超低功耗单片 MSP430

第一章：单片机概述

1.1 MSP430 系列单片机的特点

MSP430 系列单片机针对各种不同应用，包括一系列不同型号的器件。主要特点有：

1.超低功耗

MSP430 系列单片机的电源电压采用 1.8~3.6V 低电压,RAM 数据保持方式下耗电仅 0.1uA,活动模式 250uA/MIPS。而传统 MCS51 单片机约为 10~20mA/MIPS（MIPS：每秒百万条指令数),I/O 输入端口的漏电流最大仅 50nA。

2.强大的处理能力

MSP430 系列单片机是 16 位单片机，或者说是伪 16 位单片机。在单片机内部，采用 16 为数据总线进行操作，并采用目前流行的、受好评的精简指令集（RISC）结构，一个时钟期可以执行一条指令，而传统 MCS51 单片机要 12 个时钟周期才能执行一条指令。比如 MSP430 在 8MHz 晶振驱动下工作时，指令速度可达 8MIPS。

同时，MSP430 系列单片机中的某些性能更好，采用了一般只有 DSP 才有的 16 位多功能硬件乘法器、硬件乘-加功能、DMA（Direct Memory Access）等一系列先进的结构体系，大大增强了数据处理和运算能力，可以有效的实现一些数字信号处理的算法（如 FFT,DFT 等)。这种结构在其他系列单片机中尚未使用。

3.高性能模拟技术及丰富的片上外围模块

MSP430 系列单片机结合 TI 的高性能模拟技术,集成了丰富的片内外设包括:看门狗 WDT,模拟比较器 A，定时器 A 定时器 B，串口 0，1，硬件乘法器，液晶驱动器，10 位/12/14/位 ADC，12 位 DAC，IIC 总线，DMA，端口 P1~P6，基本定时器，视不同型号可能有不同组合。我们主要介绍型号 MSP430F449，这款单片机不包括上述介绍的 DMA，IIC 总线模块以及数模转换 DAC。

4.系统工作稳定

上电复位后，首先由 DCO_CLK 启动 CPU，以保证程序从正确的位置开始执行，保证晶体振荡器有足够的起振及稳定时间。然后软件可设置适当的寄存器的控制来确定最后的系统时钟。如果晶体振荡器用作 CPU 时钟 MCLK 时发生故障，DCO 为自动启动，以保证系统正常工作。

5.方便高效的开发环境

以 MSP430F449 为例，F 代表单片机为 FLASH 型，可以通过片内的 JTAG 接口下载程序到 FLASH，再由 JTAG 接口控制程序运行，读取片内 CPU 状态，以及存储器内容等信息供设计者调试，整个开发（编译，调试，下载）都可以在同一个软件集成环境中进行。开发语言有汇

编语言和 C 语言。目前比较好的开发软件是 IAR Workbench。

1.2 MSP430 操作简介

MSP430 系列单片机与 MCS51 单片不一样，不需要对总线进行操作。其丰富的外设资源全部映射到相关寄存器，也就是说，对单片机的操作实际上就是通过位寻址的方式对相关寄存器的赋值操作，并且每个寄存器都能进行位操作，使得寄存器的使用变得更加灵活方便。这是 MSP430 系列单片机与 MCS51 单片机相比另一个独特的地方。

1.3 MSP430 系列单片机在系统中的应用

由于 MSP430 单片机有功耗低，速度快，片内资源丰富，I/O 口多，数字噪声低等特点，MSP430 单片机可以用以设计多种系统。例如：

控制类：可以通过内部丰富的定时器功能产生可灵活控制的 PWM 波，以及丰富的 I/O 口和强大的中断功能进行传感器的控制。

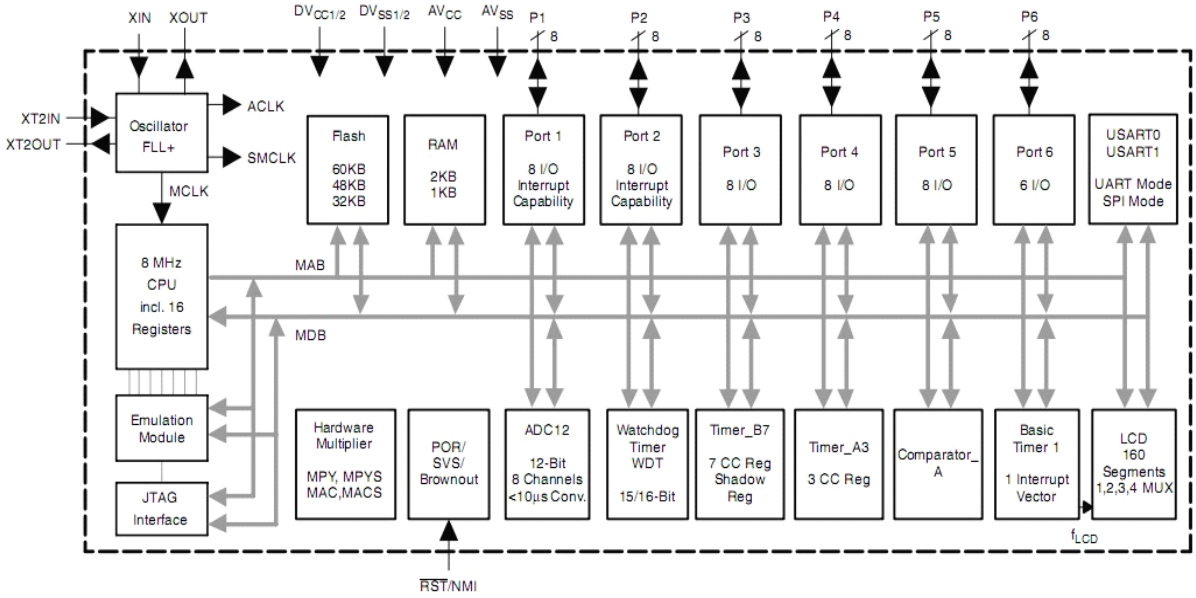
通信类：内部丰富的串口资源以及灵活的时钟设置给通信系统提供了强大的支持，利用定时器可以直接产生已调制的 ASK 信号。

电源类：利用定时器产生 PWM 波和 SPWM 波进行逆变控制。

仪器类：丰富的 I/O 口资源以及内部集成 ADC12 的模块等可以用来设计仪器类的题目，比如宽带放大器。

第二章：片内主要模块介绍

MSP430 丰富的片内模块功能齐全，操作方便。本章介绍 MSP430 单片机片内模块的结构，原理和功能，列举相关模块的应用实例，源程序以 C 语言为主。通过相关模块寄存器的介绍以及示例程序，可以较方便并快速的熟悉并使用 MSP430。另外，理解源程序时，需要结合附录——MSP430 系列单片机头文件内容。系统资源（MSP430F449）框图如



2.1 时钟模块

要熟练使用 MSP430，就得先熟悉其时钟系统，稳定适合的时钟是单片机正常运行的必需条件。另外，MSP430 系列单片机的众多片内模块都需要时钟源进行驱动。MSP430F449 单片机时钟模块由高速晶体振荡器、低速晶体振荡器、数字控制振荡器 DCO、锁相环增强版本 FLL+构成。这些给 MSP430F449 产生了三个时钟源，分别是：

a. LFXT1CLK 低频时钟，一般由 32768Hz 晶振产生；

b. XT2 CLK 高频时钟，一般由 8MHz 晶振产生；

c. DCOCLK 片内数字控制 RC 振荡器，经常用作系统和外设的时钟信号，其稳定性由 FLL+硬件控制。

2.1.1 MSP430F449 的三个时钟源可以提供四种时钟信号

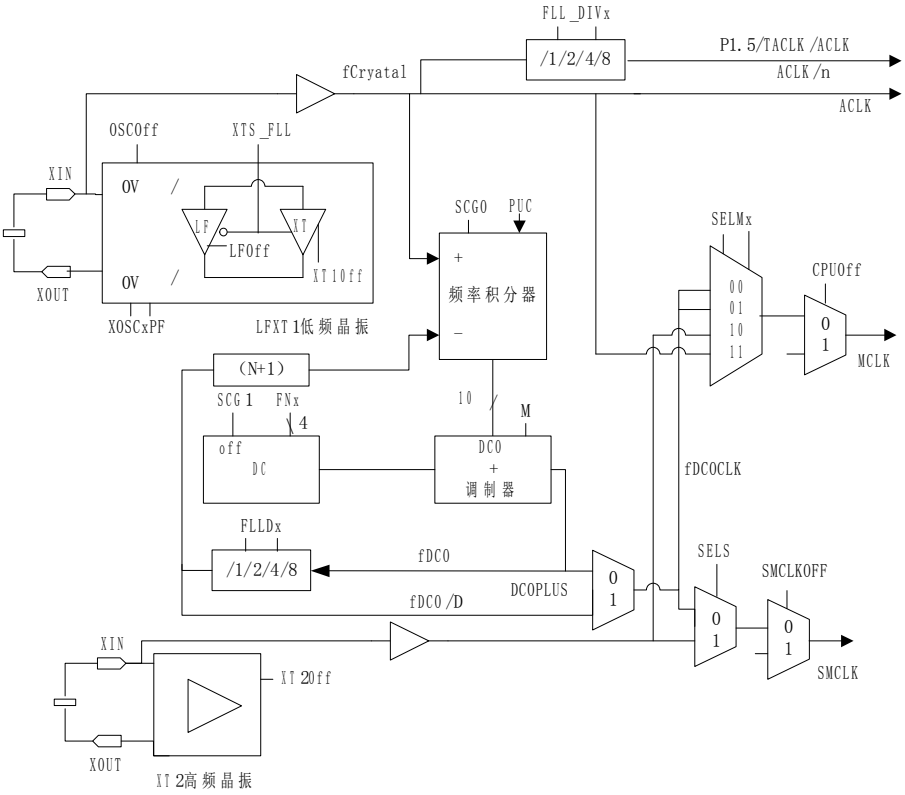
ACLK 辅助时钟：ACLK 来自 LFXT1CLK 信号，ACLK 可由软件选作各个外围模块的时钟信号，一般用于低速外设，比如显示器，键盘扫描，低速数模、模数转换器等。

ALK/n：ACLK 经 1、2、4、8 分频后由 P1.5 输出，仅供外部电路使用。

MCLK 系统主时钟：MCLK 可由软件选择来自 LFXT1CLK、XT2CLK 和 DCOCLK 三者之一，然后经 1、2、4、8 预分频得到。MCLK 主要用于 CPU 和系统。

SMCLK 子系统时钟：可由软件选择来自 XT2CLK 和 DCOCLK。SMCLK 主要用于高速外围模块。

图** MSP430F44X 时钟模块



2.1.2 MSP430F449 时钟模块寄存器

MSP430F449S 时钟模块寄存器如表 2-1 所示

表 2-1 MSP430F449 时钟模块寄存器

寄 存 器	缩 写 形 式	类 型	地 址	初 始 状 态
系统时钟控制寄存器	SCFQCTL	读写	52H	01FH
系统时钟频率积分寄存器 0	SCFI0	读写	50H	040H
系统时钟频率积分寄存器 1	SCFI1	读写	51H	复位
FLL+控制寄存器 0	FLL+CTL0	读写	53H	003H
FLL+控制寄存器 1	FLL+CTL1	读写	54H	复位

这些控制寄存器都是字节形式的，必须以字节指令来访问。

（1）SCFQCTL 系统时钟控制寄存器

寄存器 SCFQCTL 的位与含义如下：

表 2-2 寄存器 SCFQCTL 的位定义

7	6	5	4	3	2	1	0
M	N						

M 调节器使能控制

0 调制器使能；

1 调制器禁止。

N DCOCLK 倍数

如果 DCO+=0, $f_{DCOCLK}=(N+1)*f_{CRYSTAL}$

如果 DCO+=1, $f_{DCOCLK}=D*(N+1)*f_{CRYSTAL}$

复位信号后，SCFQCTL 的默认值为 31，DCO+=0。

(2)SCFI0　系统频率积分寄存器 0

表 2-3　寄存器 SCFI0 的位定义

7	6	5	4	3	2	1	0
FLLDx		FN_8	FN_4	FN_3	FN_2	MODx(LSBx)	

MODx(LSBx)是十位 DCOCLK 频率调整参数的最后两位，含义见 SCFI1 寄存器部分。这十位 DCOCLK 频率调整参数由 FLL+硬件自动完成。

FLLDx　FLL+环分频系数，DCOCLK 在 FLL+反馈环中被分频。

00　不分频；

01　2 分频；

10　4 分频；

11　8 分频。

FN_x　DCOCLK 频率的调整范围控制，如表 2-4 所示。

表 2-4　频率的可调范围

FN_8	FN_4	FN_3	FN_2	f_{DCOCLK}/MHz
0	0	0	0	0.65~6.1
0	0	0	1	1.3~12.1
0	0	1	x	2~17.9
0	1	x	x	2.8~26.6
1	X	x	x	4.2~46

(3)SCFI1　系统频率积分寄存器 1

表 2-5　寄存器 SCFI1 的位定义

7	6	5	4	3	2	1	0
DCOx					MODx(MSBx)		

DCOx　DCOCLK 频率周期控制，这 5 位控制 DCOCLK 频率周期的 29 种组合，每一个组合比前一个组合高出 10%。

MODx　调制器控制位的高 3 位，低 2 位在 SCFI0 中，这 3 位控制 32 种可能的周期混合方式。

(4)FL_CTL0　FLL+控制寄存器 0

表 2-6　寄存器 FL_CTL0 的位定义

7	6	5	4	3	2	1	0
DCO+	XTS_FLL	OscCap		XT2OF	XT1OF	LFOF	DCOF

DCO+　选择 DCO 用作 MCLK 或 SMCLK 前是否需要预分频。

0　不分频；

1　分频。

XTS_FLL　选择 LFXT1 模式。

0　低频模式；

1　高频模式。

OscCap　选择振荡器电容。

00　0~2 PF;

01　1~6 pF;

10　2~8 pF;

11　　3~10 pF。

XT2OF　XT2 振荡器失效标志。注意，在 MSP430F41X/42X 中没有 XT2。

0　　没有失效；

1　　失效。

XT1OF　LFXT1 振荡器在高频模式下失效标志。

0　　没有失效；

1　　失效。

XLOF　LFXT1 振荡器在低频模式下失效标志。

0　　没有失效；

1　　失效。

DCOF　DCO 振荡器失效标志。

0　　没有失效；

1　　失效。

(5)FLL_CTL1 FLL+控制寄存器 1

各位定义如下：

表 2-7　寄存器 FLL_CTL1 的位定义

7	6	5	4	3	2	1	0
未用	SMCLKOFF	XT2OFF	SELM		SELS	FLL_DIV	

FLL_DIV　LFXT1 频率的分频因子。

00　　不分频；

01　　2 分频；

10　　4 分频；

11　　8 分频。

SELS　选择 SMCLK 时钟源。

0　　DCOCLK；

1　　XT2XLK。

SELM　选择 MCLK 时钟源。

0，1　DCOCLK;

2　　XT2CLK;

3　　LFXT1CLK。

XT2OFF　关闭 XT2 振荡器。如果 XT2 没有被用做 MCLK 或者 SMCLK，则关闭 XT2.

0　　打开 XT2;

1　　关闭 XT2。

SMCLKOFF　关闭时钟信号 SMCLK。无需使用 SMCLK 进行高速外设驱动时，可以关闭，降低系统功耗。

0　　打开 SMCLK;

1　　关闭 SMCLK。

2.1.3　FLL+模块应用举例

例 2-2　设 LFXT1=32768Hz，XT2CLK=8M。设计系统主时钟 MCLK=4MHz，系统辅助时钟 ACLK=32768Hz，系统子时钟 SMCLK=8M。三个时钟信号分别通过 P1.1，P1.4，P1.5 输出。

分析：ACLK=LFXT1=32768Hz，MCLK=DCOCLK=4M,则 N=4M/32768-1=63,SMCLK=XT2CLK=8M。

程序如下：

```
#include <msp430x44x.h>
void main()
{
    unsigned int i;
    WDTCTL = WDTPW + WDTHOLD;          //关闭看门狗
    SCFQCTL=63;                        //N=63，DCOCLK=32768*(N+1)*2=4MHz
    SCFI0 |=FN_4;
    FLL_CTL0 |=DCOPLUS + OSCCAP1;      //DCO+=1,LFXT1 低频模式,振荡电容 1~6pF
    P5DIR |= BIT1 + BIT4 + BIT5;       //时钟输出
    P5SEL |= BIT1 + BIT4 + BIT5;       //端口作为特殊功能（见 2.3）
    FLL_CTL1 |=SELS;                   //SMCLK 时钟源 XT2CLK,MCLK 时钟源 DCOCLK
    FLL_CTL1 &=~XT2OFF;                //XT2 有效
    do
    {
        IF1 &=~OFIFG;                  //清除振荡器失效标志位
        for(i = 0;i<0xff;i++);         //稳定时间
    }
    while((IFG2 & OFIFG) ! = 0);
    while(1);
}
```

2.2 低功耗结构

TI 的 MSP430 是一个特别强调低功耗的单片机系列，尤其适用于采用电池供电的长时间工作场合。

MSP430 应用系统价格和电流消耗等因素会影响 CPU 与外围模块对时钟的需求，所以系统使用不同的时钟信号:ACLK、MCLK、SMCLK。用户通过程序可以选择低频或高频，这样可以根据实际需要来选择适合的系统时钟频率，这 3 种不同频率的时钟输出给不同的模块，从而更加合理的利用系统的电源，实现整个系统的低功耗。这一点对于电池供电的系统来讲至关重要。

2.2.1 系统工作模式

控制位 SCG1、SCG2、OscOff、CPUOff 可由软件配置成 6 种不同工作模式：1 种活动模式和 5 种低功耗模式。通过设置控制位 MSP430 可以从活动模式进入到相应的低功耗模式；而各种低功耗模式又可以通过中断方式回到活动模式。

MSP430 工作模式通过控制位设置。在各种工作模式下，时钟系统所产生的 3 种时钟活动状态时各不相同的。表 2-3 反映了各种工作模式、各控制位及 3 种时钟的活动状态之间的关系。

表 2-8 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
活动模式（AM）	SCG1=0 SCG0=0	CPU 处于活动状态 MCLK 活动

	OscOff=0 CPUOff=0	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，则支流发生器被禁止，否则仍保持活动 MCLK 被禁止 SMCLK 活动 ACLK 活动
低功耗模式 2（LPM2）	SCG1=1 SCG0=0 OscOff=0 CPUOff=1	CPU 处于禁止状态 如果 DCO 未用作 MCLK 或 SMCLK，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，支流发生器被禁止 MCLK 被禁止 SMCLK 被禁止 ACLK 活动
低功耗模式 4（LPM4）	SCG1=X SCG0=X OscOff=1 CPUOff=1	CPU 处于禁止状态 MCLK 被禁止 SMCLK 被禁止 ACLK 被禁止 所有振荡器停止工作

2.2.2 低功耗应用原则

一般的低功耗原则：

(1)最大化 LPM3 的时间，用 32768Hz 晶振作为 ACLK 时钟，DCO 用于 CPU 激活后的突发短暂运行

(2)用接口模块代替软件驱动功能；

(3)用中断控制程序运行；

(4)用可计算的分支代替标志位测试产生的分支；

(5)用快速查表代替冗长的软件计算；

(6)在冗长的软件计算中使用单周期的 CPU 寄存器；

(7)避免频繁的子程序和函数的调用；

(8)尽可能的直接用电池供电。

此外，再设计外设时还有一些常规原则：

(1)将不用的 FETI 输入端连接到 V_{ss}；

(2)JTAG 输入端 TMS、TCK、和 TDI 不要连接到 V_{ss}；

(3)CMOS 输入端不能有浮空节点，将所有输入端接适合的电平；
(4)不论对于外核还是对于外围模块，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。

2.3 I/O 端口

MSP430 的端口资源很丰富 P1~P6、S、COM，其中 P1~P6 端口都是既可以字节寻址又可以位寻址的，也就是说端口的每一位都可以独立用于输入输出。

2.3.1 MSP430 的端口

MSP430 的端口功能丰富，如表 2-9 所示。

表 2-9 MSP430F449 端口功能

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

2.3.2 端口数据输出特性

微处理器输入端口的漏电流对系统的耗电影响很大。MSP430 单片机输入端口的最大漏电流为 50nA，远低于其他系列单片机（一般为 1~10uA）。

不管是灌电流还是拉电流，每个端口的输出晶体管都能限制输出电流最大约 6mA，保证系统安全。

2.3.3 端口 P1 和 P2

端口 P1 和 P2 具有输出/输入、中断和外部模块功能，这些功能可以通过它们各自的 7 个控制寄存器的设置来实现。下面 Px 代表 P1 或 P2。

一、PxDIR 输入/输出方向寄存器

相互独立的 8 位分别定义了 8 个应缴的输入输出方向。8 位在 PUC 后都被复位，即输入状态。使用输入和输出功能时，应该先定义端口的方向，输入/输出才能满足设计者的要求。作为输入时只能读，输出时可读可写。

0 输入模式；
1 输出模式。

```
P1DIR  | = BIT0;          //P1.0 位操作输出
P1DIR  &=~BIT1;          //P1.1 位操作输入
P2DIR  =  0xff;          //P2 口字节操作输出
P3DIR  =  0x00;          //P3 口字节操作输入
```

注意程序中的按位或“|”和按位与“&”操作，这种位操作的设置的结果是每个位都是独立的，不会影响字节中的其他位的状态。

二、PxIN　输入寄存器

输入寄存器是只读寄存器。用户不能对其写入，只能通过读取该寄存器内容知道 I/O 端口的输入信号。此时引脚方向必须选定为输入。

三、PxOUT　输出寄存器

该寄存器为 I/O 端口的输出缓冲寄存器，在读取输出缓存的内容与引脚方向定义无关。改变方向寄存器的内容，输出缓存的内容不受影响。

```
P2OUT |=BIT7;              //P2.7 输出高电平
P2OUT &=~BIT6;             //P2.6 输出低电平
```

四、PxIFG　中断标志寄存器

该寄存器有 8 个标志位，标志相应的引脚是否有待处理的中断请求。

0　没有中断请求；

1　有中断请求。

五、中断触发沿寄存器

如果允许 Px 口的某个引脚中断，还需定义该引脚的中断方式。该寄存器的 8 位分别对应 Px 口的 8 个引脚。

0　上升沿使相应标志置位；

1　下降沿使相应标志置位。

注意：Px 中断的敏感性，只有跳变才能引起中断请求，而静电平不能。改变 PxIES 可能使相应标志置位，所以改变 PxIES 之后需要清除标志位。

六、PxIE　中断使能寄存器

Px 口的每一个引脚都有一位用以控制该引脚是否允许中断。

0　禁止中断；

1　允许中断。

七、PxSEL　P1 和 P2 两端口还有其他片内外设功能，为减少引脚，将这些功能与芯片外的联系通过复用 P1 和 P2 引脚的方式来实现。

0　选择引脚为 I/O 口；

1　选择引脚为外围模块端口。

2.3.4　端口 P3、P4、P5 和 P6

端口 P3、P4、P5 和 P6 没有中断能力，其余功能同 P1 和 P2，可以实现输入/输出功能和外围模块功能。除去与中断相关的 3 个寄存器，其他寄存器均与 P1 和 P2 的相同。具体用法见 2.3.3。

2.3.5　端口 COM 和 S

这些端口实现与液晶片的直接接口。COM 端口为液晶片的公共端，S 端口为液晶片的段码端。液晶片输出端也可以经软件配置为输出端口。

2.4　看门狗定时器

看门狗定时器 WDT 是 MSP430 系列单片机中常用的一种部件。在工业现场，往往会由于

供电电源、空间电磁干扰或其他原因引起强烈的干扰噪声。这些干扰作用于数字器件，极易使其产生误动作，引起MSP430发生“程序跑飞”事故。若不进行有效处理，程序就不能回到正常工作状态，从而失去应有的控制功能。看门狗定时器正是为了解决这类问题而产生的，尤其是在具有循环结构的程序任务中更为有效。在正常操作期间，一次WDT定时时间到，将产生一次期间复位。如果通过编制程序使WDT定时时间稍大于程序运行一遍所用时间，并且程序执行过程中都有对WDT清零的指令，使计数器重新计数，则程序正常运行时，就回在WDT定时间到达之前对WDT清零，不会产生WDT溢出，如果由于干扰使程序跑飞，则不会再WDT定时时间到达之前执行计数器清零指令。WDT就回产生溢出，从而产生系统复位，CPU需要重新运行用户程序，这样程序就可以又恢复正常运行状态。

MSP430看门狗除了具有上述系统监测的特定用途之外，还可以作为内部定时器来使用，当选择的时间到达以后，和其他定时器一样产生一个定时中断。此外WDT还可以被完全停止活动以支持超低功耗应用。

1. WDT寄存器

（1）计数单元WDTCNT

WDTCNT是16位增计数器，由MSP430所选定的时钟电路产生的固定周期脉冲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，那么从开始计数到技术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。

（2）控制寄存器WDTCTL

WDTCTL由两部分组成：高8位被用作口令，低8位是对WDT操作的控制命令。要写入操作WDT的控制命令，由于处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，可直接读取地址120H中的内容，读出WDTCTL的低字节，高字节始终为69H。WDTCTL除了看门狗定时器的控制位之外，还有两个位用于设置NMI引脚功能。

下面是WDTCTL寄存器各位的定义：

15~8	7	6	5	4	3	2	1	0
口令	HOLD	NMIED	NMI	TMSEL	CNCTN	SSEL	IS1	IS0

IS0,IS1 选择看狗定时器的定时输出。其中T是WDTCTL的输入时钟源周期。

0 $T\times2^{15}$

1 $T\times2^{13}$

2 $T\times2^{9}$

3 $T\times2^{6}$

SSEL 选择WDTCNT的时钟源

0 SMCLK

1 ACLK

由IS0、IS1及SSEL3位便可确定WDT定时时间，因此通过软件对计数器设置不同的初始值就可以实现不同时间的定时。与其他定时器不同之处在于，WDT最多只能定时8种和时钟源相关的时间。表****列出了WDT可选的定时时间（LFXT1=32768Hz，SMCLK=1MHz）。

SSEL	IS1	IS0	定时时间/ms	
0	1	1	0.064	$T_{smclk}\times2^{6}$
0	1	0	0.51	$T_{smclk}\times2^{9}$
1	1	1	1.95	$T_{aclk}\times2^{6}$

0	0	1	8.19	$T_{smclk}\times2^{13}$
1	1	0	15.63	$T_{aclk}\times2^{9}$
0	0	0	32.77	$T_{smclk}\times2^{15}$
1	0	1	250	$T_{aclk}\times2^{13}$
1	0	0	1000	$T_{aclk}\times2^{15}$

CNCTL　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。

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。

2. WDT 的操作

用户可以通过 WDTCTL 寄存器中的 TMSEL 和 HOLD 控制位设置 WDT 工作的看门狗模式，定时器模式和低功耗模式。

（1）看门狗模式

由于在上电复位或者系统复位时，WDTCTL 和 WDTCNT 两寄存器内容被全部清除。上述情况将导致 WDT 的运行并自动进入看门狗模式。所以，用户软件一般都需要进行如下操作：

a. 进行 WDT 的初始化，设置合适的时间。

b. 周期性地堆 WDTCNT 清零，防止 WDT 溢出，保证 WDT 的正确使用。

在看门狗模式下，如果计数器超过了定时时间，就会产生复位和激活系统上电清除信号，系统从上电复位的地址启动。

如果系统不用看门狗功能，应该在程序开始处禁止看门狗功能。

```
WDTCTL = WDTHOLD + WDTPW;
```

（2）定时器模式

WDTCTL 的 TMSEL 位置位选择定时器模式。这一模式产生选定时间的周期性中断。定时时间可以通过 WDTCTL 的 CNCTL 位置来开始。

注意：定时器模式下定时时间的改变。

a.改变定时时间而不同时清除 WDTCNT 将导致不可预料的系统立即复位或中断。定时时间改变应伴随计数器清除，并在一条指令中完成。例如：

```
WDTCTL = WDTPW + WDTCNCTL + WDTIS0;
```

b.如果先后分别进行清除的定时时间选择，则可能立即引起不可预料的系统复位或中断。

c.另外，在正常工作时，改变时钟源可能导致 WDTCNT 额外的计数时钟。

（3）低功耗模式

当系统不需要 WDT 做看门狗和定时器时，可关闭 WDT 以减小功耗。

控制位 HOLD=1 关闭 WDT，这是看门狗停止工作。

3.看门狗定时器的中断控制功能

看门狗定时器用到特殊功能寄存器 SFR 地址的两位：

a. 中断标志 WDTIFG 位于 IFG1.0,初始状态为复位。

b. 中断允许 WDTIE 位于 IE1.0,初始状态为复位。

需要注意的是，定时器模式下中断优先级要低于看门狗模式，定时器模式下中断是可屏蔽的，而看门狗模式下的中断是不可屏蔽的。

3. 看门狗应用举例

例***：设 ACLK=32768Hz,SMCLK=1MHz。设计使用看门狗定时器功能产生一个方波。

```
#include <MSP430x14x.h>
void main()
{
    WDTCTL = WDT_MDLY_32;        //定时周期 32ms，见头文件定义
    IE1 |=WDTIE;                 //使能 WDT 中断
    P1DIR |=0x01;                //P1.0 输出
    _EINT();                     //全局中断允许
    For(;;)
    {
        _BIS_SR_(CPUOff);        //进入 LPM0;
        _NOP();
    }
}
#pragma vector =WDT_VECTOR
__interrupt void watchdog_timer(void)
{
    P1OUT ^=0x01;                //P1.0 取反
}
```

2.5 定时器 A

定时器 A 是由一个 16 位定时/计数器和 3 个捕获/比较通道组成。各捕获/比较通道可以单独控制，且具有丰富完善的中断控制寄存器。中断可以由计数器溢出或者捕获/比较事件触发。

定时器 A 具有如下特点：

(1)异步 16 位计数器，4 种工作模式。
(2)多种可选可配置计数器时钟源。
(3)3 个可配置捕获/比较寄存器。
(4)输出模式可选，支持 PWM 波输出。
(5)具有异步输入输出锁存功能。
(6)完善的中断服务功能。

2.5.1 Timer_A 的结构

结构 Timer_A 由计数器、捕获/比较和输出三大部分组成，如图 2-1 所示。

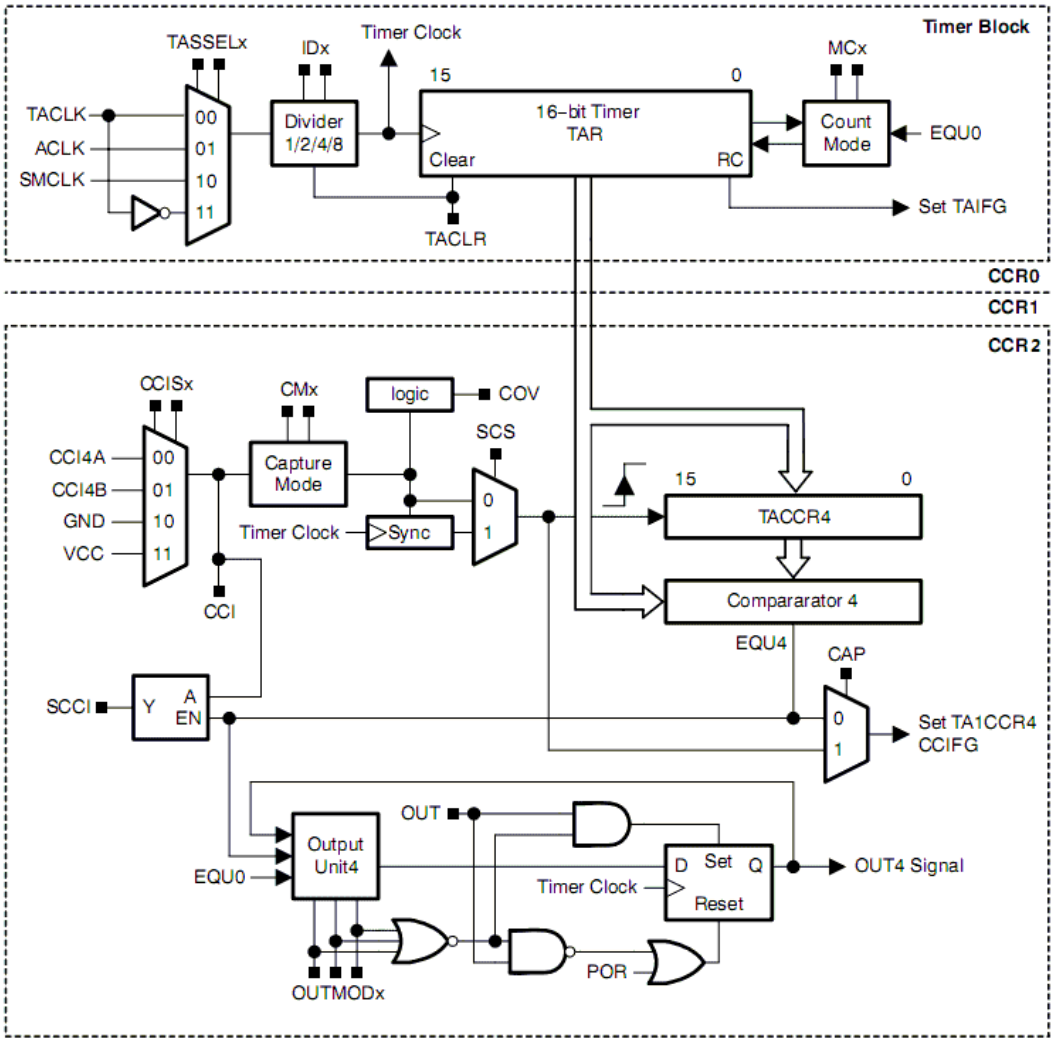


图 2-1　Timer_A 的结构

一、计数器部分

计数器部分完成计数功能。计数的时钟源选择、时钟分频系数、计数模式控制及计数主模块都在此部分。计数时钟源有 3 种选择，选定的时钟源经过分频后作为计数时钟，按照设定的计数模式完成定时/计数功能。

二、捕获/比较部分

捕获/比较部分用于捕获事件发生的时间或者产生定时时间间隔。 MSP430x449 的 Timer_A 中包含 3 个完全相同的捕获/比较器，工作模式和输入输出完全独立，各受自己的控制寄存器控制。捕获/比较功能的引入，加以中断控制可以提高 I/O 端口处理事务的能力和效率。

三、输出部分

输出部分用于产生用户所需的输出信号。Timer_A 有 8 种可选的输出模式，支持 PWM 波输出。输出部分工作只需要寄存器的预先设置，工作过程中不需要软件干涉。这样可以实现精准的定时和波形输出。

2.5.2 寄存器

用户对 Timer_A 的所有操作都是通过对相应的寄存器设置来实现。MSP430x44x 单片机包含的寄存器如表 2-10 所示。

表 2-10 MSP430x44x 单片机的寄存器

Register	Short Form	Register Type	Address	Initial State
Timer_A control Timer0_A3 Control	TACTL/ TA0CTL	Read/write	0160h	Reset with POR
Timer_A counter Timer0_A3 counter	TAR/ TA0R	Read/write	0170h	Reset with POR
Timer_A capture/compare control 0 Timer0_A3 capture/compare control 0	TACCTL0/ TA0CCTL	Read/write	0162h	Reset with POR
Timer_A capture/compare 0 Timer0_A3 capture/compare 0	TACCR0/ TA0CCR0	Read/write	0172h	Reset with POR
Timer_A capture/compare control 1 Timer0_A3 capture/compare control 1	TACCTL1/ TA0CCTL1	Read/write	0164h	Reset with POR
Timer_A capture/compare 1 Timer0_A3 capture/compare 1	TACCR1/ TA0CCR1	Read/write	0174h	Reset with POR
Timer_A capture/compare control 2 Timer0_A3 capture/compare control 2	TACCTL2/ TA0CCTL2	Read/write	0166h	Reset with POR
Timer_A capture/compare 2 Timer0_A3 capture/compare 2	TACCR2/ TA0CCR2	Read/write	0176h	Reset with POR
Timer_A interrupt vector Timer0_A3 interrupt vector	TAIV/ TA0IV	Read only	012Eh	Reset with POR

一、TACTL： 定时器 A 控制寄存器

定时器 A 控制寄存器 TACTL 中包含定时器及其操作的控制位。POR 信号后定时器 A 控制寄存器 TACTL 的所有位都自动复位，但是在 PUC 信号后不受影响。其各位定义如下（注：rw 表示可读可写，（0）表示 POR 复位后该位为 0。下同）：

表 2-11 控制寄存器 TACTL 的位定义

15	14	13	12	11	10	9	8
Unused						TASSELx	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

7	6	5	4	3	2	1	0
IDx		MCx		Unused	TACLR	TAIE	TAIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	w-(0)	rw-(0)	rw-(0)

TASSEL1，TASSEL0：BIT9，BIT8。定时器时钟源选择位。

00：TACLK

01：ACLK

10：SMCLK

11：Inverted TACLK

ID1，ID0：BIT7，BIT6。输入时钟源分频选择位。

00： /1

01： /2

10： /4

11： /8

由 TASSEL1 和 TASSEL0 两位选择时钟源，然后由 ID1, ID0 选择分频系数将输入时钟分频，分频后的信号作为计数器时钟。

MC1，MC0：BIT5~BIT4。计数模式控制位。

00：停止计数模式

01：增计数模式

10：连续计数模式

11：增减计数模式

TACLR：BIT2。定时器清除位。

0： 不清除

1： 清除

将该位置 1 将会清除 TAR、时钟分频系数和计数方向。TACLR 由硬件自动复位，其读出值始终为 0。

TAIE：BIT1。定时器 A 中断允许控制位。该位使能 TAIFG 的中断请求。

0： 中断禁止

1： 中断允许

TAIFG：BIT0。定时器 A 中断标志位。

0： 定时器 A 无中断请求

1： 定时器 A 有中断请求

二、TAR： 16 位计数器。

该寄存器就是执行技术的单元，是计数器的主体，其内容可读可写。POR 信号后 TACTL 的所有位都自动复位。为避免时钟竞争，在修改 TAR 的值时，应该先停止计数器计数，修改完成后再启动计数器计数。

三、TACCTLx： 捕获/比较控制寄存器 x。

Timer_A 有多个捕获/比较模块，每个模块都有自己的控制字寄存器 TACCTLx，这里 x 为模块序号。POR 信号后 TACTL 的所有位都自动复位，但是在 PUC 信号后不受影响。该寄存器各位意义如下：

表 2-12 捕获/比较控制寄存器 TACCTLx 的位定义

15	14	13	12	11	10	9	8
CMx		CCISx		SCS	SCCI	Unused	CAP
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	r0	rw-(0)

7	6	5	4	3	2	1	0
OUTMODx			CCIE	CCI	OUT	COV	CCIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	rw-(0)

CM1，CM0：BIT15~BIT14。捕获方式选择位。

00：禁止捕获

01：上升沿捕获

10：下降沿捕获

11：上升沿下降沿都捕获

SSIS1，CCIS0：BIT13，BIT12。捕获事件输入源选择位。

00：CCIxA

01：CCIxB

10：GND

11：Vcc

SCS：BIT11。捕获信号与定时器时钟同步选择位。

0： 异步捕获

1： 同步捕获

异步捕获模式允许在请求是立即将 CCIFG 置位且捕获定时器的值，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，则捕获寄存器可能出错。实际中经常使用异步捕获模式，且捕获总是有效的。

SCCI：BIT10。同步比较/捕获输入。必较相等信号 EQUx 将选定的捕获/比较输入信号 CCIx（CCIxA, CCIxB, GND, Vcc）进行锁存，可由 SCCI 读出。

CAP：BIT8。工作模式选择位。

0： 比较模式

1： 捕获模式

如果改变 CAP 使工作模式由比较变为捕获，就不应该同时捕获，因为这时捕获到的值是不可预料的。

OUTMODx：BIT7~BIT5。输出模式选择位。由于 EQUx=EQU0，模式 2，3，6 和 7 不适用于 TACCR0 操作。

000：输出

001：置位

010：翻转/复位

011：置位/复位

100：翻转

101：复位

110：翻转/置位

111：复位/置位

CCIE：BIT4.。捕获/比较模块中断允许位。

0： 禁止中断

1： 允许中断

CCI：BIT3。捕获/比较模块的输入信号。选择的输入信号可以由此位读出。

OUT：BIT2.。输出。在输出模式 0，该位直接控制输出状态。

0： 输出低电平

1： 输出高电平

COV：BIT1。捕获溢出标志。

0： 没有捕获溢出

1： 发生捕获溢出

在捕获模式下，如果捕获寄存器的值被读出前再次发生捕获事件，则 COV 置位。读捕获寄存器不会使 COV 复位，须用软件复位。

CCIFG：BIT0。捕获/比较中断标志位。

0： 没有中断请求

1： 有中断请求

捕获模式，寄存器 CCRx 捕获了 TAR 的值时，CCIFG 置位。

比较模式，寄存器 CCRx 的值与 TAR 值相等时，CCIFG 置位。

四、TACCRx： 捕获/比较寄存器 x。

在捕获模式，当满足捕获条件时，硬件自动将计数器 TAR 中的数据写入到该寄存器，CCR0 经常作为周期寄存器，其他 CCRx 相同。

在比较模式，用户软件根据定时时间长短，配合计数器计数频率和工作方式，写入该寄存器相应的数据即可。

五、TAIV:　中断向量寄存器。

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

捕获/比较寄存器 CCR0 具有最高的优先级，因为 CCR0 能用于定义增计数和增减计数模式的周期寄存器，因此，它需要具有最快速的服务。CCIFG0 在中断服务程序响应时能够自动复位。CCR0 中断如图 2-2 所示。

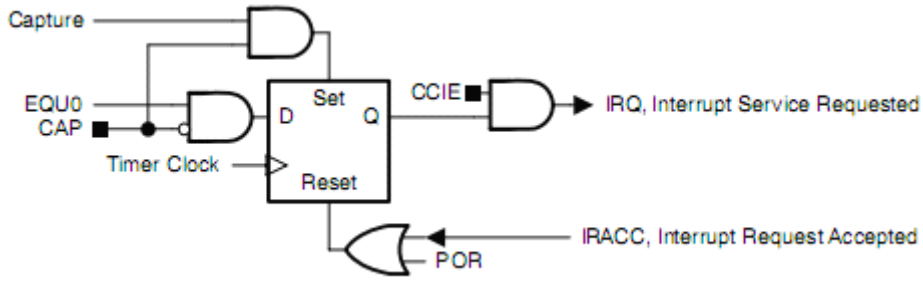


图 2-2　CCR0 中断示意图

CCR1~CCRx 和定时器共用同一个中断向量，输入多源中断。Timer_A 的中断向量寄存器 TAIV 用来确定中断请求的中断源。TAIV 为 16 位寄存器，其 15～4 位与 0 位全为 0，3～1 位的数据对应相应的中断源。具体如图 2-3 所示。

TAIV Contents	Interrupt Source	Interrupt Flag	Interrupt Priority
00h	No interrupt pending	-	
02h	Capture/compare 1	TACCR1 CCIFG	Highest
04h	Capture/compare 2	TACCR2 CCIFG	
06h	Capture/compare 3†	TACCR3 CCIFG	
08h	Capture/compare 4†	TACCR4 CCIFG	
0Ah	Timer overflow	TAIFG	
0Ch	Reserved	-	
0Eh	Reserved	-	Lowest

† Timer1_A5 only

图 2-3　中断向量表

CCIFG0 为单源中断标志，在相应的中断服务程序得到相应后会自动清除。对应 Timer_A 的多源中断标志 CCIFG1~CCIFx 和 TAIFG 在软件读中断向量寄存器 TAIV 后自动复位。如果不访问 TAIV，则硬件不会自动复位，须由软件清除。如果相应的中断允许复位（即不允许中断），则将不会产生中断请求而中断标志仍然存在，须由软件消除。

如果有 Timer_A 中断标志位置位，则 TAIV 为相应的数据，该数据与 PC 值相加则可使系统进入相应的中断服务程序。如果 Timer_A 有多个中断标志置位，则系统首先判断优先级在相应相应的中断。

2.5.3　计数模块

定时器共有 4 种计数工作模式：停止计数模式，增计数模式，连续计数模式和增减计数模式。工作模式的选择由 MCx 来决定。

一、停止计数模式（MC1=0,MC0=0）

定时器暂停，但是并不发生复位。当定时器暂停后重新开始计数时，计数器将下哦那个暂停的值开始按照暂停前的技术模式继续计数。如果这不是我们需要的，则可以通过 TACLR 来清除定时器的方向记忆特性。

二、增计数模式（MC1=0,MC0=1）

这种模式常用于周期小于 65536 个计数时钟周期的定时。Timer_A 增计数模式的周期寄存器是 CCR0。当 TAR 增计数到 CCR0 的值时（相等或大于），定时器复位并从 0 开始重新计数。如图 2-4 所示。

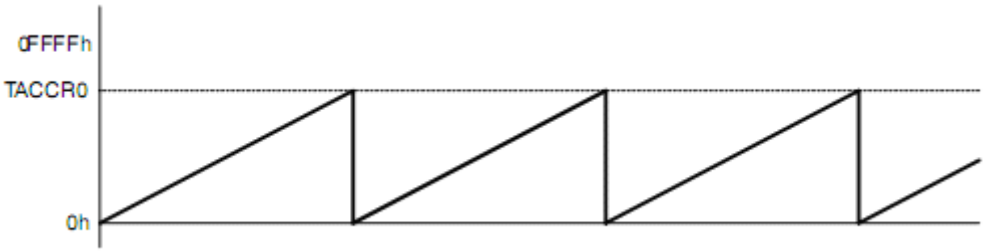


图 2-4　增计数模式

中断标志 CCIFG0 在 TAR 计数到 TACCR0 时置位，中断标志 TAIFG 在 TAR 从 TACCR0 返回到 0 时置位。中断标志的置位过程如图 2-5 所示。

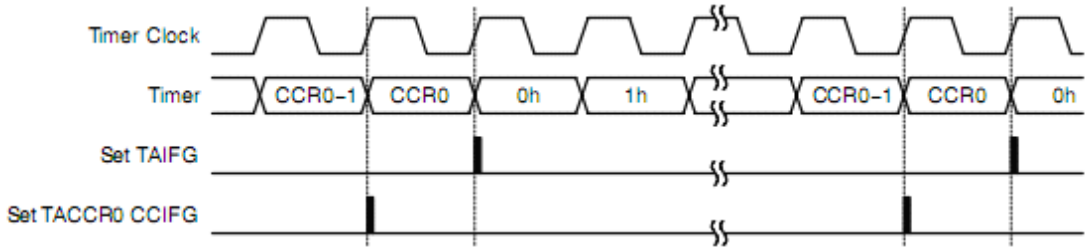


图 2-5　中断标志的置位过程

在计数器工作时，如果改变 CCR0 的值，会有不同的变化。如果新周期大于旧周期，或者大于当前的计数值，计数器将会直接计数到新周期。而如果新周期小于当前计数器的计数值，则定时器接受新周期并返回到 0 之前将会继续增加一个旧的计数周期。

三、连续计数模式（MC1=1,MC0=0）

在该模式下，定时器从当前值计数到 0FFFFH 后，又返回到 0000H 开始重新计数。如图 2-6。

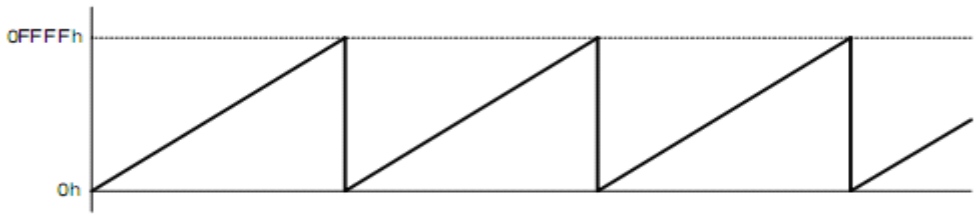


图 2-6　连续计数模式

同样是在TAR等于TACCRx时会置位TACCIFGx，当TAR由0FFFFH返回0000H时置位TAIFG。如图 2-7。

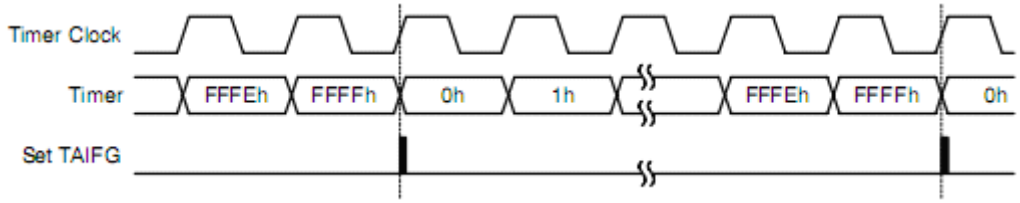


图 2-7　中断标志的置位过程

连续计数一般用于需要 65536 个时钟周期的应用场合。也可以通过设置和改变 TACCRx 的值产生独立时间间隔，输出一定频率的信号。例如，可以在相应的比较寄存器 CCRx 上加一个时间差，不断地增加 CCRx 的值，使定时器每到一定的时间间隔都能与 CCRx 相等，从而使中断标志位置位，向 CPU 申请中断，这个时间差是当前时刻到下一个中断发生时刻所经历的时间。如图 2-8 为用 TACCR0 和 TACCR1 产生的独立定时中断信号。

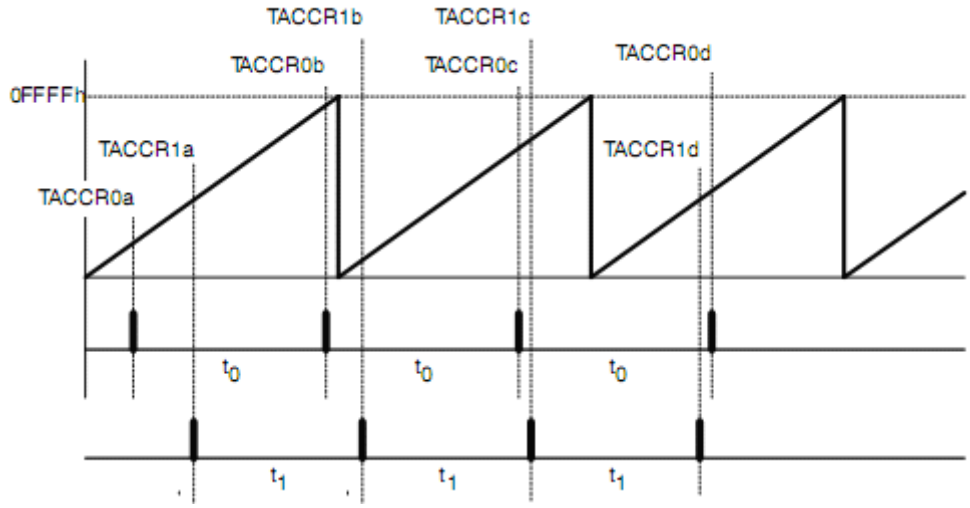


图 2-8　TACCR0 和 TACCR1 产生的独立定时中断信号

四、增减计数模式（MC1=1,MC0=1）

在该模式下，TAR 先增计数到 CCR0，然后反向减计数到 0。计数周期仍然由 CCR0 定义，但周期是 CCR0 计数值的 2 倍，所以该模式常用于生产对称波形的场合。计数周期如图 2-9 所示。

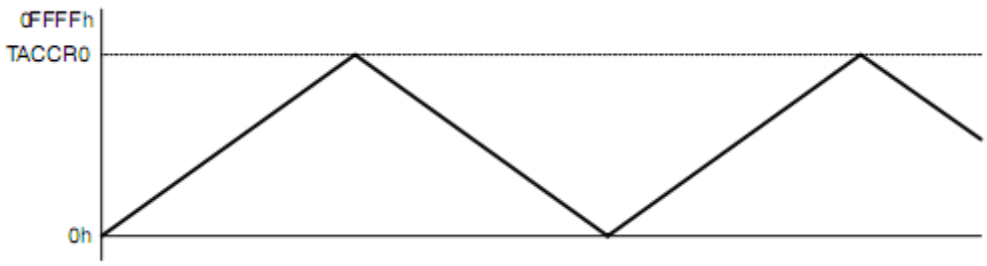


图 2-9　增减计数模式

如图 2-10，中断标志 TACCIFG0 会在 TAR 从 TACCR0-1 计数到 TACCR 时置位，TAIFG 会在 TAR 从 0001H 计数到 0000H 时置位。两者中断周期是相等的，仅存在相位上的差别。

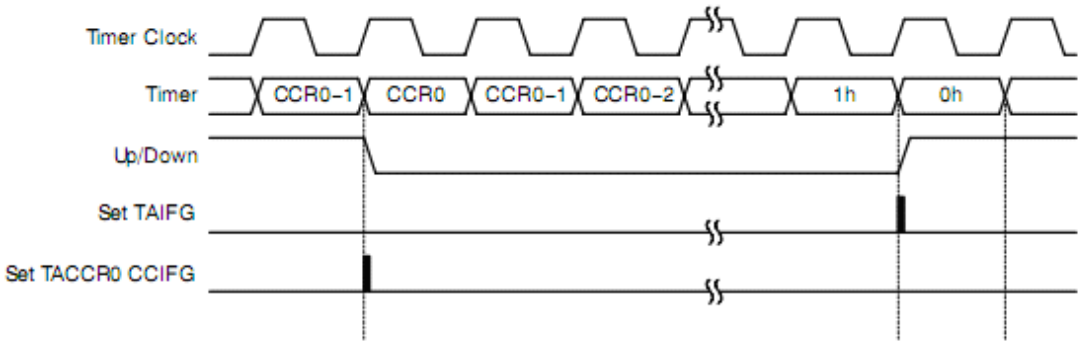


图 2-10　中断标志的置位过程

在计数器工作时，如果改变 CCR0 的值，会有不同的变化。如果计数器正在增计数，则情况与增计数模式相同。如果计数器正在减计数，则先减计数到 0000H 才开始新周期的计数。

2.5.4　捕获/比较模块

Timer_A 有多个相同的捕获/比较模块，为实时处理提供灵活的手段，每个模块都可用于捕获时间发生的时间或者产生定时间隔，同时可以引起中断。CCIS1 和 CCIS0 选择捕获出发输入源，CCMx1 和 CCMx0 选择捕获触发条件，捕获/比较寄存器与定时器总线相连，可在满足捕获条件时将 TAR 的值写入捕获寄存器，也可以在 TAR 的值与比较器的值相等时，设置标志位。

捕获模式和比较模式的选择可以通过**捕获/比较控制寄存器 CCTLx** 中位 **CAP** 的设置来实现。

一、捕获模式（CAP=1）

如果在选定的引脚上发生设定的脉冲边沿，则 TAR 中的值将写入到 TACCRx 中。所以捕获/比较寄存器常用于确定事件发生的时刻，如时间测量和频率测量等。捕获完成后，中断标志 TACCIFGx 将置位。如果捕获信号与定时器计数时钟同步，则可避免时间竞争。而如果不同步，则可能导致捕获数据无效，可以通过软件验证数据并加以校正。

二、比较模式（CAP=0）

在比较模式，与捕获有关的硬件停止工作，但计数器 TAR 的值计数到比较器中预先设定的值时就会置位标志位，产生中断请求，也可以结合输出单元产生所需的信号。该模式主要用于需要软硬件定时以及产生脉宽调制（PWM）输出的情况。

Timer_A 的 3 个捕获/比较器在比较模式时设置 EQUx 信号存在差别。如果 TAR 的值大于或等于 CCR0 中的值则 EQU0=1，如果 TAR 的值等于 CCR1 或 CCR2 中的值则 EQU1 或 EQU2=1。

三、输出模块

每个捕获/比较模块都包含一个输出单元，用于产生输出信号，比如 PWM 波。每个输出模块有 8 种输出模式，每个模式都是根据 EQU0 和 EQUx 来产生相应的输出信号。

除模式 0 外，去其它输出都在定时器时钟上升沿发生变化。输出模式 2，3，6，7 不适用于输出单元 0，因为 EQU0=EQUx。所有的输出模式定义如下：

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中的 OUT 位定义，并在 OUT 位写入位信息后立即更新。

输出模式 1：置位模式。输出信号 OUTx 在 TAR 等于 TACCRx 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：翻转/复位模式。输出在 TAR 的值等于 TACCRx 时翻转，当 TAR 的值等于 TACCR0 时复位。

输出模式3:置位/复位模式。输出在TAR的值等于TACCRx是置位,当TAR的值等于TACCR0时复位。

输出模式 4：翻转模式。输出在 TAR 的值等于 TACCRx 时翻转，输出信号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 TACCRx 时复位，并保持复位知道选择另一种输出模式。

输出模式6:翻转/置位模式。输出在TAR的值等于ACCR0时翻转,当TAR的值等于TACCR0时置位。

输出模式 7：复位/置位模式。输出电平在 TAR 的值等于 TACCRx 时复位，当 TAR 的值等于 TACCR0 时置位。

2.5.5 应用实例

一、基本定时（增计数）

利用增计数模式实现基本的定时功能，也可以利用连续计数和增减计数实现同样的功能。

例 2-2 设 ACLK=32.768KHz,MCLK=SMCLK=DCO=32×ACLK=1.048576MHz。设计利用 Timer_A 的增计数模式，通过 TAIFG 中断，由端口 P6.1 输出一个频率为 250Hz 的方波。

程序如下：

```
#include <msp430x44x.h>
void main()
{
    WDTCTL = WDTPW + WDTHOLD;
    FLL_CTL0 |= XCAP14PF;
    P6SEL &= ~BIT1;                  //I/O
    P6DIR |= BIT1;                   //Output
    TACTL  = TASSEL1 + TACLR + TAIE; //SMCLK=1.048576MHz，允许中断
    TACCR0 = 2097;                    //1048576Hz/2097=500Hz 中断频率
    TACTL |= MC0;                     //增计数.增计数模式总是以 CCR0 为周期的
    _EINT();
    while(1) {  LPM0;   }
}
#pragma vector = TIMERA1_VECTOR
__interrupt void TimerA1_ISR(void)
{
    switch(TAIV)
    {
    case 0x02: break;
    case 0x04: break;
    case 0x0A: P6OUT ^= BIT1;break;  //取反周期为 500Hz,方波频率 250Hz
    }
}
```

例 2-3 设 ACLK=32.768KHz，MCLK=SMCLK=DCO=32×ACLK=1.048576MHz。利用 Timer_A 的增计数模式，通过 TACCIFG0 中断，由端口 P6.1 输出一个频率为 250Hz 的方波。

程序如下：

```
#include <msp430x44x.h>
void main()
{
    WDTCTL = WDTPW + WDTHOLD;
    FLL_CTL0 |= XCAP14PF;
    P6SEL &= ~BIT1; //I/O
    P6DIR |= BIT1;  //Output
    TACTL   = TASSEL1 + TACLR;          //SMCLK=1.048576MHz，不允许中断
    TACCTL0 = CCIE;                     //允许中断
    TACCR0  = 2097;                     //1048576Hz/2097=500Hz 中断频率
    TACTL  |= MC0;                      //增计数.增计数模式总是以 CCR0 为周期
    _EINT();
    while(1) {  LPM0;   }
}
#pragma vector = TIMERA0_VECTOR
__interrupt void TimerA0_ISR(void)
{
    P6OUT ^= BIT1;                      //取反频率为 500Hz,方波频率 250Hz
}
```

二、独立时序信号（连续计数）

利用连续计数模式产生多个独立相位和频率的信号。

例 2-4 设 ACLK=32.768KHz，MCLK=SMCLK=DCO=32×ACLK=1.048576MHz。利用 Timer_A 的连续计数模式，由 P6.1 和 P6.2 分别输出一个频率为 250Hz 和 400Hz 的方波。

程序如下：

```
#include <msp430x44x.h>
void main()
{
    WDTCTL = WDTPW + WDTHOLD;
    FLL_CTL0 |= XCAP14PF;
    P6SEL &= ~(BIT1+BIT2); //I/O
    P6DIR |= (BIT1+BIT2);  //Output
    TACTL  = TASSEL1 + TACLR;       //SMCLK=1.048576MHz，不允许中断
    TACCTL1 = CCIE;                 //允许中断
    TACCTL2 = CCIE;                 //允许中断
    TACCR1 = 2097;                  //1048576Hz/2097=500Hz TACCIFG1 中断频率
    TACCR2 = 1311;                  //1048576Hz/1311=800Hz TACCIFG2 中断频率
    TACTL |= MC1;                   //连续计数
    _EINT();
    while(1) {  LPM0;   }
}
#pragma vector = TIMERA1_VECTOR
__interrupt void TimerA1_ISR(void)
{
```

```
        switch(TAIV)
        {
        case 0x02: P6OUT ^= BIT1; TACCR1+=2097;break; //取反周期为500Hz,方波频
率Hz
        case 0x04: P6OUT ^= BIT2; TACCR1+=1311;break; //取反周期为800Hz,方波频
率Hz
        case 0x0A: break;
        }
    }
```

三、对称波形（增减计数）

利用增减计数模式产生对称的波形输出。当TAR增计数到TACCR0时TACCIFGx中断，当TAR减计数到0时TAIFG中断。

例2-5 设ACLK=32.768KHz，MCLK=SMCLK=DCO=32×ACLK=1.048576MHz。利用Timer_A的增减计数模式，由P6.1输出以方波。

程序如下：

```
#include <msp430x44x.h>
void main()
{
    WDTCTL = WDTPW + WDTHOLD;
    FLL_CTL0 |= XCAP14PF;
    P6SEL &= ~BIT1; //I/O
    P6DIR |= BIT1;  //Output
    TACTL  = TASSEL1 + TACLR + TAIE; //SMCLK=1.048576MHz，允许中断
    TACCTL0 = CCIE;                  //允许中断
    TACCR0 = 2097;                   //1048576Hz/2097=500Hz TACCIFG1中断频
率
    TACTL |= MC1 + MC0;              //增减计数
    _EINT();
    while(1) {  LPM0;   }
}
#pragma vector = TIMERA0_VECTOR
__interrupt void TimerA0_ISR(void)
{
    P6OUT ^= BIT1;
}
#pragma vector = TIMERA1_VECTOR
__interrupt void TimerA1_ISR(void)
{
    switch(TAIV)
    {
    case 0x02: break;
    case 0x04: break;
    case 0x0A: P6OUT ^= BIT1;break;
    }
```

```
}
```

四、PWM 波（增计数，OUTx 输出）（附带说明 DA 实现）

例 2-6 设 ACLK=32.768KHz，MCLK=SMCLK=DCO=32×ACLK=1.048576MHz。利用 Timer_A 输出周期为 1/(1048576/5241)=1/2001=0.4998ms，占空比分别为 75%和 25%的 PWM 矩形波。

分析：PWM 波是一种具有固定周期和不同占空比的数字信号。如果 Timer_A 定时器工作在增计数模式，输出采用模式 7（复位/置位模式），则可以利用寄存器 TACCR0 控制 PWM 波的周期，用某个寄存器 TACCRx 控制占空比。这样 Timer_A 就可以产生可变占空比的 PWM 波形了。

程序如下：

```
#include <msp430x44x.h>
void main()
{
    WDTCTL = WDTPW + WDTHOLD;
    FLL_CTL0 |= XCAP14PF;
    TACTL = TASSEL1 + TACLR;     //时钟源选择 SMCLK=1.048576MHz
    TACCR0 = 524-1;              //周期/(1048576/5241)=1/2001=0.4998ms
    TACCTL1 = OUTMOD_7;          //输出模式，置位复位
    TACCR1 = 393;                //H:L = 393:(524-393) = 3:1
    TACCTL2 = OUTMOD_7;          //输出模式，置位复位
    TACCR2 = 131;                //H:L = 131:(524-131) = 1:3
    TACTL |= MC0;

    while(1) {  LPM0; }
}
```

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

五、

例 2-7：设 ACLK=32.768KHz，MCLK=SMCLK=200×ACLK=6.5536MHz。利用 Timer_A 输出频率为 50Hz 的正弦波。

分析：SPWM 就是在 PWM 的基础上改变了调制脉冲方式，脉冲宽度时间占空比按正弦规率排列，这样输出波形经过适当的滤波可以做到正弦波输出。它广泛地用于直流交流逆变器等，比如高级的 UPS 就是一个例子，可以由定时器 A 产生 SPWM 波。其中定时器 A 工作在比较模式，输出采用模式 3（置位/复位模式）。CCR0 确定了正弦波的频率，CCR1 代表 PWM 波的占空比，使 CCR1 的值呈正弦变化，则定时器 A 输出为 SPWM 波，通过滤波，即可得到正弦波。

程序如下：

```
#include <msp430x44x.h>

const unsigned int sin[256]=
{
128,130,133,135,138,140,143,145,148,150,152,155,157,159,162,164,166,169,171,
173,175,177,179,181,184,186,188,190,191,193,195,197,199,200,202,204,205,207,
```

```
208,210,211,212,214,215,216,217,218,219,220,221,222,223,224,224,225,226,226,
227,227,227,228,228,228,228,228,228,228,228,228,227,227,227,226,226,225,224,
224,223,222,221,220,219,218,217,216,215,214,212,211,210,208,207,205,204,202,
200,199,197,195,193,191,190,188,186,184,181,179,177,175,173,171,169,166,164,
162,159,157,155,152,150,148,145,143,140,138,135,133,130,128,126,123,121,118,
116,113,111,108,106,104,101,99,97,94,92,90,87,85,83,81,79,77,75,72,70,68,66,
65,63,61,59,57,56,54,52,51,49,48,46,45,44,42,41,40,39,38,37,36,35,34,33,32,
32,31,30,30,29,29,29,28,28,28,28,28,28,28,28,28,28,29,29,29,30,30,31,32,32,33,
34,35,36,37,38,39,40,41,42,44,45,46,48,49,51,52,54,56,57,59,61,63,65,66,68,
70,72,75,77,79,81,83,85,87,90,92,94,97,99,101,104,106,108,111,113,116,118,1
21,123,126
};  //256 点正弦波表

unsigned char count;
void main()
{
  WDTCTL=WDTPW+WDTHOLD;
  SCFQCTL=99;
  FLL_CTL0|=DCOPLUS;                    //频率为 32768*200=6.5536MHz

  TACTL=TASSEL1+TACLR;
  TACCR0=511;                          //正弦波频率为 6.5536MHz/(256*512)=50Hz
  TACCTL1=OUTMOD_3+CCIE;                //输出模式，置位/复位
  TACCR1=128;

  P1DIR|=0X04;
  P1SEL|=0X04;                         //定时器 A OUT1 输出

  count=0;
  _EINT();
  TACTL|=MC0;
  while(1) {  LPM0; };
}

#pragma vector=TIMERA1_VECTOR
__interrupt void time_a1(void)
{
  switch(TAIV)
  {
  case 2:{
  TACCR1=sin[count++];
  break;}
  case 4:break;
  case 10:break;
```

```
  default:break;
  }
}
```

2.6 液晶驱动

MSP430F449 片内具有段式液晶驱动模块。在液晶驱动电路中，液晶等效为电容。两个电极板分别为公共极与段极。公共极由 COM 口信号驱动，段极由 SEG 信号驱动。

一、MSP430 液晶驱动模块主要特点如下：

(1)具有显示缓存器。

(2)所需的 SEG,COM 信号自动产生。

(3)4 种驱动方法。

(4)多种扫描频率。

(5)段输出端口可以切换为通常输出端口。

(6)显示缓存器可作为一般存储器。

(7)用 ACLK 经 Basic Timer 产生频率。

二、液晶驱动方法

MSP430 液晶驱动模块由 4 种驱动方法，分别为静态驱动、2MUX 驱动、3MUX 驱动、4MUX 驱动，特点如表 2-13 所示。

表 2-13 液晶模块驱动方法

方法	公共极引脚数	每个引脚驱动液晶段数	需要引脚总数
静态	1	1	1+需要驱动的段数
2MUX	2	2	2+需要驱动的段数
3MUX	3	3	3+需要驱动的段数
4MUX	4	4	4+需要驱动的段数

4MUX 的操作比较方便，占用 S 口较少，下面通过示例介绍 4MUX 的驱动方法。

例 2-8 采用 4MUX 的驱动方式，一次显示 0~7 八个数字。

```
#include <msp430x44.h>
const  char  LCD[10]={
0XEB,            /*0  不同的液晶片的段式分配不同，需要根据具体情况进行修改*/
0X60,            /*1*/
0XC7,            /*2*/
0XE5,            /*3*/
0X6C,            /*4*/
0XAD,            /*5*/
0XAF,            /*6*/
0XE0,            /*7*/
0XEF,            /*8*/
0XED             /*9*/
};
void main()
{
```

```
        int i ;
        WTCTL = WDTPW + WDTOLD;
        FLL_CTL0 |=XCAP14PF;
        LCDCTL = LCDON + LCD4MUX + LCDP2;        //开启模块功能，4MUX 显示，LCDP2
表示 S0~S17
        BTCTL = BTFRFQ1;                         //驱动时钟设置
        P5SEL = 0XFC;                            //COM 口复用为外设引脚
        While(1)
        {
            for(i=0;i<7;i++)
            LCDMEM[i]=LCD[i];                //写入数据
        }

    }
```

2.7 串行通信模块的异步模式

2.7.1 MSP430 串行通信概述

串口是系统与外界联系的重要手段，在嵌入式斯通开发和应用中，经常需要使用上位机实现系统调试及现场数据的采集与控制。一般是通过上位机本身配置的串行口，通过通信技术，和嵌入式系统进行连接通信。

MSP430 系列的每一种型号都可以实现串行通信功能：

(1)USART 硬件直接实现；

(2)通过定时器软件实现。

MSP430x44x 系列单片机具有两个 USART 模块（USART0 和 USART1），USART 模块可以从任何一种低功耗模式 LPMx 开始自动工作。所有 USART 模块都可以实现两种通信方式：UART 异步通信和 SPI 同步通信。

USART 模块包含 4 个部分：

(1)波特率部分，控制串行数据接收和发送的速度；

(2)接收部分，接收串行输入的数据；

(3)发送部分，发送串行输出的数据；

(4)接口部分，完成并/串、串/并转换。

2.7.2 异步操作

MSP430 串行通信模式通过两个引脚，即接收引脚 URXD 和发送引脚 UTXD 与外界相连。

异步串行通信特点如下：

(1)异步模式，包括线路空闲/地址位通信协议；

(2)两个独立移位寄存器:输入移位寄存器和输出移位寄存器；

(3)传输 7 位或 8 位数据，可采用奇校验和偶校验或者无校验；

(4)从最低位开始的数据发送或者接收；

(5)可编程实现分频因子为整数或小树的波特率；
(6)独立的发送和接收中断；
(7)通过有效的起始位检测将 MSP430 从低功耗唤醒；
(8)状态标志检测错误或者地址位。

一、异步串行字符格式

异步通信字符格式由 4 部分组成，起始位、数据位、奇偶校验位和停止位

表 2-14　异步通信字符格式

ST	D0…D6	D7	AD	PA	SP	SP
Start bit	数据位	8^{th}data CHAR=1	MM=1	PENA=1	1^{ST}stop bit	2^{nd}stop bit SP=1

接收操作以收到有效起始位开始。起始位由检测 URXD 端口的下降沿开始，然后以 3 次采样多数表决的方法取值。如果 3 次采样至少 2 次是 0 才表明是下降沿，然后开始接收初始化操作，这一过程实现错误起始位的拒收和帧中数据的中心定位。MSP430 可以处于低功耗模式，通过上述过程识别正确起始位之后被唤醒（详见例 2-9），然后通过串行接口控制寄存器中设定的数据格式，开始接收，直到本帧采集完毕。

异步模式下，传送数据是以字符为单位来传送的。因为每个字符在起始位处可以通过起始位判别重新定位，所以传送时多个字符可以一个接一个的连续传送，也可以断续传送，并且同步时钟脉冲不传送到接收方，收发双方各自用自己的时钟源来控制发送和接受。

二、串行操作自动错误检测

USART 模块接收字符时，能够自动进行校验错误、帧错误、溢出错误和打断错误状态检测。各种检测错误的含义和标志如下：

(1)FE: 标志帧错误。当一个接受字符的停止位为 0 并被装入接受缓存，接收的为一个错误的帧，那么帧错误标志被置位。

(2)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输入缓存，发生校验错误，PE 置位。

(3)OE: 溢出错误标志。当一个字符写入接收缓存，前一个字符还没有被读出，这是前一个字符因被覆盖而丢失，发生溢出。

(4)BRK：打断检测标志。当发生一次打断同时 URXEIE 置位时，该为被置位，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 位低电平被视为打断。

当上述任何一种错误出现，为 RXERR 置位，表明有一个或多个出错标志被置位。如果 URXEIE=0，并且有错误被检测到，那么接收缓存不接收任何数据。如果 URXEIE=1，接收缓存接收字符，相应的错误检测标志置位，直到用户软件复位或者接收内容被读出，才能复位。

三、波特率的产生

在异步串行通信中，波特率是很重要的指标。它一般表示每秒钟传送二进制数码的位数。波特率反映了异步串行通信的速度。所以在进行异步通信时，波特率的产生是必须的。波特率发生器产生同步信号表明各位的位置。波特率部分由时钟输入选择和分频、波特率发生器、调整器和波特率寄存器等组成。串行通信时，数据接收和发送的速率就由这些构件控制。

整个模块的始终源来自内部 3 个时钟或者外部输入时钟，由 UxTCTL 发送控制寄存器中位 SSEL1 和 SSEL0 选择，以决定最终进入模块的时钟信号 BRCLK 的频率。时钟信号 BRCLK 送入一个 15 位的分频器，通过一系列的硬件控制，最终输出两移位寄存器使用的移位时钟 BITCLK 信号。这个信号的产生，是分频器在起作用。当计数器减计数到 0 时，输出触发器翻转，送给 BITCLK 信号周期的一半就是定时器（分频计数器）的定时时间。

下面介绍波特率的设置于计算。采集每位数据的时候，在每位数据的中间都要进行 3 次采样，以多数表决的原则进行数据标定和移位接受操作，如此一次采集。由此看出，分频因子要么很大，要么是整数，否则，由于才几点的积累偏移，会导致每帧后面的几位数据采

样点不在其数据的有效范围内。

MSP430 的波特率发生器使用一个分频计数器和一个调整器，能够用低时钟频率实现高速通信，从而在系统低功耗的情况下实现高性能的串行通信。使用分频因子加调整的方法可以实现每一帧内的各位有不同的分频因子，从而保证每个数据的 3 个采样状态都是处于有效的范围内。

分频因子 N 由送到分频计数器的时钟 BRCLK 频率和所需的波特率来决定：

N=BRCLK/波特率

如果使用常用的波特率与常用晶体产生的 BRCLK，则一般得不到整数的 N，还有小数部分。分频计数器实现分频因子的整数部分，调制器使得小数部分尽可能准确。分频因子定义如下：

N=UBR + (M7+ M6 + …… + M0) / 8

其中：N 为目标分频因子；UBR 为 UxBR0 中的 16 位数据；Mx 为调整寄存器（UxMCTL）中的各数据位。

波特率可由下式计算：

波特率 = BRCLK/N=BRCLK/(N=UBR + (M7+ M6 + …… + M0) / 8)

波特率发生器可简化为由分频器与调整器组成。分频器完成输入时钟 BRCLK 的分频功能，调整器的数据按每一位计算，将对应位的数据（0/1）加到每一次分频计数器的分频值上。

关于调整器的工作原理，下面通过实际波特率的例子来说明。

例 3　BRTCLK=32768Hz，要产生 BITCLK=2400Hz，分频器的分频系数 32768/2400=13.65。所以分频器的计数值为 13.接下来用调整寄存器的值来设置小数部分的 0.65。调整器是一个 8 位寄存器，其中每一位对应 8 次分频情况，如果对应位为 0，则分频器按照设定分频系数计数；如果对应位为 1，则分频器按照设定的分频系数加 1 分频技术。按照这个原则，0.65*8=5，也就是说，8 次分频技术过程中应该有 5 次加 1 计数，3 次不加 1 计数。调整寄存器由 5 个 1 和 3 个 0 组成。调整器的数据每 8 次周而复始循环使用，最低位最先调整。最好的原则是 5 个 1 相对分散。比如设置为 6BH(01101011)。

四、异步方式的中断

USART 模块有接收和发送两个独立的中断源。使用两个独立的中断向量，一个用于接收中断事件，一个用于发送中断事件。USART 模块的中断控制位位于特殊功能寄存器中(special reg)MSP430F44Xxilie USART 异步方式中断控制位如表 2-15、2-16 所示。

表 2-15　MSP430F44X 系列 USART0 异步方式中断控制位

特殊功能寄存器	接收中断标志位	发送中断标志位
IFG1	接收中断标志 URXIFG0	发送中断标志 UTXIFG0
IE1	接收终端使能 URXIE0	发送中断使能 UTXIE0
ME1	接收允许 URXE0	发送允许 U TXE0

表 2-16　MSP430F44X 系列 USART1 异步方式中断控制位

特殊功能寄存器	接收中断标志位	发送中断标志位
IFG2	接收中断标志 URXIFG1	发送中断标志 UTXIFG1
IE2	接收终端使能 URXIE1	发送中断使能 UTXIE1
ME2	接收允许 URXE1	发送允许 U TXE1

MSP430 异步收发器是完全独立操作的，单使用同一个波特率发生器，则接收器和发送器使用相同的波特率。

注意：接收允许位 URXE 的置位或复位，将设置允许或禁止接收器从 URXD 数据线接收数据。当禁止接收时，如果已经开始一次接收操作，则会在完成后停止下一次接受操作；如果

无接收操作，则在进行中立即停止接受操作，连起始位的检测也被禁止。

发送允许位 UTXE 的置位或复位，将设置允许或禁止发送器将串行数据发送到 TXD 线路。

当 UTXE=0 时，已被激活的发送并不停止，而将在完成发送已经写入发送缓存器内的全部数据后被禁止。在 UTXE=0 之前写入发送缓存的数据有效。也就是说，UTXE 复位时，虽然发送缓存可写入数据，但数据不发送到串行线路，而一旦 UTXE 置位，缓存内的数据立即发送到线路。

2.7.3 异步通信寄存器

MSP430F449 中有两个通信硬件模块 USART0 和 USRAT1，因此有两套寄存器。USART0，USART1 的寄存器分别如表 2-17、2-18 所示。

表 2-17 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	071H	PUC 后 001H
接收控制寄存器	U0RCTL	读/写	072H	PUC 后 000H
波特率调整控制寄存器	U0MCTL	读/写	073H	不变
波特率控制寄存器 0	U0BR0	读/写	074H	不变
波特率控制寄存器 1	U0BR1	读/写	075H	不变
接收缓存器	U0RXBUF	读	076H	不变
发送缓存器	U0TXBUF	读/写	077H	不变
SFR 模块使能寄存器 1	ME1	读/写	004H	PUC 后 000H
SFR 中断时能寄存器 1	IE1	读/写	000H	PUC 后 000H
SFR 中断标志寄存器 1	IFG1	读/写	002H	PUC 后 082H

表 2-18 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	079H	PUC 后 001H
接收控制寄存器	U1RCTL	读/写	07AH	PUC 后 000H
波特率调整控制寄存器	U1MCTL	读/写	07BH	不变
波特率控制寄存器 0	U1BR0	读/写	07CH	不变
波特率控制寄存器 1	U1BR1	读/写	07DH	不变
接收缓存器	U1RXBUF	读	07EH	不变
发送缓存器	U1TXBUF	读/写	07FH	不变
SFR 模块使能寄存器 2	ME2	读/写	005H	PUC 后 000H
SFR 中断时能寄存器 2	IE2	读/写	001H	PUC 后 000H
SFR 中断标志寄存器 2	IFG2	读/写	003H	PUC 后 020H

下面依次介绍各个寄存器（x 代表 0 和 1）

a. UxCTL 控制寄存器

SUART 模块的基本操作由此寄存器的控制位决定。各位定义如表 2-19：

表 2-19 控制寄存器 UxCTL 的位定义

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　允许校验。

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 为停止位；

1　2 位停止位。

CHAR　字符长度。

0　7 位；

1　8 位。

LISTEN　反馈选择。选择是否将发送的数据由内部反馈给接收器。

0　无反馈；

1　有反馈。

SYNV　USART 模块的的模式选择。

0　UART 异步模式；

1　SPI 同步模式。

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。该位的状态影响着其他一些控制位和状态的状态。在串行口使用的过程中，这一位是比较重要的控制位。一次正确的 USART 模块初始化应该是这样的顺序：现在 SWRST=1 的情况下设置串口；然后设置 SWRST=0；最后如果需要中断，则设置相应的中断使能。

二、UxTCTL　发送控制寄存器

该寄存器的各位定义如表 2-20：

表 2-20 发送控制寄存器 UxTCTL 的位定义

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　时钟源选择位。

0　外部时钟 UCLKI；

1　辅助时钟 ACLK；

2、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。

三、URCTL　接收控制寄存器

各位定义如表 2-21：

表 2-21　接收控制寄存器 URCTL 的位定义

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

这部分在 2.9.2 中有介绍，因此不再说明。

四、UxBR0　波特率选择寄存器 0

五、UxBR1　波特率选择寄存器 1

UxBR0 和 UxBR1 两个寄存器分别用来存放波特率分频因子的整数部分。其中 UxBR0 为低字节，UxBR1 为高字节。两个字节合起来为一个 16 位字，成为 UBR。在异步通信时，UBR 不小于 3，否则会发生不可预测的错误。

六、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

如果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，则整数部分写入 UBR 寄存器，小数部分由调整控制寄存器 UxMCTL 的内容反映。波特率则由以下公式计算：

波特率 = BRCLK/N=BRCLK/(N=UBR + (M7+ M6 + …… + M0) / 8)

其中 UBR 为 UxBR0 中的 16 位数据。调整寄存器的 8 位分别对应 8 次分频，如果 Mx=1，则相应次的分频增加一个时钟周期，否则分频计数值不变。

七、URXBUF　接收数据缓存

接收缓存存放从接收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。如果传输 7 位数据，接收数据右对齐，最高位为 0。

八、UTXBUF　发送数据缓存

发送缓存内容可以发送至发送移位寄存器，然后由 UTXDx 传输。对发送缓存进行写操作可以复位 UTXIFG。如果传输 7 位数据，发送缓存最高位为 0。

2.7.4　异步操作应用举例

例 2-9 MSP430F449 在 ACLK=LFXT1=32768Hz,MCLK=SMCLK=1048576Hz 驱动下，以波特率 2400 异步串行通信，采用中断方式接收。

计算波特率分频因子：1048576/2400=436.9;UBR=0X01B4H,UxMCTL=0xFF;

程序如下：

```
#include<msp430x44x.h>
#pragma vector=UART0RX_VECTOR
__interrupt void usart0_rx(void)
{
        static  unsigned  char  value;
        while((IFG1 & UTXIFG0) == 0);        //判断接受缓存是否有数据
        value = RXBUF0;                      //读取数据
}
void main()
```

```
{
        WDTCTL = WDTPW + WDTHOLD;
        FLL_CTL0 |=XCAP14PF;
        U0CTL    |=SWRST;
        UTCTL0   |=SSEL1;                    //选择 MCLK 作为时钟
        U0BR0    =0XB4;                      //分频因子
        U0BR1    =0X01;
        U0MCTL   =0xFF;                      //调整器设置
        U0CTL    |=CHAR;                     //8 位数据
        U0CTL    &=~SWRST;
        ME1      |=URXE0;                    //使能接收
        IE1      |=URXIE0;                   //使能接收终端
        P2SEL    |=BIT5;                     //P2.5 用作接收
        _EINT();
        for(;;)
        {
                _BIS_SR(CPUOff);             //进入 LPM0 低功耗模式，等待接收
    中断唤醒
                _NOP();
        }
}
```

2.8 模数转换

MSP430 系列单片机很多系列都具有模数转换模块，转换精度主要为 10 位、12 位及 14 位。其中，MSP430X1XX2 内具有 ADC10 高速 10 位模数转换器；MSP430F13X、MSP430F14X、MSP430F43X 和 MSP430F44X 等系列器件中含有 ADC12 模块。MSP430X32X 系列中含有 ADC14 模块，它是 12+2 位精度模数转换器。

2.8.1 ADC12 结构

一、参考电压发生器

MSP430 ADC12 内置参考电源，而且参考电压有 6 种可编程选择，分别为 V_{R+}与 V_{R-}的组合。其中，V_{R+}有 AV_{CC}（模拟电源正端）、V_{REF+}（A/D 转换器内部参考电源的输出正端）及 V_{eREF+}（外部参考源的正输入端），V_{R-}包括 AV_{SS}（模拟电源负端）和 V_{eREF-}/V_{REF-}（A/D 转换器参考电源负端——内部或外部）。ADC12 可以通过设置各控制寄存器的值灵活地设置参考电压发生器的工作。

二、模拟多路器

当对多个模拟信号进行采样并 A/D 转换时，为了共用一个转换内核，需要模拟多路器，将多个模拟信号分时地接通。MSP430 中的 ADC12 配置有 8 路外通道与 4 路内部通道，通过 A0~A7 实现外部 8 路模拟信号输入，4 路内部通道可以将 V_{eREF+}、V_{eREF-}/V_{REF-}、$(AV_{CC}-AV_{SS})/2$ 以及片内温度传感器的输出作为待转换模拟输入信号。将片内温度传感器的输出送到 ADC12 的通道 10，可以测量芯片内的温度，可以通过测量值来设置警告信息，判断芯片的工作是否正常。

将 V_{eREF+}、V_{eREF-}/V_{REF-}、$(AV_{CC}-AV_{SS})/2$ 作为 ADC12 的输入信号，可以用于有关 ADC12 的自检、校验和诊断功能。

三、具有采样保持功能的 12 位转换器内核

ADC12 内核是一个 12 位的模数转换器，并能够将结果存放在转换存储器中。该内核使用两个可编程的参考电压（V_{R+}和 V_{R-}）定义转换的最大值和最小值。输入模拟电压的最终结果转换结果满足公式：

$$N_{ADC}=4095*(V_{IN}-V_{R-})/(V_{R+}-V_{R-})$$

ADC12 内核接收到模拟信号输入并具有转换允许的相关信号之后便开始进行 A/D 转换。

四、采样及转换所需的时序控制电路

在时序控制电路指挥下 ADC12 的各部件能够协调工作，这部分提供采样及转换所需要的各种时钟信号：ADC12CLK 转换时钟、SAMPCON 采样及转换信号、SHT 控制的采样周期等。详细情况见寄存器说明部分。

五、转换结果缓存

ADC12 共有 12 个转换通道，设置了 16 个转换存储器用于暂存转换结果，合理设置之后，ADC12 硬件会自动将转换结果存放到相应的 ADC12MEM 寄存器中。每个转换存储器 ADC12MEMx 都有自己对应的控制寄存器 ADC12MCTLx。

2.8.2 ADC12 寄存器

ADC12 有大量的控制寄存器供用户使用，用户根据实际需求通过软件独立配置 ADC12 的资源，从而灵活 ADC12 的各个功能模块。用户可以操作的寄存器如表 2-22 所示。

表 2-22 ADC12 的寄存器

寄存器类型	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~ADC12MEM15	存储寄存器 0~15

一、转换控制寄存器：ADC12CTL0、ADC12CTL1

① ADC12CTL0 转换控制寄存器 0

ADC12CTL0 与 ADC12CTL1 是 ADC12 的重要寄存器，控制了 ADC12 的大部分操作，其中的 4~15 位只有在 ENC＝0 时才可被修改。该寄存器的各位含义如下：

表 2-23 转换控制寄存器 0 的位定义

15~12	11~8	7	6	5	4	3	2	1	0
SHT1	SHT0	MSC	2.5V	REFON	ADC12ON	ADC12OVIE	ADC12TOIVE	ENC	ADC12SC

ADC12SC 采样/转换控制位。在不同条件下，ADC12SC 的含义如下：

表 2-24 ADC12CTL0 的采样/转换控制位

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

其中，ENC=1 表示转换允许（必须使用）；ISSH＝0 表示采样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。用软件启动一次 A/D 转换，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 A/D 转换

1　首次转换由 SAMPCON 上升沿启动

ADC12TOIVE 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 中原有数据还没有读出，而又有新的转换结果数据要写入时，则发生溢出。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

表 2-25　ADC12CTL0 的多次采样/转换位 MSC

有效条件	MSC	含义
SHP=1 CONSEQ≠0	0	每次转换需要 SHI 信号的上升沿触发采样定时器
	1	仅首次转换由 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后立即进行

SHT1、SHT0　采样保持定时器 1、采样保持定时器 0

分别定义了保存在转换结果寄存器 ADC12MEM8~ADC12MEME15 和 ADC12MEM0~ADC12MEM7 中的转换采样时序与采样 ADC12CLK 的关系。采样周期是 ADC12CLK 周期乘 4 的整数倍，即：

$$t_{sample} = 4 \times t_{ADC12CLK} \times n$$

表 2-26　ADC12CTL0 的采样保持定时器 0、1

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

② ADC12CTL1　转换控制寄存器 1

大多数位（3~15 位）只有在 ENC=0 时才可被修改。该寄存器的各位含义如下：

表 2-27　转换控制寄存器 1 的位定义

15~12	11~10	9	8	7~5	4、3	2、1	0
CSSTARTADD	SHS	SHP	ISSH	ADC12DIV	ADC12SEL	CONSEQ	ADC12BUSY

CSSTARTADD 转换存储器地址位。该 4 位所表示的二进制数 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。分频因子为该 3 位二进数加 1

ADC12SSEL ADC12 内核时钟源选择

0 ADC12 内部时钟源：ADC12OSC

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 ENC=0 之前，测试 ADC12BUSY 位以确定是否为 0。在其他转换模式下，该位无效。

二、存储及其控制寄存器： ADC12MCTL0~ADC12MCTL15

ADC12MEM0~ADC12MEM15

① ADC12MEM0~ADC12MEM15 转换存储寄存器

该组寄存器用来存放 A/D 转换结果。只用其中的低 12 位。

② ADC12MCTLx 转换存储器控制寄存器

每一个转换存储器有一个对应的转换存储器控制寄存器，所以在 CSSTARTADD 转换存储器地址位设置的同时，也确定了 ADC12MCTLx。

该寄存器各位含义如下，所有位只有在 ENC 为低电平时可被修改。

表 2-28 转换存储器控制寄存器的位定义

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 该序列中最后一次转换

SREF　参考电压源选择位

0　$V_{R+}=AV_{CC}$，$V_{R-}=AV_{SS}$

1　$V_{R+}=V_{REF+}$，$V_{R-}=AV_{SS}$

2、3　$V_{R+}=V_{eREF+}$，$V_{R-}=AV_{SS}$

4　$V_{R+}=AV_{CC}$，$V_{R-}=V_{REF-}/V_{eREF-}$

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

6、7　$V_{R+}=V_{eREF+}$，$V_{R-}=V_{REF-}/V_{eREF-}$

INCH　选择模拟输入通道。该 4 位所表示的二进制数为所选的模拟输入通道。

0~7　A0~A7

8　V_{eREF+}

9　V_{REF-}/V_{eREF-}

10　片内温度传感器的输出

11~15　$(AV_{CC}-AV_{SS})/2$

三、中断控制寄存器：ADC12IFG、ADC12IE、ADC12IV

采用中断处理方式，可以提高 MSP430 工作效率。

① 中断标志寄存器是一个 16 位字结构，其中中断标志位 ADC12IFG.x 对应于转换存储寄存器 ADC12MEMx。

ADC12IFG.x 置位：转换结束，并且转换结果已经装入转换存储寄存器。

ADC12IFG.x 复位：ADC12MEMx 被访问(ADC12IFG.x＝0)

② 中断允许寄存器与 ADC12IFG 寄存器相对应，也是一个 16 位寄存器。

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。

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。

③ ADC12IV　中断向量寄存器

ADC12IV 是一个多源中断：有 18 个中断标志（ADC12IFG.0~ADC12IFG.15 与 ADC12TOV,ADC12OV），但只有一个中断向量。所以需要设置这 18 个标志的优先级顺序，按照优先级来安排中断标志的响应，高优先级的请求可以中断正在服务的低优先级。

2.8.3　ADC12 转换模式

ADC12 提供 4 种转换模式：

(1)单通道单次转换

(2)序列通道单次转换

(3)单通道多次转换

(4)序列通道多次转换

一、单通道单次转换模式

在 ENC 复位并再次置位之前的输入信号将被忽略，转换模式可以在转换开始但结束之前切换，新的模式会在当前转换完成之后起作用。

程序举例：

例 2-10　设 LFXT1=32768Hz。设计 ADC12 单通道单次采样。

分析：使用 ADC12SC 位启动转换，ENC 位在每次转换之间固定

```
#include "msp430x44x.h"
void main(void)
{
    WDTCTL=WDTPW+WDTHOLD;
```

```
    P6SEL|=BIT0;  //使能 A/D 转换通道 A0
    ADC12CTL0=ADC12ON+SHT0_2;  //打开 ADC12 内核，选择采样定时器的值
    ADC12CTL1=SHP;  //采样信号选择，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
    ADC12CTL0|=ENC;  //允许转换
    while(1)
    {
        ADC12CTL0|=ADC12SC; //在 ENC=1,ISSH=0,SHP=1 的条件下，ADC12SC 由 0 变
                            //为 1 启动 A/D 转换，转换完成后 ADC12SC 自动复位
        while((ADC12IFG&BIT0)==0);  //等待转换结束并用将转换结果装入转换存
                                    //储寄存器
        _NOP();
    }
}
```

二、序列通道单次转换模式

程序举例：设 LFXT1=32768Hz。设计 ADC12 单次单通道采样。

例 2-11

```
#include "msp430x44x.h"
static unsigned int results[4];  //存储转换结果
void main(void)
{
        WDTCTL=WDTPW+WDTHOLD;
        P6SEL|=BIT0+BIT1+BIT2+BIT3;  //使能 A/D 转换通道 A0,A1,A2,A3
        ADC12CTL0=ADC12ON+MSC+SHT0_2;  //打开 ADC12 内核,选择采样定时器的值,
                                       //选择多次转换
        ADC12CTL1=SHP+CONSEQ_1;  //采样信号选择，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，转换
                                 //模式选择，序列通道单次转换模式
        ADC12MCTL0=INCH_0;
        ADC12MCTL1=INCH_1;
        ADC12MCTL2=INCH_2;
        ADC12MCTL3=INCH_3+EOS;  //通道 A3,最后序列
        ADC12IE=BIT3;  //使能中断 ADC12IFG.3
        ADC12CTL0|=ENC;  //使能转换
        _EINT();  //允许全局中断
        while(1)
        {
                ADC12CTL0|=ADC12SC;  //开始转换
                _BIS_SR(LPM0_bits);  //进入 LPM0
        }
    }
#pragma vector=ADC_VECTOR
__interrupt void ADC12ISR(void)
{
```

```
        results[0]=ADC12MEM0;  //存结果，清除 IFG
        results[1]=ADC12MEM1;
        results[2]=ADC12MEM2;
        results[3]=ADC12MEM3;
        _BIC_SR_IRQ(LPM0_bits);  //退出 LPM0
}
```

三、单通道多次转换模式

程序举例：设 LFXT1=32768Hz。设计 ADC12 单通道多次采样。

例 2-12

```
#include "msp430x44x.h"
static unsigned int results[8];  //存储转换结果
void main(void)
{
    WDTCTL=WDTPW+WDTHOLD;
    P6SEL|=BIT0;  //使能 A/D 转换通道 A0
    ADC12CTL0=ADC12ON+SHT0_8+MSC;  //打开 ADC12 内核，选择采样定时器的值，选
                                   //择多次采样
    ADC12CTL1=SHP+CONSEQ_2;  //采样信号选择，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转换模式选择，
                             //单通道多次转换
    ADC12IE=BIT0;  //使能中断 ADC12IFG.0
    ADC12CTL0|=ENC;  //使能转换
    _EINT();  //允许全局中断
    ADC12CTL0|=ADC12SC;  //开始转换
    _BIS_SR(LPM0_bits);  //进入 LPM0
}

#pragma vector=ADC_VECTOR
__interrupt void ADC12ISR(void)
{
    static unsigned int index=0;
    results[index]=ADC12MEM0;//进入中断，读取并存储转换结果
    index=(index+1)%8;
}
```

四、例 2-13

序列通道多次转换模式

程序举例：设 LFXT1=32768Hz。设计 ADC12 单通道多次采样。

```
#include "msp430x44x.h"
static unsigned int A0results[8];  //存储 A0 转换结果
static unsigned int A1results[8];  //存储 A0 转换结果
static unsigned int A2results[8];  //存储 A0 转换结果
static unsigned int A3results[8];  //存储 A0 转换结果
```

```
void main(void)
{
    WDTCTL=WDTPW+WDTHOLD;
    P6SEL|=BIT0+BIT1+BIT2+BIT3;  //使能 A/D 转换通道 A0,A1,A2,A3
    ADC12CTL0=ADC12ON+SHT0_8+MSC;  //打开 ADC12 内核，选择采样定时器的值，选
                                   //择多次采样
    ADC12CTL1=SHP+CONSEQ_3;  //采样信号选择，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转换模式选择，
                             //序列通道多次转换
    ADC12MCTL0=INCH_0;
    ADC12MCTL1=INCH_1;
    ADC12MCTL2=INCH_2;
    ADC12MCTL3=INCH_3+EOS;  //通道 A3,最后通道
    ADC12IE=BIT3;  //使能中断 ADC12IFG.3
    ADC12CTL0|=ENC;  //使能转换
    _EINT();  //允许全局中断
    ADC12CTL0|=ADC12SC;  //开始转换
    _BIS_SR(LPM0_bits);  //进入 LPM0
}

#pragma vector=ADC_VECTOR
__interrupt void ADC12ISR(void)
{
    static unsigned int index=0;
    A0results[index]=ADC12MEM0;  //进入中断，读取并存储通道 A0 转换结果
    A1results[index]=ADC12MEM1;  //进入中断，读取并存储通道 A1 转换结果
    A2results[index]=ADC12MEM2;  //进入中断，读取并存储通道 A2 转换结果
    A3results[index]=ADC12MEM3;  //进入中断，读取并存储通道 A3 转换结果
    index=(index+1)%8;
}
```

2.9 FLASH 存储器模块

嵌入式 FLASH 存储器获得的技术进步在从根本上促进了微控制器的应用。显著的改变了微控制器市场。FLASH 技术结合了 OTP 存储器的成不又是和 EEPROM 得可再编程性能，可以使用尽可能小的开销来发挥 EEPROM 的最大灵活性。MSP430 嵌入式 FLASH 存储器同 EEPROM 一样是电可擦出并且可编程存储器，它的主要特点如下：

a. 编程可以使用位、字节和字操作。
b. 可以通过 JTAG/BSL 和 ISP 进行编程。
c. 1.8~3.6V 工作电压，2.7~3.6 编程电压。
d. 擦除/编程此处可达 100000 次。
e. 数据保持时间从 10 年到 100 年不等。
f. 60KB 空间编程时间<5 秒。
g. 保密熔丝烧断后不可恢复，不能再对 JTAG 进行任何访问。

h. FLASH 编程/擦除时间由内部硬件控制，无需任何软件干预。

总之，FLASH 存储器具有如下优点：掉电后数据不丢失、数据存储速度快、电可擦除、容量大、在线可编程、足够多的擦鞋次数、价格低廉、高可靠性。FLASH 基本可以取代 EEPROM，只是擦除操作不能一个字节一个字节擦除，只能一段一段进行。

2.9.1 FLASH 存储器结构

FLASH 存储器模块是 MSP430FLASH 型器件中都有的一个模块，不同型号器件的 FLASH 容量不同，所在的地址空间也不一样，但都是由 n 段主存储器与 2 段信息存储器组成。信息存储器为每段 128 字节，分别为信息存储器 A 和 B，主存储器每段为 512 字节。所有型号器件的信息存储器地址完全相同，从 1000H~10FFH。主存储器的地址范围不一样，但其实地址一样，都是第 0 段源于地址 0FFFH，不同型号的器件最后一段的结束地址不同，

2.9.2 FLASH 存储器的寄存器及操作

1.FLASH 存储器的寄存器

允许编程、擦除等操作首先要对 3 个控制寄存器（FCTL1、FCTL2 及 FCTL3）D 的各位进行定义。它们使用安全键值（口令码）来防止错误的变成和擦除周期，口令出错将产生非屏蔽中断请求。安全键值位于每个控制字的高字节，读时为 96H，写时为 5AH。

（1）FVTL1 控制寄存器 1

FCTL1 用于控制所有写或者删除操作的有效位。各位的定义如下：

15~8	7	6	5	4	3	2	1	0
安全键值	BLKWRT	WRT	保留			MERAS	ERASE	

BLKWRT 段编程位。如果有较多的连续数据需要编程到某一段或某几段，则可选择这种方式，这样可缩短编程时间。在一段程序完毕，再变成其他段时，需要对该位先复位再置位，在下一条写指令执行前 WAIT 应为 1。

0 未选用段编程方式；

1 选用段编程方式。

WRT 编程位。

0 不编程，如果 FLASH 写操作，发生非法访问，使 ACCVIFG 置位；

1 编程。

MERAS 主存控制擦除位

0 不擦除；

1 主存全擦除，对主存写时启动擦除操作，完成后 MERAS 自动复位。

ERASE 擦除一段控制位。

0 不擦除；

1 擦除一段。由空写指令带入段号来制定擦除哪一段，操作完成后自动复位。

（2）FCTL2 控制寄存器 2

FCTL2 中各位主要对进入实虚发生器的时钟经进行定义。各位定义如下：

15~8	7	6	5	4	3	2	1	0
安全键值	SSEL1	SSEL0	FN5	FN4	FN3	FN2	FN1	FN0

SSEL1,SSEL0 选择时钟源

0　ACLK;
1　MCLK;
2　SMCLK;
3　SMCLK

FN5~FN0　分频系数选择位
0　直通;
1　2 分频;
2　3 分频;
……
63　64 分频。

（3）FCTL3　控制寄存器 3

FCTL3 寄存器用于 FLASH 存储器操作，保存相应的状态标志和错误条件。对于该控制寄存器的改变，没有什么条件限制。在 PUC 期间，它的控制位置位或者复位 WAIT。但是在 POR 期间 KEYV 复位。

FCTL3 寄存器主要是一些标志位，各位的定义如下：

15~8	7	6	5	4	3	2	1	0
安全键值			EMEX	LOCK	WAIT	ACCVIFG	KEYV	BUSY

EMEX　紧急退出位。对 FLASH 的操作失控时使用该位坐紧急处理。
0　无作用。
1　立即停止对 FLASH 的操作。

LOCK　锁定位，给已经编程哈德 FLASH 存储器加锁。该位可以由用户程序写入，也可以由硬件自动设置。可在编程、段擦除、主存擦除期间置位，置位后当前操作能正常结束。在段编程模式中，如果 BLKWRT=1 且 WAIT=1 时将 LOCK 位置位，则 BLKWRT 和 WAIT 都将复位，段编程模式结束。如果在段编程模式中发生非法访问，则 ACCVIFG 和 LOCK 位将置位。
0　不加锁，FLASH 存储器可读、可写、可擦除;
1　加锁，加锁的 FLASH 可读、不可写、不可擦除。

WAIT　等待指示信号位，该位只读。在段编程模式中指示 FLASH 存储器可以接受编程数据。如果 BLKWRT 位复位或 LOCK 位置位，则 WAIT 自动复位，段编程结束使得 WAIT 置位。在 BLKWRT=1 时，每一次成功的写入之后，BUSY 位被复位，指示其他的字或字节单元可以被编成操作。
0　段编程操作已经开始，编成操作进行中;
1　段编程操作有效，当前数据已经正确的写入 FLASH 存储器，后续编成数据被列入计划。

ACCVIFG　非法访问中断标志。当对 FLASH 阵列进行编程或擦除操作时，不能访问 FLASH，否则将使得该位置位。如果非法访问中断允许同时总的中断也允许，则将执行 NMI 中断程序。
0　没有对 FLASH 存储器的非法访问;
1　有对 FLASH 存储器的非法访问。

KEYV　安全键值（口令码）出错标志位。
0　对 3 个控制寄存器的访问，写入时高字节是 0A5H;
1　对 3 个控制寄存器的访问，写入时高字节不是 0A5H，同时引发 PUC 信号。KEYV 不会自动复位，须用软件复位。

BUSY　忙标志位。该位表示 FLASH 模式现在的状态，是否正在对其操作，现在忙与否。

该位只读，为用户提供一个判断标志位。每次编程或擦除之前都应该检查 BUSY 位。当编程或擦除启动以后，时序发生器将自动设置该位为 1，操作完成后，BUSY 位自动复位。

0　FLASH 存储器不忙；

1　FLASH 存储器忙。

2.FLASH 存储器的操作

由于 FLASH 存储器由很多先对独立的段组成，因此可在一段中运行程序，而对另一个段进行擦除或写入数据等操作。如果程序的运行和擦除或变成的段为同一段，则设置标志位 BUSY=1，而使得 CPU 挂起，直至编程周期结束，标志位 BUSY=0 为止。这是才能继续 CPU 的运行，执行下一条指令。正在执行编程或擦除等操作的 FLASH 段是不能被访问的，因为这时该段是片内地址数据总线暂时断开的。

对 FLASH 模块的操作可分为 3 类，擦除、写入及读出。而擦除又可分为单段擦除和整个模块擦除；写入可分为字写入、字节写入、子连续写入和字节连续写入，同时也可分为经过 JTAG 接口的访问与用户程序的访问。

FLASH 的具体操作过程如下：

(1) 对 FLASH 的擦除要做 4 件事：

a. 对 FLASH 控制寄存器写入适当的控制位；

b. 监视 BUSY 位；

c. 空写一次；

d. 等待。

擦除开始时，FLASH 模块需要产生适当的时序信号和正确的编程电压，然后由时序发生器控制整个擦除过程，擦除完毕，编程电压撤销。段擦除需要 4819 个时钟周期，整个 FLASH 擦除需要 5297 个时钟周期。

(2) FLASH 编程操作

FLASH 存储器主要用于保存用户程序或重要的数据、信息等一些掉电后不丢失的数据，只有通过对 FLASH 的编程操作，才能将这些数据写入 FLASH 存储器。由两种方式可以对 FLASH 编程：单个字或字节写入，多个字或字节顺序写入或块写入。

对 FLASH 编程按如下顺序进行：

a. 选择适当的时钟源以及合适的分频因子；

b. 如果 LOCK=1，将它复位。

c. 监视 BUSY 位，直到 BUSY=0 才可进入下一步；

d. 如果写入单字或单字节，则将设置 WRT=1；

e. 如果是块写或多姿、多字节顺序写入，则将设置 WRT=1，BLKWRT=1。

f. 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。

(3) FLASH 操作小结

对 FLASH 模块有 3 种操作：读、写及擦除。读很简单，可使用各种寻址方式，借助指令就可以轻松完成，如果用 C 语言编程，则可以使用指针对地址进行数据读取。擦除与写入需要按照其固定的操作过程，通过控制 FLASH 模块的三个寄存器中控制字的相应的位来完成，只有控制位的唯一组合才能实现相应的功能。表***在下面的示例中，有擦除和写入的历程，可以直接模块化进行调用。

表***

功能	BLKWRT	WRT	MERAS	ERASE	BUSY	WAIT	LOCK
字或字节写入	0	1	0	0	0	0	0
块写入	1	1	0	0	0	1	0

段擦除并写入 全部擦除并写入	0 0	0 0	0 1	1 1	0 0	0	0 0

另外，FLASH 模块在 POR 信号后，处于默认的度模式，不须对控制位作任何操作，就可以读出其中数据。

例 2-14 设计 FLASH 段 A、B 的擦除和写入

程序如下：

```
#include <msp430x44x.h>
char value;

void write_segA(char value);
void copy_A2B(void);
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;                   //关闭看门狗
    FCTL2 = FWKEY + FSSEL0 + FN0;              //定时时钟
    value = 0;
    while(1)
    {
        write_segA(value++);                    //写信息段 A
        copy_A2B();                             //信息段 A 内容复制到信息段 B

    }
}
void write_segA(char value)
{
    char  *flash_ptr;
    unsigned char i;
    flash_ptr = (char *)0x1080;                 //初始化 FLASH 指针
    FCTL1 = FWKEY + ERASE;                      //允许擦除
    FCTL3 = FWKEY;                              //解锁
    *flash_ptr = 0;                             //空写，启动擦除
    FCTL1 = FWKEY + WRT;                        //允许写
    for(i=0;i<128;i++)                          //循环写信息段 A 的 128 字节
    {
        *flash_ptr++=value;
    }
    FCTL1 = FWKEY;
    FCTL3 = FWKEY + LOCK;                       //锁定
}
void copy_A2B(void)
{
    char *flash_ptrA;
    char *flash_ptrB;
```

```
    unsigned int i;
    flash_ptrA = (char *)0x1080;
    flash_ptrB = (char *)0x1000;
    FCTL1 = FWKEY + ERASE;                          //允许擦除
    FCTL3 = FWKEY;                                  //解锁
    *flash_ptrB = 0;                                    //空写，启动擦除
    FCTL1 = FWKEY + WRT;                            //允许写
    for(i=0;i<128;i++)                              //循环将信息段A的128字节复
    制到信息段B
    {
        *flash_ptrB++ = *flash_ptrA++;
    }
    FCTL1 = FWKEY;
    FCTL3 = FWKEY + LOCK;                           //锁定

}
```

第三章：典型问题分析

这一部分是一些在使用MSP430的过程中可能会遇到的一些问题，和针对这些问题的一些解决方案。

1 关于430的时钟系统分析：

一般来说，430的使用时配备两个晶振，一个高速晶振8M，一个低速晶振32768Hz。这两个晶振与芯片内部的DCO一起成为芯片的三个时钟源。记住，这只是系统的时钟源。而系统运行的时候有三个时钟：MCLK,ACLK,SMCLK。用户需要通过软件配置来选择三个时钟的时钟源，具体的选择根据不同系列的单片而异。当系统的两个晶振没有起振或者没有连接的时候，内部DCO就会自动成为系统的时钟源，提供时钟。这就是为什么会发现程序能够运行，但是时钟却不是按照晶振输入的设置。这个时候就可能是晶振没有起振，DCO成为了时钟源。导致这个问题的原因有三个：

(1)程序设置错误，并没有选择外部晶振作为时钟源或者没有等待晶振起振；解决方法：检查程序中寄存器的设置，或者增加延时，对振荡器失效标志位进行判断；

(2)硬件电路错误，晶振的外接电容一般取值22pf，太大或者太小都有可能导致晶振不起振；解决方法：更换电容；

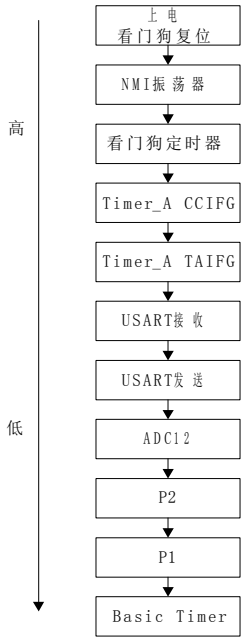
(3)晶振损坏，这是最后的判断，如果前两个问题可以排除的话，就可以断定是晶振坏了；解决方法：更换晶振。

2 看门狗：

有时候程序进行仿真时，发现全速运行时，结果错误，但是单步运行却又正确，这个时候可能需要检查下，主函数里是不是将看门狗定时器关闭了，很有可能是没有WDTCTL = WDTHOLD + WDTPW；语句，而导致看门狗一直复位。

3 按键：

(1)一般情况下，键盘设置在 P1 或者 P2 口，因为这两个口有独立的中断能力。有时候可能会发现按键时有时无，在排除硬件问题的情况下可以检查下程序中是否使用了比 P1 或者 P2 中断优先级更高的中断，比如定时器 A。中断优先级关系如下：



(2)程序有时候总往按键的中断里跳，这有可能是键盘的接法不太合理。在使用扫描法的时候，肯能会有将列线悬空的接法，这个时候的 P1 或者 P2 口比较容易受到干扰而产生中断。比较合理的接法是反转法的接法，将按键接电阻接地。

4 FLASH：

1）由于 MSP430 采用的是哈佛结构，即所有存储空间统一地址编码，因此，在使用 FLASH 的时候，最好只对 A,B 两段进行，因为其他地址会有程序或者变量的存储；

2）对 FLASH 进行写操作前，一定要先进行擦出，要注意的是，FLASH 的擦除只能一段一段的进行。

5 头文件：

在所建的项目中都会包含一个头文件<msp430x44x.h>(以 4 系列为例)，头文件中有大量的宏定义，熟练了解头文件，对 430 的学习，示例程序的阅读，以及提高程序的编写效率和可读性都有很大的帮助。

6 一种理解：

需要强调一点，对于 430 的 c 程序编程，对于模块的初始化，数据的读写，信号输入输出都是对寄存器的操作。

7 变量命名：

在编写 430 程序时，要注意变量的命名不能与寄存器同名，内部的寄存器如 N，C，Z，V 等，不能用相同的名字作为变量名。

8 I/O 口的复位：

430 的每个 IO 口在上电复位后都默认为输入口，在控制外部器件，如键盘、显示器时，要注意 IO 口的方向设置，比如在显示接口时，若把控制接口置为输入口，则会导致显示电流增大很多。在置为输出口时，可输入也可输出，在置为输入口时只能输入。

9 I/O 口的复位：

将 430 的端口设为输出口时，该口不能响应中断