采用 MSP430 LaunchPad 启动开发工作

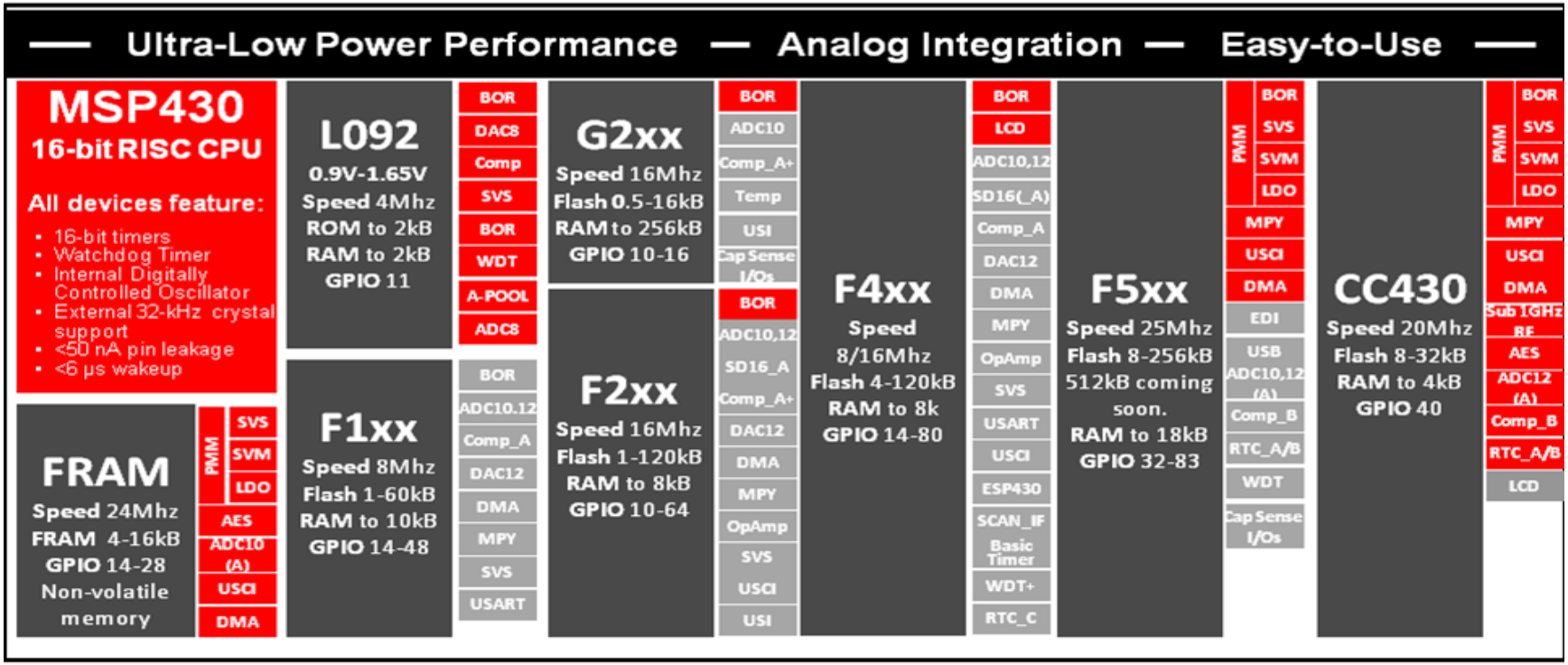
议程

- 介绍 Value Line 系列
- Code Composer Studio
- CPU 与基本时钟模块
- 中断与 GPIO
- Timer_A 与 WDT+
- MSP430低功耗设计
- ADC10 和 Comparator_A+
- 串行通信模块
- Grace
- 电容式触摸按键解决方案

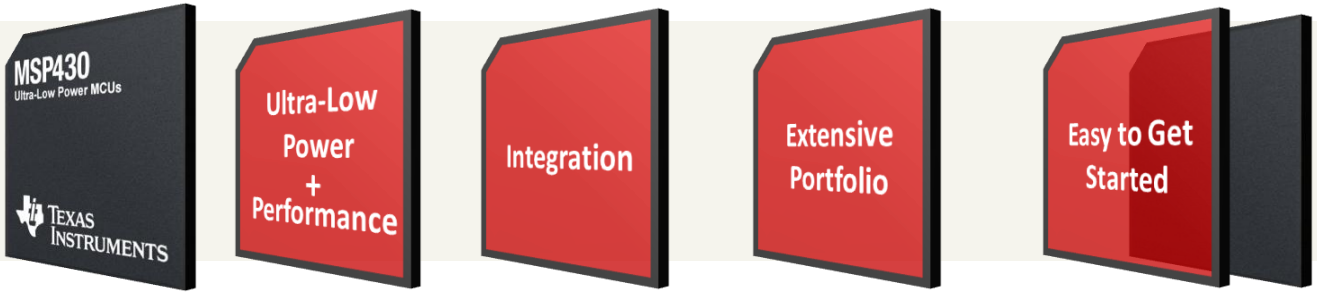
MSP430 系列MCU产品

MSP430 Portfolio at a glance

300+ Ultra-Low Power Devices Starting @ $0.25USD
Featuring: Up to 256kB Flash, 18kB RAM, 25+ Package Options, Up to 113 pins, High integration



MSP430 MCU 介绍



超低功耗

业内功耗最低的 MCU

- 超低功耗运行模式
- 7 种低功耗模式
- 即时唤醒
- 所有的 MSP430 器件均具有超低功耗特性

集成

智能型模拟与数字外设

- 外设工作于低功耗模式
- 减少外部分立器件与物料成本
- 具有 FRAM、USB、RF、电容式触摸 I/O、计量模块、LCD、ADC、DAC 等等

丰富齐全的产品线，低成本可供选择

找到适合您需要的理想 MCU

- 400 多款器件
- 容量高达 256kB 闪存，18kB RAM，超过 25 种封装可供选择
- Value Line 系列器件起售价仅 0.25 美元
- 不同性能与片上集成度的器件可供选择

易于启动开发工作

低成本与简单入手

- 完整套件起售价仅 4.30 美元
- 可提供基于 GUI 的编码及调试工具
- MSP430Ware 软件与资源库

  -包括代码范例、数据手册、用户指南等等！

MSP430 支持的应用

公用事业计量

电能表
燃气表
流量表
智能计量仪表

无线应用

远程传感器
通讯控制器
RFID

消费类电子

便携式电子产品
遥控器
个人保健
PC 外设

MSP430 MCU 可支持数以千计的应用

凭借 MSP430 MCU 的超低功耗性能、高集成度模拟与数字外设、以及易用的工具，客户可方便地实现其产品的差异化

能量收集

可再生能源
无电池设备
太阳能、热能、振动能，等等

便携式医疗

血糖计
温度计
心率监测计
可植入装置

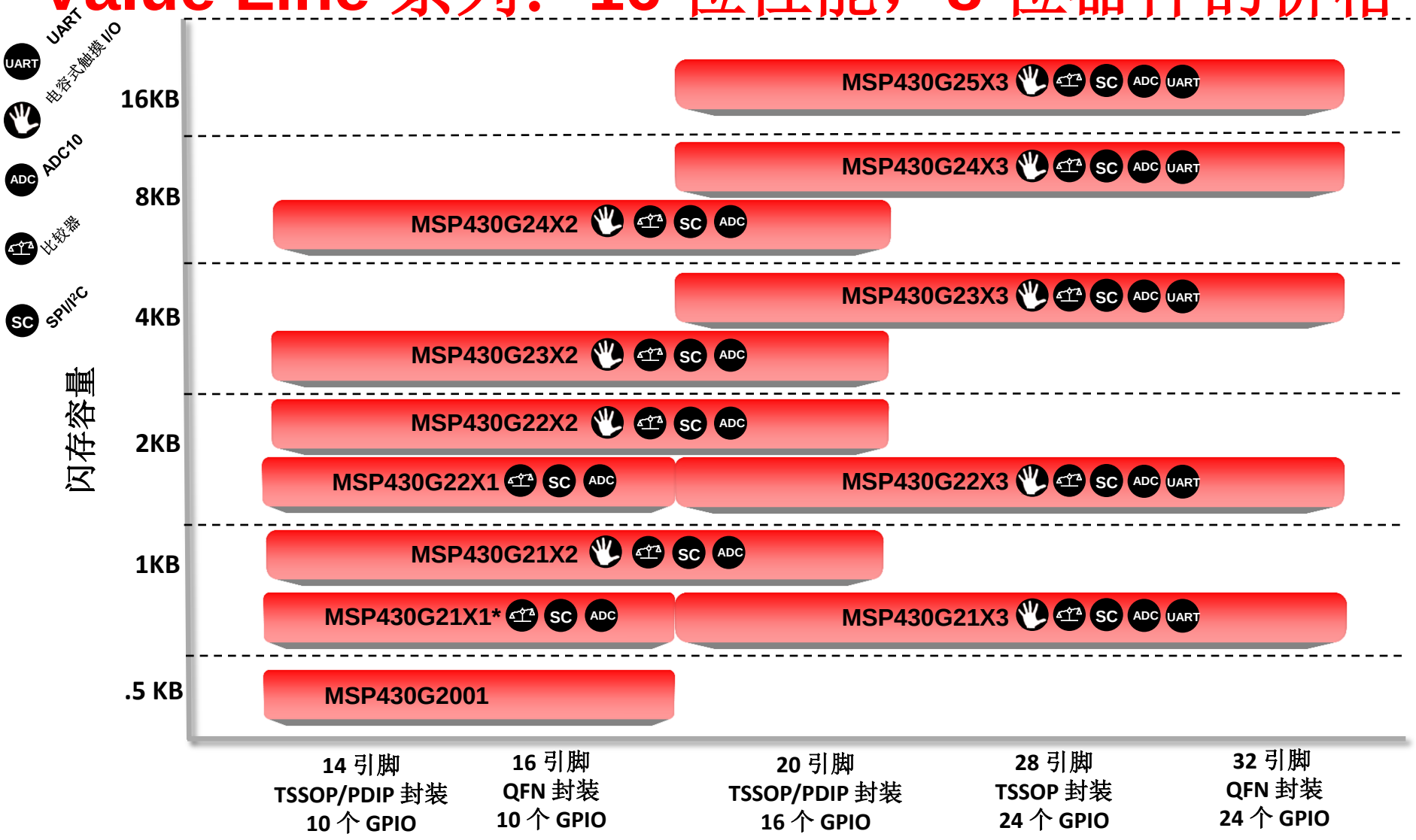
传感器与安全

烟雾探测器
运动探测器
振动检测器
智能传感器

个人健康与健身

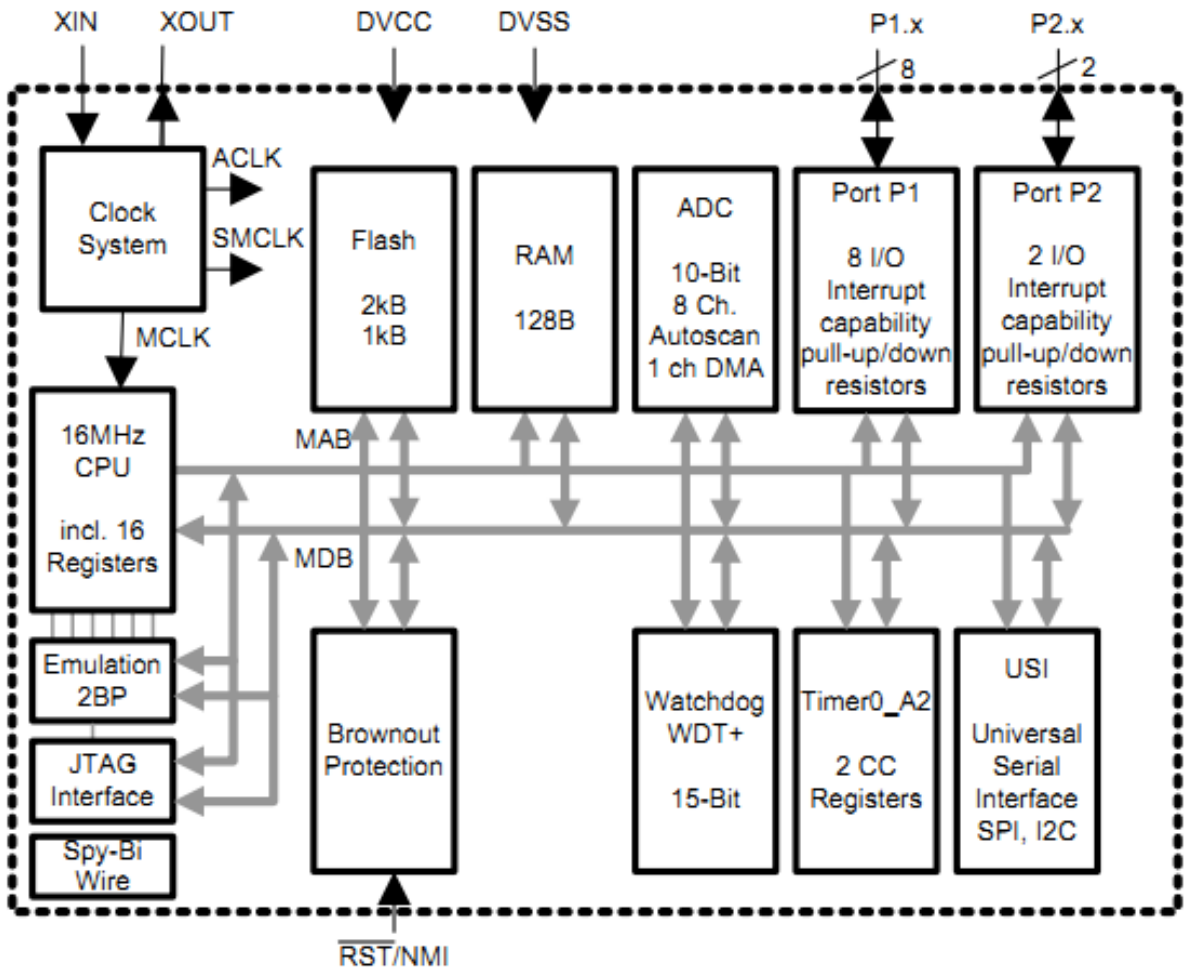
运动手表
计步器
热量计
潜水手表

Value Line 系列：16 位性能，8 位器件的价格



Value Line 功能框图

functional block diagram, MSP430G2x31



Value Line 外设

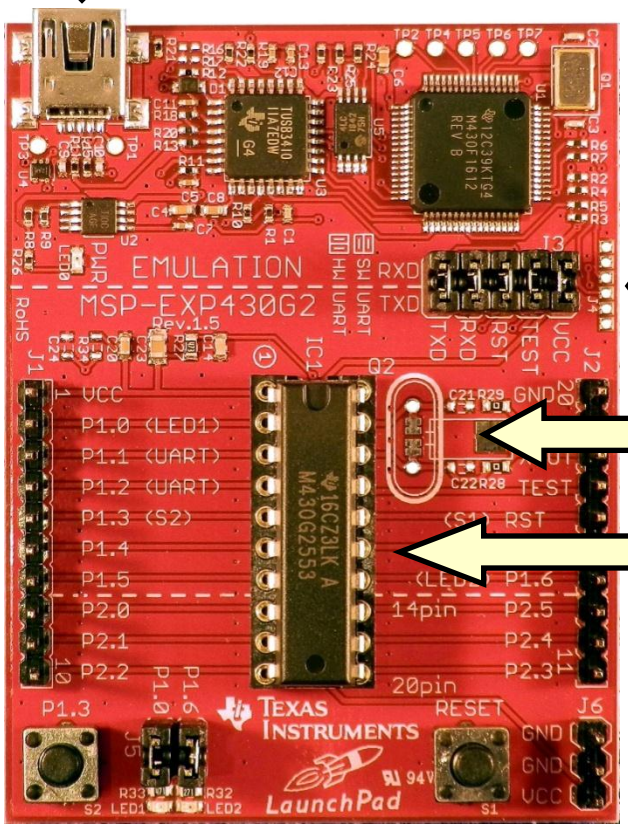
- **通用 I/O**
  - 可独立编程
  - 可提供输入、输出与中断（边沿可选）的任意组合
  - 所有寻址指令可对端口控制寄存器进行读/写访问
  - 每个 I/O 具有一个可独立编程的上拉/下拉电阻
  - 某些器件/引脚具有触摸按键模块 (PinOsc)
- **16 位 Timer_A2**
  - 2 个捕获/比较寄存器
  - 丰富的中断功能
- **WDT+ 看门狗定时器**
  - 也可用作一个普通定时器
- **欠压复位**
  - 可在上电和断电期间提供正确的复位信号
  - 功耗包含于MCU最低功耗时（LPM4）所消耗电流之中

Value Line 外设

- **串行通信**
  - 支持 I2C 和 SPI 的 USI
  - 支持 I2C、SPI 以及 UART 的 USCI
- **Comparator_A+**
  - 可设定反相和同相输入
  - 可选的 RC 输出滤波器
  - 可直接输出至 Timer_A2 捕获输入
  - 具有中断能力
- **8 通道/10 位 200 ksps SAR ADC**
  - 8 个外部通道（取决于器件）
  - 内置电压和温度传感器
  - 可编程的参考电压
  - DTC可在无需 CPU 干预的情况下将结果发送至存储器
  - 具有中断能力

LaunchPad 开发板

USB 仿真器接口



片上仿真器模块

6 针 eZ430 连接器

外置晶体接口

芯片引出脚

MSP430器件和插座

P1.3 按钮

电源连接器

LED 和跳线

P1.0 和 P1.6

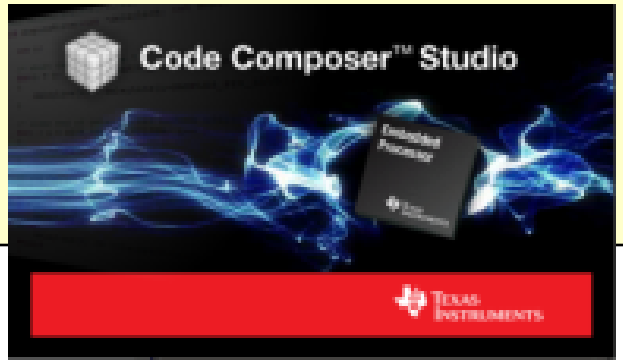
复位按钮

议程

- 介绍 Value Line 系列
- Code Composer Studio
- CPU 与基本时钟模块
- 中断与 GPIO
- Timer_A 与 WDT+
- MSP43P低功耗设计
- ADC10 和 Comparator_A+
- 串行通信模块
- Grace
- 电容式触摸按键解决方案

Code Composer Studio 简介

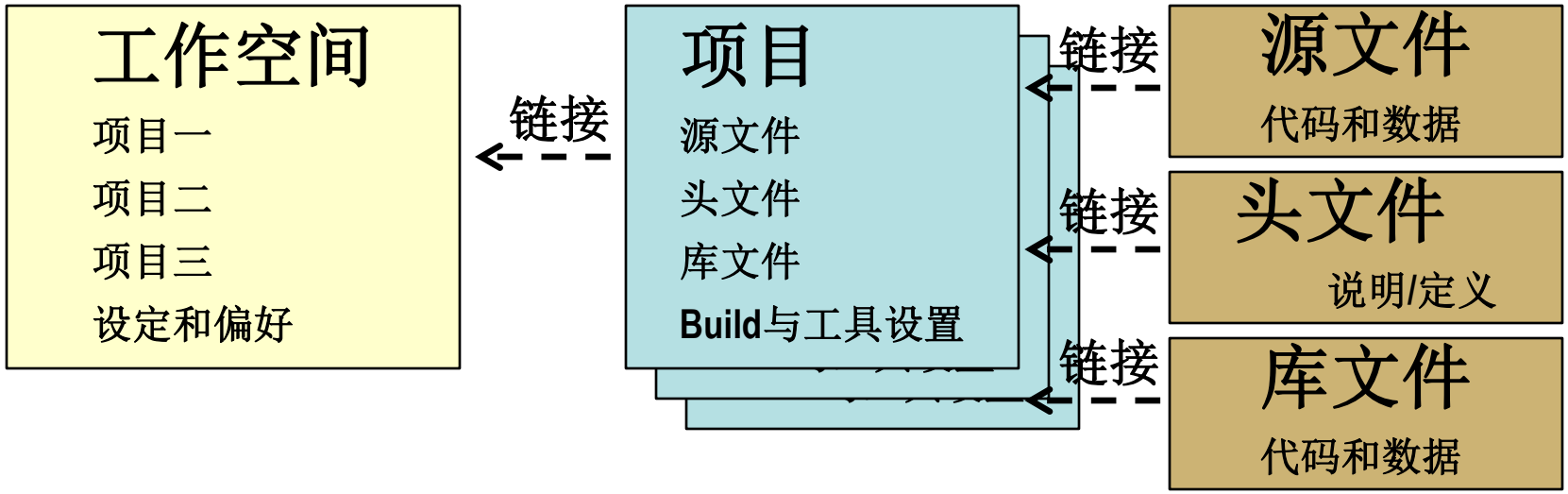
- 用于 TI 嵌入式处理器的集成型开发环境 (IDE)
  - 包括调试器、编译器、编辑器、仿真器、操作系统 (OS) …
  - 该 IDE 基于 Eclipse 开源软件框架
  - 由 TI 对其进行扩展以支持TI 嵌入式控制器
- CCSv5 基于成熟的 Eclipse（在 CCS 5.1 中采用的是 version 3.7）
  - 未来的 CCS 版本将使用 Eclipse 的最新版本
  - 由其他供应商提供的全兼容型 Eclipse 插件或使用 TI 工具并将它们置于一种现有的 Eclipse 环境之中
  - 用户能充分利用 Eclipse 的最新特性
- 集成更多的工具
  - OS 应用程序开发工具（Linux、Android…）
  - 代码分析、源控制…
- 很快支持 Linux
- 低成本！445 或 495 美元



常见任务

- 创建新项目
  - 采用模板即可非常简单地创建针对某款器件的新项目
- **Build**选项
  - 用户可使用**Build**选项对项目进行编译配置
  - 选项的更新通过编译器的发布来提供，而不依赖于 **CCS** 的更新升级
- 共享项目
  - 用户可非常方便共享项目，并包括项目的版本控制等
  - 简化操作以共享链接资源

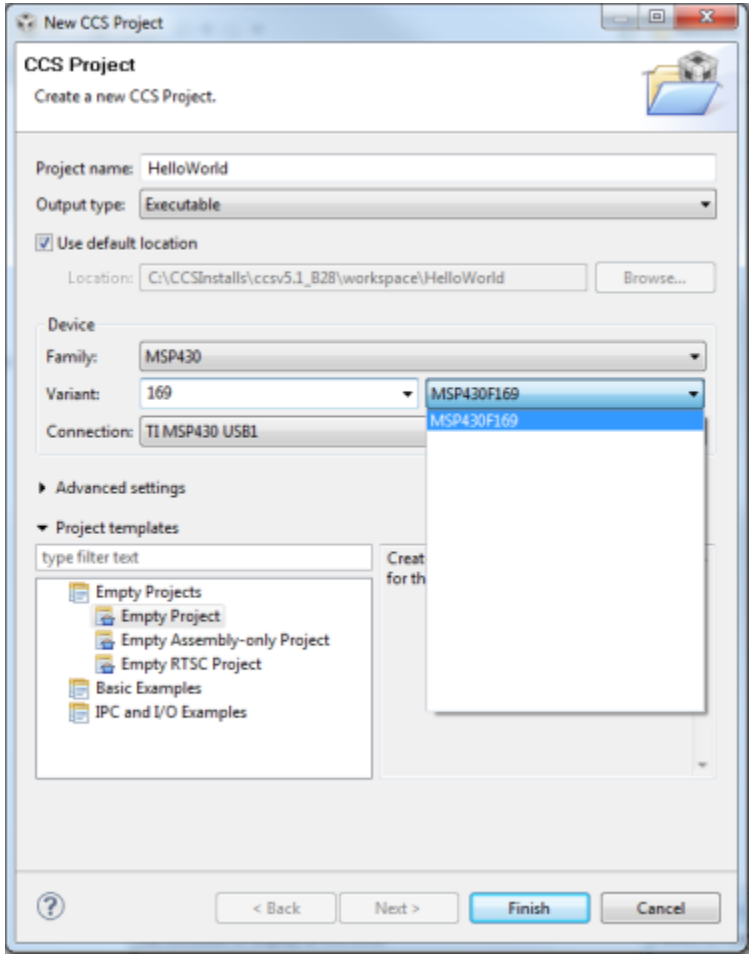
工作空间与方案



工作空间包含您的设置与偏好，以及至您的项目的链接。
将项目从工作空间中删除只是删除了链接，而并非设计文件

一个工程项目包含了您的Build选项与工具设置，以及至您的输入文件的链接。
将文件从工作空间中删除只是删除了链接，而并非设计文件

项目向导



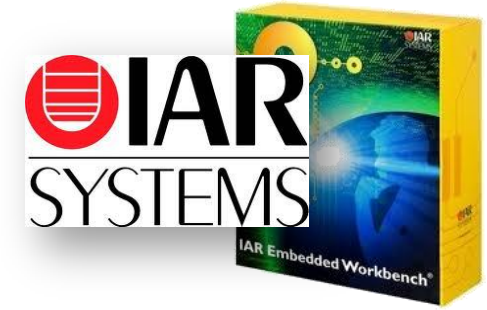
- **单页向导满足大多数情况的使用要求**
  - “下一个”按钮将在某个模板需要附加设置显示
- **包含调试器设置**
  - 假如选取了某款特定的器件，用户还可选择其调试器，并生成 ccxml 文件
- **采用默认设置使其简单易用**
  - 一些高级配置编译器版本、字节存储顺序等在 Advanced Settings下

各种不同的 IDE 选项

可提供免费的集成型开发环境 (IDE)

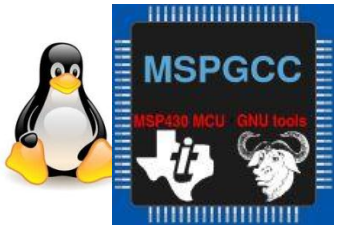
Code Composer Studio

- 基于 Eclipse 的 IDE（编译器、调试器、链接器等），适用于所有的 TI 嵌入式处理器
- 无限制版本售价 495 美元
- 可提供免费版本！
  - 免费的 16kB 代码空间限制版本可供下载
  - 可提供免费、全功能、120 天试用期限版本



IAR Embedded Workbench

- 功能强大的第三方 IDE 产品，配有项目管理工具和编辑器。包括用于所有 MSP430 器件的配置文件。
- 可提供免费版本！
  - 免费的 4/8/16kB 代码空间限制 (Kickstart) 版本可供下载
  - 可提供免费、全功能、30 天试用期限版本

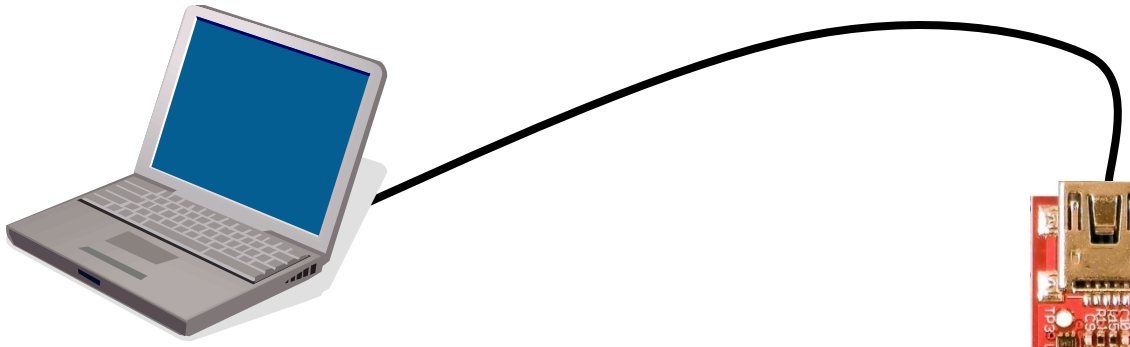


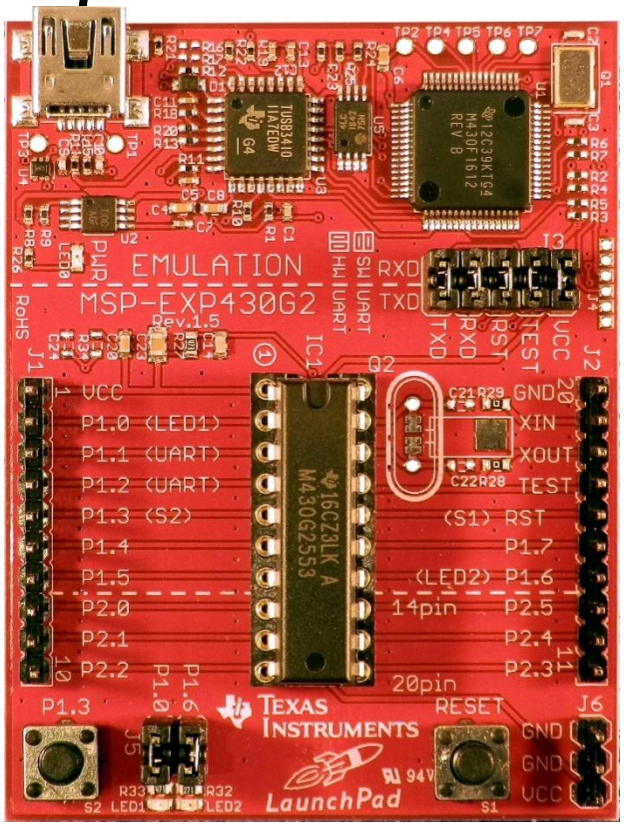
MSPGCC

- 用于 MSP430 的免费、开源、GCC 工具链
- 包含 GNU C 编译器 (GCC)、汇编器和链接器 (binutils)、调试器 (GDB)
- 工具可在 Windows、Linux、BSD 及其他大多数 Unix 版本的操作系统上使用
- 更多详情敬请访问：http://mspgcc.sourceforge.net/

可提供其他的 MSP430 IDE 选项！更多详情敬请访问 www.ti.com/msp430tools

Lab1：Code Composer Studio

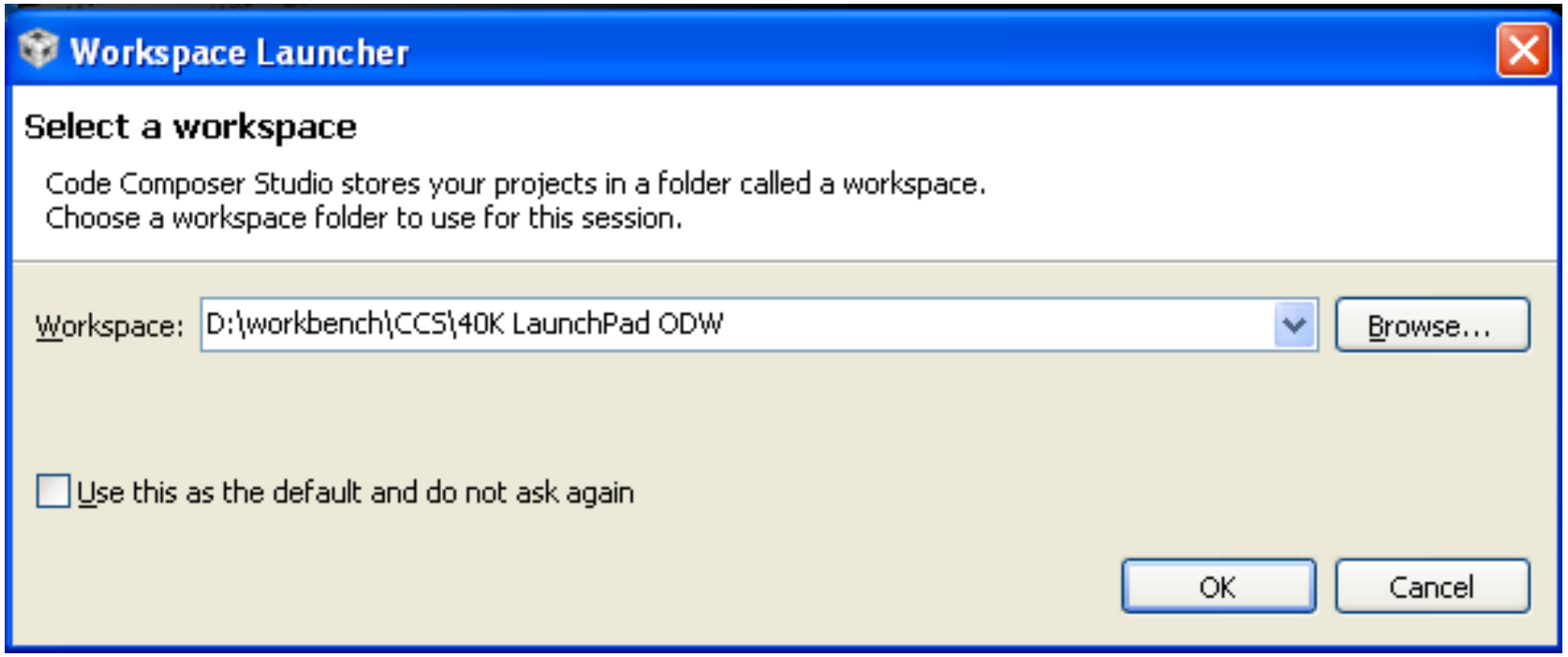




- **Lab1：**
  - 创建一个新的工作空间
  - 创建 **Lab1** 项目
  - 加进**temperature sense demo**代码
  - 编译并运行

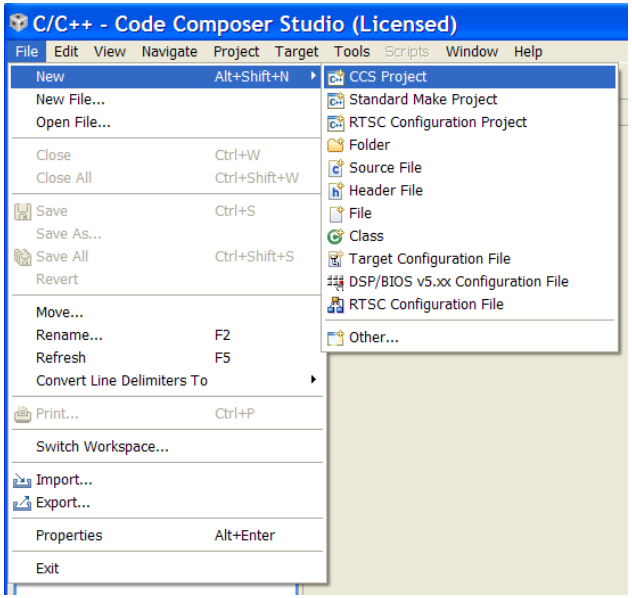
第一步：创建 CCS 工作空间

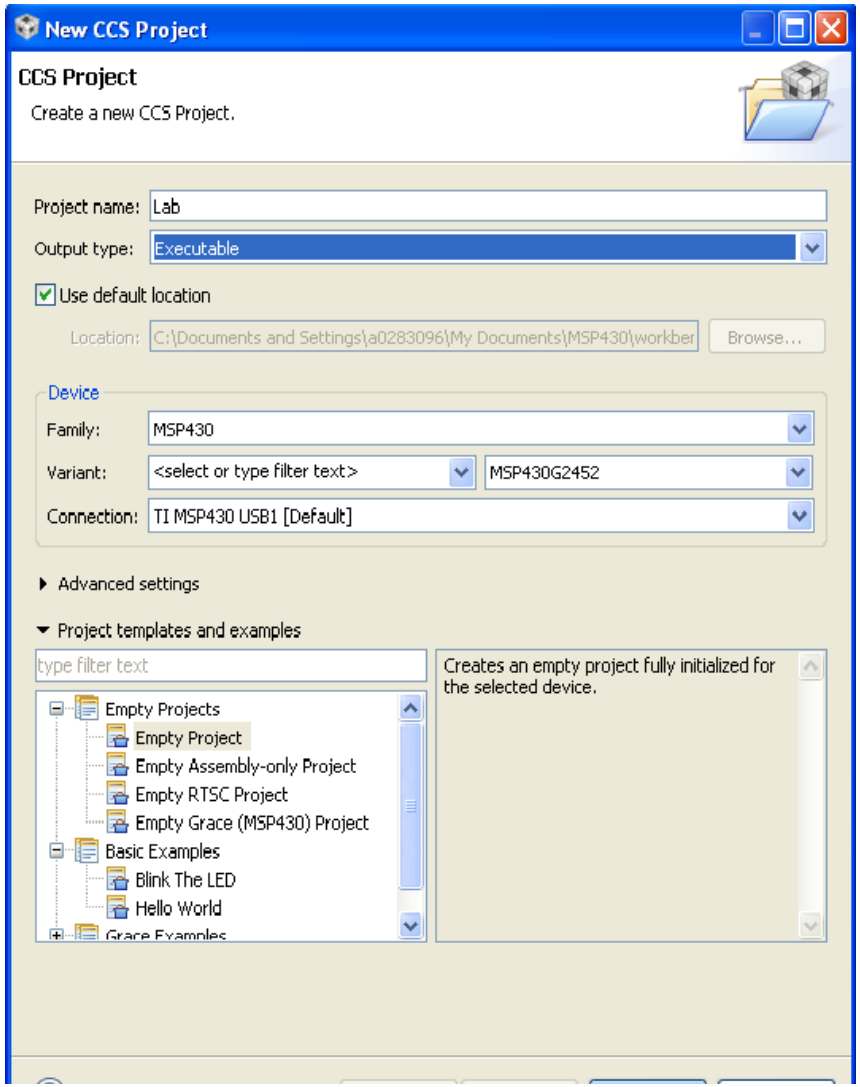
- 将 Lab 文件置于您的电脑上
- 启动 CCS v5
- *选择一个“工作空间”位置*



第二步：创建一个 CCS 项目

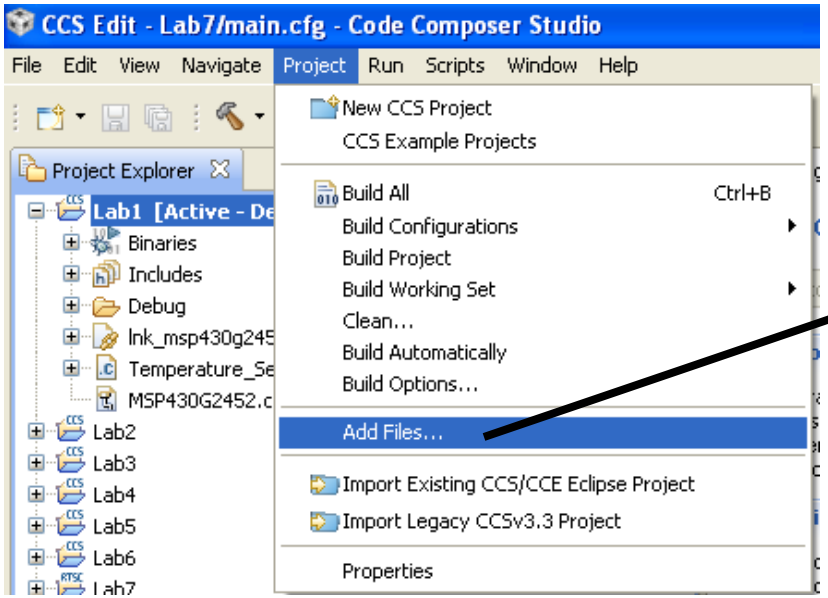
- File > New > CCS Project
- Project name：Lab1
- Device>Family：MSP430
- Variant：MSP430G2231
- Project templates and examples:
  Empty Project

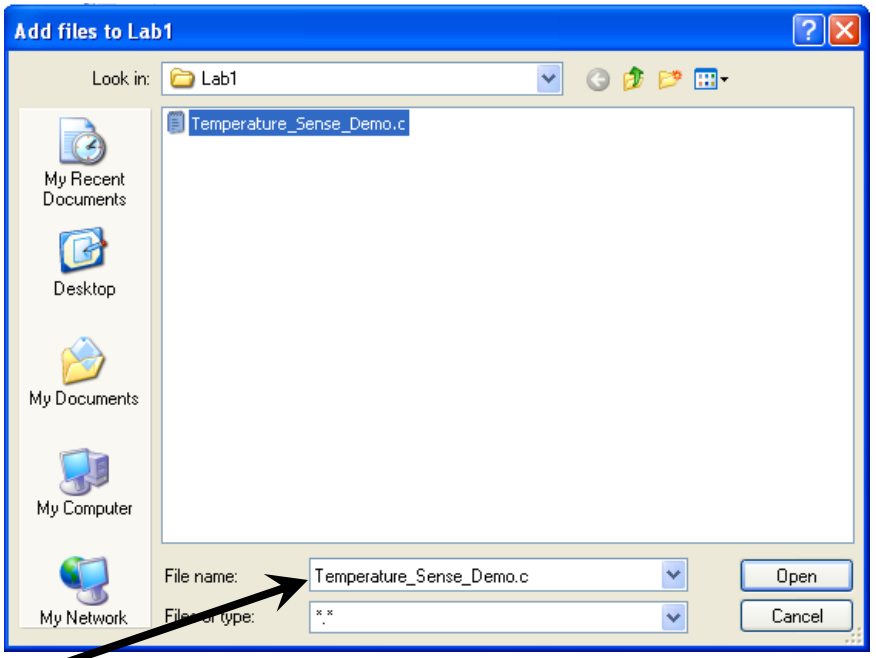


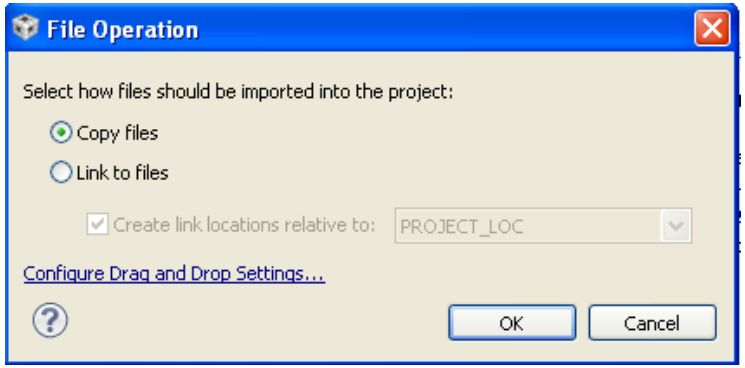


第三步：在 CCS 项目中加进一个文件

- Project > Add Files
- 选择 Lab 所在的文件夹
- 并选择：
  Temperature_Sense_Demo.c



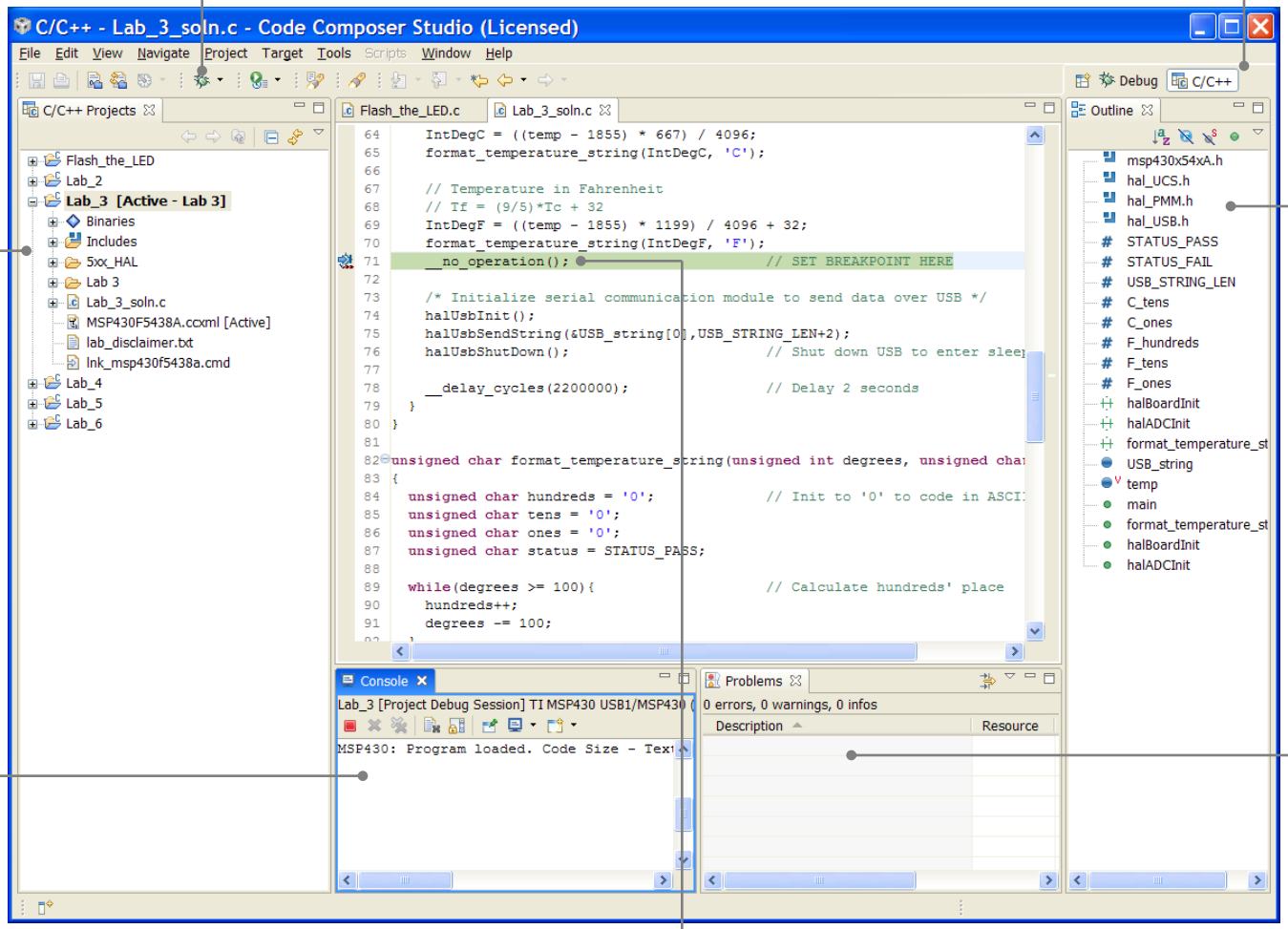




CCS 窗口 – C/C++ 透视概览

一键式项目调试

独立的调试与 C/C++ 项目透视图



项目视图
- 所有项目的列表

项目概要
- 快捷了解项目组成部分

控制台
- 构建信息

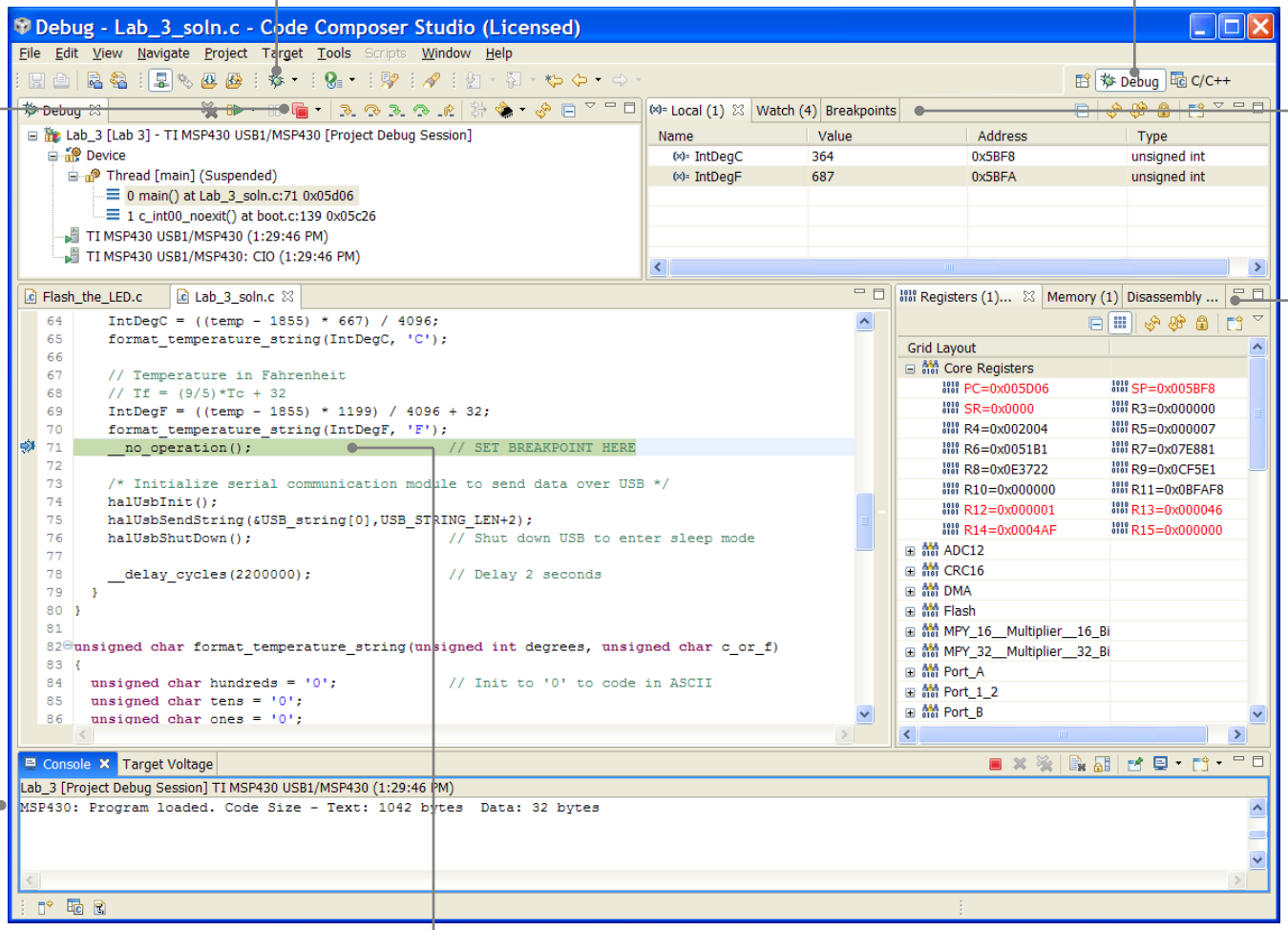
代码窗口
- 实时断点，语法高亮显示

问题观察
- 信息、警告、差错

CCS 窗口 – 调试透视概览

一键式项目调试

独立的调试与 C/C++ 项目透视图



目标控制
- 起动
- 停止
- 暂停
- 步进
- 堆栈跟踪

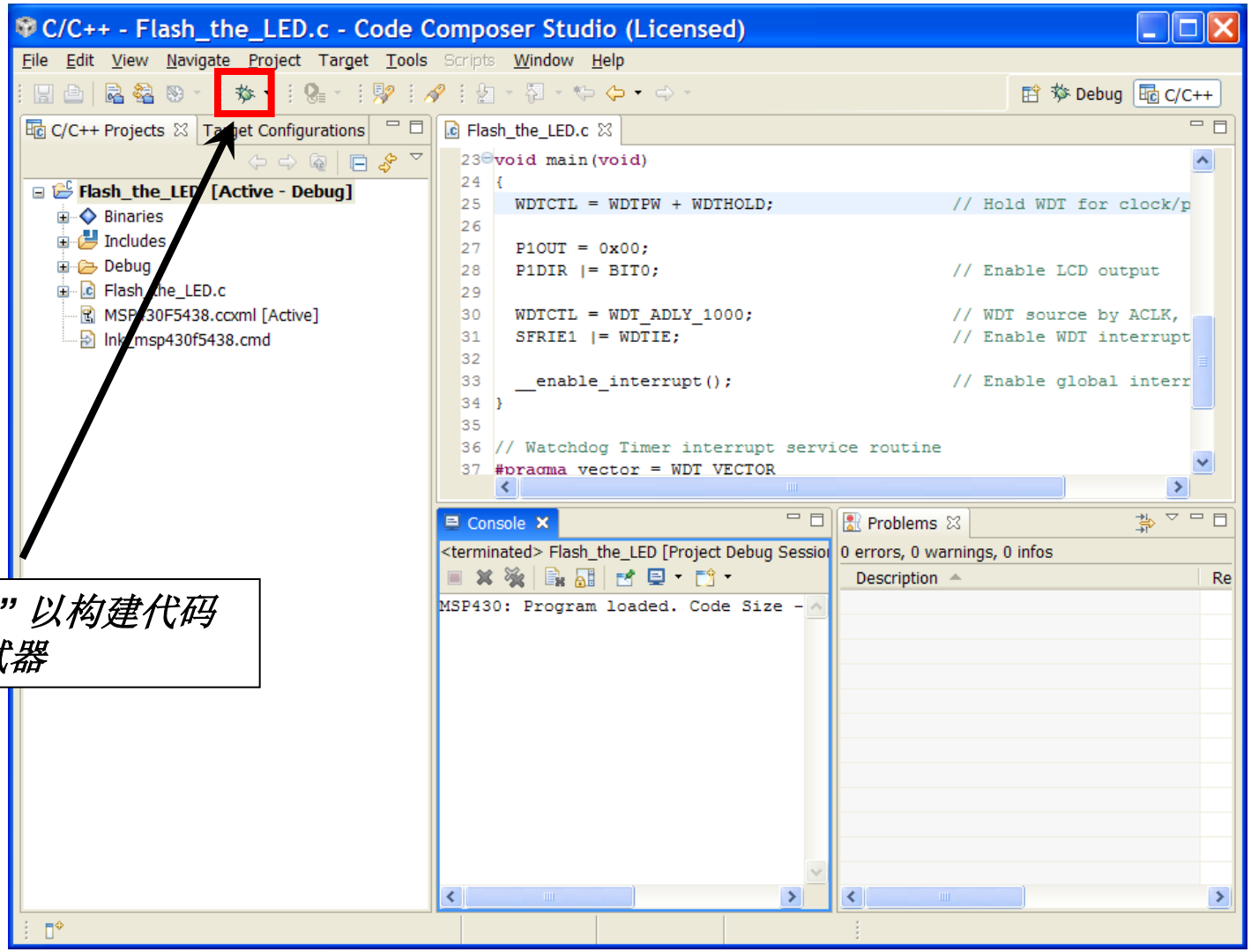
高度可配置的窗口布局
- 用户偏好
- 插件支持

实时、系统内 MSP430 信息
- 寄存器访问
- 闪存、RAM、信息段访问
- 反汇编视图

程序长度信息

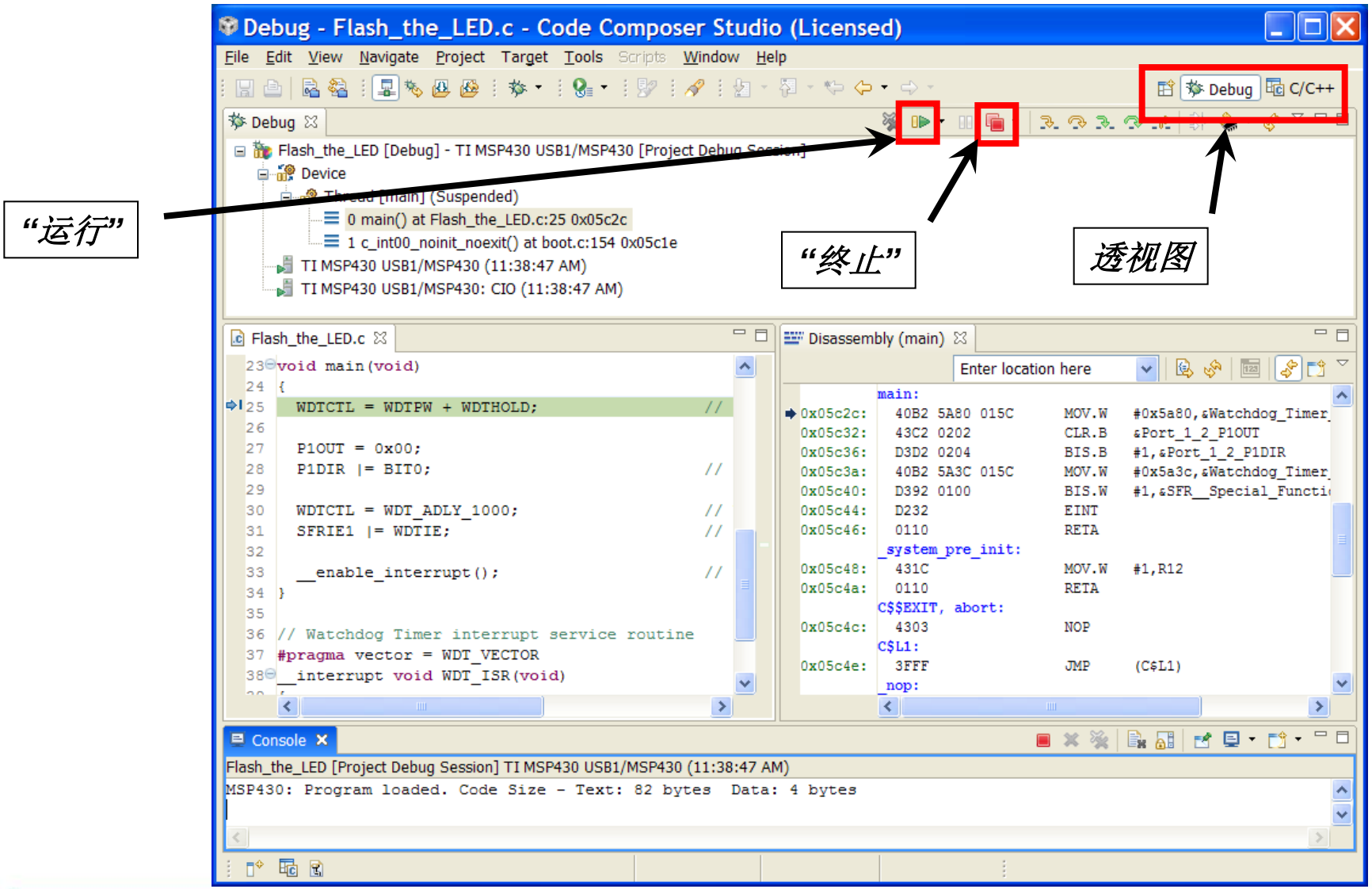
代码窗口
- 实时断点，语法高亮显示

第四步：Build及调试一个 CCS 项目



点击 "DEBUG" 以构建代码并启动调试器

第五步：运行、终止一个 CCS 项目



议程

- 介绍 Value Line 系列
- Code Composer Studio
- CPU 与基本时钟模块
- 中断与 GPIO
- Timer_A 与 WDT+
- MSP430低功耗设计
- ADC10 和 Comparator_A+
- 串行通信模块
- Grace
- 电容式触摸按键解决方案

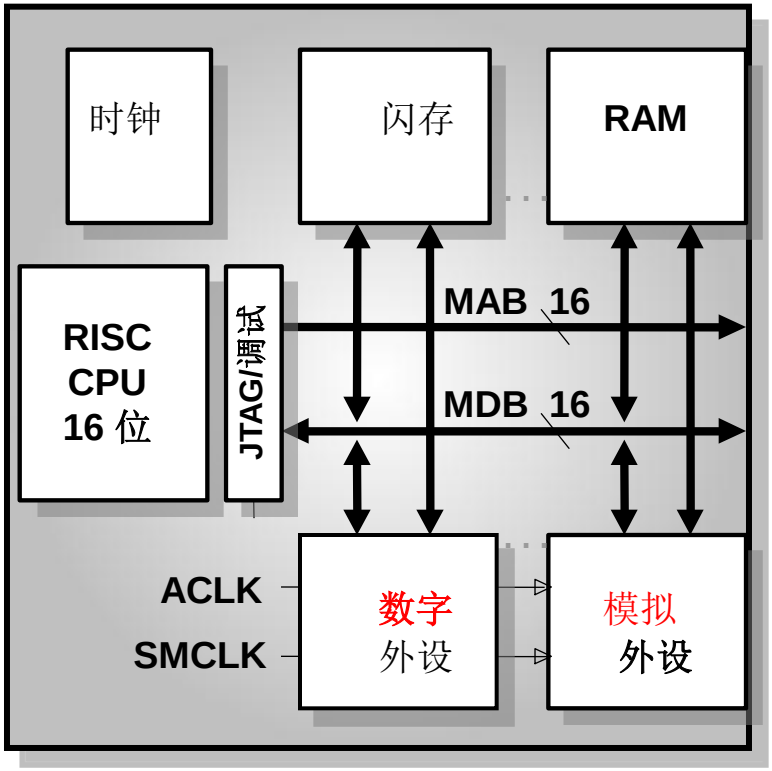
MSP430G2xx 结构

超低功耗

- 0.1uA 断电模式流耗
- 0.8uA 待机模式流耗
- 220uA / 1MIPS
- <1us 时钟启动时间
- <50nA 端口漏电流
- 零功耗欠压复位 (BOR)

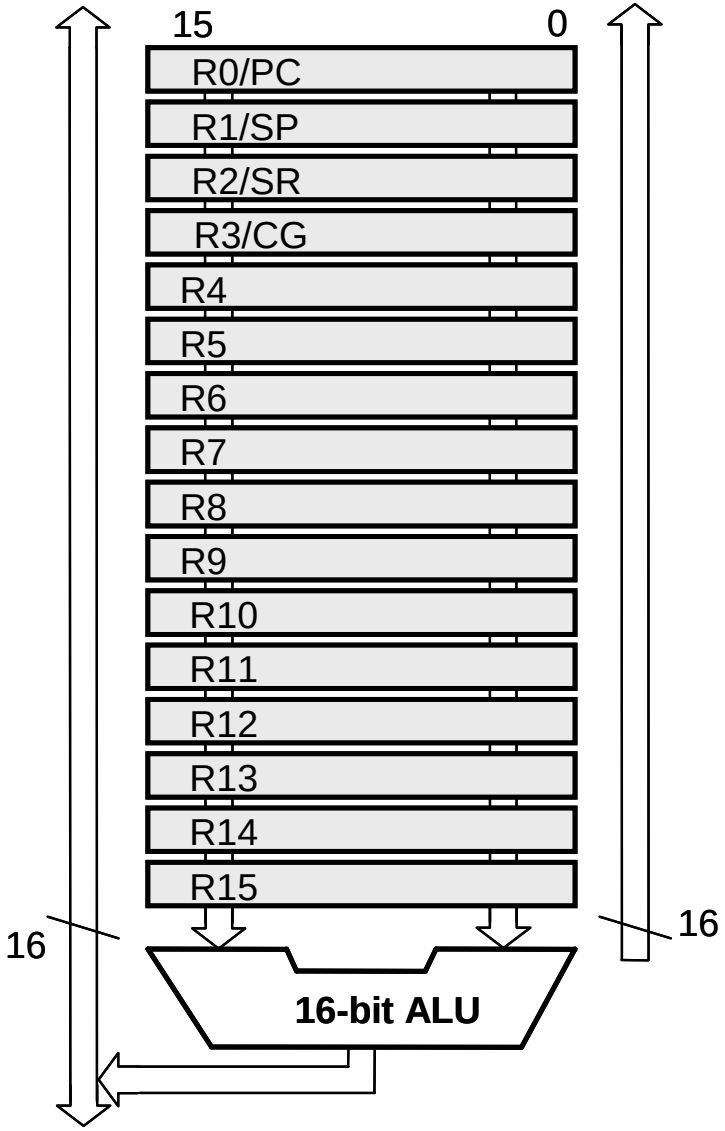
超灵活

- 0.5k 至 16kB 系统内可编程 (ISP) 闪存
- 16 位定时器
- SPI、I2C
- 10 位 ADC
- 嵌入式仿真



16 位 RISC CPU

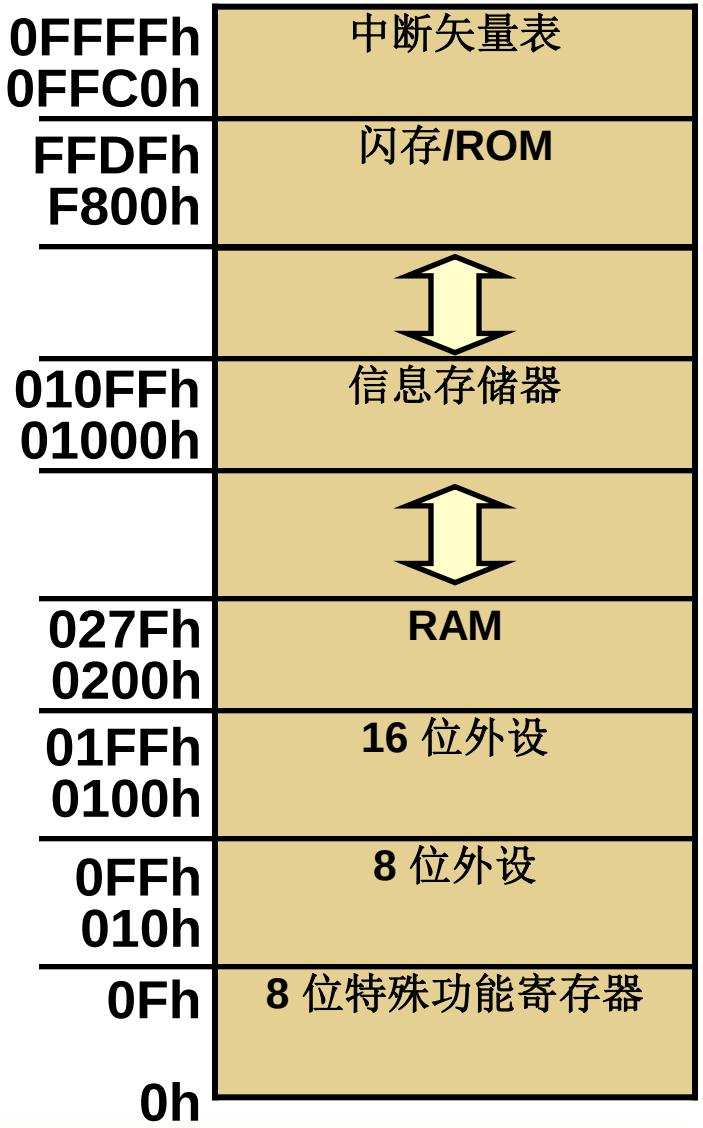
- 单周期寻址寄存器文件
  - 4 种专用型
  - 12 种通用型
  - 无累加器瓶颈
- RISC 架构
  - 27 条核心指令
  - 24 条仿真指令
  - 7 种地址模式
- Atomic内存至内存寻址
- 位、字节和字处理
- 常数发生器



存储器映射

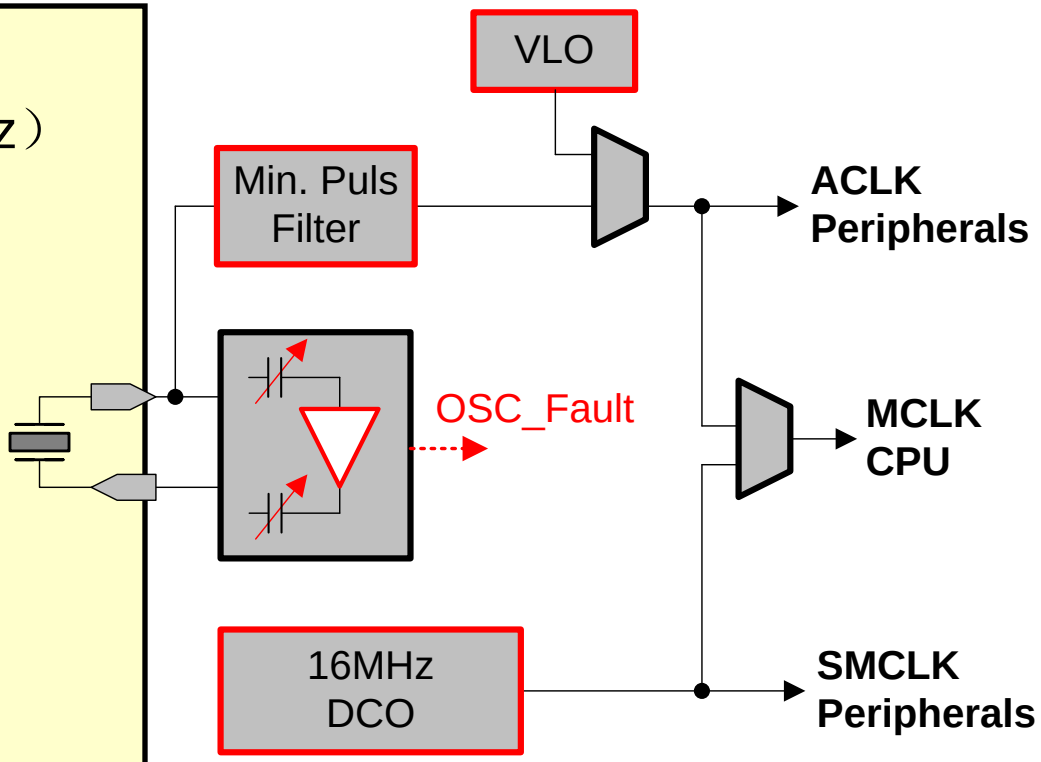
- 闪存可编程：通过 JTAG 或系统内 (ISP)
- 编程电压低至 2.2V，单字节或单字编程
- 主存储器：每段512 字节 (0-n). 可分段或全部擦除
- 信息存储器：每段64 字节 (A-D)
  - Section A 包含器件专用的校准数据，并可锁定
- 可编程闪存定时发生器

示出的是 G2231



时钟系统

- **超低功耗/低频振荡器 (VLO)**
  - 4 – 20kHz（典型值为 12kHz）
  - 500nA 待机流耗
  - 0.5%/° C 和 4%/V 漂移
- **晶体振荡器 (LFXT1)**
  - 片内集成可编程负载电容
  - 故障保险 OSC_Fault
  - 脉冲滤波器
- **数字控制振荡器 (DCO)**
  - 0 至 16MHz
  - ± 3% 容差
  - 出厂校准（保存在闪存中）



MCU PUC后，MCLK 和 SMCLK 由 DCOCLK 提供（约 1.1 MHz）。在 LF 模式中，ACLK 由 LFXT1CLK 采用一个 6pF 内部负载电容提供。

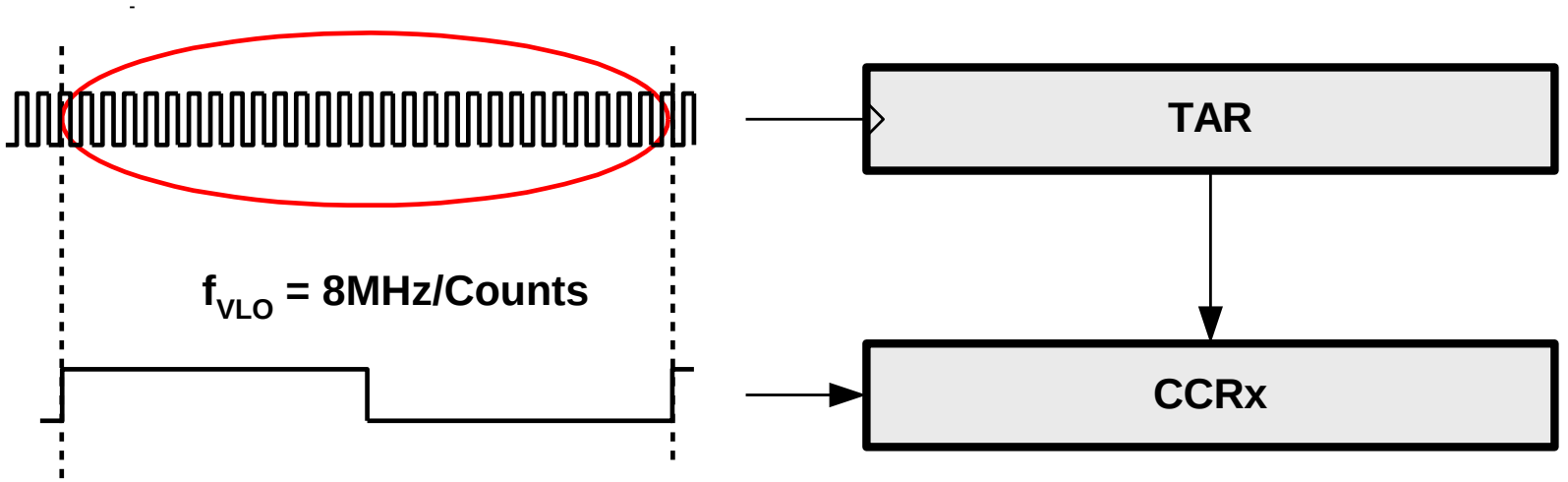
G2xxx –DCO校正

DCO Calibration Data (provided from factory in flash info memory segment A)			
DCO Frequency	Calibration Register	Size	Address
1 MHz	CALBC1_1MHz	byte	010FFh
	CALDCO_1MHz	byte	010FEh
8 MHz	CALBC1_8MHz	byte	010FDh
	CALDCO_8MHz	byte	010FCh
12 MHz	CALBC1_12MHz	byte	010FBh
	CALDCO_12MHz	byte	010FAh
16 MHz	CALBC1_16MHz	byte	010F9h
	CALDCO_16MHz	byte	010F8h

```
 // Setting the DCO to 1MHz
 if (CALBC1_1MHZ ==0xFF || CALDCO_1MHZ == 0xFF)
   while(1);                // Erased calibration data? Trap!
BCSCTL1 = CALBC1_1MHZ;      // Set range
DCOCTL  = CALDCO_1MHZ;      // Set DCO step + modulation
```

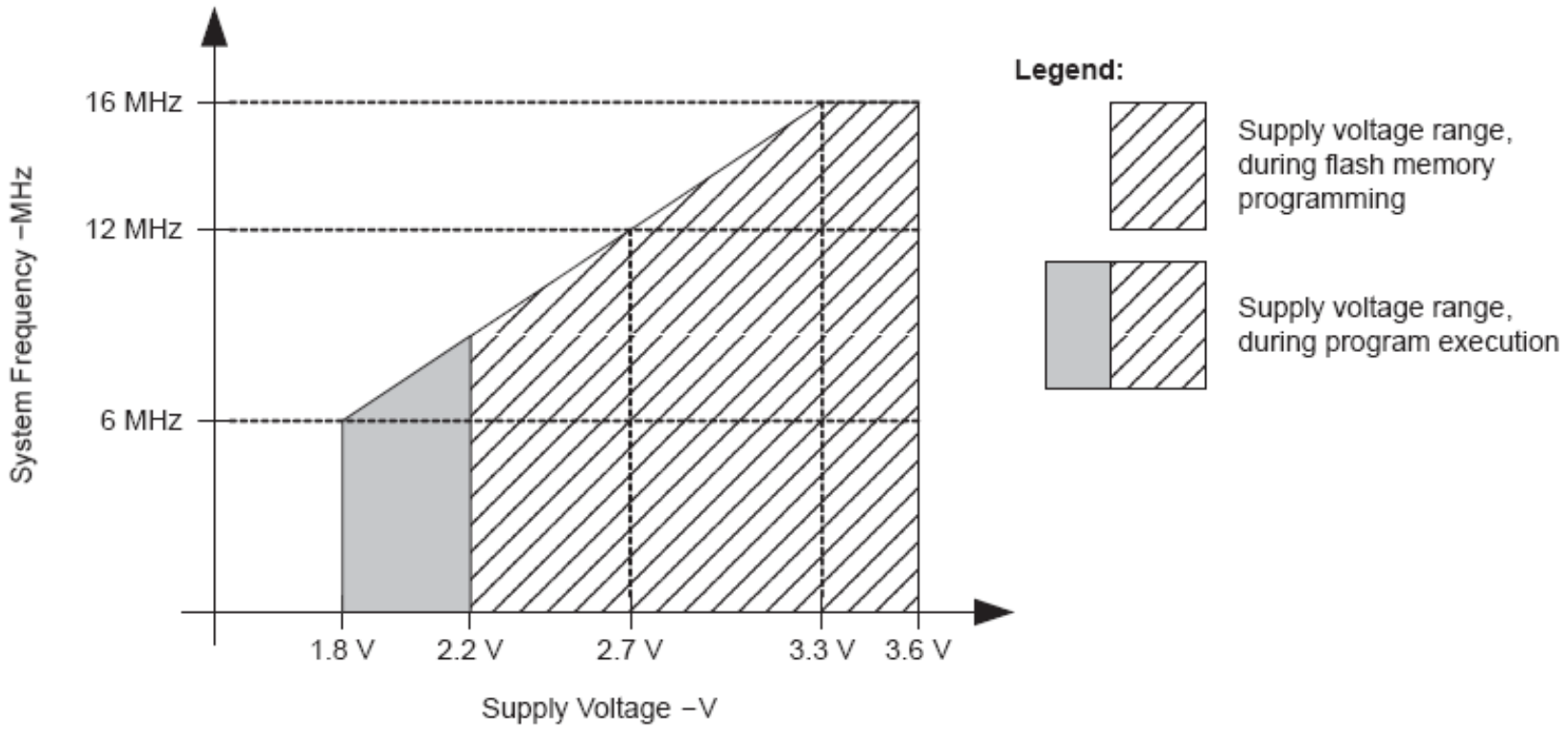
- G2xx1 器件只具有 1MHz DCO 校正参数。若需要较高的频率，客户必须自行校准。
- G2xx2 和 G2xx3 具有所有 4 个DCO校正参数校准值。

VLO 的校准



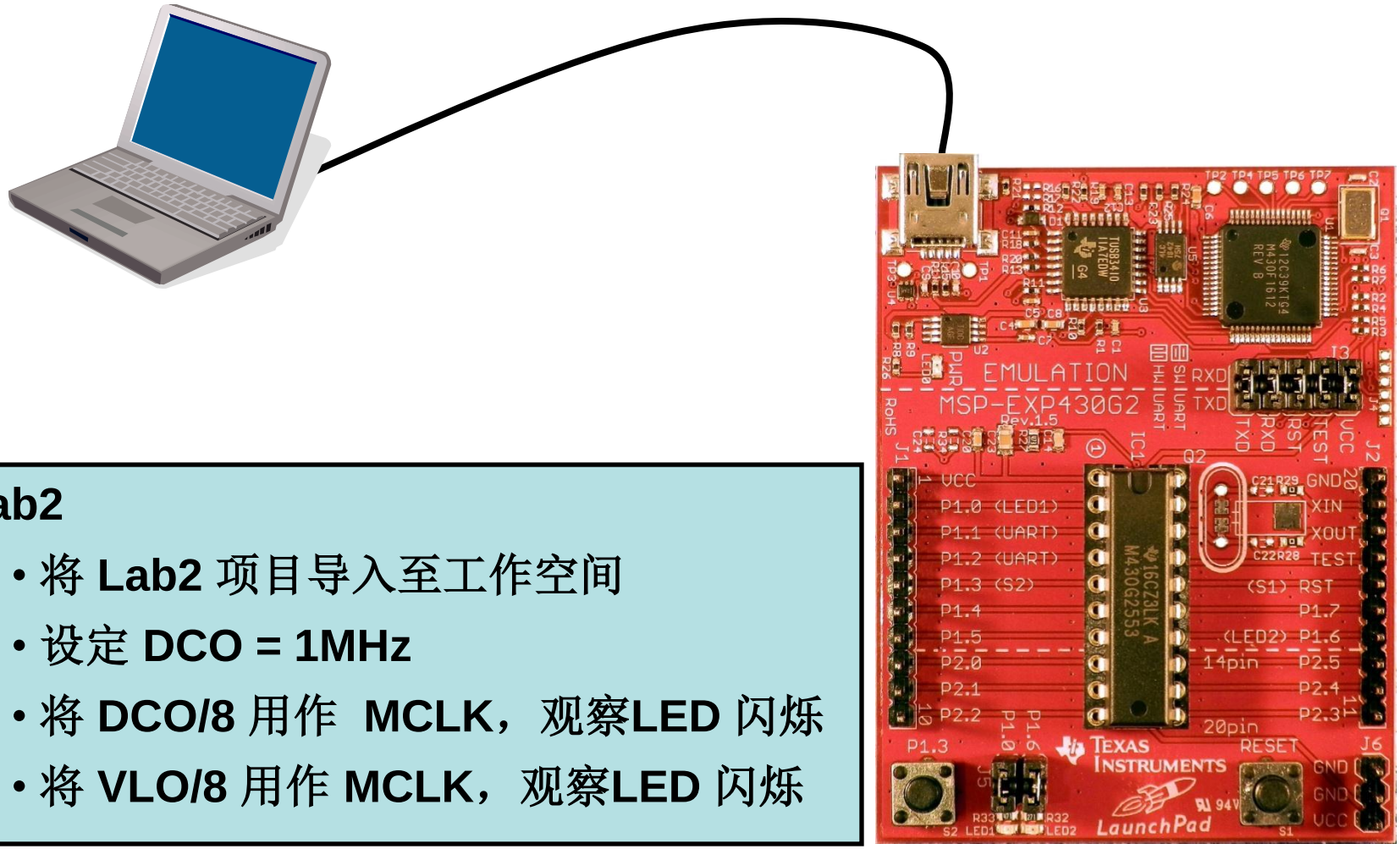
- 在运行期间可对 VLO 进行校准
- 采用校准的 1MHz DCO为Timer_A提供时钟
- 利用 VLO 提供的 ACLK/8， 捕获其上升沿
- 经运算f_{VLO} = 8MHz/计数
- *可在网上查询代码库 (SLAA340)*

系统 MCLK 和 Vcc



- 时钟速度与所需的 Vcc 必需匹配
- 需要外部 LDO 稳压器
- 如果 Vcc < 选定频率所要求的最小电压值，则会导致程序运行不确定性
- 所有 G2xxx 器件的工作频率均可高达 16MHz

Lab2：基本时钟配置



- **Lab2**
  - **将 Lab2 项目导入至工作空间**
  - **设定 DCO = 1MHz**
  - **将 DCO/8 用作 MCLK，观察LED 闪烁**
  - **将 VLO/8 用作 MCLK，观察LED 闪烁**

Lab 2：

```
 // Configure Basic Clock
BCSCTL1 = ____________; // Set range
DCOCTL = _____________;// Set DCO step + modulation
BCSCTL3 |= LFXT1S_2;// Set LFXT1
```

```
 // Configure MCLK
BCSCTL2 |=_________ + DIVM_3; // Set MCLK
```

- 参考用户指南、数据手册及原理图

Lab 2：2xx 用户指南中的 BCSCTL2

5.3.3 BCSCTL2, Basic Clock System Control Register 2

7	6	5	4	3	2	1	0
SELMx		DIVMx		SELS	DIVSx		DCOR(1)(2)
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

SELMx	Bits 7-6	Select MCLK. These bits select the MCLK source.	
		00	DCOCLK
		01	DCOCLK
		10	XT2CLK when XT2 oscillator present on-chip. LFXT1CLK or VLOCLK when XT2 oscillator not present on-chip.
		11	LFXT1CLK or VLOCLK
DIVMx	Bits 5-4	Divider for MCLK	
		00	/1
		01	/2
		10	/4
		11	/8
SELS	Bit 3	Select SMCLK. This bit selects the SMCLK source.	
		0	DCOCLK
		1	XT2CLK when XT2 oscillator present. LFXT1CLK or VLOCLK when XT2 oscillator not present
DIVSx	Bits 2-1	Divider for SMCLK	
		00	/1
		01	/2
		10	/4
		11	/8
DCOR	Bit 0	DCO resistor select. Not available in all devices. See the device-specific data sheet.	
		0	Internal resistor
		1	External resistor

Lab 2：MSP430G2231 标头文件中的 BCSCTL2

```
262 #define DIVA_2                  (0x20)          /* ACLK Divider 2: /4 */
263 #define DIVA_3                  (0x30)          /* ACLK Divider 3: /8 */
264
265 #define DIVS0                   (0x02)          /* SMCLK Divider 0 */
266 #define DIVS1                   (0x04)          /* SMCLK Divider 1 */
267 #define SELS                    (0x08)          /* SMCLK Source Select 0:DCOCLK / 1:XT2CLK/LFXTCLK */
268 #define DIVM0                   (0x10)          /* MCLK Divider 0 */
269 #define DIVM1                   (0x20)          /* MCLK Divider 1 */
270 #define SELM0                   (0x40)          /* MCLK Source Select 0 */
271 #define SELM1                   (0x80)          /* MCLK Source Select 1 */
272
273 #define DIVS_0                  (0x00)          /* SMCLK Divider 0: /1 */
274 #define DIVS_1                  (0x02)          /* SMCLK Divider 1: /2 */
275 #define DIVS_2                  (0x04)          /* SMCLK Divider 2: /4 */
276 #define DIVS_3                  (0x06)          /* SMCLK Divider 3: /8 */
277
278 #define DIVM_0                  (0x00)          /* MCLK Divider 0: /1 */
279 #define DIVM_1                  (0x10)          /* MCLK Divider 1: /2 */
280 #define DIVM_2                  (0x20)          /* MCLK Divider 2: /4 */
281 #define DIVM_3                  (0x30)          /* MCLK Divider 3: /8 */
282
283 #define SELM_0                  (0x00)          /* MCLK Source Select 0: DCOCLK */
284 #define SELM_1                  (0x40)          /* MCLK Source Select 1: DCOCLK */
285 #define SELM_2                  (0x80)          /* MCLK Source Select 2: XT2CLK/LFXTCLK */
286 #define SELM_3                  (0xC0)          /* MCLK Source Select 3: LFXTCLK */
287
```

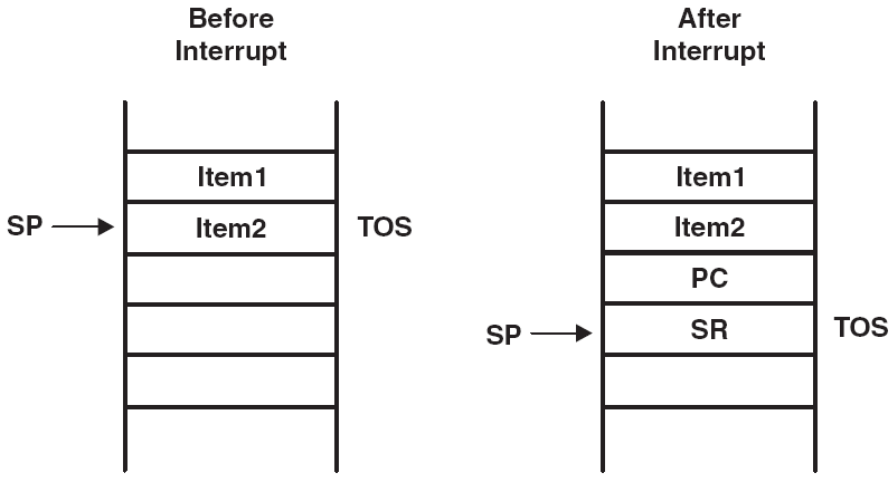
议程

- 介绍 Value Line 系列
- Code Composer Studio
- CPU 与基本时钟模块
- 中断与 GPIO
- Timer_A 与 WDT+
- MSP430低功耗设计
- ADC10 和 Comparator_A+
- 串行通信模块
- Grace
- 电容式触摸按键解决方案

中断和堆栈

进入中断服务程序

- 当前执行的指令已完成
- 指向下一条指令的 PC 被推送至堆栈上
- SR 被推送至堆栈上
- 选择了具有最高优先级的中断
- 中断请求标志为单源中断标志时自动复位；若为多源中断标志则保持于设定状态，由软件控制
- SR 被清零；这将终止任何低功耗模式；由于 GIE 位被清零，因此将禁止执行更多的中断
- 中断向量的内容被装入 PC；程序将利用位于该地址的中断服务例程继续执行



Vector Table – G2231

Interrupt Source	Interrupt Flag	System Interrupt	Word Address	Priority
Power-up External Reset Watchdog Timer+ Flash key violation PC out-of-range	PORIFG RSTIFG WDTIFG KEYV	Reset	0FFFEh	31 (highest)
NMI Oscillator Fault Flash memory access violation	NMIIFG OFIFG ACCVIFG	Non-maskable Non-maskable Non-maskable	0FFFCh	30
			0FFFAh	29
			0FFF8h	28
			0FFF6h	27
Watchdog Timer+	WDTIFG	maskable	0FFF4h	26
Timer_A2	TACCR0 CCIFG	maskable	0FFF2h	25
Timer_A2	TACCR1 CCIFG TAIFG	maskable	0FFF0h	24
			0FFEEh	23
			0FFECh	22
ADC10	ADC10IFG	maskable	0FFEAh	21
USI	USIIFG USISTTIFG	maskable	0FFE8h	20
I/O Port P2 (2)	P2IFG.6 P2IFG.7	maskable	0FFE6h	19
I/O Port P1 (8)	P1IFG.0 to P1IFG.7	maskable	0FFE4h	18
			0FFE2h	17
			0FFE0h	16
Unused			0FFDEh to 0FFCDh	15 - 0

中断处理函数编程

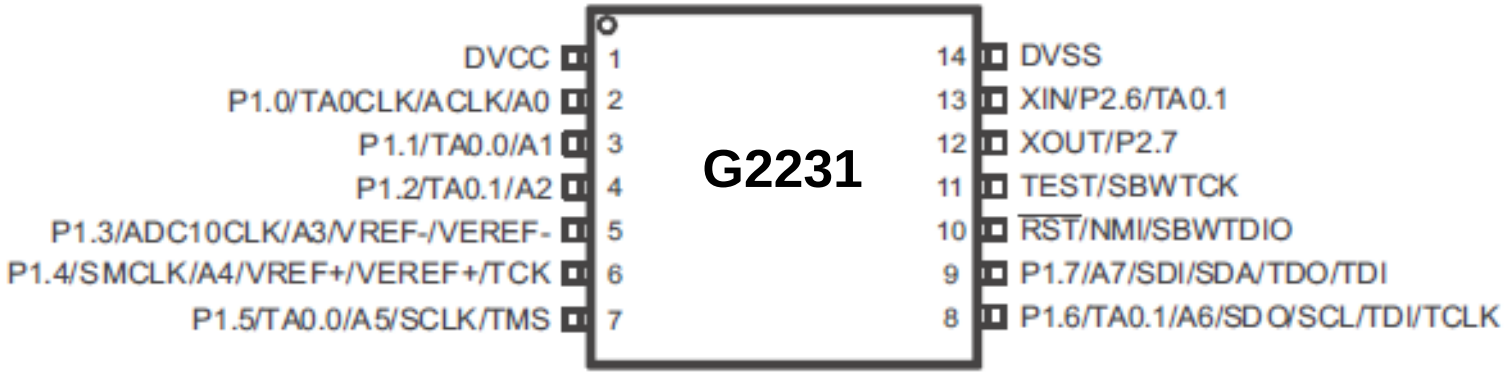
```
#pragma vector=WDT_VECTOR
__interrupt void WDT_ISR(void)
{
    IE1 &= ~WDTIE;                  // disable interrupt
    IFG1 &= ~WDTIFG;                // clear interrupt flag
    WDTCTL = WDTPW + WDTHOLD;       // put WDT back in hold state
    BUTTON_IE |= BUTTON;            // Debouncing complete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

#pragma vector – 下面的函数是一个用于所列矢量的 ISR

__interrupt void – 识别 ISR 名称

无特别需要的返回值

GPIO 端口



GPIO 寄存器

- 输入寄存器 PxIN
- 输出寄存器 PxOUT
- 方向寄存器 PxDIR
- 电阻启用 PxREN
- 功能选择 PxSEL
- 功能选择2 PxSEL2
- 中断边缘 PxIES
- 中断启用 PxIE
- 中断标记 PxIFG

用于 GPIO 中断

GPIO 代码范例

```
P1DIR |= BIT4;
P1SEL |= BIT4;
```

```
P1DIR |= BIT0;
P1OUT |= BIT0;
```

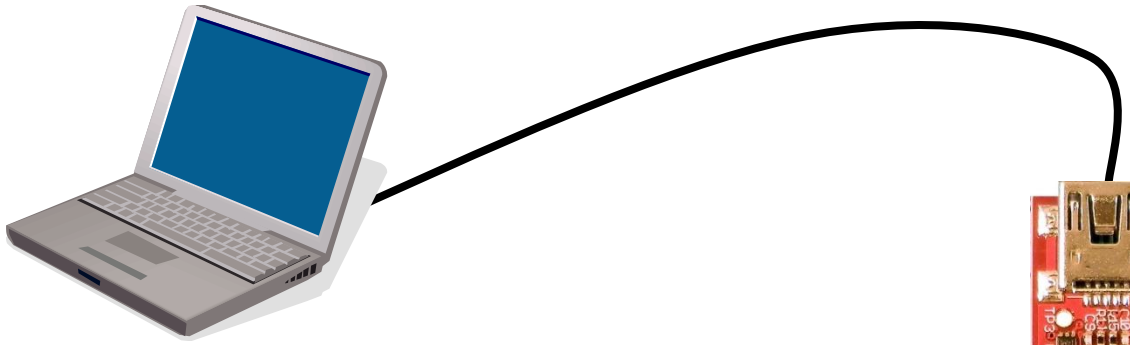
引脚复用

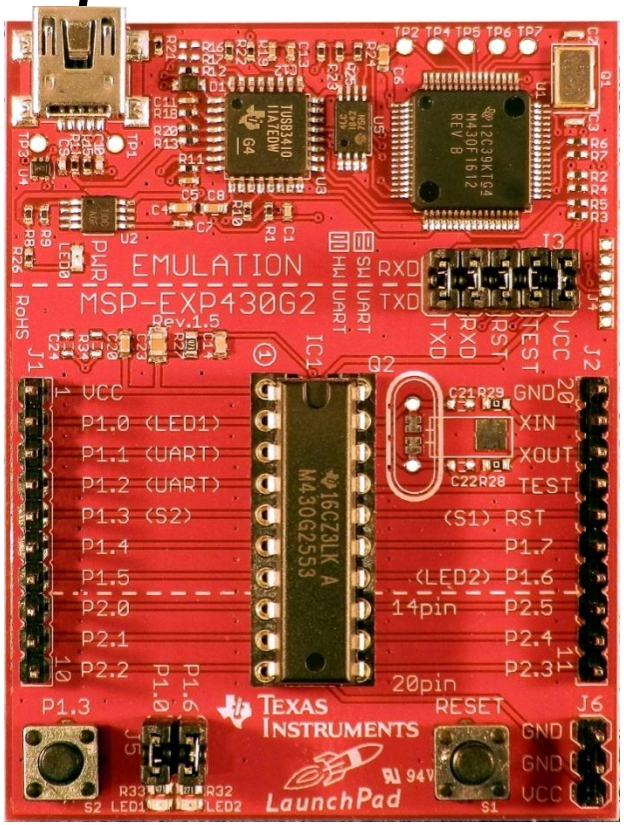
Table 16. Port P1 (P1.4) Pin Functions

PIN NAME (P1.x)	x	FUNCTION	CONTROL BITS / SIGNALS[1]					
			P1DIR.x	P1SEL.x	P1SEL2.x	ADC10AE.x (INCH.x=1)[2]	JTAG Mode	CAPD.y
P1.4/	4	P1.x (I/O)	I: 0; O: 1	0	0	0	0	0
SMCLK/		SMCLK	1	1	0	0	0	0
TA0.2/		TA0.2	1	1	1	0	0	0
		TA0.CCI2A	0	1	1	0	0	0
VREF+[2]/		VREF+	X	X	X	1	0	0
VEREF+[2]/		VEREF+	X	X	X	1	0	0
A4[2]/		A4	X	X	X	1 (y = 4)	0	0
CA4/		CA4	X	X	X	0	0	1 (y = 4)
TCK/		TCK	X	X	X	0	1	0
Pin Osc		Capacitive sensing	X	0	1	0	0	0

- 每个引脚具有多项功能
- 在对应的寄存器选择相应的引脚功能
- 具体详见各器件的数据手册

Lab3：GPIO





Lab3

- 设置 **P1.3** 为按钮
- 设置 **P1.0** 控制 **LED**
- 利用按钮进行触发**LED**翻转

Lab 3:

```
P1DIR |= BIT0; // Set P1.0 to output direction
P1IES |= BIT3; // P1.3 Hi/lo edge
_____ &= ~BIT3; // P1.3 IFG cleared
_____ |= BIT3; // P1.3 interrupt
```

```
 // Port1 interrupt service routine
#pragma vector = __________
__interrupt void Port_1(void)
```

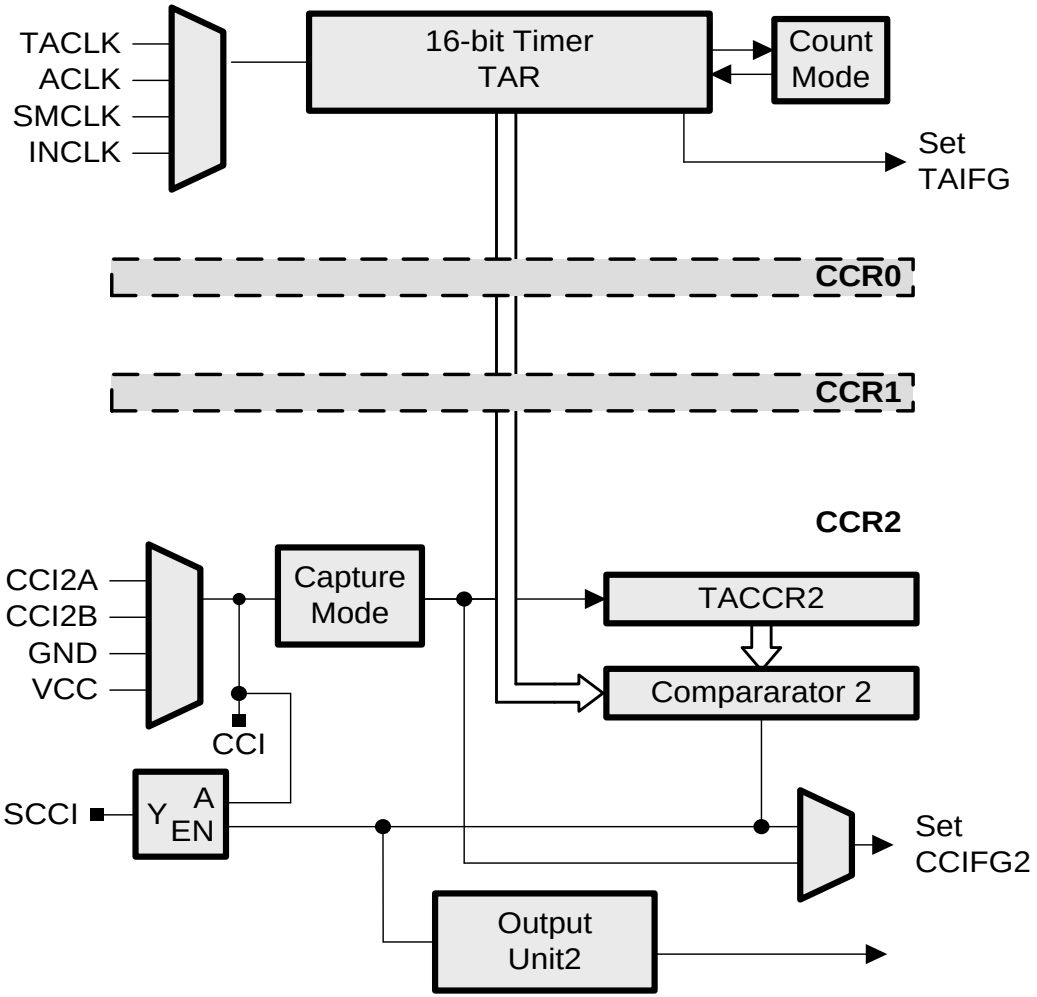
```
 // Port1 interrupt service routine
P1OUT ^= BIT0;   // P1.0 = toggle
______ &= ~BIT3; // P1.3 IFG cleared
```

议程

- 介绍 Value Line 系列
- Code Composer Studio
- CPU 与基本时钟模块
- 中断与 GPIO
- Timer_A 与 WDT+
- MSP430低功耗设计
- ADC10 和 Comparator_A+
- 串行通信模块
- Grace
- 电容式触摸按键解决方案

Timer_A

- 异步 16 位定时器/计数器
- 连续、递增-递减、递增计数模式
- 多个捕获/比较寄存器
- PWM 输出
- 中断向量寄存器用于实现中断快速响应
- 能触发 DMA 传输
- 所有 MSP430 上均有 Timer_A模块



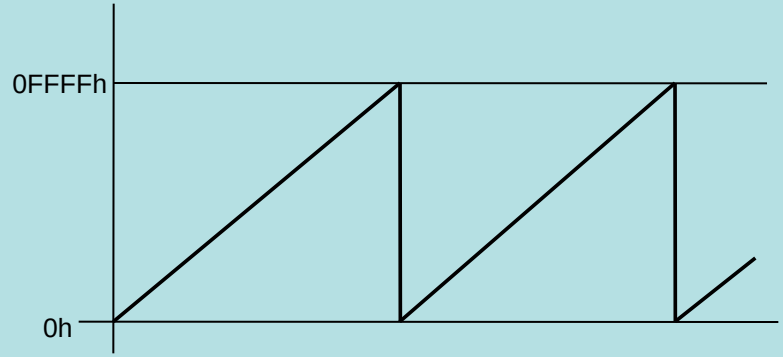
Timer_A 计数模式

停止/暂停

定时器计数停止/暂停

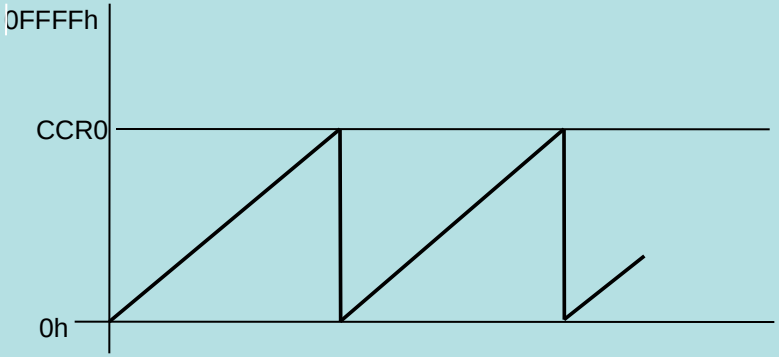
连续

定时器连续递增计数



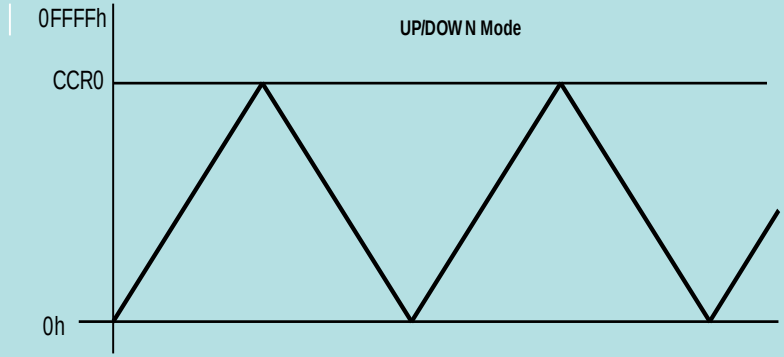
递增

定时器在 0 和 CCR0 之间计数



递增/递减

定时器在 0 – CCR0 – 0 之间计数



CCR – 计数比较寄存器

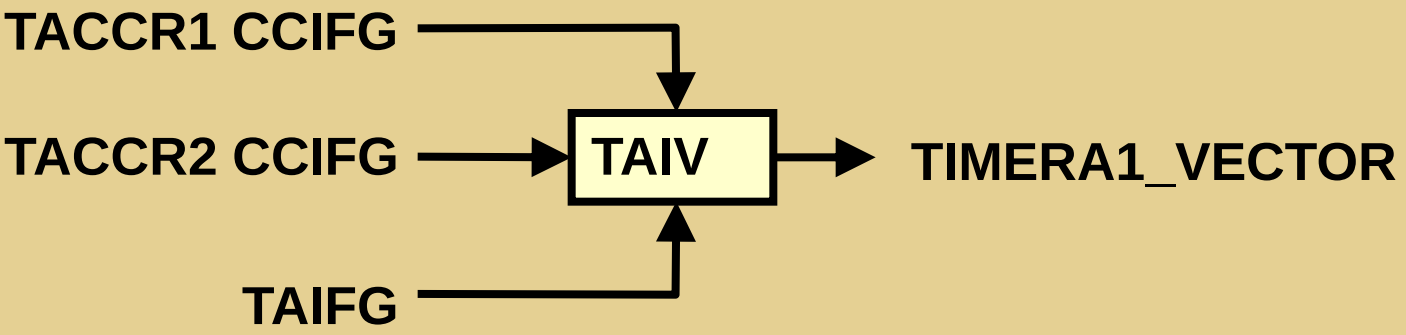
Timer_A 中断

Timer_A 捕获/比较寄存器 0 中断标记 (TACCR0) 生成单个中断向量：

TACCR0 CCIFG → **TIMERA0_VECTOR**

无需处理程序

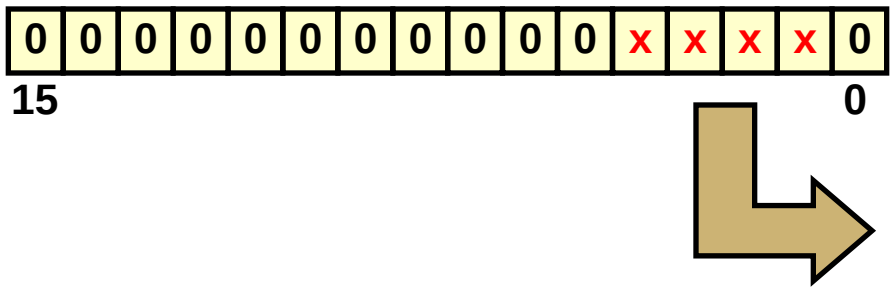
对 TACCR1、2 和 TA 中断标记进行优先级处理，并采用 Timer_A 中断向量寄存器 (TAIV) 将之组合成为另一个中断向量。



代码必须包含一个处理程序，以确定触发的是哪一个 Timer_A1 中断

TAIV 处理程序范例

TAIV



源	TAIV 内容
没有即将发生的中断	0
TACCR1 CCIFG	02h
TACCR2 CCIFG	04h
保留	06h
保留	08h
TAIFG	0Ah
保留	0Ch
保留	0Eh

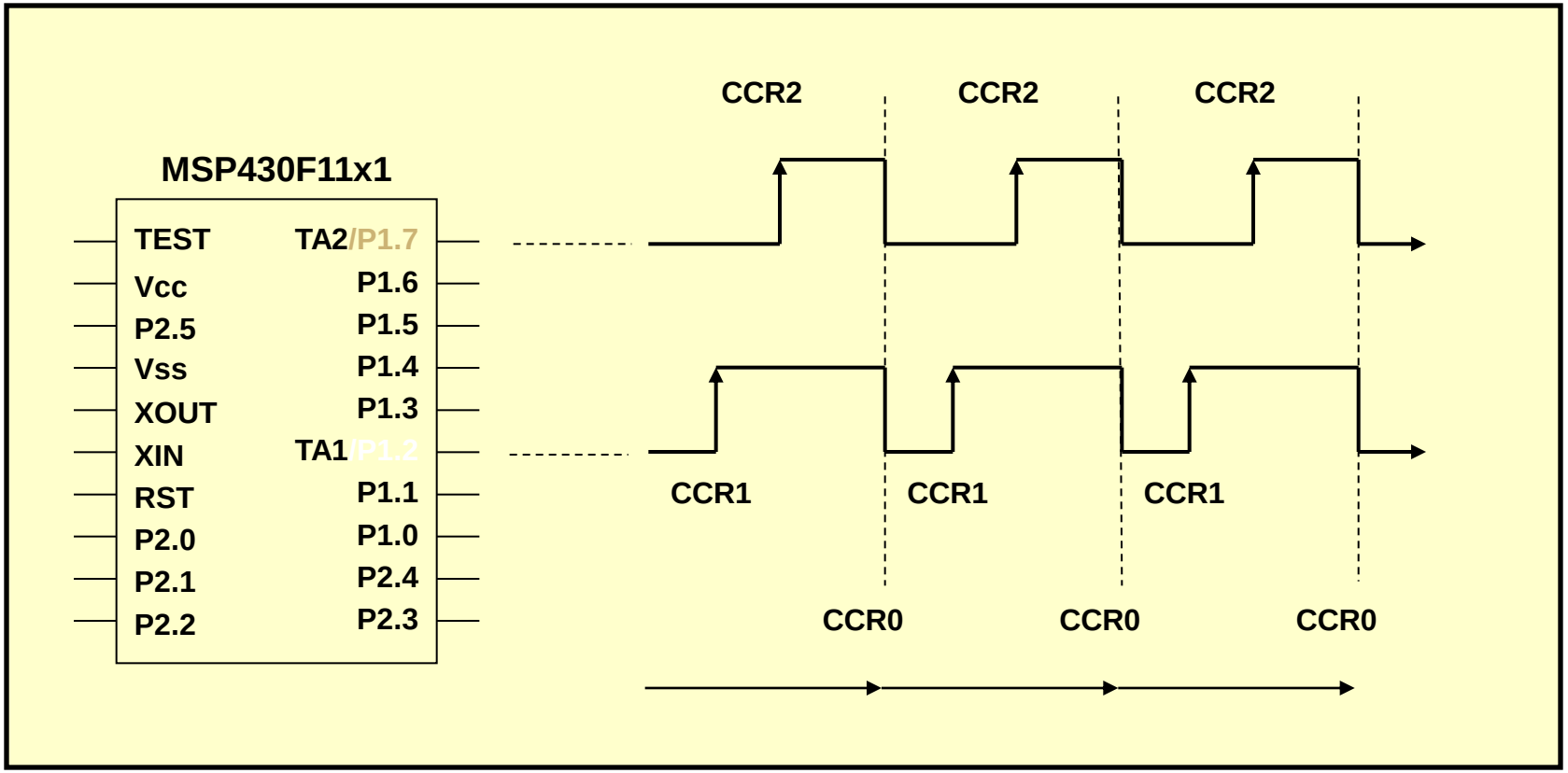
```
#pragma vector = TIMERA1_VECTOR
__interrupt void TIMERA1_ISR(void)
{
  switch(__even_in_range(TAIV,10))
  {
    case  2 :       // TACCR1 CCIFG
      P1OUT ^= 0x04; break;
    case  4 :       // TACCR2 CCIFG
      P1OUT ^= 0x02; break;
    case 10 :       // TAIFG
      P1OUT ^= 0x01; break;
   }
}
```

C 代码

```
0xF814   add.w   &TAIV,PC
0xF818   reti
0xF81A   jmp     0xF824
0xF81C   jmp     0xF82A
0xF81E   reti
0xF820   reti
0xF822   jmp     0xF830
0xF824   xor.b   #0x4,&P1OUT
0xF828   reti
0xF82A   xor.b   #0x2,&P1OUT
0xF82E   reti
0xF830   xor.b   #0x1,&P1OUT
0xF834   reti
```

汇编代码

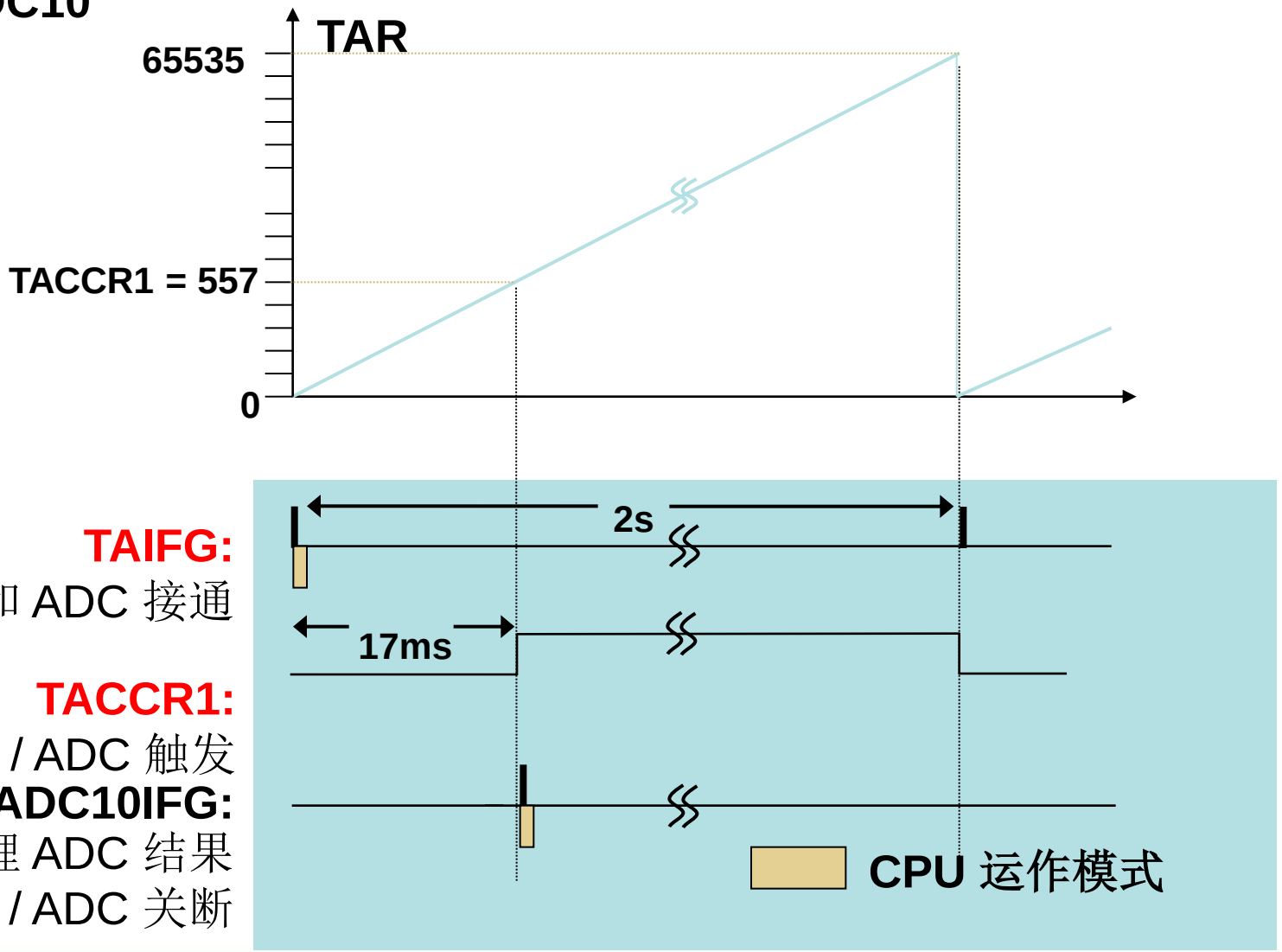
Timer_A PWM 范例



- PWM完全自动发送
- 可通过对CCR的配置，生成多路相同频率和不同占空比的PWM
- 请浏览 MSP430 的相关网址的代码范例

采用 Timer_A 的直接硬件控制

范例：**ADC10**



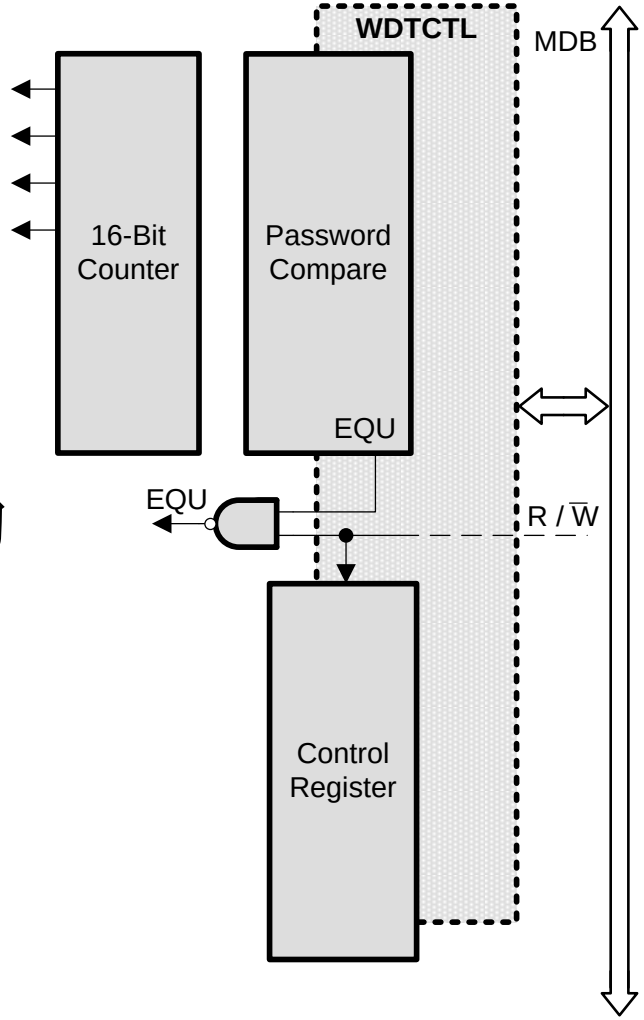
TAIFG:
基准和 ADC 接通

TACCR1:
基准延迟 / ADC 触发

ADC10IFG:
处理 ADC 结果
基准 / ADC 关断

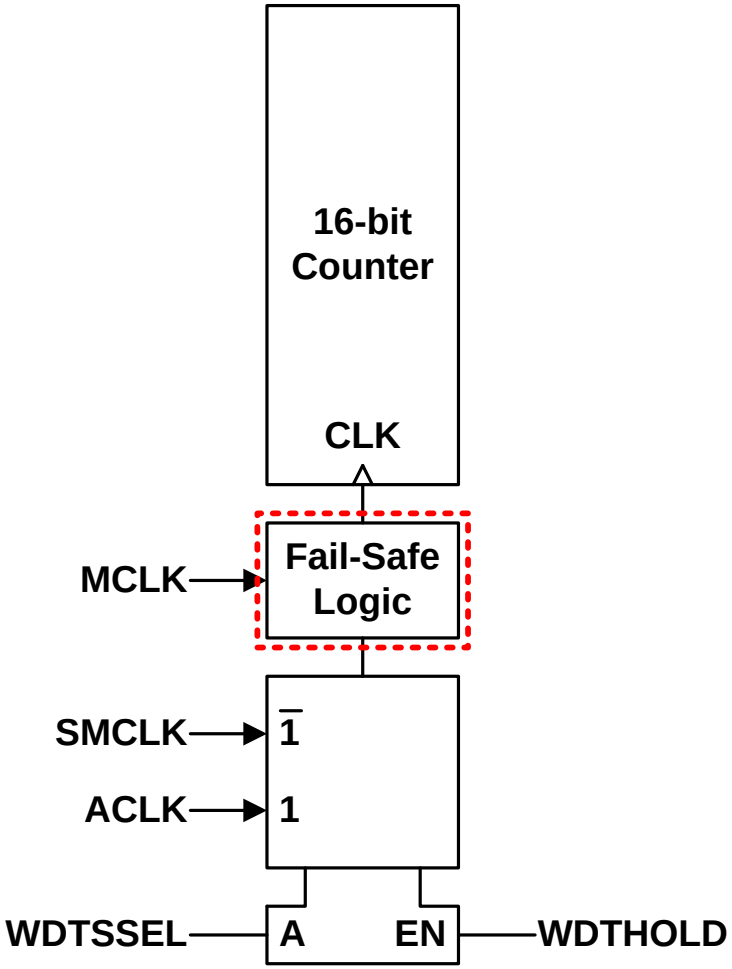
WDT+ 模块：概要

- 在所有 **MSP430** 器件上均有**WDT**
- 两种模式
  - 看门狗
  - 间隔定时器
- 访问受密码保护
- 看门狗模式和定时器模式拥有不同的单独中断向量
- **Clock**可由 **ACLK** 或 **SMCLK** 提供
- 控制 **RST/NMI** 引脚模式
- **WDT+** 添加了故障保护时钟



看门狗定时器故障保险

- 倘若 **ACLK/SMCLK** 发生故障，则时钟脉冲源 **= MCLK**
  （WDT+ 故障保护特性）
- 假如 **MCLK** 由一个晶振 **(XTAL)** 提供，且晶振发生故障，则 **MCLK = DCO**
  （XTAL 故障保护特性）



WDT：常见设计问题

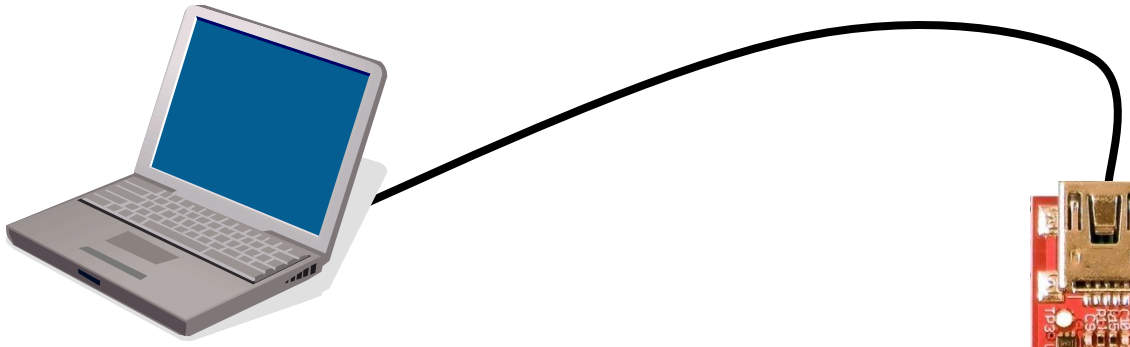
- 程序保持对其自身的复位
- 程序动作反常 – 执行是否到达清除WDT的位置？
  - 设置一个靠近 main() 起点的中断，以查看代码是否重新启动
- CPU 甚至在到达第一条指令之前似乎就出现了冻结现象
  - C 程序是不是具有大量初始化数据 ？
  - 通常只会在拥有超大RAM空间器件上出现
  - 解决方案：在 __low_level_init() 函数中关闭看门狗

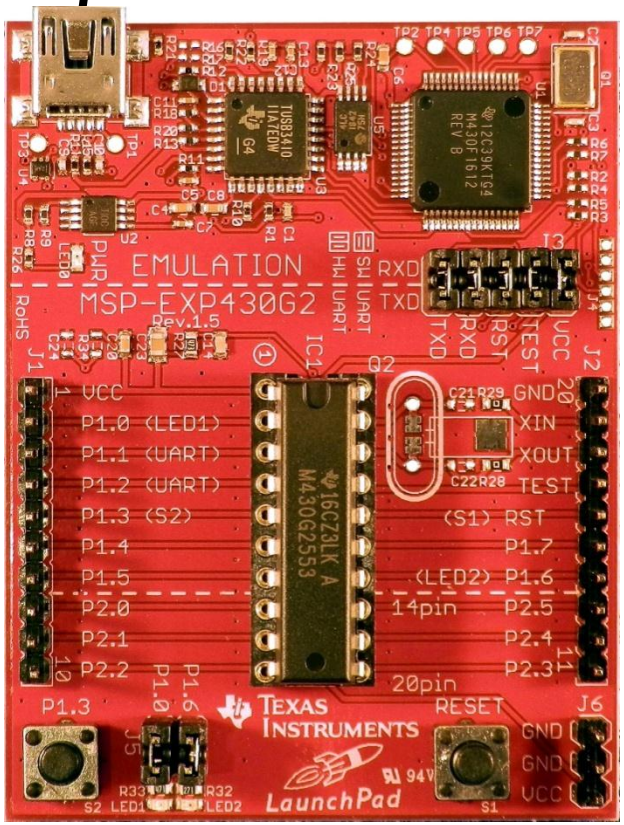
```
void main(void)
{
  WDTCTL = WDTPW+WDTHOLD;       // Stop the dog
   .
   .
}
```

WDT：间隔定时器功能

- 当定时到达时没有 **PUC** 产生
- 如果 **WDTIE** 和 **GIE** 在到达间隔时被设定，则生成一个 **WDT** 间隔中断（而不是复位中断）
- 定时间隔可编程选择

Lab4：定时器和中断





Lab4

- 采用 **Timer_A** 再次完成**Lab2**实验
- 配置 **Timer_A** 计数周期：**5100**
- 当 **TAR = 100** 时，产生一个中断触发**LED**控制

Lab 4 源代码

```
// Configure TimerA
TACTL = ____________________; // Source: ACLK, UP mode
CCR0 = 5100;                 //Timer count 5100
CCR1 = 100;                  //Timer count 100
CCTL0 = CCIE;                //CCR0 interrupt enabled
CCTL1 = CCIE;                //CCR1 interrupt enabled
```

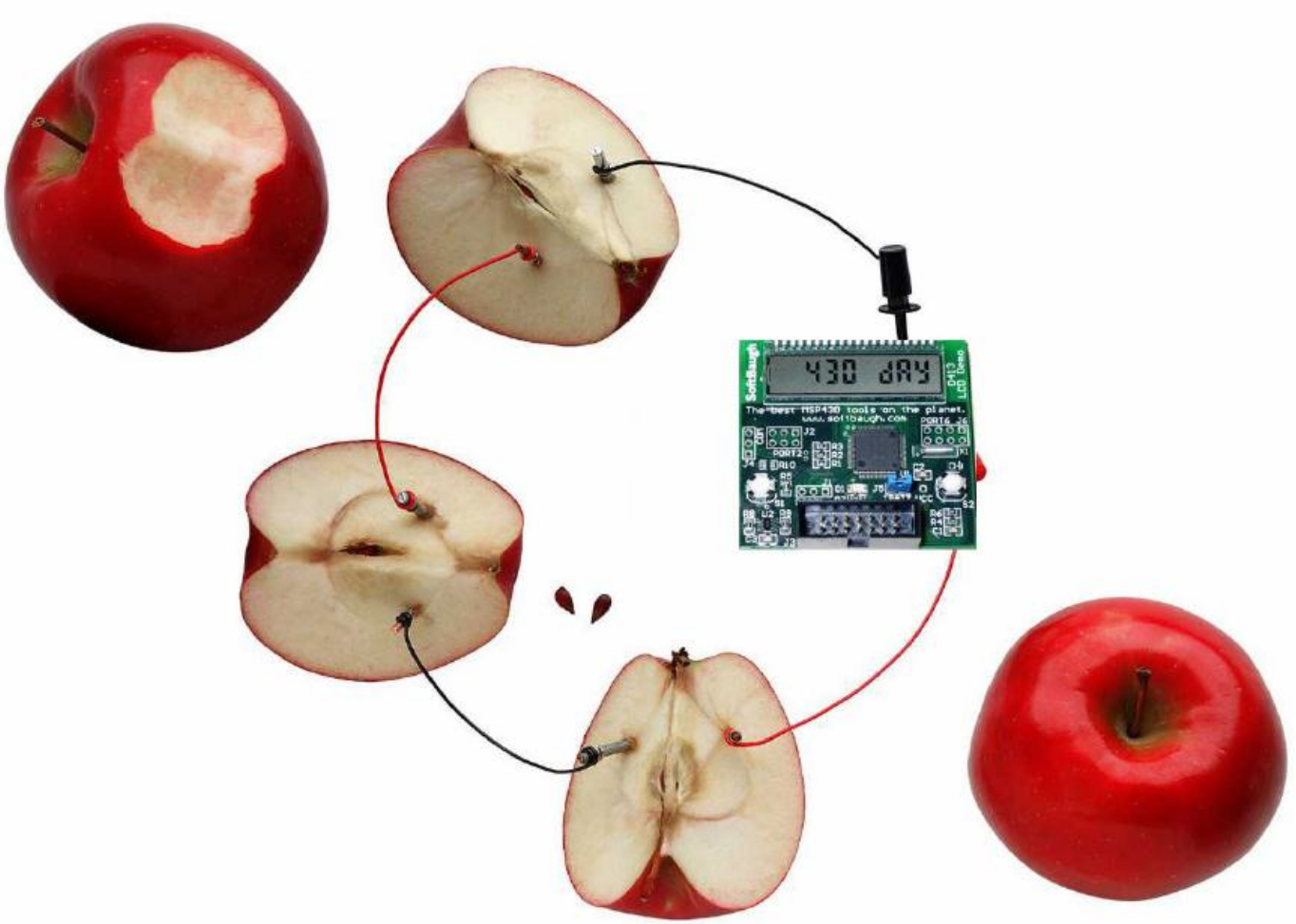
```
 // Timer A0 interrupt service routine
#pragma vector = ___________
__interrupt void Timer_A0(void)
```

```
 // Timer A1 interrupt service routine
#pragma vector = ___________
__interrupt void Timer_A1(void)
```

议程

- 介绍 Value Line 系列
- Code Composer Studio
- CPU 与基本时钟模块
- 中断与 GPIO
- Timer_A 与 WDT+
- MSP430低功耗设计
- ADC10 和 Comparator_A+
- 串行通信模块
- Grace
- 电容式触摸按键解决方案

超低功耗特性



*超低功耗*是MSP430的NDA

- MSP430 从一开始就是专为超低功耗 (ULP) 而设计的
- 外设专为减少功耗和最大限度地降低 CPU 占用率而优化
- 智能型低功耗外设能独立于 CPU 而工作，并让系统在更长的时间里处于较低功耗模式

www.ti.com/ulp

- ✓ 多种操作模式
  - 100 nA 断电（RAM 保持）
  - 0.3 µA 待机
  - 110 µA / MIPS（采用 RAM）
  - 220 µA / MIPS（采用闪存）
- ✓ 即时可稳定工作的高速时钟
- ✓ 1.8 至 3.6V 单电源操作
- ✓ 零功率、始终工作的 BOR
- ✓ <50nA 的引脚漏电流
- ✓ 可最大限度地减少每项任务的执行周期的 CPU
- ✓ 低功耗智能外设
  - 自动传输数据的 ADC
  - 功耗微乎其微的定时器
  - 100 nA 模拟比较器
- ✓ 可保证所需工作条件下的性能