实验一：非操作系统下的实验

实验目的

1. 熟悉 ADS 软件的使用
2. 熟悉程序的下载和调试流程
3. 熟悉 GPIO 的操作掌握 Arm 在非不使用操作系统下的的使用，理解一般的操作方法，蜂鸣器，中断的使用。
4. 熟悉 ARM 的中断操作
5. 掌握按键检测的方法
6. 熟悉 ARM 的定时器操作
7. 熟悉 ARM 的 PWM 操作
8. 熟悉 ARM 的串口操作

实验原理

Nor/Nand 选择开关

当从 Nand Flash 启动时,请在实验平台开机前将该开关拨到靠近绿色接口的地方。

音频输入输出接口

TQ2440 提供的音频接口完全按照标准接口提供,绿色为音频输出接口,红色为音频输 入口。

串口接口(串口 0(RS232)和三个串口扩展接口(3.3V 电平))

串口在 TQ2440 实验平台的使用中是非常重要的接口之一,TQ2440 可以通过它用 PC 和 实验平台直接进行交互操作、传输数据、完成调试等。

TQ2440 实验平台提供的标配串口线是直连串口线,在使用实验平台时请把串口线的一

段接实验平台的串口接口,另外一端接 PC 的串口接口,然后就可以通过串口进行交互等操 作了。

三串口扩展接口引出的串口的 TX 和 RX 引脚均是 3.3V 电平,如果需要使用 232 电平, 请扩展 MAX3232 或者 SP3232EEN 芯片转成 232 电平。

复位按钮

硬件重启实验平台时使用。

Jtag 接口

在 TQ2440 实验平台中,Jtag 的用途是当 Nand Flash 或 Nor Flash 中没有 uboot 时,使用 它烧写 uboot 进去;或者是进行仿真时使用它。 使用时请接上 TQ2440 提供的 Jtag 板到实验平台的 Jtag 接口和 PC 的并口接口,然后再 使用 Jtag 软件进行烧写或仿真操作。 说明:当不使用 Jtag 烧写时,请拔掉 Jtag 线。

用户按钮

使用中断功能的 4 个用户按钮。

AD 输入测试电阻

在 Linux 或无 OS 中可以测试 ADC 接口(AIN2 管脚)。

蜂鸣器(PWM 控制)

PWM 控制的蜂鸣器。

蜂鸣器/LCD 电压选择接口 打开或关断蜂鸣器的电源,选择 LCD 供电电压。

用户 LED 灯

使用 GPIO 口控制的 LED 灯。

LCD 接口(LCD 接口 1 和 2)

在接 LCD 接口时请注意排线接口的定位方向,请不要接反了。 说明:这两个接口是使用的相同的数据线和控制线,唯一不同的是一个是 FPC 接口而另 外一个是插针形式的接口。 u 电源指示灯 正常开机后,将会点亮红色的 LED 灯。

电源开关

控制着整个实验平台的供电,需要开机时请拨动该开关到靠近电源指示灯的方向,关机 时拨动该开关到反方向。

实验仪器与设备

1. 电脑
2. Jtag 调试器
3. Arm 开发板
4. 触摸屏
5. 触摸笔

实验步骤

GPIO 接口实验

1．实现控制 LED 开启关闭功能

(1)　单步调试例程，也可以双击程序中的某一行添加断点，则全速运行后程序会停在断点处。并且在调试过程中观察 LED 的变化。写出控制 LED 开启和关闭的程序语句和对应 LED 的状态，填入下表当中

序号	程序	现象
1	reg_tmp=rGPBCON;	LED 灯全灭
2	rGPBDAT|=((1<<5)|(1<<6)|(1<<7)|(1<<8));	LED 灯全灭

(2)　例程中实现的是对一个 LED 的控制，修改例程，写出一个对所有 LED 控制的通用函数，该函数的参数有两个：LED 的号码和 LED 的开关状态。写出该函数并且写出调用该函数控制 4 个 LED 状态的语句和对应现象，填入下表中：

序号	程序	现象

1	void ControlLED(unsigned int x,unsigned int state)
	{
	if(state==1)
	{
	rGPBDAT=rGPBDAT&(~(1<<x));
	//PORTB[n]为低电平
	}
	Else
	{
	rGPBDAT=rGPBDAT|(1<<x);
	//PORTB[n]为高电平
	}
	
	}	

2	rGPBDAT=rGPBDAT&(~(1<<x)); //PORTB[n]为低电平	LED 灯全灭
3	rGPBDAT=rGPBDAT|(1<<x); //PORTB[n]为高电平	LED 灯全灭

2. 实现流水灯功能

(1) 利用上面编写的通用的 LED 控制函数，实现 LED 依次亮灭的过程(流水灯)，把流水灯控制的语句填入下表中并描述 LED 的变化

序号	程序	现象

1	while(1)
	{
	for(n=5;n<=8;n++)
	{
	ControlLED(n,1);
	delay(1);
	ControlLED(n,0);
	delay(1);
	}
	}
		LED 灯一闪烁

3. 实现控制蜂鸣器功能

(1) 跟 LED 的控制类似，蜂鸣器的控制也是通过控制 IO 输出电平来控制的。蜂鸣器使用的是 GPB0，只需要对该端口操作即可

(2) 写出控制蜂鸣器鸣叫和静音的语句，并把实验现象填入下表当中

序号	程序	现象

1	rGPBUP|=((1<<1)|(1<<2)|(1<<3)|(1<<4)); //禁止上拉电阻
	rGPBDAT|=((1<<1)|(1<<2)|(1<<3)|(1<<4)); //输出全为高电平 jingyin
		蜂鸣器响

4. 实现按键控制 LED 功能

(1) 写出查询按健状态函数，并根据不同的按键控制不同的 LED 亮
(2) 按键使用的是 IO 口是 GPF0、GPF1、GPF2 和 GPF4。需要将其配置为输入的模式，然后读取数据寄存器即可
(3) 把按键检测的程序段和对应的现象写入下表中

序号	程序	现象

1	rGPBDAT=rGPBDAT&(~0x1E0);
	if((rGPFDAT&0x17) == 0x07)
	{
	rGPBDAT=rGPBDAT|(1<<5); //PORTB[5]为高电平
	delay(1);
	}
	if((rGPFDAT&0x17) == 0x16)
	{
	rGPBDAT=rGPBDAT|(1<<6); //PORTB[6]为高电平
	delay(1);
	}
	if((rGPFDAT&0x17) == 0x15)
	{
	rGPBDAT=rGPBDAT|(1<<7); //PORTB[7]为高电平
	delay(1);
	}
	if((rGPFDAT&0x17) == 0x13)
	{
	rGPBDAT=rGPBDAT|(1<<8); //PORTB[8]为高电平
	delay(1);
	}
		按下不同的键的时候，不同的灯会亮。

中断实验

1. 打开实验代码文件夹中的 irq_test 子文件夹中的工程 irq_test.mcp 工程
2. 阅读代码，单步执行，体会 void KeyScan_Test(void)函数里面对中断相关寄存器的设置方法和意义，了解中断服务函数 static void __irq Key_ISR(void)中对相关寄存器的设置方法
3. 注意：要进入中断服务函数，必须全速执行程序，不能单步执行。因此，可以在中断服务函数 static void __irq Key_ISR(void)添加一个断点，然后点击  全速执行程序。当按下按键后，程序就会停在断点处
4. 例程中只给出外部中断 0 和外部中断 4 的程序，只能实现对按键 K2 和 K4 的检测，仿照对外部中断 0 的配置，完成对按键 K1 和 K3 的检测，写出对外部中断 1 和 2 的配置代码还有对 LED 的控制程序，并进行调试，把代码和现象列入下表中

序号	程序	现象

1	rEXTINT0 &= ~(7<<0|(7<<16)|(7<<8)|(7<<4));
	
	rEXTINT0 |= (2<<0|(2<<16)|(2<<8)|(2<<4));
	//set eint0,1,2,4 falling edge int
	rEINTPEND |= (1<<4); //clear eint 4
	rEINTMASK &= ~(1<<4); //enable eint 4
	ClearPending(BIT_EINT0|BIT_EINT1|BIT_EINT2|BIT_EINT4_7);
	pISR_EINT0 = pISR_EINT1 = pISR_EINT2 = pISR_EINT4_7 = (U32)Key_ISR;
	EnableIrq(BIT_EINT0|BIT_EINT1|BIT_EINT2|BIT_EINT4_7);
	
	while(Uart_GetKey() != ESC_KEY) ;
	DisableIrq(BIT_EINT0|BIT_EINT1|BIT_EINT2|BIT_EINT4_7);
		无特别现象。实现中断 1 和 2 的配置。

2	同上。	

3	if((reg_tmp&(1<< 0)) == 0)
	{
	rGPBDAT = rGPBDAT & ~(LED4);
	//亮 LED4
	}
	else if((reg_tmp&(1<< 4)) == 0)
	{
	rGPBDAT = rGPBDAT & ~(LED2);
	//亮 LED2;
	}
	else if((reg_tmp&(1<< 1)) == 0)
	{
	rGPBDAT = rGPBDAT & ~(LED1);
	//亮 LED3;
	}
	else if((reg_tmp&(1<< 2)) == 0)
	{
	rGPBDAT = rGPBDAT & ~(LED3);
	//亮 LED1;
	}
	else
	{
	return 0xff;
	}
		按下不同的中断会有不同的灯亮

定时器和 PWM 实验

1. 打开实验代码文件夹中的 Song_test 子文件夹中的工程 song_test.mcp 工程
2. 单步调试程序，找出基本音符的放音语句，把响应的语句和现象列入下表中

序号	程序	现象
1	Buzzer_Freq_Set0(260);	do
2	Buzzer_Freq_Set0(294); □	ra

3	Buzzer_Freq_Set0(328);	mi
4	Buzzer_Freq_Set0(347); □	fa
5	Buzzer_Freq_Set0(390);	sa
6	Buzzer_Freq_Set0(438);	la
7	Buzzer_Freq_Set0(490);	xi

3. 理解蜂鸣器唱歌的过程，全速运行程序，聆听蜂鸣器唱歌
4. 更改唱歌的内容，播放另外两首歌

串口实验

1. 打开实验代码文件夹中的 uart_test 子文件夹中的工程 uart_test.mcp 工程
2. 阅读代码，单步执行，体会 Uart0_SendByte 和 Uart0_Getch()函数里面对相关寄存器的设置方法和意义，实现基本的串口收发功能，并且修改收发的数据，把对应的程序和现象记录下来

序号	程序	现象
1	Uart0_SendByte('5');	在电脑上显示 5
2	ch=Uart0_Getch();	在电脑上显示出输入的字符。

3. 修改程序，实现一个字符串的发送功能，把程序和现象记录下来

序号	程序	现象

1	char str[]="abcdefghij123456"; init(); Uart0_Init(115200); Uart0_SendString(str);	在电脑上显示出abcdefghij123456
2		

4. 修改代码，实现 PC 通过串口控制 LED 的功能，把程序和现象记录下来

序号	程序	现象

1	while(1){
	ch=Uart0_Getch();
	Uart0_SendByte(ch);
	if(ch=='y')
	{
	rGPBDAT&=~0x1E0;
	}
	else
	{
	rGPBDAT|=0x1E0;
	}
	}
		通过键盘按下'y'LED 亮，反之则 LED 灭

2		

实时时钟实验

1. 打开实验代码文件夹中的 rtc_test 子文件夹中的工程 rtc_test.mcp 工程
2. 阅读代码，单步执行，体会 void RTC_Time_Set(void)和 void RTC_Display(void)函数里面对相关秒寄存器的写入和读取过程，通过串口观察到的结果。
3. 增加对年、月、日、时、分寄存器的写入和读取功能，并通过串口观察修改后的时间。把代码和相应的现象写入下表

序号	程序	现象

1	void RTC_Time_Set(void)
	{
	rRTCCON = 1 ; //RTC read and write enable
	
	rBCDYEAR = 0x08 ; // 年闰年测试
	rBCDMON = 0x02 ; //月
	rBCDDATE = 0x28 ; // 日
	
	rBCDDAY = 0x05 ; // 星期
	rBCDHOUR = 0x23 ; // 小时
	rBCDMIN = 0x59 ; //分
	rBCDSEC = 0x50 ; //秒
	
	rRTCCON &= ~1 ; //RTC read and write disable
	}
		在电脑上看到显示出时间

2	void RTC_Display(void)
	{
	U16 year ;
	U8 month, day ; // week
	U8 hour, minute, second ;
	
	rRTCCON = 1 ; //RTC read and write enable
	
	year = 0x2000+rBCDYEAR ;
	//年
	month = rBCDMON ; //月
	day = rBCDDATE ; // 日
	
	hour = rBCDHOUR ; // 小时
	minute = rBCDMIN ; //分
	second = rBCDSEC ; //秒
	
	rRTCCON &= ~1 ; //RTC read and write disable
	
	Uart_Printf("RTC time : %04x-%02x-%02x %02x:%02x:%02x\n", year, month, day, hour, minute, second);
	Delay(900) ;
	
	}
		在电脑上看到显示出时间

4. 修改工程，改变时间设置值，尝试设置闰年和非闰年，观察实验结果，把把代码和相应的现象写入下表

序号	程序	现象

1	void RTC_Time_Set(void)
	{
	rRTCCON = 1 ; //RTC read and write enable
	
	rBCDYEAR = 0x08 ; // 年闰年测试
	rBCDMON = 0x02 ; //月
	rBCDDATE = 0x28 ; // 日
	
	rBCDDAY = 0x05 ; // 星期
	rBCDHOUR = 0x23 ; // 小时
	rBCDMIN = 0x59 ; //分
	rBCDSEC = 0x50 ; //秒
	
	rRTCCON &= ~1 ; //RTC read and write disable
	}
		在电脑上看到显示出时间

2	void RTC_Time_Set(void)
	{
	rRTCCON = 1 ; //RTC read and write enable
	
	rBCDYEAR = 0x09 ; // 年闰年测试
	rBCDMON = 0x02 ; //月
	rBCDDATE = 0x28 ; // 日
	
	rBCDDAY = 0x05 ; // 星期
	rBCDHOUR = 0x23 ; // 小时
	rBCDMIN = 0x59 ; //分
	rBCDSEC = 0x50 ; //秒
	
	rRTCCON &= ~1 ; //RTC read and write disable
	}
		在电脑上看到显示出时间

看门狗实验

1. 打开实验代码文件夹中的 wd_test 子文件夹中的工程 wd_test.mcp 工程
2. 阅读代码，了解看门狗的启动方式和喂狗的方法
3. 修改工程，比较有喂狗和没有喂狗的实验结果，把程序和相应的现象列入下表中。注意：要观察看门狗的复位，必须全速运行程序，不能单步执行

序号	程序	现象

1	//流水灯
	for(i=5;i<=8;i++)
	{
	led_con(i,ON);
	rWTCNT=2000; //喂狗
	delay(1);
	led_con(i,OFF);
	rWTCNT=2000; //喂狗
	delay(1);
	}
		LED 一直闪烁

2	//流水灯
	for(i=5;i<=8;i++)
	{
	led_con(i,ON);
	//rWTCNT=2000; // 喂狗
	delay(1);
	led_con(i,OFF);
	//rWTCNT=2000; //喂狗
	delay(1);
	}
		Cause： The processor is reset
		LED 闪一会儿就停了

思考题解答

- **从程序编写的角度,比较 51 单片机和 SC32440 的 IO 操作的不同之处**

  51 单片机的引脚都简单。往往可以直接对引脚做赋值。而 SC32440 的 IO 口操作相对麻烦，如果进行位操作还需要掌握编程技巧---通过逻辑运算来实现位操作。

- **比较实验 2.2 和实验 2.3,要实现检测按键被按下的功能,采用中断和查询的方法哪个响应 的速度会更快?两者有什么优缺点?**

  中断的响应更快。但使用查询来实现程序很简单，所以在程序很简单的时候，使用查询来实现，而在程序执行很长的时候选择中断。

- **简述 PWM 技术的其他应用例子及其原理**

  脉宽调制(PWM)是利用微处理器的数字输出来对模拟电路进行控制的一种非常有效的技术，广泛应用在从测量、通信到功率控制与变换的许多领域中。

  通过数子电路控制模拟电路，通信脉冲调制，模拟电路容易随时间漂移，因而难以调节。可以通过通过数子电路控制模拟电路，还可以进行通信脉冲调制等。

总之，PWM 既经济、节约空间、抗噪性能强，是一种值得广大工程师在许多设计应用中使用的有效技术。

- **如何采用带 FIFO 的串口进行数据收发**

  用 FIFO 发送数据时，需要保证接受端的 FIFO 不溢出，则发送过快，接受端 FIFO 满后中断末取走数据，发送端就又发数据。其只能由发送端来保证。

在实验总结

通过本次实验我了解到了很多使用的 Arm 相关的知识和技术。

在调试程序中不可避免的会遇到许多问题，一定要耐心排查才能解决。比如中断的概念就很难理解，也特别容易出错。

第二次实验： μC/OS II 操作系统下的实验

实验目的

掌握操作系统的基本原理，μC/OS II 操作系统下的基本操作方法熟悉 μC/OS II 多任务建立方法, 以及任务间的同步。

实验原理

一个较为复杂的应用程序时,也通常把一个大型任务分解成多个小任务, 然后在计算机中通过运行这些小任务,最终达到完成大任务的目的,这种做法也使用应用程 序的维护变得方便起来。因此,现代操作系统几乎都是对任务操作系统。

在 μC/OS-II 中,与上述小任务对应的是程序实体就叫做“任务”(实质是一个线程)。μ C/OS-II 就是一个能对这些小任务进行管理和调度的多任务操作系统。

从应用程序设计的角度来看,μC/OS-II 的任务就是一个线程,就是一个用来解决用户 问题的 C 语言函数和与之相关联的一些数据结构而构成的一个实体。

μC/OS-II 的任务有两种:用户任务和系统任务。有应用程序设计者编写得任务,叫做 用户任务;有系统提供的任务叫做系统任务。用户任务是为解决应用问题而编写的;系统任 务是为应用程序来提供某种服务的。

嵌入式系统中的各个任务都是以并发的方式来运行的,并为同一个大的任务服务。它们 不可避免地要共同使用一些共享资源,并且在处理一些需要多个任务共同协作来完成的工作 时,还需要相互的支持和限制。因此,对于一个完善的多任务操作系统来说,系统必须具有 完各的同步和通信机制。

任务之间这种制约性的合作运行机制叫做任务间的同步。系统中任务的同步是依靠任务 与任务之间互相发送消息来保证同步的。

实验仪器与设备

1. 电脑
2. Jtag 调试器
3. Arm 开发板
4. 串口线

实验步骤

µC/OS II 多任务建立实验

1. 打开实验代码文件夹中的 ucos_task_test 子文件夹中的工程 ucos_task_test.mcp 工程
2. 阅读代码，熟悉任务的建立方法
3. 修改例程代码，建立两个任务，程序结构如图 3-1 所示。

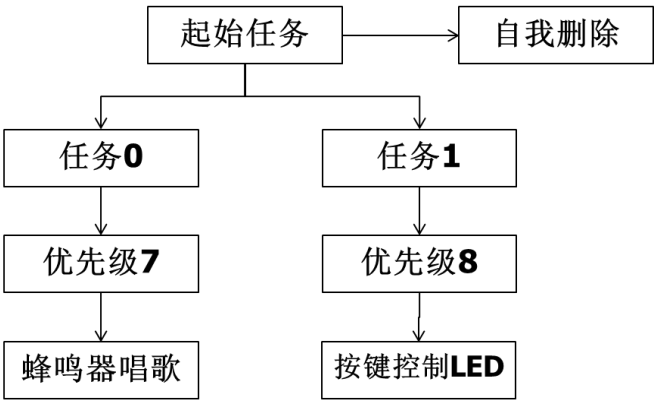

图 3-1 多任务示意图

a) 修改工程，建立任务 0：蜂鸣器唱歌任务(注意，单个音符的延迟应该采用 µC/OS II 的延迟函数 OSTimeDly 来实现，不用自己写延迟函数)，进行调试，直到能够正常唱歌
b) 建立任务 1：仿照“2.2.4 按键控制 LED 实验”建立按键检测 LED 控制任务
c) 全速运行程序，观察在唱歌的同时，按下按键，LED 能否发生变化
d) 把建立任务的关键代码、两个任务的代码列出来，填入下表中

序号	程序	现象
1	OSTaskCreate (Task0,(void *)0, &Task0Stk[Task0StkLengh - 1], 7); OSTaskCreate (Task1,(void *)0, &Task1Stk[Task1StkLengh - 1], 8);	建立任务的关键代码。

2	//任务 1，按键控制 LED 闪烁
	void Task1(void *pdata)
	
	{
	U16 task1Cnt=0;
	U32 led;
	rGPFCON&=~(0x3FF);
	led=0x1E0;
	while(1)
		按键后 LED 灯熄灭

	{
	if((rGPFDAT&0x17)!=0x17)
	{
	rGPBDAT&=~0x1E0;
	}
	else
	{
	rGPBDAT|=0x1E0;
	
	}
	OSTimeDly(50);
	}
	}	

3	void Task0(void *pdata)
	
	{
	unsigned char Temp1,Temp2;
	U16 freq;// lci 1000
	unsigned int Addr=0;
	
	while(1)
	{
	Temp1=SONG[Addr++];
	Temp2=SONG[Addr++];
	
	switch (Temp1)
	{
	case 01: freq=260;
	break;
	case 02: freq=294;
	break;
	case 03: freq=328;
	break;
	case 04: freq=347;
	break;
	case 05: freq=390;
	break;
	case 06: freq=438;
	break;
	case 07: freq=490;
	break;
	
	case 11: freq=520;
	break;
	case 12: freq=581;
	break;
	case 13: freq=657;
	break;
	case 14: freq=694;
	break;
	case 15: freq=781;
	break;
	case 16: freq=892;
		会播放歌曲 但速度有时快 有时慢

	break;
	case 17: freq=1000;
	break;
	
	case 21: freq=1041;
	break;
	case 22: freq=1190;
	break;
	case 23: freq=1315;
	break;
	case 24: freq=1388;
	break;
	case 25: freq=1562;
	break;
	case 26: freq=1785;
	break;
	case 27: freq=1923;
	break;
	default: freq='e';break;
	}
	Buzzer_Freq_Set0(freq);
	OSTimeDly(120*Temp2);
	if(Addr>440) //440，此为数字数据个数，歌唱不同的歌曲的时候需要进行调整
	{
	Addr=0;
	}
	}
	}	

4		

μC/OS II 任务间通信实验

1. 打开实验代码文件夹中的 ucos_resource_compete 子文件夹中的工程 ucos_resource_compete.mcp 工程
2. 阅读代码，熟悉任务的信号量的建立方法
3. 修改例程代码，建立三个任务，程序结构如图 3-2 所示
   a) 修改工程，建立任务 0：蜂鸣器唱歌任务(注意，单个音符的延迟应该采用 μC/OS II 的延迟函数 OSTimeDly()来实现，不用自己写延迟函数)，进行调试，知道能够正常唱歌
   b) 建立任务 1：仿照“2.3.4.3 按键控制 LED 实验”建立按键检测 LED 控制任务，并且创建信号量，当检测到按键按下后，通过 OSSemPost()函数发出一个信号量

c) 建立任务 2：利用 OSSemPend ()函数等待信号量，当等待到信号量后通过串口输出相应的按键号

d) 全速运行程序，观察在唱歌的同时，按下按键，串口能否产生相应的输出

e) 把三个任务的代码列出来，填入下表中

序号	程序	现象

1	void Task0(void pdata)
	
	{
	unsigned char Temp1,Temp2;
	U16 freq; lci 1000
	unsigned int Addr=0;
	
	while(1)
	{
	Temp1=SONG[Addr++];
	Temp2=SONG[Addr++];
	
	switch (Temp1)
	{
	case 01 freq=260;
	break;
	case 02 freq=294;
	break;
	case 03 freq=328;
	break;
	case 04 freq=347;
	break;
	case 05 freq=390;
	break;
	case 06 freq=438;
	break;
	case 07 freq=490;
	break;
	
	case 11 freq=520;
	break;
	case 12 freq=581;
	break;
	case 13 freq=657;
	break;
	case 14 freq=694;
	break;
	case 15 freq=781;
	break;
	case 16 freq=892;
	break;
	case 17 freq=1000;
	break;
	
	case 21 freq=1041;
	break;
	case 22 freq=1190;
		蜂鸣器唱歌《老鼠爱大米》

	break;
	case 23 freq=1315;
	break;
	case 24 freq=1388;
	break;
	case 25 freq=1562;
	break;
	case 26 freq=1785;
	break;
	case 27 freq=1923;
	break;
	default freq='e';break;
	}
	Buzzer_Freq_Set0(freq);
	OSTimeDly(120Temp2);
	if(Addr440) 440，此为数字数据个数，歌唱不同的歌曲的时候需要进行调整
	{
	Addr=0;
	}
	}
	
	}	

2	rGPBDAT=rGPBDAT&(~0x1E0);
	if((rGPFDAT&0x17) == 0x07)
	{
	rGPBDAT=rGPBDAT|(1<<5); //PORTB[5]为高电平
	delay(1);
	key=1;
	Count=count+1;
	OsSemPost(Semp)
	
	}
	
	if((rGPFDAT&0x17) == 0x16)
	{
	rGPBDAT=rGPBDAT|(1<<6); //PORTB[6]为高电平
	delay(1);
	key=2;
	Count=count+1;
	OsSemPost(Semp)
	}
	
	if((rGPFDAT&0x17) == 0x15)
	{
	rGPBDAT=rGPBDAT|(1<<7); //PORTB[7]为高电平
	delay(1);
	key=3;
	Count=count+1;
	OsSemPost(Semp)
	
	}
	
	if((rGPFDAT&0x17) == 0x13)
	{
	rGPBDAT=rGPBDAT|(1<<8); //PORTB[8]为高电平
	delay(1);
	key=4;
	Count=count+1;
	OsSemPost(Semp)
	
	}
		按下不同的键的时候，不同的灯会亮。

3	void Task2(void pdata)
	{
	while (1)
	{
	OSSemPend(Semp,0,&err);
	
	OSPrintf("key=%d,count=%d/n",key,cou
	nt);
	OSSemPost(Semp);
	}
	}
		按下键的后，会传过去对应的按键和按键的次数并且 print。

4		

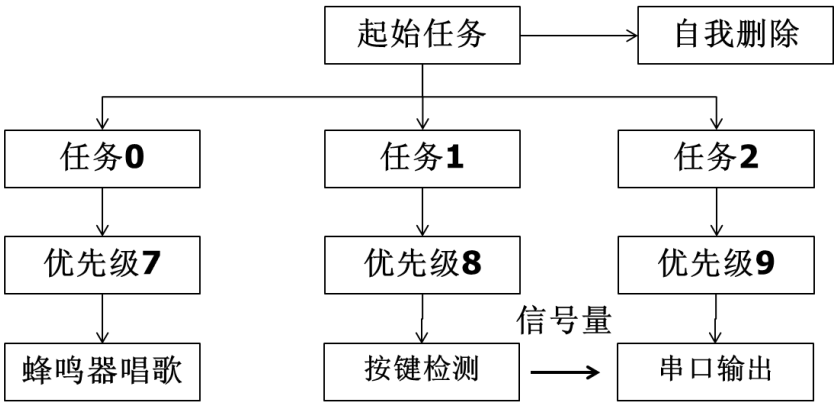

图 3-2 信号量实现任务间通信示意图

思考题解答

- **如果不采用 µC/OS II 的延迟函数 OSTimeDly()来实现,而用自己写延迟函数来实验延迟,能 否实现一边放歌一边检测按键按下的功能?**

  不能。因为 uC/OS II 的多进程是通过在一个任务延迟的时候来实现。

- **列举出 µC/OS II 的任务间通信的方式,并比较它们的异同。**

  信号量：只能传一个数据，0 和 1。

  邮件：传递消息

  信号量集：传递一系列的信号量。

实验总结

通过本次实验我了解到了很多使用的知识和技术。

在调试程序中不可避免的会遇到许多问题，一定要耐心排查才能解决。比如信号量的调试那里。

在电子技术应用领域中，嵌入式的应用愈来愈多地应用到各行各业。如：工业控制、仪器仪表、电讯技术、办公自动化和计算机外部设备、汽车与节能、商用产品、家用电器等。目前，嵌入式正朝着大容量片上存储器、多功能 i/o 接口、宽范围工作电源和低功耗方向发展。

要开发嵌入式的应用，不但要掌握嵌入式硬件和软件方面的知识，而且还要深入了解各应用系统的专业知识，只有将这两方面的知识融会贯通和有机结合，才能设计出优良的应用系统。一个好的工程设计师不仅要掌握嵌入式的工作原理，而且还要不断了解各公司最新芯片的结构和应用，在实际应用中找到最好的性能价格比。所以还要注意培养学生接受新知识的自学能力，掌握芯片发展动态。