http://www.Microcontrol.cn 微控设计网

中国 MSP430 单片机专业网站

MSP430F 常用模块应用原理

微控设计网 版主 DC 策划
原创于:2006-3-7
最后更新:2008-5-31 V8.2

为了更好地引导 MSP430 单片机爱好者的入门，微控设计网为大家整理了一个份 MS430 单片机入门资料，希望能够帮助到更多的国内单片机爱好者朋友。

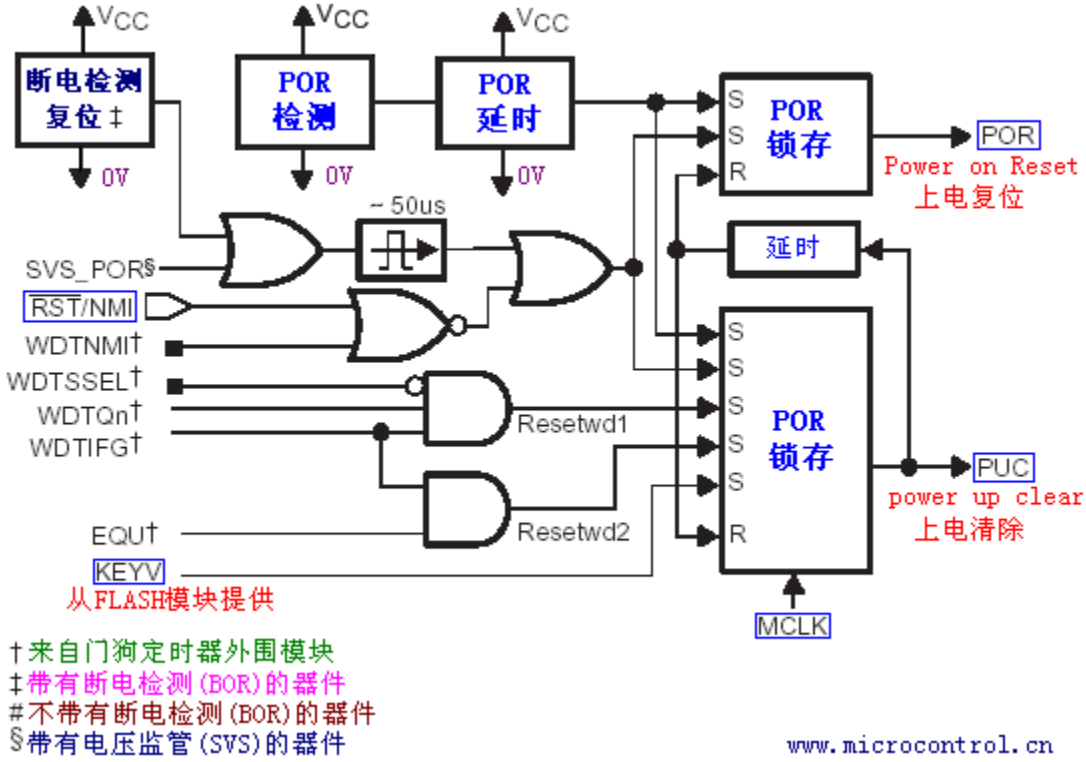
如果你在学习或应用 MSP430 单片机过程中想与同行分享成果或交流技术问题，欢迎进入我们的微控技术论坛。

欢迎购买微控设计网 MSP430 单片机开发工具

DC 版主 QQ: 108517559

模块列表

1-复位模块



MSP430 单片机系统复位电路

从上 MSP430 系统复位电路功能模块图中可以看到了两个复位信号，一个是上电复位信号 POR(Power On Reset)和上电清除信号 PUC(Power Up Clear)。

POR 信号是器件的复位信号，此信号只有在以下的事件发生时才会产生：

☐器件上电时。

☐RST/NMI 引脚配置为复位模式，当 RST/NMI 引脚生产低电平时。

当 POR 信号产生时，必然会产生 PUC 信号；而 PUC 信号的产生时不会产生 POR 信号。会引起产生 PUC 信号的事件：

☐POR 信号发生时。

☐ 启动看门狗时，看门狗定时器计满时。

☐ 向看门狗写入错误的安全参数值时。

☐向片内 FLASH 写入错误的安全参数值时。

MSP430 单片机系统复位后器件的初始

当 POR 信号或 PUC 信号发生时引起器件复位后，器件的初始化状态为：

☐RST/NMI 引脚配置为复位模式。

☐ I/O 引脚为输入模式。

□ 装态寄存器复位。

□ 程序计数器(PC)装入复位向量地址 0xFFFE,CPU 从此地址开发始执行。

□其它模块的寄存器初始化，详情请查器件手册。

POR 和 PUC 两者的关系：

POR 信号的产生会导致“系统复位”并“产生 PUC 信号”。而 PUC 信号不会引起 POR 信号的产生。

无论是 POR 信号还是 PUC 信号触发的复位,都会使 MSP430 从地址 0xFFFE 处读取复位中断向量,程序从中断向量所指的地址处开始执行。触发 PUC 信号的条件中,除了 POR 产生触发 PUC 信号外,其他的豆科一通过读取相应的中断向量来判断是何种原因引起的 PUC 信号,以便作出相应的处理。

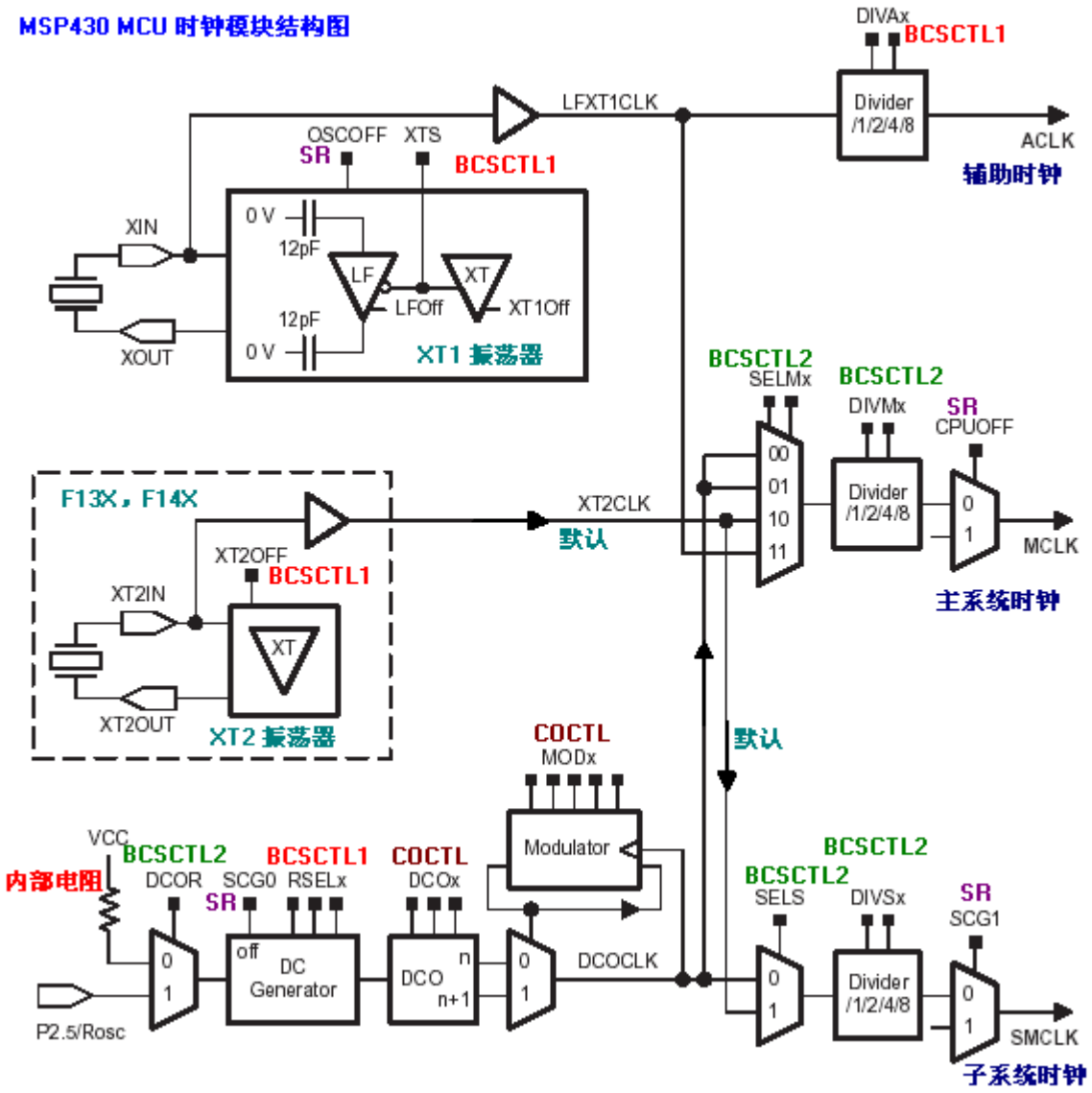
系统复位(指 POR)后的状态为：

(1) RST/NMI 管脚功能被设置为复位功能;
(2)所有 I/O 管脚被设置为输入;
(3)外围模块被初始化,其寄存器值为相关手册上的默认值;
(4)状态寄存器 SR 复位;
(5)看门狗激活,进入工作模式;
(6)程序计数器 PC 载入 0xFFFE 处的地址,微处理器从此地址开始执行程序。

典型的复位电路有一下 3 种：

(1) 在 RST/NMI 管脚上接 100K 欧的上拉电阻。
(2)在[1]的基础上再接 0.1uf 的电容,电容的一端接地,可以使复位更加可靠。
(3)在[2]的基础上,再在电阻上并接一个型号为 IN4008 的二极管,可以可靠的实现系统断电后立即上电。

2-时钟模块



MSP430F1XX 系列时钟模块图

时基模块结构如上图:

MSP430 系列单片机基础时钟主要是由低频晶体振荡器,高频晶体振荡器，数字控制振荡器(DCO)，锁频环(FLL)及 FLL+等模块构成。由于 430 系列单片机中的型号不同，而时钟模块也将有所不同。虽然不同型号的单片机的时基模块有所不同，但这些模块产生出来的结果是相同的.在 MSP430F13、14 中是有 TX2 振荡器的，而 MSP430F11X,F11X1 中是用 LFXT1CLK 来代替 XT2CLK 时钟信号的.在时钟模块中有 3 个(对于 F13,F14)时钟信号源(或 2 个时钟信号源，对于 F11X、F11X1):

1-LFXT1CLK: 低频/高频时钟源.由外接晶体振荡器,而无需外接两个振荡电容器.较常使用的晶体振荡器是 32768HZ。

2-XT2CLK: 高频时钟源.由外接晶体振荡器。需要外接两个振荡电容器，较常用的晶体振荡器是 8MHZ。

3-DCOCLK: 数字可控制的 RC 振荡器。

MSP430 单片机时钟模块提供 3 个时钟信号输出，以供给片内各部电路使用。

1-ACLK: 辅助时钟信号.由图所示,ACLK 是从 FLXT1CLK 信号由 1/2/4/8 分频器分频后所得到的.由 BCSCTL1 寄存器设置 DIVA 相应为来决定分频因子.ACLK 可用于提供 CPU 外围功能模块作时钟信号使用.

2-MCLK: 主时钟信号.由图所示,MCLK 是由 3 个时钟源所提供的。他们分别是 LFXT1CLK,XT2CLK(F13、F14，如果是 F11,F11X1 则由 LFXT1CLK 代替)，DCO 时钟源信号提供.MCLK 主要用于 MCU 和相关系统模块作时钟使用。同样可设置相关寄存器来决定分频因子及相关的设置。

3-SMCLK: 子系统时钟，SMCLK 是由 2 个时钟源信号所提供.他们分别是 XT2CLK(F13、F14)和 DCO，如果是 F11、F11X1 则由 LFXT1CLK 代替 TX2CLK。同样可设置相关寄存器来决定分频因子及相关的设置。

MSP430X1X1 系列产品中,其中 XT1 时钟源引脚接法有如 3 种应用。F13、14 的 XT1 相同。需要注意的是，LFXT1 只有工作在高频模式下才需要外接电容。
对以引脚较少的 MSPX1XX 系列产品中有着不同时基模块,具体如下:

MSP430X11X1:LFXT1CLK , DCO

MSP430F12X: LFXT1CLK , DCO

MSP430F13X/14X/15X/16X:LFXT1CLK , DCO , XT2CLK

MSP430F4XX: LFXT1CLK , DCO , XT2CLK , FLL+

时钟发生器的原理说明：
问题的提出：1、高频、以便能对系统硬件请求和事件作出快速响应
2、低频率，以便将电流消耗降制至最少
3、稳定的频率，以满足定时器的应用。
4、低 Q 值振荡器，以保证开始或停止操作没有延时
MSP430 采用了一个折衷的办法：就是用一个低频晶镇振，将其倍频在高频的工作频率上。一般采用这种技术的实用方法有两种，一个是说、锁相环、 一个是锁频环，而锁相环采用模拟的控制容易引起“失锁”和易引起电容量的改变。而 TI 采用的是锁频环技术，它采用数字控制器 DCO 和频率积分来产 生高频的运行时钟频率。

低功耗设置的技巧问题：
1、LPM4：在振荡器关闭模式期间，处理机的所有部件工作停止，此时电流消耗最小。此时只有在系统上电电路检测到低点电平或任一请求异步响应中断的外部中断事件时才会从新工作。因此在设计上应含有可能需要用到的外部中断才采用这种模式。否则发生不可预料的结果。
2、LPM3：在 DC 发生器关闭期间，只有晶振是活动的。但此时设置的基本时序条件的 DC 发生器的 DC 电流被关闭。由于此电路的高阻设计，使功耗被抑制。注：当从 DC 关闭到启动 DC0 要花一端时间 (ns-us)
3、LPM2：在此期间，晶镇振和 DC 发生器是工作的，所以可实现快速启动。
4、LPM1：在此振荡器已经工作，所以不存在启动时间延时问题。

结合上述特点，在写程序时要综合考虑低功好耗特性，对外部事件的安排也很重要。你必须在功能实现上综合考虑才能达到你预期的效果。使用 C 语言可用如下的语

句：_BIS_SR(LMP3_bits)和_BIC_SR(LPM3 bits)
LPM3 和 LPM3_EXIT
它们的定义是一样的。这里说明在 C 语言环境中有些定义的函数是不可见的。但你可以从 in430.h 文件看到它们的定义。

DCOCTL DCO 控制寄存器

7	6	5	4	3	2	1	0
DCO.2	DCO.1	DCO.0	MOD.4	MOD.3	MOD.2	MOD.1	MOD.0

DCO.0-DCO.4 定义 8 种频率之一，可以分段调节 DCOCLK 频率，相邻两种频率相差 10%。而频率由注入直流发生器的电流定义。

MOD.0-MOD.4 定义在 32 个 DCO 周期中插入的 Fdco+1 周期个数，而在下的 DCO 周期中为 Fdco 周期，控制改换 DCO 和 DCO+1 选择的两种频率。如果 DCO 常数为 7，表示已经选择最高频率，此时不能利用 MOD.0-MOD.4 进行频率调整。

BCSCTL1 基本时钟系统控制寄存器 1

7	6	5	4	3	2	1	0
XT2OFF	TXS	DIVA.1	DIVA.0	XT5V	Rsel.2	Resl.1	Resl.0

XT2OFF 控制 XT2 振荡器的开启与关闭。
TX2OFF=0，XT2 振荡器开启。
TX2OFF=1，TX2 振荡器关闭（默认为 TX2 关闭）

XTS 控制 LFXT1 工作模式，选择需结合实际晶体振荡器连接情况。
XTS=0，LFXT1 工作在低频模式（默认）。
XTS=1，LFXT1 工作在高频模式（必须连接有高频相应的高频时钟源）。

DIVA.0 DIVA.1 控制 ACLK 分频。
0 不分频（默认）
1 2 分频
2 4 分频
3 8 分频

XT5V 此位设置为 0。

Resl1.0，Resl1.1，Resl1.2 三位控制某个内部电阻以决定标称频率。
Resl=0，选择最低的标称频率。
……..
Resl=7，选择最高的标称频率。

BCSCTL2 基本时钟系统控制寄存器 2

7	6	5	4	3	2	1	0
SELM.1	SELM.0	DIVM.1	DIVM.0	SELS	DIVS.1	DIVS.0	DCOR

SELM.1 SELM.0 选择 MCLK 时钟源

0 时钟源为 DCOCLK（默认）
1 时钟源为 DCOCLK
2 时钟源为 LFXT1CLK（对于 MSP430F11/12X），时钟源为 XT2CLK（对于 MSP430F13/14/15/16X）；
3 时钟源为 LFTXTICLK。

DIVM.1 DIVM.0 选择 MCLK 分频
0 1 分频（默认）
1 2 分频
2 4 分频
3 8 分频

SELS 选择 SMCLK 时钟源
0 时钟源为 DCOCLK（默认）
1 时钟源为 LFXT1CLK（对于 MSP430F11/12X），时钟源为 XT2CLK（对于 MSP430F13/14/15/16X）。

DIVS.1 DIVS.0 选择 SMCLK 分频。
0 1 分频
1 2 分频
2 4 分频
4 8 分频

DCOR 选择 DCO 电阻
0 内部电阻
1 外部电阻

PUC 信号之后，DCOCLK 被自动选择 MCLK 时钟信号，根据需要，MCLK 的时钟源可以另外设置为 LFXT1 或者 XT2。设置顺序如下：
[1] 复位 OscOff
[2] 清除 OFIFG
[3] 延时等待至少 50us
[4] 再次检查 OFIFG，如果仍然置位，则重复[3]、[4]步骤，直到 OFIFG=0 为止。

时基模块应用范例(1)

本例程是设置时钟模块的工作方式和相关的控制寄存器.以 MSP430F149 和 C 程序编写.设置主时钟信号 MCLK=TX2,子时钟信号 SMCLK=DCOCLK,将 MCLK 从 MSP430F149 的 P5.4 口输出,在 F14X 系列中 P5.4 和 MCLK 是复用的,(详情请看相关的芯片资料可查得)

```
#include <msp430x14x.h>
```

```
void main (void)
{
  unsigned int i;
  WDTCL = WDTPW+WDTHOLD;   //停止看门狗
  P5DIR = 0x10;            //设置 P5.4 输出
  P5SEL = 0x10;            //设置 P5.4 口为外围模块用作 MCLK 信号输出
  BCSCTL1 &= ~XT2OFF;      //使 TX2 有效,TX2 上电时默认为关闭的.

do
{
  IFG1 &= ~OFIFG;          //清振荡器失效标志
  for(i= 0xff; i>0; i--); //延时,待稳定.
}

 while ((IFG1 & OFIFG)!=0);      //若振荡器失效标志有效

  BCSCTL2 |= SELM1;               //使 MCLK = XT2
  for(;;);
}
```

时基模块应用范例(2)

XT1 时钟源外接高频晶体振荡器并使其工作在高频模式.利用 P2.0 的复用引脚功能输出 ACLK(高频).同时 P1.1 输出一个方波.

```
#include <msp430x11x1.h>
void main(void)
{
  unsigned int i;
  WDTCTL = WDTPW + WDTHOLD;     // 停止看门狗
  BCSCTL1 |= XTS;
 // 设置时基寄存器 1,使 ACLK = LFXT1 = HF XTAL,也就是高频模式.
  P2DIR |= 0x01;                // 设置 P2.0 方向寄存器为输出
  P2SEL |= 0x01;     // 设置 P2.0 口为外围模块用作 ACLK 信号输出
  P1DIR |= 0x02;                // 设置 P1.1 方向寄存器为输

do
 {
  IFG1 &= ~OFIFG;               // 清振荡器失效标志
  for (i = 0xFF; i > 0; i--);   // 延时,待稳定
 }
while ((IFG1 & OFIFG));         //若振荡器失效标志有效?
```

```
BCSCTL2 |= SELM_3;
// 设置时基寄存器 2,使主时钟信号 MCLK = LFXT1 (可靠的)

for (;;)                          // 无穷循环
 {
  P1OUT |= 0x02;                  // P1.1 = 1
  P1OUT &= ~0x02;                 // P1.1 = 0
 }
}
```

3-10 端口

数字输入输出端口使用特性：

1、所有端口都可以单独进行编程。

2、可以进行输入输出和中断条件的任意组合

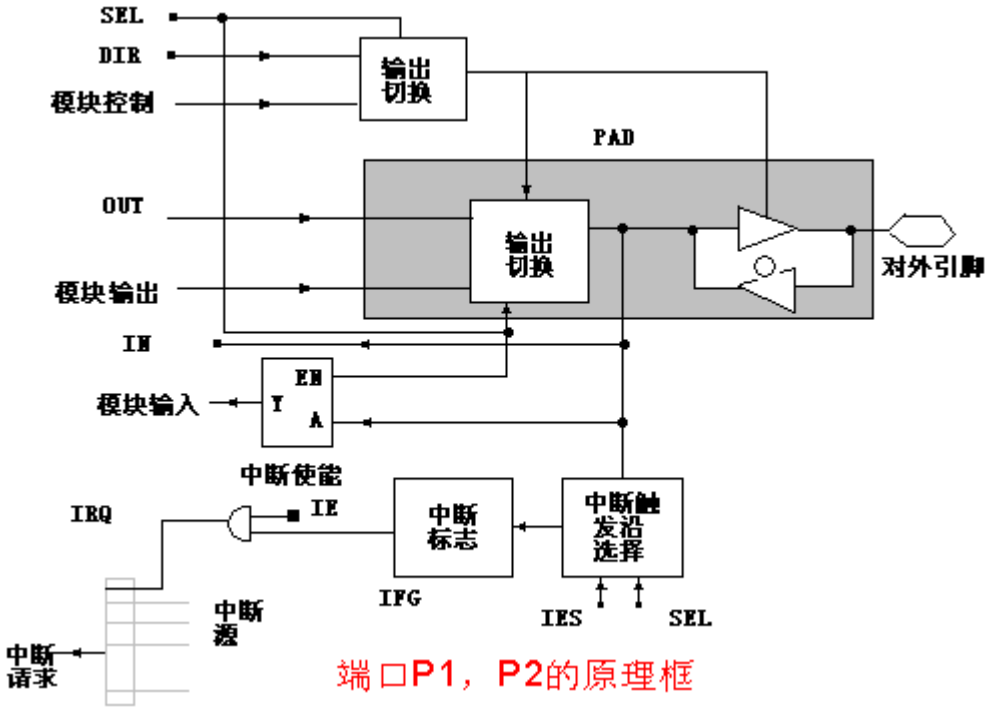
3、对具有中断功能的引脚输入沿可进行选择

4、具有第 2 功能选择，以适合不同 I/O 口操作。

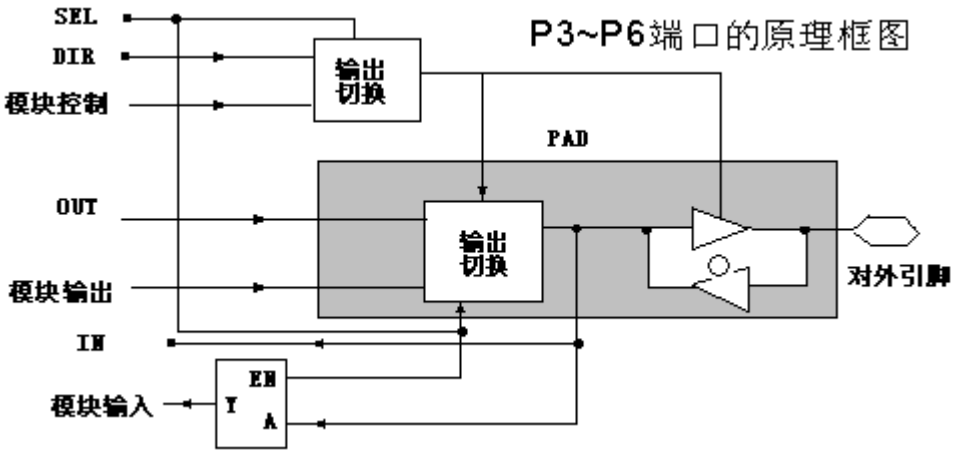
5、所有指令支持端口控制寄存器的读写操作

注：1、对输入寄存器写入会增加电流的消耗。

2、由于各端口的功能各不相同，因此它们的内部电路原理也不相同，具体可参考数据手册。



端口P1，P2的原理框



MSP430 的端口

器件	P1	P2	P3	P4	P5	P6	S	COM
MSP430F11X	√	√						
MSP430F12X	√	√	√					
MSP430F13/14/15/16	√	√	√	√	√	√		
MSP430F4XX	√	√	√	√	√	√	√	√
MSP430F20X	√	√						
MSP430F21X	√	√						
MSP430F22X	√	√	√	√				

MSP430 端口功能

端口	功能
P1、P2	I/O、中断功能、其他片内外设功能
P3、P4、P5、P6	I/O、其他片内外设功能
S、COM	I/O、驱动液晶

PxDIR 方向寄存器

7	6	5	4	3	2	1	0
P7DIR	P6DIR	P5DIR	P4DIR	P3DIR	P2DIR	P1DIR	P0DIR

0 为输入模式

1 为输出模式

在 PUC 后全都为复位，作为输入时，只能读；作为输出时，可读可定。

PxIN 输入寄存器

7	6	5	4	3	2	1	0
PxIN	PxIN	PxIN	PxIN	PxIN	PxIN	PxIN	PxIN

输入寄存器是只读的，用户不能对它写入，只能读取其 IO 内容。此时引脚方向必须为输入。

PxOUT 输出寄存器

7	6	5	4	3	2	1	0
P7OUT	P6OUT	P5OUT	P4OUT	P3OUT	P2OUT	P1OUT	P1OUT

这是 IO 端口的输出缓冲器，在读取时输出缓存的内容与脚引方向定义无关。改变方向寄存器的内容，输出缓存的内容不受影响。

PxIFG 中断标专寄存器

7	6	5	4	3	2	1	0
P7IFG	P6IFG	P5IFG	P4IFG	P3IFG	P2IFG	P1IFG	P0IFG

标志相应引脚是否有待处理中断信息。

0 没有中断请求

1 有中断请求

PxIES 中断触发沿选择寄存器

7	6	5	4	3	2	1	0
P7IES	P6IES	P5IES	P4IES	P3IES	P2IES	P1IES	P0IES

0 上升沿使相应标志置位

1 下降沿使相应标志置位

PxIE 中断使能寄存器

7	6	5	4	3	2	1	0
P7IE	P6IE	P5IE	P4IE	P3IE	P2IE	P1IE	P0IE

0 禁止中断

1 允许中断

PxSEL 功能选择寄存器

7	6	5	4	3	2	1	0
P7SEL	P6SEL	P5SEL	P4SEL	P3SEL	P2SEL	P1SEL	P0SEL

0 选择引脚为 I/O 功能。

1 选择引脚为外围模块功能

关于端口 P3、P4、P5、P6

端口 P3、P4、P5、P6 是没有中断功能的，其它功能与 P1、P2 相同。所以在此不再作详尽说明。

关于端口 COM、S

这些端口实现与 LCD 片的驱动接口，COM 端是 LCD 片的公共端，S 端为 LCD 片的段码端。LCD 片输出端也可以用软件配置为数字输出端口，详情使用请查看其手册。

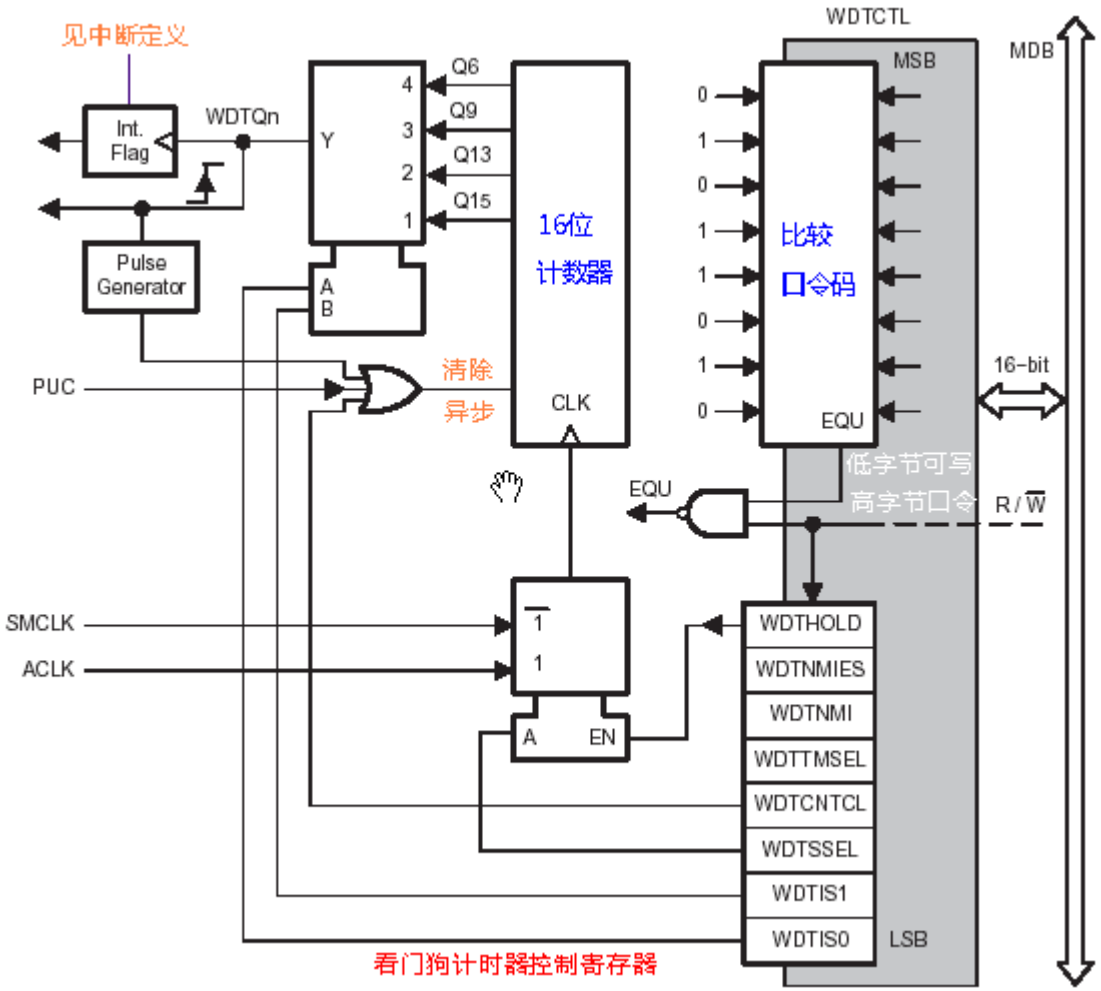
4-WDT 看门狗

看门狗定时器实际上是一个特殊的定时器，它的的功能是当程序运行发生故障时序时能使系统从新启动。其原理就是发生的故障的时间满足规定的定时时间后，产生一个非屏蔽中断，使系统的复位。这样当在调试程序或预计程序运行在某段内部可能瞬时发生时序错误时（如外部电路干扰）选用设置看门狗定时中断可以避免程序跑飞看门狗的定时时间可以通过 **WDTCTL** 中的低三位（**SSEL**、**IS1**、**IS0**）选择，当系统时钟为 **1MHz** 时，最大可设置为 **1** 秒钟，最小可设置 **64** 微秒。

WDTCTL 是一个 **16** 位寄存器，其高字节为口令，口令为 **5AH**，当对它写入操作时必须写口令才能操作，否则会导致系统复位。

另外该模块还具有定时器的功能。你可通过 **TMSET** 位进行选择。你可通过设置 **CNTCL** 来使 **WDTCNT** 从 **0** 开始计数。其定时按选定的时间周期产生中断请求。

当 **WDT** 工作在定时器模式时，**WDTCTL** 中断标志位在定时间到时置位，因该模式下定时器中断源是单源的，当得到中断服务时其 **WDTCTL** 标志位复位。



WDTCNT 计数单元

15--0

这是 16 位增计数器,由 MSP430 所选定的时钟电路产生的固定周期时钟信号对计数器进行加法计数。如果计数器事先被预置的初始状态不同，那么从开始计数到计数溢出为止所用的时间就不同。WDTCNT 不能直接通过软件存取，必须通过看门狗定时器的控制寄存器 WDTCTL 来控制。

WDTCTL 控制寄存器

15--8	7	6	5	4	3	2	1	0
口令	HOLD	NMIES	NMI	TMSEL	CNTCL	SSEL	IS1	IS0

WDTCTL 由高 8 位口令和低 8 位控制命令组成。要写入操作 WDT 的控制命令，出于安全原因必须先正确写入高字节看门狗口令。口令为 5AH，如果口令写错将导致系统复位。读 WDTCTL 时不需要口令。这个控制寄存器还可以用于设置 NMI 引脚功能。

ISO，IS1 选择看门狗定时器的定时输出。其中 T 是 WDTCNT 的输入时钟源周期。

0 T x 2(15)
1 T x 2(13)
2 T x 2(9)
3 T x 2(6)

SSEL 选择 WDTCNT 的时钟源

0 SMCLK
1 ACLK

由 ISO，IS1，SSEL3 可确定 WDT 定时时间。WDT 最多只能定时 8 种和时钟源相关的时间。下表列出了 WDT 可选的定时时间(晶体为 32768HZ，SMCLK=1MHZ)。

WDT 的定时时间表

SSEL	IS1	IS0	定时时间/ms	
0	1	1	0.056	Tsmclk x 2(6)
0	1	0	0.5	Tsmclk x 2(9)
1	1	1	1.9	Taclk x 2(6)
0	0	1	8	Tsmclk x 2(13)
1	1	0	16	Taclk x 2(9)
0	0	0	32	Tsmclk x 2(15) (PUC 复位后的值)
1	0	1	250	Taclk x 2(13)
1	0	0	1000	Taclk x 2(15)

CNTCL 当该位为 1 时，清除 WDTCNT。

TMSEL 工作模式选择

0 看门狗模式

1 定时器模式

NMI 选择 RST/NMI 引脚功能，在 PUC 后被复位。
0 RST/NMI 引脚为复位端
1 RST/NMI 引脚为边沿触发的非屏蔽中断输入。

NMIES 选择中断的边沿触发方式
0 上升沿触发 NMI 中断
1 下降沿触发 NMI 中断

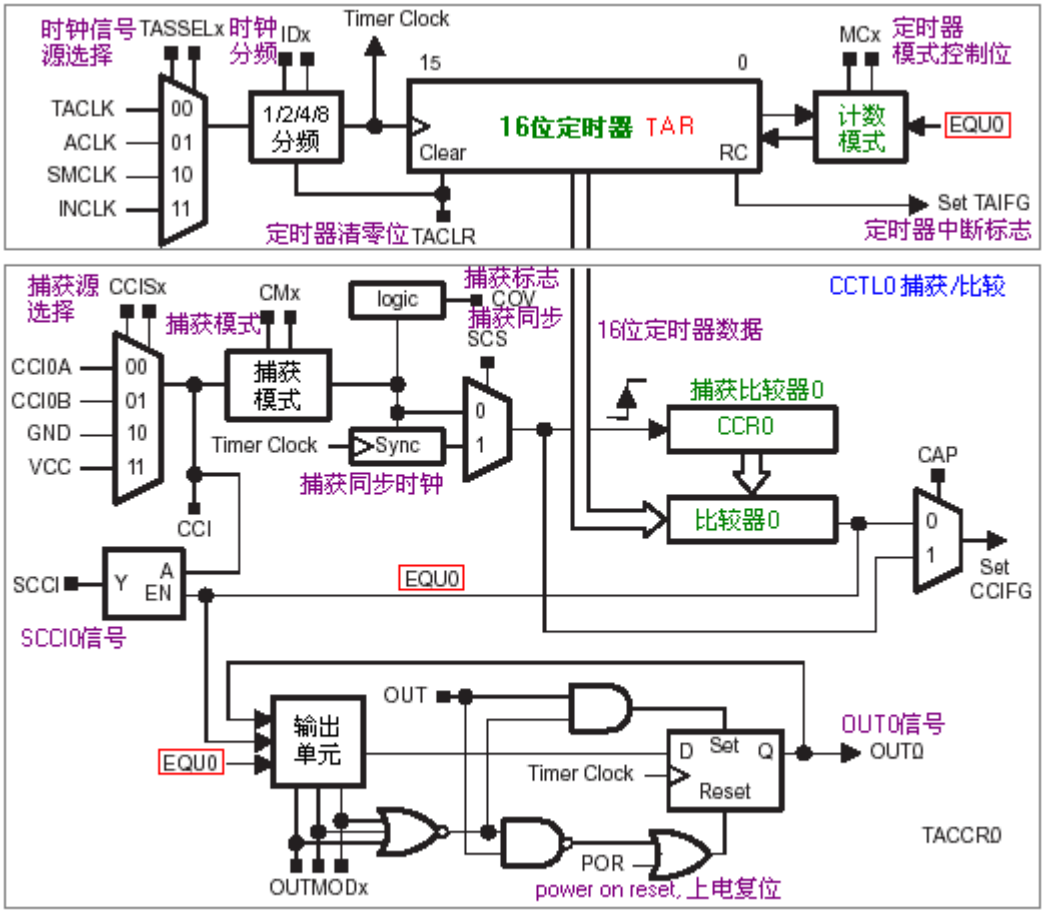
HOLD 停止看门狗定时器工作，降低功耗。
0 WDT 功能激活
1 时钟禁止输入，计数停止

5-定时器

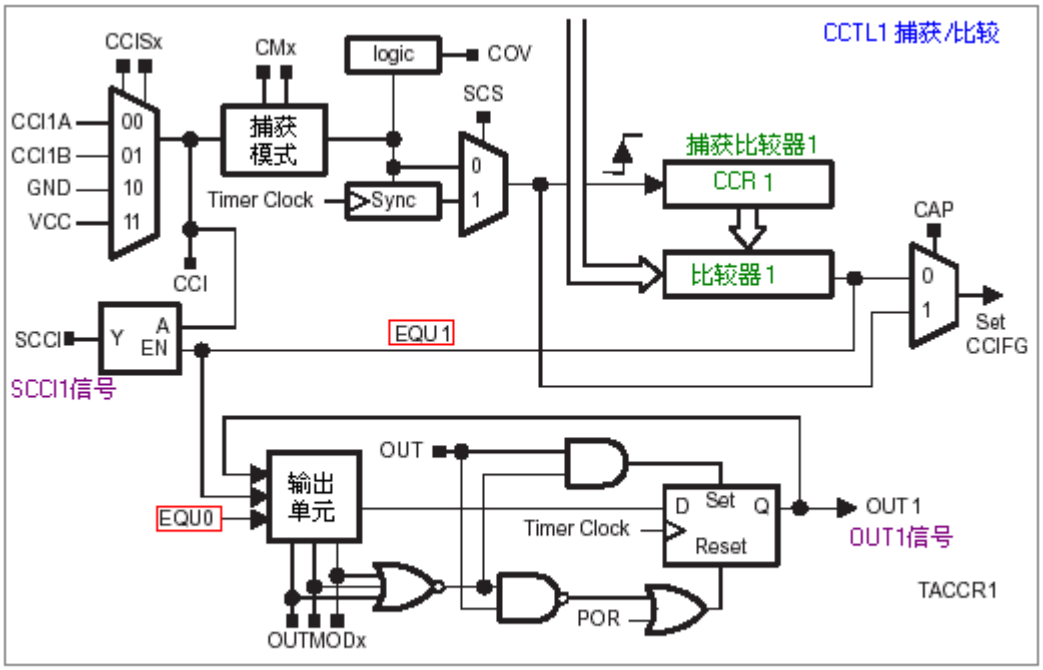
各种定时器功能

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

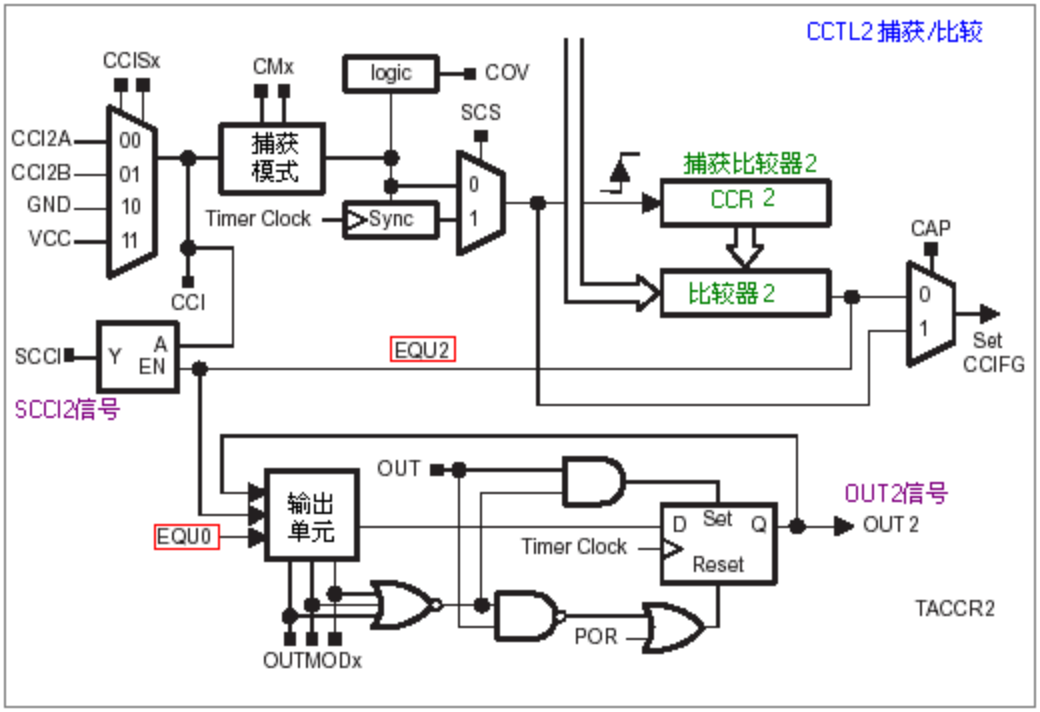
定时器 A



Timer_A第一部分



Timer A第二部分



Timer A 第三部分

Timer_A 的寄存器

寄存器	缩写	读定类型	地址	初态
Timer_A 控制寄存器	**TACTL**	**R/W**	**160H**	**POR** 复位
Timer_A 计数器	**TAR**	**R/W**	**170H**	**POR** 复位
捕捞/比较控制寄存器 **0**	**CCTL0**	**R/W**	**162H**	**POR** 复位
捕捞/比较寄存器 **0**	**CCR0**	**R/W**	**172H**	**POR** 复位
捕捞/比较控制寄存器 **1**	**CCTL1**	**R/W**	**164H**	**POR** 复位
捕捞/比较寄存器 **1**	**CCR1**	**R/W**	**174H**	**POR** 复位
捕捞/比较控制寄存器 **2**	**CCTL2**	**R/W**	**166H**	**POR** 复位
捕捞/比较寄存器 **2**	**CCR2**	**R/W**	**176H**	**POR** 复位
中断向量寄存器	**TAIV**	**R/W**	**12EH**	**POR** 复位

由于 **MSP430** 系列单片机不同系列可能包含不同数目的捕获/比较器，这里列出含有 **3** 个捕获/比较器 **Timer_A** 的寄存器。

在 **MSP430** 系列单片机中带有功能强大的定时器资源，这定时器在单片机应用系统中起到重要的作用。 利用 **MSP430**（以下称为 **430**）单片机的定时器可以用来实现计时，延时，信号频率测量，信号触发检测，脉冲脉宽信号测量，**PWM** 信号发生。另外通过软件编写可以用作串口的波特率发生器。后面我们将用定时器 **A** 作为一个波特率发生器，来编写一个串口例程给初学者参考。以加强初学者对定时器 **A** 的理解和应用。

在 **430** 的大系列产品中，不同的子系列产品定时器资源有所不同；在 **F11X**，**F11X1**

中是不带定时器 B 资源的。430 的定时器主要分为 3 部分模块：看门狗定时器，定时器 A，定时器 B。定时器 A 主要资源特点有 16 位定时计数器，其计数模式有 4 种。多种计数时钟信号供选择。3 个可配置输入的捕获/比较功能寄存器和 8 种输出模式的 3 个可配置输出单片。以上各块定时器资源可作多种组合使用，以实现强大的功能。

定时器资源功能说明

(1)看门狗定时器(WDT)：主要用于程序在生错误时用作单片机系统复位重起的。另外，也可作为一个基本定时器使用。

(2)定时器 A：作基本定时器使用，结合捕获/比较功能模块可实现时序控制，可编程波形信号发生输出。可作串口波特率发生器使用。

(3)定时器 B：作基本定时器使用，与定时器 A 基本相同,但是功能方面有某些功能会比 A 增强些。详情请看关于定时器 B 应用范例。

定时器_A(三个比较/捕获寄存器)

定时器_A 模块资源：一个 16 位计数器 TAR、三个捕获/比较寄存器 CCRx 三个捕获/比较控制寄存器 CCTLx

工作模式：

停止模式

增计数模式：定时计数器增到 CCR0。（你可在此期间设置 CCRx 来产生中断标记，但计数是一定走到 CCR0 后在循环进行，以下也一样）。

连续计数模式：从 0 到 65535 连续增计数模式。

增/减模式：先增 CCR0 后减至 0 模式。（当条件满足后中断标志置位）该工作模式可通过控制寄存器 TACTL 中的 MC1 和 MC0 选择计数器 TAR：就是存放定时器的计数值，虽然它只有 16 位寄存器，但可设置为 8、10、14 位的计数长度。该定数器是该模块的核心，它与包括定时器、捕获/比较模块打交道。如用于比较时要于它比较，捕获时是将它的值捕获保存。可用字进行读写。

比较模式：

这是该定时 器的默认模式，在此所有的捕获硬件停止工作。如果此时相应定时器中断允许打开的话，同时开始启动定时器，定时计数器 TAR 中的数值等于比较寄存器的值时，则产生中断请求。如没有中断允许，只是响应的中断标志 CCIFGx 置位。同时 EQUx 信该号位为真。否则为假利用它可以控制输出产生占空比可变的 PWM 波形输出。

当选用了比较模式时：

比较模式常用在用软件设定时中断间隔，来处理有关的事情，如 键盘扫描、事件查询处理、也可结合输出产生脉冲时序发生信号，PWM 信号等。

一个典型的例子就是利用不断的装载到 TxCCRx 中的数据与 TAR 的值比较来产生中断处理信号。

捕获模式：

主要用于利用信号的正沿、负沿或正负沿的任一组合，测量外部或内部事件，也

可以由软件停止。外部触发事件可以用 CCISx 选择 CCIxA，CCIxB,GND,和 Vcc 源。完成捕获后相应的中断标志 CCIFGx 置为位。

捕获模式应用：

捕获是当外部有信号进来后(触发)，将定时器 TxR 的值捕获到自己的锁存寄存器 TxCCRx 中,你可以随时读出。 TxCCRx 为 16 位可读可写。捕获模式用于事件的精确定位，它可用在速度（或频率）或时间测量中。一个典型的例子就是通过两次捕获外部事件来获得外部脉冲信号的宽度。

最简单的例子是：捕获模式选择任意沿，CCISx1=1,变化 CCISx0（及输入选择 Vcc 和 GND），这样在发生 Vcc 与 GND 切换（有高低电平触发时）产生捕获条件。

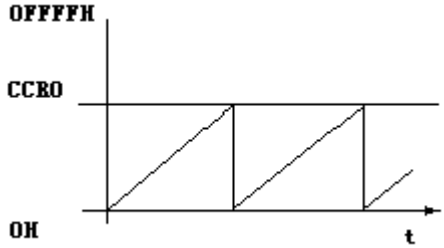
异步通讯：

再一个例子就是同时应用比较模式和捕获模式用软件实现 UART 通讯。即利用定时器比较模式来模拟通讯时序的波特率来发送数据，同时采用捕获模式来接收数据，并及时转换比较模式来选定调整接收波特率，达到接收一个字节的目的。

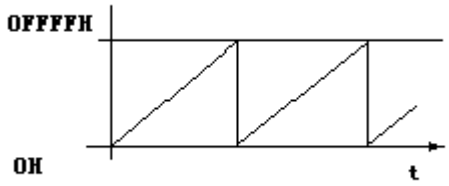
时钟源选择:

它是通过定时控制器 TACTL 中的两位完成，当从新上电或发生 POR 时（系统复位）或用软件通过 CLR 位使分频器复位。在正常操作时分频器的状态是不可见的。

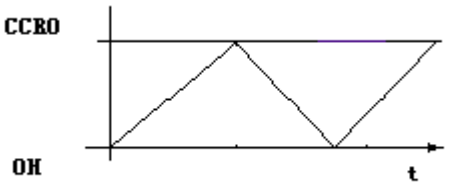
工作模式的选择：



增计数模式：当计数器增计数到 CCR0 时捕获比较中断 CCIFG 标志置位。



连续计数模式：从 0 开始到 0FFFFH 在从 0 开始，当回到 0 时 TAIFG 置位（ TAIFG 为溢出标志）但并不中断,只有 TAIE=1 时才产生中断。可以设置不同的 CCR0 的值来产生中断 CCIFG。



增/减计数模式：

增减计数模式，当增到 CCR0 时 CCIF0 置位，当减到 0 时 TAIFG 置位（ TAIFG 为溢出标志）

TACTL 控制寄存器

15--10	9	8	7	6	5	4	3	2	1	0
未用	SSEL1	SSEL0	ID1	ID0	MC1	MC0	未用	CLR	TAIE	TAIFG

SSEL1、SSEL0 选择定时器输入分频器的时钟源

Timer_A 时钟源

SSEL1	SSEL0	输入时钟源	说明
0	0	TACLK	用特定的外部引脚信号
0	1	ACLK	辅助时钟
1	0	SMCLK	子系统时钟
1	1	INCLK	见器件说明

ID1，ID0 输入分频选择

00 不分频
01 2 分频
10 4 分频
11 8 分频

MC1，MC0 计数模式控制位

00 停止模式
01 增计数模式
10 连续计数模式
11 增/减计数模式

CLR 定时器清除位

POR 或 CLR 置位时定时器和输入分频器复位。CLR 由硬件自动复位，其读出始终为 0。定时器在下一个有效输入沿开始工作。如果不是被清除模式控制暂停，则定时器以增计数模式开始工作。

TAIE 定时器中断允许位

0 禁止定时器溢出中断

1 允许定时器溢出中断

TAIFG 定时器溢出标志位

增计数模式：当定时器由 CCR0 计数到到 0 时，TAIFG 置位。

连续计数模式：当定时器由 0FFFFH 计数到 0 时，TAIFG 置位。

增/减计数模式：当定时器由 CCR0 减计数到 0 时，TAIFG 置位。

TAR 16 位计数器

15--0

这是计数器的主体，内部可读写。

[1]修改 TIMWER_A：当计数时钟不是 MCLK 时，写入应该在计数器停止计数时写，因为它与 CPU 时钟不同步，可能引起时间竞争。

[2]TIMER_A 控制位的改变：如果用 TACLK 控制寄存器中的控制位来改变定时器工作，修改时定器应停止，特别是修改输入选择位、输入分频器和定时器清除位时。输入时钟和软件所用的系统时钟异步可能引起时间竞争，使定时器响应出错。

CCTLx 捕获/比较控制寄存器

15 14	13 12	11	10	9	8	7 6 5	4	3	2	1	0
CAPTMOD1-0	CCIS1-0	SCS	SCCIx		CAP	OUTMODx	CCIEx	CCIx	OUT	COV	CCIFx

TIMER_A 有多个捕获比较模块，每个模块都有自己的控制寄存器 CCTLx

CAPTMOD1-0 选择捕获模式

00 禁止捕获模式

01 上升沿捕获

10 下降沿捕获

11 上升沿与下降沿都捕获

CCISI1-0 在捕获模式中用来定提供捕获事件的输入端

00 选择 CCIxA

01 选择 CCIxB

10 选择 GND

11 选择 VCC

SCS 选择捕获信号与定时器时钟同步、异步关系

0 异步捕获

1 同步捕获

异步捕获模式允许在请求时立即将 CCIFG 置位和捕获定时器值，适用于捕获信号的周期远大于定时器时钟周期的情况。但是，如果定时器时钟和捕获信号发生时间竞争，则捕获寄存器的值可能出错。

在实际中经常使用同步捕获模式，而且捕获总是有效的。

SSCIx 比较相等信号 **EQUx** 将选择中的捕获、比较输入信号 **CCIx(CCIxA,CCIxB,Vcc** 和 **GND)**进行锁存，然后可由 **SCCIx** 读出。

CAP 选择捕获模式还是比较模式。

0 比较模式

1 捕获模式

注意：同时捕获和捕获模式选择

如果通过捕获比较寄存器 **CCTLx** 中的 **CAP** 使工作模式从比较模式变为捕获模式，那么不应同时进行捕获；否则，在捕获比较寄存器中的值是不可预料的，推荐的指令顺序为：**[1]**修改控制寄存器，由比较模式换到捕获模式。

[2]捕获

OUTMODx 选择输出模式

000 输出

001 置位

010 PWM 翻转/复位

011 PWM 置位/复位

100 翻转/置位

101 复位

110 PWM 翻转/置位

111 PWM 复位/置位

定时器时钟上升沿时 **OUTx** 在各模式下的状态

输出模式	EQU0	EQUx	OUTx 状态(或触发器输入端 D)
0	X	X	X(OUTx 位)
1	X	0	OUTx(不变)
	X	1	1(置位)
2	0	0	OUTx(不变)
	0	1	/OUTx(与以前相反)
	1	0	0
	1	1	1(置位)
3	0	0	OUTx(不变)
	0	1	1(置位)
	1	0	0
	1	1	1(置位)
4	X	0	OUTx(不变)
	X	1	/OUTx(与以前相反)
5	X	0	OUTx(不变)
	X	1	0
6	0	0	OUTx(不变)

	0	1	/OUTx(与以前相反)
	1	0	1
	1	1	0

CCIx 捕获比较模的输入信号

捕获模式：由 **CCIS0** 和 **CCIS1** 选择的输入信号通过该位读出。

比较模式：**CCIx** 复位。

OUT 输出信号

0 输出低电平

1 输出高电平

如果 **OUTMODx** 选择输出模式 **0**(输出)，则该位对应于输入状态。

COV 捕获溢出标志

0 输出低电平

1 输出高电平

[1]当 **CAP=0** 时，选择比较模式。捕获信号发生复位，没有使 **COV** 置位的捕获事件。

[2]当 **CAP=1** 时，选择捕获模式，如果捕获寄存器的值被读出再次发生捕获事件，则 **COV** 置位。程序可检测 **COV** 来断定原值读出前是否又发生捕获事件。读捕获寄存器时不会使溢出标志复位，须用软件复位。

CCIFGx 捕获比较中断标志

捕获模式：寄存器 **CCRx** 捕获了定时器 **TAR** 值时置位。

比较模式：定时器 **TAR** 值等于寄存器 **CCRx** 值时置位。

CCRx 捕获/比较寄存器

15--0

在捕获比较模块中，可读可写。其中 **CCR0** 经常用作周期寄存器，其他 **CCRx** 相同。

TAIV 定器器 A 中断向量寄存器

15--5	4--1	0
0--0	中断向量	0

Timer_A 有两个中断向量，一个单独分配给捕获比较寄存器 **CCR0**，另一个作为共用的中断向量用于定时器和其他的捕获比较寄存器。

CCR0 中断向量具有最高的优先级，因为 **CCR0** 能用于定义是增计数和增减计数模式的周期。因此，他需要最快速度的服务。**CCIFG0** 在被中断服务时能自动复位。

CCR1-CCRx 和定时器共用另一个中断向量，属于多源中断，对应的中断标志 **CCIFG1-CCIFGx** 和 **TAIFG1** 在读中断向量字 **TAIV** 后，自动复位。如果不访问 **TAIV** 寄

存器，则不能自动复位，须用软件清除；如果相应的中断允许位复位(不允许中断)，则将不会产生中断请求，但中断标志仍存在，这时须用软件清除。

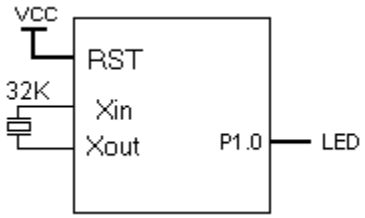
关于中断挂起和返回不包括处理约需要 **11~16** 个时钟周期。

TIMER_A 中断优先级

中断优先级	中断源	缩写	**TAIV** 的内容
最高	捕获/比较器 **1**	**CCIFG1**	**2**
	捕获/比较器 **1**	**CCIFG1**	**4**
	捕获/比较器 **x**	**CCIFGx**	
最低	定时器溢出	**TAIFG1**	**10**
	没有中断将挂起		**0**

例 1

定时器 A 自动溢出



简述： 利用 Timer_A 直接计数产生溢出，然后中断处理。
ACLK = TACLK = 32768Hz, MCLK = SMCLK = default DCO ~800kHz

例程：

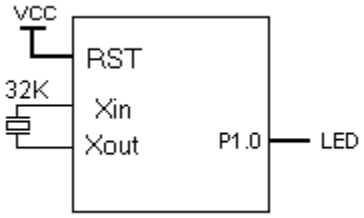
```
//*******************************************************************
#include   <msp430x14x.h>
void main(void)
 {
   WDTCTL = WDTPW + WDTHOLD;              //停止看门狗定时器
   P1DIR |= 0x01;                         // 设 P1.0 为输出
   TACTL = TASSEL_1 + MC_2 + TAIE;        // ACLK, 定时器 A 计数模式,且开中断功能

   _BIS_SR(LPM3_bits + GIE);              //进入 LPM3 低功耗模式和开总中断允许
 }

// Timer_A3 中断向量 (TAIV)处理程序
#pragma vector=TIMERA1_VECTOR
__interrupt void Timer_A(void)
 {
   switch( TAIV )                         //应用 switch 语句来处理多中断源的向量
   {                                      //向量列表通过 case 语句来分多中断源的入口
     case   2:   break;                   // CCR1 比较/捕获寄存器的中断入口，
                                          //本例子未用到。
     case   4:   break;                   // CCR2 比较/捕获寄存器的中断入口，
```

```
                                        //本例子未用到。

    case 10:    P1OUT ^= 0x01;          //定时器 A 在计满 16 位时将产生计数溢出中断
                                        //进入中断，向量值为 10。此中断处理入口
                break;
  }
 }
```

//**

例 2

CCR0 捕获/比较寄存器--比较功能中断定用



简述：本程序是利用了定时器 A 模块中的 CCR0 捕获/比较寄存器的值与定时器 A 的值进行比较。如果比较相等就产生 CCR0 中断，进入 Timer A0 中断服务程序进行中断处理。以 P1.0 作为指示，反转速度= 32768/(2*1000) = 16.384 。

ACLK = TACLK = 32768Hz, MCLK = SMCLK = default DCO ~800kHz

本程序 TA(Timer A)采用增计数模式。在增计数模式时，当 TA 中的计数值与 CCR0 的值相等时(或 TA>CCR0 的值时)，TA 被清零并且重新由 0 开始计数。在此同时，CCR0 产生中断。产生 CCIFG0 标志置位，及 TAIFG 定时器 A 溢出标志置位。CCIFG0，TAIFG 在被中断服务程序处理时是自动复位。如本例中执行__interrupt void Timer_A (void)时，CCIFG0 已被清除标志位。

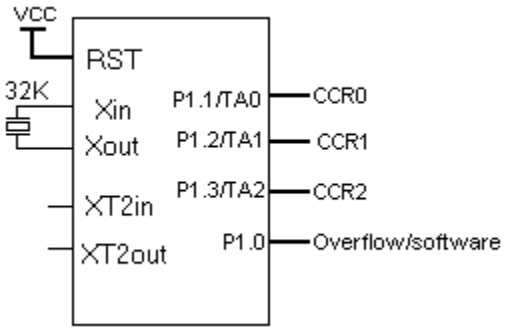
例程：

```
//**********************************************************************
#include   <msp430x14x.h>
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;          //停止看门狗定时器
  P1DIR |= 0x01;                     //设 P1.0 为输出
  CCTL0 = CCIE;                      //CCR0 开中断允许
  CCR0 = 1000-1;                     //向 CCR0 捕获/比较寄存器装入初值，用于比较
  TACTL = TASSEL_1 + MC_1;           //选 ACLK 为定时器 A 时钟源, 增计数模式

  _BIS_SR(LPM3_bits + GIE);          //进入 LPM3 模式/ 开中断允许
}

// Timer A0 中断服务程序
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)
{
  P1OUT ^= 0x01;                     //反转 P1.0
```

```
}
//*********************************************************************
```

例 3

CCR0 捕获/比较寄存器--比较功能中断定用



简述： ACLK = TACLK = 32kHz, MCLK = SMCLK = default DCO ~800kHz

实现输出

P1.1= CCR0 = 32768/(2*4) = 4096Hz

P1.2= CCR1 = 32768/(2*16) = 1024Hz

P1.3= CCR2 = 32768/(2*100) = 163.84Hz

P1.0= overflow = 32768/(2*65536) = 0.25Hz

此程序中主要用到了 CCR0，CCR1，CCR0 捕获/比较寄存器的值与 TA 的值进行比较。TA 的计数模式为连续计数。此模式适合产生多个不同时序信号，只要将改变 CCRX 中的值就可以。在本例中，CCRX 每次中断里增长了下一次中断发生时的时差值。另外，还需设置 TA 模块输出单元 OUTx 的输出模式，本例设置的输出模式为反转模式。

需要注意的，CCR0 是一个单独分配的中断向量。而 CCR1-CCRx 和 TA 的中断向量是共用的，这是一个多源中断。CCIFG1-CCIFGx 和 TAIFG1 中断标志在读中断向量 TAIV 后，硬件会自动复位这些标志；CCR0 中断服务程序执行后 CCIFG0 标志也被复位。

CCR0 和 CCR1-CCRx，TA 的中断服务程序是分开的。

__interrupt void Timer_A0(void)

__interrupt void Timer_A1(void)

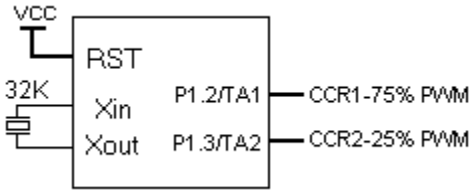
例程：

```
//*********************************************************************
#include   <msp430x14x.h>
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 //停止看门狗定时器
  P1SEL |= 0x0E;                            //设 P1.1 - P1.3 为模块功能
  P1DIR |= 0x0F;                            //设 P1.0 - P1.3 方向为输出
  CCTL0 = OUTMOD_4 + CCIE;                  // CCR0 输出为反转模式, 开中断允许
  CCTL1 = OUTMOD_4 + CCIE;                  // CCR1 输出为反转模式, 开中断允许
  CCTL2 = OUTMOD_4 + CCIE;                  // CCR2 输出为反转模式, 开中断允许
  TACTL = TASSEL_1 + MC_2 + TAIE;           // ACLK, 连续计数模式, 开 TA 中断允许

  _BIS_SR(LPM3_bits + GIE);                 //进入 LPM3/开总中断允许
}
```

```
// Timer A0 中断服务程序
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A0 (void)
{
   CCR0 += 4;                                  //向 CCR0 增加偏移量
}

// Timer_A3 中断服务程序
#pragma vector=TIMERA1_VECTOR
__interrupt void Timer_A1(void)
{
   switch( TAIV )
   {
   case   2: CCR1 += 16;                       //向 CCR1 增加偏移量
             break;
   case   4: CCR2 += 100;                      //向 CCR2 增加偏移量
             break;
   case 10: P1OUT ^= 0x01;                     // Timer_A3 定时器 A 溢出
             break;
  }
}
//*****************************************************************************
```

例 4

CCR1 CCR2 捕获/比较寄存器—产生 PWM 波形



简述：利用定时器 A 的自动产生 PWM 波形输出，而无需采用软件程序来服务生产。本程序 CCR0 的值作为 PWM 的周期宽度，而 CCR1，CCR2 的值作为 PWM 的占空宽。适当修改软件可以容易实现动态 PWM 输出或外加少量的 RC 元件要以实现 DAC 功能。

ACLK = TACLK = LFXT1 = 32768Hz, MCLK = default DCO ~800kHz.

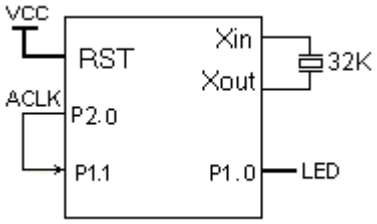
例程：

```
//*****************************************************************************
#include   <msp430x14x.h>
void main(void)
{
   WDTCTL = WDTPW + WDTHOLD;                   //停止看门狗定时器
   P1DIR |= 0x0C;                              //设 P1.2 和 P1.3 为输出
   P1SEL |= 0x0C;                              //设 P1.2 和 P1.3 TA1/2 为模块功能

   CCR0 = 512-1;                               //装入 PWM 周期值

   CCTL1 = OUTMOD_7;                           //设 CCR1 输出单元为复位/置位输出模式
   CCR1 = 384;                                 //装入 CCR1 PWM 的占空值

   CCTL2 = OUTMOD_7;                           //设 CCR2 输出单元为复位/置位输出模式
```

```
   CCR2 = 128;                               //装入 CCR2 PWM 的占空值

   TACTL = TASSEL_1 + MC_1;                  //时钟源选择 ACLK, 定时器 A 增计数模式

   _BIS_SR(LPM3_bits);                       //进入 LPM3 低功耗模式
}

//*************************************************************************************
```

例 5

定时器 A 模块的 CCR0 捕获/比较寄存器的捕获功能应用



简述：利用 CCR0 捕获/比较寄存器的捕获功能对 ACLK/8 信号进行捕获，分别捕获 16 次数据存储到数组中。在每次捕获到数据时都用新数据对旧数相比较，并将两数据的差异数据存入数组中，同时保存新数据。完成 16 次数据捕获后 LED 取反。

例程：

```
//*************************************************************************************
#include    <msp430x14x.h>
unsigned int new_cap=0;
unsigned int old_cap=0;
unsigned int cap_diff=0;

unsigned int diff_array[16];                 // 差异数组
unsigned int capture_array[16];              // 捕获数组
unsigned char index=0;
unsigned char count = 0;

void main(void)
{
   volatile unsigned int i;
   WDTCTL = WDTPW + WDTHOLD;                 //停止看门狗定时器
   for (i=0; i<20000; i++)                   //延时，等待晶体振荡器称定
   {}
   P1DIR = 0x01;                             //置 P1.0 为输出,P1.1 为输入
   P1OUT &= ~0x01;                           // LED 关
   P1SEL = 0x02;                             //设 P1.1 为 TA0 功能
   P2DIR = 0x01;                             //设 P2.0 方向为输出
   P2SEL |= 0x01;                            //设 P2.0 为 ACLK 输出
   BCSCTL1 |= DIVA_3;                        //将 ACLK/8 分频

   CCTL0 = CM_1 + SCS + CCIS_0 + CAP + CCIE;
   //上升边+ 同步+ CCI0A (P1.1 信号源) + 捕获功能 + 捕获中断允许
```

```
   TACTL = TASSEL_2 + MC_2;                  // SMCLK 时钟+连续计数模式

   _BIS_SR(LPM0_bits + GIE);                     //进入 LPM0 +开总中断允许
}

#pragma vector=TIMERA0_VECTOR
__interrupt void TimerA0(void)
{
    new_cap = TACCR0;                          //将中断时 TACCR0 值装入 new_cap
    cap_diff = new_cap - old_cap;              //差异值=新值-旧值

    diff_array[index] = cap_diff;              //向差异数组写入差异值
    capture_array[index++] = new_cap;          //向捕获数组写入捕获值
    if (index == 16)                           //数组是否已满？(1-16)
    {
       index = 0;                              //清索引变量
       P1OUT ^= 0x01;                          //反转 P1.0
    }
    old_cap = new_cap;                         //存储捕获新值
    count ++;
    if (count == 32)
    {
       count = 0;
       _NOP();
    }

}

//*******************************************************************************
```

6-比较器 A

以下图可以看出比较器 **A** 的结构大概可以分 **4** 部分构成，分别为模拟输入，比较器 **A** 核心，低通虑波器，基准电压部分和中断部分组成。

首先，整个比较器 **A** 的工作必需由 **CAON** 位置为 **1** 时才能工作的，此位属 **CACTL1** 控制寄存器。单片机上电时此位是为 **0** 的，也就是说比较器是不工作的。

以下大概讲述几个部分电路的功能和一些相关信息。

模拟输入电路：

外部模拟引脚信号 **CA0**，**CA1(**正负端**)**可以分别由 **P2CA0,P2CA1** 位控制开或关。经过软件的设置可以分别与内部的几个基准电压进行比较（**0.5VCC,0.25VCC,**三极管门值电压）或外部其中的电压进行比较。

◎应用的硬件比较可以分为以下三种组合：

☐两个外部引脚输入信号进行比较

☐其中一个外部引脚信号与内部的 **0.5VCC** 或 **0.25VCC** 比较

☐其中一个外部引脚信号与内部基准电压比较

◎参考电压发生器

参考电压电路是可以由 **CARSEL**，**CARERF0**，**CARERF1** 位来控制电压的产生。通过软件设置可以选择几种电压输出到比较器的输入中作为比较，当然此参考电压也可以通过单片机的引脚往外部提供参考电压之用。

◎比较器 **A** 核心

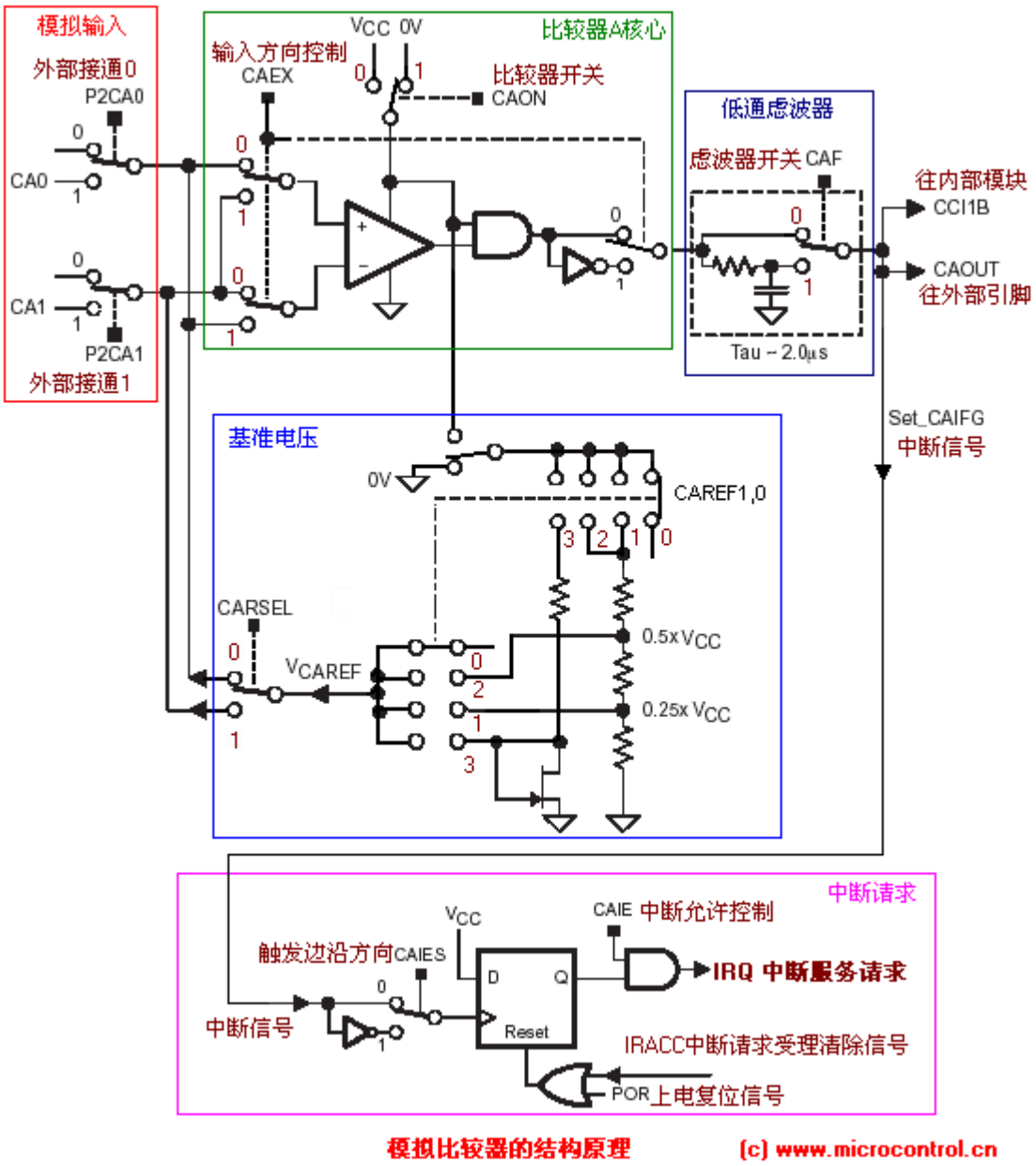
比较器 **CAON** 位控制开关，**CAEX** 位控制位控制方向。

◎低通虑波器

低通虑波器只需一个 **CAF** 位来控制此虑波器的功能开与关。此虑波器功能是用于消除比较器输出信号的毛刺，以保证信号的质量和中断请求的可靠性。

◎中断请求

比较器 **A** 模块是具有中断功能的，如比较器功能 **CAIE** 中断允许开了，在 **CAIF** 信号产生时将生产中断（当然 **GIE** 要为 **1** 时）。比较器 **A** 模块是具有中断独立向量的，是一个单独的中断，**CUP** 接受请求后会硬件自动清除中断标志位 **CAIFG**。



模拟比较器的结构原理 (c) www.microcontrol.cn

CACTL1 比较器 A 控制寄存器 1

7	6	5	4	3	2	1	0
CAEX	CARESL	CAREF1	CAREF0	CAON	CAIES	CAIE	CAIFG

CAEX 比较器的输入端,控制比较器 A 的输入信号和输出方向。

CARSEL 选择内部参考源加到比较器 A 的正端或负端。

CAEX 和 **CARSEL** 的含义

CARSE	CAEX	含义

0	0	内部参考源加到比较器的正端
	1	内部参考源加到比较器的负端
1	0	内部参考源加到比较器的负端
	1	内部参考源加到比较器的正端

CAREF1、CAREF0 选择参考源
0 使用外部参考源；
1 选择 **0.25Vcc** 为参考电压
2 选择 **0.5Vcc** 为参考电压
3 选择二极客电压为参考电压，必须见具体的芯片资料。

CAON 控制比较器 **A** 的打开和关闭
0 关闭比较器
1 打开比较顺

CAIES 中断触发沿选择
0 上升沿使中断标志 **CAIFG** 置位
1 下降沿使中断标志 **CAIFG** 置位

CAIE 中断允许
0 禁止中断
1 允许中断

CAIFG 比较器中断标志
0 没有中断请求
1 有中断请求

CACTL2 比较器 A 控制寄存器 2

7	6	5	4	3	2	1	0
CACTL2.7	CACTL2.6	CACTL2.5	CACTL2.4	P2CA1	P2CA0	CAF	CAOUT

CACTL2.7—2.4 含义请参见具体的芯片资料，例如，在 **MSP430X1XX** 系列中，这位可以被执行，但不控制任何硬件，可被用作标志位。

P2CA1 控制输入端 **CA1**
0 外部引脚信号不连接比较器 **A**
1 外部引脚信号连接比较器 **A**

P2CA0 控制输入端 **CA0**
0 外部引脚信号不连接比较器 **A**
1 外部引脚信号连接比较器 **A**

CAF 选择比较器输出端是否经过 **RC** 低通滤波器
0 不经过
1 经过

CAOUT 比较器 **A** 的输出
0 CA0 小于 **CA1**
1 CA0 大于 **CA1**

CAPD 端口禁止寄存器

7--0

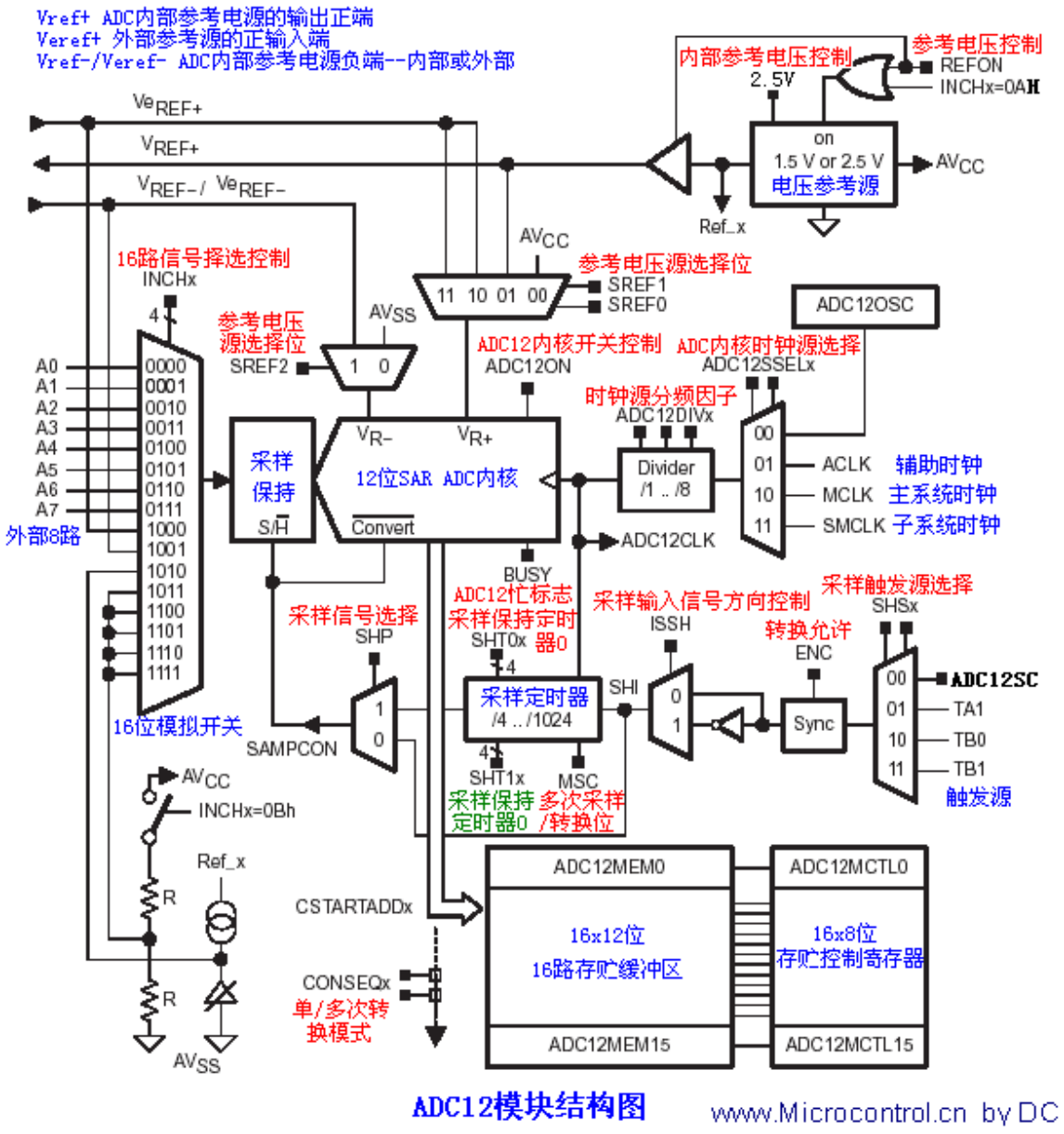
比较器 **A** 模块的输入输出与 **IO** 口共用引脚，**CAPD** 可以控制 **IO** 端口输入缓冲器的通断开关。当输入电压不接近 **Vss** 或 **Vcc** 时，**CMOS** 型的输入缓冲器可以起到分流作用。这样可以减少了由不是 **Vss** 或 **Vcc** 的输入电压所引起的流入输入缓冲器的电流。控制位 **CAPD0—CAPD7** 初始化为 **0**，则端口输入缓冲器有效。当相应控制位置 **1** 时，端口输入缓冲器无效。

程序范例：

```
#include <msp430x11x1.h>
void main (void)
  {
    WDTCTL = WDTPW + WDTHOLD;       // 停止 WDT
    CAPD |= 0x08;                   // 断开与 IO 端口输入
    CACTL2 = P2CA0;                 // 设置 P2.3 为+comp
    CCTL0 = CCIE;                   // CCR0 允许中断
    TACTL = TASSEL_2 + ID_3 + MC_2;  // SMCLK/8,计数模式
    _EINT();                        // 开总中断

  while (1)                         // 循环
  {
    CACTL1 = 0x00;                  // 没有参考电压
    _BIS_SR(LPM0_bits);             // 进入 LPM0
    CACTL1 = CAREF0 + CAON;          // 0.25*Vcc=P2.3, 比较器开
    _BIS_SR(LPM0_bits);             // 再次进入 LPM0
    CACTL1 = CAREF1 + CAON;          // 0.5*Vcc=P2.3, 比较器开
    _BIS_SR(LPM0_bits);             // 再次进入 LPM0
    CACTL1 = CAREF1 + CAREF0 + CAON;   // 0.55V on P2.3,比较器开
    _BIS_SR(LPM0_bits);             // 再次进入 LPM0
   }
 }

// Timer A0 interrupt service routine
```

```
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)
{
   _BIC_SR_IRQ(LPM0_bits);                    //退出 LMP0 模式
}
```

7-ADC12 模数转换模块



ADC12模块结构图 www.Microcontrol.cn by DC

MSP430 模数转换模块--ADC12

MSP430 单片机的 ADC12 模块是一个 12 位精度的 A/D 转换模块,他具有高速度,通用性等特点。大部分都内置了 ADC 模块.而有些不带 ADC 模块的片子,也可通过利用内置的模拟比较器来实现 AD 的

转换。在系列产品中，我们可以通过以下列表来简单地认识他们的 ADC 功能实现。

系列型号	ADC 功能实现	转换精度
MSP430X1XX2	Slope AD	10 位
MSP430F13X	ADC 模块	12 位
MSP430F14X	ADC 模块	12 位
MSP430F41X	Slope AD	12 位
MSP430F43X	ADC 模块	12 位
MSP430F44X	ADC 模块	12 位
MSP430X32X	ADC 模块	14 位
MSP430F21X	Slope AD	12 位

从以下 ADC12 结构图中可以看出，ADC12 模块中是由以下部分组成：输入的 16 路模拟开关，ADC 内部电压参考源，ADC12 内核，ADC 时钟源部分，采集与保持/触发源部分，ADC 数据输出部分，ADC 控制寄存器等组成。

输入的 16 路模拟开关

16 路模拟开关分别是由 IC 外部的 8 路模拟信号输入和内部 4 路参考电源输入及 1 路内部温度传感器源及 AVCC-AVSS/2 电压源输入。外部 8 路从 A0-A7 输入，主要是外部测量时的模拟变量信号。内部 4 路分别是 Veref+ ADC 内部参考电源的输出正端，Vref-/Veref- ADC 内部参考电源负端(内部/外部)。1 路 AVCC-AVSS/2 电压源和 1 路内部温度传感器源。片内温度传感器可以用于测量芯片上的温度，可以在设计时做一些有用的控制；在实际应用时用得较多。而其他电源参考源输入可以用作 ADC12 的校验之用，在设计时可作自身校准。

ADC 内部电压参考源

ADC 电压参考源是用于给 ADC12 内核作为一个基准信号之用的，这是 ADC 必不可少的一部分。在 ADC12 模块中基准电压源可以通过软件来设置 6 种不同的组合。AVCC(Vr+)，Vref+,Veref+,AVSS(Vr-),Vref-/Veref-。

ADC12 内核

ADC12 的模块内核是共用的，通过前端的模拟开关来分别来完成采集输入。ADC12 是一个精度为 12 位的 ADC 内核，1 位非线性微分误差，1 位非线性积分误差。内核在转换时会参用到两个参考基准电压，一个是参考相对的最大输入最大值，当模拟开关输出的模拟变量大于或等于最大值时 ADC 内核的输出数字量为满量程，也就是 0xfff；另一个则是最小值，当模拟开关输出的模拟变量大小或等于最大值时 ADC 内核的输出数字量为最低量程，也就是 0x00。而这两个参考电压是可以通过软件来编程设置的。

ADC 时钟源部分

ADC12 的时钟源分有 ADC12OSC，ACLK，MCLK，SMCLK。通过编程可以选择其中之一时钟源，同时还可以适当的分频。

采集与保持,触发源部分

ADC12 模块中有着较好的采集与保持电路，采用不同的设置有着灵活的应用。

ADC 数据输出部分

ADC 内核在每次完成转换时都会将相应通道上的输出结果存贮到相应用通道缓冲区单元中，共有 16 个通道缓冲单元。同时 16 个通道的缓冲单元有着相对应的控制寄存器，以实现更灵活的控制。

ADC12 模块的所有寄存器

寄存器	寄存器缩写	寄存器含义
转换控制寄存器	**ADC12CTL0**	转换控制寄存器 **0**
	ADC12CTL1	转换控制寄存器 **1**
中断控制寄存器	**ADC12IFG**	中断标志寄存器
	ADC12IE	中断使能寄存器
	ADC12IV	中断向量寄存器

存储及其控制寄存器	ADC12MCTL0-ADC12MCTL15	存储控制寄存器 0-15
	ADC12MEM0-ADC12MCTL15	存储寄存器 0-15

ADC12CTL0 转换控制寄存器 0

15--12	11--8	7	6	5	4	3	2	1	0
SHT1	SHT0	MSC	2.5V	REFON	ADC12ON	ADC12TOVIE	ADC12TVIE	ENC	ADC12SC

ADC12SC 采集/转换控制位
在不同条件 **ADC12SC** 的含义.

ENC=1	SHP=1	ADC12SC 由 0 变为 1 启动 AD 转换
		AD 转换完成后 ADC12SC 自动复位
ISSH=0	SHP=0	ADC12SC 保持高电平时采集
		ADC12SC 复位时启动一次转换

ENC=1 表示转换允许(必须使用)；**ISSH=0** 表示采要输入信号为同相输入(推荐使用)；**SHP=1** 表示采样信号 **SAMPCON** 来源于采样定时器；**SHP=0** 表示采样直接由 **ADC12SC** 控制。使用 **ADC12SC** 时，需注意以上表格信号的匹配。用软件启动一次 **AD** 转换，需要使用一条指令来完成 **ADC12SC** 与 **ENC** 的设置。

ENC 转换允许位
0 ADC12 为初始状态，不能启动 AD 转换
1 首次转换由 SAMPCON 上升沿启动
只有在该位为高电平时，才能用软件或外部信号启动转换。在不同转换模式，**ENC** 由高电平变为低电平的影响不同：
□ 当 **CONSEQ=0** (单通道单次转换模式)且 **ADC12BUSY=1**(ADC12 处于采样或者转换)时，中途撤走 **ENC** 信号(高电平变为低电平)，则当前操作结束，并可能得到错误结果。所以在单通道单次转换模式整个过程中，都必须保证 **ENC** 信号有效。
□ 当 **CONSEQ=0**(非单通道单次转换)时，**ENC** 由高电平变为低电平，则当前转换正常结束，且转换结果有效，在当前转换结束时停止操作。

ADC12TVIE 转换时间溢出中断允许位
0 没发生转换时间溢出
1 发生转换时间溢出
当前转换还没有完成时，又发生一次采样请求，则会发生转换时间溢出。如果允许中断，则会发生中断请求。

ADC12OVIE 溢出中断允许位
0 没有发生溢出
1 发生溢出
当 **ADC12MEMx** 中原有的数据还没有被读出，而现在又有新的转换结果数据要写入时，则会发生溢出。如果相应的中断允许，则会发生中断请求。

ADC12ON ADC12 内核控制位
0 关闭 **ADC12** 内核
1 打开 **ADC12** 内核

REFON 参考电压控制位
0 内部参考电压发生器关闭
1 内部参考电压发生器打开

2.5V 内部参考电压的电压值选择位
0 选择 **1.5V** 内部参考电压
1 选择 **2.5V** 内部参考电压

MSC 多次采样/转换位

有效条件	MSC 值	含义
SHP=1	0	每次转换需要 SHI 信号的上升沿触发采集定时器
CONSE !=0	1	仅首次转换同 SHI 信号的上升沿触发采样定时器,而后采样转换将在前一次转换完成立即进行

其中 **CONESQ≠0** 表示当前转换模式不是单通道单次转换。

SHT1，SHT0 采集保持定时器 **1**，采样保持定时器 **0**
这是定义了每通道转换结果中的转换时序与采样时钟 **ADC12CLK** 的关系。采样周期是 **ADC12CLK** 周期的整 **4** 倍，则：

$$Tsample = 4 \times T_{adc12clk} \times N$$

SHT1,SHT0 采样保持定时器 **1**,采样保持定时器 **0** 的分频因子

SHITx	0	1	2	3	4	5	6	7	8	9	10	11	12--15
N	1	2	4	8	16	24	32	48	64	96	128	192	256

ADC12CTL2 转换控制寄存器 2

15-12	11-10	9	8	7-5	4,3	2,1	0
CSSTARTADD	SHS	SHP	ISSH	ADC12DIV	ADC12SSEL	CONSEQ	ADC12BUSY

大多数位只有在 **ENC=0** 时才可被修改，如 **3-15** 位。

CSSTARTADD 转换存储器地址位,这 **4** 位表示二进制数 **0-15** 分别对应 **ADC12MEM0-15**。可以定义单次转换地址或序列转换的首地址。

SHS 采样触发输入源选择位

0 ADC12SC
1 Timer_A.OUT1
2 Timer_B.OUT0
3 Timer_B.OUT1

SHP 采样信号(SAMPCON)选择控制位
0 SAMPCON 源自采样触发输入信号
1 SAMPCON 源自采样定时器，由采样输入信号的上升沿触发采样定时器

ISSH 采样输入信号方向控制位
0 采样输入信号为同向输入
1 采样输入信号为反向输入

ADC12DIV ADC12 时钟源分频因子选择位，分频因子为 3 位二进制数加 1

ADC12SEL ADC12 内核时钟源选择
0 ADC12 内部时钟源：ADC120SC
1 ACLK
2 MCLK
3 SMCLK

CONSEQ 转换模式选择位
0 单通道单次转换模式
1 序列通道单次转换模式
2 单通道多次转换模式
3 序列通道多次转换模式

ADC12BUSY ADC12 忙标志位
0 表示没有活动的操作
1 表示 ADC12 正处于采样期间、转换期间或序列转换期间。

ADC12BUSY 只用于单通道单次转换模式，如果 ENC 复位，则转换立即停止，转换结果不可靠，需要在使 ENC=0 之前，测试 ADC12BUSY 位以确定是否为 0。在其它转换模式下此位是无效的。

ADC12MEM0-ADC12MEM15 转换存储器

15	14	13	12	11-0		
0	0	0	0	MSB		LSB

这 16 位寄存器是用来存储 AD 转换结果，只用其中低 12 位，高 4 位在读出时为 0。

ADC12MCTLx 转换存储器控制寄存器

7	6,5,4	3,2,1,0

EOS	SREF	INCH

EOS 序列结束控制位

0 序列没有结束

1 此序列中最后一次转换

SREF 参考电压源选择位

0 V_{R+} = A_{VCC} ， V_{R-}=A_{VSS}

1 V_{R+} = A_{REF+} ， V_{R-}=A_{VSS}

2,3 V_{R+} = A_{eREF+} ， V_{R-}=A_{VSS}

4 V_{R+} = A_{VCC} ， V_{R-}=V_{REF-}/V_{eREF-};

5 V_{R+} = V_{REF+} ， V_{R}-=V_{REF-}/ V_{eREF-};

6,7 V_{R+} = A_{eREF+} ， V_{R}-=V_{REF-}/ V_{eREF-};

INCH 选择模拟输入通道。用 **4** 位二进制码表示输入通道

0-7 A0-A7

8 V_{eREF+}

9 V_{eREF-}/V_{eREF-}

10 片内温度传感器的输出

11-15 (A_{VCC}-A_{VSS}) /2

中断控制寄存器：**ADC12IFG**、**ADC12IE**、**ADC12IV**

ADC12IFG 中断标志寄存器

15	14	……	1	0
ADC12IFG.15	ADC12IFG.14	……	ADC12IFG.1	ADC12IFG.0

ADC12IFG.x = 1 转换结束，并且转换结果已经装入转换存储器

ADC12IFG.x = 0 ADC12MEMx 被访问

ADC12IE 中断使能寄存器

15	14	……	1	0
ADC12IE.15	ADC12IE.14	……	ADC12IE.1	ADC12IE.0

ADC12IE.x= 1 允许相应的中断标志位 **ADC12IFG.x** 在置位时发生的中断请求服务

ADC12IE.x=0 禁止相应的中断标志位 **ADC12IFG.x** 在置位时发生的中断请求服务

ADC12IV 中断向量寄存器

ADC12 是一个多源中断：有 **18** 个中断标志**(ADC12IFG.0**—**ADC12IFG.15**、**ADC12TOV**、**ADC120V)**但只有一个中断向量。

ADC12 各中断标志对应的 ADC12IV 值																ADC12TOV	ADC120V	ADC12IV
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

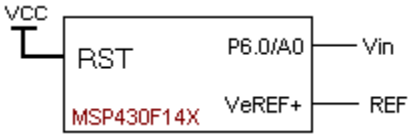
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	4
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	6
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	8
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	10
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	14
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	16
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	18
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	20
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	22
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	24
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	26
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	28
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	30
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36

优先级从高到低，分别为：数据溢出标志 **ADC12OV**，时间溢出中断标志 **ADC12TOV…ADC12IFG.15**。各中断标志产生一个 **0-36** 的偶数。**0** 表示没有中断标志。

ADC12OV、**ADC12TOV** 会在访问后 **ADC12IV** 后自动复位，但 **ADC12IFG.0--ADC12IFG.15** 对应中断服务程序响应后，其标志不自动复位，用以保证能处理发生溢出的情况。因为如在 **ADC12IFG.x** 未复位，也就是说此时转换结果的数据还未被读出。而又有新的转换数据写入 **ADC12EME** 时，会发生溢出，所以 **ADC12IFG.x** 需在用户软件中复位，或通过访问对应转换存储器 **ADC12MEMx** 自动复位。

例 1

ADC12-使用外部参考电压



简述：使用单通进行 ADC 转换，电压参考源来自外部。ADC12 的 Vr+=VeREF+，Vr-=Avss；Vr+、Vr-是 ADC12 模块的最大值和最小值的参考电压源。当输入模拟电压信号等于或高于 Vr+时，ADC12 转换满幅输出，此时输出值为 0x0FFF。而当输入模拟电压信号等于或小于 Vr-时，ADC12 转换输出为 0，此时输出值为 0x0000。

ADC12 模拟电压的转换公式为： Nadc=4095 x (Vin- Vr-) / (Vr+-Vr-)

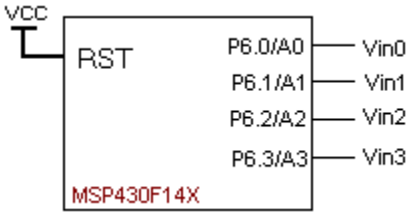
例程：

```
//*******************************************************************************
#include  <msp430x14x.h>
void main(void)
```

```
{
   WDTCTL = WDTPW+WDTHOLD;              // 停止看门狗
   P6SEL |= 0x01;                       // 打开 A0 A/D 通道输入
   ADC12CTL0 = ADC12ON+SHT0_2;          // 开 ADC12 模块+采集定时器分频系数 n=4
   ADC12CTL1 = SHP;                  // 使用采样定时器输出作采集/转换信号 SAMPCON
   ADC12MCTL0 = SREF_2;                  // 使用外部参考电压 Vr+ = VeREF+
   ADC12CTL0 |= ENC;                    // 允许转换

   while (1)
   {
      ADC12CTL0 |= ADC12SC;             // 开始转换
      while ((ADC12IFG & BIT0)==0);     //等待转换结束
      _NOP();                           //转换完成
   }
}
```

例 2

ADC12-A0-A3 重复序列转换



简述：对 AD0-AD3 进行重复序列转换。

在学习 MSP430 ADC12 模块和编写 ADC12 程序时应掌握好对 ADC12 的初始设置：

1-设置 AD 端口的输入

2-打开 ADC12 模块工作；选择 ADC12 内核时钟源；选择采集/转换信号源，若选用了采样定时器作为采集/转换信号源(SAMPCON)时还要选择转换采样时序与采样时钟 ADC12CLK 的系数（定时器的分频系数 N，由 SHITx 位组合决定）和采样定时器的触发方式(MSC 位决定)。采集/转换周期: Tsample=4 x Tadc12clk x N

3-转换模式选择：单通道单次、序列单次、单通多次、序列多次。

由 ADC12CTL1 的 CONSEQ 位决定

4-选择 ADC12 参考电压，和输入通道的选择

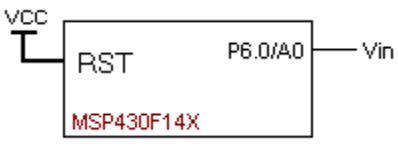
例程：

```
//*******************************************************************************
//ADC12 内核时钟源为 ADC12OSC
#include  <msp430x14x.h>

#define     Num_of_Results     8                    //定义常数
//定义全局变量数组,第个数据为 16 位.
static unsigned int A0results[Num_of_Results];   // A0 结果数组
static unsigned int A1results[Num_of_Results];   // A1 结果数组
static unsigned int A2results[Num_of_Results];   // A2 结果数组
```

```
static unsigned int A3results[Num_of_Results];    // A3 结果数组

void main(void)
{
   WDTCTL = WDTPW+WDTHOLD;                    //  停止看门狗
   P6SEL = 0x0F;                              //  打开 A0-A3 A/D 通道输入
   ADC12CTL0 = ADC12ON+MSC+SHT0_8; //开 ADC12 模块+采样信号由 SHI 仅首次触发
                                   //+采集定时器分频系数 n=64,
   ADC12CTL1 = SHP+CONSEQ_3;     //  使用采样定时器输出作采集/转换信号 SAMPCON
                                 //  重复序列采样模式

   ADC12MCTL0 = INCH_0;          //  参考电压 ref+=AVcc,  输入通道选择为  A0
   ADC12MCTL1 = INCH_1;          //  参考电压 ref+=AVcc,  输入通道选择为  A1
   ADC12MCTL2 = INCH_2;          //  参考电压 ref+=AVcc,  输入通道选择为  A2
   ADC12MCTL3 = INCH_3+EOS;      //  参考电压 ref+=AVcc,  输入通道选择为  A3
                                 //+由此通道产生序列结束控制位
   ADC12IE = 0x08;                           // A3 通道开中断 ADC12IFG.3
   ADC12CTL0 |= ENC;                         //  允许转换
   ADC12CTL0 |= ADC12SC;                     //  启动转换
   _BIS_SR(LPM0_bits + GIE);                 //  进入 LPM0 模式,  开总中断允许
}

#pragma vector=ADC_VECTOR                    //ADC 中断服务程序
__interrupt void ADC12ISR (void)
{
   static unsigned int index = 0;            //中断服务程序中的静态变量

   A0results[index] = ADC12MEM0;   //移动 A0 结果往数组，此操作的同时清除 ADC12FIG.0
   A1results[index] = ADC12MEM1;   //移动 A1 结果往数组，此操作的同时清除 ADC12FIG.1
   A2results[index] = ADC12MEM2;   //移动 A2 结果往数组，此操作的同时清除 ADC12FIG.2
   A3results[index] = ADC12MEM3; //移动 A3 结果往局数组，此操作的同时清除 ADC12FIG.3
   index = (index+1)%Num_of_Results;  //  增加结果的索引，  取 index 变量的模(余数)
}
//*****************************************************************************
```

例 3

ADC12-重复单通道转换



简述： ADC12-重复单通道转换，主要了解 ADC12 的设置。

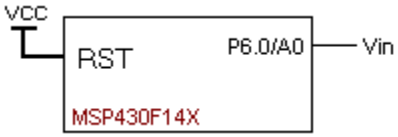
例程：

```
//*****************************************************************************
#include   <msp430x14x.h>
#define    Num_of_Results    8
```

```
static unsigned int results[Num_of_Results];              //静态结果数组

void main(void)
{
   WDTCTL = WDTPW+WDTHOLD;                    // 停止看门狗
   P6SEL |= 0x01;                             // 打开 A0 A/D 通道输入
   ADC12CTL0 = ADC12ON+SHT0_8+MSC;   // 开 ADC12 模块,+采集时间分频系数 n=64
                                              //采样信号由 SHI 仅首次触发
   ADC12CTL1 = SHP+CONSEQ_2;      // 使用采样定时器输出采集/转换信号 SAMPCON
                                  // 单通道多次转换模式
   ADC12IE = 0x01;                            // 允许 A0 中断  ADC12IFG.0
   ADC12CTL0 |= ENC;                          // 允许转换
   ADC12CTL0 |= ADC12SC;                      // 开始转换
   _BIS_SR(LPM0_bits + GIE);                  // 进入 LPM0,开中断总允许
}


#pragma vector=ADC_VECTOR
__interrupt void ADC12ISR (void)
{
   static unsigned int index = 0;             //中断服务程序中的静态变量

   results[index] = ADC12MEM0;                // 移动结果
   index = (index+1)%Num_of_Results;      // 增加结果的索引，取 index 变量的模(余数)
//*****************************************************************************
```

例 4

ADC12-单通道多次转换，采用 TA1 作为采样触发源



简述：利用 P6.0 为 ADC 输入，用 CCR1 产生 ADC12 所需的采样触发信号源。
P6.0 采用单通道多次采方式采样，共采样 512 次后进入 LPM3 模式。

例程：

```
//*****************************************************************************
TACLK = ACLK

#include   <msp430x14x.h>

#define     Num_of_Results     512
int results[Num_of_Results] = {0};

void ADC_Init(void);
```

```
void main(void)
{
   WDTCTL = WDTPW | WDTHOLD;                  // 停止看门狗
   ADC_Init();                                // 初始化 ADC12
   ADC12CTL0 |= ENC;                          // ADC 开始转换
   _BIS_SR(LPM0_bits);                        // 进入 LPM0 模式
}


void ADC_Init(void)
{
   P1DIR = 0xff;                              // 设 P1 方向为输出
   P1OUT = 0;                                 // P1 口输出 0
   P6SEL |= 0x01;                             // P6.0 设为 AD0 输入功能
   ADC12CTL0 = ADC12ON+SHT0_1+REF2_5V+REFON; // 设置 ADC12
   //ADC12 内核开+采样保持时间系数(2)+选择内部参考电压+打开内部参考电压发生器

   ADC12CTL1 = SHP+CONSEQ_2+SHS_1;
//采样触发源来自采样定时器,由采样输入信号(ADC12CLK)的上升沿触发采样定时器
//+单通道多次采样+采样触发源为 Timer_A 的输出单元 out1 输出的信号(由 CCR1 定时产生)

   ADC12MCTL0 = INCH_0 + SREF_1;
   //选择 AD 通道 0+正参考电压=Vref+,负参考电压=AVss

   ADC12IE = 0x0001;                          // 开中断允许-ADC12IFG.0

   TACCR0 = 1500;                             // 延时，允许有足够时间用于整理
   TACCTL0 |= CCIE;                           // TA 为比较模式(TA 与 CCRO 比较)，TA 开中断允许

   TACTL = TASSEL_1 | MC_1;                   // TACLK = ACLK,定时器 A 为增计数模式工作

   _BIS_SR(LPM3_bits + GIE);                  // 进入 LPM3 模式，开总中断允许，等待延时时间结束

   TACCTL0 &= ~CCIE;                          // 关闭 CCR0 捕获比较寄存器的中断

   P2SEL |= BIT3;                             // 设 P2.3 为 TimerA out1 输出功能
   P2DIR |= 0x08;                             // 设 p2.3 为输出
   TACCR0 = 7;                                // 初始化 TACCR0,采样时钟源周期=CCR0+1

   TACCR1 = 4;
// 初始化 TACCR1,用于产一个 TimerA.out1 信号给 ADC12 做采样触发信号源

   TACCTL1 = OUTMOD_3;
// CCR1 捕获比较寄存器的 OUT1 输出单元输出为置位/复位模式

   TACTL = TACLR | MC_1 | TASSEL_1;
// 定时器时钟=ACLK, 清除 TA,增计数模式
}
```

```
// Timer_A0 CCR0 中断服务程序
#pragma vector=TIMERA0_VECTOR
__interrupt void ta0_isr(void)
{
  TACTL = 0;                          // TA 控制寄存器清零
  LPM3_EXIT;                          // 在返回时退出 LPM3 模式
}

// ADC12 中断服务程序
#pragma vector=ADC_VECTOR
__interrupt void ADC12ISR (void)
{
  static unsigned int index = 0;              //静态变量

  results[index++] = ADC12MEM0;               // 移动 ADC 结果往 results 数组

  if (index == 512)                           //是否转换了 512 次?
  {
    ADC12CTL0 &= ~ENC;                        // 是，则停止转换
    index = 0;
    P1OUT |= 0x01;                            // P1.0=1
    _BIS_SR(LPM3_bits);                       // 进入 LPM3 模式
  }
}
//*******************************************************************************
```

8-USART 串行异步模式

MSP430F149 有两个 **USART** 通讯端口，其性能完全一样，每个通讯口可通过 **RS232**、**RS485** 等芯片转换，与之相应的串行接口电路通讯。**MSP430F449** 支持串口异步和同步通讯，每种方式都具有独立的帧格式和独立的控制寄存器。

以下简单的介绍异步通讯和用到几个寄存器的功能。

1、 异步通讯的模式结构。在异步模式下，接收器自身实现帧的同步，外部的通讯设备并不使用这一时钟。波特率的产生是在本地完成的。异步帧格式由 **1** 个起始位、**7** 或 **8** 个数据位、校验位（奇**/**偶**/**无）、**1** 个地址位、和 **1** 或 **2** 个停止位。一般最小帧为 **9** 个位，最大为 **13** 位。

2、 波特率发生器：**MSP430** 的波特率发生器是根据波特率选择寄存器 **UBR** 和调整寄存器 **UM** 来产生的串行数据位，因此在设置上非常灵活，

波特率 **=** 模块时钟**/**分频因子**=UBR+UM/8**

波特率：指每秒传送的位。

分频因子：是指在特定的波特率下，每传送一位数据所需要的时钟周期，用 **N** 表示。

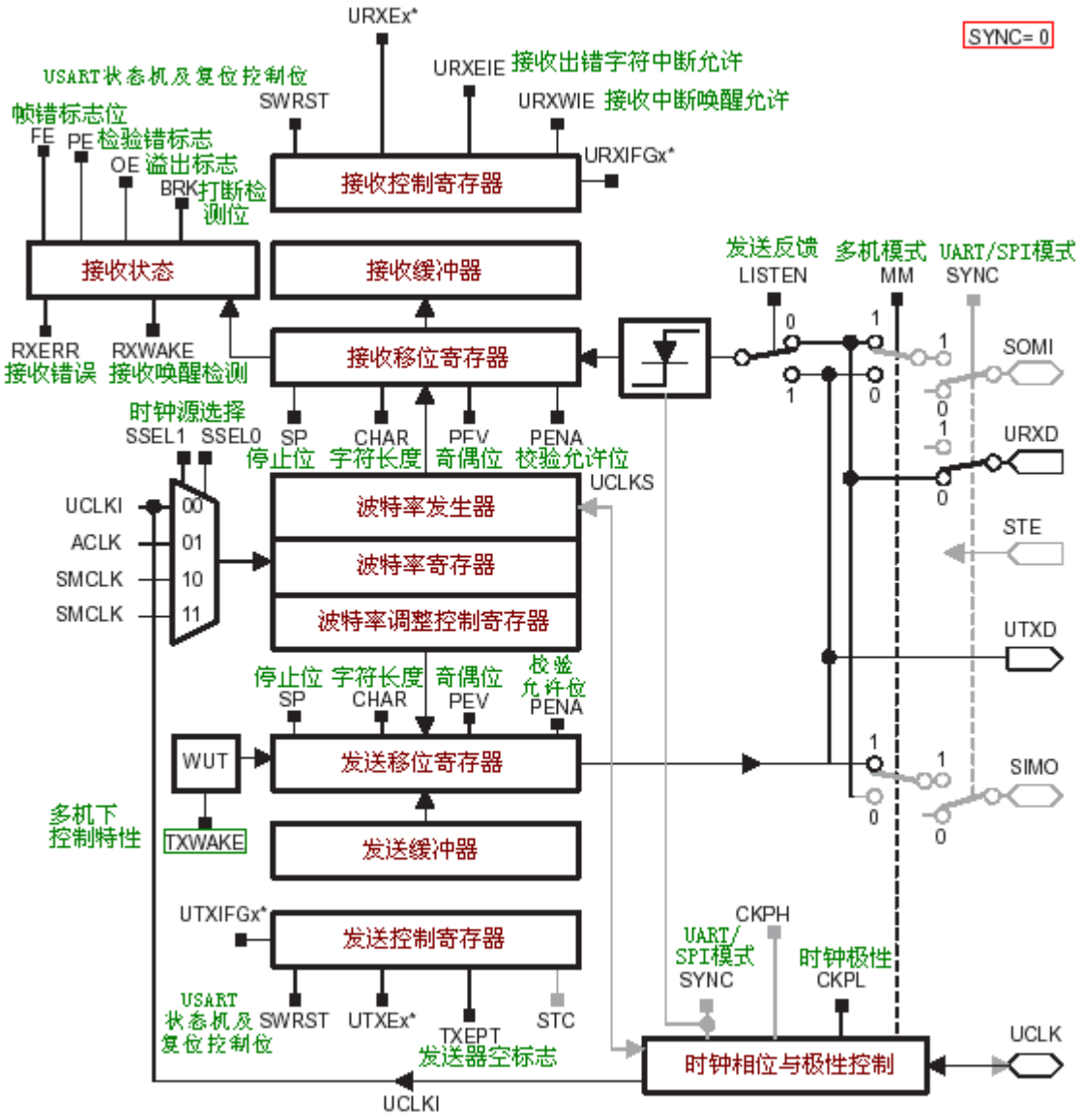
3、**UBR** 和 **UM** 寄存器的装载数据的确定

UBR 是一个 **16** 位的寄存器，它分两个 **8** 位寄存器来设顶定，**UBR00** 和 **UBR10** 注意他们的高低直接字节，使用时是按字节操作。

例如：波特率**=2400**， 模块时钟**=32768Hz** ，其分频因子为 **13.65**

由于分频因子不一定正好为整数，因此对小数部分必须通过调整寄存器来进行调整，在此可插进 **5** 个“**1**”来满足一帧数据的 **65%**的误差。但即使这样也不可能完全消除这种误差，只是

误差很小可忽略不计，这样才能准确的进行数据传送。



MSP430 USART模块结构图

MSP430F14 USART0 异步方式中断控制位

特殊功能寄存器	接收中断控制位	发送中断控制位
IFG1	接收中断标志 **URXIFG0**	接收中断标志 **UTXIFG0**
IE1	接收中断使能 **URXIE0**	接收中断使能 **UTXIE0**
ME1	接收允许 **URXE0**	接收允许 **UTXE0**

MSP430F14 USART1 异步方式中断控制位

特殊功能寄存器	接收中断控制位	发送中断控制位
IFG2	接收中断标志 **URXIFG1**	接收中断标志 **UTXIFG1**

IE2	接收中断使能 URXIE1	接收中断使能 UTXIE1
ME2	接收允许 URXE1	接收允许 UTXE1

在 MSP430 器件中有的型号有两个通信硬件模块 USART0 和 USART1,因此他们有两套寄存器.请看下表:

USART0 的寄存器

寄存器	缩写	读写类型	地址	初始状态
控制寄存器	U0CTL	读/写	070H	PUC 后 001H
发送控制寄存器	U0TCTL	读/写	71H	PUC 后 001H
接收控制寄存器	U0RCTL	读/写	72H	PUC 后 000H
波特率调整控制寄存器	U0MCTL	读/写	73H	不变
波特率控制寄存器 0	U0BR0	读/写	74H	不变
波特率控制寄存器 1	U0BR1	读/写	75H	不变
接收缓冲器	U0RXBUF	读	76H	不变
发送缓冲器	U0TXBUF	读/写	77H	不变
SFR 模块使能寄存器 1	ME1	读/写	004H	PUC 后 000H
FR 模块使能寄存器 1	IE1	读/写	000H	PUC 后 000H
FR 模块使能寄存器 1	IFG1	读/写	002H	PUC 后 082H

USART1 的寄存器

寄存器	缩写	读写类型	地址	初始状态
控制寄存器	U1CTL	读/写	078H	PUC 后 001H
发送控制寄存器	U1TCTL	读/写	79H	PUC 后 001H
接收控制寄存器	U1RCTL	读/写	7AH	PUC 后 000H
波特率调整控制寄存器	U1MCTL	读/写	7BH	不变
波特率控制寄存器 0	U1BR0	读/写	7CH	不变
波特率控制寄存器 1	U1BR1	读/写	7DH	不变
接收缓冲器	U1RXBUF	读	7EH	不变
发送缓冲器	U1TXBUF	读/写	7FH	不变
SFR 模块使能寄存器 1	ME2	读/写	005H	PUC 后 000H
FR 模块使能寄存器 1	IE2	读/写	001H	PUC 后 000H
FR 模块使能寄存器 1	IFG2	读/写	003H	PUC 后 020H

UxCTL 控制寄存器

7	6	5	4	3	2	1	0
PENA	PEV	SPB	CHAR	LISTEN	SYNC	MM	SWRST

PENA 校验允许位

0 校验禁止

1 校验允许

校验允许时,发送端发送校验,接收端接收该校验,地址位多机模式中,地址位包含校验操作.

PEV 奇偶校验位,该位在校验允许时有效

0 奇校验

1 偶校验

SPB 停止位选择.决定发送的停止位数,但接收时接收器只检测 1 位停止位.

0 1 位停止位

1 2 位停止位

CHAR 字符长度

0 7 位

1 8 位

LISTEN 反馈选择.选择是否发送数据由内部反馈给接收器

0 无反馈

1 有反馈,发送信号由内部反馈给接收器

SYNC USART 模块的模式选择

0 UART 模式[异步]

1 SPI 模式[同步]

MM 多机模式选择位

0 线路空闲多机协议

1 地址位多机协议

SWRST 控制位

上电时该位置位,此时 USART 状态机和运行标志初始化成复状态 (URXIFG=0,URXIE=0,UTXIE=0,UTXIFG=1)。所有受影响的逻辑保持在复位状态,直至 SWRST 复位。也就是说一次系统复位后，只有对 SWRST 复位，USART 才能重新被允许。而接收和发送允许标志 URXE 和 UTXE 不会因 SWRST 而更改。

SWRST 位会使 URXIE、UTXIE、URXIFG、RXWAKE、TXWAKE、RXERR、BRK、PE、OE 及 FE 等复位。

在串行口使用设置时，这一位起重要的作用。一次正确的 USART 模块初始化应该是这样设置过程的：先在 SWRST=1 时设置，设置完串口后再设置 SWRST=0；最后如需要中断，则设置相应的中断使能。

UxTCTL 发送控制寄存器

7	6	5	4	3	2	1	0

未用	CKPL	SSEL1	SSEL0	URXSE	TXWAKE	未用	TXEPT

CKPL 时钟极性控制位
- **0** **UCLKI** 信号与 **UCLK** 信号极性相同
- **1** **UCLKI** 信号与 **UCLK** 信号极性相反

SSEL1、SSEL0 时钟源选择，此两位确定波特率发生器的时钟源
- **0** 外部时钟 **UCLKI**；
- **1** 辅助时钟 **ACLK**
- **2** 子系统时钟 **SMCLK**
- **3** 子系统时钟 **SMCLK**

URXSE 接收触发沿控制位
- **0** 没有接收触发沿检测
- **1** 有接收触发沿检测

TXWAKE 传输唤醒控制
- **0** 下一个要传输的字符为数据
- **1** 下一个要传输的字符是地址

TXEPT 发送器空标志，在异步模式与同步模式时是不一样的。
- **0** 正在传输数据或者发送缓冲器**(UTXBUF)**有数据
- **1** 表示发送移位寄存器和 **UTXBUF** 空或者 **SWRST=1**

URCTL 接收控制寄存器

7	6	5	4	3	2	1	0
FE	PE	OE	BRK	URXEIE	URXWIE	RXWAKE	RXERR

FE 帧错误标志位
- **0** 没有帧错误
- **1** 帧错误

PE 校验错误标志位
- **0** 校验正确
- **1** 校验错误

OE 溢出标志位
- **0** 无溢出
- **1** 有溢出

BRK 打断检测位

0 没有被打断

1 被打断

URXEIE 接收出错中断允许位

0 不允许中断，不接收出错字符并且不改变 **URXIFG** 标志

1 允许中断，出错字符接收并且能够置位 **URXIFG**

URXWIE 接收唤醒中断允许位，当接收到地址字符时，该位能够置位 **URXIFG**，当 **URXEIE=0**，如果接收内容有错误，该位不能置位 **URXIFG**。

0 所有接收的字符都能够置位 **URXIFG**

1 只能接收到地址字符才能置位 **URXIFG**

在各种条件下 **URXEIE** 和 **URXWIE** 对 **URXIFG** 的影响

URXEIE	URXWIE	字符出错	地址字符	接收字符后的标志位 URXIFG
0	X	1	X	不变
0	0	0	X	置位
0	1	0	0	不变
0	1	0	1	置位
1	0	X	X	置位(接收所有字符)
1	1	X	0	不变
1	1	X	1	置位

RXWAKE 接收唤醒检测位。在地址位多机模式，接收字符地址位置位时，该机被唤醒，在线路空闲多机模式，在接收到字符前检测到 **URXD** 线路空闲时，该机被唤起，**RXWAKE** 置位。

0 没有被唤醒，接收到的字符是数据

1 唤醒，接收的字符是地址

RXERR 接收错误标志位

0 没有接收错误

1 有接收到错误

UxBR0、UxBR1 波特率选择寄存器

这两个寄存器是用于存放波特率分频因子的整数部分。

UxBR0 波特率选择寄存器 0

7	6	5	4	3	2	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

UxBR1 波特率选择寄存器 1

7	6	5	4	3	2	1	0

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}

UxMCTL 波特率调整控制寄存器

7	6	5	4	3	2	1	0
M7	M6	M5	M4	M3	M2	M1	M0

若波特率发生器的输入频率 **BRCLK** 不是所需波特率的整数倍，带有一小数，则整数部分写 **UBR** 寄存器，小数部分由调整寄存器 **UxMCTL** 的内容反映。波特率由以下公式计算：

波特率=BRCLK/(UBR+(M7+M6+…M0)/8)

URXBUF 接收数据缓存

7	6	5	4	3	2	1	0

接收缓存存放移位寄存器最后接收的字符，可由用户访问。读接收缓存可以复位接收时产生的各种错误标志、**RXWAKE** 位和 **URXIFGx** 位。如果传输 **7** 位数据，接收缓存内容右对齐，最高位为 **0**。

当收接和控制条件为真时，接收缓存装入当前接收到的字符。

当接收和控制条件为真时接收数据缓存结果

条件		结果			
URXEIE	URXWIE	装入 URXBUF	PE	FE	BRK
0	1	无差错地址字符	0	0	0
1	1	所有地址字符	X	X	X
0	0	无差错字符	0	0	0
1	0	所有字符	X	X	X

UTXBUF 发送数据缓存

7	6	5	4	3	2	1	0

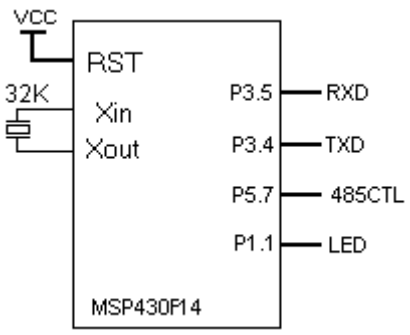
发送缓存内容可以传至发送移位寄存器，然后由 **UTXDx** 传输。对发送缓存进行写操作可以复位 **UTXIFGx**。如果传输出 **7** 位数据，发送缓存内容最高为 **0**。

常用波特率及其对应设置参数与对应误差表

baud rate	Divide by		ACLK[32768HZ 低频振荡器]						MCLK[1048576HZ 高频振荡器]				
	ACLK	MCLK	UxBR1	UxBR0	UxMCTL	Max.TX Error/%	Max.RX Error/%	Synchr TX Error/%	UxBR1	UxBR0	UxMCTL	Max.TX Error/%	Max.RX Error/%
75	436.91	13981	1	B4	FF	-0.3/0.3	-0.3/0.3	±2	36	9D	FF	0/0.1	±2

110	297.89	9532.51	1	29	FF	0/0.5	0/0.5	±3	25	3C	FF	0/0.1	±3
150	218.45	6990.5	0	DA	55	0/0.4	0/0.4	±2	1B	4E	FF	0/0.1	±2
300	109.23	3495.25	0	6D	22	-0.3/0.7	-0.3/0.7	±2	0D	A7	00	-0.1/0	±2
600	54.61	1747.63	0	36	D5	-1/1	-1/1	±2	06	D3	FF	0/0.3	±2
1200	27.31	873.81	0	1B	03	-4/3	-4/3	±2	03	69	FF	0/0.3	±2
2400	13.65	436.91	0	0D	6B	-6/3	-6/3	±4	01	B4	FF	0/0.3	±2
4800	6.85	218.45	0	06	6F	-9/11	-9/11	±7	0	DA	55	0/0.4	±2
9600	3.41	109.23	0	03	4A	-21/12	-21/12	±15	0	6D	03	-0.4/1	±2
19200		54.61							0	36	6B	-0.2/2	±2
38400		27.31							0	1B	03	-4/3	±2
76800		13.65							0	0D	6B	-6/3	±4
115200		9.1							0	09	08	-5/7	±7

例程描述：利用上位软件发送一个字符给单片机，单片机接收后发回上位软件。
MSP430 电路简图如下：



程序：

```
// ACLK = UCLK0 = LFXT1 = 32768, MCLK = SMCLK = DCO~ 800k
//  波特率在 32768hz XTAL @9600 = 32768Hz/9600 = 3.41 (0003h 4Ah )
//  软件：IAR Embedded Workbench Version: 3.4A

#include   <msp430x14x.h>
void delay (void);                                  //串口时，无此延时
void main(void)
{
```

```
  WDTCTL = WDTPW + WDTHOLD;                 // 停止看门

  P5DIR |= 0x80;                            // P5.7 为输出,RS485 控制端
  P5OUT &= ~0x80;                           // 使 SN75176B 为接收状态

  P1DIR |= 0x02;                            // 设 LED 指示
  P1OUT &= ~0x02;

  //以下是串口设置
  P3SEL |= 0x30;                            // P3.4,5 = USART0 TXD/RXD
  ME1 |= UTXE0 + URXE0;                     // 使能 USART0 模块 TXD/RXD
  UCTL0 |= CHAR;                            // 8 位字符格式
  UTCTL0 |= SSEL0;                          // 串口模块时时钟 UCLK = ACLK
  UBR00 = 0x03;                             // 设置波特率控制寄存器
  UBR10 = 0x00;                             // 32k/9600 - 3.41(ACLK)
  UMCTL0 = 0x4A;                            // 波特率调整寄存器设置
  UCTL0 &= ~SWRST;                          // 初始化 USART 状态机
  IE1 |= URXIE0;                            // 使能 USART0 接收中断

// Mainloop
  for (;;)
  {
    _BIS_SR(LPM3_bits + GIE);               // 进入 LPM3 模式/允许总中断
    while (!(IFG1 & UTXIFG0));              // USART0 发送缓冲器是否准备?
      P5OUT |= 0x80;                        // 使 SN75176B 为发送
      TXBUF0 = RXBUF0;                      // 从接收缓冲器写入发送缓冲器
      delay ();
      P5OUT &= ~0x80;                       // 使 SN75176B 为接收
      P1OUT ^= 0x02;                        //LED
  }
}

// UART0 接收中断将从 LPM3 模式退出
#pragma vector=UART0RX_VECTOR
__interrupt void usart0_rx (void)
 {
  _BIC_SR_IRQ(LPM3_bits);                   // 退出 LPM3

 }

void delay (void)
 { unsigned i=180;
   while(i!=0)
      i--;
 }
```

10-CPU 模块及全局资料

MSP430 中断嵌套机制

(1) 430 默认的是关闭中断嵌套的，除非你在一个中断程序中再次开总中断 EINT。

(2) 当进入中断程序时，只要不在中断中再次开中断，刚总中断是关闭的，此时来中断不管是比当前中断的优先级高还是低都不执行。

(3)若在中断 A 中开了总中断，刚可以响应后来的中断 B（不管 B 的优先级比 A 高还是低），B 执行完现继续执行。注意：进入中断 B 生总中断同样也会关闭，如果 B 中断程序执行时需响应中断 C，则此时也要开总中断，若不需响应中断，则不用开中断，B 执行完后中跳出中断程序进入 A 程序时，总中断会自动打开。

(4)若在中断中开了总中断，后来的中断同时有多个，则会按优先级来执行，即中断优先级只有在多个中断同时到来才起做用！中断服务不执行抢先原则。

(5)对于单源中断，只要响应中断，系统硬件自动清中断标志位，对于 TA/TB 定时器的比较/捕获中断，只要访问 TAIV/TBIV，标志位倍被自动清除；对于多源中断要手动清标志位，比如 P1/P2 口中断，要手工清除相应的标志，如果在这种中断用“EINT（）；”开中断，而在打开中断前没有清标志，就会有相同的中断不断嵌入，而导致堆栈溢出引起复位，所以在这类中断必须先清标志现打开中断开关。

关于 CPU 部分我这次主要着重讲述下 SR 状态寄存器各位功能作用,对于 C 语言写已足够用了。另外还会补充一部单片机全局性的资料。

MSP430 的中断分为 3 种：系统复位、不可屏蔽中断、可屏蔽中断。关于中断相关状态情况:

(1)系统复位的中断向量为 0xFFFE。

(2)不可屏蔽中断的中断向量为 0xFFFC。响应不可屏蔽中断时,硬件自动将 OFIE、NMIE、ACCVIE 复位。软件首先判断中断源并复位中断标志,接着执行用户代码。退出中断之前需要置位 OFIE、NMIE、ACCVIE,以便能够再次响应中断。需要特别注意点：置位 OFIE、NMIE、ACCVIE 后,必须立即退出中断相应程序,否则会再次触发中断,导致中断嵌套,从而导致堆栈溢出,致使程序执行结果的无法预料。

(3)可屏蔽中断的中断来源于具有中断能力的外围模块,包括看门狗定时器工作在定时器模式时溢出产生的中断。每一个中断都可以被自己的中断控制位屏蔽,也可以由全局中断控制位屏蔽。

多个中断请求发生时,响应最高优先级中断。响应中断时,MSP430 会将不可屏蔽中断控制位 SR.GIE 复位。因此,一旦响应了中断,即使有优先级更高的可屏蔽中断出现,也不会中断当前正在响应的中断,去响应另外的中断。但 SR.GIE 复位不影响不可屏蔽中断,所以仍可以接受不可屏蔽中断的中断请求。

中断响应的过程：

(1)如果 CPU 处于活动状态,则完成当前指令;
(2)若 CPU 处于低功耗状态,则退出低功耗状态;
(3)将下一条指令的 PC 值压入堆栈;
(4)将状态寄存器 SR 压入堆栈;
(5)若有多个中断请求,响应最高优先级中断;
(6)单中断源的中断请求标志位自动复位,多中断源的标志位不变,等待软件复位;
(7)总中断允许位 SR.GIE 复位。SR 状态寄存器中的 CPUOFF、OSCOFF、SCG1、V、N、Z、C 位复位;
(8)相应的中断向量值装入 PC 寄存器,程序从此地址开始执行。

中断返回的过程：
(1)从堆栈中恢复 PC 值,若响应中断前 CPU 处于低功耗模式,则可屏蔽中断仍然恢复低功耗模式;
(2)从堆栈中恢复 PC 值,若响应中断前 CPU 不处于低功耗模式,则从此地址继续执行程序。

CPU 的状态寄存器 SR

15-9	8	7	6	5	4	3	2	1	0
保留	V	SCG1	SCG0	OscOff	CPUoff	GIE	N	Z	C

V 溢出标志

SCG1 SCG0 时钟控制位
SCG1 置位关闭 SMCLK
SCG0 置位关闭 DCO 发生器

OscOff 晶体振荡控制位
置位 OscOff 使晶体振荡器处于停止状态，置位 OscOff 同时 CPUoff 位也置位。可用 NMI 或外部中断(系统当前中断允许)将 CPU 唤醒。

CPUoff CPU 控制位
置位 CPUoff 可使 CPU 进入关闭模式，可以用所中断允许将 CPU 唤醒。

GIE 全局中断标志位
控制可屏蔽中断
GIE 置位 CPU 可响应可屏蔽中断
GIE 置位 CPU 不响应可屏蔽中断

N 负标志

Z 零标志

C 进位标志

特殊功能寄存器

系统中断处理
当各模块发生中断请求时并且相应的中断允许和通用中断允许位(GIE)置位时,中断服务程序按以下顺序动作:
[1] CPU 处于活动状态:完成当前所执行的指令.
[2]CPU 处于停止状态:低功耗模式终止.
[3]指向下一条指令的 PC 值压入堆栈.
[4]SR 压入堆栈
[5]如果在执行上条指令时已发生多个中断请求等待服务,则选择最高优先级者.
[6]单中断源标志的中断请求位自动复位,多中断源标志仍保持置位等待软件服务.
[7]通用中断允许位 GIE 复位;CPUoff 位、OSCOff 位和 SCG1 位置位；状态位 V、N、Z 和 C

复位；SCG0 位保持不变.

[8]相应的中断向量值装入 PC，程序从此地址继续执行中断处理.

除了 CPU 的 SR 寄存器中的 GIE 位(总中断允许)，中断源都可以被控制位单独或成组地允许或禁止。中断允许标志集中于两个地址的 SFR 中。中断请求对程序流的控制可以运用中断允许和屏蔽来方便地调整。硬件只对被允许的中断源中的最高优先级者服务。

系统中断优先级

11,11x1 中断向量表

中断源	中断标志	系统中断	地址	优先级
上电	WDTIFG KEYV	复位	0FFFEH	15 最高级
外部复位				
看门狗				
FLASH				
NMI	NMIIFG	(非)屏蔽	0FFFCH	14
振荡器故障	OFIFG	(非)屏蔽		
FLASH 存储器非法访问	ACCVIFG	(非)屏蔽		
			0FFFAH	13
			0FFF8H	12
比较器 A	CMPAIFG	可屏蔽	0FFF6H	11
WDT	WDTIFG	可屏蔽	0FFF4H	10
Timer_A3	CCIFG0	可屏蔽	0FFF2H	9
Timer_A3	CCIFG1 CCIFG2 CCIFG	可屏蔽	0FFF0H	8
			0FFEEH	7
			0FFECH	6
			0FFEAH	5
			0FFE8H	4
P2	P2IFG.0-7	可屏蔽	0FFE6H	3
P1	P1IFG.0-7	可屏蔽	0FFE4H	2
			0FFE2H	1
			0FFE0H	0 最低

13,14 中断向量表

中断源	中断标志	系统中断	地址	优先级
上电		复位	0FFFEH	15 最高

外部复位 看门狗 FLASH	WDTIFG KEYV			级
NMI	NMIIFG	(非)屏蔽	OFFFCH	14
振荡器故障	OFIFG	(非)屏蔽		
FLASH 存储器非法访问	ACCVIFG	(非)屏蔽		
Timer_B7	BCCIFG0	可屏蔽	0FFFAH	13
Timer_B7	BCCIFG1-6	可屏蔽	0FFF8H	12
	TBIFG	可屏蔽		
比较器 A	CMPAIFG	可屏蔽	0FFF6H	11
WDT	WDTIFG	可屏蔽	0FFF4H	10
USART0 接收	URXIFG0	可屏蔽	0FFF2H	9
USART0 发送	UTXIFG0	可屏蔽	0FFF0H	8
ADC12	ADCIFG	可屏蔽	0FFEEH	7
Timer_A3	CCIFG0	可屏蔽	0FFECH	6
	CCIFG1	可屏蔽	0FFEAH	5
Timer_A3	CCIFG2	可屏蔽		
	CCIFG	可屏蔽		
P1	P1IFG.0-7	可屏蔽	0FFE8H	4
USART1	URXIFG1	可屏蔽	0FFE6H	3
USART1	UTXIFG1	可屏蔽	0FFE4H	2
P2	P2IFG.0-7	可屏蔽	0FFE2H	1
		可屏蔽	0FFE0H	0 最低

44x 中断向量表

中断源	中断标志	系统中断	地址	优先级
上电 外部复位 看门狗 FLASH	WDTIFG KEYV	复位	0FFFEH	15 最高级
NMI	NMIIFG	(非)屏蔽	0FFFCH	14
振荡器故障	OFIFG	(非)屏蔽		
FLASH 存储器非法访问	ACCVIFG	(非)屏蔽		
Timer_B7	BCCIFG0	可屏蔽	0FFFAH	13

Timer_B7	BCCIFG1-6	可屏蔽	0FFF8H	12
	TBIFG	可屏蔽		
比较器 A	CMPAIFG	可屏蔽	0FFF6H	11
WDT	WDTIFG	可屏蔽	0FFF4H	10
USART0 接收	URXIFG0	可屏蔽	0FFF2H	9
USART0 发送	UTXIFG0	可屏蔽	0FFF0H	8
ADC12	ADCIFG	可屏蔽	0FFEEH	7
Timer_A3	CCIFG0	可屏蔽	0FFECH	6
	CCIFG1	可屏蔽	0FFEAH	5
Timer_A3	CCIFG2	可屏蔽		
	CCIFG	可屏蔽		
P1	P1IFG.0-7	可屏蔽	0FFE8H	4
USART1	URXIFG1	可屏蔽	0FFE6H	3
USART1	UTXIFG1	可屏蔽	0FFE4H	2
P2	P2IFG.0-7	可屏蔽	0FFE2H	1
基础定时器	BTIFG	可屏蔽	0FFE0H	0 最低

ME1 模块允许寄存器 1

7	6	5	4	3	2	1	0
UTXE0	URXE0/ USPIE0						

初始状态:PUC 后为 000H

UTXE0 USART0 发送允许位
0 不允许
1 可允许

URXE0 USART0 接收允许位
0 不允许
1 可允许

USPIE0 USART0 发送与接收允许位(在 SPI 模式)

IFG1 中断标志寄存器 1

7	6	5	4	3	2	1	0
UTXIFG0	URXIFG0		NMIIFG			OFIFG	WDTIFG

初始状态:PUC 后为 082H

UTXIFG0 USART0 发送中断标志位(F14、15、16、44)

此位上电为 UTXIFG0=1,表示可以向发送缓冲器写操作。对发送缓存进行写操作时可以复位 UTXIFG0。

URXIFG0 USART0 接收中断标志位(F14、15、16、44)

0 无接收到有效字符

1 接收到有效字符，读接收缓存可以复位接收时产生的各种错误标志、RXWAKE 位和 URXIFGx 位。

NMIIFG NMI/RST 引脚信号位

OFIFG 振荡器失效时置位

0 无振荡器失效

1 振荡器失效，当 XT 扫荡器丢失大约 100 个振荡周期时设置 TX 振荡器失效标志 OSCFault。OSCFault 标志设置振荡器失效中断标志 OFIFG，如果这时振荡器失效中断允许(OFIE)置位，则将产生非屏蔽中断请求。OFIFG 标志必须由用户软件来清除。

WDTIFG 看门狗中断标志

看门狗模式时溢出或密钥不符时产生置位

IE1 中断使能寄存器 1

7	6	5	4	3	2	1	0
UTXIE0	URXIE0	ACCVIE	NMIIE			OFIE	WDTIE

初始状态:PUC 后为 000H

UTXIE0 USART 发送中断允许位

0 不允许

1 允许

URXIE0 USART0 接收中断使能位允许

0 不允许

1 允许

ACCVIE FLASH 存贮器非法访问中断允许

NMIIE NMI 中断允许

OFIE 振荡器失效中断允许

0 不允许

1 允许

WDTIE 看门狗允许,选看门狗模式无效
0 不允许
1 允许

ME2 中断使能寄存器 2

7	6	5	4	3	2	1	0
		UTIE1	URXE1/ USPIE1				

初始状态:PUC 后为 000H

UTXE1 USART1 发送允许位
0 不允许
1 可允许

URXE1 USART1 接收允许位
0 不允许
1 可允许

USPIE1 USART0 发送与接收允许位(在 SPI 模式)

IFG2 中断标志寄存器 2

7	6	5	4	3	2	1	0
		UTXIFG1	URXIFG1				

初始状态:PUC 后为 020H

UTXIFG1 USART1 发送中断标志位(F14、15、16、44)
此位上电为 UTXIFG0=1,表示可以向发送缓冲器写字符。对发送缓存进行写操作时可以复位 UTXIFG0。

URXIFG1 USART1 接收中断标志位(F14、15、16、44)
0 无接收到有效字符
1 接收到有效字符，读接收缓存可以复位接收时产生的各种错误标志、RXWAKE 位和 URXIFGx 位。

IE2 中断使能寄存器 2

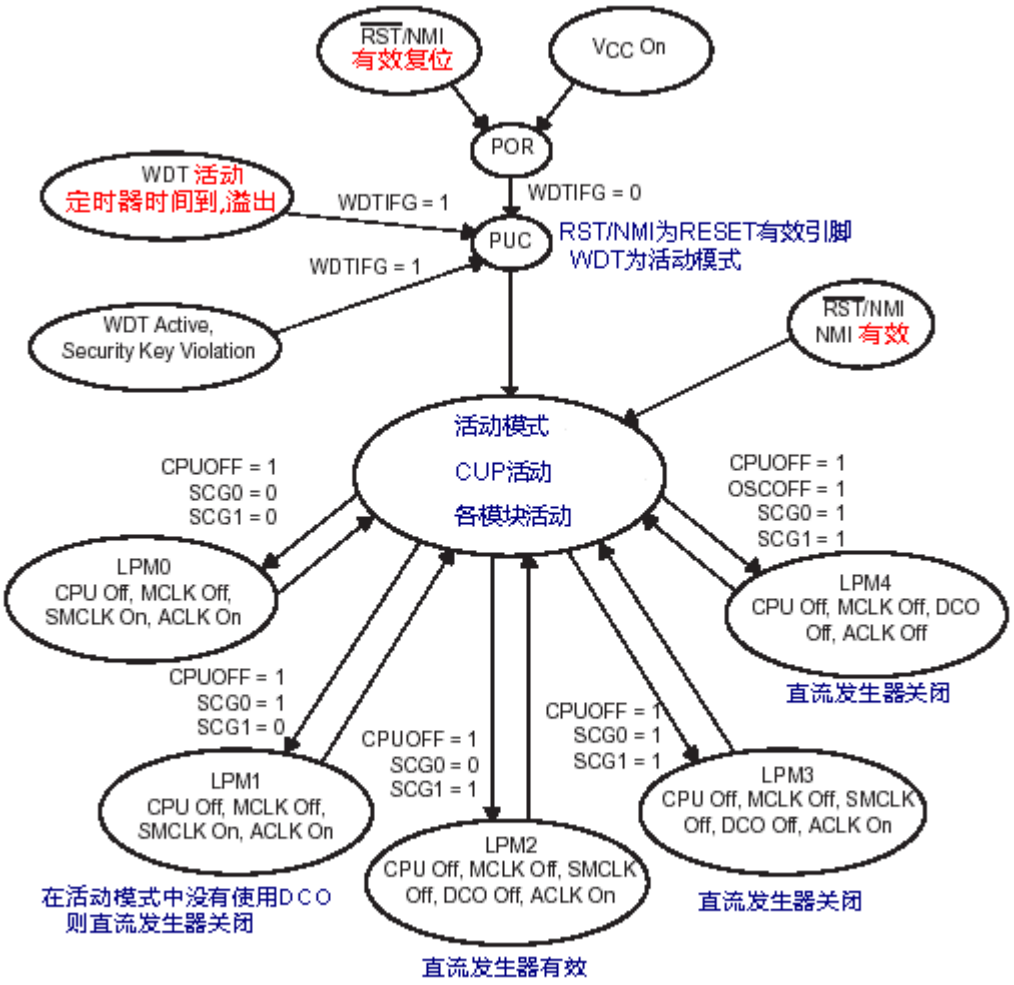
7	6	5	4	3	2	1	0
		UTXIE1	URXIE1				

初始状态:PUC 后为 000H

UTXIE1 USART1 发送中断允许位(F14、15、16、44)

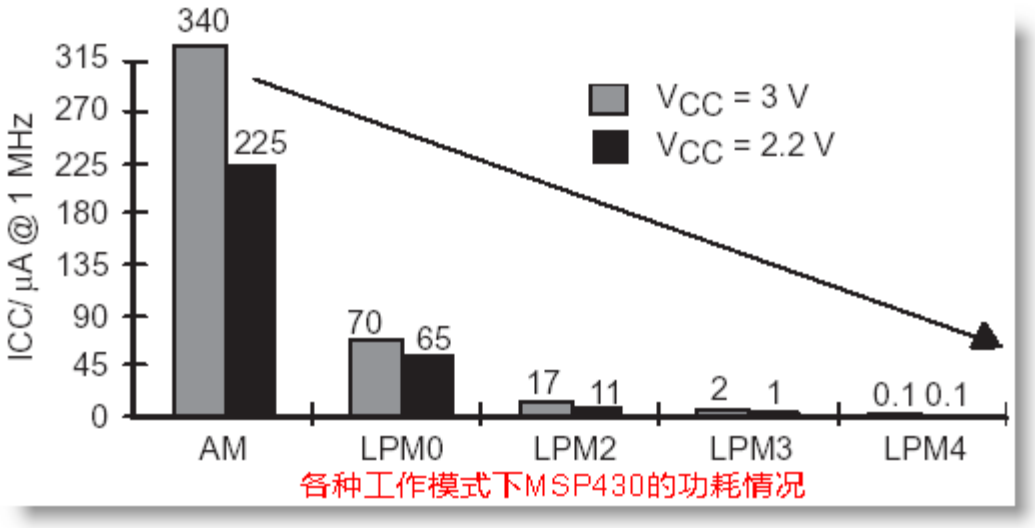
URXIE1 USART1 接收中断允许位(F14、15、16、44)

MSP430 低功耗模式



MSP430F工作模式状态图

低功耗是 **MSP430** 单片机其中一种的最大特色，所以要用得其所，就要好好了解其低功耗模式的不同状态的不同设置。



各种工作模式、各控制位及时钟的活动状态

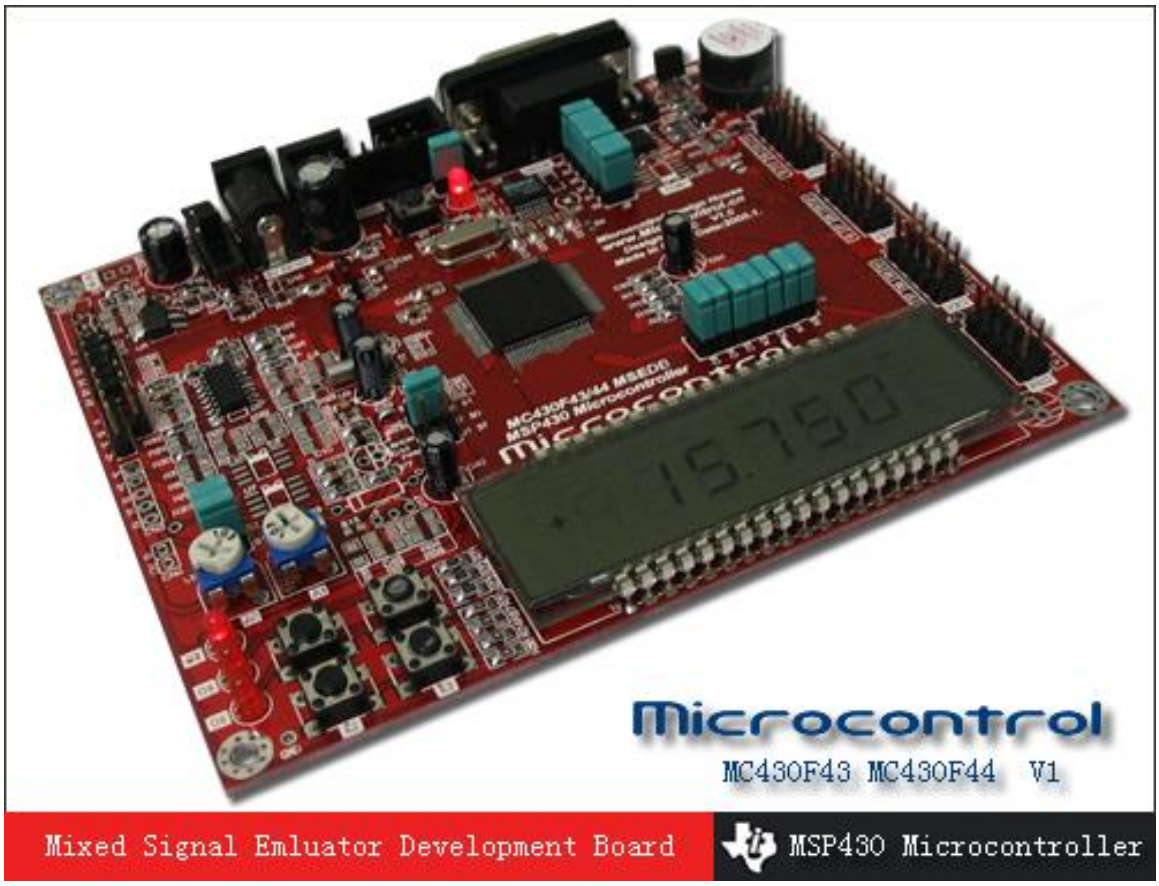
工作模式	控制位	CPU 状态、振荡器及时钟状态
活动模式(AM)	SCG1=0 SCG0=0 OscOff=0 CPUoff=0	CPU 处于活动状态 MCLK 活动 SMCLK 活动 ACLK 活动
低功耗模式 0 (LPM0)	SCG1=0 SCG0=0 OscOff=0 CPUoff=1	CPU 处于禁止状态 MCLK 被禁止 SMCLK 活动 ACLK 活动
低功耗模式 1 (LPM1)	SCG1=0 SCG0=1 OscOff=0 CPUoff=1	CPU 处于禁止状态 如果 DCO 未用作 MCLK 或 SMCLK，则直流发生器被禁止，否则仍然保持活动。 SMCLK 活动 ACLK 活动
低功耗模式 2 (LPM2)	SCG1=0 SCG0=1 OscOff=0 CPUoff=1	CPU 处于禁止状态 如果 DCO 未用作 MCLK 或 SMCLK，则自动被禁止。 MCLK 被禁止 SMCLK 被禁止 ACLK 活动
低功耗模式 3 (LPM3)	SCG1=1 SCG0=1 OscOff=0 CPUoff=1	CPU 处于禁止状态 DCO 被禁止，直流发生器被禁止。 MCLK 被禁止 SMCL 被禁止 ACLK 活动

低功耗模式 4 (LPM4)	SCG1=X SCG0=X OscOff=0 CPUoff=1	CPU 处于禁止状态 DCO 被禁止，直流发生器被禁止。 MCLK 被禁止 SMCL 被禁止 ACLK 活动 所有振荡器停止工作

参考文献

1. Texas Instruments Incorporated MSP430F14X C code Example
2. Texas Instruments Incorporated MSP430x1xx Family slau049e.pdf
3. Texas Instruments Incorporated MSP430F149 Datasheet
4. <<MSP430 系列 16 位超低功耗单片机原理与应用>> 沈建华 杨艳琴 骁曙 清华大学出版社

后面将相继推出其它模块寄存器说明，请密切留意微控设计网的更新 [注意版号与更新日期]



MSP430FET 仿真器