华东师范大学计算机科学技术系

DEP.OF COMPUTER SCIENCEEAST

CHINZ NORMAL UNIVERSITY

MSP430系列超低功耗16位单片机原理与应用

http://www.emlab.net

- 概述
- MSP430单片机结构
- MSP430指令系统与程序设计
- MSP430单片机片内外围模块
- MSP430单片机应用

概述

- 单片微型计算机
  - 单片机的概念
  - 单片机的特点
  - 单片机的应用
- MSP430系列单片机
  - MSP430系列单片机的特点
  - MSP430系列单片机的发展与应用
- MSP430应用选型
  - MSP430系列单片机命名规则
  - MSP430系列单片机选项
- 思考题与习题

微处理器的发展

- 一是朝着面向数据运算、信息处理等功能的系统机方向发展。系统机以速度快、功能强、存储量大、软件丰富、输入/输出设备齐全为主要特点，采用高级语言编程，适用于数据运算、文字信息处理、人工智能、网络通信等场合。
- 另一方面，在一些应用领域中，如智能化仪器仪表、电讯设备、自动控制设备、汽车乃至家用电器等，要求的运算、控制功能相对并不很复杂，但对体积、成本、功耗等的要求却比较苛刻。为适应这方面的需求，产生了一种将中央处理器、存储器、I/O接口电路以及连接它们的总线都集成在一块芯片上的计算机，即所谓的单片微型计算机，简称单片机（Single Chip Microcomputer）。单片机在设计上主要突出了控制功能，调整了接口配置，在单一芯片上制成了结构完整的计算机，因此，单片机也称为微控制器（MCU）

单片机的特点

- 小巧灵活、成本低、易于产品化，它能方便地组装成各种智能式控制设备以及各种智能仪器仪表。
- 面向控制，能针对性地解决从简单到复杂的各类控制任务，因而能获得最佳性能价格比。
- 抗干扰能力强，适应温度范围宽，在各种恶劣环境下都能可靠地工作，这是其他机型无法比拟的。
- 可以很方便地实现多机和分布式控制。使整个系统的效率和可靠性大为提高。

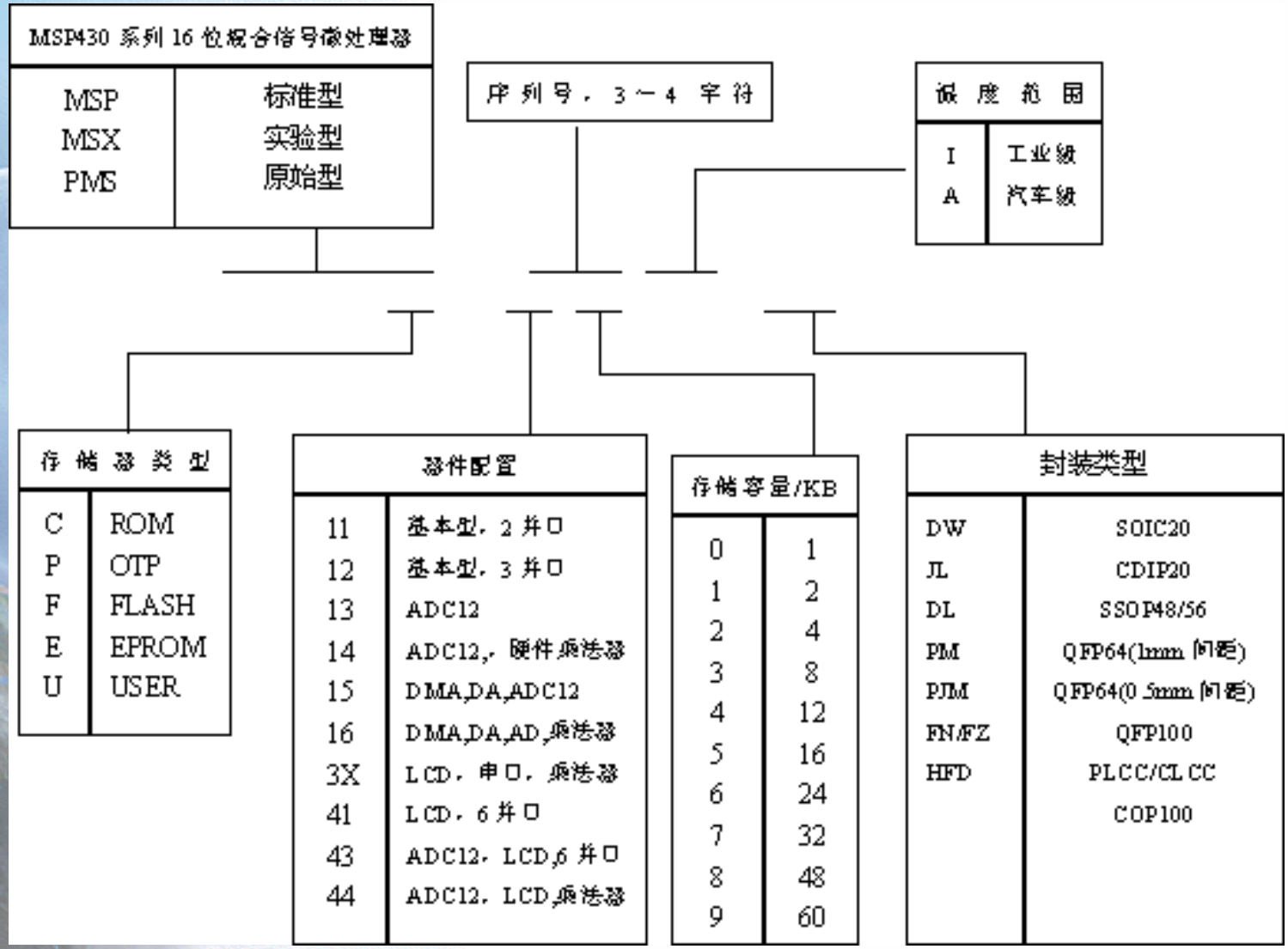
单片机的应用

- 工业控制：单片机的结构特点决定了它特别适用于各种控制系统。它既可以作单机控制器，有可作为多级控制的前沿处理机用于控制系统，应用领域相当广泛。例如：用于各种机床控制、电机控制、工业机器人、各种生产线、各种过程控制、各种检测系统等。在军事工业中：导弹控制、鱼类制导控制、智能武器装置、航天导航系统等。在汽车工业中：点火控制、变速器控制、防滑刹车、排气控制等。
- 智能化的仪器仪表：单片机用于包括温度、湿度、流量、流速、电压、频率、功率、厚度、角度、长度、硬度、元素测定等和各类仪器仪表中，使仪器仪表数字化、智能化、微型化，功能大大提高。
- 日常生活中的电器产品：单片机可用于电子秤、录像机、录音机、彩电、洗衣机、高级电子玩具、冰箱、照相机、家用多功能报警器等。
- 计算机网络与通信方面：单片机可用BIT BUS、CAN、以太网等构成分布式网络的系统，还可以用于调制解调器、各种智能通信设备（例如小型背负式通信机、列车无线通信等）、无线遥控系统等。
- 计算机外部设备：单片机可以用于温氏硬盘驱动器、微型打印机、图形终端、CRT显示器等。

MSP430系列单片机特点

- 超低功耗
- 强大的处理能力
- 高性能模拟技术及丰富的片上外围模块
- 系统工作稳定
- 方便高效的开发环境

MSP430系列单片机命名规则



第一章习题

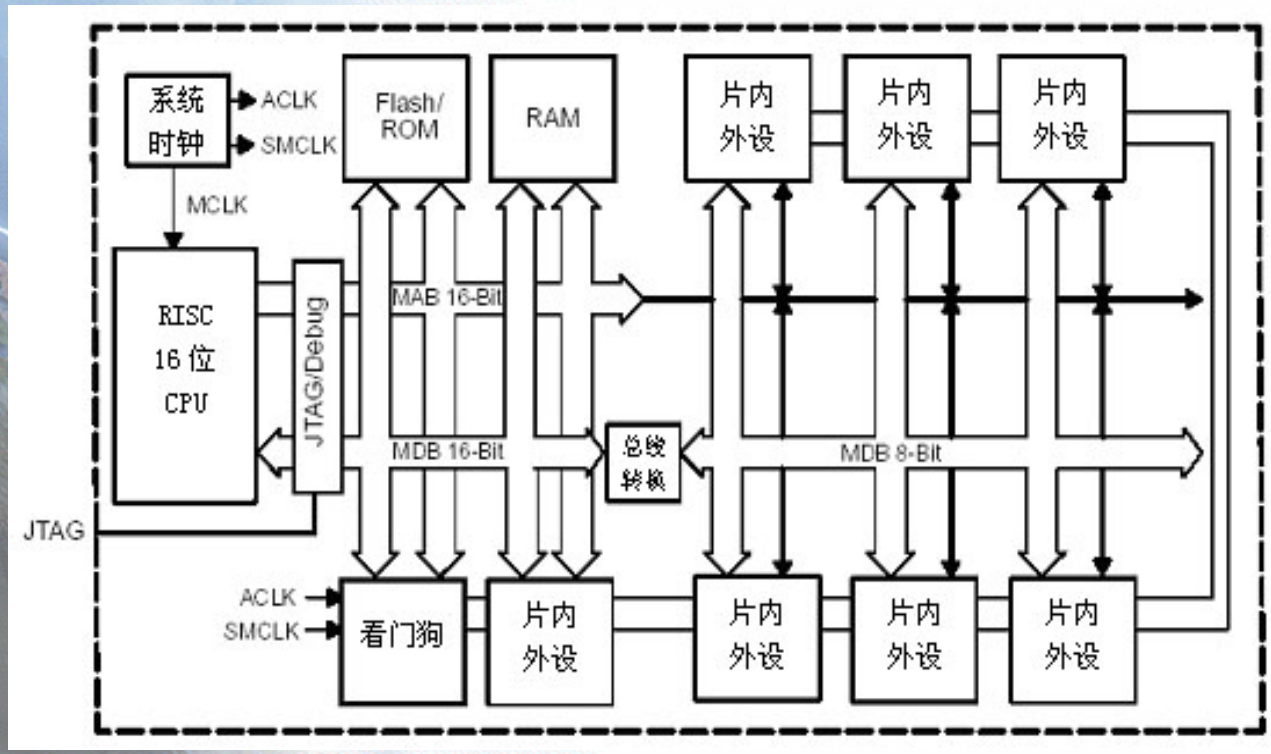
- 微处理器的发展方向是什么？
- 单片机的概念是什么？
- 单片机和我们通常所用的微型计算机有什么区别和联系？
- 单片机常见的领用领域有哪些？
- 如何理解MSP430系列单片机的“单片”解决能力？
- MSP430系列单片机最显著特性是什么？
- 如何理解MSP430系列单片机的低功耗特性？
- 为什么MSP430系列单片机特别适用于电池供电和手持设备？
- 如何理解MSP430系列单片机的强大处理能力？在开发环境方面，MSP430系列单片机和传统单片机相比，有哪些显著优势？
- 构成MSP430系列单片机的各类存储器有什么特点？各自适用于哪些场合？
- MSP430系列单片机应用选型的依据是什么？

MSP430单片机结构

- MSP430单片机结构概述
- MSP430系列产品
  - 无LCD驱动系列产品
  - 有LCD驱动系列产品
- MSP430 CPU结构和特点
- MSP430存储器和地址空间
  - 程序存储器
  - 数据存储器
  - 外围模块寄存器
- 思考题与习题

MSP430结构

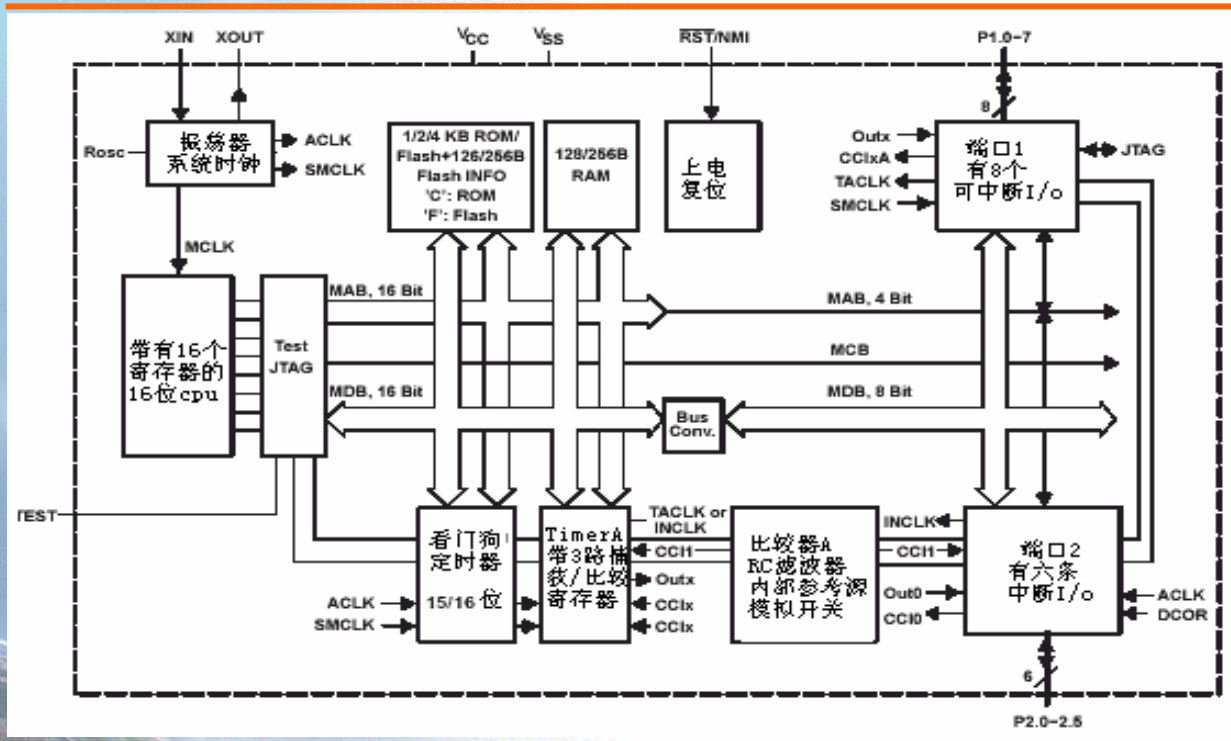
- 16位CPU通过总线连接到存储器和外围模块。
- 直接嵌入仿真处理，具有JTAG接口。
- 能够降低功耗，降低噪声对存储器存取的影响。
- 16位数据宽度，数据处理更为有效。

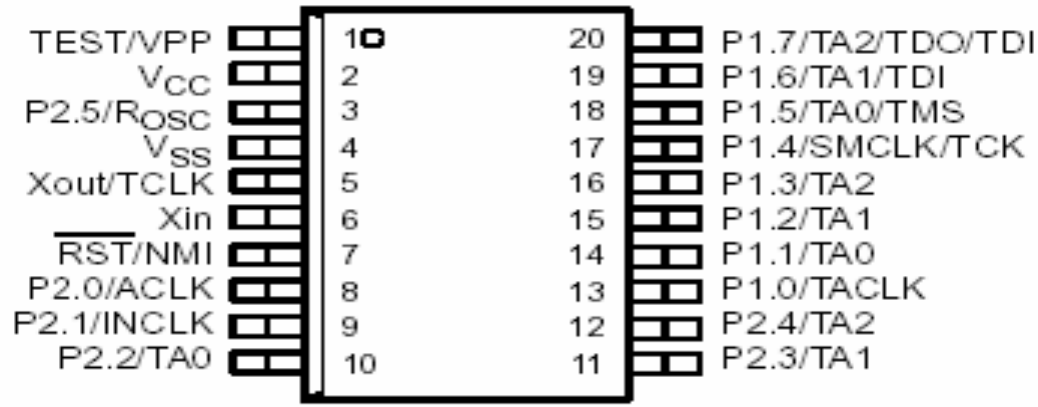


MSP430系列单片机包含以下主要功能部件：

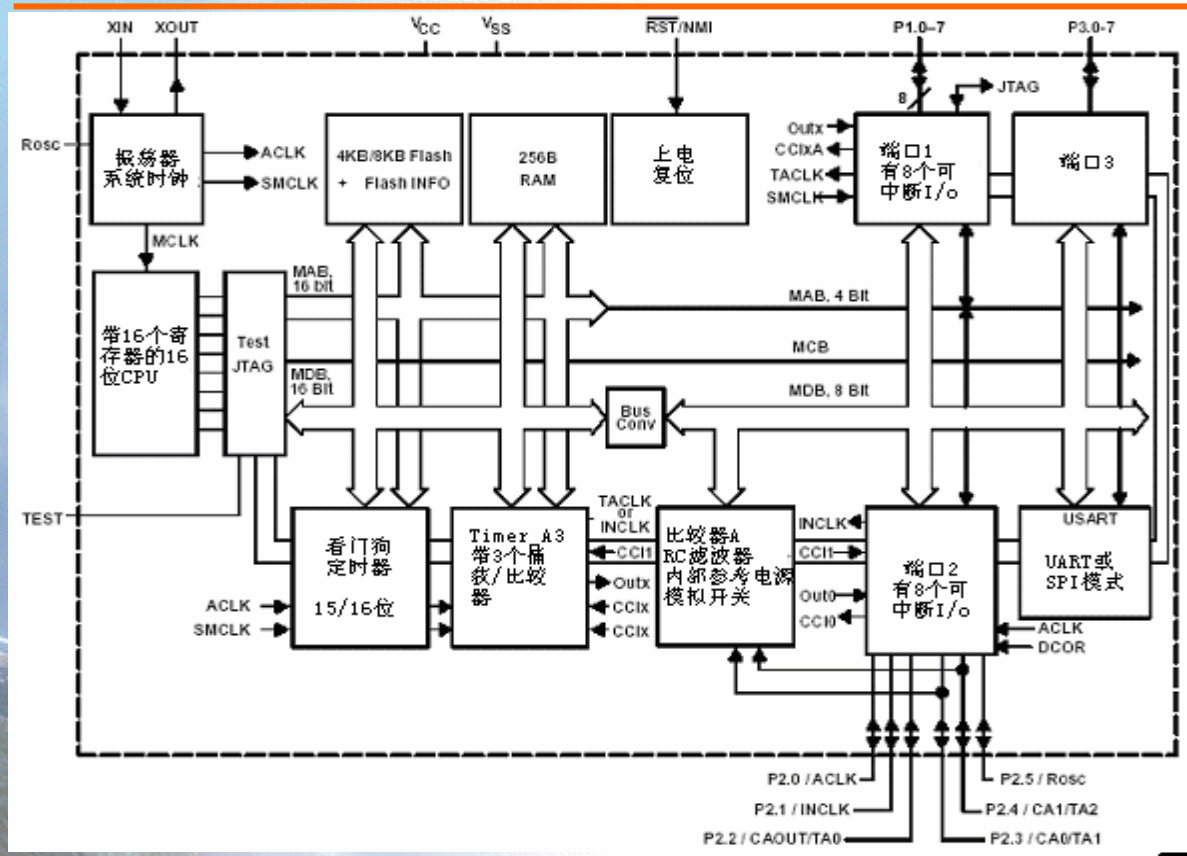
- **CPU**：MSP430系列单片机的CPU和通用微处理器基本相同，只是在设计上采用了面向控制的结构和指令系统。MSP430的内核CPU结构是按照精简指令集和高透明的宗旨而设计的，使用的指令有硬件执行的内核指令和基于现有硬件结构的仿真指令。这样可以提高指令执行速度和效率，增强了MSP430的实时处理能力。
- **存储器**：存储程序、数据以及外围模块的运行控制信息。有程序存储器和数据存储器。对程序存储器访问总是以字形式取得代码，而对数据可以用字或字节方式访问。其中MSP430各系列单片机的程序存储器有ROM、OTP、EPROM和FLASH型。
- **外围模块**：经过MAB、MDB、中断服务及请求线与CPU相连。MSP430不同系列产品所包含外围模块的种类及数目可能不同。它们分别是以下一些外围模块的组合：时钟模块、看门狗、定时器A、定时器B、比较器A、串口0、1、硬件乘法器、液晶驱动器、模数转换、数模转换、端口、基本定时器、DMA控制器等。

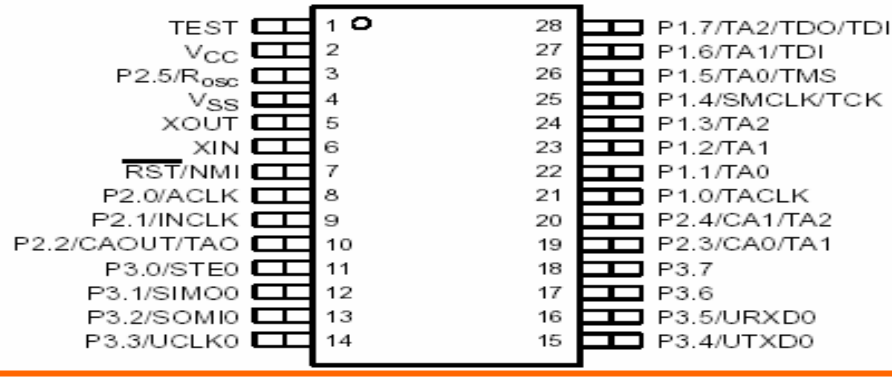
MSP430X11X系列



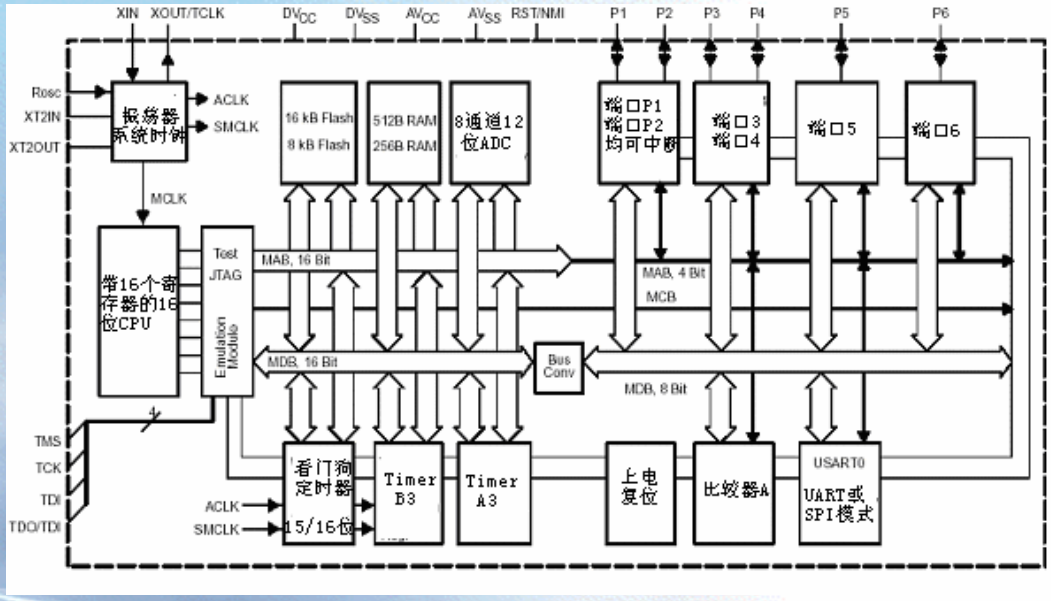


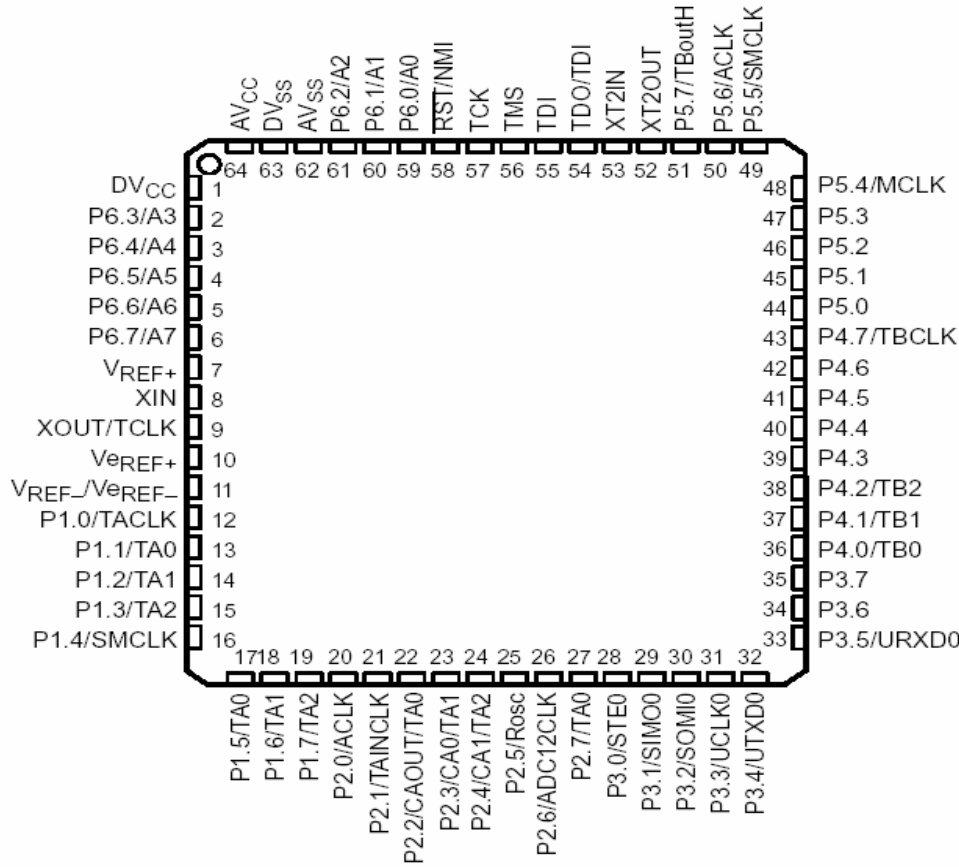
MSP430X12X系列



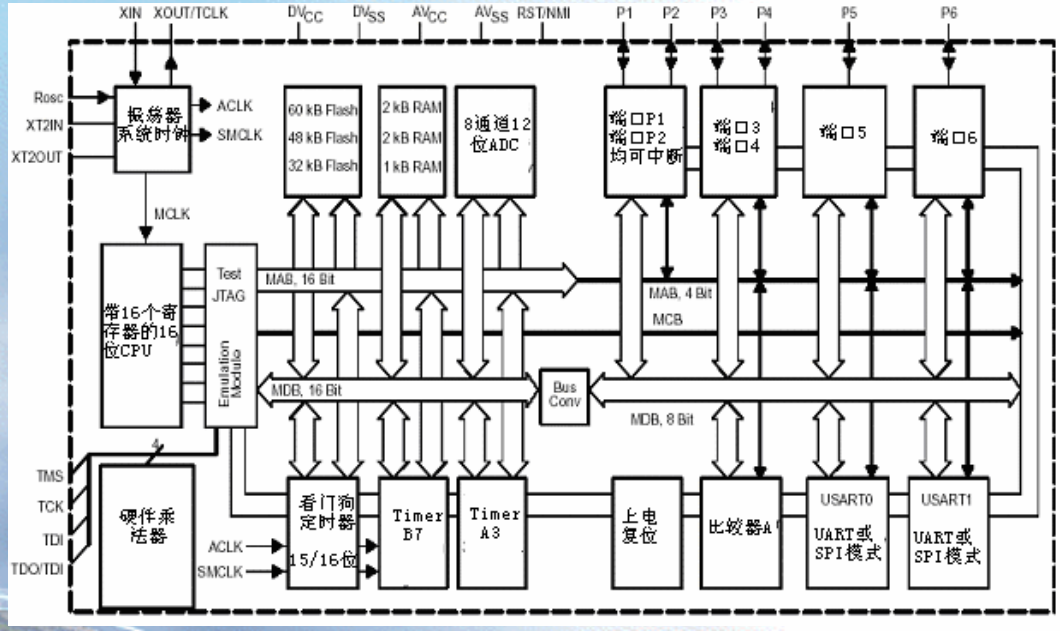


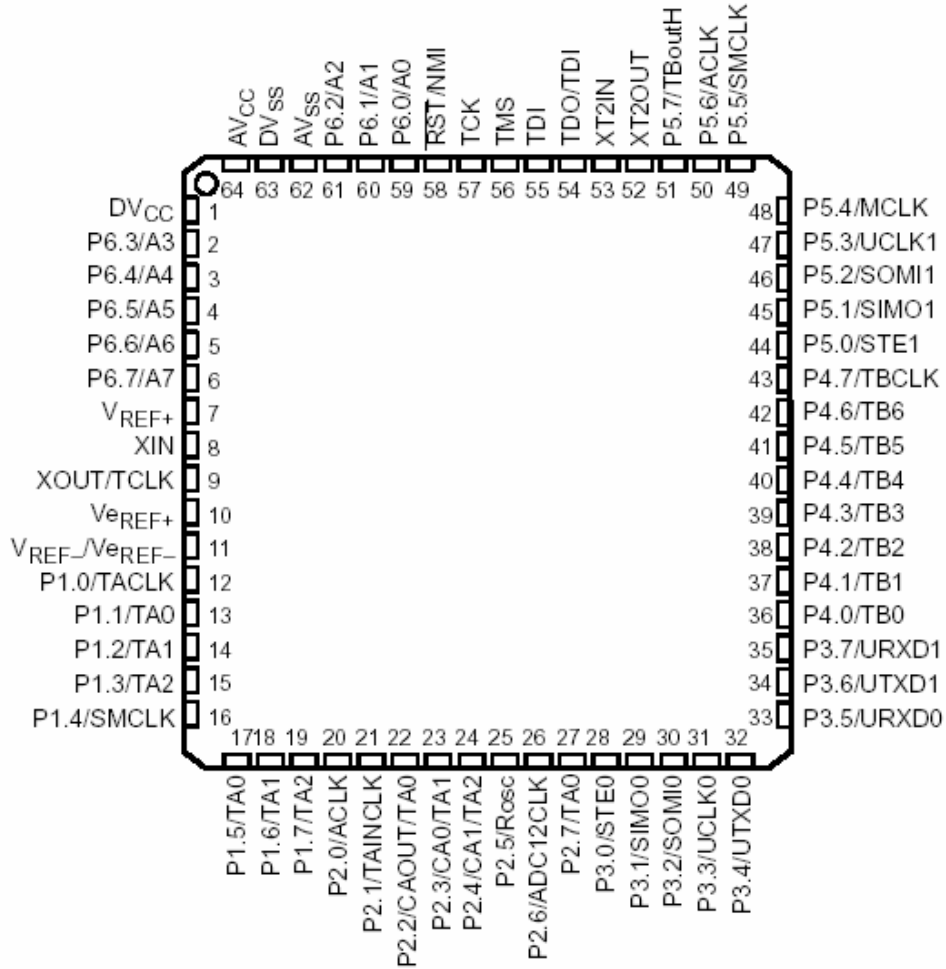
MSP430X13X系列



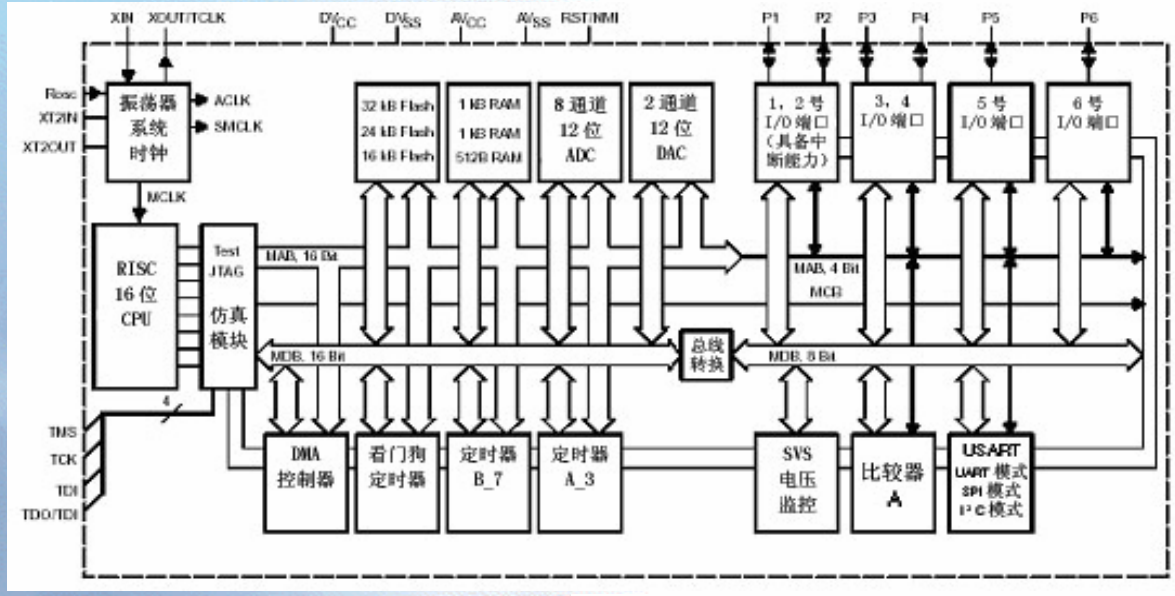


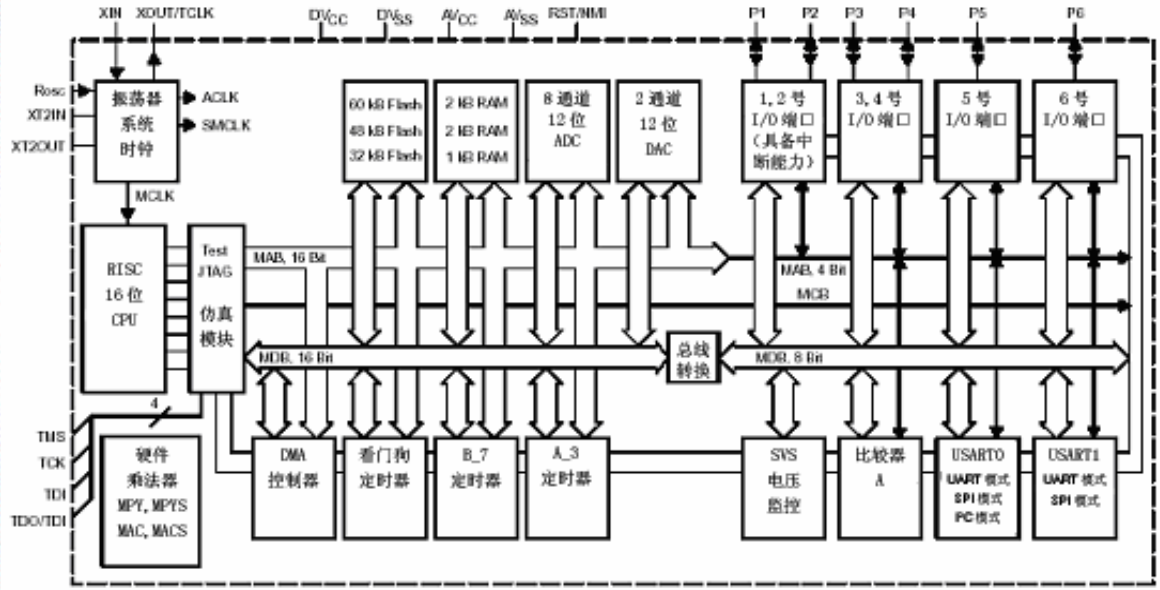
MSP430X14X系列



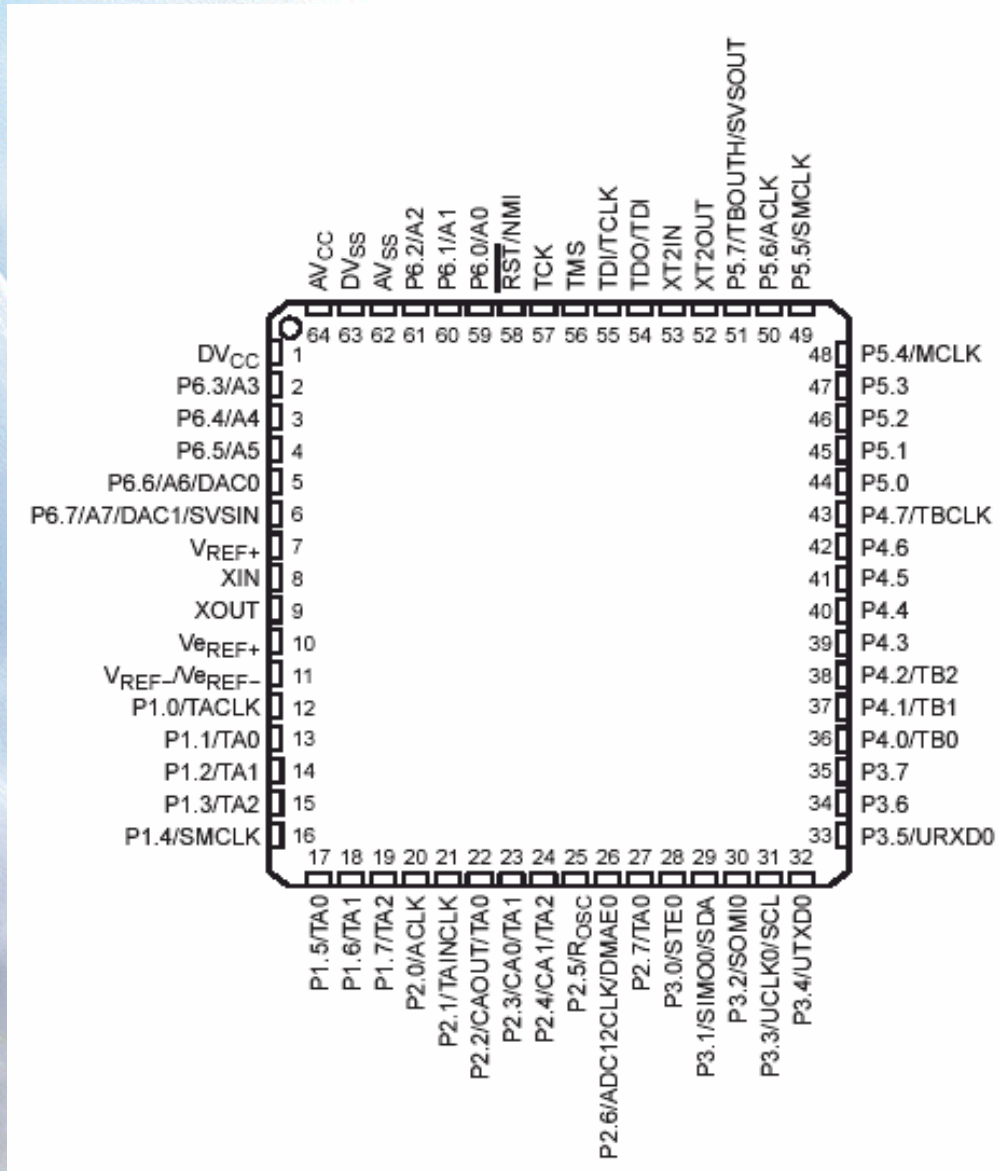


MSP430F15X/F16(1)X 系列

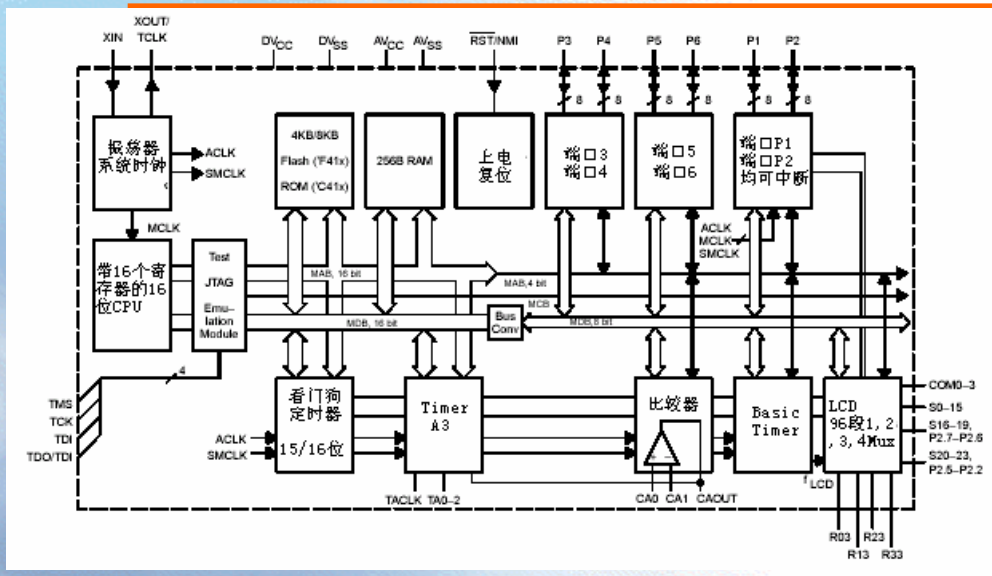


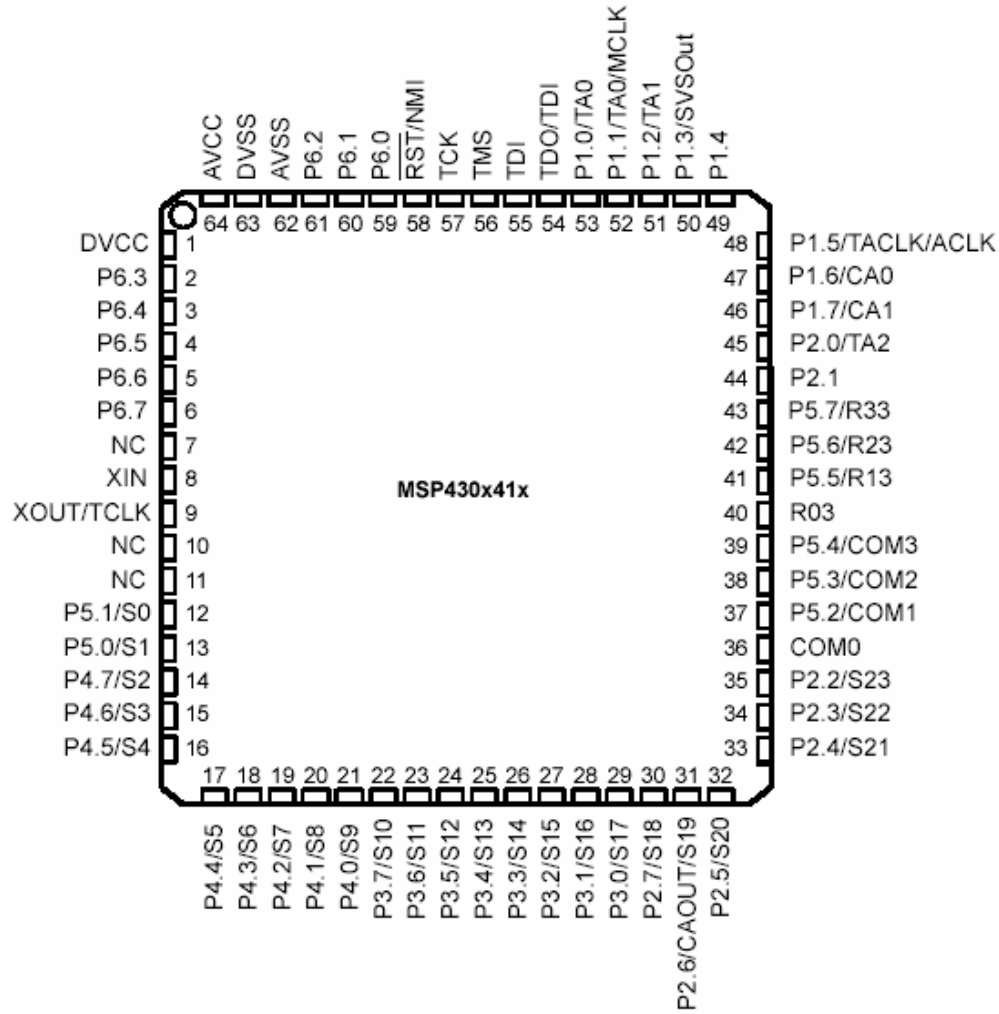


MSP430F15X/F16(1)X 系列

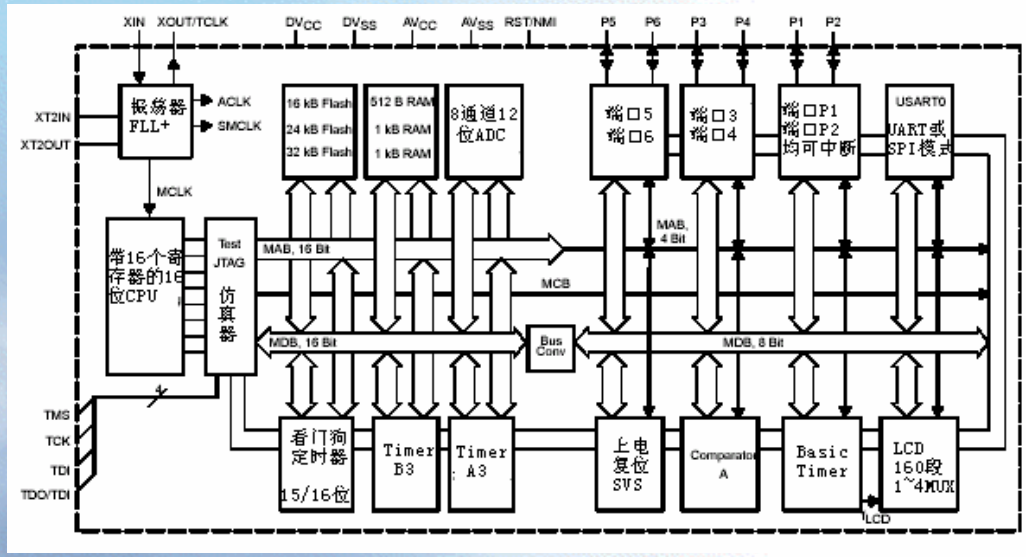


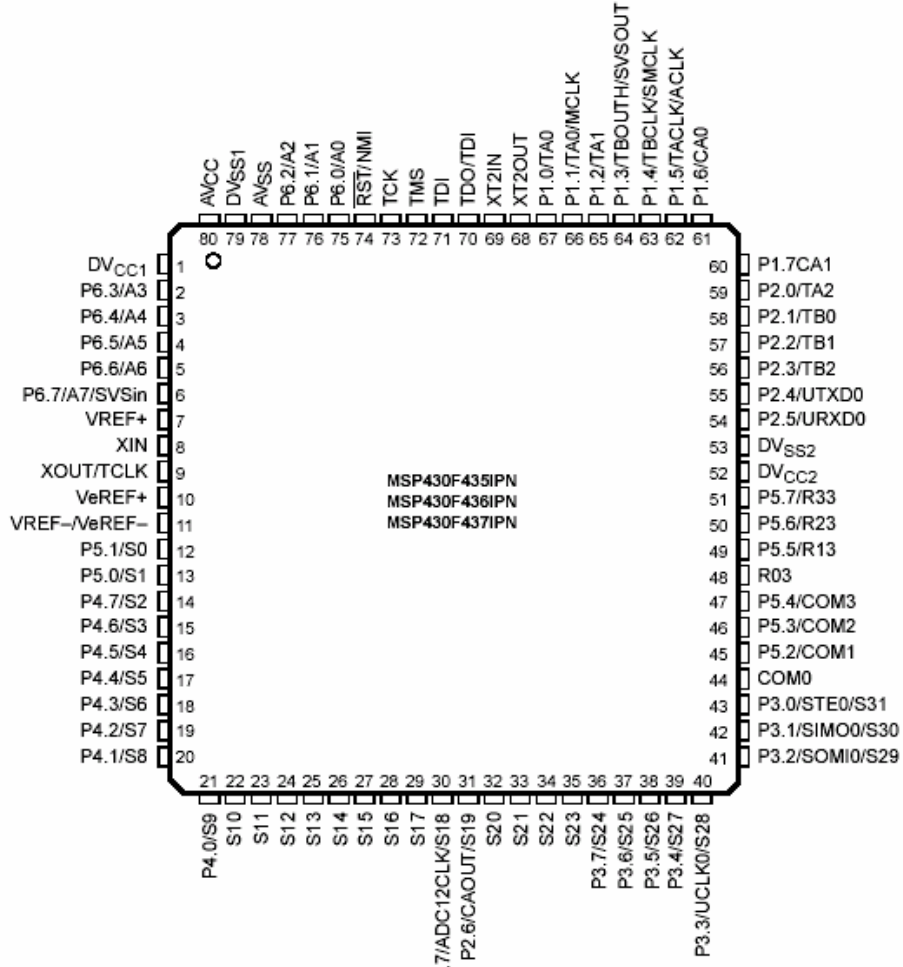
MSP430X41X系列



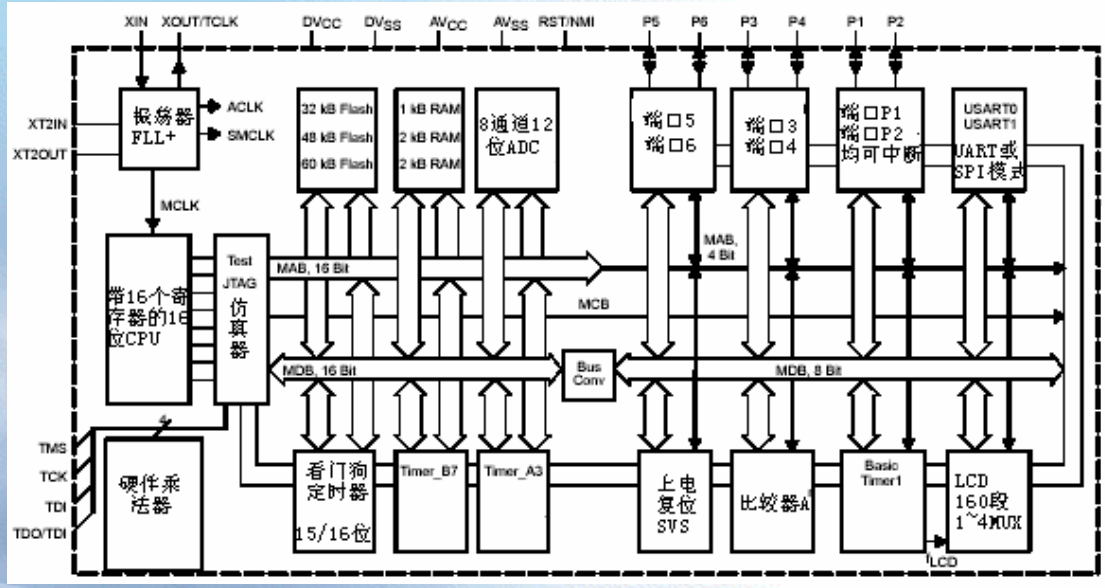


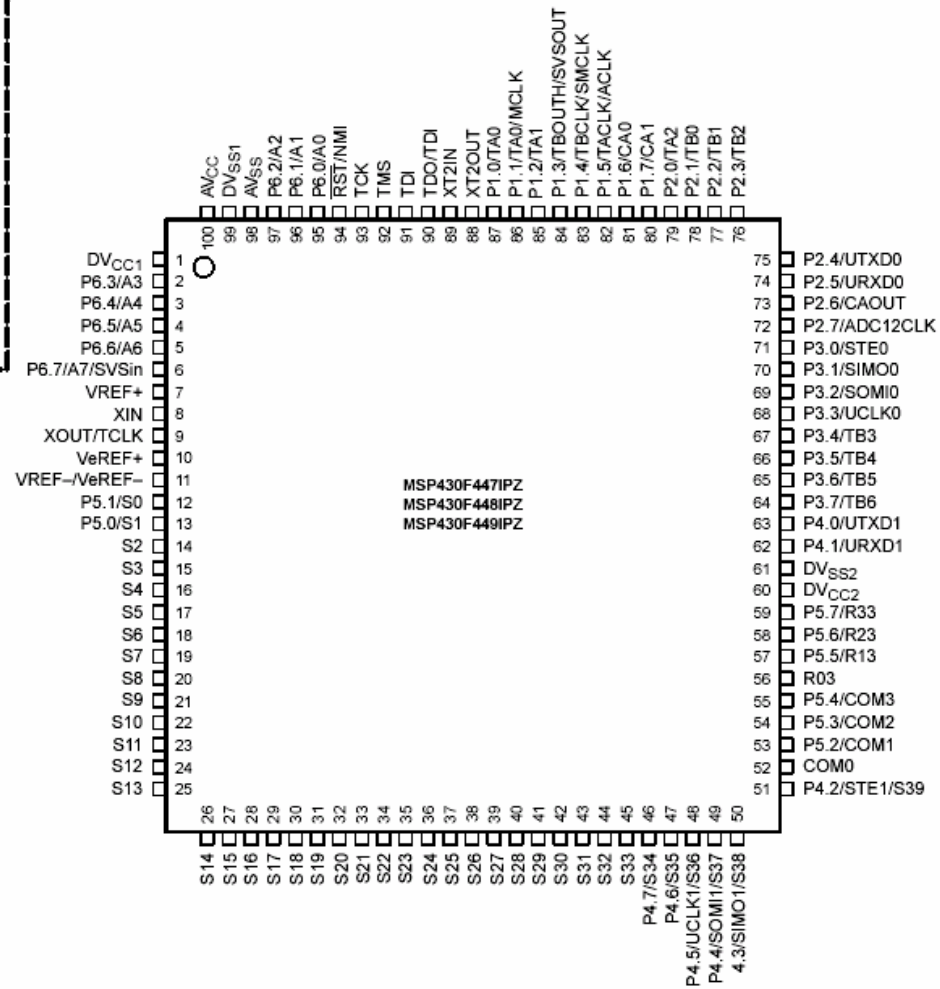
MSP430F43X系列





MSP430F44X系列

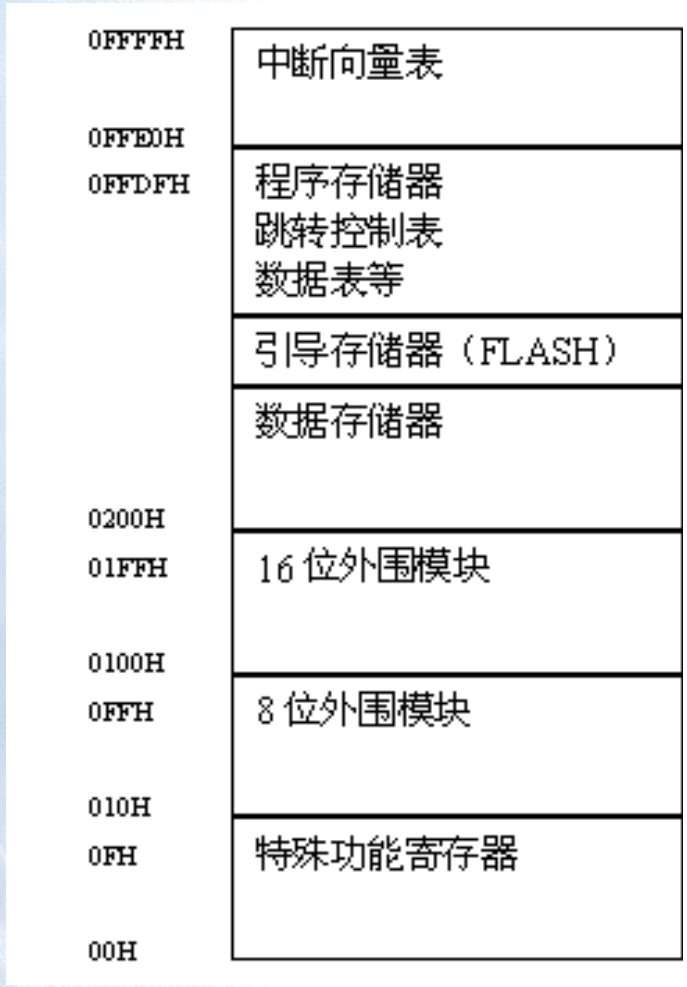




MSP430 CPU的主要特征

- 精简指令集高度正交化
- 寄存器资源丰富
- 寄存器操作为单周期
- 16位地址总线
- 常数发生器
- 直接的存储器到存储器访问

MSP430存储器结构和地址空间



本章小结

- 在结构上MSP430系列单片机集成了一部计算机的各个基本组成部分。虽然其工作原理与普通微机并无差异，但MSP430系列单片机在结构上更加突出了体积小、功能强、面向控制的特点，具有很高的性能价格比。
- MSP430系列单片机由CPU、存储器和外围模块组成，这些部件通过内部地址总线、数据总线和控制总线相连构成单片微机系统。
- MSP430的内核CPU结构是按照精简指令集的宗旨来设计的。具有丰富的寄存器资源、强大的处理控制能力和灵活的操作方式。
- MSP430的存储器结构采用了统一编址方式，可以使得对外围模块寄存器的操作象普通的RAM单元一样方便、灵活。MSP430存储器的信息类型丰富，并具有很强的系统外围模块扩展能力。

思考题与习题

1. MSP430系列FLASH型单片机有什么优势？
2. MSP430X1XX系列单片机的主要特征是什么？
3. MSP430X4XX系列单片机的主要特征是什么？
4. MSP430X1XX和MSP430X4XX系列单片机有什么区别和联系？
5. MSP430F15/16X和其他型号系列单片机相比有哪些特点？
6. MSP430F15X和MSP430F16X有什么区别和联系？
7. 单片机和典型微型计算机在结构上有什么区别？
8. MSP430系列单片机内部包含哪些主要功能部件？
9. MSP430系列单片机的CPU有哪些“面向控制”的特性？
10. MSP430系列单片机的CPU寄存器有什么特点？应该如何正确应用？
11. MSP430系列单片机的直接寻址能力为多少字节？
12. MSP430系列单片机CPU状态寄存器的作用是什么？各位的含义是什么？
13. MSP430系列单片机CPU常数发生器的作用是什么？
14. MSP430系列单片机存储器的组织方式是什么？
15. MSP430系列单片机存储器的组织方式与CPU的RISC结构有什么关系？
16. 为什么说MSP430系列单片机还有很大的系统外围模块扩展能力？
17. MSP430系列单片机具有怎样的中断处理能力？
18. MSP430系列单片机数据存储器的最低地址是什么？程序存储器的最高地址是什么？
19. 程序存储器一般用来存储哪几类信息？各类信息的含义是什么？
20. 数据存储器由那些部分组成？这些部分分别用来存储什么类型的数据？
21. 外围模块寄存器所对应的存储单元在操作上和普通RAM单元有什么区别和联系？
22. MSP430内部数据总线有那些形式？这么安排有什么好处？
23. MSP430系列FLASH型单片机的串行在线可编程的含义是什么？

MSP430指令系统与程序设计

- 指令系统概述
- 寻址方式
- 指令系统介绍
  - 数据传送类指令
  - 数据运算类指令
  - 逻辑操作指令
  - 位操作指令
  - 跳转与程序流程的控制类指令
- 程序设计
  - 程序设计基础
  - 汇编语言程序设计
  - C语言程序设计
- 思考题与习题

MSP430单片机片内外围模块

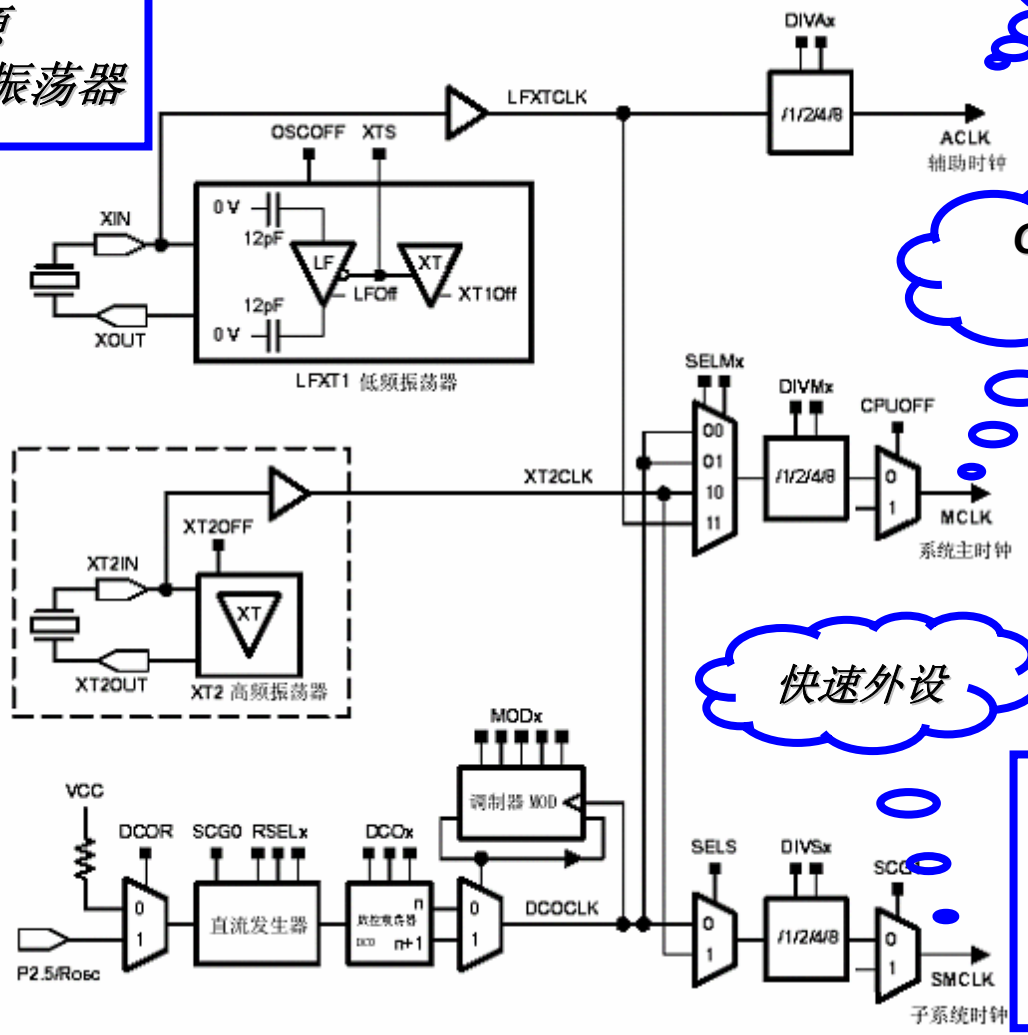
- 时钟模块
- 低功耗结构
- MSP430各种端口
- 定时器
- MSP430液晶驱动模块
- 硬件乘法器
- FLASH存储器模块
- 比较器A
- DMA控制器
- MSP430系列通用串行通信模块的异步模式
- MSP430系列通用串行通信模块的同步模式
- MSP430系列通用串行通信模块的I2C模式
- MSP430模数转换模块
- MSP430数模转换模块
- 思考题与习题

MSP430单片机应用

- MSP430常用接口设计
  - 键盘接口
  - LED显示接口
  - 液晶显示接口
  - 常用LED驱动功率接口
  - 继电器型驱动接口
- MSP430片内外围模块应用
  - 定时器
  - 比较器
  - SPI同步操作
  - A/D D/A 和DMA
- MSP430单片机应用设计举例
  - 自校准变频电源
  - 超低功耗手持式电子斜度计/加速度计
- 思考题与习题

时钟模块

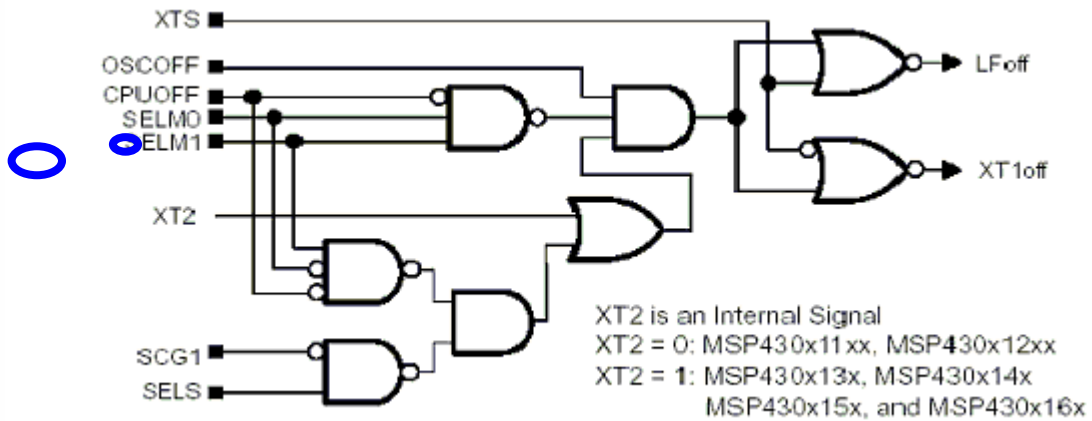
时钟输入源：
LFXT1CLK　　低频时钟源
XT2CLK　　高频时钟源
DCOCL　数字控制RC振荡器

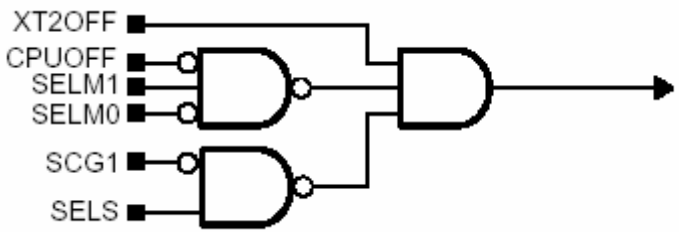


时钟输出信号
ACLK 辅助时钟
MCLK主系统时钟
SMCLK子系统时钟

振荡器控制逻辑

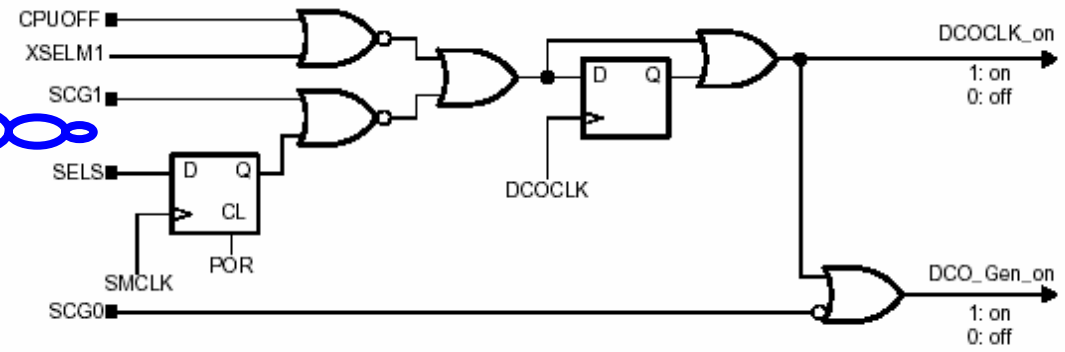
LFXT1
振荡器控制逻辑



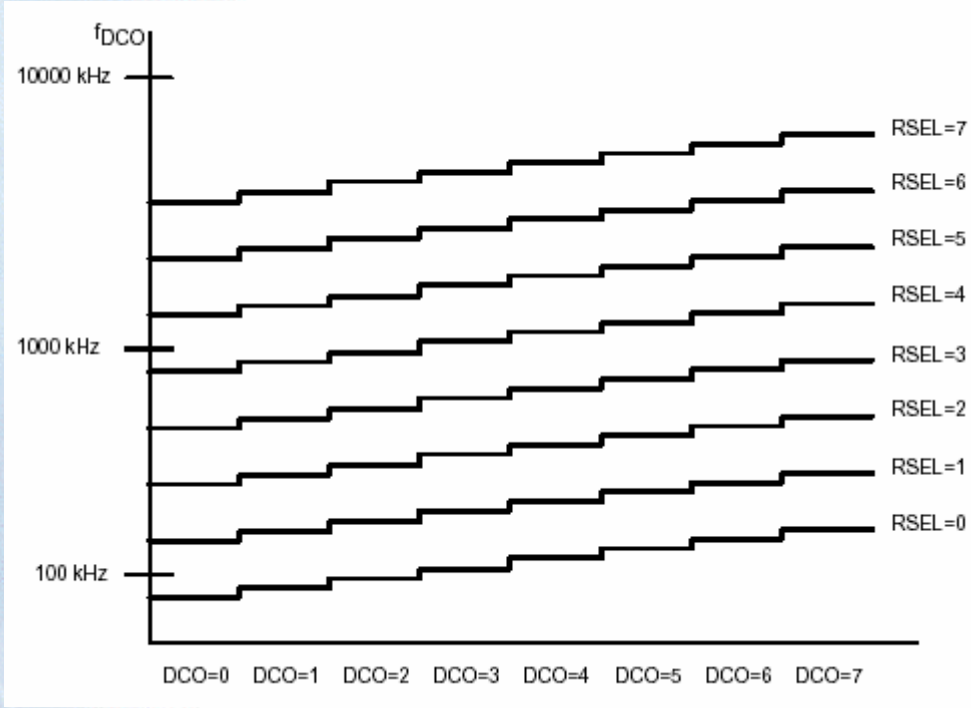


XT2振荡器控制逻辑

DCO振荡器控制逻辑



DCO频率的调节


fDCO
10000 kHz
1000 kHz
100 kHz
RSEL=7
RSEL=6
RSEL=5
RSEL=4
RSEL=3
RSEL=2
RSEL=1
RSEL=0
DCO=0
DCO=1
DCO=2
DCO=3
DCO=4
DCO=5
DCO=6
DCO=7

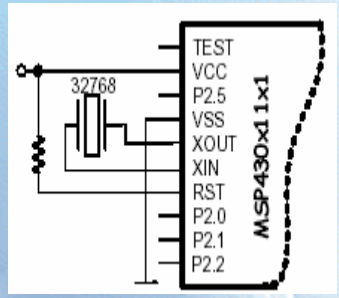
基础时钟模块工作方式和相关寄存器设置

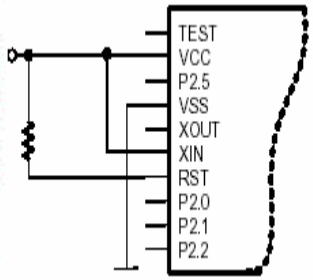
例1设MCLK = XT2, SMCLK = DCOCLK，将MCLK由P5.4输出。（MSP430X14X中引脚P5.4和MCLK复用）。

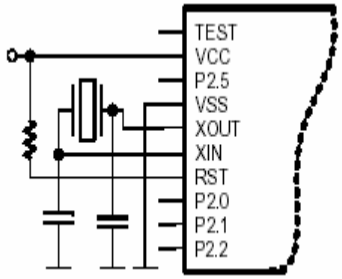
实现上述功能的程序如下：

```
#include <msp430x14x.h>
void main(void)
{
  unsigned int i;
  WDTCTL = WDTPW + WDTHOLD;          // 停止看门狗
 P5DIR |= 0x10;                        // P5.4 输出
P5SEL |= 0x10;                        // P5.4 用作MCLK输出
BCSCTL1 &= ~XT2OFF;                   // XT2有效
do
  {
IFG1 &= ~OFIFG;                  //清除振荡器失效标志
for (i = 0xFF; i > 0; i--);              // 稳定时间
  }
 while ((IFG1 & OFIFG) != 0);          // 如果振荡器失效标志存在
 BCSCTL2 |= SELM1;                  // MCLK = XT2
    for (;;);
 }
```

根据实际连接情况，确定ACLK、SMCLK和MCLK时钟源。







ACLK:
LFTX1 (32768)

MCLK:
DCOCLK或者LFTX1

SMCLK:
DCOCLK或者LFTX1

ACLK:
0

MCLK:
DCOCLK

SMCLK:
DCOCLK

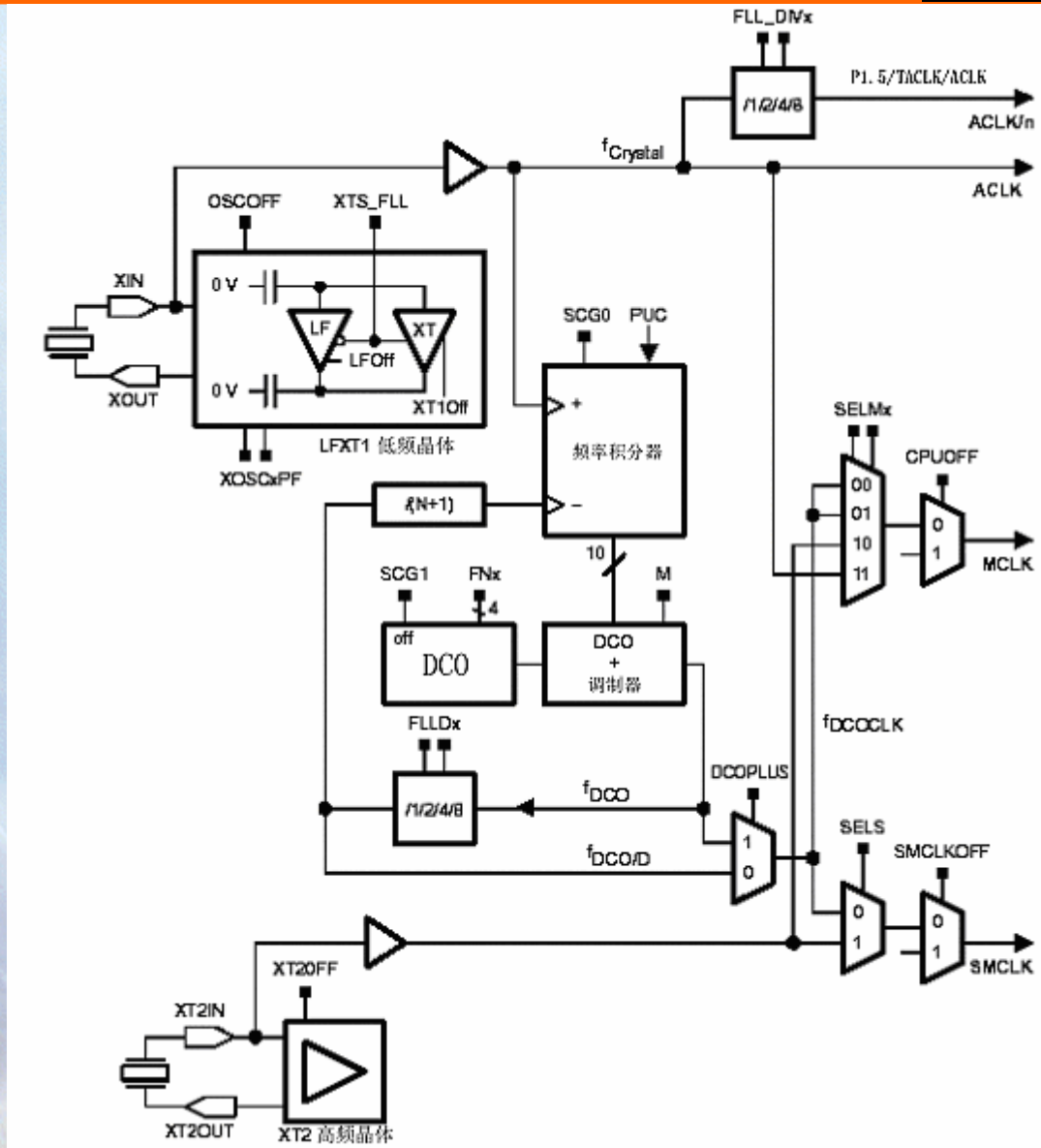
ACLK:
LFTX1 (高频模式)

MCLK:
DCOCLK或者LFTX1(高频模式)

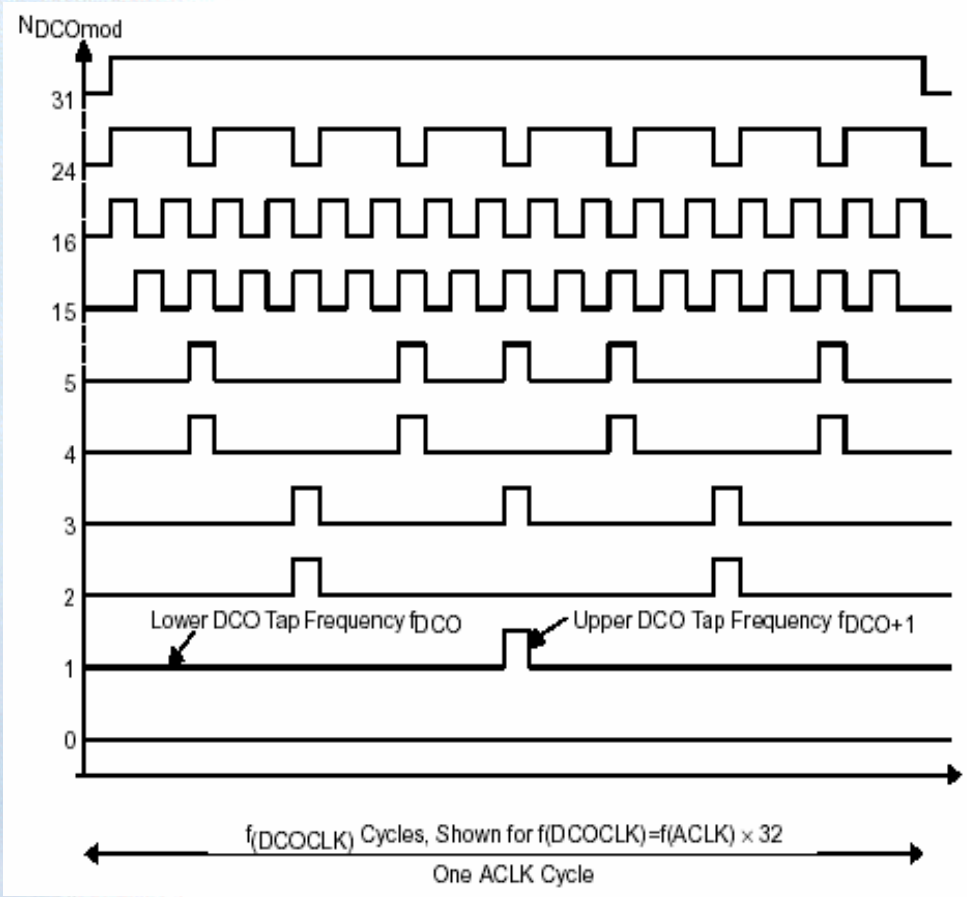
SMCLK:
DCOCLK或者LFTX1(高频模式)

ACLK只能来源于LFXT1。
MSP430X11X1内部只有LFXT1和DCO，没有XT2。
LFXT1只有工作于高频模式才需要外接电容。

MSP430F4XX系列时钟模块



FLL+工作模式

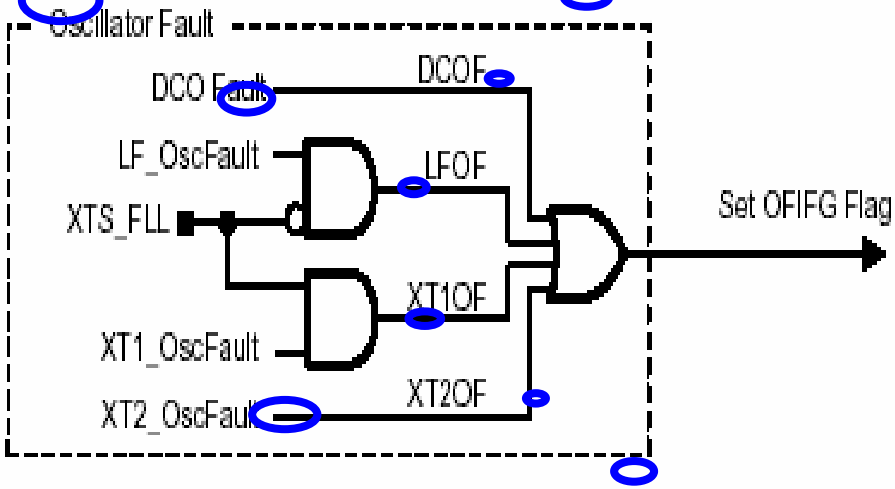


在每32个DCOCLK时钟周期中，调制器通过混合相邻两个DCOCLK周期可以克服周期累加的变化

FLL+失效控制

LFXT1振荡器在低频模式(LF)下失效

DCO振荡器失效



LFXT1振荡器在高频模式(HF)下失效

XT2振荡器失效

调整FLL+

- 保证FLL+锁定位（SCG0在状态寄存器中）并把它置位；关闭反馈环控制
- 把新数值装入调整寄存器SCFQCTL（调整位M，乘数N）
- 将DCO控制位置位，调整器高位置位：SCFI1=OFH，使得芯片以尽可能低的频率工作
- 选择DCO+控制位为1或者0
- 将控制寄存器SCFI0装入新的数值
- 还原或设置FLL+控制位

FLL+模块应用举例

例1 设：ACLK = LFXT1 = 32768Hz，令MCLK = SMCLK = DCOCLK = (n+1) × ACLK，并将MCLK和ACLK分别通过P1.1和P1.5输出。

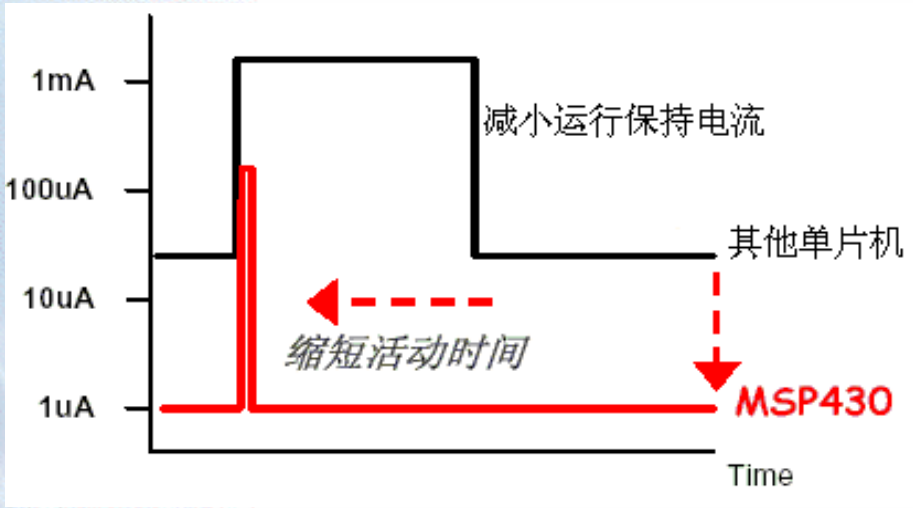
程序代码如下

```
#include "msp430x44x.h"
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;     // 停止看门狗
  SCFI0 |= FN_2;
FLL_CTL0 = XCAP18PF;
SCFQCTL = 74;                   // (74+1) ×32768 = 2.45Mhz
P1DIR = 0x22;                   // P1.1 & P1.5 输出
  P1SEL = 0x22;                 // P1.1 & P1.5输出 MCLK & ACLK
  while(1);
}
```

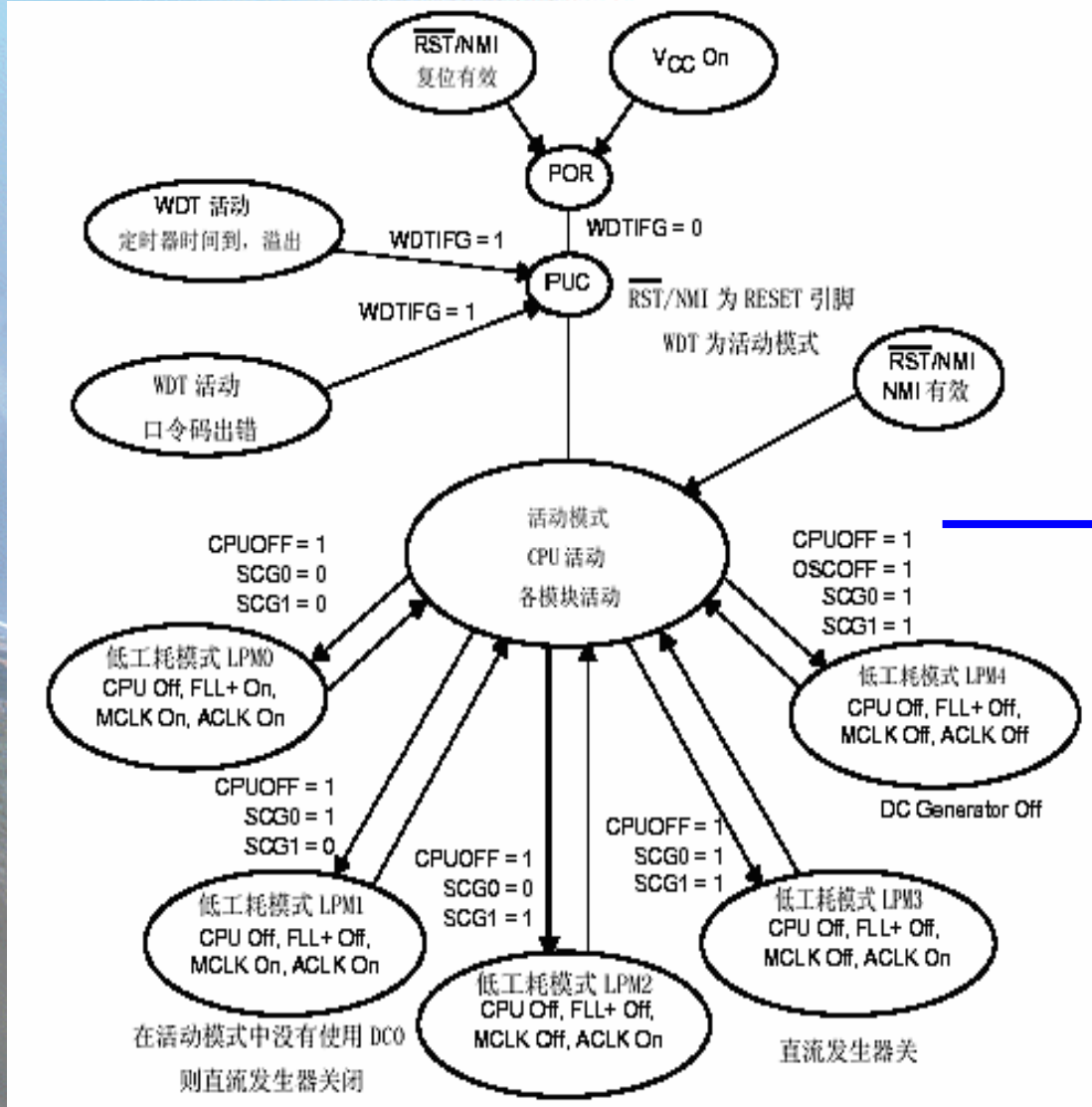
内部DCO ＝2.45Mhz，P1.1--> MCLK = 2.45Mhz，P1.5--> ACLK = 32khz

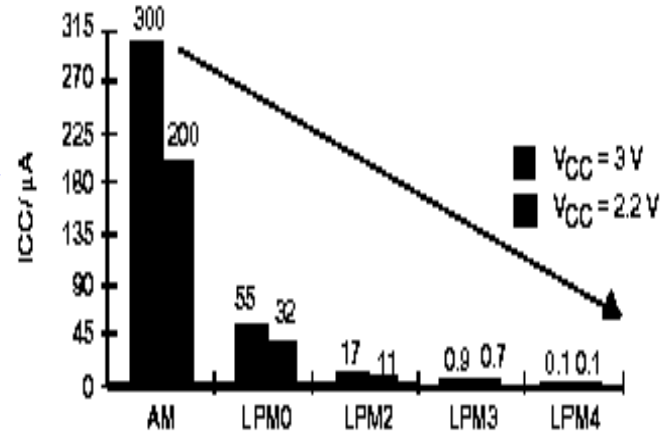
低功耗

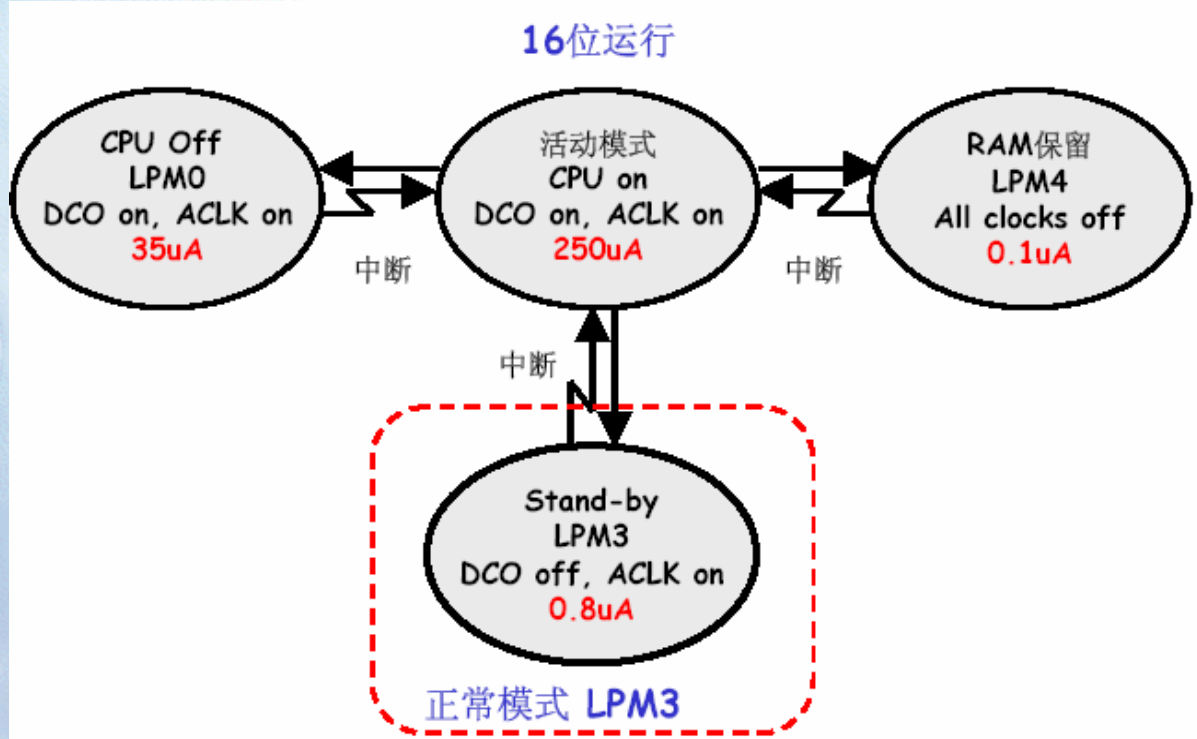
- 使用内部时钟发生器（DCO）无需外接任何元件
- 选择外接晶体或陶瓷谐振器，可以获得最低频率和功耗
- 采用外部时钟信号源
- 瞬间响应特性



MSP430 工作模式状态





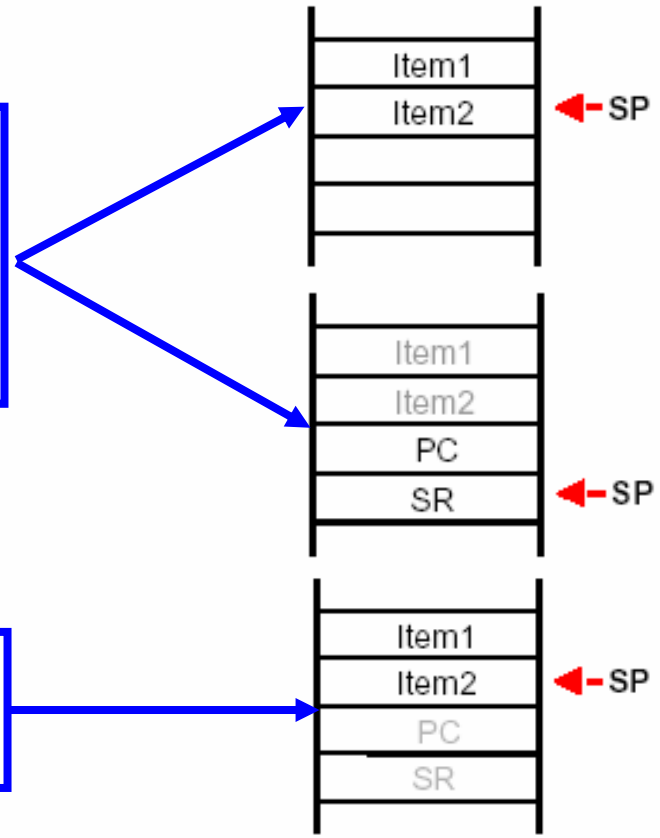


为了充分利用CPU低功耗性能，可以让CPU工作于突发状态。
在通常情况下，根据需要使用软件将CPU设定到某一种低功耗工作模式下，
在需要时使用中断将CPU从休眠状态中唤醒，完成工作之后又可以进入相应的休眠状态。

系统响应中断的过程

- 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）



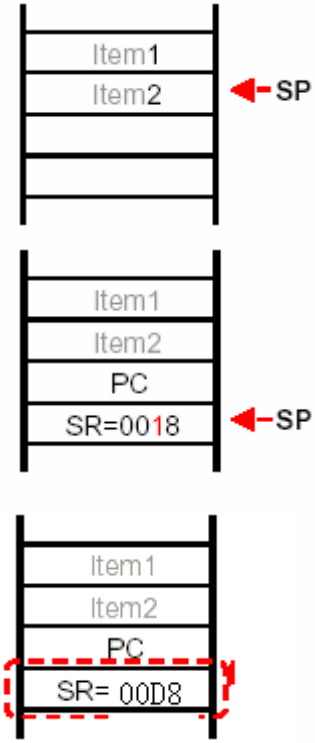
- 执行中断处理子程序
- 执行RETI指令（中断返回）

SR出栈（恢复原来的标志）
PC出栈

低功耗0转变为低功耗3

例：系统初始化完毕之后工作于低功耗模式0，中断事件触发到活动模式，中断处理结束后进入到低功耗模式3。

```
；主程序
.....；初始化操作开始
......;
......；初始化完毕
BIS   #GIE+CPUOFF，SR ；主程序中设置低功耗模式0
;.....；程序在这里停止
;
；中断子程序
......；中断处理开始
......
.....;中断处理结束
BIS   #GIE+CPUOFF+SCG1+SCG0，0（SP）；设置SR为低功耗模式3
RETI；中断返回
；系统进入低功耗模式3。
.....
```



低功耗原则

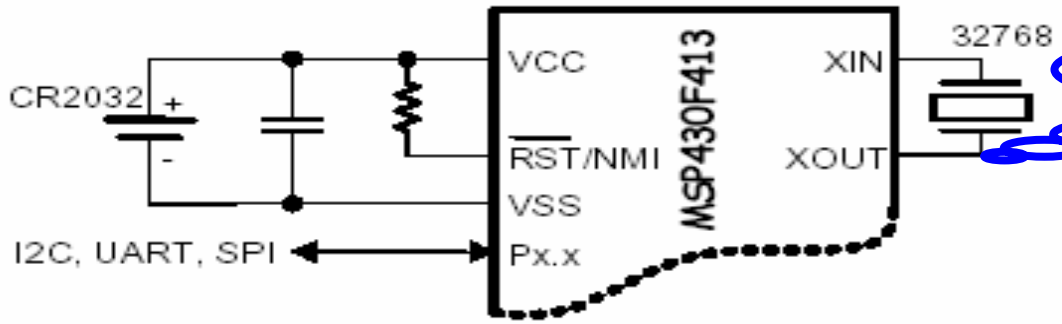
- 一般的低功耗原则：

最大化LPM3的时间，用32KHz晶振作为ACLK时钟，DCO用于CPU激活后的突发短暂运行
用接口模块代替软件驱动功能。
用中断控制程序运行
用可计算的分支代替标志位测试产生的分支
用快速查表代替冗长的软件计算
在冗长的软件计算中使用单周期的CPU寄存器
避免频繁的子程序和函数调用
尽可能直接用电池供电

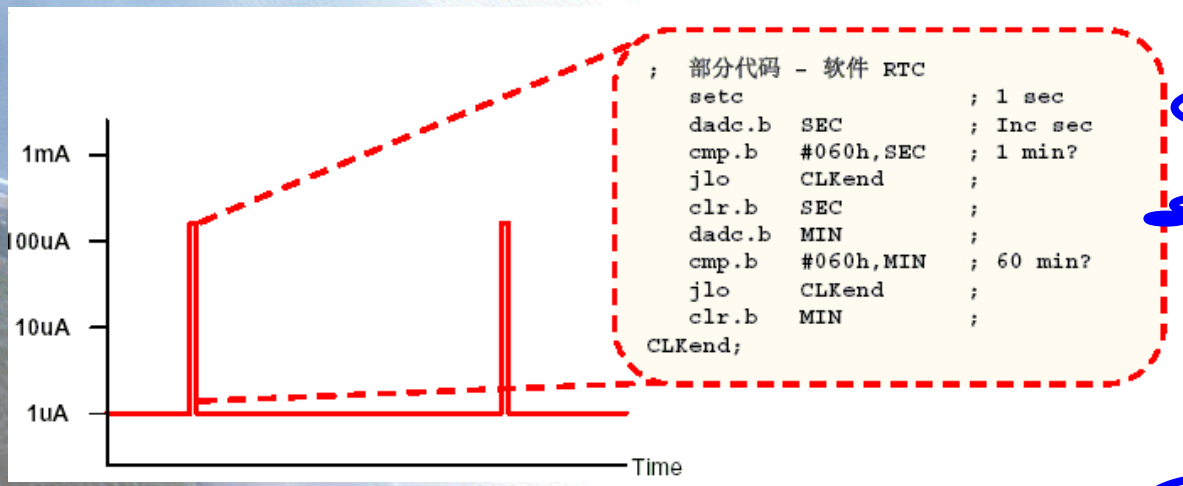
- 设计外设时的常规原则：

将不用的FETI输入端连接到VSS
JTAG端口TMS、TCK和TDI不要连接到VSS
CMOS输入端不能有浮空节点，将所有输入端接适当的电平
不论对于内核还是对于各外围模块，选择尽可能低的运行频率，如果不影响功能应设计自动关机

超低功耗嵌入式实时时钟



MSP430F413通过外接32768Hz晶体




```
;  部分代码 - 软件 RTC
    setc                    ; 1 sec
    dadc.b   SEC            ; Inc sec
    cmp.b    #060h,SEC      ; 1 min?
    jlo      CLKend         ;
    clr.b    SEC            ;
    dadc.b   MIN            ;
    cmp.b    #060h,MIN      ; 60 min?
    jlo      CLKend         ;
    clr.b    MIN            ;
CLKend;
```

MSP430 CPU工作于突发状态

平均电流＝LPM3电流+工作期间电流
＝0.80uA+250uA×100us/1000000us
=0.80 uA +0.030uA
=0.83uA

运行期间电流消耗250uA
在1S时间段内程序运行时间仅为100us
LPM3电流 为0.80uA

MSP430的端口

- 类型丰富

P1，P2，
P3，P4，P5，P6，

S和COM

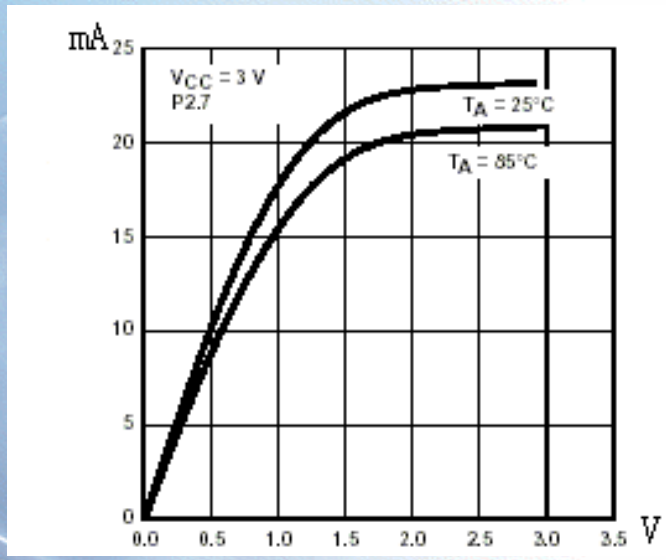
- 功能丰富

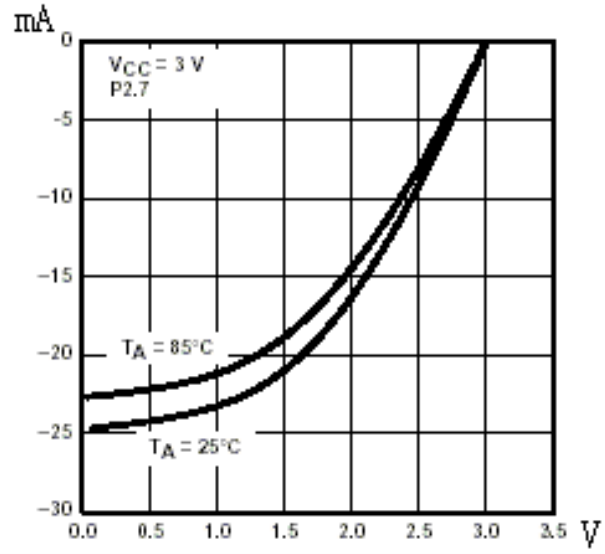
I/O
中断能力
其他片内外设功能
驱动液晶

- 寄存器丰富

P1与P2各有7个寄存器
P3、P4、P5、P6有四个寄存器

端口数据输出特性





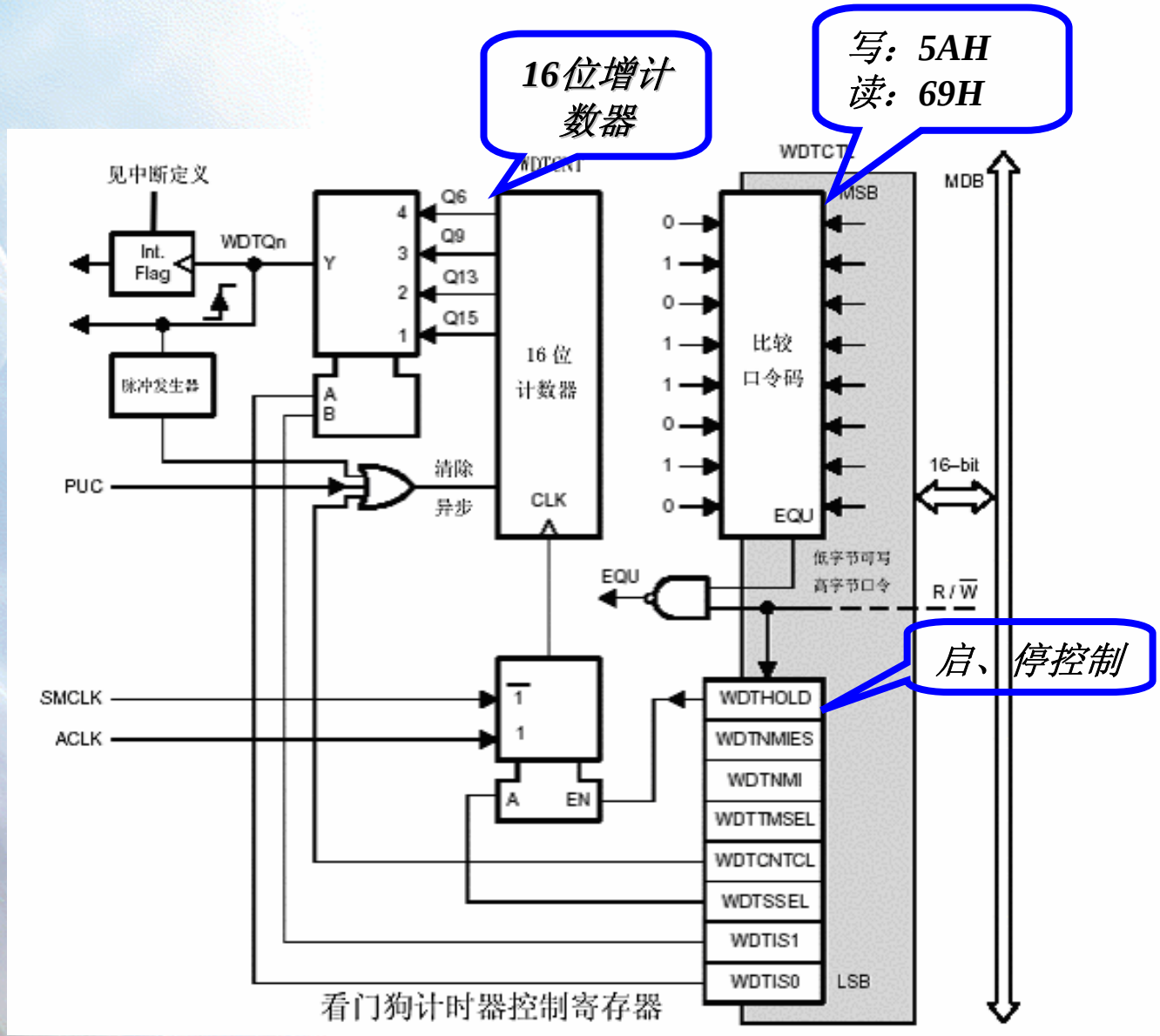
低电平输出特性

高电平输出特性

定时器

- 看门狗定时器
- 基本定时器
- 定时器A
- 定时器B

看门狗定时器

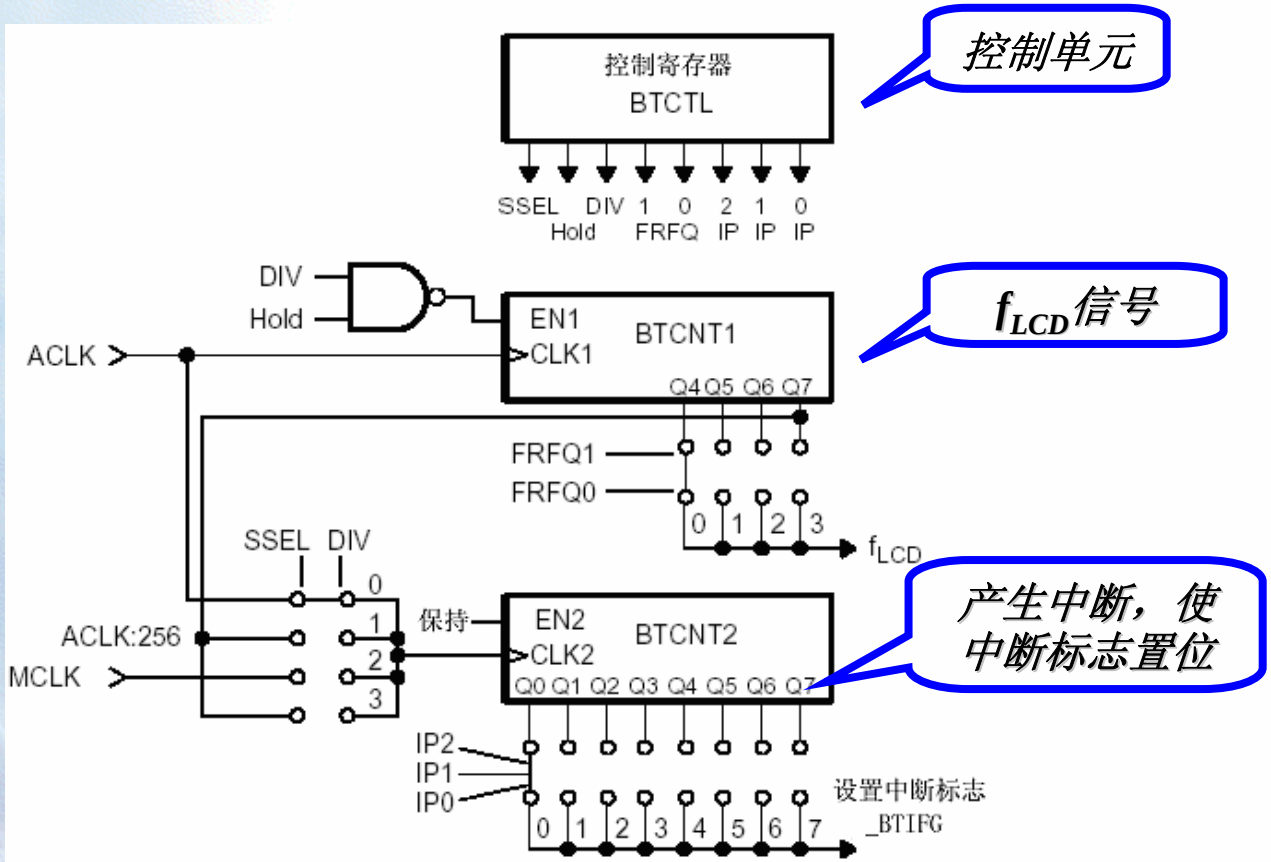

16位增计数器
写：5AH
读：69H
WDTCTL
MSB
MDB
见中断定义
Int. Flag
WDTQn
Y
4
3
2
1
Q6
Q9
Q13
Q15
16 位计数器
CLK
比较口令码
EQU
0
1
0
1
1
0
1
0
脉冲发生器
A
B
PUC
清除
异步
16–bit
低字节可写
高字节口令
R / W
EQU
SMCLK
ACLK
1
1
A
EN
启、停控制
WDTHOLD
WDTNMIES
WDTNMI
WDTTMSEL
WDTCNTCL
WDTSSEL
WDTIS1
WDTIS0
LSB
看门狗计时器控制寄存器

看门狗定时器应用举例

```
# include  <msp430x14x.h>
void main(void)
{
WDTCTL = WDT_MDLY_32;// 定时周期为32ms
IE1 |= WDTIE;              // 使能WDT中断
  P1DIR |= 0x01;             // P1.0输出
  _EINT();                   // 系统中断允许
    for (;;)
  {
    _BIS_SR(CPUOFF);         // 进入 LPM0
    _NOP();
  }
}
// 看门狗中断服务子程序
#pragma vector= WDT_VECTOR
__interrupt void watchdog_timer (void)
{
  P1OUT ^= 0x01;               // P1.0取反
}
```

基本定时器

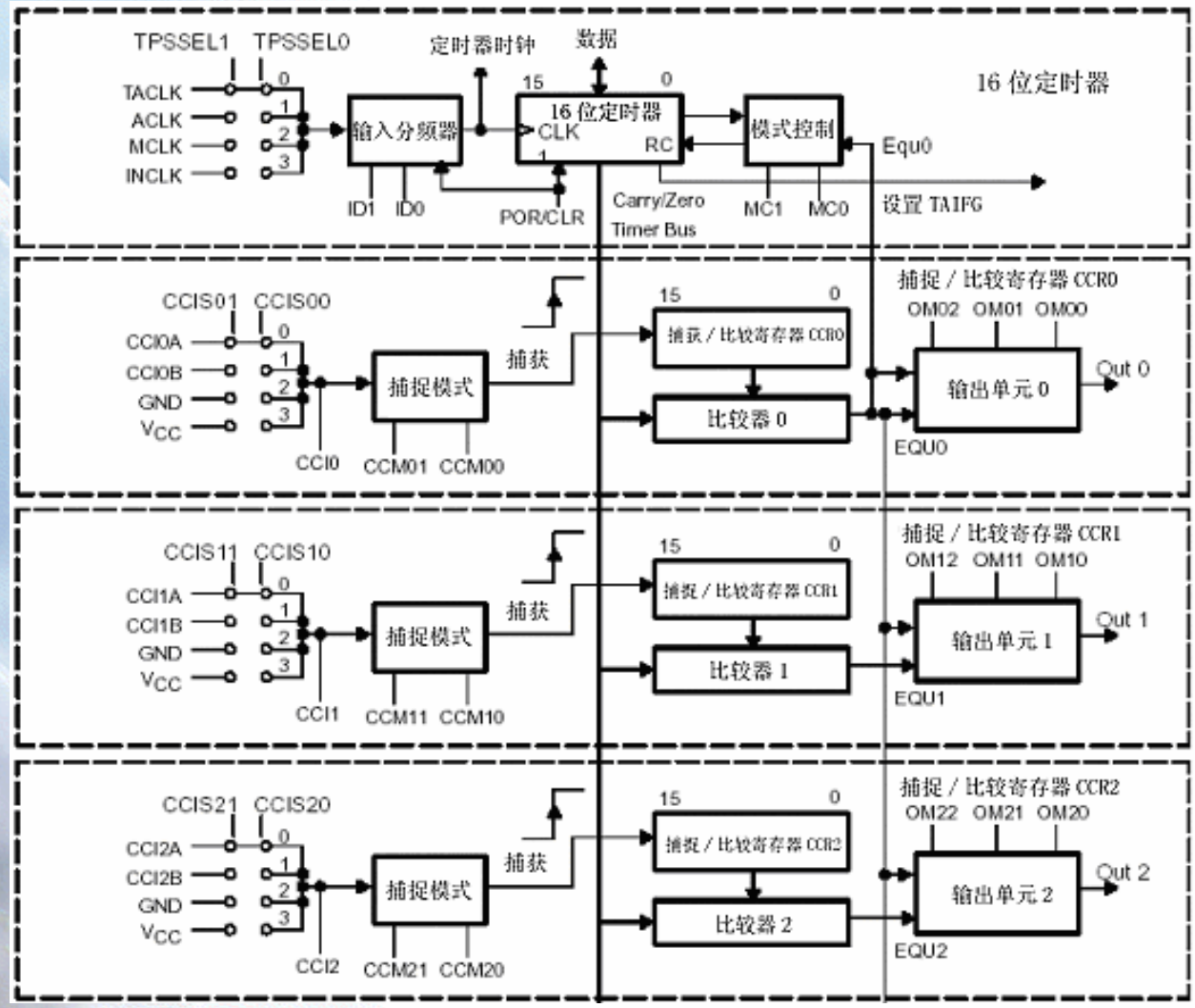
- 支持软件和各种外围模块工作在低频率、低功耗条件下



定时器A特性

- 输入时钟可以有多种选择，可是慢时钟，快时钟以及外部时钟
- 虽然没有自动重载时间常数功能，但产生的定时脉冲或PWM（脉宽调制）信号没有软件带来的误差。
- 不仅能捕获外部事件发生的时间还可锁定其发生时的高低电平。
- 可实现串行通讯
- 完善的中断服务功能
- 4种计数功能选择
- 8种输出方式选择
- 支持多时序控制
- DMA使能

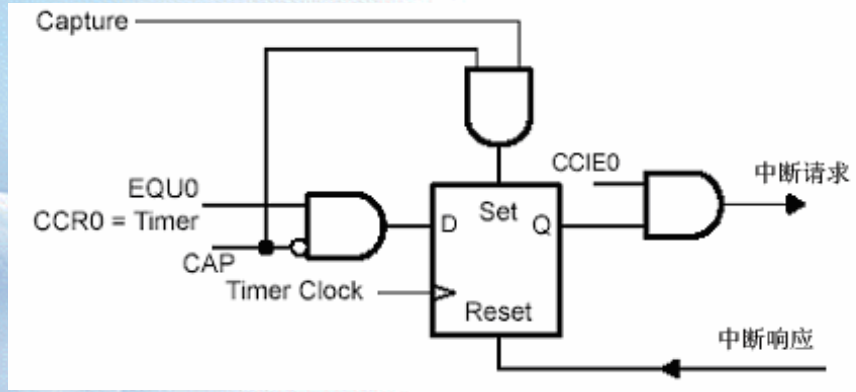
定时器A结构


TPSSEL1
TPSSEL0
定时器时钟
数据
16 位定时器
TACLK
ACLK
MCLK
INCLK
输入分频器
16 位定时器
CLK
RC
模式控制
Equ0
ID1
ID0
POR/CLR
Carry/Zero
Timer Bus
MC1
MC0
设置 TAIFG
捕捉 / 比较寄存器 CCR0
OM02 OM01 OM00
CCIS01
CCIS00
CCI0A
CCI0B
GND
Vcc
捕捉模式
捕获
捕获 / 比较寄存器 CCR0
比较器 0
输出单元 0
Out 0
EQU0
CCI0
CCM01 CCM00
捕捉 / 比较寄存器 CCR1
OM12 OM11 OM10
CCIS11
CCIS10
CCI1A
CCI1B
捕获 / 比较寄存器 CCR1
比较器 1
输出单元 1
Out 1
EQU1
CCI1
CCM11 CCM10
捕捉 / 比较寄存器 CCR2
OM22 OM21 OM20
CCIS21
CCIS20
CCI2A
CCI2B
捕获 / 比较寄存器 CCR2
比较器 2
输出单元 2
Out 2
EQU2
CCI2
CCM21 CCM20

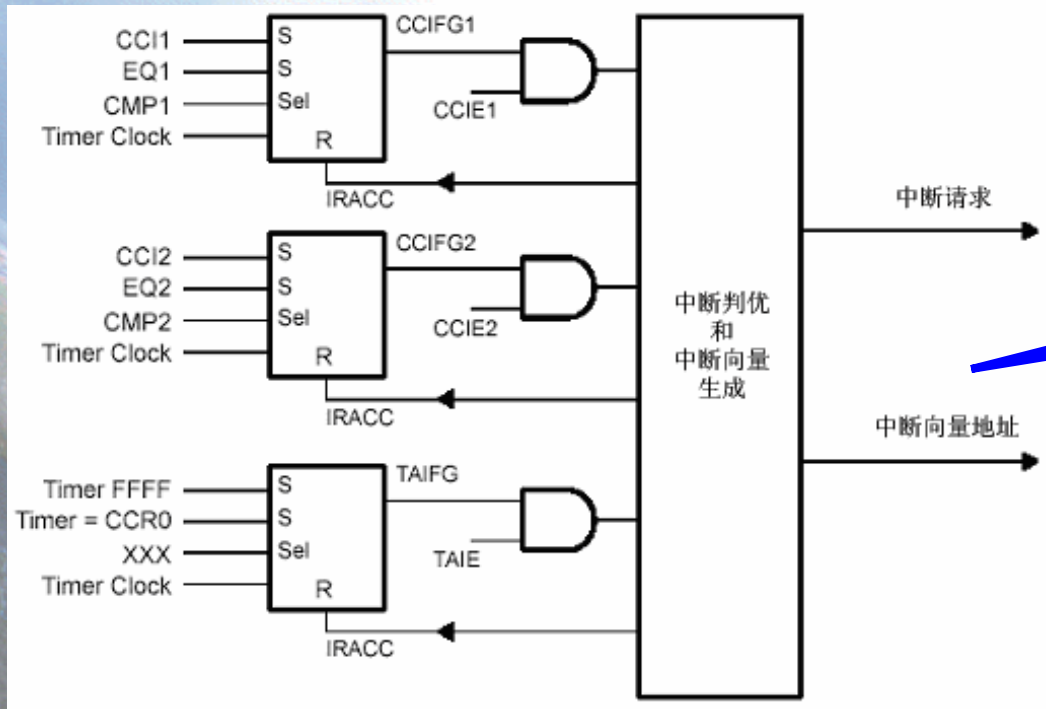
定时器A功能模块

- **计数器部分：**输入的时钟源具有4种选择，所选定的时钟源又可以1、2、4或8分频作为计数频率，Timer_A可以通过选择4种工作模式灵活的完成定时/计数功能
- **捕获/比较器：**用于捕获事件发生的时间或产生时间间隔，捕获比较功能的引入主要是为了提高I/O 端口处理事务的能力和速度。不同的MSP430单片机，Timer_A模块中所含有的捕获/比较器的数量不一样，每个捕获/比较器的结构完全相同，输入和输出都决定于各自所带的控制寄存器的控制字，捕获/比较器相互之间工作完全独立。
- **输出单元：**具有可选的8种输出模式，用于产生用户需要的输出信号。支持PWM

定时器A中断

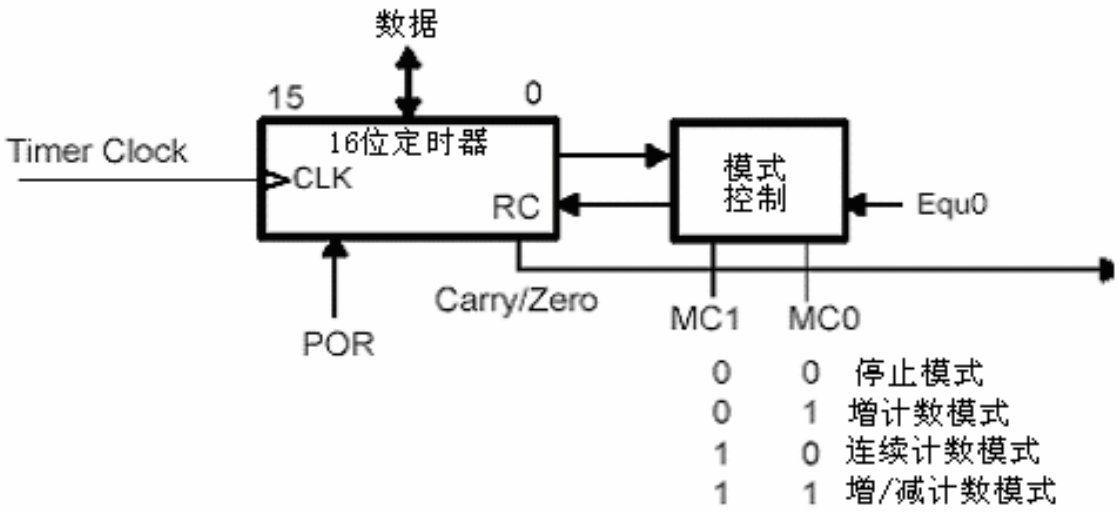


CCR0中断



CCR1～CCRx
和定时器中断

定时器工作模式

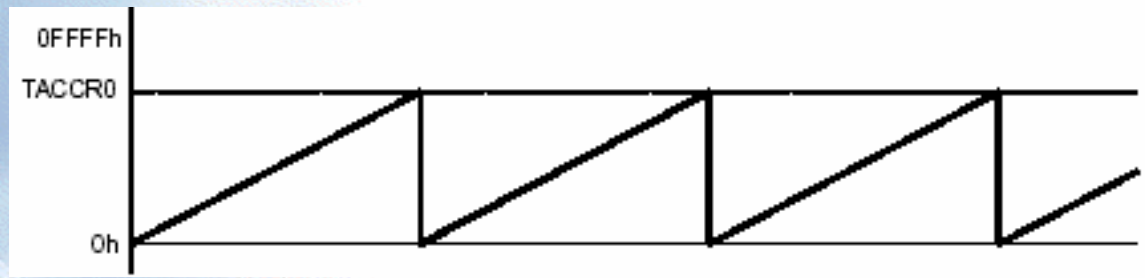

数据
15
0
Timer Clock
16位定时器
CLK
RC
模式
控制
Equ0
Carry/Zero
POR
MC1
MC0
0 0 停止模式
0 1 增计数模式
1 0 连续计数模式
1 1 增/减计数模式

停止模式

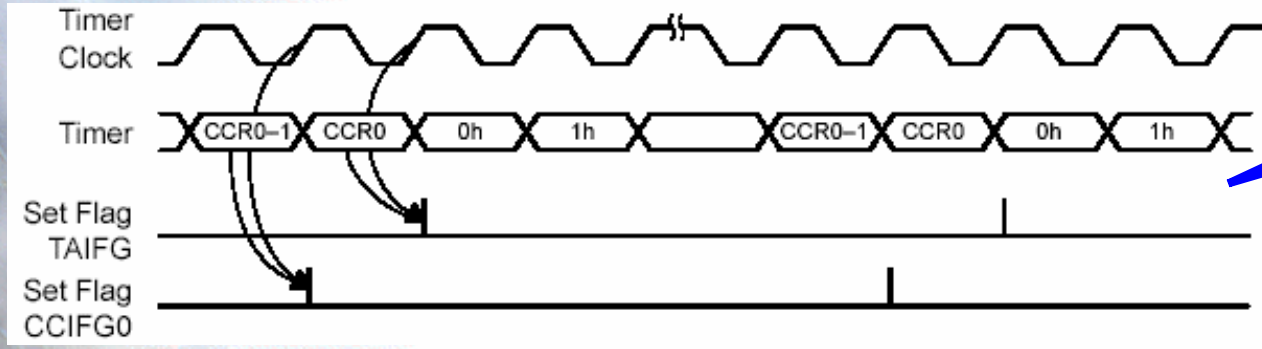
- 停止模式用于定时器暂停，并不发生复位，所有寄存器现行的内容在停止模式结束后都可用。当定时器暂停后重新计数时，计数器将从暂停时的值开始以暂停前的计数方向计数。例如，停止模式前，Timer_A工作于增/减计数模式并且处于下降计数方向，停止模式后，Timer_仍然工作于增/减计数模式，从暂停前的状态开始继续沿着下降方向开始计数。如果不能这样，则可通过TACTL中的CLR控制位来清除定时器的方向记忆特性。

增计数模式

- 捕获/比较寄存器CCR0用作Timer_A增计数模式的周期寄存器，因为CCR0为16位寄存器，所以该模式适用于定时周期小于65 536的连续计数情况。计数器TAR可以增计数到CCR0的值，当计数值与CCR0的值相等(或定时器值大于CCR0的值)时，定时器复位并从0开始重新计数。

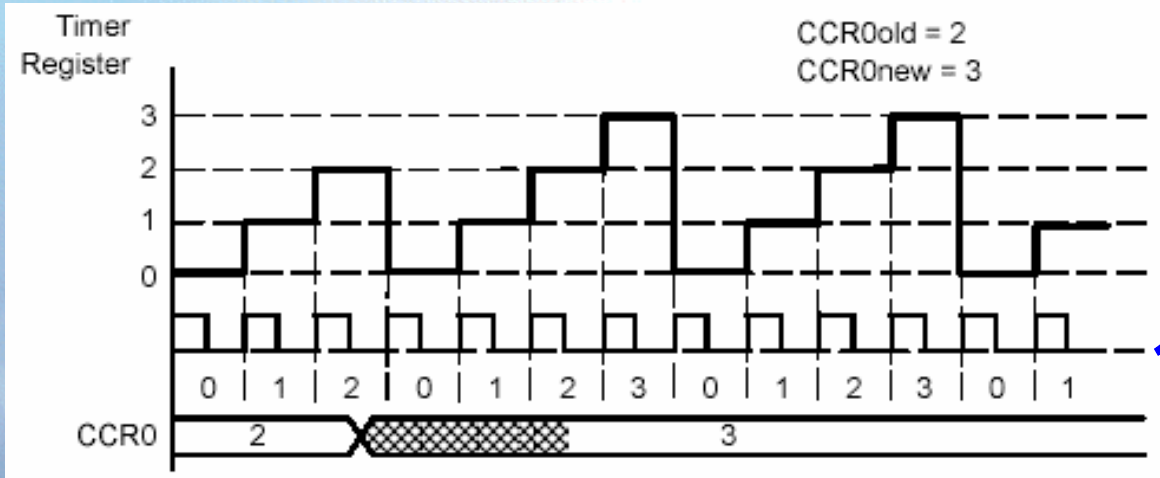


增计数模式的计数过程

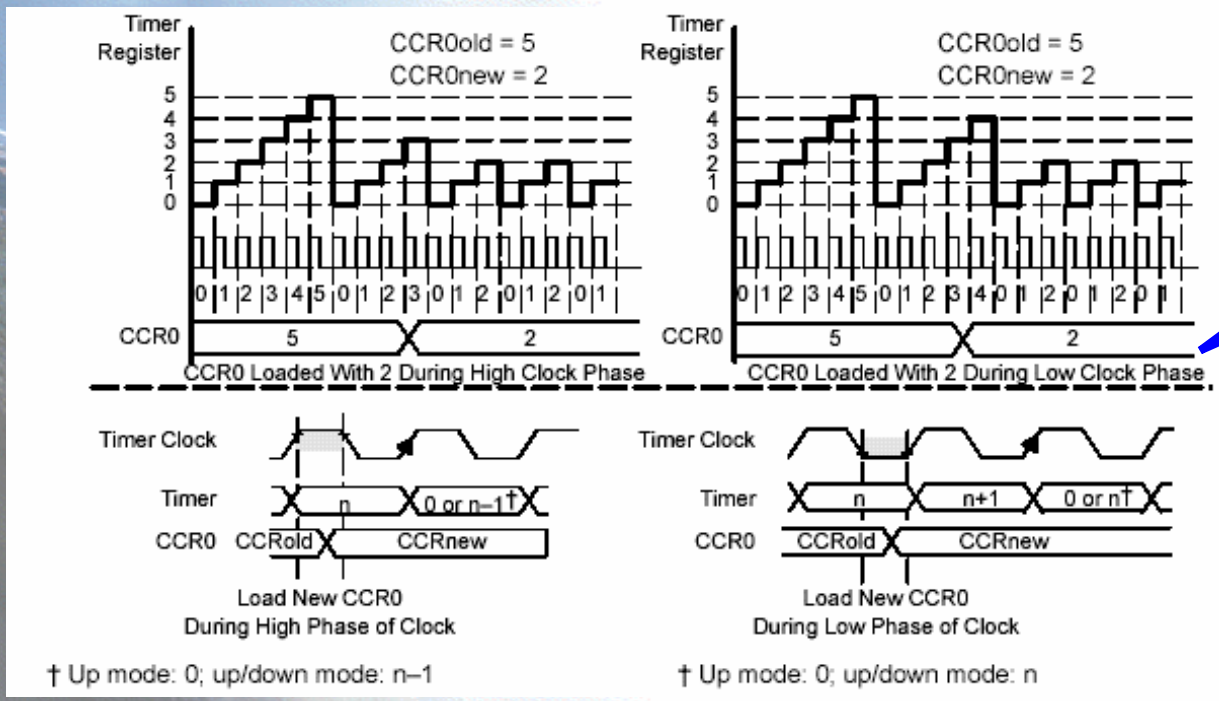


增计数模式的中断标志位设置

改变CCR0值重置计数周期—增计数方式



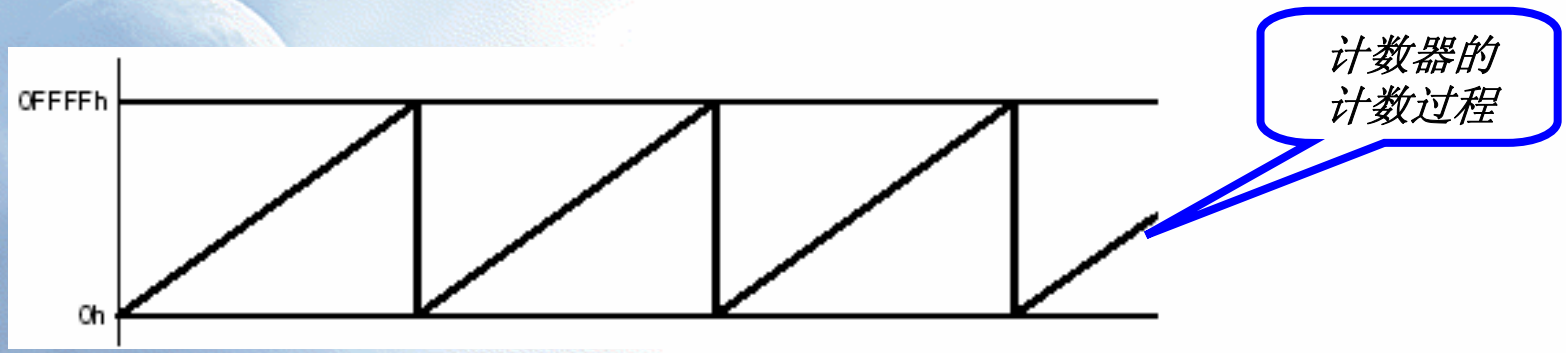
新周期大于
旧周期的响应

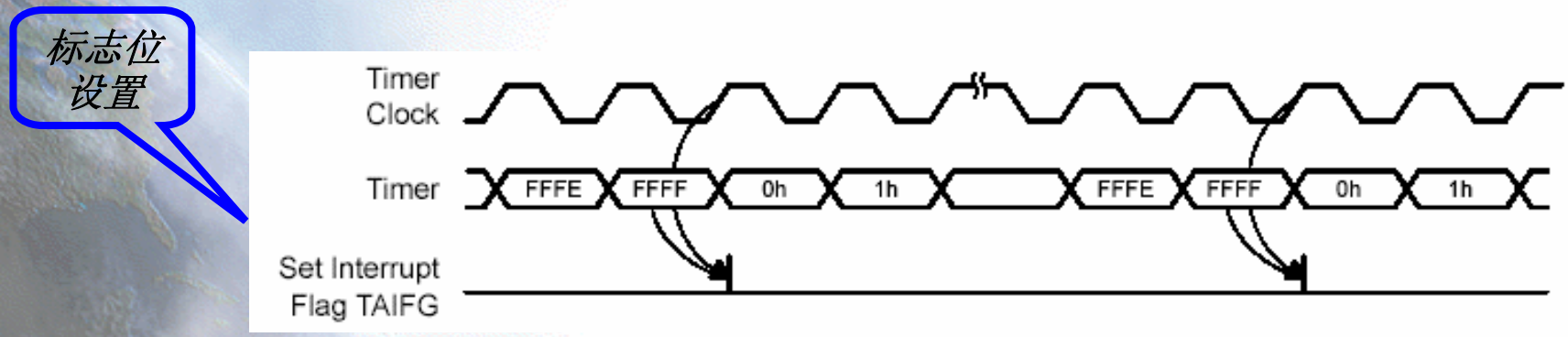


新周期小于
旧周期的响应

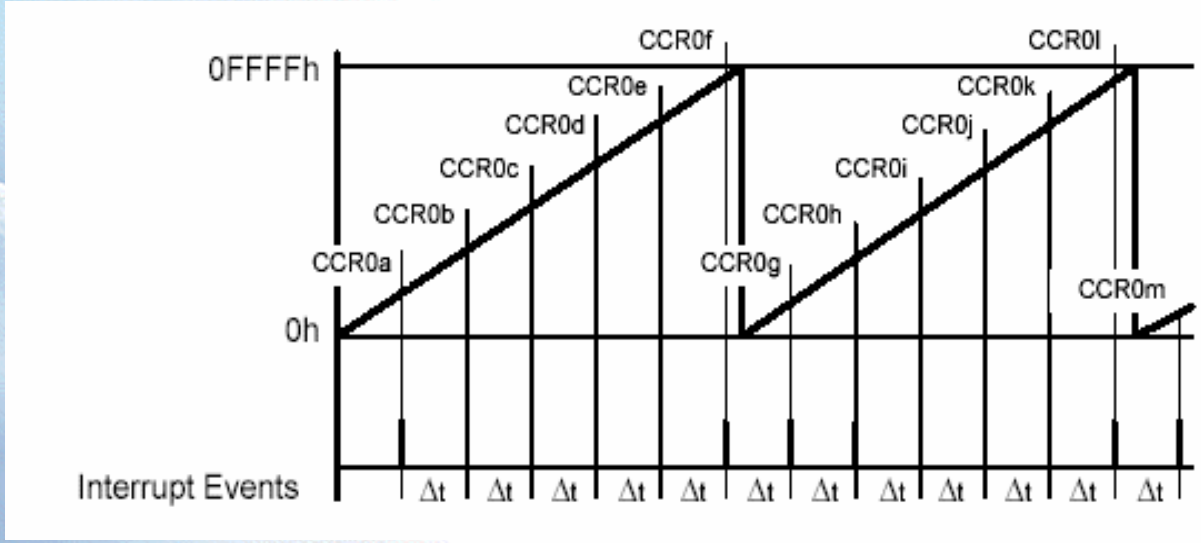
连续计数模式

- 在需要65 536个时钟周期的定时应用场合常用连续计数模式。定时器从当前值计数到0FFFFH后，又从0开始重新计数





连续计数模式的典型应用

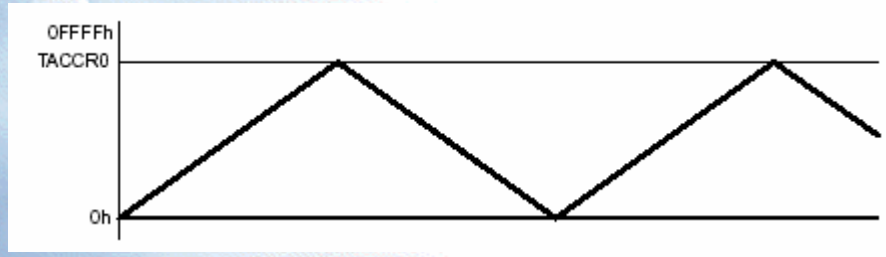


产生多个独立的时序信号：利用捕获比较寄存器捕获各种其他外部事件发生的定时器数据

*产生多个定时信号：通过中断处理程序在相应的比较寄存器**CCRx**上加上一个时间差来实现。这个时间差是当前时刻（既相应的**CCRx**中的值）到下一次中断发生时刻所经历的时间*

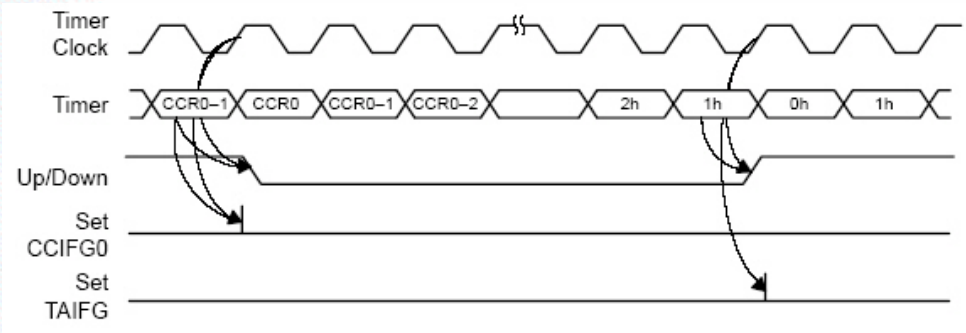
增/减计数模式

- 需要对称波形的情况经常可以使用增/减计数模式，该模式下，定时器先增计数到CCR0的值，然后反向减计数到0。计数周期仍由CCR0定义，它是CCR0计数器数值的2倍。

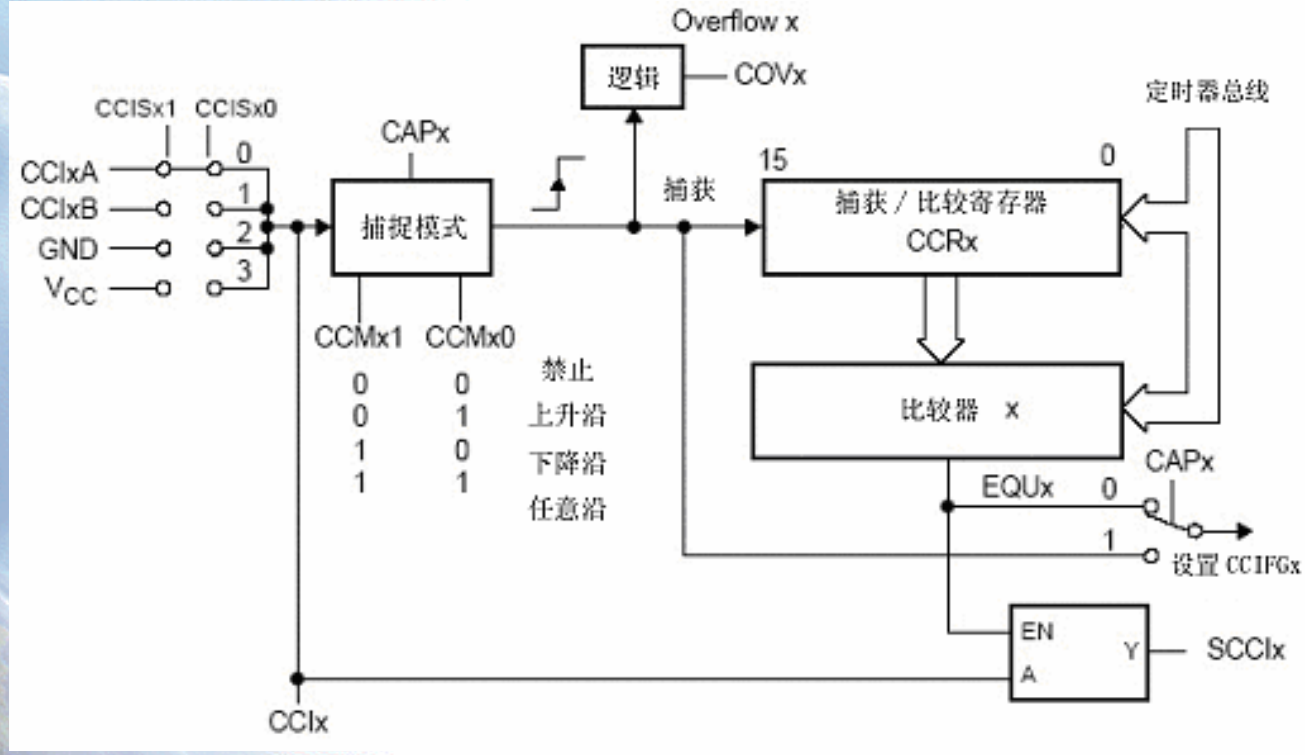


计数器的计数过程

标志位设置



捕获/比较模块

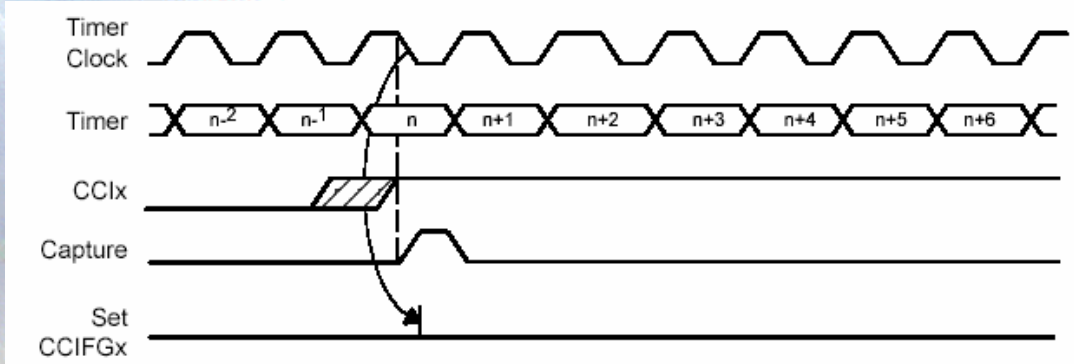

Overflow x
逻辑
COVx
定时器总线
CCISx1
CCISx0
CAPx
15
0
CCIxA
CCIxB
GND
Vcc
0
1
2
3
捕捉模式
捕获
捕获 / 比较寄存器
CCRx
CCMx1
CCMx0
0 0 禁止
0 1 上升沿
1 0 下降沿
1 1 任意沿
比较器 x
CAPx
EQUx
0
1
设置 CCIFGx
EN
A
Y
SCCIx
CCIx

捕获模式

- 当CCTLx中的CAPx=1，该模块工作在捕获模式。这时如果在选定的引脚上发生设定的脉冲触发沿（上升沿、下降沿或任意跳变），则TAR中的值将写入到CCRx中。
- 每个捕获\比较寄存器能被软件用于时间标记。可用于各种目的

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

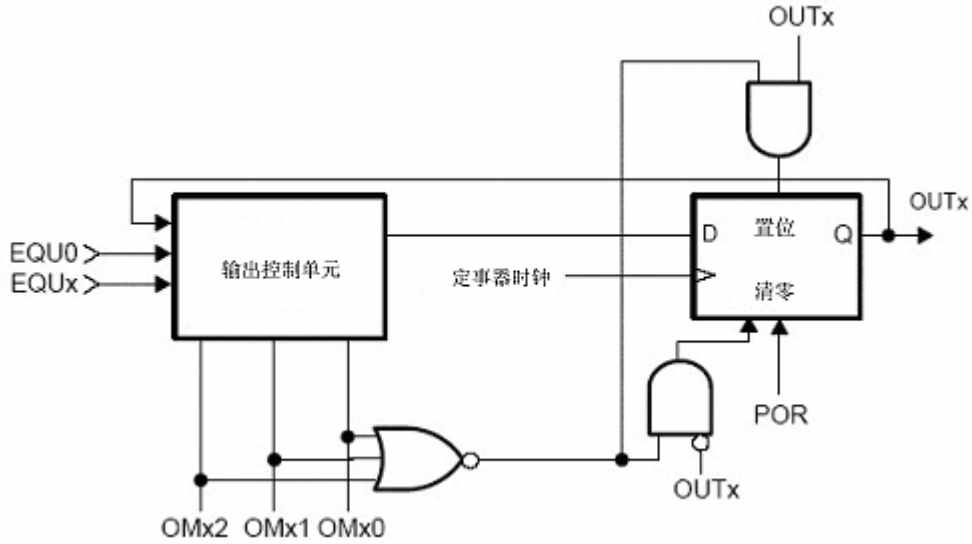
- 当捕获完成后，中断标志位CCIFGx 被置位。



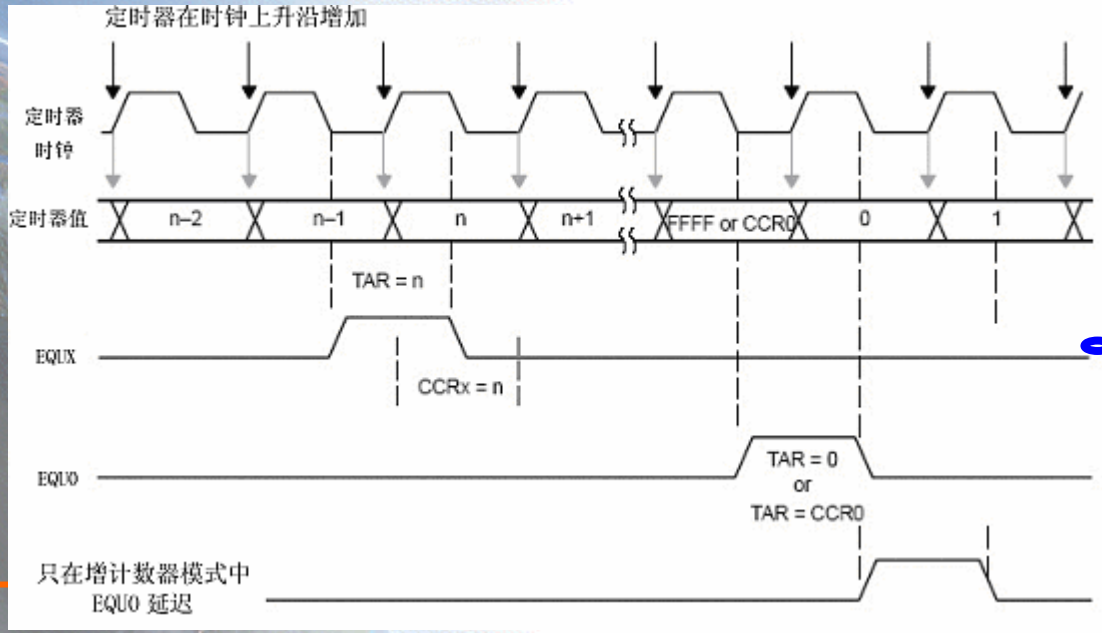
输出单元

- 每个捕获/比较模块包含一个输出单元，用于产生输出信号

输出单元的结构



输出单元时序



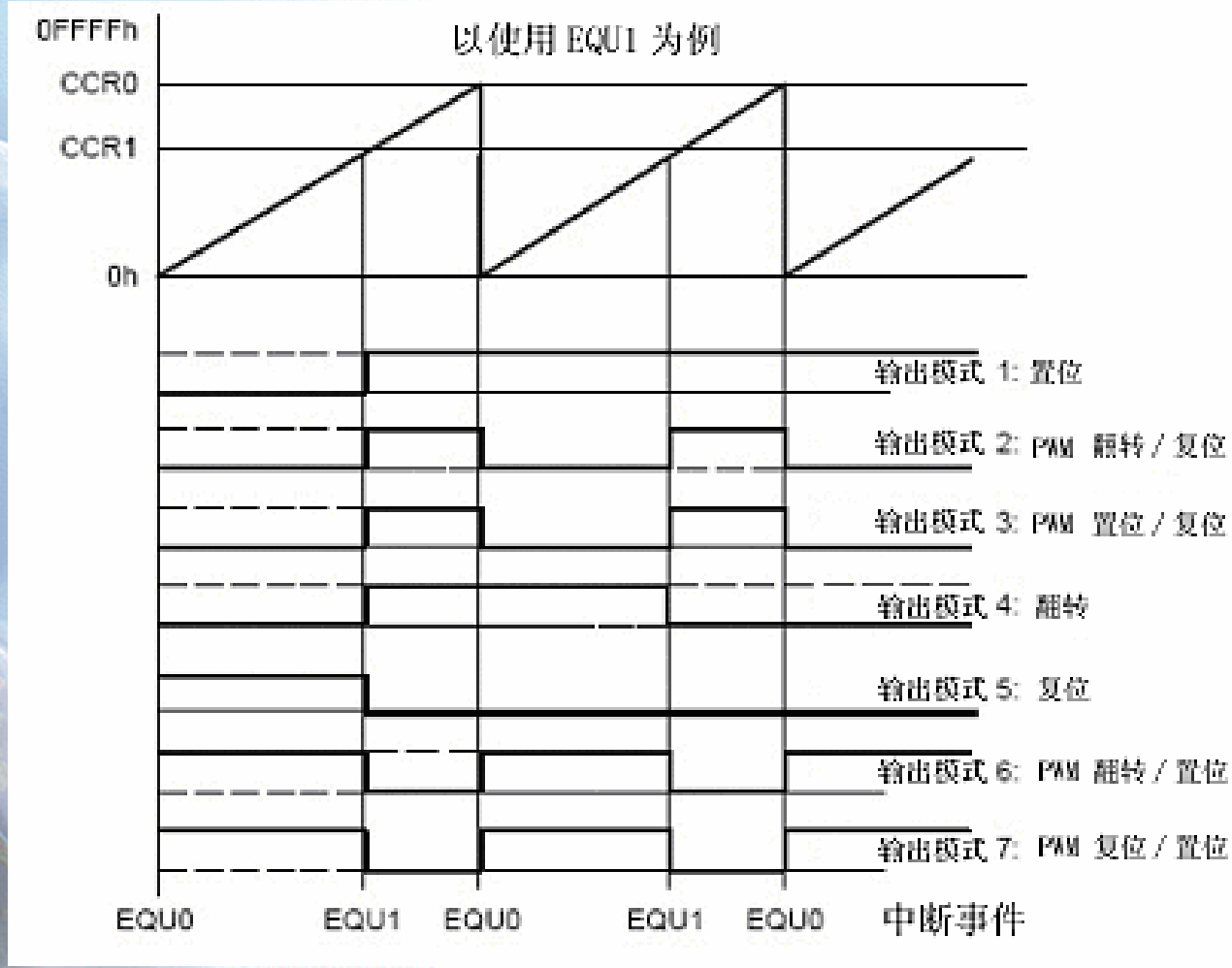
输出模式

- 输出模式0　输出模式：输出信号OUTx由每个捕获/比较模块的控制寄存器CCTLx中的OUTx位定义，并在写入该寄存器后立即更新。最终位OUTx直通。
- 输出模式1 置位模式：输出信号在TAR等于CCRx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。
- 输出模式2 PWM翻转/复位模式：输出在TAR的值等于CCRx时翻转，当TAR的值等于CCR0时复位。
- 输出模式3 PWM置位/复位模式：输出在TAR的值等于CCRx时置位，当TAR的值等于CCR0时复位。

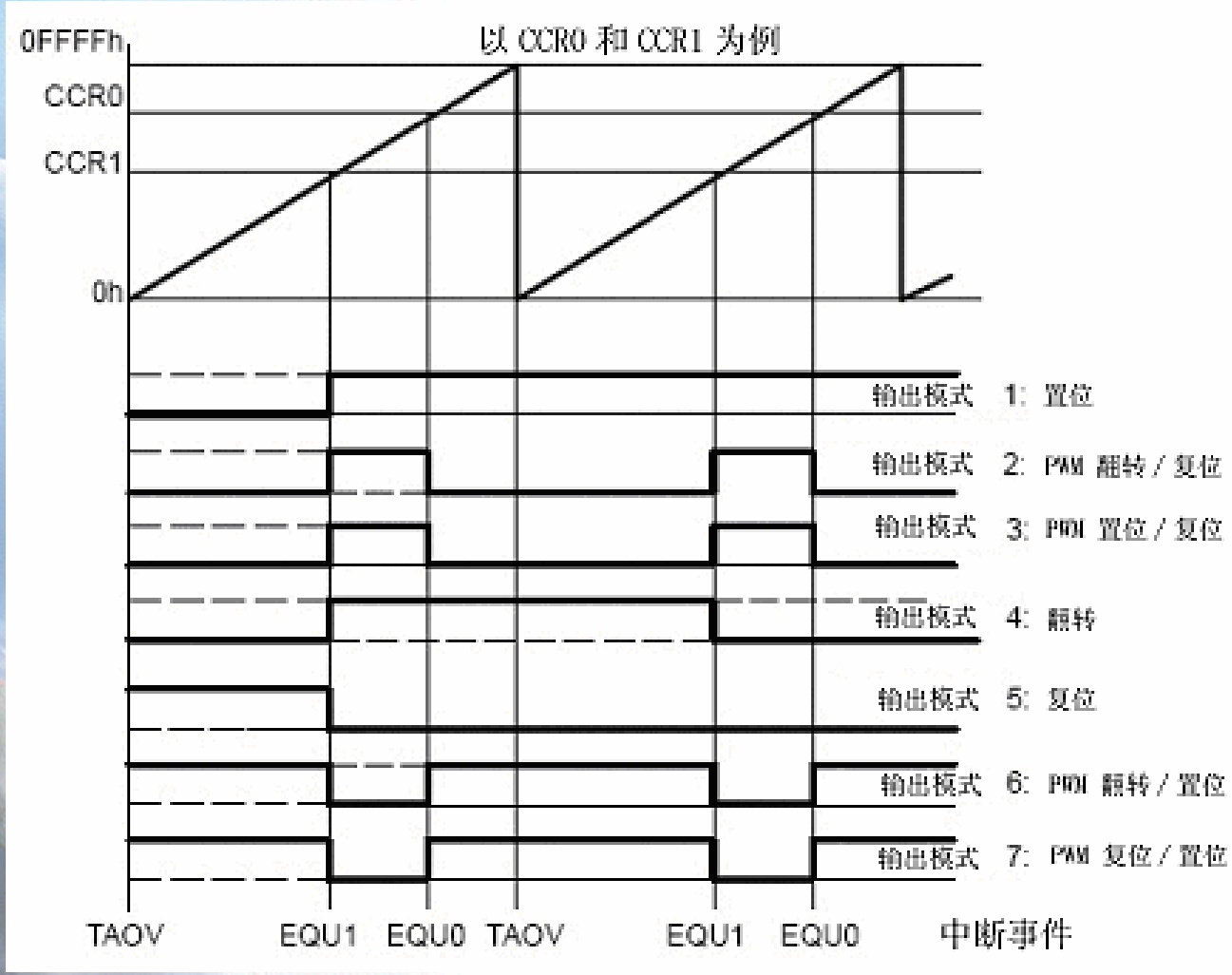
输出模式

- 输出模式4　　翻转模式：输出电平在TAR的值等于CCRx时翻转，输出周期是定时器周期的2倍。
- 输出模式5 复位模式：输出在TAR的值等于CCRx时复位，并保持低电平直到选择另一种输出模式。
- 输出模式6 PWM翻转/置位模式：输出电平在TAR的值等于CCRx时翻转，当TAR值等于CCR0时置位。
- 输出模式7 PWM复位/置位模式：输出电平在TAR的值等于CCRx时复位，当TAR的值等于CCR0时置位。

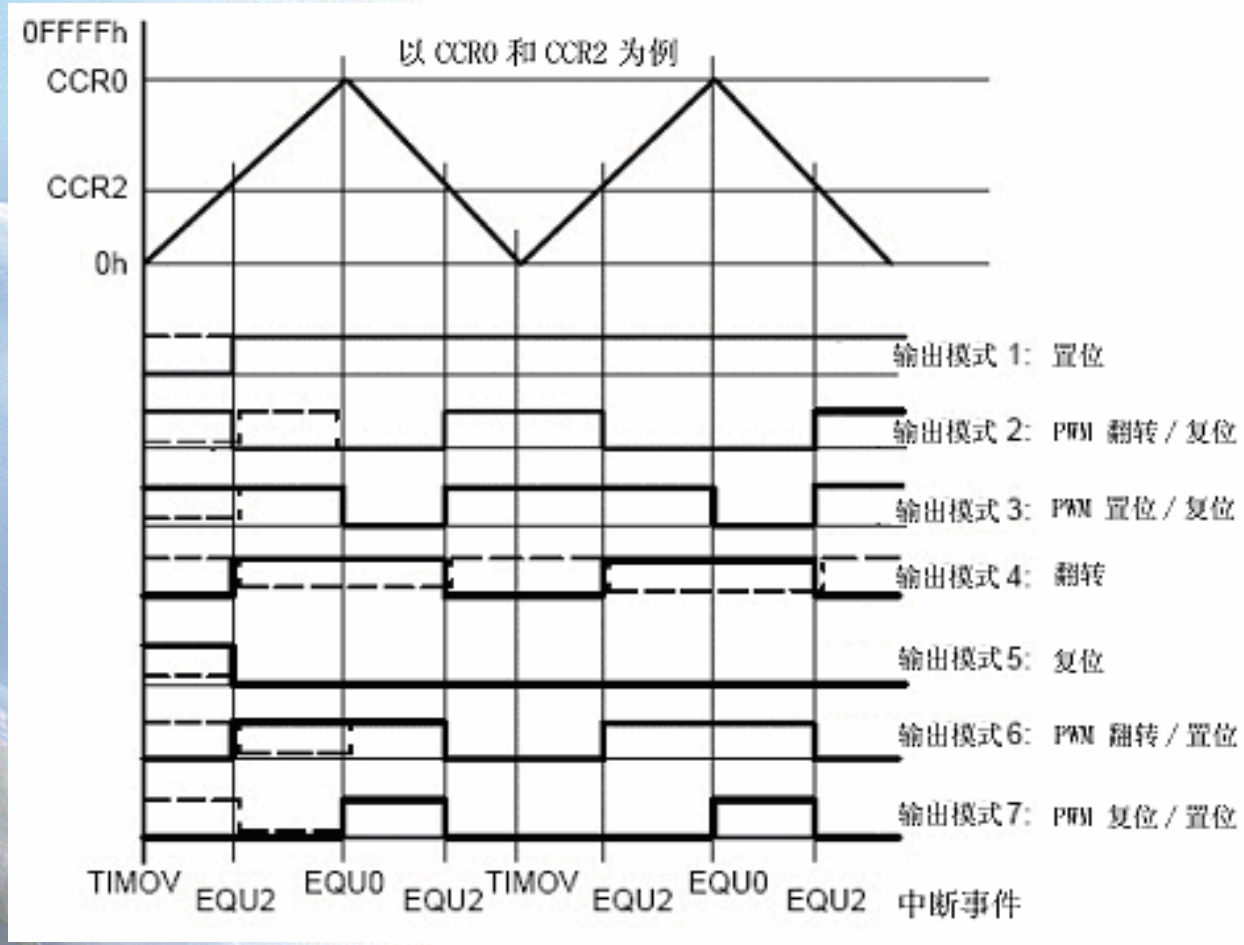
增计数模式输出实例


以使用 EQU1 为例
0FFFFh
CCR0
CCR1
0h
输出模式 1: 置位
输出模式 2: PWM 翻转 / 复位
输出模式 3: PWM 置位 / 复位
输出模式 4: 翻转
输出模式 5: 复位
输出模式 6: PWM 翻转 / 置位
输出模式 7: PWM 复位 / 置位
EQU0
EQU1
EQU0
EQU1
EQU0
中断事件

连续计数模式下的输出波形



增/减计数模式下的输出实例



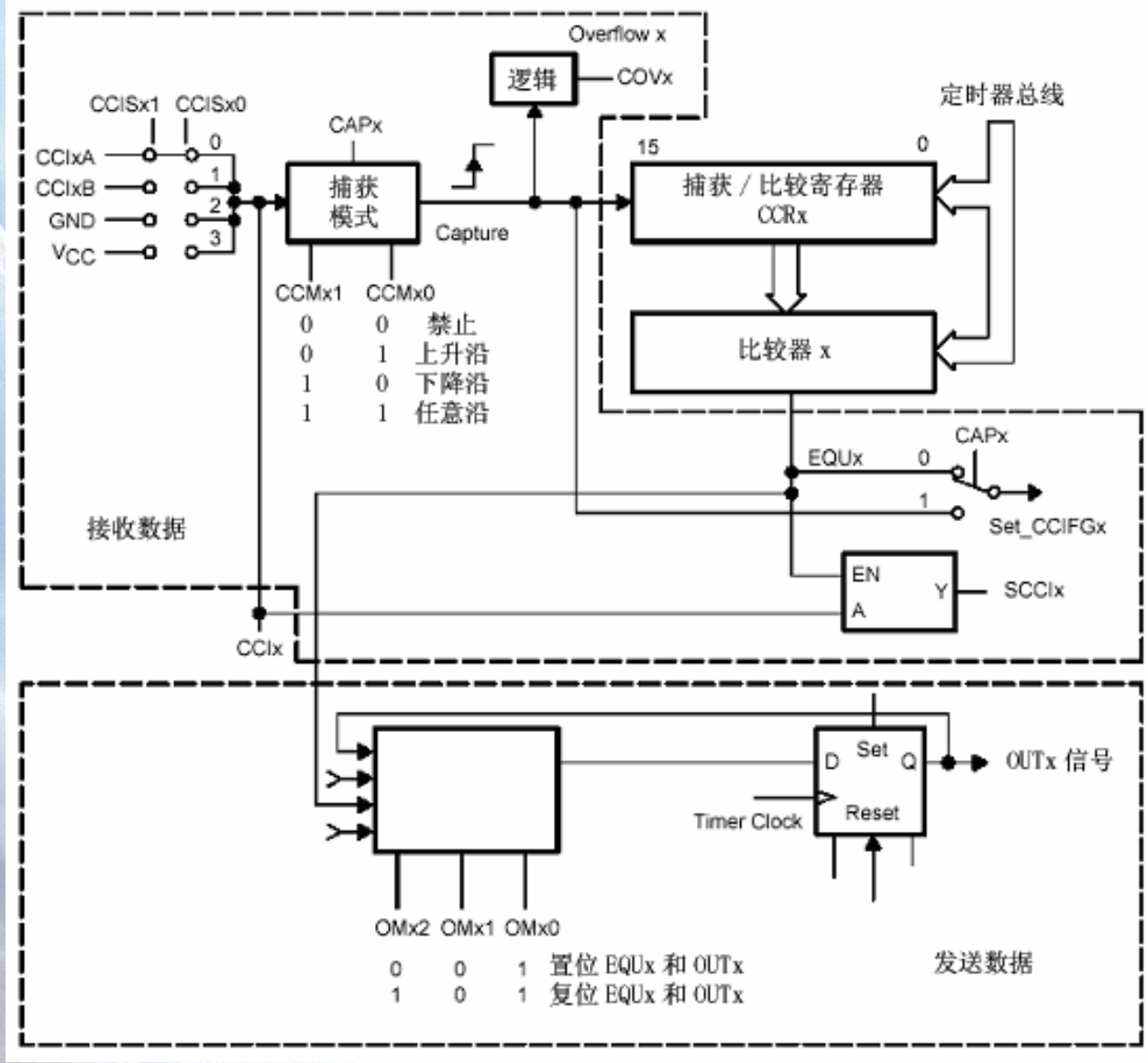
TIMER_A应用——实现异步串行通信

- Timer_A具有实现异步串行通信的一些特征

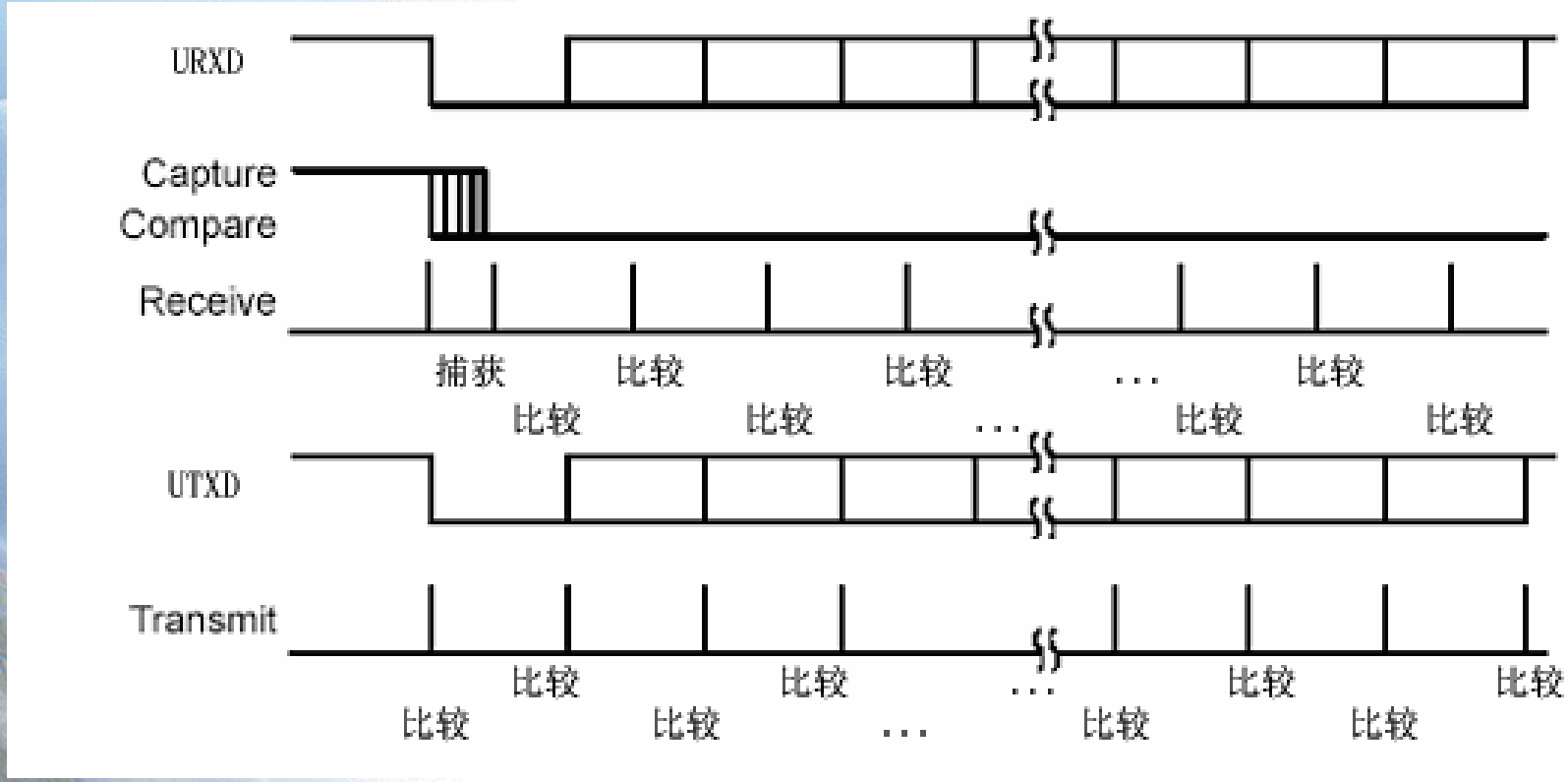
  能够自动检测起始位
  可以硬件方式产生波特率，范围从75～115200波特
  硬件锁存接收和发送的数据
  全双工方式

- 捕获功能可以捕捉选定输入引脚的状态的变化，它可以选择捕捉上升沿、下降沿、前后沿。如果捕捉到了相应的变化，则定时器计数值将被复制到捕获比较寄存器CCRx中，并会产生相应的中断。在串行通信中正是利用捕获功能的特点来捕捉起始位的信息。
- 比较功能是借助比较器不断地将CCRx中的设定值与定时器中的计数值相比较，当二者相等时就产生中断，并产生设定的输出，利用比较功能可以获得精确的时间间隔，利用该特性可以构造一个精确的波特率发生器，为串行通信提供时间基准。

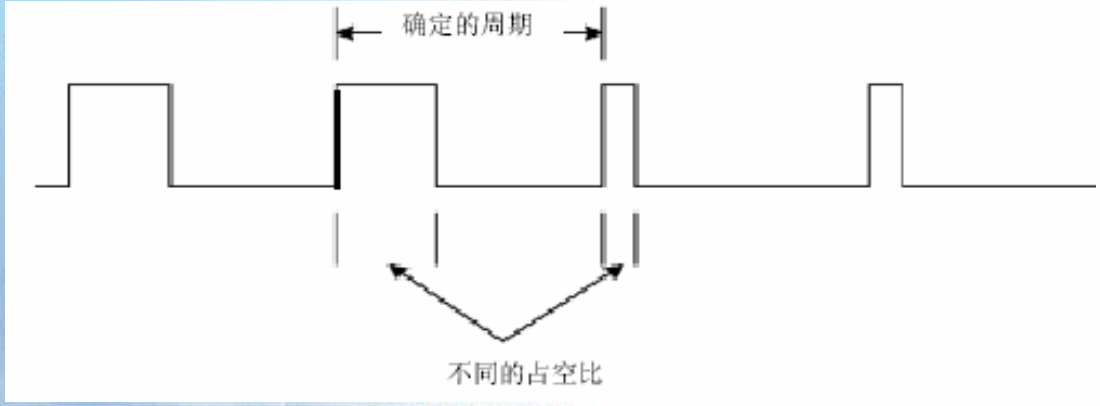
TIMER_A应用——实现异步串行通信



TIMER_A应用——实现异步串行通信

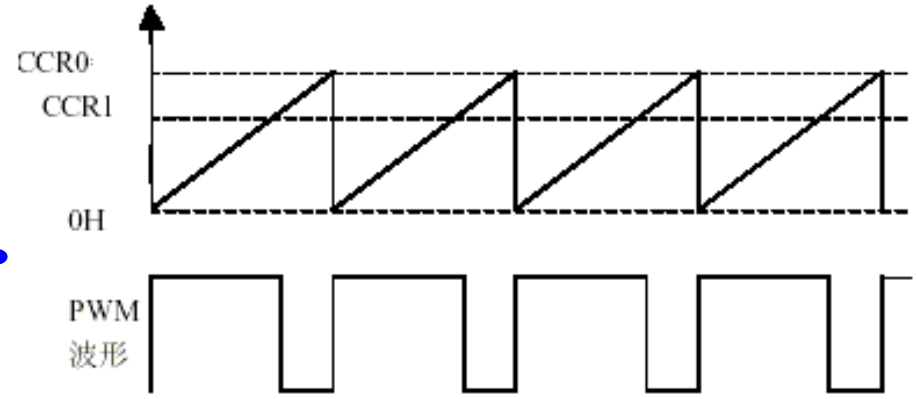


Timer_A实现PWM



PWM信号示意

Timer_A产生PWM

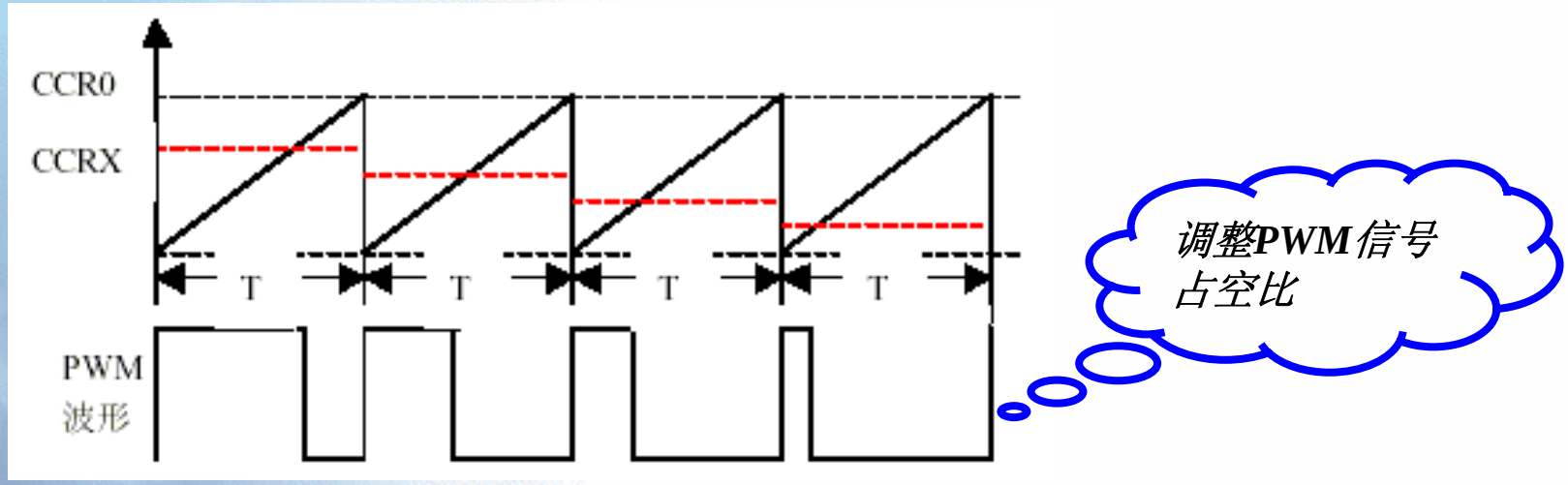


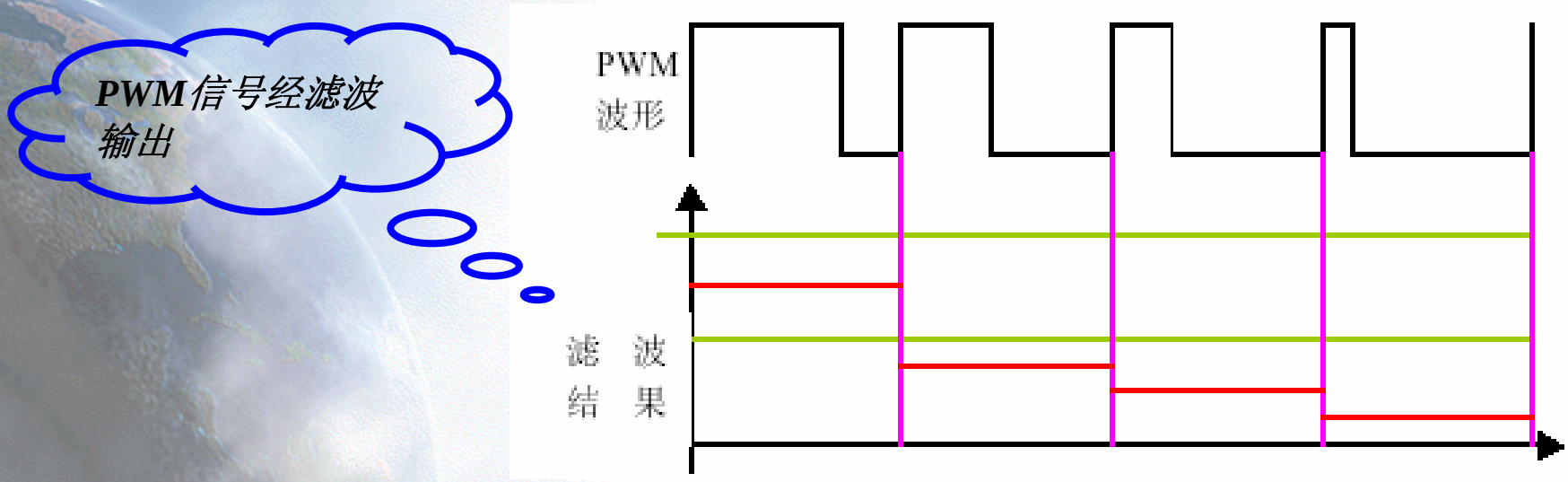
Timer_A实现PWM举例

例：设ACLK = TACLK = LFXT1 = 32768, 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：

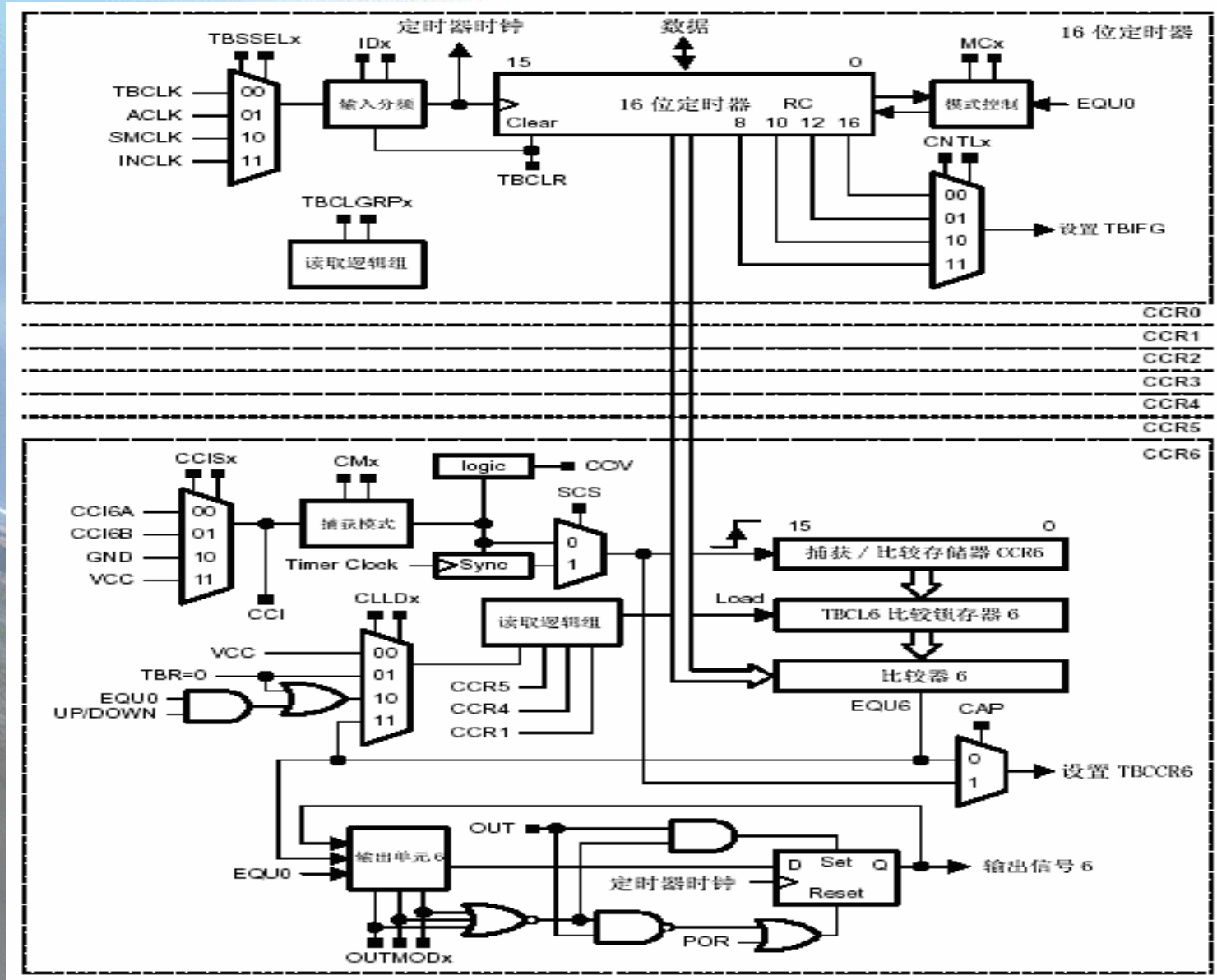
```
#include <msp430x44x.h>
void main(void)
{
WDTCTL = WDTPW +WDTHOLD;
FLL_CTL0 |= XCAP14PF;
TACTL = TASSEL0 + TACLR;                  // ACLK, 清除 TAR
  CCR0 = 512-1;                             // PWM周期
  CCTL1 = OUTMOD_7;
  CCR1 = 384;                 //占空比   384/512=0.75
  CCTL2 = OUTMOD_7;
  CCR2 = 128;                  //占空比      128/512=0.25
  P1DIR |= 0x04;                            // P1.2 输出
  P1SEL |= 0x04;                            // P1.2 TA1
  P2DIR |= 0x01;                            // P2.0 输出
  P2SEL |= 0x01;                            // P2.0 TA2
  TACTL |= MC0;                            // Timer_A 增计数模式
    for (;;)
  {
    _BIS_SR(LPM3_bits);                     // 进入 LPM3
    _NOP();
  }
}
```

PWM信号





TIMER_B


16 位定时器
定时器时钟
数据
TBSSELx
IDx
MCx
TBCLK
ACLK
SMCLK
INCLK
00
01
10
11
输入分频
15
0
16 位定时器
RC
Clear
8
10
12
16
模式控制
EQU0
TBCLR
CNTLx
TBCLGRPx
读取逻辑组
设置 TBIFG
CCR0
CCR1
CCR2
CCR3
CCR4
CCR5
CCR6
CCISx
CMx
logic
CCV
SCS
CCI6A
CCI6B
GND
VCC
捕获模式
Timer Clock
Sync
CCI
CLLDx
读取逻辑组
VCC
TBR=0
EQU0
UP/DOWN
CCR5
CCR4
CCR1
15
0
捕获 / 比较存储器 CCR6
Load
TBCL6 比较锁存器 6
比较器 6
EQU6
CAP
设置 TBCCR6
OUT
输出单元 6
EQU0
定时器时钟
D
Set
Q
Reset
输出信号 6
POR
OUTMODx

Timer_B和Timer_A共同的特征

- 4种工作模式
- 具有可选，可配置的计数器输入时钟源
- 有多个独立可配置捕获/比较模块
- 有多个具有8种输出模式的可配置输出单元
- DMA使能
- 中断功能强大，中断可能源自于计数器的溢出，也可能源自于各捕获/比较模块上发生的捕获事件或比较事件。

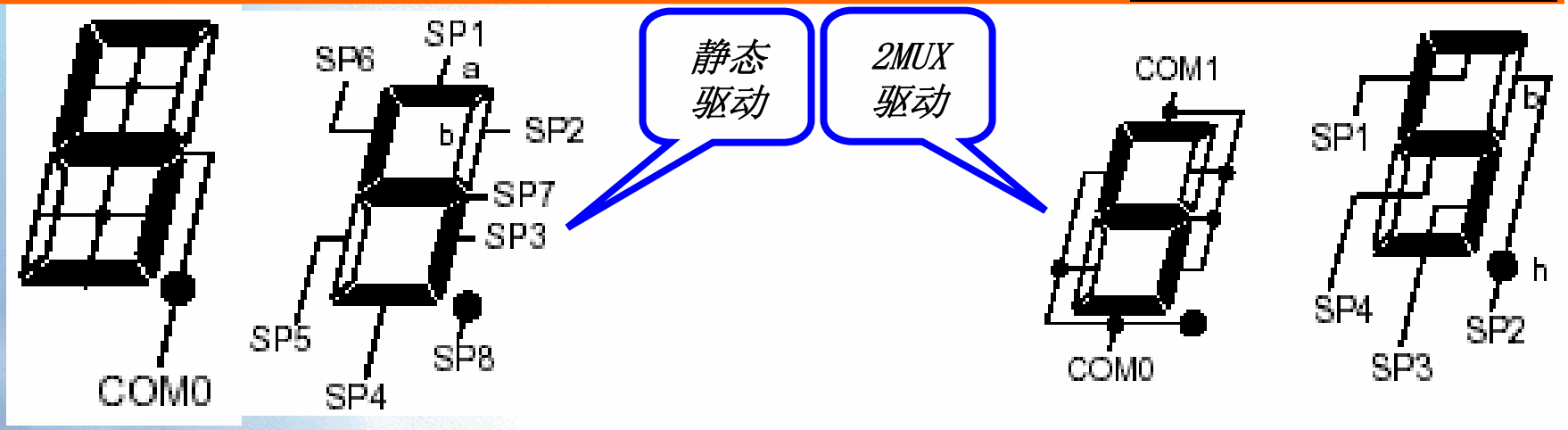
Timer_B与Timer_A不同之处

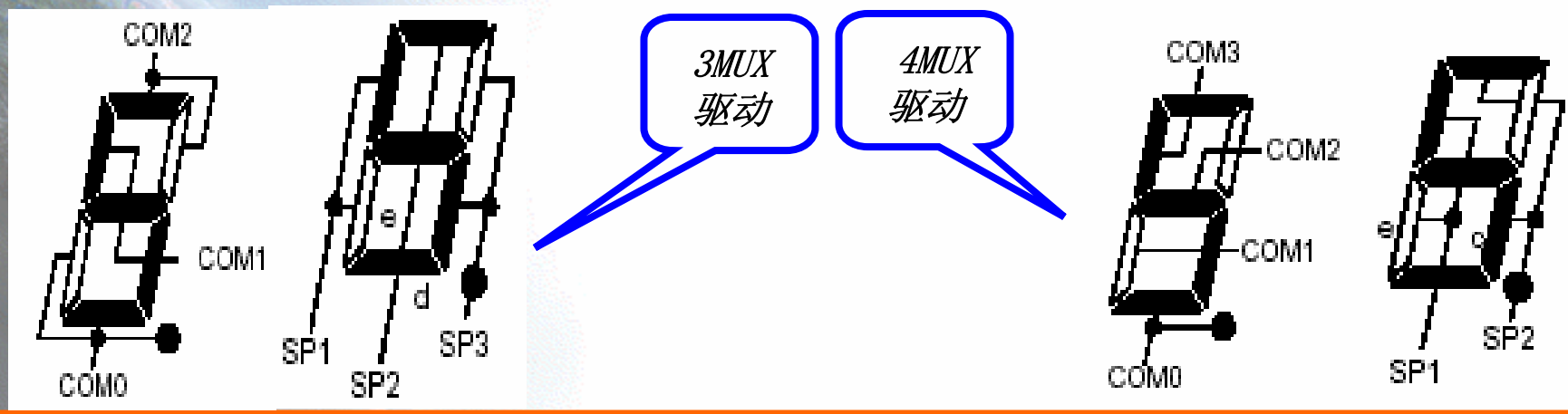
- Timer_B计数长度为8位，10位，12位和16位可编程，而Timer_A的计数长度固定为16位。
- Timer_B中没有实现Timer_A中的SCCI寄存器位的功能。
- Timer_B在比较模式下的捕获/比较寄存器功能与Timer_A不同，增加了比较锁存器
- 有些型号芯片中的Timer_B输出实现了高阻输出
- 比较模式的原理稍有不同：在Timer_A中，CCRx寄存器中保存与TAR相比较的数据；而在Timer_B中，CCRx寄存器中保存的是要比较的数据，但并不直接与定时器TBR相比较，而是将CCRx送到与之相对应的锁存器之后，由锁存器与定时器TBR相比较。从捕获/比较寄存器向比较锁存器传输数据的时机也是可以编程的，可以是在写入捕获/比较寄存器后立即传输，也可以是由一个定时事件来触发。
- Timer_B支持多重的、同步的定时功能；多重的捕获/比较功能；多重的波形输出功能（比如PWM信号）。而且，通过对比较数据的两级缓冲，可以实现多个PWM信号周期的同步更新

MSP430液晶驱动模块主要特点

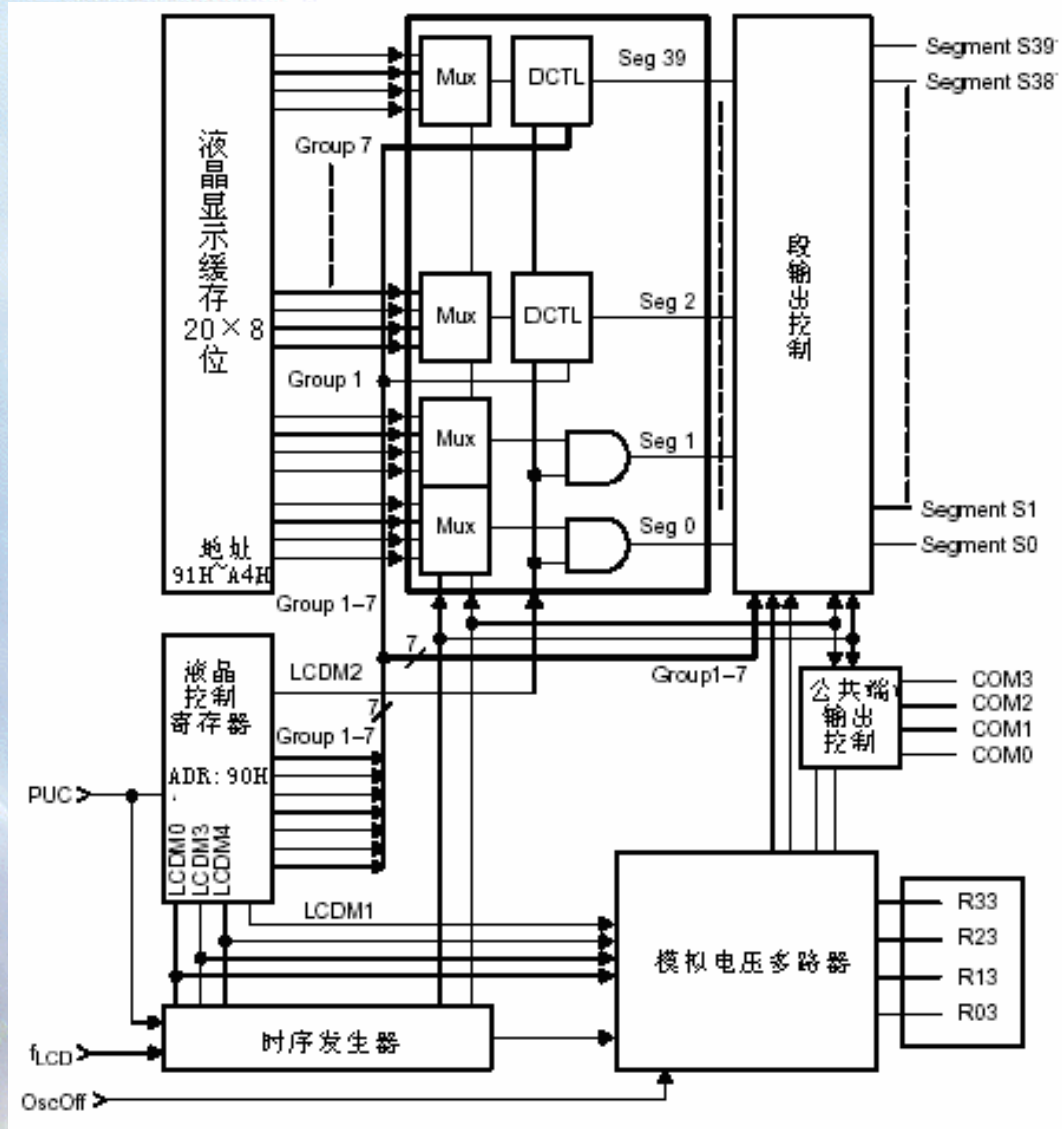
- 具有显示缓存器
- 所需的SEG、COM信号自动产生
- 4种驱动方法
- 多种扫描频率
- 段输出端口可以切换为通常输出端口
- 显示缓存器可作为一般存储器
- 用ACLK经Basic Timer产生频率

液晶驱动方法


COM0
SP6
SP1
a
b
SP2
SP7
SP3
SP5
SP8
SP4
静态
驱动
2MUX
驱动
COM1
COM0
SP1
b
h
SP4
SP2
SP3

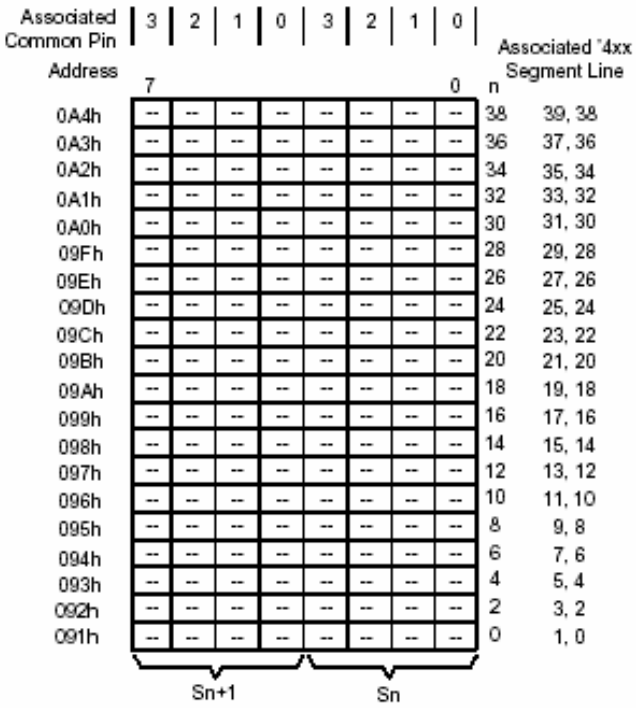

COM2
COM1
COM0
e
d
SP1
SP2
SP3
3MUX
驱动
4MUX
驱动
COM3
COM2
COM1
COM0
e
c
SP2
SP1

液晶驱动模块功能结构


液晶显示缓存 20×8 位
地址 91H~A4H
Group 7
Group 1
Group 1–7
Mux
DCTL
Seg 39
Seg 2
Seg 1
Seg 0
段输出控制
Segment S39
Segment S38
Segment S1
Segment S0
液晶控制寄存器
ADR: 90H
LCDM0
LCDM3
LCDM4
LCDM2
LCDM1
7
Group1–7
公共端输出控制
COM3
COM2
COM1
COM0
PUC
fLCD
OscOff
时序发生器
模拟电压多路器
R33
R23
R13
R03

液晶显示缓存器和段、公共极输出控制

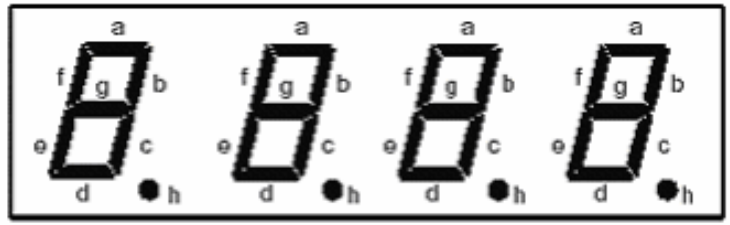
- 液晶显示缓存器各个位与液晶的段一一对应。存储位置位则可以点亮对应的液晶段，存储位复位液晶段变暗。段、公共极输出控制能够自动从显示缓存器读取数据，送出相应信号到液晶玻璃片上。因为不同器件驱动液晶的段数不同，所以液晶显示缓存器的数量也不一样。数量越大，驱动能力越强，显示的内容就越多。



静态方式显示缓存器中位与液晶段的对应关系

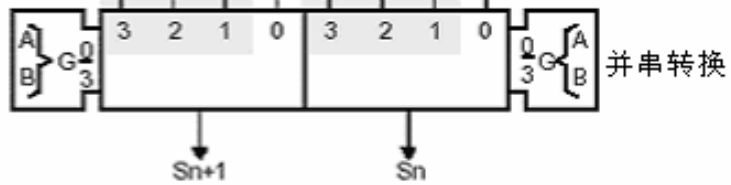
输出引脚与显示元件的连接

430 Pins		LCD Pinout PIN	COM0
S0	⇔	1	1a
S1	⇔	2	1b
S2	⇔	3	1c
S3	⇔	4	1d
S4	⇔	5	1e
S5	⇔	6	1f
S6	⇔	7	1g
S7	⇔	8	1h
S8	⇔	9	2a
S9	⇔	10	2b
S10	⇔	11	2c
S11	⇔	12	2d
S12	⇔	13	2e
S13	⇔	14	2f
S14	⇔	15	2g
S15	⇔	16	2h
S16	⇔	17	3a
S17	⇔	18	3b
S18	⇔	19	3c
S19	⇔	20	3d
S20	⇔	21	3e
S21	⇔	22	3f
S22	⇔	23	3g
S23	⇔	24	3h
S24	⇔	25	4a
S25	⇔	26	4b
S26	⇔	27	4c
S27	⇔	28	4d
S28	⇔	29	4e
S29	⇔	30	4f
S30	⇔	31	4g
S31	⇔	32	4h
COM0	⇔	33	COM0
COM1	NC		
COM2	NC		
COM3	NC		



显示缓存器

COM	3	2	1	0	3	2	1	0		
MAB 0A0h	--	--	--	h	--	--	--	g	n = 30	
09Fh	--	--	--	f	--	--	--	e	28	
09Eh	--	--	--	d	--	--	--	c	26	Digit 4
09Dh	--	--	--	b	--	--	--	a	24	
09Ch	--	--	--	h	--	--	--	g	22	
09Bh	--	--	--	f	--	--	--	e	20	Digit 3
09Ah	--	--	--	d	--	--	--	c	18	
099h	--	--	--	b	--	--	--	a	16	
098h	--	--	--	h	--	--	--	g	14	
097h	--	--	--	f	--	--	--	e	12	Digit 2
096h	--	--	--	d	--	--	--	c	10	
095h	--	--	--	b	--	--	--	a	8	
094h	--	--	--	h	--	--	--	g	6	
093h	--	--	--	f	--	--	--	e	4	Digit 1
092h	--	--	--	d	--	--	--	c	2	
091h	--	--	--	b	--	--	--	a	0	

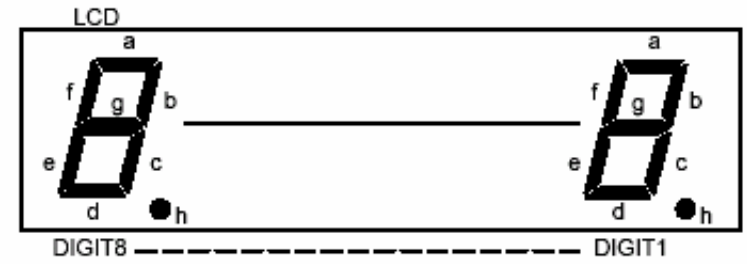


2MUX方式显示缓存器中位与液晶段的对应关系

Pinout and Connections

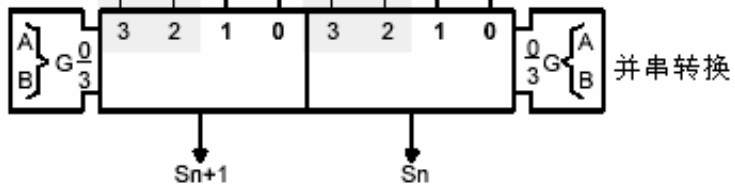
Connections

430 Pins		LCD Pinout PIN	COM0	COM1
S0	⟷	1	1f	1a
S1	⟷	2	1h	1b
S2	⟷	3	1d	1c
S3	⟷	4	1e	1g
S4	⟷	5	2f	2a
S5	⟷	6	2h	2b
S6	⟷	7	2d	2c
S7	⟷	8	2e	2g
S8	⟷	9	3f	3a
S9	⟷	10	3h	3b
S10	⟷	11	3d	3c
S11	⟷	12	3e	3g
S12	⟷	13	4f	4a
S13	⟷	14	4h	4b
S14	⟷	15	4d	4c
S15	⟷	16	4e	4g
S16	⟷	17	5f	5a
S17	⟷	18	5h	5b
S18	⟷	19	5d	5c
S19	⟷	20	5e	5g
S20	⟷	21	6f	6a
S21	⟷	22	6h	6b
S22	⟷	23	6d	6c
S23	⟷	24	6e	6g
S24	⟷	25	7f	7a
S25	⟷	26	7h	7b
S26	⟷	27	7d	7c
S27	⟷	28	7e	7g
S28	⟷	29	8f	8a
S29	⟷	30	8h	8b
S30	⟷	31	8d	8c
S31	⟷	32	8e	8g
COM0	⟷	33	COM0	
COM1	⟷	34		COM1
COM2	NC			
COM3	NC			



Display Memory

	COM	3	2	1	0	3	2	1	0		
MAB	0A0h	--	--	g	e	--	--	c	d	n = 30	1/2 Digit 8
	09Fh	--	--	b	h	--	--	a	f	28	
	09Eh	--	--	g	e	--	--	c	d	26	Digit 7
	09Dh	--	--	b	h	--	--	a	f	24	
	09Ch	--	--	g	e	--	--	c	d	22	Digit 6
	09Bh	--	--	b	h	--	--	a	f	20	
	09Ah	--	--	g	e	--	--	c	d	18	Digit 5
	099h	--	--	b	h	--	--	a	f	16	
	098h	--	--	g	e	--	--	c	d	14	Digit 4
	097h	--	--	b	h	--	--	a	f	12	
	096h	--	--	g	e	--	--	c	d	10	Digit 3
	095h	--	--	b	h	--	--	a	f	8	
	094h	--	--	g	e	--	--	c	d	6	Digit 2
	093h	--	--	b	h	--	--	a	f	4	
	092h	--	--	g	e	--	--	c	d	2	Digit 1
	091h	--	--	b	h	--	--	a	f	0	



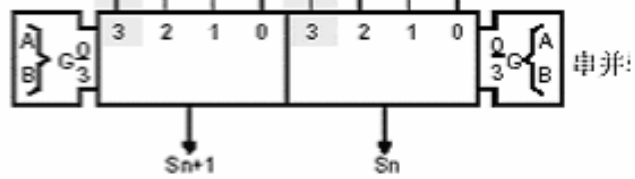
3MUX方式显示缓存器中位与液晶段的对应关系

输出引脚与显示元件的连接

'430 Pins		LCD Pinout PIN	COM0	COM1	COM2
S0	↔	1	1e	1f	1y
S1	↔	2	1d	1g	1a
S2	↔	3	1h	1c	1b
S3	↔	4	2e	2f	2y
S4	↔	5	2d	2g	2a
S5	↔	6	2h	2c	2b
S6	↔	7	3e	3f	3y
S7	↔	8	3d	3g	3a
S8	↔	9	3h	3c	3b
S9	↔	10	4e	4f	4y
S10	↔	11	4d	4g	4a
S11	↔	12	4h	4c	4b
S12	↔	13	5e	5f	5y
S13	↔	14	5d	5g	5a
S14	↔	15	5h	5c	5b
S15	↔	16	6e	6f	6y
S16	↔	17	6d	6g	6a
S17	↔	18	6h	6c	6b
S18	↔	19	7e	7f	7y
S19	↔	20	7d	7g	7a
S20	↔	21	7h	7c	7b
S21	↔	22	8e	8f	8y
S22	↔	23	8d	8g	8a
S23	↔	24	8h	8c	8b
S24	↔	25	9e	9f	9y
S25	↔	26	9d	9g	9a
S26	↔	27	9h	9c	9b
S27	↔	28	10e	10f	10y
S28	↔	29	10d	10g	10a
S29	↔	30	10h	10c	10b
COM0	↔	31	COM0		
COM1	↔	32		COM1	
COM2	↔	33			COM2
COM3	NC				

显示缓存器

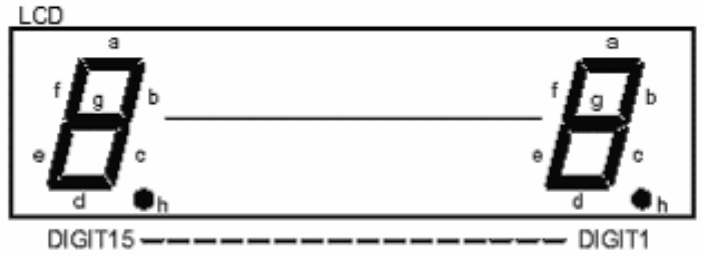
	COM	3	2	1	0	3	2	1	0		
MAB	09Fh	--	a	g	d	--	y	f	e	n = 30	
	09Eh	--	b	c	h	--	a	g	d	28	Digit 1
	09Dh	--	y	f	e	--	b	c	h	26	Digit 9
	09Ch	--	a	g	d	--	y	f	e	24	
	09Bh	--	b	c	h	--	a	g	d	22	Digit 8
	09Ah	--	y	f	e	--	b	c	h	20	Digit 7
	099h	--	a	g	d	--	y	f	e	18	
	098h	--	b	c	h	--	a	g	d	16	Digit 6
	097h	--	y	f	e	--	b	c	h	14	Digit 5
	096h	--	a	g	d	--	y	f	e	12	
	095h	--	b	c	h	--	a	g	d	10	Digit 4
	094h	--	y	f	e	--	b	c	h	8	Digit 3
	093h	--	a	g	d	--	y	f	e	6	
	092h	--	b	c	h	--	a	g	d	4	Digit 2
	091h	--	y	f	e	--	b	c	h	2	Digit 1
		--	a	g	d	--	y	f	e	0	



4MUX方式显示缓存器中位与液晶段的对应关系

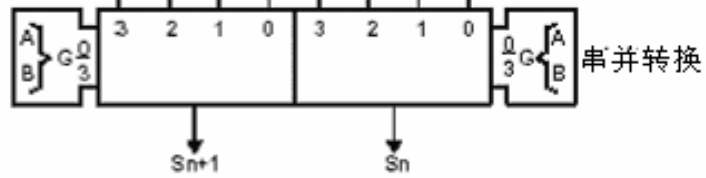
输出引脚与显示元件的连接

430 Pins		PIN	LCD Pinout COM0	COM1	COM2	COM3
S0	↔	1	1d	1e	1g	1f
S1	↔	2	1h	1c	1b	1a
S2	↔	3	2d	2e	2g	2f
S3	↔	4	2h	2c	2b	2a
S4	↔	5	3d	3e	3g	3f
S5	↔	6	3h	3c	3b	3a
S6	↔	7	4d	4e	4g	4f
S7	↔	8	4h	4c	4b	4a
S8	↔	9	5d	5e	5g	5f
S9	↔	10	5h	5c	5b	5a
S10	↔	11	6d	6e	6g	6f
S11	↔	12	6h	6c	6b	6a
S12	↔	13	7d	7e	7g	7f
S13	↔	14	7h	7c	7b	7a
S14	↔	15	8d	8e	8g	8f
S15	↔	16	8h	8c	8b	8a
S16	↔	17	9d	9e	9g	9f
S17	↔	18	9h	9c	9b	9a
S18	↔	19	10d	10e	10g	10f
S19	↔	20	10h	10c	10b	10a
S20	↔	21	11d	11e	11g	11f
S21	↔	22	11h	11c	11b	11a
S22	↔	23	12d	12e	12g	12f
S23	↔	24	12h	12c	12b	12a
S24	↔	25	13d	13e	13g	13f
S25	↔	26	13h	13c	13b	13a
S26	↔	27	14d	14e	14g	14f
S27	↔	28	14h	14c	14b	14a
S28	↔	29	15d	15e	15g	15f
S29	↔	30	15h	15c	15b	15a
COM0	↔	31	COM0			
COM1	↔	32		COM1		
COM2	↔	33			COM2	
COM3	↔	34				COM3

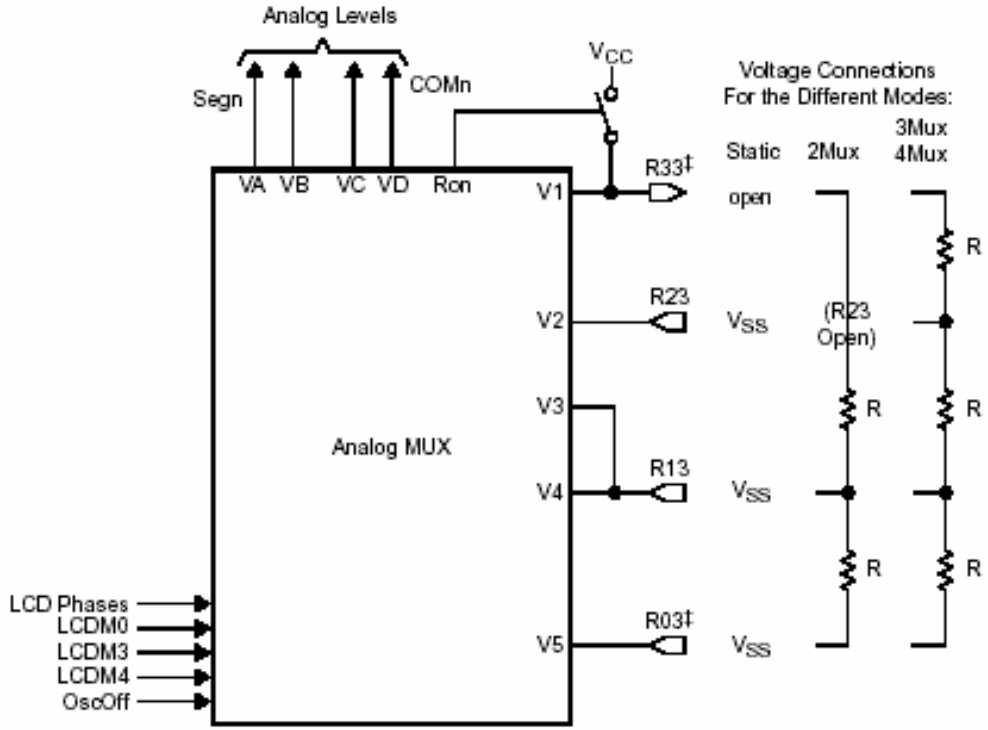


显示缓存器

	COM	3	2	1	0	3	2	1	0	
MAB	09Fh	a	b	c	h	f	g	e	d	n = 30 Digit
	09Eh	a	b	c	h	f	g	e	d	28 Digit
	09Dh	a	b	c	h	f	g	e	d	26 Digit
	09Ch	a	b	c	h	f	g	e	d	24 Digit
	09Bh	a	b	c	h	f	g	e	d	22 Digit
	09Ah	a	b	c	h	f	g	e	d	20 Digit
	099h	a	b	c	h	f	g	e	d	18 Digit
	098h	a	b	c	h	f	g	e	d	16 Digit
	097h	a	b	c	h	f	g	e	d	14 Digit
	096h	a	b	c	h	f	g	e	d	12 Digit
	095h	a	b	c	h	f	g	e	d	10 Digit
	094h	a	b	c	h	f	g	e	d	8 Digit
	093h	a	b	c	h	f	g	e	d	6 Digit
	092h	a	b	c	h	f	g	e	d	4 Digit
	091h	a	b	c	h	f	g	e	d	2 Digit
										0 Digit



液晶模拟电压多路器



OscOff	LCDM4	LCDM3	LCDM0	VA	VB	VC	VD	RONt
X	X	X	0	0	0	0	0	OFF
1	X	X	X	0	0	0	0	OFF
0	0	0	1	V5/V1	V1/V5	V5/V1	V1/V5	ON
0	0	1	1	V5/V1	V1/V5	V3/V3	V1/V5	ON
0	1	X	1	V5/V1	V2/V4	V4/V2	V1/V5	ON

2个输入端：R23、R13，V1＝Vcc，V5＝Vss
3个输入端：R23、R13、R03，V1＝Vcc
4个输入端：R23、R13、R03、R33

4MUX显示举例

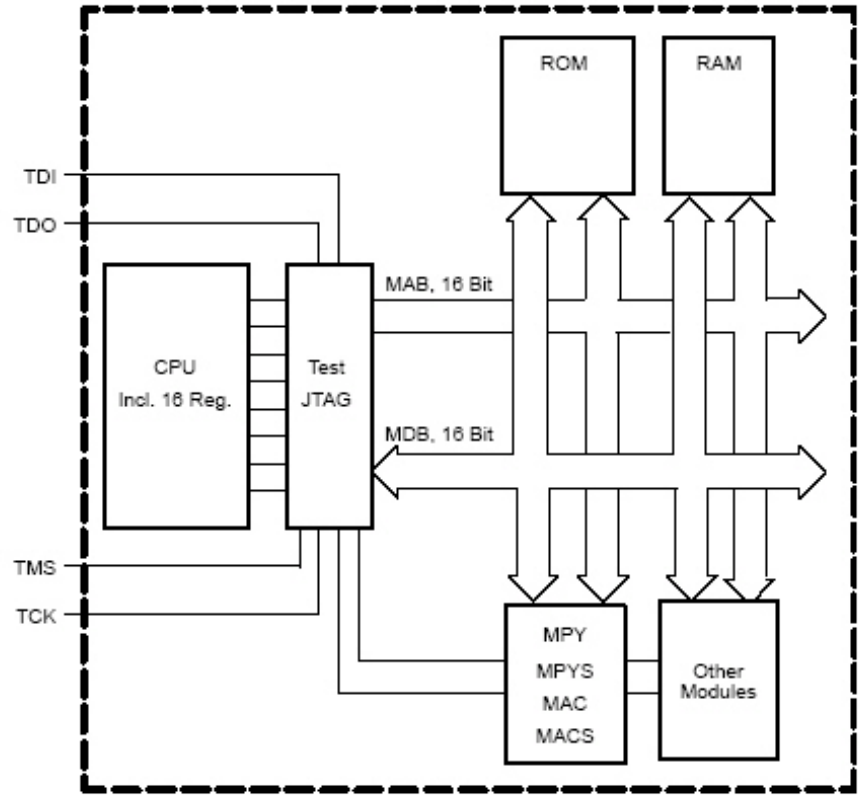
```
#include  <msp430x44x.h>
char digit[10] = {
0xEB,   /* "0" LCD segments a+b+c+d+e+f   ，每个字的8段安排在一个字节中。*/
0X60,   /* "1" */
0XC7,   /* "2" */
0XE5,   /* "3" */
0X6C,   /* "4" */
0XAD,   /* "5" */
0XAF,   /* "6" */
0XE0,   /* "7" */
0XEF,   /* "8" */
0XED    /* "9" */
};

void main(void)
{
   int i;
  WDTCTL = WDTPW + WDTHOLD;  // 停看门狗
  FLL_CTL0 |= XCAP14PF;    // 配置FLL+
  LCDCTL = LCDON + LCD4MUX + LCDP2; // 4Mux, S0-S17
  BTCTL = BTFRFQ1;        // 基本定时器输出f_LCD
  P5SEL = 0xFC;         // 公共极和 Rxx 选择

  for (;;)
  {
  for (i=0; i<7; ++i)           // 显示"6543210"
  LCDMEM[i] = digit[i];
  }
}
```

硬件乘法器通过内部总线与CPU相连

- MSP430可以在不改变CPU结构和指令的情况下增加功能。这种结构特别适用于对运算速度要求很严格的情况。硬件乘法器大大加强了MSP430的功能并提供了软硬件相兼容的范围，提高了数据处理能力。

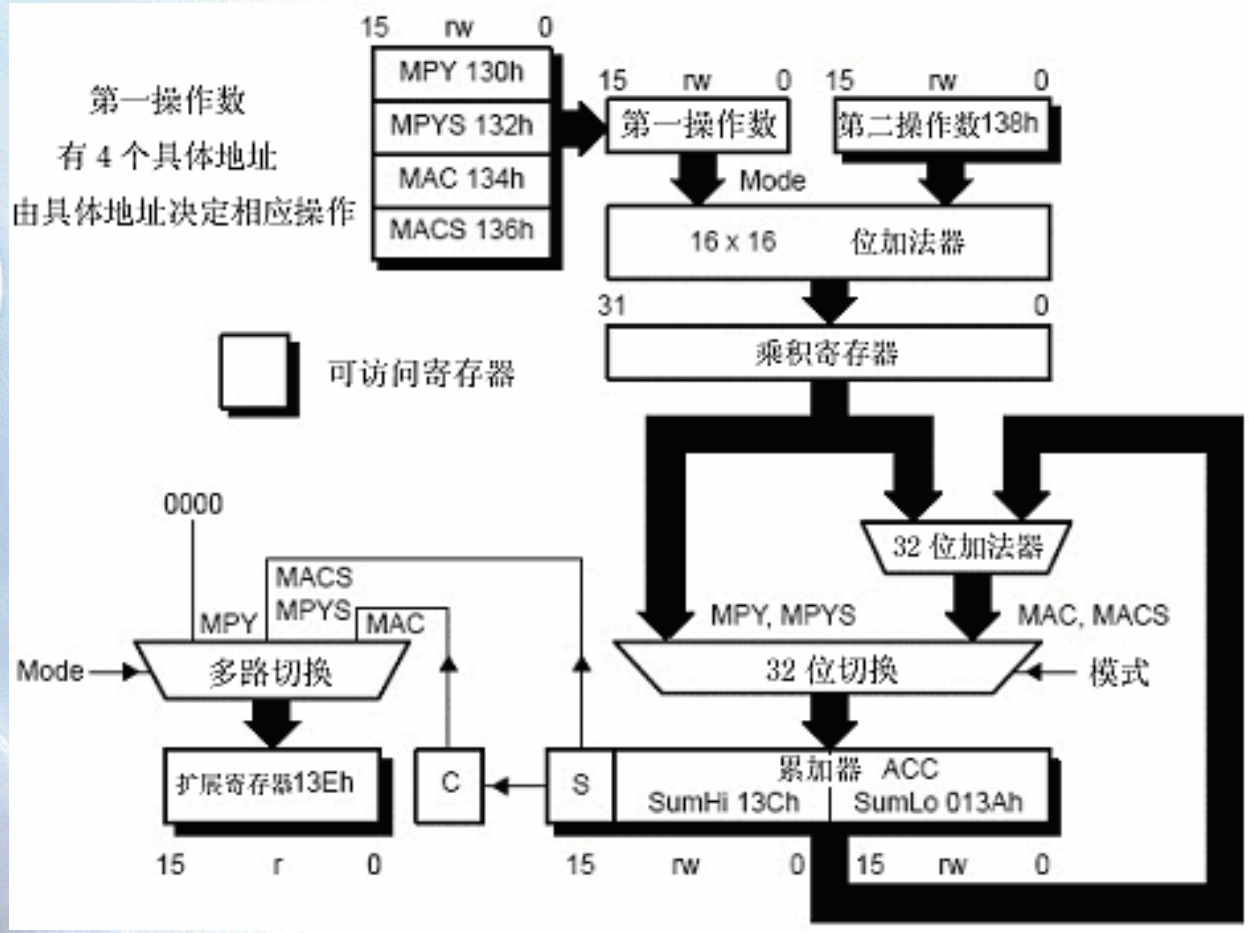


硬件乘法器的结构

- 操作数寄存器：OP1和OP2，第一个操作数可来源于4个寄存器：MPY，MPYS，MAC及MACS，它们能确定乘法的类型。当第二个操作数写入后，相应的乘法操作立即执行，一般需要4个周期数。

- 结果寄存器：结果高字寄存器（RESHI）、结果低字寄存器（RESLO）及结果扩展寄存器（SUMEXT）。寄存器RESHI和RESLO的内容为两个16位数相乘的32位乘积结论。而寄存器SUMEXT的内容由执行的乘法模式及乘积的结果决定。

16×16位硬件乘法器的结构



硬件乘法器寄存器

- MPY　操作数1，指示操作数为无符号数相乘
- MPYS　操作数1，指示操作数为有符号数相乘。
- MAC　操作数1，指示操作数为无符号数累加。
- MACS　操作数1，指示操作数为有符号数累加。
- OP_2　操作数2。
- RESLO　结果低字寄存器。
- RESHI　结果高字寄存器。
- SUMEXT　结果扩展寄存器。

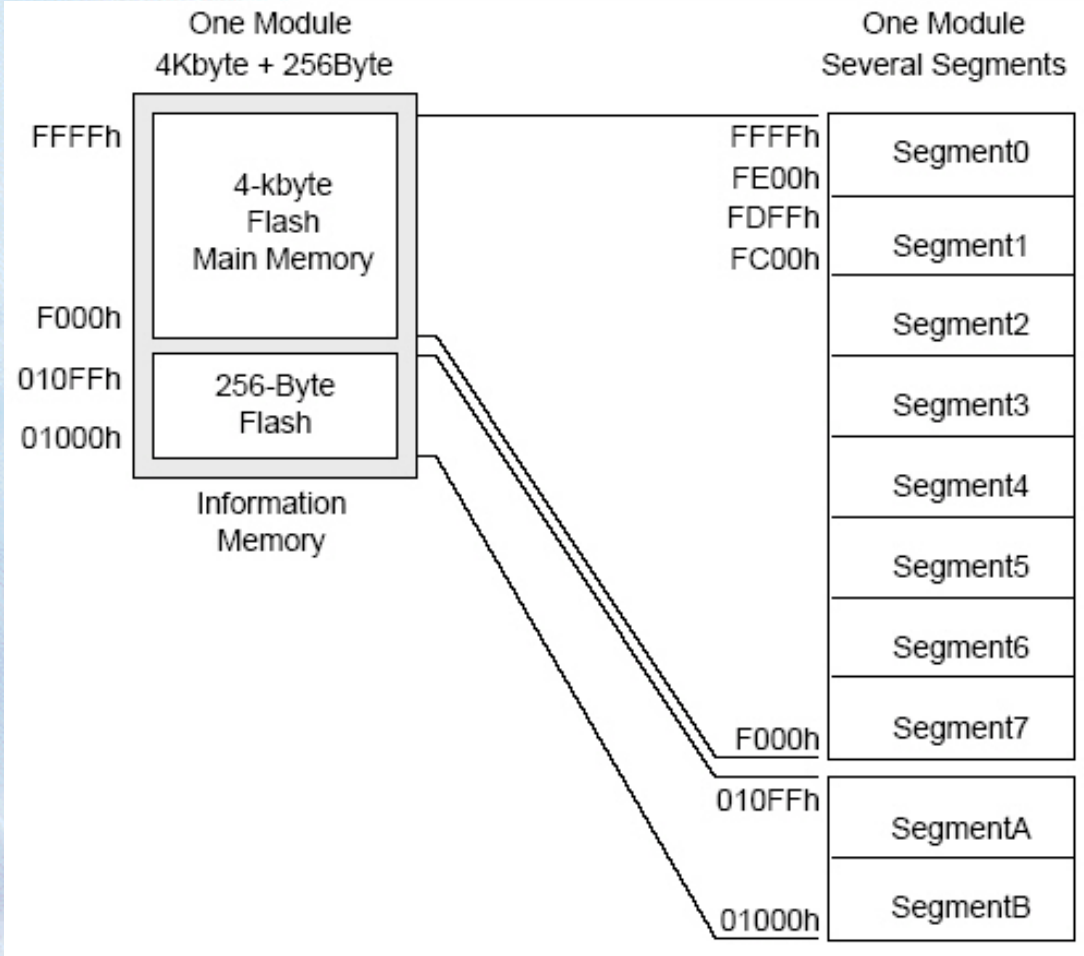
硬件乘法器使用注意事项

- 第二个操作数写入完毕，乘法运算就开始。一般在取出结果之前插入1～2条指令，以保证运算时间的需要。
- 其次，在一个器件中只有一个硬件乘法器，如果遇到多处使用的情况，必须在每一次使用完成后再进行下一次使用
- 结果扩展寄存器（SUMEXT）的内容，与运算类型及运算结果都有关系。
- 不论进行何种运算，只要操作数类型为8×8型，操作过程就要使用寄存器的绝对地址，而不能使用符号形式。寄存器MPY，MPYS，MAC，MACS和OP2的地址依次为：0130h，0132h，0134h，0136h，0138h。

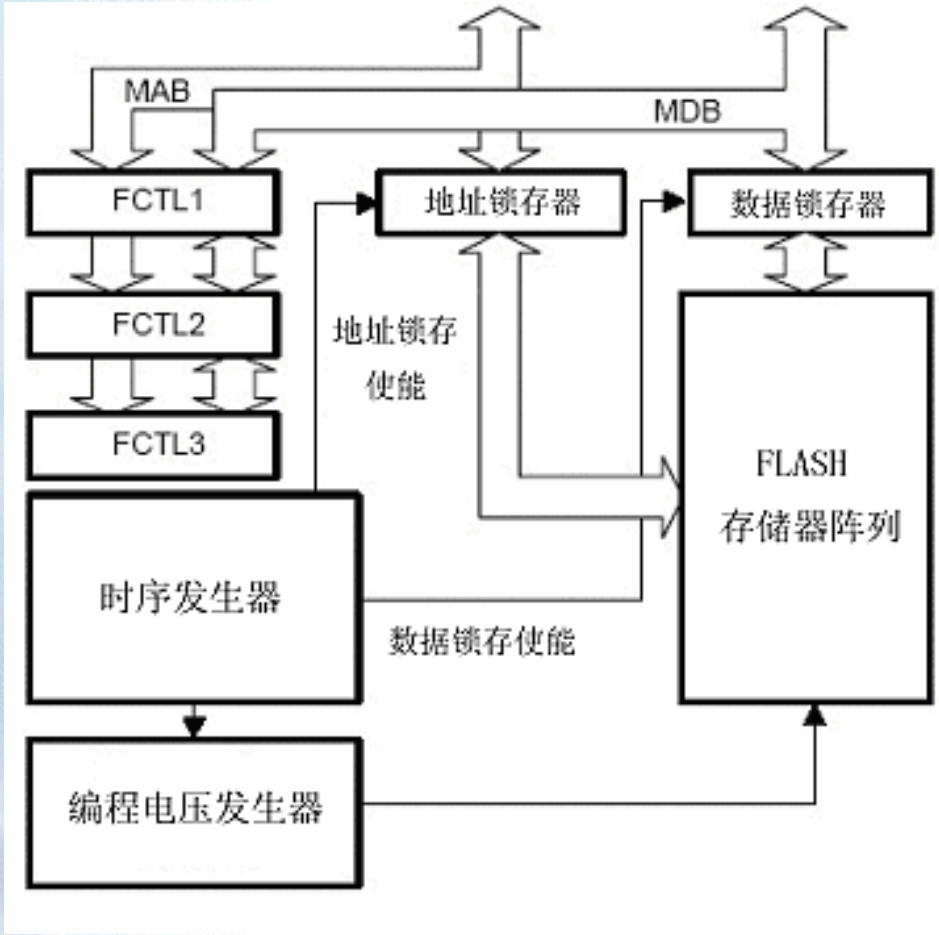
FLASH存储器模块 主要特点

- 编程可以使用位、字节和字操作
- 可以通过JTAG、BSL和ISP进行编程
- 1.8~3.6V工作电压，2.7~3.6V编程电压
- 100K的擦除/编程周期
- 数据保持时间从10年到100年不等
- 可编程次数从100到100,000次
- 60K空间编程时间<5秒
- 保密熔丝烧断后不可恢复，不能再对JTAG进行任何访问。
- FLASH编程/擦除时间由内部硬件控制，无需任何软件干预

部分容量的FLASH段与地址的对应关系



FLASH存储器的结构框图



FLASH存储器的主要功能部件

- 控制寄存器：控制FLASH存储器的檫除与写入
- FLASH存储器阵列：存储体
- 地址数据锁存器：檫除与编程时执行锁存操作
- 编程电压发生器：产生编程电压
- 时序发生器：产生檫除与编程所需所有时序控制信号

擦除操作

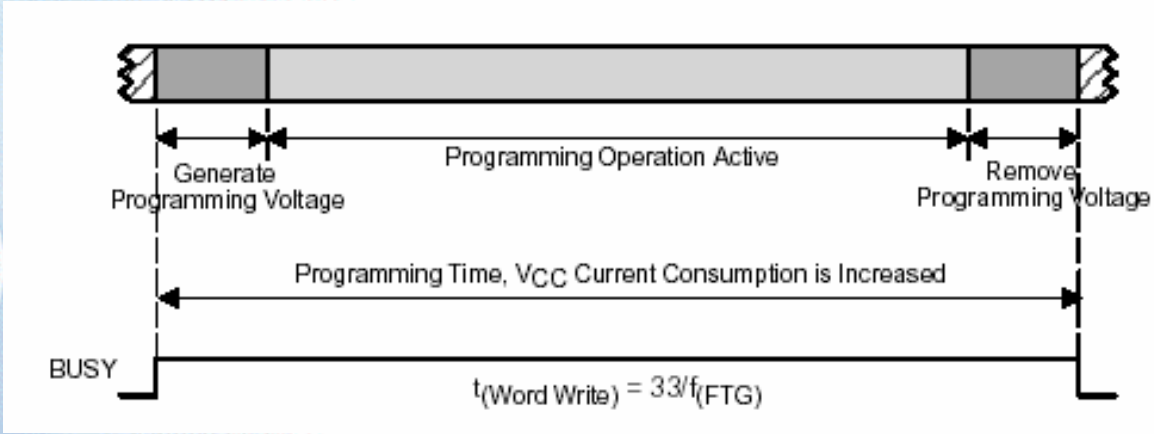
- 选择适当的时钟源和分频因子，为时序发生器提供正确时钟输入
- 如果Lock=1，则将它复位
- 监视BUSY标志位只有当BUSY=0时才可以执行下一步，否则一直监视BUSY。
- 如果擦除一段，则设置ERASE=1。
- 如果擦除多段，则设置MERAS=1
- 如果整个FLASH全擦除，则设置RASE=1同时MERAS=1。
- 对擦除的地址范围内任意位置作一次空写入，用以启动擦除操作。如果空写的地址在不能执行擦除操作的段地址范围内，则写入操作不起作用

FLASH编程操作

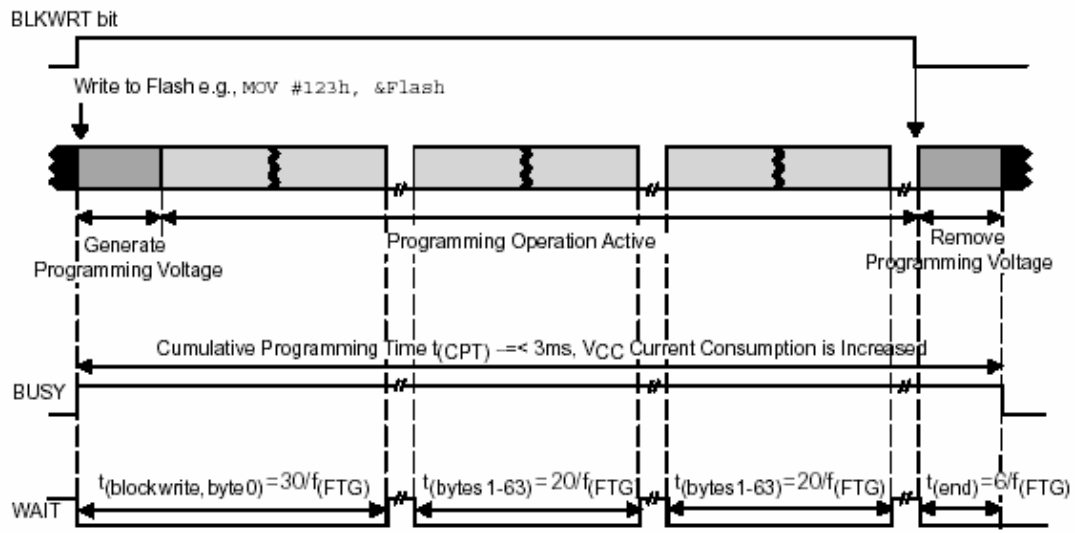
- 选择适当的时钟源以及合适的分频因子
- 如果Lock=1，将它复位
- 监视BUSY位，直到BUSY=0是才可进入下一步
- 如果写入单字或单字节，则将设置WRT=1
- 如果是块写或多字、多字节顺序写入，则将设置WRT=1，BLKWRT=1
- 将数据写入选定地址时启动时序发生器，在时序发生器的控制下完成整个过程

单字或单字节写入与块写入的控制时序

- 单字或单字节写入周期



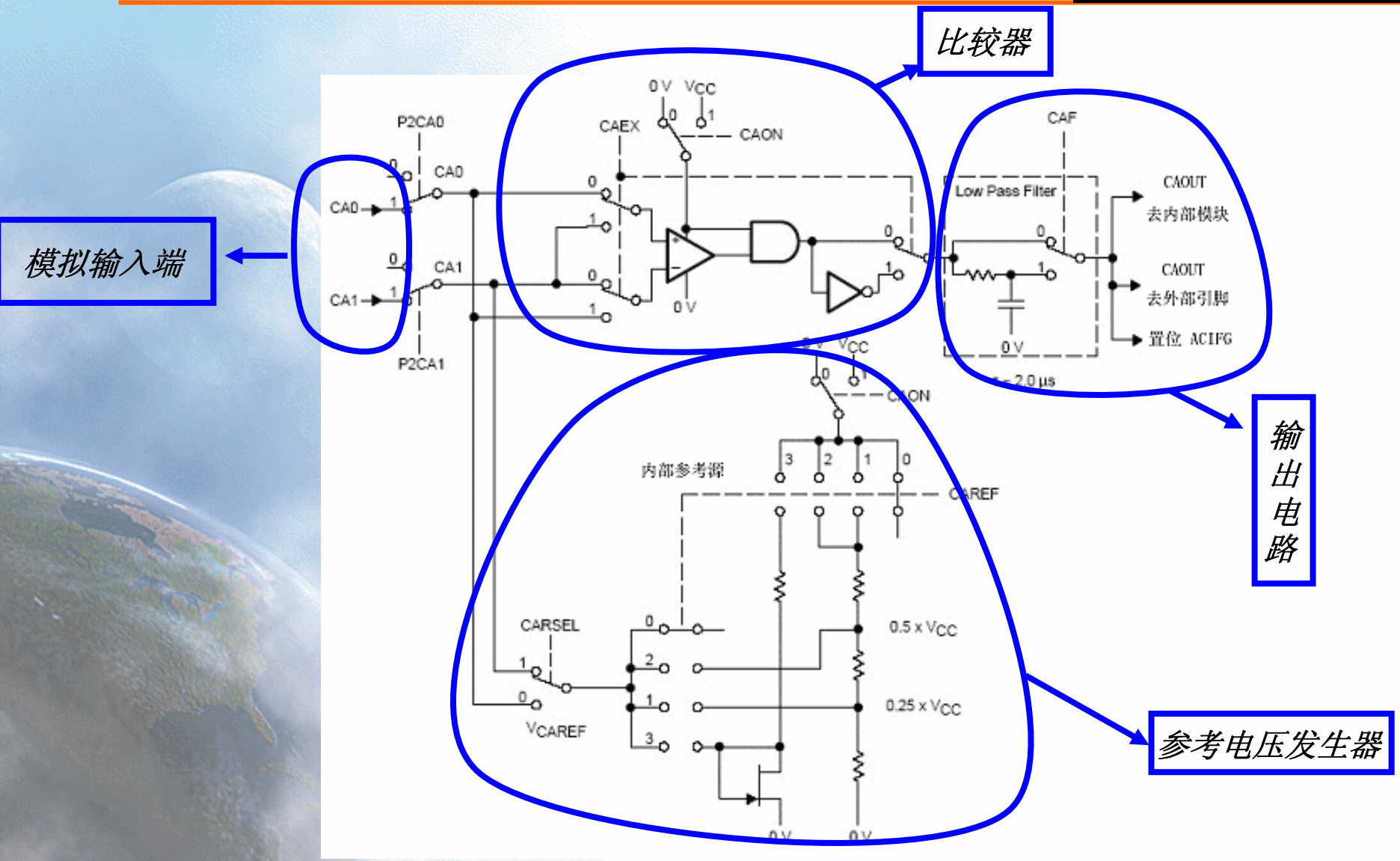
- 块写入周期



FLASH错误操作的处理

- 如果写入高字节口令码错误，则引发PUC信号：小心操作可以避免；
- 在对FLASH操作期间读FLASH内容，会引发ACCVFIG状态位的设置：小心操作可以避免
- 因为在对FLASH操作期间，需要较长的时间，如果这时看门狗定时器的数据将近尾声，则看门狗定时器溢出：建议用户程序在进行FLASH操作之前先停掉看门狗定时器，等操作结束之后再打开看门狗
- 所有的FLASH类型的MSP430器件的0段都包含有中断向量等重要的程序代码，如果对其进行擦除操作，将会引起严重的后果：建议用户程序在进行FLASH操作之前，先将该段的重要数据（或程序代码）保存到RAM中或写入到其他暂时未用的段中，等待该段操作完毕再还原那些数据（或程序代码）；同时一定不要使正在执行的程序处在正要被擦除的段中；也不要在FLASH操作期间允许中断的发生。

比较器A的结构


比较器
模拟输入端
输出电路
参考电压发生器
P2CA0
CA0
CA1
P2CA1
CAEX
CAON
0 V
VCC
CAF
Low Pass Filter
CAOUT
去内部模块
CAOUT
去外部引脚
置位 ACIFG
~2.0 µs
内部参考源
CAREF
CARSEL
VCAREF
0.5 x VCC
0.25 x VCC

比较器A的功能

- 比较器A的主要功能是指出两个输入电压CA0和CA1的大小关系，然后设置输出信号CAOUT的值。如果CA0>CA1 则：CAOUT＝1，否则CAOUT＝0。

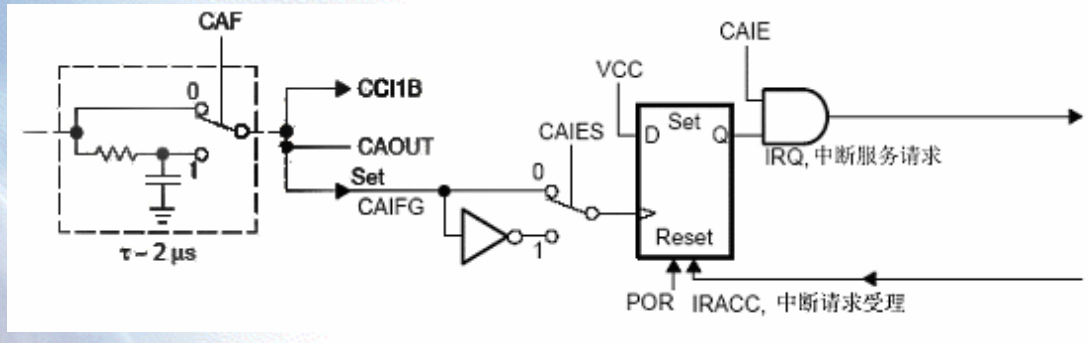
- 参与比较的两个电压CA0和CA1可以是外部或者内部基准电压。任何组合都是可能的。

两个外部输入比较
每个外部输入与0.5V_{CC}或0.25V_{CC}比较
每个外部输入与内部基准电压比较

比较器A的中断

- 比较器A响应中断的条件为

有中断源：比较器模块有比较结果输出。
设置中断标志：CAIES选择比较器输出的上升沿或下降沿使中断标志CAIFG置位。
中断允许：比较器A中断允许（CAIE置位）、系统总中断允许（GIE置位）



中断响应后，因为比较器A具有独立中断向量，是单源中断，硬件会自动清除中断标志位CAIFG

比较器A应用举例

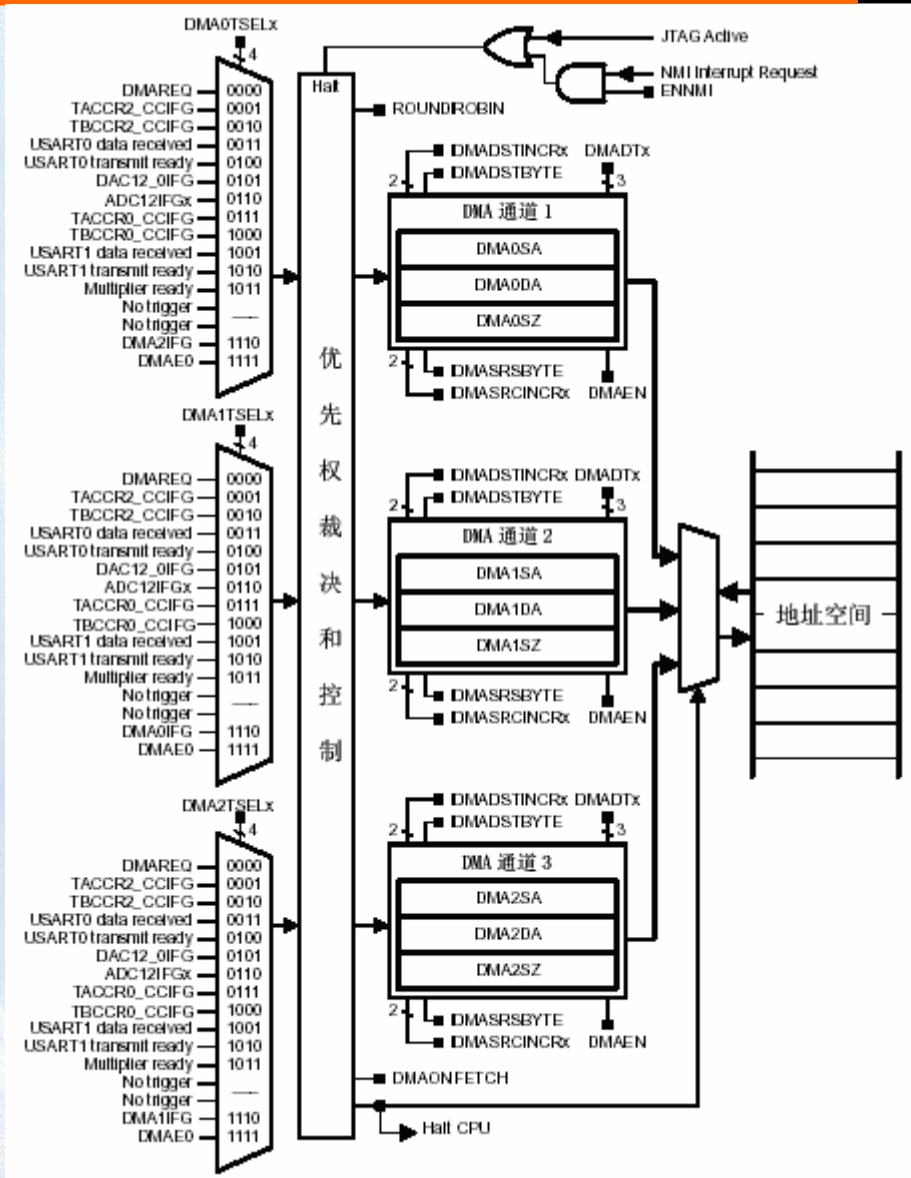
- 电压检测：P2.3输入的未知电压接到比较器A正端，片内参考电压0.25Vcc接到比较器A负端，如果未知电压大于0.25Vcc，P1.0置位，否则P1.0复位。

```
#include  "msp430x11x1.h"
void main (void)
{
  WDTCTL = WDTPW + WDTHOLD; // 停止看门狗
P1DIR |= 0x01;                          // P1.0 输出
  CACTL1 = CARSEL + CAREF0 + CAON; // 0.25 Vcc = -comp
  CACTL2 = P2CA0;                       // P2.3 = +comp
  while (1)
  {
    if ((CAOUT & CACTL2))
    P1OUT |= 0x01;                      // CAOUT =1, 置位 P1.0
    else P1OUT &= ~0x01;                // 否则复位
  }
}
```

MSP430 DMA控制器的特性

- 数据传送不需要CPU介入，完全由DMA控制器自行管理。
- 在整个地址空间范围内传输数据，块方式传输可达65536字节
- 能够提高片内外设数据吞吐能力，实现高速传输，每个字或者字节的传输仅需要2个MCLK
- 减少系统功耗，即使在片内外设进行数据输入或输出时，CPU也可以处于超低功耗模式而不需唤醒
- 字节和字数据可以混合传送：DMA传输可以是字节到字节、字到字、字节到字或者字到字节。当字到字节传输时，只有字中较低字节能够传输，当从字节到字传输时，传输到字的低字节，高字节被自动清零
- 四种传输寻址模式：固定地址到固定地址、固定地址到块地址、块地址到固定地址以及块地址到块地址
- 触发方式灵活：边沿或者电平触发。
- 单个、块或突发块传输模式：每次触发DMA操作，可以根据需要传输不同规模的数据

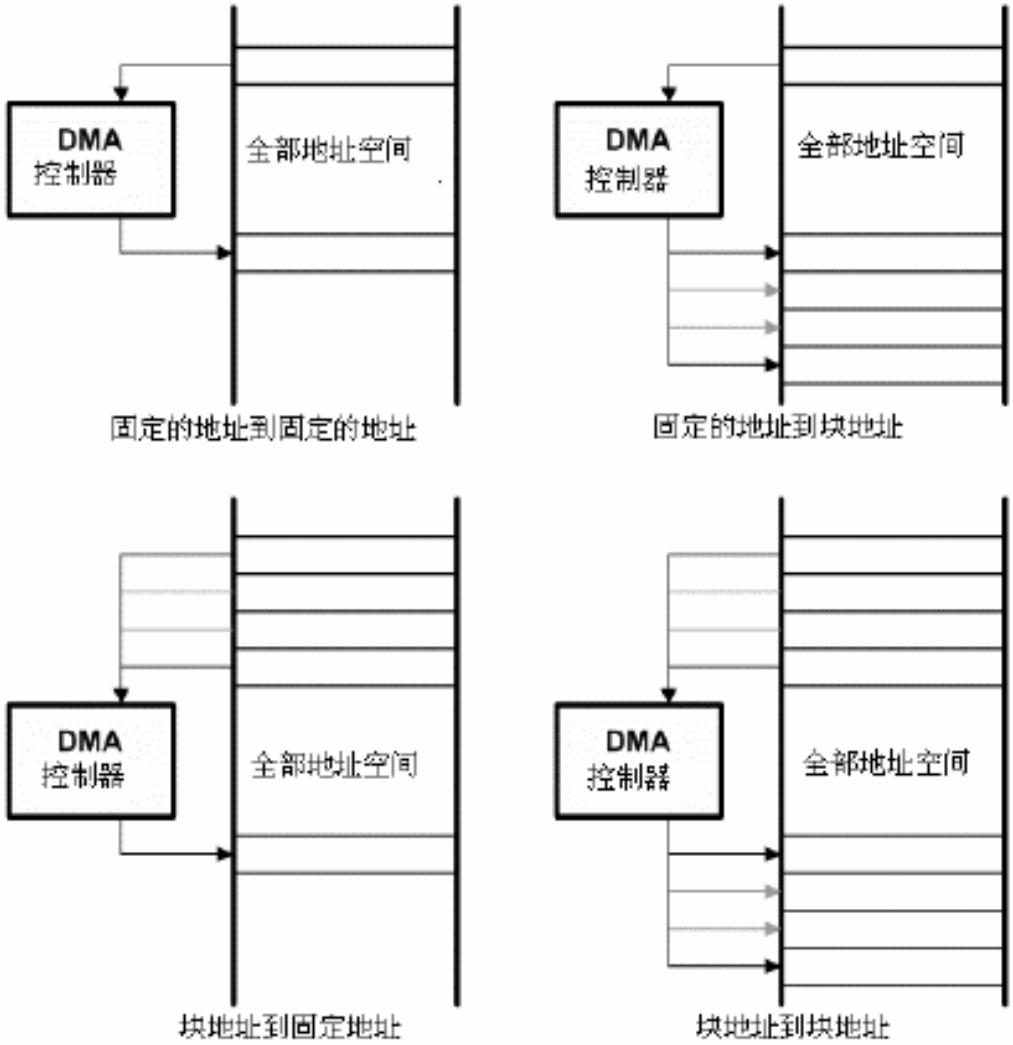
MSP430 DMA控制器的结构



DMA控制器的功能模块：

- 3个独立的传输通道：通道0、通道1和通道2。每个通道都有源地址寄存器、目的地址寄存器、传送数据长度寄存器和控制寄存器。每个通道的触发请求可以分别允许和禁止
- 可配置的通道优先权：优先权裁决模块，传输通道的优先级可以调整，对同时有触发请求的通道进行优先级裁决，确定哪个通道的优先级最高。MSP430的DMA控制器可以采用固定优先级，还可以采用循环优先级。
- 程序命令控制模块，每个DMA通道开始传输之前，CPU要编程给定相关的命令和模式控制，以决定DMA通道传输的类型
- 可配置的传送触发器：触发源选择模块，DMAREQ（软件触发）、Timer_A CCR2输出、Timer_B CCR2输出、I^2C 数据接收准备好、I^2C 数据发送准备好、USART接收发送数据、DAC12模块DAC12IFG、ADC12模块 的ADC12IFGx、DMAxIFG、DMAE0 外部触发源。并且还具有触发源扩充能力

DMA控制器的寻址


DMA
控制器
全部地址空间
固定的地址到固定的地址
DMA
控制器
全部地址空间
固定的地址到块地址
DMA
控制器
全部地址空间
块地址到固定地址
DMA
控制器
全部地址空间
块地址到块地址

DMA传输模式

- 单字或者单字节传输
- 块传输
- 突发块传输
- 重复单字或者单字节传输
- 重复块传输
- 重复突发块传输

DMA应用举例

- 利用DMA控制器将数据块由RAM的220h-240h单元传输到240h-260h单元

```
#include  <msp430x16x.h>
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;    // 停看门狗
  P1DIR |= 0x01;                          // P1.0输出
  DMA0SA = 0x0220;                      // 起始地址
  DMA0DA = 0x0240;                   // 目的地址
  DMA0SZ = 0x010;                    // 传输规模
  DMA0CTL = DMADT_5 + DMASRCINCR_3 + DMADSTINCR_3 + DMAEN; // 重复块传输，起始地址、
            目的地址增量
  for (;;)                                // 重复块传输
  {
    P1OUT |= 0x01;                        // 置位P1.0
    DMA0CTL |= DMAREQ;                 // 触发块传输
    P1OUT &= ~0x01;                     // 清除P1.0
  }
}
```

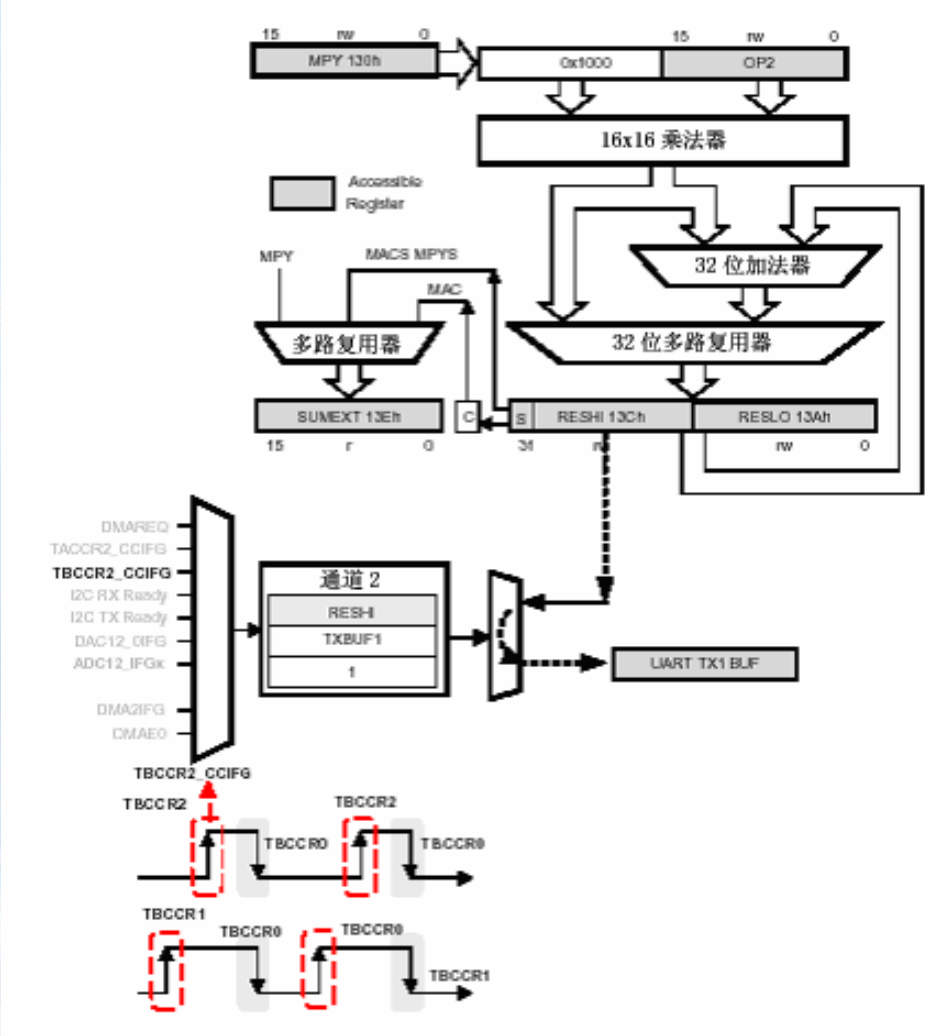
DMA应用举例

- 通过TACCR2触发DMA控制器给端口P1输出一个字节串

```
#include  <msp430x16x.h>
const unsigned char testconst[] = { 0x00, 0x03, 0x02, 0x03, 0x00, 0x01 };
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;
P1DIR |= 0x03;              // P1.0/1.1 输出
  DMACTL0 = DMA0TSEL_1;    // CCR2IFG 触发
  DMA0SA = (unsigned int)testconst;  // 起始地址
  DMA0DA = P1OUT_;          // 目的地址
  DMA0SZ = sizeof testconst;     // 传输规模
  DMA0CTL = DMADT_4 + DMASRCINCR_3 + DMASBDB + DMAEN; // 重复单字节传输，起始地址增量，DMA使能
  TACTL = TASSEL_2 + MC_2;     // SMCLK, 连续计数模式
   _BIS_SR(LPM0_bits);         //进入 LPM0
}
```

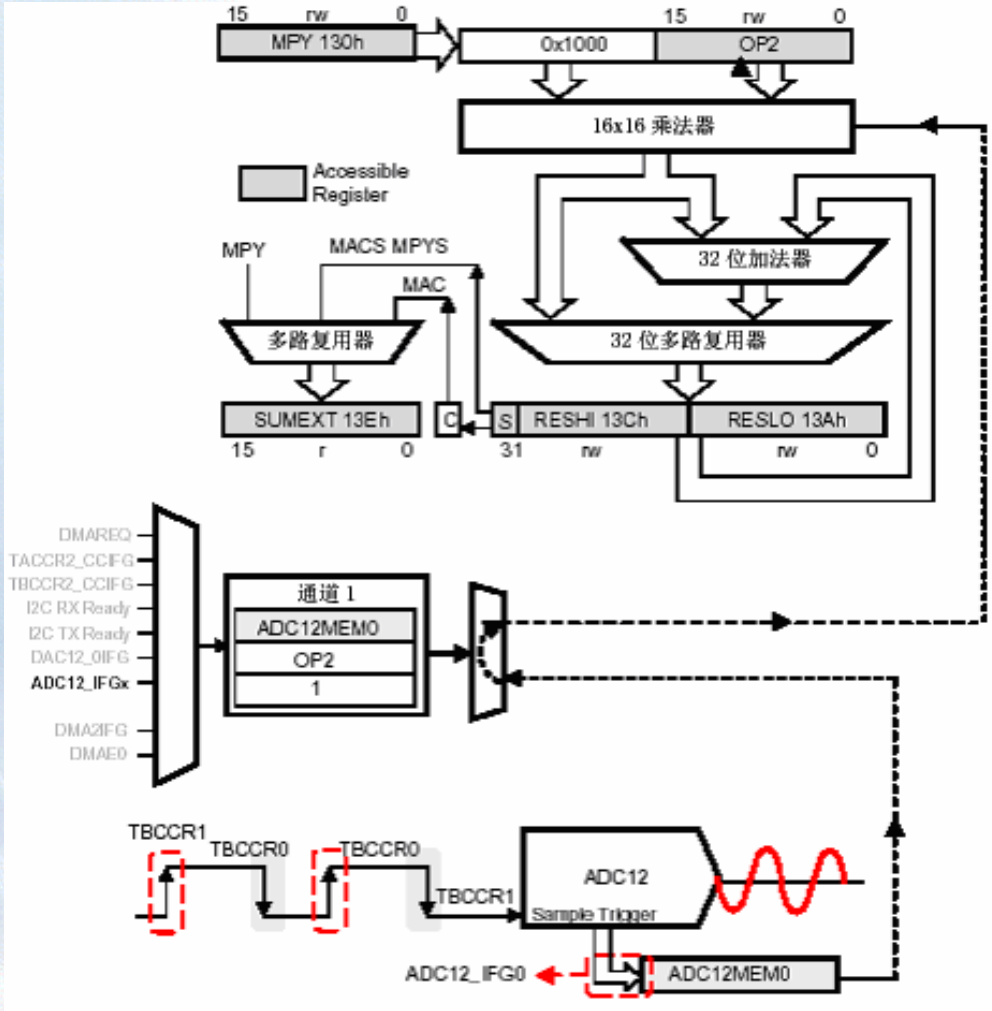
DMA应用举例

- DMA传输使硬件乘法器的运算结果通过串口输出。

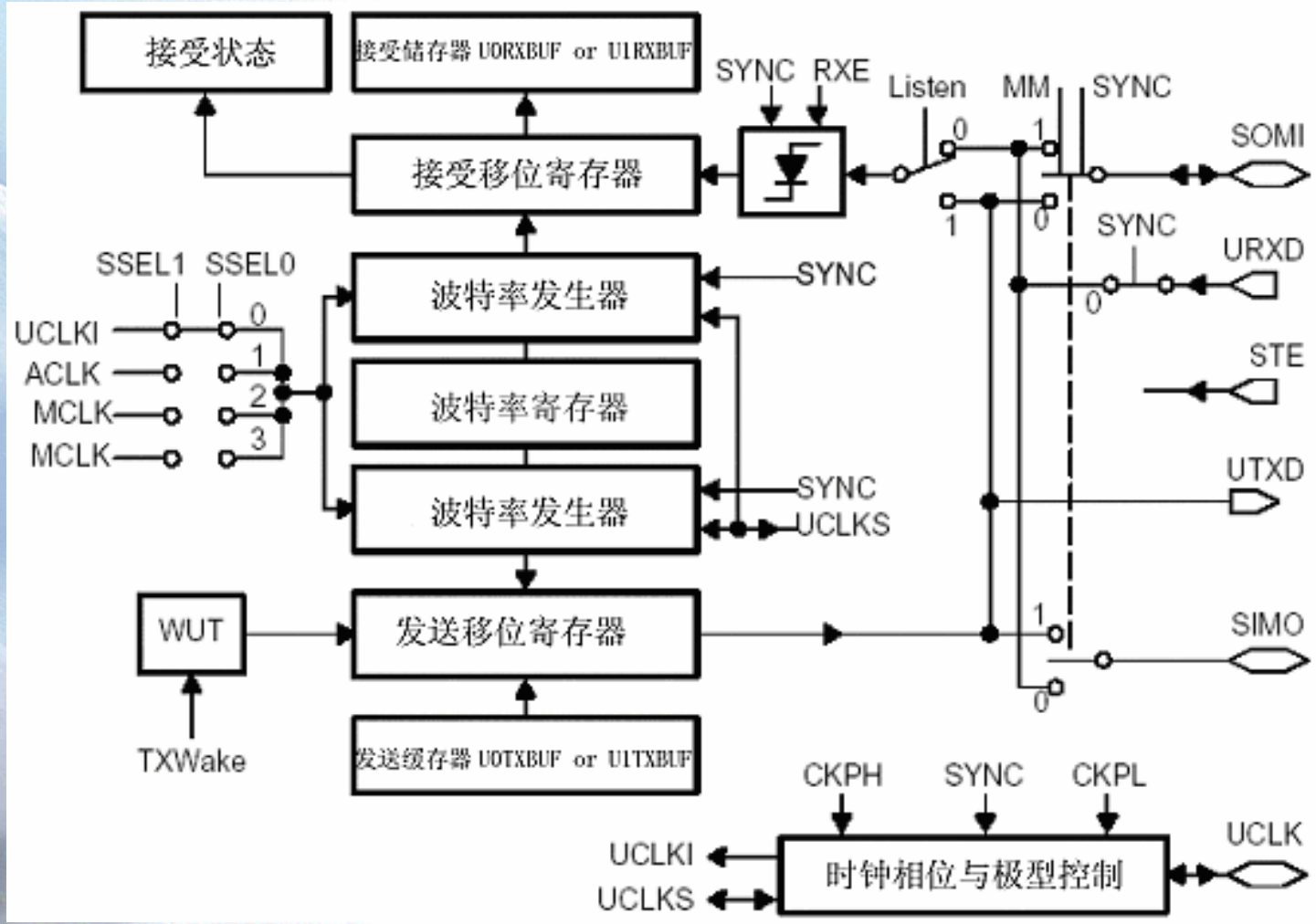


DMA应用举例

- ADC12转换的结果通过 DMA 控制器传送至高速的运算部件硬件乘法器MPY



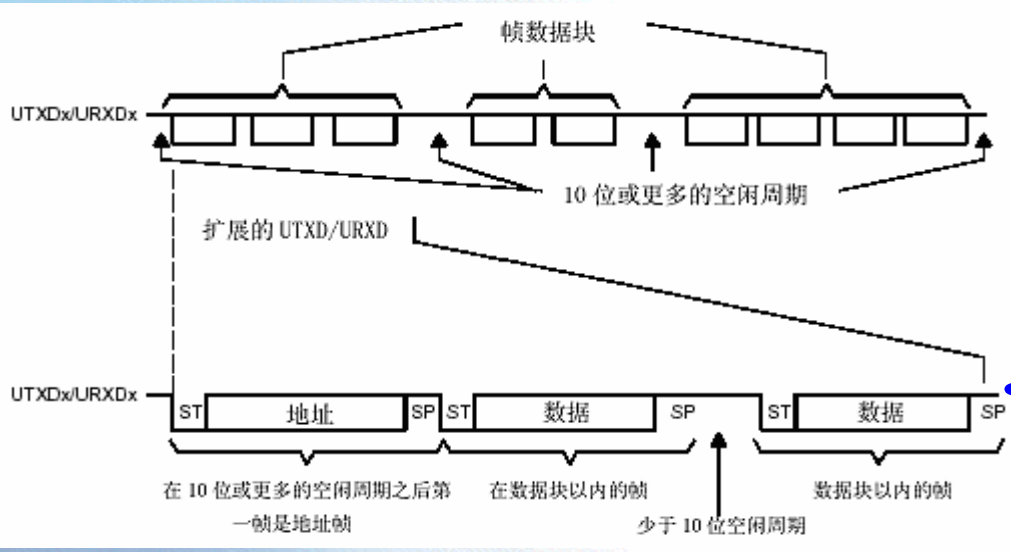
USART模块结构


接受状态
接受储存器 U0RXBUF or U1RXBUF
SYNC
RXE
Listen
MM
SYNC
SOMI
接受移位寄存器
SSEL1
SSEL0
SYNC
URXD
UCLKI
ACLK
MCLK
MCLK
波特率发生器
波特率寄存器
波特率发生器
SYNC
UCLKS
STE
UTXD
WUT
发送移位寄存器
SIMO
TXWake
发送缓存器 U0TXBUF or U1TXBUF
CKPH
SYNC
CKPL
UCLK
UCLKI
UCLKS
时钟相位与极型控制

串行异步通信特点

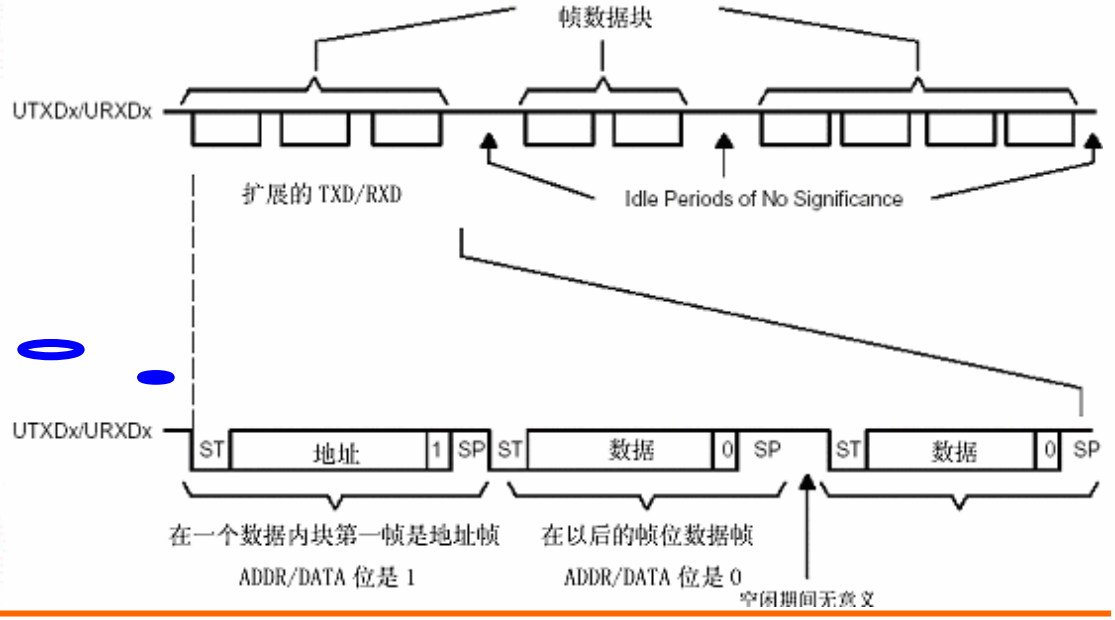
- 异步模式，包括线路空闲/地址位通信协议
- 两个独立移位寄存器：输入移位寄存器和输出移位寄存器
- 传输7位或8位数据，可采用奇校验或偶校验或者无校验
- 从最低位开始的数据发送和接收
- 可编程实现分频因子为整数或小数的波特率
- 独立的发送和接收中断
- 通过有效的起始位检测将MSP430从低功耗唤醒
- 状态标志检测错误或者地址位

异步多机通信模式



线路空闲多机模式

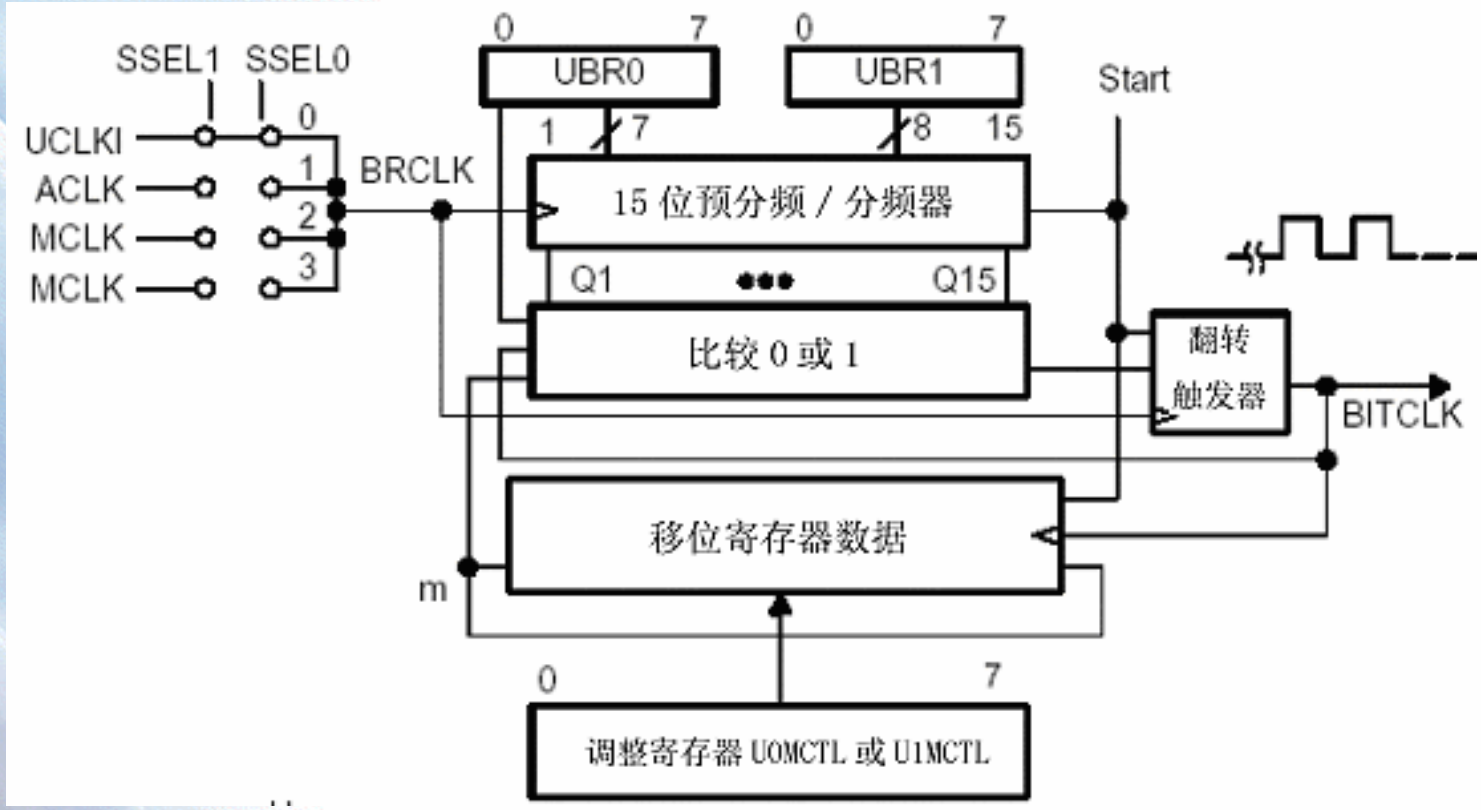
地址位多机模式



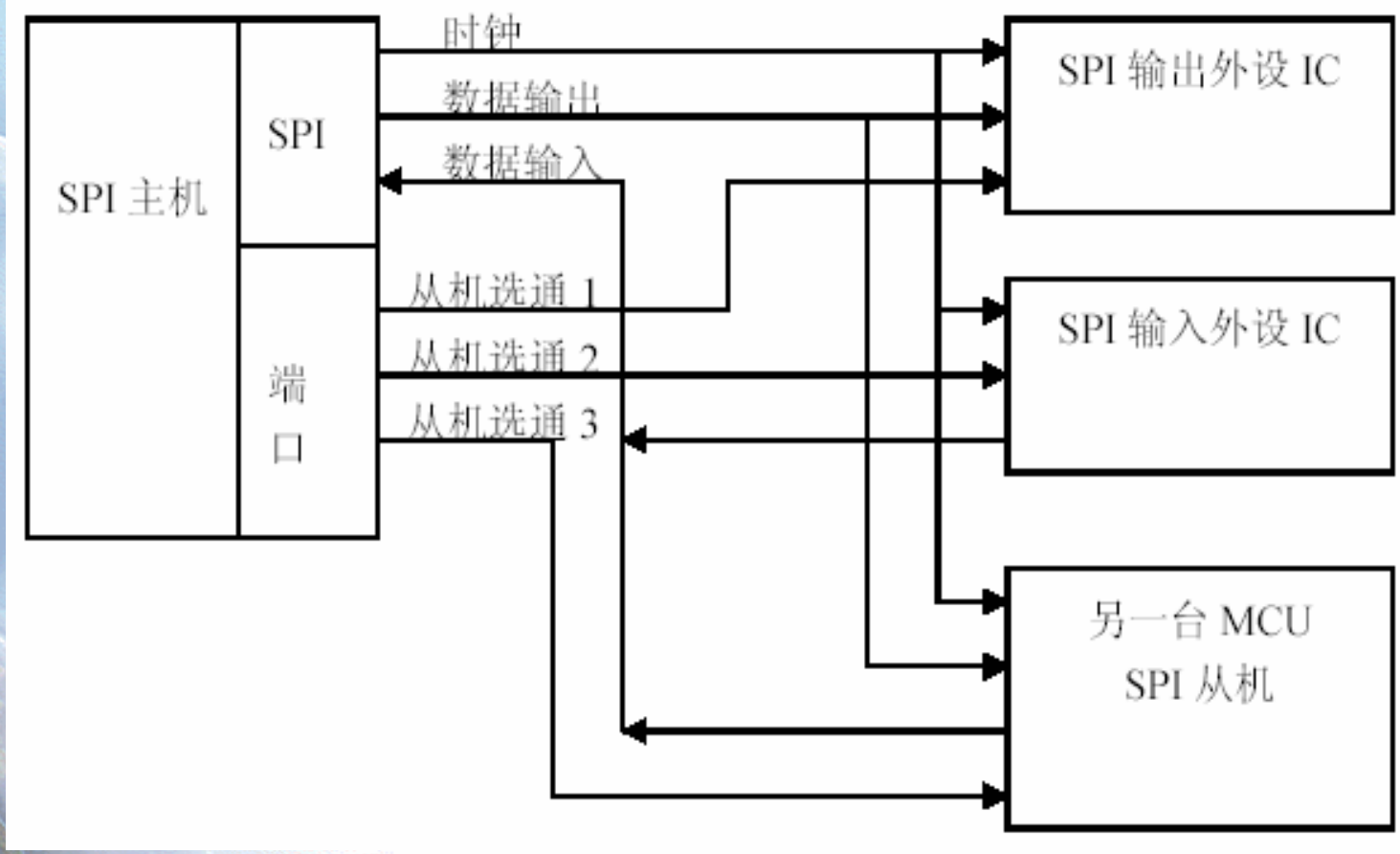
串行操作自动错误检测

- FE 标志帧错误：当一个接收字符的停止位为0并被装入接收缓存，接收的为一个错误的帧，那么帧错标志被设置成1，即使在多停止位模式时也只检测第一个停止位。同样，丢失停止位意味着从起始位开始的同步特性被丧失，也是一个错误帧。在同步的4线模式时，因总线冲突使有效主机停止，并在STE引脚信号出现下降沿时使FE位设置为1
- PE 奇偶校验错误：当接收字符中1的个数与它的校验位不相符，并被装入接收缓存时，发生校验错，设置PE为1
- OE 溢出错误标志：当一个字符写入接收缓存URXBUF时，前一个字符还没有被读出，这时前一个字符因被覆盖而丢失，发生溢出（同步与异步情况相同）
- BRK 打断检测标志：当发生一次打断同时URXEIE置位时，该位被设置为1，表示接收过程被打断过。RXD线路从丢失的第一个停止位开始连续出现至少10位低电平被识别为打断

波特率的产生


SSEL1
SSEL0
UCLKI
ACLK
MCLK
MCLK
0
1
2
3
BRCLK
0
7
UBR0
0
7
UBR1
Start
1
7
8
15
15 位预分频 / 分频器
Q1
Q15
比较 0 或 1
翻转
触发器
BITCLK
移位寄存器数据
m
0
7
调整寄存器 U0MCTL 或 U1MCTL

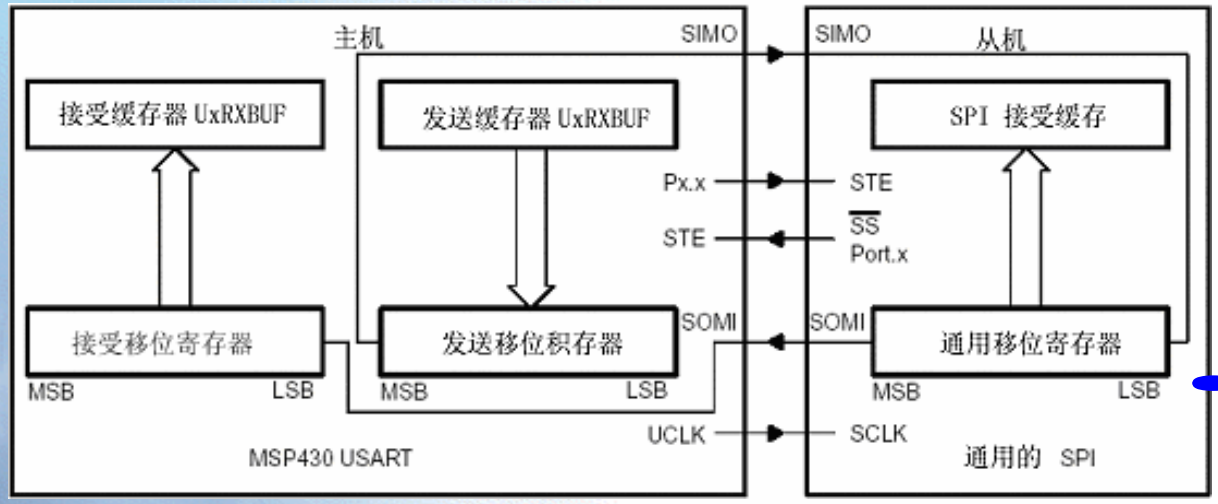
SPI典型结构


SPI 主机
SPI
端口
时钟
数据输出
数据输入
从机选通 1
从机选通 2
从机选通 3
SPI 输出外设 IC
SPI 输入外设 IC
另一台 MCU
SPI 从机

MSP430的同步通信模块特点

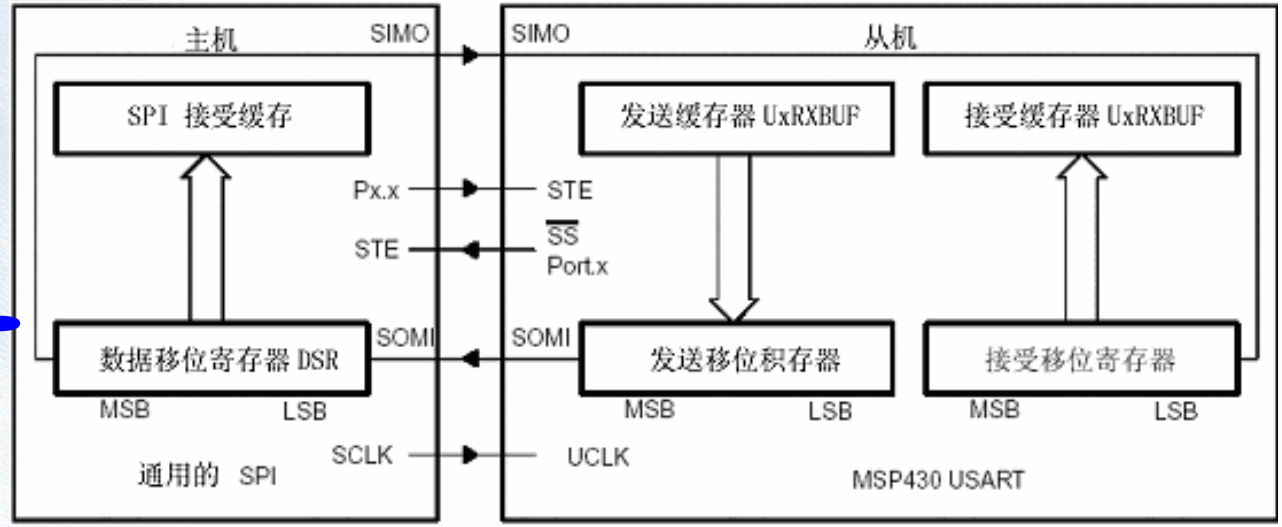
- 支持3线或4线SPI操作
- 支持主机模式与从机模式
- 接收和发送有单独的移位寄存器
- 接收和发送有独立的缓冲器
- 接收和发送有独立的中断能力
- 时钟的极性和相位可编程
- 主模式的时钟频率可编程
- 7位或8位字符长度

SPI的主机模式和从机模式

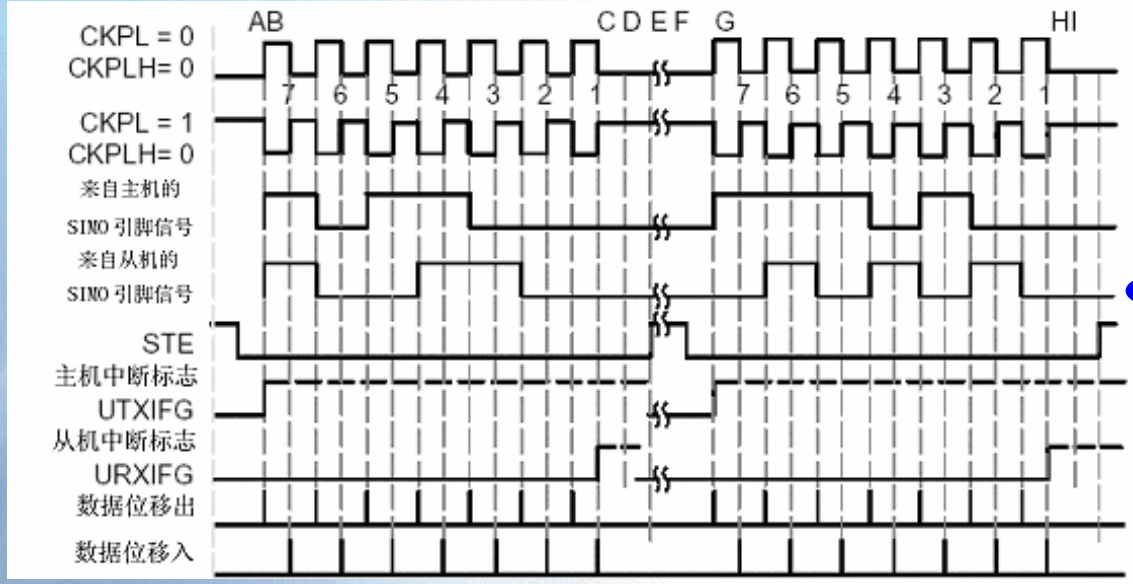


USART为主机模式

USART为从机模式

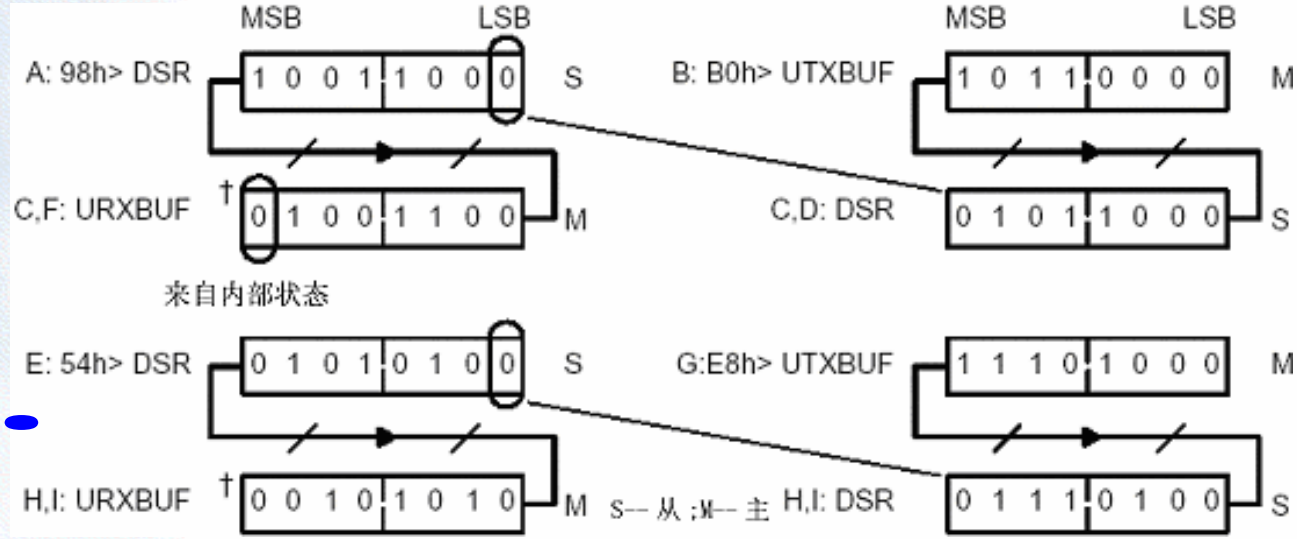


同步通信举例



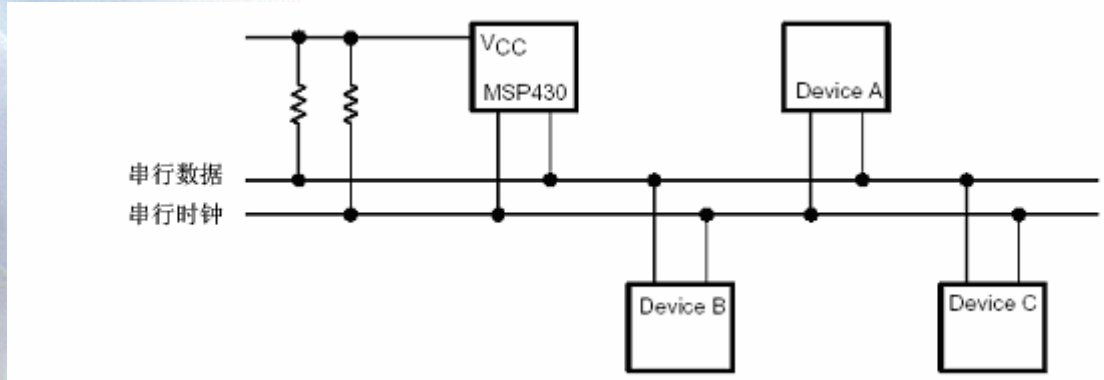
同步串行数据通信

数据传输循环

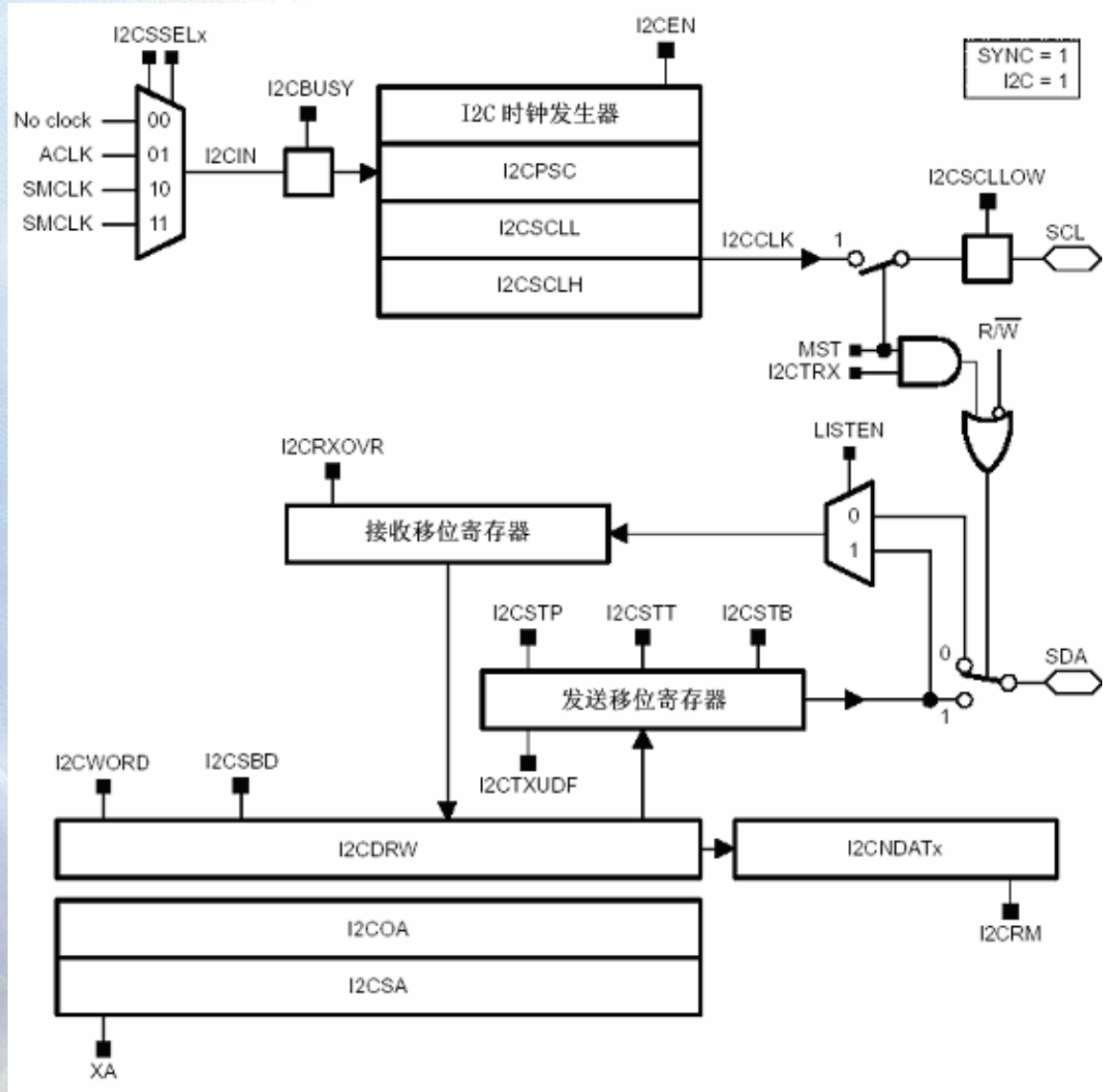


I2C概述

- 在现代电子系统中，有为数众多的IC需要进行相互之间以及与外界的通信。为了提高硬件效率和简化电路设计而广泛使用Inter-IC。
- Inter-IC（I2C）总线是一种用于内部IC控制的具有多端控制能力的双线双向串行数据总线系统。能够用于替代标准的并行总线，连接各种集成电路和功能模块。I2C器件的应用能够减少电路间连线，减小电路板尺寸，降低硬件成本，并提高了系统可靠性
- MSP430和有关设备互连



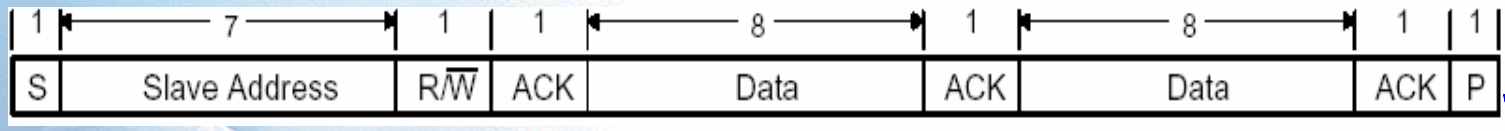
MSP430 I2C模块结构



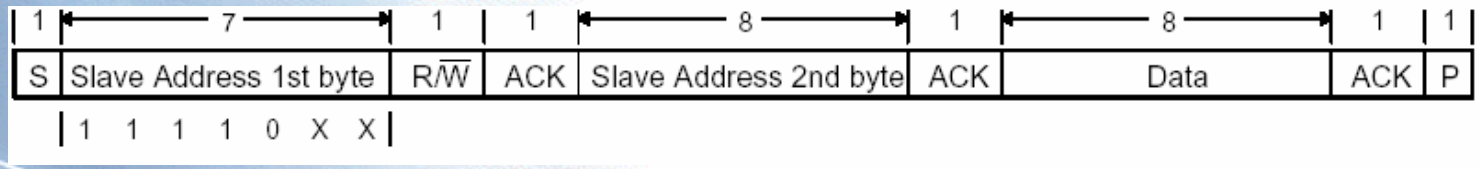
MSP430系列I2C模块的主要特征

- 符合I2C规范V2.1
- 读写采取先进先出缓冲结构
- 可编程时钟发生器
- 16位数据访问可达到总线的最大吞吐率
- 自动数据字节计算
- 支持低功耗模式
- 从接收根据检测到开始信号自动将MSP430从LPMx模式唤醒
- 两个DMA触发源
- 中断功能丰富
- 只能用USART0实现I2C操作

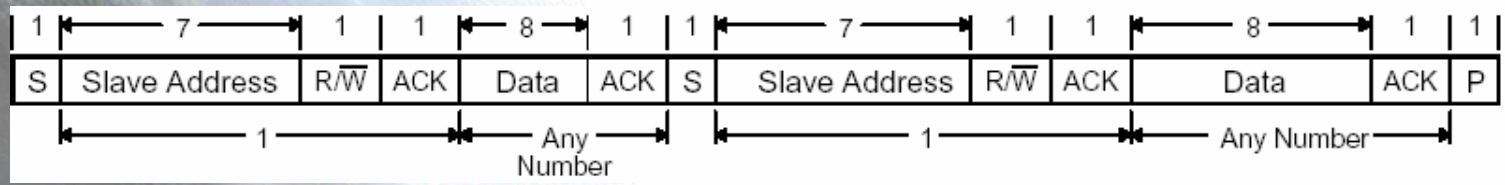
I2C的寻址模式



7位寻址模式



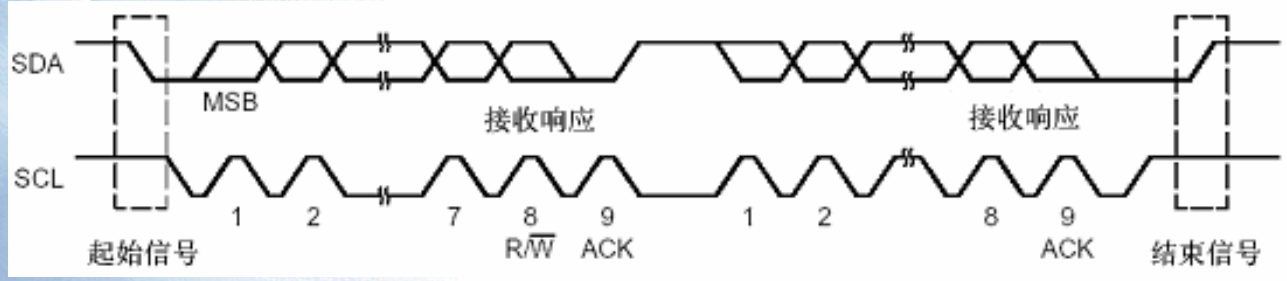
10位寻址模式



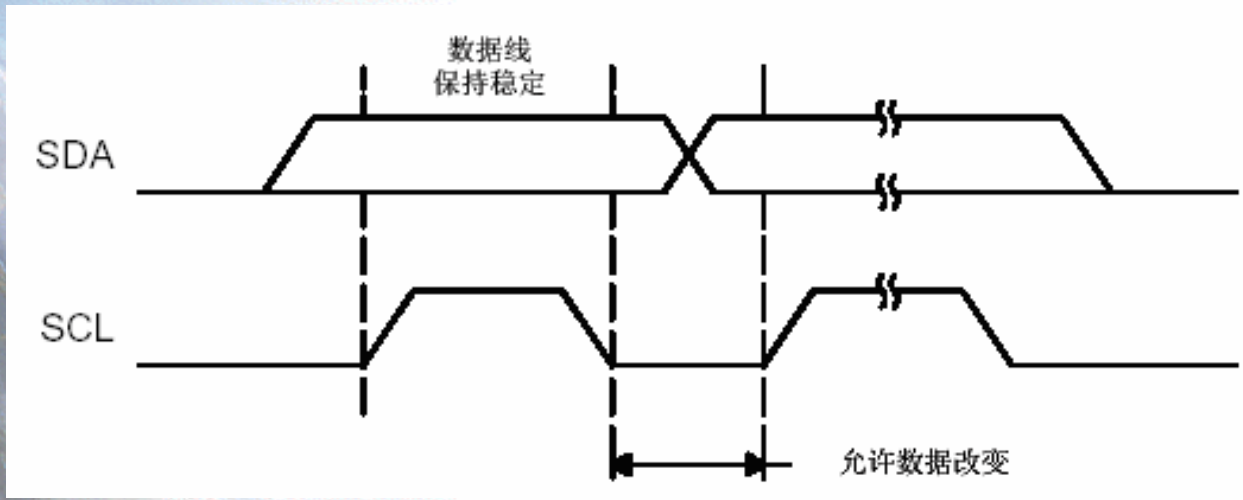
重复产生起始

电气特性

- 起始位：SCL=1时，SDA上有下降沿
- 停止位：SCL=1时，SDA上有上升沿



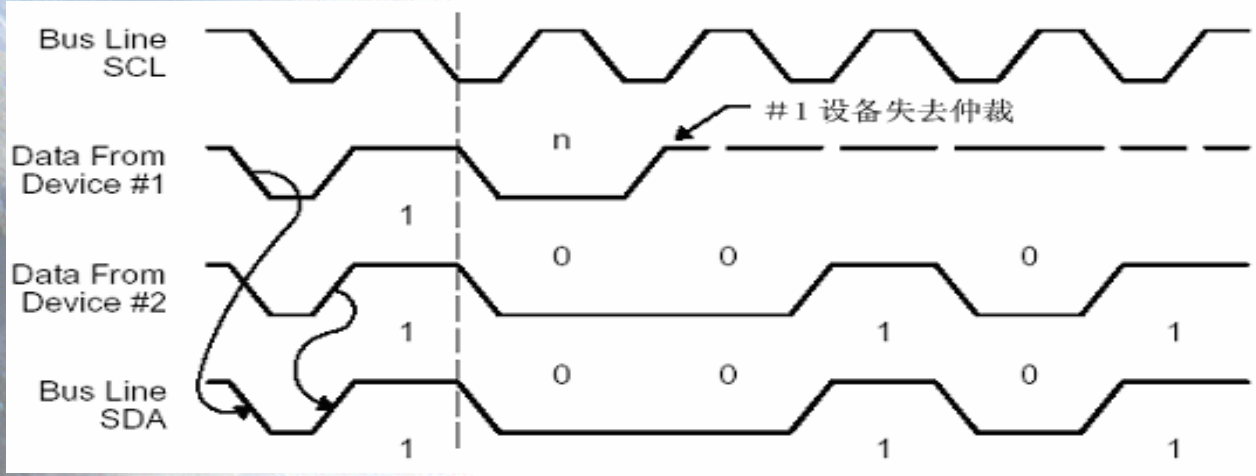
I2C模块数据传输



I2C总线位传输

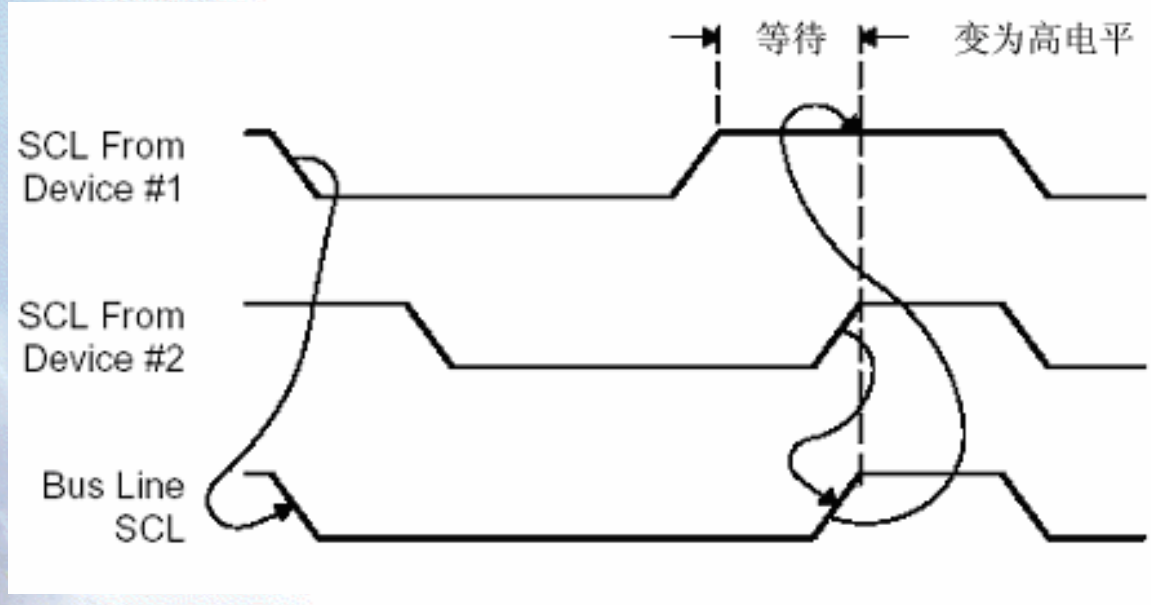
总线仲裁

- 当两个设备同时发出起始位进行数据传输时，相互竞争的设备使它们的时钟保持同步，正常发送数据。没有检测到冲突之前，每个设备都认为只有自己在使用总线
- 仲裁过程中使用的数据就是相互竞争的设备发送到SDA线上的数据。第一个检测到自己发送的数据和总线上数据不匹配的设备就失去仲裁能力。如果两个或更多的设备发送的第一个字节的内容相同，那么仲裁就发生在随后传输中。也许直到相互竞争的设备已经传输了许多字节后，仲裁才会完成。



时钟同步

- 仲裁过程中，要对来自不同主设备的时钟进行同步处理。在SCL上第一个产生低电平的主设备强制其他主设备也发送低电平，SCL保持为低，如果某些主设备已经结束低电平状态，就开始等待，直到所有的主设备都结束低电平时钟。



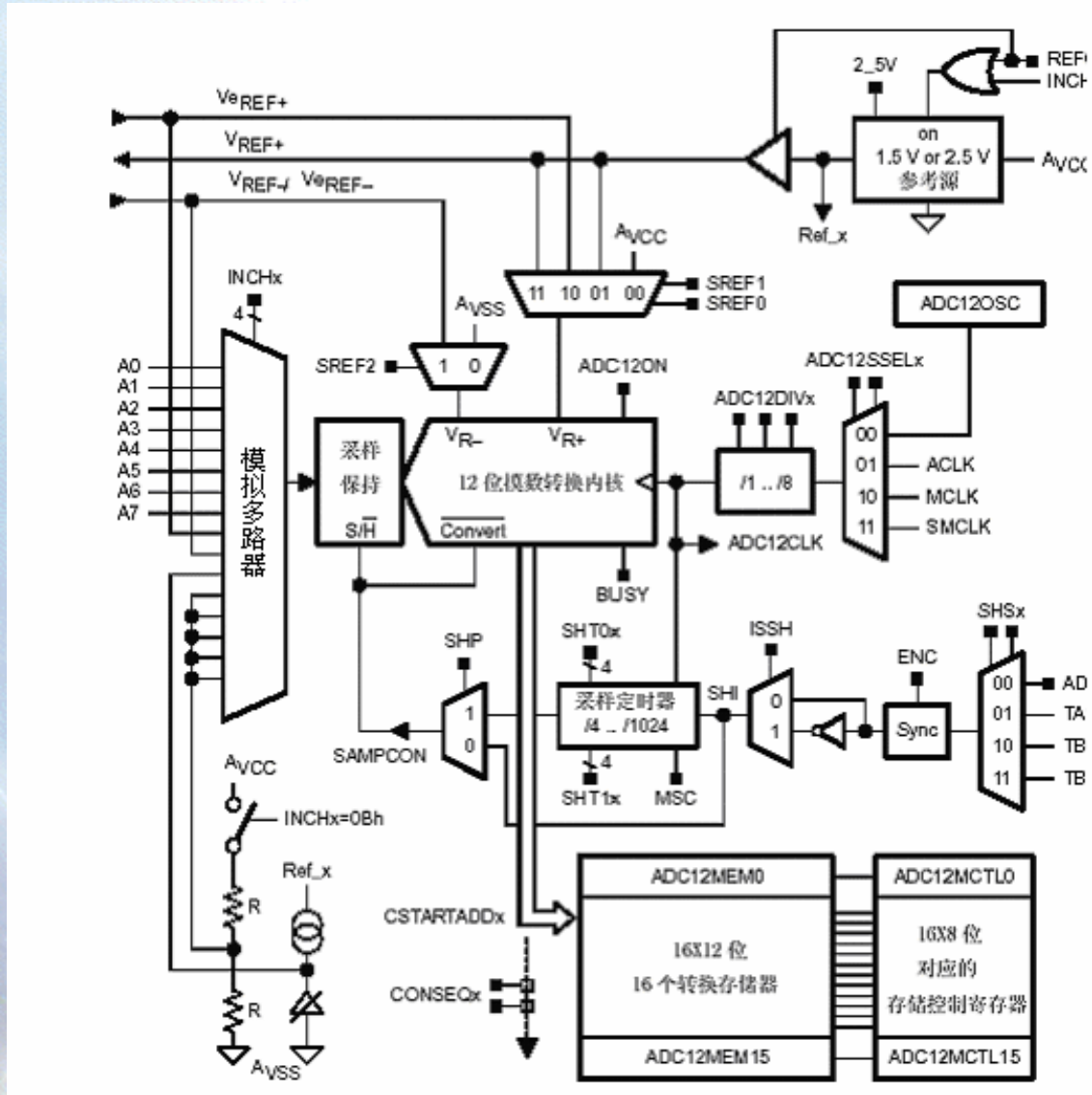
重新配置I2C模块进行UART或者SPI操作

- 清除I2C、I2CEN和SYNC位（CLR.B &U0CTL）
- 设置SWRST位（MOV.B #SWRST, &U0CTL）
- 进行UART或者SPI模式的初始化
  - 在SWART=1情况下初始化所有USART寄存器（包括UxCTL）
  - 通过特殊功能寄存器MEx使能USART模块（URXEx，UTXEx之一或全部）
  - 软件清除SWRST位（BIC.B #SWRST, &UxCTL）
  - 通过特殊功能寄存器IEx中断使能（URXIEx，UTXIEx之一或全部）（可选）

复位条件下配置I2C模式

- 在SWRST＝1情况下选择I2C模式（BIS.B #SYNC+I2C，&U0CTL）
- 清除I2CEN位（BIC.B #I2CEN, &U0CTL）
- 在I2C＝0情况下重新配置I2C模块
- 软件设置I2CEN（BIS.B #I2CEN, &U0CTL）

ADC12的结构


VeREF+
VREF+
VREF-/ VeREF-
2_5V
on
1.5 V or 2.5 V
参考源
AVCC
Ref_x
11 10 01 00
SREF1
SREF0
ADC12OSC
INCHx
4
AVSS
SREF2
1 0
ADC12ON
ADC12SSELx
ADC12DIVx
A0
A1
A2
A3
A4
A5
A6
A7
模拟多路器
采样
保持
S/H
VR-
VR+
12 位模数转换内核
Convert
/1 .. /8
00
01
10
11
ACLK
MCLK
SMCLK
ADC12CLK
BUSY
SHP
SHT0x
ISSH
SHSx
ENC
采样定时器
/4 .. /1024
SHI
Sync
SAMPCON
SHT1x
MSC
AD
TA
TB
TB
AVCC
INCHx=0Bh
Ref_x
R
R
AVSS
CSTARTADDx
CONSEQx
ADC12MEM0
16X12 位
16 个转换存储器
ADC12MEM15
ADC12MCTL0
16X8 位
对应的
存储控制寄存器
ADC12MCTL15

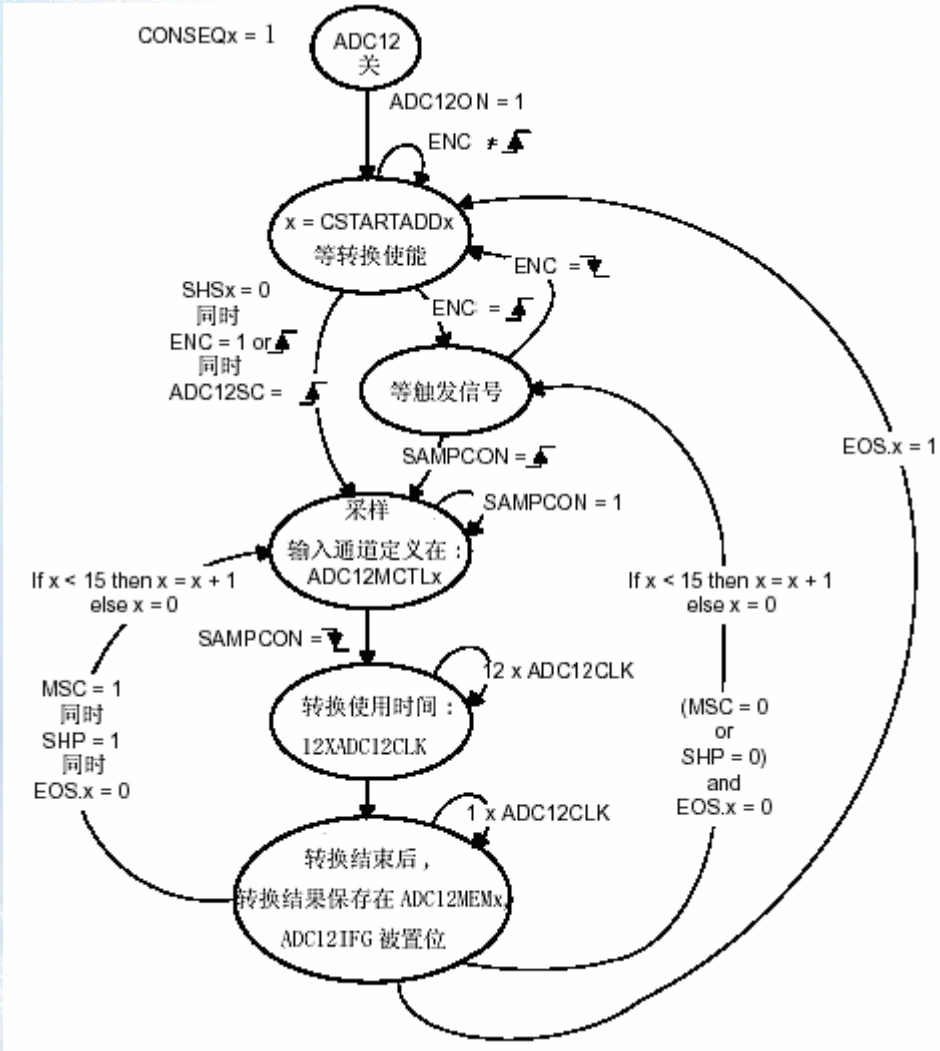
ADC12主要特点

- 12位转换精度，1位非线性微分误差，1位非线性积分误差
- 有多种时钟源提供给ADC12模块，而且模块本身内置时钟发生器
- 内置温度传感器
- Timer_A/Timer_B硬件触发器
- 配置有8路外部通道与4路内部通道
- 内置参考电源，并且参考电压有6种组合
- 模数转换有4种模式
- 16字转换缓存
- ADC12可关断内核支持超低功耗应用
- 采样速度快，最高可达200ksps
- 自动扫描
- DMA使能

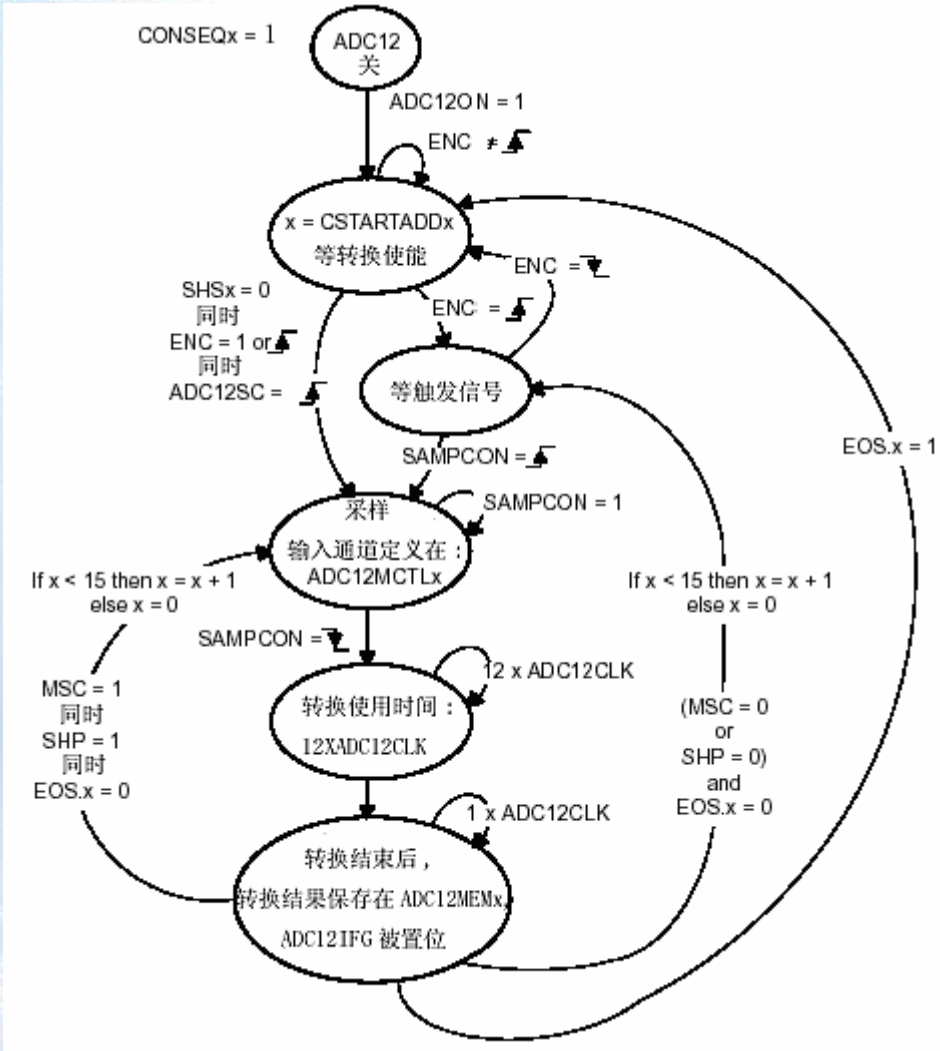
ADC12转换模式

- 单通道单次转换
- 序列通道单次转换
- 单通道多次转换
- 序列通道多次转换

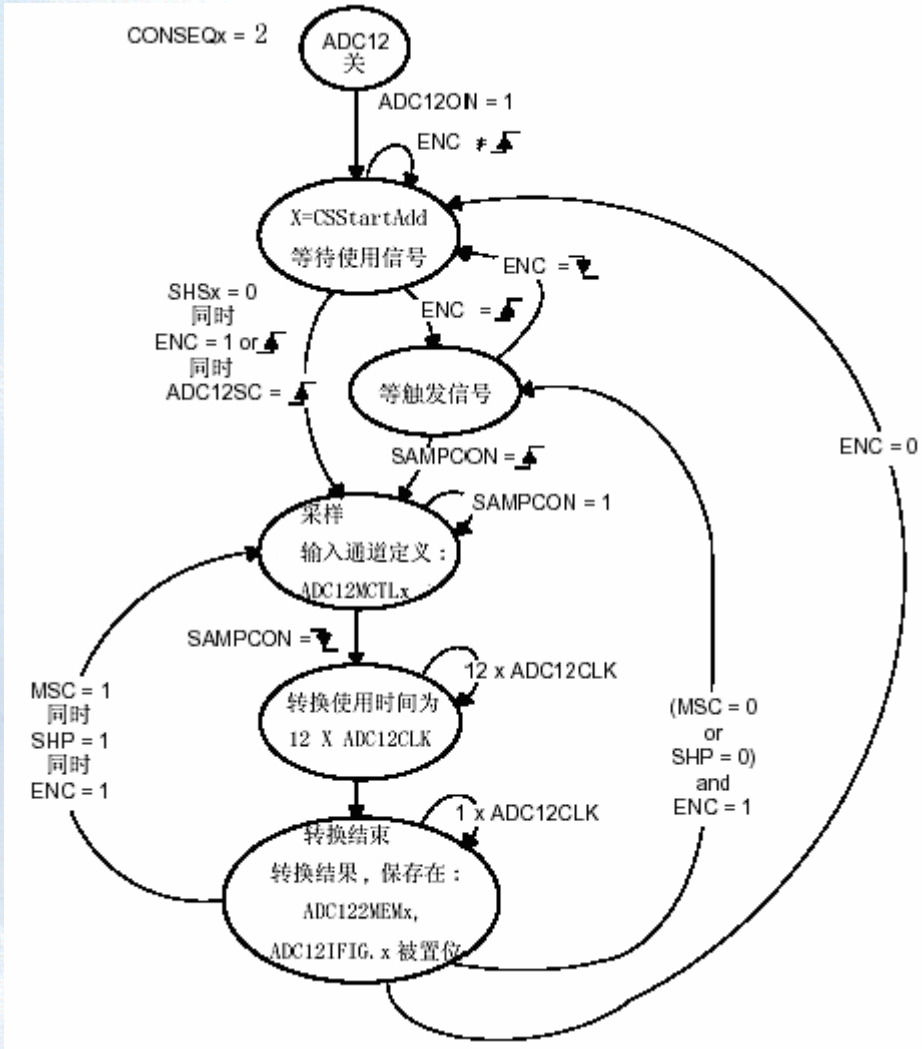
单通道单次转换模式状态



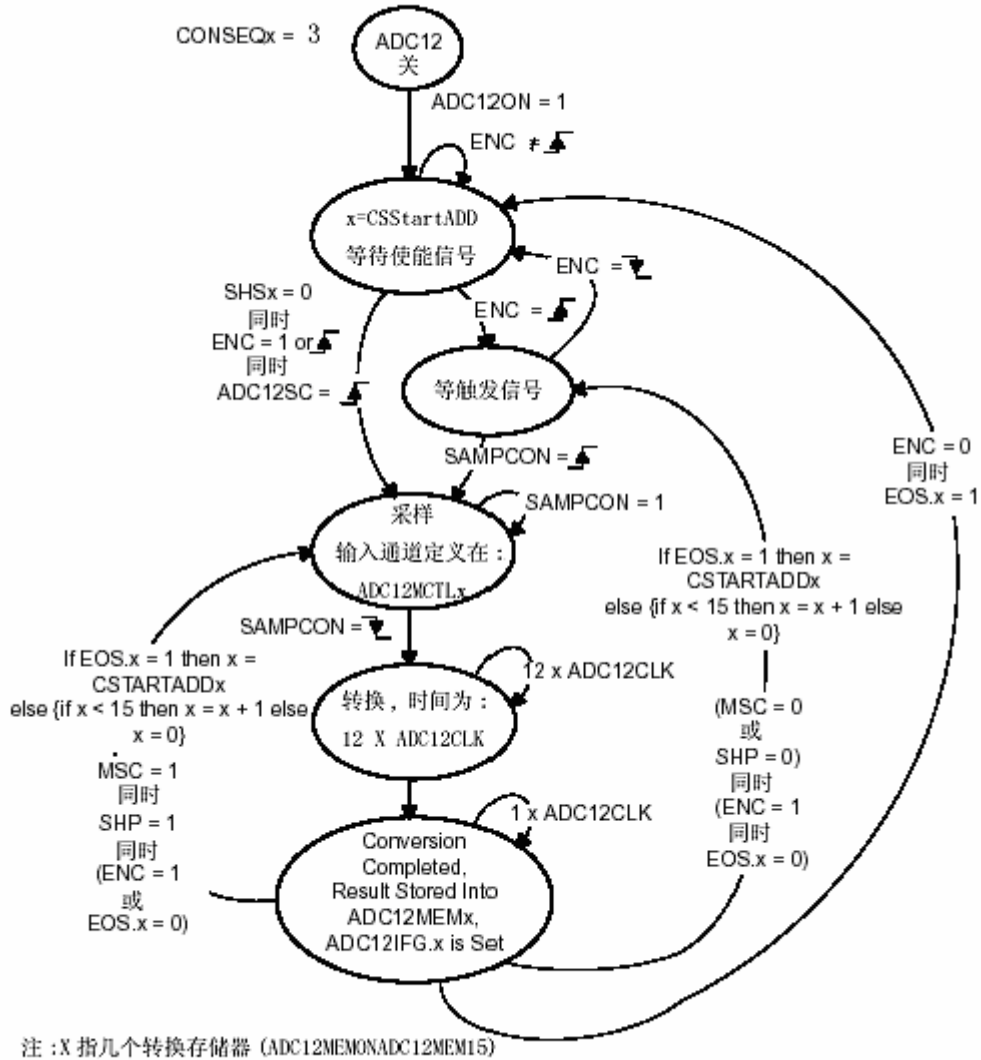
序列通道单次转换状态


CONSEQx = 1
ADC12关
ADC12ON = 1
ENC =
x = CSTARTADDx
等转换使能
ENC =
ENC =
SHSx = 0
同时
ENC = 1 or
同时
ADC12SC =
等触发信号
SAMPCON =
SAMPCON = 1
采样
输入通道定义在：
ADC12MCTLx
If x < 15 then x = x + 1
else x = 0
SAMPCON =
12 x ADC12CLK
转换使用时间：
12XADC12CLK
1 x ADC12CLK
转换结束后，
转换结果保存在 ADC12MEMx
ADC12IFG 被置位
MSC = 1
同时
SHP = 1
同时
EOS.x = 0
If x < 15 then x = x + 1
else x = 0
(MSC = 0
or
SHP = 0)
and
EOS.x = 0
EOS.x = 1

单通道多次模式的状态


CONSEQx = 2
ADC12
关
ADC12ON = 1
ENC
X=CSStartAdd
等待使用信号
ENC
SHSx = 0
同时
ENC = 1 or
同时
ADC12SC =
ENC
等触发信号
SAMPCON =
SAMPCON = 1
采样
输入通道定义：
ADC12MCTLx
SAMPCON =
12 x ADC12CLK
转换使用时间为
12 X ADC12CLK
1 x ADC12CLK
转换结束
转换结果，保存在：
ADC122MEMx,
ADC12IFIG.x 被置位
MSC = 1
同时
SHP = 1
同时
ENC = 1
(MSC = 0
or
SHP = 0)
and
ENC = 1
ENC = 0

序列通道多次转换状态



注：X 指几个转换存储器（ADC12MEMONADC12MEM15）
和转换存储控制寄存器（ADC12MCTLO—ADC12MCTL15）

ADC12应用举例

- 使用外部参考源

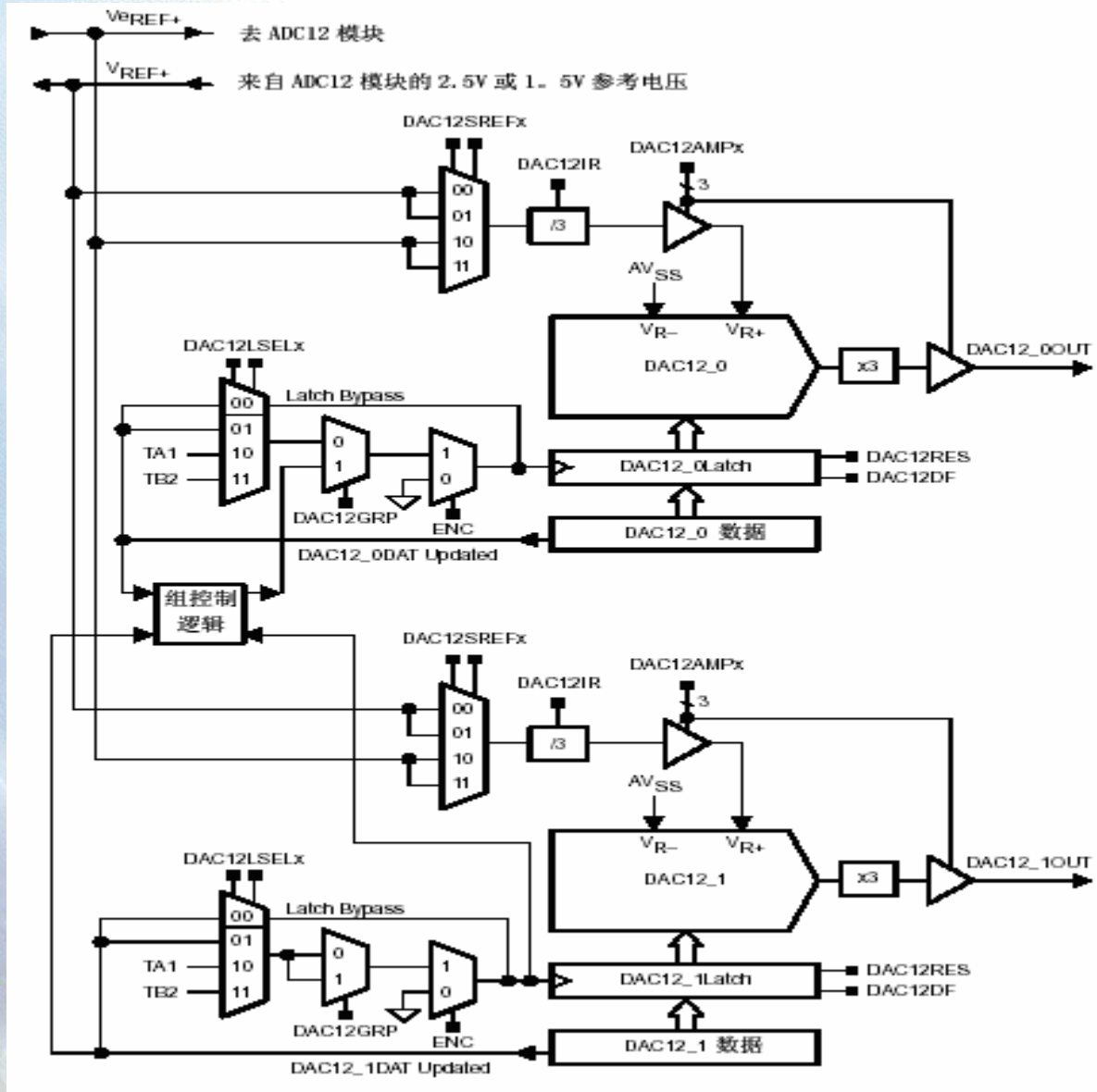
```
#include          "msp430x44x.h"
void main(void)
{
WDTCTL = WDTPW+WDTHOLD;
P6SEL |= 0x01;                                  // 使能A/D 通道A0
 ADC12CTL0 = ADC12ON+SHT0_2; // 打开 ADC12, 设置采样时钟
  ADC12CTL1 = SHP;                              // 使用采样时钟
  ADC12MCTL0 = SREF_2;                  // Vr+ = VeREF+ (外部)
  ADC12CTL0 |= ENC;                             // 使能转换
     while (1)
  {
    ADC12CTL0 |= ADC12SC;                       // 开始转换
    while ((ADC12IFG & ADC12BUSY)==0);
    _NOP();
  }
}
```

ADC12应用举例

- 使用内部参考源

```
#include          "msp430x44x.h"
void main(void)
{
  unsigned int i;
WDTCTL = WDTPW+WDTHOLD;
P6SEL |= 0x01;                              // 使能 A/D 通道A0
  ADC12CTL0 = ADC12ON+SHT0_2+REFON+REF2_5V;
ADC12CTL1 = SHP;
ADC12MCTL0 = SREF_1;                        // Vr+=Vref+
  for ( i=0; i<0x3600; i++)                 // 为参考源启动提供延迟
  {
  }
  ADC12CTL0 |= ENC;                           // 使能转换
  while (1)
  {
    ADC12CTL0 |= ADC12SC;                     // 开始转换
    while ((ADC12IFG & BIT0)==0);
    _NOP();
  }
}
```

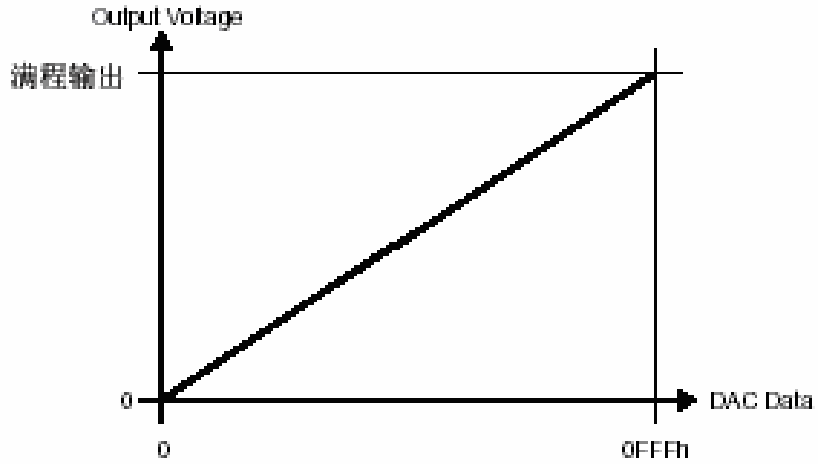
DAC12的结构


VeREF+
去 ADC12 模块
VREF+
来自 ADC12 模块的 2.5V 或 1.5V 参考电压
DAC12SREFx
DAC12IR
DAC12AMPx
00
01
10
11
/3
3
AVSS
VR-
VR+
DAC12_0
x3
DAC12_0OUT
DAC12LSELx
Latch Bypass
TA1
TB2
DAC12GRP
ENC
DAC12_0Latch
DAC12RES
DAC12DF
DAC12_0 数据
DAC12_0DAT Updated
组控制
逻辑
DAC12_1
DAC12_1OUT
DAC12_1Latch
DAC12_1 数据
DAC12_1DAT Updated

DAC12的主要特征

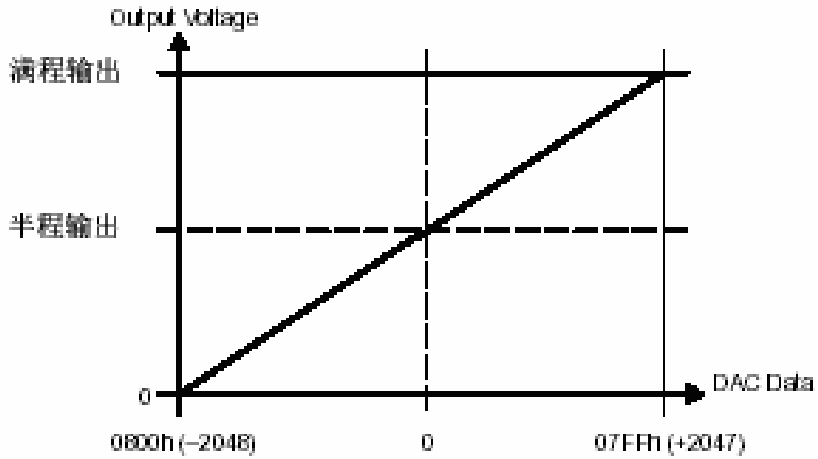
- 8位、12位分辨率
- 可编程的时间对能量消耗
- 内部或外部参考电压
- 支持无符号和有符号数据输入
- 具有自校验功能
- 二进制或者二的补码形式
- 多路DAC同步更新
- 可直接存储器存取

DAC12_xDAT的数据格式

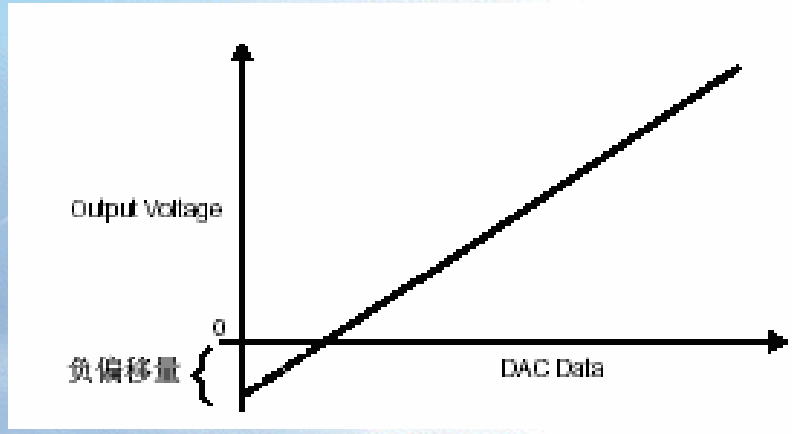


DAC12采用12位二进制数格式

DAC12使用2的补码形式

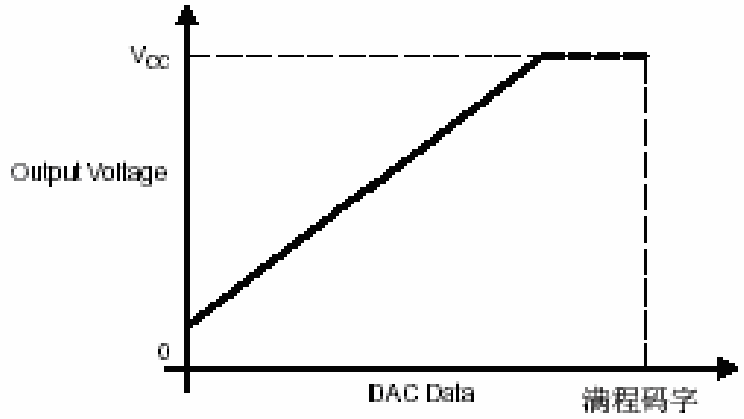


校正DAC12输出



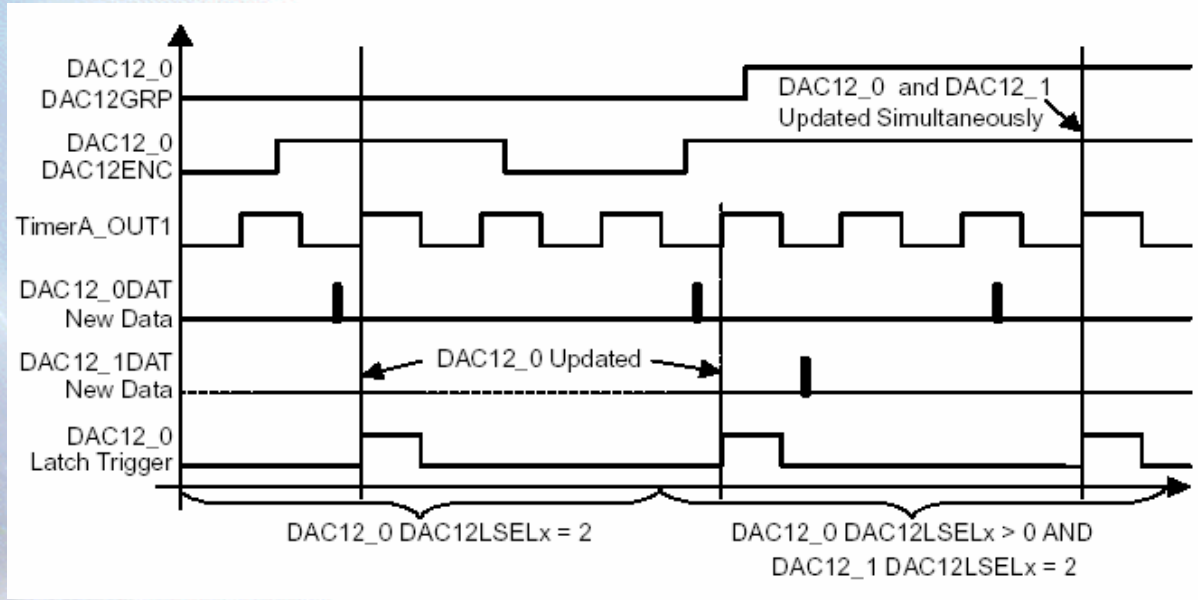
负偏移量

正偏移量



组合多个DAC12模块

- MSP430x15x以及MSP430x16x中，DAC12_0和DAC12_1通过设置DAC12_0的DAC12GRP位实现组合。当DAC12_0和DAC12_1处于组合状态，只有两个转换通道的DAC12LSELx大于零并且DAC12ENC置位这两个条件同时满足情况下，才可以由DAC12_1DAC12LSELx位选择两个DAC的更新触发源。



DAC12应用举例

- **阶梯波的产生：**在一定时间范围内，每隔一段时间，输出幅度递增一个恒定值。阶梯波可以通过延迟程序或定时器来配合DAC12产生。
- **三角波的产生：**三角波是由两段直线组成，先输出一个线性增长的波形，达到最大值时，再送出一个线性减少的波形，这两个波形合到一起就成为三角波。可通过控制DAC12的输入值递增、递减来实现
- **不规则信号的产生：**可以把不规则信号的采样值，存储在程序存储器中，然后用查表的方法读出这些值，送到DAC12一个通道后输出到Y轴上，同时利用另一个DAC12通道在X轴送出锯齿波，以产生水平扫描线。两个DAC12通道信号的频率应保持一定的比例关系，从而能够使显示波形保持同步。当然也可用这种方法产生规则的波形，如正弦波等。