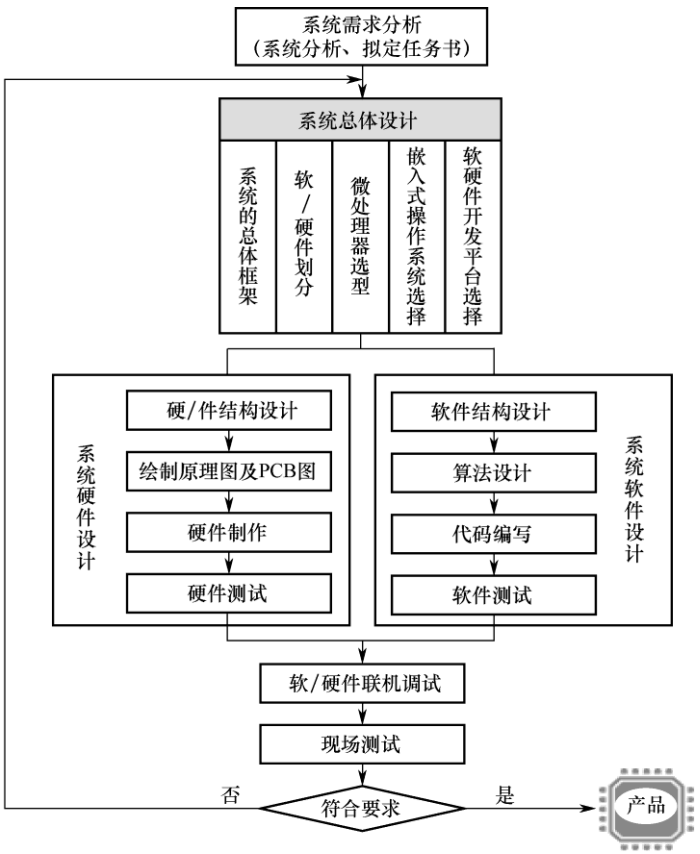
DSP 具有以下两种含义。

（1）用作 Digital Signal Processing（数字信号处理，简称 DSP），是用数值计算的方式对信号进行加工处理的理论和技术。信息化的基础是数字化，对数字信号进行处理就用到数字信号处理技术。

（2）Digital Signal Processor（数字信号处理器，简称 DSP），是一种专用于数字信号处理领域的微处理器芯片，将数字信号处理算法用具体的器件实现。嵌入式微处理器在控制方面具有优势，但对需要进行大量离散时间信号高速实时处理的数字信号处理算法，微处理器无论是在高速实时运算上还是在效率上都无法满足要求，因此，专门针对数字信号处理的 DSP 芯片应运而生。

1-5 简述嵌入式系统的开发流程。

嵌入式系统自身的特性决定了嵌入式系统开发与通用计算机系统的开发有着明显的区别。嵌入式系统开发主要由系统总体开发、嵌入式软/硬件开发、系统测试等三大部分组成，具体又细分为系统需求分析、系统总体设计、系统硬件设计、系统软件设计、系统软/硬件测试，各个阶段之间往往要求不断的反复和修改，直至完成最终设计目标。其开发流程如下图所示。




1-6 从功能概述、系统结构组成、功能模块等方面设计实现餐厨垃圾智能监测系统。（或选择一个嵌

入式系统进行案例分析，分别从功能概述、系统结构组成、功能模块等方面进行分析。）

答:功能概述：一个相对完善的智能家居监控系统主要包括以下四个功能:

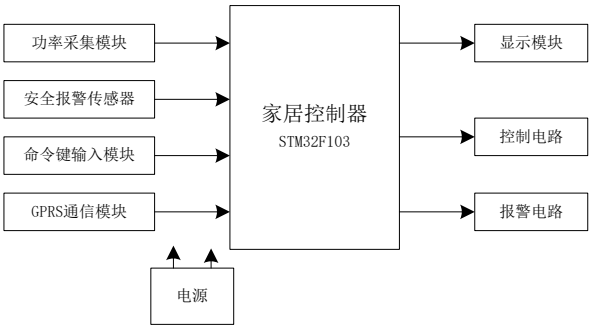
（1）家庭安全监护。家居中的门窗、燃气、火灾等异常状况，应及时将相应警报信息发送到移动终端，帮助用户及时了解家中的安全状况。

（2）家庭环境优化服务。控制家中的温度、湿度等，智能启动加湿器、空调，对家居环境进行智能调节室温和空气湿度，给用户一个舒适的生活环境。

（3）家庭能源管理。远程管理电源、用电设备及异常，能够智能降低功耗，监控用电设备安全。

（4）家居智能控制服务。对家中的所有家电，可通过移动终端实现远程或本地的开关控制。

系统硬件结构框图如下图所示。




各功能模块概述：

（1）通信电路设计。短信监控功能通过 GPRS 模块对短信进行操作实现，采用 AT 指令进行通信操作。本系统中采用基于 USB 的通信模块 SIM908，通过 STM32 的串口进行连接。

（2）显示模块电路。液晶显示屏 TFT-LCD 具有驱动电压低、功耗小、反应速度快、显示画面质量好等优点，本系统采用 LCD 作为显示设备。LCD 的端口包括数据端口和控制端口，其中数据端口与主控芯片的 PB0-PB5 进行连接，控制端口与主控芯片的 PC6-PC9 连接。

（3）数据采集模块电路。温度采集采用数字温度传感器 DS18B20。功率采集电路主要由电流检测电路、电压检测电路、电能计量芯片 ADE7755 及其外围电路组成。家电开断控制电路采用三极管驱动继电器来实现家用电器供电控制。

# 第 2 章 习题

## 习题 2

2-1 ARM Cortex-M3 处理器主要由哪些部分构成？

Cortex-M3 是基于 ARMv7-M 体系结构设计的 32 位处理器内核，Cortex-M3 微处理器主要由两大部分组成：Cortex-M3 内核和调试系统。Cortex-M3 内核主要包括以下 5 部分：

（1）中央处理器核心（Cortex-M3 Core）

（2）嵌套向量中断控制器（NVIC）

（3）系统时钟（SYSTICK）

（4）存储器保护单元（MPU）

（5）总线矩阵（AHB 互连网络）

调试系统主要用于调试和测试，其主要包括：串行线/串口 JTAG 调试端口（SW-DP/J.TAG-DP）、基于 AHB 总线的通用调试接口（AHB-AP）、嵌入式跟踪单元（ETM）和数据观察点触发器（DWT）等，其中两个调试端口提供对系统中（包括处理器寄存器）所有寄存器和存储器的调试访问。

2-2 计算机体系结构主要有冯·诺依曼结构与哈佛结构，请简述其区别？

冯·诺依曼结构也称普林斯顿结构，是一种指令和数据采用二进制表示，且存储在同一个存储器中，并经同一条总线传输的存储器结构，冯·诺依曼结构中指令地址和数据地址指向同一个存储器的不同物理位置，两者统一编址且宽度相同。

哈佛结构是一种将程序指令和数据分开存储的存储器结构，目的是为了打破程序访问时的瓶颈。指令存储器和数据存储器独立编址、独立访问，数据和指令分别存储在两个独立的存储器中，并且使用两条独立的总线分别与 CPU 进行信息交换，数据和指令的存储可以同时进行，执行时可以预先读取下一条指令，容易实现指令流水。

冯·诺依曼结构多用于桌面处理器和移动处理器，哈佛结构主要用于嵌入式系统领域。与冯·诺依曼结构处理器比较，哈佛结构处理器有以下明显的特点。

（1）程序存储和数据存储相互独立，使用两个独立的存储器模块，分别存储指令和数据，每个存储模块都不允许指令和数据并存。

（2）使用独立的两条总线，分别作为 CPU 与每个存储器之间的专用通信路径，而这两条总线之间毫无关联。

（3）CPU 采用不同的指令来访问程序和数据。

2-3 现有一个 32 位的数据 0x12345678，存放在内存地址为 0x4000～0x4003 处，若按小端模式进行存放，则其最低字节数据 0x78 存放在内存地址___0x4000____处；若以大端模式存储，则最高字节数据 0x12 存放在内存__0x4000___处。

2-4 现采用 3 级流水线结构完成一条指令的取指、译码和执行，若这三部分的时间分别是 $t_{取指}$=2ns，$t_{分析}$=2ns，$t_{执行}$=1ns，则完成 100 条指令需要___203___ns。

2-5 Cortex-M3 内核能够寻址的最大空间为__4GB__，寻址范围为__0x00000000～0xFFFFFFFF __。

2-6 简述基于 ARM 内核的嵌入式领域常见的半导体公司及其代表性产品。

ARM 公司不生产芯片，只是将设计出的 IP Core 授权给各大半导体厂商，由各大半导体厂商根据自己的技术优势，生产出针对各应用领域的性能各异的芯片。如 ST 公司的 STM32、NXP 公司的 LPC2000、Freescale 的 Kinets 系列等，下表列出了嵌入式领域部分半导体公司及其 ARM 架构的代表性产品。

| 半导体公司 | 代表性产品（内核架构） |
| --- | --- |
| TI（德州仪器） | LM3Sxxxx 系列（Cortex-M3）<br>LM4Fxxxx 系列（Cortex-M4） |
| NXP（恩智浦） | LPC1800（Cortex-M3）<br>LPC4300（Cortex-M4） |
| Atmel（爱特梅尔） | SAM3S 系列（Cortex-M3）<br>SAM4S 系列（Cortex-M4） |
| Freescale（飞思卡尔） | Kinetis X 系列（Cortex-M4） |
| ST（意法半导体） | STM32、STM8 |

2-7 简述 ARM Cortex-M3 内核和 ARM Cortex-M4 内核的区别。

Cortex-M3 基于 ARMv7-M 架构和 Cortex-M4 基于 ARMv7-ME 架构，都具有高性能，且都是为微控制器应用设计的，与 Cortex-M3 相比，Cortex-M4 提供了更丰富的指令集，还有一个可选的 MPU 单元。在指令集方面，M4 比 M3 多了浮点运算指令、单周期的 MAC 指令、SIMD 指令和更多的饱和指令，具有浮点运算能力以及增强的 DSP 处理指令，其可以执行一些数字信号处理程序。

2-8 简述 STM32 的时钟系统。该系统有哪几个时钟源，各自的时钟频率是多少？

STM32 的时钟系统由 RCC（Reset and Clock Control，复位与时钟控制器）产生，用来为系统和各种外设提供所需的时钟频率，以确定各外设的工作速度。

根据 ST 官方用户手册，STM32 具有以下 5 个时钟源。

HSI：High Speed Internal，高速内部时钟，由内部 8MHz 的 RC 振荡器生成，可作为系统时钟或经 2 分频后作为 PLL 输入。特点：时钟频率精度差，不稳定。

HSE：High Speed External，高速外部时钟，可外接一个外部时钟源，或者通过 OSC_IN 和 OSC_OUT 引脚外接晶振，允许外接的晶振频率范围为 4～16MHz，通常使用 8MHz。特点：精度高，稳定。

LSI：Low Speed Internal，低速内部时钟，由内部 RC 振荡器产生，频率约 40kHz，主要为独立看门狗（IWDG）和自动唤醒单元提供时钟。

LSE：Low Speed External，低速外部时钟，通过 OSC32_IN 和 OSC32_OUT 引脚外接频率为 32.768kHz 的晶振，为 RTC（Real-Time Clock，实时时钟部件）提供低速高精度的时钟源。

PLL：Phase Locked Loop，锁相环，是一种反馈控制电路，用于外部输入时钟信号与内部振荡信号的同步（频率和相位相同），以确保输出频率的稳定。另一方面，也可用于倍频 HSI 或 HSE，其时钟输入源可选择为 HSI/2、HSI 或者 HSE，倍频可选择为 2～16 倍，但其输出频率最大不得超过 72MHz。

2-9 STM32 的存储空间分为哪几个部分？每部分的地址范围是多少？

STM32 共有 4GB 大小的存储器空间，划分为 8 个 512MB 等份区域，分别为：

1. 代码区（0x0000_0000 - 0x1FFF_FFFF）。该区是可以执行指令的，缓存属性为 WT（“写通”，Write Through），即不可以缓存。此区亦可写数据。在此区上的数据操作是通过数据总线接口的（读数据

使用 D‐Code，写数据使用 System），且在此区上的写操作是缓冲的。

2. SRAM 区（0x2000_0000 – 0x3FFF_FFFF）。此区用于片内 SRAM，写操作是缓冲的，并且可以选择 WB‐WA(Write Back, Write Allocated)缓存属性。此区亦可以执行指令，以允许把代码拷贝到内存中执行——常用于固件升级等维护工作。

3. 片上外设区(0x4000_0000 – 0x5FFF_FFFF)。该区用于片上外设，因此是不可缓存的，也不可以在此区执行指令（这也称为 eXecute Never, XN。ARM 的参考手册大量使用此术语）。

4. 外部 RAM 区的前半段（0x6000_0000 – 0x7FFF_FFFF）。该区用于片外 RAM，可缓存（缓存属性为 WB‐WA），并且可以执行指令。

5. 外部 RAM 区的后半段（0x8000_0000 – 0x9FFF_FFFF）。除了不可缓存(WT)外，同前半段。

6. 外部外设区的前半段(0xA000_0000 – 0xBFFF_FFFF)。用于片外外设的寄存器，也用于多核系统中的共享内存（需要严格按顺序操作，即不可缓冲）。该区也是个不可执行区。

7. 外部外设区的后半段(0xC000_0000 – 0xDFFF_FFFF)。目前与前半段的功能完全一致。

8. 系统区(0xE000_0000 – 0xFFFF_FFFF)。此区是私有外设和供应商指定功能区。此区不可执行代码。系统区涉及到很多关键部位，因此访问都是严格序列化的（不可缓存，不可缓冲）。而供应商指定功能区则是可以缓存和缓冲的。

---参考《Cortex M3 权威指南(中文)》

2-10 ARM Cortex-M3 微处理器采用存储器地址与 I/O 设备地址统一编址的方式，通过 不同的地址代码 来区分内存单元和 I/O 设备。

# 第 3 章习题

## 习题 3

3-1 根据 STM32 的命名规则，芯片型号 STM32F103ZET6 中各符号代表什么含义？该芯片主要有哪些特征？

STM32 代表 ARM Cortex-M3 内核的 32 位微控制器，ST 为意法半导体，M 为微控制器的缩写，32 代表是 32 位的芯片，F 代表芯片的系列为基础级，103 为增强型系列为主流入门级，Z 表示此芯片有 144 管脚，E 表示具有 512KB 的 Flash，T 表示此芯片采用 LQFP 封装，6 表示芯片工作温度范围为 40～+85℃。

STM32F103ZET6 是一款基于 ARM Cortex-M3 内核的微控制器，主频为 72MHz，该芯片具有 512KB 的 Flash 存储器和 64KB 的 SRAM，支持多种通信接口，如 SPI、I2C、USART，包含电机控制、USB 和 CAN 等接口，具有高性能、低功耗、丰富的外设和强大的处理能力，广泛应用于工业控制、汽车电子、智能家居等领域。

3-2 简述 STM32 微控制器三种开发模式的区别与联系。

ST 公司为 STM32 提供了以下三种开发模式。

（1）寄存器开发方式。

（2）标准外设库开发方式。

（3）HAL（Hardware Abstraction Layer，硬件抽象层）库开发方式。

标准外设库（Standard Peripherals Library）简称标准库，其开发方式是将底层寄存器的操作进行了统一封装，包括所有标准器件外设的驱动器，采用 C 语言实现，开发人员只需要熟悉并调用相应的 API（Application Programming Interface，应用程序编程接口）函数，即可实现对相关外设的驱动操作。由于标准库只是将一些基本的寄存器操作封装成了 C 函数，因此使用标准库可以很简单地跟踪到底层寄存器，这对学习和掌握硬件底层相关的内容十分有利。需要注意的是，标准库是针对某一系列的芯片开发的，其跨平台移植性很差。

HAL 库开发是与 STM32CubeMX 软件（STM32CubeMX 是一个配置 STM32 代码的工具）配套使用的，它把底层硬件相关的内容封装起来并进行抽象，通过图形化的操作方式自动生成相关外设的驱动代码，简单易用，但若要从 HAL 库跟踪代码并理解其架构却很难。

以上三种开发方式各有利弊，寄存器开发需要对底层寄存器十分熟悉，开发过程慢，但代码量小，代码执行效率高。基于库的开发方式已成为当今嵌入式开发的主流开发方式，也是各大半导体公司和嵌入式行业大力推崇的一种方式。

3-3 什么是 CMSIS？ARM 公司为什么要制定 CMSIS 标准？

CMSIS 标准：Cortex Microcontroller Software Interface Standard，Cortex 微控制器软件接口标准。由于 ARM 公司产品是由众多合作公司形成的生态半导体产业链，在这条产业链上，ARM 公司只负责芯片内核的架构设计，而半导体厂商则根据 ARM 公司提供的内核标准设计各自的芯片，所以任何一个基于 Cortex 生产的芯片其内核结构都是一样的，区别在于存储器容量、片上外设、I/O 以及其他模块的设计。为了解决不同芯片厂商生产的基于 Cortex 内核的微处理器在软件上的兼容问题，ARM 公司与众多芯片和软件厂商共同制定了 CMSIS 标准，旨在将所有基于 Cortex 内核产品的软件接口标准化。

3-4 STM32 标准外设库中的 startup_stm32f10x_hd.s 文件主要有哪些功能？

startup_stm32f10x_hd.s 文件是 STM32F10x 系列芯片的启动文件，负责初始化系统硬件和中断向量表配置，并提供主函数入口点，它为系统启动提供了必要的支持和准备工作。主要有初始化堆栈指针、设置中断向量表的入口地址、配置外部 SRAM 作为数据存储器、配置系统时钟、初始化程序计数器指针 PC 等功能。

3-5 下列程序为 STM32 标准外设库 v3.5.0 的时钟初始化函数，分析此函数，补充（1）～（6）相应的注释。

```
void RCC_Configuration(void)//时钟初始化函数
{
  ErrorStatus HSEStartUpStatus;//等待时钟稳定
  RCC_DeInit();（1）
  RCC_HSEConfig(RCC_HSE_ON);（2）
  HSEStartUpStatus=RCC_WaitForHSEStartUp();//等待外部高速时钟晶振就绪
  If(HSEStartUpStatus==SUCCESS)
```

```
    {
      FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);
      FLASH_SetLatency(FLASH_Latency_2);
      RCC_HCLKConfig(RCC_SYSCLK_Div1); //AHB 使用系统时钟
      RCC_PCLK2Config(RCC_HCLK_Div1);（3）
      RCC_PCLK1Config(RCC_HCLK_Div2);（4）
      RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);（5）
      RCC_PLLCmd(ENABLE); //启动 PLL
      while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY)==RESET){}//等待 PLL 启动
      RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);（6）
      while(RCC_GetSYSCLKSource()!=0x08){}//等待系统时钟源的启动
    }
    … …
  }
```

答：

（1）将 RCC 寄存器重新设置为默认值

（2） 打开外部高速时钟晶振 HSE

（3）设置高速 APB2 时钟为 HCLK，即不分频

（4） 设置低速 APB1 时钟为 HCLK 的一半，即二分频

（5）配置系统时钟 PLLCLK = 8MHz * 9 = 72 MHz

（6）将 PLL 设置为系统时钟源

3-6 STM32F103 系列产品中的高密度、中密度和低密度是按什么划分的？

STM32F1 系列产品中，每个系列又有多种型号供用户选择，各个子系列按闪存容量的大小又分为小容量、中容量和大容量产品。

（1）低密度（LD，Low-Density）：闪存容量为 16～32KB 的小容量产品。

（2）中密度（MD，Medium-Density）：闪存容量为 64～128KB 的中容量产品。

（3）高密度（HD，High-Density）：闪存容量为 256～512KB 的大容量产品。

（4）超密度（XL，XL-Density）：闪存容量为 768KB～1MB 的大容量产品。

3-7 简述嵌入式系统的开发方式。嵌入式系统开发由哪几部分组成？嵌入式系统软件开发一般分为哪几个阶段？

由于软硬件资源受限，嵌入式系统的开发不能在其平台上直接进行，因此一般采用交叉开发方式实现。所谓交叉开发，就是利用由宿主机（Host）、目标机（Target）和交叉连接宿主机与目标机的工具（调试、下载工具、仿真器）三部分组成的交叉开发系统进行嵌入式系统开发。

嵌入式系统软件开发一般分为 4 个阶段：代码编辑、程序编译、链接、调试和下载，每个阶段都需要使用不同的工具来完成，这些工具都需要运行在宿主机上。

# 第 4 章习题

## 习题 4

4-1 简述 STM32 的最小系统，复位电路有哪几种方式？

最小系统也称为单片机最小应用系统，是指用最少的元件组成单片机可以工作的系统。对 STM32 微控制器来说，典型的最小系统一般包括电源电路、晶振电路、复位电路和程序下载电路，往往还包含 LED 指示电路。

STM32 复位电路有以下两种复位方式。

（1）上电复位：在上电瞬间，由于电容两端的电压不能突变，因此 RESET 出现短暂低电平，芯片自动复位，然后电容充电，充电时间由电阻和电容共同决定，系统完成复位，之后，单片机开始正常工作，电容相当于断路，复位端 RESET 一直为高电平。

（2）手动复位：按键 S1 按下时，RESET 与地连通，产生低电平，实现复位。

4-2 STM32 的程序下载方式有哪些？若 STM32F103 系列微控制器从 Flash 的起始地址 0x80000000 处启动，则 BOOT0 和 BOOT1 如何设置？

STM32 支持 JTAG（Joint Test Action Group）调试接口和 SWD（Serial Wire Debug，串行单总线调试）调试接口, 还可以采用串口方式进行下载，通常采用 USB 转串口芯片 CH340G 将上位机的 USB 映射为串口。

当 BOOT0=0，BOOT1=1 时，STM32 从用户 Flash 的起始地址 0x80000000 处启动运行代码。

4-3 简述嵌入式 C 语言中的 const、static、volatile、extern 关键字的区别与应用场合。

const 关键字用于定义只读的变量，其值在编译时不能被改变，注意，const 关键字定义的是变量而不是常量。使用 const 关键字是为了在编译时防止变量的值被误修改，提高程序的安全性和可靠性，一般放在头文件中或者文件的开始部分。

static 关键字可以用来修饰变量，使用 static 关键字修饰的变量，称为静态变量。若在该全局变量前加上关键字 static，则该全局变量被定义成一个静态全局变量，其作用范围只在定义该变量的源文件内有效，其他源文件不能引用该全局变量，这样就避免了在其他源文件中因引用相同名字的变量而引发的错误，有利于模块化程序设计。

volatile 就是不让编译器进行优化，即每次读取或者修改值时，都必须重新从内存中读取或修改，而不是使用保存在寄存器的备份。

extern 关键字用于指明此函数或变量的定义在其他文件中，提示编译器遇到此函数或变量时到其他模块中寻找其定义。这样，extern 声明的函数或变量就可以在本模块或其他模块中使用，因此，使用 extern 是一个声明而不是重新定义。

4-4 简述 STM32 的 HAL 库的回调函数的形式和作用。

回调函数原型如下：

__weak void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin);

该回调函数的函数体基本上是空的，其默认实现声明为__weak 属性，weak 关键字表示弱声明，其作用在于一旦声明了用户自己编写的同名函数，链接器就会替换掉原来用__weak 声明的函数，链接用户编写的同名函数。所以，STM32 的 HAL 库的回调函数的函数体需要用户自己编写，其实质是通过中断处理函数 void HAL_GPIO_EXTI_IRQHandler (uint16_t GPIO_Pin)调用回调函数来实现中断服务功能。

4-5 STM32 标准外设库和 HAL 库是通过什么来实现内存空间的定义和操作的？

利用 C 语言的宏定义和指针来封装总线和外设的基地址，通过指针可以对内存空间进行操作。

4-6 请简述 STM32 嵌入式开发中常用的数据类型及其表示的数据范围。

STM32 采用 CMSIS 软件标准 C99 标准中定义的数据类型，ST 公司也为开发人员提供了基于 C 语言的标准外设库，其定义的数据类型如下表所示。

| C 语言的数据类型 | STM32 对应的数据类型 | 功能描述 |
|---|---|---|
| unsigned char | uint8_t | 8 位无符号数据（0～255） |
| unsigned short int | uint16_t | 16 位无符号数据（0～65535） |
| unsigned int | uint32_t | 32 位无符号数据（$0\sim2^{32}-1$） |
| unsigned long long | uint64_t | 64 位无符号数据（$0\sim2^{64}-1$） |
| signed char | int8_t | 8 位有符号数据（−128～+127） |
| signed short int | int16_t | 16 位有符号数据（−32768～+32767） |
| signed int | int32_t | 32 位有符号数据（$-2^{31}\sim2^{31}-1$） |
| signed long long | int64_t | 64 位有符号数据（$-2^{63}\sim2^{63}-1$） |

4-7 STM32 开发中使用了很多英文缩写，请解释下列英文缩写的含义。

（1）RCC （2）RTC（3）EXTI（4）PWM（5）PLL （6）OSC （7）NVIC （8）ISP （9）UART （10）SPI （11）EIA （12）API （13）AFIO （14）IRQ （15）BKP （16）APB

（1）RCC：Reset and Clock Control，即复位和时钟控制。

（2）RTC ：Real Time Clock，实时时钟。

（3）EXTI：External Interrupt/Event Controller，外部中断/事件控制器

（4）PWM：Pulse Width Modulation，简称脉宽调制

（5）PLL ：Phase Locked Loop，锁相环

（6）OSC ：Oscillator，简称 OSC

（7）NVIC ：Nested Vectored Interrupt Controller，嵌套向量中断控制器

（8）ISP ：In-System Programming，在系统可编程（在线编程）

（9）UART：Universal Asynchronous Receiver/Transmitter，通用异步收发器

（10）SPI：Serial Peripheral Interface，串行外设接口

（11）EIA ： Electronics Industry Association，电子工业协会

（12）API ： Application Programming Interface，应用程序编程接口

（13）AFIO ： Alternate Function Input/Output，复用 I/O 端口

（14）IRQ ： Interrupt ReQuest，中断请求

（15）BKP ： BACKUP，备份寄存器

（16）APB：Advanced Peripheral Bus，先进外设总线，

4-8 嵌入式开发中大量用到宏定义#define，简述#define 与 typedef 的区别。

#define 是 C 语言的预处理命令，它用于宏定义，用来将一个标识符定义为一个字符串，该标识符称为宏名，被定义的字符串称为替换文本，采用宏定义的目的主要是方便程序编写，一般放在源文件的前面，称为预处理部分。

typedef 用于为复杂的声明定义一个简单的别名，它不是一个真正意义上的新类型。在编程中使用 typedef 的目的一般有两个：①为变量起一个容易记忆且意义明确的新名称；②简化一些比较复杂的类型声明。

#define 与 typedef 的区别为：typedef 是在编译阶段处理的，具有类型检查的功能，而 #define 是在预处理阶段处理的，即在编译前，只进行简单的字符串替换，不进行任何检查。

# 第 5 章习题

# 习题 5

5-1 简述 I/O 引脚的通用功能和复用功能。

GPIO（General Purpose Input/Output，通用输入/输出）是微控制器最基本的片上外设，微控制器通过 GPIO 实现与外界的信息交换。

STM32F103 系列微控制器最多有 144 个引脚，其中的 GPIO 引脚除作为通用输入/输出引脚使用外，为减少芯片引脚数量，提高引脚利用率，大多数引脚还通过复用技术兼具其他专用功能，即一个引脚可以作为多个外设引脚使用，称为复用 I/O 端口（Alternate Function I/O，AFIO），但某一时刻一个引脚只能使用复用功能中的一个。

当 I/O 引脚用作复用功能时，可选择复用推挽输出模式或复用开漏输出模式，在选择复用开漏输出模式时，需外接上拉电阻。更多复用功能请查阅相关芯片数据手册。

5-2 通过阅读 STM32 的标准外设库和 HAL 库的 GPIO 输入/输出函数源代码，可以看出其实质是通过操作什么来实现的？

通过操作相关寄存器来是实现的。

5-3 简述 STM32 推挽输出模式和开漏输出模式的区别及应用场合。

在推挽输出模式下，当 I/O 引脚输出高电平时，P-MOS 导通。当 I/O 引脚输出低电平时，N-MOS 导通。这样两个 MOS 管轮流导通，其负载能力和开关速度得到提高。使用推挽输出模式的目的是增大输出电流，即提高输出引脚的驱动能力，提高电路的负载能力和开关的速度。

开漏输出模式下，I/O 引脚只能输出低电平，即当输出寄存器输出为低电平时，N-MOS 导通；当输出寄存器输出为高电平时，N-MOS 截止，此时无论写入什么数据，P-MOS 都截止，相当于断开了 P-MOS 管，输出为高阻状态。若要输出高电平，则需外接电阻，所接的电阻称为上拉电阻，此时输出电平取决于该引脚外接的上拉电阻及外部电源电压情况。因此，开漏输出模式可以用来匹配电平，适用于电平不匹配的场合。

5-4 STM32 的 HAL 库和标准外设库中定义了很多布尔类型的常量，请简述下列常量的含义。（1）SET（2）RESET（3）ENABLE（4）DISABLE

（1）SET：置位，将某一位设置为 1。

（2）RESET：复位，将某一位设置为 0。

（3）ENABLE：允许，使能

（4）DISABLE：关闭，禁止

5-5 编写程序：控制多个 LED 灯全亮函数（void LED_On_All(void)）、LED 灯全熄灭函数（void LED_Off_All(void)）、控制某个 LED 灯亮函数（void LED_On(uint16_t led_unmber)）及控制某个 LED 灯熄灭函数（void LED_Off(uint16_t led_unmber)）。

# 第 6 章习题

# 习题 6

6-1 简述中断的处理流程。

一个完整的中断处理过程分为 4 个步骤：中断请求、中断响应、中断服务和中断返回。

6-2 中断和异常有什么不同？ARM Cortex-M3 内核的 NVIC 支持多少种异常和中断？

单片机的中断有两类：外部中断和内部中断。由外部原因或事件导致的中断称为外部中断，即可以人工控制的中断；由自身因素导致的中断称为内部中断（Cortex-M3 称为异常）。

ARM 公司设计的 Cortex-M3 内核可支持 256 种中断，包括 15 个内核中断和 240 个外部中断，并具有 256 级的可编程中断优先级设置，即除 3 个固定的高优先级（Reset、NMI、硬件失效）外，其他中断和异常的优先级是可以由用户进行设置的。但使用 Cortex-M3 内核的芯片制造商（如 ST、NXP 公司等）不需要用到那么多中断，故可以对中断进行精简。

6-3 简述 STM32 的中断和异常。

STM32 通过其内部的 NVIC（Nested Vectored Interrupt Controller，嵌套向量中断控制器）来进行管理和配置，实现中断通道的设置、优先级分配及中断使能等功能。

STM32F10x 系列产品有 84 个中断通道，包括 16 个内核中断和 68 个可屏蔽中断，具体到 STM32F103 系列芯片只有 60 个可屏蔽中断，STM32F107 系列有 68 个可屏蔽中断。

6-4 已知 3 个中断：中断 3（RTC 中断）、中断 6（外部中断 0）和中断 7（外部中断 1）。其优先级分别为：中断 3（RTC 中断）的抢占优先级为 2，响应优先级为 1；中断 6（外部中断 0）的抢占优先级为 3，响应优先级为 0；中断 7（外部中断 1）的抢占优先级为 2，响应优先级为 0。

假定设置中断优先级组为 2，请写出这 3 个中断的中断优先级顺序。

答： 中断优先级顺序：中断 7 > 中断 3> 中断 6

6-5 STM32 的外部中断 EXTI10-15 共用一个中断服务函数 void EXTI15_10_IRQHandler(void)，若此时 PA10、PA11、PA13、PA15 和 PB15 同时产生中断，则实际开发中应如何区分和处理来自这 5 个引脚的外部中断？

GPIO 的中断是以组为单位的，同组的外部中断共用一条外部中断线。STM32 的外部中断 EXTI10-15 共用一个中断服务函数，实际应用中，可以通过判断语句判断触发中断来自哪个具体的引脚来实现不同的中断处理。

6-6 编写程序：通过两个按键（按键 1 和按键 2）控制两个 LED 灯（LED1 和 LED2）循环点亮，要求按键 2 的按键中断能中断按键 1 的按键中断。

# 第 7 章习题

## 习题 7

7-1 在数字通信中，比特率和波特率有什么区别和联系？

波特率是指每秒钟传输的二进制位数，单位为比特每秒（bit/s，bps），是衡量串行数据传输速度快慢的指标。波特率决定了异步串行通信中每位数据占用的时间。如波特率为 115200bit/s，表示每秒传输 115200 位二进制数据，每位数据在数据线上持续的时间约为 1/115200≈8.68us。

通信时，数据是逐位传送的，而 1 个字符往往由若干位组成，因此每秒所传输的字符数（字符速率）和波特率是两个概念，比特率（Bit rate）是单位时间内传输或处理的比特的位数，单位为“位每秒”（ bit/s, b/s），也写作 bps(bit per second）。在异步串行通信中，所说的传输速率是指波特率，而不是字符速率，两者关系是：波特率=字符速率×每个字符包含的位数。

7-2 简述 RS232 的接口组成及其电平标准。

RS-232 标准主要规定了通信接口的机械特性、信号用途及信号电平标准等电气特性。通常采用 DB-9（9 针）或 DB-25（25 针）的形式，以 DB-9 最为常见， DB-9 通常采用三线制串口，仅需发送（Tx）、接收（Rx）和地（GND）三条线，即可实现全双工通信，最高传输速率可达 20kbit/s。若需要进行高可靠性传输，则采用硬件流控制，由专用信号线 CTS/RTS（请求发送/清除发送）进行传输，如利用 DB-9 接口中的 4 号、6 号、7 号和 8 号引脚实现数据流控制，通常应用在调制解调器中。

RS-232 的电平标准采用负逻辑，即逻辑 0 为+3V～+15V，逻辑 1 为−3V～−15V，这种方式有利于减少信号衰减，RS-232 的传输距离最远可达 15m。

7-3 UART 数据帧格式由哪几部分组成？

异步串行通信的数据帧由起始位、数据位、校验位、停止位 4 部分组成。

起始位：占一位，位于数据帧的开头，其值为 0，即以逻辑 0 表示传输数据的开始。

数据位：要发送的数据，数据长度可以是 5 位、6 位、7 位或 8 位，低位在前，高位在后，数据通常用 ASCII（American Standard Code for Information Interchange，美国信息交换标准代码）码表示。传输数据时先传送字符的低位，后传送字符的高位，即采用小端方式由最低有效位（Least Significant Bit，LSB）到最高有效位（Most Significant Bit，MSB）一位一位地传输。

校验位：占一位，用于检测数据是否有效，该位为可选项。若校验位为 0，则表示不对数据进行校验。若校验位为“1”，则对数据位进行奇校验或偶校验。设置奇偶校验位是为了提高数据传输的准确率。

停止位：一帧传送结束的标志，根据实际情况而定，可以是 1 位、1.5 位或 2 位。

空闲位：数据传输完毕，数据帧之间用 1 表示空闲位，即用 1 表示当前线路上没有数据传输。

7-4 简述 STM32 引脚的复用功能和重映射功能；简述 STM32 的 USARTx 的各端口引脚的复用和重映射情况。

复用功能：STM32 微控制器的 I/O 引脚通过一个复用器连接到板载外设/模块，该复用器一次仅允许一个外设的复用功能 (AF) 连接到 I/O 引脚。这样可以确保共用同一个 I/O 引脚的外设之间不会发生冲突。STM32 所有内置外设的外部引脚都是与标准 GPIO 引脚复用的，如果有多个复用功能模块对应同一个引脚，只能使能其中之一，其它模块保持非使能状态。

重映射功能：为了使不同的器件封装的外设 I/O 功能数量达到最优，可以把一些复用功能重新映射到其他的引脚上。STM32 中有许多的内置外设的输入、输出引脚都具有重映射（Remap）的功能。

STM32 参考手册的通用 I/O 和复用 I/O 章节中对 USART 复用功能重映射进行了说明，比如，USART1 端口的 Tx 引脚默认情况下（USART1_REMAP=0）复用 PA9 引脚，USART1 端口的 Rx 引脚复用 PA10 的引脚，若此时 PA9 和 PA10 引脚已经连接了其他设备，则 USART1 端口引脚需要进行重新映射，设置 USART1_REMAP=1，将 USART1_TX 引脚映射到 PB6 引脚，USART1_RX 引脚映射到 PB7 引脚。

7-5 编写程序：基于 HAL 库利用 STM32F103 系列微控制器的 USART4 和 USART5 实现异步串行通信的收发功能。

7-6 编写程序：实现可接收任意字节的串口通信程序。

7-7 项目实训：串口数据记录仪，将通过串口的数据存储在 SD 卡上或 AT2400 芯片上。

# 第 8 章习题

# 习题 8

8-1 简述 DMA 的工作原理。

直接存储器存取（Direct Memory Access，DMA）既是一种数据传输技术，又是微控制器的一种片上外设。DMA 传输方式无须 CPU 干预，通过硬件直接实现外设与存储器之间或存储器与存储器之间的数据传输，适用于大批量高速数据传输。相比于中断方式，DMA 方式不需要像中断处理方式那样保留现场、恢复现场，而是由 DMA 控制器直接控制，这样节省了 CPU 资源，提高了传输效率。

8-2 STM32 有几个 DMA？简述各 DMA 通道的 DMA 映射及优先权。

STM32 最多有 2 个 DMA 控制器，DMA1 有 7 个通道，DMA2 有 5 个通道，每个通道都专门用来管理一个或多个外设对存储器访问的请求。

外设（TIMx（x=1,2,3,4）、ADC1、SPI1、IICx（x=1,2）、USARTx（x=1,2,3））产生的 DMA1 请求传送到 DMA1 控制器，且同一时刻只有一个请求有效。外设（TIMx（x=5,6,7,8）、ADC3、SPI/I2S3、USART4、DAC 通道 1、2 和 SDIO）产生 5 个通道的 DMA2 请求传送到 DMA2 控制器，同样在某一时刻只能有一个请求有效，

当有多个 DMA 请求时，DMA 控制器通过内部的仲裁器进行优先权管理。通道的优先权分为 4 个等级：最高优先级（Very High）、高优先级（High）、中等优先级（Medium）和低优先级（Low），高优先级的通道优先获得总线响应。若 2 个请求具有相同的软件优先级，则较小编号的通道比较大编号的通道具有较高的优先权，此外，在大容量产品和互联型产品中，DMA1 控制器拥有高于 DMA2 控制器的优先级。

8-3 STM32 的 DMA 控制器支持哪几种 DMA 传输方式？

STM32 的 DMA 工作模式有两种：循环模式和普通模式。循环模式用于处理一个环形的缓冲区，每轮传输结束时，数据传输的配置会自动更新为初始状态，DMA 传输会连续不断地进行。普通模式在 DMA 传输结束时，DMA 通道被自动关闭。

8-4 STM32 的 DMA 通道发送的数据宽度有 8 位、16 位和 32 位，那么 DMA 通道一次传输的最大数据量为多少？

DMA 传输的数据量是可以通过相关寄存器的设置进行可编程的，最大数据传输量为 65536。

8-5 编写程序：利用串口采用 DMA 方式发送数据到主存，统计发送数据中包含的字符个数。

# 第 9 章习题

## 习题 9

9-1 简述通用定时器、SysTick 定时器和看门狗定时器的区别。

STM32 定时器种类多，且功能强大，主要包括 1 个 SysTick 定时器、1 个实时时钟（RTC）、2 个看门狗（WatchDog）定时器、2 个高级定时器、4 个通用定时器和 2 个基本定时器，这些定时器完全独立、互不干扰，可以同步操作。

STM32 具有 4 个独立的 16 位通用定时器，具有定时、测量输入信号的脉冲长度（输入捕获）、输出所需波形（输出比较、产生 PWM、单脉冲输出等）等功能。

SysTick 系统时钟位于 Cortex-M3 内核中，是一个 24 位的递减计数器，主要用于精确延时，在多任务操作系统中为系统提供时间基准（时基），用于任务切换，为每个任务分配时间片。

看门狗定时器的作用是当微控制器受到外部干扰或程序中出现不可预知的逻辑故障导致应用程序脱离正常的执行流程时（俗称程序跑飞），在一定的时间间隔内使系统复位，回到初始状态，因此看门狗定时器是用来监视 MCU 程序运行状态的，确保系统可靠、稳定运行。看门狗定时器一旦启动就开始计数，达到溢出条件后会使 MCU 复位，所以用户需要在程序中使用专门语句在看门狗定时器达到溢出条件前，对计数值进行重置，以免 MCU 复位，这个过程俗称“喂狗”。

9-2 简述通用定时器与高级定时器的区别和应用场合。

STM32F103 微控制器内部集成了多个可编程定时器，分为基本定时器、通用定时器和高级定时器 3 大类，其中，通用定时器有 4 个：TIM2、TIM3、TIM4 和 TIM5；高级定时器有 2 个：TIM1 和 TIM8。STM32 系列微控制器中只有大容量的 STM32F103/107 系列微控制器具有高级定时器，从功能上看，高级定时器的功能强于通用定时器的功能，通用定时器的功能强于基本定时器的功能。

通用定时器具有定时、测量输入信号的脉冲长度（输入捕获）、输出所需波形（输出比较、产生 PWM、单脉冲输出等）等功能。

高级定时器除具有通用定时器具备的功能外，还提供控制三相六步电机的接口，具有刹车、死区时间控制等功能，主要用于电机控制。

9-3 计算采用 STM32 通用定时器分别精确延时 100μs、50μs、1ms、5ms 时的预分频系数 PSC 和 ARR 的值。

答：定时器的定时时间主要取决于定时周期和预分频因子，计算公式为：

定时时间=(ARR+1)×(预分频值 PSC+1)/输入时钟频率

或

T=(TIM_Period+1)×(TIM_Prescaler+1)/TIMxCLK

若预分频系数 PSC=36000−1，则 ARR=200-1=199

此时，定时时间=200×36000/72000000=0.1s。

若预分频系数 PSC=36000−1，则 ARR=10-1=9

此时，定时时间=10×36000/72000000=0.005s =5ms 。

9-4 简述 PWM 的工作原理。

PWM（Pulse Width Modulation，脉冲宽度调制），是一种利用脉冲宽度实现对模拟信号进行数字控制的一种技术。是占空比可以调制的脉冲波形。

STM32 实现 PWM 信号的输出，需要用到三个寄存器：自动重载寄存器 ARR，捕获/比较寄存器 CCR 以及计数寄存器 CNT，并通过通道引脚 TIMx_CHn 输出 PWM 信号。启动定时器后，计数器从 0 开始计数，计数过程中，不断将计数值 CNT 与捕获/比较值 CCR 以及自动重载值 ARR 相比较，当 CNT 小于 CCR 时，输出高电平；当 CNT 等于 CCR 时，输出电平翻转，变为低电平。计数器继续计数，当 CNT 等于 ARR 时，输出电平再次翻转，变为高电平，开始下一个周期的输出。根据 PWM 信号的输出过程可以得知，自动重载寄存器 ARR 用于控制 PWM 信号的周期，捕获/比较寄存器 CCR 用于控制 PWM 信号的占空比。

9-5 编写程序：基于 HAL 库实现 PWM 呼吸灯。

答：参考答案不唯一，以下程序仅供参考

```
while (1)
  {
     while (PWM_Val< 500)
   {
       PWM_Val++;
            __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, PWM_Val);
       HAL_Delay(1);
     }
  while (PWM_Val)
  {
       PWM_Val--;
       __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, PWM_Val);
       HAL_Delay(1);
     }
   HAL_Delay(200);
  }
  /* USER CODE END 3 */
}
```

# 第 10 章习题

## 习题 10

10-1 简述 A/D 转换的过程及主要技术参数。

A/D 转换过程一般包括采样、保持、量化和编码 4 个步骤。

ADC 的技术参数主要从转换精度和转换速度两方面考虑，一般用分辨率和转换误差来描述转换精度，用转换时间或转换速率描述转换速度。

10-2 STM32 的 ADC 有哪些通道？有哪几种转换模式？

STM32 的 ADC 有 16 个模拟输入通道，ADCx_IN0～ADCx_IN15 为 ADC 的输入信号通道，即为 ADC 的输入 GPIO 引脚，ADC 的 GPIO 引脚均为复用功能，使用时需将其配置为模拟输入模式。ADC1 还有两个内部通道：温度传感器通道和 VREFINT，分别用来测量芯片内部的温度和内部参考电压。温度传感器连接到 ADC1_IN16 通道，内部参考电压 VREFINT 连接到 ADC1_IN17 通道。

STM32 设计了按组进行转换的模式，可以由程序设置实现对多个模拟通道自动地进行逐个采样转换，STM32 分为两种组转换模式：规则组和注入组。

ADC 的转换模式主要有以下 4 种。

1．单次模式

单次转换模式下，ADC 只执行一次转换。

2．连续模式

连续转换模式下，当前 ADC 转换结束后就会立即启动下一个转换。

3．扫描模式

扫描模式用来扫描一组模拟通道，这组通道可以来规则通道组也可以来自注入通道组。开启扫描模式后，ADC 将扫描被选中通道组的所有通道，若将此时的转换模式设置为单次转换，则在扫描完本组所有通道后，ADC 自动停止；若将转换模式设置为连续模式，则在扫描完本组所有通道后，再从第一个通道继续扫描。

4．间断模式

间断模式用于多个通道的规则通道组和注入通道组。

10-3 简述 STM32 的 ADC 的转换时间。STM32F103 系列微控制器的 ADC 最短转换时间是多少？

答： STM32 芯片参考手册中规定：ADC 的时钟频率不能超过 14MHz，该频率由 PCLK2 经分频产生。由 STM32 的时钟结构可知，ADC 的时钟（ADCCLK）是由 APB2（PCLK2）经 ADC 预分频器分频得到的，分频值可设置为 2、4、6 或 8。

A/D 转换在采样时信号需保持一定的时间，以保证 A/D 转换的正确实现。STM32 的 ADC 每条通道的采样时间都可以进行选择，采样时间可选择为采样周期的 1.5 倍、7.5 倍、13.5 倍、28.5 倍、41.5 倍、55.5 倍、71.5 倍或 239.5 倍。

ADC 总的转换时间=采样时间+12.5 个周期

由于 ADC 的时钟 ADCCLK 最大不能超过 14MHz，因此 STM32F103 系列微控制器的 ADC 最短转换时间为 1us。采样时间越长，转换结果越稳定。

10-4 编写程序：基于 HAL 库采用查询方式和 DMA 两种方式采集 ADC1 通道 11 上的外部电压，将采样的数字量和对应计算得出的电压值通过串口调试助手显示出来。（提示：需要配置 USARTx 和 ADC1。）

# 第 11 章习题

# 习题 11

11-1 对智能小车进行需求分析，写出需求分析报告。

11-2 智能家居以住宅为平台，采用嵌入式技术、传感器技术、网络通信技术将家居设施融合关联在一起，实现对家居生活的智能化控制，给人们提供安全、舒适、便捷的居住环境。

1. 请以 STM32 为主控芯片，从功能概述、系统总体设计（组成框图）、功能模块介绍等方面阐述智能家居控制系统。

答:功能概述：

一个相对完善的智能家居监控系统主要包括以下四个功能:

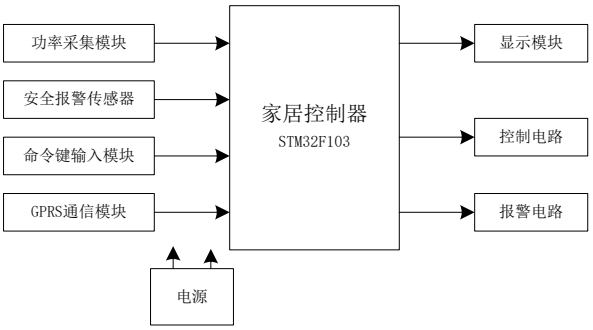
1. 家庭安全监护。家居中的门窗、燃气、火灾等异常状况，应及时将相应警报信息发送到移动终端，帮助用户及时了解家中的安全状况。

2. 家庭环境优化服务。控制家中的温度、湿度等，智能启动加湿器、空调，对家居环境进行智能调节室温和空气湿度，给用户一个舒适的生活环境。

3. 家庭能源管理。远程管理电源、用电设备及异常，能够智能降低功耗，监控用电设备安全。

4. 家居智能控制服务。对家中的所有家电，可通过移动终端实现远程或本地的开关控制。

系统硬件结构框图如下图所示。




各功能模块概述：

1. 通信电路设计。短信监控功能通过 GPRS 模块对短信进行操作实现，采用 AT 指令进行通信操

作。本系统中采用基于 USB 的通信模块 SIM908，通过 STM32 的串口进行连接。

2. 显示模块电路。液晶显示屏 TFT-LCD 具有驱动电压低、功耗小、反应速度快、显示画面质量好等优点，本系统采用 LCD 作为显示设备。LCD 的端口包括数据端口和控制端口，其中数据端口与主控芯片的 PB0-PB5 进行连接，控制端口与主控芯片的 PC6-PC9 连接。

3. 数据采集模块电路。温度采集采用数字温度传感器 DS18B20。功率采集电路主要由电流检测电路、电压检测电路、电能计量芯片 ADE7755 及其外围电路组成。家电开断控制电路采用三极管驱动继电器来实现家用电器供电控制。

2. 假如该智能家居控制系统中需要用到三个中断向量，其优先级配置如表 1 所示。试简述 STM32 中断机制，并针对表中中断向量说明其优先级次序。

**表 1　智能家居中断优先级配置**

| 中断向量 | 抢占优先级 | 响应优先级 |
|---|---|---|
| A | 0 | 0 |
| B | 1 | 0 |
| C | 1 | 1 |

答：STM32 中断机制：STM32 微控制器通过其内部的 NVIC（Nested Vectored Interrupt Controller，嵌套向量中断控制器）来进行管理和配置，实现中断通道的设置、优先级分配及中断使能等功能。无论是来自内核的异常还是来自外设的中断，都由 NVIC 进行控制和处理，STM32 的内部中断处理机制如下图所示。

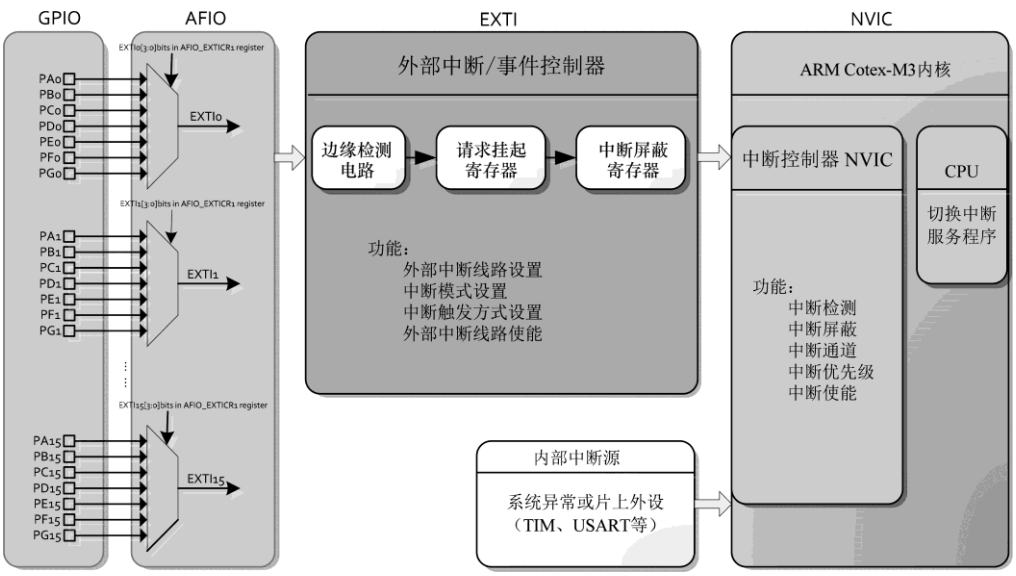



图　STM32 的内部中断处理机制

若内核正在执行 C 的中断服务函数，则它能被抢占优先级更高的中断 A 打断，而由于 B 和 C 的抢

占优先级相同，所以 C 不能被 B 打断。但如果 B 和 C 中断是同时到达的，内核就会首先响应响应优先级别更高的 B 中断。

3. 假设该智能家居控制系统中采用 LED 闪烁进行控制状态的显示，系统利用定时器 TIM3 产生 1s 的定时中断实现 LED 状态翻转，LED 连接在 STM32 的 PE5 引脚上，且低电平有效，硬件电路设计示意图如图 1 所示。

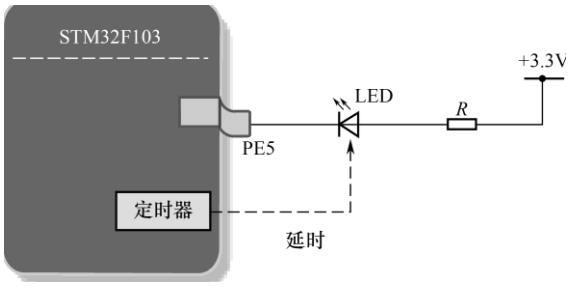



图 1　硬件电路设计示意图

该程序基于 HAL 库实现代码如下，请填写相关信息。

```
MX_TIM3_Init();
    /* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim3); //启动定时器 TIM3 定时中断
    /* USER CODE END 2 */

/* USER CODE BEGIN 4 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  if(htim->Instance == _________________)
  {
    HAL_GPIO_TogglePin(_____________,_____________);
  }
}
   /* USER CODE END 4 */
```

答：

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if(htim->Instance ==  htim3.Instance)
    {
        HAL_GPIO_TogglePin( GPIOE, GPIO_PIN_5);
    }
}
```

# 第 12 章习题

## 习题 12

12-1 嵌入式操作系统都有哪些功能？

嵌入式操作系统具有通用计算机操作系统的基本功能和特点，能够有效管理嵌入式系统的全部软、硬件资源的分配和任务调度。

针对不同应用场合，嵌入式操作系统包含的功能组件可能有所不同，但一般来说，都会有一个内核（Kernel），所谓内核是指系统的一个组件，包含了操作系统最基本、最主要的功能，如任务管理、存储管理、I/O 设备管理等，具有任务调度、任务管理、时间管理、内存管理、任务间的通信和同步等基本功能。

12-2 uC/OS-III 的内核包括哪几部分？

uC/OS-III 的内核包含了任务调度、任务管理、时间管理、内存管理、任务间的通信和同步等基本功能，是一个抢占式多任务处理内核。

12-3 uC/OS-III 的任务间通信与同步的实现方式有哪些？简述其工作原理。

uC/OS-III 提供了信号量、事件标志组等机制来实现任务之间的同步或者任务与中断服务程序（ISR）之间的通信。信号量机制用于控制对共享资源的保护，当有多个任务同时访问共享资源时，就需要为该共享资源设立一个标志，用于表示该共享资源的占用情况，当有任务需要访问该共享资源时，先查询共享资源的标志，判断这个共享资源是否被占用。信号量有两种形式：二值型信号量和多值型信号量。

信号量机制更多地被用来实现任务间的同步以及任务和 ISR 之间的同步，而当一个任务与多个任务发生同步时，推荐使用事件标志组。

uC/OS-III 通过消息队列等机制来提供通信服务，用来在任务之间传送任意数量的数据。

12-4 uC/OS-III 在创建任务时需要完成哪些工作？

uC/OS-III 的任务由三部分组成：任务控制块（Task Control Block, TCB）、任务堆栈和任务函数。uC/OS-III 在创建任务时需要定义任务栈、定义控制块 TCB、定义任务主体函数。

任务堆栈用来在切换任务和调用其他函数的时候保存现场，每一个任务都有自己的堆栈。

任务控制块用来记录任务的各个属性。uC/OS-III 任务控制块是用来记录与任务相关的信息的数据结构，如任务堆栈指针、任务当前状态、任务优先级等。每个任务都必须有各自的任务控制块。

任务函数由用户编写的任务处理代码，是应用功能的具体实现。uC/OS-III 应用程序由多个任务组成，任务本质上是一个带有 void *指针参数、无返回值的无限循环函数，参数是 void *类型，目的是可以传递不同类型的数据或函数。

12-5 嵌入式操作系统中任务调度有哪几种方式？

任务调度主要是协调任务对计算机系统资源的争夺使用，对系统资源非常匮乏的嵌入式系统来说，任务调度尤为重要，它直接影响着系统的实时性能。通常，任务调度主要有三种方式：基于优先级的抢占式调度、不可抢占式调度和时间片轮转调度。

12-6 时钟中断在 uC/OS-III 中的作用是什么？简述其工作原理。

时钟节拍是 uC/OS-III 操作系统的心脏，要实现时间管理（延时、超时确认等）就必须依赖于时钟节拍，这个时钟节拍就是操作系统的时基。STM32 采用 Cortex-M3 内核提供的 SysTick 定时器产生系统时钟节拍，是一个周期为毫秒级的系统时钟，SysTick 定时器能产生操作系统所需的定时中断，

uC/OS-III 中时钟中断函数负责处理时钟中断，它通过调用系统函数 OSTimeTick( )将等待时钟信号的高优先级任务进行中断级任务切换。uC/OS-III 允许用户在 os_cfg_app.h 文件中通过配置宏的值来设定系统时钟节拍的频率，范围为 10～1000Hz，当设置 OS_CFG_TICK_RATE_HZ 为 100Hz 时，也就是系统时钟节拍的周期为 10ms，这里 OS_CFG_TICK_RATE_HZ 默认为 1000Hz，也就是时钟节拍的周期为 1ms，从而实现每隔一定的时间就会产生一次中断，维持整个系统的运行。