工程训练（电子工艺实习）

电子产品嵌入式软件设计

实验与创新实践教育中心
哈尔滨工业大学（深圳）

工程训练（电子工艺实习）

工程项目开发与管理——电子工艺实习项目制作

课程规则：

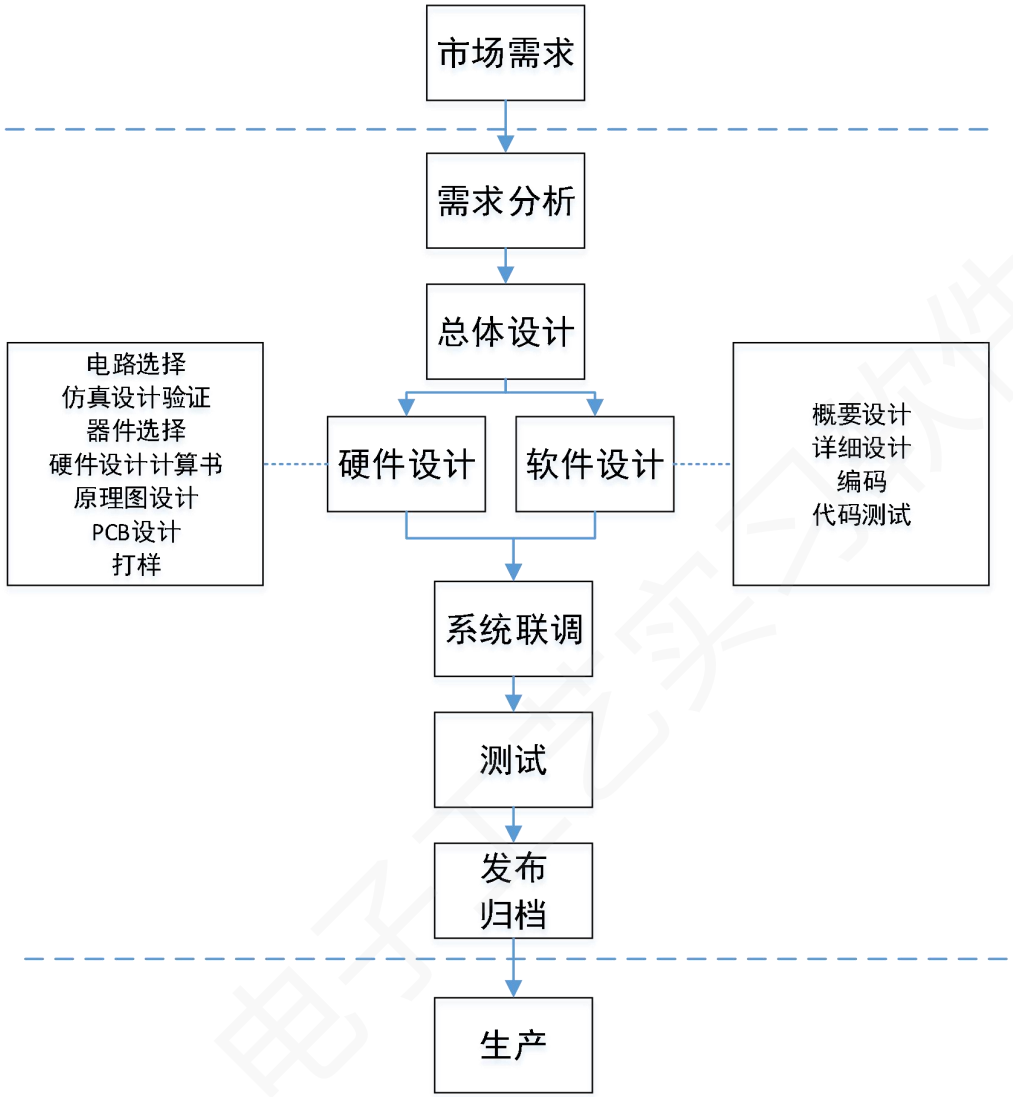
- 运用课程中所学的项目管理、电子产品设计制造相关知识，按照项目开发和管理流程完成一个电子产品项目的实施。
- 提交项目文档，发布项目（路演）。

具体要求在项目制作1课上会有负责的老师来讲解布置！

工程项目开发


项目开发
技术开发
生产交付
软件
硬件
结构
硬件生产
软件生产
库存、原材料管理
项目交付及运营
软件设计
元器件识别
硬件设计
PCB设计
焊接调试
PCB制造
SMT生产工艺
电子封装设计
电子可靠性
软件版本管理
软件上线/烧录
智能楼宇

嵌入式电子产品开发流程



市场部

研发部

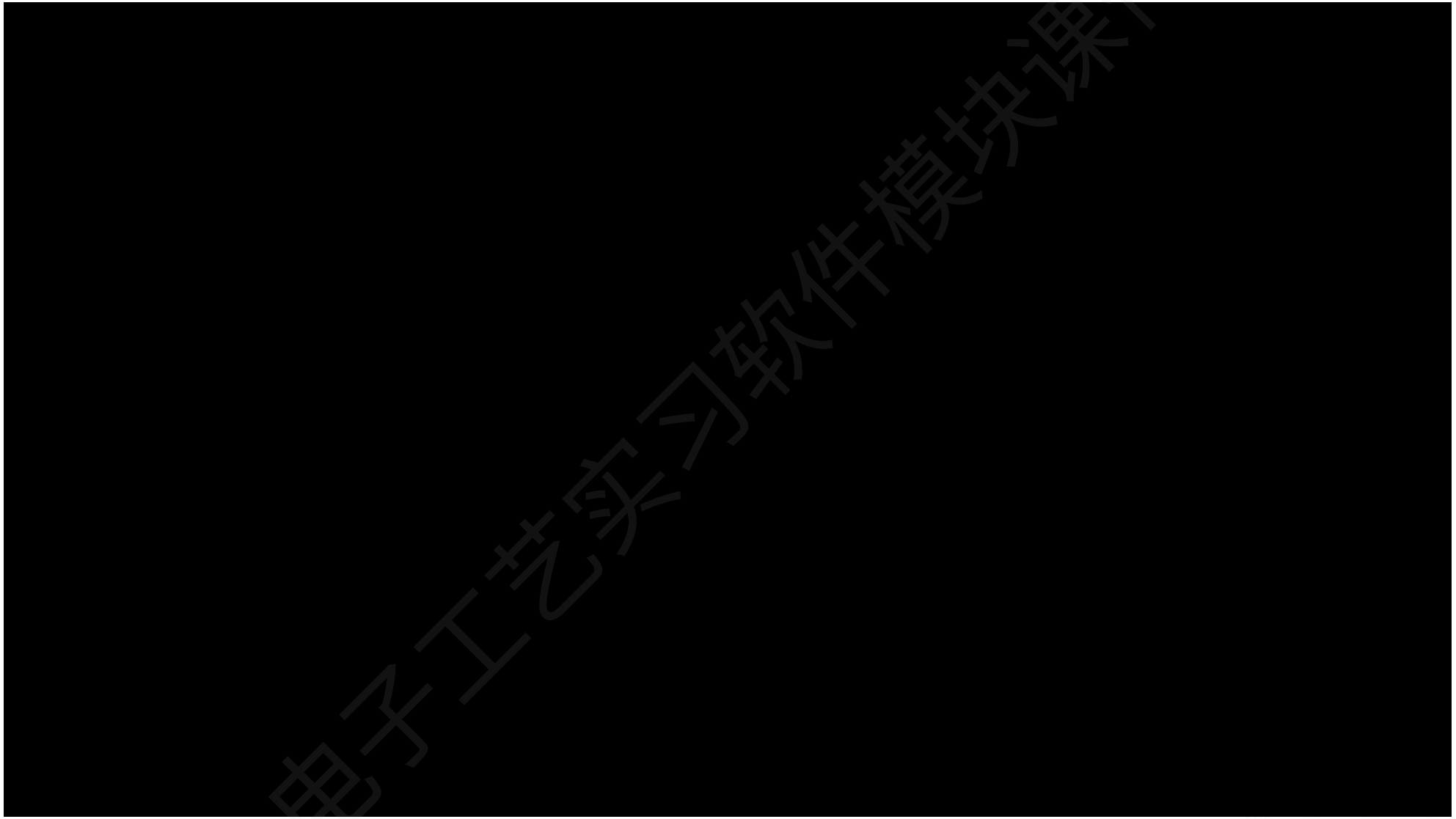
制造部

版本管理

教学目标

1. 了解电子产品系统设计流程；
2. 了解产品开发项目管理基本概念；
3. 了解嵌入式软件设计流程及设计文档撰写；
4. 熟悉嵌入式软件编码、调试基本方法；
5. 熟悉相关技术文档查阅方法；
6. 熟悉嵌入式软件开发工具的使用方法。

我们设计的电子产品——往届学生作品分享



实验案例体验项目

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，以达到最佳展示效果。如果一定时间内都没有检测到声音信号，则自动关闭LED灯，以达到节能目的。

从中提取关键信息：

1.能感知声音信号

2.能感知光强信号

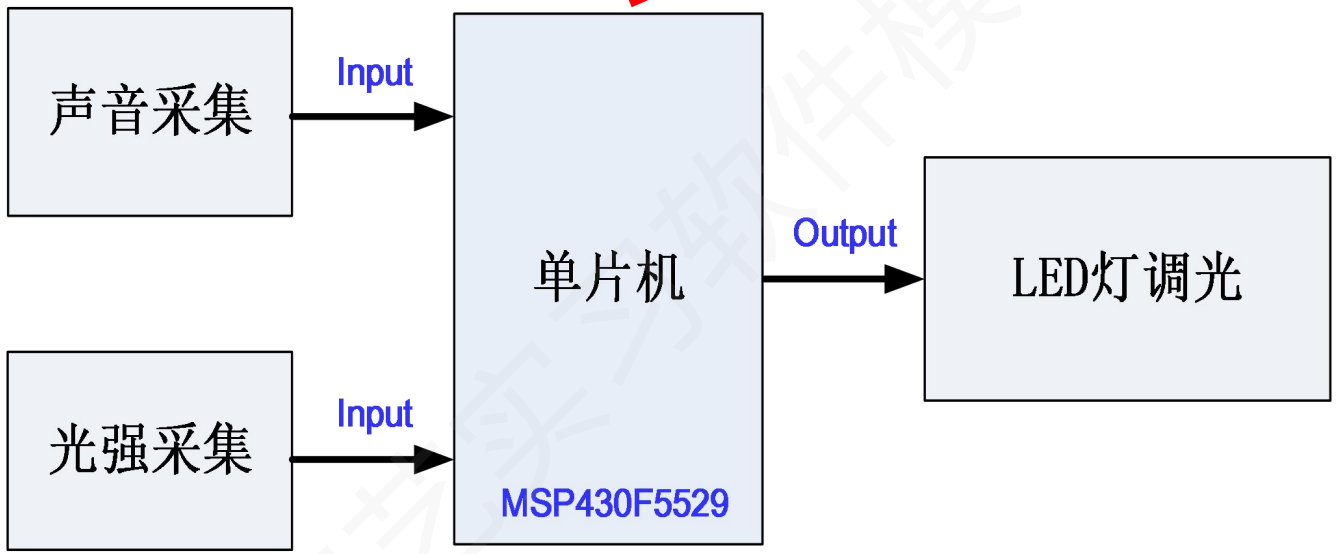
3.能调节LED灯亮度

4.能自动关闭LED灯

声控灯+自动调节亮度

3个模块

主控芯片选择：资源（主频、存储、外设……）、成本、体积、功耗、是否主流等

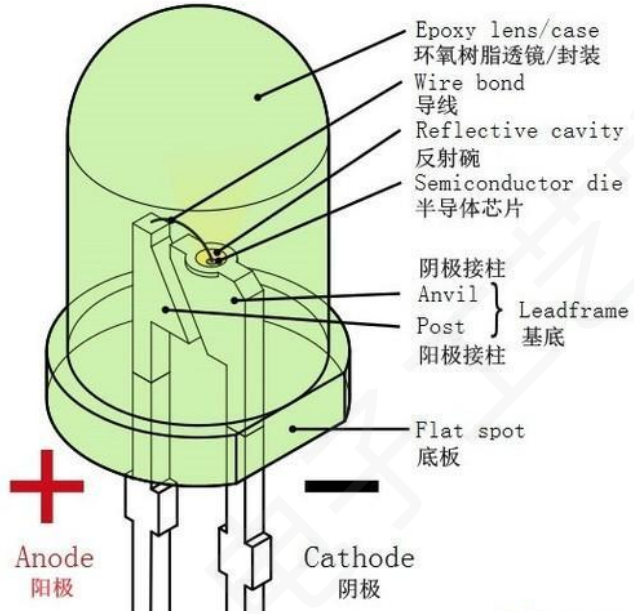


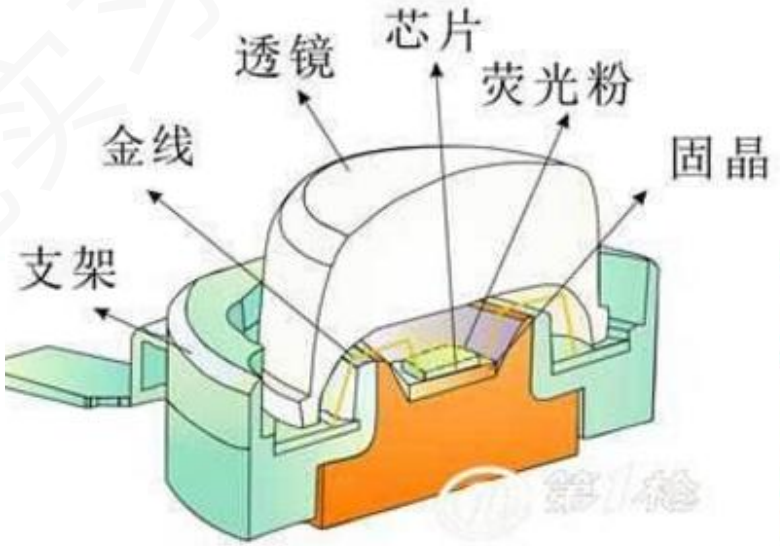
展馆灯光控制系统原理框图

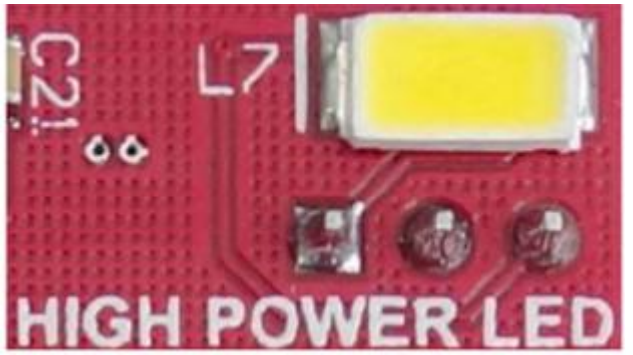
受控对象——大功率LED

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。

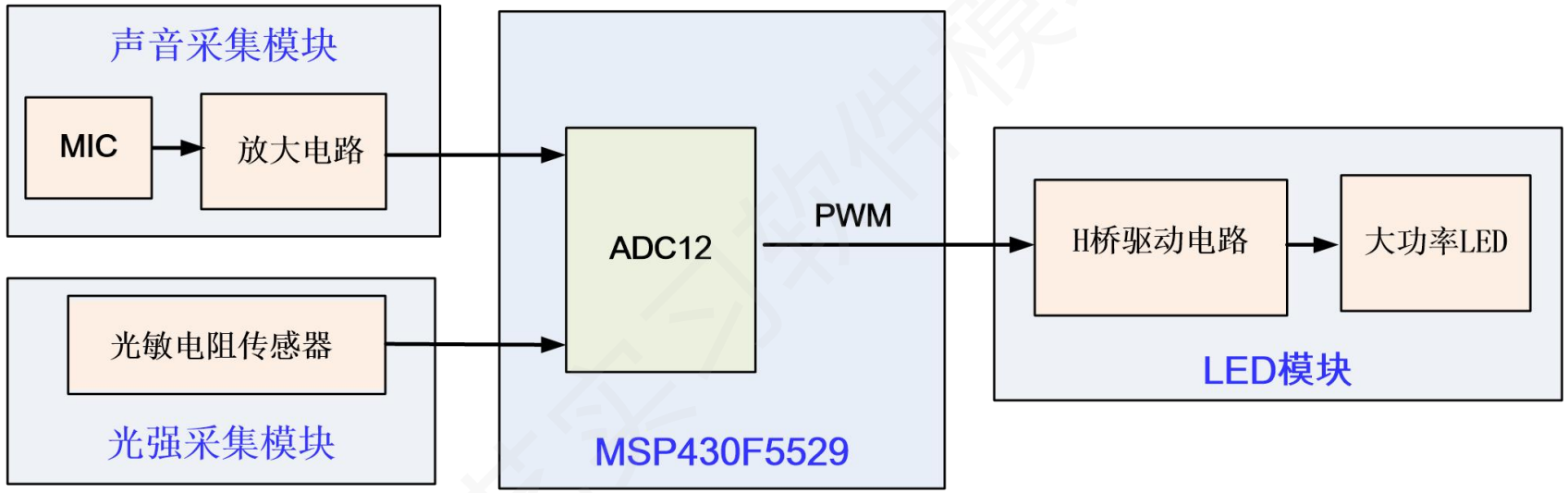
- 在一定范围内，电流越大，亮度越大。
- 单片机的输出电流不足以驱动大功率LED

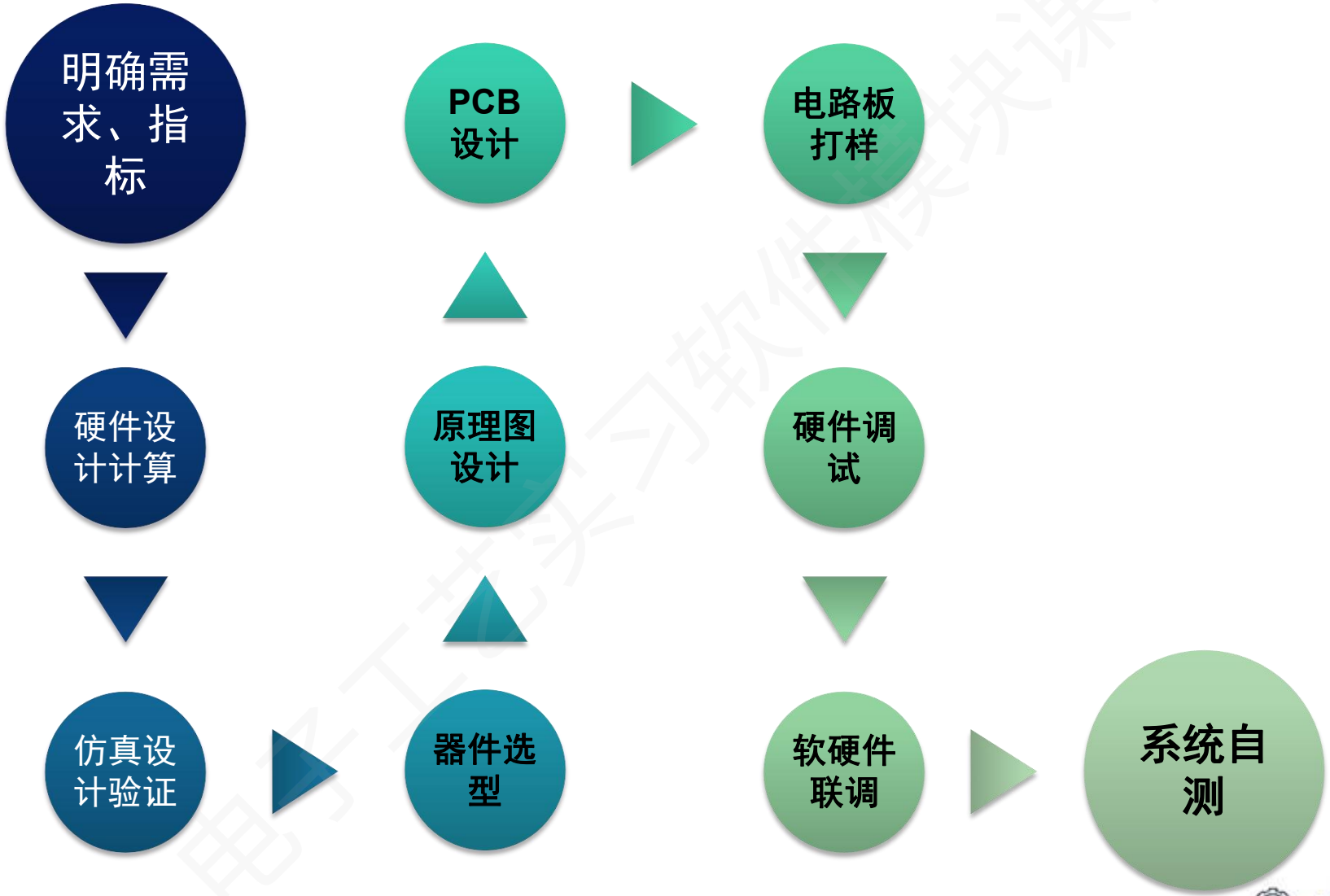


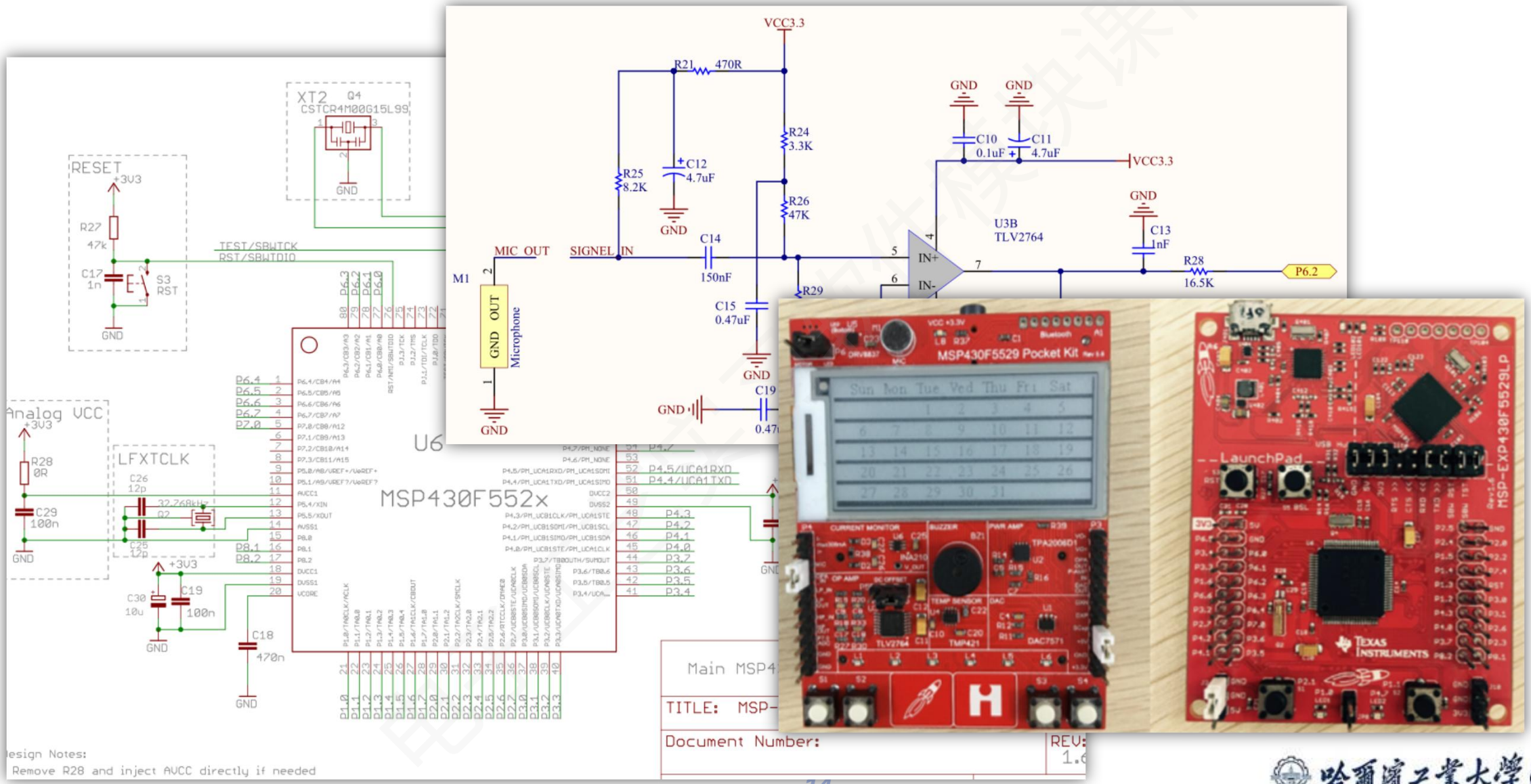


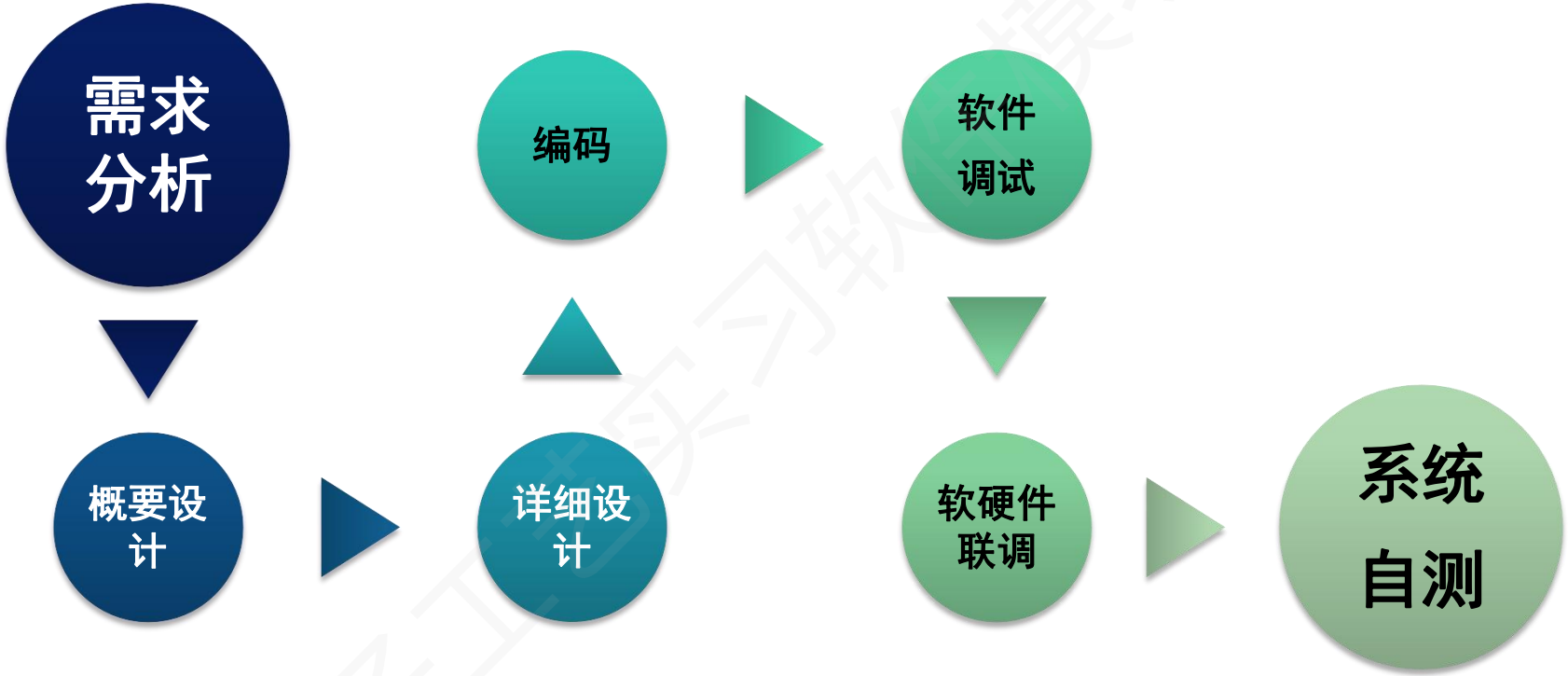


原理框图

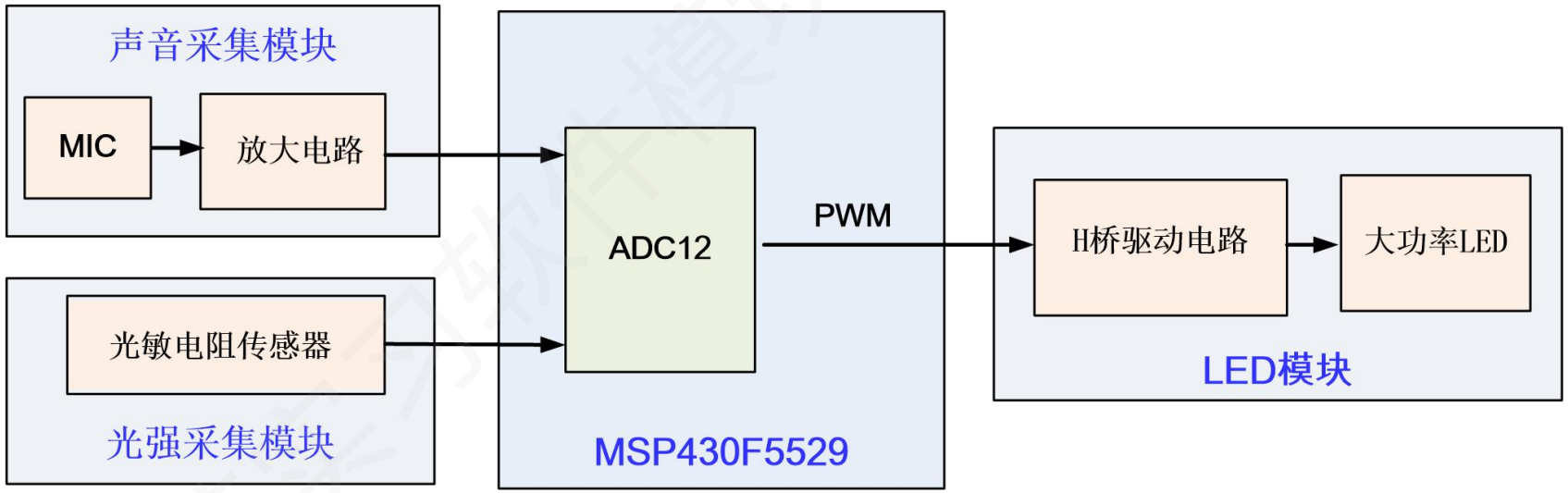


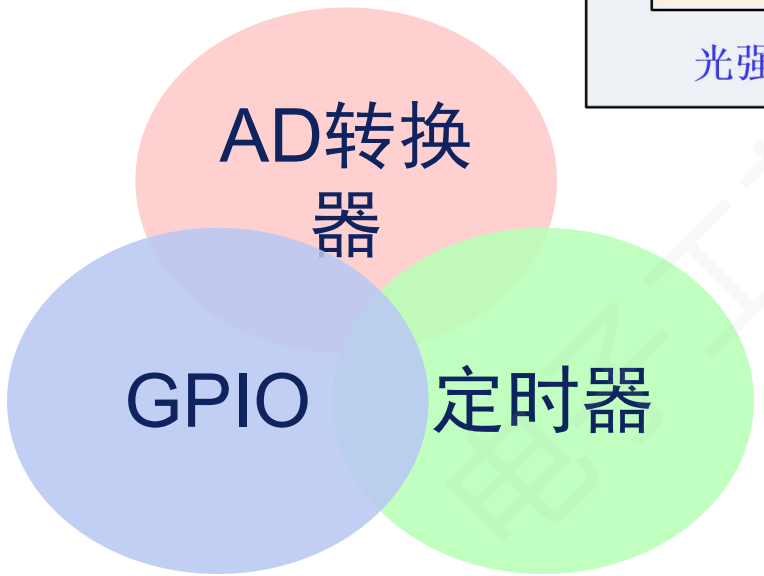






展馆灯光控制软件
——需求分析



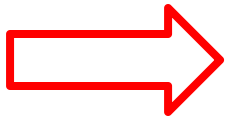


软件需求文档

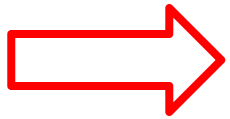
软件设计——概要设计

确定软件系统的基本框架

软件需求文档



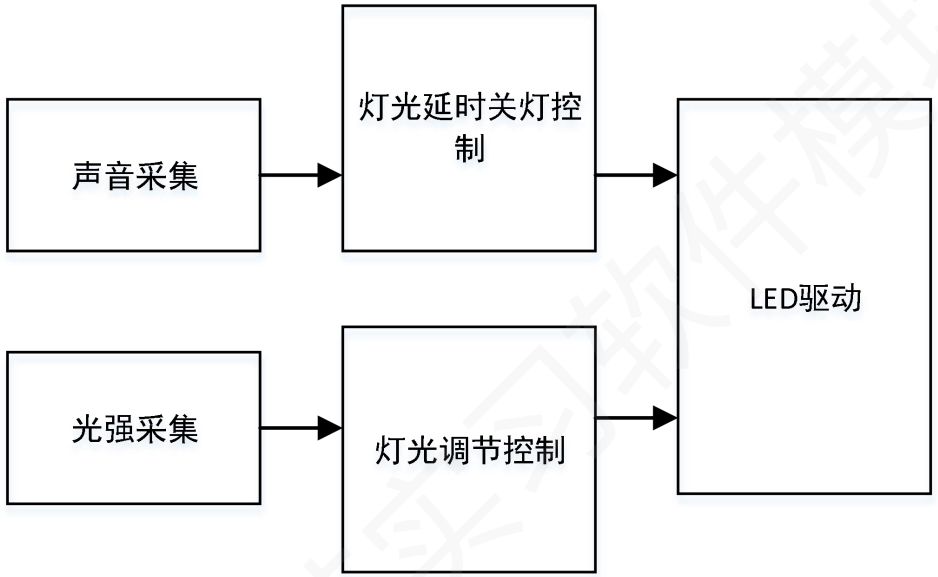
- 将采用什么样的体系构架？
- 需要创建哪些功能模块？
- 模块之间的关系如何？
- 数据结构如何？
- 安全性设计
- 故障处理设计
- 可维护性设计……



概要设计说明书

展馆灯光控制软件

概要设计
详细设计
编码
代码测试

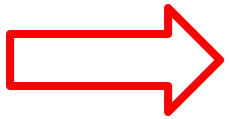


软件模块划分

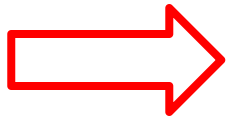
软件设计——详细设计

设计每个模块实现算法，所需的局部结构

概要设计说明书



- 给出每个模块的简要描述：功能、性能
- 确定模块输入输出接口的细节。
- 确定每一模块使用的数据结构。
- 每个模块确定采用的算法、流程逻辑：常选用**流程图**进行描述。
- 程序调用层次关系。

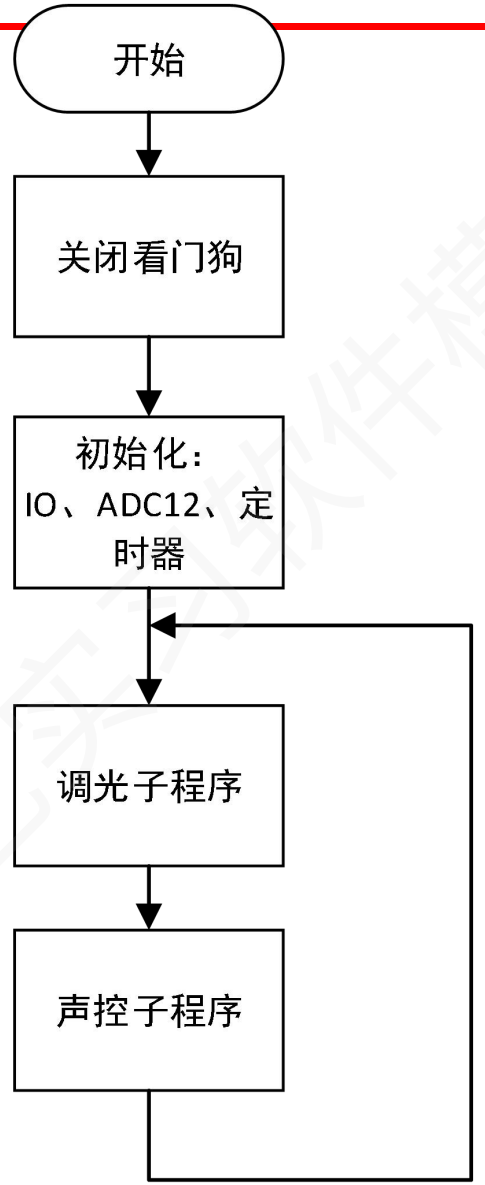


详细设计说明书

展馆灯光控制软件

概要设计
详细设计
编码
代码测试

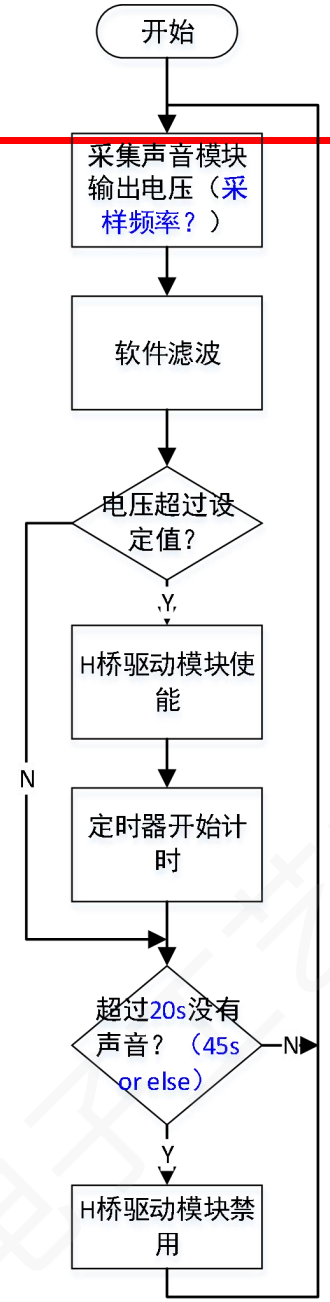
主程序

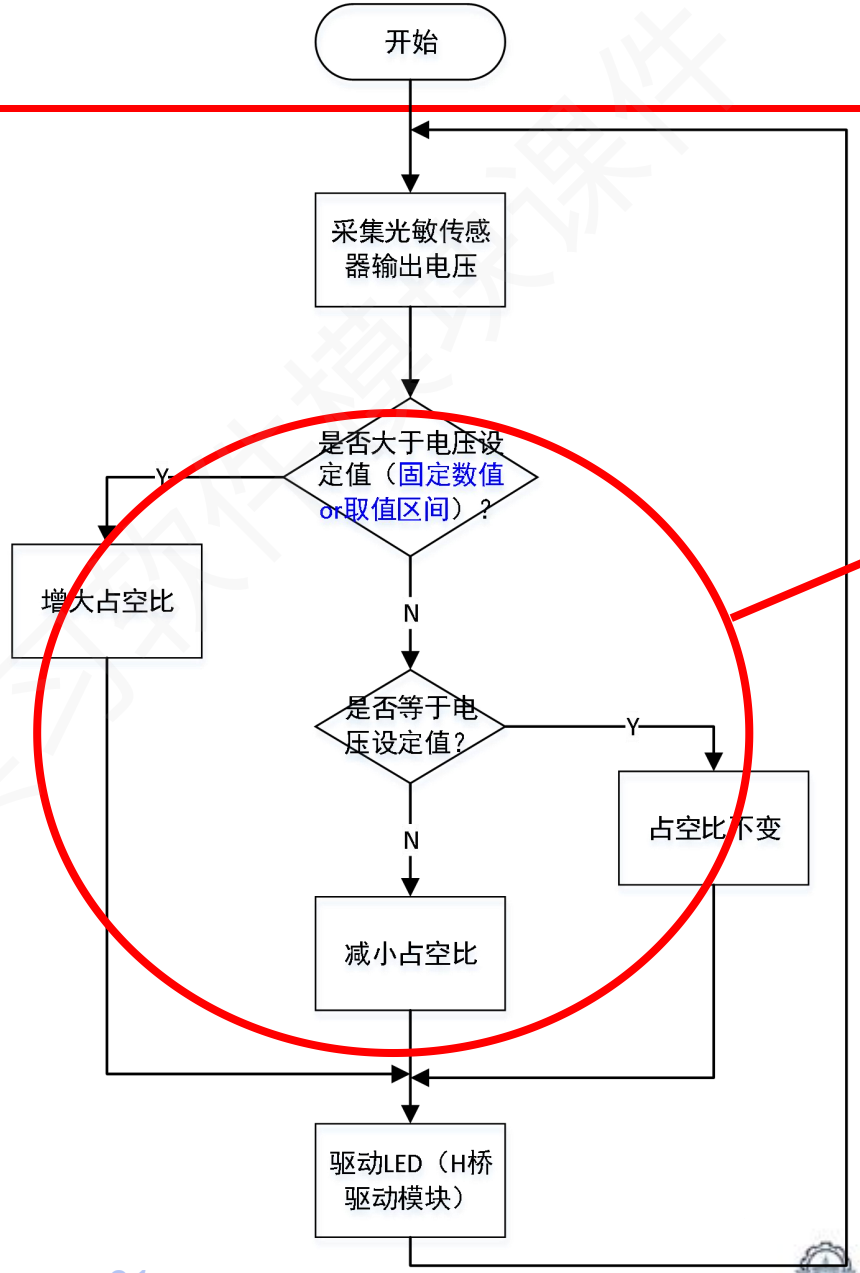


展馆灯光控制软件

概要设计
详细设计
编码
代码测试

声控子程序

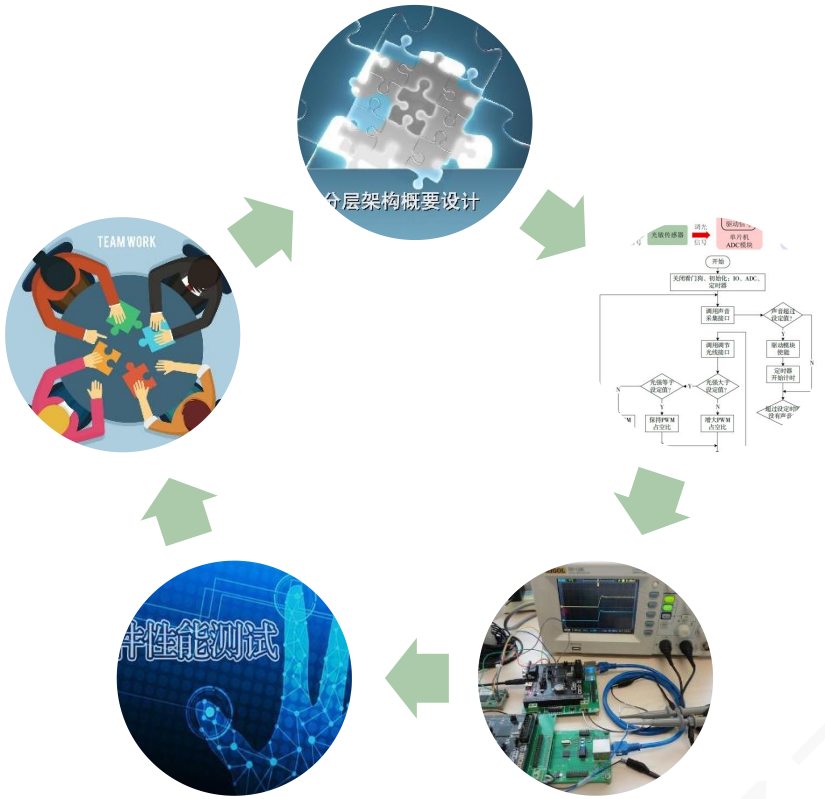




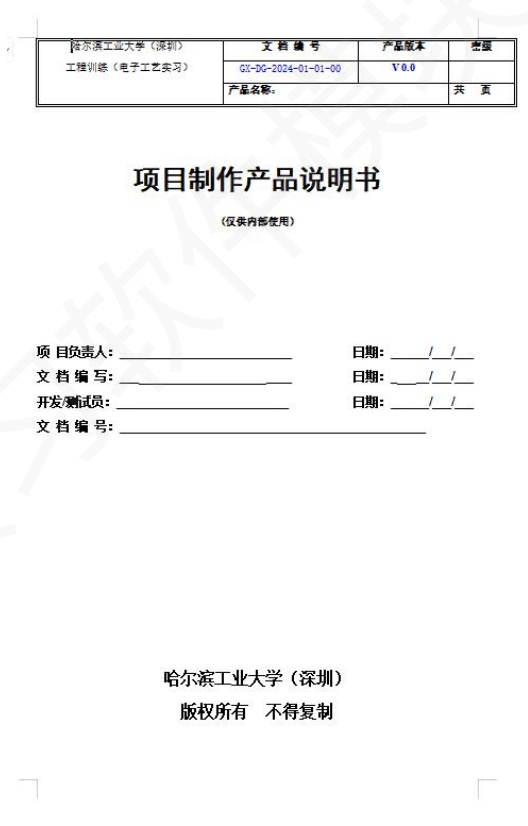
PID控制？

调光子程序

项目制作输出文档



输出



哈尔滨工业大学（深圳）

工程训练（电子工艺实习）

文档编号	产品版本	密级
GX-DG-2024-01-01-00	V 0.0	
产品名称：		共 页

项目制作产品说明书

（仅供内部使用）

项 目负责人：_____ 日期：____/__/__

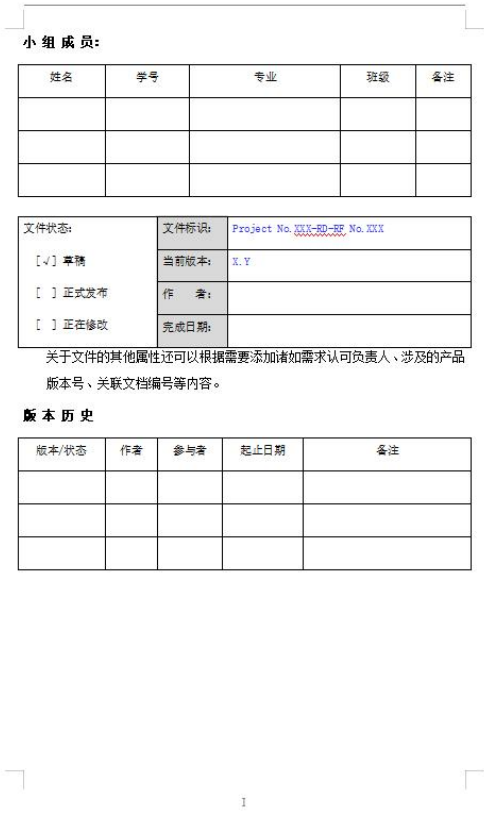
文 档 编 写：_____ 日期：____/__/__

开发/测试员：_____ 日期：____/__/__

文 档 编 号：_____

哈尔滨工业大学（深圳）

版权所有 不得复制



小 组 成 员：

姓名	学号	专业	班级	备注

文件状态：	文件标识：	Project No. XXX-RD-RF No. XXX
[√] 草稿	当前版本：	X.Y
[ ] 正式发布	作 者：	
[ ] 正在修改	完成日期：	

关于文件的其他属性还可以根据需要添加诸如需求认可负责人、涉及的产品版本号、关联文档编号等内容。

版 本 历 史

版本/状态	作者	参与者	起止日期	备注

嵌入式处理器

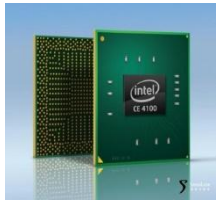
- 是控制、辅助系统运行的**硬件单元**。
- 最初4位处理器，发展到8位、16位单片机，再到最新的32位，64位嵌入式CPU。

MCU

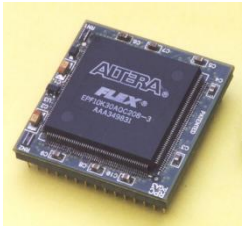
Micro Controller Unit
单片机

EMPU

EDSP



SoC



FPGA

CPU

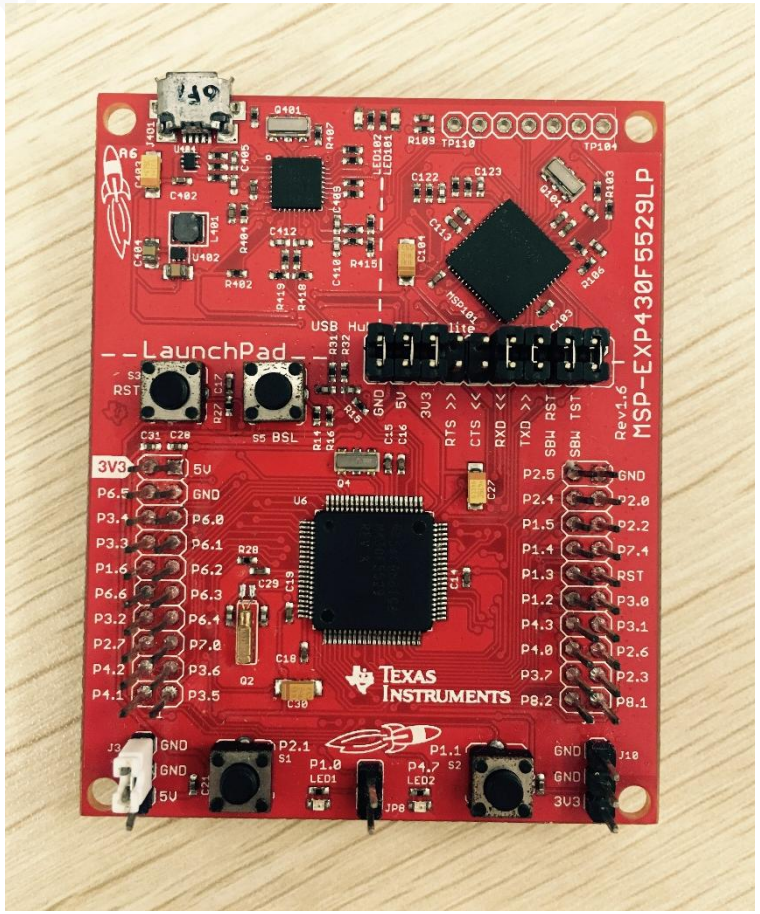
单片机是何方神圣？

MSP430F5529 硬件开发环境

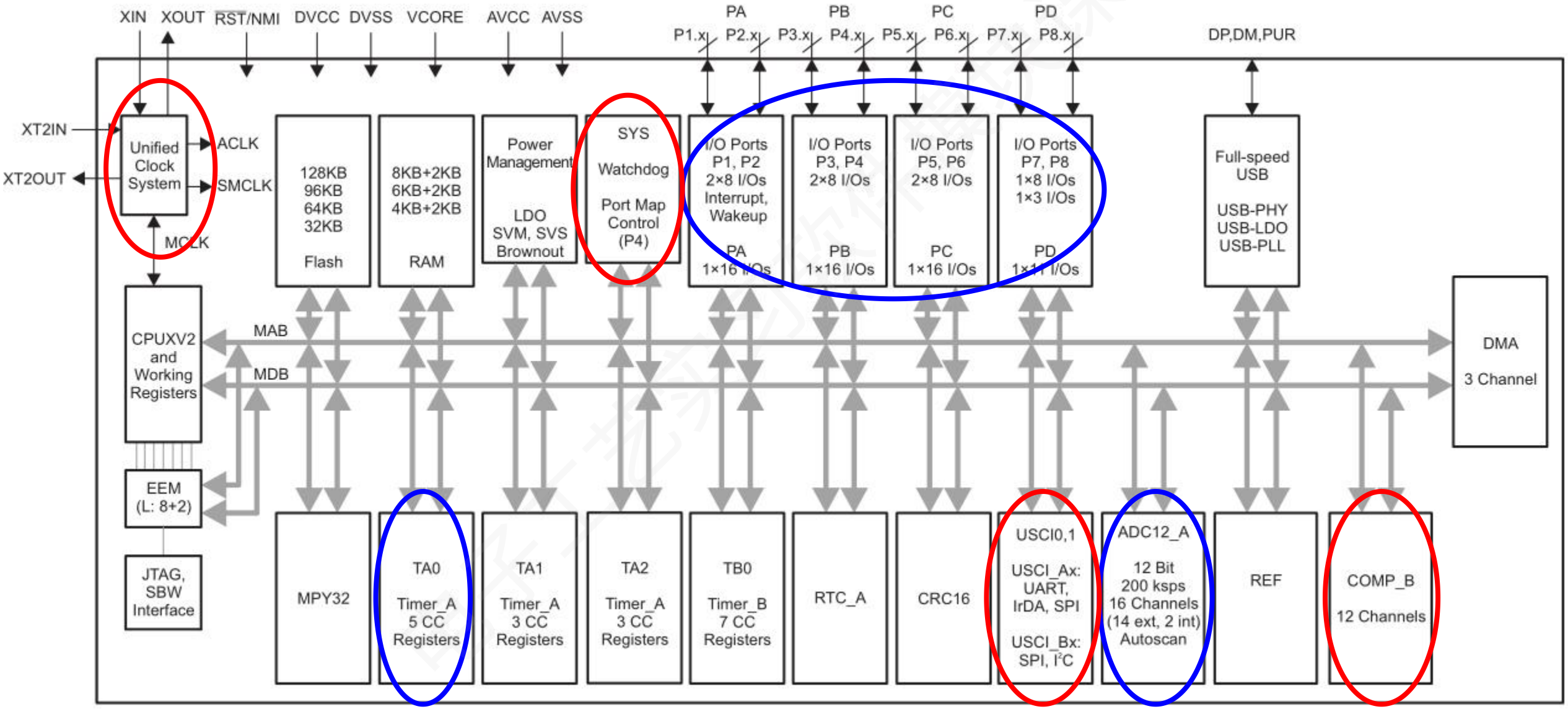
哈尔滨工业大学（深圳）
实验与创新实践教育中心

主要特性

- 16位单片机
- 超低功耗：激活模式，待机模式，
  关闭模式，关断模式
- 高达25MHz的系统时钟
- 统一时钟系统
- 4个16位定时器， 配有若干捕获/比较寄存器
- 2个通用串行通信口
- 全速通用串行总线
- 12位AD转换器



功能框图


XIN
XOUT
RST/NMI
DVCC
DVSS
VCORE
AVCC
AVSS
PA
PB
PC
PD
P1.x
P2.x
P3.x
P4.x
P5.x
P6.x
P7.x
P8.x
DP,DM,PUR
XT2IN
XT2OUT
Unified Clock System
ACLK
SMCLK
MCLK
128KB 96KB 64KB 32KB Flash
8KB+2KB 6KB+2KB 4KB+2KB RAM
Power Management LDO SVM, SVS Brownout
SYS Watchdog Port Map Control (P4)
I/O Ports P1, P2 2×8 I/Os Interrupt, Wakeup PA 1×16 I/Os
I/O Ports P3, P4 2×8 I/Os PB 1×16 I/Os
I/O Ports P5, P6 2×8 I/Os PC 1×16 I/Os
I/O Ports P7, P8 1×8 I/Os 1×3 I/Os PD 1×11 I/Os
Full-speed USB USB-PHY USB-LDO USB-PLL
CPUXV2 and Working Registers
MAB
MDB
DMA 3 Channel
EEM (L: 8+2)
JTAG, SBW Interface
MPY32
TA0 Timer_A 5 CC Registers
TA1 Timer_A 3 CC Registers
TA2 Timer_A 3 CC Registers
TB0 Timer_B 7 CC Registers
RTC_A
CRC16
USCI0,1 USCI_Ax: UART, IrDA, SPI USCI_Bx: SPI, I²C
ADC12_A 12 Bit 200 ksps 16 Channels (14 ext, 2 int) Autoscan
REF
COMP_B 12 Channels

MSP430F5529 实验开发系统

包含：

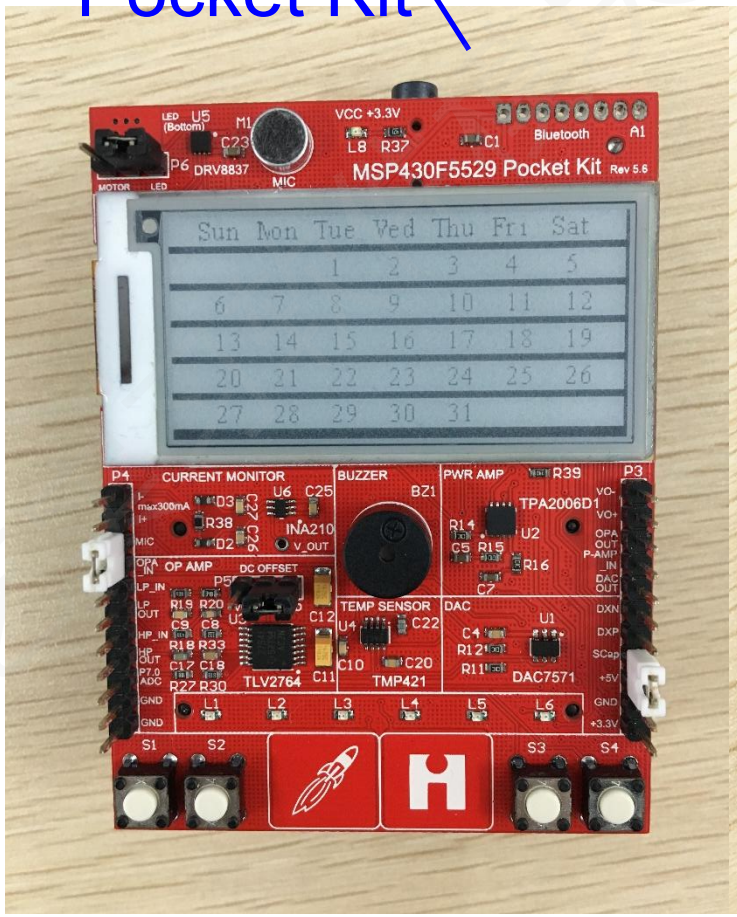
- MSP430F5529 主板
  （ MSP430F5529 LaunchPad ）
- MSP430F5529 口袋板
  （MSP430F5529 Pocket Kit）

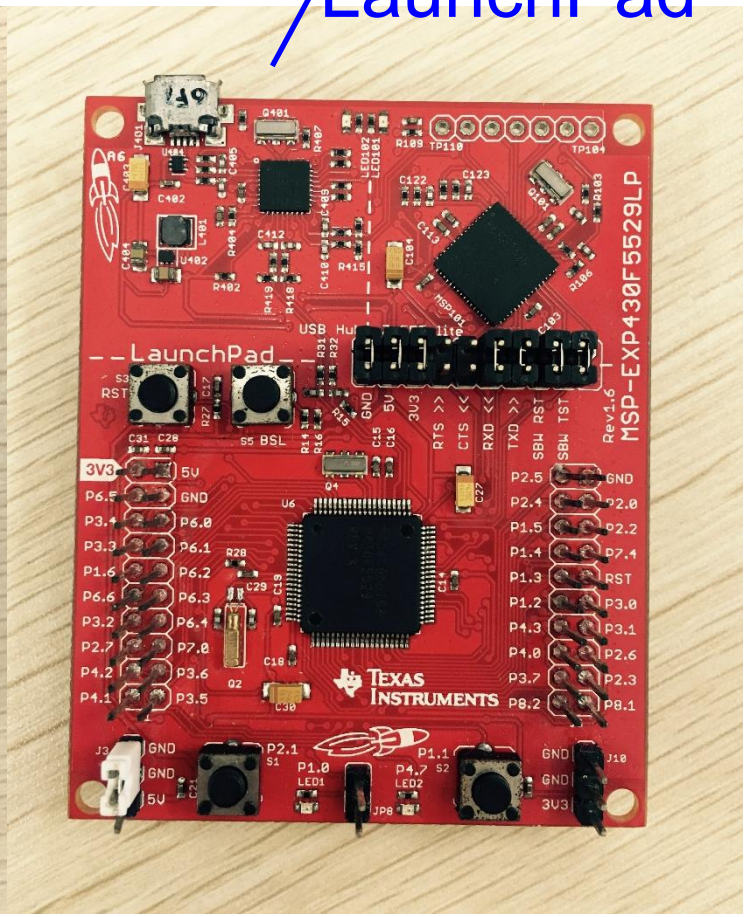
功能：

- 音频播放、音量调节、录音
- 温度测量
- 通信、屏幕显示
- 电压测量等

口袋板
Pocket Kit

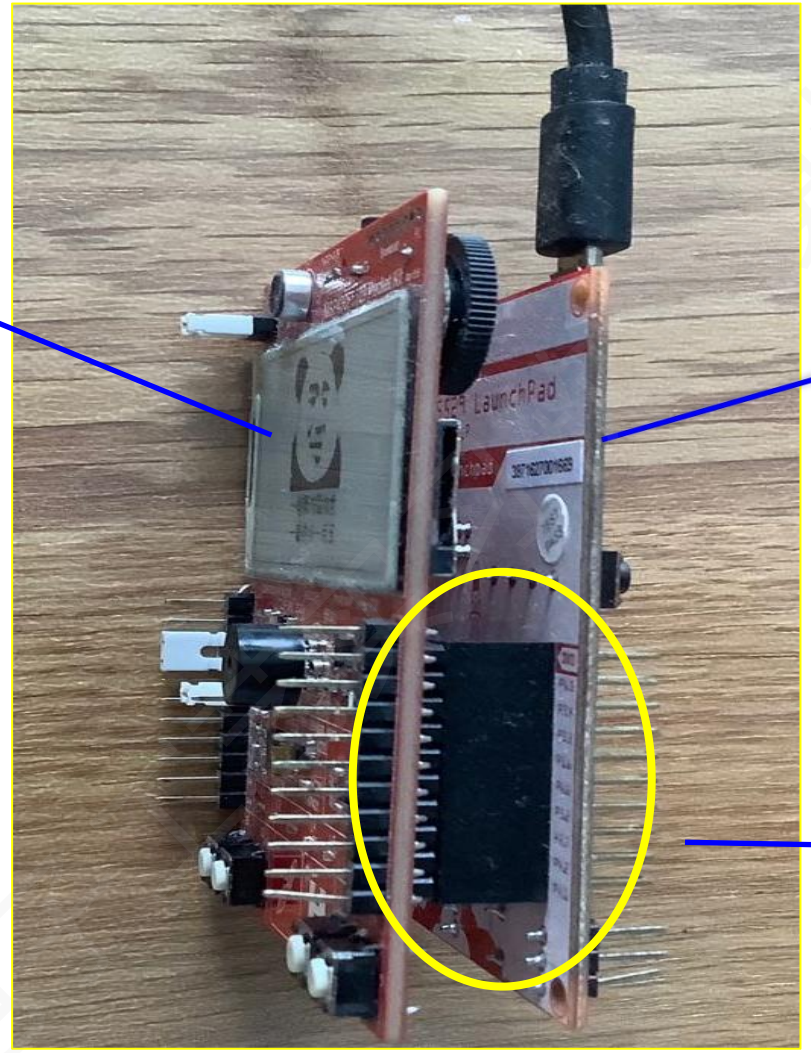
主板
LaunchPad





MSP430F5529 实验开发系统

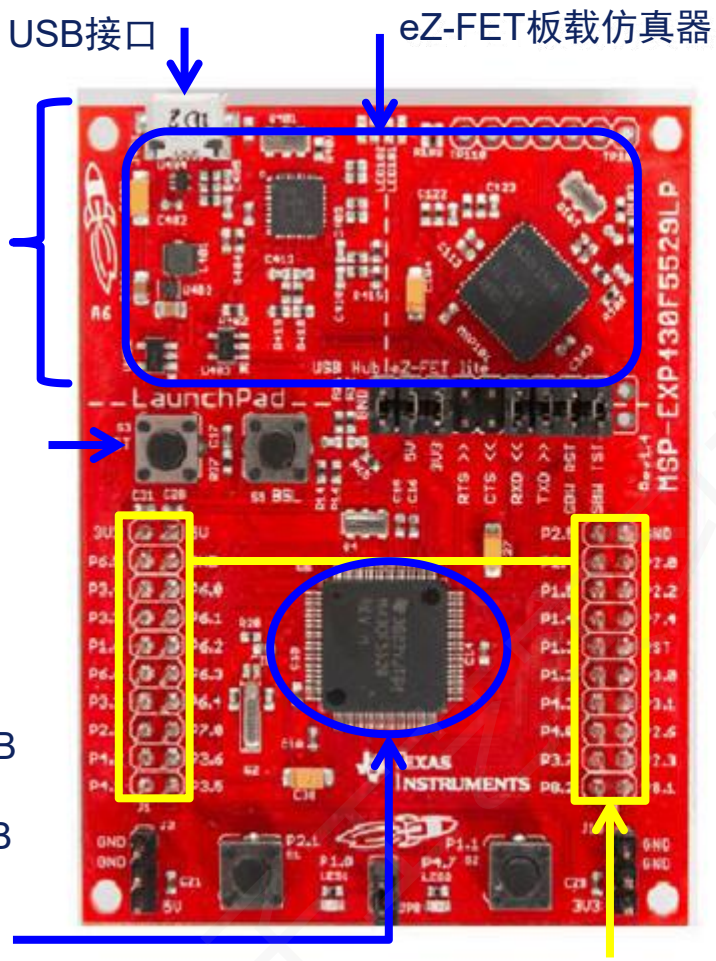
口袋板
Pocket Kit



主板
LaunchPad

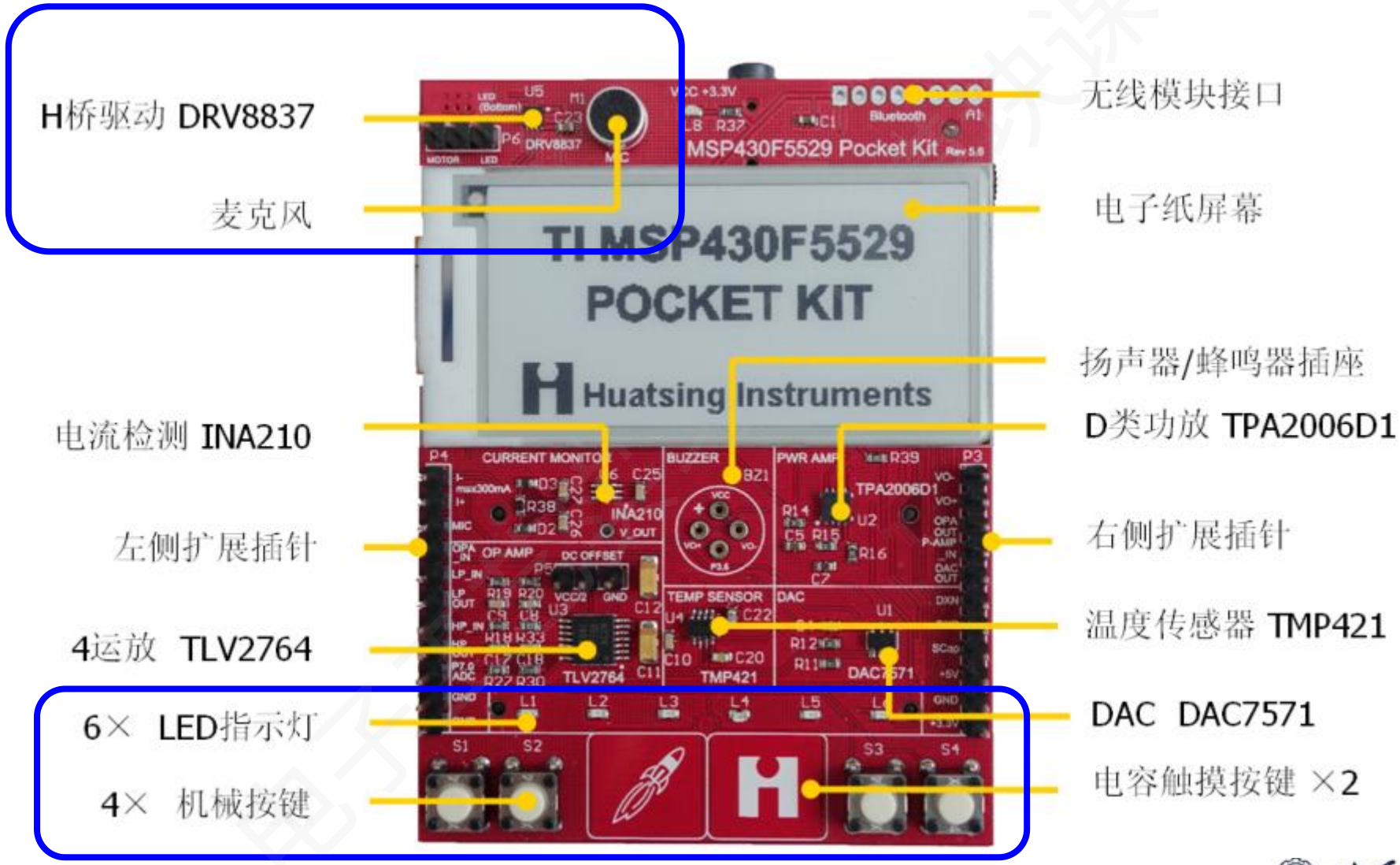
插针连接

主板

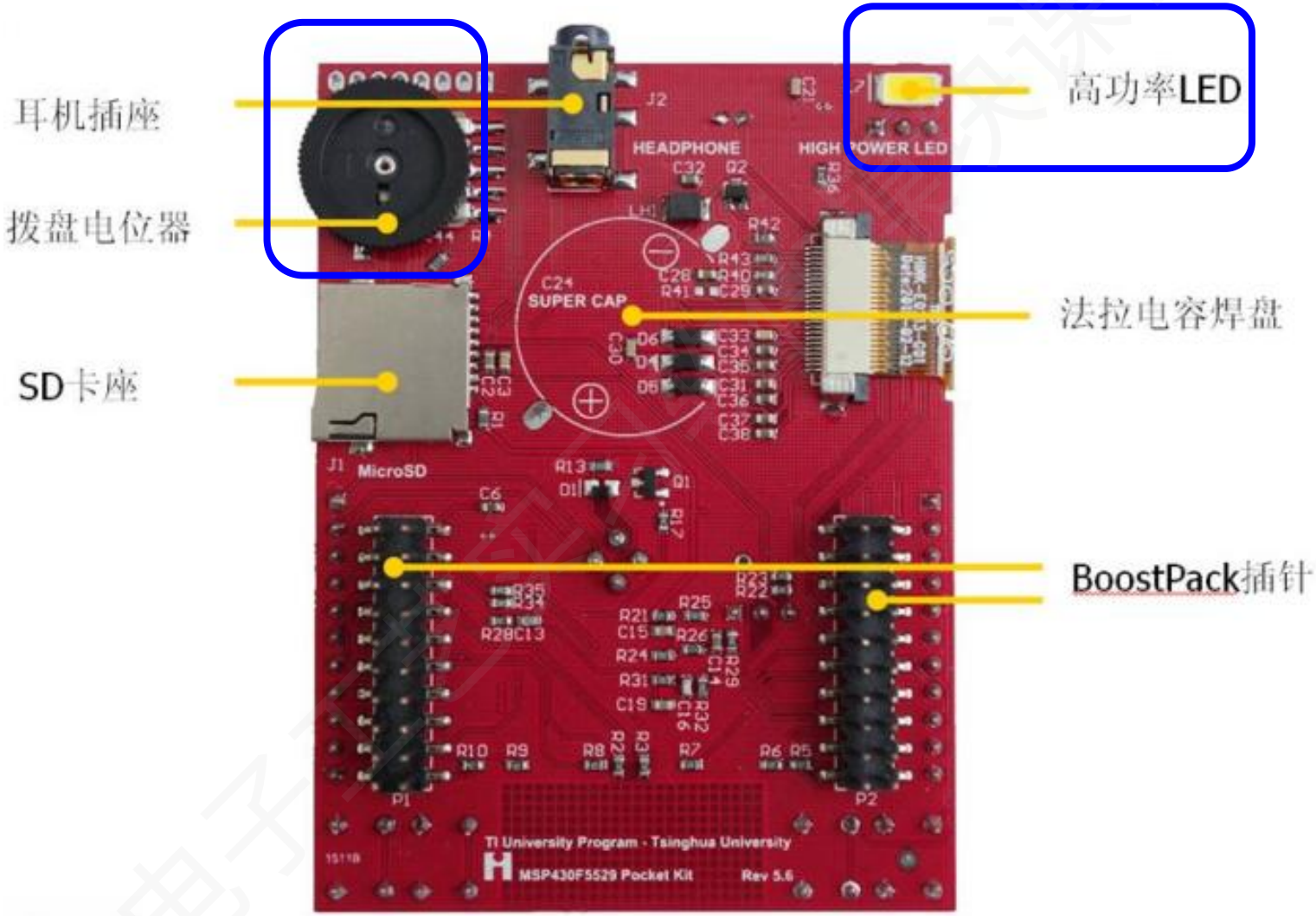


- 带USB的 MSP430F5529 16位单片机。
- eZ-FET板载仿真器，包含UART接口
- 带有USB Hub，可同时用USB接口仿真和开发USB应用程序
- USB供电，内置3.3V电压转换器
- 配备40引脚的BoosterPack
- 与40引脚的LaunchPad BoosterPack ™ 开发工具标准兼容

正面资源图

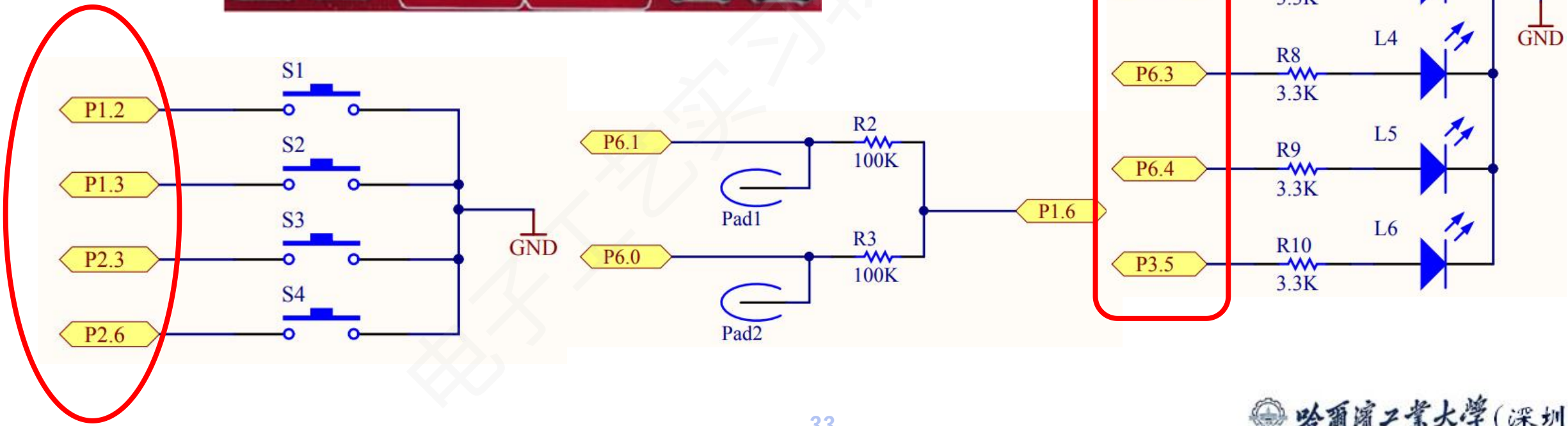


背面资源图



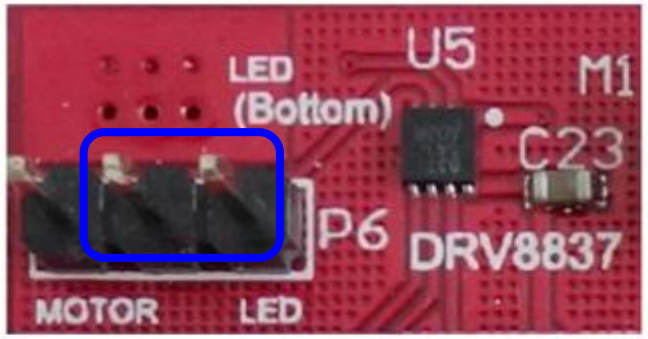
◆ 按键及LED灯

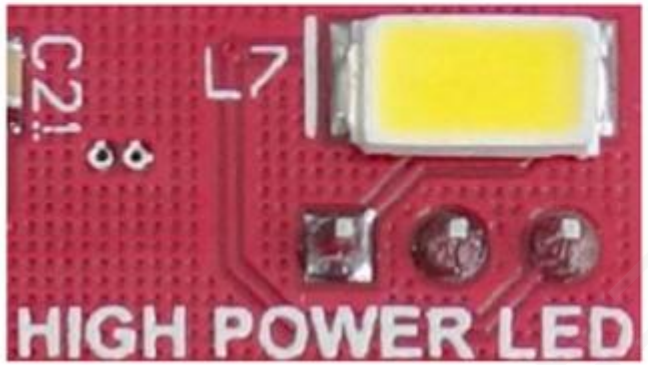
- ✓ 用户 LED 模块：6 个 LED 灯（ L1~L6 ）可供用户编程使用
- ✓ 用户按键模块：4个机械按键（S1~S4），2个电容触摸按键（Pad1~Pad2）可供用户编程使用

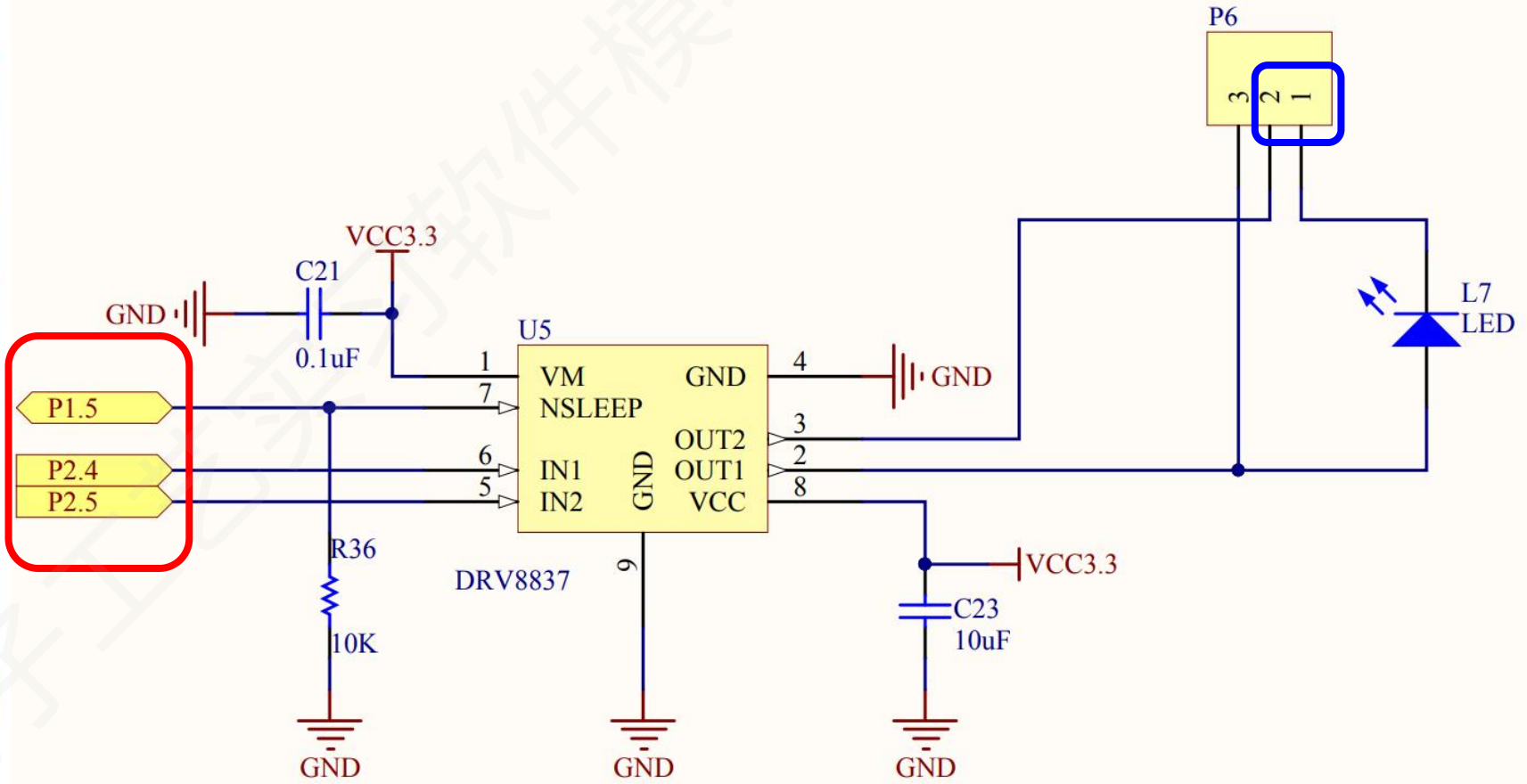


◆ H桥驱动和高功率LED

口袋板正面左上角是低压 H 桥驱动器 DRV8837，用它来驱动口袋板背面的高功率 LED（L7），也预留出了电机接口，可以用来驱动扩展的电机模块。

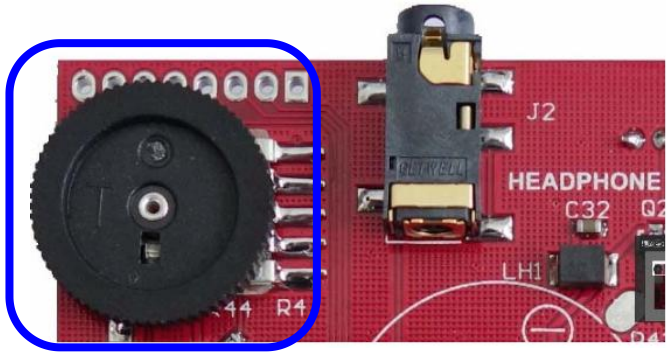


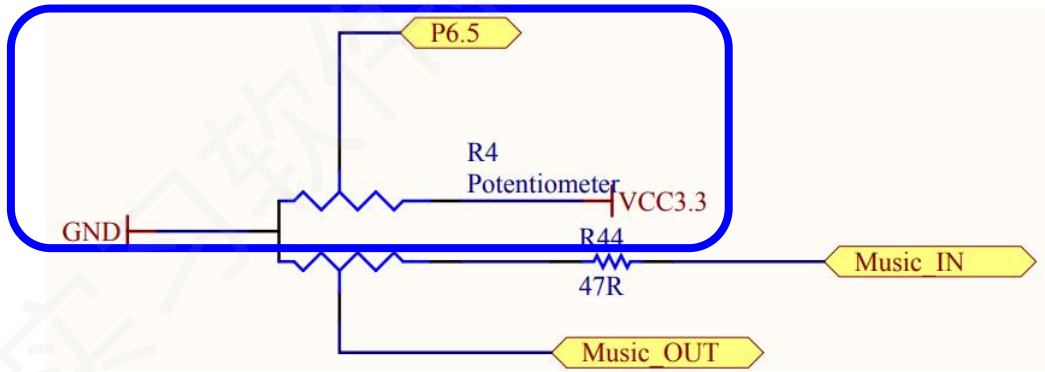




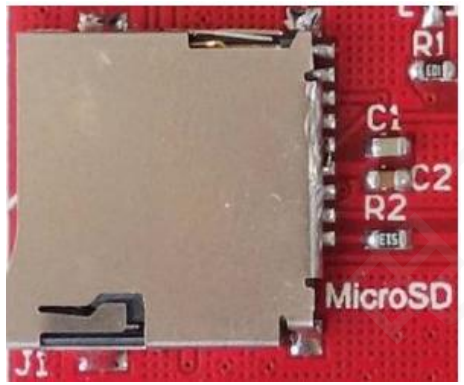
拨盘电位器/耳机插座/SD卡座

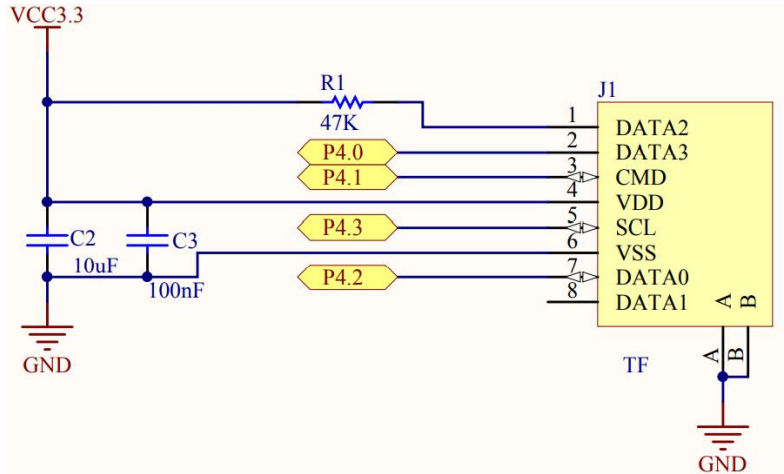
- ✓ 口袋板背面设置了一个双路 50KΩ 的拨盘电位器，一个抽头连接到了 F5529 的 ADC端（P6.5），另一个抽头连接到 TPA2006D1 芯片的输入端，同时也连接到耳机插座的输出端，起到对信号的衰减作用。
- ✓ 耳机插座在口袋板背面上方，可以连接标准 3.5mm 耳机插头，由于整个信号链路都是单声道的，因此将左右声道并接到了一起。该插座也可以做 Line Out 接口使用，输出的信号电压受拨盘电位器控制。





- ✓ 1 个 MicroSD 卡座，我们可以在 SD 卡中存放音频或图像文件，然后通过口袋板进行播放或显示。





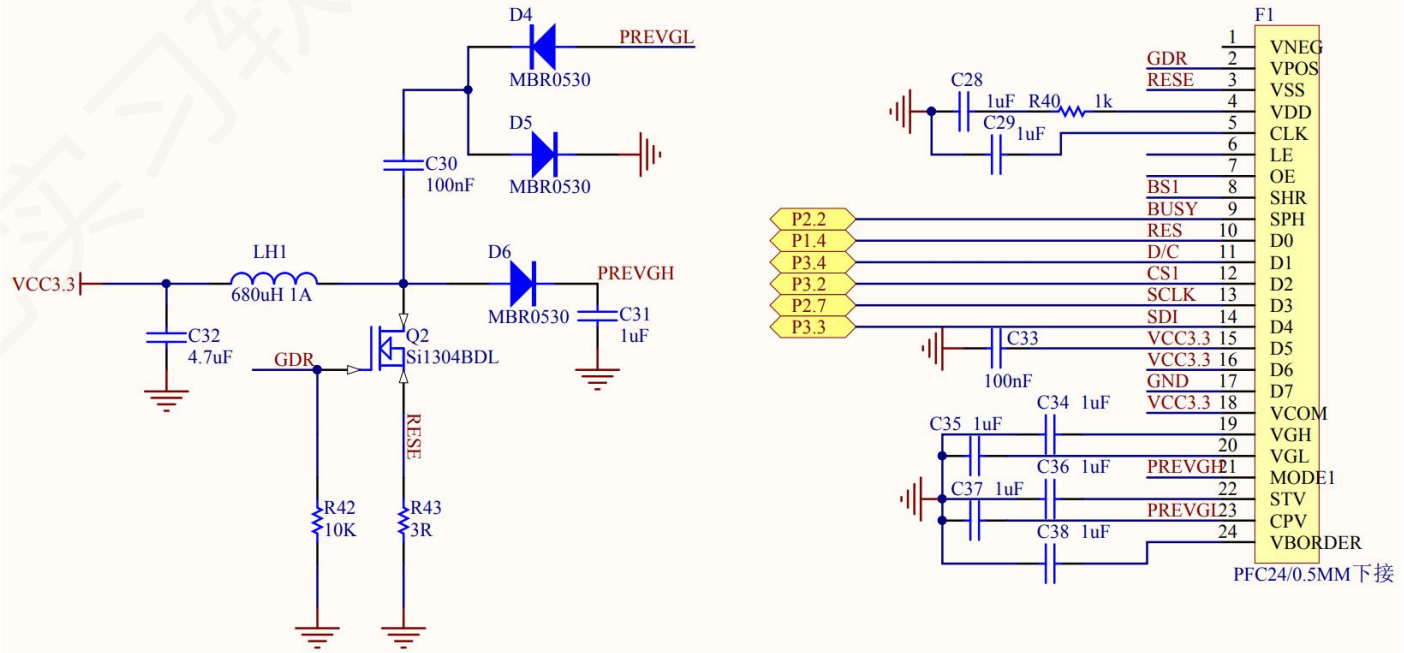
电子纸屏幕

2.1英寸的电子纸屏幕（电子墨水屏），分辨率250×122，SPI接口

超级省电—只有在屏幕刷新时消耗电能，维持显示无需供电

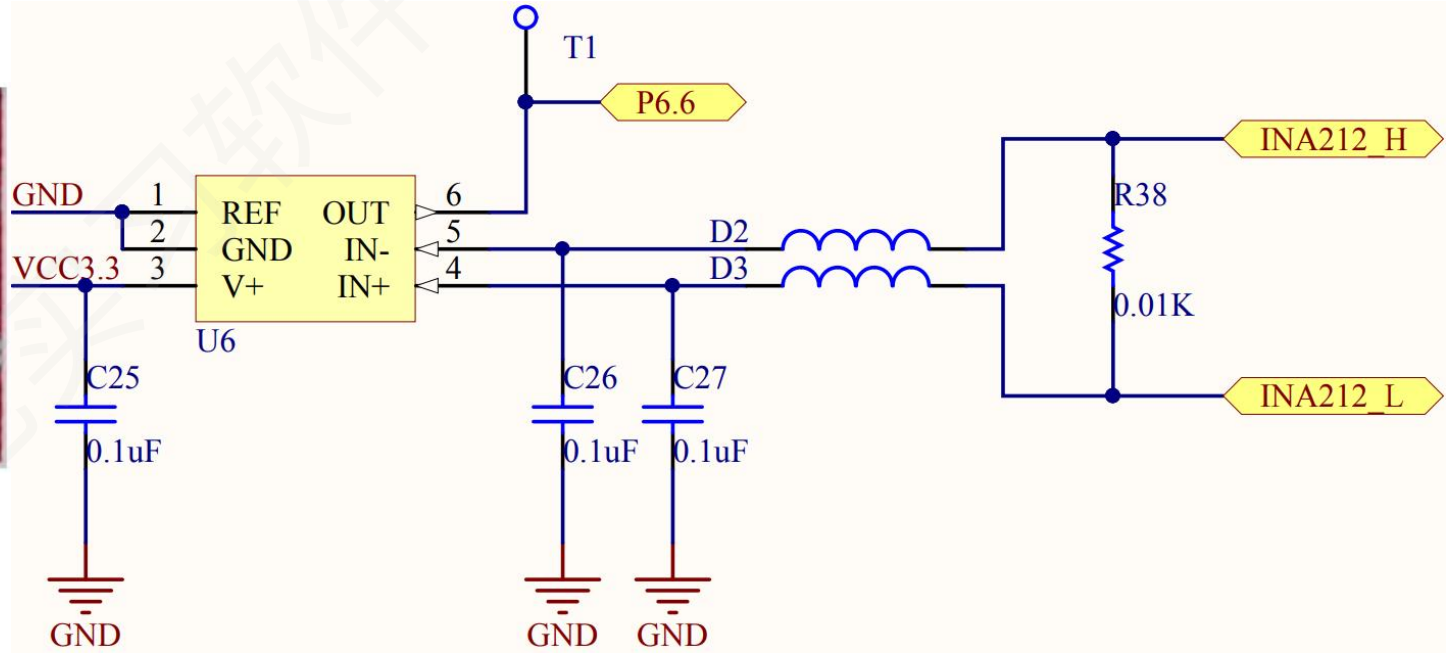
易阅读性—反射环境光来显示图案，强烈的阳光下依然清晰可视，视角度几乎达到180°，具有传统印刷品显示效果

轻薄灵活—最薄可以做到0.1mm，和纸张的厚度差不多，可制成可弯曲型屏幕



电流测量单元

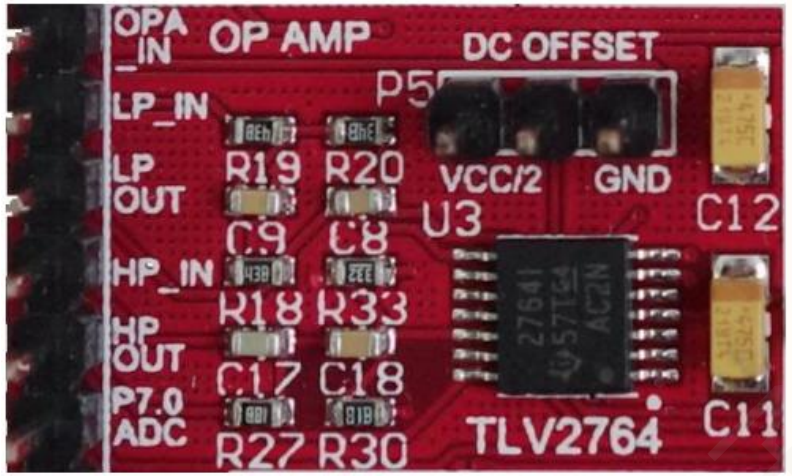
- ✓ 在口袋板屏幕下方就是电压输出型电流检测芯片 INA210，将模块串联接入电路中,就会测量出电路中的电流，输入电流与输出电流分别接入左侧扩展插针 P4.2（I+）与 P4.1（I-）端，最大输入电流不可超过 300mA。

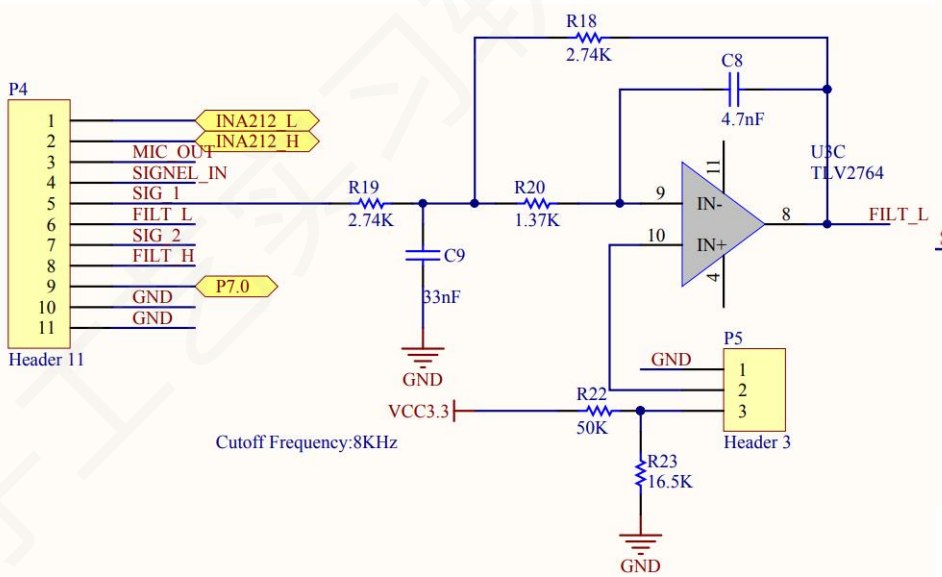


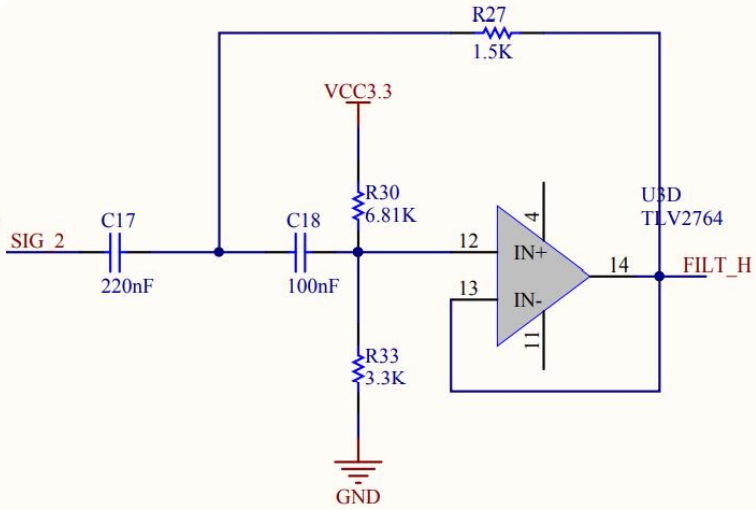
滤波器与前级放大单元

TLV2764 四运放模块，内部集成了 4 个独立的运算放大器。

- ✓ 一个二阶有源低通滤波器：多重反馈型（multi-feedback）型，截止频率是 8KHz
- ✓ 一个二阶有源高通滤波器： Sallen-Key 结构，截止频率 500Hz
- ✓ 一个对小信号进行电压放大的前级放大器，可对口袋板上麦克风输出的微小信号进行前级放大。

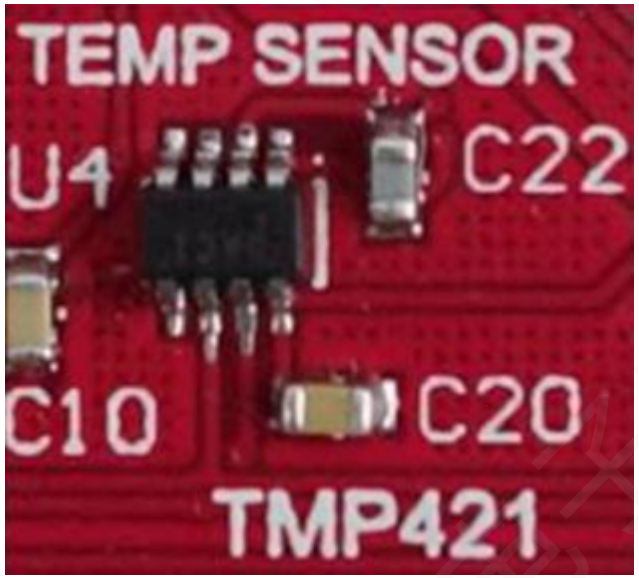


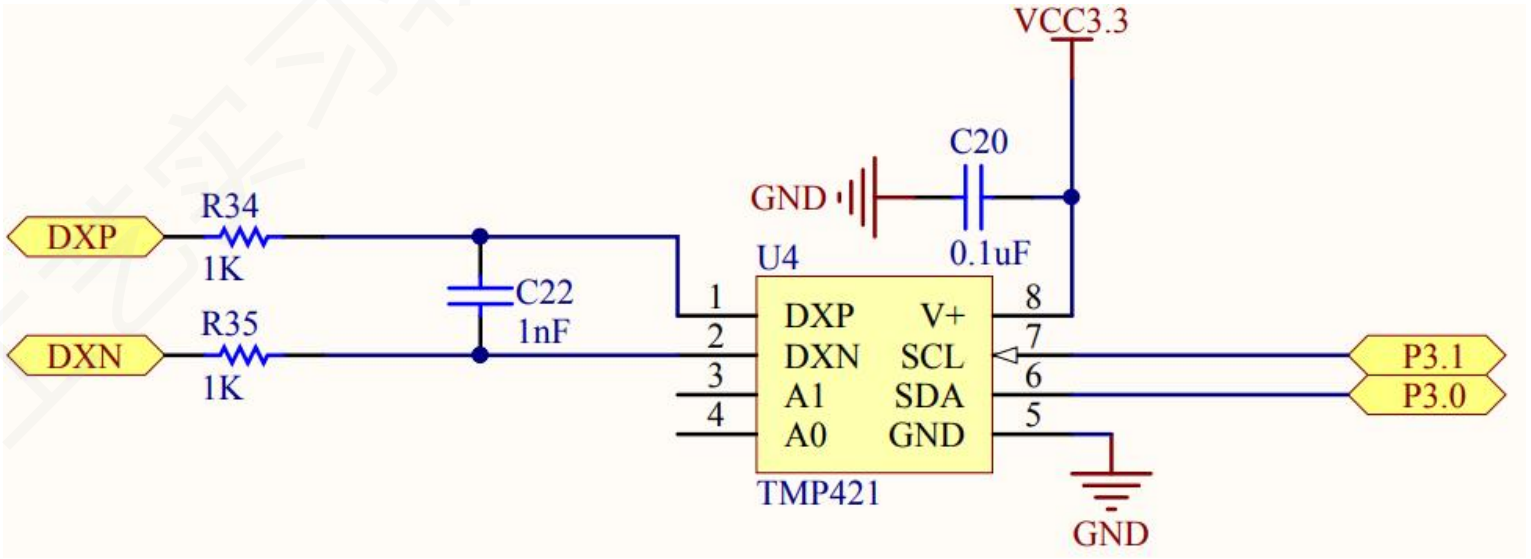




温度传感器单元

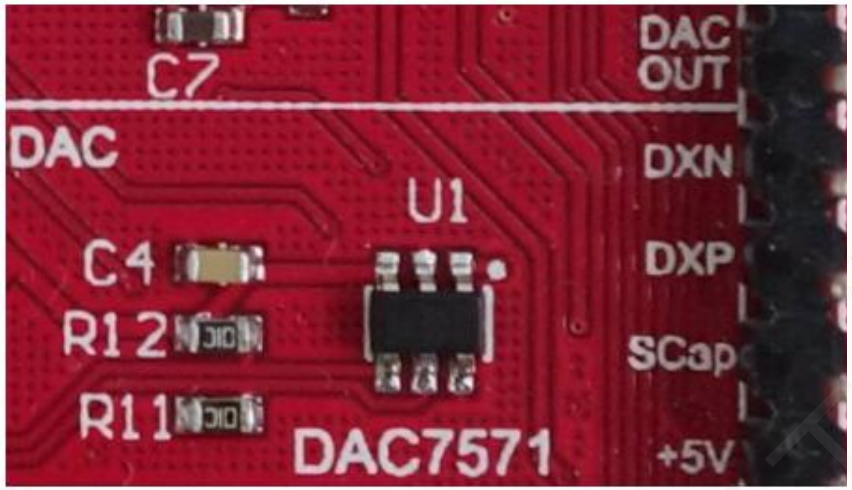
- ✓ IIC 总线的双温区数字温度传感器芯片 TMP421，该芯片能够测量本地（Local）温度（芯片端的温度）与远程（Remote）温度；
- ✓ 测量远程温度需要将一段 2-pin等长的杜邦线一端连接在右侧扩展接口 P3.6（DXN）与 P3.7（DXP）上，另一端按一定规则连接一个指定型号的三极管，三极管端的温度即为远程温度。

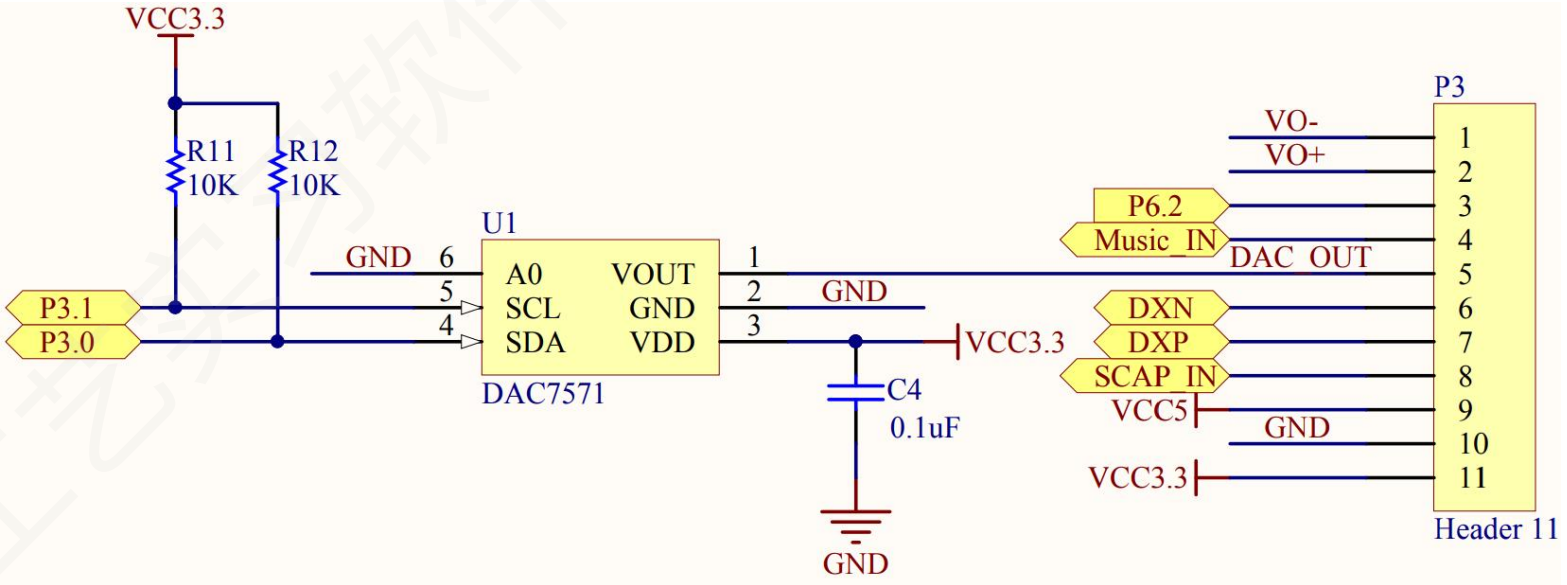




DAC单元

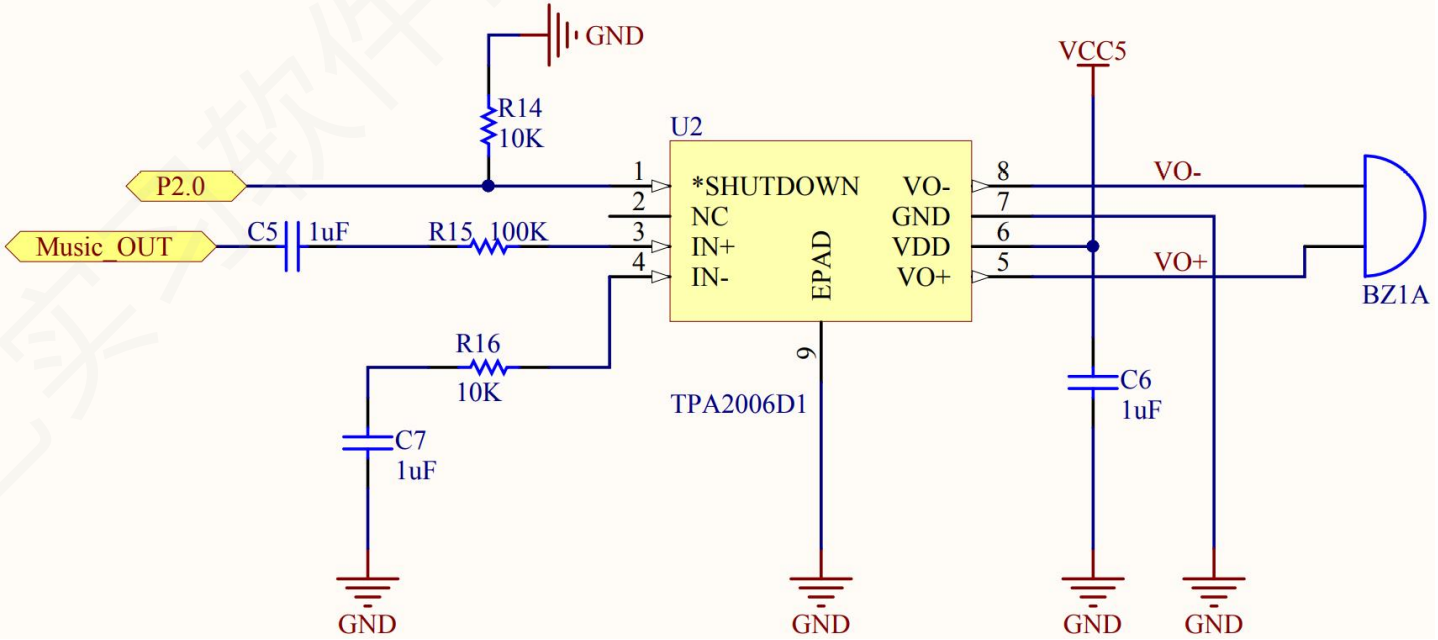
- ✓ DAC7571是低功耗，单通道12位DA转换器，I2C接口，时钟的最高速度3.4Mbps。利用DAC7571可实现音频信号播放





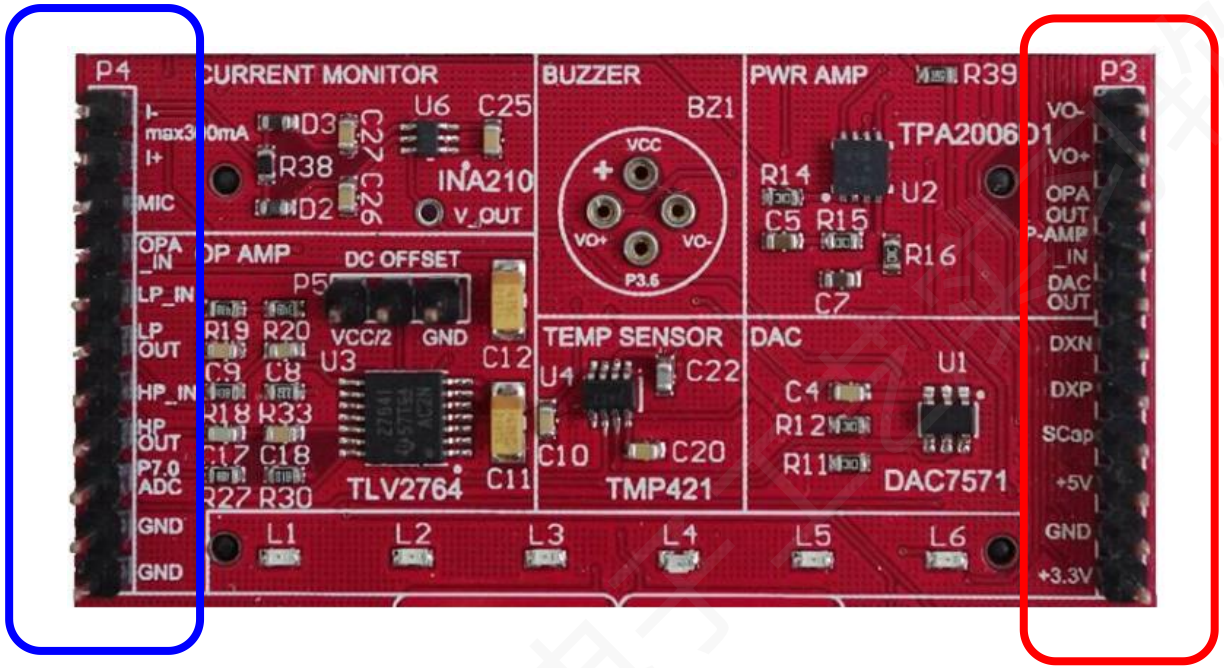
音频功放单元

- ✓ 利用TPA2006 D类功放，可与前级放大电路、低通、高通滤波器、DAC单元配合，实现音频信号的处理和放大流程



口袋板接口说明

- ✓ 口袋板背面的 BoosterPack 接口用于和MSP430F5529LaunchPad 进行连接，两块板子采用背对背的连接方式
- ✓ 口袋板正面中间两侧有两列单排插针作为一些信号端口使用，我们可以通过杜邦线将信号引入或引出。



信号接口（P4、P3）引脚定义见下：

位置	序号	定义	说明
P4（左侧）	1	I-	电流测试输入负端与正端，最大允许通过电流 300mA
	2	I+	
	3	MIC	麦克风输出信号
	4	OPA_IN	前级放大器输入信号
	5	LP_IN	低通滤波器输入端
	6	LP_OUT	低通滤波器输出端
	7	HP_IN	高通滤波器输入端
	8	HP_OUT	高通滤波器输入端
	9	P7.0/ADC	ADC 输入端，最大允许输入电压 3.3V
	10	GND	地信号，可以作为跳线帽收纳端子使用
	11	GND	
P3（右侧）	1	VO-	功放芯片 BTL 输出负端
	2	VO+	功放芯片 BTL 输出正端
	3	OPA OUT	前级放大器输出端
	4	P-AMP_IN	功率放大器输入端
	5	DAC OUT	DAC 输出端
	6	DXN	远程温度信号连接端
	7	DXP	远程温度信号连接端
	8	SCap	法拉电容正极
	9	+5V	+5V 信号
	10	GND	地信号
	11	+3.3V	+3.3V 信号

MSP430F5529 软件开发环境

哈尔滨工业大学（深圳）
实验与创新实践教育中心

安装包下载

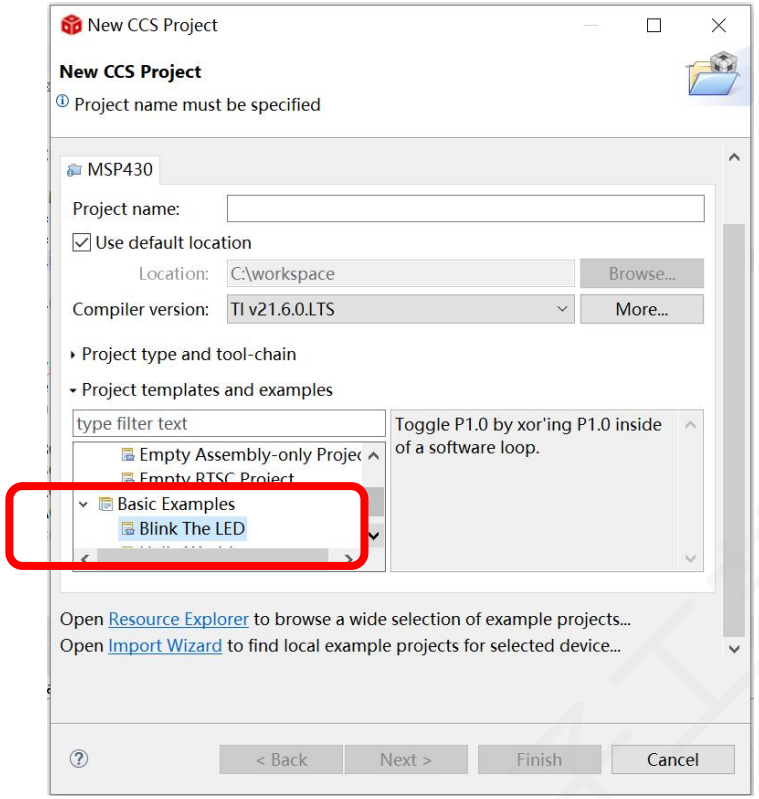
CCS(Code Composer Studio)：一种集成开发环境 (IDE)，支持 TI 的微控制器和嵌入式处理器产品系列。具有环境配置、源文件编辑、程序调试、跟踪和分析等功能。帮助用户在一个软件环境下完成编辑、编译、链接、调试和数据分析等工作。

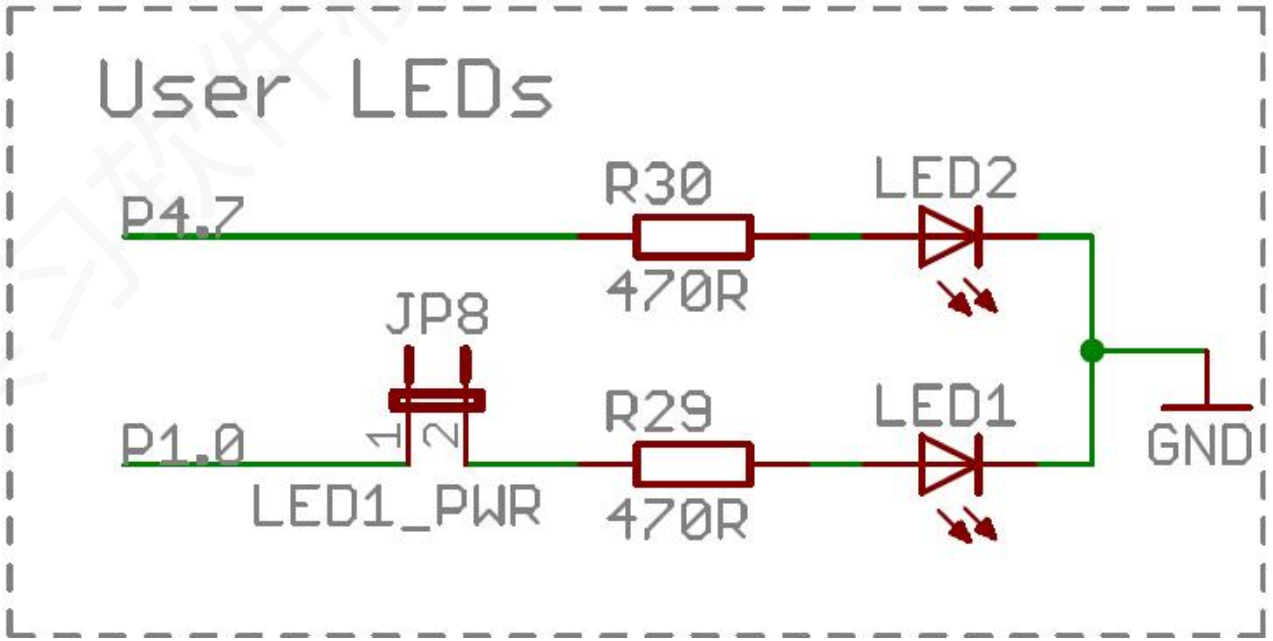
- **目前版本已更新到CCS12.7.1，可以在官网下载安装包，选择Off-line Installers（安装时不需网络环境）或者On-line Installers（安装时需要网络环境）下的windows系统安装包。**
- **注意：所有名称（计算机名和用户名）和路径全部为非中文字符。**

下载路径：*http://software-dl.ti.com/ccs/esd/documents/ccs_downloads.html*

熟悉软件环境

动手尝试建立Basic Example工程并调试，Blink the LED。





回顾

进制的简化符号

进 制	符 号	数 码
二 进 制	B(Binary)	0~1
八 进 制	(Octal)	0~7
十 进 制	Decimal)	0~9
十六进制	H(hexadecimal)	0~9,A~F

将八进制数 307 转换成二进制数是： [填空1]

O->D->B　　307O=3 x 8 x 8 + 7 = 199D = 1100 0111B

八进制的一位可以用二进制的三位表示

3　0　7　O

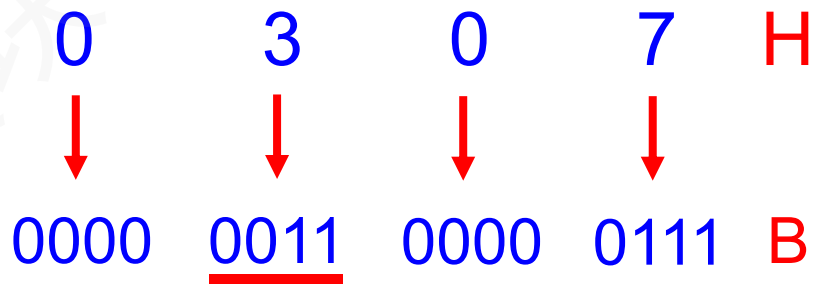
↓　↓　↓

011　000　111　B

答案： 1100 0111B

进制的简化符号

进 制	符 号	数 码
二 进 制	B(Binary)	0~1
八 进 制	(Octal)	0~7
十 进 制	Decimal)	0~9
十六进制	H(hexadecimal)	0~9,A~F



写法：

0x1314

0x66

88H

运算符

操作符	说明
&&	逻辑与
||	逻辑或
!	逻辑非
&	按位与
|	按位或
^	按位异或
~	按位取反
>>	右移
<<	左移

```
P1OUT=P1OUT|BIT0;

P1OUT|=BIT0;

P1OUT&=~BIT0;

P1OUT^=BIT0;

P1IN&BIT0
```

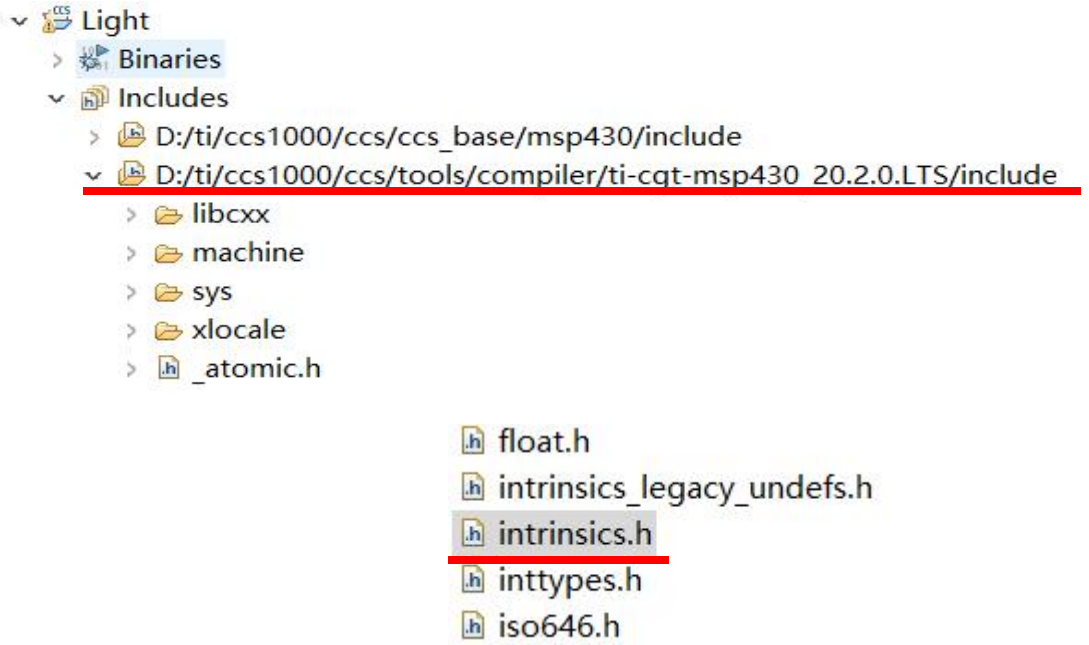
```
#define BIT0    (0x0001)
#define BIT1    (0x0002)
#define BIT2    (0x0004)
#define BIT3    (0x0008)
#define BIT4    (0x0010)
#define BIT5    (0x0020)
#define BIT6    (0x0040)
#define BIT7    (0x0080)
```

执行次数	PxOUT	BIT0
0	0100 1110	0000 0001
1	0100 1111	0000 0001
2	0100 1110	0000 0001

与（&）	0 & 0 = 0	1 & 0 = 0	0 & 1 = 0	1 & 1 = 1
或（|）	0 | 0 = 0	1 | 0 = 1	0 | 1 = 1	1 | 1 = 1
异或（^）	0 ^ 0 = 0	1 ^ 0 = 1	0 ^ 1 = 1	1 ^ 1 = 0

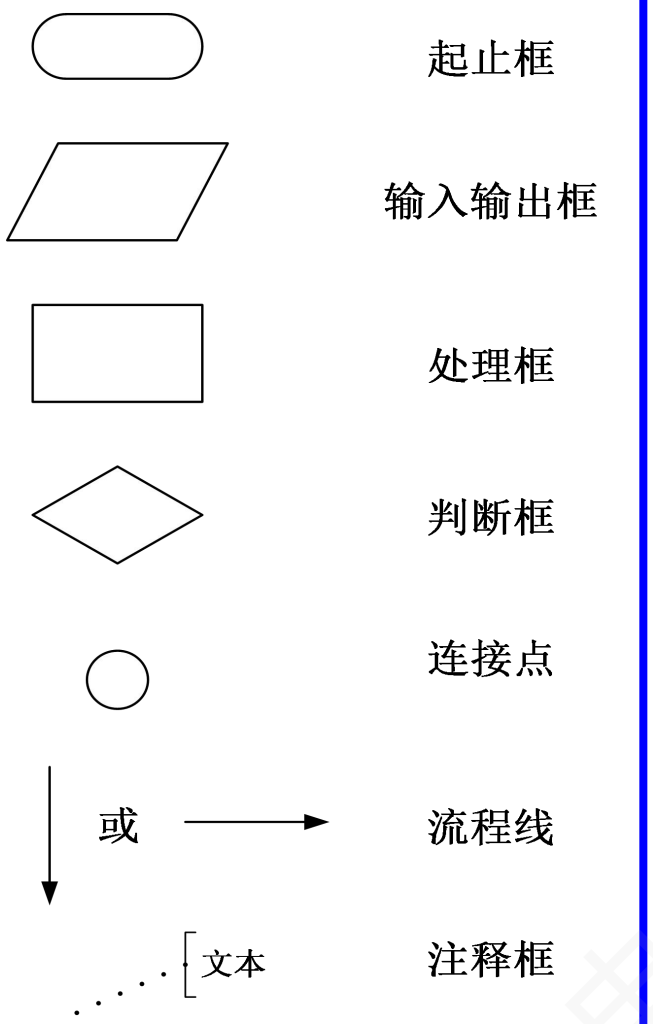
常用的系统函数

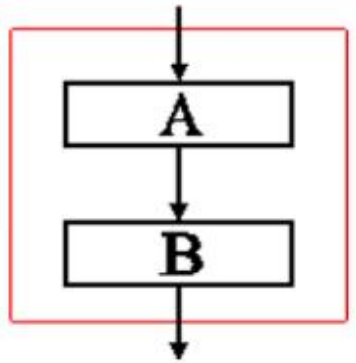
```
#define _nop()                  __no_operation()      //空操作
#define _no_operation()         __no_operation()
#define _enable_interrupt()     __enable_interrupt()  //开全局中断
#define _enable_interrupts()    __enable_interrupt()
#define _disable_interrupt()    __disable_interrupt()  //关全局中断
#define _disable_interrupts()   __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
```



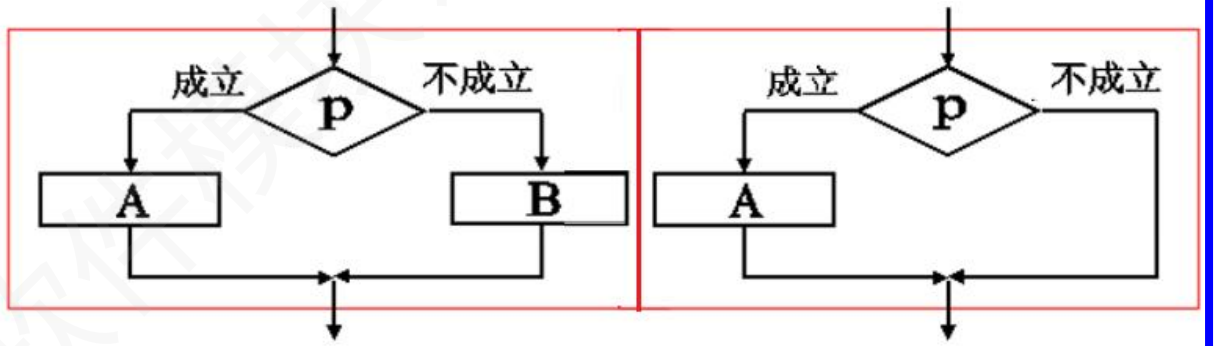
程序流程图

- 用标准的图形符号表示程序算法，能够体现其处理顺序与逻辑关系。

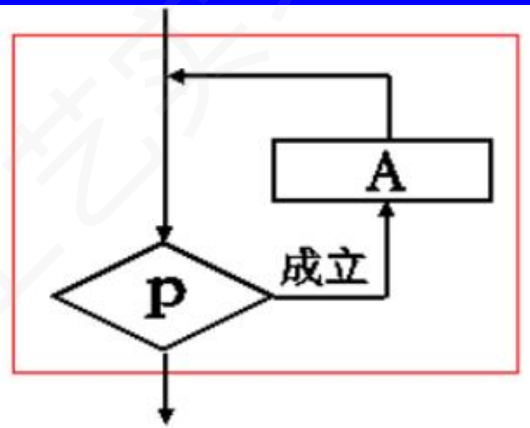


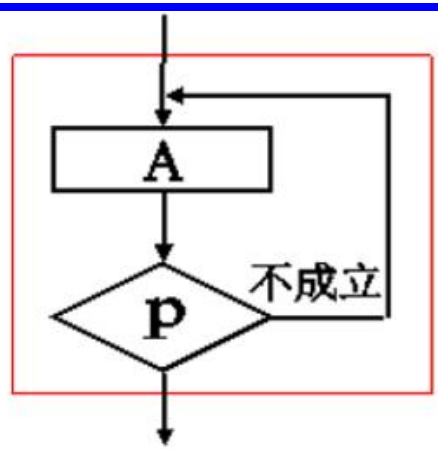


顺序结构



选择结构



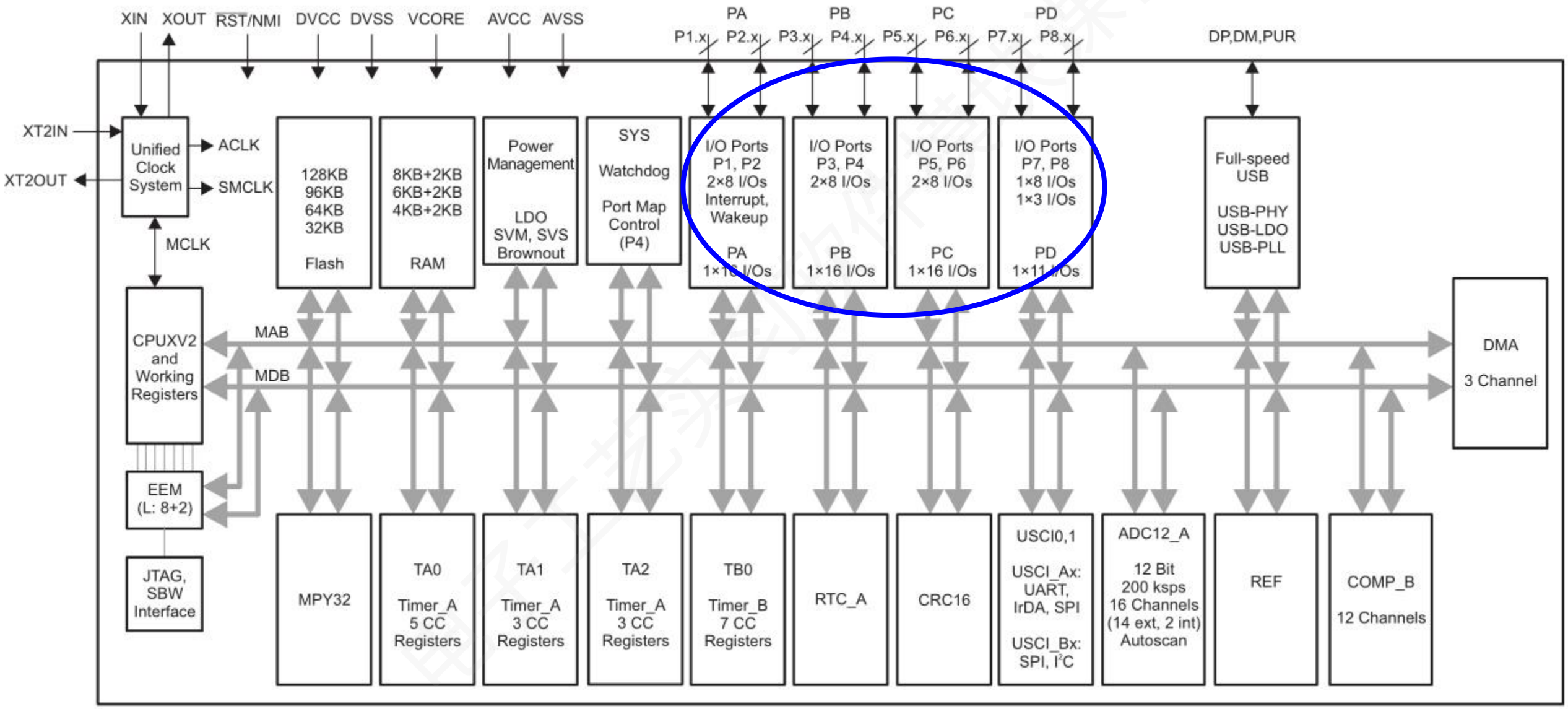


循环结构

MSP430 GPIO应用

实验与创新实践教育中心
哈尔滨工业大学（深圳）

展馆灯光控制系统功能需求分析-总体设计



参考教材和资源

- 《MSP430超低功耗单片机原理与应用（第3版）》

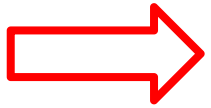
  作者：沈建华、杨艳琴、王慈

  出版社：清华大学出版社

- MOOC，设置习题

MSP430x5xx and MSP430x6xx Family User Guide.pdf

本节内容



- GPIO 概述
- MSP430 GPIO 特点
- MSP430 GPIO 寄存器
- MSP430 GPIO 库函数
- GPIO 编程示例
- GPIO 实验内容

GPIO 概述

GPIO (General Purpose I/O)　通用输入输出端口

应用

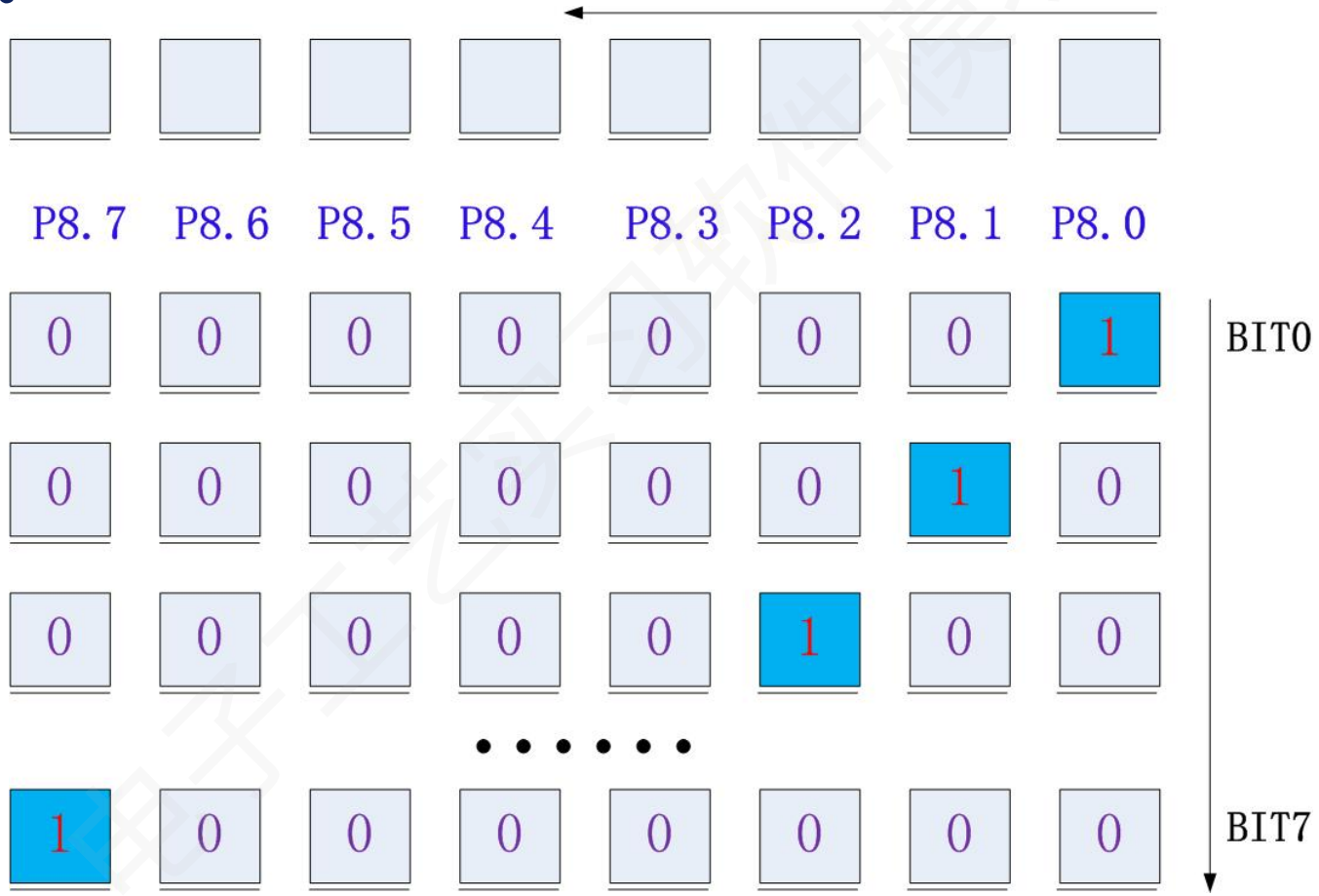
- 用于芯片与片外器件或设备的交互。

特性

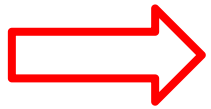
- 可以独立控制每个GPIO口的方向（输入/输出模式）；
- 可以独立设置每个GPIO的输出状态（高/低电平）；
- 所有GPIO口在复位后都有个默认方向（或输入或输出）。

GPIO 概述

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。



本节内容



- GPIO 概述
- MSP430 GPIO 特点
- MSP430 GPIO 寄存器
- MSP430 GPIO 库函数
- GPIO 编程示例
- GPIO 实验内容

MSP430 GPIO 特点

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：

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	√	√	√	√	√	√	√	√	√			√	√	√

MSP430 GPIO 特点

功能丰富

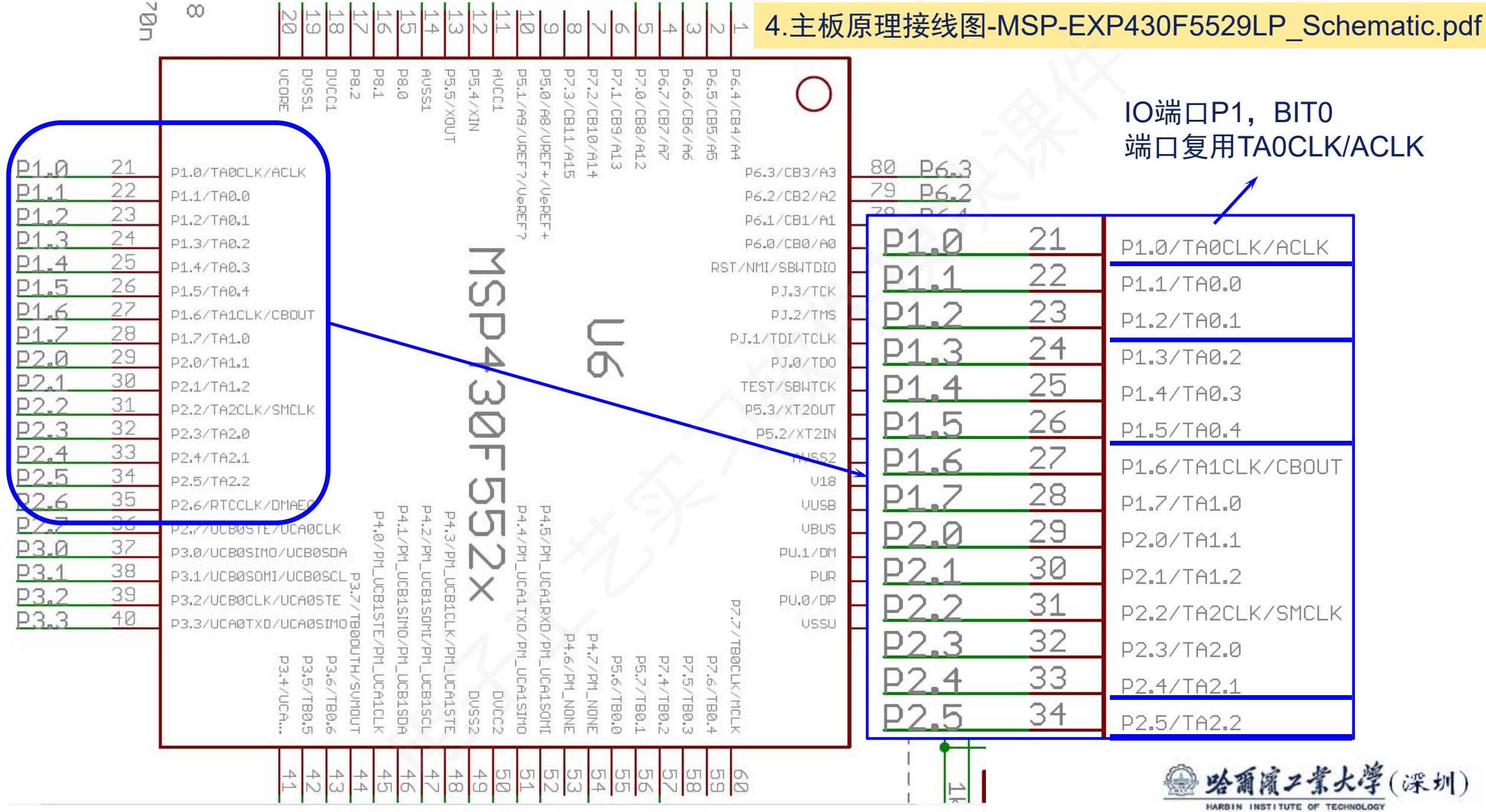
MSP430各端口和功能，如下表所示：

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）。

4.主板原理接线图-MSP-EXP430F5529LP_Schematic.pdf



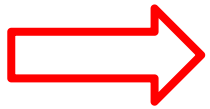
MSP430 GPIO 特点

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）。

本节内容



- GPIO 概述
- MSP430 GPIO 特点
- MSP430 GPIO 寄存器
- MSP430 GPIO 库函数
- GPIO 编程示例
- GPIO 实验内容

GPIO 寄存器

名称	缩写	BIT=1	BIT=0
方向寄存器	PxDIR	输出模式	输入模式
输入寄存器	PxIN	输入高电平	输入低电平
输出寄存器	PxOUT	输出高电平	输出低电平
上下拉电阻使能寄存器	PxREN	使能	禁用
功能选择寄存器	PxSEL	外设功能	IO端口
驱动强度寄存器	PxDS	高强度	低强度
中断使能寄存器	PxIE	允许中断	禁止中断
中断触发沿寄存器	PxIES	下降沿置位	上升沿置位
中断标志寄存器	PxIFG	有中断请求	无中断请求

通用控制寄存器

中断控制寄存器

PxDIR方向寄存器

Port x Direction Register

Figure 12-11. PxDIR Register

7	6	5	4	3	2	1	0
PxDIR							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-13. PxDIR Register Description

Bit	Field	Type	Reset	Description
7-0	PxDIR	RW	0h	Port x direction 0b = Port configured as input 1b = Port configured as output

- 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。
- 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; // 设置P1端口P1.0引脚为输出方向
```

PxIN 输入寄存器

Port x Input Register

Figure 12-9. PxIN Register

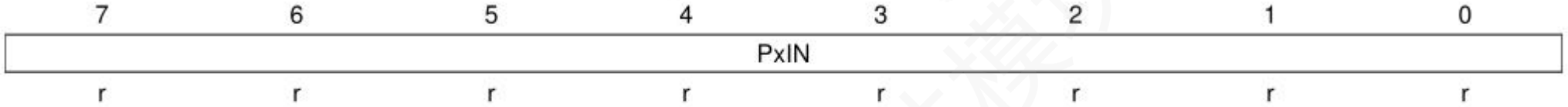


Table 12-11. PxIN Register Description

Bit	Field	Type	Reset	Description
7-0	PxIN	R	undefined	Port x input. Read only.

- 该寄存器是只读寄存器，即用户不能对它写入。
- 这个寄存器是只读的，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)：

  Bit = 1：输入为高电平；

  Bit = 0：输入为低电平；

PxOUT 输出寄存器

Port x Output Register

Figure 12-10. PxOUT Register

7	6	5	4	3	2	1	0
PxOUT							
rw	rw	rw	rw	rw	rw	rw	rw

Table 12-12. PxOUT Register Description

Bit	Field	Type	Reset	Description
7-0	PxOUT	RW	undefined	Port x output When I/O configured to output mode: 0b = Output is low 1b = Output is high When I/O configured to input mode and pullups/pulldowns enabled: 0b = Pulldown selected 1b = Pullup selected

- 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

1. 将需要的值写入该寄存器，控制输出引脚的电平状态
2. 作为输入且上下拉使能时，选择引脚的上拉或下拉

PxREN上拉或下拉电阻使能寄存器

Port x Pullup/Pulldown Resistor Enable Registers

Figure 12-12. PxREN Register

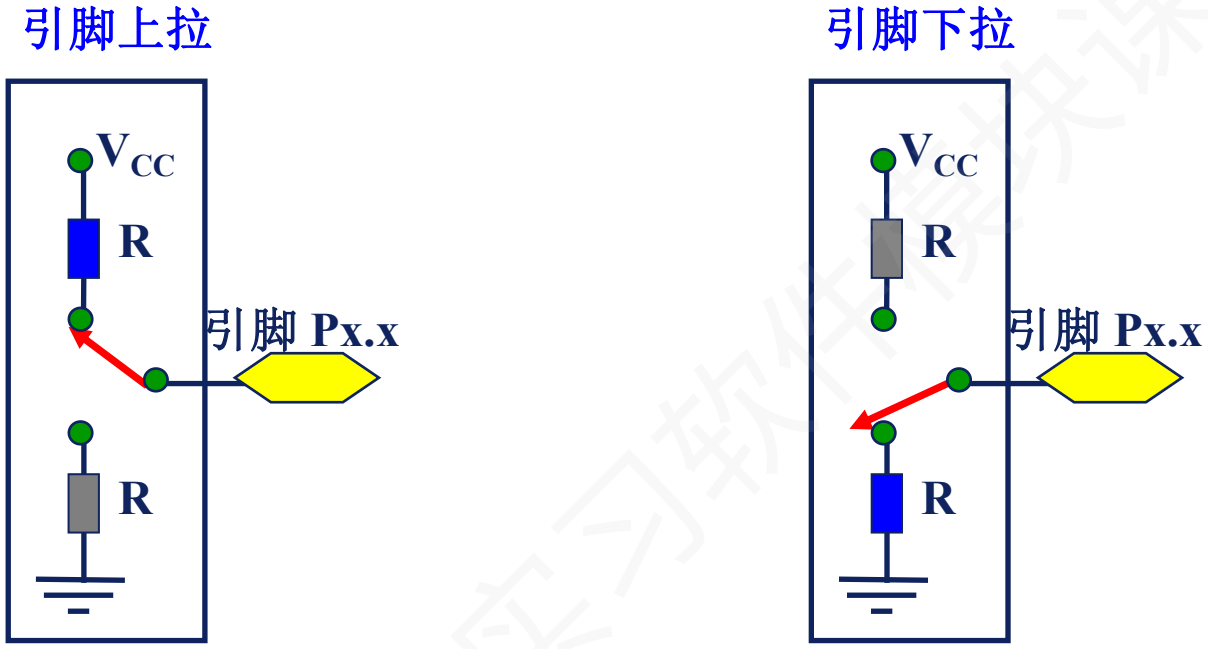
7	6	5	4	3	2	1	0
PxREN							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-14. PxREN Register Description

Bit	Field	Type	Reset	Description
7-0	PxREN	RW	0h	Port x pullup/pulldown resistor enable. When respective port is configured as input, setting this bit will enable the pullup or pulldown. See Table 12-1 0b = Pullup or pulldown disabled. 1b = Pullup or pulldown enabled.

- 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。
- 当使能引脚上拉或下拉功能时，通过设置PxOUT相应位来选择。

上下拉电阻？



PxOUT

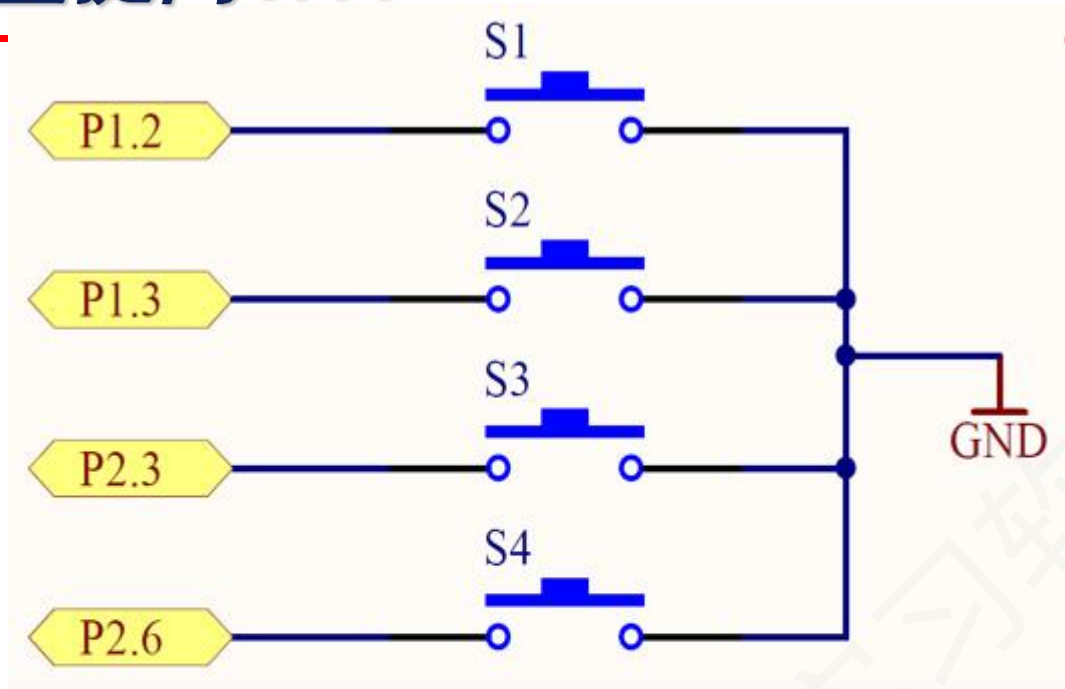
- **Bit = 1: 引脚选择上拉;**

  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。

- **Bit = 0: 引脚选择下拉;**

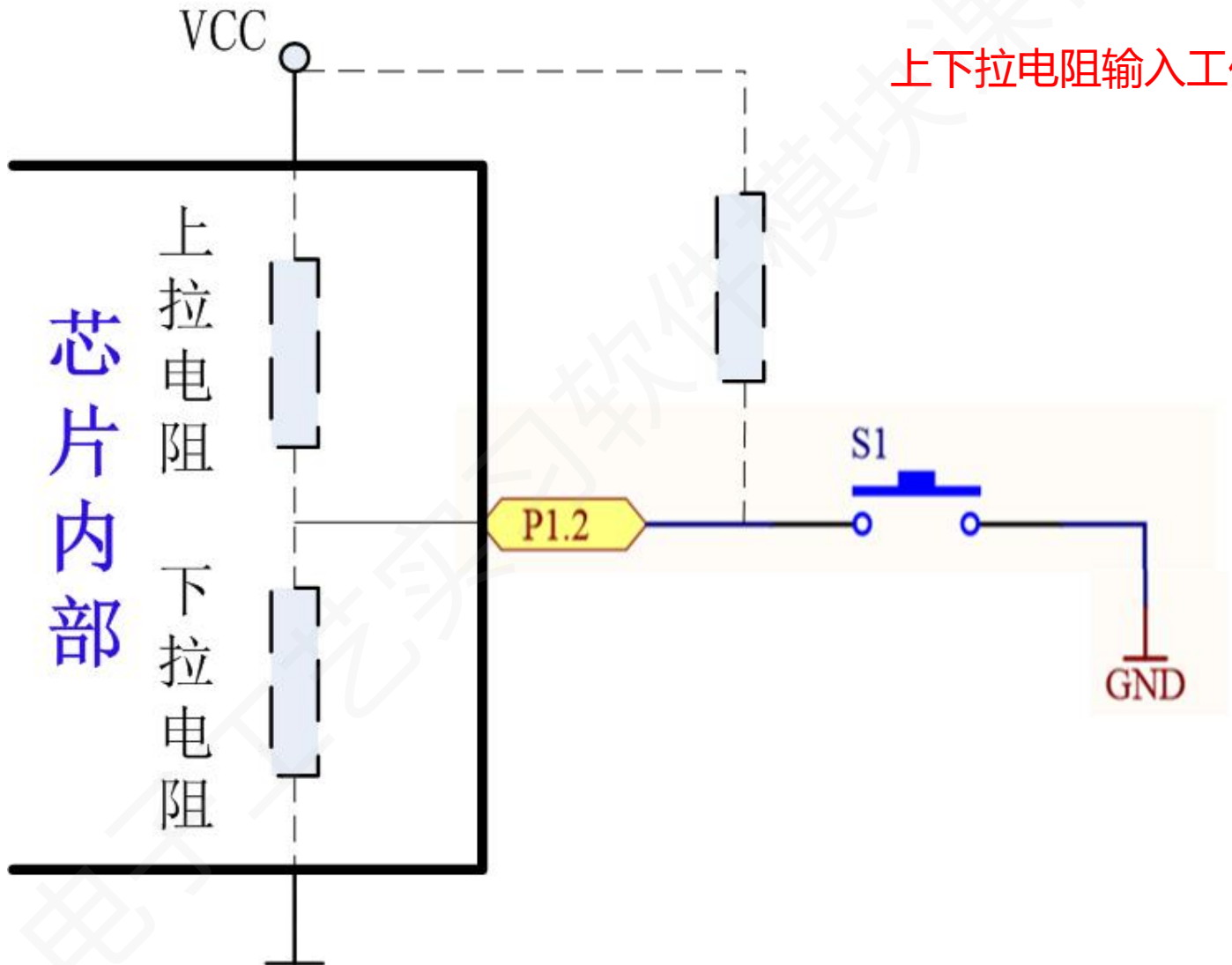
  下拉电阻则是把电平拉低，电阻接到GND地线上。

课堂提问1.1：



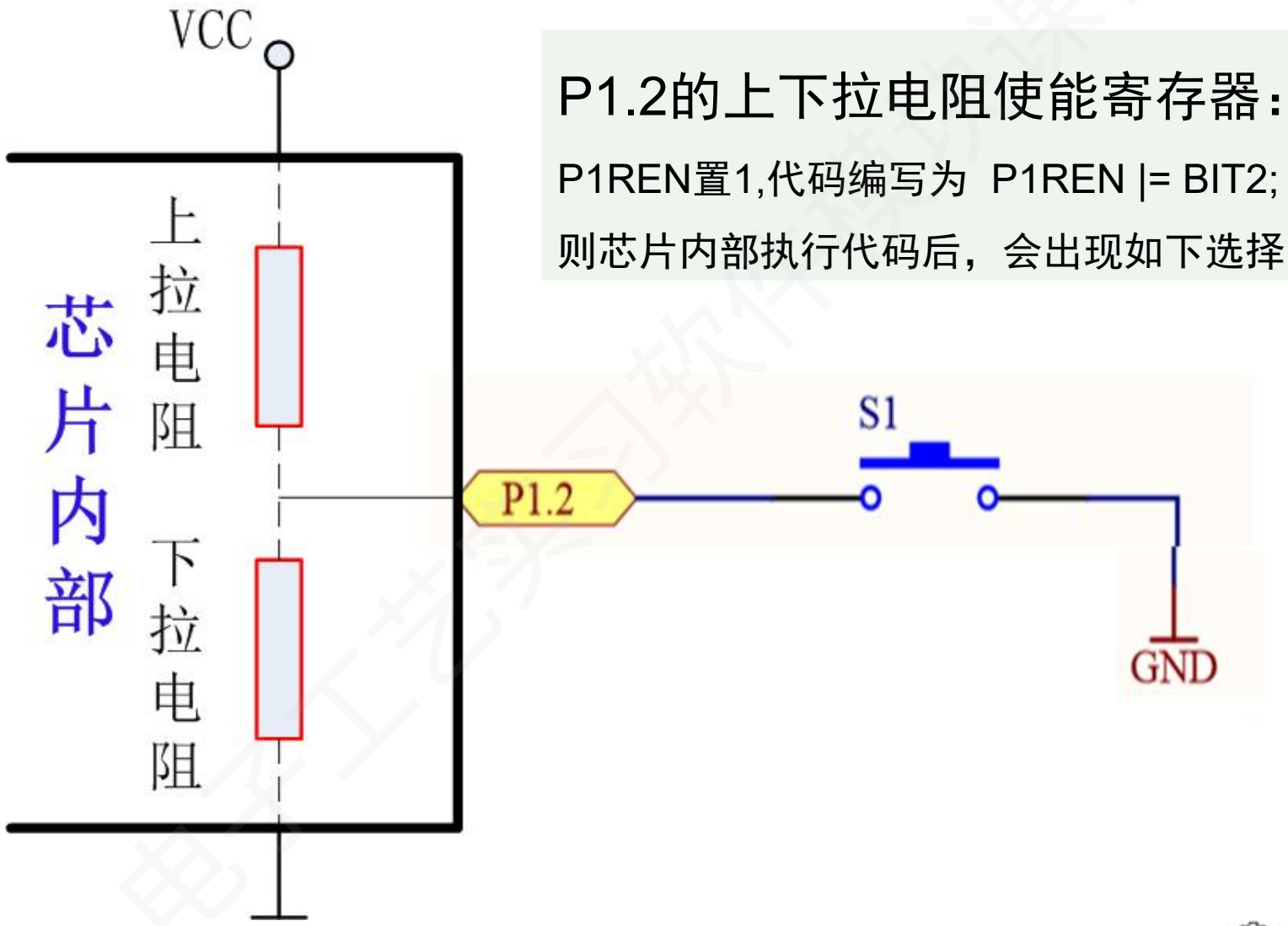
此按键接线下如何设置引脚的工作状态？上拉？下拉？

★GPIO 工作模式



上下拉电阻输入工作模式

★GPIO 工作模式

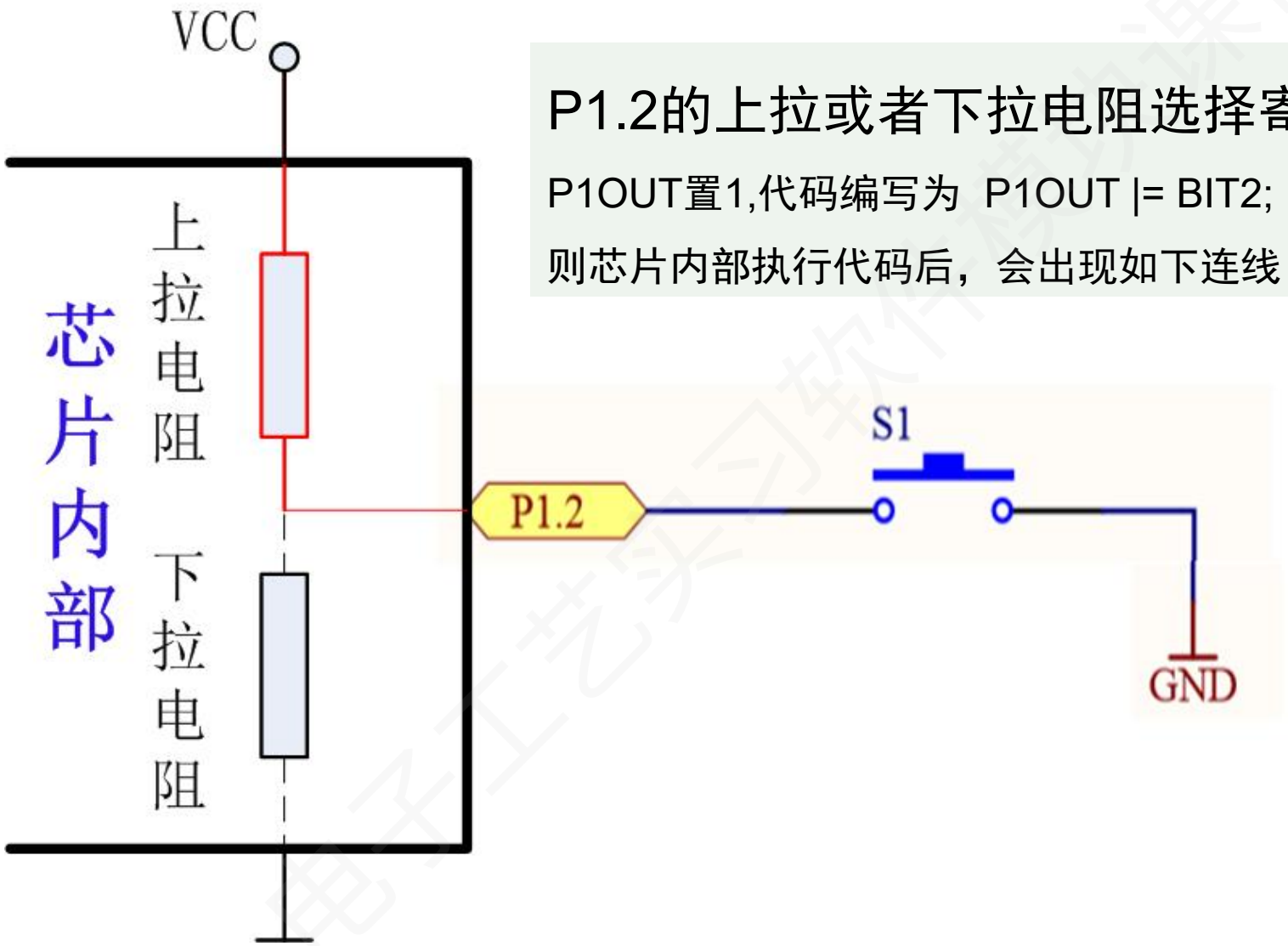


P1.2的上下拉电阻使能寄存器：

P1REN置1,代码编写为 P1REN |= BIT2;

则芯片内部执行代码后，会出现如下选择：

★GPIO 工作模式



P1.2的上拉或者下拉电阻选择寄存器 （输出寄存器）：

P1OUT置1,代码编写为 P1OUT |= BIT2;

则芯片内部执行代码后，会出现如下连线：

PxSEL功能选择寄存器

REGISTERDESCRIPTION

Figure 12-14. PxSEL Register

Table 12-16. PxSEL Register Description

Bit	Field	Type	Reset	Description
7-0	PxSEL	RW	0h	Port x function selection 0b = I/O function is selected 1b = Peripheral module function is selected

- 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端口功能与外设模块功能。
- 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。

PxDS输出驱动强度寄存器

Port x Drive Strength Register

Figure 12-13. PxDS Register

7	6	5	4	3	2	1	0
PxDS							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-15. PxDS Register Description

Bit	Field	Type	Reset	Description
7-0	PxDS	RW	0h	Port x drive strength 0b = Reduced output drive strength 1b = Full output drive strength

- 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。

寄存器位操作

与（&）	0 & 0 = 0	1 & 0 = 0	0 & 1 = 0	1 & 1 = 1
或（|）	0 | 0 = 0	1 | 0 = 1	0 | 1 = 1	1 | 1 = 1
异或（^）	0 ^ 0 = 0	1 ^ 0 = 1	0 ^ 1 = 1	1 ^ 1 = 0

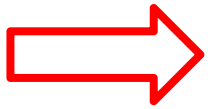
每一位代表一个IO引脚

```
P1OUT |= BIT0;        //P1.0置1
P1OUT &= ~BIT0;       //P1.0清零
P1OUT ^= BIT0;        //P1.0取反
P1IN&BIT0             //读取P1.0的值
```

字节操作
与位操作
的区别？

```
P1OUT = BIT0;         //P1.0置1, P1.1~P1.7清零
P1OUT |= BIT0;        //P1.0置1
```

本节内容



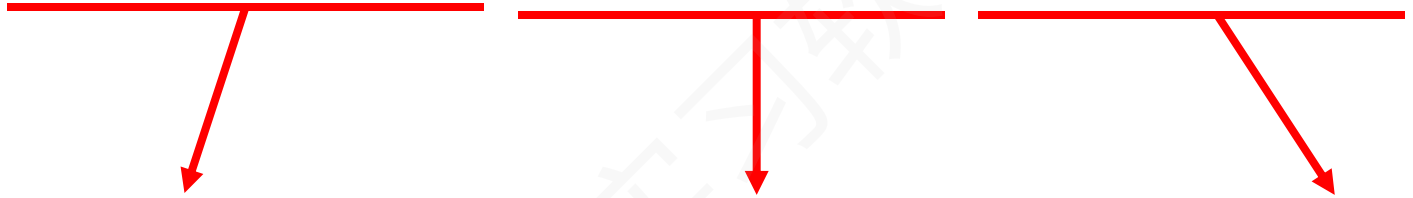
- GPIO 概述
- MSP430 GPIO 特点
- MSP430 GPIO 寄存器
- MSP430 GPIO 库函数
- GPIO 编程示例
- GPIO 实验内容

MSP430 GPIO库函数

GPIO API 库函数：

引脚配置函数、引脚状态函数、中断处理函数

```
void GPIO_setAsOutputPin (uint8_t selectedPort, uint16_t selectedPins)
```



函数名：功能描述　　输入参数1：端口号　　输入参数2：引脚号

函数名、形参定义、使用方法：

3.库函数用户手册-MSP430F5xx_6xx_DriverLib_Users_Guide.pdf

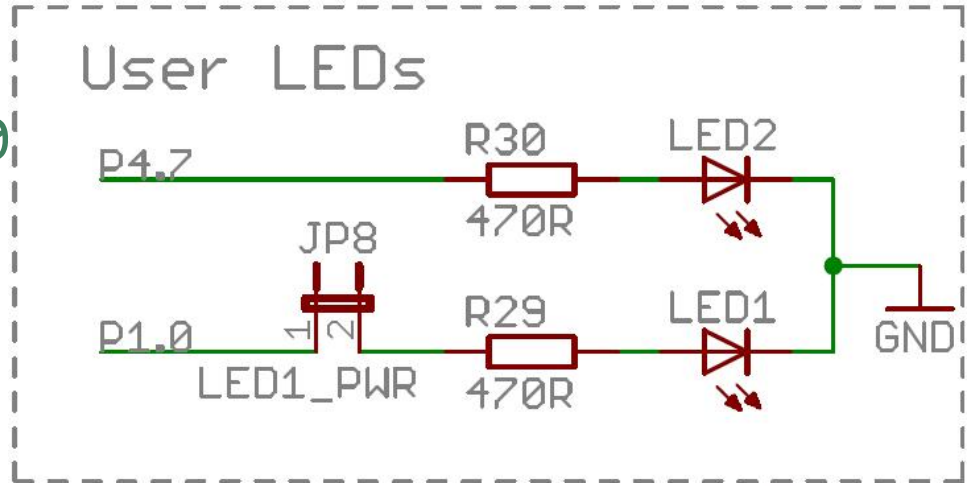
本节内容

- GPIO 概述
- MSP430 GPIO 特点
- MSP430 GPIO 寄存器
- MSP430 GPIO 库函数
- GPIO 编程示例
- GPIO 实验内容

演示实验1.1

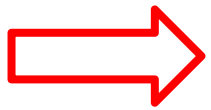
```
void main(void)
{
    WDTCTL = WDTPW | WDTHOLD;// stop watchdog timer
    P1DIR |= 0x01;                    // configure P1.0 as output

    volatile unsigned int i; // volatile to prevent optimization

    while(1)
    {
      P1OUT ^= BIT0;                  //toggle P1.0
      for(i=10000; i>0; i--); // delay
    }
}
```

还记得这个工程吗



本节内容

- GPIO 概述
- MSP430 GPIO 特点
- MSP430 GPIO 寄存器
- MSP430 GPIO 库函数
- GPIO 编程示例
- GPIO 实验内容



演示实验1.2 按键对 LED 灯的控制实验（查询方式）

- 实验目的

（1）掌握对IO口的查询操作和IO基本操作的流程；

（2）对部分引脚功能有个初步了解；

- 实验要求 （指导书1.4.2.1）

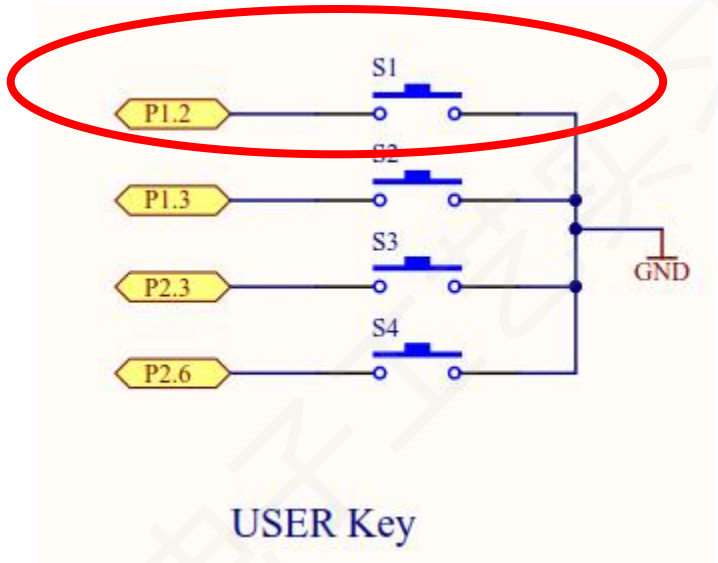
通过按键S1对IO口P1.2的操作，以查询方式实现高低电平检测，从而对与L1相连的P8.1的电平控制，实现L1灯的亮灭。

实验现象

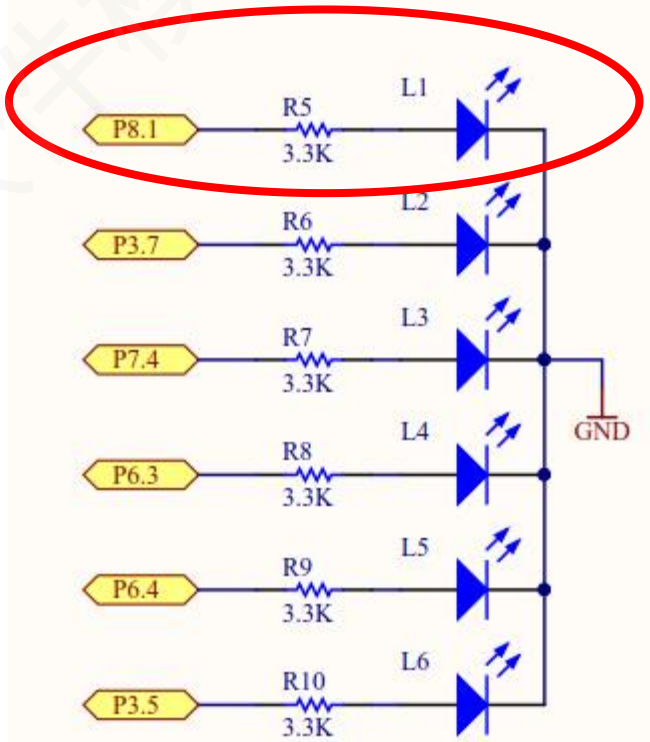
演示实验1.2 按键对 LED 灯的控制实验（查询方式）

一、硬件接线图

5.口袋板接线原理图-MSP430F5529_Pocket_Lab_Schematic.pdf



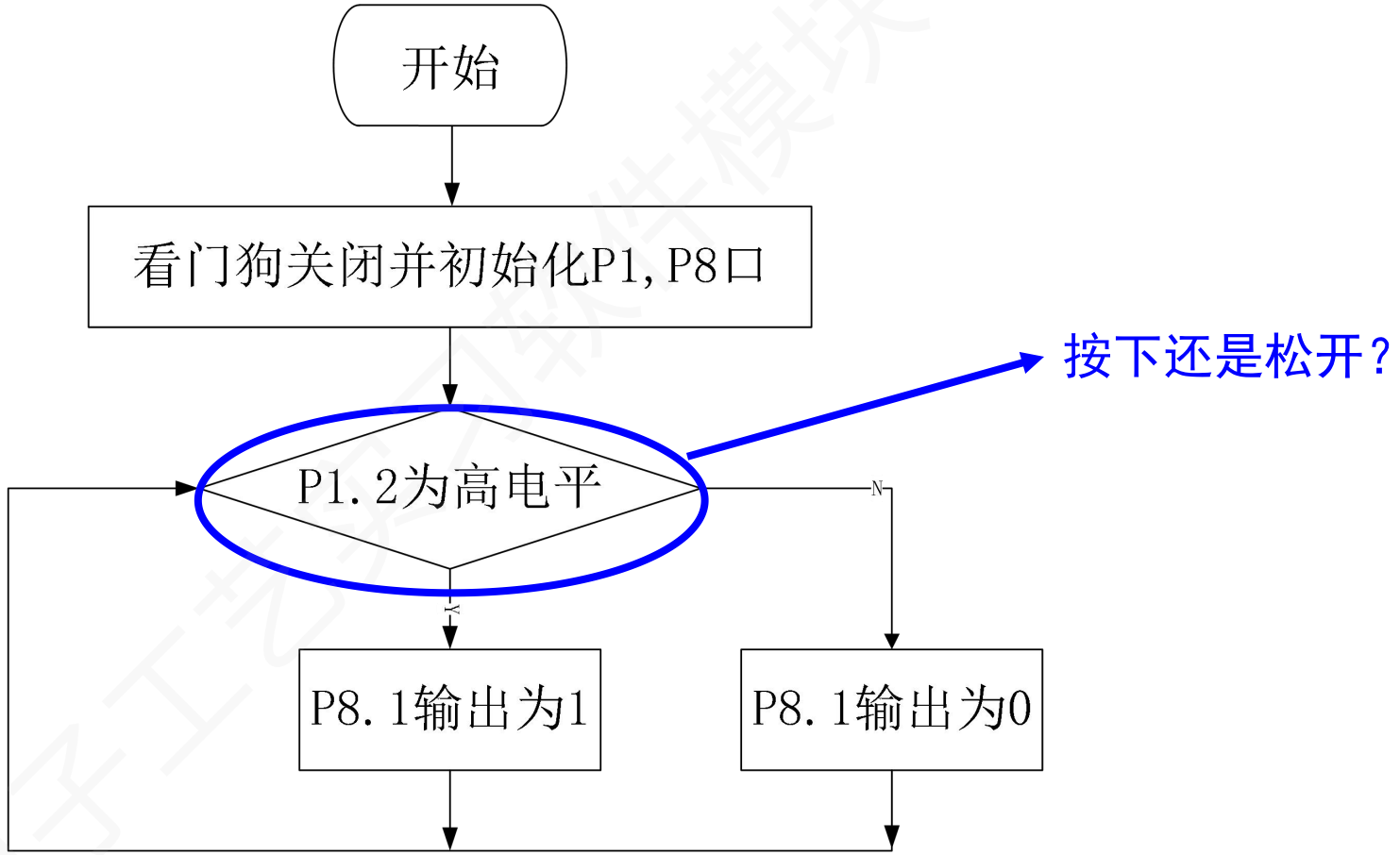
USER Key



LED

演示实验1.2 按键对 LED 灯的控制实验（查询方式）

二、程序流程图



演示实验1.2 按键对LED灯的控制实验（查询方式）

三、编写代码

```
int main(void)
{
    WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer
    P8DIR|=BIT1;              // 设置P8.1口为输出模式
    P8OUT&=~BIT1;             // 设置P8.1口初始状态
    P1REN|=BIT2;              // 使能P1.2上下拉电阻功能
    P1OUT|=BIT2;              // 置P1.2为上拉电阻方式
    while (1)
    {         //判断按键状态，如果S2按键按下则P1.2=0，抬起则P1.2=1
        if (!(P1IN&BIT2))
            P8OUT|=BIT1; // P8.1输出高（LED1点亮）
        else
            P8OUT&=~BIT1;// P8.1输出低（LED1熄灭）
    }
    return 0;
}
```

初始化LED灯

初始化按键

查询按键状态

点灯

灭灯

演示实验1.2 按键对 LED 灯的控制实验（查询方式）

- 四、CCS环境下编码、编译、下载、调试

（1）打开CCSv10并确定工作区间，然后选择File-->New-->CCS Project，新建一个CCS工程。

（2）在 Project name 中输入新建工程的名称，在此输入 lab1_1。

（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），单击Finish。

（5）在新窗口中输入编写的代码，编译并调试。

课堂实验1.1

请动手练习，实现现象：

通过按键S2（S3/S4任选1个）的操作，以查询方式进行高低电平检测，从而实现L2灯(L3~L6任选1个)的亮灭控制(S2按下L2灭，S2松开L2亮)。

MSP430 中断

实验与创新实践教育中心
哈尔滨工业大学（深圳）

本节内容

- 中断系统概述
- MSP430 GPIO 中断
- 中断嵌套
- MSP430 GPIO 中断实验内容

课堂提问1.2

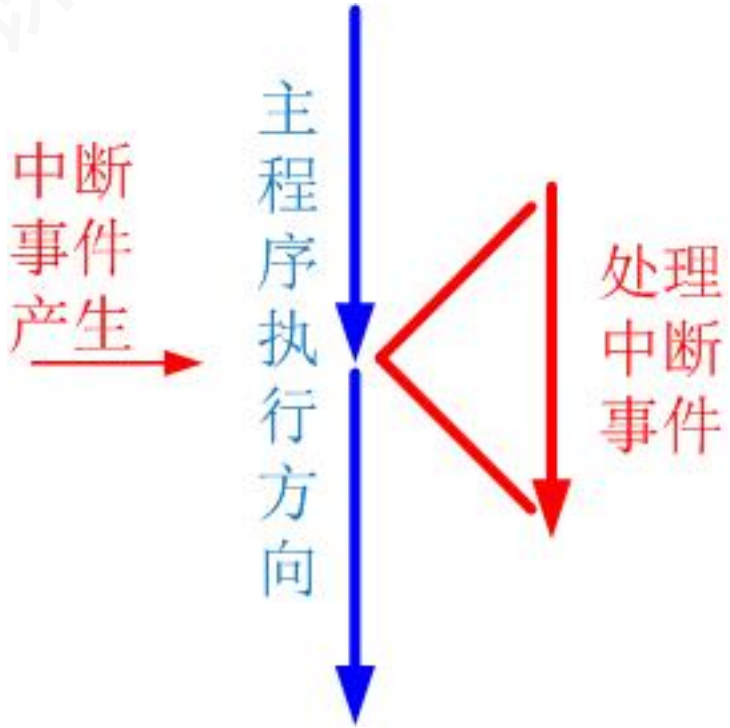
你对中断概念是如何理解的?

中断系统简介

中断是暂停CPU正在运行的程序，转去执行相应的中断服务程序（ISR）；当中断服务程序执行结束后，返回到被中断的程序继续运行的过程。

中断系统是实现中断的软硬件系统。

有效利用中断可以简化程序、提高执行效率、满足实时需求。



中断响应的过程

中断源发出中断请求，征得系统允许

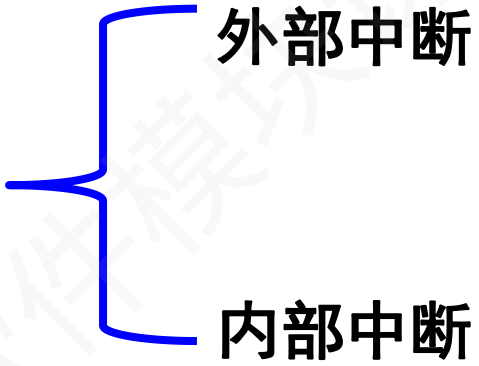
保护中断现场，执行中断服务程序

结束中断服务程序，恢复中断现场

- 硬件自动中断服务
  - PC入栈
  - SR入栈
  - 中断向量赋给PC
  - GIT、CPUOFF、OSCOFF和SCG1清除
  - IFG标志位清除（单源中断标志）
- 执行中断子程序
- 执行RETI指令（中断返回）
  - SR出栈（恢复原来的标志）
  - PC出栈

中断分类

根据中断触发信号的来源分类

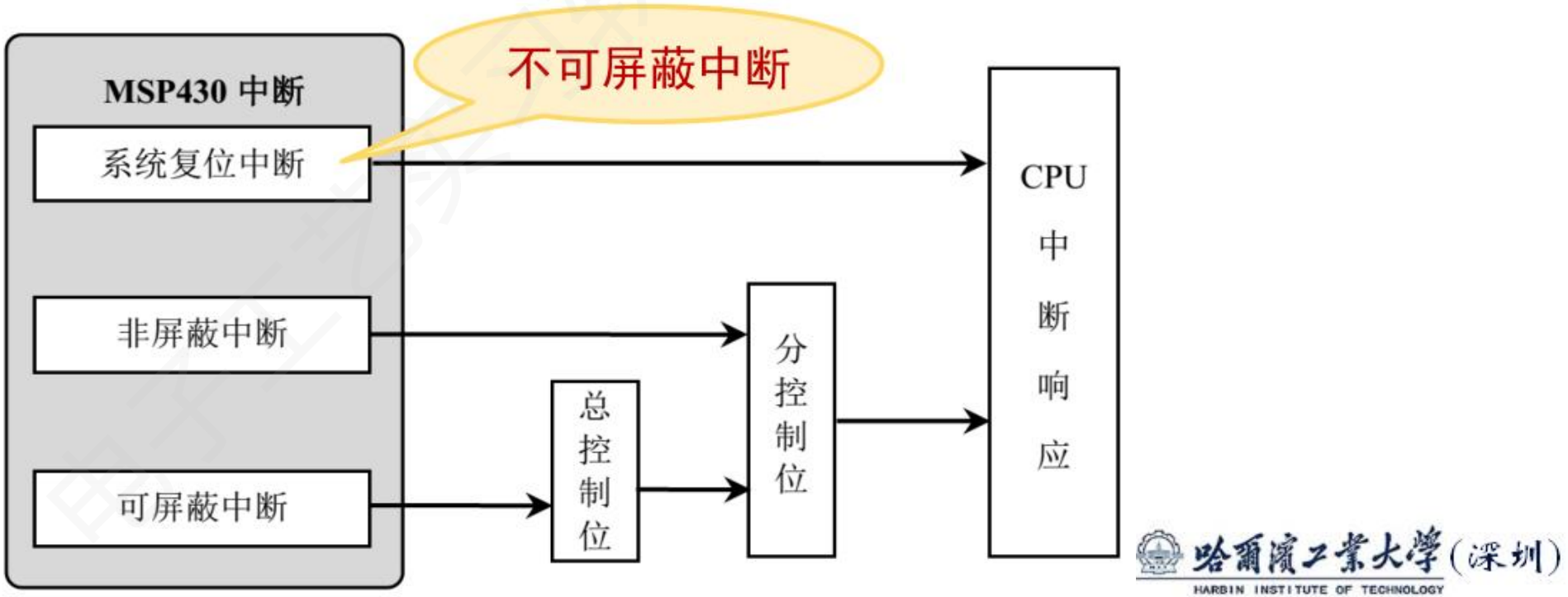


- 外部中断：外部引脚触发的中断。
- 内部中断：单片机内部模块触发的中断，例如：定时器中断、串行通信接收与发送中断。

中断分类

根据中断源的可控程度可分为：

- 系统复位中断(System Reset Interrupt)
- 非屏蔽中断(Non-maskable Interrupt)
- 可屏蔽中断(Maskable Interrupt)



中断向量表

中断向量即为中断函数入口地址。MSP430的中断向量位于0xFFFF~0xFF80地址范围，不同系列设备中断向量表可能不同。

中断源	中断标志	系统中断	字地址	优先级
复位上电 外部复位看门狗 FLASH密码	…… WDTIFG KEYV	…… 复位	…… 0xFFFE	…… 最高
系统NMI:PMM	--	不可屏蔽	0xFFFC	……
用户NMI 晶振错误 内存访问错误	NMIIFG OFIFG ACCVIFG	不可屏蔽 不可屏蔽 不可屏蔽	0xFFFA	
设备特定	--	--	0xFFF8	……
……	--	--		
看门狗定时器	WDTIFG	可屏蔽	……	……
……	--	--		
设备特定	--	--	……	……
保留	--	可屏蔽	……	最低

中断函数（中断服务程序）

中断向量，名称参考示例程序或头文件

```
#pragma vector=WDT_VECTOR
__interrupt void WDT_ISR(void)
{
  IE1 &= ~WDTIE;                    // disable interrupt
  IFG1 &= ~WDTIFG;                  // clear interrupt flag
  WDTCTL = WDTPW + WDTHOLD;         // put WDT back in hold state
  BUTTON_IE |= BUTTON;              // Debouncing complete
}
```

用户自定义的函数名

中断服务子程序

本节内容

- 中断系统概述
- MSP430 GPIO 中断
- 中断嵌套
- MSP430 GPIO 中断实验内容

GPIO中断

具有中断能力的端口：P1和P2

GPIO中断有关寄存器：

PxIE	端口x中断使能寄存器
PxIFG	端口x中断标志寄存器
PxIES	端口x中断边沿选择寄存器

PxIE中断使能寄存器 （仅P1和P2）

Port 1 Interrupt Enable Register

Figure 12-4. P1IE Register

7	6	5	4	3	2	1	0
P1IE							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-6. P1IE Register Description

Bit	Field	Type	Reset	Description
7-0	P1IE	RW	0h	Port 1 interrupt enable 0b = Corresponding port interrupt disabled 1b = Corresponding port interrupt enabled

- 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。

PxIES中断触发沿选择寄存器（仅P1和P2）

Port 1 Interrupt Edge Select Register

Figure 12-3. P1IES Register

7	6	5	4	3	2	1	0
P1IES							
rw	rw	rw	rw	rw	rw	rw	rw

Table 12-5. P1IES Register Description

Bit	Field	Type	Reset	Description
7-0	P1IES	RW	undefined	Port 1 interrupt edge select 0b = P1IFG flag is set with a low-to-high transition. 1b = P1IFG flag is set with a high-to-low transition.

- 中断使能后，还需定义该引脚的**中断触发方式**。
- 该寄存器可读可写。
- PxIES的配置：Bit = 0：上升沿使相应中断标志置位；
  Bit = 1：下降沿使相应中断标志置位。

PxIFG中断标志寄存器（仅P1和P2）

Port 1 Interrupt Flag Register

Figure 12-5. P1IFG Register

Table 12-7. P1IFG Register Description

Bit	Field	Type	Reset	Description
7-0	P1IFG	RW	0h	Port 1 interrupt flag 0b = No interrupt is pending 1b = Interrupt is pending

- 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。
- 中断使能后，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。

使用端口中断的具体步骤

配置IO端口：端口方向、上/下拉电阻

↓

设置中断触发方式：上升沿/下降沿

↓

开启中断使能：IO中断使能、全局中断使能

↓

编写中断子函数：if语句查询产生中断的具体IO口

↓

编写事件处理函数：IO口中断发生后要进行的操作（例如：点亮LED、发送通信数据）

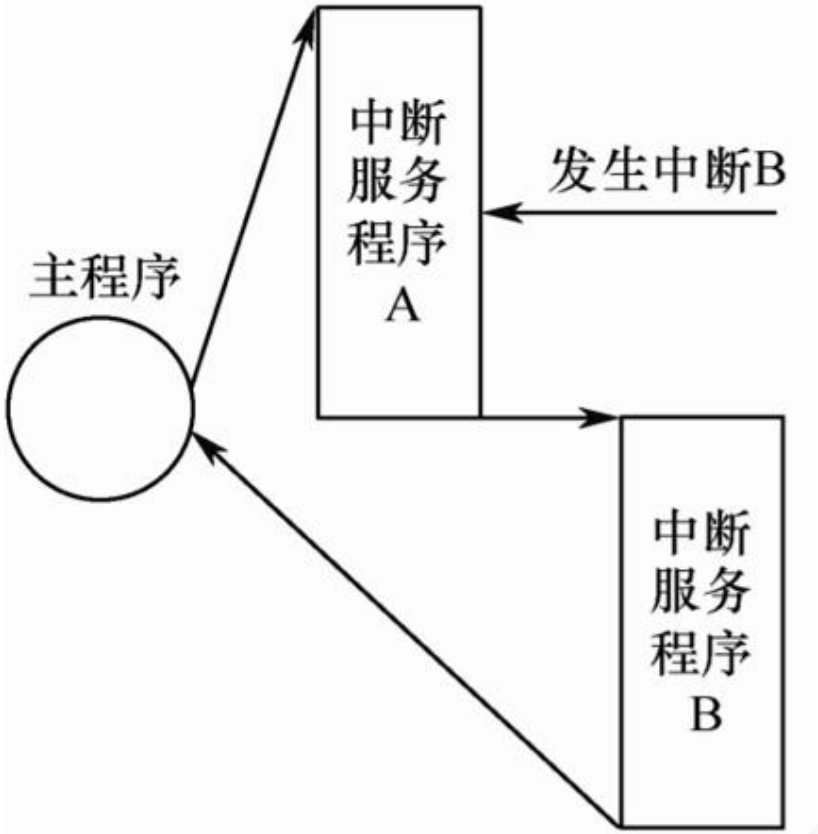
↓

清除中断标志位：PxIFG=0

本节内容

- 中断系统概述
- MSP430 GPIO 中断
- 中断嵌套
- MSP430 GPIO 中断实验内容

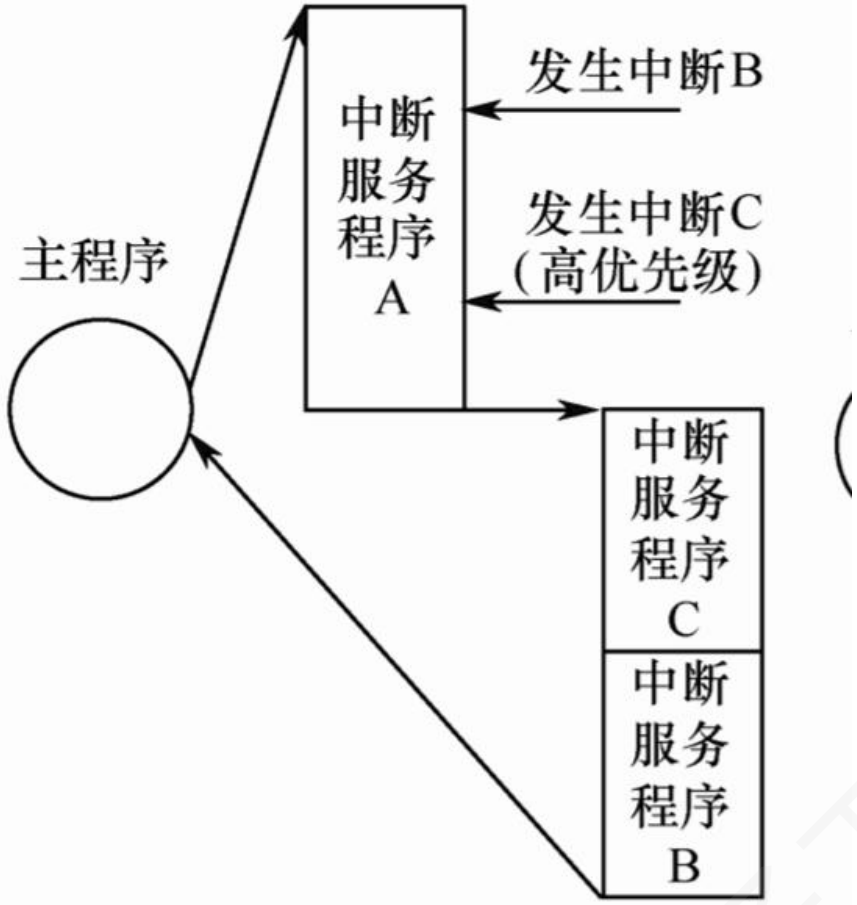
多种中断发生情况示意图



（a）A中断服务程序未开中断，发生中断B

- 当进入中断入口处，MSP430单片机会自动清除中断允许标志位GIE，即默认不能发生中断嵌套，即使高优先级中断也不能打断低优先级中断的执行。
- 如图（a）所示，如果在执行中断服务程序A时，发生了中断请求B，B的中断标志位置1，但不会立即响应B的中断，需等待A执行完成返回后（GIE自动恢复），才进入B的中断服务程序。

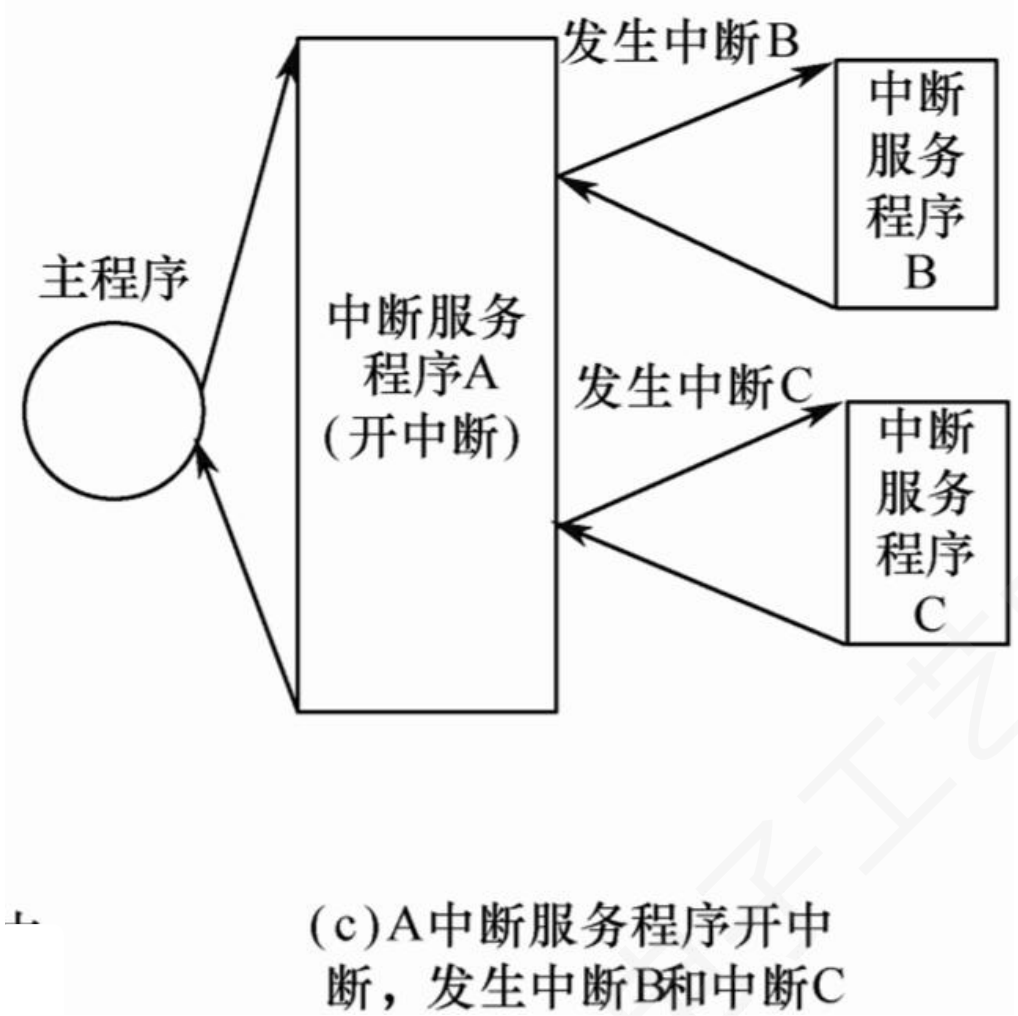
多种中断发生情况示意图



(b)A中断服务程序未开中断，发生中断B和中断C

- 如图（b）所示，如果在执行中断服务程序A时，有多个中断发生，会在A中断执行完毕后，依照中断优先级由高至低的顺序依次执行各个待执行的中断服务程序。
- MSP430x5xxx/6xxx系列单片机的中断优先级是固定不可设的。

中断嵌套



(c)A中断服务程序开中断，发生中断B和中断C

（a）（b）两种情况，先发生的中断将会导致后发生的中断处理延迟。为所有中断都尽快执行，则需允许中断嵌套，如图（c）所示。需要在所有中断的入口处都加一句开中断的语句：_EINT（），恢复总的中断允许（GIE）。中断嵌套被允许后，所有中断能够立即被执行，可以保证事件的实时性要求。

课堂提问1.3

如何在MSP430系列处理器中允许中断嵌套?

中断嵌套

```
#pragma vector = TIMER0_A0_VECTOR
__interrupt void Timer_A (void)

{
    __enable_interrupt(); //开全局中断GIE//_EINT();
    ......
}
```

本节内容

- 中断系统概述
- MSP430 GPIO 中断
- 中断嵌套
- MSP430 GPIO 中断实验内容

演示实验1.3：中断方式实现按键对 LED 灯的控制

- 实验目的

（1）了解 MSP430F5529 中断系统；

（2）掌握 IO 口中断的使用和编程；

实验现象

- 实验要求 （指导书1.4.2.3）

通过中断方式，实现按键S1控制L1灯的亮灭（按一次，L1状态改变一次）

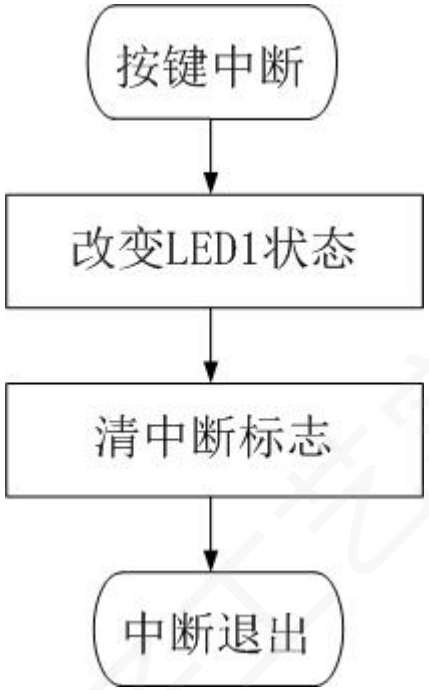
- 解析

口袋板 S1 按键连接到 P1.2 口，当 S1 按键被按下再松开时，P1.2 口电平由高变低再变高，下降沿或者上升沿触发一个中断事件，然后在 P1 口的中断函数中填写代码改变 L1 灯的状态。

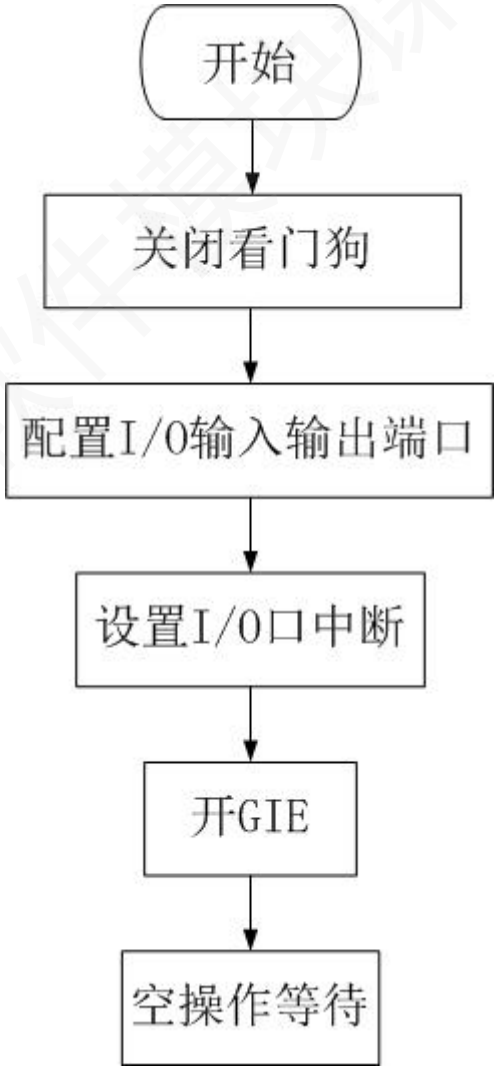
演示实验1.3：中断方式实现按键对 LED 灯的控制

- 程序流程图

中断处理函数



主程序



演示实验1.3：中断方式实现按键对 LED 灯的控制

◆ 编程思路

- 设置 P8.1 口为输出状态以控制 LED1 灯状态
- 使能 P1.2 口的上拉电阻，设置为上拉电阻输入模式
- 选择 P1.2 口中断沿，使能中断，清中断标志位，等待
- 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。
- 设置寄存器 P8DIR、 P8OUT、 P1REN、P1OUT、P1IES、P1FG、P1IE。
- 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”来表明。

```
#include <msp430.h>
int main(void)
{
        WDTCTL = WDTPW | WDTHOLD;  // stop watchdog timer
        P8DIR|=BIT1;  //p8.1 output
        P8OUT&=~BIT1;

        P1DIR&=~BIT2;  //p1.2 input,pull up
        P1REN|=BIT2;
        P1OUT|=BIT2;

        P1IE|=BIT2; //enable P1.2 interrupt
        P1IES|=BIT2; //high-low transition
        P1IFG&=~BIT2;

        _enable_interrupt();

        return 0;
}
```

```
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{
    if(P1IFG&BIT2){
        P8OUT^=BIT1;
        P1IFG&=~BIT2;
    }

    if(P1IFG&BIT3){
        ; //?
        ; //?
    }
}
```

课堂实验1.2

请动手练习，完成实验要求：

通过中断方式，实现按键S1控制L1灯的亮灭、按键S2控制L2灯的亮灭、按键S3控制L3灯的亮灭，以及按键S4控制L4灯的亮灭（按一次，灯状态改变一次）

注意：按键和灯的硬件引脚查询，P1和P2端口分别有一个中断向量，如何书写中断函数？

Q&A