哈尔滨工业大学（深圳）

工程训练(电子工艺实习)
电子产品软件设计与应用指导书

编者：杨静、刘能锋

2018 年

实验与创新实践教育中心

目 录

实验一 嵌入式系统软件设计与开发概述

1.1 实验目的

（1） 了解嵌入式系统软件设计与开发流程，熟悉 CCS 的基本使用方法；
（2） 掌握 MSP430 系列单片机程序开发的基本步骤；
（3） 掌握对 IO 口的查询操作和 IO 基本操作的流程；

1.2 实验仪器

（1） MSP430F5529LP+MSP430F5529 POCKET KIT 开发板一套；
（2） PC 机操作系统 Windows 7，CCSv7.3 集成开发环境。

1.3 实验准备

1.3.1 嵌入式系统软件设计与开发

1.3.1.1 嵌入式系统介绍

微控制器（MCU）、微处理器（MPU）、数字信号处理器（DSP）是目前最常用的 3 种可编程处理器，他们根据确定的程序执行相应的置零，其架构特性均来源于 1971 年开发的第一款微处理器。微控制器具有有限的输入和输出，能够嵌入到完整系统中。

嵌入式系统的功能软件集成于硬件系统之中，系统的应用软件与硬件一体化。在嵌入式系统的硬件设备中，嵌入处理器是整个系统的核心部件，其性能的好坏直接决定整个系统的运行效果。

嵌入式系统开发面向具体应用，不同领域的应用市场需要不同款式和性能指标的处理器来开发，于是在嵌入式处理器市场中，中低端的 4 位、8 位和 l6 位处理器依然存在，高性能的 32 位处理器也有很多产品。随着超大规模集成电路技术和微电子技术发展，包含嵌入式处理器以及部分外围电路的微控制器产品也进入市场，这些产品的上市，不仅丰富了嵌入式处理器产品，而且也更加方便了工程技术人员进行嵌入式系统的技术开发和扩大嵌入式产品的应用领域。

我们课程所学的单片机属于微控制器（MCU），具有体积小、功能强、价格低等优点，是微机应用产品的最佳选择。单片机的出现也改变了传统的电路设计方法，过去经常采用模拟电路、脉冲电路、组合逻辑实现的电路系统，现在相当一部分可用各种单片机取代。传统的逻辑设计方法正在演变成软件和硬件相结合的设计方法，许多电路设计问题都转化为程序设计问题。

在实际工业应用中，硬件设计基本确定后，软件设计完成电路的各种功能，软件和硬件结合达到最终的电子线路实际应用功能，产品就可以急性集成测试出厂了。

软件开发流程如下，

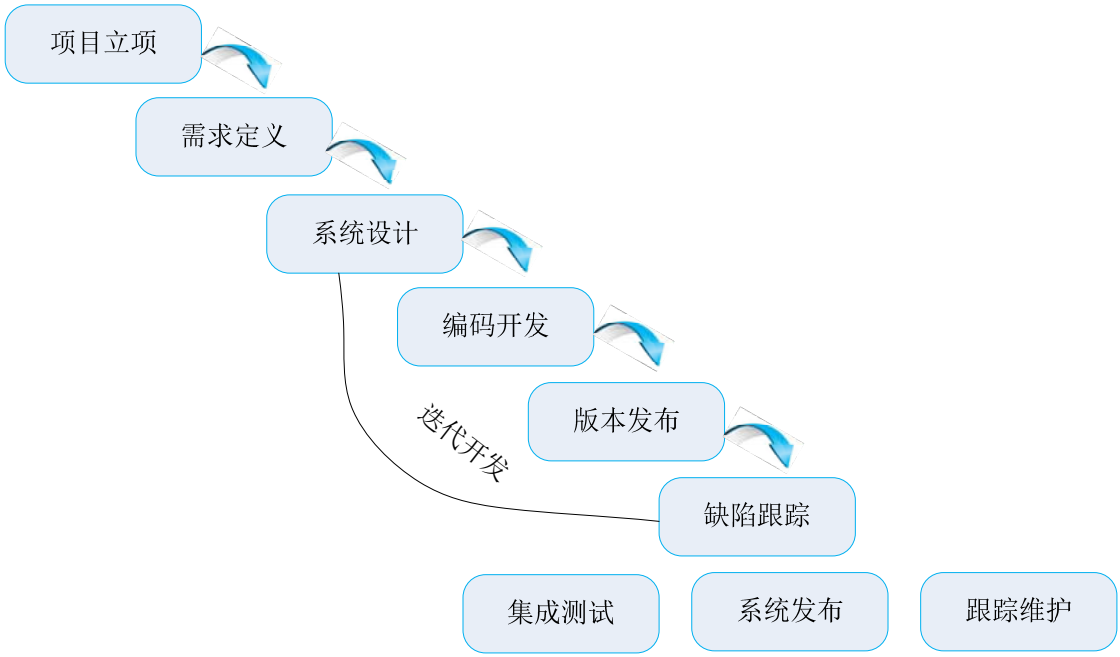


图 1-1 软件开发流程图

输出存档文件有《软件需求分析书》、《软件总体概要设计说明》、《软件**模块或功能概要设计》、《软件**模块或功能详细设计》、《软件**模块或功能测试说明文档》、《软件总体设计集成测试发布文档》、《软件总体设计**版本发布》、《软件总体设计**版本发布》等。

后期在产品使用和维护阶段，还要不断更新维护两种文档，《**软件缺陷分析记录》和《**软件版本升级记录》

1.3.1.2 MSP430F5529 口袋板实验开发系统

MSP430F5529 口袋板（MSP430F5529 Pocket Kit，以下简称 F5529PK）实验开发系统配合 TI MSP430F5529 LaunchPad 开发板（以下简称 LP 板）使用，板子上集成了电子纸屏幕（电子墨水屏）、模拟滤波器、用户 LED 与按键、蜂鸣器、音频功放、温度传感器、DAC、SD 卡座、耳机插座、信号输入/输出接口等模块，大大扩展了 MSP430F5529LaunchPad 开发板的实验内容与应用领域。

通过口袋板上的众多模块，不但可以完成 MCU、数字电路、模拟电路等基础实验项目，而且可以完成音频播放、录音回放、温度测量等趣味性综合实验。口袋板非常适合实验室之外（宿舍、家中）进行一些科技创新、电子设计大赛活动时使用。

（1）MSP430F5529 LP 微控制器介绍

- 带 USB 的 16 位单片机；
- eZ-FET 板载仿真器，包含 UART 接口；
- 带有 USB Hub ，可同时用 USB 接口仿真和开发 USB 应用程序；
- USB 供电，内置 3.3V 电压转换器；
- 配备 40 引脚的 BoosterPack，与 40 引脚的 LaunchPad BoosterPack ™ 开发工具标准兼容；
- Launch Pad 使用 USB 接口提供的 5V 电源供电；

- 仿真器板包含 DC-DC 变换器提供 3.3V 电源；
- 目标板的 5V 或 3.3V 电源均可由外部提供。

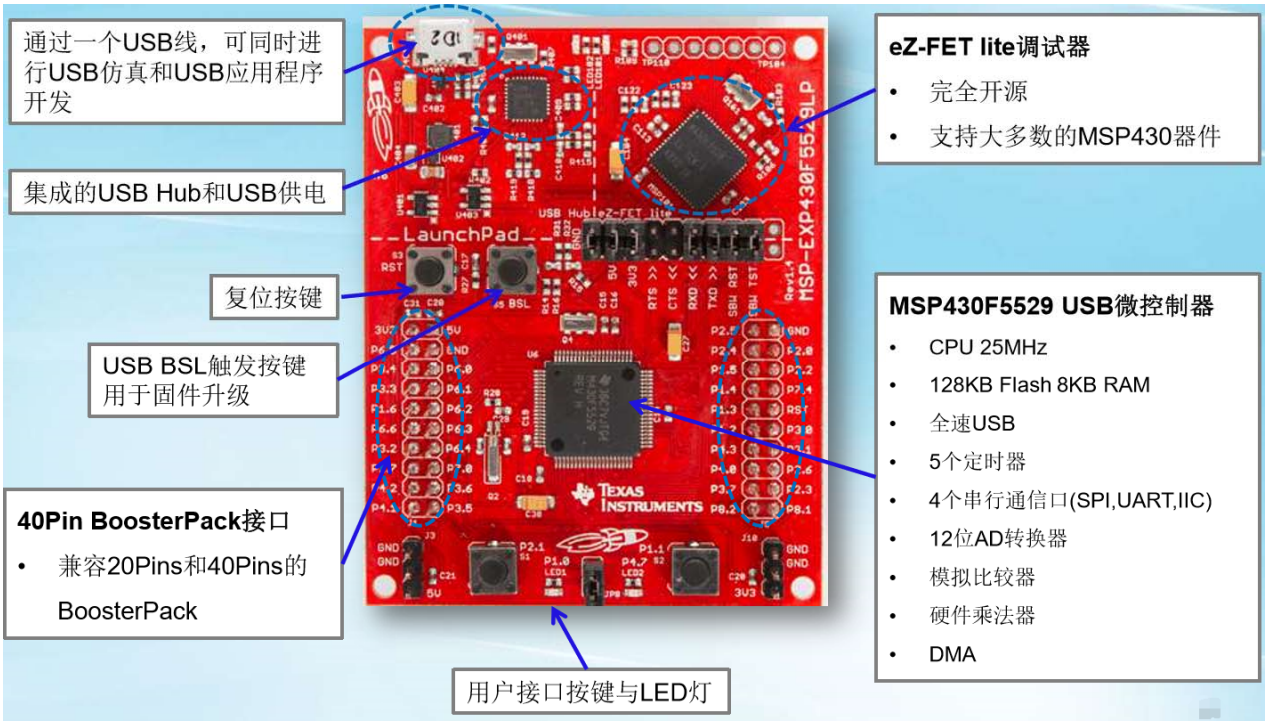


图 1-2 MSP430F5529 LP 硬件资源功能图

（3） MSP430F5529LP 引脚图及结构框图

MSP430F5529 的引脚图如图 1-3 所示，结构框图如图 1-4 所示。

（4） MSP430F5529 口袋板硬件资源

MSP430F5529 口袋板上主要集成了如下模块。

- 电子纸显示屏（电子墨水屏）：I2C 接口，分辨率 250×122；
- 高功率 LED 模块；
- 电流检测模块；
- 有源滤波器模块：对用户开放一个二阶低通滤波器和一个二阶高通滤波器，其中低通滤波器的直流偏置电压可在）0V 与 1/2VCC 间进行切换；
- 无源蜂鸣器/扬声器模块；
- 音频功率放大器模块；
- 温度传感器模块；
- 串行 DAC 模块；
- 用户 LED 模块：6 个 LED 灯可供用户编程使用；
- 用户按键模块：4 个机械按键、2 个电容触摸按键可供用户编程使用；
- 无线扩展模块接口；
- 耳机插座：可以连接耳机，也可以做立体声 Line Out 接口使用；
- MicroSD 卡插座；
- BoosterPack 插针：用于连接 MSP430F5529LaunchPad 板子的上的 BoosterPack 插座（背对背连接）；

• 信号接口：可以通过杜邦线将信号引入/引出。

口袋板硬件资源图请看图 1-5（分别为口袋板正面与背面）。

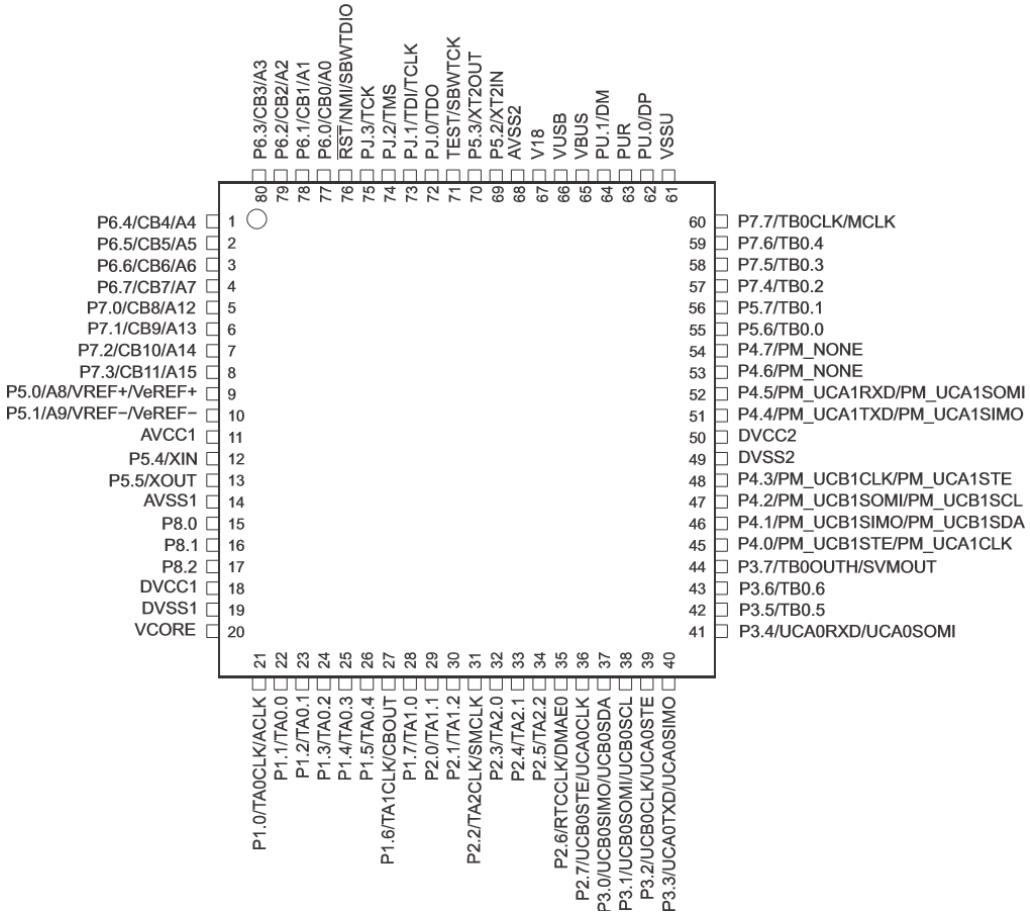


图 1-3 MSP430F5529LP 芯片引脚定义

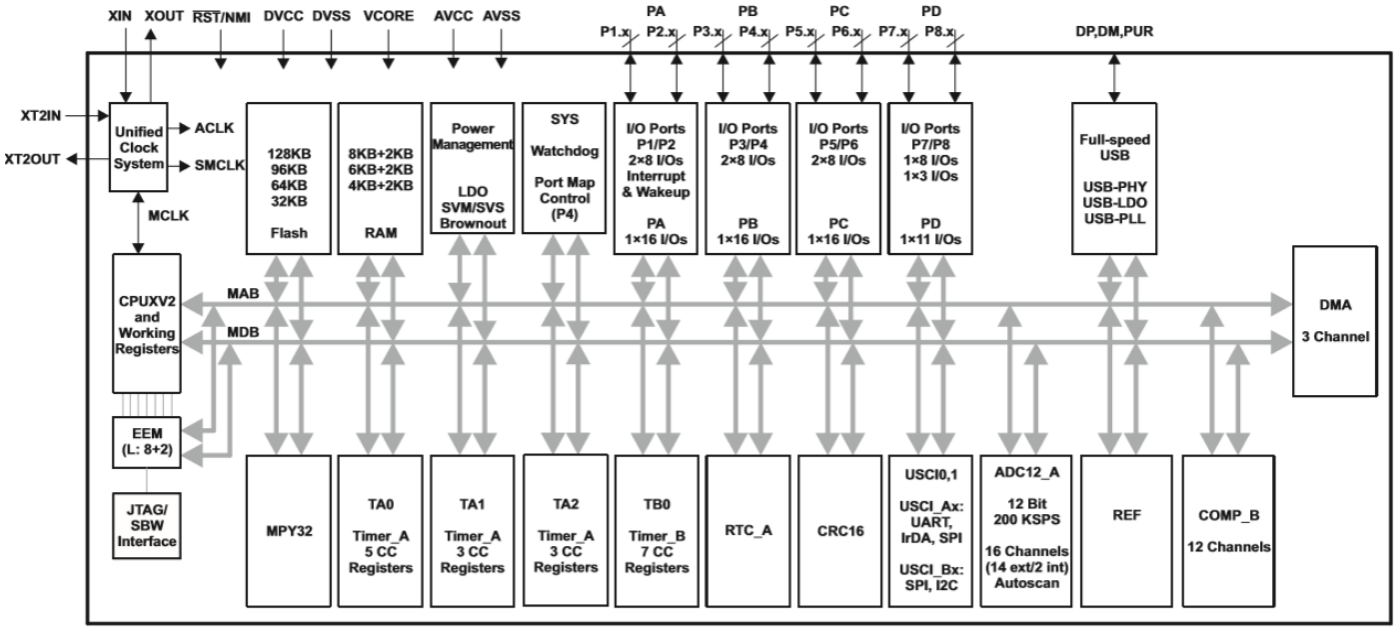


图 1-4 MSP430F5529LP 功能框图

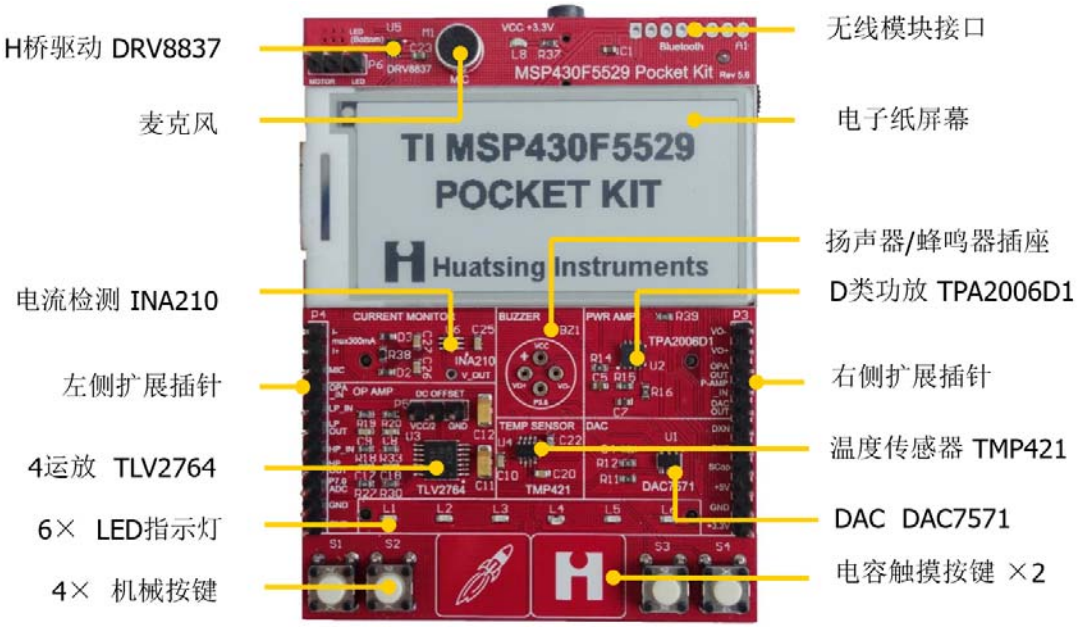


图 1-5（a） 口袋板硬件资源功能图

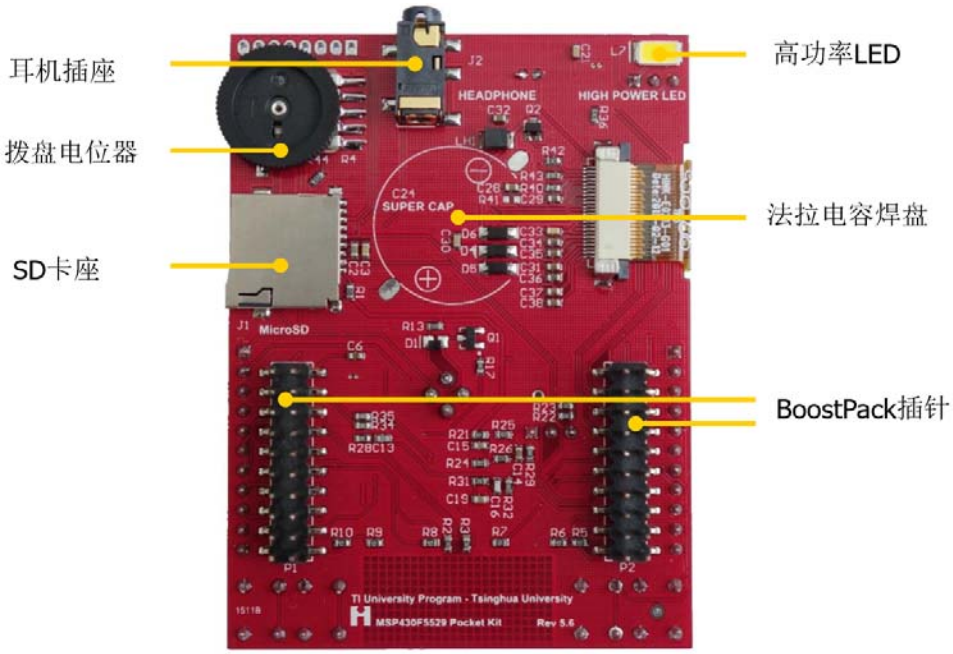


图 1-5（b） 口袋板硬件资源功能图

如图 1-6（a）所示，口袋板正面设置了 6 个 LED 指示灯（LED1-LED6），4 个机械按键（S1-S4），2 个电容触摸按键（Pad1-Pad2），背面上部有拨盘电位器，供学习 MSP430 单片机基本 GPIO、时钟系统、定时器、ADC 等模块使用。

如图 1-6（b）所示，电子纸屏幕正下方是蜂鸣器/扬声器插座，有 4 个圆孔座，当把无源蜂鸣器的两个管脚插入上下两个圆孔座中，蜂鸣器作为无源蜂鸣器使用，我们可以通过 F5529 的 P3.6 口控制蜂鸣器的发声频率。当把蜂鸣器插入左右两个圆孔座中，是将无源蜂鸣器当做扬声器使用，因此我们一定要选用低阻抗（比如 16Ω）的无源蜂鸣器，否则会因为阻抗太大，声音很小。当然，我们也可以将正规扬声器的两个引脚接入这两个圆孔座。左右两个圆孔座分

别与右侧扩展接口 P3.2（VO+）、P3.1（VO-）是连通的。

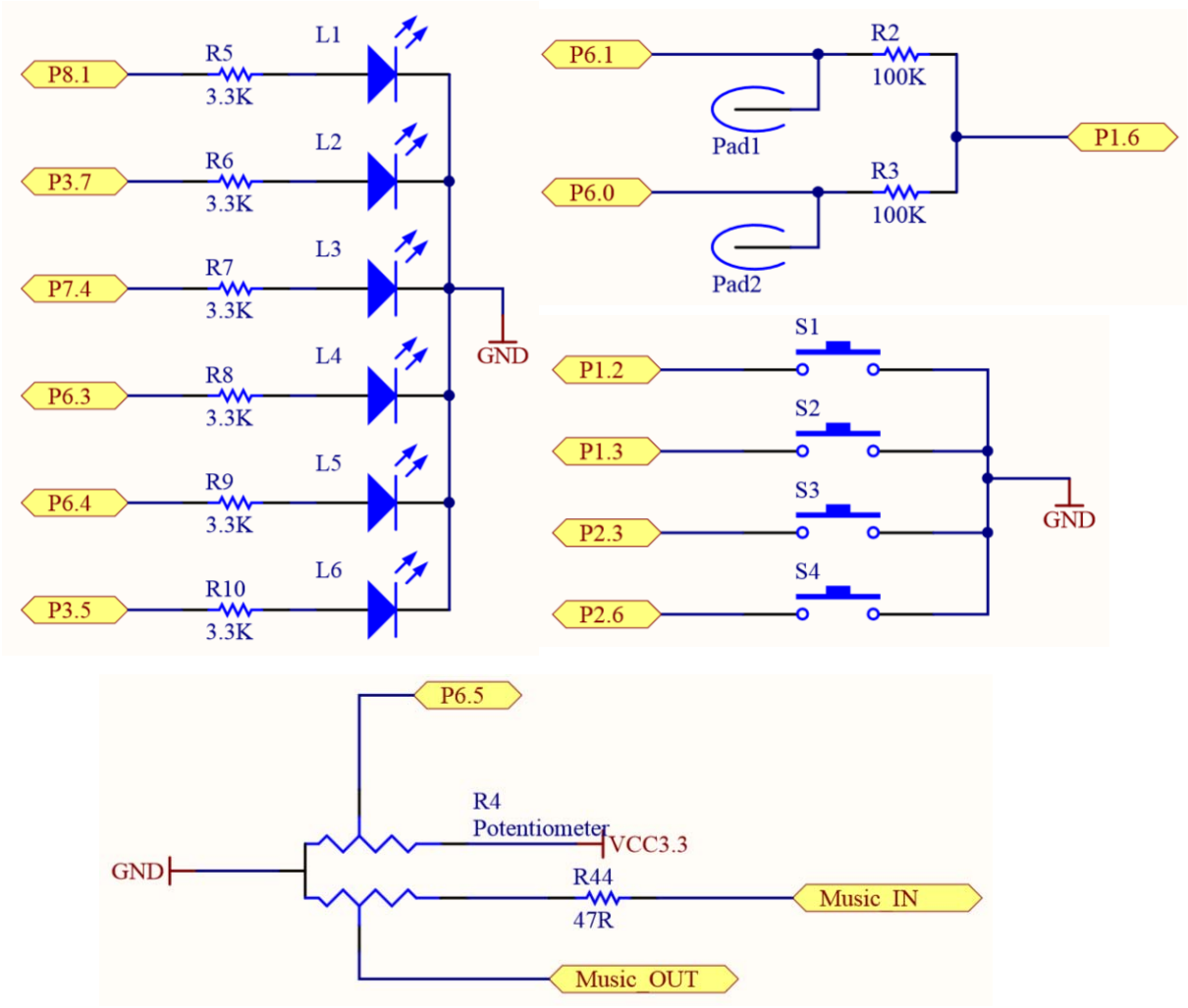


图 1-6（a） 部分按键、指示灯、拨码盘原理接线图

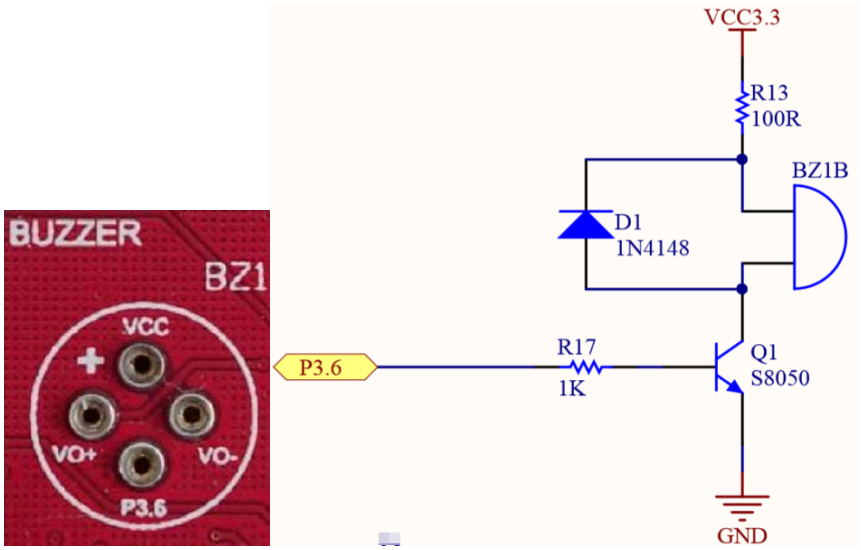


图 1-6（b）蜂鸣器 PCB 板及原理接线图

1.3.2 编译环境介绍

CCS（Code Composer Studio）是 TI 公司研发的一款具有环境配置、源文件编辑、程序调试、跟踪和分析等功能的集成开发环境，能够帮助用户在一个软件环境下完成编辑、编译、链接、调试和数据分析等工作。CCSv7.3 为 CCS 软件的较新版本，功能更强大、性能更稳定、可用性更高，是 MSP430 软件开发的理想工具。

1.3.2.1 CCSv7 的安装

（1）运行下载的安装程序 ccs_setup_xxx.exe（xxx 代表 CCS 版本号），安装包路径不能包含中文字符，只认英文字符，这是很多编译软件安装时都有的特点，要特别注意。当运行到如图时，选择 CCS 安装路径，默认路径是 c:\ti，但如果 C 盘装有还原卡或是空间很小，请选择安装到其他硬盘分区。

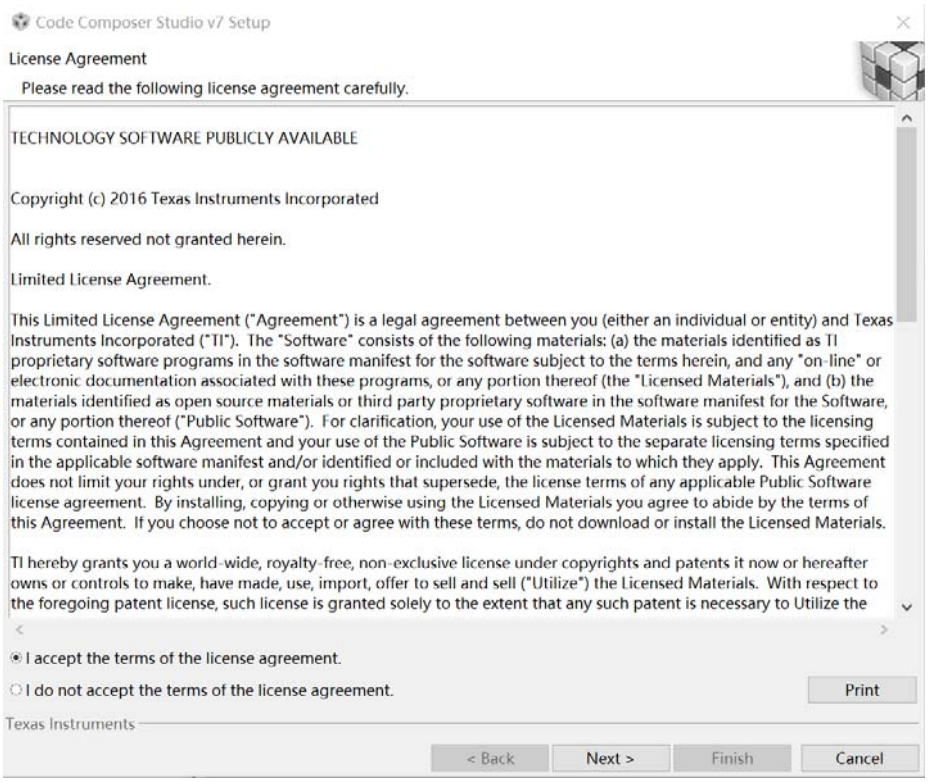


图 1-7（a） CCS 安装流程图例 License Agreement

（2）单击 Next 得到图 1-7（c）选择产品处理器窗口，为了安装快捷，在此只选择支持 MSP430 UltraLow Power MCUs 的选项即可。单击 Next，继续安装。

（3）按照默认选择继续安装，直到如图 1-8 提示安装成功。

（4）单击 Finish，将运行 CCS，弹出如图 1-9 所示窗口，我们可以在电脑中合适的区域建立工作空间（workspace），尽量**不要**将"Use this as the default and do not ask again"选项勾选上，因为以后想要更改工作空间将非常麻烦。

（5）单击 OK，至此软件安装配置工作全部完成，如图 1-10 所示。

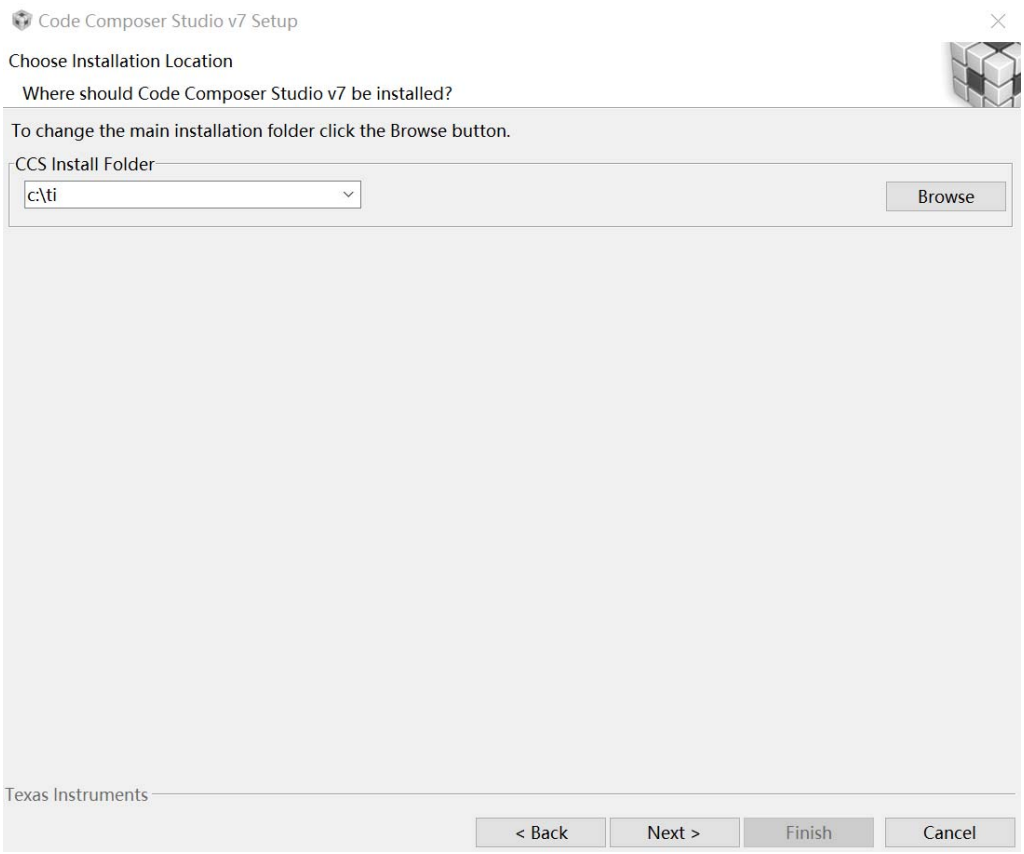


图 1-7（b） CCS 安装路径图例

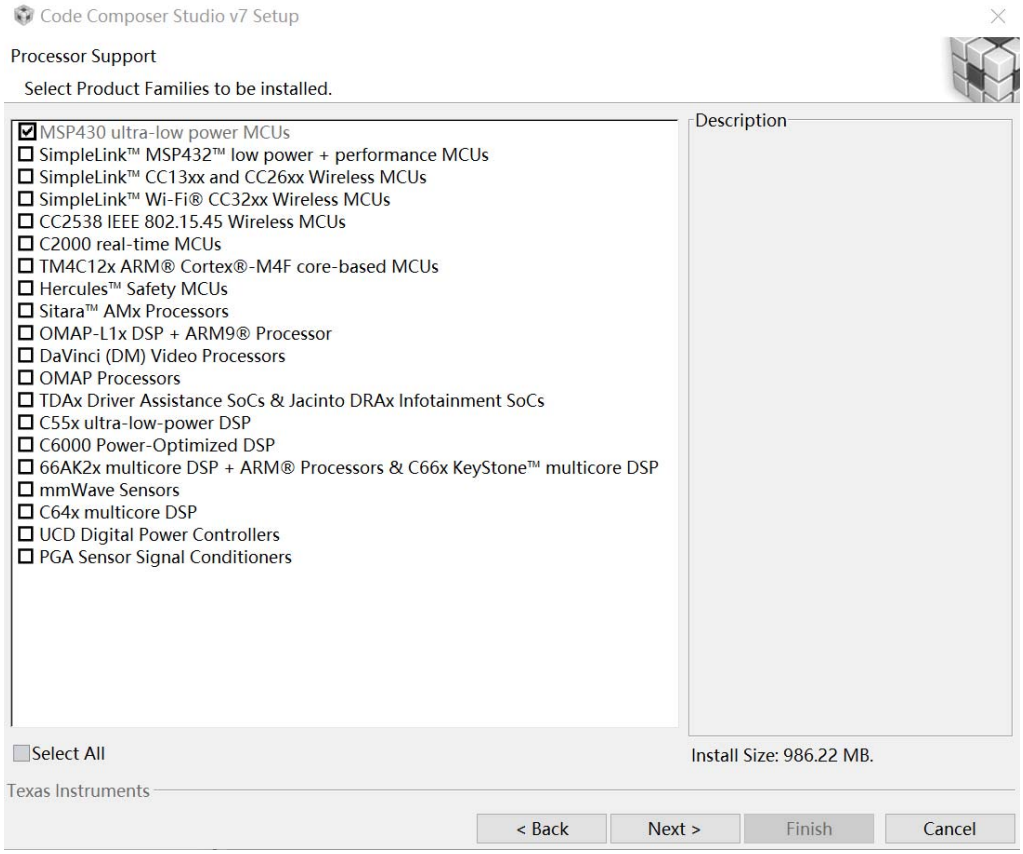


图 1-7（c） CCS 安装选择产品处理器图例

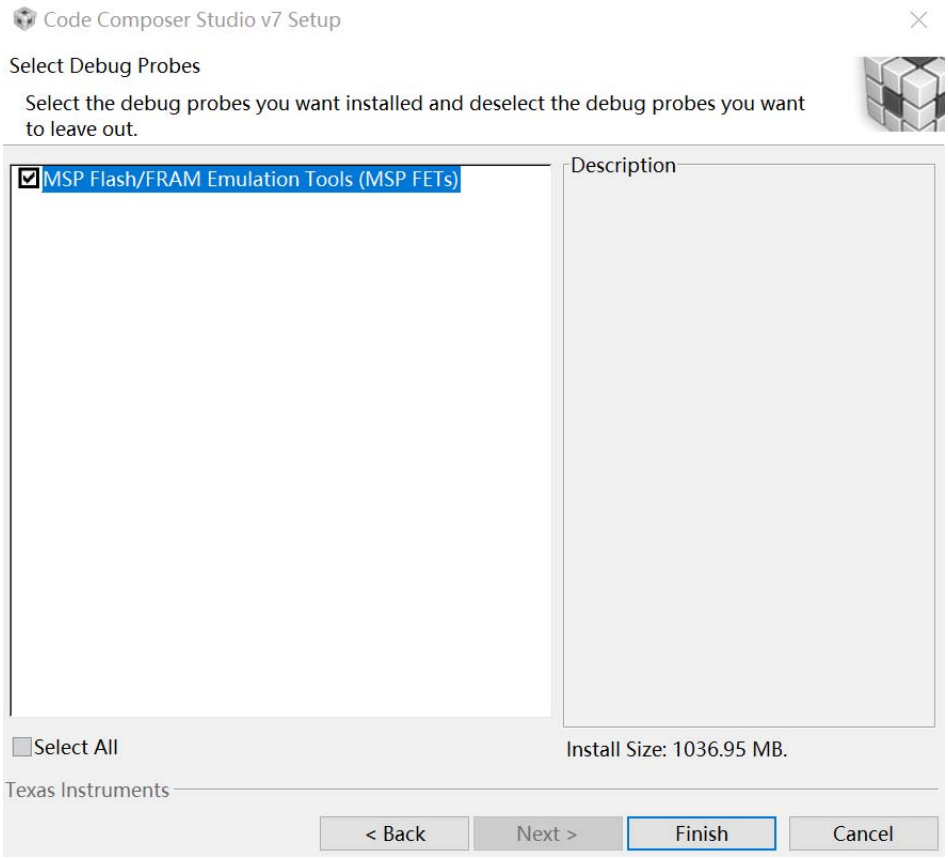


图 1-7（d） CCS 安装 Select Debug Probes 仿真工具图例

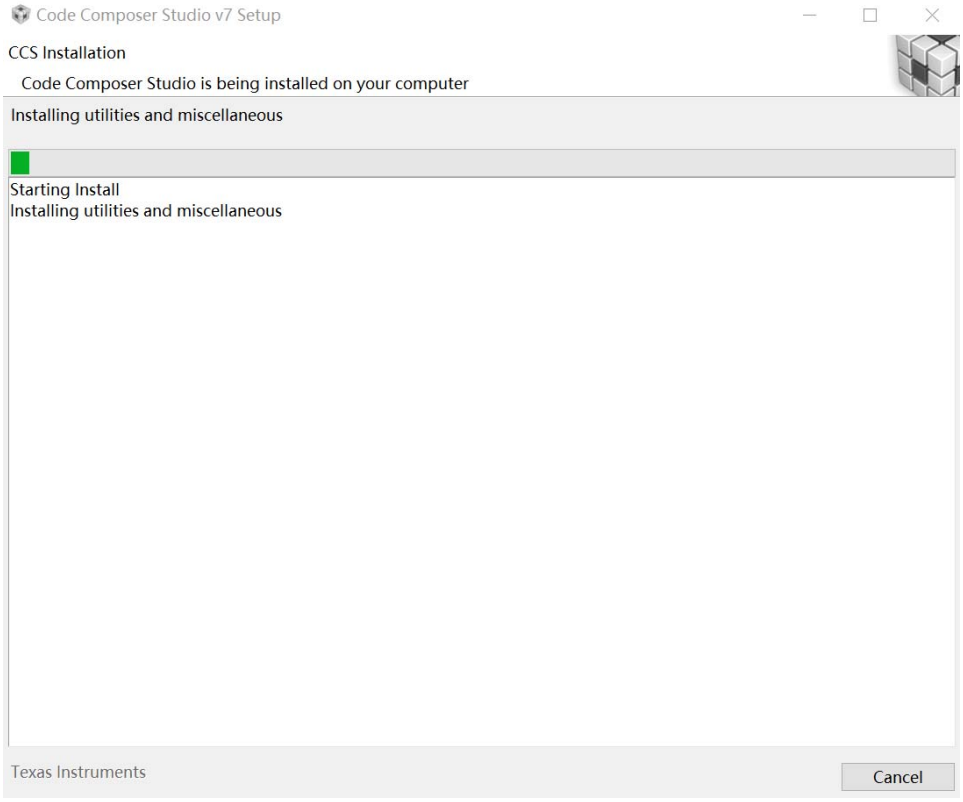


图 1-7（e） CCS 安装正在安装过程图例

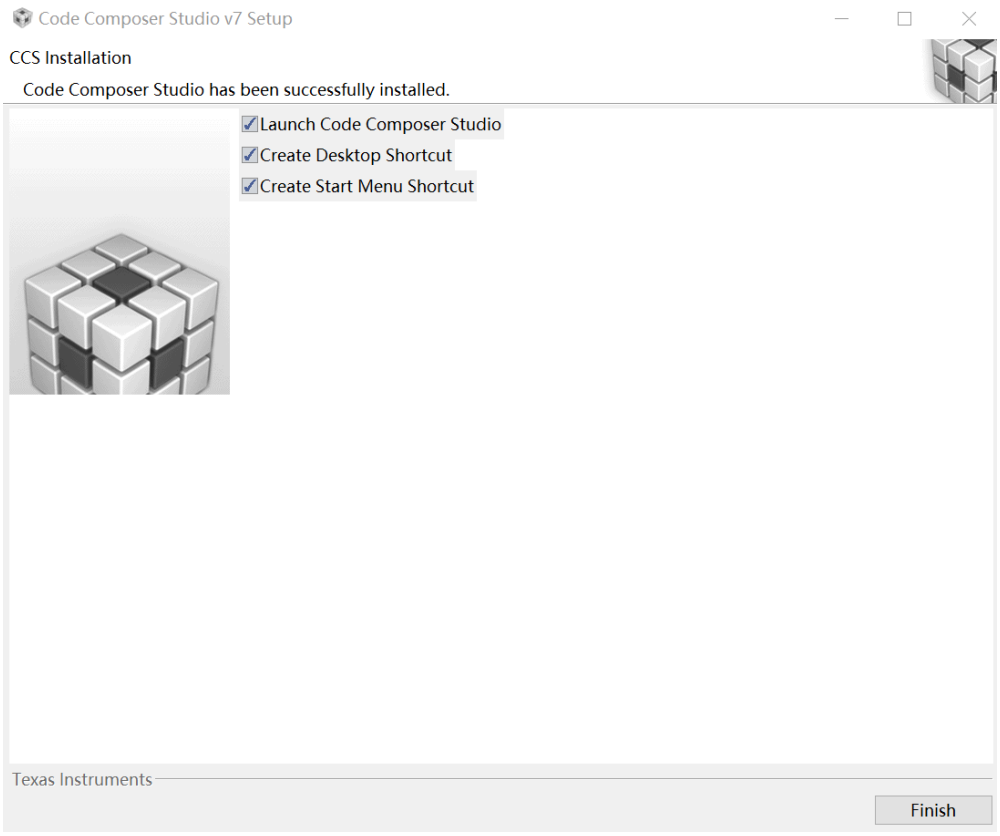


图 1-8　CCS 安装完成图例

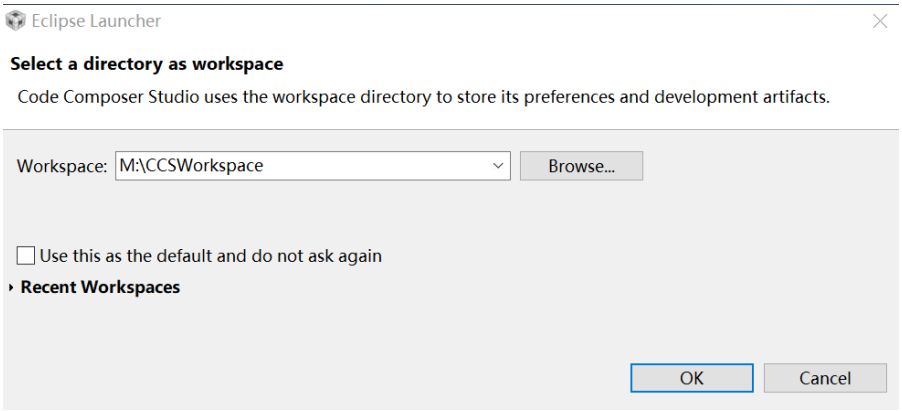


图 1-9 建立工作空间（workspace）

1.3.2.2　利用 CCS 导入已有工程

（1）打开 CCS 后，在 File 菜单栏 Import 选项中，或者在工程浏览窗口（Project Explore）空白处点击右键 Import 选项，或者在 Project 菜单栏中选择 Import CCS Project，打开如图 1-11 界面。

其中第一项为需要导入的工程所在目录（源目录）；最下面的选项是问要不要把源目录中的工程复制到当前所建的 workspace（目标目录）中，我们选择“要”，然后单击 Finish，即可完成既有工程的导入。

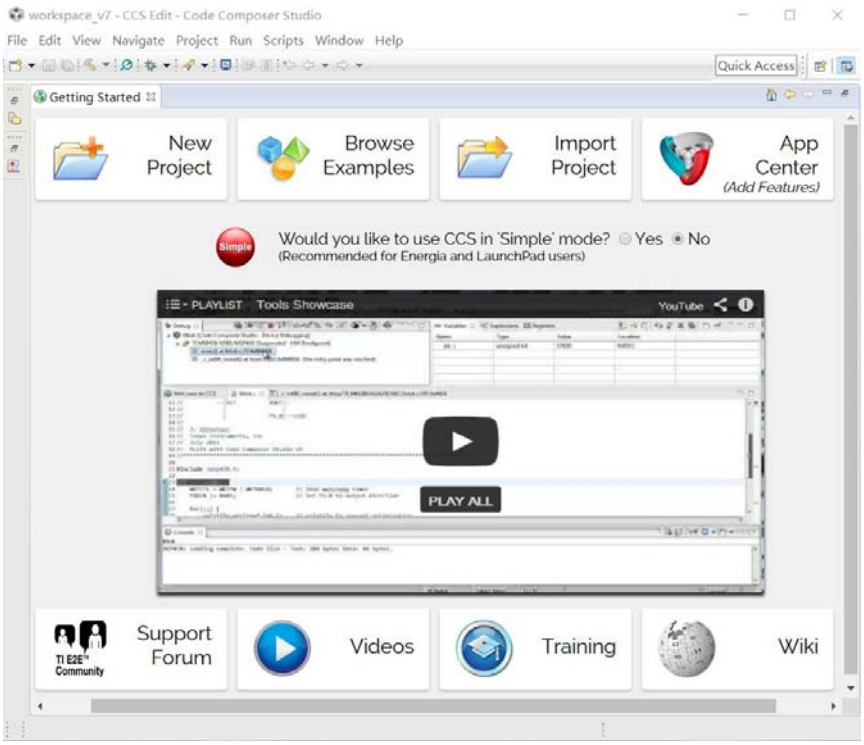


图 1-10 CCS 工作界面

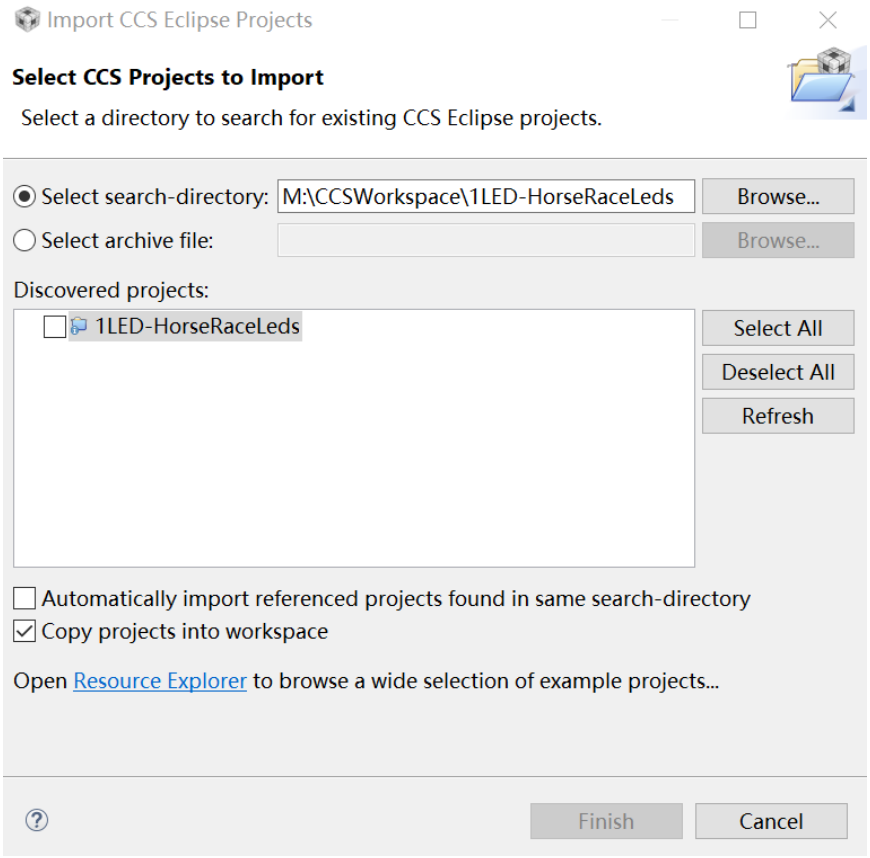


图 1-11 导入已有工程界面

1.3.2.3 利用 CCS 新建工程

（1）在 File 菜单栏 New 选项中，或者在工程浏览窗口（Project Explore）空白处点击右键 New 选项，或者在 Project 菜单栏中选择“New CCS Project”选项，然后将正确的器件系列，型号填入 Target 对话框。接着在在 Project name 中输入新建工程的名称，其余选择默认选项即可。注意，根据编程习惯，在建立空工程时，可选择工程自带 main.c 的选项，源文件中自动引用 msp430.h 头文件。

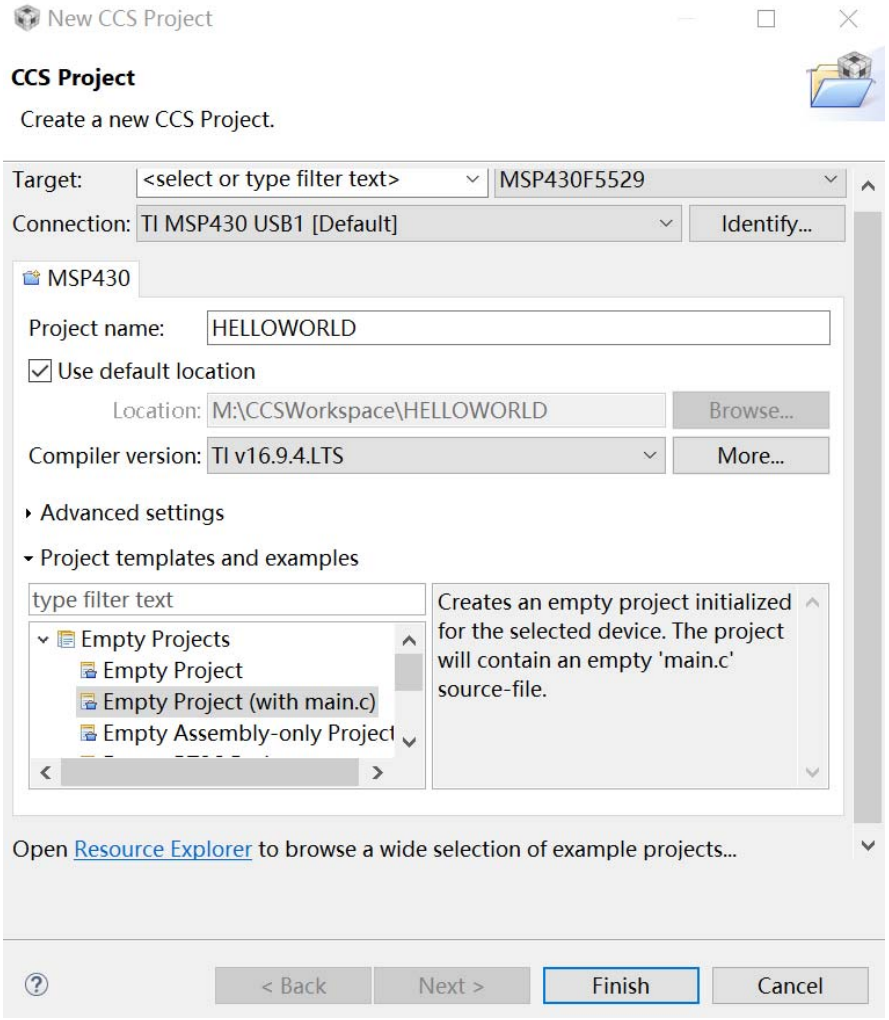


图 1-12 新建工程界面

1.3.2.4 利用 CCS 调试工程

（1）单击 CCS 工具栏上绿色的 Debug 按钮 ，就可以对工程文件进行编译、链接，生成可执行文件，然后下载到 MSP430F5529LD 芯片中。在编译过程中会出现低功耗语法扫描对话框，其中含义是问我们是否对对程序进行低功耗语法扫描，如果我们选择进行语法扫描（选择 Proceed），当语法正确但不符合低功耗规则时，编译器会给出警告信息；我们也可以选择不进行低功耗扫描（选择 Cancel），自行选择即可。

（2）编译下载完成后的 CCS 界面如下图所示。

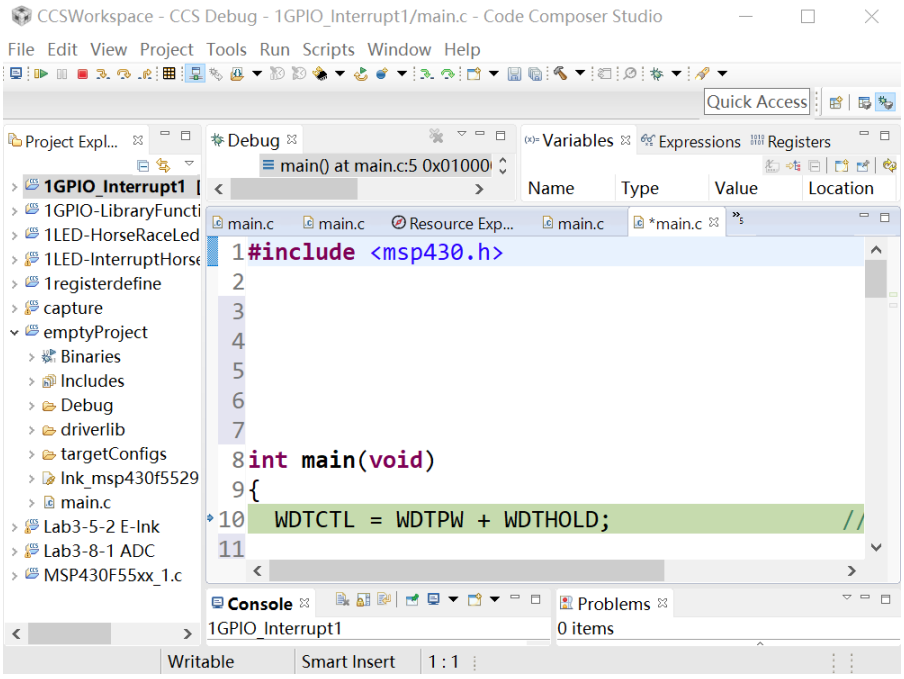


图 1-13 工程编译完成后界面

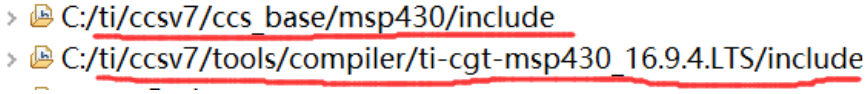
单击运行图标运行程序，观察显示的结果。在程序调试的过程中，可通过设置断点来调试程序：选择需要设置断点的位置，右击鼠标选择 Breakpoints->Breakpoint，断点设置成功后将显示图标，可以通过双击该图标来取消该断点。程序运行的过程中可以通过单步调试按钮配合断点单步的调试程序，可通过中止按钮返回到编辑界面。

1.3.2.5 CCS 下系统函数及宏定义相关资源说明

嵌入式系统编程有硬件底层下的汇编语言编程，基于寄存器的 C 语言编程以及基于 API 的应用层 C 语言编程等几种开发方式。越接近底层的编程对硬件资源掌握的要求越高，应用层的特定功能的开发反而相对简单，只要实现相应的算法，即使对硬件不很熟悉都可以完成相应的功能。电子工艺实习要求的是基于寄存器的 C 语言编程方式，需要学生学会使用寄存器的设置方式，进而通过软件来控制硬件的工作状态。

CCS 编译环境已经将开发板上面的所有寄存器进行了宏定义，并且对相应的精简指令集（RISC）中所有指令进行了封装，同时对字节和字以及位都进行了宏定义，方便了使用者直接使用。相关的头文件 in430.h 、msp430.h、msp430f5529.h、legacy.h 等。对于特定的模块，如系统时钟等，TI 提供了部分官方 API，可以在 CCS 中联网下载使用。

相应的资源可由如下路径查询。



1.3.3 C 语言编程基础

在写 C 语言的过程中，尽量消除不同 CPU 的差异，或者将差异集中到一个地方做修改，那么就能方便的实现代码移植。我们现在写 C 程序，就必须按此要求严格要求自己。这样才能一通百通，才能减少重复劳动。

（1） 变量

变量按照二进制位数有 8 位的字节，16 位的字，32 位的双字等，各有用处。一些特殊的关键字有

```
const unsigned char Table[7]={1,2,3,4,5,6,7}
static      int a;      //本地全局变量
volatile    int b;      //不被优化
_no_init    int c;      //不对其初始化
```

（2） 数学运算

首先，尽可能避免浮点数运算。运算非常慢且占用 RAM 多，所以应尽量避免使用浮点数 float。其次，防止定点数溢出，如，

```
long int x;
int   a;
x=a*1000;            //a 和 1000 都是 int 型，a>65 溢出
```

应改为

```
x=a*long(1000); 或  x=(long)a*1000;
```

（3） 位操作

精简指令处理器如何写 IO 口？

```
P2OUT = P2OUT | 0x01;       // P2.0 置高，按位或
P2OUT |=0x01;               //  一般均简写成这样
P2OUT &=~0x01;              // P2.1 置低，按位与
P2OUT ^=0x04;               // P2.2 取反，按位异或
#define BIT0     （0x01）     //  宏定义
......
P2OUT |=BIT0;                   // P2.0 置高
P1OUT &=~(BIT1+BIT2+BIT3) // P1.1、P1.2、P1.3 置低
```

精简指令处理器如何读 IO 口？

```
char Key;
If((P1IN&BIT5)==0)          P2OUT |=BIT0;
If(P1IN&BIT5)               P2OUT |=BIT0;
If(P1IN&(BIT5+BIT6))        P2OUT |=BIT0;
If(P1IN&BIT5)       Key=1;                  //读 P1.5 值赋 Key
else                Key=0;
```

（4） 寄存器操作

处理器是已经设计完成的具有完备功能的模拟电路，那么操作单片机的寄存器，即通过选择一系列开关，就可以完成各种不同的功能。越是功能强大的处理器，需要配置的寄存器越多。处理器的用户手册就是用来查寄存器功能的。

查用户手册，找到控制串口收发的是 IE1 寄存器的最高两位，我们可以用下面的赋值。

IE1 |= BIT6　　//开串口收中断
IE1 |= BIT7　　//开串口发中断
为便于记忆和理解，头文件中有如下宏定义：
#define URXIE0　　（0x40）　//在 MSP430x42x.h
#define UTXIE0　　（0x80）　//头文件中已有
IE1 |=URXIE0+UTXIE0
以后我们接触高级处理器的程序中，大部分都是这么写，不会像 51 单片机里面直接写 TMOD=0x20 这样 。
特别注意：使用“|=”赋值不会影响其他位，但要搞清楚是不是要先对标志位清 0。

（5） 内部函数及库函数

IAR EW430 提供 100 个库函数，如，
Ctype.h　　字符处理类
Math.h　　数学类
Stdio.h　　输入和输出类
Stdlib.h　　通用子程序类
String.h　　字符串处理类
库函数是 C 语言通用的，有些模块库函数可以在 CCS 的 Resource Explorer 中下载和寻找。系统函数与特定处理器有关，有几个常用的系统函数如下。

```
#define _nop()                  __no_operation()              // 空操作
#define _no_operation()         __no_operation()              // 空操作
#define _enable_interrupt()     __enable_interrupt()          // 开全局中断
#define _enable_interrupts()    __enable_interrupt()
#define _disable_interrupt()    __disable_interrupt()         // 关全局中断
#define _disable_interrupts()   __disable_interrupt()
#define _set_interrupt_state(x) __set_interrupt_state(x)
#define _get_interrupt_state()  __get_interrupt_state()
#define _delay_cycles(x)        __delay_cycles(x)             // 延时
```

1.4 基础 GPIO 实验

1.4.1 GPIO 概述

GPIO (General Purpose I/O)，通用输入输出端口。GPIO 基本都是用于芯片与片外器件或设备的交互。如，检测数字输入，如键盘或开关信号；驱动 LED，蜂鸣器或 LCD 等其他指示器；控制片外器件，较高级的使用可以用它们（通过程序）模拟很多器件的时序达到控制相应器件的目的，比如模拟 SPI 和模拟总线等。
GPIO 是 MCU 与外界交互的重要途径，它具有如下的特性：
（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 的输出状态（高/低电平）；
（3）所有 GPIO 口在复位后都有个默认方向（或输入或输出）。
GPIO 口都是按组规划，有的芯片是 8 个 GPIO 口一组，有的是 16 个或 32 个为一组。一

般每个 GPIO 口都需要做两个寄存器位：一是选择口线方向（输入输出）二是需要一个数据位（用于设置输出数据和读取输入数据）。所以一组 GPIO 口至少会有两个寄存器 GPIOxDIR 和 GPIOxDATA。

（1）GPIOxDIR:控制各个 GPIO 口的方向；

（2）GPIOxDATA:用于各个 GPIO 口的输入输出数据。

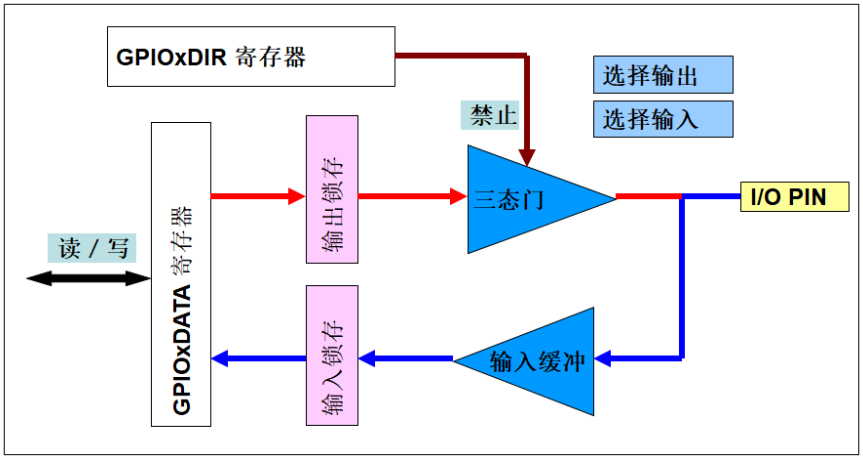


图 1-14 寄存器工作图

1.4.1.1 MSP430 的 GPIO 特点

（1） 端口类型丰富

有端口 P1、P2、P3、P4、P5、P6、P7、P8、P9 、P10 、P11 、S 和 COM。产品因型号不同可包含上述所有或部分端口。如下表所示：

表 1-1 MSP430 端口资源

器件	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	PJ	S	COM
MSP430X13/14/15/16	√	√	√	√	√	√								
MSP430X4XX	√	√	√	√	√	√							√	√
MSP430F5438/36/19	√	√	√	√	√	√	√	√	√	√	√	√		
MSP430F5529/27/25	√	√	√	√	√	√	√	√				√		
MSP430F663X	√	√	√	√	√	√	√	√	√			√	√	√

具有中断能力的端口：P1 和 P2。端口 P1 和 P2 具有输入输出、中断和外部模块功能。这些功能可以通过它们各自 9 个控制寄存器的设置来实现。

不具有中断能力的端口：P3 和其他端口。P3 和其他端口没有中断能力，其余功能同 P1 和 P2，可以实现输入 / 输出功能和外围模块功能。

（2） 端口功能丰富

MSP430 各端口和功能，如下表所示。

表 1-2 MSP430 端口功能

端口	功能
P1、P2	I/O、中断能力、其他片内外设功能
P3、P4、P5、P6、P7、P8、P9、P10、P11	I/O、其他片内外设功能
PJ	I/O、JTAG 功能复用
S、COM	I/O、驱动液晶

端口引脚还可以独立的配置成特殊功能，例如：

- USART – 通用串行同步/异步通信模块；
- 模拟信号比较器；
- 模拟—数字转换器；
- 其他功能 (请参见具体芯片的数据手册)。

（3） 端口寄存器丰富

MSP430 各种端口有大量的控制寄存器供用户操作。最大限度提供了输入/输出的灵活性。

- 每个 I/O 口都可以独立编程。
- 输入或输出可任意组合。
- P1 和 P2 所有 I/O 口都具有边沿可选的输入中断功能。
- 可以按字节输入输出，也可按位进行操作。
- 可设置 I/O 口的上拉或下拉功能。
- 可配置 I/O 驱动能力（高驱动强度或低驱动强度）。

1.4.1.2 GPIO 寄存器

（1） PxDIR 输入 / 输出方向寄存器

相互独立的 8 位分别定义了 8 个引脚的输入/输出方向。使用输入和输出功能时，应该先定义端口的方向。

PxDIR 配置：

- Bit = 1: 将端口引脚设置为输出模式；
- Bit = 0: 将端口引脚设置为输入模式。

例，设置 P1 端口的 P1.0 引脚为输出方向，其余引脚（P1.1~P1.7）设置为输入方向。

- P1DIR = 0x01;

（2） PxIN 输入寄存器

该寄存器是只读寄存器，即用户不能对它写入，其中的每一位都反映了其对应的 I/O 引脚的输入信号(引脚配置为通用 I/O)。

PxIN 配置:

- Bit = 1: 输入为高电平;
- Bit = 0: 输入为低电平。

（3） PxOUT 输出寄存器

输出寄存器是可读可写的。这个寄存器的每个位都反映了写入相应输出引脚的值。将需要的值写入该寄存器，控制输出引脚的电平状态。

PxOUT 配置:

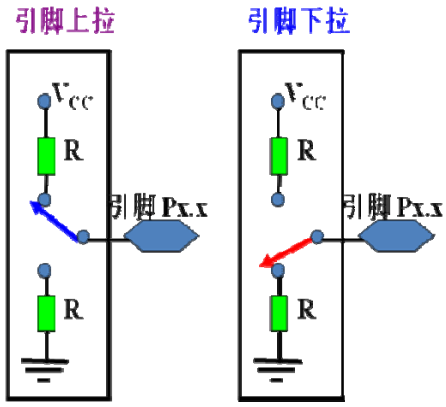
- Bit = 1: 输出为高电平;
- Bit = 0: 输出为低电平。

（4） PxREN 上拉或下拉电阻使能寄存器

PxREN 寄存器中的每一位可使能或禁用相应 I/O 引脚的上拉/下拉电阻。

PxREN 配置:

- Bit = 1: 使能上拉/下拉电阻;
- Bit = 0: 禁用上拉/下拉电阻。



当使能引脚上拉或下拉功能时，通过设置 PxOUT 相应位来选择。上拉电阻简单来说就是把电平拉高，通常用 4.7-10K 的电阻接到 Vcc 电源。下拉电阻则是把电平拉低，电阻接到 GND

地线上。

（5） PxSEL 功能选择寄存器

I/O 端口还具有其他片内外设功能，为减少引脚，将这些外设功能与 I/O 端口引脚复用来实现。PxSEL 来选择引脚的 I/O 端口功能与外围模块功能。

PxSEL 的配置:

- Bit = 0: 选择引脚为 I/O 端口;
- Bit = 1: 选择引脚为外设功能。

（6） PxDS 输出驱动强度寄存器

PxDS 寄存器的每个位，设置引脚的输出强度为高驱动强度或低驱动强度。默认值为低驱动强度。

PxDS 的配置:

- Bit = 0: 低驱动强度；
- Bit = 1: 高驱动强度。

（7） PxIE 中断使能寄存器（仅中断端口 P1 和 P2）

该寄存器的 8 位与端口的 8 个引脚一一对应，其中某一位置位表示允许对应的引脚在电平变化（上升沿或下降沿）时产生中断，否则，表示禁止该位的中断。

每个 PxIE 位使能的中断请求都与相应的 PxIFG 中断标志相关联，可通过写 PxOUT 和 PxDIR 来设置 PxIFG。

PxIE 的配置:

- Bit = 1: 允许中断；
- Bit = 0: 禁止中断。

（8） PxIES 中断触发沿选择寄存器（仅中断端口 P1 和 P2）

如果允许 Px 口的某个引脚中断(即 PxIE 和 GIE 已设置)，还需定义该引脚的中断触发方式。

该寄存器可读可写，寄存器的 8 位分别对应 Px 口 8 个引脚。

PxIES 的配置:

- Bit = 1: 下降沿使相应中断标志置位；
- Bit = 0: 上升沿使相应中断标志置位。

（9） PxIFG 中断标志寄存器（仅中断端口 P1 和 P2）

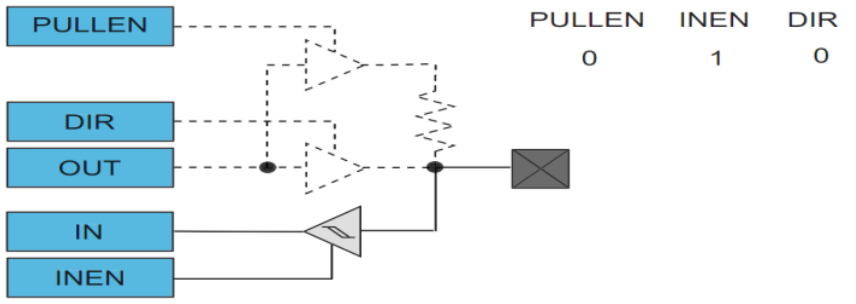
该寄存器用来表示对应引脚是否产生了由 PxIES 设定的电平跳变。如果在 GIE 置位，引脚对应的中断使能寄存器 PxIE 位置位，则会向 CPU 请求中断处理。中断标志 PxIFG.0～PxIFG.7 共用一个中断向量， PxIFG.0～PxIFG.7 不会自动复位。必须用软件来判定是对哪一个事件服务，并将相应的标志复位。

PxIFG 的配置:

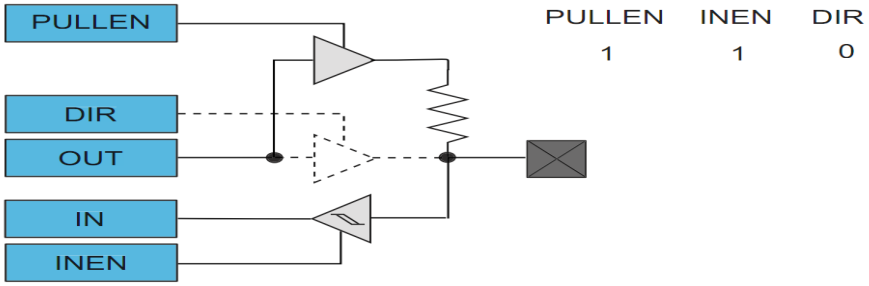
- Bit = 0: 没有中断请求；
- Bit = 1: 有中断请求。

1.4.1.3 GPIO 工作模式

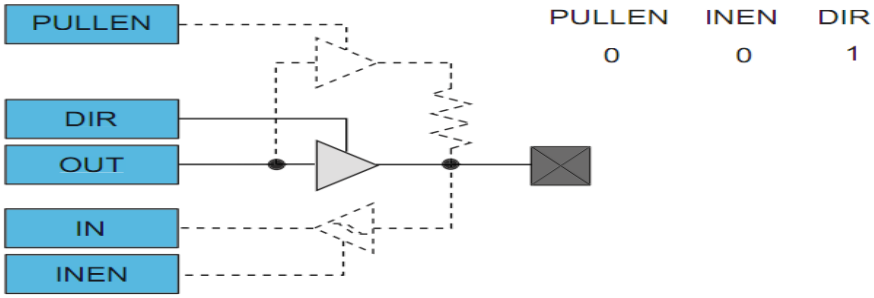
（1） 标准输入



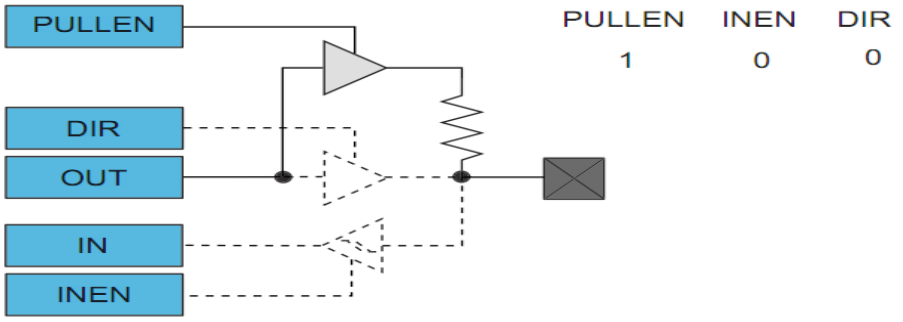
（2） 上（下）拉输入



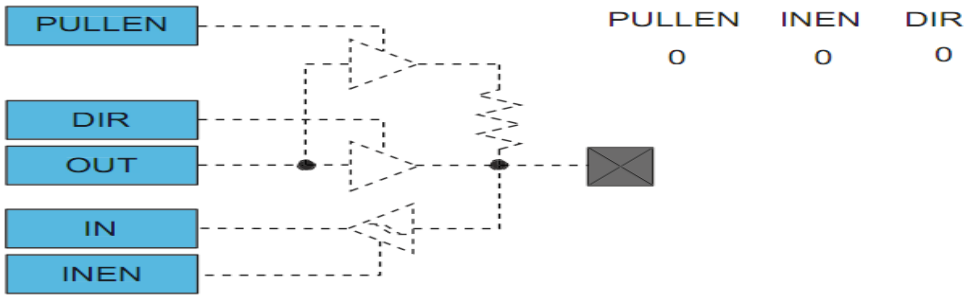
（3） 输入禁用的推拉输出



（4） 上（下）拉输出



（5） 复位或模拟 I/O



1.4.1.4 GPIO 官方库函数

GPIO 的 API 被分成三组函数：GPIO 的引脚配置函数，处理中断的函数，以及引脚状态的函数。库函数可在 CCS 的 Resource Explorer 中下载和寻找。

（1） GPIO 的引脚配置函数

a) GPIO_setAsOutputPin() // 设置为输出引脚
b) GPIO_setAsInputPin() // 设置为输入引脚
c) GPIO_setAsInputPinWithPullDownResistor () // 设置为输入下拉
d) GPIO_setAsInputPinWithPullUpResistor () // 设置为输入上拉
e) GPIO_setDriveStrength() // 设置引脚驱动强度
f) GPIO_setAsPeripheralModuleFunctionOutputPin() // 设置为外设输出引脚
g) GPIO_setAsPeripheralModuleFunctionInputPin() // 设置为外设输入引脚

（2） GPIO 中断处理函数

a) GPIO_enableInterrupt() // 使能中断
b) GPIO_disbleInterrupt() // 禁止中断
c) GPIO_clearInterrupt() // 清除中断标志位
d) GPIO_getInterruptStatus() // 获取中断状态
e) GPIO_interruptEdge() // 选择中断沿

（3） GPIO 引脚状态的函数

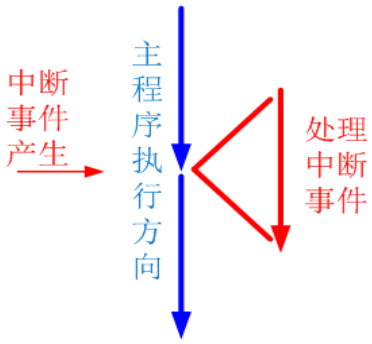
a) GPIO_setOutputHighOnPin() // 引脚输出为高
b) GPIO_setOutputLowOnPin() // 引脚输出为低
c) GPIO_toggleOutputOnPin() // 翻转引脚
d) GPIO_getInputPinValue() // 获取引脚输入值

1.4.1.5 GPIO 中断

什么是中断？

先不以专业术语来描述。首先大家理解，单片机是按照程序一步一步执行的，点击单步运行就可以看到光标向下移动一步，即单片机向下移动一步，就是单步运行。就像人一样，在写作业时需要去把烧开的水关掉，就需要停止手头的事情去关掉开水，这就产生了一个中断的概念。对于单片机也是这样，之前写的程序中，都会有一个死循环，在这个死循环里面完成所需事件，但这时，如果系统告诉我们去点亮一个灯，那单片机就需要停止死循环里面的动作，

先去点灯，然后再跑回来继续执行死循环里面的代码。通过图示我们可以更加了解他的机制。



中断装置和中断处理程序统称为中断系统，中断系统是单片机中重要的组成部分。中断是 CPU 对系统发生的某个事件作出的反应，暂停正在运行中的程序，转去执行处理相应的中断事件，完毕后返回被中断的程序继续运行。中断系统的使用大大提高了 CPU 的效率，中断的实现由软件和硬件综合完成。

MSP430F5529 的中断装置比较多，其主要的有 IO 口中断、定时器中断、串口中断、外部中断、ADC 转换中断、看门狗中断、捕获比较中断等，这些装置都能引发中断事件。中断分为不可屏蔽（nonmaskable）中断与可屏蔽（maskable）中断两大类，各种中断的优先级也是不同的，MSP430F5529 的一些中断源、中断标志及优先级等具体可查阅参考书或者用户手册。

中断的具体过程如下：

- 事先将中断服务程序入口地址装入中断向量表；
- 中断发生后，如果中断被允许（可屏蔽中断），CPU 将当前程序地址和 CPU 状态寄存器 SR 压入堆栈；
- 跳转到中断服务程序入口，备份寄存器入堆栈；
- 开始执行中断服务程序；
- 退出中断前，恢复寄存器。CPU 取回 SR 寄存器，跳转回中断前主程序地址。

对于 430 来讲，P1 和 P2 口都可以作为单片机的中断入口，由三个寄存器进行控制。具体是 PxIFG，PxIE，PxIES。有些器件可能会有些不同，具体要看他们的数据手册。那么这三个寄存器分别是什么呢？具体可参考 1.4.1.2 节寄存器的介绍。

PxIFG（Port x Interrupt Flag Register） 端口 x 中断标志寄存器
PxIE（Port x Interrupt Enable Register） 端口 x 中断使能寄存器
PxIES（Port x Interrupt Edge Select Register）端口 x 中断边沿选择寄存器

使用端口中断的一个具体的步骤是什么？

（1）配置端口，设置端口的方向。

也就是设置 PxDIR 的方向，设置上下拉电阻等。注意：如果外部硬件上面没有上下拉电阻的话，这里一定要进行配置，否则将导致输入电平不稳定。如果是机械按键输入，可以通过 PxREN 启用内部上下拉电阻，根据按键的接法，设定 PxOUT（决定最终是上拉电阻还是下拉电阻）。

（2）设置中断的触发模式，通过写 PxIES，上升沿还是下降沿或者两者均可以产生中断。

（3）开启中断，就是设置 PxIE，通过 PxIE 开启 IO 中断，通过_EINT()开启总中断。

（4）编写中断子函数，在中断子函数中，通过 if 语句查询具体中断的 IO 口，如果是机械按键输入，还需有消抖代码。

（5）根据具体 IO 的输入，编写事件处理函数。

（6）清除中断标志位，设置 PxIFG ，PxIFG = 0。

端口中断的一个具体的步骤大概是如上的一个操作流程，要根据具体的使用情景和方式进行微调。

使用按键时有个现象要特别注意，就是按键按下时会有抖动，那么就会对正常按键按下产生干扰，这时我们就需要消抖处理。

如图所示，机械按键按下和弹起时，会有毛刺干扰。在一次按键过程中，会有若干次下降沿，只有 1 号是真正的按键事件。如何避免其他几次下降沿“中断”的影响呢？

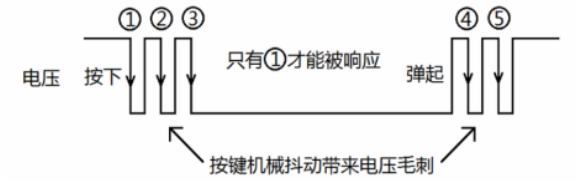


图 1-15 机械按键抖动产生的毛刺

在此提供以下几种方式消除按键抖动，仅供参考。

方法 1： 理论上 Push_Key 的值应该等于 P1IN。 P1IN 和 P1IFG 的区别在于 P1IN 可能有双键被按下的可能性，而中断标志位 P1IFG 只记录最早按下的键。但有一个“干扰”因素，设置为输出口的 IO 其对应的 P1IFG 位也可能是 1，所以用了一个按位与操作代码 P1IFG&(~P1DIR)，将输出 IO 排除在外。

注：P1IN 为 IO 输入寄存器；P1OUT 为 IO 输出寄存器；P1IFG 为 IO 中断标志位寄存器；P1DIR 为 IO 方向寄存器，高电平代表 IO 设为输出，低电平代表 IO 口设为输入。

方法 2：足够长的延时，实际上就是阻塞 CPU，下降沿 1 检测到后，由于延时，下降沿 2 、 3 将不被检测。下降沿 4 检测到后，下降沿 5 也会因为延时不会被检测。

方法 3：延时是无法消灭下降沿 4 的。所以，另加了一条判据，当检测到下降沿 4 ，进中断，延时一段时间再看按键电平时，按键电平将是高，这就说明它是按键弹起阶段的第一个毛刺，可以被排除掉。另一个要说明的是，按键按下后，按键 IO 的电平是 0，而中断标志 IFG 是 1，正常按下时，两个值应该不等。所以判据代码是(P1IN&Push_Key)==0。

方法 4：在用 switch 判断具体中断 IO 时，没有用 P1IN 的值，因为 P1IN 有可能不止 1 个“1”（多个按键被按下）。而应该用 PIIFG 滤除输出 IO 影响后的 Push_Key，Push_Key 有且只有 1 个“1”。

1.4.2 实验内容

1.4.2.1 按键对 LED 灯的控制实验

查询方式实现按键对 LED 灯的控制。此实验目的为，掌握对 IO 口的查询操作和 IO 基本操作的流程，对部分引脚功能有个初步了解。通过按键对 IO 口 P1.2 的操作，查询实现高低电平检测，从而对与 LED 相连的 P8.1 的电平控制，实现 LED 灯的亮灭。程序流程图如图 1-16 所示。

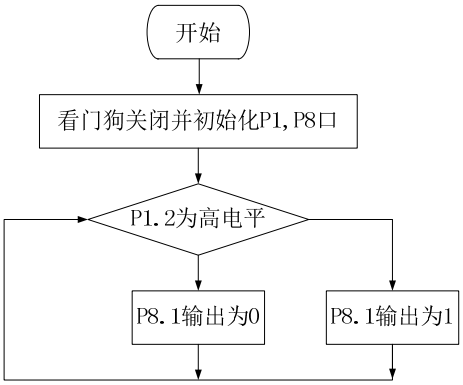


图 1-16 按键对 LED 灯的控制实验程序流程图

实验步骤如下：

（1）打开 CCSv7 并确定工作区间，然后选择 File-->New-->CCS Project 弹出前面图 1-12 对话框；
（2）在 Project name 中输入新建工程的名称；
（3）在 Device 部分选择器件的型号，在此 Family 选择 MSP430，Variant 选择 MSP430X5XX family，芯片选择 MSP430F5529，其余保持默认；
（4）在左下角对话框中，选择 Empty Projects 下拉菜单下的 Empty Project（空工程）或者 Empty Project（with main.c），单击 Finish。
（5）在新窗口中输入代码进行实验和调试。

实验中可能用到的寄存器设置参考：

（1）一个 S1(P1.2)按键输入，选择上（下）拉输入模式，设置两个寄存器 P1REN 和 P1OUT，查询一个寄存器 P1IN 的状态。

★ 使能 P1.2 上下拉电阻功能
★ 置 P1.2 为上拉电阻方式
★ 查询 P1.2 的输入电平

（2）一个 LED1(P8.1)灯输出，设置寄存器 P8DIR。

★ 设置 P8.1 端口为输出模式

实验要求：

（1）实现现象：按键 S1 按下，LED1 灯点亮，按键 S1 松开，LED1 灯熄灭；

（2）使用三种编程方法实现，配置寄存器法，直接调用头文件#include <msp430f5529.h>法，通过使用固件库 driverlib 配置 GPIO 引脚控制法。

- **第一种编程方法**

查找端口寄存器的内存基址，偏移地址，自己定义并且操作寄存器实现。msp430f5529-datasheet.pdf 里面说明了端口基地址和偏移地址。如图 1-17 所示。

Table 6-26. Port P1 and P2 Registers (Base Address: 0200h)

REGISTER DESCRIPTION	REGISTER	OFFSET
Port P1 input	P1IN	00h
Port P1 output	P1OUT	02h
Port P1 direction	P1DIR	04h
Port P1 pullup or pulldown enable	P1REN	06h
Port P1 drive strength	P1DS	08h
Port P1 selection	P1SEL	0Ah
Port P1 interrupt vector word	P1IV	0Eh
Port P1 interrupt edge select	P1IES	18h
Port P1 interrupt enable	P1IE	1Ah
Port P1 interrupt flag	P1IFG	1Ch
Port P2 input	P2IN	01h
Port P2 output	P2OUT	03h
Port P2 direction	P2DIR	05h
Port P2 pullup or pulldown enable	P2REN	07h
Port P2 drive strength	P2DS	09h
Port P2 selection	P2SEL	0Bh
Port P2 interrupt vector word	P2IV	1Eh
Port P2 interrupt edge select	P2IES	19h
Port P2 interrupt enable	P2IE	1Bh
Port P2 interrupt flag	P2IFG	1Dh

图 1-17（a） datasheet 中 P1P2 端口基地址和偏移地址详表

Table 6-29. Port P7 and P8 Registers (Base Address: 0260h)

REGISTER DESCRIPTION	REGISTER	OFFSET
Port P7 input	P7IN	00h
Port P7 output	P7OUT	02h
Port P7 direction	P7DIR	04h
Port P7 pullup or pulldown enable	P7REN	06h
Port P7 drive strength	P7DS	08h
Port P7 selection	P7SEL	0Ah
Port P8 input	P8IN	01h
Port P8 output	P8OUT	03h
Port P8 direction	P8DIR	05h
Port P8 pullup or pulldown enable	P8REN	07h
Port P8 drive strength	P8DS	09h
Port P8 selection	P8SEL	0Bh

图 1-17（b） datasheet 中 P7P8 端口基地址和偏移地址详表

有关编程语句示例如下：

（1）定义基地址

#define P1P2_BASE_Address 0x0200

（2）定义位操作常用的 BIT0-BIT8

#define BIT0 (0x0001)
#define BIT1 (0x0002)
#define BIT2 (0x0004)
#define BIT3 (0x0008)
#define BIT4 (0x0010)
#define BIT5 (0x0020)
#define BIT6 (0x0040)
#define BIT7 (0x0080)

（3）定义相关寄存器

#define P8DIR (*((volatile unsigned char*)(P7P8_BASE_Address + 0x05)))

嵌入式系统编程，要求程序员能够利用 C 语言访问固定的内存地址。既然是个地址，那么按照 C 语言的语法规则，这个表示地址的量应该是指针类型。所以，知道要访问的内存地址后，比如 0x0200，第一步是要把它强制转换为指针类型(unsigned char *)(0x0200)，所以 0x0200 强制转换为指向 unsigned char 类型。volatile（可变的）这个关键字说明这变量可能会被意想不到地改变，这样编译器就不会优化此变量；第二步，对指针变量解引用，就能操作指针所指向的地址的内容了 *(volatile unsigned char *)(0x0200)；第三步，小心地把#define 宏中的参数用括号括起来，这是一个很好的习惯。

- **第二种编程方法**

调用芯片头文件实现该现象。CCS 提供了丰富的宏定义，用户可以自行选择使用芯片的头文件中程序资源，集中在 msp430.h 中，我们使用的是 msp430f5529.h，可以直接调用该头文件，这样就不用自己编写宏定义。

调用芯片头文件

#include <msp430.h>

或者 **#include <msp430f5529.h>**

新建工程时可选择 Empty Project（with main.c）选项。如果选择此选项，请思考里面的内嵌语句是什么作用？（如下所示）

WDTCTL = WDTPW | WDTHOLD;

- **第三种编程方法**

调用 TI 官方库函数实现该现象。官方库函数下载可参考如下方法。

在 CCS 编译环境下，点击菜单 View，选择 Resource Explore，联网后，选择下面路径内的空工程导入工程文件栏，即可获得库函数。

Software/MSP430Ware-v:3.80.03.07/Libraries/Driver Library/MSP430F5xx_6xx/Example Projects

注意：在导入工程之前要首先下载并安装 MSP430Ware 插件如下图 1-18 所示。

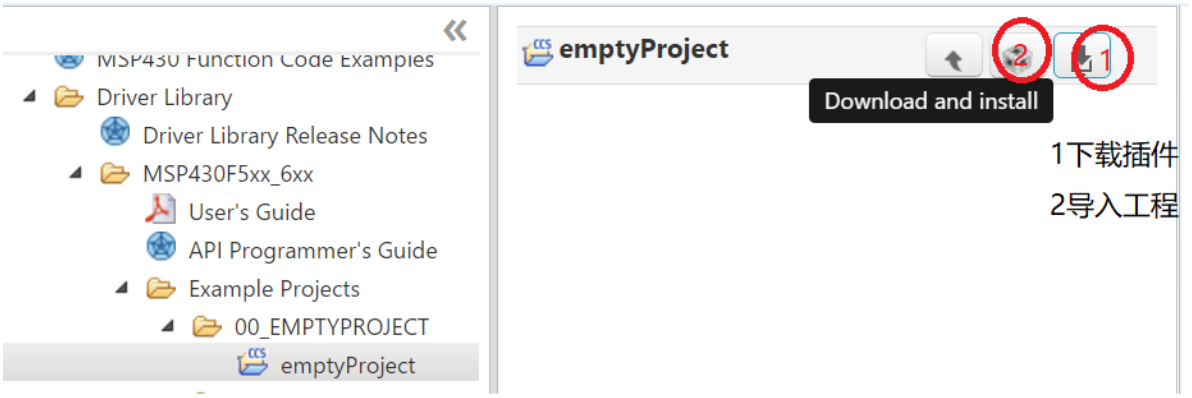


图 1-18 下载官方库函数示例图

在下载导入成功后，即可使用此工程来使用库函数编码实现所需功能。在工程的 driverlib 文件夹中，本次实验可用到 gpio.c 和 gpio.h。

可能用到的库函数（忽略形参）如下，可在 gpio.h 中自行查找具体用法。

GPIO_setAsOutputPin( );

GPIO_setAsInputPinWithPullUpResistor( );

GPIO_getInputPinValue( );

GPIO_setOutputHighOnPin ( );

GPIO_setOutputLowOnPin ( );

1.4.2.2 LED 跑马灯实验（查询方式）

查询方式实现 LED 跑马灯实验。此实验目的为，掌握对 IO 口的查询操作和 IO 基本操作的流程。该实验要求实现 LED 灯的连续点亮的跑马灯效果。板上共有 LED1~LED6 六个 LED 灯，顺序点亮，连续不断。点亮的频率不要求，肉眼可分辨即可。

实验要求：

（1）实现现象：LED 灯的连续点亮的跑马灯效果（肉眼可分辨）；

（2）选取熟悉的或者喜欢的任意一种编程方法实现，配置寄存器法，直接调用头文件 #include <msp430f5529.h>法，通过使用固件库 driverlib 配置 GPIO 引脚控制法。

实验步骤参照 1.4.2.1 节。

1.4.2.3 按键对 LED 灯的控制实验（中断方式）

本实验应用的是 MSP430F5529 的 IO 口中断，主板 S1 上按键连接到 P1.2 口，当 S1 按键被按下时，P1.2 口电平由高变低，触发一个中断事件，然后在 P1 口的中断函数中填写代码改变 LED1 灯的状态。

编程思路，整个编程过程比较简单，只需要配置好相应的 IO 口就可以了，需要设置 P8.1 口为输出状态以控制 LED1 灯状态，还需要使能 P1.2 口的上拉电阻，选择 P1.2 口中断沿，使能中断，清中断标志位，然后进入等待状态，最后再写一个 P1 口的中断服务函数，在函数中改变 LED1 灯的点亮状态。当有按键被按下既产生中断事件，程序转向中断服务函数并改变 LED 灯的状态。

用到的寄存器有 P8DIR、P8REN、P8OUT、P1OUT、P1IES、P1FG、P1IE，具体功能可查看用户手册有关寄存器章节。写中断函数不同于写普通函数，中断函数不需要和普通函数那样声明，在 MSP430 中用拓展关键字“__interrupt”来标明。

示例如下，

```
#pragma vector=PORT1_VECTOR
__interrupt void Port_1_Key(void)
{

}
```

程序流程图如下，

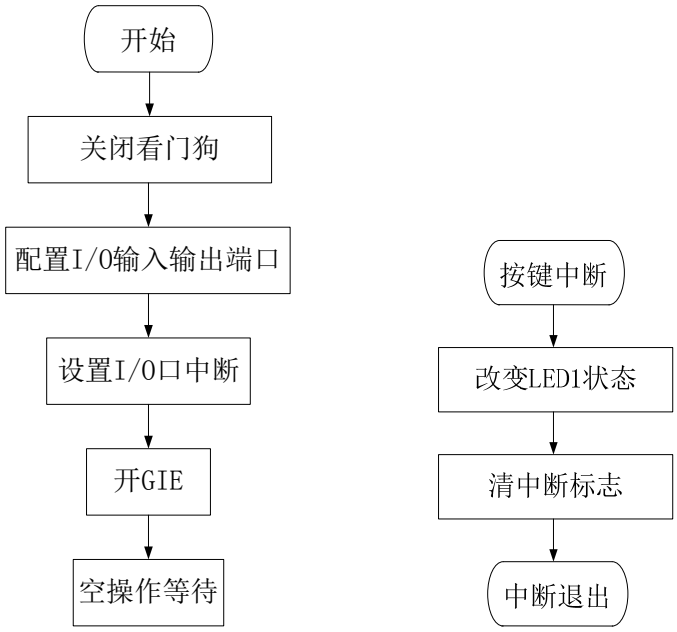


图 1-19 按键对 LED 灯的控制实验（中断方式）程序流程图

实验要求：

（1）实现现象：当 S1 按键被按下松开后，LED1 灯状态翻转（由亮到灭，或由灭到亮）；

（2）选取熟悉的或者喜欢的任意一种编程方法实现，配置寄存器法，直接调用头文件 #include <msp430f5529.h>法，通过使用固件库 driverlib 配置 GPIO 引脚控制法。

1.4.3 本章作业

1.4.3.1 作业 1

对于 1.4.2.1 节，该实验代码的效果是只有按键按下时，LED 灯才会点亮。按键松开后，LED 熄灭。如何修改程序，使按下按键一次，LED 点亮，再次按下按键时，LED 熄灭？

请编程实现。

1.4.3.2 作业 2

实验要求：

使用 GPIO 中断（P1 端口或 P2 端口），实现按键跑马灯。频率不要求，肉眼可分辨即可。

实现现象：

（1）使用口袋板上的任一按键或多个按键，和 L1~L6 的灯，实现按下按键，L1~L6 顺序点亮一次（或者多次不限）；

（2）实现（1）中的跑马灯现象后，选取另外一个按键 S，此按键 S 能控制现象（1），如何控制自行设定。比如说，按下按键 S，则现象（1）不能实现，放开反之。不允许使用查询方式实现，要使用中断方式实现。

1.5 思考与分析

（1） MSP430 的 IO 口都具备哪些功能？
（2） 与 IO 口相关的寄存器有哪些？
（3） 如何处理机械按键产生的毛刺、抖动现象？
（4） 去掉该句代码，程序是否能正常运行？为什么？
WDTCTL = WDTPW+WDTHOLD; // 关闭看门狗

实验二 MSP430F5529LP 系统时钟与定时器

2.1 实验目的

（1） 了解 MSP430F5529LP 时钟系统；
（2） 学习看门狗定时器原理，熟练掌握看门狗定时中断；
（3） 学习 Timer_A 定时器原理，掌握 Timer_A 的定时中断；
（4） 掌握蜂鸣器的使用方法，熟练应用 GPIO 控制 LED 亮灭。

2.2 实验仪器

（1） MSP430F5529LP+MSP430F5529 POCKET KIT 开发板一套；
（2） PC 机操作系统 Windows 7，CCSv7.3 集成开发环境。

2.3 实验准备

2.3.1 系统时钟

单片机各部件能有条不紊自动工作，实际上是在其系统时钟作用下，控制器指挥芯片内各个部件自动协调工作，使内部逻辑硬件产生各种操作所需的脉冲信号而实现的。

为适应系统和具体应用需求，单片机的系统时钟必须满足以下不同要求：

- 高频率，用于对系统硬件需求和外部事件快速反应；
- 低频率，用于降低电流消耗；
- 稳定的频率，以满足定时应用，如实时时钟 RTC；
- 低 Q 值振荡器，用于保证开始及停止操作最小时间延迟。

图 2-1 为 MSP430X5XX / 6XX 系列单片机时钟模块结构(可参考用户手册中 Figure 5-1. UCS Block Diagram)。可以看出，时钟模块有 5 个时钟输入源：

- XT1CLK，低频或高频时钟源：可以使用标准晶振，振荡器或者外部时钟源输入 4MHz～32MHz。XT1CLK 可以作为内部 FLL 模块的参考时钟。
- XT2CLK，高频时钟源：可以使用标准晶振，振荡器或者外部时钟源输入 4MHz～32MHz。
- VLOCLK，低功耗低频内部时钟源：典型值为 10KHZ；
- REFOCLK，低频修整内部参考时钟源：典型值为 32768Hz，作为 FLL 基准时钟源；
- DCOCLK，片内数字控制时钟源：通过 FLL 模块来稳定。

基础时钟模块可提供 3 种时钟信号：

- ACLK，辅助时钟：ACLK 可由软件选择来自 XT1CLK、REFOCLK、VLOCLK、DCOCLK、DCOCLKDIV、XT2CLK（由具体器件决定）这几个时钟源之一。然后经 1、2、4、8、16、32 分频得到。ACLK 可由软件选作各个外设模块的时钟信号，一般用于低速外设模块。

- MCLK，系统主时钟： MCLK 可由软件选择来自上述 5 种时钟源，同样可经过分频得到。MCLK 主要用于 CPU 和系统。
- SMCLK，子系统时钟：可由软件选择来自上述 5 种时钟源，同样可经过分频得到。SMCLK 可由软件选作各个外设模块的时钟信号，主要用于高速外设模块。

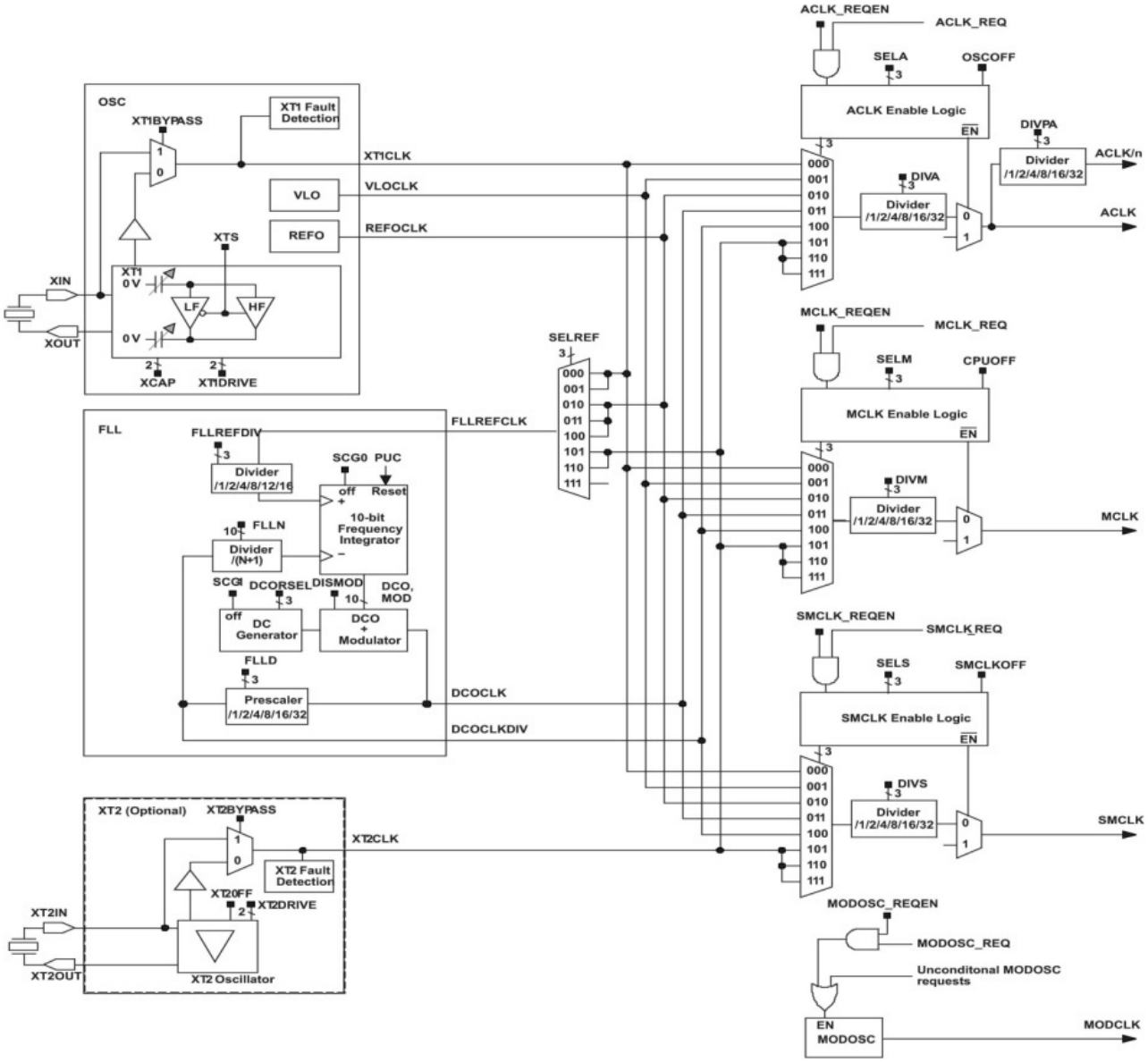


图 2-1 MSP430X5XX / 6XX 系列单片机时钟模块结构

MCLK、SMCLK 和 ACLK 三者关系。需要用 MCLK 的时候不多，一般情况都处于休眠状态，以节约功耗。能只用 SMCLK 解决的问题，就别动用 MCLK，待 SMCLK 完成自己的任务后，再请 MCLK 出马。当没有实际任务的时候，MCLK 和 SMCLK 都可以休眠，但是要 ACLK 工作，检测到任务后唤醒 MCLK 和 SMCLK。

MSP430F5XX/6XX 时钟模块寄存器，可翻看用户手册中 UCS Module Registers 章节，有详细介绍。在此列出几个寄存器用法示例。

（1） UCSCTL4 标准时钟系统控制寄存器

该寄存器各位的定义如下：

15	14	13	12	11	10	9	8
Reserved					SELA		
r0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
Reserved	SELS			Reserved	SELM		
r0	rw-1	rw-0	rw-0	r0	rw-1	rw-0	rw-0

- SELA：Bits10~8，选择 ACLK 的时钟源。
  - 000 XT1CLK；
  - 001 VLOCLK；
  - 010 REFOCLK；
  - 011 DCOCLK；
  - 100 DCOCLKDIV；
  - 101 XT2CLK，如 XT2CLK 不可用，默认为 DCOCLKDIV；
  - 110 保留，默认为 XT2CLK（如果可用），否则默认为 DCOCLKDIV；
  - 111 保留，默认为 XT2CLK（如果可用），否则默认为 DCOCLKDIV。
- SELS：Bits6~4，选择 SMCLK 的时钟源。详细信息请查阅用户手册。
- SELM：Bits2~0，选择 MCLK 的时钟源。详细信息请查阅用户手册。

（2） UCSCTL6 标准时钟系统控制寄存器

该寄存器各位的定义如下：

15	14	13	12	11	10	9	8
XT2DRIVE		Reserved	XT2BYPASS	Reserved			XT2OFF
rw-1	rw-1	r0	rw-0	r0	r0	r0	rw-1
7	6	5	4	3	2	1	0
XT1DRIVE		XTS	XT1BYPASS	XCAP		SMCLKOFF	XT1OFF
rw-1	rw-1	rw-0	rw-0	rw-1	rw-1	rw-0	rw-1

- XT2DRIVE：Bits15~14，XT2 的起振电流可以调节到合适值。详细信息请查阅用户手册。
- XT2BYPASS：Bit12，XT2 旁路模式选择。详细信息请查阅用户手册。
- XT2OFF：Bit8，关闭 XT2 晶振。
  - 0 假如 XT2 已经通过端口选择，并且非旁路模式，那么 XT2 被打开；
  - 1 假如 XT2 没有被用作 ACLK、MCLK 以及 SMCLK 的时钟源，或者没有用作 FLL 的校准源，XT2 关闭。
- XT1DRIVE：Bits7~6，XT1 的起振电流可以调节到合适值。详细信息请查阅用户手册。
- XTS：Bit5，XT1 模式选择。
  - 0 低频模式，XCAP 定义 XIN 和 XOUT 两个引脚的电容；
  - 1 高频模式。该位无效。
- XT1BYPASS：Bit4，XT1 旁路模式选择。详细信息请查阅用户手册。
- XCAP：Bits3~2，低频晶振电容选择。详见数据手册相关章节。
- SMCLKOFF：Bit1，SMCLK 关闭该位用来关闭 SMCLK 信号。
  - 0 SMCLK 开启；
  - 1 SMCLK 关闭。
- XT1OFF：Bit0，关闭 XT1 晶振。
  - 0 假如 XT1 已经通过端口选择，并且非旁路模式，那么 XT1 被打开；

1　假如 XT1 没有被用作 ACLK、MCLK 以及 SMCLK 的时钟源，或者没有用作 FLL 的校准源，XT1 关闭。

（3） UCSCTL7 标准时钟系统控制寄存器

该寄存器各位的定义如下：

15	14	13	12	11	10	9	8
Reserved		Reserved		Reserved		Reserved	
r0	r0	rw-0	rw-(0)	rw-(1)	rw-(1)	r-1	r-1
7	6	5	4	3	2	1	0
Reserved			Reserved	XT2OFFG(1)	XT1HFOFFG(1)	XT1LFOFFG	DCOFFG
r0	r0	r0	rw-(0)	rw-(0)	rw-(0)	rw-(1)	rw-(1)

- XT2OFFG：Bit3，XT2 晶振失效标志位。如果该位置位，那么 OFIFG 也置位。只要 XT2 失效条件存在，XT2OFFG 标志位就会置位。XT2OFFG 可以通过软件清零。
  0　最近一次复位之后没有失效条件产生；
  1　XT2 失效，最近一次复位之后产生失效条件。
- XT1HFOFFG：Bit2，XT1 晶振失效标志位(高频模式)。如果该位置位，那么 OFIFG 也置位。只要 XT1 失效条件存在，XT1HFOFFG 标志位就会置位。XT1HFOFFG 可以通过软件清零。
  0　最近一次复位之后没有失效条件产生；
  1　XT1 失效，最近一次复位之后产生失效条件。
- XT1LFOFFG：Bit1，XT1 晶振失效标志位(低频模式)。如果该位置位，那么 OFIFG 也置位。只要 XT1 失效条件存在，XT1LFOFFG 标志位就会置位。XT1LFOFFG 可以通过软件清零。
  0　最近一次复位之后没有失效条件产生；
  1　XT1 失效(LF)，最近一次复位之后产生 XT1(LF)失效条件。
- DCOFFG：Bit0，DCO 失效标志位。如果该位置位，那么 OFIFG 也置位。如果 DCO={0}或者 DCO={3}，DCOFFG 标志位就会置位。DCOFFG 可以通过软件清零。
  0　最近一次复位之后没有失效条件产生；
  1　DCO 失效，最近一次复位之后产生 DCO 失效条件。

2.3.2 定时器

定时器是 MCU 中重要的功能单元之一，有着多种用法，能够实现多种功能。下面主要介绍 MSP430F5529 中的看门狗定时器（WDT）、定时器 A（TIMER_A）。

WDT（Watch Dog Timer）俗称看门狗，是单片机非常重要的一个片内外设。什么是看门狗呢？看门狗实际就是一个定时器，只不过在定时到达时，可以复位单片机。这个功能对于实际工程应用中的产品非常有用。在很多应用中，单片机要经年累月的连续工作，如果期间单片机由于各种意外死机（俗称跑飞），则单片机就经年累月的不工作了。有了看门狗，就可以避免这种意外的发生。单片机都是循环工作的，比如完成整个循环所需时间最长不超过 1 秒，则可以把看门狗定时器的定时值设为 1.2 秒，在主循环中加入看门狗定时值清零的代码（俗称喂狗）。这样一来，假如程序运行正常，则总会在看门狗定时器到点前“喂狗”，从而避免单片机复位。如果程序死机，则不会及时“喂狗”，单片机复位。 复位后看门狗依然默认开启，继续守护程序的正常运行。

简言之，看门狗定时器（WDT）实际上是一个计数器，初始计数值为 0，处理器上电复

位(PUC)后看门狗开始计数。如果程序运行正常，过一段时间 CPU 应发出指令让看门狗复位，重新开始计数；如果看门狗溢出就认为程序没有正常工作，产生看门狗中断。

综上所述，看门狗的原理就 8 个字“定时喂狗，狗饿复位”。

复位后单片机状态为：

（1） I/O 引脚切换成输入模式。

（2） I/O 标志位清除。

（3） 其它外围模块及寄存器实现初始化。

（4） 状态寄存器复位。

（5） CPU 从内存的 0FFFE 地址开始执行代码。

MSP430X5XX / 6XX 系列单片机的看门狗定时器原理，如下图所示，

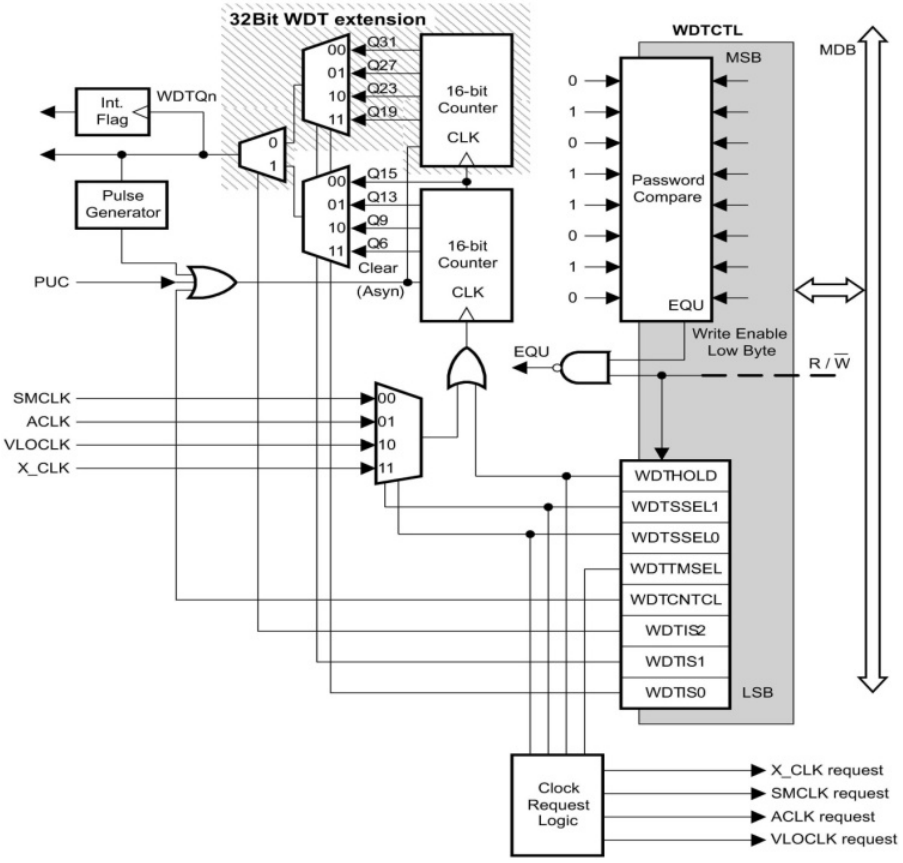


图 2-2 看门狗定时器原理

WDT 由控制寄存器 WDTCTL 控制其计数器不可由程序访问。控制寄存器 WDTCTL 为 16 位寄存器，其高八位用作口令，低八位用作对 WDT 的实际控制，下面为控制寄存器各域的说明：

- WDTPW：需要注意每次操作 WDTCTL 时必须写入本段，即对 WDTCTL 的操作均应当为 WDTCTL= WDTPW+....这种格式，否则将触发器件复位；
- WDTHOLD：看门狗使能控制，向该位写入 1 时关闭看门狗；
- WDTTMSEL：模式选择，‘0’ 看门狗模式，‘1’定时器模式；
- WDTCNCTL：计数器清除控制,当被置'1'后会自动复位为'0'，'0' 无动作，'1'计数器清零；
- WDTSSEL：选择 WDTCN 的时钟源，'0' 选择 SMCLK，'1'选择 ACLK。
- WDTISx： 看门狗的定时间隙选择。

用户可以通过 WDTCTL 寄存器中的 WDTTMSEL 和 WDTHOLD 控制位设置 WDT 工作在看门狗模式、定时器模式和低功耗模式。

◆ 看门狗模式

PUC 后，WDT 进入默认状态。如果系统不用看门狗功能，应该在程序开始处禁止看门狗功能。在看门狗模式下，如果计数器超过了定时时间，就会产生复位和激活系统上电清除信号。用户软件一般都需要进行如下操作：

（1）进行 WDT 的初始化：设置合适的时间。

（2）周期性地对 WDTCNT 清零：防止 WDT 溢出。

◆ 定时器模式

WDTTMSEL 设置为 1 时，WDT 工作在定时器模式。 在定时器模式下，定时间隔到以后，WDTIFG 标志位置 1。

◆ 低功耗模式

当不需要看门狗定时器时，可使用 WDTHOLD 位来停止看门狗计数器 WDTCNT，以降低功耗。

MSP430X5XX / 6XX 系列单片机的 Timer_A 共有 4 种计数模式，如表 2-1 所示：

表 2-1 MSP430X5XX / 6XX 系列单片机的 Timer_A 计数模式

MCx	模式	说明
00	停止模式	定时器停止
01	增计数模式	定时器重复从 0 计数到 TAxCCR0
10	连续计数模式	定时器器重复从 0 计数到 0FFFFh
11	增/减计数模式	定时器重复从 0 增计数到 TAxCCR0 再减计数到 0

定时功能模块是 MSP430 应用系统中经常用到的重要部分，可用来实现定时控制、延迟、频率测量、脉宽测量和信号产生、信号检测等等。一般来说，MSP430 所需的定时信号可以用软件和硬件两种方法来获得。MSP430 系列有丰富定时器资源：看门狗定时器（WDT），定时器 A（Timer_A），定时器 B（Timer_B）和定时器 D（Timer_D）等。MSP430 系列定时器部件功能，如表 2-2 所示.。

定时器 A 由一个 16 位定时器和多路捕获/比较通道组成。MSP430F5529 中包含有 3 个定时器 A 的子模块和 7 个捕捉/比较模块。定时器 A 支持多重捕获/比较，PWM 输出和内部定时。定时器还有扩展中断功能，中断可以由定时器溢出产生或由捕获/比较寄存器产生。定时器 A 的特性包括：

- 四种运行模式的异步 16 位定时/计数器
- 可选择配置的时钟源
- 可配置的 PWM 输出
- 异步输入和输出锁存
- 对所有 TA 中断快速响应的中断向量寄存器

表 2-2 MSP430 系列定时器部件功能

定时器	功能
看门狗定时器	基本定时、当程序发生错误时执行一个受控的系统重启动
基本定时器	基本定时、支持软件和各种外围模块工作在低频率、低功耗条件下
定时器A	基本定时、支持同时进行的多种时序控制、多个捕获/比较功能和多种输出波形（PWM），可以以硬件方式支持串行通信。
定时器B	基本定时、功能基本同定时器A,但比定时器A灵活，功能更强大
定时器D	基本定时、功能基本同定时器A,但比定时器A灵活，功能更强大

Timer_A 主要有 TAxCTL，TAxR，TAxCTLn，TAxCCRn，TAxIV、TAxEX0 几个寄存器（其中 x 指定时器的实例号，对 MSP430F5529 可以是 0~2，即 TA0~TA2；n 指捕捉/比较模块号）。其中最主要的是 TAxCTL 寄存器，用于设定 Timer_A 的输入时钟信号源、工作模式，复位、中断使能等。定时器 A 大致可分计数器和若干个捕捉比较模块。计数器在工作时由选定的时钟信号驱动计数，并且在计数归零时触发对应的定时器的 TAIFG；计数器配合几个比较/捕获寄存器便可实现定时、PWM、脉冲测量等功能。

TIMER_A 有 3 种定时/计数方式，方式选择由 TAxCTL 寄存器中的 MC 位控制。

增计数模式（01）：设置 MCx=01，主定时器将工作在增计数模式下。与连续计数不同的是， CCR0 模块可以提前将 TAR 寄存器清零。如图所示，当 TAR 的值与 TACCR0 预设值相等时，TAR 被强迫清零。时钟表盘只能在灰色区域活动。

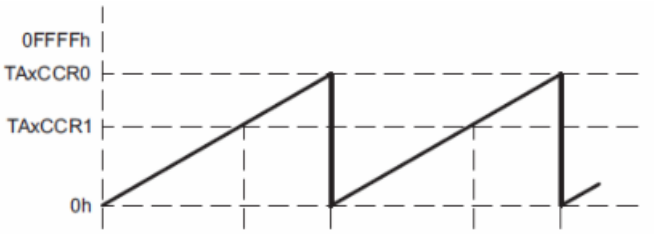


图 2-3 增计数模式

连续计数模式（10）：设置 MCx=10，主定时器将工作在连续计数模式下。TAR 寄存器最大值 65535，计满则清零。如图所示，时钟的周期仅由时钟源的频率决定，频率越高则越快计数至 65535，周期越短。

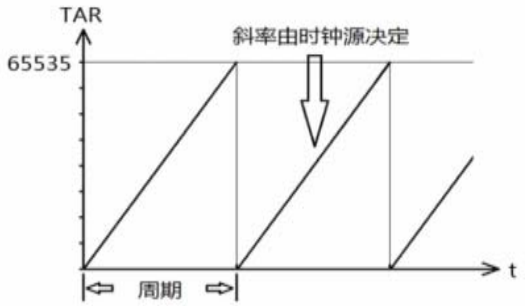


图 2-4 连续计数模式

增减计数模式（11）：TAxR 从 0 增加到 TAxCCR0 然后再从 TAxCCR0 减到 0。

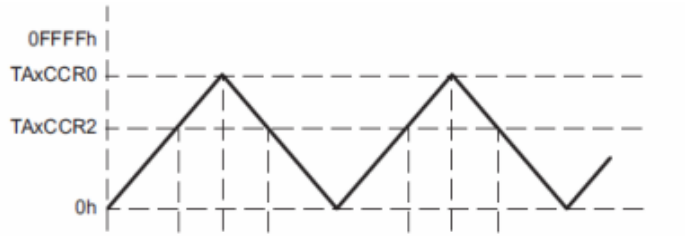


图 2-5 增减计数模式

Timer_A 有多个相同的捕获/比较模块，为实时处理提供灵活的手段，每个模块都可用于捕获事件发生的时间或产生定时间隔。通过 TACCTLx 中的 CAP 位选择模式，该模块既可用于捕获模式，也可用于比较模式。当发生捕获事件或定时时间到都将引起中断。捕获/比较模块的结构，如下图所示。

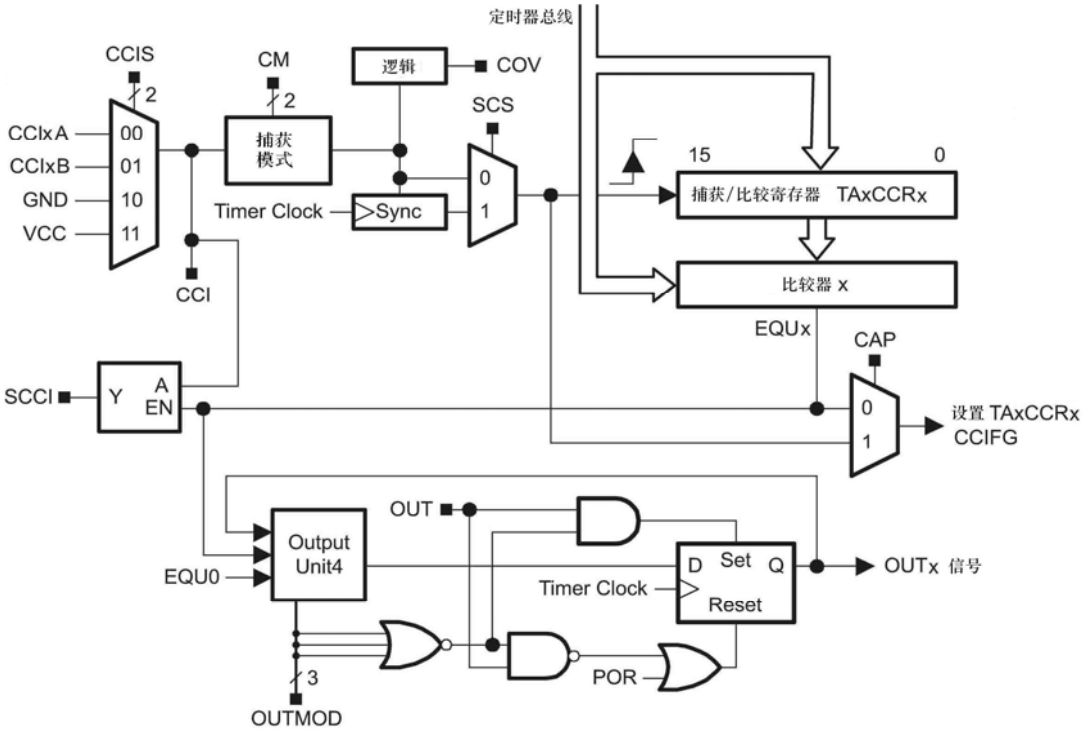


图 2-6 捕获/比较模块的逻辑结构

（1）捕获模式

当 TACCTLx 中的 CAP = 1，该模块工作在捕获模式。每个捕获/比较寄存器可以用来记录时间事件，例如：

▲ 测量软件程序所用时间

▲ 测量硬件事件之间的时间

▲ 测量系统频率

用 CM1 和 CM0 位选择捕获条件，可以选择禁止捕获、上升沿捕获、下降沿捕获或者上升沿下降沿都捕获。当捕获完成后，定时器的值被复制到 TAxCCRn 寄存器，并且中断标志 CCIFG 置位。如果总的中断允许位 GIE 允许，相应的中断允许位 CCIE 也允许，则将产生中断请求。如下图所示：

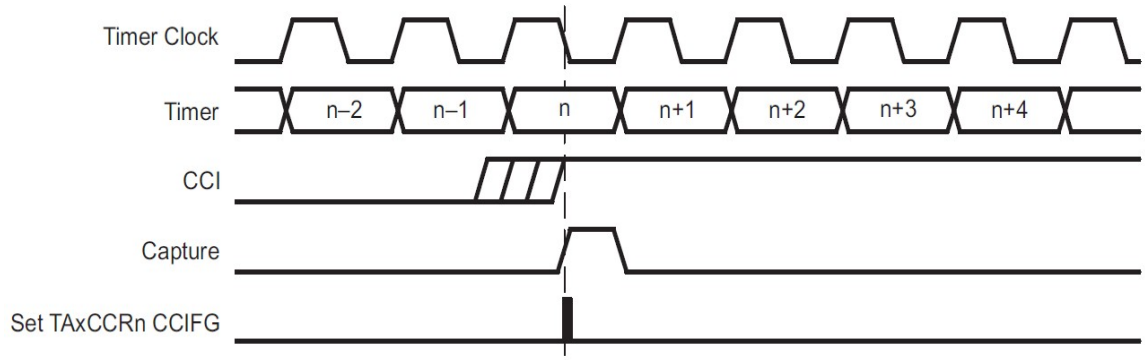


图 2-7 捕获模式的信号

（2）比较模式

当 TACCTLx 中的 CAP = 0，该模块工作在比较模式。比较方式主要用于为软件或应用硬件产生定时，还可为 D/A 转换功能或者马达控制等各种用途产生脉宽调制（PWM）输出信号。在计数器 TAxR 计数到 TAxCCRn（n 代表具体的捕获比较寄存器）的值时，中断标志 CCIFG 置位，内部信号 EQUx=1，EQUx 根据输出模式影响输出，输入信号 CCI 被锁存在 SCCI。

（3）输出单元

每个捕获/比较模块都包含一个输出单元，用于产生输出信号。每个输出单元有 8 种工作模式，可产生基于 EQUx 的多种信号。除模式 0 外，其他模式的输出都在定时器时钟上升沿时发生变化。输出模式 2，3，6，7 不适合输出单元 0，因为 EQUx=EQU0。输出单元在输出控制位 OUTMODx 的控制下，有 8 种输出模式输出信号。这些模式与 TAxR、TACCTLx、TAxCCR0 的值有关，如下表所示。

表 2-3 MSP430X5XX / 6XX 系列单片机的 Timer_A 计数模式

OUTMODx	模式	说明
000	输出模式0：输出	输出信号取决与寄存器 TACCTLx 中的 OUT位。当 OUT位更新时，输出信号立即更新。
001	输出模式1：置位	输出信号在TAxR等于TAxCCRn时置位，并保持置位到定时器复位或选择另一种输出模式为止。
010	输出模式2：翻转/复位	输出在TAxR的值等于TAxCCRn时翻转，当TAxR的值等于TAxCCR0时复位。
011	输出模式3：置位/复位	输出在TAxR的值等于TAxCCRn时置位，当TAxR的值等于TAxCCR0时复位。
100	输出模式4：翻转	输出电平在TAxR的值等于TAxCCRn时翻转，输出周期是定时器周期的2倍。
101	输出模式5：复位	输出在TAxR的值等于TAxCCRn时复位，并保持低电平直到选择另一种输出模式。
110	输出模式6：翻转/置位	输出电平在TAxR的值等于TAxCCRn时翻转，当TAxR值等于TAxCCR0时置位。
111	输出模式7：复位/置位	输出电平在TAxR的值等于TAxCCRn时复位，当TAxR的值等于TAxCCR0时置位。

定时器在增计数模式的输出实例如下图所示，在增计数模式下，当计数器 TAxR 增加到 TAxCCRx 或从 TAxCCR0 计数到 0 时，OUTn 信号按选择的输出模式发生变化。

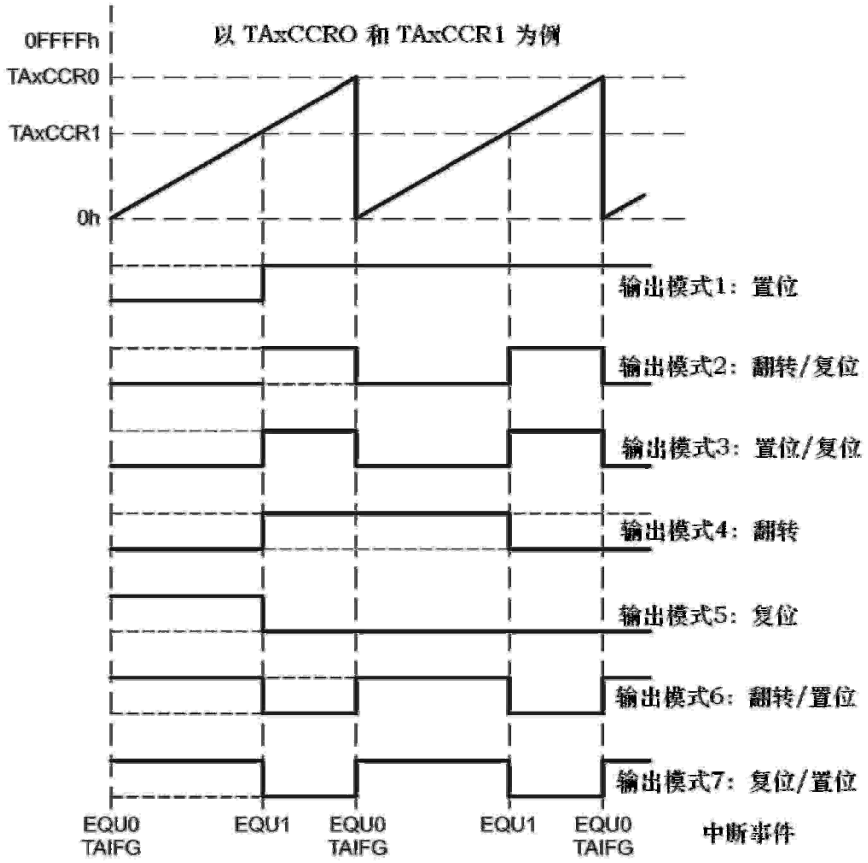


图 2-8 定时器在增计数模式的输出模式

PWM 信号是一种具有固定周期不定占空比的数字信号，如下图所示：

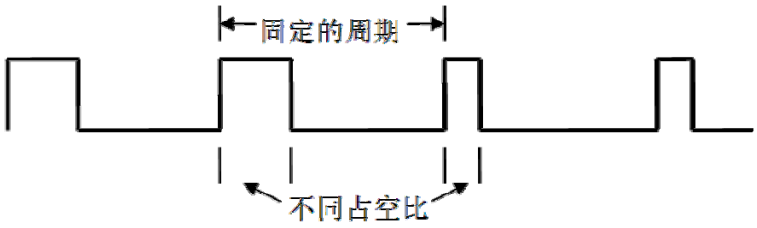


图 2-9 定时器在增计数模式的输出模式

如果 Timer_A 定时器的计数器工作在增计数方式，输出采用输出模式 7（复位/置位模式），则可利用寄存器 TAxCCR0 控制 PWM 波形的周期，用某个寄存器 TAxCCRx 控制占空比。这样 Timer_A 就可以产生出任意占空比的 PWM 波形。如下图所示：

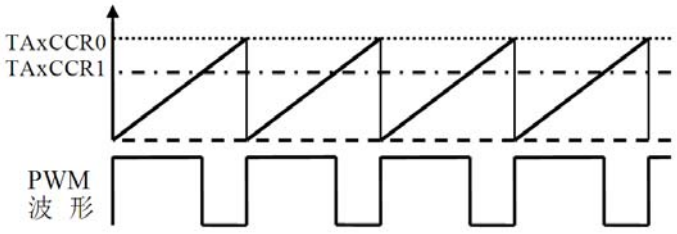


图 2-10 定时器在增计数模式的输出模式

可以随时间变化任意改变 PWM 信号的占空比，保持 CCR0 值（周期不变），改变 CCRx 值（改变占空比）。PWM 不需要修改占空比和时间时，CPU 在做完 Timer_A 初始化工作之后，

Timer_A 就能自动输出 PWM，而不需利用中断维持 PWM 输出，此时 CPU 就可以进入低功耗状态。

MSP430F5XX/6XX 定时器模块寄存器，可翻看用户手册中 WDTCTL Register、TAxCTL Register、TAxR Register、TAxCCTLn Register、TAxCCRn Register、TAxIV Register、TAxEX0 Register 等章节，有详细介绍。以看门狗定时器为例，如下图表所示。

Figure 16-2. WDTCTL Register

15	14	13	12	11	10	9	8
WDTPW							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

7	6	5	4	3	2	1	0
WDTHOLD	WDTSSEL		WDTTMSEL	WDTCNTCL	WDTIS		
rw-0	rw-0	rw-0	rw-0	r0(w)	rw-1	rw-0	rw-0

Table 16-2. WDTCTL Register Description

Bit	Field	Type	Reset	Description
15-8	WDTPW	RW	69h	Watchdog timer password. Always read as 069h. Must be written as 5Ah; if any other value is written, a PUC is generated.
7	WDTHOLD	RW	0h	Watchdog timer hold. This bit stops the watchdog timer. Setting WDTHOLD = 1 when the WDT is not in use conserves power. 0b = Watchdog timer is not stopped. 1b = Watchdog timer is stopped.
6-5	WDTSSEL	RW	0h	Watchdog timer clock source select 00b = SMCLK 01b = ACLK 10b = VLOCLK 11b = X_CLK; VLOCLK in devices that do not support X_CLK
4	WDTTMSEL	RW	0h	Watchdog timer mode select 0b = Watchdog mode 1b = Interval timer mode
3	WDTCNTCL	RW	0h	Watchdog timer counter clear. Setting WDTCNTCL = 1 clears the count value to 0000h. WDTCNTCL is automatically reset. 0b = No action 1b = WDTCNT = 0000h
2-0	WDTIS	RW	4h	Watchdog timer interval select. These bits select the watchdog timer interval to set the WDTIFG flag and/or generate a PUC. 000b = Watchdog clock source /(2^31) (18h:12m:16s at 32.768 kHz) 001b = Watchdog clock source /(2^27) (01h:08m:16s at 32.768 kHz) 010b = Watchdog clock source /(2^23) (00h:04m:16s at 32.768 kHz) 011b = Watchdog clock source /(2^19) (00h:00m:16s at 32.768 kHz) 100b = Watchdog clock source /(2^15) (1 s at 32.768 kHz) 101b = Watchdog clock source /(2^13) (250 ms at 32.768 kHz) 110b = Watchdog clock source /(2^9) (15.625 ms at 32.768 kHz) 111b = Watchdog clock source /(2^6) (1.95 ms at 32.768 kHz)

2.4 实验内容

2.4.1 系统时钟实验

设 ACLK = XT1 = 32768Hz，ACLK 通过 P1.0 输出。编程实现 ACLK 的时钟源为 XT1 外接晶振，并且用示波器观察到 P1.0 输出信号的频率，验证信号是否为晶振频率。

实验要求：

（1）实现现象：引脚 P1.0 输出一个频率为 32768Hz 的方波，示波器可观察到；

（2）请自行查看硬件接线原理图，找到 XT1 晶振 XT1IN 和 XT1OUT 的 GPIO 复用引脚，选取熟悉的或者喜欢的任意一种编程方法实现，配置寄存器法，直接调用头文件 #include <msp430f5529.h>法，使用固件库 driverlib 的时钟库函数模块。

实验步骤：

参照第 1 节相关实验步骤，新建工程文件，编写代码，编译调试后，下载程序看实验现象。

2.4.2 看门狗定时功能实验 1

使用看门狗的定时功能产生一个周期为 64ms 的方波。选取某一个 IO 端口，使某个引脚输出一个周期为 64ms 的方波，并且用示波器可以清晰的观察到。

实验要求：

（1）实现现象：引脚 P1.0(或者合理的选择其他引脚均可) 输出一个周期为 64ms 的方波，示波器可观察到；

（2）选取熟悉的或者喜欢的任意一种编程方法实现，配置寄存器法，直接调用头文件 #include <msp430f5529.h>法，使用固件库 driverlib 的看门狗库函数模块。

实验步骤：

参照第 1 节相关实验步骤，新建工程文件，编写代码，编译调试后，下载程序看实验现象。

2.4.3 看门狗定时器实验 2

本实验使用了 MSP430F5529 中的看门狗定时器模块，运用看门狗定时中断功能实现了对 LED 的定时亮灭控制。

编程思路：本实验的关键在于 WDT 的定时功能应用，首先得配置好看门狗的工作模式和时钟源，然后通过中断的方式实现定时地对 LED 的亮灭进行控制，当按键按下不停喂狗时，不会产生中断。

程序流程图如下，

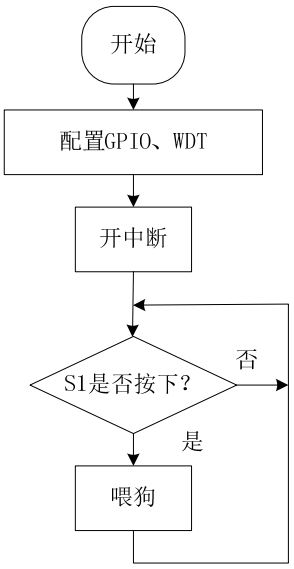


图 2-11 程序流程图

实验要求：

（1）实现现象：口袋板左下角 L1 指示灯以 1Hz 左右的频率闪烁（表示系统工作异常），看门狗定时器（WDT）产生了中断，当我们不停的按下 S1 按键时，发出一个“喂狗”的信号（实际是让 WDT 定时器复位，不产生中断，表示系统工作正常），结果就是 L1 指示灯的状态不再变化（长亮或长灭）；当我们不按下 S1 键或按下的频率比 WDT 定时器产生中断的频率低的时候，L1 指示灯就以 1Hz 左右的频率不停闪烁。

（2）选取熟悉的或者喜欢的任意一种编程方法实现，配置寄存器法，直接调用头文件#include <msp430f5529.h>法，使用固件库 driverlib 的看门狗函数模块。

实验步骤：

参照第 1 节相关实验步骤，新建工程文件，编写代码，编译调试后，下载程序实现要求的实验现象。

2.4.4 定时器 Timer_A 实验 1

使用定时器定时功能产生两个固定占空比 PWM 波。设 ACLK = TACLK = LFXT1 = 32768Hz， MCLK = SMCLK = DCOCLK = 32×ACLK = 1.048576MHz，利用 Timer_A 输出周期为 512 /32768 = 15.625ms、占空比分别为 75%和 25%的 PWM 矩形波。

实验要求：

（1）实现现象：P 1.2 端口引脚输出占空比 75% PWM，P1.3 端口引脚输出占空比 25% PWM，周期均为 15.625ms，请用示波器验证波形。

（2）请自行查找硬件接线原理图，找出相应端口和定时器对应的外设功能关系，选取熟悉的或者喜欢的任意一种编程方法实现，配置寄存器法，直接调用头文件#include <msp430f5529.h>法，使用固件库 driverlib 的时钟库函数模块。

实验步骤：

参照第 1 节相关实验步骤，新建工程文件，编写代码，编译调试后，下载程序看实验现象。

2.4.5 定时器 Timer_A 中断实验

本实验使用了 MSP430F5529 中的 Timer_A 定时器模块，运用定时中断功能实现对 LED 指示灯的定时亮灭控制，并且控制蜂鸣器的发声频率。

编程思路：实验开始必须先关闭看门狗定时器，然后配置好连接蜂鸣器和 LED 灯的 GPIO 寄存器，重点设置 TA0CTL 寄存器，使 Timer_A 工作在增计数模式，并选择 SMCLK 为其时钟源。使能比较器中断后，设定比较值为 50000（请分析此数值代表的意义），最后打开总中断。

实验步骤：

（1）根据原理图，把蜂鸣器相关跳线接上（板件默认是跳上的，如未跳上请做此步骤）。

（2）实现控制代码的设计，画出流程图；

（3）完成编码，并将代码烧录到开发板中；

（4）观察 LED 的状态变化，验证结果；

（5）听蜂鸣器的声音，验证效果。

要求实现的实验现象：

口袋板上 S1 按键按下后 L1 指示灯每隔 50ms 闪烁一次，同时蜂鸣器发出响声；S2 按

键按下后指示灯不再闪烁，蜂鸣器停止发声。

程序流程图如下图 2-12 所示。

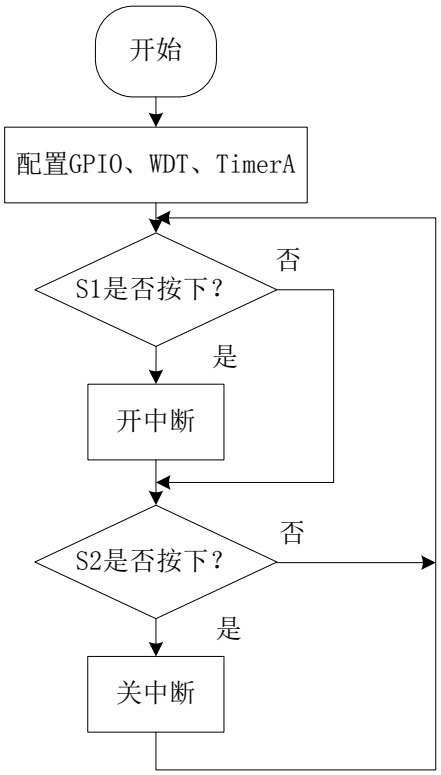


图 2-12 程序流程图

2.4.6 本章作业

（1） 参考 2.4.1 节系统时钟实验要求，自己查找硬件接线原理图，找到 XT2 晶振 XT2IN 和 XT2OUT 的 GPIO 复用引脚，同时找到 SMCLK 时钟输出的 GPIO 复用引脚。硬件接线：SMCLK 外部晶振 XT2 为 4MHz，SMCLK 通过 P2.2 输出。

要求：编程，使 SMCLK 的源为外部晶振的 4MHz，用示波器查看 P2.2 引脚输出时钟信号，是否为硬件接线中晶振的频率 4MHz，如果不是，那么频率是多少？分析为什么会出现此现象？

（2） 参考 2.4.3 节看门狗定时器实验要求，添加一个功能，实现如下现象：

- 蜂鸣器安装在 P3.6 引脚后，编程实现蜂鸣器发声，发声频率为 1Hz，周期为 1s。人耳听到的声音是类似钟表的滴答声。
- 按键 S1 按下后，观察实验现象，蜂鸣器发声情况。
- 编程实现改变蜂鸣器发声频率为 4Hz，周期为 250ms。

提示：

P8.1 引脚在定时中断后翻转，LED1 灯闪烁，那么 P3.6 引脚跟 P8.1 引脚功能一样，只不过驱动的是蜂鸣器。

（3）参考 2.4.5 节定时器中断实验，通过定时器设定改变蜂鸣器发声的音调（频率），高声调一个，低声调一个，做对比，分析蜂鸣器发生频率和定时器中断频率之间的关系。

2.5 思考与分析

（1） 主函数中，没有调用中断子程序，中断子程序为什么能被执行？何时被执行?
（2） Timer_A 定时器都有哪些工作模式，如何配置？
（3） 如何通过定时器设定蜂鸣器发声的音调？

实验三　MSP430F5529LP 的 ADC 实验

3.1 实验目的

（1） 了解比较器的工作原理，熟悉比较器 B 的使用；
（2） 了解 ADC 的工作原理，熟悉 ADC12 的 4 种转换模式；
（3） 掌握 MSP430F5529LP 的 ADC12 基本使用方法；
（4） 结合电位器与 ADC12 模块实现对 LED 灯的控制。

3.2 实验仪器

（1） MSP430F5529LP+MSP430F5529 POCKET KIT 开发板一套；
（2） PC 机操作系统 Windows 7，CCSv7.3 集成开发环境。

3.3 实验原理

3.3.1 比较器 B

比较器被用于判断输入信号电位之间的相对大小，它至少有两个输入端及一个输出端。比较器 B 是为精确的比较测量而设计的，如电池电压监测、产生外部模拟信号、测量电流、电容和电阻，结合其他模块还可实现精确的 A/D 模数转换功能。比较器 B 是工业仪表、手持式仪表等产品设计中的理想选择。5 系列之前的 MSP430 单片机仅有比较器 A，5/6 系列 430 单片机升级为比较器 B。

Comp_B 是一个模拟电压比较器，涵盖了多达 16 通道的通用比较器功能。Comp_B 模块主要特性有：

- 正向反向终端输入多路选择器
- 通过软件选择比较器输出的 RC 滤波
- 可输出到 TA 的捕获输入
- 软件控制端口输入缓冲
- 具有中断能力
- 可选的参考电压发生器、电压磁滞发生器
- 参考电压输入可选择共用参考电压
- 超低功耗的比较模式
- 低功耗模式支持中断驱动测量系统

比较器 B 的结构，如下图所示：

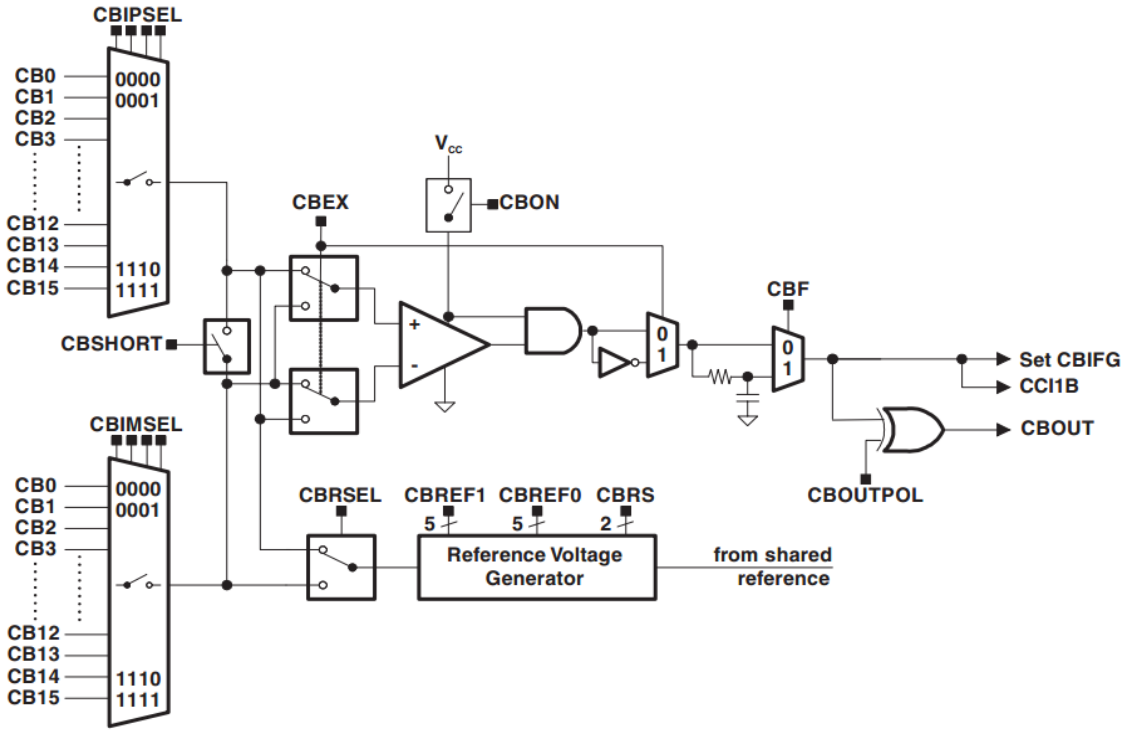


图 3-1 比较器 B 的结构

比较器 B 的主要功能：指出两个输入电压 CB0 和 CB1 的大小关系，然后设置输出信号 CBOUT 的值。如果 CB0>CB1 则：CBOUT＝1，否则 CBOUT＝0。比较器 B 的结构图所示，比较器位于模拟信号输入和滤波输出之间的部分。比较器对正和负输入终端的模拟信号进行比较。如果正端信号大于负端，则比较器输出 CBOUT 为高。其中与门负责将比较输出信号进行整形。可以通过 CBON 位来关闭或打开比较器。控制位 CBEX 选择正向或反向输出。

◆模拟输入端

参与比较的两个模拟信号通过正、负两个输入电压端：CB0 和 CB1 进入比较器 B，输入电流极小。这两个输入端可由用户软件设置，最终能够选择 6 种信号（CB0、CB1、0.5VCC、0.25VCC、三极管阀值电压和外部参考源），而且能够进行多种组合比较。硬件提供的比较组合如下：

- 两个外部输入比较
- 每个外部输入与 0.5VCC 或 0.25VCC 比较
- 每个外部输入与内部基准电压比较

◆模拟输入开关

- 通过 CBIPSELx 及 CBIMSELx 位，用于选择两个比较器输入终端与相应端口管脚之间连接还是断开。比较器的输入终端可以分别进行控制。
- 通过配置 CBIPSELx/CBIMSELx 位可以实现：
  - ▲ 将外部信号连接到比较器的正端或负端
  - ▲ 内部参考电压到相应输出端口管脚选择一个路径
  - ▲ 将外部电流源应用到比较器的正端或负端

▲ 内部多路选择器的两个端口到外部的映射

- CBEX 位控制输入多路选择器，改变比较器正端或负端输入信号的顺序。另外，当比较器终端顺序发生改变时，比较器输出信号也发生反转，这使用户可以检测或补偿比较器输入端的偏置电压。

◆输出电路

- 最终输出信号的上升沿或下降沿可以设置为具有中断能力。
- 如果不使用中断，可将输出信号送给内部其他模块，作为其他模块的一个输入信号；还可以由外部引脚引出。

3.3.2 AD 转换器

模拟量--数字量转换模块（ADC）与数字量--模拟量转换模块（DAC）是非常有用的功能模块，很多单片机中只有 ADC 模块、没有 DAC 模块，MSP430F5529 也一样，只有一路 ADC，但是通过片内的多路复用模块，扩展出了 12 个通道。这里我们先介绍 F5529 内部的 ADC 模块的原理及应用，与之对应的 DAC 因为没有集成在 F5529 内部，可扩展为片外设备。下面首先简要介绍一下模拟量与数字量的概念以及他们之间的关系。

模拟信号是指用连续变化的物理量表示的信息，其信号的幅度、频率或相位随时间作连续变化，如目前广播的声音信号，或图像信号等。当模拟信号采用连续变化的电磁波来表示时，电磁波本身既是信号载体，同时作为传输介质；而当模拟信号采用连续变化的信号电压来表示时，它一般通过传统的模拟信号传输线路（例如电话网、有线电视网）来传输。

数字信号指幅度的取值是离散的，幅值表示被限制在有限个数值之内。二进制码就是一种数字信号。它具有抗干扰能力强、无噪声积累、便于加密处理、便于存储、处理和交换等特点。

模拟信号与数字信号之间的相互转换：模拟信号一般通过 PCM 脉冲编码调制(Pulse Code Modulation)方法量化为数字信号，即让模拟信号的不同幅度分别对应不同的二进制值，例如采用 8 位编码可将模拟信号量化为 2^8=256 个量级，实用中常采取 24 位或 30 位编码；数字信号一般通过对载波进行移相(Phase Shift)的方法转换为模拟信号。计算机、计算机局域网与城域网中均使用二进制数字信号，目前在计算机广域网中实际传送的则既有二进制数字信号，也有由数字信号转换而得的模拟信号。但是更具应用发展前景的是数字信号。

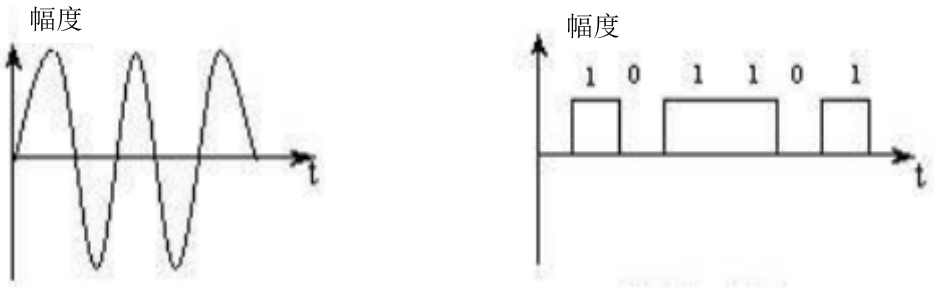


图 3-2 模拟信号（左）与数字信号（右）

AD 转换：AD 转换器根据其原理主要可以分为积分型、逐次逼近型和并行比较型。在获取相同的转换精度时，积分型 AD 转换器的电路最为简单、成本低、同时转换耗时最长，并行比较型则拥有最高的转换速度和最复杂的电路，逐次逼近型的性能介于另两种之间。MSP430 的 AD 模块为逐次逼近型，其转换器核心由一个比较器、一个 DA 转换器和逐次比较逻辑组

成。工作时转换器会从最高位到最低位依次生成试探电压的数字值，将其送入 DA 转换器生成试探电压信号，并将这一信号与输入信号进行比较来决定每一位的取值，对于 n 位的转换器，每次转换过程需要比较 n 次。

模数转换器的作用是把从传感器采集到的连续变化的模拟电压信号转换为单片机可识别的数字信号。在转换的过程中，数字输出代码与模拟输入电压之间的关系如图 5-2 所示。

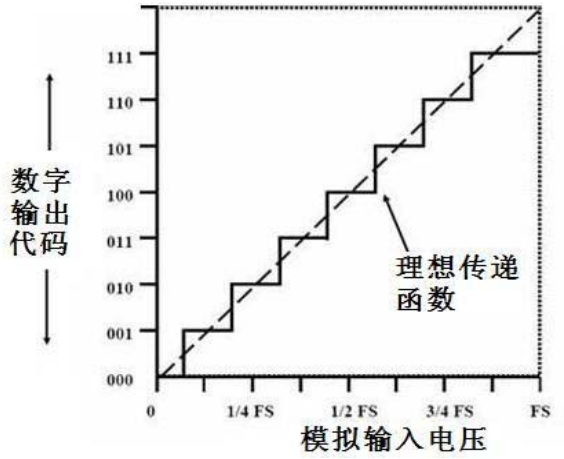


图 3-3 数字输出代码与模拟输入电压之间的关系

在进行模／数(即 A／D)转换时，通常按取样、保持、量化、编码四个步骤进行。

所谓取样，就是对模拟信号 Ui(t) 进行周期性抽取样值的过程。

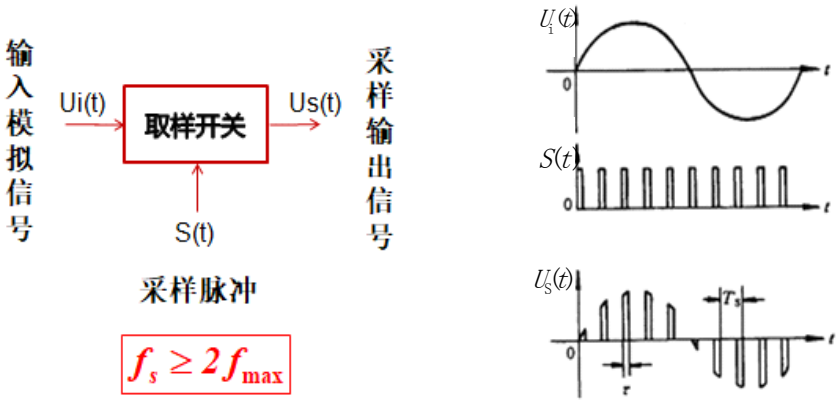


图 3-4 取样电路及图形

保持：模拟信号经采样后，得到一系列样值脉冲。采样脉冲宽度 τ 一般是很短暂的，在下一个采样脉冲到来之前，应暂时保持所取得的样值脉冲幅度，以便进行转换。因此，在取样电路之后须加保持电路。

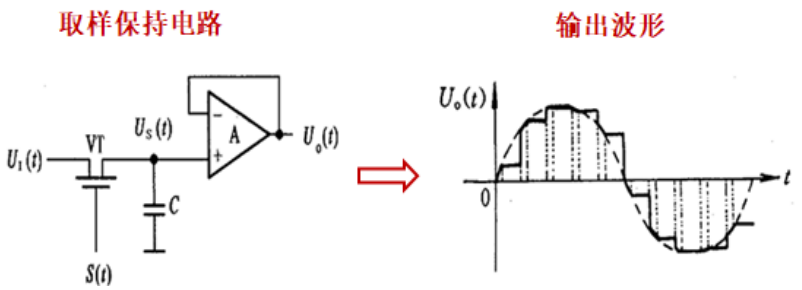


图 3-5 取样保持电路及输出图形

上图电路中，

①在采样脉冲 S(t)到来的时间 τ 内，VT 导通，UI(t)向电容 C 充电，假定充电时间常数远小于 τ，则有：UO(t)＝US(t)＝UI(t)。－－采样

场效应管 VT 为采样门，电容 C 为保持电容，运算放大器为跟随器，起缓冲隔离作用。

②采样结束，VT 截止，而电容 C 上电压保持充电电压 UI(t)不变，直到下一个采样脉冲到来为止。－－保持

取样保持后所得**阶梯波**仍是一个可以连续取值的**模拟量**，但 n 位数字量只能表示 2^n 个数值。因此，用数字量来表示连续变化的模拟量时就有一个类似于四舍五入的近似问题。

量化：用数字信号的最低位 1 (LSB)所对应的模拟电压作为量化单位，用 Δ 表示，将样值电压变换为量化单位(Δ)电压整数倍的过程。

编码：量化后的离散量用相应的二进制码表示，称作编码。

数字量的计算方法：

ADC 内核一般要使用两个参考电压 VR+和 VR-，一般这两个电压可以是用户接入或者是使用内部参考电压。VR+是定义的转换最大值，VR-则是转换的最小值。

以 12 位分辨率为例。

- Vin> =VR+　　ADC12 输出满量程值 0x0FFF；
- Vin< =VR-　　ADC12 输出 0；
- VR-<Vin<VR+　ADC12 的转换结果满足如下公式：

$$NADC = 4095 * (Vin- VR-)/(VR+ - VR-)$$

分辨率 R：

- 可以转换成数字量的模拟电压量的最小值；
- 最低有效位（LSB），即分辨率单位：$R = \frac{1}{2^n}$
- 分辨率只是规定了数字量输出的位数，而不是 ADC 的性能。

转换精度：

- 表示模拟电压实际值与其对应数字量的相对误差；
- 可以表达为“真实度”。

转换时间：

- 指 ADC 模块完成一次模拟数字转换所需要的时间，转换时间越短越能适应输入信号的变化。转换时间与 ADC 模块的结构和位数有关。

MSP430F5529 的 ADC12_A 模块支持快速 12 位模数转换，该模块包含 12 位 ADC 核、采样选择控制、参考电流发生器。ADC12_A 特征如下：

- 大于 200 ksps 的最大转换速率；
- 无失码的 12 位单调转换器；
- 软件或定时器控制的可编程采样保持周期；
- 通过软件或定时器控制转换开始；
- 软件可选择的片上参考电压生成器（1.5 V，2.0 V，2.5 V）；
- 软件选择的内部或外部参考；
- 多达 12 个可单独配置的外部输入通道；
- 内部温度传感器的转换通道，AVCC，和外部参考；
- 可选择的转换时钟源；

- 单通道，重复单通道，序列（自动扫描），重复序列（重复自动扫描）转换模式；
- ADC 内核和参考电压可单独实现掉电；
- 快速解码的 18 位 ADC 中断的中断向量寄存器；
- 16 个转换结果存储寄存器。

ADC12_A 功能框图如下所示。

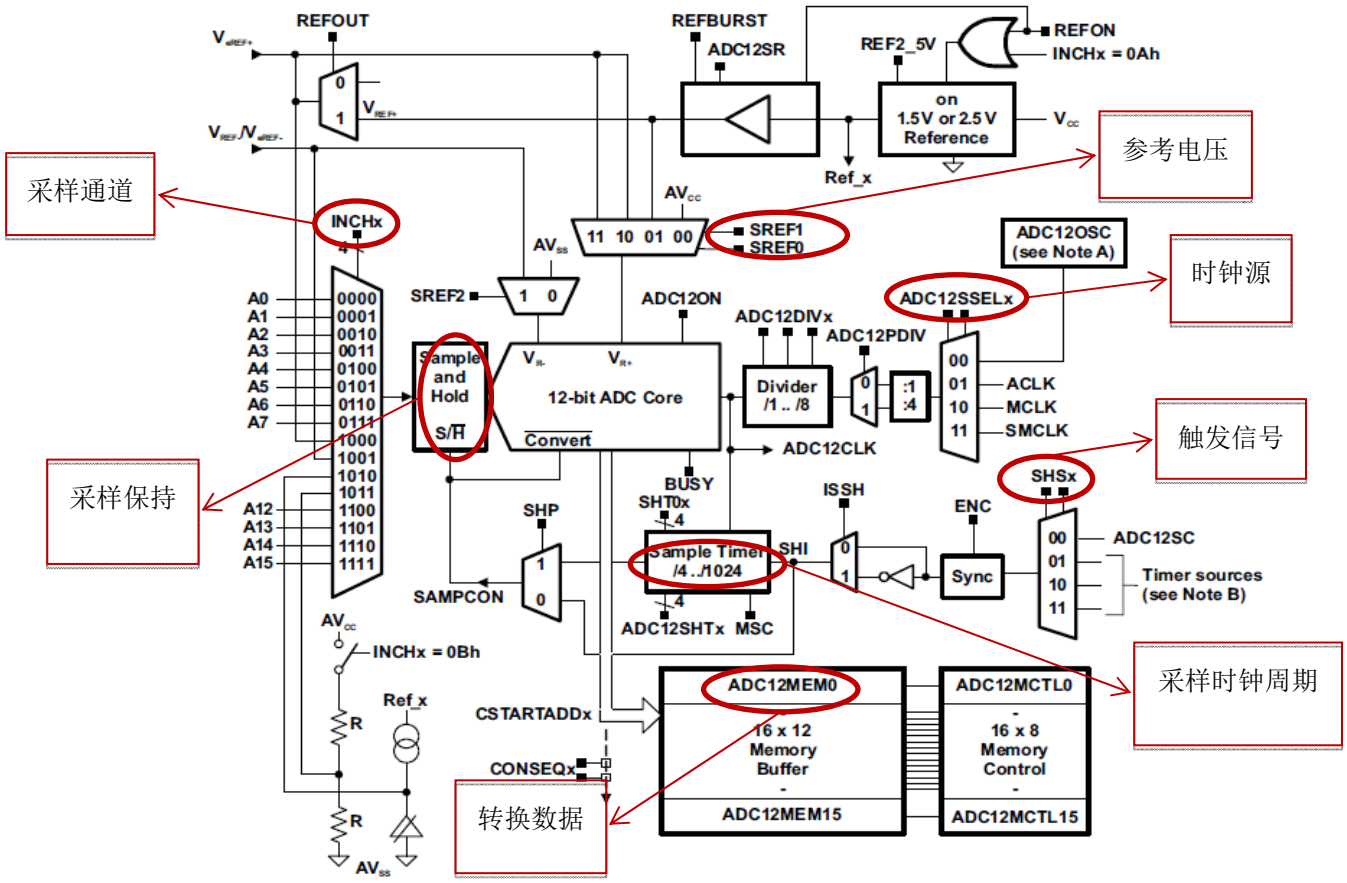


图 3-6　ADC12_A 功能框图

（1）12 位 ADC 核(使能位为 ADC12ON 位)

将模拟输入转化为 12 位数字表示，在 ADC12MEMx 寄存器中存储结果，转换结果的上限和下限: VR+ 、VR- 。

数码输出(N_{ADC}) 结果:

- 最大值: N_{ADC} = 0FFFh, 输入信号>=V_{R+};
- 零: N_{ADC} = 0000h, 输入信号<=V_{R-};

转换结果:

二进制形式:　　$N_{ADC} = 4096 \frac{V_{in} - V_{R-}}{V_{R+} - V_{R-}}$

（2）数模转换由 SHI 信号（采样输入信号）的上升沿启动

SHI (SHSx 位选择) 的来源可以是:

- 00　ADC12SC 位;
- 01　Timer_A 输出单元 1;
- 10　Timer_B 输出单元 0;
- 11　Timer_B 输出单元 1。

（3）采样时序方式

- ADC12SHP = 1: 脉冲模式
- ADC12SHP = 0: 扩展采样模式

（4）16 个 ADC12MEMx 转换存储寄存器

由相关的 ADC12MCTLx 控制寄存器配置存储转换结果。

- 非连续转换(单一或重复单一通道)：
  ADC12CSTARTADDx 定义转换的第一个和单个的 ADC12MCTLx.
- 连续转换(序列或重复序列通道)：
  序列从 ADC12MCTLx 寄存器中的值开始，该值由 ADC12CSTARTADDx 指定;
  指针自动递增到下一个 ADC12MCTLx，从而开启下一次转换;
  ADC12MCTL15 之后的转换是 ADC12MCTL0;
  序列保持运行直至 ADC12EOS 位信号出现,此信号是实际运行序列的最后一次转换;
  16 个 ADC12MCTLx 寄存器可以包含多个序列。

（5）ADC12_A 包含 18 个中断源

- ADC12IFG0-ADC12IFG15
  当 ADC12MEMx 存储寄存器加载转化结果时，其对应的 ADC12IFGx 位将会置位;
- ADC12OV, ADC12MEMx 溢出位
  在上一个转换结果未被读取之前，向任意一个 ADC12MEMx 写入新的转换结果时，ADC12OV 位将会置位;
- ADC12TOV, ADC12_A 转换时间溢出位
  当前转换未完成就请求另一个采样和转换时，ADC12TOV 位将会置位。

（6）中断向量寄存器 ADC12IV 用来确定是哪个 ADC12_A 中断源发出了中断请求。

（7）ADC12_A 寄存器

- ADC12CTL0，ADC12_A 控制寄存器 0
- ADC12CTL1，ADC12_A 控制寄存器 1
- ADC12CTL2，ADC12_A 控制寄存器 2
- 转换存储寄存器：ADC12MEM0～ADC12MEM15
- ADC12MCTLx 转换存储器控制寄存器 ADC12MCTL0～ADC12MCTL15
- ADC12IE， ADC12_A 中断使能寄存器
- ADC12IFG，ADC12_A 中断标志寄存器
- ADC12IVx 中断向量寄存器

具体寄存器用法请详细翻看用户手册。

3.4 实验内容

3.4.1 比较器实验

使用拨盘电位器，获得 3.3V 以下需要的电压，设计参考电压为 1.5V，与拨盘电位器获得的电压作比较，驱动 LED 灯亮灭。

实验要求：

（1）实现现象：拨动拨盘电位器当对应电压达到 1.5V 以上时，LED 灯（任选）点亮，未达到时 LED 灯熄灭；

（2）自行查看硬件接线原理图，选取熟悉的或者喜欢的任意一种编程方法实现，配置寄存器法，直接调用头文件#include <msp430f5529.h>法，使用固件库 driverlib 的比较器库函数模块。

实验步骤：

参照第 1 节相关实验步骤，新建工程文件，编写代码，编译调试后，下载程序看实验现象。

3.4.2 ADC 实验

ADC12 模块的主要寄存器有 ADC12CTL0（转换控制寄存器）、ADC12CTL1（终端控制寄存器）、ADC12IFG（中断标志寄存器）)、 ADC12IE（中断使能寄存器）、ADC12IV（中断向量寄存器）。

采样方式分为单通道单次采样、多通道单次采样、单通道循环采样、多通道循环采样四种方式，我们本次实验采取的是单通道循环采样方式。当 ADC12ON 为高电平时，ADC 转换器启动并等待转换开始的信号；此时当 ADC12ENC 位为 1 且 ADC12SC 出现上升沿时开始转换过程，并把每次转换的数据保存在 ADC12MEN0 中。在单通道循环采样方式下转换过程会循环进行，直到将 ADC12ENC 复位。

F5529 口袋板上配置的是双路拨盘电位器（R4），其他一路用来调节 ADC（P6.5）输入端的电压（0~3.3V）；另外一路用来控制 DAC 输出电压的大小，通过这一路就可以实现对耳机插座与扬声器端输出的音频信号幅值进行控制。这个实验是改变 ADC 端输入电压，然后依据电压高低分为几档通过 LED1~LED5 显示出来。

（1）实验要求

实现拨动拨码电位器，LED1~LED6 顺序点亮的现象。

（2）程序分析

编程思路：熟悉了 MSP430F5529 中的 ADC12 模块原理之后便可对其控制寄存器进行配置，设计采样模式，时刻得到电位器的输出端电压值。并通过其大小，设定范围从而来控制 LED 灯的亮灭。程序流程图如图 3-7 所示。

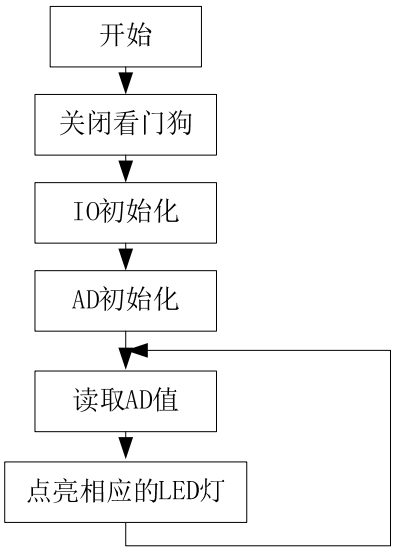


图 3-7 程序流程图

（3）实验步骤

① 根据编程思路设计结构与实现方法，按照流程图实现代码编写，并在编译器上进行编译改错，通过调节电位器查看 LED 灯的变化是否符合设计要求，根据伪代码，自行编写程序，编译、调试并下载程序到开发板。
② 运行程序，旋转拨盘电位器，查看 LED 指示灯的变化。

3.4.3 本章作业

（1） 自行学习触摸电容原理（下发电子文档供参考），调试触摸电容伪代码，实现电子文档中要求的实验现象。
（2） 完成 ADC 实验 PPT 中的思考题，并使用 ADC12MEM1 寄存器存储转换数据，再次实现 3.4.2 节中要求的实验现象。

3.5 思考与分析

（1）如何采集负电压信号？
（2）如何对正弦波或三角波进行模数转换实现计数功能？
（3）请尝试编写程序将采集的信号波形在电子纸屏幕上显示。

实验四 MSP430F5529 口袋板 SPI 工作模式

4.1 实验目的

（1） 了解 MSP430F5529 通用串行通信接口概念，了解电子墨水屏技术；
（2） 掌握 SPI 工作模式及编程方法。

4.2 实验仪器

（1） MSP430F5529LP+MSP430F5529 POCKET KIT 开发板一套；
（2） PC 机操作系统 Windows 7，CCSv7.3 集成开发环境。

4.3 实验准备

通用串行通信接口（USCI）模块是 MSP430F5529 芯片上的重要通信接口，很多外设模块使用这一接口。5529 中 USCI 接口有 2 组：USCI_Ax 与 USCI_Bx，工作模式主要有：UART、SPI、I^2C 等等，不同的 USCI 模块所具有的功能也不尽相同，其中 USCI_Ax 模块支持 UART 模式、SPI 模式，该模块还可用于 IrDA 通信的脉冲整形以及 LIN 通信的自动波特率检测等；USCI_Bx 模块支持的特性包括 I^2C 模式和 SPI 模式。本节介绍 SPI 模式。

4.3.1 通用串行通信接口

在同步模式下，USCI 通过 3 个或者 4 个引脚把 MSP430 连接到一个外部系统中，这些引脚分别是：UCxSMIO，UCxSOMI，UCxCLK 和 UCxSTE。同步位 UCSYNC 被置位且模式选择位 UCMODE 位选择 SPI 时 USCI 模块工作于 SPI 模式。

SPI 的特点如下，

- 具有 7 位或 8 位的数据位选择；
- 支持 3 线或 4 线 SPI；
- 支持主从模式；
- 独立的发送和接受寄存器；
- 分离的发送和接受缓冲寄存器；
- 支持低位在前或高位在前的收发；
- 独立的接收中断和发送中断；
- 主模式下时钟频率可编程；
- 从模式可工作在 LPM4 下。

SPI 的结构如下，

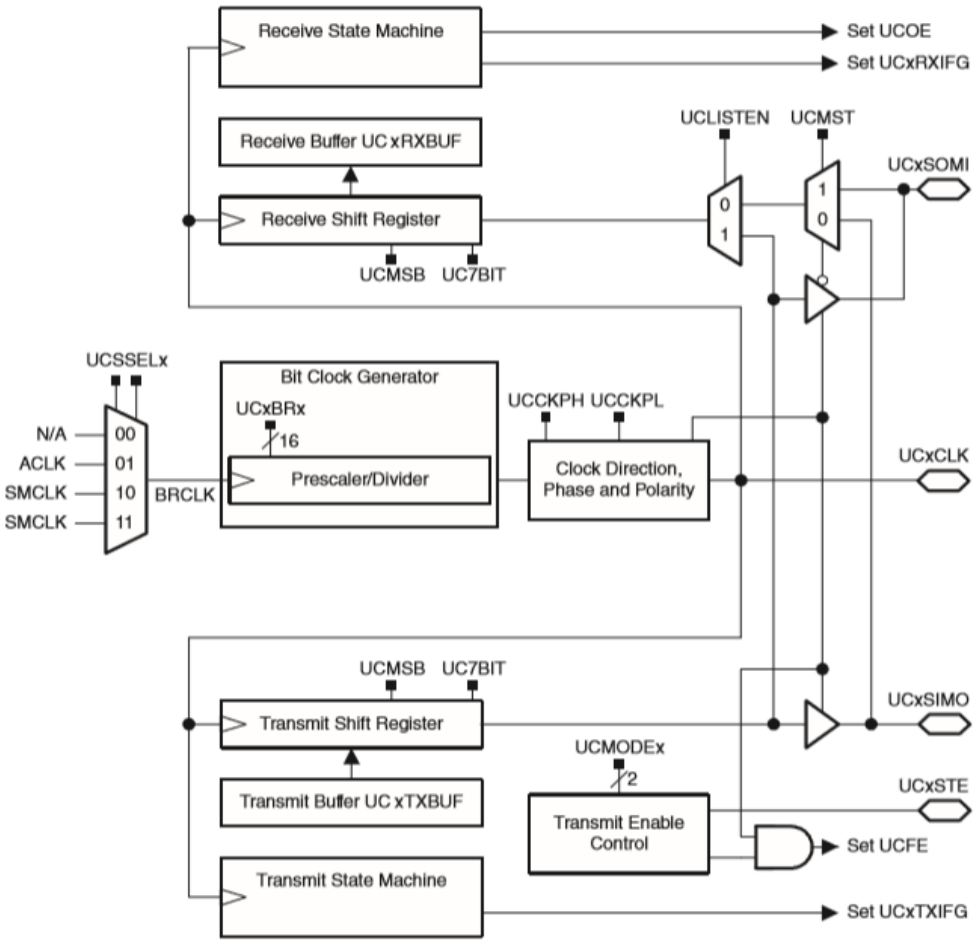


图 4-1 SPI 接口结构框图

如上图所示，在 SPI 模式下，数据的发送和接收是由多个器件共享一个由时钟运行的，该时钟是由一个主机提供的。UCxSIMO 是从机输入、主机输出，UCxSOMI 是从机输出、主机输入；UCxCLK 是 USCI 的 SPI 模式的时钟；UCxSTE 是从模式下的发送使能端，由主机控制，用于 4 线模式中选择一个从机接收和发送数据。

USCI 初始化和复位如下，

USCI 可以被 PUC 或者 UCSWRST 位复位。在 PUC 后，UCSWRST 位自动置位，保持 USCI 在复位状态。当置位时，UCSWRST 位复位 UCRXIE、UCTXIE、UCRXIFG、UCOE、UCFE 位并置 UCAXTXIFG 位。清除 UCSWRST 位可以是 USCI 进入工作状态。相应的 USCI 初始化/重配置的过程如下：

(1) 设置 UCSWRST；

(2) UCSWRST=1 时初始化所有的 UCSI 寄存器（包括 UCxCTL1）；

(3) 配置端口；

(4) 软件复位 UCSWRST；

(5) 通过 UCRXIE 或 UCTXIE 使能中断（可选）。

字符格式，在 SPI 模式下的 USCI 模块支持由 UC7BIT 位来选择 7~8 位字符长度。在 7 位数据模式下，只能选择 LSB；UCMSB 位控制着数据发送的方向且选择低位在前还是高位在前，缺省情况下是低位在前。

主模式如下，

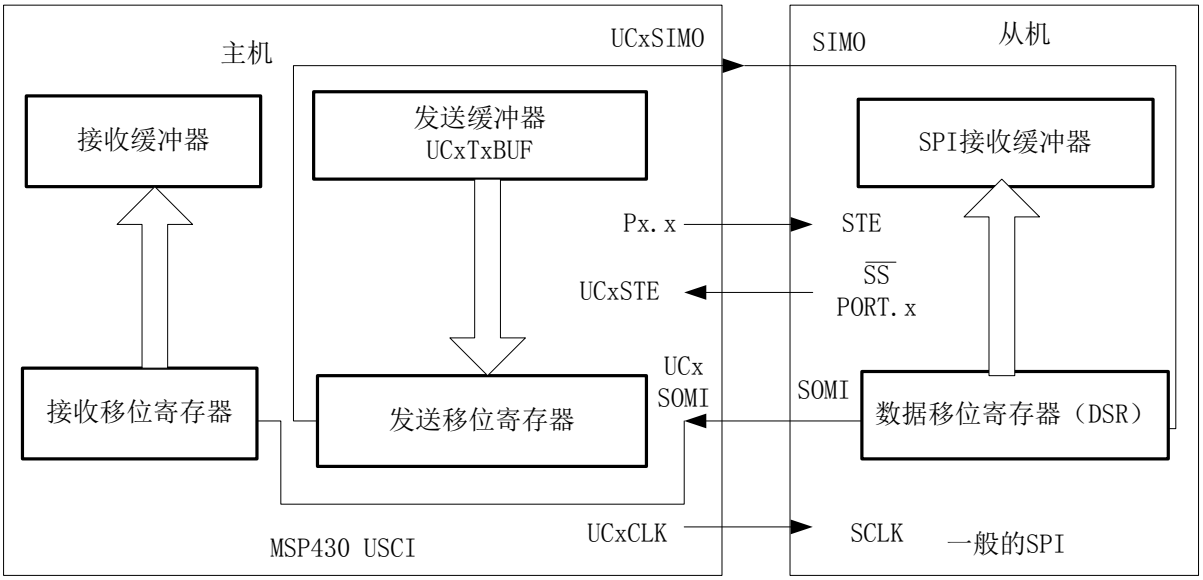


图 4-2 主机与外部从机结构

如上图所示 MSP430 的 USCI 作为主机，当数据移动到数据发送缓冲区时，USCI 开始数据发送。当 TX 移位寄存器为空的时候，缓冲区的数据会移动到 TX 移位寄存器，开始在 UCxSIMO 口进行数据发送，并且由 UCMSB 位的设置决定高位在前还是低位在前。在相反的时钟边沿，在 UCxSOMI 端的数据被移位到数据接收寄存器。当一个字节的数据接收完成，被接收的数据就会从 RX 寄存器被移动到数据接收缓冲区 UCxRXBUF，而且接收中断标志位 UCRXIFG 被置 1。

发送中断标志位 UCTXIFG=1 意味着数据已经从发送缓冲寄存器被完整移动到发送移位寄存器，此时发送缓冲区已经准备好了发送一组新的数据，但并不意味着数据的发送和接受工作已经完成了。在主模式下，由于数据的接收和发送是并行工作的，如需要接收数据到 USCI，必须向发送缓冲区写入数据。

从模式如下，

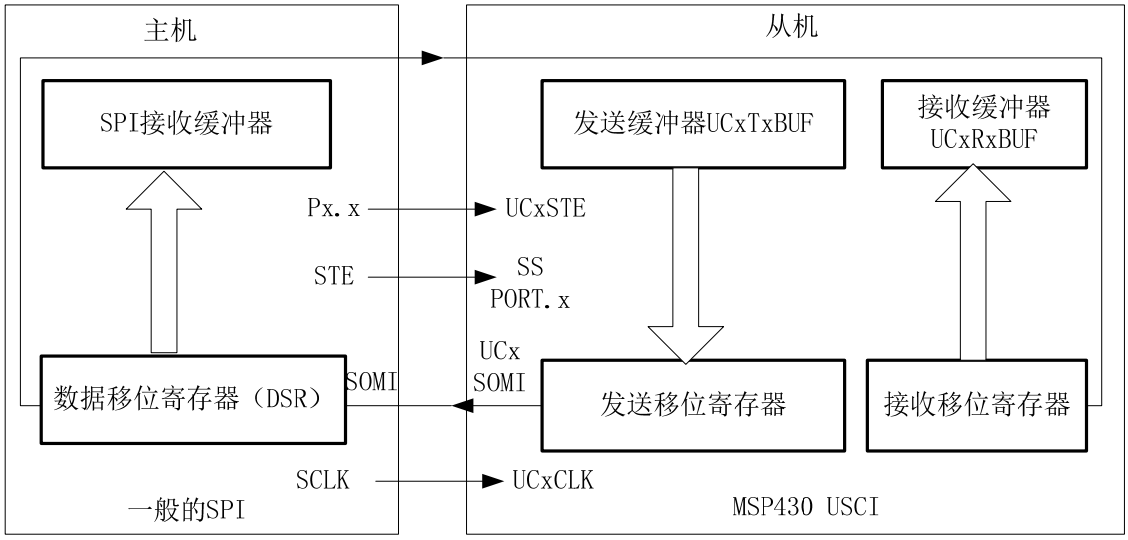


图 4-3 从机与外部主机结构

如上图所示 MSP430 的 USCI 作为从机。UCxCLK 用于 SPI 的时钟输入而且必须有外

部主机供电，数据传输的速率由这一时钟决定。待发数据应当在 UCxCLK 的起始边沿之前被写入并送至发送移位寄存器，并且会从 UCxSOMI 端口上发送出去。在时钟的另一边沿，UCxSIMO 端的数据会被移动到接收移位寄存器里，并且在预设位数的数据被接收完成后送入接收缓冲区 UCxRXBUF。当数据由接收移位寄存器移动到接收缓冲区时，接收的中断标志位被置 1，表明了数据已被接收。当之前传送的数据在新数据到来之前没有被接收缓冲区读到时会出现溢出错误，同时溢出错误标志位被置 1。

串行时钟控制介绍如下，

UCxCLK 由 SPI 总线上的主机提供，当 UCMST=1 时，位时钟由 UCxCLK 引脚上的 USCI 位时钟产生器提供，并且由 UCSSELx 位来进行时钟选择；当 UXMST=0 时，USCI 时钟由主机的 UCxCLK 引脚提供，此时不使用位时钟产生器，并且 UCSSELx 位并不起作用。SPI 的数据接收器和发送器并行工作，使用同一个时钟源。位于位速率控制寄存器 UCxxBR1 和 UCxxBR0 中的 16 位值的 UCBRx 是 USCI 时钟源的分频数。主模式下可生成的最高频率的位时钟是 BRCLK。在 SPI 模式中不使用调制，同时 SPI 模式时的 USCI-A 模块下 UCAxMCTL 应该被清零。

SPI 中断介绍如下，

每个 USCI 模块只有一个由发送和接收共享的中断向量，但 USCI-Ax 和 USCI-Bx 不共享同一个中断向量。发送中断标志位 UCTXIFG 是由发送器置 1 的，以便用来指示发送缓冲区 UCxTXBUF 做好接收下个字符的准备；此时如果 UCTXIE 和 GIE 也被置”1“，那么一个中断到来的时候就会产生一个中断请求。当一个字符被写入发送数据缓冲区时 UCTXIFG 会被自动复位。当 PUC 或者 UCSWRST=1 时会置位 UCTXIFG 并复位 UCTXIE。每次当接收一个字符并把字符装载到数据接收缓冲区时接收数据的中断标志位 UCRXIFG 就会被置 1，同时若 UCRXIE 和 GIE 被置 1 则也会有一个中断请求产生。UCRXIFG 和 UCRXIE 也会由 PUC 或者 UCSWRST=1 复位，当读取接收数据缓冲区时接收中断标志位也会自动复位。中断向量寄存器 UCxIV 可以被用来判断当前产生的是接收中断还是发送中断。

4.3.2 电子墨水屏显示技术

MSP430F5529 口袋板上没有采用常见的点阵液晶、段式液晶、TFT 等屏幕，而是采用了电子纸（电子墨水屏）这种革新的信息显示设备，它与传统的显示屏幕有很大不同，下面介绍一下其特点。

电子纸作为一种革新的信息显示的新方法和设备，区别于其他显示技术的具有以下几个突出优势：

（1）易阅读性

电子纸是靠反射环境光来显示图案的，它具有纸张印刷般的效果。与传统透射式液晶显示屏（TFT 等）相比，即使是在强烈的阳光底下，依然清晰可视；可视角度几乎达到了 180°。另外电子纸显示柔和、无闪烁，因此“电纸书”都是采用电子纸做屏幕。

（2）超级省电

电子纸只有在刷新屏幕的时候才会消耗电能，而且断电后能保持断电前最后一帧图片的显示，这是其他屏幕都做不到的，再加上不需要背光，因此电子纸适合作为电子标签使用。

（3）轻薄灵活

与其他显示屏幕相比，电子纸不管是在厚度还是重量上都有明显的优势，最薄可以做到 0.1mm，和纸张的厚度差不多。如果采用塑料薄膜作为基材，还具有可弯曲的特点。

以上说的都是电子纸的优势，下面说说电子纸目前的不足，这些技术不足，以及成本上的居高不下今后随着技术发展会逐渐得到解决的。

（1）刷新速度慢：尤其是电泳型电子纸刷新速度是比较慢的，目前无法做动画或动态视频显示。

（2）色彩还原不好：目前电子纸色彩还原还不如 TFT 等屏幕真实、鲜艳，无法很好的显示彩色照片，再加上成本原因，目前电子纸以点色灰阶型为主。

本实验所使用电子墨水屏相关数据手册会在课前提供给大家电子版，作为预习使用。

4.4 实验内容

F5529 口袋板使用了一片分辨率为 250×122 的 2.1 英寸的电子纸，SPI 接口。

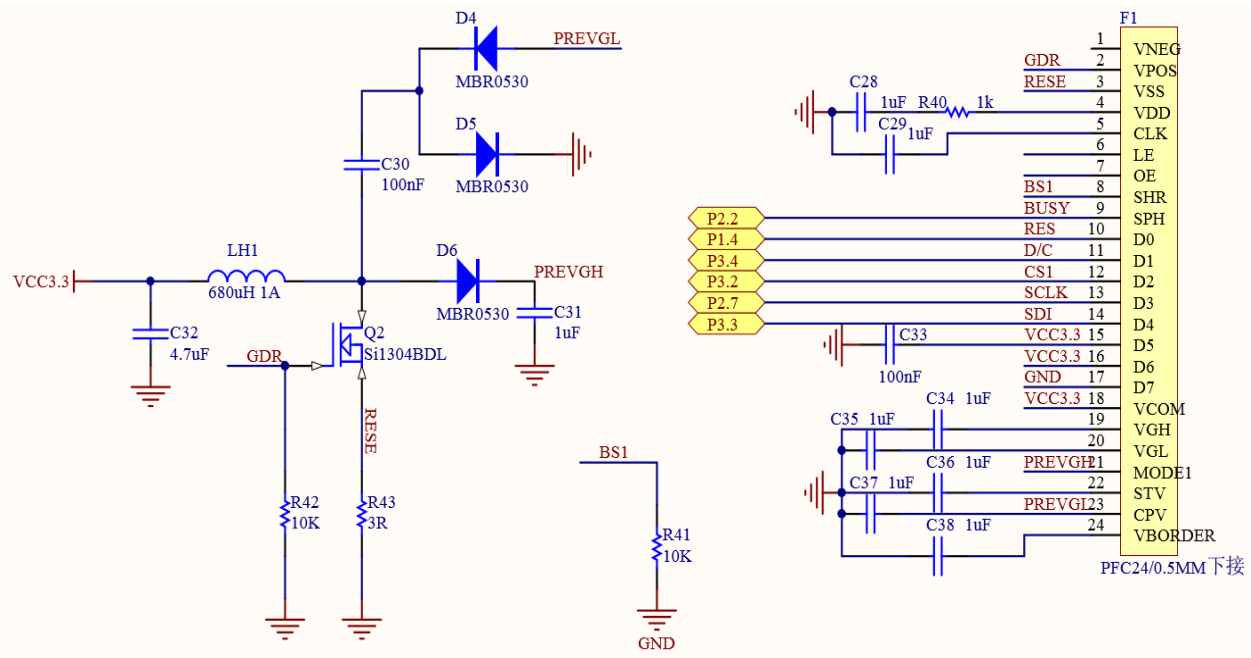


图 4-4 电子纸屏幕接口电路

4.4.1 电子墨水屏字符显示实验

编程思路：在写程序操作屏幕显示之前首先要配置好系统时钟，即为系统时钟配置合适的晶振，然后初始化屏幕，初始化过程中包括了配置 MSP430F5529 的 SPI 接口以及通过 SPI 接口对屏幕进行读写操作初始化屏幕。初始化 MSP430F5529 的 SPI 时序接口需要配置的寄存器有 UCA0CTL1、UCA0CTL0、UCA0BRW、P3SEL 等，寄存器详细功能可查阅文档。配置完 SPI 时序后还要通过 SPI 对屏幕的读写操作初始化屏幕后才算完成整个初始化过程；初始化完成后在通过 SPI 对屏幕的读写使屏幕上显示出图片和文字来。初始化和显示屏幕的操作可通过调用驱动来完成，相关的驱动函数及函数功能如下：

```
void display( unsigned char *str, //字符串
              unsigned int xsize, //x 方向位置
              unsigned int ysize, //y 方向位置
              unsigned int font, //字体 0,1,2
```

unsigned int size, //字号 0,1
unsigned int size, //字号 0,1
unsigned int size,) //字号 0,1

实验步骤如下，

（1） 导入已有工程代码，学习程序流程和编程思路，思考字符显示是以什么方式实现的（点阵还是 ASCII 码？），完成代码调试，并将代码烧录到开发板中；

（2）观察屏幕的状态变化，验证结果；

实验结果：

图 4-5 屏幕显示

4.4.2 电子墨水屏显示设计型实验

完成 4.4.1 小节实验后，对电子墨水屏显示程序的驱动接口有了一定了解。本节需要学会灵活使用程序中的驱动程序（包括更新显示缓存接口，SPI 通信进行显示等接口），请参照 4.4.1 节的工程程序，自行设计一个字符动态显示功能，可利用前面学过的定时器定时功能，或者 GPIO 中断，或者 AD 中断等，实现字符动态显示。

实验现象示例如下，仅供参考，可以自己设计更有趣生动的现象来实现。

（1） 按 S1 按键，屏幕显示：“I love you!”

按 S2 按键，屏幕显示：“Just Kidding!”;

（2） 屏幕共有 n 行字符显示，每按一次 S1 按键，屏幕刷新一行的选项，如下图显示更新。

Chapter 1	A moving and speaking puppet
Chapter 2	**Growing nose**
Chapter 3	Donkey ears
Chapter 4	A good boy Pinocchio

按键 S1 按下一次

Chapter 1	A moving and speaking puppet
Chapter 2	Growing nose
Chapter 3	**Donkey ears**
Chapter 4	A good boy Pinocchio

按键 S1 再按下一次

4.5 思考与分析

（1）电子纸屏幕是否可以显示汉字字符？如果可以，如何实现？

实验五　综合实验——展馆灯光控制

5.1 实验目的

（1） 了解光敏电阻等光线传感器的原理；
（2） 了解 ADC 数据采集软件滤波算法及其运用；
（3） 了解单片机 PWM 波形产生的原理及其在 LED 调光的运用；
（4） 了解单片机软件设计文件编写及程序编写规范。

5.2 实验仪器

（1） MSP430F5529LP+MSP430F5529 POCKET KIT 开发板一套；
（2） 光敏电阻传感器模块一套；
（3） 照度计一台；
（4） MSO2012B 示波器一台；
（5） PC 机操作系统 Windows 7，CCSv7.3 集成开发环境。

5.3 实验准备

5.3.1 光敏电阻的原理

光敏电阻是用硫化隔或硒化隔等半导体材料制成的特殊电阻器，其工作是基于内光电效应原理。光照愈强，阻值就愈低，随着光照强度的升高，电阻值迅速降低，亮电阻值可小至 1KΩ 以下。光敏电阻对光线十分敏感，其在无光照时，呈高阻状态，暗电阻一般可达 1.5MΩ。光敏电阻的特殊性能，随着科技的发展将得到极其广泛应用。

通常，光敏电阻器通常由光敏层、玻璃基片（或树脂防潮膜）和电极等组成，制成薄片结构，以便吸收更多的光能。当它受到光的照射时，半导体片（光敏层）内就激发出电子—空穴对，参与导电，使电路中电流增强。为了获得高的灵敏度，光敏电阻的电极常采用梳状图案。光敏电阻器。光敏电阻器在电路中用字母“R”或“RL”、“RG”表示

光敏电阻常用硫化镉（CDS）制成。它分为环氧树脂封装和金属封装两款，同属于导线型（DIP 型），环氧树脂封装光敏电阻按陶瓷基板直径分为 Ø3mm、Ø4mm、Ø5mm、Ø7mm、Ø11mm、Ø12mm、Ø20mm、Ø25mm 。

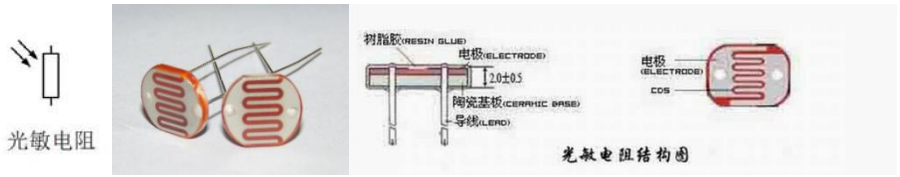


图 6-1 电阻符号、外形及结构

参数特性：根据光敏电阻的光谱特性，可分为三种光敏电阻器：紫外光敏电阻器、红外光敏电阻器、可见光光敏电阻器。

光敏电阻的主要参数是：

（1）光电流、亮电阻。光敏电阻器在一定的外加电压下，当有光照射时，流过的电流称为光电流，外加电压与光电流之比称为亮电阻，常用“100LX”表示。

（2）暗电流、暗电阻。光敏电阻在一定的外加电压下，当没有光照射的时候，流过的电流称为暗电流。外加电压与暗电流之比称为暗电阻，常用“0LX”表示(用照度计测量光的强弱，其单位为勒克斯 lx)。

（3）灵敏度。灵敏度是指光敏电阻不受光照射时的电阻值（暗电阻）与受光照射时的电阻值（亮电阻）的相对变化值。

（4）光谱响应。光谱响应又称光谱灵敏度，是指光敏电阻在不同波长的单色光照射下的灵敏度。若将不同波长下的灵敏度画成曲线，就可以得到光谱响应的曲线。

（5）光照特性。光照特性指光敏电阻输出的电信号随光照度而变化的特性。从光敏电阻的光照特性曲线可以看出，随着的光照强度的增加，光敏电阻的阻值开始迅速下降。若进一步增大光照强度，则电阻值变化减小，然后逐渐趋向平缓。在大多数情况下，该特性为非线性。

（6）伏安特性曲线。在一定照度下，加在光敏电阻两端的电压与电流之间的关系称为伏安特性。在给定偏压下,光照度较大，光电流也越大。在一定的光照度下，所加的电压越大，光电流越大，而且无饱和现象。但是电压不能无限地增大，因为任何光敏电阻都受额定功率、最高工作电压和额定电流的限制。超过最高工作电压和最大额定电流，可能导致光敏电阻永久性损坏。

（7）温度系数。光敏电阻的光电效应受温度影响较大，部分光敏电阻在低温下的光电灵敏较高，而在高温下的灵敏度则较低。

（8）额定功率。额定功率是指光敏电阻用于某种线路中所允许消耗的功率，当温度升高时，其消耗的功率就降低。

（9）指光敏电阻器从光照跃变开始到稳定亮电流的。

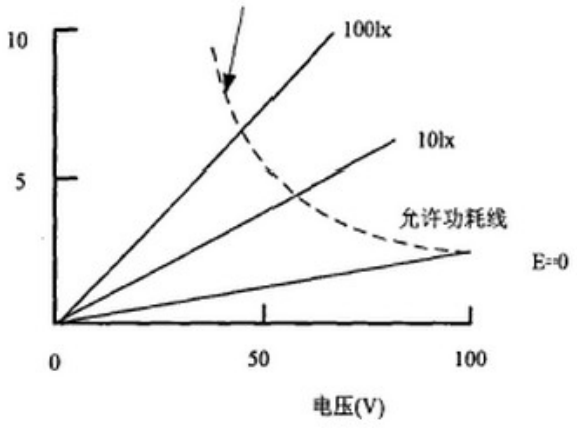


图 6-2 光敏电阻特性曲线

频率特性：当光敏电阻受到脉冲光照射时，光电流要经过一段时间才能达到稳定值，而在停止光照后，光电流也不立刻为零，这就是光敏电阻的时延特性。由于不 同材料的光敏，电阻时延特性不同，所以它们的频率 特性也不同。硫化铅的使用频率比硫化镉高得多，但多数光敏电阻的时延都比较大，所以，其不能用在要求快速响应的场合。

光照特性：光电流 I 和光通量 F 的关系曲线，称光照特性。不同的光敏电阻的光照特性是

不同的。但在大多数情况下，曲线的形状类似。由于光照特性是非线性的，不适宜做成线性的敏感器件。适合做开关电传感器。

本实验采用的是光敏电阻传感器模块，其工作电压 3.3V-5V，输出形式有两种，一种是 DO 数字开关量输出（0 和 1）只要由内宽电压 LM393 比较器实现，另一种为 AO 模拟电压输出。

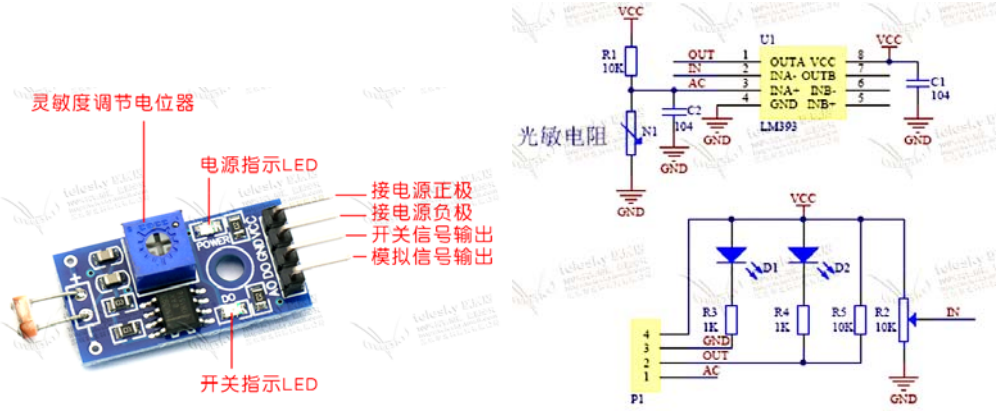


图 6-3 光敏电阻传感器模块其电路原理图

建议连接方式：

光敏电阻传感器模块 VCC-->MSP430F5529 开发板 3.3V。

光敏电阻传感器模块 GND -->MSP430F5529 开发板 GND。

光敏电阻传感器模块 AO-->MSP430F5529 开发板 P6.1。

5.3.2 麦克风信号的前置放大

一般的声音源种类很多，如传声器（话筒）、电唱机、CD 唱机及线路传输等，这些声音源的输出信号的电压差别很大，从零点几毫伏到几百毫伏。而一般功率放大器的输入灵敏度是一定的，这些不同的声音源信号如果直接输入到功率放大器中的话，对于输入过低的信号，功率放大器输出功率不足，不能充分发挥功放的作用；如果输入信号的幅值过大，功率放大器的输出信号将严重过载失真。所以一个实用的音频功率放大系统必须设置前置放大器，以便使放大器适应不同的的输入信号，或放大，或衰减，或进行阻抗变换，使其与功率放大器的输入灵敏度相匹配。

由于话筒输出信号非常微弱，一般只有 100 微伏~几毫伏，因此需要增加前置放大器，其主要功能一是匹配话筒的输出阻抗与前置放大器的输入阻抗；二是匹配前置放大器的输出电压幅度与功率放大器的输入灵敏度。F5529 口袋实验板上使用 TLV2764 集成运放作为麦克风信号的前置放大器，有兴趣的同学可以通过示波器观察一下麦克风输出信号的幅值与经过 TLV2764 放大后信号的幅值有什么区别。

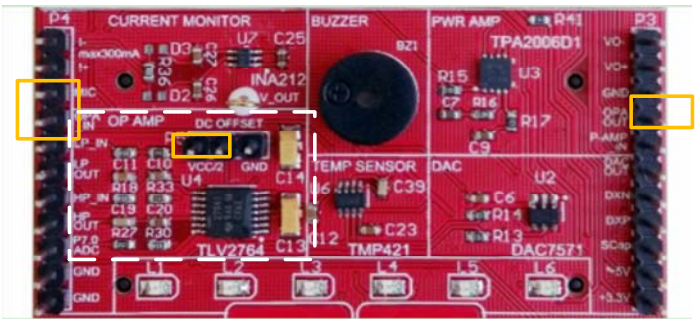


图 6-4 口袋板线路连接图

连接方式：

麦克风的输出信号与 TLV2764 运放的输入端连接起来，即将 F5529 口袋板左侧 MIC 与 OPA_IN 用跳线帽短接。

F5529 口袋板右侧 OPA_OUT 与 MSP430F5529LP 用杜邦线连接。

5.3.3 AD 转换数据采集软件滤波

由于干扰的存在，AD 转换数据往往出现大幅波动，影响其转换数据的准确性，为降低干扰的影响，除在外部增加滤波电路外，软件滤波工程中也经常用到，下面介绍几种经典的软件滤波方法的程序和优缺点分析：

（1）限幅滤波

1) **方法：** 根据经验判断，确定两次采样允许的最大偏差值（设为 A）；

每次检测到新值时判断：

如果本次值与上次值之差 <= A，则本次值有效；

如果本次值与上次值之差 > A，则本次值无效，放弃本次值，用上次值代替本次值。

2) 优点：能有效克服因偶然因素引起的脉冲干扰；

3) 缺点：无法抑制那种周期性的干扰 ，平滑度差。

4) 程序：

```
/* A 值可根据实际情况调整
value 为有效值，new_value 为当前采样值
滤波程序返回有效的实际值 */
#define A 10
char value;
char filter()
{
    char new_value;
    new_value = get_ad();
    if ( ( new_value - value > A ) || ( value - new_value > A ) )
    return value;
    else
    return new_value;
}
```

（2）中位值滤波法

1) 方法：连续采样 N 次（N 取奇数），其采样值按大小排列，取中间值为本次有效值。

2) 优点：能有效克服因偶然因素引起的波动干扰，对温度、液位的变化缓慢的被测参数有良好的滤波效果。

3) 缺点：对流量、速度等参数快速变化场合不适用。

4) 程序：

```
/* N 值可根据实际情况调整
排序采用冒泡法*/
#define N 11
char filter()
{
    char value_buf[N];
```

```
    char count,i,j,temp;
    for ( count=0;count<N;count++)
    {
        value_buf[count] = get_ad();
        delay();
    }
    for (j=0;j<N-1;j++)
    {
        for (i=0;i<N-j－1;i++)
        {
            if ( value_buf[i]>value_buf[i+1] )
            {
                temp = value_buf[i];
                value_buf[i] = value_buf[i+1];
                value_buf[i+1] = temp;
            }
        }
    }
    return value_buf[(N-1)/2];
}
```

（3）算术平均滤波法

1) 方法： 连续取 N 个采样值进行算术平均运算

N 值较大时：信号平滑度较高，但灵敏度较低；

N 值较小时：信号平滑度较低，但灵敏度较高；

N 值的选取：一般流量，N=12；压力：N=4

2) 优点：适用于对一般具有随机干扰的信号进行滤波，这样信号的特点是有一个平均值，信号在某一数值范围附近上下波动。

3) 缺点：对于测量速度较慢或要求数据计算速度较快的实时控制不适用，比较浪费 RAM

4) 程序：

```
#define N 12
char filter()
{
    int sum = 0;
    for ( count=0;count<N;count++)
    {
        sum + = get_ad();
        delay();
    }
    return (char)(sum/N);
}
```

（4）递推平均滤波法（又称滑动平均滤波法） (FIR 前身)

1) 方法： 把连续取 N 个采样值看成一个队列，队列的长度固定为 N，每次采样到一个

新数据放入队尾,并扔掉原来队首的一次数据。(先进先出原则) ，把队列中的 N 个数据进行算术平均运算,就可获得新的滤波结果。

N 值的选取：流量，N=12；压力：N=4；液面，N=4~12；温度，N=1~4。

2) 优点：对周期性干扰有良好的抑制作用，平滑度高 ，适用于高频振荡的系统。

3) 缺点：灵敏度低，对偶然出现的脉冲性干扰的抑制作用较差，不易消除由于脉冲干扰所引起的采样值偏差。

不适用于脉冲干扰比较严重的场合

比较浪费 RAM

1) 程序：

```
#define N 12
char value_buf[N];
char i=0;
char filter()
{
    char count;
    int sum=0;
    value_buf[i++] = get_ad();
    if ( i == N ) i = 0;
    for ( count=0;count<N,count++)
    sum+ = value_buf[count];
    return (char)(sum/N);
}
```

（5）中位值平均滤波法（又称防脉冲干扰平均滤波法）

1) **方法：**相当于“中位值滤波法”+“算术平均滤波法”

连续采样 N 个数据，去掉一个最大值和一个最小值，然后计算 N-2 个数据的算术平均值

N 值的选取：3~14；

2) **优点：**融合了两种滤波法的优点；对于偶然出现的脉冲性干扰，可消除由于脉冲干扰所引起的采样值偏差 。

3) **缺点：**测量速度较慢，和算术平均滤波法一样；比较浪费 RAM。

4) 程序：

```
#define N 12
char filter()
{
    char count,i,j;
    char value_buf[N];
    int sum=0;
    for (count=0;count<N;count++)
    {
        value_buf[count] = get_ad();
        delay();
    }
    for (j=0;j<N-1;j++)
```

```
    {
        for (i=0;i<N-j-1;i++)
        {
            if ( value_buf[i]>value_buf[i+1] )
            {
                temp = value_buf[i];
                value_buf[i] = value_buf[i+1];
                value_buf[i+1] = temp;
            }
        }
    }
    for(count=1;count<N-1;count++)
    sum += value[count];
    return (char)(sum/(N-2));
}
```

（6）限幅平均滤波法

1) **方法：**

相当于“限幅滤波法”+“递推平均滤波法”。

每次采样到的新数据先进行限幅处理，再送入队列进行递推平均滤波处理。

2) **优点：**融合了两种滤波法的优点 对于偶然出现的脉冲性干扰，可消除由于脉冲干扰所引起的采样值偏差 。

3) **缺点：**比较浪费 RAM。

4) **程序：**略 参考子程序 1、3。

（7）一阶滞后滤波法

1) **方法：**取 a=0~1；

本次滤波结果=（1-a）*本次采样值+a*上次滤波结果 。

2) **优点：**对周期性干扰具有良好的抑制作用 适用于波动频率较高的场合

3) **缺点：**相位滞后，灵敏度低 滞后程度取决于 a 值大小不能消除滤波频率高于采样频率的 1/2 的干扰信号。

4) **程序：**

```
/* 为加快程序处理速度假定基数为 100，a=0~100 */
#define a 50
char value;
char filter()
{
    char new_value;
    new_value = get_ad();
    return ((100-a)*value + a*new_value);
}
```

（7）加权递推平均滤波法

1) **方法：**该方法是对递推平均滤波法的改进，即不同时刻的数据加以不同的权，通常是，越接近现时刻的数据，权取得越大。给予新采样值的权系数越大，则灵敏度越高，但信号平滑

度越低 .

2) **优点：**适用于有较大纯滞后时间常数的对象和采样周期较短的系统

3) **缺点：**对于纯滞后时间常数较小，采样周期较长，变化缓慢的信号不能迅速反应系统当前所受干扰的严重程度，滤波效果差。

4) **程序：**

```
/* coe 数组为加权系数表，存在程序存储区。*/
#define N 12
char code coe[N] = {1,2,3,4,5,6,7,8,9,10,11,12};
char code sum_coe = 1+2+3+4+5+6+7+8+9+10+11+12;
char filter()
{
    char count
    char value_buf[N];
    int sum=0;
    for (count=0,count<N;count++)
    {
        value_buf[count] = get_ad();
        delay();
    }
    for (count=0,count<N;count++)
    sum += value_buf[count]*coe[count];
    return (char)(sum/sum_coe);
}
```

（8）消抖滤波法

1) **方法：**设置一个滤波计数器 ，将每次采样值与当前有效值比较：

如果采样值 ＝ 当前有效值，则计数器清零。

如果采样值 < 或 > 当前有效值，则计数器 +1，并判断计数器是否 >= 上限 N(溢出)

如果计数器溢出,则将本次值替换当前有效值，并清计数器。

2) **优点：**对于变化缓慢的被测参数有较好的滤波效果, 可避免在临界值附近控制器的反复开/关跳动或显示器上数值抖动。

3) **缺点：**对于快速变化的参数不宜。如果在计数器溢出的那一次采样到的值恰好是干扰值,则会将干扰值当作有效值导入系统。

4) **程序：**

```
#define N 12
char filter()
{
    char count=0;
    char new_value;
    new_value = get_ad();
    while (value !=new_value)
    {
        count++;
```

```
        if (count>=N) return new_value;
        delay();
        new_value = get_ad();
    }
    return value;
}
```

（9）限幅消抖滤波法

1) **方法：**相当于“限幅滤波法”+“消抖滤波法”，先限幅，后消抖 。

2) **优点：**继承了“限幅”和“消抖”的优点 ，改进了“消抖滤波法”中的某些缺陷,避免将干扰值导入系统。

3) **缺点：**对于快速变化的参数不宜，程序略 参考子程序 1、9。

5.3.4 LED 发光原理

LED（Light Emitting Diode），发光二极管，是一种能够将电能转化为可见光的固态的半导体器件，它可以直接把电转化为光。LED 晶片的一端附在一个支架上，一端是负极，另一端连接电源的正极，使整个晶片被环氧树脂封装起来。半导体晶片由两部分组成，一部分是 P 型半导体，在它里面空穴占主导地位，另一端是 N 型半导体，在这边主要是电子。但这两种半导体连接起来的时候，它们之间就形成一个 P-N 结。当电流通过导线作用于这个晶片的时候，电子就会被推向 P 区，在 P 区里电子跟空穴复合，然后就会以光子的形式发出能量，这就是 LED 发光的原理。而光的波长也就是光的颜色，是由形成 P-N 结的材料决定的。

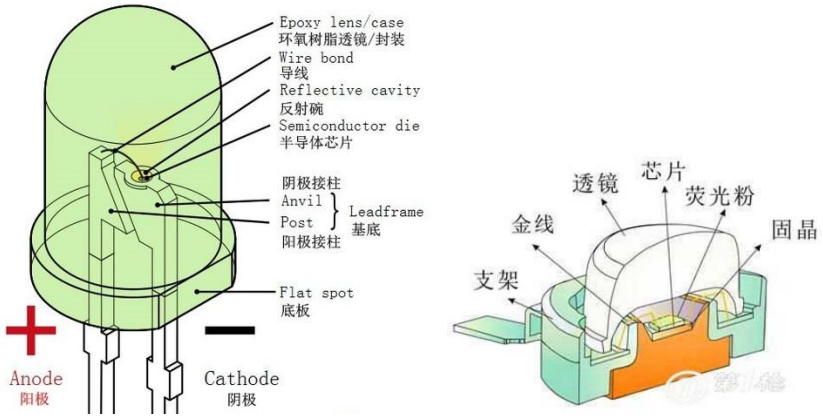


图 6-5 发光二极管结构图

LED 通常由含镓（Ga）、砷（As）、磷（P）、氮（N）等的化合物制成，砷化镓二极管发红光，磷化镓二极管发绿光，碳化硅二极管发黄光，氮化镓二极管发蓝光。

发光二极管的压降与使用材料及型号有关，对于同型号的 LED，其压降基本相同，通过 LED 的电流决定其发光强度。LED 的工作电流由下式计算：

$$I_F = \frac{V_{CC} - (V_F + V_{CS})}{R}$$

其中，V_F 为 LED 正向压降（黄绿、黄、红 LED 压降在 1.8-2.4v，可取 2.0v；蓝、绿、白光 LED 压降在 2.8-4.0v，可取 3.3v）；V_{CS} 为 LED 驱动器的压降，R 为 LED 限流电阻，V_{CC} 为

电源电压；I_F 为 LED 工作电流（LED 的工作电流一般为 5～20mA）。

限流电阻的选取：

$$R=\frac{V_{CC}-(V_F+V_{CS})}{I_F}$$

如果V_{CC} 为 5V，I_F 为 10mA,V_F 为 2V，V_{CS} 为 0.2V，则：

$$R=0.28(k\Omega)$$

5.3.5 脉宽调制波 PWM

PWM 是脉宽调制波英文 PWM（Pulse-Width Modulation）的缩写，PWM 信号是一种具有固定周期不定占空比的数字信号，如下图所示：

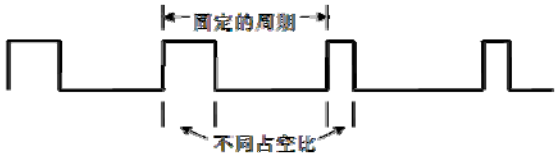


图 6-6　PWM 波形

如果 Timer_A 定时器的计数器工作在增计数方式，输出采用输出模式 7（复位/置位模式），则可利用寄存器 TAxCCR0 控制 PWM 波形的周期，用某个寄存器 TAxCCRx 控制占空比。这样 Timer_A 就可以产生出任意占空比的 PWM 波形。如下图所示：

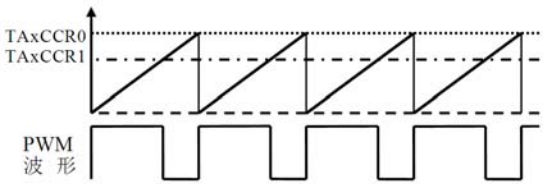


图 6-7 波形产生原理

可以随时间变化任意改变 PWM 信号的占空比，具体做法：

- 保持 CCR0 值（周期不变）；
- 改变 CCRx 值（改变占空比）。

如下图所示：

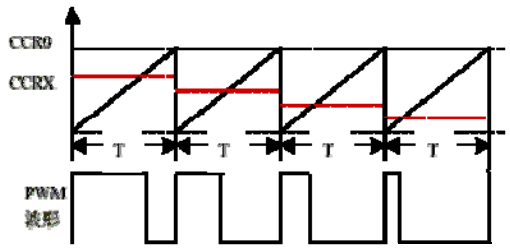


图 6-8 随时间变化任意改变 PWM 信号波形

如果 PWM 信号占空比随时间变化，那么经过滤波之后的输出信号就是幅度变化的模拟信号，因此通过控制 PWM 信号的占空比，就可以产生不同的模拟信号，实现 D/A 转换。如下图所示：

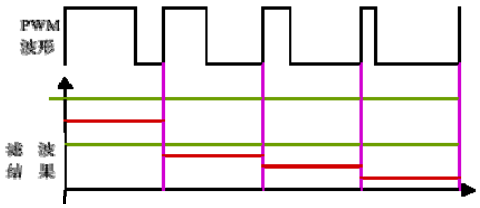


图 6-9 PWM 信号滤波结果

PWM 不需要修改占空比和时间时，CPU 在做完 Timer_A 初始化工作之后，Timer_A 就能自动输出 PWM，而不需利用中断维持 PWM 输出，此时 CPU 就可以进入低功耗状态。

5.3.6 H 桥驱动电路

电机或高功率 LED 灯运行时需要的较大电流，单片机普通 IO 口的驱动能力是不够，所以需要加入驱动电路，在 MSP430F5529 POCKET KIT 中，典型的驱动电路最就是 H 桥驱动电路。

下图所示为一个典型的直流电机控制电路。电路得名于“H 桥驱动电路”是因为它的形状酷似字母 H。4 个三极管组成 H 的 4 条垂直腿，而电机就是 H 中的横杠。如图所示，H 桥式电机驱动电路包括 4 个三极管和一个电机。要使电机运行，必须导通对角线上的一对三极管。因为根据不同三极管对的导通情况，电流可能会从左至右或从右至左流过，所以在驱动 LED 过程中，必须控制 LED 两端的电压极性，否则容易击穿二极管。

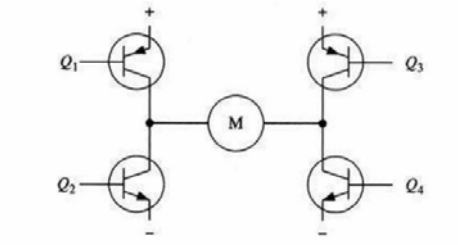


图 6-10 H 桥驱动电路

要使 LED 灯点亮，必须使对角线上的一对三极管导通。如下图所示，当 Q1 管和 Q4 管导通时，电流就从电源正极经 Q1 从左至右穿过 LED，然后再经 Q4 回到电源负极。按图中电流箭头所示，该流向的电流将驱动 LED。

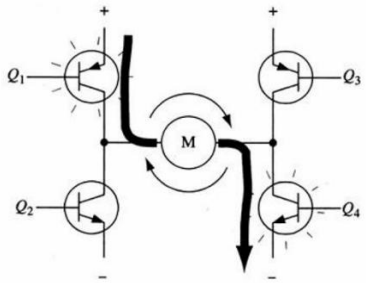


图 6-11 H 桥驱动电流示意图

驱动电机时，保证 H 桥上两个同侧的三极管不会同时导通非常重要。如果三极管 Q1、Q2 同时导通，那么电流就会从正极穿过两个三极管直接回到负极。此时，电路中除了三极管外没有其他任何负载，因此电路上的电流就可能达到最大值（该电流仅受电源性能限制），甚至烧坏三极管。基于上述原因，在实际驱动电路中通常要用硬件电路方便地控制三极管的开关。

下图所示就是基于这种考虑的改进电路，它在基本 H 桥电路的基础上增加了 4 个与门 2

个非门。4 个与门同一个“使能”导通信号相接，这样，用这一个信号就能控制整个电路的开关。而 2 个非门通过提供一种方向输入，可以保证任何时候在 H 桥的同侧都只有一个三极管能导通。（与本节前面的示意图一样，图 6-12 所示也不是一个完整的电路图，特别是图中与门和三极管直接连接是不能正常工作的。）

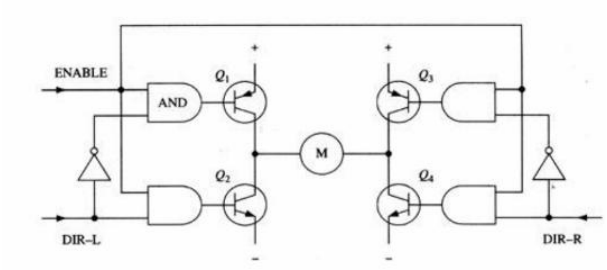


图 6-12 具有使能控制和方向逻辑的 H 桥电路

采用以上方法，H 桥就只需要用三个信号控制：两个方向信号和一个使能信号。如果 DIR-L 信号为 0，DIR-R 信号为 1，并且使能信号是 1，那么三极管 Q1 和 Q4 导通，电流流向是从左至右（如下图所示）；如果 DIR－L 信号变为 1，而 DIR-R 信号变为 0，那么 Q2 和 Q3 将导通，电流则反向流过电机。

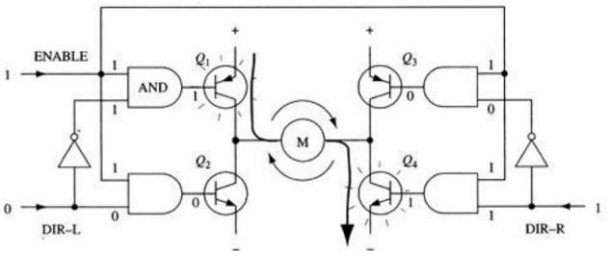


图 6-13 使能信号与方向信号的使用

由于分立元件制作 H 桥较为复杂，实际中往往使用封装好的 H 桥集成电路，只需连接电源、电机和控制信号即可，在额定的电压和电流内使用非常方便可靠。

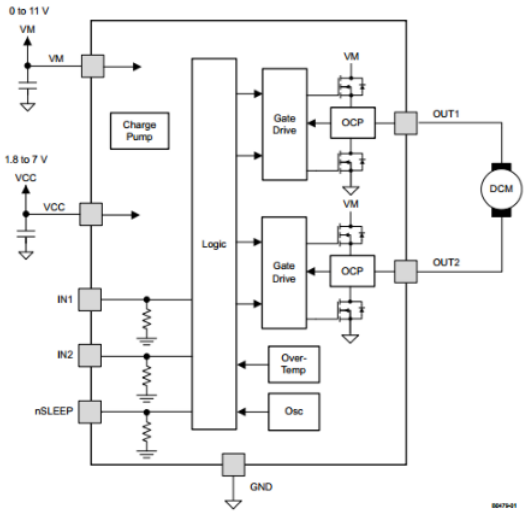


图 6-14 TI DRV8837 芯片内部结构图

口袋板上使用了 TI DRV8837 低电压电机驱动芯片，芯片的基础构造就是用的前面介绍的 H 桥。DRV8837 的主要特性与优势：

●延长电池使用寿命：280 毫欧低 RDS(ON)与仅为 35 nA 的超低睡眠电流可为玩具、智能气表或水表、电子锁、微型打印机以及摄像机等应用提高散热性能，延长电池使用寿命；

●提高性能：逻辑与电机的分开供电可提高启动及停止条件下电池供电应用的性能；

●宽/低电压工作范围：1.8 V 至 11 V 宽泛工作电压支持多达 6 节碱性电池或 2 节锂离子电池组应用；

●减少板级空间：微小型 2 毫米×2 毫米 WSON 封装支持更小、更时尚的设计；

●高级片上保护：过流、过温、贯通以及欠压保护可提高系统可靠性，降低设计复杂性。

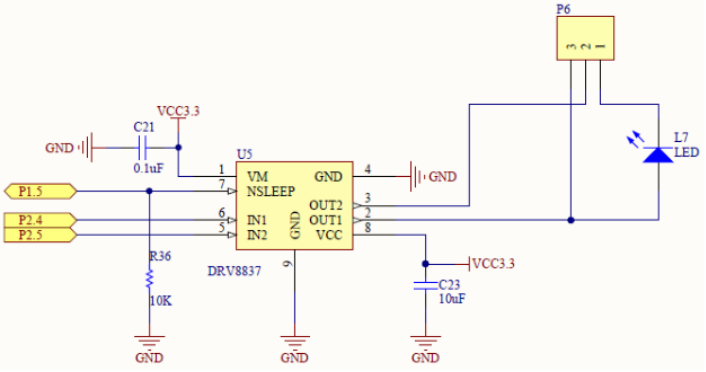


图 6-15 H 桥驱动模块原理图

如图所示，P1.5 为控制引脚 NSLEEP，把该引脚置高驱动电路生效，把该引脚置低，驱动电路失效。驱动 LED 时，可设为需将跳线帽短接 P6 的 1-2 插针。

5.4 程序分析

5.4.1 功能描述

该系统主要用于展览馆等需要解说员解说，且需要调节光线以达到最佳演示效果的场合。系统检测到外界声音后打开灯光，系统根据周边环境光的情况自动调整 LED 灯的亮度，已达到最佳展示效果。30 秒后，如果没有检测到声音信号，则自动关闭 LED 灯，以达到节能目的。

5.4.2 总体设计

系统主要由声音模块、光敏电阻传感器模块、LED 及其驱动模块组成。声音模块主要由麦克风及放大电路组成，通过单片机 AD12 通道 0 采集声音信号。光敏电阻传感器模块直接连到单片机 AD12 通道 1，采集光线强度信号。LED 及其驱动模块由 H 桥电路和大功率 LED 组成，通过 P1.5、P2.4、P2.5 控制其开关和亮度。

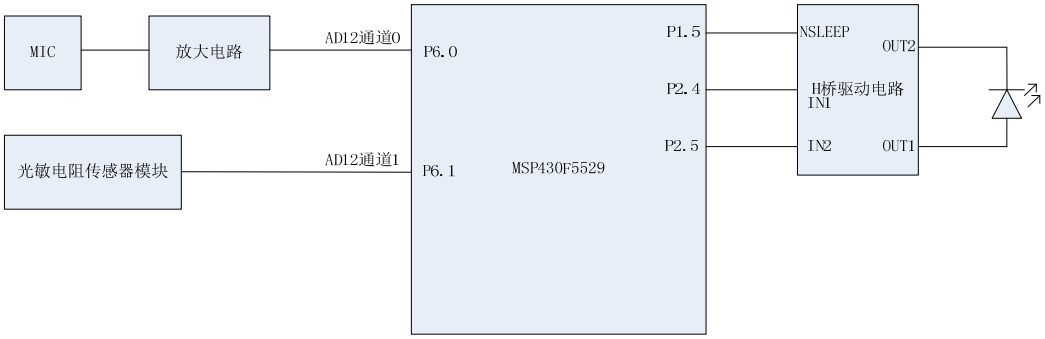


图 6-16 系统总体设计

5.4.3 编程思路

编程思路：判断声音采集电压，如果超过一定设定电压值，则判定讲解员正在讲解。此时打开 LED 使能，同时开始计时 20 秒，如果期间声音采集电压还超过设定电压值，则重新开始计时，直到 20 秒内没有超过设定电压值。光敏电阻的光强电压则一直在检测，随时调整占空比，控制 LED 电流。

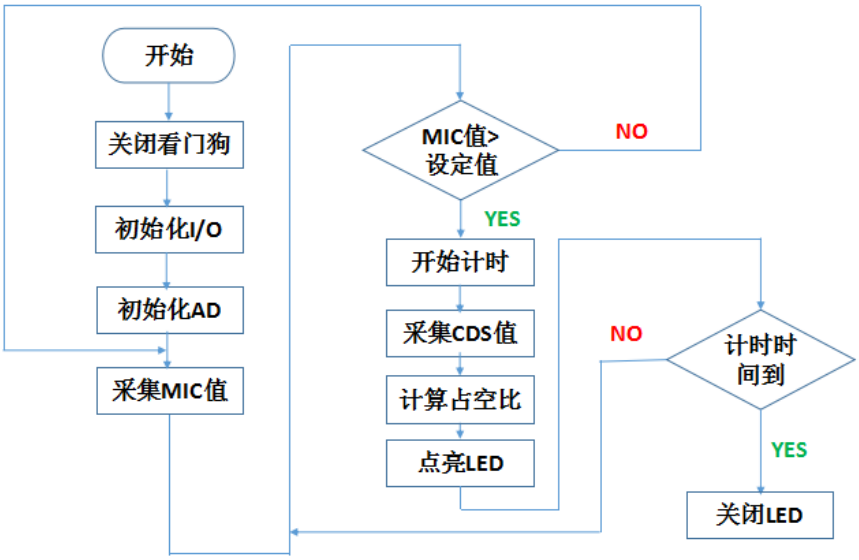


图 6-17 程序流程图

5.5 功能验证实验

5.5.1 LED 驱动控制实验

大功率 LED 发光时需要较大电流,因此本实验利用板载 DRV8837 驱动 LED ，通过 PWM 调节 LED 亮度。

（1）**电路连接**　用跳线帽短接图上插针。

图 6-18 实验接线示意图

（2）编程思路

首先按照驱动电机的原理，让 LED 满额长亮。然后用设置 PWM 占空比，用 PWM 信号控制电机驱动电路工作和停止。通过调节占空比就可以控制功耗了。

（3）实验现象

高功率 LED 灯点亮，并随着占空比的改变亮度也改变。

5.5.2 声音模块采集实验

（1）电路连接

麦克风的输出信号与 TLV2764 运放的输入端连接起来，即将 F5529 口袋板左侧 MIC 与 OPA_IN 用跳线帽短接。

F5529 口袋板右侧 OPA_OUT 与 MSP430F5529LP 用杜邦线连接。

（2）编程思路

熟悉了 MSP430F5529 中的 ADC12 模块原理之后便可对其控制寄存器进行配置，设计采样模式，得到声音模块的输出端电压值。同时将电压值显示在墨水屏中。

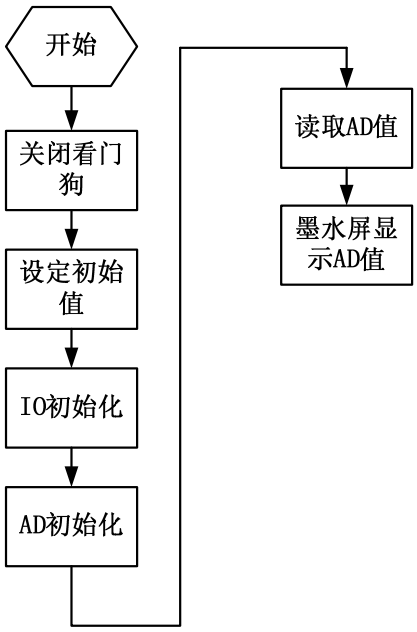


图 6-19 声音模块电压采集及显示程序流程图

（3）实验步骤

a) 按照电路连接要求，连接模块与开发板，检查无误后，才能上电

b) 根据编程思路设计结构与实现方法，按照流程图实现代码编写，并在编译器上进行编译改错。可参考导入已有工程，编译、调试并下载程序到开发板。

c) 运行程序，改变环境音量，查看墨水屏显示数据变化。

（4）实验现象

墨水屏显示数据随音量变化而变化。

5.5.3 ADC 采集软件滤波实验

（1）电路连接

参照实验五。

（2）编程思路

本实验采用中位值平均滤波法（又称防脉冲干扰平均滤波法）相当于“中位值滤波法”+“算术平均滤波法” 连续采样 N 个数据，去掉一个最大值和一个最小值，然后计算 N-2 个数据的算术平均值。该方法融合了两种滤波法的优点；对于偶然出现的脉冲性干扰，可消除由于脉冲干扰所引起的采样值偏差。

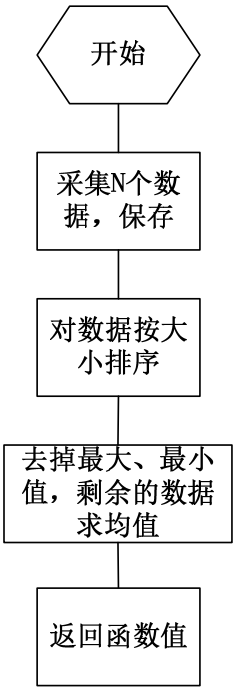


图 6-20 软件滤波程序流程图

（3）实验步骤

a) 按照电路连接要求，连接模块与开发板，检查无误后，才能上电。

b) 将采集光强数据文件，AD 通道改为 P6.5（A5），观察 LED 的变化。

c) 根据编程思路设计结构与实现方法，按照流程图实现代码编写，并在编译器上进行编译改错（注：请给 6.3.2 中提供的例程挑错），查看 LED 灯的变化是否符合设计要求。可参考导入已有工程，编译、调试并下载程序到开发板。

d) 运行程序，在不调整电位器的条件下，观察加入软件滤波后 LED 灯的变化。

（4）实验现象

加入软件滤波的程序 LED 闪烁现象消除。

5.6思考与分析

（1） 墨水屏幕刷新很慢，如何做到不总在刷屏？

（2） 请尝试 AD 采集麦克风的信号，设计跟随音乐的音频幅度变化而闪亮的 LED 灯。