第5章

GPIO 模块及其应用

GPIO(通用输入/输出接口)是嵌入式处理器最重要和最常用的外设,几乎每个应用系统里都会用到。本章介绍 LPC2114/2124 处理器的引脚功能选择模块和 GPIO 模块的功能、相关的寄存器以及编程操作方式。

5.1 输入/输出设备与片内外设

5.1.1 概述

对嵌入式计算机的硬件组成来说,CPU、存储器和输入/输出设备是其硬件的基本组成部分。

输入/输出设备又称 IO 设备或外围设备(简称“外设”),由于嵌入式系统对体积的要求,常在处理器芯片的内部集成多种外设,也称“片内外设”。对于嵌入式系统而言,典型的外设有:通用 IO 口(GPIO)、串行通信口、定时计数器、AD 转换器等等,这些外设对于系统与外部的信息交换非常重要。

针对不同的应用领域,对外设的配置需求也各不相同,嵌入式处理器芯片的各种型号通常都有不同的外设配置,我们可根据应用领域对外设的不同需求,选定外设配置比较适合的处理器芯片型号。

5.1.2 通用输入/输出端口 GPIO

CPU 处理的对象通常是多位二进制数据,处理的结果也是多位二进制数据,因此通用 IO 口(GPIO)是可以按位或者按多位并行输入输出数字信号的最常用最基本的外设。

当外部信号以高低电平的形式输入 CPU 时,要通过输入端口(Input Port);当 CPU 处理的结果要输出控制外部设备时,要通过输出端口(Output Port)。GPIO 端口是兼具有输入、输出功能的双向端口,可以编程控制其信号流向,因此称为“通用输入/输出端口”。

LPC2000 系列 ARM 的 GPIO(General Purpose input output,通用输入/输出)具有如下特性:

(1)可以独立控制每个 GPIO 口的方向(输入/输出模式);

(2)可以独立设置每个 GPIO 口的输出状态(高/低电平);

(3)所有 GPIO 口在复位后默认为输入状态。

LPC2114 和 LPC2124 具有两个端口:P0 和 P1。P0 和 P1 最多有 46 个 I/O 口可供使用。具体描述如表 5-1 所示。

表 5-1 LPC2114/2124 系列 GPIO 描述

端口	I/O 口端口号 LPC2114/2124
P0	P0.0～P0.25,P0.27～P0.30
P1	P1.16～P1.31, P1.0～P1.1,P1.16～P1.31
P2	无 P2.0～P2.31
P3	无 P3.0～P3.31
最多可用 I/O 口数量	46 个

由于嵌入式处理器设计时,考虑到使用的灵活性,端口大多设计成多功能端口,它们除了具有 GPIO 端口的功能以外还具有其他功能,在应用时具体作为哪种功能来使用,可以通过编程来设定,这就涉及到下面将要谈到的“引脚功能连接模块”。

5.2 LPC2114 的引脚功能连接模块

5.2.1 引脚功能连接模块的作用

LPC2000 系列微控制器的大部分管脚都具有多种功能,即管脚复用,但是同一引脚在同一时刻只能使用其中一个功能,在 LPC2000 系列处理器中,“引脚功能连接模块”相当于一个可以编程控制的多路开关,如图 5-1 所示,可以通过对它的设置来指定某个引脚选择何种功能。具体来说,就是通过配置“引脚功能连接模块”的相关寄存器控制多路开关来连接引脚与片内外设。

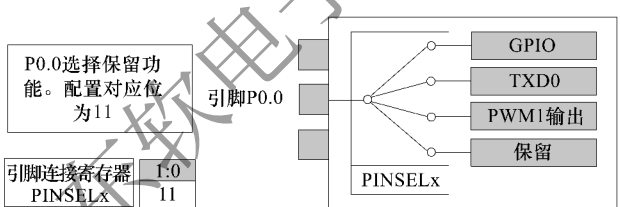

图 5-1 引脚功能连接模块的结构

5.2.2 引脚功能连接模块的相关寄存器

在使用端口时,我们是通过设置引脚功能模块的相关寄存器来完成设置任务的。LPC2000 系列 ARM 具有 3 个 PINSEL 寄存器,这 3 个 PINSEL 寄存器都是 32 位宽度的,其中 PINSEL0 和 PINSEL1 控制端口 0,PINSEL2 根据芯片的不同控制的端口数量也不同,具体描述如表 5-2 所示。

表 5-2 引脚连接模块寄存器映射

名称	描述	访问	复位值	地址
PINSEL0	引脚选择寄存器 0	读/写	0x0000 0000	0xE002 C000
PINSEL1	引脚选择寄存器 1	读/写	0x1540 0000	0xE002 C004
PINSEL2	引脚选择寄存器 2	读/写	见表 6-9	0xE002 C014

1. 引脚功能选择寄存器 0(PINSEL0)

PINSEL0 寄存器按照表 5-3 中的设定来控制引脚的功能。IOxDIR 寄存器中的方向

控制位只有在引脚选择 GPIO 功能时才有效。对于其他功能，方向是自动控制的。

表 5-3　　引脚功能选择寄存器 0

PINSEL0	引脚名称	00	01	10	11	复位值
1：0	P0.0	GPIO P0.0	TxD(UART0)	PWM1	保留	00
3：2	P0.1	GPIO P0.1	BxD(UART0)	PWM3	EINT0	00
5：4	P0.2	GPIO P0.2	SCL(1^2C)	捕获 0.0(TIMER0)	保留	00
7：6	P0.3	GPIO P0.3	SDA(1^2C)	匹配 0.0(TIMER0)	EINT1	00
9：8	P0.4	GPIO P0.4	SCK(SPI0)	捕获 0.1(TIMER0)	保留	00
11：10	P0.5	GPIO P0.5	MISO(SPI0)	匹配 0.1(TIMER0)	保留	00
13：12	P0.6	GPIO P0.6	MOSI(SPI0)	捕获 0.2(TIMER0)	保留	00
15：14	P0.7	GPIO P0.7	SSEL(SPI0)	PWM2	EINT2	00
17：16	P0.8	GPIO P0.8	TxD(UART1)	PWM4	保留	00
19：18	P0.9	GPIO P0.9	RxD(UART1)	PWM6	EINT3	00
21：20	P0.10	GPIO P0.10	RTS(UART1)	捕获 1.0(TIMER1)	保留	00
23：22	P0.11	GPIO P0.11	CTS(UART1)	匹配 1.1(TIMER1)	保留	00
25：24	P0.12	GPIO P0.12	DSR(UART1)	匹配 1.0(TIMER1)	保留	00
27：26	P0.13	GPIO P0.13	DTR(UART1)	捕获 1.1(TIMER1)	保留	00
29：28	P0.14	GPIO P0.14	CD(UART1)	EINT1	保留	00
31：30	P0.15	GPIO P0.15	RI(UART1)	EINT2	保留	00

2. 引脚功能选择寄存器 1(PINSEL1)

PINSEL1 寄存器按照表 5-4 中的设定来控制引脚的功能。IOxDIR 寄存器中的方向控制位只有在引脚选择 GPIO 功能时才有效。对于其他功能，方向是自动控制的。

表 5-4　　引脚功能选择寄存器 1

PINSEL1	引脚名称	00	01	10	11	复位值
1：0	P0.16	GPIO P0.16	EINT0	匹配 0.2(TIMER0)	保留	00
3：2	P0.17	GPIO P0.17	捕获 1.2(TIMER1)	SCK(SPI1)	匹配 1.2(TIMER1)	00
5：4	P0.18	GPIO P0.18	捕获 1.3(TIMER1)	MISO(SPI1)	匹配 1.3(TIMER1)	00
7：6	P0.19	GPIO P0.19	匹配 1.2(TIMER1)	MOSI(SPI1)	匹配 1.3(TIMER1)	00
9：8	P0.20	GPIO P0.20	匹配 1.3(TIMER1)	SSEL(SPI1)	EINT3	00
11：10	P0.21	GPIO P0.21	PWM5	保留	捕获 1.3(TIMER1)	00
13：12	P0.22	GPIO P0.22	保留	捕获 0.0(TIMER0)	匹配 0.0(TIMER0)	00
15：14	P0.23	GPIO P0.23	保留	保留	保留	00
17：16	P0.24	GPIO P0.24	保留	保留	保留	00
19：18	P0.25	GPIO P0.25	保留	保留	保留	00
21：20	P0.26	GPIO P0.26	保留			
23：22	P0.27	GPIO P0.27	AIN0(A/D 转换器)	捕获 0.1(TIMER0)	匹配 0.1(TIMER0)	01
25：24	P0.28	GPIO P0.28	AIN1(A/D 转换器)	捕获 0.2(TIMER0)	匹配 0.2(TIMER0)	01
27：26	P0.29	GPIO P0.29	AIN2(A/D 转换器)	捕获 0.3(TIMER0)	匹配 0.3(TIMER0)	01
29：28	P0.30	GPIO P0.30	AIN3(A/D 转换器)	EINT3	捕获 0.0(TIMER0)	01
31：30	P0.31	GPIO P0.31	保留			00

3. 引脚功能选择寄存器 2(PINSEL2)

PINSEL2 寄存器按照表 5-5 当中的设定来控制引脚的功能。IOxDIR 寄存器中的方向控制位只有在引脚选择 GPIO 功能时才有效。对于其他功能,方向是自动控制的。

在表 5-4 中的"复位值"这一栏表示微控制器复位时对应位的值;对于 PINSEL2 的 bit2、bit3,复位时由 P1.26、P1.20 引脚的电平决定,倘若引脚接有上拉电阻(如 10kΩ 上拉电阻),相应位的值被设置为 0;倘若引脚接有下拉电阻(如 4.7kΩ 下拉电阻),相应位的值被设置为 1。

表 5-5　　LPC2114/2124 引脚功能选择寄存器 2

PINSEL2	描述	复位值
1:0	保留	00
2	该位为 0 时,P1.31～P1.26 用作 GPIO;该位为 1 时,P1.31～P1.26 用作调试端口	$\overline{\text{P1.26}}$/$\overline{\text{RTCK}}$
3	该位为 0 时,P1.25～P1.16 用作 GPIO;该位为 1 时,P1.25～P1.16 用作跟踪端口	$\overline{\text{P1.20}}$/$\overline{\text{TRACESYNC}}$
31:4	保留	00

小结:我们从表 5-4 和表 5-5 中可以看出,PINSELx 寄存器中,每 2 位控制了一个引脚的功能选择,每个引脚的四个功能选择分别对应于这两位的 00、01、10、11 四个值,而当设置为 00 时,这个引脚总是 GPIO 功能。LPC2114/2124 引脚功能选择寄存器 0、1、2 所对应的引脚编号如表 5-6 所示。

表 5-6　　LPC2114/2124 引脚功能选择寄存器 0、1、2

寄存器	LPC2114/2124	备注
PINSEL0	P0[0:15]	每个引脚由 PINSEL0 中对应的 2 位控制
PINSEL1	P0[16:31]	每个引脚由 PINSEL1 中对应的 2 位控制
PINSEL2	P1[16:31]	每个引脚由 PINSEL2 中对应的 2 位控制

5.2.3 引脚功能连接模块的应用编程

【例 5-1】 要求将 P0.8、P0.9 设置为 TxD1、RxD1。

通过查阅 PINSE0 寄存器设置表,得到 P0.9 和 P0.8 的控制位为 PINSEL0[19:16],当该域设置为[0101](0x05)时选择 RxD1 和 TxD1。

```
PINSEL0 = 0x05 << 16;
```

为了不影响别的管脚连接设置,通常选择下面的设置方法。

```
PINSEL0 = (PINSEL0 & 0xFFF0FFFF)| (0x05 << 16);
```

"PINSEL0 & 0xFFF0FFFF"这部分是将 PINSEL0 中将被设置的 4 位(对应 P0.9 和 P0.8 两个引脚)都清零,而其他位保持不变;"| (0x05 << 16)"这部分是先将一个全零的 32 位二进制数的相应 4 位设置为需要的值,再和刚才得到的那个数相或(组合),就达到了只设定 PINSEL0 中某些位而使其他位保持不变的目的。

注意:这个编程方法很重要,在后面的程序中会经常出现,请读者重点掌握。

5.3 LPC2114 的通用输入/输出模块 GPIO

5.3.1 GPIO 端口的基本结构和工作原理

LPC2114/2124 微控制器具有两个端口:P0 和 P1,可以作为 GPIO 使用的引脚数为

46 个。

GPIO 端口的基本结构如图 5-2 所示，该图清晰地说明了端口相关的那些寄存器及其各种控制功能。

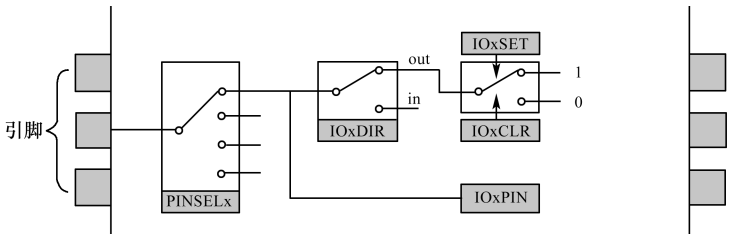

图 5-2 GPIO 结构原理图

这里有 5 个相关的重要寄存器。

最左边是信号的 IO 引脚，引脚的功能由“引脚功能选择寄存器”PINSELx 来编程指定(如 5.2 所述)，当引脚连接模块的多路开关连接到最上面的触点时(对应于 PINSELx 寄存器中相关位为 00 的情况)，引脚作为 GPIO 功能来使用。

然后，“端口方向控制寄存器”IOxDIR 寄存器负责设定端口信号的流动方向，这相当于一个 2 路开关的作用，当选择位输入(in)，则信号由引脚流入芯片后，直接进入下面的“引脚值寄存器”IOxPIN，这个寄存器的作用就是获得输入信号的电平。

如果 IOxDIR 寄存器选择了输出(out)，那么内部的将要输出的高电平和低电平分别通过“输出置位寄存器”IOxSET 和“输出清零寄存器”IOxCLR 两个寄存器来设置。

以上就是整个 GPIO 工作的基本原理。

5.3.2 GPIO 相关寄存器的详细说明

LPC2114/2124 的 P0 口和 P1 口由两组(每组 4 个)寄存器控制，如表 5-7 所示。

表 5-7 GPIO 寄存器映射——P0 和 P1

通用名称	描述	访问	复位值	PORT0 地址 & 名称	PORT1 地址 & 名称
IOxPIN	IOPIN 引脚值寄存器。不管方向和模式如何设定，引脚的当前状态都可以从该寄存器中读出。	只读	NA	0xE002 8000 IO0PIN	0xE002 8010 IO1PIN
IOxSET	IOSET 输出置位寄存器。该寄存器和 IOCLR 寄存器一起控制输出引脚的状态。写入 1 使对应引脚输出高电平；写入 0 无效。	读/位置	0x00000000	0xE002 8004 IO0SET	0xE002 8014 IO1SET
IOxDIR	IODIR 方向控制寄存器。该寄存器单独控制每个 I/O 口的方向。	读/写	0x00000000	0E002 8008 IO0DIR	0E002 8018 IO1DIR
IOxCLR	IOCLE 输出清零寄存器。该寄存器控制输出引脚的状态。写入 1 使对应引脚输出低电平清零 IOSET 寄存器中的对应位；写入 0 无效。	只清零	0x00000000	0xE002 800C IO0CLR	0xE002 801C IO1CLR

1. GPIO 引脚值寄存器(IOnPIN，n=0、1、2、3)

读该寄存器可以了解到 GPIO 引脚当前的电平状态。写该寄存器会将值保存到输出寄存器中，可用于 I/O 测试。该特性在应用中几乎毫无用处，原因是不可能对该寄存器

中单个字节执行写操作。IOPIN 寄存器描述如表 5-8 所示。

表 5-8 GPIO 引脚值寄存器

IOxPIN	描述	复位值
31：0	GPIO 引脚值。IO0PIN 的位对应于 P0.0,…,位 31 对应于 P0.31	未定义

2. GPIO 方向寄存器(IOxDIR，x=0、1、2、3)

当引脚配置为 GPIO 模式时,可使用该寄存器控制引脚的方向。比如某引脚用作输出功能,IODIR 寄存器的相应位必须设置为 1。IODIR 寄存器描述如表 5-9 所示。

表 5-9 GPIO 方向寄存器

IODIR	描述	复位值
31：0	方向控制位(0=输入,1=输出)。IO0DIR 的位 0 控制 P0.0,...,位 31 对应于 P0.31	0

3. GPIO 输出置位寄存器(IOxSET，x=0、1、2、3)

当引脚配置为 GPIO 输出模式时,可使用该寄存器控制引脚输出高电平。写入 1 使对应引脚输出高电平;写入 0 无效。如果一个引脚被配置为输入或第二功能,写 IOSET 无效。

读 IOSET 寄存器返回 GPIO 输出寄存器的值。该值由前一次对 IOSET 和 IOCLR(或前面提到的 IOPIN)的写操作决定。该值不反映任何外部环境对引脚的影响。

IOSET 寄存器描述如表 5-10 所示。

表 5-10 GPIO 输出置位寄存器

IOSET	描述	复位值
31：0	输出置位。IO0SET 的位 0 对应于 P0.0,...,位 31 对应于 P0.31	0

4. GPIO 输出清零寄存器(IOxCLR，x=0、1、2、3)

当引脚配置位 GPIO 输出模式时,可使用该寄存器从引脚输出低电平。写入 1 使对应引脚输出低电平,并清零 IOSET 寄存器中相对应的位;写入 0 无效。如果一个引脚被配置为输入或第二功能,写 IOCLR 无效。IOCLR 寄存器描述如表 5-11 所示。

表 5-11 GPIO 输出清零寄存器

IOCLR	描述	复位值
31：0	输出清零。IO0CLR 的位对应于 P0.0,...,位 31 对应于 P0.31	0

5.3.3 GPIO 输出控制编程

【例 5-2】 设置 P0.0 输出高电平。

如图 5-3 所示。

【例 5-3】 GPIO 读/写操作:程序读取 P0.7～P0.4 引脚值,然后从 P0.3～P0.0 输出。

```
bak = IO0PIN; //读取引脚上的值 ,保存于变量 bak 中
IO0CLR = 0x0000000F; //将 P0.0～P0.3 输出 0
IO0SET = (bak&0x000000F0)>> 4; //设置 P0.0～P0.3 输出(为 1 的位输出 1)
```

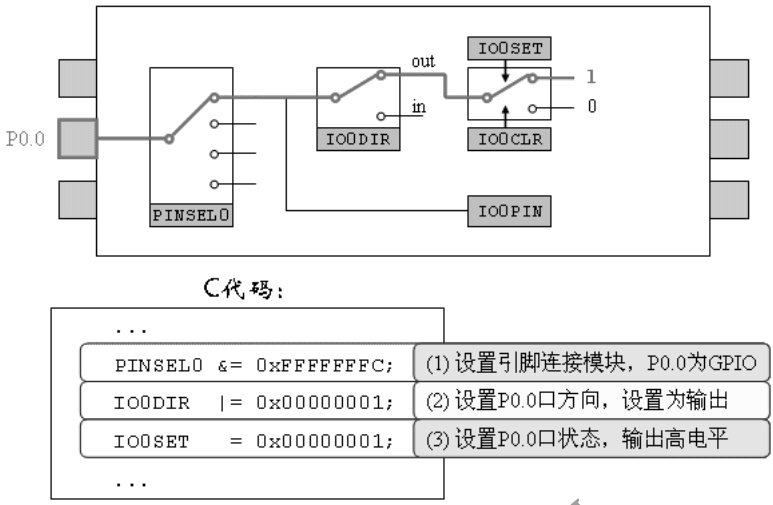

图 5-3 设置 P0.0 输出高电平

【例 5-4】 取反 P0.0 的输出。

分析：程序先从 IO0SET 读取当前输出寄存器的值，而不是去读引脚上的电平值（即读 IO0PIN），然后判断 P0.0 的输出是高电平还是低电平，再控制输出相反。

```
if((IO0SET&0x00000001) = = 0)    IO0SET = 0x00000001;
else  IO0CLR = 0x00000001;
```

【例 5-5】 将 8 位无符号整形变量 Data 的值输出到 P0.0～P0.7。

在需要将多位数据同时输出到某几个 IO 口线时，通常使用 IOxSET 和 IOxCLR 来实现，在某些情况下也可以使用 IOxPIN 寄存器实现。后者可以在多个 IO 口上直接输出 0 和 1 电平。

(1)使用 IOxSET 和 IOxCLR 实现。

```
#define  DataBus  0xFF
PINSEL0 &= 0xFFFF0000;      //(1)设置引脚连接模块，P0.0～7 为 GPIO
IO0DIR  |= DataBus;         //(2)设置 P0.0 口方向，设置为输出
IO0CLR   = DataBus;         //(3)清零 8 位 IO 口的输出状态
IO0SET   = Data;            //(4)Data 变量中为 1 的位将输出高电平
...
```

(2)使用 IOxPIN 实现。

```
#define  DataBus  0xFF
PINSEL0 &= 0xFFFF0000;      //(1)设置引脚连接模块，P0.0 为 GPIO
IO0DIR  |= DataBus;         //(2)设置 P0.0 口方向，设置为输出
IO0PIN   = (IO0SET & 0xFFFFFF00)| Data;  //(3)写 IO0PIN，输出数据
```

使用 GPIO 注意要点：

(1)引脚设置为输出方式时，输出状态由 IOxSET 和 IOxCLR 中最后操作的寄存器决定。

(2)大部分 GPIO 输出为推挽方式（个别引脚为开漏输出），正常拉出/灌入电流均为 4mA（短时间极限值 40mA）。

(3)复位后默认所有 GPIO 为输入模式。

5.3.4 基础实训(1) 蜂鸣器输出控制

1. 系统功能

用 LPC2114 的一个引脚输出高低电平,控制蜂鸣器间歇鸣叫。

2. 效果演示

播放“实训演示视频”文件夹中“\实训 5_1.avi”。

3. 硬件原理

电路设计如图 5-4(a)所示。

(1)用 LPC2114 的一个引脚 P0.7 控制晶体管 Q1 的基极,周期性输出高低电平;

(2)PNP 晶体管 Q1 放大电流,驱动控制蜂鸣器;

(3)P0.7 端口输出高低电平控制 Q1 的基极;

(4)当 P0.7 控制电平输出 0 时,Q1 导通,蜂鸣器蜂鸣;

(5)当 P0.7 控制电平输出 1 时,Q1 截止,蜂鸣器停止蜂鸣。

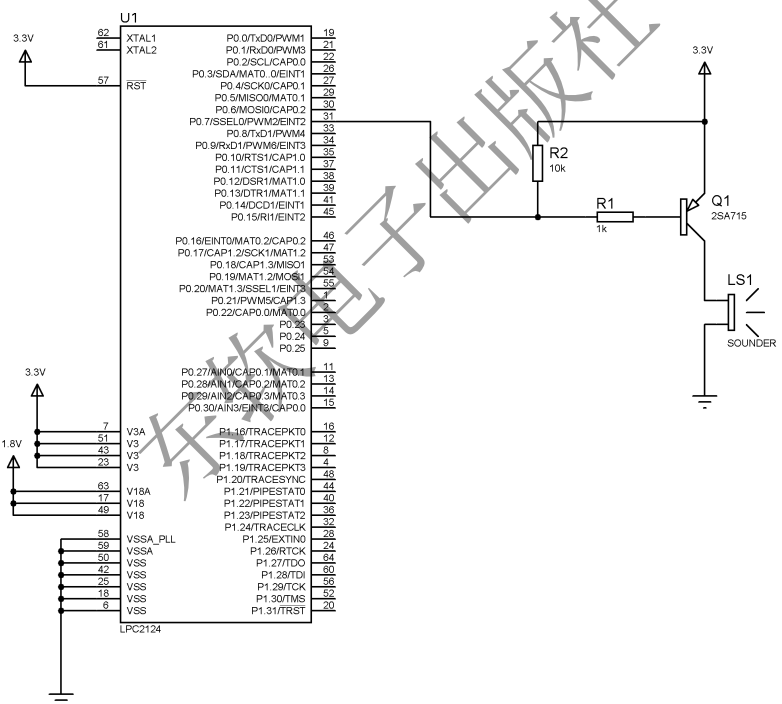

图 5-4(a) 电路原理图

4. 程序设计

要使得蜂鸣器周期性鸣叫,须给它输出一个周期性高低电平信号,这个信号实际上可以是一个方波,也就是高电平持续若干时间、低电平再持续若干时间,如此循环往复。

为了获得一个固定的延时,我们可以采用软件延时子程序,让指令的执行来产生延时效果。这种软件延时的方法一般多用在对定时精度要求不高的场合。

然后在主程序中,使用一个无限循环作为程序主体(while(1){… }语句来实现),在每次循环过程中需完成以下任务:

(1)设置 P0.7 端口输出高电平。

(2)调用延时子程序。

(3)设置 P0.7 端口输出低电平。

(4)调用延时子程序。

这样就可以实现系统要求的功能了。

当然,在主程序开始无限循环体之前,必须要对 P0.7 端口作出正确的初始设置(简称“初始化”),要通过端口功能选择模块寄存器去设置 P0.7 为 GPIO 功能,还要把它的 IO 方向设置为“输出”。

程序流程图如图 5-4(b)所示。

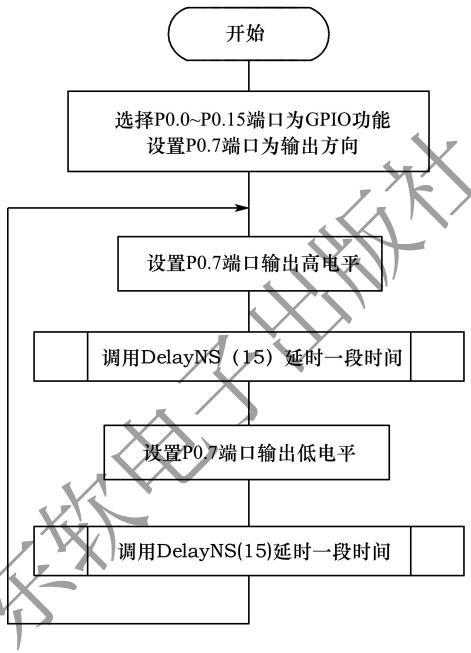

图 5-4(b) 主程序流程图

5. 代码详析

按照第 4 章所讲述的步骤,利用 ARM 开发工具 RealView MDK 建立 GPIO1.Uv2 工程,并添加代码,编译并链接工程,生成 GPIO1.HEX 文件。核心代码如下:

```
#include "LPC21xx.h"   //引入头文件,其中包含了对 LPC21xx 芯片全部寄存器地址的定义
#define BEEPCON 0x00000080      //定义该常数是为了方便控制寄存器的编程
void DelayNS(unsigned int dly )  //延时子程序定义
                                 //入口参数 dly——延时参数,值越大,延时越久。
{
  unsigned int i;
  /*   下面用 2 层嵌套循环构成延时子程序的主体部分 ,参数 dly 决定了延时的长度 */
  for (;dly>0 ;dly--)            //第 1 层循环
  for ( i=0; i<5000; i++);       //第 2 层循环
}
int main(void)                   //主程序
```

```
{
  PINSEL0 = 0x00000000;          // 设置引脚连接 GPIO
  IO0DIR = BEEPCON;              //设置 I/O 为输出
  while (1)                      //该循环为无限循环,循环体就是程序的主体
  {
    IO0SET = BEEPCON;            //BEEPCON = 1 输出高电平使蜂鸣器鸣叫
    DelayNS(15);                 //延时
    IO0CLR = BEEPCON;            //BEEPCON = 0 输出低电平使蜂鸣器关闭
    DelayNS(15);                 //延时
  }
}
```

6. 电路仿真

(1)在 Proteus 软件中,连接电路原理如图 5-4(a)所示。

(2)右击 LPC2124,在弹出的 Edit Component 对话框中,为 LPC2124 芯片设置 Program File 文件 GPIO1.HEX 路径。

(3)启动仿真,蜂鸣器产生蜂鸣。

思考:把本实训的输出端改为 P0.18,且 LED 熄灭的时间是发光的时间的 1/2,重新修改电路和编写程序,仿真看效果。

5.3.5 GPIO 输入检测编程

【例 5-6】 读取 P0.0 引脚状态,如图 5-5 所示。

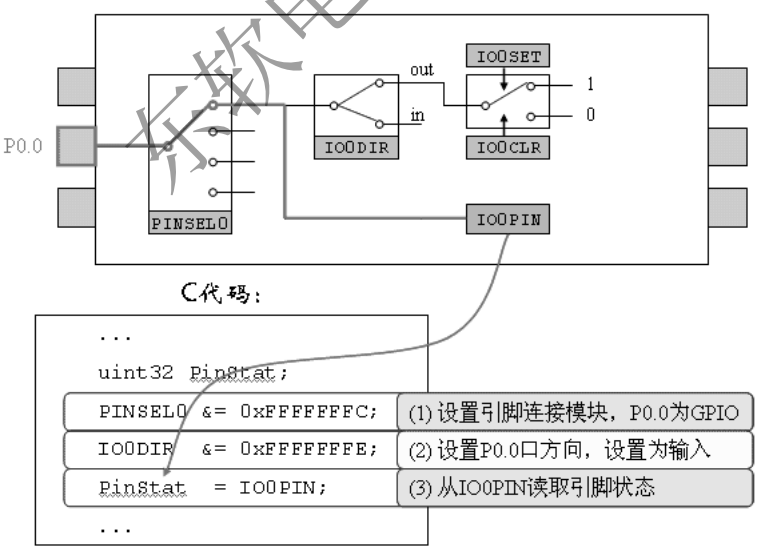

图 5-5 读取 P0.0 引脚的输入

5.3.6 基础实训(2) 单个 LED 显示单个按键的状态

1. 系统功能

若按键被按下,则 LED1 点亮;否则 LED1 熄灭。

2. 效果演示

播放“实训演示视频”文件夹中“\实训 5_2.avi”。

3. 硬件原理

电路设计如图 5-6(a)所示。

(1)P0.14 连接按键输入电路。

(2)GPIO 是双向的 I/O 口,内部无上拉电阻,所以当用作按键输入时,需要加上拉电阻,如图 5-6(a)中所示的电阻 R2 为 P0.14 引脚的上拉电阻。使用 P0.14 作为按键输入,每一次有效按键即对 P0.7 引脚连接的 LED1 点亮;断开按键则 LED1 熄灭。

(3)按键被按下时,P0.14 输入低电平;否则输入高电平。

(4)P0.7 端口输出高低电平控制 LED1 的状态。

(5)输出“1”则 LED1 熄灭;输出“0”则 LED1 点亮。

4. 程序设计

要用 LED 表示按键的状态(按下时亮,松开时灭),须周期性地检测按键的状态(通过输入引脚的电平来确定),若得到高电平,表示按键松开,则给 LED 控制端输出信号使得 LED 熄灭;若得到低电平,表示按键闭合,给 LED 输出控制信号使其发光。如此循环往复。

然后在主程序中,使用一个无限循环作为程序主体,在每次循环过程中需完成以下任务:

(1)读取 P0.14 的 IO0PIN 寄存器,得到端口电平(按键状态)。

(2)若 P0.14 检测到高电平,P0.7 输出高电平使得 LED 熄灭;否则 P0.7 输出低电平使得 LED 点亮。

(3)调用延时子程序。

当然,在主程序开始无限循环体之前,必须要对 P0.7 和 P0.14 端口作出正确的初始设置(简称“初始化”),要通过端口功能选择模块寄存器去设置 P0.7 和 P0.14 为 GPIO 功能,还要把 P0.7 的 IO 方向设置为“输出”,P0.14 的方向设置为输入。

程序流程如图 5-6(b)所示。

5. 代码详析

利用 ARM 开发工具 RealView MDK 建立 GPIO2.Uv2 工程,并添加代码,编译并链接工程,生成 GPIO2.HEX 文件。核心代码如下:

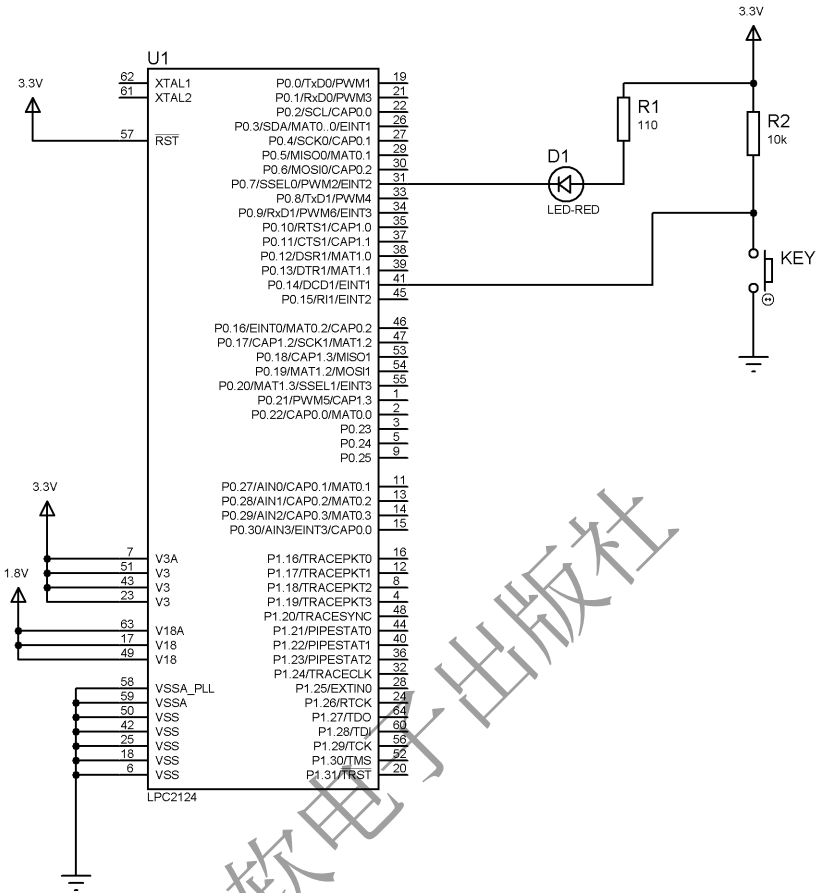

图 5-6(a) 电路原理图

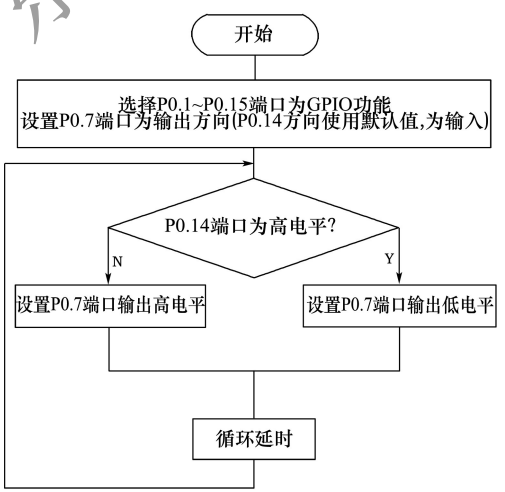

图 5-6(b) 主程序流程图

```
#include "LPC21XX.h"
#define LED1 0x00000080
#define PIN_P014 0x00004000

int main(void)
{
  unsigned int i;
  PINSEL0 = 0x00000000;//端口选择为 GPIO 功能
  IO0DIR = LED1;//设置 P0.7 为输出(P0.14 方向使用默认值,为输入)
  while (1)  //主循环体
  {
    if ((IO0PIN & PIN_P014)! = 0)  IO0SET = LED1; //若 P0.14 检测到高电平,P0.7 输出高电平使得 LED 熄灭
    else  IO0CLR =  LED1; //若 P0.14 检测到低电平,P0.7 输出低电平使得 LED 点亮
    for (i = 0 ;i<1000; i + + ); //延迟一段时间,再进入下一次循环
  }
}
```

6. 电路仿真

(1)在 Proteus 软件中,连接电路原理图如图 5-6(a)所示。

(2)右击 LPC2124,在弹出的 Edit Component 对话框中,为 LPC2124 芯片设置 Program File 文件 GPIO2. HEX 路径。

(3)启动仿真,按下 KEY1 键则 LED1 点亮,否则 LED1 熄灭。

思考:把本实训的输出端改为 P0.15,输入端改为 P0.19,重新修改电路和编写程序,仿真看效果。

5.4 本章小结

本章介绍了 LPC2114 芯片的引脚功能连接模块和 GPIO 模块,这是处理器输入输出的最基本的方式,我们要了解了两者的工作原理和组成结构;更为重要的是必须熟悉其相关的控制寄存器和掌握其编程方法。

5.5 强化练习

1. 引脚连接模块有什么作用,相关寄存器有哪些?

2. GPIO 相关的有哪些控制寄存器,各有什么作用?