# 第一章

1.什么是嵌入式系统

2.嵌入式系统与通用计算机系统的异同点？

3.嵌入式系统的特点主要有哪些？

4. ARM Cortex-M3处理器有几个类别，分别应用于哪些领域？

5.简要说明ARM Cortex-M3内核和ARM Cortex-M3 MCU的关系。

6.STM32微控制器产品线包括哪三个类别？

7.以STM32F103ZET6微控制器为例说明STM32 微控制器命名规则？

8.嵌入式系统的软件分为哪两种体系结构？

9.常见的嵌入式操作系统有哪几种？

10.对于STM32嵌入式系统软件，如何选择操作系统？

1. 嵌入式系统是指嵌入到其他设备或系统中的计算机系统，它通常具有实时性、稳定性、可靠性等特点，并能够实现特定的功能。

2. 嵌入式系统与通用计算机系统的主要区别在于硬件平台和应用场景。通用计算机系统是以PC机为代表的通用计算机平台，具有通用性和灵活性，主要用于人机交互、办公、娱乐等领域；而嵌入式系统则是为了满足特定的应用需求而设计的，通常采用定制化的硬件平台和实时操作系统，主要应用于汽车、家电、医疗设备、工业控制等领域。

3. 嵌入式系统的特点主要包括：实时性、稳定性、可靠性、节能性、体积小、功耗低、成本低等。这些特点是为了满足特定的应用需求而设计的。

4. ARM Cortex-M3处理器有3个类别：Cortex-M3、Cortex-M4和Cortex-M7。Cortex-M3适用于功耗和成本敏感的应用，如工业控制、安防监控、智能家居等；Cortex-M4增加了DSP指令集和浮点运算能力，适用于音频处理、图像处理、机器人控制等应用；Cortex-M7则增加了高级优化特性，适用于高端工业控制、自动驾驶、机器人等应用。

5. ARM Cortex-M3内核是一种基于ARMv7-M架构的32位RISC处理器核心，提供了高性能、低功耗、实时性等特性。ARM Cortex-M3 MCU是基于ARM Cortex-M3内核的微控制器产品，集成了处理器核心、存储器、外设和调试接口等，可以直接用于嵌入式系统设计。

6. STM32微控制器产品线包括三个类别：STM32F系列、STM32L系列和STM32H系列。其中，STM32F系列是高性能、高集成度的产品，适用于工业控制、汽车电子、通信等领域；STM32L系列是低功耗、高性能比的产品，适用于消费电子、医疗设备、智能家居等领域；STM32H系列是高性能、安全可靠的产品，适用于航空航天、国防等高端应用。

7. STM32F103ZET6微控制器的命名规则为：STM32为产品线名称，F表示高性能系列，103为产品系列中的型号，Z表示封装形式，ET表示扩展温度范围，6表示Flash存储器的大小为512KB。不同的型号在命名规则中可能会有略微不同的标识，但大体上都包括产品系列、性能等级、封装形式、存储器容量等信息。

8. 嵌入式系统的软件分为裸机体系结构和操作系统体系结构两种。裸机体系结构是指在没有操作系统支持的情况下，直接编写裸机程序进行硬件操作；操作系统体系结构则是指在操作系统的支持下，以任务为单位进行软件设计，具有更好的可维护性和可扩展性。

9. 常见的嵌入式操作系统有：FreeRTOS、uC/OS、Linux嵌入式版、Windows嵌入式版等。其中，FreeRTOS和uC/OS是常用的实时操作系统，适用于资源有限、要求实时响应的嵌入式系统；Linux嵌入式版和Windows嵌入式版则适用于需要更强大的计算能力和丰富的应用支持的嵌入式系统。

10. 在选择嵌入式操作系统时，需要考虑系统的实时性、资源占用、可维护性、应用支持等因素。对于STM32嵌入式系统软件，可以根据具体的应用需求选择不同的操作系统，如FreeRTOS适用于实时性要求较高的应用，Linux嵌入式版适用于需要更强大的计算能力和应用支持的应用等。同时，也可以根据开发人员的熟悉程度、社区支持等因素进行选择。

# 第二章

1.开发板使用的微控制器具体型号是什么？

2.开发板电源模块可以提供几种供电电压？

3.LED指示灯采用共阳接法还是共阴接法，其电源电压为多少伏？

4.当开发板的按键按下时端口的为高电平还是低电平？

5.开发板有几个外接晶振，频率分别为多少，各自的作用是什么？

6.数码管采用共阳接法还是共阴接法，其笔画位和公共端分别由MCU的哪些GPIO控制？

1. STM32F103ZET6开发板使用的微控制器型号是STM32F103ZET6。

2. STM32F103ZET6开发板电源模块可以提供多种不同的供电电压，如3.3V、5V等。

3. LED指示灯的接法可能因不同的开发板而异。一般来说，STM32F103ZET6开发板上的LED指示灯采用共阴接法，其电源电压为3.3V。

4. 当STM32F103ZET6开发板上的按键按下时，端口的电平会变为低电平。

5. STM32F103ZET6开发板有两个外接晶振，分别为8MHz和32.768kHz。8MHz的晶振用于MCU的主频时钟，而32.768kHz的晶振用于实时时钟和低功耗模式下的时钟。

6. STM32F103ZET6开发板上的数码管采用共阳接法。笔画位和公共端由MCU的不同GPIO控制，具体的GPIO引脚如下：

- 笔画位A由GPIOA的第0位控制；

- 笔画位B由GPIOA的第1位控制；

- 笔画位C由GPIOA的第2位控制；

- 笔画位D由GPIOA的第3位控制；

- 笔画位E由GPIOA的第4位控制；

- 笔画位F由GPIOA的第5位控制；

- 笔画位G由GPIOA的第6位控制；

- 数码管公共端由GPIOB的第12位控制。

# 第三章

1.什么是STM32固件库？

2.目前通用的STM32F10x固件库版本是多少？

3.STM32固件库下载网址是什么？并从该网站下载最新的固件库。

4.请指出固件库中文件core\_cm3.c文件的路径。

5.请指出固件库中文件stm32f10x.h文件的路径。

6.Keil MDK -ARM软件的下载网址是什么？并从该网站下载最新的软件包。

7.Keil MDK -ARM软件建立的工程文件的扩展名是什么？

8.参照书中工程模板创建方法，建立自己的工程模板。

# 第四章

1.什么是 GPIO?

2. STM32F103微控制器 GPIO 的引脚是如何命名的？

3. STM32F103微控制器 GPIO有几种输入工作模式？

4. STM32F103微控制器GPIO有几种输出工作模式？

5. STM32F103微控制器GPIO输出速度有哪几种？在应用中应如何选择？

6. STM32F1033微控制器GPIO相关寄存器有哪些？

7.如果需要设置GPIOE端口所有引脚输出低电平，则GPIOE\_ODR寄存器的值为多少？

8.如果需要设置 GPIOE 端口所有引脚输出高电平，则 GPIOE\_BSRR寄存器的值为多少？

9.STM32微控制器软件设计有哪两种常用开发方式？

1. STM32固件库是STMicroelectronics公司针对其STM32系列微控制器推出的一套软件库，包含了丰富的驱动程序、示例代码和工具函数，用于简化STM32系列微控制器的开发流程。

2. 目前通用的STM32F10x固件库版本是V3.5.0。

3. STM32固件库下载网址是 https://www.st.com/content/st\_com/zh/products/embedded-software/mcu-mpu-embedded-software/stm32-embedded-software/stm32-standard-peripheral-libraries/stsw-stm32054.html#overview ，可以从该网站下载最新的固件库。

4. 固件库中文件core\_cm3.c文件的路径是: STM32F10x\_StdPeriph\_Lib\_V3.5.0/Libraries/CMSIS/CM3/CoreSupport/core\_cm3.c

5. 固件库中文件stm32f10x.h文件的路径是: STM32F10x\_StdPeriph\_Lib\_V3.5.0/Libraries/STM32F10x\_StdPeriph\_Driver/inc/stm32f10x.h

6. Keil MDK-ARM软件的下载网址是 https://www.keil.com/demo/eval/armv4.htm ，可以从该网站下载最新的软件包。

7. Keil MDK-ARM软件建立的工程文件的扩展名是.uvprojx。

8. 创建自己的工程模板可以按照以下步骤进行：

为了创建自己的工程模板，可以按照以下步骤进行操作：

1)在Keil软件中创建一个新的工程，选择一个合适的芯片型号和存储路径，点击“Save”。

2)在新建的工程中添加必要的源文件、头文件和库文件。

3)在菜单栏中选择“Project”，然后选择“Options for Target”。在弹出的对话框中，设置相关的编译和链接选项，如搜索路径、优化选项等。

4)如果需要使用调试器进行调试，可以选择“Debug”选项卡，然后设置调试器类型和调试选项。

5)最后，将新建的工程保存为模板文件。在菜单栏中选择“Project”，然后选择“Manage Project Items”。在弹出的对话框中，选择“Export Project Template”，然后设置模板文件的存储路径和名称，点击“Export”。

完成以上步骤后，就可以使用自己的工程模板来创建新的工程了。在Keil软件中选择“Project”，然后选择“New uVision Project from Template”，在弹出的对话框中选择自己创建的模板文件，然后设置新工程的名称和存储路径，点击“Save”即可。

# 第五章

1.基于库函数开发方式的特点有哪些？

2. 函数 RCC\_APB2PeriphClockCmd 的功能有哪些？

3.函数 GPIO\_Init 的功能有哪些？有哪些参数？

4.函数 GPIO\_Write 的功能是什么，有哪些参数？

5. 简述函数 GPIO\_Write、GPIO\_SetBits 和 GPIO\_ResetBits 的异同点。

6.简要说明SysTick 定时器的概况以及使用该定时器的好处。

7. SysTick定时器相关的控制寄存器有哪些？

8. SysTick定时器常用的延时函数有哪两个？

# 第六章

1.GPIO输入函数有哪些，名称、功能、输入参数、返回值各是什么？

2.单片机控制系统中，按键应如何连接？不同连接方式，配置GPIO工作模式时应如何选择？

3.蜂鸣器的工作原理是什么？什么是有源蜂鸣器？什么是无源蜂鸣器？在单片机控制系统中应如何控制它们？

4.按键输入时是如何实现去抖动的，除了软件去抖动而外，如何进行硬件去抖动？

5.如何实现按键功能复用？分别对单击、双击、长按等编写不同的响应程序。

6.如何利用蜂鸣器发出有节奏的声音？参考网上资料，利用实验板上的硬件资源，编写简单的乐曲演奏程序。

1. GPIO输入函数有以下两个：

（1） uint8\_t GPIO\_ReadInputDataBit(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)

功能：读取指定GPIO口的输入状态

输入参数：GPIO\_TypeDef\* GPIOx GPIO端口，GPIO\_Pin：GPIO引脚

返回值：0或1，对应输入引脚的电平状态

（2） uint16\_t GPIO\_ReadInputData(GPIO\_TypeDef\* GPIOx)

功能：读取指定GPIO口所有输入端口的状态

输入参数：GPIO\_TypeDef\* GPIOx GPIO端口

返回值：对应GPIO端口所有引脚的状态值，以16位二进制数形式返回

2. 按键通常采用两种连接方式：一种是将按键连接到单片机的普通IO口上，需要配置为输入模式；另一种是采用外部中断方式，将按键连接到外部中断输入口上。在配置GPIO工作模式时，对于第一种连接方式，应选择GPIO输入模式；对于第二种连接方式，应选择外部中断输入模式。

3. 蜂鸣器的工作原理是将交流信号通过控制板上的音频功率放大电路输出到扬声器上，从而使扬声器发出声音。有源蜂鸣器是内置了音频功放芯片，可直接连接到外部信号源上，通常具有较高的音量和质量；无源蜂鸣器则需要外接电路将音频信号放大后再输出到扬声器上。在单片机控制系统中，通过控制IO口的电平状态来控制蜂鸣器的开关。

4. 去抖动可以通过软件和硬件两种方式实现。软件去抖动的方法是：在按键按下时，延时一定的时间再读取按键的状态，如果读取的状态为按下状态，则认为按键有效；否则，认为按键无效。硬件去抖动的方法是：在按键输入时通过RC滤波电路滤除按键输入信号中的抖动部分，从而得到一个干净的信号。

5. 实现按键功能复用的方法是：在程序中通过定时器或者外部中断来监测按键的状态，并根据不同的按键状态执行不同的功能。例如，对于单击、双击、长按等按键事件，可以通过定时器来记录按键的时间，并在一定的时间内判断按键的状态，从而实现不同的响应程序。

#include "stm32f10x.h"

#define KEY\_PORT GPIOA

#define KEY\_PIN GPIO\_Pin\_0

#define SINGLE\_CLICK 1

#define DOUBLE\_CLICK 2

#define LONG\_PRESS 3

// 定义按键状态变量

volatile uint8\_t key\_state = 0;

// 记录按键按下的时间

volatile uint32\_t key\_down\_time = 0;

// 检测按键状态

void check\_key\_state(void) {

static uint8\_t press\_cnt = 0;

static uint8\_t long\_press\_flag = 0;

// 检测按键是否按下

if (!GPIO\_ReadInputDataBit(KEY\_PORT, KEY\_PIN)) {

// 按键按下

if (key\_state == 0) {

key\_state = 1;

key\_down\_time = SysTick->VAL;

}

// 按键长按检测

if ((SysTick->VAL - key\_down\_time) > 3000) {

if (!long\_press\_flag) {

long\_press\_flag = 1;

key\_state = LONG\_PRESS;

}

}

// 按键双击检测

if (press\_cnt == 1) {

if ((SysTick->VAL - key\_down\_time) < 500) {

key\_state = DOUBLE\_CLICK;

}

press\_cnt = 0;

} else {

press\_cnt++;

}

} else {

// 按键松开

if (key\_state != 0) {

// 如果是单击或长按，执行单击或长按程序

if (key\_state == SINGLE\_CLICK || key\_state == LONG\_PRESS) {

// 单击或长按处理程序

} else if (key\_state == DOUBLE\_CLICK) {

// 双击处理程序

}

// 清空按键状态变量

key\_state = 0;

press\_cnt = 0;

long\_press\_flag = 0;

}

}

}

int main(void) {

// 初始化系统时钟、GPIO等

// ...

while (1) {

// 轮询方式检测按键状态

check\_key\_state();

}

}

6. 利用蜂鸣器发出有节奏的声音可以通过PWM控制蜂鸣器的工作频率和占空比来实现。具体步骤如下：

这里提供一种可能的实现方法，仅供参考。

1. 硬件资源：实验板上的蜂鸣器和定时器TIM2

2. 软件实现：通过调节TIM2的定时周期和占空比，产生不同频率和持续时间的PWM波，驱动蜂鸣器发声。

3. 程序流程：

1. 配置TIM2为PWM输出模式

2. 根据需要定义不同频率和持续时间的音符，并将其转化为对应的TIM2定时周期和占空比参数。

3. 将音符序列按照乐曲的节奏和顺序组合起来，形成完整的乐曲。

4. 通过循环依次输出每个音符，控制蜂鸣器发声。

第七章

1.数码管显示的原理是什么，共阳和共阴显示码如何确定？显示码分别是什么？

2.什么是静态显示？什么是动态显示？两种显示方法的特点分别是什么？应如何进行选择？

3.数码管动态扫描的延时时间长短对显示效果有何影响，延时程序一般应如何编写？

4.在本章的扩展项目中，为什么要使用数组，数组的数据类型如何确定，如何定义数组引用数组元素？

5.如何改写学号显示程序？使其通用性更强，即由一个学号显示更改为另一个学号显示，其序修改量越少越好。

6.如何在一个开发板上显示两位或多位同学的学号？每一个学号显示持续时间大概3秒。

7.如何实现滚动显示？例如，在六位数码管上显示11位的手机号码。

8将章数码显示与第6章按键输入结合在一起，当有不同按键按下时，数码管上显示不同的内容。

9.编写一通用程序，将一个无符号整型数据显示在六位数码管上。

10.已知有一浮点数，其数值小于1000，且小数点后保留两位小数，如何将其显示在数码管上？

1. 数码管显示原理：数码管是由多个LED组成的，LED的发光是通过加电使其正向导通，电流从LED的正极流向负极，使LED发光。共阳数码管的阳极连接在LED的正极，共阴数码管的阴极连接在LED的负极。共阳数码管在显示时，需要将要显示的数字通过流向对应的LED，即在对应的段选端口输出低电平；共阴数码管需要将要显示的数字通过断开对应的LED，即在对应的段选端口输出高电平。共阳数码管的显示码对应表和共阴数码管的显示码对应表不同。共阳数码管的显示码为：0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80, 0x90；共阴数码管的显示码为：0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x67。

2. 静态显示是指在某一个时刻，每个数码管上显示的数字都是固定的，不会随时间变化。动态显示是指在一段时间内，不同的数码管上显示不同的数字，随时间变化。静态显示适用于需要同时显示多个数字的场合，动态显示适用于只需要显示一个数字的场合。选择显示方式需要根据实际需要进行判断。

3. 数码管动态扫描的延时时间长短对显示效果有一定影响。如果延时时间太短，会导致数码管上显示的数字不稳定，闪烁或者不全；如果延时时间太长，会导致数码管上显示的数字变化过慢，无法正常显示。延时程序一般需要根据具体的硬件和延时时间进行编写。

4. 使用数组可以简化程序，将相同或类似的操作通过循环来实现。在扩展项目中，数组的数据类型应为整型，数组元素的数量应与学号位数相同。数组可以通过定义和引用的方式来使用，例如定义一个6位学号的数组：int stu\_no[6]，引用数组元素：stu\_no[0]表示第一个数字。

5. 可以通过将学号作为参数传入函数，函数内部进行对应的显示操作，从而实现通用性。具体来说，可以编写一个函数，将学号作为参数传入，并在函数内部将学号转换成数字，并进行相应的显示操作。

6. 要在一个开发板上显示两位或多位同学的学号，可以使用数码管动态扫描技术，将每一位的数码分别输出到数码管上，然后通过延时的方式实现每一个学号显示持续时间大约为3秒。可以将学号存储在一个数组中，然后通过循环依次输出每一位的数码，每次输出之间的延时为3秒，如下所示：

```c

#include <reg52.h>

// 数码管显示码

unsigned char code numCode[] = {

0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80, 0x90

};

// 学号数组

unsigned char code stuNum[] = "20210001";

void main() {

unsigned char i, j;

while(1) {

for (i = 0; i < sizeof(stuNum); i++) { // 循环输出每一位学号

for (j = 0; j < 50; j++) { // 数码管动态扫描延时

P2 = 0x01; // 第一位数码管

P0 = numCode[stuNum[i] - '0'];

Delay(1);

P2 = 0x02; // 第二位数码管

P0 = numCode[stuNum[i+1] - '0'];

Delay(1);

}

}

}

}

// 延时函数

void Delay(unsigned int n) {

unsigned int i, j;

for (i = 0; i < n; i++) {

for (j = 0; j < 125; j++);

}

}

```

7. 实现滚动显示可以通过定时器中断来完成。具体步骤如下：

- 将需要显示的数据存储在数组中。

- 开启定时器中断，每个中断周期显示数组中的一个元素。

- 在显示下一个元素之前，将当前数码管的值清零或显示其他内容，避免出现残影。

- 循环执行以上步骤，直到全部数据都被显示完毕。

例如，要在六位数码管上滚动显示一个11位的手机号码，可以将该手机号码存储在一个长度为11的数组中，然后使用定时器中断来不断刷新数码管的显示，从而实现滚动显示的效果。

8. 数码管和按键可以通过中断来实现按键输入的响应和数码管显示的更新。具体步骤如下：

- 配置数码管的GPIO口为输出模式，按键的GPIO口为输入模式。

- 在主循环中不断检测按键的状态，如果按键被按下，则触发中断，并在中断服务函数中更新需要显示的内容。

- 在定时器中断中不断更新数码管的显示内容，根据需要显示的内容选择对应的数码管码值进行显示。

例如，可以通过按下不同的按键来切换数码管的显示内容，例如显示学号、班级、姓名等等。

9. 编写一通用程序，将一个无符号整型数据显示在六位数码管上，可以按照以下步骤进行：

- 将需要显示的无符号整型数据转换为字符串。

- 在字符串的前面补零，使得字符串长度为6。

- 将字符串中的每个字符转换为对应的数码管码值。

- 在数码管上显示转换后的结果。

例如，要将无符号整型数123456显示在六位数码管上，可以将其转换为字符串"123456"，在字符串前面补零得到"123456"，然后将每个字符转换为对应的数码管码值，在数码管上显示结果。

10. 要将一个浮点数显示在数码管上，可以按照以下步骤进行：

- 将浮点数乘以100并四舍五入，得到一个整数。

- 将整数转换为字符串。

- 在字符串中插入小数点，使得小数点位于第4位。

- 将字符串中的每个字符转换为对应的数码管码值。

- 在数码管上显示转换后的结果。

例如，要将浮点数3.14显示在数码管上，可以将其乘以100得到314，然后将其转换为字符串"314"，在字符串中插入小数点得到"3.14"，最后将字符串中的每个字符转换为对应的数码管码值，在数码管上显示结果。

# 第八章

1.什么是中断？为什么要使用中断？

2.什么是中断源？STM32F103支持哪些中断源？

3.什么是中断屏蔽？为什么要进行中断屏蔽？如何进行中断屏蔽？

4.中断的处理过程是什么？包含哪几个步骤？

5.什么是中断优先级？什么是中断嵌套？

6.STM32F103优先级分组方法中，什么是抢占优先级？什么是响应优先级？

7.什么是中断向量表？它通常存放在存储器的哪个位置？

8.什么是中断服务函数？如何确定中断函数的名称？在哪里编写中断服务程序

9.中断服务函数与普通的函数相比有何异同？

10.什么叫断点？什么叫中断现场？断点和中断现场保护和恢复有什么意义？

11.对本章所介绍时间调节项目进行修改，时间调节设有三个按键：一个按键用来选择调节位置，一个按键是数字加，一个按键是数字减。

12.设计并完成项目，开发板上电LED指示灯L1亮，设置两个按键：一个用于LED灯左移，另一个用于LED灯右移。

1.中断是指在CPU运行期间，来自硬件设备或其他软件模块的异步事件，强制CPU立即停止当前的工作流程并转而处理这个事件。使用中断可以提高系统的响应能力和效率，避免CPU忙等待。

2.中断源是指产生中断的硬件设备或软件模块。STM32F103支持多达82个中断源，包括外部中断、定时器中断、UART中断、SPI中断、I2C中断等。

3.中断屏蔽是指禁止某些中断，使其在处理优先级较低的中断时不被响应。进行中断屏蔽是为了避免高优先级中断的不必要干扰，从而提高系统的实时性能。中断屏蔽可以通过修改中断控制器的寄存器或设置CPU的特殊指令实现。

4.中断的处理过程包括以下几个步骤：中断请求发生->CPU停止当前工作流程->CPU保存现场->CPU跳转到中断服务函数开始执行->中断服务函数执行完毕->CPU恢复现场->CPU返回之前的工作流程。

5.中断优先级是指中断请求被响应的优先级，它可以决定CPU在处理多个中断请求时的顺序。中断嵌套是指当一个中断正在处理时，另一个中断请求发生，此时CPU可以暂停当前中断的处理，转而响应更高优先级的中断请求。

6.STM32F103优先级分组方法中，抢占优先级是指中断请求发生时，如果当前正在执行的中断优先级低于抢占优先级，则CPU会立即停止当前中断处理，转而响应抢占优先级更高的中断请求。响应优先级是指中断请求发生时，如果当前正在执行的中断的优先级与新中断的优先级相同，则CPU会根据响应优先级决定是否暂停当前中断，转而响应新中断请求。

7.中断向量表是存储中断向量地址的表格，用于指示中断服务程序的入口地址。它通常存放在芯片的内部Flash或RAM中的固定地址处，且是只读的。

8.中断服务函数是处理中断请求的程序，它包含在中断处理过程中，以响应中断请求。中断函数的名称一般按照规定的格式来命名，格式为"IRQn\_Handler()"，其中n表示中断号。中断服务程序可以在C语言中编写，但需要遵循一定的规范，如不能使用浮点数、不能调用普通函数等。

9. 中断服务函数与普通的函数相比有何异同？

中断服务函数（ISR）和普通函数（Function）在使用方式、参数传递、返回值等方面有很大的不同。

（1）使用方式：ISR是由硬件中断触发执行的函数，而普通函数则是由程序调用执行的函数。

（2）参数传递：ISR的参数通常是由硬件中断传递的参数，而普通函数的参数是由程序传递的参数。

（3）返回值：ISR通常没有返回值，而普通函数通常需要有返回值。

（4）可重入性：ISR必须是可重入的，因为当一个中断正在执行时，另一个中断可能会发生，而普通函数则不需要考虑这个问题。

（5）执行时间：ISR的执行时间必须非常短，以避免影响其他中断或主程序的执行，而普通函数则没有这个限制。

10. 什么叫断点？什么叫中断现场？断点和中断现场保护和恢复有什么意义？

（1）断点：在程序中设置断点就是指程序在执行到指定的代码行时停止执行，以便程序员查看当前程序状态和调试代码。

（2）中断现场：在CPU执行中断服务程序时，需要保护中断现场（即CPU在进入中断服务程序之前的状态）以便中断服务程序执行完后恢复CPU的状态并继续执行主程序。中断现场通常包括CPU的寄存器、状态字和栈指针等。

（3）断点和中断现场保护和恢复的意义：断点可以帮助程序员在调试程序时快速定位和解决问题。中断现场保护和恢复可以确保中断服务程序的正常执行，避免因为中断服务程序对现场的改变而导致主程序出错，提高系统的可靠性和稳定性。

11.修改时间调节项目，增加选择调节位置的按键和数字加、数字减的按键，可以按照以下步骤进行：

1)确定硬件连接，将选择调节位置的按键连接到单片机的一个GPIO口上，将数字加、数字减的按键分别连接到两个不同的GPIO口上，同时需要连接上电源和地线。

2)修改程序，根据选择调节位置的按键的状态来确定数字加、数字减的目标寄存器。例如，当选择调节小时数时，数字加、数字减的目标寄存器为小时寄存器；当选择调节分钟数时，数字加、数字减的目标寄存器为分钟寄存器。

3)修改程序，增加中断服务函数。当选择调节位置的按键被按下时，触发外部中断，在中断服务函数中修改目标寄存器，并更新数码管显示；当数字加、数字减的按键被按下时，触发按键中断，在中断服务函数中修改目标寄存器，并更新数码管显示。

4)修改程序，增加状态变量。用于标记当前选择的调节位置和当前按下的数字加、数字减的按键。

5)修改程序，增加按键检测函数。在主循环中，检测选择调节位置的按键、数字加、数字减的按键是否被按下，若被按下则更新状态变量。

6)修改程序，根据状态变量来决定是否进入中断服务函数。

7)测试程序是否正常运行，通过选择调节位置的按键和数字加、数字减的按键，修改时间并显示到数码管上。

注意：在增加外部中断和按键中断时，需要设置优先级，并根据优先级进行相应的中断处理。同时需要注意中断嵌套问题，避免出现多重中断的情况。

12.这个项目的主要思路是通过读取两个按键的状态来控制 LED 的位置。为了达到这个目的，需要使用输入输出口和中断。

1)硬件连接

首先，将两个按键连接到开发板的两个 GPIO 端口上，同时将 LED 连接到另一个 GPIO 端口上。在本项目中，假设按键连接到了 PA0 和 PA1 上，LED 连接到了 PB12 上。

2)配置 GPIO 端口

使用 CubeMX 工具或手动编写代码来配置 GPIO 端口。需要将 PA0 和 PA1 配置为输入模式，PB12 配置为输出模式。同时，要在 PA0 和 PA1 上启用外部中断，使得按键状态变化可以被检测到。

3)配置中断优先级

在 STM32F103 中，有 16 个中断优先级，其中抢占优先级和响应优先级各占 4 位。可以使用 CubeMX 工具来配置中断优先级，也可以手动编写代码。在本项目中，假设要求 LED 移动的优先级比较高，所以将其抢占优先级设为 1，响应优先级设为 0。同时，将按键中断的优先级设为低于 LED 移动的优先级，例如抢占优先级为 2，响应优先级为 0。

4)编写中断服务函数

需要编写两个中断服务函数：一个用于处理 PA0 的中断，一个用于处理 PA1 的中断。在中断服务函数中，需要读取按键状态，如果按键按下，则改变 LED 的位置。

5)编写主程序

在主程序中，需要初始化 GPIO 端口和中断优先级，然后进入一个死循环。在死循环中，不断地读取 LED 的位置并将其输出到 PB12 上。

6)测试

编译程序并下载到开发板上，测试 LED 是否可以移动。如果按键按下时 LED 可以移动，说明程序运行正常。

总的来说，本项目的实现思路比较简单，主要就是读取按键状态，根据按键状态来改变 LED 的位置。但是，在实现过程中，需要注意中断优先级的设置，以确保 LED 移动的优先级比按键中断的优先级高，从而避免按键中断被优先处理而导致 LED 移动不稳定的问题。

12. 这个项目的主要思路是通过读取两个按键的状态来控制 LED 的位置。为了达到这个目的，需要使用输入输出口和中断。

1)硬件连接

首先，将两个按键连接到开发板的两个 GPIO 端口上，同时将 LED 连接到另一个 GPIO 端口上。在本项目中，假设按键连接到了 PA0 和 PA1 上，LED 连接到了 PB12 上。

2)配置 GPIO 端口

使用 CubeMX 工具或手动编写代码来配置 GPIO 端口。需要将 PA0 和 PA1 配置为输入模式，PB12 配置为输出模式。同时，要在 PA0 和 PA1 上启用外部中断，使得按键状态变化可以被检测到。

3)配置中断优先级

在 STM32F103 中，有 16 个中断优先级，其中抢占优先级和响应优先级各占 4 位。可以使用 CubeMX 工具来配置中断优先级，也可以手动编写代码。在本项目中，假设要求 LED 移动的优先级比较高，所以将其抢占优先级设为 1，响应优先级设为 0。同时，将按键中断的优先级设为低于 LED 移动的优先级，例如抢占优先级为 2，响应优先级为 0。

4)编写中断服务函数

需要编写两个中断服务函数：一个用于处理 PA0 的中断，一个用于处理 PA1 的中断。在中断服务函数中，需要读取按键状态，如果按键按下，则改变 LED 的位置。

5)编写主程序

在主程序中，需要初始化 GPIO 端口和中断优先级，然后进入一个死循环。在死循环中，不断地读取 LED 的位置并将其输出到 PB12 上。

6)测试

编译程序并下载到开发板上，测试 LED 是否可以移动。如果按键按下时 LED 可以移动，说明程序运行正常。

总的来说，本项目的实现思路比较简单，主要就是读取按键状态，根据按键状态来改变 LED 的位置。但是，在实现过程中，需要注意中断优先级的设置，以确保 LED 移动的优先级比按键中断的优先级高，从而避免按键中断被优先处理而导致 LED 移动不稳定的问题。

# 第九章

1.软件延时和可编程定时延时特点是什么？分别应用于什么场合？

2.嵌入式系统中，定时器的主要功能有哪些？

3.STMF103微控制器定时器的类型有哪几种？不同类型的定时器有什么区别？

4.STM32F103微控制器通用定时器的常用工作模式有哪些？

5.什么是 PWM? PWM的实现方式有哪几种？

6.定时器初始化时，如何确定预分频寄存器TIMx\_PSC和自动重装值寄存器的TIMxARR的值？

7.在PWM输出模式中，如何确定PWM波的频率？如何确定PWM波的占空比？如何确定PWM波输出的极性？

8.STM32F103微控制器通用定时器有哪几种计数方式？何时可以产生更新事件？

9.利用定时器实现开发板LED灯秒闪烁功能，要求亮灭各500ms。

10.利用定时器产生精确的1s的定时，秒数值从0开始向上累加，并将数值显示于六位数码管。

11.编写程序，使开发板微控制器的PC7引脚输出频率为1kHz,占空比为50%的PWM方波。

12.定时器如何对外部脉冲进行计数？编写程序对按键次数进行统计，并显示于数码管上。

1. 软件延时和可编程定时延时的特点和应用场合如下：

- 软件延时：通过程序代码中的循环、计数等方式实现，具有可编程性强、灵活性好的特点，适用于简单的时间控制和延时等场合。

- 可编程定时延时：通过定时器模块实现，具有精度高、可靠性好、占用CPU资源少的特点，适用于需要精确时间控制的场合，如测距、脉冲计数、PWM波控制等。

2. 嵌入式系统中，定时器的主要功能包括：

- 生成精确的时间延时

- 计数外部信号脉冲

- 产生PWM波控制信号

- 定时触发ADC采样

- 实现定时中断等功能。

3. STMF103微控制器的定时器类型包括基本定时器（TIM6/TIM7）、通用定时器（TIM2/TIM3/TIM4/TIM5）和高级定时器（TIM1）。它们的区别在于定时器功能和硬件资源的配置不同，高级定时器还支持更高级的功能如编码器模式和PWM输入模式等。

4. STM32F103微控制器通用定时器的常用工作模式包括：

- 确定时间周期的向上计数模式

- 周期性计数模式

- 单脉冲模式

- 编码器模式

- PWM输出模式

- PWM输入模式等。

5. PWM（Pulse Width Modulation）是一种通过调整脉冲宽度来控制电路输出信号的技术。它可以实现模拟信号的数字化、数字信号的模拟化以及电源电压调节等功能。PWM的实现方式包括：

- 基于定时器的PWM

- 基于比较器的PWM

- 基于高速I/O口的PWM等。

6. 在定时器初始化时，可以根据需要确定预分频寄存器TIMx\_PSC和自动重装值寄存器的TIMx\_ARR的值。预分频寄存器决定了定时器时钟的分频系数，自动重装值寄存器决定了定时器计数器计数到多少值后产生更新事件。具体的计算公式为：定时器的计数频率 = 定时器时钟源的频率 / (TIMx\_PSC + 1)，定时时间 = TIMx\_ARR / 定时器计数频率。

7. 在PWM输出模式中，可以通过设置预分频寄存器和自动重载寄存器的值来确定PWM波的频率和占空比，同时可以通过设置控制寄存器来确定PWM波输出的极性。

确定PWM波的频率需要先确定计数器的时钟频率，这个时钟频率是通过分频器的分频系数和微控制器的主频来计算得出的。然后根据所需的PWM波频率和计数器时钟频率，计算出自动重载寄存器的值。

确定PWM波的占空比，需要根据所需的占空比和自动重载寄存器的值来计算得出占空比对应的计数值，这个计数值就是占空比所对应的高电平时间。

确定PWM波输出的极性，需要设置控制寄存器中的输出极性位来选择正极性输出还是负极性输出。正极性输出意味着高电平时输出高电压，低电平时输出低电压，而负极性输出则相反。

8. STM32F103微控制器通用定时器有以下三种计数方式：

1)向上计数模式：定时器从0开始计数，当定时器计数值达到自动重载寄存器的值时，产生更新事件，并将计数器清零重新计数。

2)向下计数模式：定时器从自动重载寄存器的值开始计数，当计数器计数值为0时，产生更新事件，并将计数器自动加载自动重载寄存器的值重新计数。

3)向上/向下计数模式：定时器从0开始计数，当计数器计数值达到自动重载寄存器的值时，产生更新事件，同时改变计数方向为向下，当计数器计数值为0时再次产生更新事件并改变计数方向为向上，如此循环。

更新事件是指计数器达到自动重载寄存器的值时产生的中断或者触发的事件，可以用于定时器的重载或者周期性的产生中断信号等应用。产生更新事件的条件取决于计数模式、时钟源、预分频系数等参数的设置。

9.利用定时器实现开发板LED灯秒闪烁功能，要求亮灭各500ms。

实现步骤如下：

1）初始化定时器 TIMx，并设置预分频系数和自动重装值，使其定时周期为 500ms；

2）开启定时器 TIMx；

3）在定时器中断服务函数中，翻转 LED 灯的状态。

代码示例：

```c

#include "stm32f10x.h"

#define LED\_PIN GPIO\_Pin\_13

#define LED\_PORT GPIOC

void TIMx\_Init(uint16\_t arr, uint16\_t psc)

{

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseInitStructure;

NVIC\_InitTypeDef NVIC\_InitStructure;

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM2, ENABLE);

TIM\_TimeBaseInitStructure.TIM\_Period = arr;

TIM\_TimeBaseInitStructure.TIM\_Prescaler = psc;

TIM\_TimeBaseInitStructure.TIM\_ClockDivision = TIM\_CKD\_DIV1;

TIM\_TimeBaseInitStructure.TIM\_CounterMode = TIM\_CounterMode\_Up;

TIM\_TimeBaseInit(TIM2, &TIM\_TimeBaseInitStructure);

NVIC\_InitStructure.NVIC\_IRQChannel = TIM2\_IRQn;

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 0;

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 0;

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;

NVIC\_Init(&NVIC\_InitStructure);

TIM\_ITConfig(TIM2, TIM\_IT\_Update, ENABLE);

TIM\_Cmd(TIM2, ENABLE);

}

void LED\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOC, ENABLE);

GPIO\_InitStructure.GPIO\_Pin = LED\_PIN;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(LED\_PORT, &GPIO\_InitStructure);

}

void TIM2\_IRQHandler(void)

{

if (TIM\_GetITStatus(TIM2, TIM\_IT\_Update) != RESET)

{

GPIO\_WriteBit(LED\_PORT, LED\_PIN, (BitAction)(1 - GPIO\_ReadOutputDataBit(LED\_PORT, LED\_PIN)));

TIM\_ClearITPendingBit(TIM2, TIM\_IT\_Update);

}

}

int main(void)

{

LED\_Init();

TIMx\_Init(9999, 7199);

while (1)

{

}

}

```

在上述代码中，定时器 TIM2 的时钟频率为 72 MHz，通过设置预分频系数和自动重装值，使得定时周期为 500ms。在 TIM2 中断服务函数中翻转 LED 灯的状态即可实现 LED 灯的秒闪烁功能。

10. 要实现这个功能，需要使用STM32微控制器中的定时器和外部中断。具体实现步骤如下：

1)配置定时器TIM2

首先，需要将定时器TIM2配置为1秒一次中断的计时器。通过分析可知，STM32F103的系统时钟频率为72MHz，而TIM2定时器时钟频率为72MHz / 72 = 1MHz。因此，每计数1,000,000次就会产生1秒的中断，实现精确的1秒定时。

具体配置步骤：

(1) 使能TIM2定时器时钟，将TIM2计数模式设置为向上计数。

(2) 将预分频系数设置为71，即将1MHz的定时器时钟分频为1MHz / (71 + 1) = 10kHz。

(3) 将自动重装载值设置为9999，即当定时器计数值达到9999时，自动将计数值清零并产生更新事件。

(4) 使能TIM2的更新中断，并开启定时器计数。

2)配置外部中断

由于需要实现秒数值从0开始向上累加的功能，因此需要使用外部中断来监听定时器TIM2的更新事件，并在每次中断中累加秒数值。

具体配置步骤：

(1) 使能GPIO的时钟，将PA0引脚配置为浮空输入模式。

(2) 配置外部中断模式为上升沿触发模式，并使能外部中断。

(3) 在外部中断的中断服务函数中累加秒数值并更新数码管显示。

3)数码管显示

在每次秒数值更新后，需要将秒数值转换成数码管可以显示的形式，并将其显示在数码管上。

具体显示步骤：

(1) 使用数字转换函数将秒数值转换成6位数码管可以显示的BCD码。

(2) 将BCD码通过数码管驱动芯片输出到数码管上进行显示。

# 第十章

1.什么叫串行通信和并行通信？各有什么特点？

2.什么叫异步通信和同步通信？各有什么特点？

3.什么叫波特率？串行通信对波特率有什么基本要求？

4.串行通信按数据传送方向来划分共有几种制式？

5.试述串行通信常用的差错校验方法。

6.USArt 数据帧由哪些部分组成？

7.简述STM32F103微控制器USART 的主要特点。

8.概述STM32F103微控制器的USART 的内部结构。

9.简述STM32F103微控制器USART 串口初始化一般包含哪些步骤。

10.STM32F103微控制器的USArt 有哪些中断事件？可以产生哪些DMA请求？

11.已知异步通信接口的帧格式由1个起始位，8个数据位，无奇偶校验位和1位停止位组成。当该接口每分钟传送3600个字符时，试计算其波特率。

12.设计一个项目，使单片机能够通过串口接收到上位机发送过来的数据，对其加1再回传至上位机。

1. 串行通信和并行通信是计算机领域中常见的两种数据传输方式。串行通信是指将数据位逐个地传输，而并行通信是指将多个数据位同时传输。串行通信的特点是使用少量的信号线，但传输速率较慢，而并行通信则需要较多的信号线，但传输速率较快。

2. 异步通信和同步通信是两种常见的串行通信方式。异步通信是指在传输数据时，每个数据位都在一个时钟周期内传输，而时钟周期的长度并不是固定的。同步通信则是指在传输数据时，数据位的传输是依靠外部时钟信号进行同步的。异步通信的特点是传输速率相对较低，但传输距离可以较远，而同步通信的传输速率较高，但传输距离受到时钟信号传输的限制。

3. 波特率是指串行通信中单位时间内传输的比特数，通常用bps（bits per second）作为单位来表示。串行通信对波特率有一个基本要求，即接收端和发送端的波特率必须相同才能正确地进行数据传输。

4. 串行通信按照数据传输方向来划分共有两种制式，分别是单向制式和双向制式。单向制式是指数据只能在一个方向上进行传输，如RS-232C协议；而双向制式则是指数据可以在两个方向上进行传输，如I2C协议。

5. 串行通信常用的差错校验方法有奇偶校验、CRC校验和校验等。奇偶校验是指在传输数据时，在每个字节的最高位加上一个校验位，使得整个字节中1的个数为偶数或奇数，接收端在接收到数据后统计1的个数来判断是否出错。CRC校验是指在传输数据时，通过计算生成一个校验码，并将其与传输数据一起发送，接收端在接收到数据后同样进行计算生成校验码并与接收到的校验码进行比较来判断是否出错。校验和校验是指在传输数据时，在数据末尾添加一个校验和，接收端在接收到数据后同样计算校验和并与接收到的校验和进行比较来判断是否出错。

6. USART数据帧由以下部分组成：

- 起始位：逻辑0位，标识数据传输的开始

- 数据位：传输的数据，可以是5、6、7或8位

- 校验位：可选，用于检查数据的准确性

- 停止位：逻辑1位，标识数据传输的结束

7. STM32F103微控制器USART的主要特点包括：

- 支持多种通信模式：USART、SPI、I2C等

- 可以进行半双工或全双工通信

- 可以配置不同的波特率、数据位数、停止位数、奇偶校验等参数

- 支持中断和DMA传输

- 可以同时与多个外设通信

8. STM32F103微控制器的USART内部结构包括以下部分：

- 数据缓冲区：用于存储接收或发送的数据

- 波特率发生器：用于产生不同的波特率

- 控制寄存器：用于配置通信参数和控制USART的工作状态

- 状态寄存器：用于记录USART的状态和中断标志位

9. STM32F103微控制器USART串口初始化一般包括以下步骤：

- 配置GPIO引脚为USART的功能引脚

- 配置USART的通信参数，如波特率、数据位数、停止位数、奇偶校验等

- 使能USART模块并配置中断和DMA传输

10. STM32F103微控制器的USART有以下中断事件：

- 接收寄存器非空中断

- 发送寄存器空中断

- 帧错误中断

- 奇偶校验错误中断

- 传输完成中断

USART可以产生以下DMA请求：

- 发送DMA请求

- 接收DMA请求

11. 题目给出每分钟传送3600个字符，因此传输速率为3600个字符/60秒 = 60个字符/秒。由于该帧格式包含10位，因此实际数据传输速率为60/10=6波特。

这个项目需要实现串口接收和发送功能，以及数据的加1处理。以下是一个简单的实现过程：

1. 硬件准备：连接开发板的USART串口引脚（TX、RX）与上位机相应的串口引脚。

2. 在STM32CubeMX中配置串口通信的时钟和引脚，生成代码。

3. 在生成的代码中找到相应的USART初始化函数，根据需求进行参数配置。

4. 在主程序中加入串口接收和发送的代码。

第十一章

1.通常，SPI接口由哪几根线组成？它们分别有什么作用？

2.SPI接口的连接方式有几种？分别画出其连接示意图。

3.SPI的数据格式有哪几种？传输顺序可分为哪几种？

4.在SPI时序控制中，CPOL和CPHA的不同取值对时序有什么影响？

5.简述STM32F103微控制器SPI主要特点。

6.什么是OLED?OLED显示屏的主要特点是什么？

7.SM32F103微控制器与OLED显示屏的SPI连接方式有哪两种？

8.OLED12864显示屏共划分多少行？每行有多少个点？整个屏幕共有多少像素点？

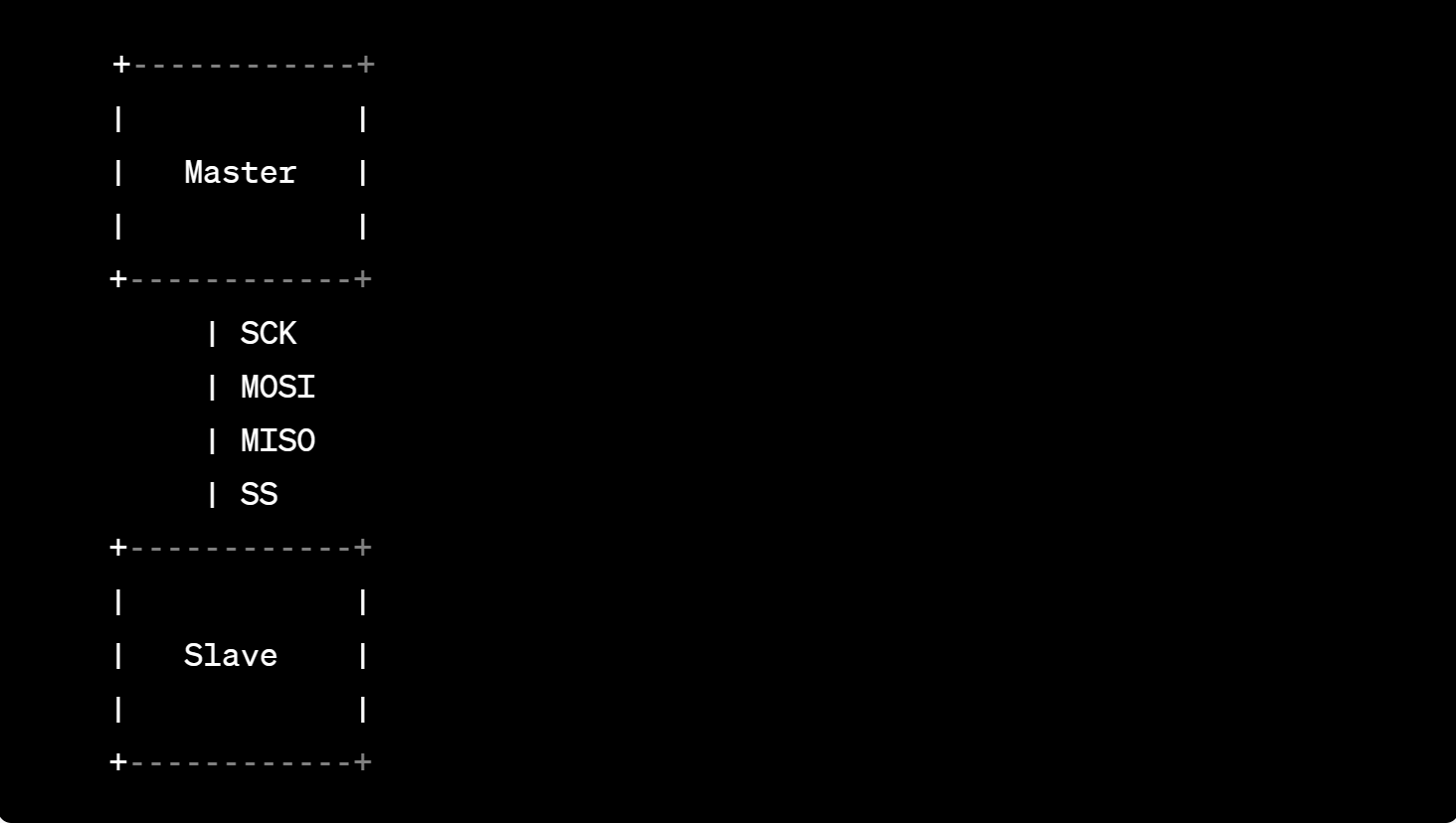
9.如何进行普通汉字 (16\*16) 显示？取字模时应如何进行字模选项设置？

10.如何进行大号汉字 (32\*32) 显示？取字模时应如何进行字模选项设置？

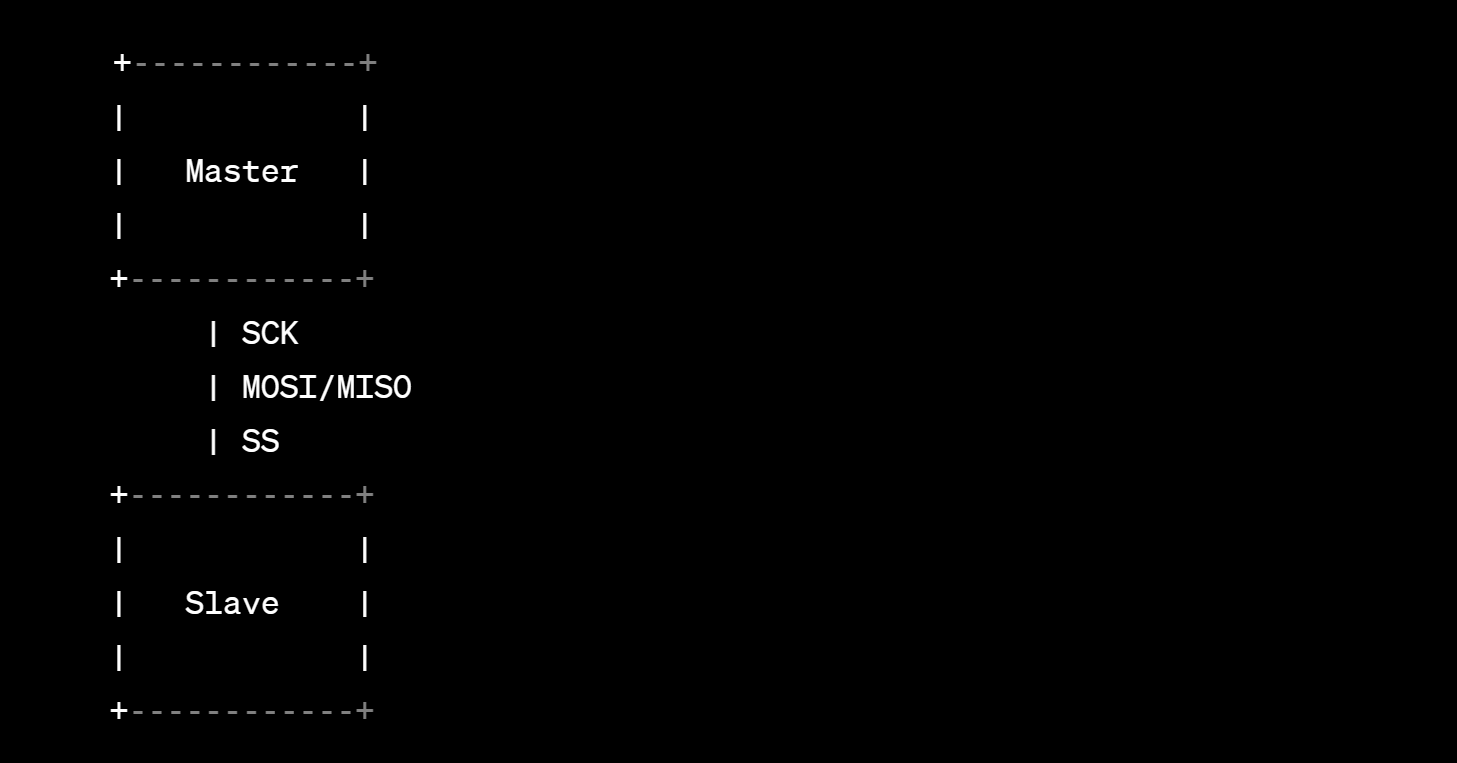
1. SPI接口通常由四根线组成：主设备的时钟线（SCK）、主设备输出的数据线（MOSI）、主设备输入的数据线（MISO）和从设备的片选线（SS）。其中，SCK是时钟信号，MOSI是主设备输出从设备输入的数据信号，MISO是从设备输出主设备输入的数据信号，SS是片选信号。

2. SPI接口的连接方式一般有三种：单向全双工、半双工和双向全双工。其中，单向全双工和双向全双工需要四根线，而半双工只需要三根线。下图为这三种连接方式的示意图：

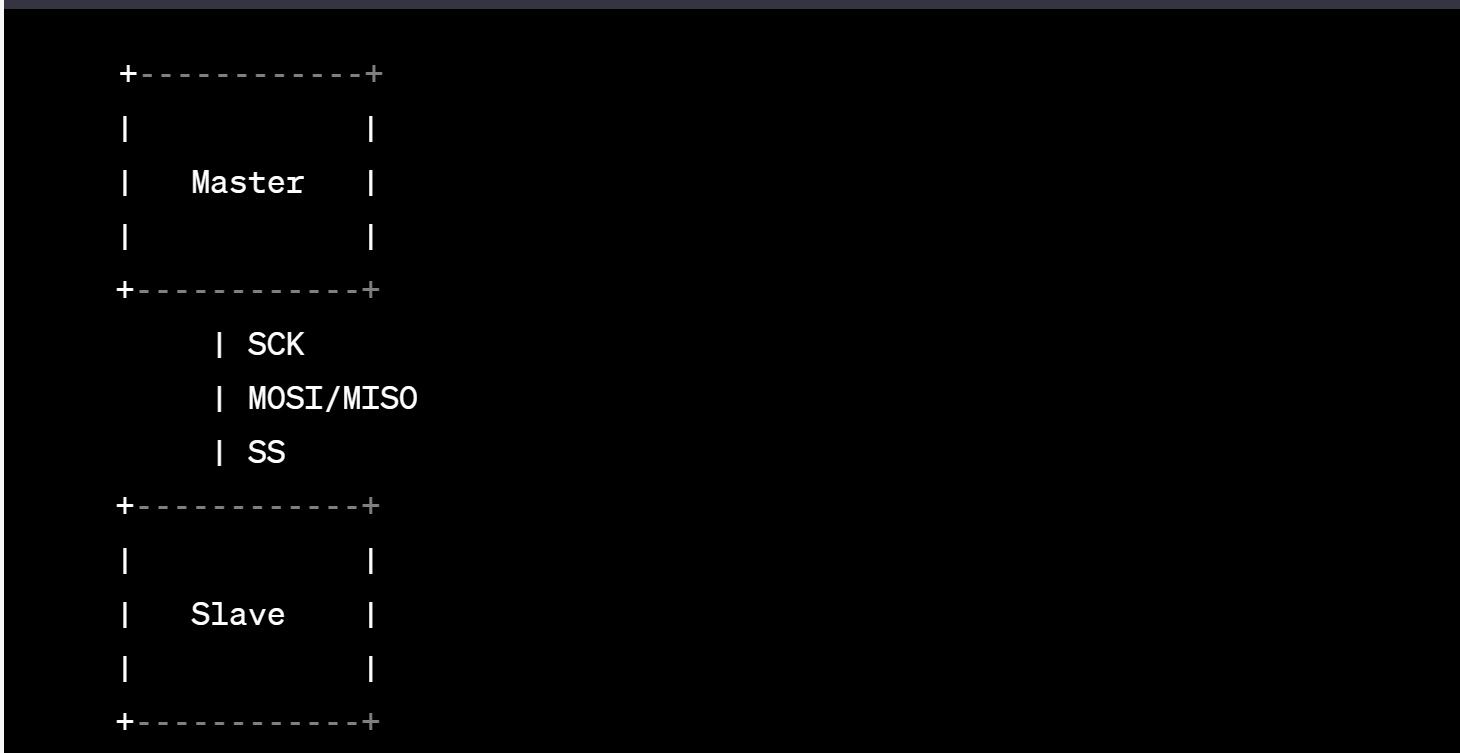
单向全双工：



半双工：



双向全双工：



3. SPI的数据格式有两种：帧格式和数据格式。帧格式是指数据帧的格式，一般包括数据位数、传输顺序、时钟极性和时钟相位等。常见的数据位数有8位、16位和32位。数据格式是指数据的编码方式，一般包括两种：二进制补码和反码。传输顺序可以分为两种：MSB（最高位优先）和LSB（最低位优先）。

4. SPI时序控制中，CPOL和CPHA的不同取值对时序的影响如下：

- CPOL（Clock Polarity）为时钟极性，决定时钟信号的空闲状态是高电平（CPOL=1）还是低电平（CPOL=0）。

- CPHA（Clock Phase）为时钟相位，决定数据采样的时机是在时钟的上升沿（CPHA=0）还是下降沿（CPHA=1）。

不同的CPOL和CPHA取值组合会影响SPI总线的数据传输顺序和时钟相位。常见的四种模式为：Mode 0、Mode 1、Mode 2和Mode 3。其中，Mode 0和Mode 3的CPOL都为0，Mode 1和Mode 2的CPOL都为1，Mode 0和Mode 2的CPHA为0，Mode 1和Mode 3的CPHA为1。

5. STM32F103微控制器SPI的主要特点包括：

- 支持SPI1和SPI2两个外设，每个外设最多可连接2个从设备。

- 支持主从模式和全双工模式，可根据应用场景选择合适的模式。

- 支持四种SPI模式（Mode 0、Mode 1、Mode 2、Mode 3），可根据外设特性进行配置。

- 支持硬件控制和软件控制两种片选方式，灵活性较高。

- 支持DMA传输和中断处理两种数据传输方式，可根据应用场景进行选择。

6. OLED（Organic Light-Emitting Diode）是一种新型的显示技术，它采用有机材料作为发光材料，具有自发光、反应速度快、可视角度大等特点。OLED显示屏的主要特点包括：

- 自发光，不需要背光源，显示效果更加鲜明、清晰。

- 可视角度大，对于不同角度的观察，显示效果不会受到影响。

- 反应速度快，对于动态图像的显示，响应速度更快。

- 厚度薄，体积小，方便携带。

7. STM32F103微控制器与OLED显示屏的SPI连接方式有以下两种：

- 4线SPI接口：使用SCK、MOSI、MISO和CSN四根线进行连接。

- 3线SPI接口：使用SCK、SDA和CSN三根线进行连接。

8. OLED12864显示屏共划分64行，每行128个点，整个屏幕共有8192像素点。

9. 普通汉字(16x16)的显示可以通过取汉字的字模来实现。常用的方法是先在PC上用字模软件制作好汉字的字模，然后将字模转化为16进制数，并将其存放在单片机的Flash或RAM中。在程序中通过读取相应的字模数据，然后按照具体的显示方式将其显示在OLED屏幕上。在取字模时，需要根据OLED屏幕的显示方向和字模的存储方式进行相应的选项设置。

10. 大号汉字(32x32)的显示方式与普通汉字类似，只是字模的大小不同。同样需要先在PC上用字模软件制作好汉字的字模，然后将字模转化为16进制数，并将其存放在单片机的Flash或RAM中。在程序中通过读取相应的字模数据，然后按照具体的显示方式将其显示在OLED屏幕上。在取字模时，需要根据OLED屏幕的显示方向和字模的存储方式进行相应的选项设置。需要注意的是，由于大号汉字的字模较大，所以在存储和读取时需要考虑数据的存储方式和读取速度等问题。

下面是STM32外部中断的一般配置步骤：

1. 初始化IO口为输入。使用GPIO\_Init()函数初始化对应的IO口，并设置为输入模式。

2. 开启IO口复用时钟。对于一些特殊的IO口（如PE4~PE7），需要打开AFIO时钟，以便可以配置它们的复用功能。使用RCC\_APB2PeriphClockCmd()函数开启相应的时钟。

3. 设置IO口与中断线的映射关系。使用GPIO\_EXTILineConfig()函数将IO口与对应的中断线联系起来。

4. 初始化线上中断，设置触发条件等。使用EXTI\_Init()函数初始化对应的中断线，设置中断触发类型（上升沿、下降沿、双边沿等），并使能中断线。

5. 配置中断分组（NVIC），并使能中断。使用NVIC\_Init()函数配置中断分组，将对应的中断线使能。

6. 编写中断服务函数。在中断服务函数中编写具体的处理代码。

7. 清除中断标志位。在中断服务函数中，需要调用EXTI\_ClearITPendingBit()函数清除中断标志位，以便下一次中断可以正确触发。

需要注意的是，不同型号的STM32微控制器可能会略有不同，具体的配置步骤还需根据对应的数据手册进行参考。