第 3 章　GPIO 的用途实验

GPIO 是 General Purpose Input Output 的简称，中文名称为“通用输入/输出接口”。它是嵌入式系统硬件平台的重要组成部分，许多外部设备的控制通常是通过 GPIO 端口来进行的。在本章实验中，我们将具体学会使用 S3C2440 芯片的 GPIO 端口，主要是这些 GPIO 端口的功能初始化。

3.1　实验目标及要求

本实验通过 1 小段程序，来验证 S3C2440 的 GPIO 端口功能及特性，从而熟悉并掌握 S3C2440 的 GPIO 端口使用。具体的实验目的及要求是：

1. 熟悉并掌握 S3C2440 的 GPIO 端口功能，了解其引脚的多功能特性。
2. 熟悉并掌握 GPIO 端口的控制寄存器格式。
3. 熟悉并掌握 GPIO 端口控制寄存器的初始化编程。
4. 熟悉并掌握 GPIO 端口的数据寄存器及其编程。

3.2　实 验 步 骤

实验步骤：

首先检查实验箱与宿主机之间是否用串口线(即 RS-232 接口线)连接好。若连接好，则启动宿主机，并在宿主机上运行超级终端，配置好其参数。然后，利用 ADS 1.2 工具建立本实验程序的工程项目，并配置好工程项目的参数(具体操作参见 1.3 节和 1.4 节。由于这些步骤在以后的实验中也类似，后续章节中不再介绍)。

3.3　实验关键点

S3C2440 芯片共有 130 个输入/输出引脚，分属于 9 个 GPIO 端口。这 9 个 GPIO 端口均为多功能复用端口，端口功能可以编程设置。9 个 GPIO 端口是：

- 端口 A (GPA)——有 25 条输出引脚的端口,但最高 2 位对应的引脚未用(即保留)。
- 端口 B (GPB)——有 11 条输入/输出引脚的端口。
- 端口 C (GPC)——有 16 条输入/输出引脚的端口。
- 端口 D (GPD)——有 16 条输入/输出引脚的端口。
- 端口 E (GPE)——有 16 条输入/输出引脚的端口。
- 端口 F (GPF)——有 8 条输入/输出引脚的端口。
- 端口 G (GPG)——有 16 条输入/输出引脚的端口。
- 端口 H (GPH)——有 11 条输入/输出引脚的端口。
- 端口 J (GPJ)——有 13 条输入/输出引脚的端口。

上述 9 个 GPIO 端口根据系统配置和设计的不同需求,设计者可以选择这些 GPIO 端口的功能。若选定某个 GPIO 端口的功能,设计者应在使用该引脚功能之前编程设置对应的控制寄存器,从而确定所需 GPIO 端口的功能。如果某个 GPIO 引脚不用于特定专用功能的话,那么该引脚就可以设置用于普通的输入/输出功能。

每个 GPIO 端口中,均有 3 个寄存器,分别是 GPxCON、GPxDAT、GPxUP(注:x 代表 GPIO 端口的序号:A、B、……、J),它们的作用分别是端口控制寄存器、端口数据寄存器、端口上拉电阻寄存器。

端口控制寄存器 GPxCON 的作用是用来设定 GPIO 端口 x 的引脚功能。端口数据寄存器 GPxDAT 的作用是进行读/写端口引脚的信息,例如,若端口 E 的 GPE0 配置为输出时,向端口 E 的数据寄存器最低位写入 1 时,GPE0 引脚输出高电平,否则,写入 0 时,GPE0 引脚输出低电平。端口上拉电阻寄存器 GPxUP 确定端口的引脚是否使用内部上拉电阻,注意:端口 A 没有端口上拉电阻寄存器。

3.3.1 端口引脚功能介绍

上面提到,每个 GPIO 端口的引脚功能均是多功能的,具体的引脚功能请参考主教材《嵌入式系统原理及接口技术》的第 6 章。在本章实验示例中,由于用到 GPIO 端口 C 的若干引脚,因此,在此具体介绍 GPIO 端口 C 的引脚功能。

端口 C 的 I/O 引脚共有 16 条,每条引脚的功能如表 3-1 所示。

表 3-1 端口 C 的引脚功能

引 脚 标 号	功能 1	功能 2	功能 3
GPC15	普通输入/输出	VD7	—
GPC14	普通输入/输出	VD6	—
GPC13	普通输入/输出	VD5	—
GPC12	普通输入/输出	VD4	—
GPC11	普通输入/输出	VD3	—
GPC10	普通输入/输出	VD2	—
GPC9	普通输入/输出	VD1	—

续表

引 脚 标 号	功能 1	功能 2	功能 3
GPC8	普通输入/输出	VD0	—
GPC7	普通输入/输出	LCDVF2	—
GPC6	普通输入/输出	LCDVF1	—
GPC5	普通输入/输出	LCDVF0	—
GPC4	普通输入/输出	VM	—
GPC3	普通输入/输出	VFRAME	—
GPC2	普通输入/输出	VLINE	—
GPC1	普通输入/输出	VCLK	—
GPC0	普通输入/输出	LEND	—

端口 C 的引脚有 2 种功能，第 1 种功能是作为普通的输入/输出信号线，第 2 种功能主要是用作彩色 LCD 显示器接口的控制信号线以及数据信号线。

3.3.2　端口的控制寄存器介绍

端口控制寄存器的作用是用来设置端口引脚功能，如 GPCCON 是端口 C 的控制寄存器，用来设置端口 C 中每个 GPIO 引脚的功能。它是可读/写的，其地址为：0x56000020，复位后的初值为 0x0。GPCCON 寄存器的具体格式如表 3-2 所示。

表 3-2　GPCCON 寄存器的格式

符　　号	位	描　　述	初 始 状 态
GPC15	[31:30]	00＝输入　01＝输出 10＝VD7　11＝保留	00
GPC14	[29:28]	00＝输入　01＝输出 10＝VD6　11＝保留	00
GPC13	[27:26]	00＝输入　01＝输出 10＝VD5　11＝保留	00
GPC12	[25:24]	00＝输入　01＝输出 10＝VD4　11＝保留	00
GPC11	[23:22]	00＝输入　01＝输出 10＝VD3　11＝保留	00
GPC10	[21:20]	00＝输入　01＝输出 10＝VD2　11＝保留	00
GPC9	[19:18]	00＝输入　01＝输出 10＝VD1　11＝保留	00
GPC8	[17:16]	00＝输入　01＝输出 10＝VD0　11＝保留	00
GPC7	[15:14]	00＝输入　01＝输出 10＝LCDVF2　11＝保留	00

续表

符　号	位	描　述	初始状态
GPC6	[13:12]	00＝输入　01＝输出 10＝LCDVF1　11＝保留	00
GPC5	[11:10]	00＝输入　01＝输出 10＝LCDVF0　11＝保留	00
GPC4	[9:8]	00＝输入　01＝输出 10＝VM　11＝保留	00
GPC3	[7:6]	00＝输入　01＝输出 10＝VFRAME　11＝保留	00
GPC2	[5:4]	00＝输入　01＝输出 10＝VLINE　11＝保留	00
GPC1	[3:2]	00＝输入　01＝输出 10＝VCLK　11＝保留	00
GPC0	[1:0]	00＝输入　01＝输出 10＝LEND　11＝保留	00

GPCDAT 是端口 C 数据寄存器，它是可读/写的。当端口 C 为输入功能时，从该寄存器可读取端口 C 连接的外部数据信息；当端口 C 为输出功能时，向该寄存器写入的数据将通过端口 C 输出。其地址为：0x56000024，复位后的初值不确定。GPCDAT 寄存器的具体格式如表 3-3 所示。

表 3-3　GPCDAT 寄存器的格式

符　号	位	描　述	初始状态
GPC15:0	[15:0]	存放端口 C 的数据	—

GPCUP 是端口 C 上拉设置寄存器，它是可读/写的。用来确定端口 C 的 GPIO 引脚是否内部接上拉电阻。其地址为：0x56000028，复位后的初值为 0x0。GPCUP 寄存器的具体格式如表 3-4 所示。

表 3-4　GPCUP 寄存器的格式

符　号	位	描　述	初始状态
GPC15:0	[15:0]	1＝对应的 GPIO 引脚上拉电阻不使能 0＝对应的 GPIO 引脚上拉电阻使能	0x0000

3.3.3　实验程序源码解释

本段源程序的功能是利用 GPIO 端口 C 的 GPC5、GPC6、GPC7 引脚来控制 3 个 LED 显示器显示。显示器的控制程序采用 C 语言编写，在 main()函数中实现。目标系统启动时，先执行启动引导程序，然后引导应用程序的 main()函数进行执行。目标系统

的硬件电路上，用 GPC5、GPC6、GPC7 引脚控制 3 个 LED 灯的阴极，3 个 LED 灯的阳极通过电阻直接连电源。

该示例利用 ADS 1.2 进行开发时，建立的工程项目主界面如图 3-1 所示。

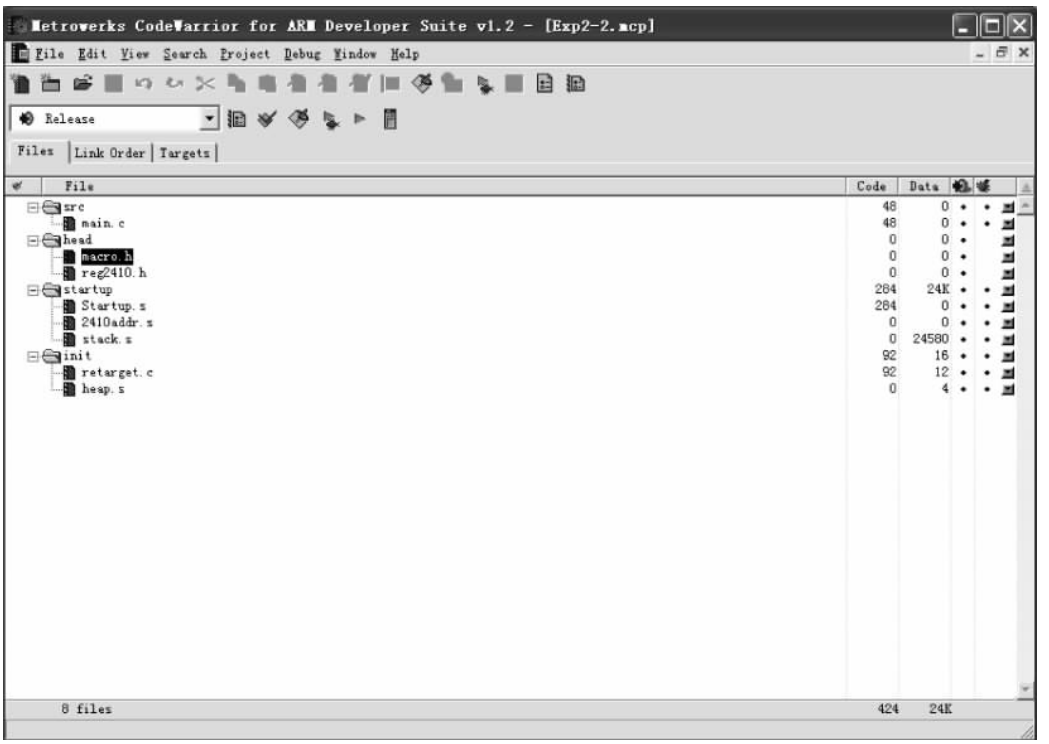

图 3-1　GPIO 用途实验的工程项目主界面

从图 3-1 中可以看到，该工程项目中包含了 2410addr. s、reg2410. h、Startup. s、main. c 等源程序文件。2410addr. s、reg2410. h 文件中定义了 S3C2440 芯片内部寄存器对应的变量，具体代码请参见附录 A。Startup. s 文件中的代码在第 2 章的源代码二中已解释过，此处不再赘述。下面介绍工程项目中的其他文件。

main. c 文件中的 main()函数代码设计如下：

```
#include <string.h>
#include <stdio.h>
#include "inc/macro.h"
#include "inc/reg2410.h"
void delay(int time1,int time2);
int main(void)
{
    //初始化端口 C 的 GPC5、GPC6、GPC7 的功能为输出
    rGPCCON = (rGPCCON | 0x00005400) & 0xffff57ff;
    //下面是控制 3 个 LED 灯闪烁的语句
     while(1)
```

```
        {
            //下面语句使得 GPC5 输出 0,GPC6、GPC7 输出 1
            rGPCDAT = (rGPCDAT & 0xffdf) | 0x00c0;
            delay(10000,10000);              //延时
            //下面语句使得 GPC5 输出 1,GPC6、GPC7 输出 0
            rGPCDAT = (rGPCDAT & 0xff3f) | 0x0030;
            delay(10000,10000);              //延时
        }
    return 0;
}
//延时用的函数
void delay(int time1,int time2)
{
    int i,j;
    for(i=0;i<time1;i++)
    {
        for(j=0;j<time2;j++);
    }
}
```

stack.s 文件中的代码定义一些堆栈区,分别对应了用户模式、管理模式、未定义模式、IRQ 模式、中止模式、FIQ 模式等,其程序代码如下:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;定义各工作模式下的堆栈区
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        AREA    Stacks, DATA, NOINIT

        EXPORT UserStack
        EXPORT SVCStack
        EXPORT UndefStack
        EXPORT IRQStack
        EXPORT AbortStack
        EXPORT FIQStack

        SPACE   4096
UserStack   SPACE   4096
SVCStack    SPACE   4096
UndefStack  SPACE   4096
AbortStack  SPACE   4096
IRQStack    SPACE   4096
FIQStack    SPACE   4

        END
```