TEXAS INSTRUMENTS

MSP430 系列混合信号微控制器 结构及模块

用户指南

目录

本书用途及表述约定

MSP430 用户指南以方便工程师及程序员使用的方式提供软件和硬件资料，以帮助开发应用 MSP430 系列的产品。

以下是表示信号和处理机状态的符号的简要说明：

- ADC　　A/D 转换器。
- CPUOff mode　保持 RAM 及 I/O 信号不变的低功耗模式。
  用辅助时钟(32768Hz 晶振)工作的模块处于活动状态。
- DCO　　数字控制振荡器。
- LCD　　液晶显示器。
- FF　　触发器。
- MAB　　存储器地址总线。位于各内部模块之间。可以是从 4 位至 16 位的任意宽度。它与 MS 信号一起定义了物理地址。
- MDB　　存储器数据总线。位于各内部模块之间。可以是 8 位或 16 位宽度。
- MS　　模块选择。为预解码地址空间。与 MAB 一起定义了物理地址。
- MSFR　　模块特殊寄存器。是特殊寄存器的预解码地址空间（0h 至 0Fh）。
- OSCOff mode　最低功耗模式。保持 RAM 及 I/O 信号不变。晶振停止。
- OTP　　单次可编程。
- POR　　上电复位。
- PUC　　上电清除，“1”设置处理机启动条件。
- SAR　　逐位逼近寄存器。
- SCI　　处理同步及异步协议的串行通信接口。
- SCG　　系统时钟发生器。
- SFR　　特殊功能寄存器。
- SPI　　串行外围接口（广泛应用的同步串行通信协议）。
- TBD　　待定义。
- TOS　　堆栈顶。
- UART　　通用异步收发（最广泛应用的串行通信协议）。
- USART　　通用同步异步收发。
- WD，WDT　　看门狗，看门狗定时器。

寄存器位类型约定

- rw：　　读/写。
- r：　　只读。
- r0：　　读出为“0”。
- w：　　只写。
- (w)：　　无寄存功能，写“1”将产生一个脉冲。读出总是为“0”。
- -0，-1：　　发生 PUC 后的状态。
- -(0)，-(1)：　发生 POR 后的状态。
- h0：　　由硬件复位。

符号

运算符

@	寄存器间接寻址
&	绝对寻址
→	数据传递方向
+	加
-	减
x	乘
/	除
.AND.	逻辑与
.OR.	逻辑或
.XOR.	逻辑异或
.NOT.	逻辑非

寄存器符号

R0 或 PC	寄存器 0 或程序计数器
R1 或 SP	寄存器 1 或堆栈指针
R2 或 SR/CG1	寄存器 2 或状态寄存器/常数发生器 1
R3 或 CG2	寄存器 3 或常数发生器 2
R4 至 R15	通用工作寄存器

状态寄存器内容

C	进位或借位标志位
Z	零标志位
N	负数标志位
CPUOff	CPU 关闭位
OSCOff	系统振荡器关闭位
GIE	总控中断允许位
SCG0	系统时钟发生器控制位 0
SCG1	系统时钟发生器控制位 1
V	溢出标志位

其它

＝	等于
≠	不等
＞，＜，≥，≤	比较符
″ ″	包含 ASCII 字符
h	16 进制数
b	2 进制数
#	立即数
E	指数
&	绝对寻址模式指示

汇编程序伪指令

.equ	等价命令
.sect	区域指示
.word	word 数据
.byte	byte 数据
;	注解指示

1. MSP430 系列

本章讨论 MSP430 系列微控制器的特性及对模拟信号处理控制的特殊能力。系列的全部成员均为软件兼容，软件通过公共的软件库、设计技术及开发工具，可以方便地在系列中的各型号间移植。

CPU 设计成适合各种应用的 16 位结构。它采用“冯-纽曼结构”，因此 RAM、ROM 和全部外围模块都在同一个地址空间内。

目录 页号

1.1 特性与功能

- 多达 64K 字节寻址空间，包含 ROM、RAM、闪存 RAM 和外围模块，将来计划扩大至 1M。
- 通过堆栈处理，中断和子程序调用层次无限制。
- 仅 3 种指令格式，全部为正交结构。
- 尽可能做到 1 字/指令。
- 源操作数有 7 种寻址模式。
- 目的操作数有 4 种寻址模式。
- 外部中断引脚：I/O 引脚具有中断能力。
- 中断优先级：同时发生的中断按优先级别处理。
- 嵌套中断结构：中断程序可以被更高优先级的中断请求打断。
- 外围模块的存储器分配：全部寄存器不占用 RAM 空间，均在模块内。
- 片上 USART：发送与接收有各自的中断。
- 定时器中断可作事件计数、时序发生、PWM、等等。
- 看门狗功能。
- A/D 转换器(10 位或更高精度) 有 8 个输入端，可作为恒流源工作。
- 具有 EPROM 型及 OTP 型。
- 具有 LCD 驱动电路。
- 用 FLL 和 32768Hz 晶振获得稳定的处理机时钟频率。
- 正交指令简化了程序开发：所有指令可以用所有寻址模式。
- 已开发了 C-编译器。
- 模块设计思想：所有模块采用存储器分配。

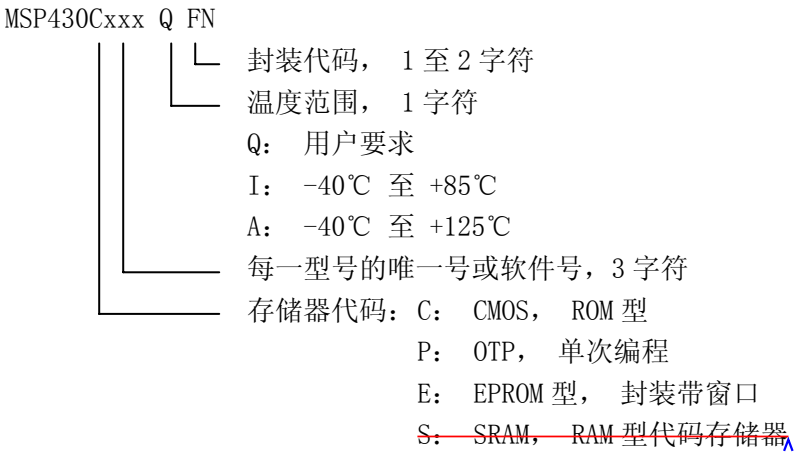
1.2 系统关键特性

- 超低电流消耗： CPUOff and OSCOff 模式。
- 可在电压降至 2.5 V 情况下工作。
- 系统内置模块：LCD 驱动，A/D 转换，I/O 端口，UART，看门狗，定时器，EPROM，等等。
- 只有微计算机模式; 无微处理机模式。
- 方便使用：强大而方便的指令组加速软件的开发。
- 软件可在 RAM 中运行。程序可经 UART 或测试引脚装入 RAM，并能在实时条件下运行。这样可降低试验和调整的开销。
- 在 64K 字节公共空间中有可能实现任意的 ROM/RAM 混合分配。
- 具有高级语言编程能力
  大寄存器组（12 个通用寄存器）
  面向堆栈
  大 ROM 和 RAM 空间
  正交指令组
  利用寻址模式实现查表处理
- 有实现 16 与 10 进制转换的专门指令 DADD
- ROM 读取、RAM 存取、数据处理、I/O 及其它外围操作都使用公共的指令：无特殊指令。
- CPU 的能力远超出智能化传感系统的要求。它的实时处理能力及各种外围模块使得可应用在其它低功耗领域。如有线远程通信的 DTM 收发器。

1.3 MSP430 系列的各型号

MSP430 系列的各型芯片的命名和特性归纳如下：

- 命名：



- 开发工具包括软件仿真器 DT430,汇编器及连接器 ASM430/LNK430,C-编译器 CS430/CW430，以及硬件在线仿真器 ICE430。所有开发工具都工作在 PC 环境下，并且符合 Windows 的 SAA 标准。

对 PC 的最低要求为：

IBM 兼容
DOS 5.0 或更高版本
Windows 3.1， 3.11 or ’95
PC 有 486 或更高性能处理机
8 MB 存储器
3.5″软盘驱动器
硬盘有 5 MB 可用空间

	MSP430x310	MSP430x320	MSP430x330
最大内部时钟频率	1.1 MHz @3V 3.3 MHz @5V	1.1 MHz @3V 2.2 MHz @5V	1.1 MHz @3V 2.2 MHz @5V
晶振频率	32.768 kHz		32.768 kHz
工作温度	-40℃至+85℃	-40℃至+85℃	-40℃至+85℃
程序存储器 MSP430Cxxx： MSP430Pxxx： MSP430Exxx： 存储器扩展	 4/8/12k ROM 8K OTP 8K EPROM 无	 8k ROM 16K OTP 16K EPROM 无	 24k ROM 32K OTP 32K EPROM 无
内部 RAM	256/512	256/512	1024
数据 EEPROM	无	无	无
模块 硬件乘法器 P0，8 位，支持中断 P1，8 位，支持中断 P2，8 位，支持中断 P3 P4 看门狗定时器 BasicTimer1/实时钟 8-bit 定时器/计数器 定时器/端口，1x8 位 Timer_A，16 位 同步通信 异步通信 LCD 驱动 ADC/恒流源 DAC	 无 有 有 有 有 有 无 无 定时器/端口+软件 23x4 段 见定时器/端口 无	 无 有 有 有 有 有 无 无 定时器/端口+软件 21x4 段 有 无	 有 有 有 有 有 有 有 有 有 有 有 SPI 模式 UART 模式或 定时器/端口+软件 30x4 段 见定时器/端口 无
I/O 引脚 输入引脚 输出引脚	9 1 27	9 7 25	40 1 34
中断/复位 外部中断 向量数	 11 16	 11 16	 1+24 16
封装类型	56 QFP	64 SSOP	100 QFP

表 1.1：MSP430 系列特性汇总

2．结构概述

本章说明 MSP430 系统的基本功能。

目录 页号

MSP430 系列产品包含以下主要功能：

- CPU
- 程序存储器(ROM 或 EPROM)
- 数据存储器(RAM 或 EEPROM)
- 运行控制
- 外围模块
- 振荡器和倍频器

MSP430 系列采用存储器--存储器结构，即用一个公共的空间对全部功能模块寻址，同时用精简指令组操作所有功能模块。

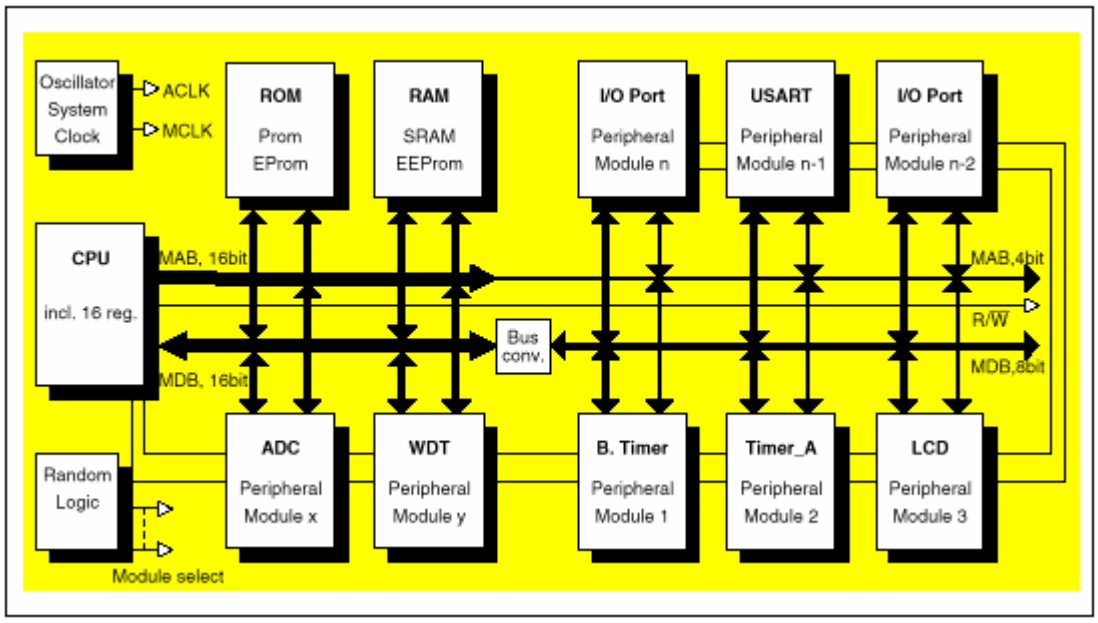


图 2.1：MSP430 系统结构

2.1 CPU

CPU 运行正交设计、对模块高度透明的精简指令集。它由一个 16 位 ALU、16 个寄存器和一套指令控制逻辑组成。其中 4 个寄存器有特殊用途，即程序计数器 PC、堆栈指针 SP、状态寄存器 SR 和常数发生器 CG2。除了 R3/CG2 和 R2/CG1，所有寄存器都可作为通用寄存器用所有指令操作。常数发生器是用于在指令执行时提供常数而不是存储数据。对 CG1 访问的寻址模式可以区分常数的数据。PC、SR 和 SP 配合精简指令组所实现的控制使应用开发可实现复杂的寻址模式和软件算法。

2.2 代码存储器

对代码存储器的访问总是以字形式取得代码，而对数据可以用字或字节方式访问。每次访问需要 16 条数据线和访问当前存储器模块所需的地址线。存储器模块由模块允许信号自

动选中，这是一项减少总电流消耗的技术。程序存储器可以是 EPROM 或 ROM。MSP430 系列目前的产品支持 OTP 型和掩膜编程型。支持外部扩展存储器是将来性能增强的目标。最低的 64K 字节空间的顶部 16 个字，即 0FFFFh 到 0FFE0h，保留存放复位和中断的向量。

程序对程序存储器可以任意读取，但不能写入。

未来的改进：

寻址空间将改进为分段方式。扩展的寻址空间主要增加 3 种方式：长跳转及调用指令、代码段指针 CSP 和数据指针 DPP。代码段指针在状态寄存器 SR 中。扩展的地址空间以如下方式用于代码访问指令（CSP+PC）和数据存储器访问指令（[DPPi]+操作数地址）中：

MAB = CSP * 10000h + PC　　访问代码存储器

MAB = DDPi * 4000h + Rs/d　　访问堆栈和数据存储器

在 64K 空间内寻址，CSP 和 DPP 的数值在存储器地址总线上不起作用。

2.3 数据存储器(RAM)

数据存储器经两条总线与 CPU 相连：存储器地址总线（MAB）和存储器数据总线（MDB）。数据存储器可以以字或字节宽度集成在片内。

所有指令可以对字节或字操作。但是对堆栈和 PC 的操作是按字宽度的，寻址时必须对准偶地址。

2.4 运行控制

MSP430 系列微控制器的运行主要受控于存储在特殊寄存器（SFR）中的信息。不同的 SFR 中的位可以允许中断，支撑取决于中断标志状态的软件，以及定义外围模块的工作模式。被禁止的外围模块将停止它的功能以减少电流的消耗。而所有存储在模块寄存器中的数据被保留。外围模块的工作模式可以用特定的区域来识别。

2.5 外围模块

外围模块经 MAB、MDB 和中断服务及请求线与 CPU 相连。对大多数外围模块，MAB 通常是 5 位。MDB 是 8 位或 16 位。8 位数据总线的模块经总线转换电路连到 16 位的 CPU。这些模块的数据交换毫无例外地要用字节指令处理。而对 16 位模块的操作指令就没有任何限制。大部分外围模块是工作在字节方式下的。SFR 的处理也毫无例外为 8 位。对 8 位外围模块的操作要依据顺序说明。

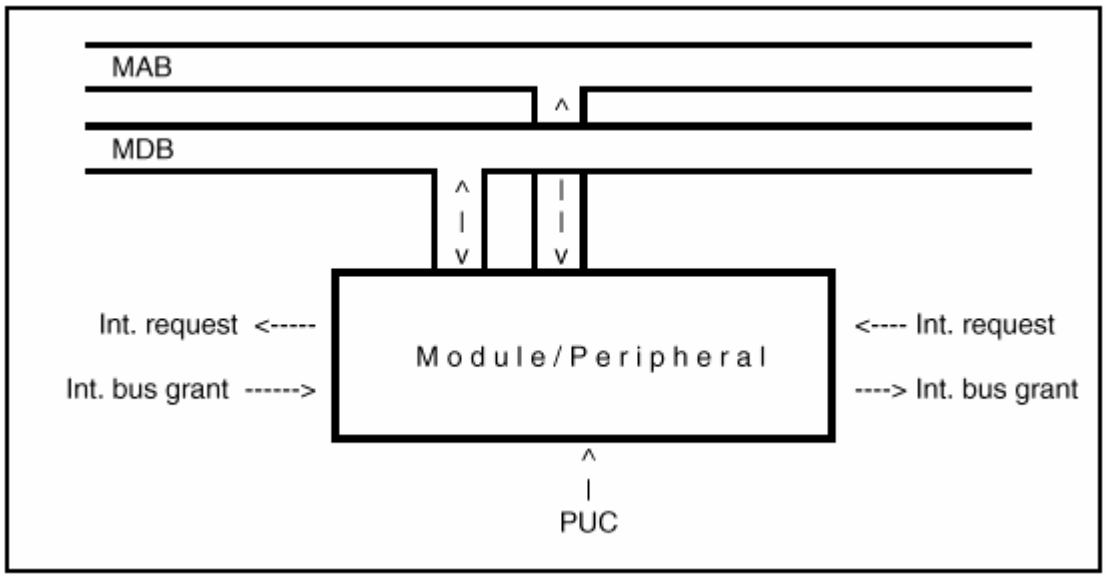


图 2.2：外围模块的连接总线

2.6 振荡器、倍频和时钟发生器

振荡器是专门为通用的时钟低功耗 32768Hz 晶振设计的。除了晶体外接外，所有的模拟元件都集成在片内。

这一振荡器对于一些以低频工作的模块是直接信号源。对于 CPU 和其它模块，晶振频率用一个锁频环电路（FLL）倍频。FLL 在上电后以最低频率开始工作，并通过控制一个数控振荡器（DCO）来调整到适当的频率。

长时间的频率偏离受到晶体和振荡器的稳定性的限制。

供处理机工作的时钟发生器的频率固定在晶振的倍频上。并提供时钟信号 MCLK。

3．系统复位、中断和运行模式

目录 页号

3.1 系统复位和初始化

MSP430 可以有 4 种复位来源：在 VCC 端加上供电电源，在 RST*/NMI 端输入低电平信号，可编程看门狗定时器超时和在对 WDTCTL 寄存器写入时密钥不符。

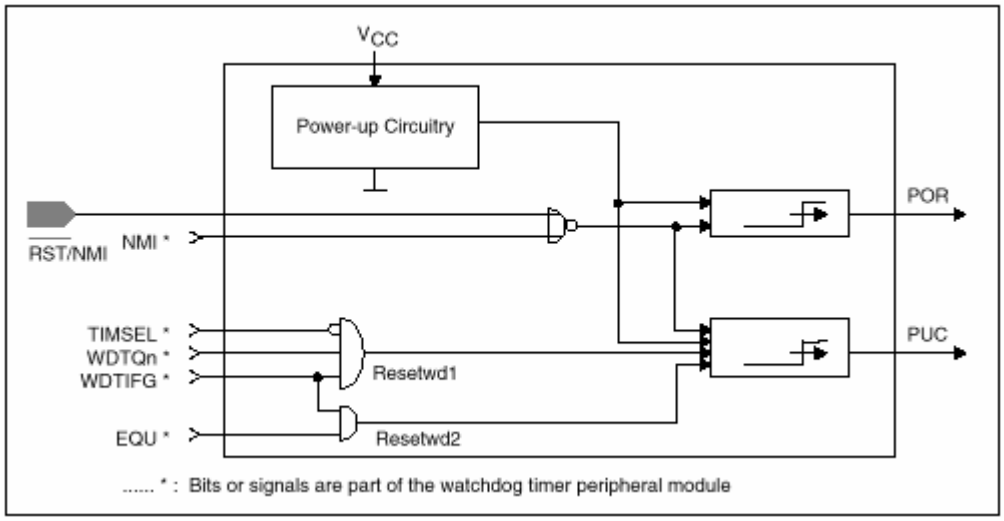


图 3.1：系统复位功能

发生复位后，程序查询各复位源的标志。程序能确定复位源，以执行适当的复位操作。MSP430 在发生 VCC 上电后开始硬件初始化：

- 全部 I/O 引脚切换成输入状态。
- I/O 标志被复位，见各外围模块说明。
- 在复位向量地址 0FFFEh 中含的地址加载入 PC。
  CPU 从这个上电清除（PUC）向量中含的地址开始运行。
- 状态寄存器（SR）清除。
- 用户程序必须对除 PC 与 SR 外的全部寄存器作初始化（如 SP、RAM 等）。
- 决定工作频率的系统时钟从 DCO 的最低频率开始工作。启动晶振时钟后，频率调整到目标值。

RST*/NMI 引脚在加载 VCC 后形成复位功能。引脚的复位功能一直保持到不选此功能为止。处于复位功能状态下，在 RST*/NMI 引脚上拉低至 GND，然后释放，MSP430 按以下顺序开始工作：

- 在复位向量地址 0FFFEh 中含的地址加载入 PC。
- 在释放 RST*/NMI 引脚后，CPU 从复位向量中含的地址开始运行。
- 状态寄存器 SR 复位。
- 除 PC 与 SR 外，用户程序对全部寄存器作初始化（如 SP、RAM 等）。
- 对外围模块中的寄存器作处理。
- 决定工作频率的系统时钟从 DCO 的最低频率开始工作。启动晶振时钟后，频率调整到目标值。

3.2 中断系统结构

有 3 类中断：

- 系统复位
- 非屏蔽中断
- 可屏蔽中断

引起系统复位的中断源有：

- 加电源电压 @POR，PUC
- RST*/NMI 引脚加低电平（选择复位模式） @POR，PUC
- 看门狗定时器溢出（选择看门狗模式） @PUC
- 看门狗定时器密钥不符（写 WDTCTL 是口令错） @PUC

非屏蔽中断由以下情况产生：

- RST*/NMI 引脚有上升沿（选择 NMI 模式）。
- 振荡器故障。

注意： 振荡器故障
振荡器故障可由允许位 OFIE 屏蔽。在通用中断允许复位后被禁止。

可屏蔽中断源有：

- 看门狗定时器溢出（选择定时器模式）。
- 其它模块的中断。

MSP430 中断优先级系统各模块的中断优先级由模块连接链决定：越接近 CPU/NMIRS 的模块中断优先级越高

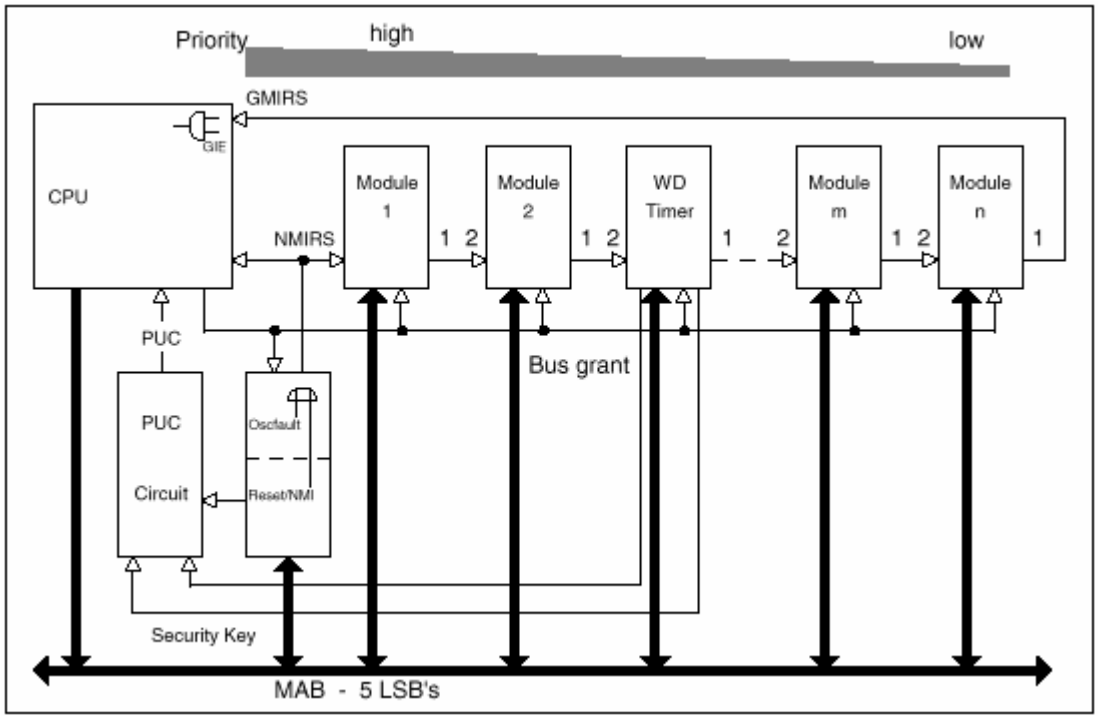


图 3.2：中断优先级结构

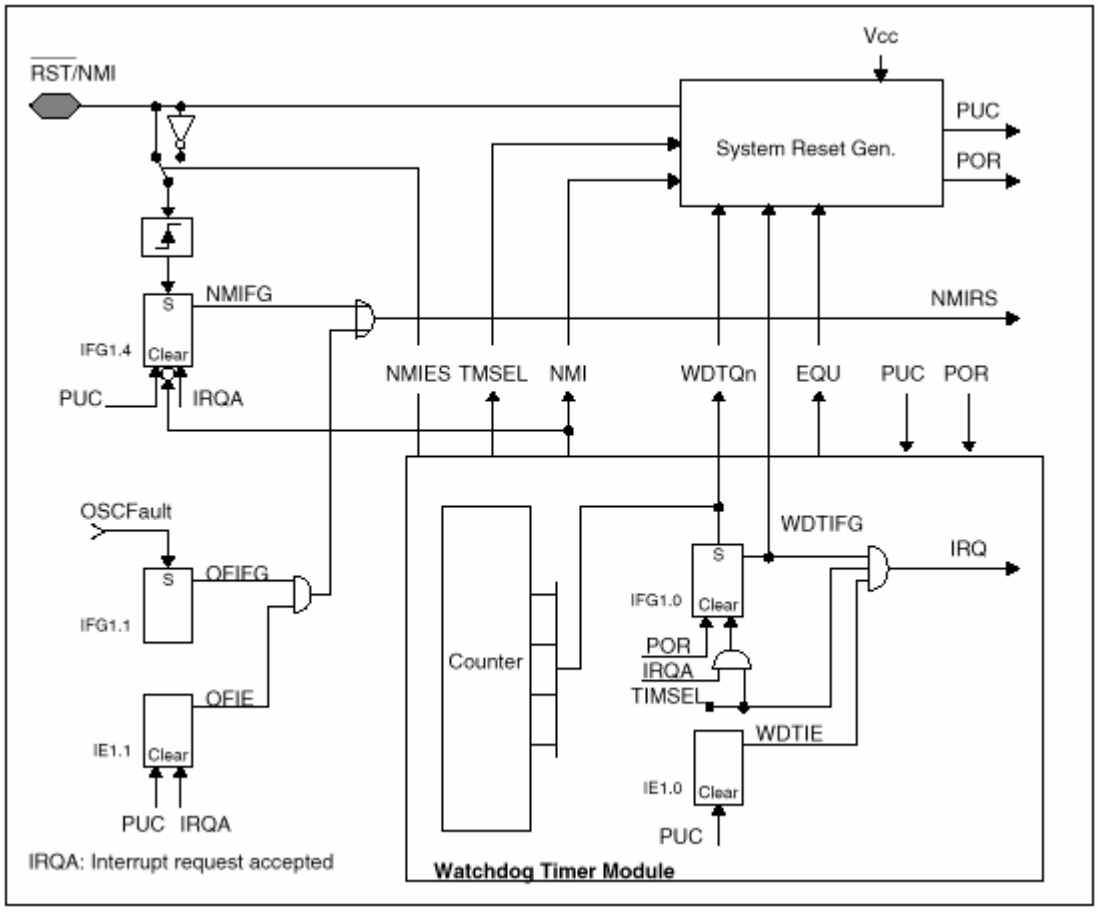


图 3.3：复位/非屏蔽中断模式选择

因为作用在同一个引脚上，复位和 NMI 功能只能交替工作。有关的控制位位于看门狗定时器控制寄存器（WDTCTL）内，也有口令保护。

WDTCTL（0120h）

7							0
HOLD	NMIES	NMI	TMSEL	CNTCL	SSEL	IS1	IS0
rw-0	rw-0	rw-0	rw-0	(w)-0	rw-0	rw-0	rw-0

位 5：　NMI 位选择 RST*/NMI 引脚的功能。PUC 后复位。

NMI=0：　RST*/NMI 引脚按复位输入端工作。

RST*/NMI 引脚保持低电平时，内部 PUC 信号有效（电平敏感）。

NMI=1： RST*/NMI 引脚作为边沿敏感的非屏蔽中断输入端。

位 6： 选择 RST*/NMI 引脚上触发 NMI 功能的有效边沿。

NMIES=0： 上升沿触发非屏蔽中断。

NMIES=1： 下降沿触发非屏蔽中断。

中断操作 - 复位/NMI

如选择复位功能，当 RST*/NMI 引脚保持“低”时 CPU 一直保持复位状态。当引脚电平变“高”，CPU 开始从 0FFFEh(复位向量)所含的地址执行程序。

如选择 NMI 功能，一个按 NMIES 位规定的触发沿会产生无条件中断，程序从 0FFFCh 所含的地址重新开始执行，并且 SFR 中的 RST*/NMI 标志(IFG1.4)也被置位。它在中断请求服务程序执行时自动复位。RST*/NMI 引脚不可永久地保持“低”状态。当发生激活 PUC 的状态，WDTCTL 寄存器中的复位会强制在 RST*/NMI 引脚上产生复位功能。在 RST*/NMI 引脚上的持续“低”电平引起复位和系统暂停。

注意： NMI 触发沿选择

当已选择了 NMI 模式并改变 NMI 沿选择位，根据 RST*/NMI 引脚上的实际电平可能会发生 NMI。在 NMI 模式选择前改变 NMI 沿选择位就不会发生 NMI。

中断操作 -- 振荡器故障控制

如振荡器部分所述，FLL 振荡器甚至在晶振失效时也能持续工作，但是它将运行在可能的最低频率上。第二个极限是可能的最高频率。这两种情况通常都是出错条件，必须能被 CPU 检测到。因此 SFR 中的 IE1.1 位可允许振荡器故障信号产生一个 NMI 请求。通过测试 SFR 中的中断标志 IFG1.1，CPU 能确定中断是否由振荡器故障引起。

中断操作 -- 上电清除(PUC)

3 种信号源或事件会产生系统复位：

- 上电逻辑电路信号
- RST*/NMI 输入
- 看门狗定时器溢出

因 RST*/NMI 和看门狗引发的复位可以由软件通过测试 SFR 中的中断标志位 IFG1.0 来确定。

3.3 中断处理

MSP430 的可编程中断结构可以组成灵活的片上和外部中断体系，以适应实时中断驱动系统的需要。中断可由处理机的运行状态来启动，如看门狗溢出、外围模块或外部发生的事件等。每个中断源可以用中断允许位单独关闭，而状态寄存器中的通用中断允许位 GIE 可以禁止全部中断。

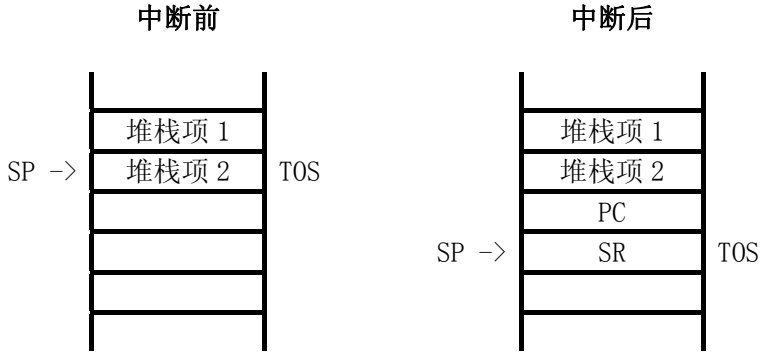
当中断请求发生并且相应的中断允许位和通用中断允许位置位，中断服务程序按以下顺序激活：

- CPU 处于活动状态： 完成当前执行指令。

CPU 处于省电状态：　　终止低功耗模式。

- 指向下一条指令的 PC 值压入堆栈。
- SR 压入堆栈。
- 如在执行上条指令时已有多个中断请求发生并且等待服务，选择最高优先级者。
- 在单一中断源标志中的中断请求标志位自动复位，多中断源标志仍保持置位以等待软件服务。
- 通用中断允许位 GIE 复位；
  CPUOff 位、OscOff 位和 SCG1 位 *）复位；
  状态位 V、N、Z 和 C 复位。
- 相应的中断向量值装入 PC，程序从该地址继续执行中断处理。

*）SCG0 不改变，FLL 环路控制保持原有工作状态。



中断响应从接受中断请求开始到执行相应的中断服务程序的首条指令，持续 6 个周期。

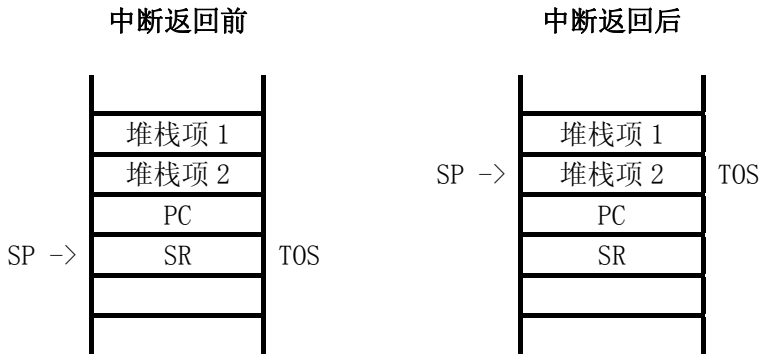
中断处理程序结束的指令是：

RETI

它执行以下内容：

- SR 从堆栈中推出。
  被中断的程序回到与中断前完全相同的状态，包括 OscOff、CPUOff 和 GIE 位。
  SR 中的 GIE 位在中断服务期间的值被取代，它总是“1”，因为为了接受中断请求它预先已置位。
- PC 从堆栈推出。

以 RETI 指令从中断服务程序返回，需 5 个周期。



如果 GIE 位在中断处理程序内置位，允许中断请求嵌套。

含有 GIE 位的状态寄存器 SR/R2 如下：

15 … 9	8	7							0
留作将来增强用	V	SCG1	SCG0	OscOff	CPUOff	GIE	N	Z	C
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

图 3.4：状态寄存器 SR

除了 GIE 位，中断源可以被控制位单独或成组的允许/禁止。中断允许标志集中位于两个地址的 SFR 中。受中断请求对程序流的控制可以方便地通过充分运用中断允许和屏蔽来调整。硬件只对被允许的中断源中的最高优先级服务。

3.3.1 SFR 中的中断控制位

大多数中断控制位、中断标志和中断允许位集中在少数几个 SFR 中。这些 SFR 以字节形式位于低地址区。SFR 只能以字节指令访问。

地址	7 … 0
000Fh	未定义或未实现
000Eh	未定义或未实现
000Dh	未定义或未实现
000Ch	未定义或未实现
000Bh	未定义或未实现
000Ah	未定义或未实现
0009h	未定义或未实现
0008h	未定义或未实现
0007h	未定义或未实现
0006h	未定义或未实现
0005h	模块允许 2； ME2.x
0004h	模块允许 1； ME1.x
0003h	中断标志 2; IFG2.x
0002h	中断标志 1; IFG1.x
0001h	中断允许 2； IE2.x
0000h	中断允许 1； IE1.x

MSP430 各型号支持各个模块内的 SFR。除了 NMI，各模块中断源可以单独允许以实现中断功能及操作。配置位的完全软件控制使得应用软件在中断允许屏蔽时重新激活系统。

中断允许 1、2

位	缩写	初始状态*	说明
IE1.0	WDTIE	复位	看门狗定时器允许，选中 watchdog 模式时无效
IE1.1	OFIE	复位	振荡器故障中断允许
IE1.2	P0IE.0	复位	针对 I/O P0.0
IE1.3	P0IE.1	复位	针对 I/O P0.1 或 8 位定时器/计数器

IE1.4		复位	保留，未定义
IE1.5		复位	保留，未定义
IE1.6		复位	保留，未定义
IE1.7		复位	保留，未定义
IE2.0	URXIE	复位	USART 接收中断允许
IE2.1	UTXRIE	复位	USART 发送中断允许
IE2.2	ADIE/TPIE	复位	ADC 或定时器/端口中断允许（适合 320 型）
IE2.3	TPIE	复位	定时器/端口(适合 310、330 型)
IE2.4		复位	保留，未定义
IE2.5		复位	保留，未定义
IE2.6		复位	保留，未定义
IE2.7	BTIE	复位	Basic Timer 中断允许

*初始状态是指发生 PUC 后的状态. 对于 WDTIFG 见有关说明

中断标志寄存器 1、2

位	缩写	初始状态	说明
IFG1.0	WDTIFG	不变	溢出或密钥不符时置位
		复位	VCC 上电或 RST*/NMI 引脚有复位条件
IFG1.1	OFIFG	置位	振荡器发生故障时置位
IFG1.2	P0IFG.0	复位	针对 I/O P0.0
IFG1.3	P0IFG.1	复位	针对 I/O P0.1 或 8-bit 定时器/计数器
IFG1.4	NMIIFG	复位	RST*/NMI 引脚信号
IFG1.5			保留，未定义
IFG1.6			保留，未定义
IFG1.7			保留，未定义
IFG2.0	URXIFG		USART 接收标志
IFG2.1	UTXIFG		USART 发送就绪
IFG2.2	ADIFG	复位	ADC 转换结束时置位
IFG2.3			保留，未定义
IFG2.4			保留，未定义
IFG2.5			保留，未定义
IFG2.6			保留，未定义
IFG2.7	BTIFG	不变	Basic Timer 标志

模块允许 1、2

位	缩写	初始状态	说明
ME1.0			保留，未定义
ME1.1			保留，未定义
ME1.2			保留，未定义
ME1.3			保留，未定义
ME1.4			保留，未定义
ME1.5			保留，未定义
ME1.6			保留，未定义
ME1.7			保留，未定义
ME2.0	URXE		USART 接收允许
ME2.1	UTXE		USART 发送允许
ME2.2			保留，未定义
ME2.3			保留，未定义

ME2.4	保留，未定义
ME2.5	保留，未定义
ME2.6	保留，未定义
ME2.7	保留，未定义

中断向量地址

中断向量和上电起始地址位于 ROM 中的 0FFFFh - 0FFE0h。向量包含各中断处理程序的 16 位入口地址。中断向量的优先级按递降排列。

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	WDTIFG	复位	0FFFEh	15，最高
NMI 振荡器故障	NMIIFG 0FIFG*	非屏蔽 可屏蔽	0FFFCh	14
I/0 专用	P0IFG.0	可屏蔽	0FFFAh	13
I/0 专用	P0IFG.1	可屏蔽	0FFF8h	12
		可屏蔽	0FFF6h	11
看门狗定时器	WDTIFG	可屏蔽	0FFF4h	10
Timer_A	CCIFG0	可屏蔽	0FFF2h	9
Timer_A	TAIFG**	可屏蔽	0FFF0h	8
USART 接收	URXIFG	可屏蔽	0FFEEh	7
USART 发送	UTXIFG	可屏蔽	0FFECh	6
ADC，定时器/端口 2）	ADCIFG	可屏蔽	0FFEAh	5
定时器/端口 1）		可屏蔽	0FFE8h	4
P2	P2IFG.07*、**	可屏蔽	0FFE6h	3
P1	P1IFH.07*、**	可屏蔽	0FFE4h	2
Basic Timer	BTIFG	可屏蔽	0FFE2h	1
P0	P0IFG.27*、**	可屏蔽	0FFE0h	0，最低

*） 多源标志

**）暂定

1） 定时器/端口在 320 型中的向量

2） 定时器/端口在 310、330 型中的向量

表 3.1：中断源、标志和向量

3.3.2 外部中断

端口 P0、P1 和 P2 的全部 8 位都可实现外部事件的中断处理。每一个 I/O 位都可独立编程。

由于可能组合任意输入、输出和中断条件，因此能灵活适应不同 I/O 结构。

注意：外部中断信号的最小脉冲宽度
所有外部中断信号必须具有至少 1.5MCLK 的脉冲宽度以保证稳定的中断响应,但是更窄的脉冲信号仍可能产生中断请求。

端口 P0

P0 模块安排了 3 个向量。P0.0 的信号、P0.1 的信号和其余端口 P0.2 至 P0.7 的信号是这 3 个向量的源。通过中断事件可将向量地址装入 PC。

P0 有 6 个控制 I/O 引脚的寄存器：

- 输入寄存器
- 输出寄存器
- 方向寄存器
- 中断标志　　有 6 个标志，含有用作中断输入的 I/O 引脚的信息
  0：无等待响应的中断
  1：因引脚信号跳变产生等待响应的中断
  写入“0”将中断标志复位。写入“1”将中断标志置位。当发生中断事件时即以同样的方式来处理。
- 中断沿选择　寄存器对每一 I/O 引脚有一中断触发跳变选择位。
  0：发生低/高跳变时中断标志置位
  1：发生高/低跳变时中断标志置位
- 中断允许　　寄存器对引脚 P0.2 至 P0.7 有 6 位分别允许中断事件触发中断请求。
  0：禁止中断请求
  1：允许中断请求

I/O 引脚 P0.2 至 P0.7 中断处理： 编程举例

```
; I/O 引脚 P0.2 至 P0.7 处理中断
;
IOINTR      PUSH    R5                  ; 保存 R5
            MOV.B   &P0IFG, R5          ; 读中断标志
            BIC.B   R5, &P0IFG          ; 用读入数据清除状态标志
                                        ; 其它各位不清除!
            EINT                        ; 允许中断嵌套
;
; R5 包含引起中断的 I/O 引脚的信息:
; 处理由此开始。
;
```

```
            .........
            .........
            POP     R5                  ; 处理完成： 恢复 R5
            RETI                        ; 从中断返回
            .........
            .........
; 中断向量表定义
            .sect   "IO27_vec", 0FFE0h
            .WORD   IOINTR              ; ROM 中 I/O 引脚(2 至 7)向量
            ;
            .sect   "RST_vec", 0FFFEh   ; 中断向量
            .WORD   RESET
```

端口 P1， P2

P1 与 P2 完全相同。对 P1 和 P2 模块分配一个单独向量。引脚 P1.0 至 P1.7 和 P2.0 至 P2.7 可用作中断源。向量包含因中断事件引发装入 PC 的存储器地址。

P1 和 P2 分别有 7 个寄存器用于控制 I/O 引脚：

- 输入寄存器
- 输出寄存器
- 方向寄存器
- 中断标志　　有 8 个标志，含有用作中断输入的 I/O 引脚的信息
  0：无等待响应的中断
  1：因引脚信号跳变产生等待响应的中断
  写入“0”将中断标志复位。写入“1”将中断标志置位。当发生中断事件时即以同样的方式来处理。
- 中断沿选择　寄存器对每一 I/O 引脚有一中断触发跳变选择位。
  0：发生低/高跳变时中断标志置位
  1：发生高/低跳变时中断标志置位
- 中断允许　　寄存器对引脚 P0.2 至 P0.7 有 6 位分别允许中断事件触发中断请求。
  0：禁止中断请求
  1：允许中断请求
- 功能选择寄存器

注意：数字端口 P0、P1 和 P2 的中断处理

只有跳变（不是静态电平）才引起中断。 中断程序必须对多源中断标志复位。所谓多源中断标志是指 P0IFG.2 至 P0IFG.7，P1IFG.0 至 P1IFG.7 和 P2IFG.0 至 P2IFG.7。单源标志 P0IFG.0 和 P0IFG.1 在得到服务时被复位。当 RETI 执行后仍有中断标志置位（因引脚信号跳变发生在中断服务期间），在 RETI 指令执行完成后会再次发生中断。这保证了软件能发现每一次跳变。

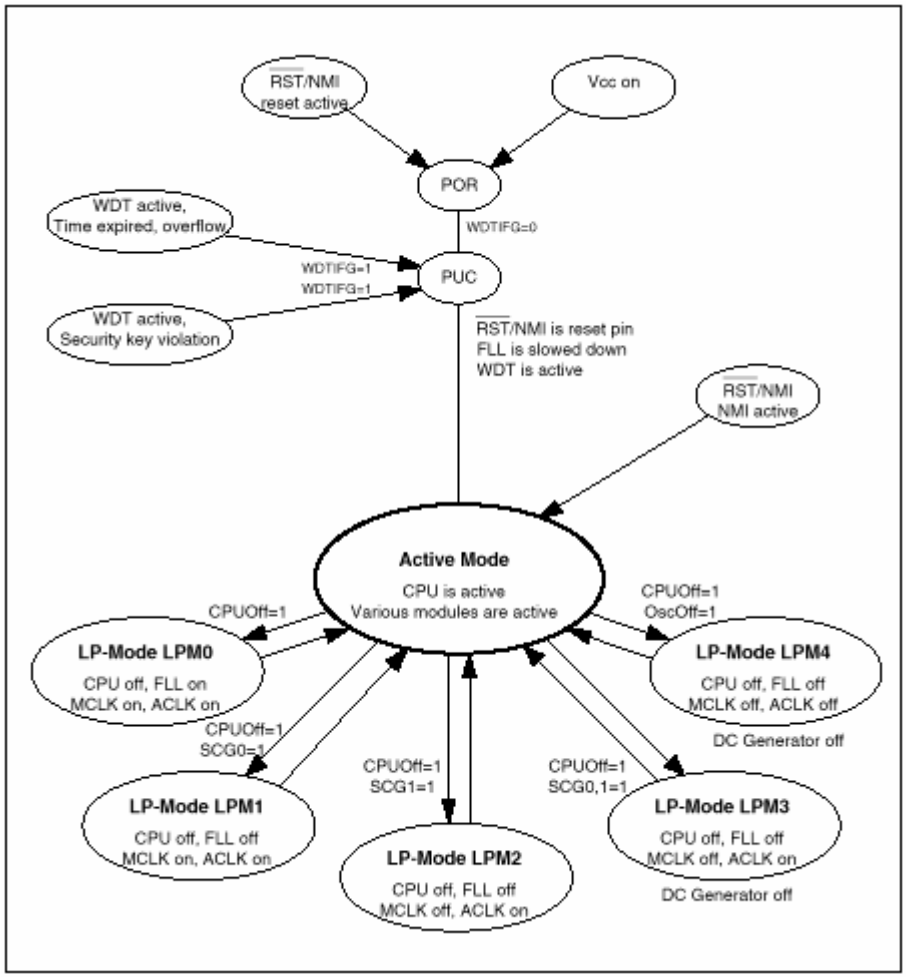
3.4 运行模式

MSP430 的运行模式以先进的方式支持超低功率和超低能耗的各种要求。这是通过各模块的智能化运行管理和 CPU 的状态组合而成。一个中断事件将系统从各种运行模式中唤醒，而 RETI 指令又使运行返回到中断事件发生前的运行模式。

MSP430 系列为超低功耗应用开发出采用不同功耗水平的运行模式。

用 CMOS 技术设计的超低功耗系统有三个主要目的：

- 解决运行速度和数据流量与低功耗设计的冲突
- 将各模块的电流消耗降至最低
- 限制活动状态至最低要求。



有 5 种运行模式，可由软件组合：

• 活动模式，AM，

各种活动的外围模块的组合。

• 低功耗模式 0，LPM0，

CPUOff 置位，CPU 停止活动，

CPUOff 不会使外围模块停止运行，

ACLK 和 MCLK 信号活动，MCLK 的锁频环控制活动。

@ SCG1=0， SCG0=0， OSCOff=0， CPUOff=1

• 低功耗模式 1，LPM1，

CPUOff 置位，CPU 停止活动，

CPUOff 不会使外围模块停止运行，

MCLK 的锁频环控制停止，

ACLK 和 MCLK 信号活动。

@ SCG1=0， SCG0=1， OSCOff=0， CPUOff=1

• 低功耗模式 2，LPM2，

CPUOff 置位，CPU 停止活动，

CPUOff 不会使外围模块停止运行，

MCLK 的锁频环控制停止，

ACLK 信号活动。

@ SCG1=1， SCG0=0， OSCOff=0， CPUOff=1

• 低功耗模式 3，LPM3，

CPUOff 置位，CPU 停止活动，

CPUOff 不会使外围模块停止运行，

MCLK 的锁频环控制和 MCLK 信号停止，

DCO 的 DC 发生器（到 MCLK 发生器）关闭。

ACLK 信号活动。

@ SCG1=1， SCG0=1， OSCOff=0， CPUOff=1

• 低功耗模式 4，LPM4，

CPUOff 置位，CPU 停止活动，

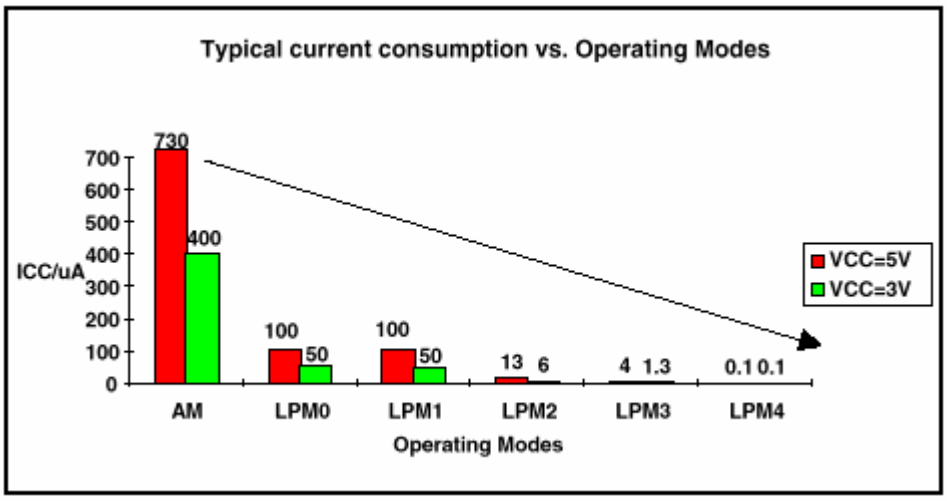
CPUOff 不会使外围模块停止运行，

MCLK 的锁频环控制停止，

DCO 的 DC 发生器（到 MCLK 发生器）关闭。

ACLK 信号停止，晶振停止。

@ SCG1=X， SCG0=X， OSCOff=1， CPUOff=1



来源：TI Data sheet SLASE07， January 1996 (MSP430C312/314)

对于各外围模块和 CPU 的活动状态通过适当的低功耗模式、选择停止外围模块的部分运行、或者将它们整个停止等方法来控制。用根据应用需求设计的软件有多种途径来组成最低电流消耗的系统。在 SFR 中含有能允许和禁止外围模块运行功能的控制位。一个例子是 LCD 外围模块中的模拟电压发生器的允许/禁止：它通过一个寄存器位来打开或关闭。最常用的影响电流消耗并支持从低功耗模式快速起动的控制位在状态寄存器 SR 中。有 4 位用于控制 CPU 和系统时钟发生器。

对活动模式 AM 的间歇运行和限制全速运行模式的时间周期，这 4 个控制位非常有用，即 CPUOff、OscOff、SCG0 和 SCG1。运行模式控制位包含在 SR 中的最大优点是在中断服务期间它作为运行状态保护在堆栈中。只要 SR 信息不改变，在 RETI 指令执行后处理机能继续保持在中断事件发生前的运行模式。通过处理保存在堆栈中的数据或改变 SP 也可以选择另一个程序流向。方便地访问堆栈和 SP 的指令组使得设计者可以分别地优化程序结构。

3.5 低功耗模式

SFR 中的各模块允许位确定各自功耗控制器工作状态的配置。由用户程序定义外围模块的活动或停止。被禁止模块的电流降低到被禁止各部分的漏电流。模块中唯一活动的是完成受控使模块进入允许状态或传递中断请求（例如发生外部硬件中断）给 CPU 的部分。

各模块允许的选择，有多达 5 种可能的省电模式：CPU 关闭模式（LPM0）和 4 种系统时钟发生器的运行组合。当控制位 CPUOff、SCG1、SCG0、OscOff（位于 SR 中）被置位就会进入上述状态。系统时钟发生器模块因 SCG1、SCG0 和 OscOff 位的状态而从 4 种省电模式中重新激活，将在系统时钟发生器部分详细说明。

进入中断程序

如果一个被允许的中断唤醒 MSP430，就会进入中断程序开始处理：

- SR 和 PC 保存入堆栈，保存了中断事件发生时的现场。
- 随后 SR 中的运行模式控制位 OscOff、SCG1 和 CPUOff 自动被复位。

从中断程序返回

有两种从中断服务程序返回的方法：

- 置位低功耗模式位后返回

从中断返回后，PC 指向下一条指令。由于被恢复后的低功耗模式禁止了 CPU 的活动，PC 所指的指令不执行。

- 复位低功耗模式位后返回

从中断返回后，程序从对 SR 中的 OscOff 或 CPUOff 置位的指令之后的地址继续执行。

3.5.1 低功耗模式 0、1，LPM0 和 LPM1

对 SR 中的 CPUOff 置位可选择进入低功耗模式 0 或 1。置位后 CPU 立即停止运行，系统内核的常规操作停止。CPU 的操作暂停直至有任一中断请求或复位发生。所有内部总线停止活动。系统时钟发生器的继续工作和时钟信号 MCLK 及 ACLK 的活动取决于 SR 中的其它 3 位，即 SCG0、SCG1 和 OscOff。SCG1 定义 MCLK 运行于 ACLK 的倍数或按 DCO 最近一次的控制信号

运行。

被允许并得到 MCLK 或 ACLK 信号的外围模块处于活动状态。I/O 端口的全部引脚和 RAM 及寄存器保持不变。所有被允许的中断事件可以从此状态唤醒程序。

```
; === Main program flow with switch to CPUOff Mode ========================
;
            BIS     #18h, SR     ; 进入 LPM0 并允许中断控制位 GIE.
                                 ; PC 在执行这一指令时增加，指向程序的后续步骤
            ..........           ; 如中断服务期间 CPUOff 复位，程序继续
; === Interrupt service routine ============================================
            ..........
            ..........
            RETI                 ; RETI 恢复中断前的 CPU 状态
                                 ; SR 中的 GIE、CPUOff、OscOff、SGC1 和 SCG0 在执
                                 ; 行从中断返回时恢复
```

3.5.2 低功耗模式 2、3， LPM2 和 LPM3

对 SR 中的 CPUOff 和 SCG1 置位可选择进入低功耗模式 2 或 3。置位后 CPU 和 MCLK 立即停止运行。它们暂停直至有任一中断请求或复位发生。所有内部总线停止活动。SCG1 定义 MCLK 在系统回到活动模式时运行于 ACLK 的倍数或按 DCO 最近一次的控制信号运行。

被允许并得到 ACLK 信号的外围模块处于活动状态。工作时需要 MCLK 信号的外围模块因为 MCLK 信号停止活动而停止。I/O 端口的全部引脚和 RAM 及寄存器保持不变。所有被允许的、不依赖于 MCLK 的中断事件可以从此状态唤醒。

3.5.3 低功耗模式 4， LPM4

全部活动部件停止，只有 RAM、端口和寄存器的内容保持。只能由被允许的外部中断唤醒。

在起动 LPM4 前，软件要考虑在这一低功耗模式期间系统需要的条件。最重要的两点是针对运行环境的的，即对 DCO 和周期性操作的影响。运行环境定义的频率合成数值应保持或校正。校正在周围环境需要系统对频率作大的改变时可能发生。当存在周期性操作应用时，应该考虑锁频环可能失控，余留的时间片不足以将锁频环保持在校正操作范围之内。

以下例子说明低功耗模式 4 的进入（OscOff）：

```
        BIS #B8h, SR     ; 进入 LPM4，并允许 GIE。
                         ; CPU 切换到 LPM 模式。DCO 操作被允许。
                         ; 在中断程序期间 LPM4 将结束时，DCO 运行已准备好。
        ..........       ; 如在中断程序中 OscOff 复位，程序继续，
        ..........       ; 否则仍保持在 OscOff 模式。
```

3.6 低功耗应用要点

当电流消耗是系统应用的重要指标时，应该考虑一些常规原则：

- 将不用的FETI输入端连接到V_{SS}

- 关闭 LCD 及模块，可能时包括外部的模拟电压发生器
- JTAG端口TMS、TCK和TDI不要连接到V_{SS}
- CMOS 输入端不能有浮空的节点：将所有输入端接适当的电平
- 选择尽可能低的运行频率 - 既针对内核，同样也针对各外围模块
- 如用了 LCD，选择尽可能弱的驱动能力，或者将它关闭
- 充分利用中断驱动软件的特性 – 程序能快速地起动执行。

4. 存储器组织

目录 页号

MSP430 系列的存储器空间采用“冯-纽曼结构”，代码存储器（ROM、EPROM、RAM）和数据存储器（RAM、EEPROM、ROM）由同一组地址及数据总线安排在一个地址空间中。

全部在物理上分离的存储区域，如内部的 ROM、RAM、SFR 和外围模块及外接的存储器，都位于这一公共的地址空间。总的寻址空间在小存储模式时为 64KB，大存储模式时为 1MB。小存储模式采用线性寻址空间。大存储模式时，代码访问的地址空间为 16 个 64KB 段，数据访问的地址空间为 16 个 64KB 页。

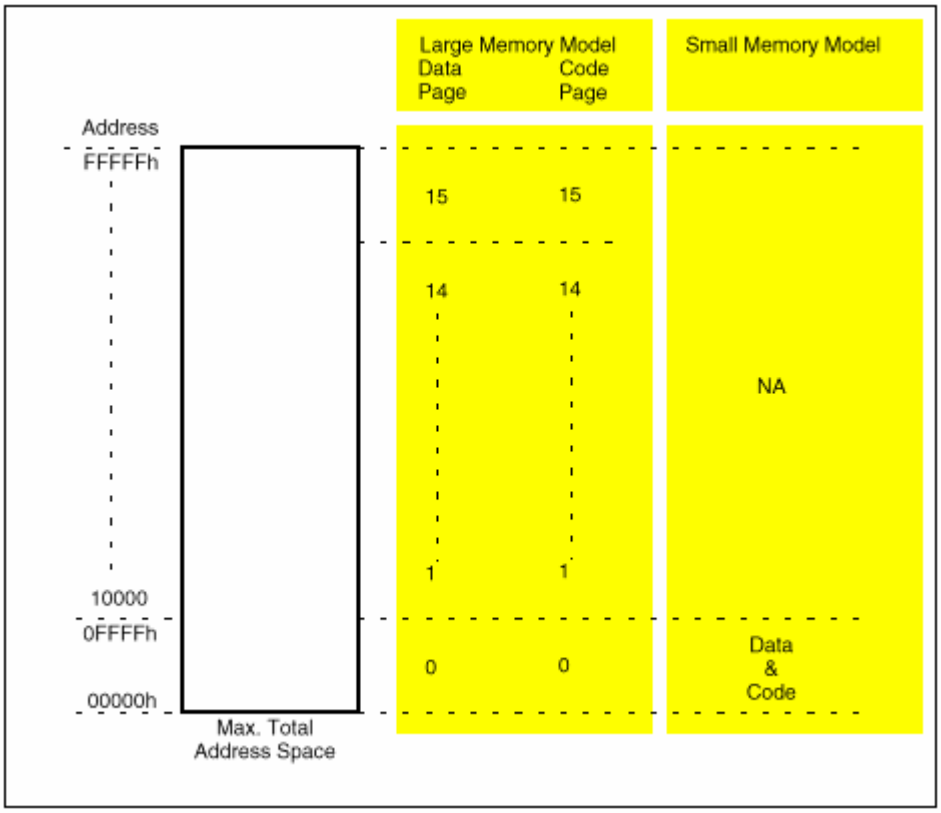


图 4.1：总的存储器地址空间

存储器构成为 64KB 或更少时采用小存储模式，安排在最低的 64KB，这时不必考虑寻址时的代码段和数据页。

小存储模式的空间结构和数据宽度如下：

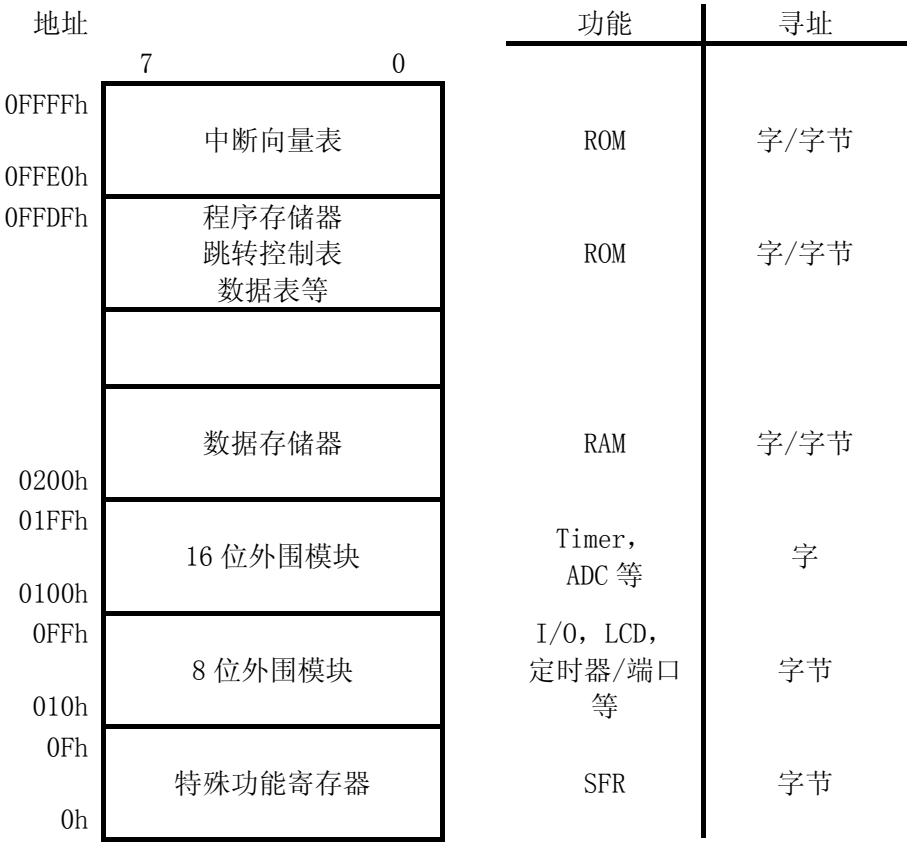
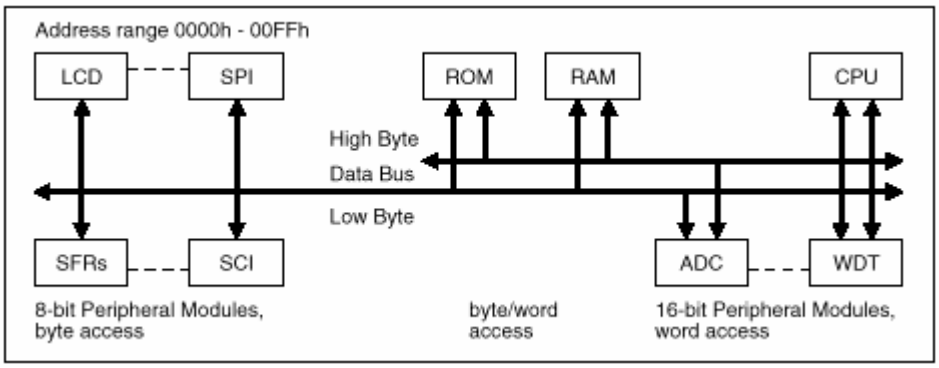


图 4.2：基本存储空间的存储器分配

数据总线可以是 16 位或 8 位宽度。可以以字访问的模块，数据宽度始终是 16 位。而其它的 8 位模块只能用字节指令访问。程序存储器（ROM）和数据存储器（RAM）既可用字节也可用字指令访问。部分外围模块以 16 位或 8 位实现，访问它们时必须用适当的字指令或字节指令。

许多外围模块与 CPU 连接通过 8 位存储器数据总线（MDB）、存储器地址总线（MAB）的低 5 位及 2 个模块允许信号、2 条中断控制/请求线和上电信号。访问这些模块总是用字节指令。其它 16 位外围模块的连接通过 16 位 MDB，完全支持字处理，访问必须用字指令。



4.1 存储器中的数据

字节数据可以定位在偶地址或奇地址。字数据定位在偶地址：低字节在偶地址，高字节在下一个奇地址。

图 4.3：字节结构存储器中的位、字节和字

4.2 片内 ROM 组织

片内 ROM 可能有各种体积直到 64KB。公共的地址空间是由 SFR、外围模块寄存器、数据和代码存储器共享的。而 SFR 和外围模块安排在 0 到 01FFh 的空间中。余下的地址空间从 0200h 到 0FFFFh 由数据和代码存储器分享。

不同体积 ROM 的开始地址都在同一个地址 0FFFEh。从最高优先级开始的中断向量表位于这一最高的 ROM 字地址。PC 计数，也即指令执行顺序，是按另一方向的。就是由低地址到高地址。根据执行指令的寻址模式 PC 将增加 2、4 或 6，但不包括程序流向控制指令，即跳转、分支跳转和子程序调用。

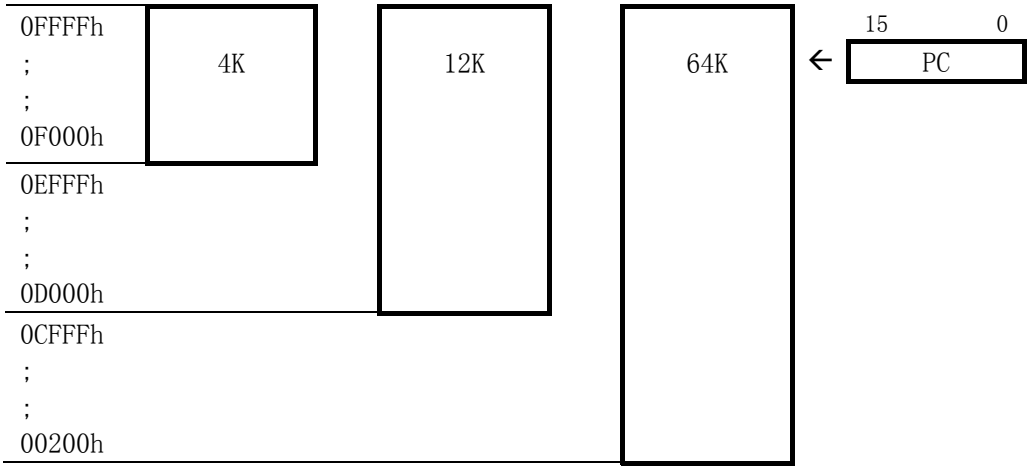


图 4.4：ROM 组织

中断向量和上电向量在 ROM 中，从地址 0FFFEh 开始。向量中含有相应的中断处理指令序列的 16 位地址。

4.2.1 ROM 表的处理

MSP430 的结构允许 ROM 存放大的数表。访问这些表可以用所有的字和字节指令。这一点对编程的灵活性和节省 ROM 空间带来各种好处：

- 在 ROM 中存放用于显示字符转换的输出可编程逻辑阵列（Output-PLA）
- 可根据需要设计 OPLA 的项数 (项数无限制)
- OTP 型自动包含 OPLA 编程能力
- 计算表访问（如棒图显示）
- 表处理支持的程序流向控制。

表处理是十分重要的特性,它能导致非常快速清晰的编程风格。特别是对于传感器应用，为了线性化和补偿将传感器数据存入表中作表处理是一个优点。

4.2.2 计算分支跳转和子程序调用

用标准指令实现计算分支跳转和子程序调用是可能的。CALL 和 BR 指令与其它指令采用同样的寻址模式。（见编程举例）

计算分支跳转和子程序调用最理想的寻址模式是间接-间接。充分利用这一编程特点可以得到与常规 8 位和 16 位微控制器不同的程序结构。利用软件的状态处理可以方便地处理大量的子程序调用，而不是利用“标志”来控制程序流向。

计算分支跳转和子程序调用在一个 64KB 代码段内有效。

4.3 RAM 和外围模块组织

利用适当的指令后缀，整个 RAM 可按字节或字访问。外围模块安排在两个地址空间：

- SFR 硬件是面向字节的，安排在地址 0h 至 0Fh。
- 硬件面向字节的外围模块安排在地址 010h 至 0FFh。
- 硬件面向字的外围模块安排在地址 100h 至 01FFh。

4.3.1 RAM

RAM 可以用作代码存储器和数据存储器。而代码访问总是对偶地址的。

指令助记符的后缀定义了访问的数据是字还是字节。

举例：

```
ADD.B   &TCDATA，TCSUM_L                        ;字节访问
ADDC.B  TCSUM_H                                 ;字节访问
ADD     R5，SUM_A    ≡   ADD.W   R5，SUM_A      ;字访问
ADDC    SUM_B        ≡   ADDC.W  SUM_A          ;字访问
```

一个字由两个字节组成，高字节（位 15-8）和低字节（位 7-0），他总是对准偶地址。

图 4.5：字节和字操作

对堆栈和 PC 的全部操作是字操作，并且对准偶地址。

字-字和字节-字节操作能同样准确得到操作数结果和状态位设置。

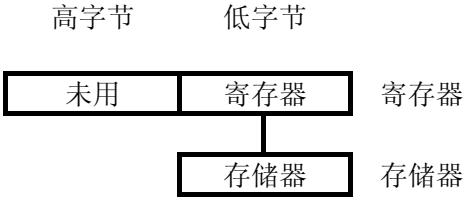
字-字操作：

```
R5 = 0F28Eh
EDE .EQU 0212h
Mem(0F28Eh) = 0FFFEh
Mem(0212h) = 00112h

ADD     @R5，&EDE

Mem(0212h) = 00110h
C = 1，  Z = 0，  N = 0
```

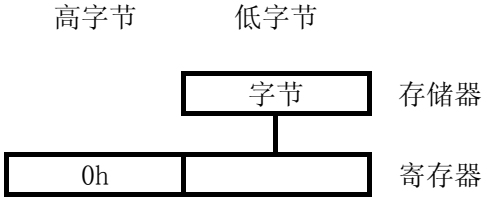
字节-字节操作：

```
R5 = 0223h
EDE .EQU 0202h
Mem(0223h) = 05Fh
Mem(0202h) = 043h

ADD.B   @R5，&EDE

Mem(0202h) = 0A2h
    C = 0，  Z = 0，  N = 1
```

寄存器-字节操作：



字节-寄存器操作：



例：寄存器-字节操作

R5 = 0A28Fh
R6 = 0203h
Mem(0203h) = 012h

ADD.B R5，0(R6)

```
  08Fh
+ 012h
  0A1h
```

Mem(0203h) = 0A1h
C = 0， Z = 0， N = 1

（寄存器低字节）
+（寻址字节）
->(寻址字节)

例：字节-寄存器操作

R5 = 01202h
R6 = 0223h
Mem(0223h) = 05Fh

ADD.B @R6，R5

```
  05Fh
+ 002h     ;R5 的低字节
  061h     ;存入 R5，高字节为 0
```

R5 = 061h
C = 0， Z = 0， N = 0

（寻址字节）
+（寄存器低字节）
->(寄存器低字节，高字节填 0)

注意： 字-字节操作

在存储器中不支持字-字节操作或者字节-字操作。
每一次寄存器-字节操作和字节-寄存器操作执行时相当于字节操作。

4.3.2 外围模块 - 地址定位

所有外围模块用软件访问和控制。可以用全部的数据交换指令。由于有因字结构而利用整个 MDB 的模块和只用 MDB 低 8 位的模块，地址空间从 0100h 到 01FFh 留作安排字模块，地址空间从 00h 到 0FFh 留作安排字模块。

安排在字地址空间的外围模块必须用字指令（如 MOV R5，&WDTCTL）访问，安排在字节地址空间的外围模块必须用字节指令（如 MOV #1，&TCCTL）访问。

这两种寻址可经绝对寻址模式实现，也可经过采用变址、间接或间接增量寻址模式的工作寄存器来实现。

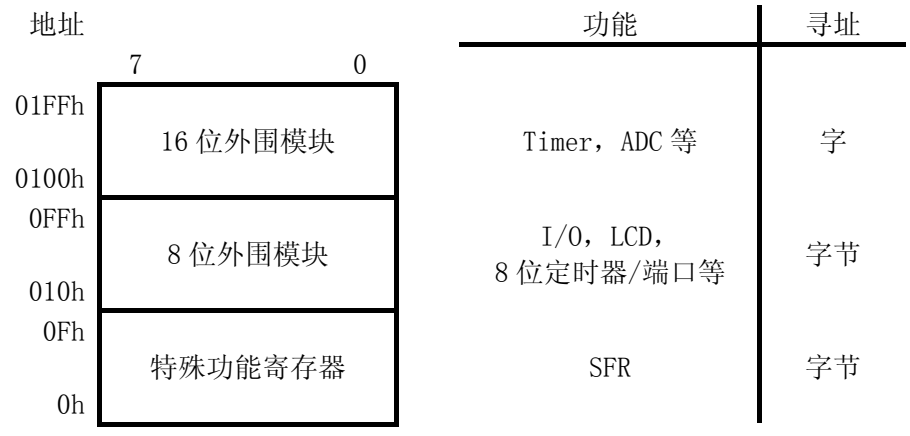


图 4.6：外围模块组织

字模块

字模块是经整个 16 位 MDB 连接的模块。访问字模块总是用字格式，因为硬件的字结构不支持字节访问。外围模块的寻址空间组织成 16 个帧，每帧有 8 个字。

地　　址	说　　明
1F0h - 1FFh	保留
1E0h - 1EFh	保留
1D0h - 1DFh	保留
1C0h - 1CFh	保留
1B0h - 1BFh	保留
1A0h - 1AFh	保留
190h - 19Fh	保留
180h - 18Fh	保留
170h - 17Fh	Timer_A
160h - 16Fh	Timer_A
150h - 15Fh	保留
140h - 14Fh	保留
130h - 13Fh	乘法器
120h - 12Fh	看门狗定时器
110h - 11Fh	A/D
100h - 10Fh	保留

图 4.7：外围模块地址分配 - 字模块

字节模块

字节模块是经 MDB 的低 8 位连接的模块。总是以字节格式来访问字节模块。在写操作时由于硬件的原因模块只取了低字节。

字节指令对字节模块的操作无任何限制。用字指令对字节外围模块作读访问会在高字节产生不可预知的结果。写入字节模块的字数据，低字节写入相应的模块寄存器，而高字节可忽略不计。

外围模块的地址空间被组织成 16 个帧。

地　　址	说　　明
00F0h - 00FFh	保留
00E0h - 00EFh	保留
00D0h - 00DFh	保留
00C0h - 00CFh	保留
00B0h - 00BFh	保留
00A0h - 00Afh	保留
0090h - 009Fh	保留
0080h - 008Fh	保留
0070h - 007Fh	USART
0060h - 006Fh	保留

0050h	-	005Fh	系统时钟发生器
0040h	-	004Fh	Basic Timer， 定时器/计数器，定时器/端口
0030h	-	003Fh	LCD
0020h	-	002Fh	P3，P4
0010h	-	001Fh	P0，P1，P2
0000h	-	000Fh	SFR

图 4.8：外围模块地址分配 - 字节模块

4.3.3 外围模块 - SFR

系统的构成和外围模块对处理机操作模式的各种响应主要由 SFR 来定义。SFR 位于更低的地址区，并以字节方式实现。SFR 只能用字节指令来访问。即使某些 SFR 位共享相同的地址空间，它们实际上是物理地存在于相应的外围模块中。

地 址	数 据 总 线 7 0
000Fh	无定义/未实现
000Eh	无定义/未实现
000Dh	无定义/未实现
000Ch	无定义/未实现
000Bh	无定义/未实现
000Ah	无定义/未实现
0009h	无定义/未实现
0008h	无定义/未实现
0007h	无定义/未实现
0006h	无定义/未实现
0005h	模块允许 2： ME2.x
0004h	模块允许 1： ME1.x
0003h	中断标志 2：IFG2.x
0002h	中断标志 1：IFG1.x
0001h	中断允许 2： IE2.x
0000h	中断允许 1： IE1.x

图 4.9：SFR 地址分配

MSP430 系列各型号支持它们各自的外围模块 SFR 的正确逻辑和功能。每个模块可以单独允许，以实现中断功能和操作。配置位的完全软件控制使得应用软件在中断允许屏蔽时重新激活系统。

系统的功耗受到被允许的模块数和它们的功能的影响。禁止一个活动模块会减少功耗。有两个部分是不能禁止的：ROM 和 RAM。处理机内核可以切换到禁止模式，即 CPUOff 模式，这时所有的内部功能都被禁止了：CPU 和总线停止了活动。

5．16 位 CPU

目录 页号

由于 PC 和工作寄存器的相等宽度带来了新的特性，例如，7 种寻址模式。

MSP430 采用“冯-纽曼结构”，RAM、ROM 在同一地址空间中，使用一组地址数据总线。

5.1 CPU 寄存器

14 个 16 位寄存器（R0，R1，R4 至 R15）用于存放数据和地址。这些寄存器在 CPU 内。利用它们能寻址达 64KB（ROM，RAM，EEPROM，外围模块，……）而不必分段。

全部 CPU 寄存器见下表。其中对有特殊用途的寄存器作了标明。寄存器 R0，R1，R2，R3 由于它们的特殊功能在通用性上有限制，将在以后说明。

程序计数器 PC	R0
堆栈指针 SP	R1
状态寄存器 SR/常数发生器 CG1	R2
常数发生器 CG2	R3
工作寄存器 R4	R4
工作寄存器 R5	R5
:	:
:	:
工作寄存器 R13	R13
工作寄存器 R14	R14
工作寄存器 R15	R15

表 5.1：寄存器功能

5.1.1 程序计数器 PC

16 位程序计数器 PC 确定了将要执行的下一条指令。每条指令占偶数字节：2、4 或 6 字节。指令访问发生在字边界，因此 PC 是对准偶地址的。PC 在取指令周期期间增加 2 以指向紧接当前执行指令后的字。利用紧跟当前指令的字的信息可能有两种寻址模式（立即寻址模式和符号寻址模式）。

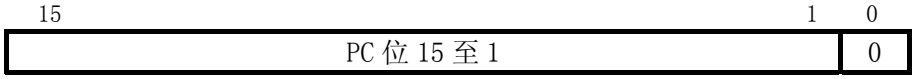


图 5.1：程序计数器 PC

5.1.2 系统堆栈指针 SP

系统堆栈指针 SP 总是对准偶地址，因此在中断服务期间以字数据访问堆栈。系统堆栈指针 SP 用于保存子程序调用和中断服务时的返回地址。它采用先减后加的方案。这一方案的优点是栈顶（TOS）数据项可用。SP 可能被用户软件所用（PUSH 和 POP 指令），但是必须记住 CPU 也在使用 SP。

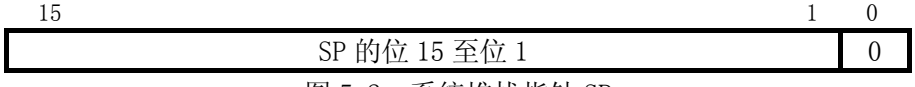


图 5.2：系统堆栈指针 SP

注意： 通用寄存器用作软件堆栈指针

通用寄存器 R4 至 R15 可以用作软件堆栈指针。

将数据项压入 Rn 控制的字软件堆栈：

```
DECD        Rn          ; 软件堆栈指针 Rn 减少 2
MOV         item，0(Rn) ; PUSH 数据项入软件堆栈
```

将数据项从软件堆栈推出：

```
MOV         @Rn+，item  ; POP 数据项出软件堆栈
```

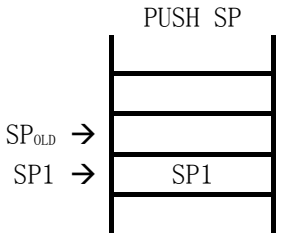
将数据项压入 Rm 控制的字节软件堆栈：

```
DEC         Rm          ; 软件堆栈指针 Rm 减少 1
MOV.B       item，0(Rm) ; PUSH 数据项入软件堆栈
```

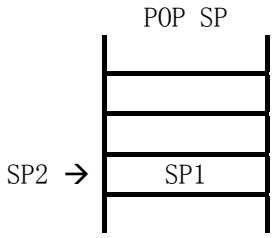
将数据项从软件堆栈推出：

```
MOV.B       @Rm+，item  ; POP 数据项推出软件堆栈
```

系统堆栈指针 PUSH 和 POP 的特殊情况。



SP 先减少 2，压入栈顶，
PUSH SP 指令后 SP 不变



SP 装入 POP SP 指令前，
SP 所指存储器地址的数据

在以下序列执行时

```
PUSH    SP          ; SP1 是执行 PUSH SP 指令后的堆栈指针
  |
  |
POP     SP          ; SP2 是执行 POP SP 指令后的堆栈指针
```

堆栈指针比序列执行前低 2 字节。

例：系统堆栈指针寻址：（参见下图中的堆栈）

```
MOV     SP，R4      ; 0xxxh - 4 送 R4
MOV     @SP，R5     ; 数据项 I3（栈顶）送 R5
MOV     2(SP)，R6   ; 数据项 I2 送 R6
MOV     R7，0(SP)   ; 用 R7 覆盖栈顶内容
MOV     R8，4(SP)   ; 用 R8 改数据项 I1
PUSH    R12         ; R12 存入 0xxxh - 6; SP 指向同一地址
POP     R12         ; 从 0xxxh - 6 恢复数据入 R12; SP 指向 0xxxh - 4
MOV     @SP+，R5    ; 数据项 I3 送 R5 (从堆栈推出); 与 POP 指令相同
PUSH    #1
POP     R8
```

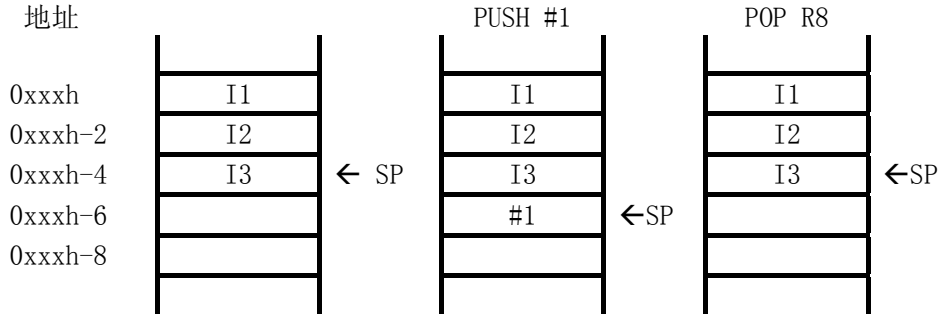


图 5.3：堆栈使用

5.1.3 状态寄存器 SR

状态寄存器 SR 含有 CPU 的个状态位：

- V　　溢出位
- SCG1　　系统时钟发生器控制位 1
- SCG0　　系统时钟发生器控制位 0
- OscOff　　晶振关闭位
- CPUOff　　CPU 关闭位
- GIE　　通用中断允许位
- N　　负数位
- Z　　零位
- C　　进位位

15 … 9	8	7							0
留作未来增强用	V	SCG1	SCG0	OscOff	CPUOff	GIE	N	Z	C
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

图 5.4：状态寄存器 SR

状态位说明

- V：　　当算术运算结果超出有符号数范围时置位，对字节和字格式数均有效：
  ADD(.B)， ADDC(.B)　　当出现以下情况时置位：
  正数 + 正数 = 负数
  负数 + 负数 = 正数
  其它情况下均复位
  SUB(.B)， SUBC(.B)，CMP(.B)　　当出现以下情况时置位：
  正数 - 负数 = 负数
  负数 - 正数 = 正数
  其它情况下均复位
- SCG1，SCG0：　　这些位控制系统时钟发生器的 4 种活动状态，因此影响处理机运行。
- OscOff：　　置位使晶振进入关闭模式：除 RAM 内容、端口和寄存器保持外，全部活动部件停止。只可能在 GIE 置位时由外部中断或由 NMI 唤醒。如不同时对 CPUOff 置位，不能对它置位。
- CPUOff：　　置位使 CPU 进入关闭模式：除 RAM、端口、寄存器和特别允许的外围模块，如 Basic Timer，UART ... 保持活动外，全部活动部分停止。所有允许的中断可以唤醒。

- GIE： 置位使被允许的中断可处理，复位禁止所有中断。GIE 位由中断复位，由 RETI 指令恢复。它也可用指令改变。
- N： 运算结果为负时置位。
  字运算： N 位设置为运算结果的 15 位。
  字节运算： N 位设置为运算结果的 7 位。
- Z： 当运算结果为 0 时置位，不为 0 时复位。
- C： 当运算结果产生进位时置位，无进位时复位。
  字运算： 进位是指字运算结果发生进位。
  字节运算： 进位是指字节运算结果发生进位。
  某些指令以 Z 位的非来修改 C 位。

注意： 状态位 V、N、Z 和 C
状态位 V、N、Z 和 C 只在某些指令执行时改变。请看指令组详细说明，MSP 软件用户指南。

5.1.4 常数发生寄存器 CG1 和 CG2

经常使用的常数可由常数发生器 R2 和 R3 产生，而不必占用一个 16 位字。所用的常数的数值由寻址位 As 来定义：

寄存器	As	常数	说明
R2	00	--------	寄存器模式
R2	01	(0)	绝对寻址模式
R2	10	00004h	+4，位处理
R2	11	00008h	+8，位处理
R3	00	00000h	0，字处理
R3	01	00001h	+1
R3	10	00002h	+2，位处理
R3	11	0FFFFh	-1，字处理

表 5.2：常数发生器 CG1、CG2 的值

使用这种方法产生常数的优点：

- 不需要特殊的指令
- 对 7 种最常用的常数不需要额外的字操作数
- 缩短指令周期：不经过 MDB 就能直接访问

当 6 种常数之一被用作立即寻址模式的源操作数，汇编程序会自动利用 R2 或 R3 模式。状态寄存器 SR/R2（用作源或目的寄存器）只能用于寄存器模式。其它的寻址位 As 的组合支持绝对寻址和位处理而不必增加其它代码。R2 与 R3 用于常数模式时不能显式寻址，它们只能作为一个寄存器数据源。

常数发生器允许一些指令用别的指令来模拟。用这种方法 CPU 变得异常简单。整个指令组只需要 27 条指令。例如单操作数指令：

```
CLR     dst
```

可以用双操作数指令以同样长度来模拟：

```
                MOV R3，dst
或等价地有      MOV #0，dst
```

其中#0 由汇编程序以 As=00 的 R3 来取代，优点是：

- 单字指令
- CPU 内不需要额外的硬件和控制操作
- 源操作数为寄存器寻址：对常数(#0)的读取不需要额外的取周期。

5.2 寻址模式

对源操作数的全部7种寻址模式和对目的操作数的全部4种寻址模式可以访问整个地址空间。由 As 和 Ad 模式位的内容确定。

As/Ad	寻址模式	语法	说明
00/0	寄存器模式	Rn	寄存器内容为操作数。
01/1	变址模式	X(Rn)	(Rn+X)指向操作数，X 存于后续字中。
01/1	符号模式	ADDR	(PC+X)指向操作数，X 存于后续字中，使用了变址模式的 X(PC)。
01/1	绝对模式	&ADDR	指令后续字含绝对地址。
10/-	间接寄存器模式	@Rn	Rn 为指针指向操作数。
11/-	间接自动增量模式	@Rn+	Rn 为指针指向操作数，然后 Rn 增加。
11/-	立即模式	#N	指令后续字含立即数 N，使用了间接自动增量模式的@PC+。

注意： 寻址模式
寻址模式将 PC 当作工作寄存器是寻址模式的标准作用。PC 指向当前执行指令后续 ROM 字形成特定寻址模式。

以下用例子详细解说 7 种寻址模式。大多数例子对源和目的操作数用了相同的寻址模式，但是对于一条指令，任何有效的源和目的操作数的寻址模式组合都是可能的。

5.2.1 寄存器模式

汇编源程序	ROM 中内容
MOV R10，R11	MOV R10，R11

长度： 1 或 2 字
操作： 移动 R10 内容到 R11。R10 不受影响。
备注： 对源和目的操作数均有效。
举例： MOV R10，R11

执行前		执行后	
R10	0A023h	R10	0A023h

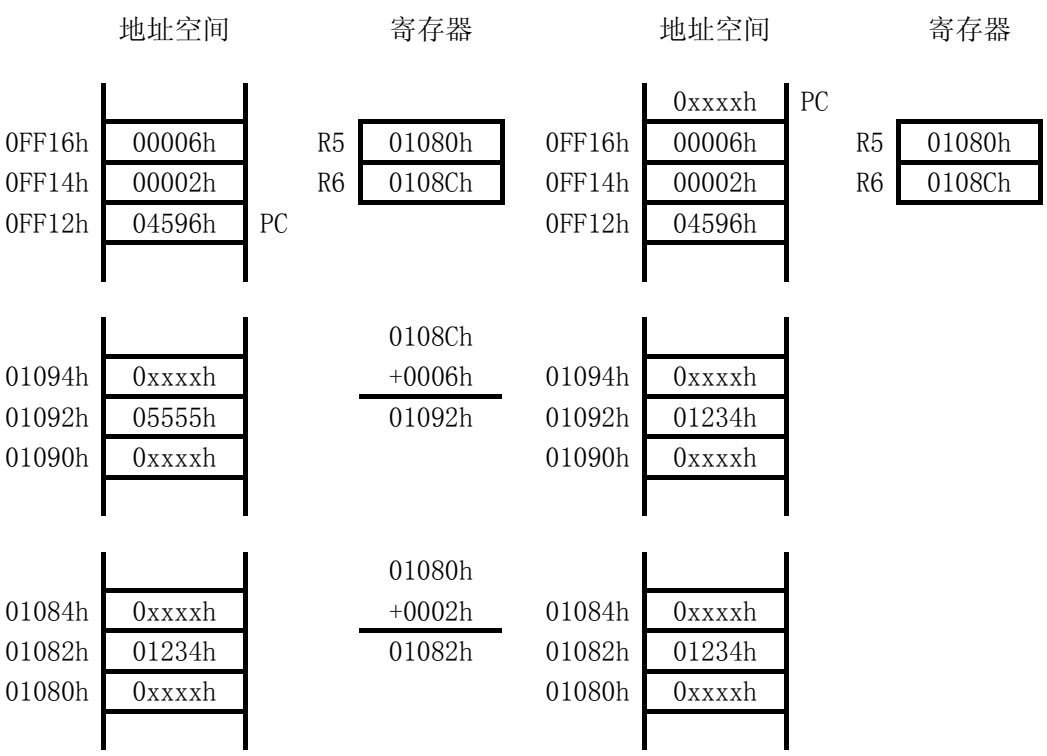
注意： 寄存器中的数据
因为寄存器中的数据是字数据，任何寄存器-寄存器操作总是字操作，必须用字指令。

5.2.2 变址模式

汇编源程序	ROM 中内容
MOV 2(R5)，6(R6)	MOV X(R5)，Y(R6) X = 2 Y = 6

长度： 2 或 3 字
操作： 移动源地址内容(R5+2 的内容)到目的地址(R6+6 的内容)。源和目的寄存器(R5 和 R6)不受影响。变址模式中 PC 自动增加，因此程序继续执行下条指令。
备注： 对源和目的操作数均有效。
举例： MOV 2(R5)，6(R6)

执行前 **执行后**



5.2.3 符号模式

汇编源程序	ROM 中内容
MOV EDE, TONI	MOV X(PC), Y(PC) X = EDE-PC Y = TONI-PC

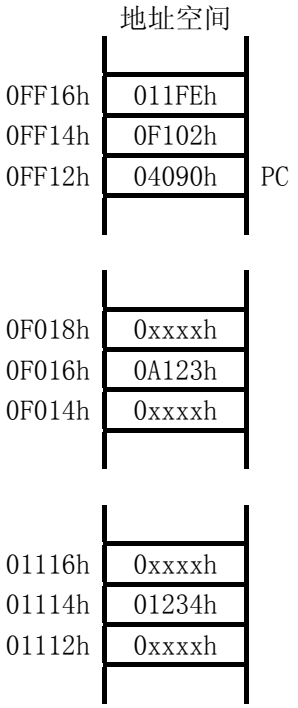
长度： 2 或 3 字

操作： 移动源地址 EDE 内容(PC+X 的内容)到目的地址 TONI(PC+Y 的内容)。指令的后续字含 PC 与源和目的地址的差。汇编程序自动计算并插入偏移量 X 与 Y。符号模式中 PC 自动增加，因此程序继续执行下条指令。

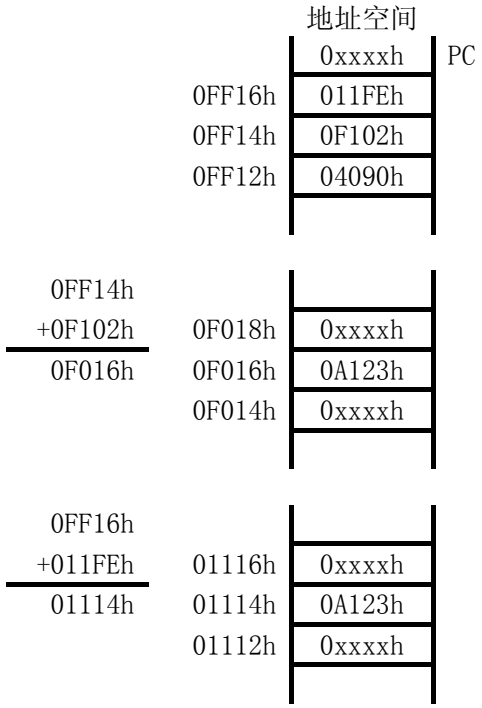
备注： 对源和目的操作数均有效。

举例：
```
MOV     EDE, TONI       ;源地址 EDE=0F016h
                        ;目的地址 TONI=01114h
```

执行前



执行后



5.2.4 绝对模式

汇编源程序	ROM 中内容
MOV &EDE，&TONI	MOV X(0)，Y(0) X = EDE Y = TONI

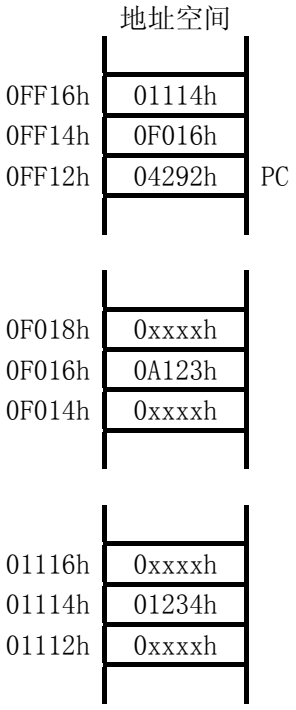
长度： 2 或 3 字

操作： 移动源地址 EDE 内容到目的地址 TONI。指令的后续字含源和目的地址的绝对地址。汇编程序自动计算并插入偏移量 X 与 Y。绝对模式中 PC 自动增加，因此程序继续执行下条指令。

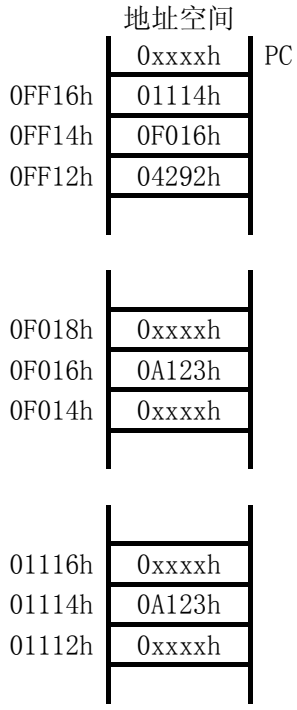
备注： 对源和目的操作数均有效。

举例：
```
MOV      &EDE，&TONI      ;源地址 EDE=0F016h
                          ;目的地址 TONI=01114h
```

执行前



执行后



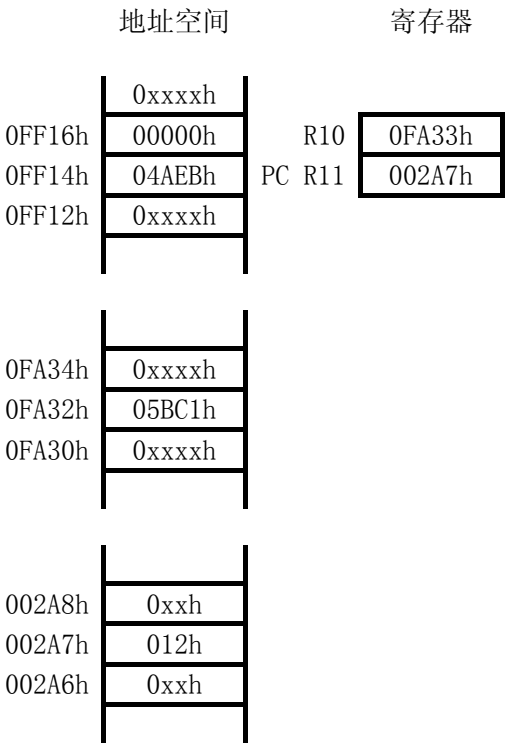
此寻址模式主要用在定位于绝对的、固定地址的硬件外围模块。对它们用绝对寻址可保证软件的透明度，如位置独立代码（PIC）编程技术。绝对模式只用于代码段 0。

5.2.5 间接模式

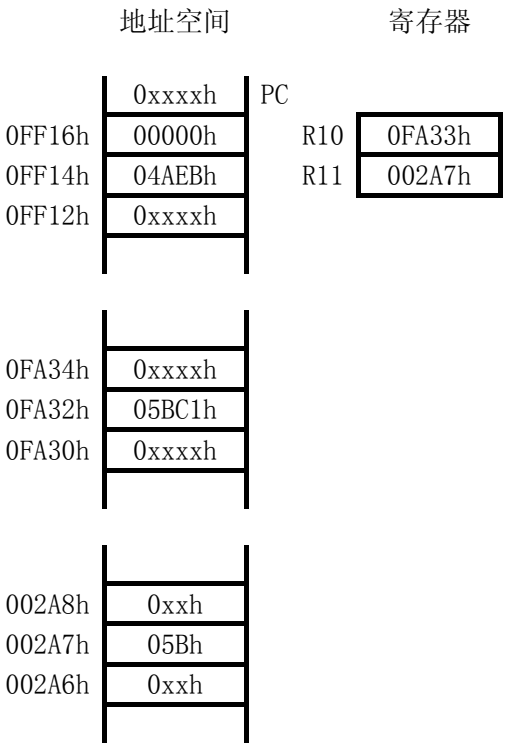
汇编源程序	ROM 中内容
MOV @R10，0(R11)	MOV @R10，0(R11)

长度： 1 或 2 字
操作： 移动源地址内容(R10 的内容)到目的地址(R11 的内容)。寄存器不受影响。
备注： 仅对源操作数有效。对目的操作数的替代方法是 0(Rd)。
举例： MOV.B @R10，0(R11)

执行前



执行后



5.2.6 间接增量模式

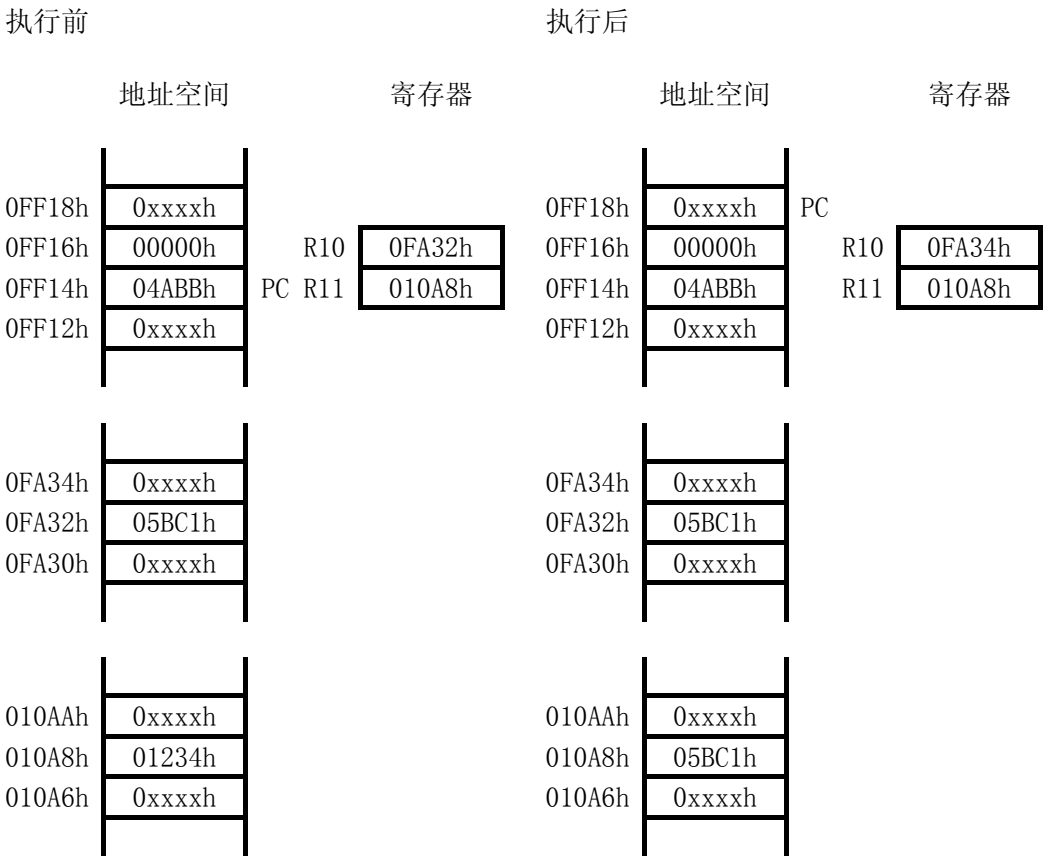
汇编源程序	ROM 中内容
MOV @R10+, 0(R11)	MOV @R10+, 0(R11)

长度： 1 或 2 字

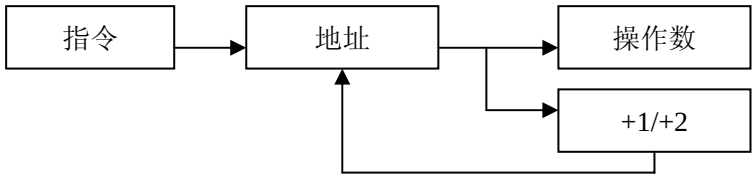
操作： 移动源地址内容(R10 的内容)到目的地址(R11 的内容)。R10 取数后增加 1（字节操作）或增加 2（字操作）：它不需要额外开销现已指向下一地址。这对于表处理非常有用。

备注： 仅对源操作数有效。对目的操作数的替代方法是 0(Rd)并加 1 条指令 INCD Rn。

举例： MOV @R10+, 0(R11)



在执行时寄存器的自动增量在取操作数之后。



5.2.7 立即模式

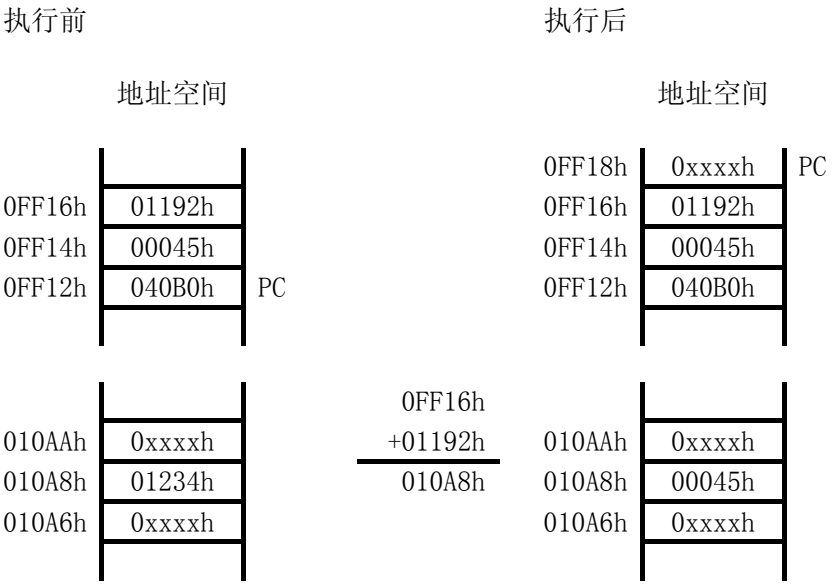
汇编源程序	ROM 中内容
MOV #45h, TONI	MOV @PC+, X(PC) 45 X = TONI-PC

长度： 2 或 3 字

操作： 移动含在后续字中的立即数 45 目的地址 TONI。当取得源操作数时 PC 指向后续字并移动数据到目的地址。

备注： 仅对源操作数有效。

举例： MOV #45h, TONI



5.2.8 指令的时钟周期与长度

CPU 的运行速度不依赖于个别指令。它取决于指令的格式和寻址模式。表中时钟周期数是针对内部振荡器频率的。

I 类格式指令

寻址模式		周期数	指令长度	实例
As	Ad			
00，Rn	0，Rm	1	1	MOV R5，R8
	0，PC	2	1	BR R9
00，Rn	1，x(Rm)		2	ADD R5，3(R6)
	1，EDE	4	2	XOR R8，EDE
	1，&EDE		2	MOV R5，&EDE
01，x(Rn)	0，Rm		2	MOV 2(R5)，R7
01，EDE		3	2	AND EDE，R6
01，&EDE				MOV &EDE，R8
01，x(Rn)	1，x(Rm)		3	ADD 3(R4)，6(R9)
01，EDE	1，EDE	6	3	CMP EDE，TONI
01，&EDE	1，&EDE		3	MOV 2(R5)，&TONI
				ADD EDE，&TONI
10，@Rn	0，Rm	2	1	AND @R4，R5
10，@Rn	1，x(Rm)		2	XOR @R5，8(R6)
	1，EDE	5	2	MOV @R5，EDE
	1，&EDE		2	XOR @R5，&EDE
11，@Rn+	0，Rm	2	1	ADD @R5+，R6
	0，PC	3	1	BR @R9+
11，#N	0，Rm	2	2	MOV #20，R9
	0，PC	3	2	BR #2AEh
11，@Rn+	1，x(Rm)		2	MOV @R9+，2(R4)
11，#N	1，EDE	5	3	ADD #33，EDE
11，@Rn+	1，&EDE		2	MOV @R9+，&EDE
11，#N			3	ADD #33，&EDE

II 类格式指令

寻址模式 A(s/d)	周期数 RRA RRC SWPB SXT	周期数 PUSH/ CALL	指令长度	实例
00，Rn	1	3/4	1	SWPB R5
01，x(Rn)	4	5	2	CALL 2(R7)
01，EDE	4	5	2	PUSH EDE
01，&EDE				SXT &EDE
10，@Rn	3	4	1	RRC @R9
11，@Rn+ *	3	4/5	1	SWPB @R10+
11，#N			2	CALL #81h

注意：II 类格式指令的立即模式（见上表中 * 号）
指令 RRA，RRC，SWPB 和 SXT 不能对目的操作数用立即模式寻址。这会产生不可预知的程序运行结果。

III 类格式指令

Jxx 指令无论跳转与否都需要相同的周期数。

时钟周期： 2
指令长度： 1 字

其它指令或操作

RETI	时钟周期：	5
	指令长度：	1 字
中断	时钟周期：	6
WDT 复位	时钟周期：	4
复位(RST*/NMI)	时钟周期：	4

5.3 指令组概述

以下对指令组作简单介绍。

指令对 SR 位的影响表示如下：

*	影响状态位
-	不影响状态位
0	状态位复位
1	状态位置位

指令的源操作数和目的操作数部分每个定义为 2 个字段（寻址模式见前述）：

src	源操作数，由 As 和 S-reg 定义
dst	目的操作数，由 Ad 和 D-reg 定义
As	寻址位，表示源操作数的寻址模式
S-reg	源操作数使用的工作寄存器
Ad	寻址位，表示目的操作数的寻址模式
D-reg	源操作数使用的工作寄存器
B/W	字节或字操作， 0： 字操作 1： 字节操作

注意： 目的地址
目的地址可以是 64KB 范围内任何一处。数据写入的操作数目的地址必须是数据能够写入的地址，否则数据将丢失。

5.3.1 双操作数指令

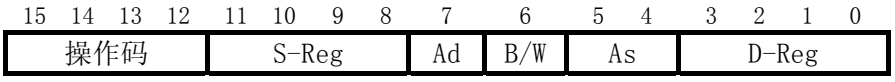


图 5.5：双操作数指令格式

			状态位			
			V	N	Z	C
MOV	src，dst	src → dst	-	-	-	-
ADD	src，dst	src + dst → dst	*	*	*	*
ADDC	src，dst	src + dst + C → dst	*	*	*	*
SUB	src，dst	dst + .not.src + 1 → dst	*	*	*	*
SUBC	src，dst	dst + .not.src + C → dst	*	*	*	*
CMP	src，dst	dst - src	*	*	*	*
DADD	src，dst	src + dst + C → dst (dec)	*	*	*	*
AND	src，dst	src .and. dst → dst	0	*	*	*
BIT	src，dst	src .and. dst	0	*	*	*
BIC	src，dst	.not.src .and. dst → dst	-	-	-	-
BIS	src，dst	src .or. dst → dst	-	-	-	-
XOR	src，dst	src .xor. dst → dst	*	*	*	*

注意： CMP 和 SUB 指令
除了结果的保存，指令 CMP 与 SUB 相同。这一点对于指令 BIT 与 AND 也是同样的。

5.3.2 单操作数指令

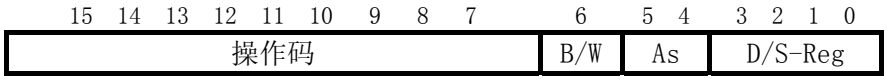


图 5.6：单操作数指令格式

			状态位			
			V	N	Z	C
RRC	dst	C → MSB → .....LSB → C	*	*	*	*
RRA	dst	MSB → MSB →.....LSB → C	0	*	*	*
PUSH	src	SP - 2 → SP， src → @SP	-	-	-	-
SWPB	dst	字节交换	-	-	-	-
CALL	dst	SP - 2 → SP	-	-	-	-
		PC+2 → 堆栈， dst → PC				
RETI		TOS → SR， SP ← SP + 2	*	*	*	*
		TOS → PC， SP ← SP + 2				
SXT	dst	Bit7 → Bit8 ..... Bit15	0	*	*	*

CALL 指令可以用所有寻址模式。如用符号模式(地址)、立即模式(#N)、绝对模式(&EDE)或变址模式(X(Rn))，指令的后续字含地址信息。

5.3.3 条件跳转

条件跳转指令可使程序产生相对于 PC 的分枝跳转。可跳转地址是相对于跳转指令执行时 PC 的-511 至+512 字范围内。10 位的 PC 偏移量是一个 10 位的有符号数，乘 2 后与 PC 相加。条件跳转不影响状态位。

取指令码和 PC 增加按以下公式：

PCnew = PCold + 2 + PCoffset * 2

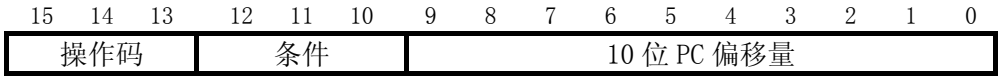


图 5.7：条件跳转指令格式

JEQ/JZ	Label	Z = 1 时，跳转到 Label
JNE/JNZ	Label	Z = 0 时，跳转到 Label
JC	Label	C = 1 时，跳转到 Label
JNC	Label	C = 0 时，跳转到 Label
JN	Label	N = 1 时，跳转到 Label
JGE	Label	(N .XOR. V) = 0 时，跳转到 Label
JL	Label	(N .XOR. V) = 1 时，跳转到 Label
JMP	Label	无条件跳转到 Label

5.3.4 模拟指令的短格式

使用常数发生器的基本指令可以表示为更为流行的模拟指令。对状态位的影响依赖于基本指令执行的结果。

指令助记符		说明	状态位 V N Z C	替代的模拟指令	
算术指令					
ADC[.W]	dst	C 位加到目的操作数	* * * *	ADDC	#0，dst
ADC.B	dst	C 位加到目的操作数	* * * *	ADDC.B	#0，dst
DADC[.W]	dst	C 位十进制加到目的操作数	* * * *	DADD	#0，dst
DADC.B	dst	C 位十进制加到目的操作数	* * * *	DADD.B	#0，dst
DEC[.W]	dst	目的操作数减 1	* * * *	SUB	#1，dst
DEC.B	dst	目的操作数减 1	* * * *	SUB.B	#1，dst
DECD[.W]	dst	目的操作数减 2	* * * *	SUB	#2，dst
DECD.B	dst	目的操作数减 2	* * * *	SUB.B	#2，dst
INC[.W]	dst	目的操作数加 1	* * * *	ADD	#1，dst
INC.B	dst	目的操作数加 1	* * * *	ADD.B	#1，dst
INCD[.W]	dst	目的操作数加 2	* * * *	ADD	#2，dst
INCD.B	dst	目的操作数加 2	* * * *	ADD.B	#2，dst
SBC[.W]	dst	从目的操作数减去 C 位	* * * *	SUBC	#0，dst
SBC.B	dst	从目的操作数减去 C 位	* * * *	SUBC.B	#0，dst
逻辑指令					
INV[.W]	dst	目的操作数取反	* * * *	XOR	#0FFFFh，dst
INV.B	dst	目的操作数取反	* * * *	XOR.B	#0FFFFh，dst
RLA[.W]	dst	算术左循环	* * * *	ADD	dst，dst
RLA.B	dst	算术左循环	* * * *	ADD.B	dst，dst
RLC[.W]	dst	经 C 位算术左循环	* * * *	ADDC	dst，dst
RLC.B	dst	经 C 位算术左循环	* * * *	ADDC.B	dst，dst
数据指令(常用)					
CLR[.W]		清除目的操作数	- - - -	MOV	#0，dst
CLR.B		清除目的操作数	- - - -	MOV.B	#0，dst
CLRC		C 位复位	- - - 0	BIC	#1，SR
CLRN		N 位复位	- 0 - -	BIC	#4，SR
CLRZ		Z 位复位	- - 0 -	BIC	#2，SR
POP	dst	数据出栈	- - - -	MOV	@SP+，dst
SETC		C 位置位	- - - 1	BIS	#1，SR
SETN		N 位置位	- 1 - -	BIS	#4，SR
SETZ		Z 位置位	- - 1 -	BIS	#2，SR
TST[.W]	dst	目的操作数测试	0 * * *	CMP	#0，dst
TST.B	dst	目的操作数测试	0 * * *	CMP.B	#0，dst
程序控制指令					
BR	dst	跳转	- - - -	MOV	dst，PC
DINT		关中断	- - - -	BIC	#8，SR
EINT		开中断	- - - -	BIS	#8，SR
NOP		空操作	- - - -	MOV	#0h，#0h
RET		从子程序返回	- - - -	MOV	@SP+，PC

5.3.5 其它指令

对有些操作数，如 CPUOff 等，没有相应的指令。这些功能的打开和关闭是通过对 SR 及外围模块寄存器中各位的复位和置位来实现的。还有的是用双操作数指令来模拟。

以下是一些例子：

```
BIC     #1，SR          ; C 位复位
MOV     #0，#0          ; 空操作
BIC     #8，SR          ; 关中断
BIS     #28h，SR        ; 进入 OscOff 模式，并允许通用中断 GIE 位
BIS     #18h，SR        ; 进入 CPUOff 模式，并允许通用中断 GIE 位
BIC     #SVCC，ACTL     ; 关闭 SVCC
```

5.4 指令分布

以下的指令分布表说明了指令是如何编码的。空格表示根据需要还可增加新的指令。

	000	040	080	0C0	100	140	180	1C0	200	240	280	2C0	300	340	380	3C0
0x																
04x																
08x																
0Cx																
10x	RRC	RRC.B	SWPB		RRA	RRA.B	SXT		PUSH	PUSH.B	CALL		RETI			
14x																
18x																
1Cx																
20x	JNE/JNZ															
24x	JEQ/JZ															
28x	JNC															
2Cx	JC															
30x	JN															
34x	JGE															
38x	JL															
3Cx	JMP															
40x..4Cx	MOV, MOV.B															
50x..5Cx	ADD, ADD.B															
60x..6Cx	ADDC, ADDC.B															
70x..7Cx	SUBC, SUBC.B															
80x..8Cx	SUB, SUB.B															
90x..9Cx	CMP, CPM.B															
A0x..ACx	DADD, DADD.B															
B0x..BCx	BIT, BIT.B															
C0x..CCx	BIC, BIC.B															
D0x..DCx	BIS, BIS.B															
E0x..ECx	XOR, XOR.B															
F0x..FCx	AND, AND.B															

图 5.8：核心指令分布

6．硬件乘法器

由于硬件乘法器可视为一个未集成入 CPU 的 16 位外围模块，因此 CPU 的结构和指令都没有因此改变。乘法中两个操作数均被存入乘法器的寄存器中，用户在输入第二个操作数之后便可以读取结果，也就是说该乘法操作并不需要额外的时钟周期。

目录 页号

Msp430 系列采用的硬件乘法器模块能够在不改变基本结构的条件下增加其它的功能。可采用的乘法操作有：

- 16 位 x 16 位
- 16 位 x 8 位
- 8 位 x 16 位
- 8 位 x 8 位

硬件乘法器模块支持 3 种乘法：无符号数相乘（MPY）；有符号数相乘 (MPYS)；无符号乘加 (MAC)。

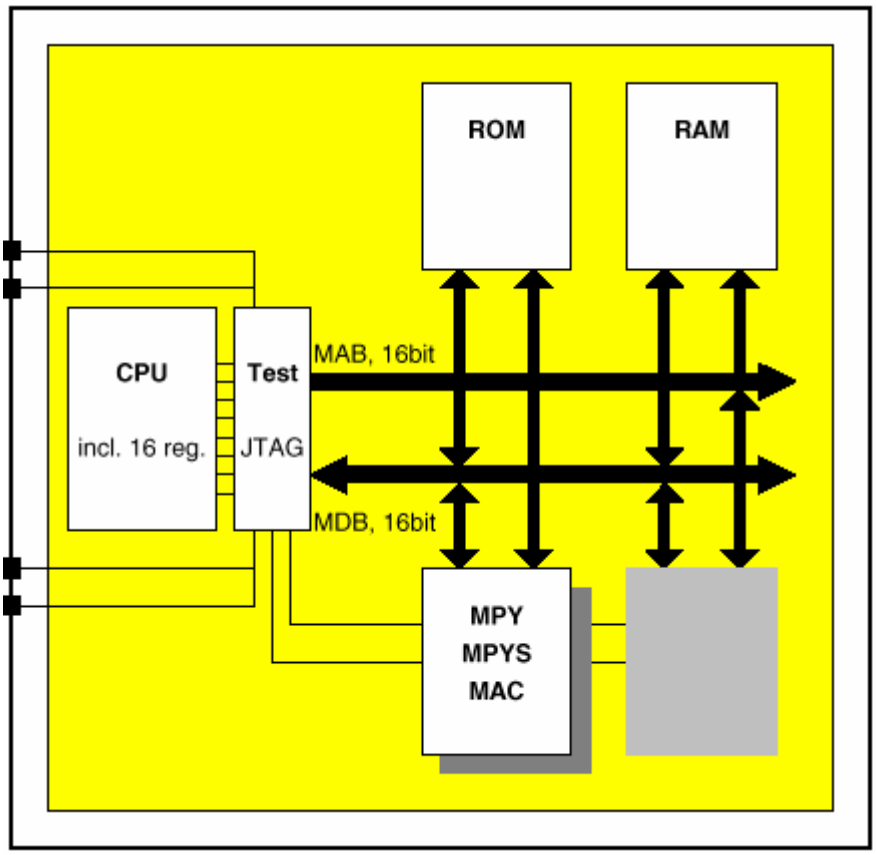


图 6.1：硬件乘法器模块与总线系统的连接

6.1 硬件乘法器操作

硬件乘法器有 2 个存放操作数的 16 位的寄存器和 3 个存放相乘结果的寄存器。只有当用户将第一操作数在第二操作数之前存入操作数寄存器中，乘法操作才会被正确执行。当第一操作数写入相应的寄存器时，乘法的种类即被确定。当用户写入第二个操作数时，乘法操作便开始执行并在用户用变址寻址模式读结果寄存器之前结束。如果用户采用间接寻址或间接自动增量寻址模式读写第二个操作数的寄存器和结果寄存器，就必须在写第二个操作数与读结果寄存器两条指令之间插入一条指令。硬件乘法器中 2 个操作数的寄存器均可用前面章节中所述的 7 种访问模式中的任一种进行访问。另外，硬件乘法器的操作与 CPU 的实时操作

和中断无关。

无符号数相乘，16 位 x 16 位；16 位 x 8 位；8 位 x 16 位；8 位 x 8 位

```
*******************************************************************************
*   TRANSFER BOTH OPERANDS TO THE REGISTERS IN THE HARDWARE                   *
*   MULTIPLIER MODULE                                                         *
*   USE CONSTANT OPERAND1 AND OPERAND 2 TO IDENTIFY BYTE DATA                 *
*******************************************************************************
OPERAND1    .EQU        0           ; 0: OPERAND1 IS WORD (16BIT)
                                    ; 8: OPERAND1 IS BYTE ( 8BIT)
OPERAND2    .EQU        0           ; 0: OPERAND2 IS WORD (16BIT)
                                    ; 8: OPERAND2 IS BYTE ( 8BIT)
MPY         .EQU        0130H
MPYS        .EQU        0132H
MAC         .EQU        0134H
OP2         .EQU        0138H
RESLO       .EQU        013AH
RESHI       .EQU        013CH
SUMEXT      .EQU        013EH
            .BSS        OPER1,2,200H
            .BSS        OPER2,2
            .BSS        RAM,8
            .IF OPERAND1=8
            MOV.B       &OPER1,&MPY ; LOAD 1ST OPERAND,
                                    ; DEFINES ADD. UNSIGNED MULTIPLY
            .ELSE
            MOV         &OPER1,&MPY ; LOAD 1ST OPERAND,
                                    ; DEFINES ADD. UNSIGNED MULTIPLY
            .ENDIF
            .IF OPERAND1=8
            MOV.B       &OPER2,&OP2 ; LOAD 2ND OPERAND AND START
                                    ; MULTIPLICATION
            .ELSE
            MOV         &OPER2,&OP2 ; LOAD 2ND OPERAND AND START
                                    ; MULTIPLICATION
            .ENDIF
*******************************************************************************
*   EXAMPLE TO ADD THE RESULT OF THE HARDWARE MULTIPLICATION                  *
*   TO THE RAM DATA, 64BITS                                                   *
*******************************************************************************
            ADD         &RESLO,&RAM     ; ADD LOW RESULT TO RAM
            ADDC        &RESHI,&RAM+2   ; ADD HIGH RESULT RO RAM+2
            ADC         &RAM+4          ; ADD CARRY TO EXTENSION WORD
            ADC         &RAM+6          ; IF 64 BIT LENGHT IS USED
```

程序代码 32 字节，执行周期，32（16 位 x 16 位）

有符号数相乘，16 位*16 位；16 位*8 位；8 位*16 位；8 位*8 位

```
*******************************************************************************
*   TRANSFER BOTH OPERANDS TO THE REGISTERS IN THE HARDWARE                   *
*   MULTIPLIER MODULE                                                         *
*   IF ONE OF THE OPERANDS IS 8BIT, SIGN EXTENSION IS NEEDED                  *
*   USE CONSTANT OPERAND1 AND OPERAND 2 TO IDENTIFY BYTE DATA                 *
*******************************************************************************
OPERAND1    .EQU        0           ; 0: OPERAND1 IS WORD (16BIT)
                                    ; 8: OPERAND1 IS BYTE ( 8BIT)
OPERAND2    .EQU        0           ; 0: OPERAND2 IS WORD (16BIT)
                                    ; 8: OPERAND2 IS BYTE ( 8BIT)
MPY         .EQU        0130H
MPYS        .EQU        0132H
MAC         .EQU        0134H
OP2         .EQU        0138H
RESLO       .EQU        013AH
RESHI       .EQU        013CH
SUMEXT      .EQU        013EH
            .BSS        OPER1,2,200H
            .BSS        OPER2,2
            .BSS        RAM,8
            .IF OPERAND1=0
            MOV         &OPER1,&MPYS; LOAD 1ST (WORD) OPERAND
                                    ; DEFINES ADD. SIGNED MULTIPLY
            .ELSE
            MOV.B       &OPER1,&MPYS; LOAD 1ST (BYTE) OPERAND,
                                    ; DEFINES ADD. SIGNED MULTIPLY
            SXT         &MPYS       ; EXPAND BYTE TO SIGNED WORD DATA
            .ENDIF
            .IF OPERAND2=0
            MOV         &OPER2,&OP2 ; LOAD 2ND (WORD) OPERAND AND
                                    ; START SIGNED MULTIPLICATION
            .ELSE
            MOV.B       &OPER2,&OP2 ; LOAD 2ND (BYTE) OPERAND,
            SXT         &OP2        ; RE-LOAD 2ND OPERAND AND START
                                    ; SIGNED 'FINAL' MULTIPLICATION
            .ENDIF
*******************************************************************************
*   EXAMPLE TO ADD THE RESULT OF THE HARDWARE MULTIPLICATION                  *
*   TO THE RAM DATA, 64BITS                                                   *
*******************************************************************************
            ADD         &RESLO,&RAM     ; ADD LOW RESULT TO RAM
            ADDC        &RESHI,&RAM+2   ; ADD HIGH RESULT RO RAM+2
            ADDC        &SUMEXT,&RAM+4  ; ADD SIGN WORD TO EXTENSION WORD
            ADDC        &SUMEXT,&RAM+6  ; IF 64 BIT LENGHT IS USED
```

程序代码 36 字节，执行周期 36（16 位 x 16 位）

无符号数乘加，16 位*16 位；16 位*8 位；8 位*16 位；8 位*8 位

```
*******************************************************************************
*   TRANSFER BOTH OPERANDS TO THE REGISTERS IN THE HARDWARE                   *
*   MULTIPLIER MODULE                                                         *
*   THE RESULT OF THE MULTIPLICATION IS ADDED TO THE CONTENT                  *
*   OF BOTH RESULT REGISTERS, RESLO AND RESHI                                 *
*   USE CONSTANT OPERAND1 AND OPERAND 2 TO IDENTIFY BYTE DATA                 *
*******************************************************************************
OPERAND1    .EQU        0           ; 0: OPERAND1 IS WORD (16BIT)
                                    ; 8: OPERAND1 IS BYTE ( 8BIT)
OPERAND2    .EQU        0           ; 0: OPERAND2 IS WORD (16BIT)
                                    ; 8: OPERAND2 IS BYTE ( 8BIT)
MPY         .EQU        0130H
MPYS        .EQU        0132H
MAC         .EQU        0134H
OP2         .EQU        0138H
RESLO       .EQU        013AH
RESHI       .EQU        013CH
SUMEXT      .EQU        013EH
            .BSS        OPER1,2,200H
            .BSS        OPER2,2
            .BSS        RAM,8
            .IF OPERAND1=8
            MOV.B       &OPER1,&MAC ; LOAD 1ST OPERAND,
                                    ; DEFINES ADD. UNSIGNED MULTIPLY
            .ELSE
            MOV         &OPER1,&MAC ; LOAD 1ST OPERAND,
                                    ; DEFINES ADD. UNSIGNED MULTIPLY
            .ENDIF
            .IF OPERAND1=8
            MOV.B       &OPER2,&OP2 ; LOAD 2ND OPERAND AND START
                                    ; MULTIPLICATION
            .ELSE
            MOV         &OPER2,&OP2 ; LOAD 2ND OPERAND AND START
                                    ; MULTIPLICATION
            .ENDIF
*******************************************************************************
*   EXAMPLE TO ADD THE RESULT OF THE HARDWARE MULTIPLICATION                  *
*   TO THE RAM DATA, 64BITS                                                   *
*   THE RESULT OF THE MULTIPLICATION IS HELD IN RESLO AND                     *
*   RESHI REGISTERS. THE UPPER TWO WORDS IN THE EXAMPLE ARE                   *
*   FURTHER LOCATED IN THEIR RAM LOCATION                                     *
*******************************************************************************
            ADDC        &SUMEXT,&RAM+4  ; ADD SUMEXTENSION TO RAM+4
            ADC         &RAM+6          ; IF 64 BIT LENGHT IS USED
```

程序代码 32 字节，执行周期 32（16 位 x 16 位）

6.2 硬件乘法器寄存器

硬件乘法器的硬件模块虽然是以字为基本单位，但是用户仍可以用处理字或字节的指令对字或字节进行操作。

寄存器	缩写	寄存器类型	地址	初始状态
● 无符号数相乘/操作数 1	MPY	读写	0130	不改变
● 有符号数相乘/操作数 1	MPYS	读写	0132	不改变
● 有符号数乘加/操作数 1	MAC	读写	0134	不改变
● 保留			0136	不改变
● 操作数 2	OP2	读写	0138	不改变
● 结果寄存器低位字寄存器	ResLo	读写	013A	未定义
● 结果寄存器高位字寄存器	ResHi	读写	013C	未定义
● 相加总和延伸寄存器	SumExt	只读	013E	未定义

两个操作数 OP1 和 OP2 均有各自的寄存器，其中 OP1 用 3 种不同地址访问同一个寄存器。不同的地址信息经解码定义了执行 3 种乘法操作（无符号数相乘、有符号数相乘和有符号数乘加）之一。

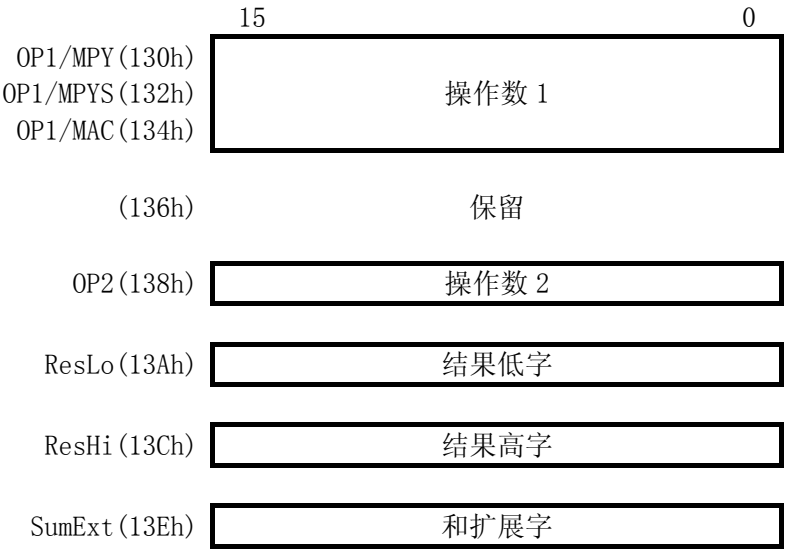


图 6.2：硬件乘法器的寄存器

结果位于 2 个字寄存器中，分别为结果高字 ResHi 和结果低字 ResLo。和扩展寄存器 SumExt 保存有符号 16 位 x 16 位乘法的符号，或者保存乘加（MAC）结果的溢出。

所有寄存器的最低有效位为 b0，最高有效位为 b7（字节数据）或 b15（字数据）。

6.3 硬件乘法器 SFR 位

由于硬件乘法器的乘法运算十分快速且不需要中断介入，因此未用 SFR 位。

6.4 硬件乘法器的软件限制

在使用硬件乘法器时有两种情况必须引起注意：

- 用间接或间接增量寻址模式处理结果。
- 在中断例程中使用硬件乘法器。

6.4.1 硬件乘法器的软件限制------寻址模式

访问乘法结果的方式有 3 种：变址寻址、间接寻址或间接增量寻址。对于变址寻址模式，包括符号寻址和绝对寻址模式，访问结果寄存器没有任何限制。但是对于用间接寻址和间接增量寻址模式访问结果寄存器，在装入操作数 2 与读取任一结果寄存器的指令之间至少插入一条指令。

```
*******************************************************************************
*   EXAMPLE: MULTIPLY OPERAND1 AND OPERAND 2                                  *
*******************************************************************************
RESLO       .SET        013AH       ; RESLO = ADDRESS OF RESLO
            PUSH        R5          ; R5 WILL HOLD THE ADDRESS OF
            MOV         #RESLO,R5   ; THE RESLO REGISTER
            MOV         &OPER1,&MPY ; LOAD 1ST OPERAND,
                                    ; DEFINES ADD. UNSIGNED MULTIPLY
            MOV         &OPER2,&OP2 ; LOAD 2ND OPERAND AND START
                                    ; MULTIPLICATION
*******************************************************************************
*   EXAMPLE TO ADD THE RESULT OF THE HARDWARE MULTIPLICATION                  *
*   TO THE RAM DATA, 64BITS                                                   *
*******************************************************************************
            NOP                     ; MIN. ONE CYLES BETWEEN MOVING
                                    ; THE OPERAND2 TO HW-MULTIPIER
                                    ; AND PROCESSSING THE RESULT WITH
                                    ; INDIRECT ADDRESS MODE
            ADD         @R5+,&RAM   ; ADD LOW RESULT TO RAM
            ADDC        @R5,&RAM+2  ; ADD HIGH RESULT RO RAM+2
            ADC         &RAM+4      ; ADD CARRY TO EXTENSION WORD
            ADC         &RAM+6      ; IF 64 BIT LENGHT IS USED
            POP         R5
```

上面的例子表明间接寻址或间接增量寻址模式需要较绝对寻址模式更多的时钟周期和代码来将乘法结果传送到目的地址。显然并不需要用间接或间接增量寻址模式来访问绝对地址的硬件乘法器。

6.4.2 硬件乘法器的软件限制------中断程序

整个乘法程序有三步：

- 将第一操作数 OP1 写入硬件乘法器，同时确定了乘法的类型。
- 将第二操作数 OP2 写入硬件乘法器，乘法开始。
- 处理 ResLo、ResHi、SumExt 寄存器中的乘法结果。

如在主程序中使用硬件乘法，考虑以下情况是有用的。当硬件乘法不在主程序而在中断程序中时，由于进入中断服务程序将把通用中断允许位复位，硬件乘法器不会受到后续中断的影响。而一般情况下，为了遵循中断程序尽量简短的原则，乘法及整个数据处理过程最好是在中断程序外完成。

在中断程序中的乘法将对主程序中的乘法程序产生影响：

中断发生在第一操作数 OP1 传入硬件乘法器后

第一操作数 OP1 的地址的 2 位最低有效位决定了乘法的类型，而这个信息无法用任何后来的操作恢复。因此在最初两步即传送第一及第二操作数给乘法器之间，不允许接受中断请求。

中断发生在第二操作数 OP2 传入硬件乘法器后

经过最初两步，乘法结果已经被存放在相应的寄存器 ResLo、ResHi 和 SumExt 中，也可以暂存在堆栈中（PUSH）和在做完另一次乘法后恢复（POP）。但是这样将使中断程序为此增加代码和运行周期。如果在进行乘法操作前关中断（DINT）并在乘法结束后开中断（EINT），就可以避免上述情况的发生。但是这一方法有个缺点，即因为中断事件发生在这段时间内，紧急的中断被抑制的可能大大增加了。

原则建议

一般说来，当主程序中已经用了硬件乘法就要避免在中断程序中使用硬件乘法。专门的应用软件、应用软件库或其它已包含入系统的软件都应考虑在内。各种不同方法的讨论都表明，那样应用的结果负面影响多于正面的。遵循原则建议，使中断程序简短些是最好的实现方法。

7．振荡器与系统时钟发生器

目录 页号

振荡器和系统时钟发生器的主要设计目标是系统廉价和低功耗

为达到系统廉价，外接器件缩减只有一个通用晶振。使用低频晶体和配合倍频器的振荡器满足了系统时钟速度与低功耗两个要求。

小电流应用的特性

通常，极低功耗设备增加各种运行模式强制实现一些功能，例如：启动时序，长期的相对于电压、温度时间的频率稳定性，高度稳定实时时钟时基等。

小电流的实时应用有两个互相矛盾的要求：满足节能的低频系统时钟和为了快速重新激活以响应事件请求的高频系统时钟。尤其对于电池供电的应用系统特别关注电流消耗。在实时应用中为响应外部时间和定时请求，偶尔也需要快速。

理论上，一个能快速启动并允许用于各种不同的功耗模式的处理机时钟发生器可以解决这两难局面。但另一方面，快速启动通常伴随着不可接受的低频率稳定性。设计多个时钟源或为时钟设计各种不同的运行模式可以解决某些外围部件实时应用的时钟要求，如低频通信、LCD 显示、定时器、计数器等。

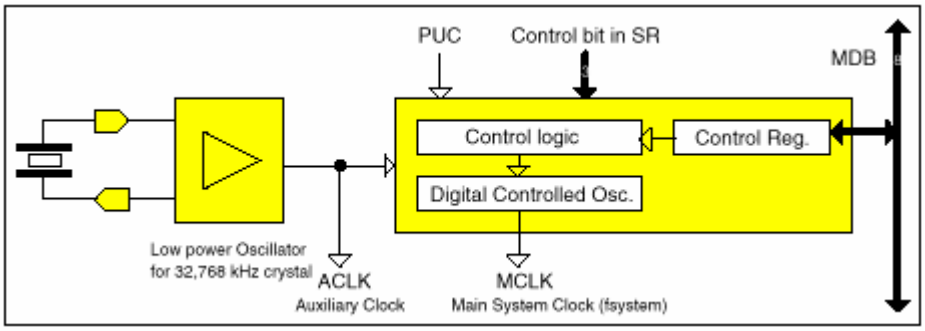


图 7.1：时钟发生器的原理

低频晶振的输出为 CPU 运行和外部模块提供了时钟信号。MSP430 采用被广泛采用的晶振且并不需要其它元件。

从降低电流消耗的观点，CPU 和外部模块的不同要求，需要 2 种时钟信号：

- 由晶振频率提供的辅助时钟信号 ALCK
- 具有更高频率的系统时钟信号 MLCK，频率晶振频率的整倍数。

7.1 晶体振荡器

特殊设计的振荡器满足了低功耗及使用 32768 Hz 晶振的要求，晶振只经两个引脚连接不需要外部元件。所有保证工作稳定的元件或移相电容都集成在芯片中。

两个因素决定了选择熟悉和广泛使用的手表晶振：

- 低功耗的振荡器和时基
- 廉价。

由于状态寄存器 SR 中的 OscOff 为复位，振荡器在上电后即开始工作。对 OscOff 位置位可使它停止。

15 … 9	8	7							0
留作未来增强用	V	SCG1	SCG0	OscOff	CPUOff	GIE	N	Z	C
rw-0	rw-0	Rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

图 7.2：状态寄存器 SR

7.2 处理机时钟发生器

为适应系统和具体应用的不同需要，微控制器的系统时钟必须满足不同的要求：

- 高频率，能对系统硬件请求和事件快速反应
- 低频率，将电流消耗降至最少，EMI，.....
- 对定时应用的频率稳定性，如实时钟 RTC
- 晶振的品质因素（Q 值）低，开始及停止操作没有时间延迟。

以上所提的基本的、互相抵触的要求是无法实现的，或者用高 Q 值高频晶体振荡器，或者用低 Q 值 RC 振荡器。上述的电流消耗和频率稳定要求需要低频晶振。MSP430 的折衷办法是用一个低频晶振，并将其倍频至标称的工作频率范围：

f(系统) = N * f(晶振)

有许多将晶振频率倍频至系统频率的方法，其中的一些是实用的。最熟悉是锁相环技术（PLL）和锁频环技术（FLL）。

频繁而不定时地间断运行模式使 PLL 技术在系统中有两大缺点。PLL 是一个跟随二阶响应的系统。所有的开关操作模式都是由相位失锁引起的，因此需要连续不断作“失步模式”的处理。宽范围的关闭时间状态与闭环电路中的模拟滤波积分器的应用是矛盾的。最终利用相位与/或频率的偏离和频率不合适引起电容电量的自动改变，直至系统重新锁相。

结合数字控制振荡器（DCO）的 FLL 技术避免了这两个严重的问题。

DCO 主要有以下特点：

- 快速启动
- 用数字信号（而非模拟信号）控制。

与这些优点相比，有一点需要仔细考虑：即频率随供电电压、环境温度的变化。

DCO 是完全单调的。

FLL 工作时可看作一个连续频率积分器。一个增/减计数器根据环路控制持续修正倍频系数 N。跟随或更新的速率与晶振频率相一致。采用 32768Hz 晶振时，速率是 30.5uS。

频率的积累误差与晶振相同。相邻机器周期间的时间偏差通常不超过 10%。

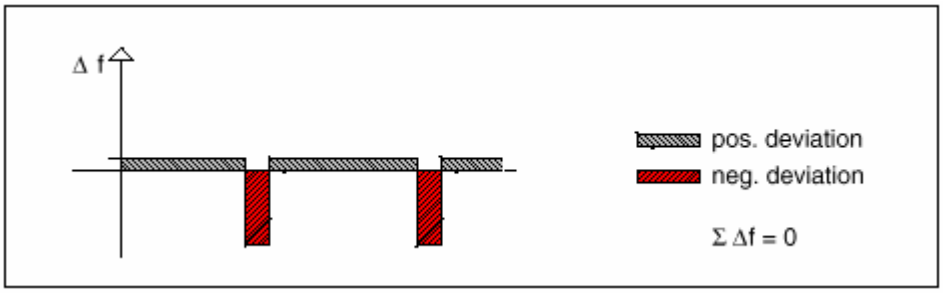


图 7.3：系统频率---时间的关系

系统时钟的启动操作取决于原有的机器状态。在一次 PUC 中，DCO 复位至它的最低频率。在 PUC 状态消除后控制逻辑开始工作，而控制逻辑的正常工作需要稳定的晶振。

10 位频率积分器控制 DCO 的工作频率。在 PUC 之后开始于 0 的频率积分器增计数至所选定的针对系统频率的数值 N。这过程需要稍长于晶振频率的周期数而不是所选择的 10 位数值，当频率积分器需要最大值数时需要 1024 周期。控制逻辑系统的运行是不定期的。

控制器运行在间断情况下的应用要注意系统频率控制的处理条件。频率积分器的修正可能是每一晶振周期（30us @ 32768Hz）只增加f_{System}/N的周期。为了避免时间积累误差，长的积分周期是必须的。

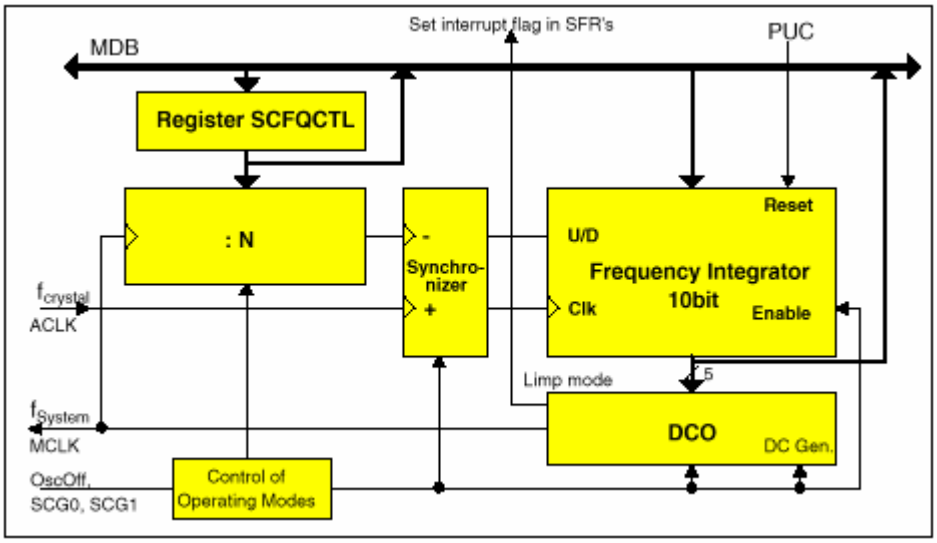


图 7.4：系统频率发生器原理图

如果 DCO 处于它的上限或下限频率时，SFR 中的两个标志位允许应用程序恢复对系统的控制。

运行在频率的上限或下限可以方便地通过访问标志 SCF10 与/或 SCF11 来检测到。

7.3 系统时钟运行模式

系统时钟发生器和晶振是由位于状态寄存器中的 3 个控制信号位来进行控制的。（CG0，SCG1，OcsOff）。这 3 个信号在 4 中不同的上电条件下复位。

这 3 个控制信号使系统工作在不同的运行条件下，对优化整个系统的功耗提供了最大的灵活性。在这 3 个信号的某些组合中系统时钟 MCLK 停止工作，而频率积分器中的已有控制值被保留。

SCG1	SCG0	OscOff	晶振	DC 发生器	DCO	环路控制	说明
0	0	0	ON	ON	ON	ON	PUC 后状态 晶振、DCO 活动 环路控制工作
0	1	0	ON	ON	ON	OFF	省电模式 LPM1 晶振、DCO 活动 环路控制停止
1	0	0	ON	ON	OFF	OFF	省电模式 LPM2 晶振、DC 发生器活动 DCO、环路控制停止
1	1	0	ON	OFF	OFF	OFF	省电模式 LPM3 晶振活动 其余功能停止
X	X	1	OFF	OFF	OFF	OFF	省电模式 LPM4 全部功能停止 $f_{MCLK} = f_{ACLK} = 0Hz$

这3个控制信号提供了5种不同功耗的工作模式，充分利用它们能支持超低功耗的应用。所有这些不同的模式为系统应用提供了这种可能性，即可能工作在最小的时间片内并在每一时间片内优化电流消耗。

SCG0 位控制 FLL 环，为 0 时，FLL 环工作，为 1 时，FLL 环关闭。

从 PUC 启动

系统时钟控制寄存器 SCFQCTL 由 PUC 置为 01Fh。同时频率积分器复位。这使得系统频率被置为最低值，然后计数持续增加直至锁定在系统频率上，这一频率等于晶振频率的 N 倍。

低功耗模式 LPM4（晶振关闭）

在振荡器关闭模式期间处理机的所有部件均停止工作，电流消耗在最小限度。只有在系统上电电路检测到低电平或任一外部中断事件请求异步响应的中断时才会开始工作。在此程序流程期间应当适当允许可能用到的中断。

系统时钟发生器脱离振荡器关闭模式的启动顺序为：

- 由频率积分器输出数值和 DCO 特征值定义的现有系统频率继续产生。
- 在晶振尚未开始工作时，频率积分器连续以f_{System}/N的频率连续减计数直至DCO工作在它的最低频率。
- 晶振开始工作后，环路控制将频率积分器调整到跟随$f_{system} = N * f_{crystal}$的数值。

低功耗模式 LPM3（DC 发生器关闭）

在DC发生器关闭模式期间只有晶振活动。设置基本时序条件的DC发生器的DC电流关闭。因高阻设计功耗被抑制。从DC发生器关闭的省电模式启动DCO要化一段时间（t_{DCGon}）才能工作在选择的频率下。它的范围在ns至us之间。

低功耗模式 LPM2（DCO 关闭）

在 LPM2 模式期间晶振和 DC 发生器仍然活动，因此可以快速启动。启动的延时延迟限于几个门延时。

低功耗模式 LPM1（FLL 关闭）

在LPM1 模式期间晶振、DC发生器及DCO仍然活动。处理机及它的全部外围模块拥有全部功能而无任何限制。工作频率取决于频率积分器的输出数值。这一数值和DCO的特征值共同决定了MCLK信号的频率，它与系统频率f_{system}相同。

因为振荡器已经工作，启动没有时间延迟。环路控制异步激活并伴有轻微的频率偏移，但是它的调节是快速的和不定期的。

7.4 系统时钟控制寄存器

系统时钟发生器通过 3 个通用模块寄存器和 SFR 与处理机的其它部分相关联。通用模块寄存器位于较低的安排字节模块的外围模块地址。3 个工作状态控制线 SCG0、SCG1 和 OscOFF 由 CPU 的状态寄存器 SR 提供。

7.4.1 模块寄存器

两个8位的寄存器控制系统时钟发生器。由用户软件将乘法系数N存入其中一个寄存器。另一个存放控制位用于各种工作模式的信号。它只能用字节指令访问。

系统时钟频率控制

SCFQCTL（052h）

7							0
M	2^6	2^5	2^4	2^3	2^2	2^1	2^0
rw-0	rw-0	rw-0	rw-1	rw-1	rw-1	rw-1	rw-1

寄存器 SCFQCTL 控制晶振频率的倍数。由 7 位指明范围 3+1 到 127+1。

$$f_{System} = (x*2^6+x*2^5+x*2^4+x*2^3+x*2^2+x*2^1+x*2^0+1) * f_{crystal}$$

寄存器 SCFQCTL 在 PUC 后的缺省值是 31，它产生的因子为 32。

f_{system}的范围是理论上的，它依赖于DCO的频率可调整范围（详细资料见电气特性）。

注意：SCG 的倍频因子

寄存器SCFQCTL（2^6至 2^0）控制晶振频率的倍数。7 位的值必须在 3 到 127 范围内。任何低于 3 的值会引起不可预知的结果，而超过 127 的值会强制MCLK的频率超过器件的性能指标。

系统时钟频率积分器

SCFI0（050h）

7							0
0	0	0	FN_4	FN_3	FN_2	2^1	2^0
r	r	r	rw-0	rw-0	rw-0	rw-0	rw-0

SCFI1（051h）

7							0
2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

系统时钟频率积分器的输出控制 DCO，值可由 SCF11 和 SCF10 读出。表达式为：

$$N_{DCO} = (x*2^9+x*2^8+x*2^7+x*2^6+x*2^5)+(1-M)*(x*2^4+x*2^3+x*2^2+x*2^1+x*2^0)$$

SCF10 中的bit1...3: 这 3 位定义DCO的标称频率(f_{NOM})。

FN_4	FN_3	FN_2	频率
0	0	9	f_{NOM}
0	0	1	$2xf_{NOM}$
0	1	X	$3xf_{NOM}$
1	X	X	$4xf_{NOM}$

7.4.2 与系统时钟发生器相关的 SFR 位

SFR 中的 2 位根据系统时钟发生器执行的功能来控制系统的相互控制。这 2 位是：

- OscFault 中断标志 OFIFG（位于 IFG1.1，初始状态不变）
- OscFault 中断允许 OFIFE（位于 IE1.1，初始状态为复位）。

中断标志位是多中断源请求机制的一部分。如选择 NMI 功能，同一个中断向量也用于 RST*/NMI 引脚的事件。这一中断是不可屏蔽的，这意味着通用中断允许位 GIE 不能禁止这一中断请求。由于共用同一个中断向量，且在 PUC 之后振荡器出错信号为有效，中断标志位不会自动复位。

有 3 种情况需要软件来处理：

- 在 PUC 之后，必须编制一段程序来识别或设置振荡器状态以因防止 OscFault 信号的有效而将 OFIFG 永久置位。OFIFG 必须由软件复位。PUC 将 OFIE 复位，因此不会发生中断请求。
- 当从 OscFault 来的中断请求被服务，中断允许位 OFIE 自动复位以阻止继续到来的中断请求，直至软件作出适当的响应使 OscFault 信号无效。在达到这一状态后，OFIE 可以按模块中断的一般规则重现置位。由振荡器出错事件不受通用中断允许位的影响。
- 中断标志 OFIFG 可用于在中断服务程序的开始识别中断源。OFIFG 的置位与 NMI 事件无关。它的作用是支配性的。

注意：中断标志 OFIFG

当中断请求被接受并得到服务，中断标志 OFIFG 仍保持置位。因为它是与 NMI 中断一起构成多源中断并要指示软件处理振荡器失效事件，所以是强制性的。首先服务 OFIFG 的条件使得这一事件的优先级比 NMI 事件高。

7.5 DCO 典型特性

DCO 随供电电压和温度而变化。在锁频环工作情况下这对于应用并不重要。因为控制周期与 ACLK 的周期相同。用 32768Hz 晶振时仅为 30.5uS。

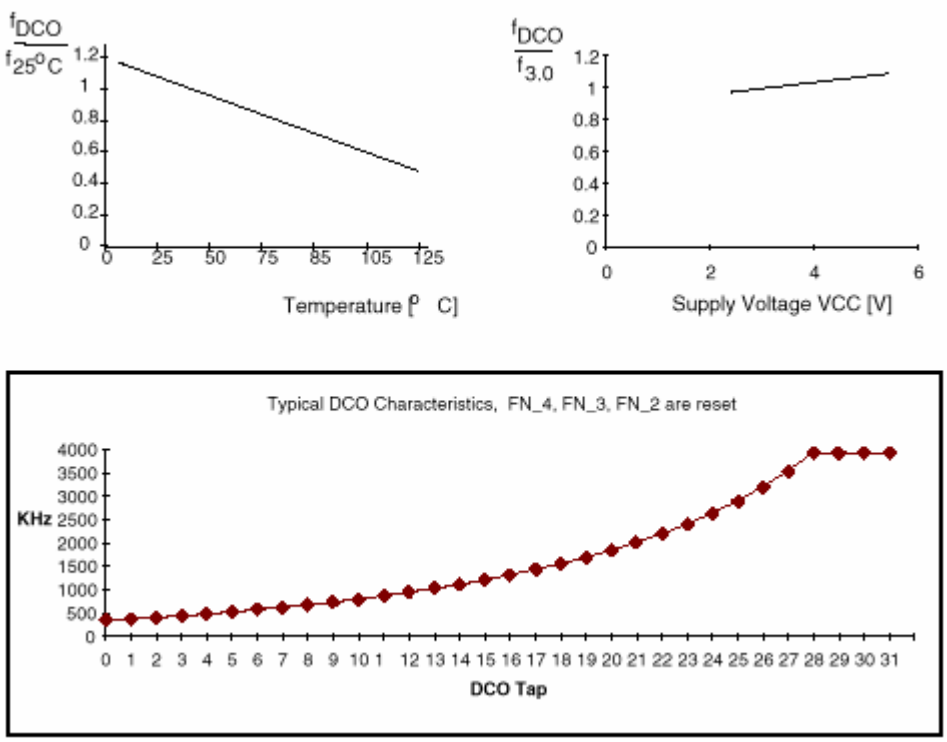


图 7.5：DCO 的特征值

注意：DCO Taps
系统时钟频率积分器寄存器 SCFI1 的高 5 位送入 DCO。如果寄存器 SCFQCTL 中的调制位 M 置位，系统频率仅由 DCO taps 来确定。

8．数字 I/O 配置

目录 页号

8.1 通用端口 P0

通用端口 P0 及其所有功能可以按引脚单独选择，并且每个信号可以作为一个中断源。有 6 个寄存器用于端口 I/O 引脚的控制。

通用模块寄存器位于较低的安排字节模块的外围模块地址。这些寄存器必须由字节指令，用绝对寻址模式访问。

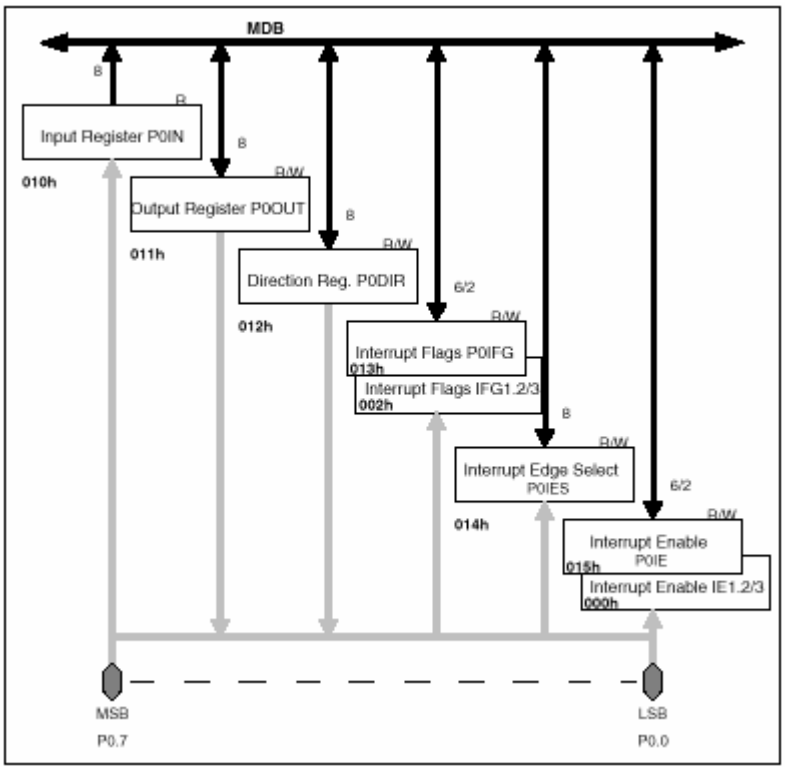


图 8.1：端口 P0 的结构

8.1.1 P0 的控制寄存器

端口 P0 通过以 8 位构成的 MDB 及 MAB 和处理机内核相连，必须通过字节指令访问。

它的 6 个控制寄存器为数字 I/O 功能提供了最大的灵活性：

- 每个 I/O 位均可独立编程。
- 可以任意组合输入、输出和中断条件。
- P0 的 8 个引脚均可实现外部事件的中断处理。

这 6 个寄存器是：

寄 存 器	缩写	寄存器类型	地址	初始状态
输入寄存器	P0IN	只读	010h	
输出寄存器	P0OUT	读写	011h	不变
方向寄存器	P0DIR	读写	012h	复位
中断标志	P0IFG	读写	013h	复位
中断触发沿选择	P0IES	读写	014h	不变
中断允许	P0IE	读写	015h	复位

除了中断标志寄存器和中断允许寄存器的最低 2 位外，其它寄存器均为 8 位。这 2 位包含在 SFR 中。这些寄存器必须用字节指令访问。

输入寄存器 P0IN

输入寄存器是扫描 I/O 引脚信号的只读寄存器。引脚的方向必须选定为输入。

注意：对只读寄存器 P0IN 写入
对这一只读寄存器写入，会在写操作有效期间增加电流消耗。

输出寄存器 P0OUT

输出寄存器是输出信息的缓存，由一个 8 位寄存器保存用作输出的 I/O 引脚的信息。输出缓存可以用所有写指令修改。读出时可得到寄存器的内容而与是否定义为输出无关。引脚方向定义的改变也不会改变输出缓存的内容。

方向寄存器 P0DIR

方向寄存器有互相独立的 8 位，分别定义了 I/O 引脚的方向。各位在 PUC 后被复位。

0： I/O 引脚切换成输入模式

1： I/O 引脚切换成输出模式

中断标志寄存器 P0IFG

中断标志寄存器有 6 个标志位，包含是否有待处理中断和是否对应相应引脚的信息。

P0IFG（013h）

7							0
P0IFG.7	P0IFG.6	P0IFG.5	P0IFG.4	P0IFG.3	P0IFG.2		
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r0	r0

0： 没有挂起中断。

1： 由于 I/O 引脚电平跳变引起待处理中断。

对寄存器 P0OUT 和 P0DIR 的操作也会使 P0IFG 置位。

对中断标志寄存器写入 0，能使它复位。

定位于 P0IFG.2 至 P0IFG.7 的 6 个标志对应于引脚 P0.2 至 P0.7。余下的 P0.1 和 P0.0 的中断标志位于 SFR 中。

注意：中断标志 P0FLG2 至 7

中断标志 P0IFG.2 至 P0IFG.7 只用了一个中断向量，是多源中断。当任一事件引起的中断被服务时 P0IFG.2 至 P0IFG.7 不会自动复位。必须用软件来判定是对哪一个事件服务并将相应的标志复位。

外部中断事件必须保持大于 1.5 倍 MCLK 的时间或足以保证相应中断标志位置位的时间。

中断触发沿选择寄存器 P0IES

对应每个 I/O 引脚，中断触发沿选择寄存器都有一位选择哪一种电平跳变触发中断标志。每一位有以下意义：

0：电平由低到高跳变使标志位置位

1：电平由高到低跳变使标志位置位

注意：改变 P0IES 位

P0IES 中的各位的任何改变都可能使相应的中断标志位置位。

中断允许寄存器 P0IE

在寄存器 P0IE 中，6 个 I/O 引脚 P0.7 至 P0.2 都有一位用以允许因中断事件产生中断请求。P0.0 和 P0.1 的中断允许位位于 SFR IE1.2 和 IE1.3 中。

P0IE（015h）

7							0
P0IE.7	P0IE.6	P0IE.5	P0IE.4	P0IE.3	P0IE.2		
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r0	r0

各位的意义如下：

0：禁止中断。

1：允许中断。

注意：P0 的中断敏感性

只有跳变，而不是静态电平，才能引起中断请求。

中断程序必须对多用途中断标志 P0IFG2 至 7 复位。而单源标志 P0IFG.0 和 P0IFG.1 会在被服务时自动复位。

如果在执行 RETI 指令后中断标志仍然置位（因为在中断服务期间又发生跳变），在 RETI 指令完成后中断将再次发生。这一点保证了每次跳变都能被软件发现。

8.1.2 P0 的原理

端口 P0 的 P0.3 至 P0.7

端口 P0 的引脚逻辑是由 5 个寄存器（P0DIR、P0OUT、P0IFG、P0IE、P0IES 和只读的 P0IN）构成。P0.3 至 P0.7 在设计上完全相同。

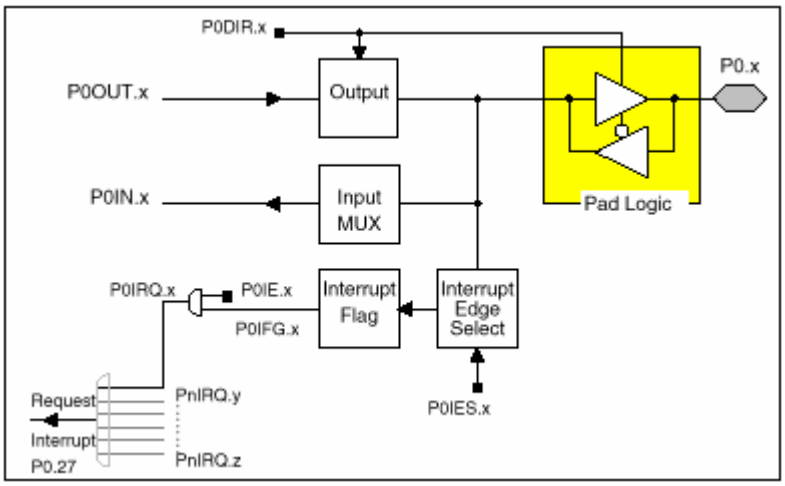


图 8.2：P0.3 至 P0.7 原理图

中断标志位可由适当的输入条件置位，也可以由软件置位。另外，修改方向控制位或中断触发沿选择位也可能发生触发条件。

P0 的 P0.2 至 P0.7 共用一个公共的中断向量。当 P0.2 至 7 中断被接受时中断标志位不会自动复位。P0IFG.2 至 P0IFG.7 的各标志位必须用软件在相应的中断服务程序中复位。

端口 P0 的 P0.2

P0.2 与 P0.3 至 7 稍有不同。它的输出信号既可由 P0OUT.2 也可由定时器/计数器的信号 TXD 决定。当输出控制寄存器的输出 TXE 置位时，TXD 信号被选中成为输出信号，同时 Pad 逻辑切换成输出，这时引脚与方向寄存器 P0DIR.2 无关。

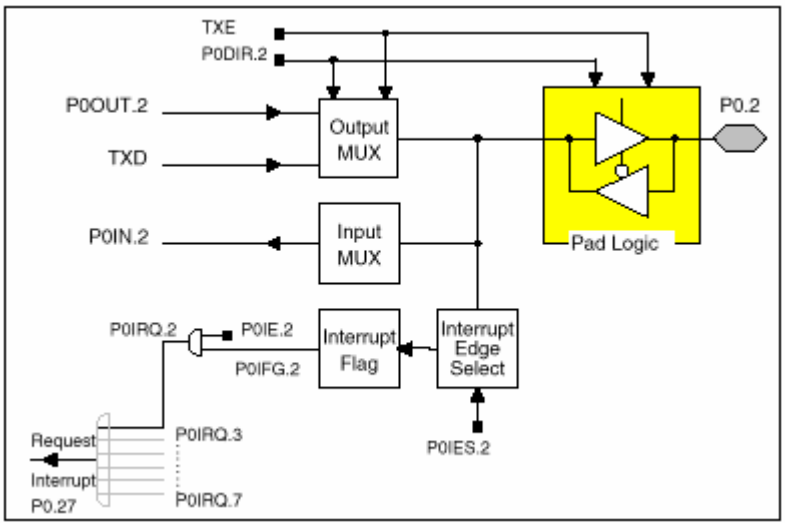


图 8.3：端口 P0.2 的原理图

中断标志 P0IFG.2 与中断标志 P0IFG.3 至 7 共享中断向量。

端口 P0.1

P0.1 与 P0.3 至 7 稍有不同。这个引脚的中断源可以是端口 P0.1 的输入，也可以是定时器/计数器的进位。当在定时器/计数器控制寄存器 TCCTL 中的中断源控制位 ISCTL 置位，中断源便从端口 P0.1 的输入信号切换至定时器/计数器的进位信号。

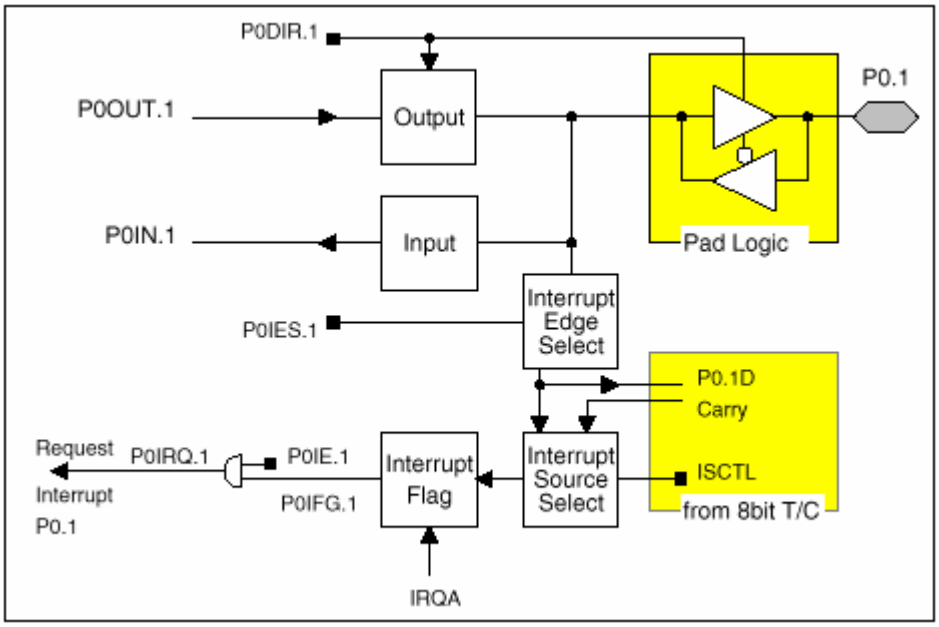


图 8.4：端口 P0.1 原理图

中断标志位 P0IFG.1 在中断请求被接受时（IRQA）自动复位。

端口 P0.0

P0.0 和端口 P0.3 至 P0.7 一样，但是用单独的中断向量。

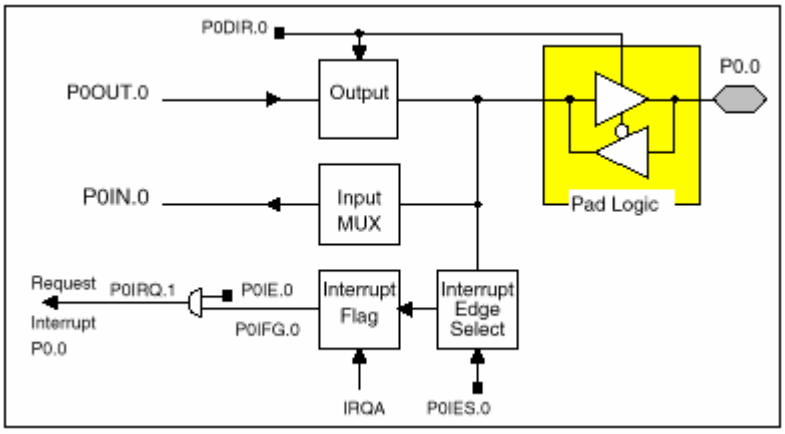


图 8.5：端口 P0.0 原理图

中断标志位 P0IFG.0 在中断请求被接受时（IRQA）自动复位。

8.1.3 P0 的中断控制功能

端口 P0 有 8 位中断标志位;8 位中断允许位;8 位中断事件触发沿选择和 3 个中断向量。

3 个中断向量分别分配给：

- P0.0
- P0.1/RXD
- P0.2 至 P0.7

P0 中的 2 个信号 P0.0 和 P0.1/RXD 用于特定的信号处理。有 SFR 中的 4 位和 P0 地址帧（Address Frame）的 2 位用来处理中断事件：

- P0.0 的中断标志位 P0IFG.0（位于 IFG1.2 中，初始状态为复位）
- P0.1/RXD 中断标志位 P0IFG.1（位于 IFG1.3 中，初始状态为复位）
- P0.0 中断允许位 P0IE.0（位于 IE1.2 中，初始状态为复位）
- P0.1/RXD 中断允许位 P0IE.1（位于 IE1.3 中，初始状态为复位）
- P0.0 中断触发沿选择位（位于 P0IES.0，初始状态为复位）
- P0.1/RXD 中断触发沿选择位（位于 P0IES.1，初始状态为复位）

两个中断标志都为单源中断并且都会在中断服务后自动复位。中断允许位和中断触发沿选择位保持不变。

余下的 6 个 I/O 信号 P0.2 至 P0.7 的中断控制位位于 I/O 地址帧。每一个信号都有 3 位用以定义对中断事件的反应：

- 中断标志位（P0IFG.2 至 P0IFG.7）
- 中断允许位（P0IE.2 至 P0IE.7）
- 中断触发沿选择位（P0IES.2 至 P0IES.7）

中断标志位 P0IFG.2 至 P0IFG.7 是多源中断的一部分。当满足两个条件：P0IE.x（$2 \le x \le 7$）置位且通用中断允许位 GIE 置位，P0.2 至 P0.7 中的一个或多个中断事件将产生中断请求。它们共用一个中断向量。因此中断标志位不会在中断服务后自动复位。

中断服务程序软件处理中断源检测和将中断并将相应的中断标志位复位。

注意：多源中断标志 P0IFG.2 至 P0IFG.7

多源中断标志 P0IFG.2 至 P0IFG.7 在中断请求被接受和得到服务时保持置位。因为是多源中断，这是必须的。每个标志必须在它自己的中断服务程序中复位。

8.2 通用端口 P1、P2

通用端口 P1 和 P2 将各个引脚选择的功能来组成全部功能，并且每一个信号都可用作中断源。

各有 7 个寄存器来控制端口的引脚。

通用模块寄存器位于安排字节外围模块的低端地址，寄存器必须用字节指令以绝对寻址模式进行访问。

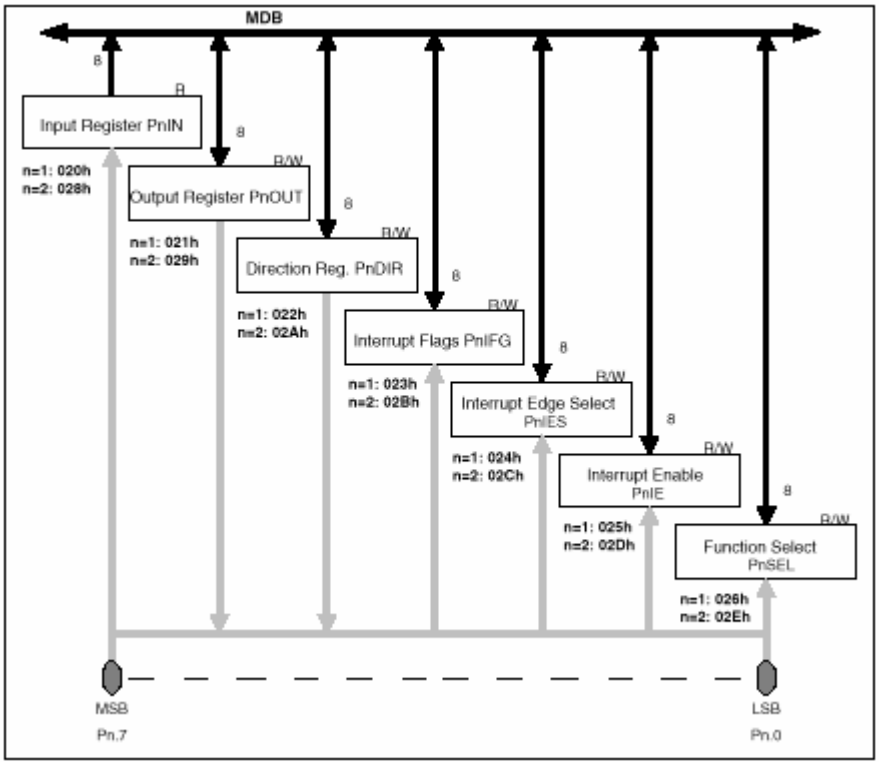


图 8.6：端口 P1、P2 的结构

8.2.1 P1 和 P2 的控制寄存器

端口 P1 和 P2 通过 8 位 MDB 及 MAB 和处理机内核相连，它们必须通过字节指令来访问。7 个控制寄存器为数字 I/O 功能的应用提供了最大的灵活性：

- 所有 I/O 位均可独立编程。
- 可能有各种输入、输出和中断条件的组合。
- P1 和 P2 的 8 位实现了对外部事件的中断处理。

P1 和 P2 的 7 个寄存器分别是：

寄存器	缩写	寄存器类型	地址	初始状态
输入寄存器	P1IN	只读	020h	
输出寄存器	P1OUT	读写	021h	不变
方向寄存器	P1DIR	读写	022h	复位
中断标志位	P1IFG	读写	023h	复位
中断触发沿选择	P1IES	读写	024h	不变
中断允许	P1IE	读写	025h	复位
功能选择寄存器	P1SEL	读写	026h	复位
输入寄存器	P2IN	只读	028h	
输出寄存器	P2OUT	读写	029h	不变
方向寄存器	P2DIR	读写	02Ah	复位
中断标志位	P2IFG	读写	02Bh	复位
中断触发沿选择	P2IES	读写	02Ch	不变
中断允许	P2IE	读写	02Dh	复位
功能选择寄存器	P2SEL	读写	02Eh	复位

以上所有寄存器为 8 位。必须用字节指令以绝对寻址模式进行访问。

输入寄存器 P1IN、P2IN

这两个只读寄存器扫描 I/O 引脚上的信号。这些引脚应选择为输入方向。

注意：对输入寄存器 P1IN、P2IN 写入
对只读寄存器的写入会在写有效期间增加电流消耗。

输出寄存器 P1OUT、P2OUT

如果 I/O 引脚用作输出，每个 8 位寄存器提供 8 位输出缓存信息。输出缓存可用所有有目的操作数的指令修改。在读取时，输出缓存的内容和引脚定义方向无关。

方向寄存器 P1DIR 和 P2DIR

每个寄存器含有相互独立的 8 位，用于定义 I/O 引脚的方向。各位在 PUC 后复位。

0：I/O 引脚切换成输入模式
1：I/O 引脚切换成输出模式

中断标志寄存器 P1IFG 和 P2IFG

每个寄存器有对应于 I/O 引脚的 8 个标志位，表示是否有中断挂起。

0：没有中断挂起

1：由于 I/O 引脚电平跳变引起中断挂起

对寄存器 P1OUT 和 P1DIR，以及 P2OUT 和 P2DIR 也能引起 P1IFG 和 P2IFG 置位。对某一个中断标志写 0 使它复位。

注意：中断标志 P1IFG0 至 7、P2IFG0 至 7

每一组中断标志 P1IFG.0 至 P1IFG.7 和 P2IFG.0 至 P2IFG.7 只用一个中断向量，它们都是多源中断向量。当中断服务时这些标志位不会自动复位。由中断服务程序确定服务的事件，并将相应的标志复位。

任何外部中断事件必须长于 1.5 倍 MCLK 的时间，以保证被接受并使相应的中断标志置位。

中断触发沿选择寄存器 P1IES 和 P2IES

寄存器中对应于每一个 I/O 引脚 P1.0 至 P1.7 和 P2.0 至 P2.7 都有一位选择触发中断标志的电平跳变。意义如下：

0：电平由低到高使标志置位

1：电平由高到低使标志置位

注意：P1IES、P2IES 各位的改变

改变 P1IES 或 P2IES 中的位可能会引起相应中断标志位置位。

PnIES.x	PnIN.x	PnIFG.x
0 → 1	0	不变
0 → 1	1	可能置位
1 → 0	0	可能置位
1 → 0	1	不变

中断允许寄存器 P1IE 和 P2IE

每一个 I/O 引脚 P1.0 至 P1.7 和 P2.0 至 P2.7 都在寄存器 P1IE 和 P2IE 中有一相应的允许位，用以允许中断事件。

允许位的意义：

0：禁止中断请求

1：允许中断请求

注意：P1、P2 的中断敏感性

只有跳变，而不是静态电平，能够引起中断。

由于它们是多源中断，中断程序必须清除全部中断标志。

如果在执行 RETI 指令后中断标志仍然置位（因为在中断服务期间又发生跳变），在 RETI 指令完成后中断将再次发生。这一点保证了每次跳变都能被软件发现。

功能选择寄存器 P1SEL 和 P2SEL

每个寄存器都有相互独立的 8 位来定义 I/O 引脚的功能。端口功能或者定义为模块功能或者定义为数据输出到引脚和从引脚输入数据。各位在 PUC 后复位：

0：端口功能，引脚是端口的输出或输入数据

1：模块功能，引脚是模块的输出或输入数据。

注意：用 P1SEL、P2SEL 选择功能
如 PnSEL.x 置位，中断触发沿选择电路被禁止。输入信号将不能改变中断标志。
当功能选择位 PnSEL 复位，中断触发沿选择和中断标志位具有完全特性。

8.2.2 P1、P2 的原理

端口 P1 和 P2 的各引脚的逻辑是相同的，都可读写。

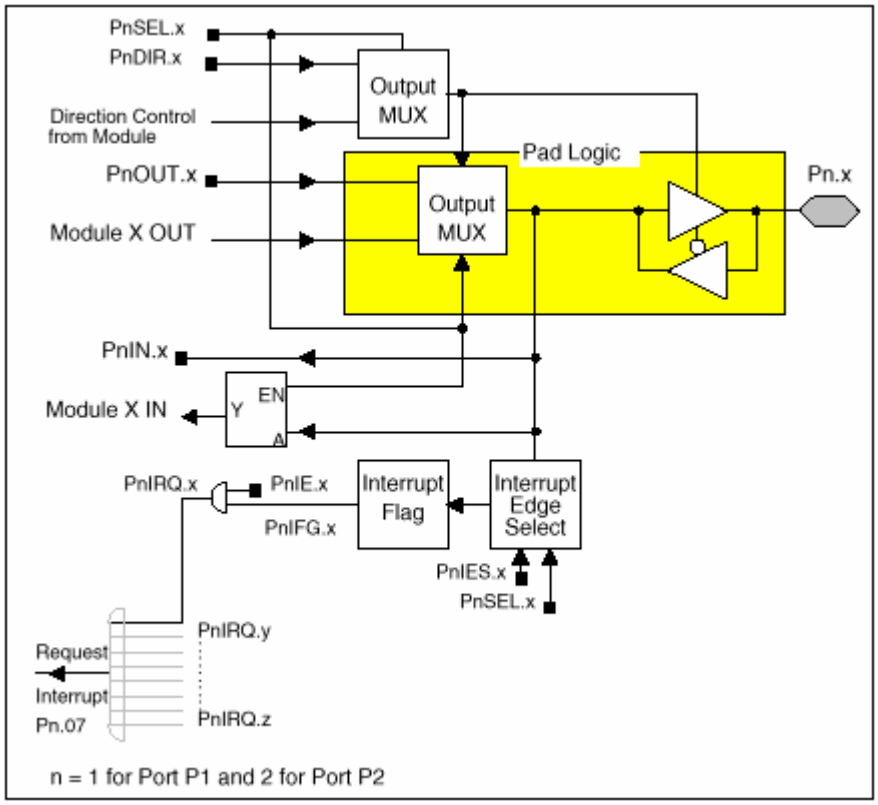


图 8.7：端口 P1、P2 的原理图

模块 X 输入功能

当 PnSEL.x 置位选择了模块功能，输入信号传入引脚对应的外围模块。当 PnSEL.x 复位，输入模块的信号将保持最后输入信号的电平。将复位的控制位置位，如果保持的电平与引脚电平同会改变输入模块的信号。

8.2.3 P1、P2 的中断控制功能

端口 P1 和 P2 有 8 位中断标志位；8 位中断允许位；8 位中断事件触发沿选择，并各有一个中断向量。

所有中断控制位位于 I/O 地址帧，每个中断信号各用 3 位来定义对中断事件的响应：

- 中断标志位：P1IFG.0 至 P1IFG.7 和 P2IFG.0 至 P2IFG.7
- 中断允许位：P1IE.0 至 P1IE.7 和 P2IE.0 至 P2IE.7
- 中断触发沿选择位：P1IES.0 至 P1IES.7 和 P2IES.0 至 P2IES.7

中断标志位 P1IFG.0 至 P1IFG.7 和 P2IFG.0 至 P2IFG.7 是多源中断的一部分。当满足两个条件：PnIE.x（0 ≤ x ≤ 7）置位且通用中断允许位 GIE 置位，P1.0 至 P1.7 或 P2.0 至 P2.7 中的一个或多个中断事件将产生中断请求。它们共用一个中断向量。因此中断标志位不会在中断服务后自动复位。

中断服务程序软件处理中断源检测和将中断并将相应的中断标志位复位。

注意：多源中断标志 P1IFG.0 至 P1IFG.7、P2IFG.0 至 P2.7
多源中断标志 P1IFG.0 至 P1IFG.7 和 P2IFG.0 至 P2IFG.7 在中断请求被接受和得到服务时保持置位。因为是多源中断，这是必须的。每个标志必须在它自己的中断服务程序中复位。

8.3 通用端口 P3 和 P4

通用端口 P3 和 P4 相同，由各个引脚选择的功能来组成全部功能。每个引脚可以选择按端口特性工作或在内部的外围模块控制下工作。

各有 4 个寄存器控制 P3 和 P4。

通用模块寄存器位于安排字节外围模块的低端地址，寄存器必须用字节指令以绝对寻址模式进行访问。

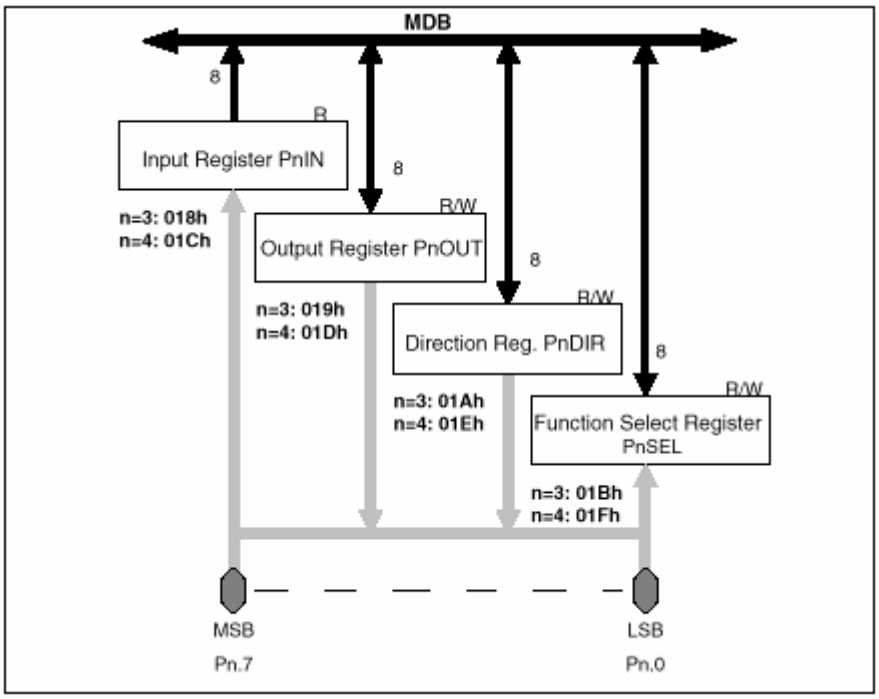


图 8.8：端口 P3、P4 的结构

8.3.1 端口 P3 和 P4 的控制寄存器

端口 P3 和 P4 通过 8 位 MDB 及 MAB 和处理机内核相连，它们必须通过字节指令以绝对寻址模式访问。

4 个控制寄存器为数字 I/O 功能的应用提供了最大的灵活性：

- 所有 I/O 位均可独立编程。
- 可能有各种输入组合。
- 可能有各种端口和模块功能的组合。

每个端口的 4 个寄存器是：

寄存器	缩写	寄存器类型	地址	初始状态
输入寄存器	P3IN	只读	018h	
输出寄存器	P3OUT	读写	019h	不变
方向寄存器	P3DIR	读写	01Ah	复位
端口选择寄存器	P3SEL	读写	01Bh	复位
输入寄存器	P4IN	只读	01Ch	
输出寄存器	P4OUT	读写	01Dh	不变
方向寄存器	P4DIR	读写	01Eh	复位
端口选择寄存器	P4SEL	读写	01Fh	复位

以上所有寄存器为 8 位。必须用字节指令访问。

输入寄存器 P3IN、P4IN

这两个只读寄存器扫描 I/O 引脚上的信号。这些引脚应选择为输入方向。

注意：对输入寄存器 P3IN、P4IN 写入
对只读寄存器的写入会在写有效期间增加电流消耗。

输出寄存器 P3OUT、P4OUT

如果 I/O 引脚用作输出，每个 8 位寄存器提供 8 位输出缓存信息。输出缓存可用所有有目的操作数的指令修改。在读取时，输出缓存的内容和引脚定义方向无关。改变方向不会改变输出缓存的内容。

方向寄存器 P3DIR 和 P4DIR

每个寄存器含有相互独立的 8 位，用于定义 I/O 引脚的方向。各位在 PUC 后复位。

0：I/O 引脚切换成输入模式

1：I/O 引脚切换成输出模式

功能选择寄存器 P3SEL 和 P4SEL

每个寄存器都有相互独立的 8 位来定义 I/O 引脚的功能。端口功能或者定义为模块功能或者定义为数据输出到引脚和从引脚输入数据。各位在 PUC 后复位：

0：端口功能，引脚是端口的输出或输入数据

1：模块功能，引脚是模块的输出或输入数据。

8.3.2 端口 P3 和 P4 的原理图

端口 P3 和 P4 各引脚的逻辑是在特殊设备配置中定义的。在这些设备规定中定义了纯数字端口或兼有数字端口和模块功能的引脚功能。

只作为数字端口的引脚只受相应的端口寄存器位来控制。

兼有数字端口和模块功能的引脚由以下控制位控制：

- 端口控制位，当相应的选择位 PnSEL.x 复位
- 模块控制位，当相应的选择位 PnSEL.x 置位

全部 8 个端口可由硬件分别配置成以下状态：

- 端口引脚只读
- 端口引脚只实现模块功能
- 由软件配置端口或模块功能

具体的实现在器件数据手册中介绍。

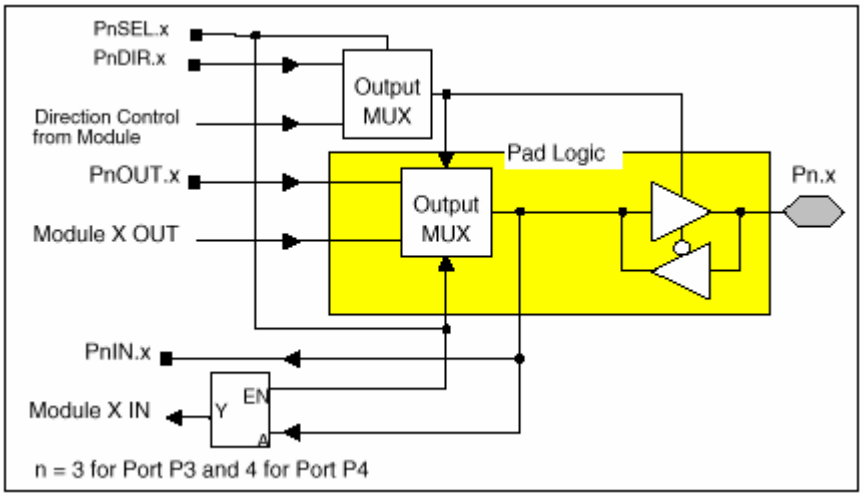


图 8.9：端口 P3.x、P4.x 的原理图

模块 X 输入功能

当 PnSEL.x 置位选择了模块功能，输入信号传入引脚对应的外围模块。当 PnSEL.x 复位，输入模块的信号将保持最后输入信号的电平。将复位的控制位置位，如果保持的电平与引脚电平同会改变输入模块的信号。

8.4 LCD 端口

LCD 端口可选择作为 LCD 显示驱动或作为提供静态信号的数字输出。LCD 的控制用公共端（COM）和段码端（SEG）输出台阶信号来驱动以 2MUX 或更高速率复合扫描所需的模拟信号。

LCD 输出

当 LCD 端口用作驱动 LCD 显示，由门电路将模拟电压送至输出引脚。LCD 输出也可由软件配置作为数字输出。

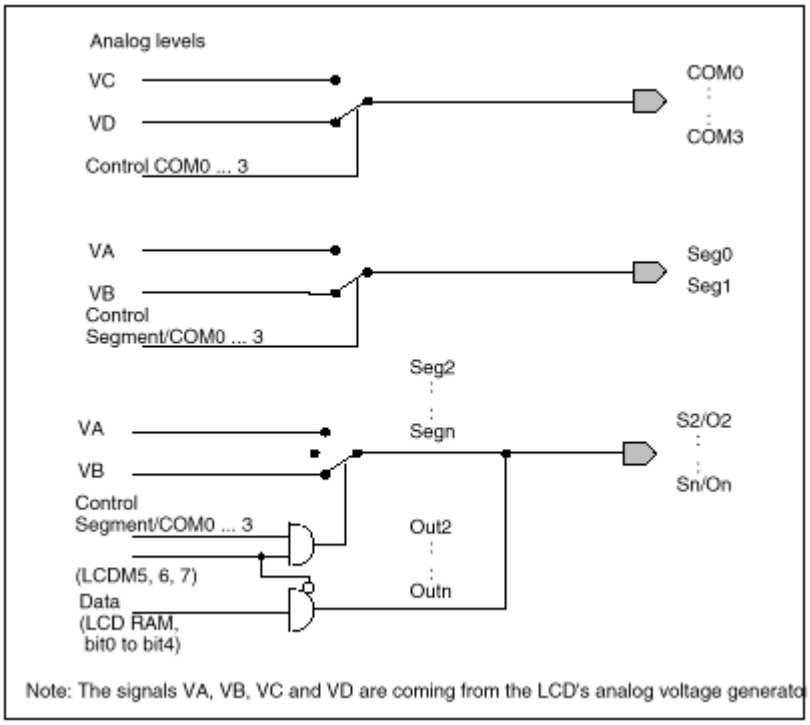


图 8.10：LCD 原理图

LCD 控制寄存器的 3 位 LCDM5、LCDM6 和 LCDM7 控制所有功能。输出信号控制的详细内容可见 LCD 章节。

8.5 LCD 端口，定时器/端口比较器

和定时器/端口模块配合的比较器和 LCD 端口共用一根 SEG 线。在 PUC 后引脚选择为 SEG

线功能。当定时器/端口模块中的 CPON 位置位后引脚选择为比较器输入。在 PUC 引起复位前一直保持置位。

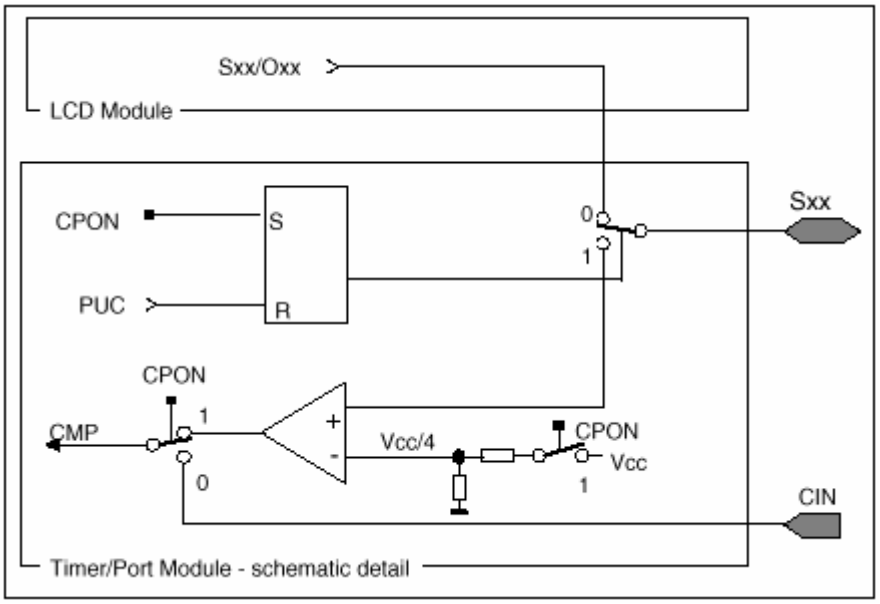


图 8.11：LCD 引脚原理图，定时器/端口比较器

9．通用定时器/端口模块

目录 页号

通用定时器/端口（Timer/Port）模块支持多种系统功能：

- 多达 6 个独立输出
- 2 个 8 位计数器，可级联成 16 位模式
- 用斜率转换原理的 A/D 转换器的精密比较器

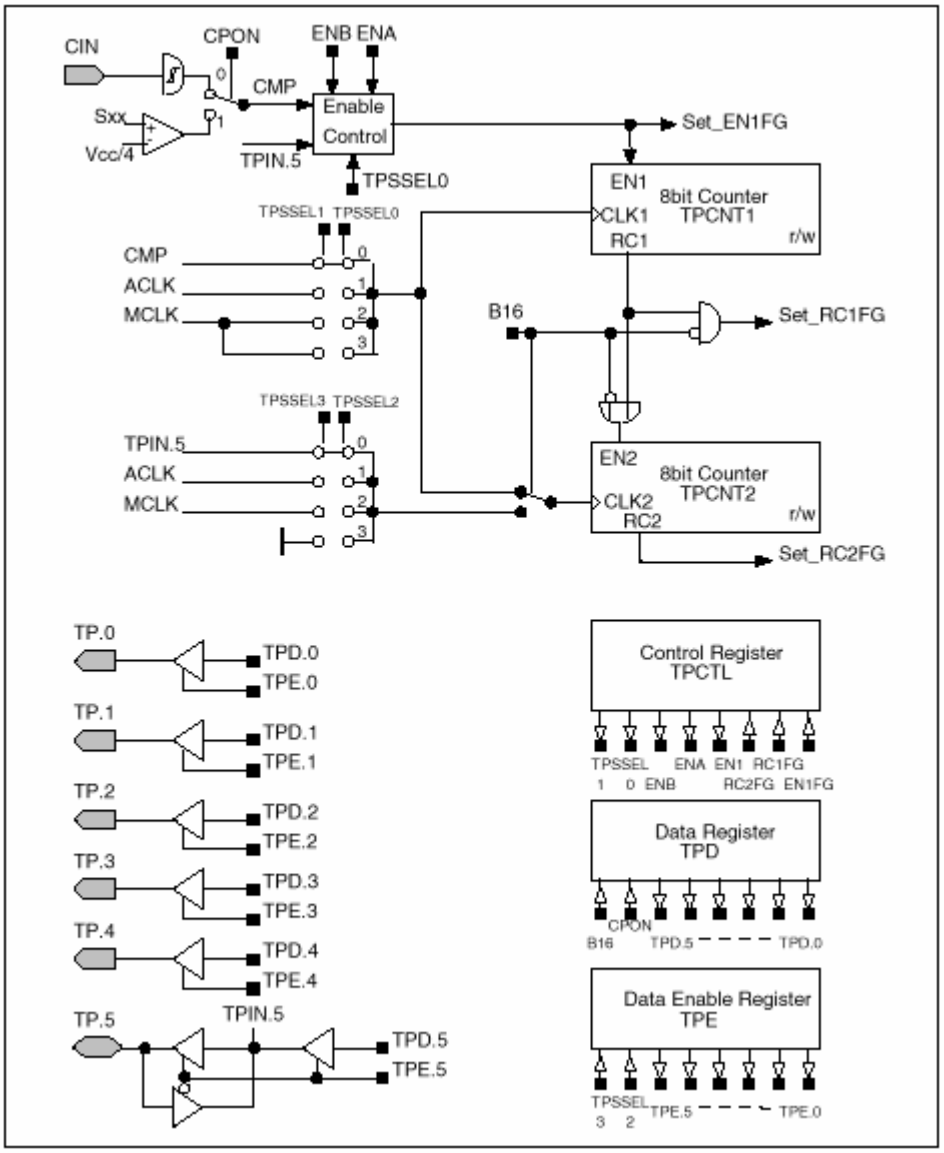


图 9.1：定时器/端口(Timer/Port)的结构

9.1 定时器/端口的操作

定时器/端口模块可由控制寄存器 TPCNTL 来配置，以工作在不同模式中。

9.1.1 定时器/端口计数器 TPCNT1，8 位操作

计数器 TPCNT1 可用适当的指令读写。当 CMP 或 ACLK 选中时，定时器数据读操作可与时钟异步。用软件作 2 到 3 次采样并作多数表决可保证数据的准确性。

当时钟源是 MCLK 时读出数据总是正确的，因为 MCLK 同样也是指令的时钟频率。读出数据仅是计数器状态的一个样本，如果 EN1 置位，计数器将持续增加。

计数器可在任何时刻写入。如果时钟在写操作和读操作之间作用，在写入数据后重新读出，数据可能不同。

计数器可能用 3 种时钟源：MCLK、ACLK 和 CMP。

当计数器的允许输入 EN1 置位，计数器在输入时钟的上升沿增 1。当 ENA 和 ENA 信号的一个或 2 两个置位，极数器允许。因系统复位，这两位也复位，计数器被禁止。将它们复位能冻结计数值。

在计数值为 0FFh 时，进位信号 RC1 变高。它的下降沿使寄存器 TPCTL 的 RC1FG 位置位。

RC1FG 在计数器由 0FFh 卷回到 0h 时置位。在不是因 ENA 和 ENB 发生 EN1 无效时，标志位 EN1FG 置位。当允许位 TPIE 置位且 3 个标志 RC1FG、RC2FG 和 EN1FG 中任一个置位时发生中断请求。标志位 RC1FG、RC2FG 和 EN1FG 必须由软件复位。

9.1.2 定时器/端口计数器 TPCNT2，8 位操作

计数器 TPCNT2 在允许信号和时钟信号上和计数器 TPCNT1 不同。计数器总是以 8 位工作。它有 3 种时钟信号源可供选择：MCLK、ACLK 和 TPIN.5。

当计数值为 0FFh 时，进位信号 RC2 变高，它的下降沿使寄存器 TPCTL 的 RC2FG 位置位。

中断标志 RC2FG 在 TPCNT2 由 0FFh 卷回到 0h 时置位。

9.1.3 定时器/端口计数器，16 位操作

8 位计数器 TPCNT1 和 TPCNT2 可级联成一个 16 位计数器。将控制寄存器的 B16 位置位实现这一操作。

以字节访问对计数器作读写操作。对 TPCNT1 和 TPCNT2 数据的读写按顺序进行。当访问是在计数时进行时这一点特别要考虑。

TPCNT1 的允许位也是 16 位计数器的允许位。TPCNT1 和 TPCNT2 的时钟信号是相同的。选择由 TPSSEL0 和 TPSSEL1 决定。这一模式中 TPSSEL2 和 TPSSEL3 不起作用。

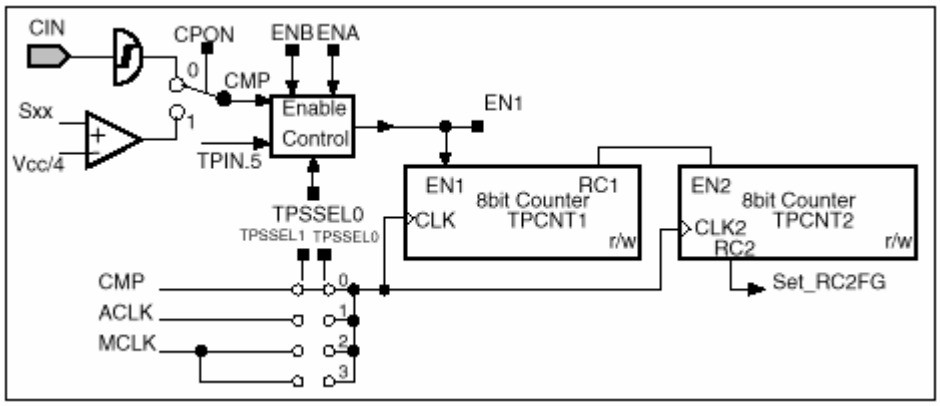


图 9.2：定时器/端口计数器

由 ENA、ENB、TPSSEL0 和 TPSSEL1 四个信号控制级联计数器，计数允许和时钟源如下：

ENB	ENA	TPSSEL1	TPSSEL0	EN1	CLK
0	0	0	0	0	CMP
0	0	0	1	0	ACLK
0	0	1	x	0	MCLK
0	1	0	0	1	CMP
0	1	0	1	1	ACLK
0	1	1	x	1	MCLK
1	0	0	0	TPIN.5*	CMP
1	0	0	1	TPIN.5*	ACLK
1	0	1	0	TPIN.5*	MCLK
1	0	1	1	TPIN.5*	MCLK
1	1	0	0	CMP*	CMP
1	1	0	1	CMP*	ACLK
1	1	1	0	CMP*	MCLK
1	1	1	1	CMP*	MCLK

引脚 CIN、Sxx 或伴随 3 个时钟信号 ACLK、MCLK、CMP 之一可使计数器停止或计数。

将正反 TPIN.5 或正反 CMP 信号作用于计数器允许信号 EN1，16 位计数器便会在每个 ACLK 或 MCLK 周期增 1，这一特性可用于测量 TPIN.5 或 CMP 引脚的信号的宽度。

进位信号 RC2 在计数值为 0FFFFh 时置位。

当计数器由 0FFFFh 卷回到 0h 时，标志位 RC2FG 置位。当允许信号 EN1 禁止，标志位 EN1FG 置位。允许信号 EN1 的源是 TPIN.5 或 CMP 信号。因此经软件用 ENA 和 ENB 禁止 EN1，EN1FG 不会置位。

9.2 定时器/端口寄存器

定时器/端口模块的硬件是字节结构，需由字节指令访问（后缀“B”）。

寄存器	缩写	寄存器类型	地址	初始状态
● TP 控制寄存器	TPCTL	读写	04Bh	复位
● TP 计数器 1	TPCNT1	读写	04Ch	不变
● TP 计数器 2	TPCNT2	读写	04Dh	不变
● TPO/P 数据寄存器	TPD	读写	04Eh	复位
● TP 数据允许寄存器	TPE	读写	04Fh	复位

定时器/端口控制寄存器

控制寄存器中的信息决定定时器/端口模块的操作。

TPCTL（04Bh)

7							0
TPSSEL1	TPSSEL0	ENB	ENA	EN1	RC2FG	RC1FG	EN1FG
rw-0	rw-0	rw-0	rw-0	r-0	rw-0	rw-0	rw-0

图 9.3：定时器/端口控制寄存器

位 0： 如允许信号来自 CMP 或 TPIN.5，计数器 TPCNT1 的允许信号 EN1 的下降沿使允许标志 EN1FG 置位。EN1FG 由事件置位，需由软件复位，否则将保持置位。
EN1FG 可以在定时器/端口中断服务程序中检查，以确定是允许位 EN1 还是进位标志 RC 在计数器从 0FFh 卷回到 0h 时置位引起中断。

位 1： 计数器 TPCNT1 由 0FFh 卷回到 0h 由 RC1FG 指示（溢出)。这一事件使它置位，同时必须由软件复位，否则将保持置位。
可用于中断服务程序中判断中断事件来源。

位 2： 计数器 TPCNT2 由 0FFh 卷回到 0h 由 RC2FG 指示（溢出)。这一事件使它置位，同时必须由软件复位，否则将保持置位。
可用于中断服务程序中判断中断事件来源。

位 3、4、5： 计数器 TPCN1 的允许信号 EN1 可读。它的值由 ENA、ENB、TPSSEL0 决定。

ENB	ENA	TPSSEL0	EN1
0	0	x	0
0	1	x	1
1	0	0	TPIN.5*
1	0	1	TPIN.5
1	1	0	CMP*
1	1	1	CMP

位 6、7： TPSSEL0 和 TPSSEL1 控制选择计数器 TPCNT1 的 3 个时钟源之一。

TPSSEL1	TPSSEL0	CLK1
0	0	CMP
0	1	ACLK
1	x	MCLK

定时器/端口计数器 TPCNT1 和 TPCNT2

两个计数器都是 8 位的，必须通过字节指令访问。

TPCNT1（04Ch）

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

TPCNT2（04Dh）

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

图 9.4：定时器/端口计数器寄存器

两个计数器都可以独立读写。用 CLEAR 指令对计数器复位。

定时器/端口数据寄存器

数据寄存器有 6 位输出数值和 2 位比较器控制位。

TPD（04Eh）

7							0
B16	CPON	TPD.5	TPD.4	TPD.3	TPD.2	TPD.1	TPD.0
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

图 9.5：定时器/端口数据寄存器

位 0 至 5：　TPD.0 至 TPD.5 保存输出到引脚 TP.0 至 TP.5 的输出值。当 TPE.0 至 TPE.5 允许这些 3 态输出端，数据会加到引脚上。它们在 PUC 后复位。
TP.5 已在模块内利用，可通过控制寄存器 TPCTL 的允许位 EN1 读取。

位 6：　比较器的 CPON 位开关比较器的供电电源。用于节省复位时的电流。PUC 后 CPON 位复位，比较器停止。

位 7：　B16 选择计数器 TPCNT1 和 TPCNT2 的工作模式。它们可以是两个独立的 8 位计数器，也可以是一个 16 位的计数器。它们只能以字节模式访问。在 16 位模式下，访问是分别对 TPCNT1 和 TPCNT2 操作的。
B16 = 0：两个 8 位计数器
B16 = 1：一个 16 位计数器。TPCNT1 为低字节，TPCNT2 为高字节。当 TPCNT1 由 0FFh 卷回到 0h 时，TPCNT2 加 1。

定时器/端口允许寄存器

定时器/端口允许寄存器有 6 位控制位和 2 位计数器溢出指示位。

TPE（04Fh）

7							0
TPSSEL3	TPSSEL2	TPE.5	TPE.4	TPE.3	TPE.2	TPE.1	TPE.0
rw-1	rw-1	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

图 9.6：定时器/端口允许寄存器

位 0 至 5：　TPE.0 至 TPE.5 控制 TP.0 至 TP.5 的 3 态。在 PUC 后复位使输出成高阻态。

位 6、7：　B16 复位时，TPSSEL2 和 TPSSEL3 控制选择计数器 TPCNT2 的 4 个时钟源之一。

B16 置位时，TPSSEL2 和 TPSSEL3 不起作用，TPCNT2 时钟源和 TPCNT1 相同。

B16	TPSSEL3	TPSSEL2	CLK2
0	0	0	TPIN.5
0	0	1	ACLK
0	1	0	MCLK
0	1	1	“0”或 VSS
1	x	x	≡CLK1

9.3 定时器/端口的 SFR 位

定时器/端口模块有一个中断向量，多源中断标志（不在 SFR 中）和一个中断允许位。

中断允许位 TPIE 位于寄存器 IE.2，初始状态为复位。

多源中断标志位 RC1FG、RC2FG 和 EN1FG 位于 TPCTL，初始状态为复位。

中断标志位 RC1FG、RC2FG 和 EN1FG 在中断服务时不能由硬件自动复位。计数器允许信号 EN1 的下降沿使标志 EN1FG 置位并指示计数器暂停。当用软件通过控制信号 ENA 和 ENB 使 TPCNT1 由允许切换到禁止时，EN1FG 不会置位。

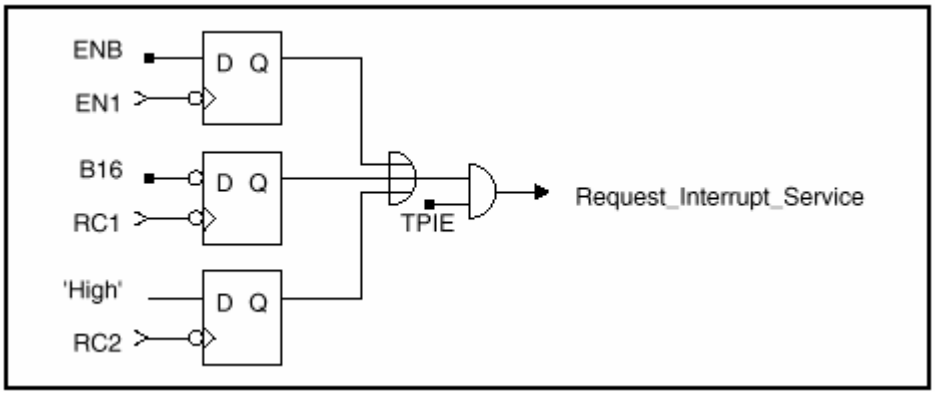


图 9.7：定时器/端口中断原理图

当定时器/端口发生中断请求，中断程序必须判断中断源并决定如何处理。当选为 8 位计数器模式，3 个中断源可能请求中断服务：EN1 下降沿、TPCNT1 溢出（RC1）、TPCNT2 溢出（RC2）。在 16 位计数器模式（B16 置位），2 个中断源可能请求中断服务：EN1 下降沿、TPCNT2 溢出。

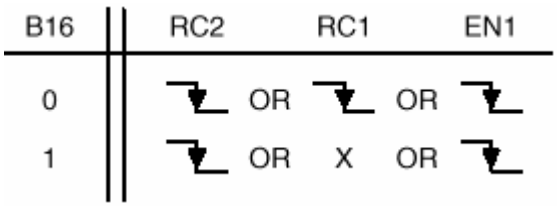


图 9.8：定时器/端口的中断请求

中断请求位必须在中断程序中复位。否则，在 GIE 置位或 RETI 指令执行后立即又发生中断请求。PUC 后定时器/端口中断允许位 TPIE 复位，这时不会发生中断请求。当 TPIE 和 GIE 置位而系统/CPU 不发生 PUC 和 NMI 时会执行中断服务。

9.4 定时器/端口在 A/D 中的应用

定时器/端口模块可为电阻或电容式传感器提供 A/D 转换功能。

对于温度测量最常见的传感器是具有正、负温度系数的热敏电阻。硅传感器、NTC 电阻和铂电阻都是这类传感器。

9.4.1 R/D 转换原理

测量电容的放电时间可以将电阻值转变为数字信号。这个电容在放电前先行充电。

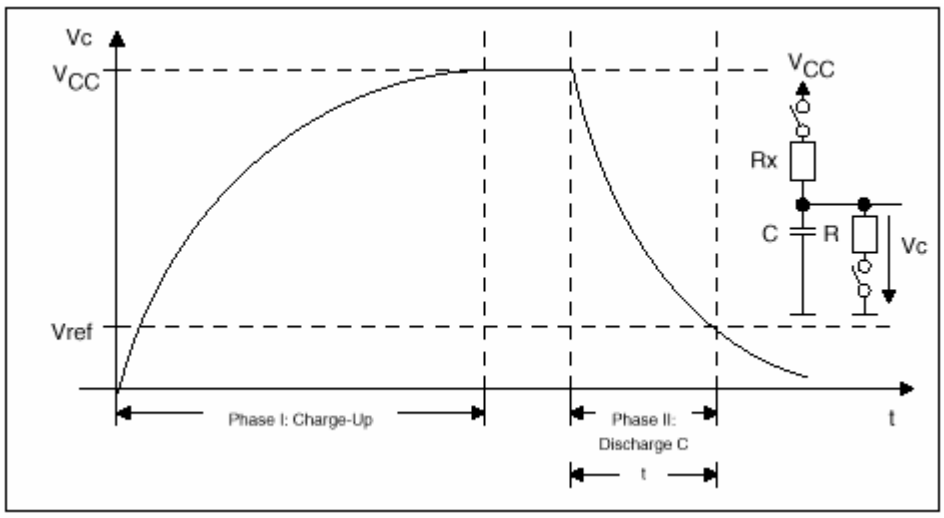


图 9.9：RC 充放电时序

用电压比较器和计数器测量电容放电至参考电压 Vref 的时间。由于比较器输出作为计数器的允许信号，电容 C 的电压在 Vref 之上时计数器一直增加。

这种时间测量方法的原理公式是：

$$t = - R * C * \ln(Vref/Vcc)$$

$$t = N * t_{clock}$$

$$N * t_{clock} = - R * C \ln(Vref/Vcc)$$

$$N = - R * C * f_{clock} * \ln(Vref/Vcc)$$

其中C，f_{clock}和Vref/VCC已知即可确定阻值。利用一个稳定的参考电阻作变换，传感器测量可下式得出；

$$Nmeas/Nref = (- Rmeas * C * \ln(Vref/Vcc))/(- Rref * C \ln(Vref/Vcc))$$

$$Nmeas/Nref = Rmeas/Rref$$

$$Rmeas = Rref * (Nmeas/Nref)$$

其中，假定电路用同一个电容，在两个测量阶段的电压和时钟周期是相同的。

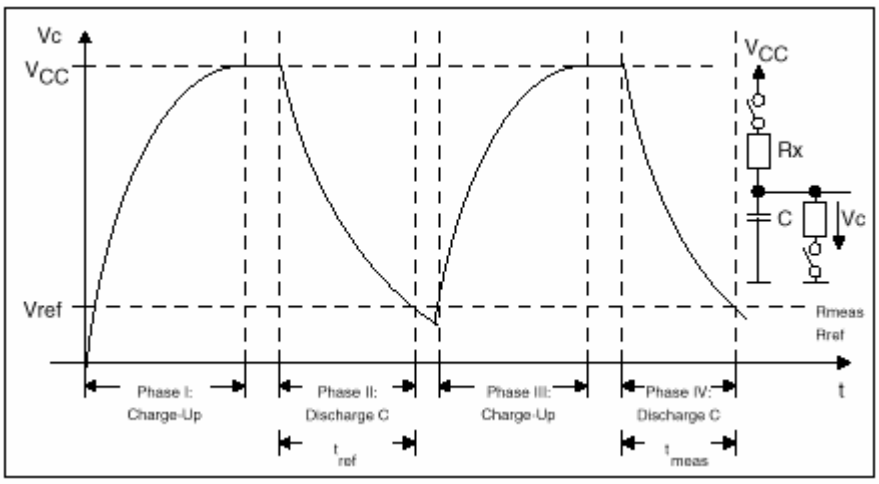


图 9.10：在 R/D 变换中利用 Rref 和 Rmeas 的充放电时序

在阶段 I 和 III，电容 C 通过 Rx 充电至 VCC，它通过 Rref 或 Rmeas 放电。如果只用 Rref，电容的放电时间就可以精确测定。

图 9.11：转换原理电路图

电阻接至 Vss 时电容 C 放电。所有的寄生放电通路都会影响电容放电至 Vref 的时间。

9.4.2 分辨率大于 8 位的转换

转换由比较器和 16 位计数器和数字输出来实现。

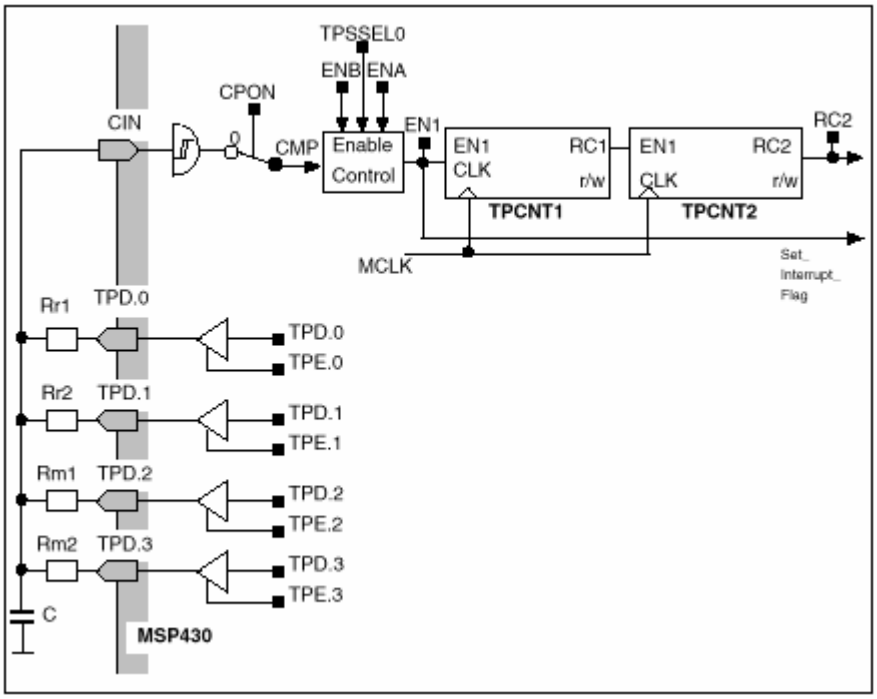


图 9.12：ADC 应用实例

在实例中，外部元件是 2 个参考电阻 Rr1 和 Rr2 和 2 个待测电阻 Rm1 和 Rm2。

用控制寄存器 TPCTL 来配置图中的电路。图中电路已简化为只有有效连接和功能模块。控制寄存器 TPCTL 写入 9Eh，即 B16、TPSSEL1、TPSSEL0、ENB 和 ENA 置位。

电容 C 通过 Rr1 与/或 Rr2 充电至 VCC。

四个放电阶段的时间计算公式是；

$$tr1 = Nr1 * t_{MCLK} = - Rr1 * C * \ln(Vref/Vcc)$$
$$tr2 = Nr2 * t_{MCLK} = - Rr2 * C * \ln(Vref/Vcc)$$
$$tm1 = Nm1 * t_{MCLK} = - Rm1 * C * \ln(Vref/Vcc)$$
$$tm2 = Nm2 * t_{MCLK} = - Rm2 * C * \ln(Vref/Vcc)$$

电阻 Rm1 和 Rm2 的计算公式是；

$$(Rr1-Rr2)/(Rmx-Rr2) = (Nr1-Nr2)/(Nmx-Nr2)$$
$$Rmx = Rr2 + (Rr1-Rr2) * (Nmx-Nr2)/(Nr1-Nr2)$$
$$Rm1 = Rr2 + (Rr1-Rr2) * (Nm1-Nr2)/(Nr1-Nr2)$$
$$Rm2 = Rr2 + (Rr1-Rr2) * (Nm2-Nr2)/(Nr1-Nr2)$$

10. 定时器

目录 页号

10.1 Basic Timer1

Basic Timer 工作的目的是支持软件和各种外围模块工作在低频率、低功耗条件下。下面是软件功能受到晶振稳定性控制的一些例子：

- 实时时钟 RTC
- 键盘去抖动
- 软件时间自加一特性

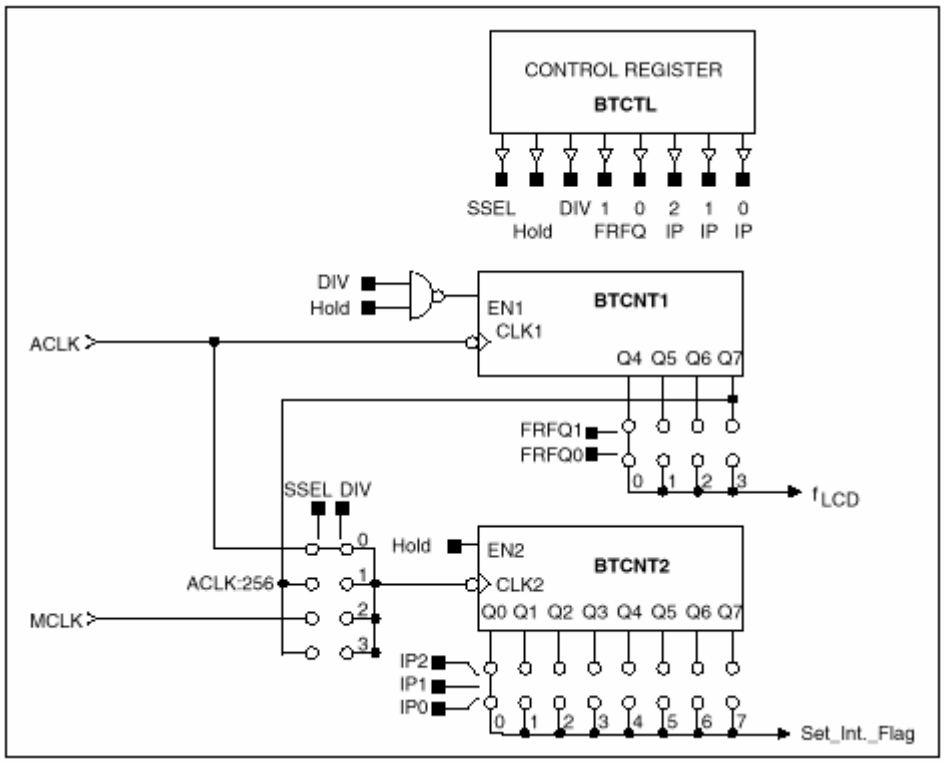


图 10.1：Basic Timer 结构

Basic Timer 向别的外围模块提供低频控制信号。软件可以访问 2 个 8 位计数器。

控制寄存器 BTCTL 的标志位控制或选择不同的工作模式。寄存器 BTCTL、8 位计数器 BTCNT1 和 BTCNT2 可用软件控制。当系统上电复位、看门狗溢出或其他情况发生时，寄存器各位保持无定义或不变状态。用户程序通常在初始化时定义 Basic Timer 的工作状态。

10.1.1 Basic Timer1 寄存器

Basic Timer1 模块的硬件是字节结构的，必须用字节指令访问（后缀“B”）。

寄存器	缩写	类型	地址	初始状态
BT1 控制寄存器	BTCTL	读写	040h	不变
BT 计数器 1	BTCNT1	读写	046h	不变
BT 计数器 2	BTCNT2	读写	047h	不变

Basic Timer1 控制寄存器

控制寄存器的信息决定了 Basic Timer 的运行，不同的位选择频率源、中断频率和 LCD 控制电路的帧频率。

BTCTL（040h）

7							0
SSEL	HOLD	DIV	FRFQ1	FRFQ0	IP2	IP1	IP0
rw	rw	rw	rw	rw	rw	rw	rw

图 10.2：Basic Timer1 寄存器

位 0 至 2： 最低 3 位 IP2 至 0 决定中断间隔时间。即中断标志 BTIFG 置位间隔时间。

位 3 至 4： FRFQ1 和FRFQ2 选择频率信号f_{LCD}。与片上LCD外围模块一起工作的设备用此频率产生COMM和SEG行的时序信号。

位 5： 参见位 7

位 6： HOLD 位停止计数器的工作
HOLD 位置位，BTCNT2 停止工作
HOLD 位和 DIV 位置位，BTCNT1 停止工作

位 7： SSEL 和 DIV 位选择 BTCNT2 的输入频率

SSEL	DIV	CLK2
0	0	ACLK
0	1	ACLK/256
1	0	MCLK
1	1	MCLK/256

BTCTL（040h）

7							0
SSEL	HOLD	DIV	FRFQ1	FRFQ0	IP2	IP1	IP0
rw	rw	rw	rw	rw	rw	rw	rw

FRFQ1	FRFQ0	IP2	IP1	IP0	中断频率
		0	0	0	f_{CLK2}：2
		0	0	1	f_{CLK2}：4
		0	1	0	f_{CLK2}：8
		0	1	1	f_{CLK2}：16
		1	0	0	f_{CLK2}：32
		1	0	1	f_{CLK2}：64
		1	1	0	f_{CLK2}：128
		1	1	1	f_{CLK2}：256
0	0				$f_{LCD}=f_{ACLK}$：32
0	1				$f_{LCD}=f_{ACLK}$：64
1	0				$f_{LCD}=f_{ACLK}$：128
1	1				$f_{LCD}=f_{ACLK}$：256

图 10.3：Basic Timer 寄存器功能

Basic Timer1 计数器 BTCNT1

BTCNT1 对辅助时钟 ACLK 分频。LCD 驱动的帧频从计数器高 4 位输出选择。最高位触发器输出可作为第二个计数器 BTCNT2 的时钟输入。计数器的 Q0 至 7 输出可读，并且能用软件写入。

BTCNT1（046h）

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

Basic Timer1 计数器 BTCNT2

BTCNT2 对输入时钟频率分频。输入时钟源可选自 MCLK、ACLK 或 ACLK/256。中断周期可以由 BTCTL 中的 IP0 至 2 位选择 8 个触发器输出之一。

BTCNT2（047h）

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

计数器的 Q0 至 7 的输出可读。计数器(Q0 至 7)可以用软件写入。

10.1.2 SFR 位

SFR 的各位根据 Basic Timer 的选定功能，实现与系统的交互控制。

- Basic Timer 中断标志 BTIFG(位于 IFG2.7)
- Basic Timer 中断允许位 BTIE(位于 IE2.7)

HOLD 位禁止模块的所有功能。并把功耗降低到最低程度，即漏电流。

当计数器允许或禁止时不会发生额外计数。系统对通用模块寄存器 BTCTL 的访问不会影响计数。它可以用通常的方式进行读写操作。

中断标志和中断允许位遵循一般的模块中断规则。除了受各自的中断允许控制外，中断请求还受控于通用中断允许位 GIE。PUC 使中断允许标志 BTIE 复位。当 Basic Timer 的中断请求被接受，中断标志 BTIFG 复位。

10.1.3 Basic Timer1 的操作

Basic Timer 总是按时钟 ACLK 或 MCLK 作增计数。SSEL 控制信号选择辅助时钟 ACLK 或主时钟 MCLK（系统时钟 fsystem）作为 BTCNT2 的时钟。

中断可用作系统的控制，它是单源中断。

Basic Timer 可以工作在两种不同模式下：

- 2 个独立的 8 位定时器/计数器
- 1 个 16 位定时器/计数器

8 位计数器模式

在 8 位计数器模式，BTCNT1 总是按 ACLK 作增计数。计数器读出时必须考虑计数器时钟(ACLK)和系统时钟(MCLK)的异步特性，计数器可用软件作与计数时钟异步的写操作。

BTCNT2 时钟信号可由 SSEL 和 DIV 选自 MCLK、ACLK 或 ACLK/256 之一。BTCNTL 按选定时钟作增计数。

8 个定时器输出之一可用于对 Basic Timer 中断标志置位。当选择 ACLK 或 ACLK/256 为时钟，读写访问可以是异步的。

16 位计数器模式

DIV 置位选择 16 位定时器/计数器模式。这时 BTCNT1/BTCNT2 时钟源是 ACLK 信号。

HOLD 位停止这 2 个 8 位计数器工作。

10.1.4 Basic Timer1 的操作：f_{LCD}信号

LCD外围模块用f_{LCD}产生COMM和SEG行的时序信号。f_{LCD}由ACLK产生。用 32768Hz晶振时，f_{LCD}可以是 1024Hz、512Hz、256Hz或 128Hz。用FRFQ1 和FRFQ2 可正确选择帧频，合适的f_{LCD}频率取决于LCD每帧的字符数据量和LCD重复速率。

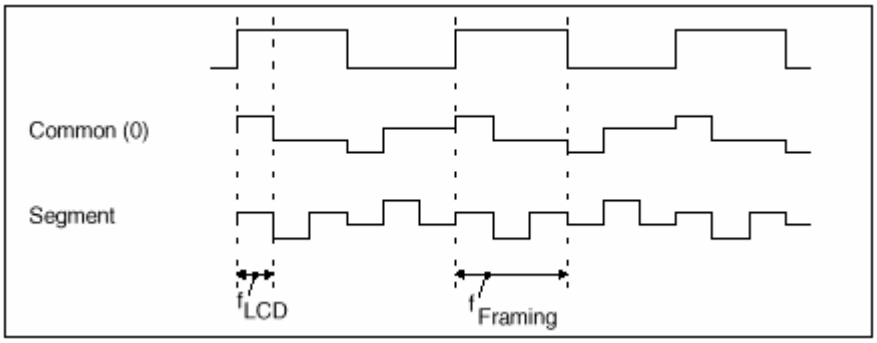


图 10.4：选作 LCD 的频率(3MUX 的例子)

3MUX 举例：

LCD数据： $f_{Framing}$ = 100Hz ... 30Hz

频率： $f_{LCD} = 6 * f_{Framing}$

f_{LCD} = 6 * 100Hz ... 6 * 30Hz = 600Hz ... 180Hz

可选 flcd: 1024Hz、512Hz、256Hz 或 128Hz

f_{LCD} =256 FRFQ1=1; FRFQ0=0

10.2 8 位间隔（Interval）定时器/计数器

8 位间隔定时器支持 3 个主要功能：

- 串行通信或数据交换
- 脉冲计数或脉冲累加
- 定时器

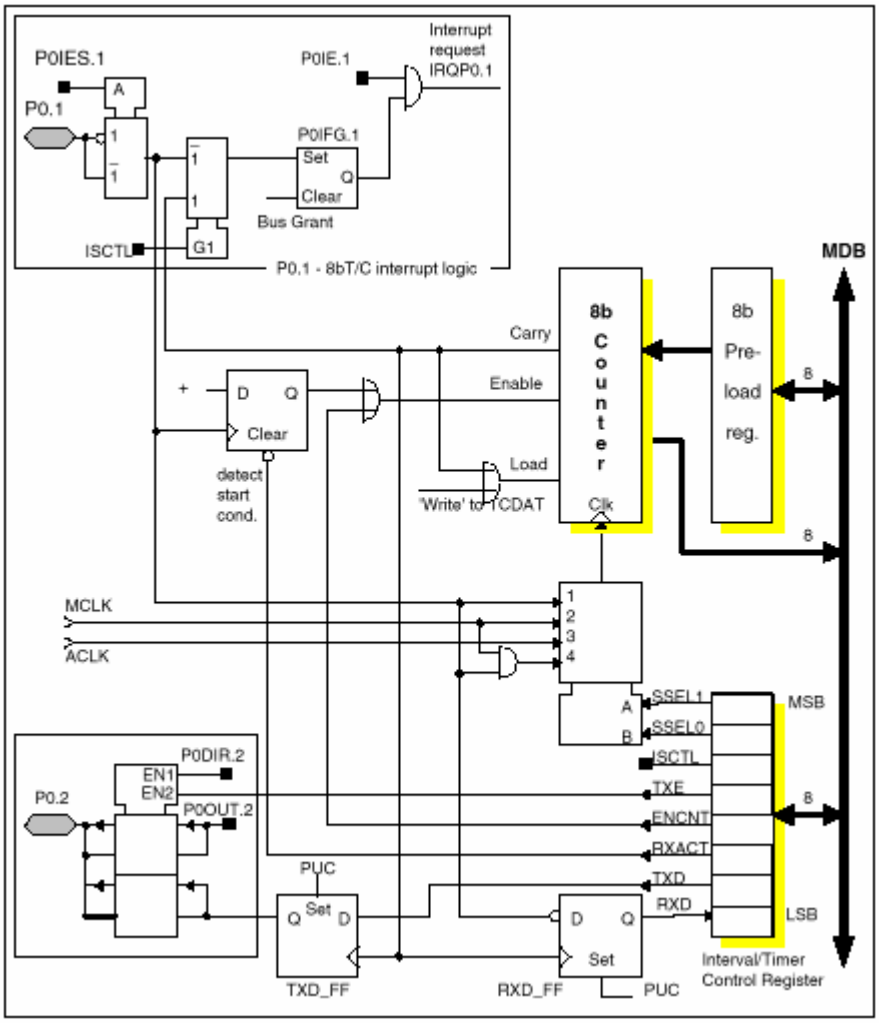


图 10.5：8 位定时器/计数器原理图

10.2.1 8 位定时器/计数器操作

8 位定时器/极树器包括以下主要模块：

- 8 位带预置数寄存器的增计数器
- 8 位控制寄存器
- 输入时钟选择器
- 触发沿检测，例如异步通信协议中的起始位
- 输入和输出数据锁存，由 8 位计数器的进位信号触发。

8 位带预置数寄存器的增计数器

8 位计数器由控制寄存器的 2 位(SEL0，SEL1)选定的输入时钟作增计数。计数器的 2 个输入(LOAD，ENABLE)控制它的操作。

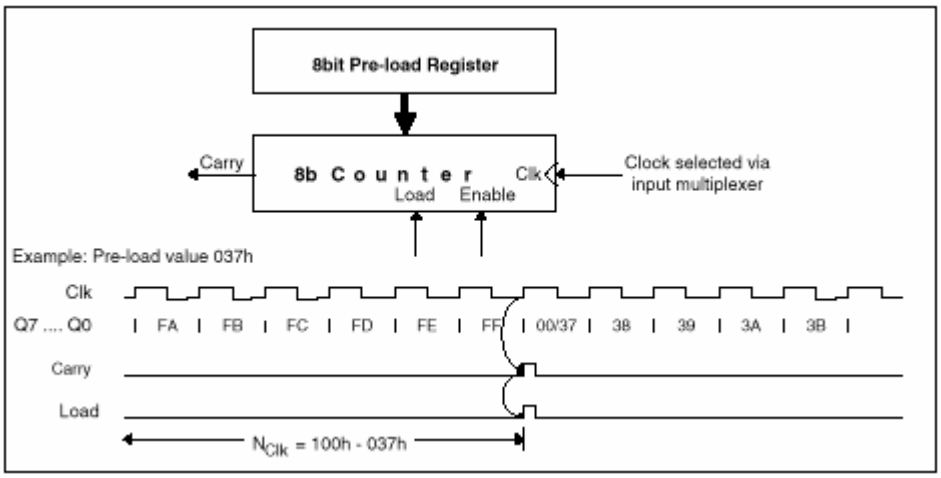


图 10.6：8 位计数器原理图

两输入之一控制预置数功能。预置操作将预置数寄存器中数据装入计数器。对计数器的写操作把预置数寄存器的内容写入计数器。

所有指令均可对预置数寄存器进行读写操作。预置数寄存器相当于一个缓存器，并且可以在计数器完成预置后立即进行写操作。

另一个输入允许计数操作。当允许信号置位，计数器在每个输入时钟的上升沿加 1。

8 位控制寄存器

8 位控制寄存器的内容选择定时器/计数器的工作方式和控制各种功能。

输入时钟选择器

8 位控制寄存器外的两个信号选择 8 位增计数器的时钟源。4 个时钟源是 MCLK、ACLK、由 P0.1 脚进入的外部信号及由 P0.1 引脚和 MCLK 进行逻辑与(.AND.)后的信号。

边沿检测

类似 UART 协议的串行协议需要检测起始位边沿以确定接收到的发送数据的起始位。

输入和输出数据锁存，RXD_FF 和 TXD_FF

将数据锁存入输入数据和输出数据锁存器的时钟信号是 8 位计数器的进位信号。两个锁存器用作 1 位缓存器并按预先定义时序改变它们的输出。

10.2.2 8 位定时器/计数器寄存器

定时器/计数器模块硬件通过 8 位 MDB 和 MAB 控制。它必须用字节指令访问。

寄存器	缩写	类型	地址	初始状态
● T/C 控制寄存器	TCCTL	读写	042h	复位
● 预置数寄存器	TCPLD	读写	043h	不变
● 计数器	TCDAT	读(写)	044h	不变

8 位定时/计数控制寄存器

控制寄存器的信息决定了定时器/计数器的操作。

TCCTL（042h)

7							0
SSEL1	SSEL0	ISCTL	TXE	ENCNT	RXACT	TXD	RXD
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r(-1)

图 10.7：8 位定时器/计数器

位 0： RXD 位只读。由 P0.1 来的外部信号在 8 位计数器进位时锁存。外部信号由固定的时序来扫描，与程序运行时间的变化无关。

位 1： TXD 位是由 8 位计数器进位来定时从 P0.2 输出的信号的缓存。

位 2： RXACT 位控制边沿检测逻辑。边沿检测要用 ENCNT 位复位来允许计数操作。
RXACT=0：边沿检测 FF 清除，它不能作为允许计数操作的信号。
RXACT=1：边沿检测 FF 允许工作，由 P0IES.1 选定的 P0.1 引脚的上升沿或下降沿使 FF 置位，计数器为计数操作好准备。如果 FF 置位它将一直保持。

位 3： ENCNT 位设置计数允许信号。8 位计数器在每个时钟输入上升沿增 1。
与 RXACT 位一起提供启/停操作。

位 4： TXE 信号控制 TXD 的三态输出缓存器。
TXE=0：三态
TXE=1：输出缓存激活

位 5： ISCTL 信号选择 P0.1 和 8 位计数器进位作为中断源
ISCTL=0：I/O 引脚 P0.1 是 P0IFG.1 的中断源
ISCTL=1：8 位计数器进位是 P0IFG.1 的中断源。

位 6,7： SSEL0 和 SSEL1 选择输入时钟源

SSEL1	SSEL0	时钟源
0	0	由 P0IES.1 决定的 P0.1 引脚信号
1	0	MCLK
0	1	ACLK
1	1	由 P0IES.1 决定的 P0.1 引脚信号.AND.MCLK

8 位定时器/计数器预置数寄存器

对计数器(TCDAT)作写操作时，预置数寄存器中的信息被装入到 8 位计数器中。

```
;========= Definitions ========================================
Dummy       .EQU    0       ; Value for dummy is not loaded into
                            ; counter
TCDAT       .EQU    044h    ; Address of 8-bit Timer/Counter
;========= Write pre-load register content to 8-bit Timer/Counter =
;
            MOV.B   #Dummy,&TCDAT
;
```

预置数寄存器 TCPLD 可以用地址 043h 访问。

8 位计数器数据

8 位计数器的数据可以用地址 044h 读出，对计数器进行写操作将把预置数寄存器中的数据而不是指令中的数据装入计数器。

10.2.3 与 8 位定时器/计数器有关的 SFR 位

8 位定时器/计数器没有独立的中断位：它和 P0 端口共享中断位。TCCTL 中的 ISCTL 位选择中断标志的中断源。

P0/RXD.1 信号或 8 位计数器的进位用作中断源。SFR 中的 2 位和 P0 地址帧中的 1 位用于处理 P0/RXD.1 的中断事件：

- P0/RXD.1 中断允许 P0IE.1(位于 IE1.3，初始状态为复位)
- P0/RXD.1 中断边沿选择 P0IES.1(位于 P0IES，初始状态为复位)

这一中断标志是单源中断标志，它在中断服务时自动复位，而允许位和边沿选择位保持不变。

10.2.4 8 位定时器/计数器在 UART 中应用

定时器/计数器外围模块具有支持用软件进行串行数据交换的特性。数据交换包括接收周期和发送周期。外围硬件支持半双工协议。

软件操作可以分为 3 个类型以满足各种应用的不同情况和不同需求：

- 在每个接收周期后立即控制位信息
- 在每个接收周期后立即控制一帧的所有位信息
- 在接收周期完成后数据帧存入内存并校验。

异步通信 UART 协议

在 UART 协议是串行位流协议，它的帧包括起始位、1 至 8 个数据位，1 个可选择校验位、1 个可选择地址位和 1 至 2 个停止位，最低位在起始位后首先发出。

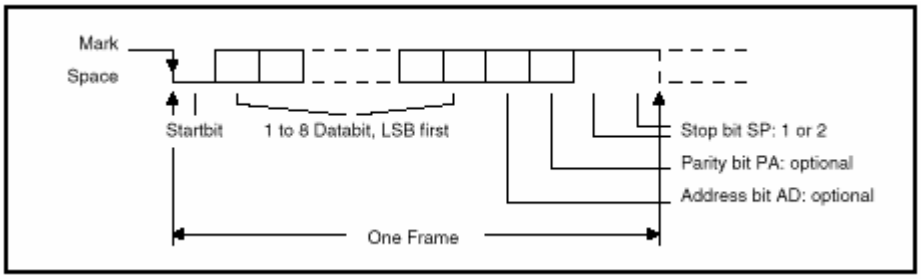


图 10.8：异步通信格式

UART 协议接收模式

定时器/计数器作为定时器工作，它的进位锁存引脚 P0.1 上的位信息，下降沿，即起始位的传号至空号跳变，表明一帧开始。每 1 位在中间扫描读取。

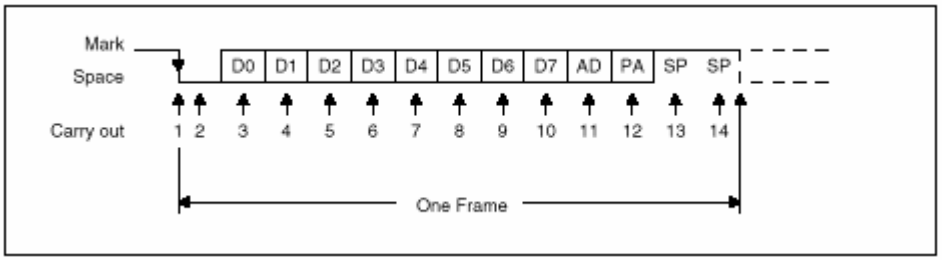


图 10.9：异步通信帧中位的扫描

软件 UART 紧密结合了接收周期的开始和波特率定时技术。如果定时器的时钟频率不是波特率的整数倍，各位的读取时序会变化。在本节的例子中工作在波特率 2400 和晶振频率 32768Hz 的条件下。结果是每一位都有自己的间隔时序，因此有各自的预置值。

UART 协议发送模式

定时器/计数器作为定时器工作，它的进位将控制寄存器 TCCTL 中 TXD 位锁存。软件 UART 装入了要从 P0.2 引脚发送的值。定时器的第二个进位把 TXD 位送到 TXD_FF。当控制寄存器中的 TXE 位置位，数据允许从 P0.2 脚发送。TXE 复位禁止缓存器与 P0.2 的连接，同时使 TXD_FF 输出置位。它相当于 UART 格式中定义的传号状态。

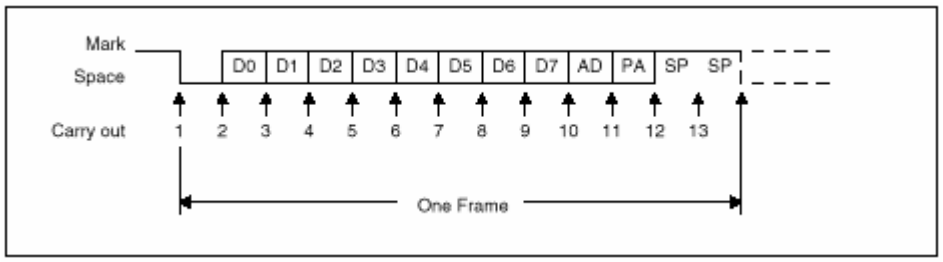


图 10.10：异步通信帧中位的发送

软件 UART 的发送时序取决于波特率。如果定时器的时钟频率不是波特率的整数倍，各位的发送时序会变化。在本节的例子中工作在波特率 2400 和晶振频率 32768Hz 的条件下。结果是每一位都有自己的间隔时序，因此有各自的预置值。

每种通信都要有识别数据传输中发生错误的能力。有 4 种出错状态定义如下：

- 校验错
- 溢出错
- 帧错
- 打断检测(Break detect)

在这些传统的原理之外，可选择一个支持通信协议处理的功能：识别多帧数据块的起始和报文的目的地址。工业标准采用了两种不同的方式进行识别：线路空闲多处理机协议和地址位多处理机协议。

线路空闲多处理机模式的格式

数据块被空闲时间分隔。在字符第一个停止位后收到 10 个以上传号状态即检测到线路空闲。当采用两位停止位时，第二个停止位被看作线路空闲的第一个传号。线路空闲后收到的第一个字符是地址字符。下图是接收器检测有一个或两个停止位的线路空闲周期：

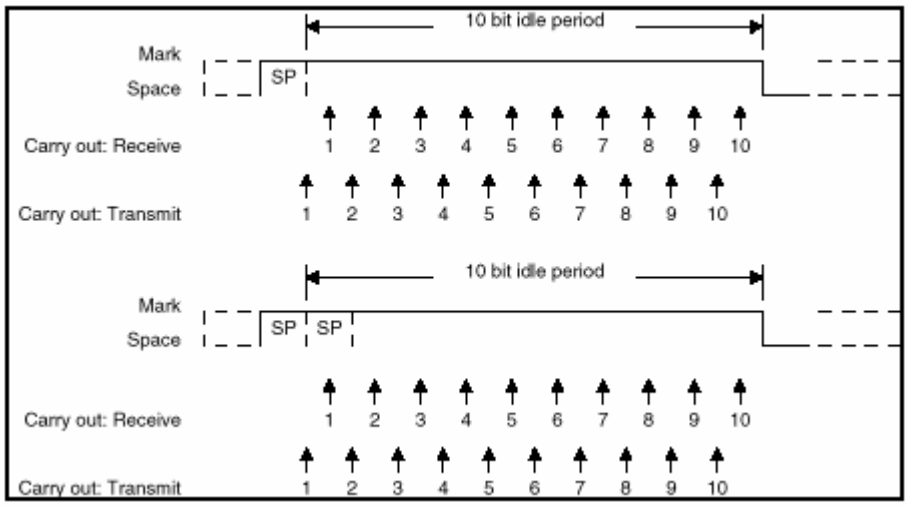


图 10.11：UART 空闲周期

推荐发送 11 位而不是 10 位的空闲周期。

精确的空闲周期导致有效的地址字符识别。多帧数据块的第一个字符可以被识别为地址。数据块中的帧间空闲周期不应超过检测 10 位空闲周期的时间。

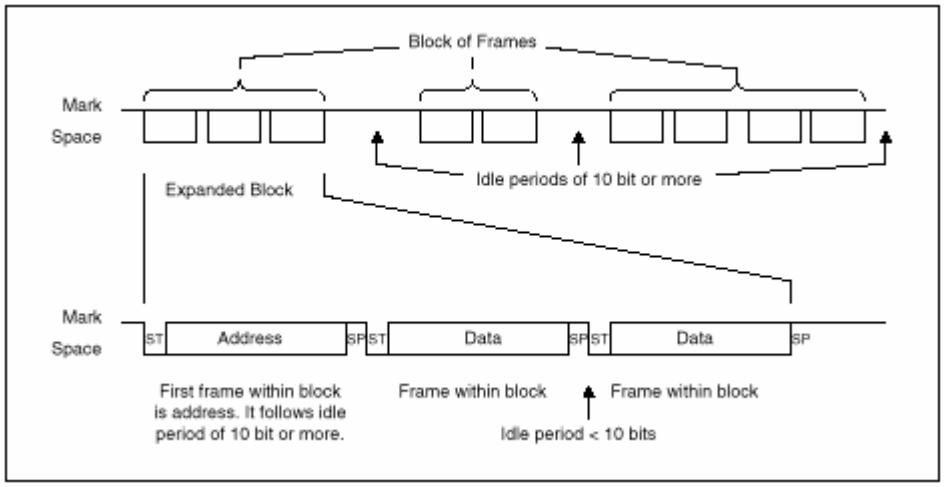


图 10.12：线路空闲多处理协议

地址位多处理机模式格式

每个字符额外含 1 位作地址识别，数据块（报文）的第一个字符含有置位的地址位。这表示这个字符是地址。

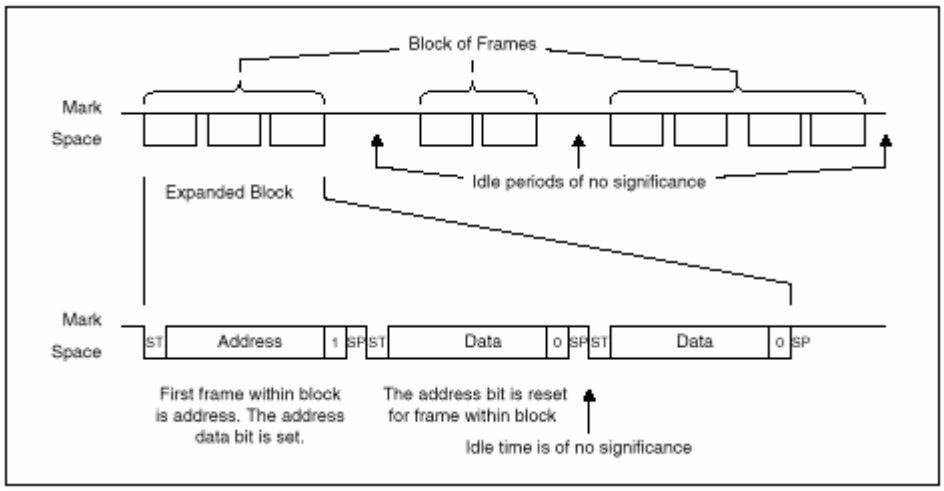


图 10.13：地址位多处理机模式格式

发送/接收应用实例

这个程序实例是采用 8 位定时器/计数器特性的串行异步通信协议例子，有以下特点：

- 波特率 2400
- ACLK = 32768 Hz

- 偶校验
- 2 个停止位
- 半双工

定时器/计数器的进位信号取代 P0.1 信号作中断源。相应的中断向量中含发送/接收中断处理程序。程序第一条指令控制程序进入发送接收部分，并用 TXRX 作运行模式指示。

```
; --------  Define interrupt vector   --------------------------------
            .SECT   "RXTX_vec",0FFF8H    ;Vector of P0.1 or carry from
                                         ;8-bit Timer/Counter
            .Word   VECRXTX              ;Address of UART handler
            ...........
```

两个进位信号间的时间间隔对每次接收或发送周期都不相同。选定波特率 2400 和晶振 32768Hz 要求除数为 32768/2400=13.67，用 14-13-14 序列作除数来实现这一理想的除数。

```
; Definitions of used expressions
RXD         .EQU    1               ; Receive data bit in control register TCCTL
TXD         .EQU    2               ; Transmit data bit in control register TCCTL
RXACT       .EQU    4
ENCNT       .EQU    8               ; Counter enable bit in control register TCCTL
TXE         .EQU    010h            ; 1: TX buffer active, 0: TX buffer 3-state
ISCTL       .EQU    020h
TCCTL       .EQU    042h            ; Address of Timer/Counter control register
TCPLD       .EQU    043h            ; Address of Timer/Counter pre-load register
TCDAT       .EQU    044h            ; Address of Timer/Counter
BitTime1    .EQU    0100h - 0Eh     ; 14/32768sec. = 427.2 us bit length
BitTime1_2  .EQU    0100h - 07h     ; Half of bitime1
BitTime2    .EQU    0100h - 0Dh     ; 13/32768sec. = 396.7 us bit length
AdP0_0      .EQU    0h              ; Interrupt enable 1 register address (SFR)
IEnP0_0     .EQU    08h             ; Bit in Interrupt enable 1 register (SFR)
ParVal      .EQU    0h              ; Parity Even selected
;
(P0.1 及 TC 中断允许)
; Registers or RAM used for data handling
Rcstatus    .EQU    0200h           ; RAM (or Register), stores actual status of
                                    ; receive sequence
TXStatus    .EQU    0201h           ; RAM (or Register), stores actual status of
                                    ; transmit sequence
TXData      .EQU    R6              ; Register that contains the transmit data
RCData      .EQU    R6              ; Stores receive data (RXD) in HighByte
Parity      .EQU    0yyyh           ; LSB is actual status of parity. The start
                                    ; value determines odd or even parity
Bend        .EQU    2 x 12
```

在程序的主循环中,用表的数据作为输出并把接收数据放入表中来演示发送和接收数据序列的功能。

```
;------------- Transmitting of frames: a table is to be output --------------
; Ry points to the table
;
            MOV     #Table2,Ry      ; Start of table copied to Ry
L$5         CRL.B   &TXStatus
            CMP     #TabEnd+1,Ry    ; All frames transmitted?
            JEQ     TabFin          ; Yes, stop transmision and continue program
            MOV.B   @Ry+,TXData     ; Info to TXData
            CALL    #TXInit         ; No, initialize transmission
TXStat      CMP     #Bend,TXStatus  ; output of one frame completed?
            JEQ     L$5             ; Yes, transmit next data of table!
            JMP     TXStat          ; No, wait for completion
            ......
            ......
Table2      .       0xxh Byte       ; Start of table containing data for trans.
            ......
            ......
TabEnd      .       0zzh Byte       ; End of table containing data for trans.
            ......
TabFin      ......                  ; Transmission of table is completed
                                    ; Continue program here
; ----------------- Prepare receiving of one frame ---------------------------
; Rx points to the table
;
            MOV     #Table1,Rx      ; Start of receive table copied into Rx
            CALL    #RCPrep         ; Receive of next frame
            ......
;---------- Processing part of received frame: Store frame in table1 ----------
RECCMPL     RLA     RCData          ; Adjust info to HighByte (remove parity bit)
            SWPB    RCData          ; Swap info to LowByte
            MOV.B   RCData,0(Rx)    ; Store info in table1
            INC     Rx              ; and pre-increment of table pointer
            CALL    #RCInit         ; Prepare for next frame
            ......                  ; Continue with background program
```

发送模式应用实例：

2400 波特，ACLK，8 位数据，偶校验，2 停止位。

发送模式用了 8 位定时器/计数器、预置数寄存器、控制寄存器、时钟选择和 TXD 数据锁存等功能。

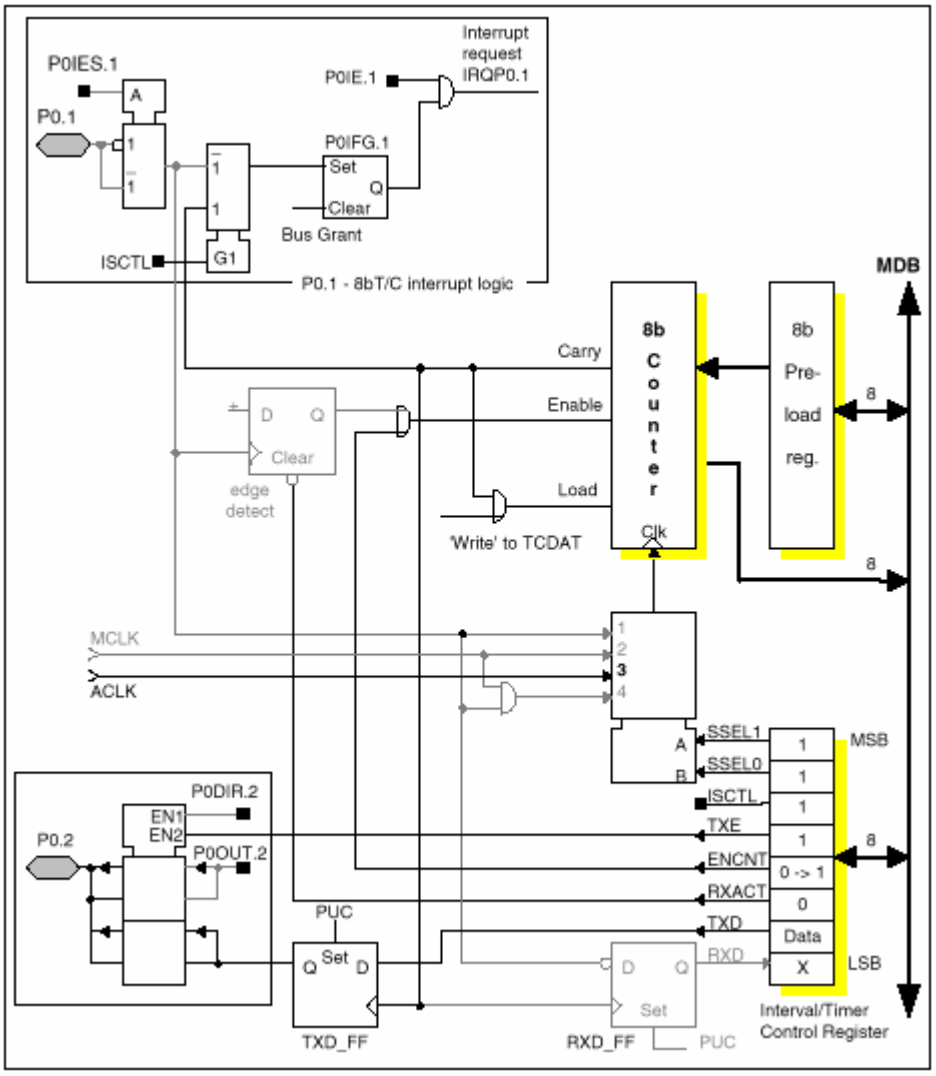


图 10.14：8 位定时器/计数器结构发送应用实例，2400 波特，ACLK

在开始串行通信字符发送前，有一些工作状态需定义：

- 允许输出缓存 -> TXE 置位。
- 选定输入 8 位定时器的时钟 -> 用 SSEL0 复位和 SSEL1 置位选 ACLK。
- 用中断源控制位 ISCTL 选定时器进位信号。

P0.1 的方向和中断触发沿应适当选定

- 预置数寄存器装入数据。
- 对计数器的写操作将把预置数寄存器中的数据装入定时器。

- RXACT 位复位。

```
; ---------------- Prepare Transmit Cycle ---------------------------------
TXINIT  MOV.B   #BitTime1_2,&TCPLD  ; Load time until start bit starts
                                    ; Disable P0.1 O/P buffer (in) if needed
        MOV.B   #072h,&TCCTL        ; TXD = 1 : Defined Start, ACLK selected
                                    ; TXEN = 1
        MOV.B   #0,&TCDAT           ; Dummy write to load 8b counter/timer
        MOV.B   #BitTime1,&TCPLD    ; Load bit time of first bit for transmission
                                    ; into pre-load register, time of Startbit
        BIS.B   #ENCNT,&TCCTL       ; Set transmit start condition
        BIS.B   #IEnP0_0,&AdP0_0    ; Interrupt enabled for P0.1 in SFR,
                                    ; address is 0.
        CLR.B   &TXStatus           ; Temporary register is prepared.
        MOV.B   #ParVal,Parity      ; ParVal = 0 for Even, ParVal = 1 for Odd
                                    ; Parity
        RET
; ------------- Acknowledge Transmit/Receive Cycle, UART Handler --------------
; The following two instructions decide whether to transmit or to receive data
; It is necessary because they use a common interrupt vector address
VECRCTX BIT.B   #RXACT,&TCCTL       ; Test which interrupt handler is active
        JNZ     RCINTRPT            ; Receive mode is active -> Jump
;
TXINTRPT PUSH   R5                  ; RXACT = 0 --> Transmit
        MOV.B   &TXStatus,R5        ; Use TXStatus for
        BR      TXTAB(R5)           ; branching
TXTAB   .Word   TXStat0             ; Startbit ; Bitime2, 13 clocks of ACLK
        .Word   TXStat1             ; Bit 0,LSB ; Bitime1, 14 clocks of ACLK
        .Word   TXStat1             ; Bit 1     ; Bitime1, 14 clocks of ACLK
        .Word   TXStat2             ; Bit 2     ; Bitime2, 13 clocks of ACLK
        .Word   TXStat1             ; Bit 3     ; Bitime1, 14 clocks of ACLK
        .Word   TXStat1             ; Bit 4     ; Bitime1, 14 clocks of ACLK
        .Word   TXStat2             ; Bit 5     ; Bitime2, 13 clocks of ACLK
        .Word   TXStat1             ; Bit 6     ; Bitime1, 14 clocks of ACLK
        .Word   TXStat1             ; Bit 7     ; Bitime1, 14 clocks of ACLK
        .Word   TXPar               ; Parity bit; Bitime2, 13 clocks of ACLK
        .Word   TXStop              ; Stop bit 1; Bitime1, 14 clocks of ACLK
        .Word   TXStop              ; Stop bit 2; Bitime1, 14 clocks of ACLK
        .Word   TXCCmpl             ; Frame transmitted
TXStat0 BIC.B   #TXD,&TCCTL
        MOV.B   #BitTime2,&TCPLD    ;Load time 13/32768s into pre-load register
        JMP     TXRET
TXStat2 MOV.B   #BitTime2,&TCPLD    ;Load time 13/32768s into pre-load register
```

```
        JMP     L$3                 ; Shift next bit out at P0.2
TXStat1 MOV.B   #BitTime1,&TCPLD    ;Load time 14/32768s into pre-load register
L$3     RRA     TXData              ; LSB is shifted to Carry
        JNC     L$1                 ; Jump to L$1 if bit = 0
L$2     BIS.B   #TXD,&TCCTL         ; Bit =1, set TXD bit in control register TCCTL
        XOR.B   Parity              ; Count 1's for parity
        JMP     TXRET               ; Bit output completed
L$1     BIC.B   #TXD,&TCCTL         ; Bit=0, reset TXD bit in control reg. TCCTL
TXRET   INCD.B  &TXStatus           ; Bit output completed
TXStat12 POP    R5
        RETI                        ; Transmit of one bit completed
; ---------- Parity bit check: Count of 1's in Parity must be even ------------
TXPar   MOV.B   #BitTime2,&TCPLD
        BIT.B   #1,Parity           ; Check parity bit value
        JNZ     L$2                 ; Parity bit should be Mark
        JMP     L$1                 ; Parity bit should be Space
; ----------------- Output of stop bit(s) -------------------------------------
TXStop  MOV.B   #BitTime1,&TCPLD
        JMP     L$2                 ; Send stop bit 1 or 2
; ----------------- Output of one frame completed -----------------------------
TXCmpl  BIC.B   #IEnP0_0,&AdP0_0    ; Interrupt disabled for P0.2 in SFR,
                                    ; address is 0.
;       BIC.B   #ENCNT,&TCCTL       ; Stop counter to conserve power consumption
        JMP     TXStat12
; ----------------- End of transmit interrupt handler -------------------------
```

接收模式应用实例

2400 波特，ACLK，8 位数据，偶校验，2 停止位。

接收模式用了 8 位定时器/计数器、预置数寄存器、控制寄存器、时钟选择、边沿检测逻辑和 TXD 数据锁存等功能。

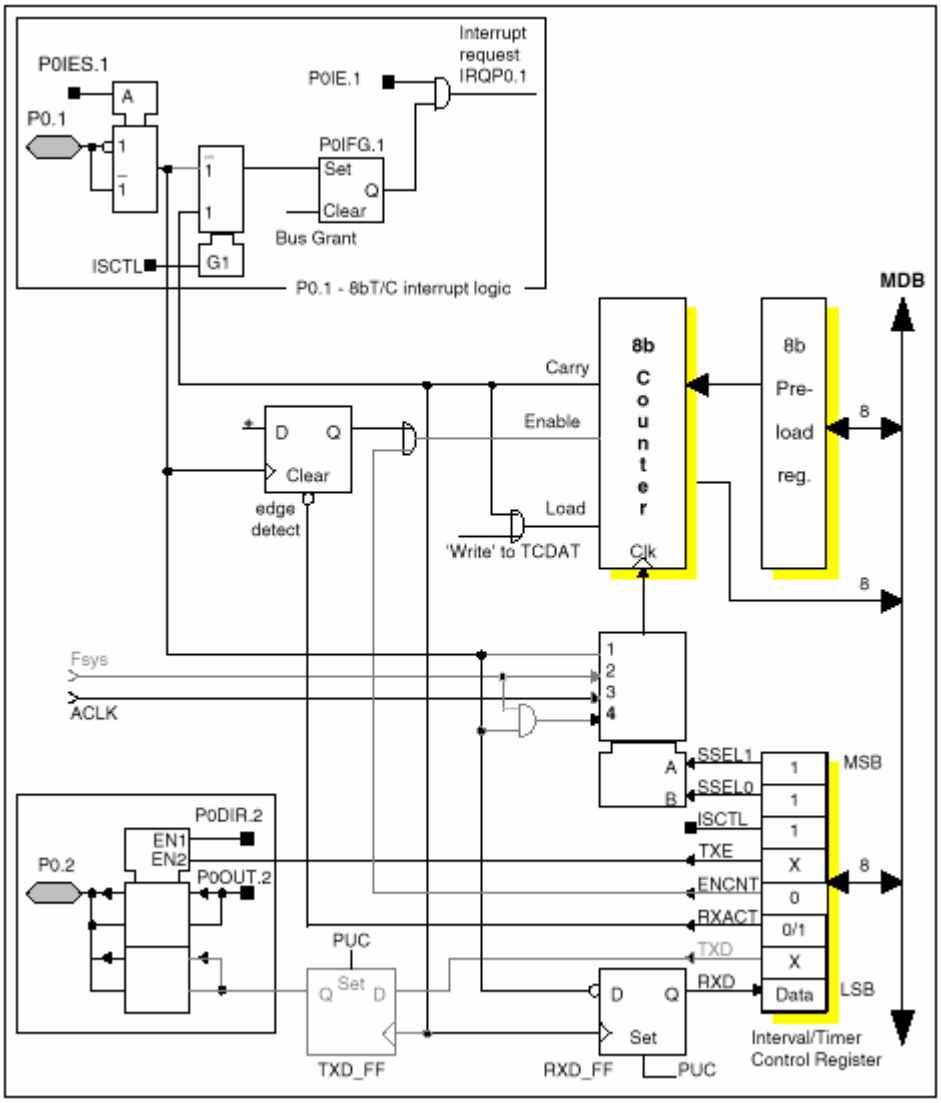


图 10.15：8 位定时器/计数器结构，2400 波特，ACLK 时钟下接收实例

在开始串行通信字符接收前，有一些工作状态需定义（假定 RXACT 复位）：

- 禁止输出缓存 -> TXE 复位。
- 选定输入 8 位定时器的时钟 -> 用 SSEL0 复位和 SSEL1 置位选 ACLK。
- 用中断源控制位 ISCTL 选定时器进位信号。

P0.1 的方向和中断触发沿应适当选定

- 预置数寄存器装入数据。

- 对计数器的写操作将把预置数寄存器中的数据装入定时器。
- RXACT 置位。

```
;           -----------------           Prepare          Receive          Cycle
-----------------------------------RCPREP  MOV.B    #062h,&TCCTL    ; SSEL0 = 0,
SSEL1 = ISCTL = 1,
                                ; all other bits are cleared
                                ; Select ACLK for clock source to 8-bit timer
                                ; Use #072h if TXEN should be enabled
RCINIT  MOV.B   #BitTime1_2,&TCPLD  ;Set Preload register with t1-2 = 0100h - 7
        MOV.B   #0,&TCDAT       ; Prepare timer interval for start bit scanning
        MOV.B   #BitTime1,&TCPLD    ; Set Preload register with Bittime 1
                                ; for receive of first data bit
        CLR     RCstatus        ; Prepare temporary registers
        MOV.B   #ParVal,Parity  ; Register Parity=0 for Even parity receive mode
                                ; and Parity=1 for Odd parity
        BIS.B   #RXACT,&TCCTL   ; activate neg. edge detect of P0.1
                                ; ( -> RX data )
        BIS.B   #IENP0_0,&ADP0_0    ; Enable interrupt according to P0.1
                                ; Interrupt source is carry from 8-bit timer
                                ; according to state of ISCTL
        RET
```

只要 RXACT 和 ENCNT 复位，定时器/计数器就停止。将 RXACT 置位以允许下降沿检测开始进入接收状态。P0.1 端口上的起始位第一个边沿将边沿检测锁存器的输出置位，它将置位到下一个 RXACT 复位。

一旦边沿检测锁存器置位，定时器开始工作。第一个定时时间开始，经过程序设定的时间，P0.1 的逻辑电平锁存到 RXD 锁存器中。此后中断请求发生。

对第一位的中断处理是可选的，可以测试起始位是否存在。缺少起始位时接收被中止。接收周期继续进行，下一个定时需要在预置数寄存器中装入适当的值。

后续各位在中断程序中的处理都相同的。

中断服务程序中应处理：

- 存储 RXD 位信息
- 准备下一位定时
- 可选：更新校验信息
  - 当校验出错时决定程序流向
  - 寻找地址位信息
- 如接收位应是停止位，作停止位测试。

注意：UART 协议，LSB/MSB 顺序
异步通信中首先发送最低位，为了接收时顺序正确，用 RRC 指令收集数据。

```
; ---------------- Receive interrupt handler -------------------------------
RCINTRPT    PUSH    R5      ; Receiver interrupt routine
                            ; R5 is used temporary as pointer
                            ; of receive bit
            MOV.B   &RCstatus,R5
            BR      RCTAB(R5)
;
RCTAB       .Word   RCstat0 ; Receive start bit
                            ; set receive time bit0/LSB, 13ACLK
            .Word   RCstat1 ; Receive bit 0 ; set receive time bit1, 14ACLK
            .Word   RCstat1 ; Receive bit 1 ; set receive time bit2,
            .Word   RCstat2 ; Receive bit 2 ; set receive time bit3
            .Word   RCstat1 ; Receive bit 3 ; set receive time bit4
            .Word   RCstat1 ; Receive bit 4 ; set receive time bit5
            .Word   RCstat2 ; Receive bit 5 ; set receive time bit6
            .Word   RCstat1 ; Receive bit 6 ; set receive time bit7/MSB
            .Word   RCstat1 ; Receive bit 7 ; set receive time parity bit
            .Word   RCstat2 ; Receive parity bit
                            ; set receive time stop bit 1
            .Word   RCstop1 ; Receive stop bit 1
                            ; set receive time stop bit 2
            .Word   RCstop2 ; Receive stop bit 2
                            ; set receive time stop bit 2
            .Word   RCCmpl  ; Frame received
; ---------------- Receive start bit: Test for space ------------------------
RCstat0     BIT.B   #RXD,&TCCTL      ; Check start bit
            JC      RCError          ; Error: start bit is Mark not Space
            MOV.B   #BitTime2,&TCPLD ; Start bit fine, load pre-load register
            JMP     RCRET
;
RCstat2     MOV.B   #BitTime2,&TCPLD ; Load pre-load register with bit time 2
            JMP     RCBit
RCstat1     MOV.B   #BitTime1,&TCPLD ; Load pre-load register with bit time 1
RCBit       BIT.B   #RXD,&TCCTL      ; RXD bit -> Carry bit
            JNC     RCRET            ; RXD bit = Carry bit = 0 ?, Yes, jump
            RRC     RCData           ; RXD bit -> MSB, Negative bit
            INC.B   &Parity          ; RXD bit = 1, increment '1'-counter
            JMP     RCRET1
RCRET       RRC     RCData           ; RXD bit -> MSB, Negative bit
RCRET1      INCD.B  &RCstatus
RCCmpl      POP     R5
            RETI
;
; Parity bit was received just like all other bits.
```

```
; During first stop bit parity is tested
;
RCstop1     BIT.B   #1,&Parity        ; Check parity bit. Bit must be zero.
            JNZ     RCError           ; Parity bit false.
;
RCstop2     MOV.B   #BitTime1,&TCPLD ; Load pre-load register with bit time 1
            BIT.B   #RXD,&TCCTL       ; Check stop bit for Mark
            JNZ     RCRET             ; Stop bit is Mark -> Ok
;
; Error handling: a new start is tried
RCError     POP     R5
            CALL    RCINIT            ; Initialize receive routine again
            RETI
```

10.3 看门狗定时器

看门狗定时器(WDT)的主要功能是当程序发生错误时执行一个受控的系统重启动。如果超过了选定的定时时间，发生系统复位。如果应用中不需要此功能，可以把它当作定时器。当选定定时时间到达后将产生中断。

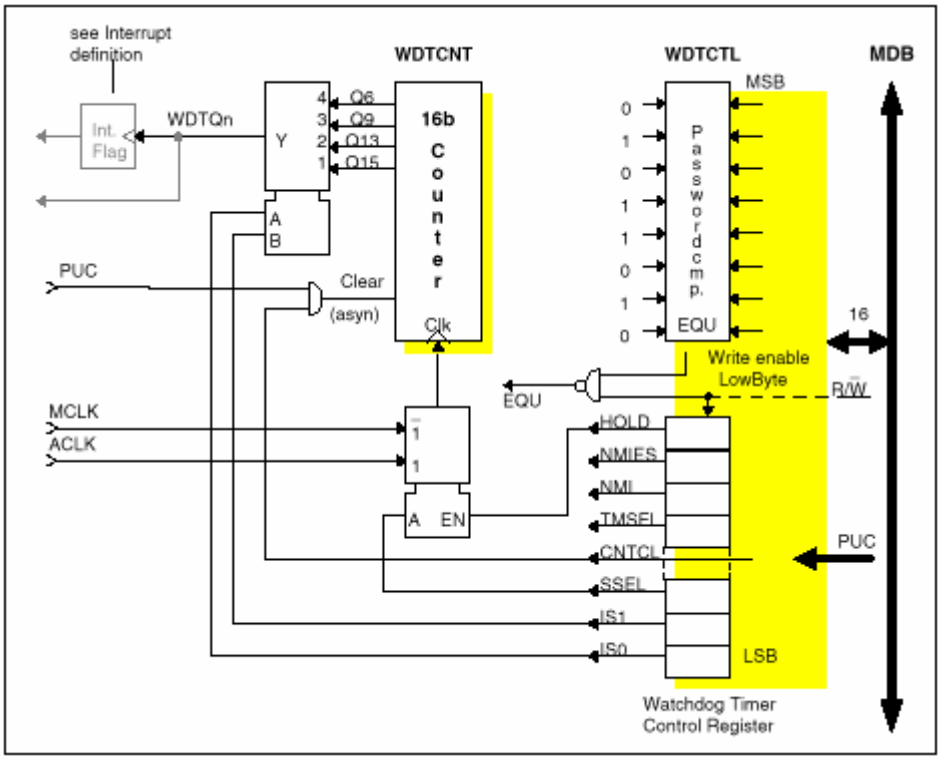


图 10.16：看门狗定时器原理图

特性：

- 8 种软件可选定时时间
- 2 种工作模式：看门狗或定时器
- 在看门狗模式下超过定时后产生系统复位，在定时模式下产生中断请求。
- 出于安全原因，对 WDT 控制寄存器的写操作需用口令。
- 用停止模式支持超低功耗特点。

10.3.1 看门狗定时寄存器

看门狗定时器的计数器 WDTCNT 是 16 位增计数器，它不能直接用软件访问。WDTCNT 的控制是通过 WDTCTL，它是位于字地址 0120h 的低位字节的 16 位读/写寄存器。所有读写操作都要用字指令，即不带后缀或用后缀“.W”。在两种工作模式（看门狗或定时器）中，只有含有正确口令的指令才能写入 WDTCTL。

WDTCTL（0120h）

7							0
HOLD	NMIES	NMI	TMSEL	CNCTT	SSEL	IS1	IS0
rw-0	rw-0	rw-0	rw-0	r0(w)	rw-0	rw-0	rw-0

图 10.17：看门狗定时器控制寄存器

位 0,1： IS0，IS1 位选择 WDTCNT 中 4 个输出之一。

设晶振频率=32768Hz，系统频率=1MHz，可以有以下定时时间：

ssel	IS1	IS0	定时(ms)	
0	1	1	0.064	t_{MCLK} x 2^6
0	1	0	0.5	t_{MCLK} x 2^9
1	1	1	1.9	t_{ACLK} x 2^6
0	0	1	8	t_{MCLK} x 2^{13}
1	1	0	16	t_{ACLK} x 2^9
0	0	0	32	t_{MCLK} x 2^{15} (PUC后的值)
1	0	1	250	t_{ACLK} x 2^{13}
1	0	0	1000	t_{ACLK} x 2^{15}

位 2：SSEL 位选择 WDTCNT 的时钟源

SSEL=0：WDTCNT 以系统频率为时钟

SSEL=1：WDTCNT 以 ACLK 为时钟，即晶振频率(32768Hz)

位 3：CNTCL 位：在两种模式中，写入“1”使 WDTCNT 从 00000h 重启动，读出无定义。

位 4：TMSEL 位选择工作模式：看门狗或定时器

TMSEL=0：看门狗模式

TMSEL=1：定时器模式

位 5：NMI 位选择 RST*/NMI 引脚功能。在 PUC 后它被复位。

NMI=0：RST*/NMI 输入作为复位输入

只要 RST*/NMI 引脚保持低，内部 PUC 信号一直有效（电平触发）。

NMI=1：RST*/NMI 输入作为边沿敏感的非屏蔽中断输入。

位 6：如果选定 NMI 功能，NMIES 选择 RST*/NMI 输入的有效触发沿，PUC 后复位。

NMIES=0：上升沿触发 NMI 中断。

NMIES=1：下降沿触发 NMI 中断。

位 7：HOLD 位停止看门狗定时器的全部工作，为了达到超低功耗这是必须的。时钟多路切换禁止并且使计数停止增加，这一状态一直保持到 HOLD 复位，然后工作继续进行。PUC 后 HOLD 复位。

HOLD=0：所有功能激活。

HOLD=1：时钟和计数停止。

访问 WDTCTL 看门狗定时器的控制寄存器

- 读操作

读 WDTCTL 不受口令限止。简单地对字地址 0120h 访问可进行读操作。它的低字节含有 WDTCTL 的值。高字节是 069h。高位字节被选为 069h，从而限止了指令对 WDTCTL 寄存器的改变。

读 WDTCTL：

WDTCTL（0120h）

15							8	7							0
0	1	1	0	1	0	0	1	读出数据							
r	r	r	r	r	r	r	r	rw-x,(w)							

- 写操作

只有在高字节中写入正确的口令才能对 WDTCL 进行写操作。对 0120h 写操作将改变 WDTCTL 寄存器。低字节含写入 WDTCTL 的数据，高字节应是口令 05Ah。如果高位中写入除 05Ah 外的其他值将产生系统复位。

写 WDTCTL：

WDTCTL（0120h）

15							8	7							0
0	1	0	1	1	0	1	0	写入数据							
(w)	(w)	(w)	(w)	(w)	(w)	(w)	(w)	rw-x,(w)							

10.3.2 看门狗定时器中断控制功能

看门狗定时器用到 SFR 地址的两位：

- WDT 中断标志 WDTIFG(位于 IFG1.0，初始状态是复位)
- WDT 中断允许 WDTIE(位于 IE1.0，初始状态是复位)

当上电或 RST*/NMI 引脚复位时，WDT 中断标志复位。这一信号被称作 POR。看门狗中断标志指示是看门狗还是上电复位引起 PUC。中断向量地址在 0FFFEh，与中断允许无关。

看门狗定时器工作在两种模式。当 WDT 被设为看门狗模式，看门狗时间溢出或口令错误将触发 PUC 信号，它自动将整个系统中相应的寄存器位复位。而对于 WDTCNT 的各位将影响系统配置，使 WDT 置为看门狗模式，使 RST*/NMI 引脚被设为复位工作结构。

当 WDT 被设为定时器模式，在设定的定时时间到达后 WDTIFG 标志置位，它请求标准的中断服务。WDT 是单源中断，当系统处理中断时中断标志被自动复位。允许位保持不变。WDT 中断允许位和通用中断允许 GIE 位应置为允许中断状态。定时模式的中断向量地址与看门狗模式的不同。

10.3.3 看门狗定时器操作

看门狗定时器模块可置成两种模式：看门狗模式和间隔定时器模式。

看门狗模式

上电或系统复位后，看门狗定时器自动进入看门狗模式，此时在看门狗控制寄存器 WDTCTL 中各位和看门狗计数器 WDTCNT 中各位复位。WDTCTL 寄存器中的初始状态导致定时时间为 32ms(在系统时钟为 1MHz 时)。由于系统时钟发生器中的数字控制振荡器 DCO 置为以最低频率工作，软件有大约 32600 个周期去响应这一重要事件。初始状态选为 WDTCNT 在系统频率下工作，允许应用程序开始运行在看门狗时间在定时时间的中间。

用作看门狗模式时，软件必须周期性地在 WDTCTL 的 CNTCL 位中写“1”来使 WDTCNT 复位以防止超过选定的定时时间。如果因程序发生问题，计数器不再被复位而超过了定时时间，将产生复位，系统上电清除(PUC)激活。系统从上电复位的地址重启动。可通过检测 SFR 中的 IFG1 的 0 位来判断复位原因。定时时间可以通过 SSEL、IS0 和 IS1 位来选定。

定时器模式

WDTCTL 的 TMSEL 位置位选择定时器模式。这一模式提供选定时间周期性中断。定时时间可以通过软件对 WDTCTL 寄存器的 CNTCL 位置位来开始。

注意： 看门狗定时器，改变定时时间

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在一条指令中完成，例如 MOV #05A0Ah，&WDTCTL。如果先后分别进行清除和定时时间选择可能立即引起不可预料的系统复位或中断。

在正常工作时改变时钟源可能导致 WDTCNT 额外的计数时钟。

工作在低功耗模式

系统时钟发生器可运行在 5 种不同的模式下。其中有三种 MCLK 和 ACLK 是活动的。一种只有 ACLK 活动，还有一种 ACLK 和 MCLK 都不活动。

应用程序要求设置结合不同 ACLK 和 MCLK 条件下硬件响应的看门狗定时器处理。

CPUOFF 模式：

程序运行停止。软件必须定义在此模式期间的工作条件。

MCLKOFF 模式/LMP2、3：

ACLK 活动而 MCLK 停止。选择 ACLK(32768Hz)，看门狗定时器继续工作，根据选定的工作模式通过系统复位或定时器中断(如果允许)来唤醒 CPU。选择 MCLK，看门狗定时器停止工作直到 MCLK 重新启动。

OSCOff 模式/LPM4：

MCLK 和 ACLK 都停止，看门狗定时器停止计数直到系统重启动。根据应用需要，程序可以在进入 OSCOff 模式将计数器复位。

停止功能支持低功耗模式。当应用程序采用了多种低功耗模式，看门狗定时器也许应该停止。

软件实例

```
; After RESET or power-up, the WDTCTL reg and WDTCNT are cleared and the initial
; operating conditions are watchdog mode with a time interval of 32 ms.
;
; Constant definitions:
;
WDTCTL  .EQU    0120h   ; Address of Watchdog timer
WDTPW   .EQU    05A00h  ; Password
T250MS  .EQU    5       ; SSEL, IS0, IS1 set to 250 ms
T05MS   .EQU    2       ; SSEL, IS0, IS1 set to 0.5 ms
CNTCL   .EQU    8       ; Bit position to reset WDTCNT
TMSEL   .EQU    010h    ; Bit position to select timer mode
;
; As long as watchdog mode is selected, watchdog reset has to be done periodically
; through a instruction e.g.:
;
        ........
        ........
        MOV     #WDTPW+CNTCL,&WDTCTL
        ........
        ........
;
; To change to timer mode and a time interval of 250 ms, the following instruction
; sequence can be used:
;
        MOV     #WDTPW+CNTCL+TMSEL+T250MS,&WDTCTL
                        ; Clear WDTCNT and select 250 ms and timer mode
        ........
        ........
; Note: The time interval and clear of WDTCNT should be modified within one
; instruction to avoid unexpected reset or interrupt
;
; Switching back to watchdog mode and a time interval of 0.5 ms is performed by:
;
        ........
        ........
        MOV     #WDTPW+CNTCL,&WDTCTL ; Reset WDT counter
;
        MOV     #WDTPW+T05MS,&WDTCTL ; Select watchdog mode and 0.5 ms
        ........
        ........
```

10.4 8 位脉宽调制定时器 PWM

用 8 位定时器计数器，PWM 外围模块产生一个占空比可以是 0-100% 的方波输出。占空比由 8 位占空比控制寄存器 PWMDT 来确定

PWM 定时器模块有以下特性：

- 可选 8 个时钟源
- 占空比为 0 至 100%，分辩率为 1/254
- 可以以正或负逻辑输出

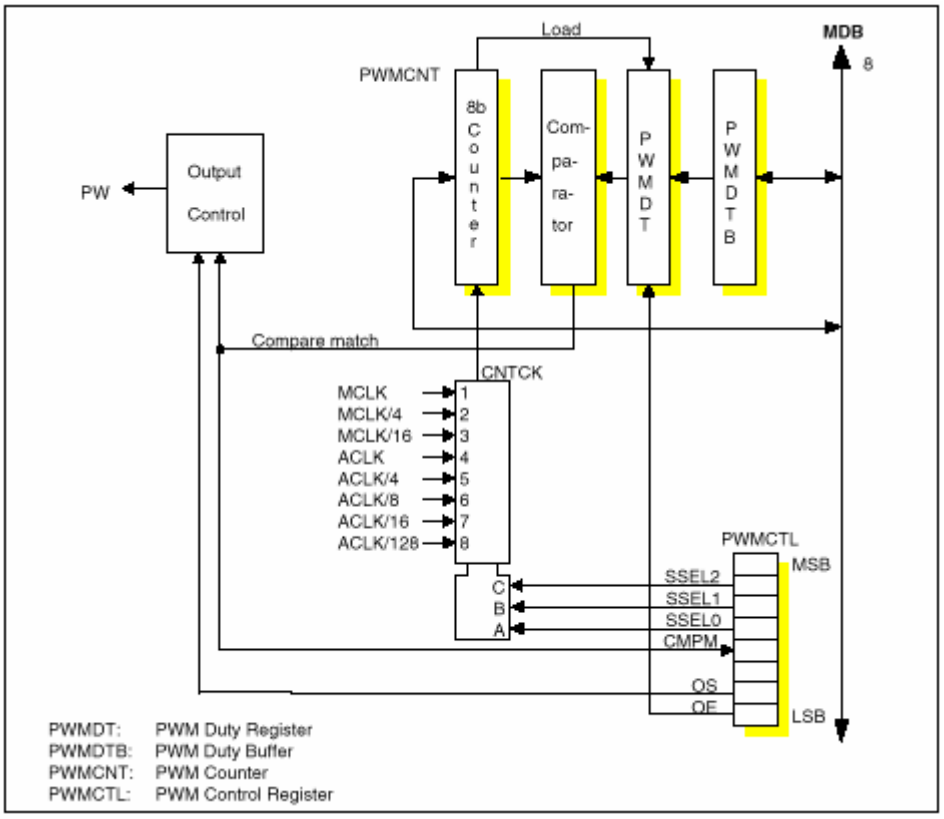


图 10.18：PWM 定时器原理框图

10.4.1 操作

PWM 定时器工作描述假设输出极性控制信号 OS 复位。PWMDT 寄存器的值表示 PWM 输出为高时脉冲的个数。

当 OE=0，定时器计数器保持为 00h，PWM 输出复位。任何值写入 PWMDT 立即生效。

当 OE=1，定时器计数器开始增加，PWM 输出变高（图中 b 状态）。

当计数器到达 PWMDT 的值，在下一个时钟脉冲时 PWM 输出变低（图中 c 状态）。

如果 PWMDT 值改变（图中写入 M 值）。在定时器计数由 0FDh 到 00h（图中 d 状态）后新的计数值变为有效。

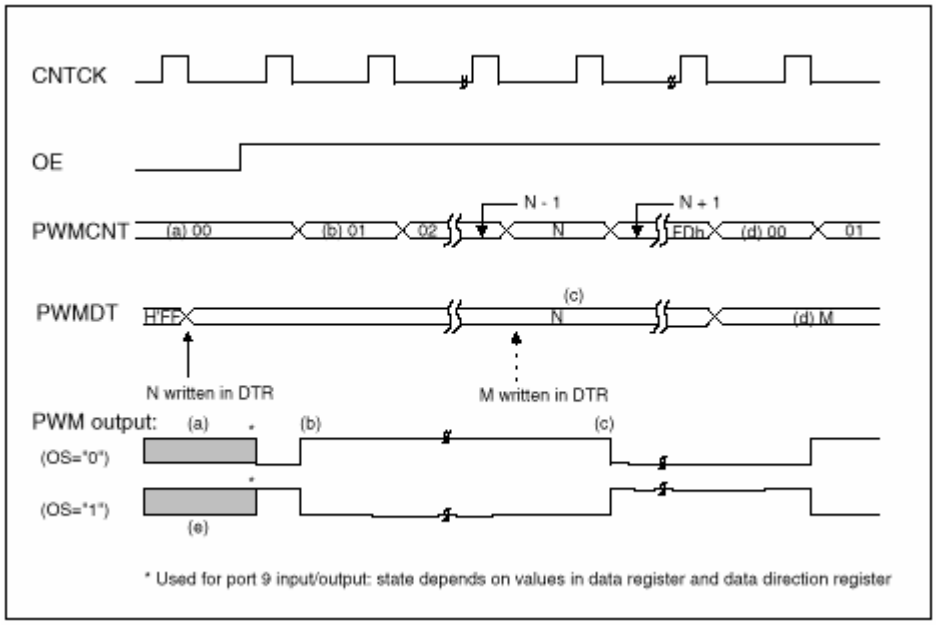


图 10.19：PWM 时序原理

PWMCTL 寄存器中的控制标志 OS 定义 PWM 输出的极性。
当 OS=0，PWMDT 中的值表示 PWM 输出为高的 PWM 计数时钟脉冲。
当 OS=1，PWMDT 中的值表示 PWM 输出为零前 PWM 计数时钟脉冲。

10.4.2 PWM 寄存器说明

通过 8 位结构 MDB 和 MAB 的操作来控制 PWM 定时器模块。它必须用字节指令来访问。

寄存器	缩写	寄存器类型	地址	初始状态
PWM 定时器控制寄存器	PWMCTL.1	读写	58h	复位
PWM 占空比缓存	PWMDTB.1	读写	59h	复位
PWM 占空比寄存器	PWMDTR.1	读写	5Ah	复位
PWM 定时器计数器	PWMCNT.1	读(写)*	5Bh	复位
PWM 定时器控制寄存器	PWMCTL.2	读写	5Ch	复位
PWM 占空比缓存	PWMDTB.2	读写	5Dh	复位
PWM 占空比缓存	PWMDTR.2	读写	5Eh	复位
PWM 定时器计数器	PWMCNT.2	读(写)*	5Fh	复位

注意：改变定时器的计数
定时器的计数器是读写寄存器，但写功能只用于测试。
实际应用程序不应写这些寄存器。

定时器的计数器 PWMCNT

PWMCNT（05Bh 或 05Fh）

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

PWMCNT 是 8 位增计数器，PWMCTL 中的输出允许位 OE 置位，计数器开始按时钟选择位 SSEL2 至 SSEL0 选定的内部时钟源脉冲计数。从 00h 计数到 0FDh 后计数又从 00h 开始。

PWM 定时器的计数器可读写。但是写功能只用于测试，应用程序不能对这些定时器计数器写入。因为这将产生不可预料的效果。

发生 PUC 时，PWM 定时器计数器初始化为 00h，同时 OE 位复位。

占空比缓存寄存器 PWMDTB

PWMDTB（059h 或 05Dh）

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

占空比缓存寄存器保存占空比因子。当计数器从 0FDh 计数到 00h 时，这一占空比因子写入占空比寄存器

PWMDTB 在 OSCOff 模式下和系统复位时置为 00h

注意：改变 PWM 占空比因子
只有在 PWMDTB 中写入新值才能改变占空比。任何直接对占空比寄存器的写操作将导致在运行周期中产生一个随机占空比。下一周期会用已进入占空比缓存的新占空比因子运行。

占空比寄存器 PWMDT

PWMDT（05Ah 或 05Eh）

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

占空比寄存器确定输出脉冲的占空比。可以 1/254 的分辨率选择从 0 到 100% 的任何占空比。在 PWMDT 中写 0h 给定 0% 的占空比，写 127(07Fh)给定 50% 占空比，写 254(0FEh)给定 100% 占空比。

定时器计数器连续与占空比寄存器的内容作比较。如 PWMDT 值不为 0，当计数器从 00h 增到 01h 时 PWM 输出置位。当计数器增到 PWMDT 的值时，PWM 输出又回到零。如果 PWMDT 值是 0(占空比 0%)，PWM 输出恒定为 0。

PWMDT 是双缓存的。当定时器计数器在运行时对 PWMDTB 写入新值，只有当计数器发生从 0FDh 到 00h 的变化后才变为有效。PUC 后，OE 位复位，定时器计数器停止，这时新值写入立即生效。对 PWMDT 读操作，读出的是当前的有效值。

占空比寄存器在发生系统复位和在 OSCOff 模式下被初始化成 0FFh。

PWM 定时器控制寄存器 PWMCTL

PWMCTL（058h 或 05Ch）

7							0
---	SSEL2	SSEL1	SSEL0	CMPM	---	OS	OE
rw-0	rw-0	rw-0	rw-0	r	rw-0	rw-0	rw-0

PWM 控制寄存器是 8 位可读写寄存器，它选择时钟源并控制 PWM 的输出。

位 0：输出允许(OE)，允许 PWM 计数器和 PWM 输出

OE=0：PWM 输出禁止。PWMTC 清除为 00h 并停止。

OE=1：PWM 输出允许。PWMTC 运行。

位 1：输出选择(OS)，选择正或负逻辑信号作为 PWM 输出。

OS=0：正逻辑：正向脉冲，1=高(初始值)

OS=1：负逻辑：负向脉冲，1=低

位 2、7：保留；不能改变并且读出为零。

位 3：CMPM：只读，可检测比较信号输出。只要定时器计数器 PWMCNT 和占空比寄存器 PWMDT 相同，它始终读出为“1”。

位 4-6：时钟选择。选择 8 个由 MCLK 和 ACLK 分频得来的时钟源之一。

SSEL2	SSEL1	SSEL0	时钟源
0	0	0	MCLK
0	0	1	MCLK/4
0	1	0	MCLK/16
0	1	1	ACLK
1	0	0	ACLK/4
1	0	1	ACLK/8
1	1	0	ACLK/16
1	1	1	ACLK/128

根据时钟源频率、分辨率、周期，PWM 的输出频率可以计算出来。

分辨率 = 1/时钟源频率

PWM 周期 = 分辨率 * 254 = 254/时钟源频率

PWM 频率 = 1/PWM 周期 = 时钟源频率/254

占空比 = PWMDT/254

11．Timer_A

本章介绍 MSP430 系统中的通用 16 位 Timer_A 的基本功能。

目录 页号

11.1 Timer_A 的操作

16 位 Timer_A 的主要功能模块有：

- 一个可连续增计数至预定值并返回 0 的计数器，也可使它停止。
- 软件可选择时钟源。
- 选定的时钟源可被 1、2、4 和 8 分频。
- 5 个捕获/比较寄存器，每个有独立的捕获事件：硬件或软件控制 2 个捕获信号。
- 2 个输出模块，支持脉宽调制要求。

定时器控制寄存器 TACTL 的各位控制 Timer_A 的配置。这个寄存器定义 16 位定时器的基本操作。可选择原始频率或分频后的输入时钟源及 4 种工作模式。另外还包括清除功能和溢出中断控制位。定时器计数到 0000h，定义为定时器溢出。此定义与定时器是增计数或减计数无关。

5 个捕获/比较寄存器的操作相同，它们通过各自的控制寄存器作配置。

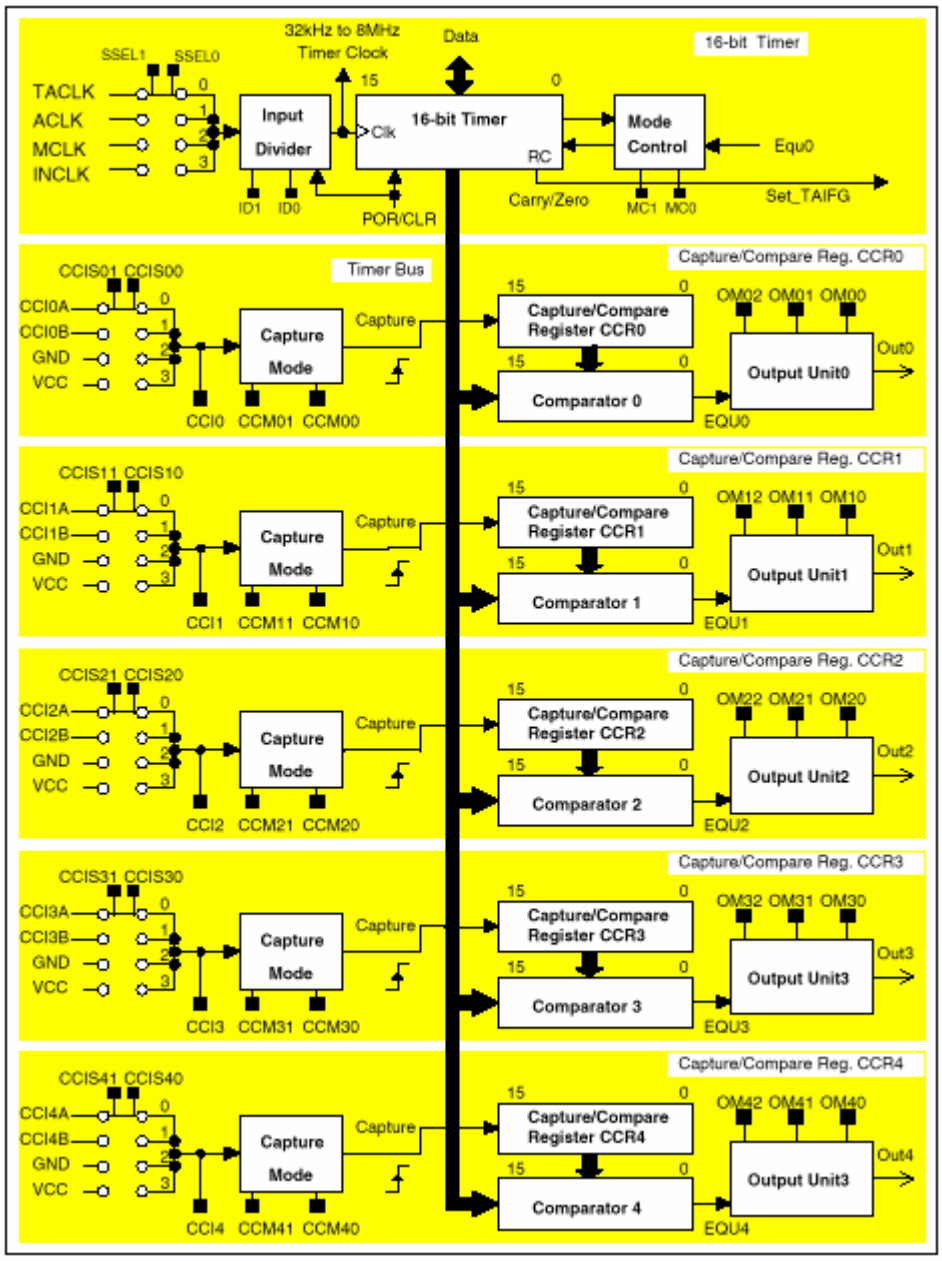


图 11.1：Timer_A 原理图

11.1.1 定时器操作

通过定义控制寄存器 TACTL 中的 2 个控制位 MC1 和 MC0，加上捕获/比较 0 模块中的比较器输出 EQU0 信号，可以提供 16 位定时器工作在四种不同模式。TACTL 的 SSEL1 和 SSEL0 选择定时器时钟源。选定的时钟源直接或经 2、4、8 分频进入 16 位定时器，时钟源可来自内部时钟也可由外部提供。

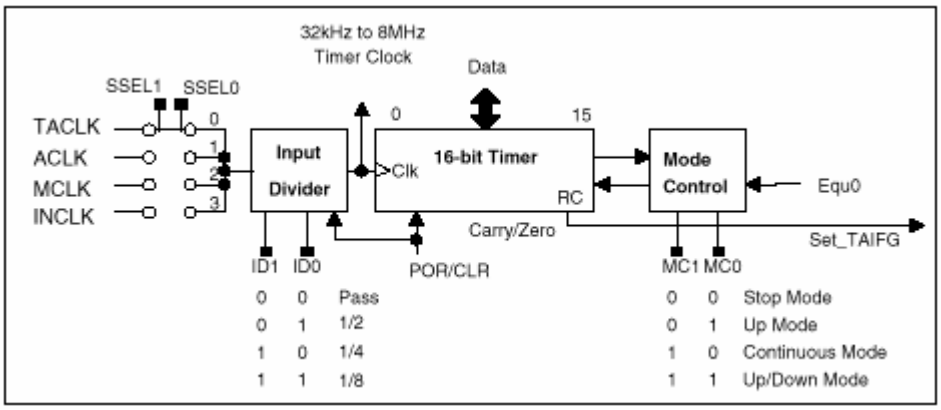


图 11.2：16 位定时器原理图

时钟源选择和输入分频。

时钟源由 SSEL0 和 SSEL1 来选定。多路切换的输出选定的信号和电平。两个控制位选择时钟源加到输入分频器有一个短暂中间状态。时钟源的改变能使输入分频器接收到额外的时钟信号。当 VCC 上电或 RST*/NMI 引脚检测到复位状态（即发生 POR 信号）或定时器通过 CLR 位复位时，输入分频器复位。CLR 位于控制寄存器 TACTL。定时器修改时，即使是写入 0，输入分频器始终保持在现有状态。正常操作时，输入分频器的现有状态对于软件是不可见的。

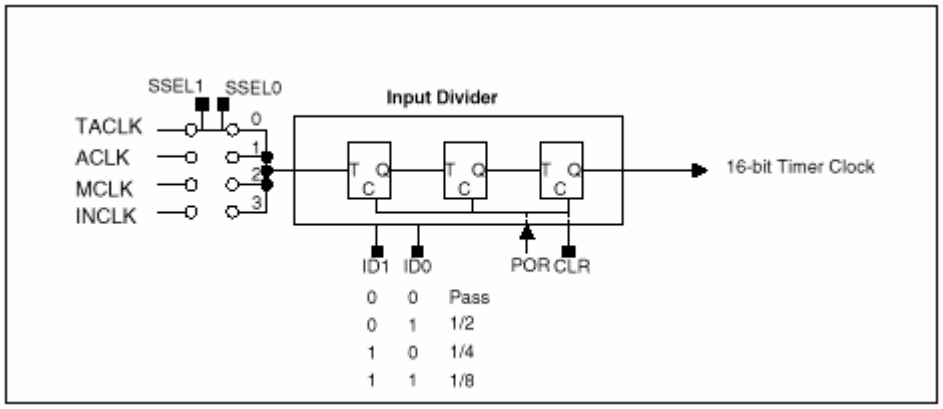


图 11.3：时钟选择和输入分频器原理图

工作模式控制和 16 位定时器

16 位定时器随每个时钟上升沿作增计数或减计数。通过标准的外围模式访问可以用软件对它读写。用 MC1 和 MC0 位来选择不同的工作模式。

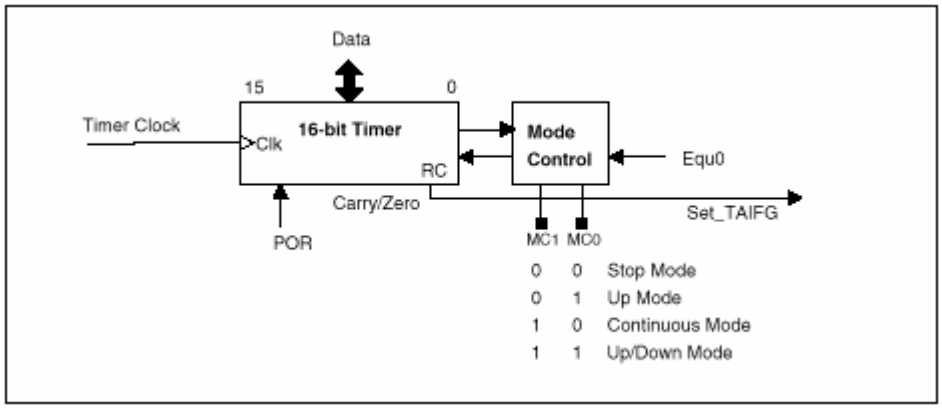


图 11.4：定时器原理图

在定时器时钟的低电平状态完成所有操作准备，并在后续的上升沿执行。绝大多数将分别讨论的特定状况是基于这一点。这一特性的一个例子，如果计数器计数到 X，此后捕获/比较寄存器也装入这一数据 X，这时比较会失败。

有 4 种定时器工作模式：

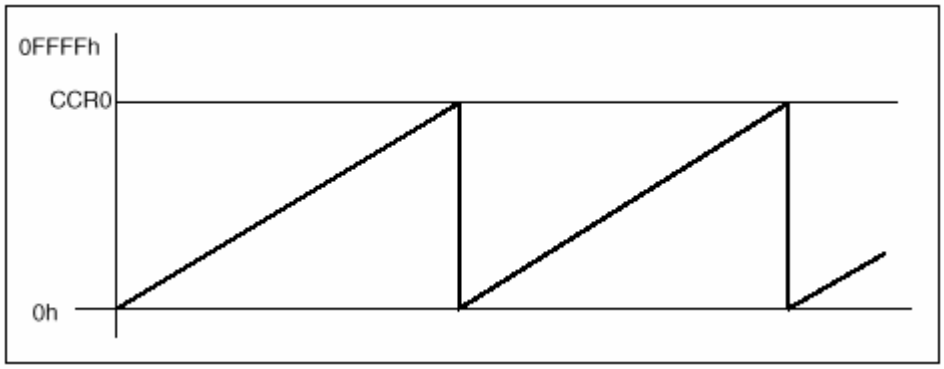
模式控制		模式	含义
MC1	MC0		
0	0	停止	定时器暂停
0	1	增计数	定时器增计数至等于比较寄存器 0 的值
1	0	连续	定时器连续增计数
1	1	增/减计数	定时器增计数到等于比较寄存器 0 的值又减计数到 0

停止模式

定时器暂停。当暂停释放时，它根据选定的工作模式当前数值开始计数。计数方向与暂停时相同。它不产生复位，所有寄存器保存的内容都可用。

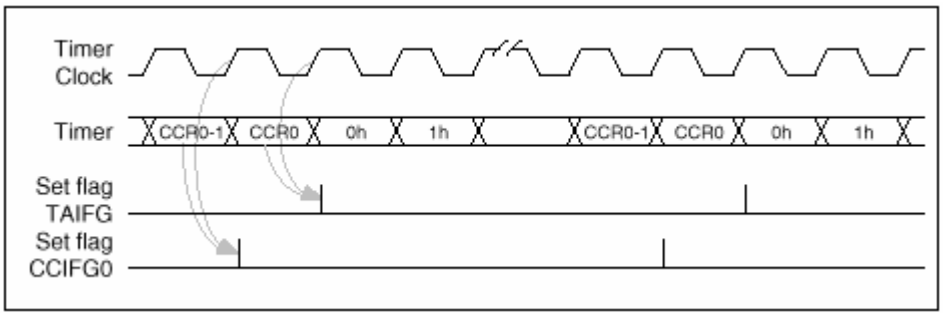
增计数模式

计数器增计数到比较寄存器 CCR0 的值。定时器从定时器寄存器的值开始计数，当计数值与比较寄存器 CCR0 值相等时，定时器复位并又从 0 重新计数。



电计

在增计数模式中，比较寄存器用作周期寄存器。当计数值等于或大于 CCR0 中值时，在下一个时钟计数器又回到 0。

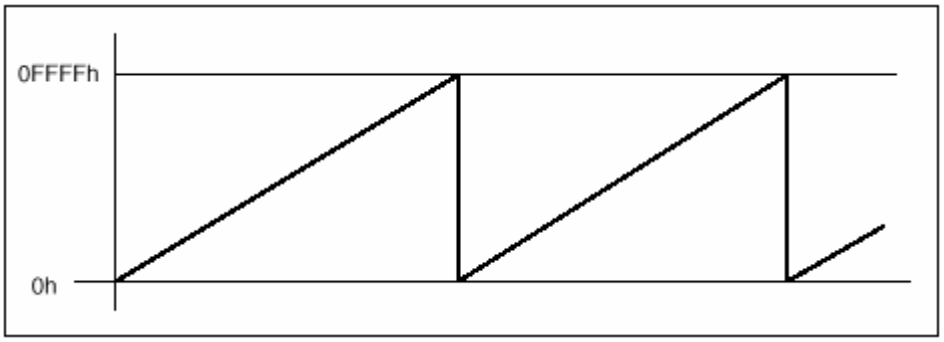


当定时器值等于 CCR0 的值，CCIFG0 标志置位。定时器从 CCR0 值计数到 0 时，TAIFG 标志置位。所有中断标志置位与各自的中断允许位无关。

如果通用中断允许位和相应的中断允许位置位，这时将产生中断请求。

连续模式

定时器从寄存器中当前值开始计数。当计到 0FFFFh 后又从 0 重新计数。



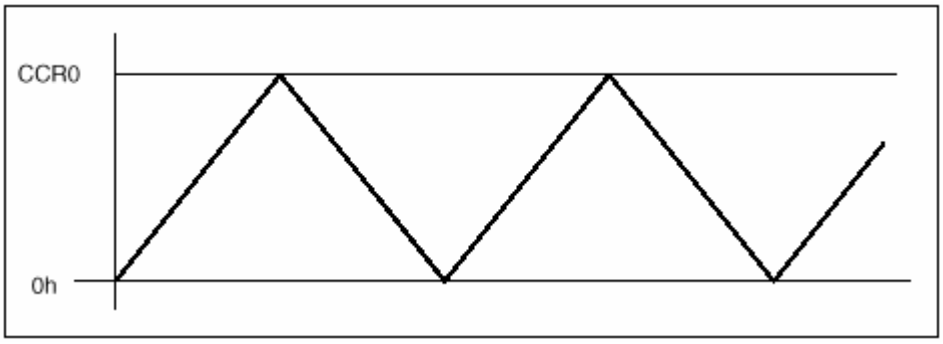
如果需要多个定时信号可用连续计数模式。中断处理时间要比相应的比较寄存器 CCRx 的时间长，这个时间差是所设时间（CCRx 中数据）与到下一个中断所需时间。

当定时器计数从 0FFFFh 计数到 0 时 TAIFG 置位。中断标志置位与相应的中断允许位无关。如果通用中断允许位和各自相应的中断允许位置位将产生中断请求。

捕获/比较寄存器 CCR0 工作方式与“连续模式”中的其他寄存器一样。

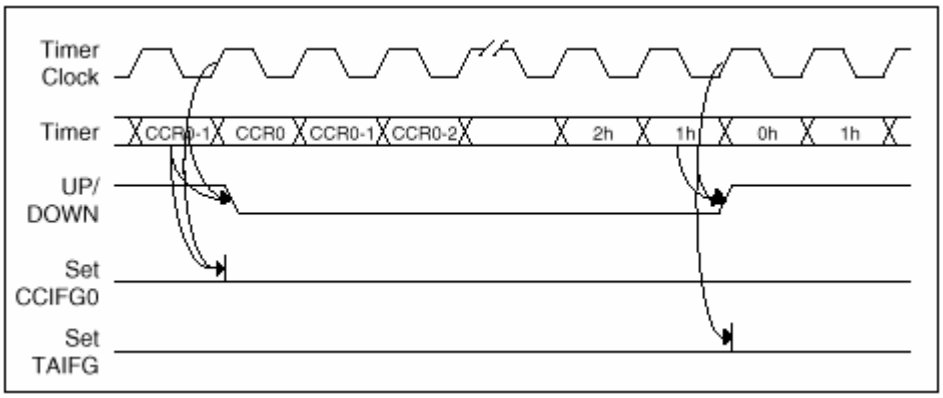
增减计数模式

定时器增计数到与 CCR0 的值相等，然后反向计数到 0。



计数方向由触发器 FF 锁存。FF 在 0000h 时置位使定时器增计数，当计数到等于 CCR0 的值时复位，使定时器锁存减计数模式。

计数周期由比较寄存器 CCR0 定义。它是 CCR0 寄存器值的两倍。



当定时器从“CCR1”增计数到“CCR0”时，中断标志 CCIFG0 置位。

当定时器从 0001h 减计数到 0000h 时，中断标志 TAIFG 置位。

捕获/比较模块

5 个相同的模块为实时处理提供了灵活的控制手段。任一模块寄存器都可用于捕获应用事件的发生时间，或产生定时间隔。如果相应的中断允许，每完成一个时间捕获或一次个定时间隔，捕获/比较模块将产生中断。CCTLX 中的模式位 CAPx 选择比较(CAPx 复位)或捕获(CAPx 置位)工作模式。在捕获模式下，控制字 CCTLX 中的 CCMx1 和 CCMx0 位定义产生捕获功能的条件：如未捕获，捕获将在上升沿、下降沿或两个沿发生。

中断允许位 CCIEx 和中断标志位 CCIFGx 用于捕获和比较模式中。发生捕获或比较事件，CCIFG 置位，控制位 CAPx 定义用于捕获或比较模式。

捕获输入 CCIxA 和 CCIxB 连接外部引脚或内部信号。不同的 MSP430 型号连到 CCIxA 和 CCIxB 的信号也不同。

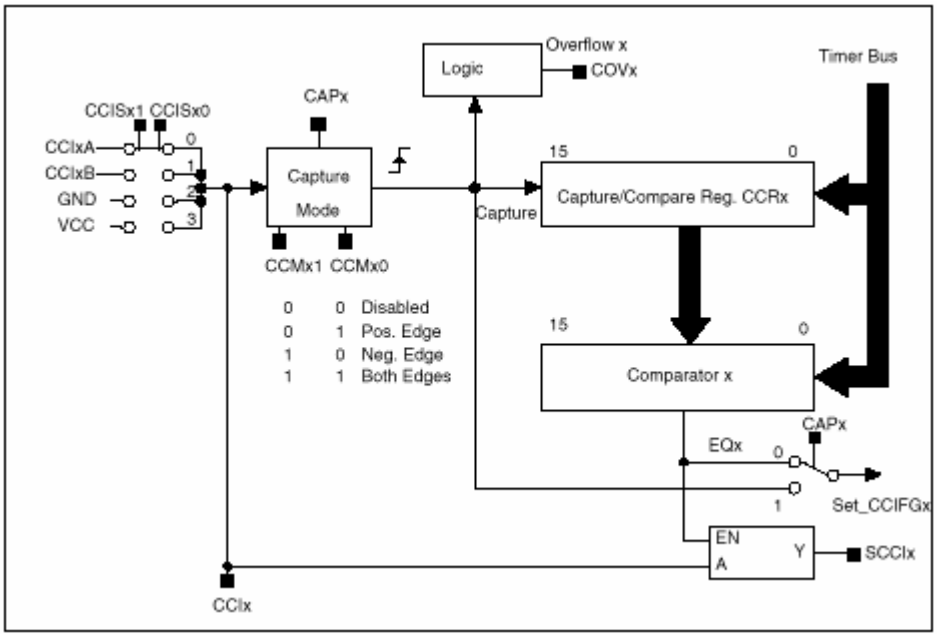
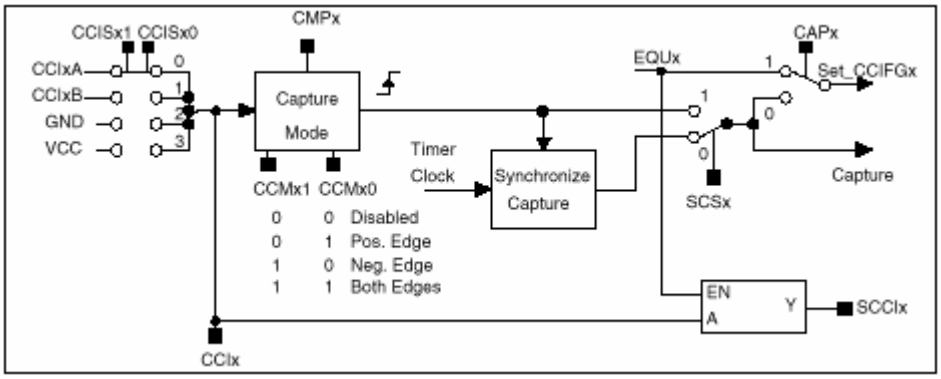
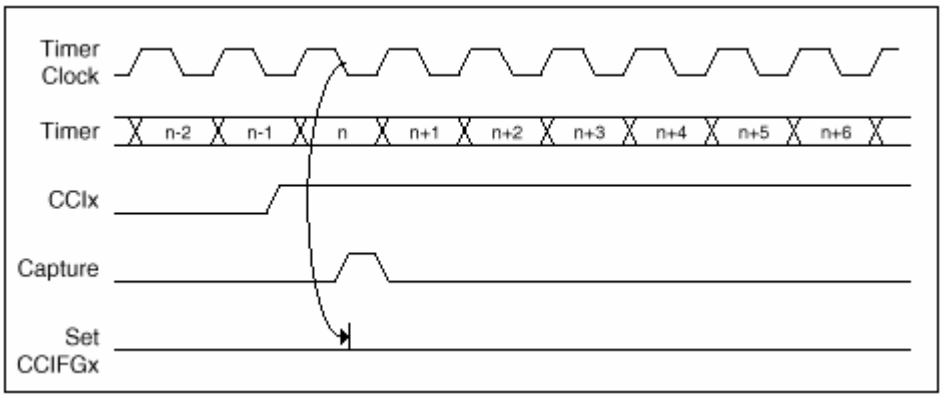


图 11.5：捕获/比较模块

由控制位 CCISx1 和 CCISx0 选择捕获逻辑的输入信号源。经过 CCIx 位或者与比较信号 EQUx 同步，可以用软件直接读取。同步位 SCCIx 支持串行通信协议的软件处理。捕获信号与定时器时钟可以异步。采用同步或异步捕获信号可支持不同的应用状况。



捕获信号与定时器时钟同步，它将中断标志置位并将定时器数值存入捕获寄存器。它们的同步可避免定时器数据和捕获信号的时间竞争。在捕获/比较控制寄存器 CCTLx 中的同步捕获信号位 SCSx 选择捕获信号模式。



非同步捕获信号支持低定时器时钟的应用。捕获事件与定时器时钟会产生时间竞争，因而导致无效的捕获数据。软件应验证数据并校正。

```
; Software example for the handling of asynchronous capture signals
;
; The data of the capture/compare register CCRx are taken by the software
; in the according interrupt routine - they are taken only after a CCRIFG
; was set. The timer clock is much slower than the system clock MCLK
;
CCRx_Int_hand   ...             ; Start of interrupt handler
                ...
                ...
                CMP &CCRx,&TAR  ; Test if the data CCRX = TAR
                JEQ Data_Valid
                MOV &TAR,&CCRx  ; The data in CCRx is wrong,
                                ; use the timer data
Data_Valid      ... ...         ; The data in CCRx are valid
                ...
                ...
                RETI
;
```

11.1.2 捕获模式

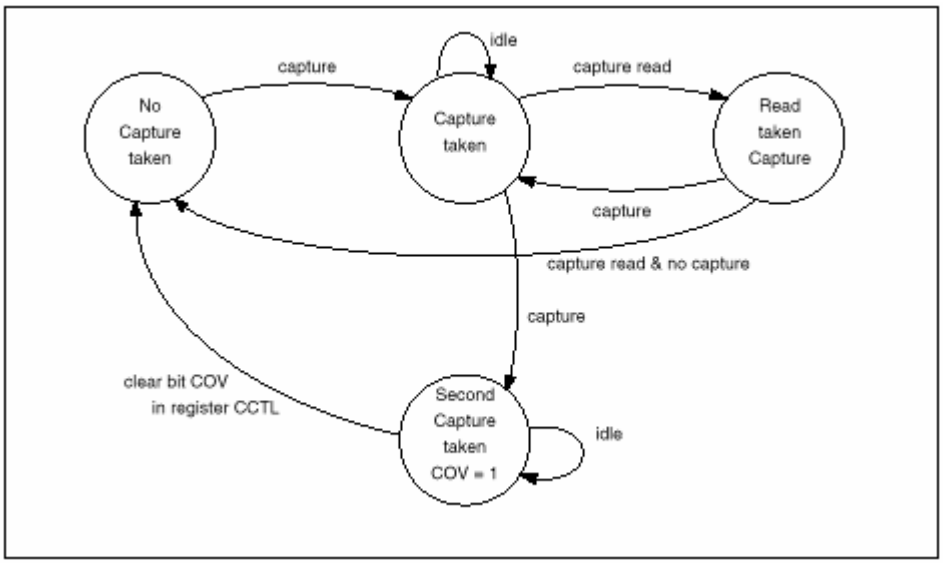
如果位于控制字 CCTLx 中的模式位 CAPx 置位，将选定捕获模式。捕获模式用于时间事件的精确定位。它可能用于速度计算或时间测量中。如果选定引脚发生选定脉冲触发沿(上升沿、下降沿或两者皆可)，定时器计数值将复制到捕获寄存器 CCRx 中。可选择 3 个不同的输入源：CCIxA，CCIxB 或者通过捕获/比较控制寄存器 CCTLx 中的 CCISx1/CCISx0 位选定的 CPU/软件信号。

捕获完成后：

- 控制字 CCTLx 中的中断标志 CCIFGx 置位。
- 如通用中断允许位 GIE 和相应的中断允许位 CCIEx 置位，将产生中断请求。

需用字指令来对捕获/比较寄存器 CCRx 操作。计数器的最新值复制并保存在其中。它提供溢出逻辑。复位表示在下一次捕获完成前捕获数据已被读取。当捕获数据还未读取时第二次捕获数据已锁存，CCTx 的溢出位 COVx 置位。这使得程序可从失去同步中恢复。

只有在另一次捕获发生前，捕获数据已完成读取，捕获才复位。如果没有完成读取溢出标志位置位。



需要用软件来复位溢出位 COVX。

```
; Software example for the handling of captured data looking for overflow
; condition
;
; The data of the capture/compare register CCRx are taken by the software
; and immediately with the next instruction the overflow bit is tested
; and a decision is made to proceed regularly or with an error handler
;
CCRx_Int_hand   ...                 ; Start of handler Interrupt
                ...
                ...
                MOV &CCRx,RAM_Buffer
                BIT #COV,&CCTLx
                JNZ Overflow_Hand
                ...
                ...
```

```
                ...
                RETI
Overflow_Hand   BIC #COV,&CCTLx      ; reset capture overflow flag
                                     ; get back to lost synchronization
                ...
                ...
;               RETI
```

注意：捕获与定时器暂停
当定时器暂停时捕获停止，顺序应是先停止捕获，再停止定时器，捕获功能重新开始时，顺序是：先开始捕获，再开始定时器。

11.1.3 比较模式

如果 CAPx 复位则选择比较模式。CAPx 位于控制字 CCTLx 中。这时所有捕获硬件停止工作。如果定时器计数值等于比较寄存器 x 中的值，那么：

- 位于控制字 CTLx 中的中断标志 CCIFGx 置位。
- 如果 GIE 和 CCIEx 置位将产生中断请求。
- EQUx 信号输出到输出单元 OUx 中。根据选定的输出模式，信号可以是置位、复位或将输出 OUTx 翻转。（如果 OUTMODx ＞ 0）

必须用字指令对捕获/比较寄存器 CCRx 访问。它保持比较值。捕获模式的溢出逻辑在此模式下无效。

当定时值大于等于 CCR0 值时，EQU0 信号为真。当定时器值等于相应的 CCR1 至 CCR4 的值时，EQU1 至 EQU4 信号为真。

11.1.4 输出单元

输出单元支持 PWM 或 DAC 的应用。按 3 个控制位选择的功能，捕获/比较寄存器的输出 EQU0 和 EQUx 控制输出逻辑。由对应于捕获/比较模块的 5 个输出单元 OU0 至 OU4 执行。控制位 OMx0、OMx1 和 OMx2 位于控制寄存器 CCTLx 中。

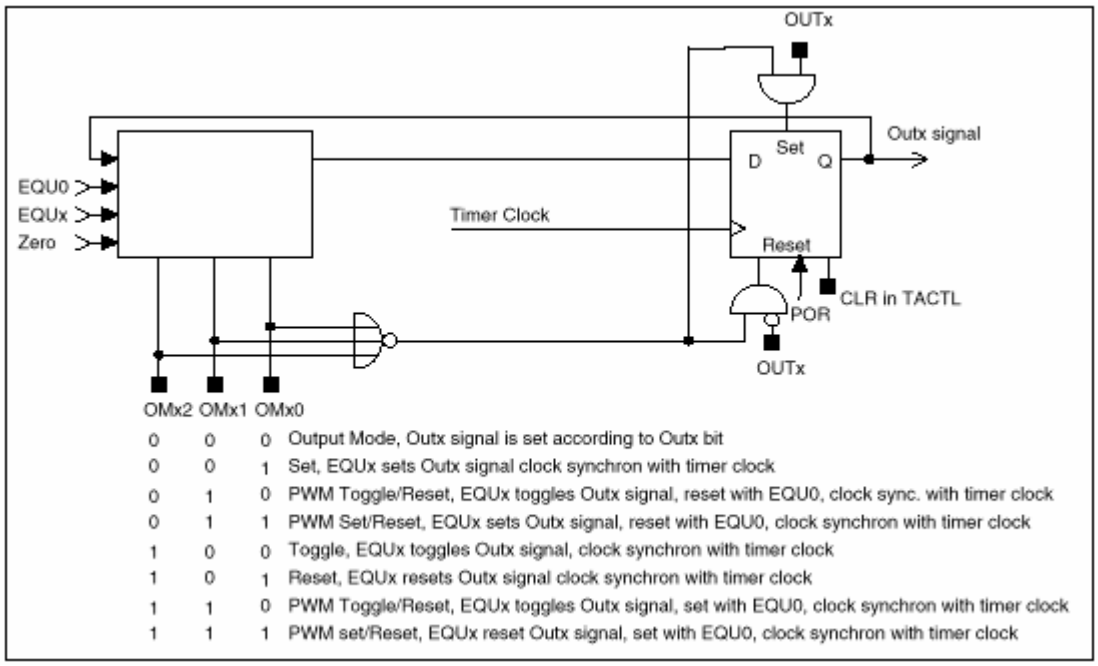


图 11.6：输出单元

如果由 OMx，OMx1，OMx2 选定了输出模式 0，则控制位 OUTx 决定 Outx 信号。输出从现有电平开始而与选定的模式无关。

增计数模式

当定时器增计数到 CCRx 并由 CCR0 到 0 时，OUTx 信号将变化。OUTx 信号根据选定的输出模式来改变。

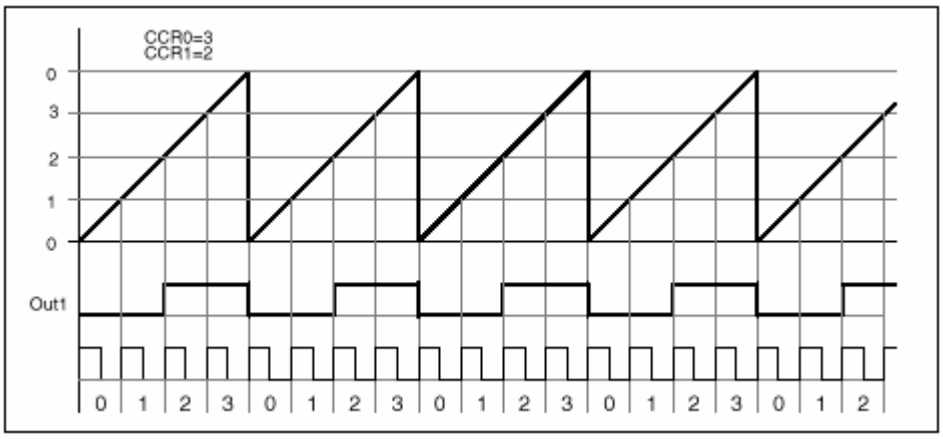


图 11.7：输出单元：在输出模式 3 时增计数模式实例

连续模式

当定时器计数到 CCRx 和计数到 CCR0 时，OUTx 信号变化。OUTx 信号根据选定的输出模式改变。

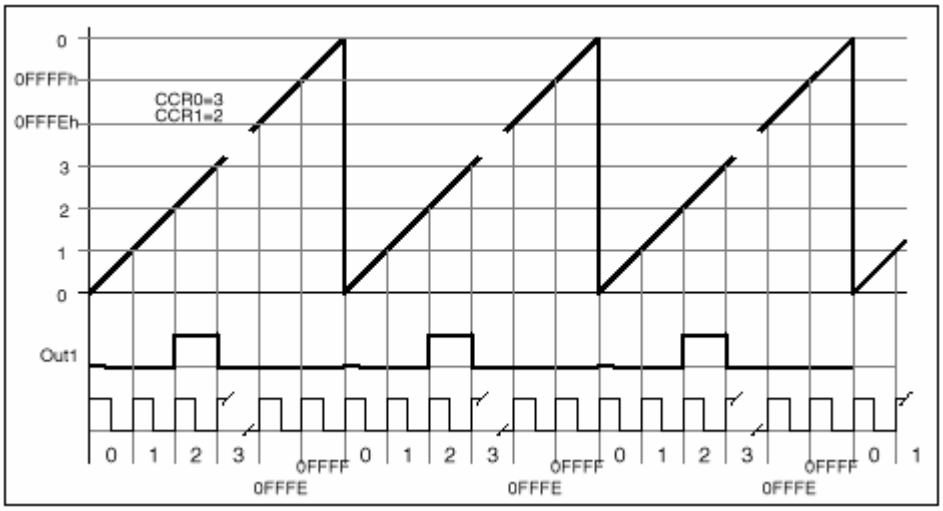


图 11.8：输出单元：输出模式 3 时连续模式实例

增/减计数模式

当定时器增计数到 CCRx 和减计数到 CCFx 时，OUTx 信号变化。OUTx 信号根据选定的输出模式改变。

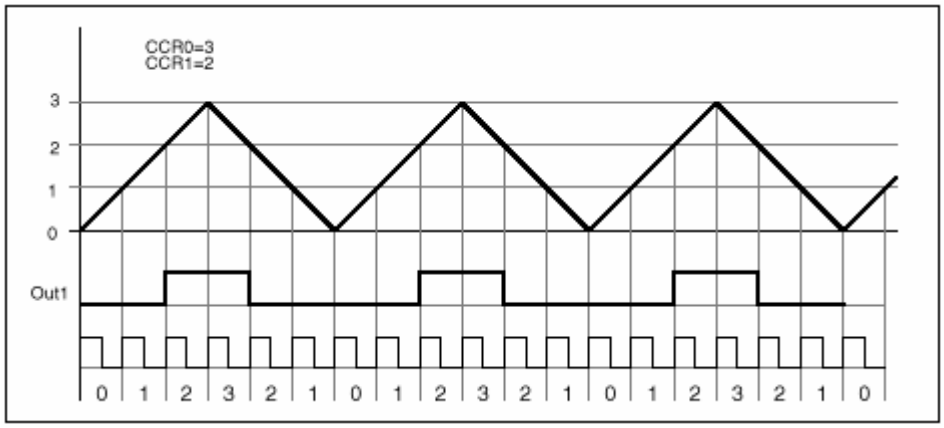


图 11.9：输出单元：输出模式 3 时增/减计数模式实例

11.2 Timer_A 的寄存器

16 位定时器模块的硬件是字结构的，必须用字指令来访问。

寄存器	缩写	寄存器类型	地址	初始状态
• Timer_A 控制寄存器	TACTL	读写	160h	POR 复位
• Timer_A 寄存器	TAR	读写	170h	POR 复位
• 捕获/比较控制寄存器 0	CCTL0	读写	162h	POR 复位
• 捕获/比较寄存器 0	CCR0	读写	172h	POR 复位
• 捕获/比较控制寄存器 1	CCTL1	读写	164h	POR 复位
• 捕获/比较寄存器 1	CCR1	读写	174h	POR 复位
• 捕获/比较控制寄存器 2	CCTL2	读写	166h	POR 复位
• 捕获/比较寄存器 2	CCR2	读写	176h	POR 复位
• 捕获/比较控制寄存器 3	CCTL3	读写	168h	POR 复位
• 捕获/比较寄存器 3	CCR3	读写	178h	POR 复位
• 捕获/比较控制寄存器 4	CCTL4	读写	16Ah	POR 复位
• 捕获/比较寄存器 4	CCR4	读写	17Ah	POR 复位
• 中断向量寄存器	TAIV	只读	12Eh	（POR 复位）

地址 16Ch、16Eh、17Ch 和 17Eh 保留作将来扩展用。

11.2.1 Timer_A 控制寄存器 TACTL

所有关于定时器和它的操作的控制位都位于定时器控制寄存器 TACTL 中。在 POR 信号后所有位都自动复位。但 PUC 不影响它们。控制寄存器必须用字指令访问。

TACTL（160h）

15														0
未用						SSEL2-0		ID1-0		MC1-0		未用	CLR	TA-IE	TA-IFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

位 0：TAIFG：指示定时器溢出事件

增计数模式： 如果定时器由 CCR0 计到 0000h 则 TAIFG 置位。

连续模式： 如果定时器由 0FFFFh 计到 0000h 则 TAIFG 置位。

增/减计数模式： 如果定时器向下计到 0000h 则 TAIFG 置位。

位 1：定时器溢出中断允许位 TAIE。如置位，溢出将产生中断请求，复位不产生。

位 2：定时器清除位 CLR。定时器和输入分频器在 POR 后或 CLR 置位时复位。CLR 由硬件自动复位，读出始终为 0。定时器在下一个有效输入沿开始工作。如果不是因清除模式控制位暂停，定时器以增计数模式开始工作。

位 3：未用

位 4-5：模式控制位。

MC1	MC0	控制模式	说明
0	0	停止	定时器暂定
0	1	增计数到 CCR0	计数至 CCR0 并从 0 重新开始
1	0	连续，增计数	增计数，总共 65536 步
1	1	增减计数	增计数至 CCR0，减计数至 0,.....

位 6-7：输入分频控制位。

ID1	ID0	分频	说明
0	0	PASS	输入信号直通定时器

0	1	/2	输入信号 2 分频
1	0	/4	输入信号 4 分频
1	1	/8	输入信号 8 分频

位 8-10：定时器进入输入分频器的输入时钟源选择。

SSEL2	SSEL1	SSEL0	输入信号	说明
0	0	0	TACLK	用特定的外部引脚信号
0	0	1	ACLK	用辅助时钟 ACLK
0	1	0	MCLK	用系统时钟 MCLK
0	1	1	INCLK	见器件说明
1	X	X		保留

位 11-15：未用。

注意：修改 Timer_A

如果定时器工作在 ACLK 或外部时钟 TACLK，任何对定时器寄存器 TAR 的写入将导致不可预料的结果。CPU 的 MCLK 和定时器时钟的异步可能出现时间竞争。

注意：Timer_A 控制位的改变

如果用 TACTL 控制寄存器中的控制位改变定时器工作，在修改时定时器应停止，重要的改变是输入选择位、输入分频位和定时器清除位。异步的输入时钟和系统时钟（用于软件）可能发生竞争使定时器响应失败。

推荐的指令段落为：

1.修改控制寄存器和停止定时器。

2.启动定时器工作。

```
例：MOV #0286h,&TACTL     ;ACLK/8，定时器停止，定时器复位。
    BIS #10h,&TACLT       ;开始连续增计数
```

11.2.2 捕获/比较控制寄存器 CCTL

每个捕获/比较模块都有它自己的控制字 CCTLx。

CCTLx（162h）

15												0
捕获模式	输入选择	SCS	SCCI	未用	CAP	OUTMODx			CCIE	CCI	OUT	COV	CCIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

POR 使 CCTLx 所有位复位，PUC 不影响这些位。

位 0：捕获/比较中断标志 CCIFGx。

捕获模式：　如置位，表示在寄存器 CCRx 中捕获了定时数值。

比较模式：　如置位，表示定时器计数值等于比较寄存器 CCRx 之值。

CCIFG0 标志：　按 MSP430 中断原理，中断请求被接受，CCIFG0 自动复位。

CCIFG1-4 标志：　读中断向量后，确定中断向量的中断标志自动复位。如中断允许位复位就不会生成中断向量，而中断标志仍能置位。这时中断标志 CCIFG1 至 CCIFG4 需要由软件来复位。

位 1：捕获溢出标志 COV

CAP=0，选择比较模式。捕获信号复位。捕获事件不会使 COV 置位。

CAP=1，选择捕获模式。如果在捕获寄存器的值被读出前发生第二次捕获事件，溢出标志 COV 置位，COV 支持程序检测在先前的值被读出前是否又发生了捕获事件。读捕获寄存器时不会使溢出标志复位。

位 2：如果 OUTMODx 是零，OUTx 位是对应的输出状态。

位 3：捕获/比较输入信号 CCIx

捕获模式：选定的输入信号(CCIxA，CCIxB，VCC 或 GND)可读出

比较模式：CCI 复位

位 4：中断允许 CCIEx：允许或禁止捕获/比较模块 x 的中断请求信号。当中断允许位、CCIFGx 和 GIE 置位时，中断请求激活。

0：中断禁止，1：中断允许

位 5-7：

输出模式	说明
0：仅输出	OUTx 位的数字即 Outx
1：置位	比较信号 EQUx 使 Outx 置位
2：PWM 翻转/复位	比较信号 EQUx 使 Outx 翻转，EQU0 使 Outx 复位
3：PWM 置位/复位	比较信号 EQUx 使 Outx 置位，EQU0 使 Outx 复位
4：翻转	比较信号 EQUx 使 Outx 翻转
5：复位	比较信号 EQUx 使 Outx 复位
6：PWM 翻转/置位	比较信号 EQUx 使 Outx 翻转，EQU0 使 Outx 置位
7：PWM 复位/置位	比较信号 EQUx 使 Outx 复位，EQU0 使 Outx 置位

位 8：CAP：定义捕获/比较模块和相应的中断模块工作在捕获模式或比较模式。

0：比较模式，1：捕获模式

位 9：只读，始终是零。

位 10：捕获/比较输入信号 SCCIx。由比较输出 EQUx 同步：选定的输入信号(CCIxA,CCIxB，VCC 或 GND)用比较器相等信号 EQUx 锁存在透明锁存器中，可读。

位 11：捕获/比较信号可以用于与定时时钟异步或同步模式。

异步模式(SCS 复位)允许在请求时立即将 CCIFG 置位和立即捕获定时器值。如果捕获源的周期远小于定时器时钟这一模式很有用。如果定时器时钟和捕获源周期发生时间竞争，捕获寄存器的值可能会出错。

同步模式(SCS 置位)是常用的且捕获总是有效。

0：异步捕获，1：同步捕获

位 12-13：输入选择，CCIS1 和 CCIS0

在捕获模式下此两位定义提供捕获事件的输入源。在比较模式下这两位无用。

0：选择 CCIxA，1：选择 CCIxB，2：GND，低，3：VCC，高

位 14-15：

捕获模式	说明
0 禁止	捕获模式禁止
1 上升沿	上升沿捕获
2 下降沿	下降沿捕获
3 上升下降沿	在上升沿与下降沿都捕获

注意：同时捕获和捕获模式选择

如果通过捕获/比较寄存器 CCRx 中的 CAP 位使工作模式从比较模式变为捕获模式，不应同时进行捕获。那样在捕获/比较寄存器中的值是不可预料的。推荐的程序段落为：

1．修改控制寄存器，由比较模式切换到捕获模式

```
2．捕获
例：BIS #CAP,&CCTL2       ; 用 CCR2 选择捕获模式
    XOR #CCIS1,&CCTL2     ; 软件捕获：CCIS0=0，捕获模式=3
```

11.2.3 Timer_A 中断向量寄存器

16 位 Ttimer_a 有 2 个中断向量：

- 在 Timer_A 中断中，捕获/比较寄存器 CCR0 中断向量具有最高优先级，捕获/比较寄存器 CCR0 能用于定义在增计数和增/减计数模式中的周期。因此它需要最快速的服务。
- 其他捕获/比较寄存器共用一个复合向量。16 位中断向量字 TAIV 指示当前最高级中断。

CCR0 中断向量

如果定时器计数值等于比较寄存器的值，与 CCR0 相关的中断标志置位。

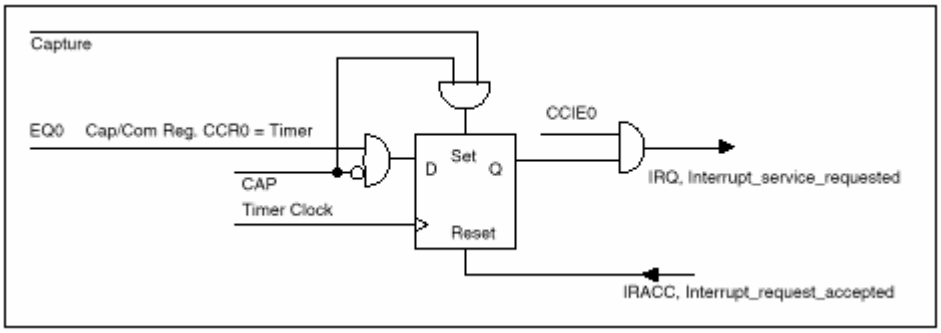


图 11.10：捕获/比较中断标志

捕获/比较寄存器 0 具有最高优先级，用它自己的中断向量来加速实时处理。

向量字 TAIFG，CCIFG1 到 CCIFG4 标志

有一个向量与 TAIFG 标志及其他 4 个捕获/比较寄存器 CCR1 至 CCR4 相关，结合成一个中断优先级结构中：具有最高优先级的 CCIFGx 标志产生从 0(无标志置位)到 12 的优先级级别码。这一编码可以加到 PC 上以进入与中断相应的软件。向量字 TAIV 以 16 位字加到 PC 上（见软件实例）。

从向量字寄存器读当前中断向量 TAIV 将定义向量的 CCIFGx 复位。

TAIV（12Eh）

15															0
0	0	0	0	0	0	0	0	0	0	0	中断向量				0
r0	r0	r0	r0	r0	r0	r0	r0	r0	r0	r0	r-(0)	r-(0)	r-(0)	r-(0)	r0

中断优先级	中断源	缩写	向量地址	向量寄存器内容
最高	捕获/比较 0	CCIFG0	X	N.A.
	捕获/比较 1	CCIFG1	Y	2
	捕获/比较 2	CCIFG2	Y	4

	捕获/比较 3	CCIFG3	Y	6
	捕获/比较 4	CCIFG4	Y	8
	定时器溢出	TAIFG	Y	10
最低	保留		Y	12
	没有中断挂起		Y	0

如果 CCIEx 或 TAIE 置位，并且 GIE 置位，CCIFGx 或 TAIFG 置位产生定时器中断请求。具有最高优先级的中断请求服务。读取定时器向量 TAIV，中断服务请求位(CCIFGx 或 TAIFG)将自动复位。现在，具有次高优先级的标志位定义向量 TAIV。如果有任何中断允许位(CCIEx 或 TAIE)和相应的中断标志已置位，会立即产生一个中断请求

CPU 可对所有中断标志 CCIFGx 和 TAIFG 进行完整的操作是它的特点。

注意：对只读寄存器 TAIV 写入

对 TAIV 进行写操作使决定向量字的当前中断标志复位。其后的软件处理会漏过正在请求服务的中断事件。此外，对只读寄存器写操作将在写信号有效时增加电流消耗。

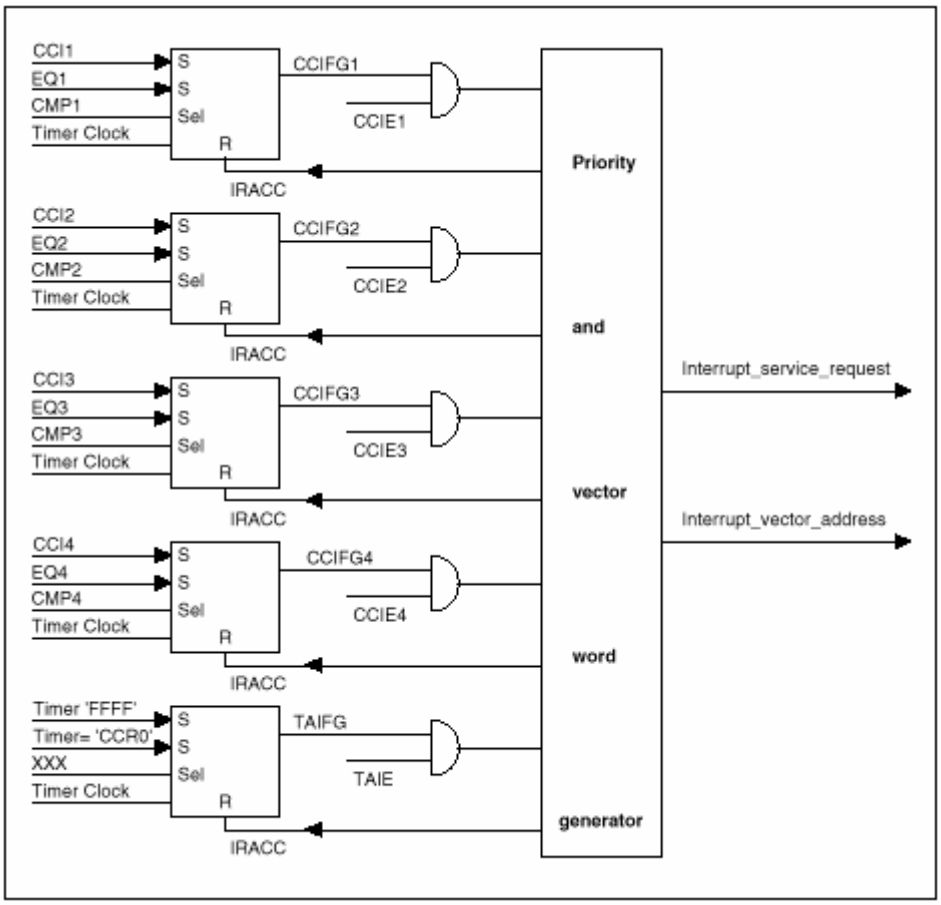


图 11.11：捕获/比较寄存器中断向量原理图

定时器中断向量寄存器，软件实例：

软件实例说明了中断向量 TAIV 的应用和处理开销。右边的数字显示每个指令需要的时钟周期。此程序是为连续模式写的：与下一个中断的时间差已加入相应的比较寄存器中。

```
; Software example for the interrupt part                              Cycles
;
; Interrupt handler for Capture/Compare Module 0.
; The interrupt flag CCIFG0 is reset automatically
;
TIMMOD0     ...                     ; Start of handler Interrupt latency    6
            RETI                                                            5
;
; Interrupt handler for Capture/Compare Modules 1 to 4.
; The interrupt flags CCIFGx and TAIFG are reset by hardware
; Only the flag with the highest priority responsible for the
; interrupt vector word is reset.
TIM_HND     $                       ; Interrupt latency                     6
            ADD     &TAIV,PC        ; Add offset to Jump table              3
            RETI                    ; Vector 0: No interrupt                5
            JMP     TIMMOD1         ; Vector 2: Module 1                    2
            JMP     TIMMOD2         ; Vector 4: Module 2                    2
            JMP     TIMMOD3         ; Vector 6: Module 3                    2
            JMP     TIMMOD4         ; Vector 8: Module 4                    2
;
; Module 5. Timer Overflow Handler: the Timer Register is
; expanded into the RAM location TIMEXT (MSBs)
;
TIMOVH                              ; Vector 12: TIMOV Flag
            INC     TIMEXT          ; Handle Timer Overflow                 4
            RETI                                                            5
;
TIMMOD2                             ; Vector 4: Module                      2
            ADD     #NN,&CCR2       ; Add time difference                   5
            ...                     ; Task starts here
            RETI                    ; Back to main program                  5
;
;
TIMMOD1                             ; Vector 2: Module 1
            ADD     #MM,&CCR1       ; Add time difference                   5
            ...                     ; Task starts here
            RETI                    ; Back to main program                  5
; The Module 3 handler shows a way to look if any other interrupt is
; pending: 5 cycles have to be spent, but 9 cycles may be saved if
; another interrupt is pending
```

```
;
TIMMOD3                             ; Vector 6: Module 3
            ADD     #PP,&CCR3       ; Add time difference                5
            ...                     ; Task starts here
            JMP     TIM_HND         ; Look for pending intrpts           2
;
            .SECT   "VECTORS",0FFF0h ; Interrupt Vectors
            .WORD   TIM_HND         ; Vector for Capture/Compare Module 1..4
; and timer overflow TAIFG
            .WORD   TIMMOD0         ; Vector for Capture/Compare Module 0
```

如果 FLL 关闭，需要增加 2 个额外时钟周期，来使 CPU 系统起动和系统时钟 MCLK 同步。

对不同的中断源，程序的开销包括中断等待和从中断返回周期数（非处理任务本身）。

- 捕获/比较模块 CCR0　　11 个周期
- 捕获/比较模块 CCR1 至 CCR4　　16 个周期
- 定时器溢出 TAIFG　　14 个周期

定时极限

用 TAIV 寄存器和上述软件，利用一个比较寄存器时两个事件之间的最短重复时间是：

$$t_{CRmin} = t_{taskmax} + 16 * t_{cycle}$$

其中：　$t_{taskmax}$　在中断程序完成任务(例如计数器加 1)的最长时间（最差情况）

t_{cycle}　MCLK频率下时钟周期

利用一个捕获寄存器时两个事件之间的最短重复时间是：

$$t_{CLmin} = t_{taskmax} + 16 * t_{cycle}$$

11.3 Timer_A 应用

11.3.1 Timer_A 增计数模式应用

如果定时周期不是 65536Hz，可用增计数模式，在连续模式定时周期是 65536Hz。CCR0 用于定义定时周期。

输出单元 OU0 的性能

由于 CCR0 也用于定义定时周期，输出单元 OU0 通常工作在 4 种模式，即输出模式 0、输出模式 1、输出模式 4 和输出模式 5 之一。因为 EQU0 信号同时有其它用途，其他 4 种模式不用。

输出单元 OU1 到 OU4 的性能

输出单元 OU1 至 OU4 和它们的驱动电路完全相同，具有相同的特性。每个都可以工作在相同或不同的模式下。

应用软件可以选择和控制产生信号或捕获定时值。下图介绍了不同输出模式的基本功能。举例时用 OUT1 作为说明。

定时器： 定时器重复从 0 增计数到 CCR0。

输出模式 0： 输出信号 OUTx 由每个捕获/比较模块中的控制寄存器 CCTLx 中的 OUTx 位定义，它与任何定时功能无关并且在软件的完全控制下。

输出模式 1： 当定时器计数值等于 CCR1 时输出置位。由 EQU0 信号(CCIFG0)引起的中断可用于比较寄存器 x 的修改。

输出模式 2： 当定时器计数值等于 CCR1 时输出翻转。当定时器计数值等于 CCR0 时复位，同时定时器也复位。基本应用是 PWM 功能或与其他输出一起产生相位关系。

输出模式 3： 当定时器计数值等于 CCR1 时输出置位，当定时器计数值等于 CCR0 时复位，同时定时器也复位。基本应用是 PWM 功能或与其他输出一起产生相位关系。

输出模式 4： 当定时器计数值等于 CCR1 时输出翻转，输出周期是定时器周期的两倍。与任何其他输出的相位关系取决于选择 CCRx 数据。

输出模式 5： 当定时器计数值等于 CCR1 时输出复位，由 EQU0 信号(CINT0)引起的中断可用于修改比较寄存器 X。

输出模式 6： 当定时器计数值等于 CCR1 时输出翻转，当定时器计数值等于 CCR0 时置位。基本应用是 PWM 功能或与其他输出一起产生相位关系。

输出模式 7： 当定时器计数值等于 CCR1 时输出复位，当定时器计数值等于 CCR0 时置位，同时定时器也复位。基本应用是 PWM 功能或与其他输出一起产生相位关系。

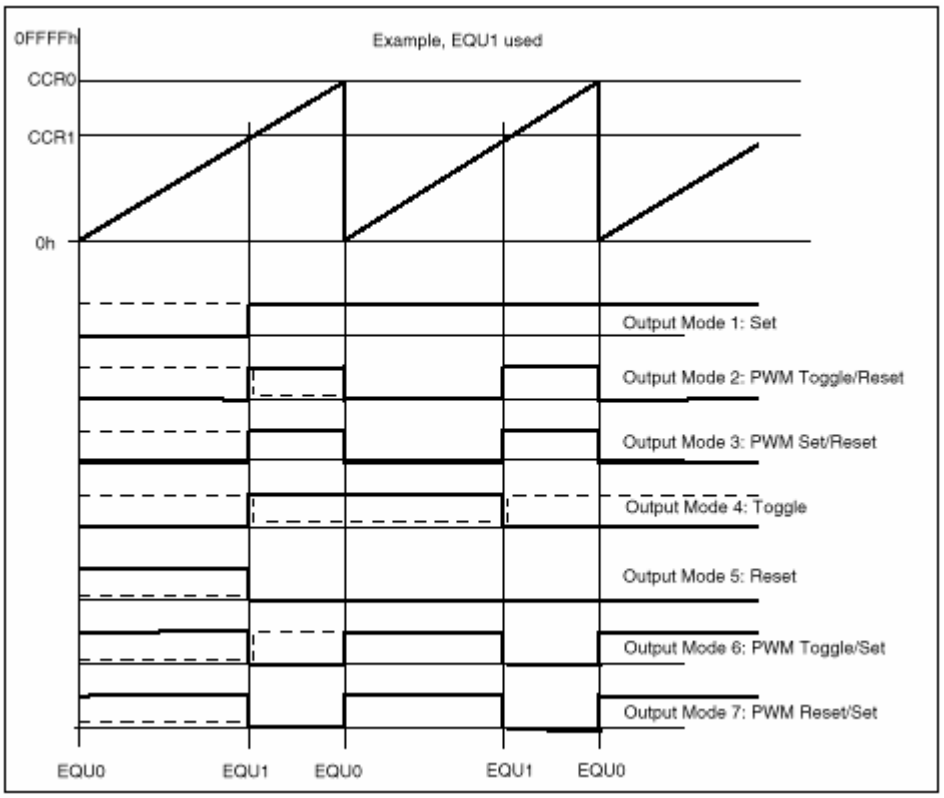


图 11.12：增计数模式的输出单元

11.3.2 Timer_A 连续模式应用

如果定时器周期是 65536 时钟脉冲对应用无关紧要时，可用这一模式。连续模式的主要应用是产生独立的软件时序。捕获/比较寄存器 CCR0 与其他 4 个捕获/比较寄存器 CCRx 的应用方法相同。

所有输出模式对不同类型应用都是有用的。通过 CCTLx 寄存器的输出模式字 OMX2 至 OMX0 可选择各种输出模式下可能的输出信号。

应用软件可以选择和控制产生信号或捕获定时值。下图介绍了不同输出模式的基本功能。输出 OUT0 和 OUT1 仅作为说明之用。CCR0 中的值大于 CCR1 中的值。

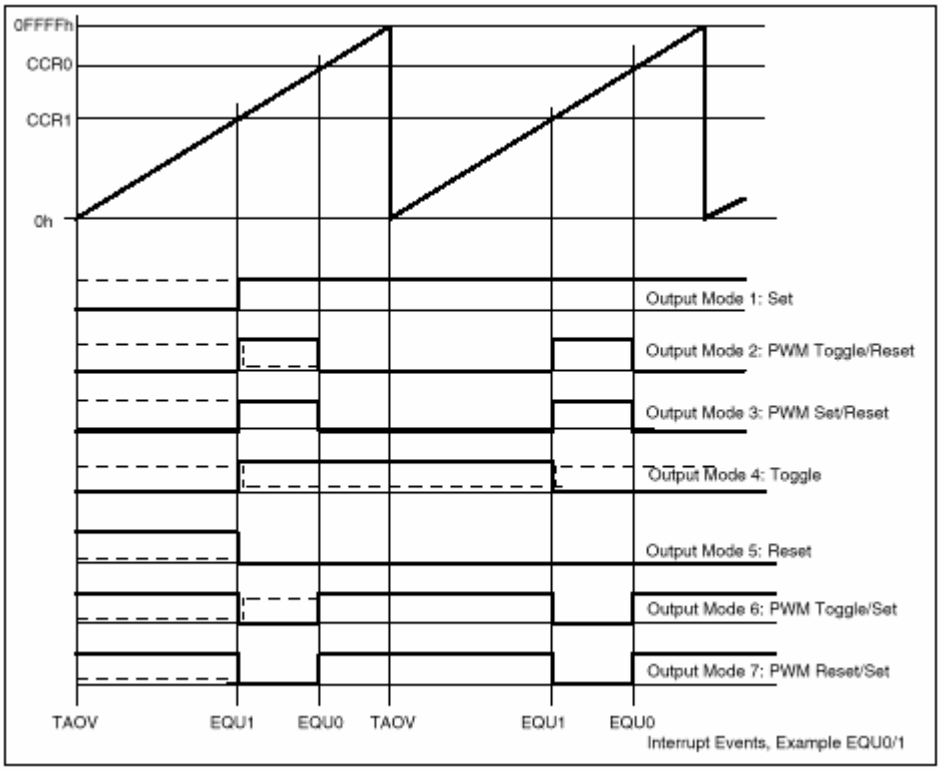


图 11.13：连续模式中的输出单元

定时器： 定时器重复从 0 增计数到 0FFFFh。

输出模式 0： 输出信号 OUTx 由每个捕获/比较模块中的控制寄存器 CCTLx 中的 OUTx 位定义，它与任何定时功能无关并且在软件的完全控制下。

输出模式 1： 当定时器计数值等于 CCR1 时输出置位。由 EQU0 信号(CCIFG0)引起的中断可用于比较寄存器 x 的修改。

输出模式 2： 当定时器计数值等于 CCR1 时输出翻转。当定时器计数值等于 CCR0 时复位，同时定时器也复位。基本应用是产生脉冲。

输出模式 3： 当定时器计数值等于 CCR1 时输出置位，当定时器计数值等于 CCR0 时复位。基本应用是产生脉冲。

输出模式 4： 当定时器计数值等于 CCR1 时输出翻转，输出周期是定时器周期的两倍。与任

何其他输出的相位关系取决于选择 CCRx 数据。

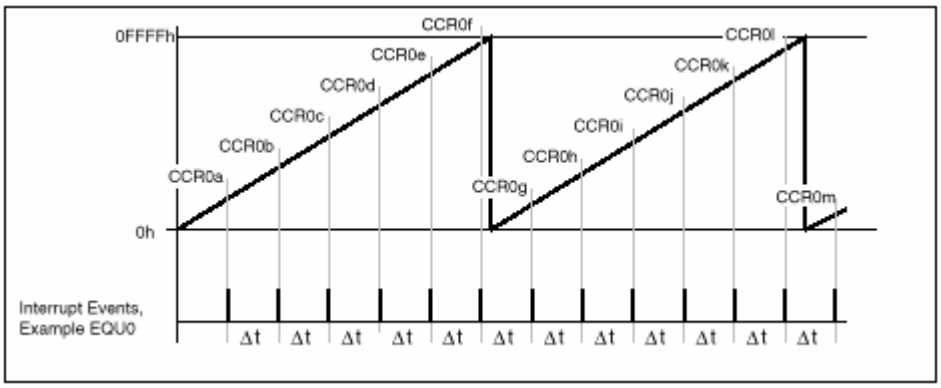
输出模式 5： 当定时器计数值等于 CCR1 时输出复位，由 EQU0 信号(CINT0)引起的中断可用于修改比较寄存器 X。

输出模式 6： 当定时器计数值等于 CCR1 时输出翻转，当定时器计数值等于 CCR0 时置位。基本应用是产生脉冲。

输出模式 7： 当定时器计数值等于 CCR1 时输出复位，当定时器计数值等于 CCR0 时置位。基本应用是产生脉冲。

连续模式，产生时间间隔

连续模式可方便地被应用软件用于产生定时时间。如果中断允许，在每次定时完成时会产生中断。在中断服务程序中，到达下一个事件的时间长度加到用于这一功能的捕获/比较寄存器 CCRx 上。用所有 5 个捕获/比较模块最多可实现 5 个独立的定时事件。



如果 CCR0 用作周期寄存器，时间间隔也可以由其他模式来产生。由于旧的 CCRx 和新的周期之和可能大于 CCR0 寄存器，因此处理更复杂。当 CCRxold 加上△t 的和大于 CCR0 的值时，这个和必须减去 CCR0 的值以校正时间间隔。

11.3.3 Timer_A 增/减计数模式应用

如果定时周期不是 65536 时钟周期并且需要产生对称的脉冲波形时，可用增减计数模式。CCR0 用于定义定时器周期，定时器周期是 CCR0 值的 2 倍。

输出单元 OU0 的能力

捕获/比较寄存器用于定义定时器周期。输出单元仅工作在输出模式 0、1、4 和 5。因为定时器已被 CCR0 相等信号 EQU0 控制，所有其他模式无效，

输出单元 OU1 至 OU4 的能力

输出单元 OU1 至 OU4 及其驱动电路完全相同，4 个单元具有相同的功能，能工作在不同模式下。

应用软件可以选择和控制产生信号或捕获定时值。下图介绍了不同输出模式的基本功能。输出 OUT3 仅作为说明之用。

在增减计数模式中连续运行会产生 2 个中断：由捕获/比较模块 CCR0 产生的中断和当定时器在减计数到达 0 时产生的中断。这 2 个中断可用于进行适当的输出脉冲修改。

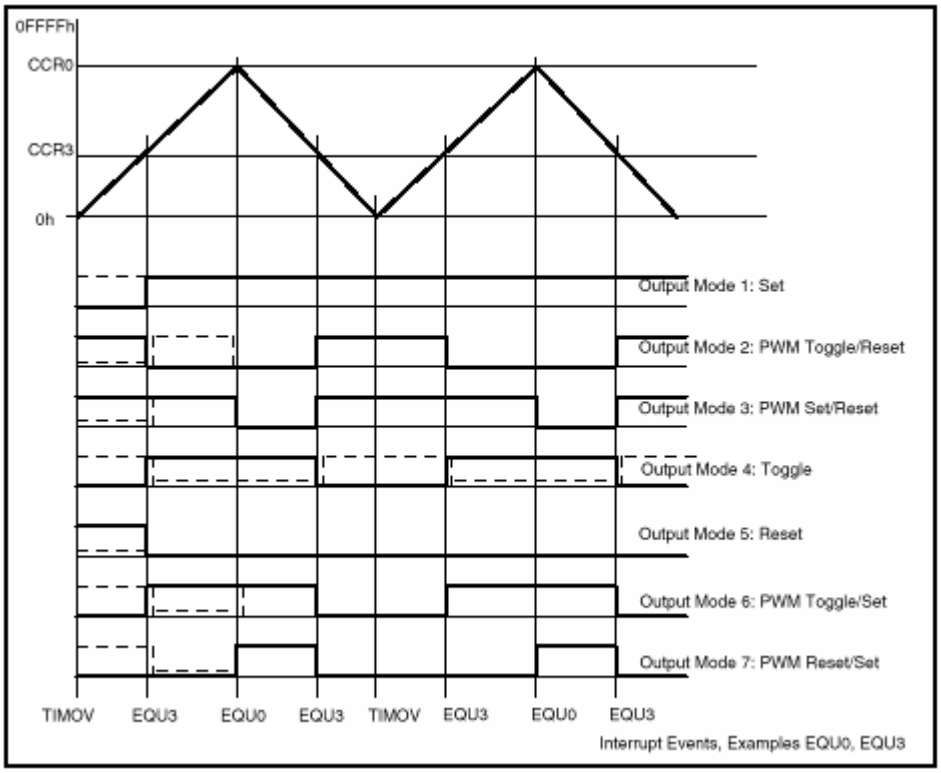


图 11.14：增减计数模式输出单元（I）

增减计数模式使应用有可能强制利用两个输出信号之间的“死区时间”。例如，两个输出驱动一个H-桥时必须不同时输出为高，以防止过载。在短暂的编程时间里，即死区时间，两个输出都切换为低。相反的情况也是可行的，如果需要，两个输出也可编程为永不同时为低。在下例中，t_{dead}为：

$$t_{dead} = t_{timer} * (CCR1-CCR3)$$

t_{dead}　　两个输出都必须为低的时间

t_{timer} 定时器寄存器输入频率的周期长度

CCRx　　比较寄存器 x 的内容

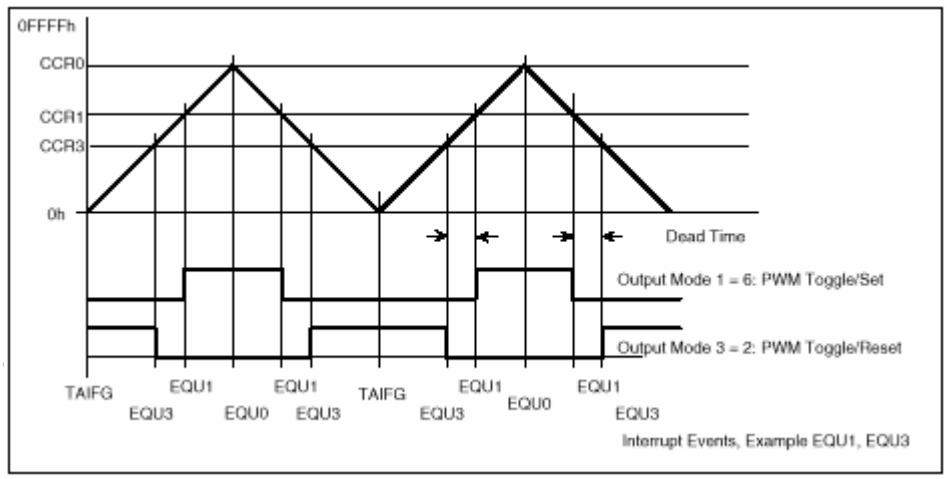


图 11.15：增减计数模式的输出单元（II）

11.3.4 Timer_A 软件捕获

每个捕获/比较寄存器能被软件用于获得时间标记，可用于各种目的：

- 测量软件程序所用时间
- 测量硬件事件之间的时间
- 测量系统频率
- ……

用 CCISx1 和 CCISx0 及通过 CCMx1 和 CCMx0 选定的捕获模式可执行软件捕获。捕获模式可选择用捕获信号 CCIx 的上升沿、下降沿或两个沿进行捕获。当捕获模式选择在两个沿进行捕获时实现最简单。捕获输入信号选择为 VCC/高或 GND/低。CCISx1 置位，根据 CCISx0 选择是 VCC/高或 GND/低作为捕获信号。

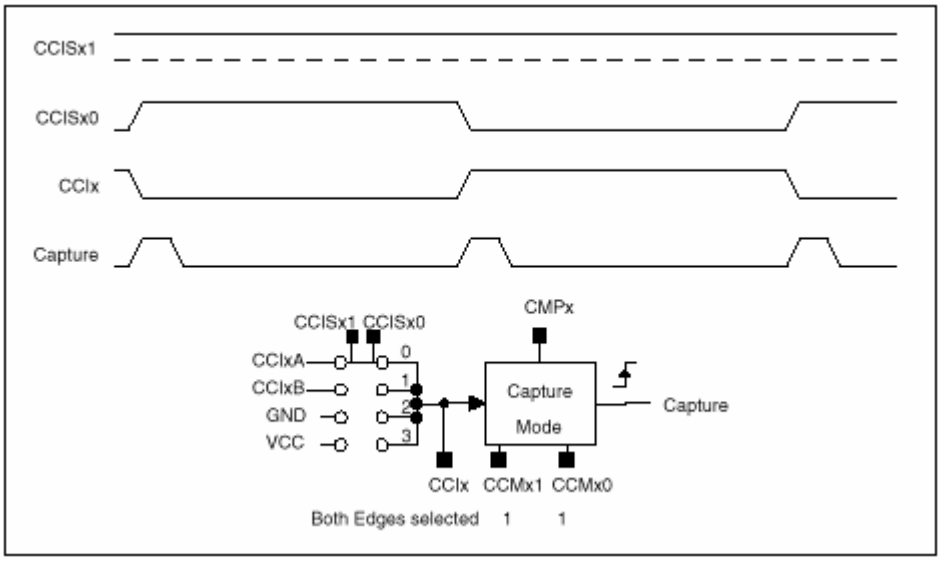


图 11.16：软件捕获实例

```
; Software example to capture data performed by software
;
; The data of the capture/compare register CCRx are taken by the software
; It is assumed that CCMx1, CCMx0 and CCISx1 bits are set.
; The bit CCISx0 selects the CCIx signal to be high or low
;
            ...
            ...
            XOR     #CCISx0,&CCTLx
```

. . .
. . .
. . .

11.3.5 Timer_A 处理异步串行通信协议

串行异步通信用给定的波特率发送和接收数据。工业标准规定了几种波特率。接收与发送可使用相同或不同的波特率。接收开始于信号的下降沿。接收器与此下降沿同步，后续各位按选定的波特率接收。

用比较功能经输出单元从选定引脚上移出数据可以实现发送功能。每次中断程序中设定数据可确保波特率。用模式 1 使输出引脚置位，模式 5 使引脚复位。

用捕获/比较功能，以控制寄存器的 SCCIx 位从引脚上的数据移入存储器可以实现接收功能。捕获定时器数据即接收信号的下降沿可识别接收开始时间。然后同一个捕获/比较模块改为比较功能。比较数据是捕获时间加上由波特率决定的半位时间。第一位用第一个比较事件 EQUx 锁存。后续位的扫描根据选定的波特率以同样方法完成。与位扫描相关的中断服务程序收集一个字的所有位供以后软件处理之用。

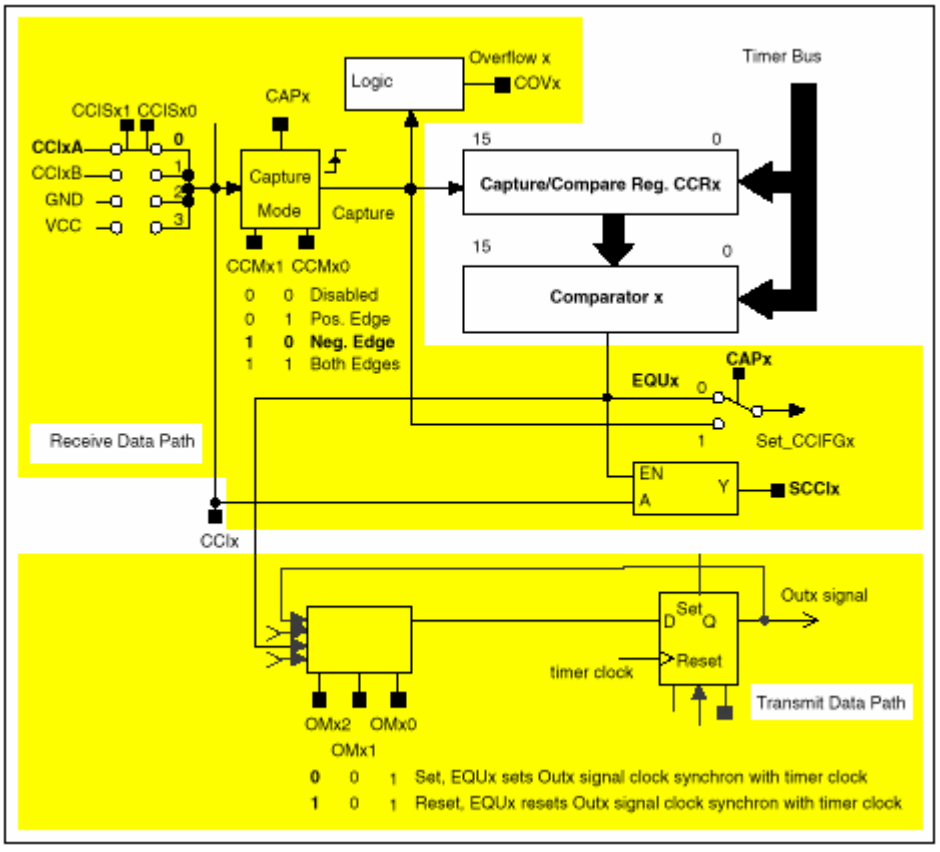


图 11.17：Timer_A 用于处理异步通信协议

当选定为半双工通信需用一个捕获/比较模块。用 2 个捕获/比较模块可实现全双工模式。在半双工模式中，接收和发送必须依次运行，可以只用一条数据线。在全双工模式中，接收和发送可以并行执行。

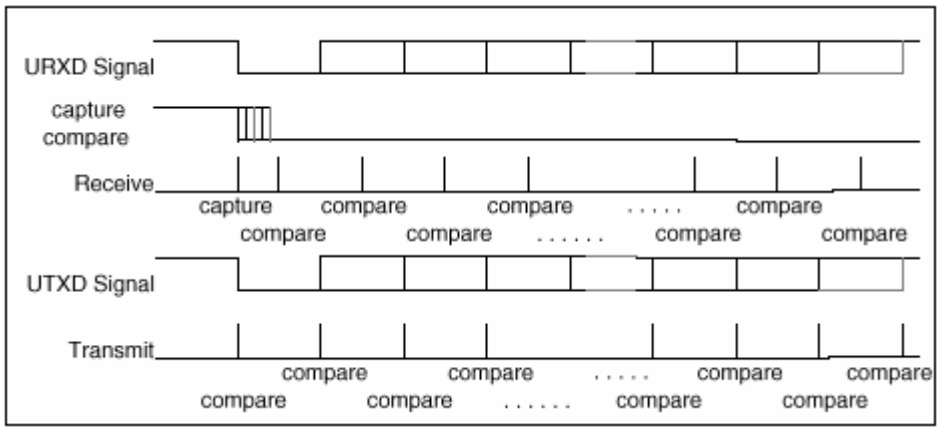
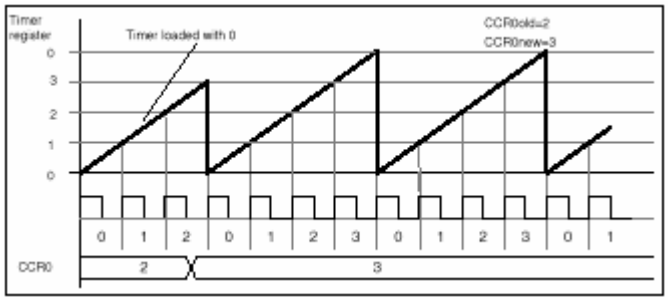


图 11.18：Timer_A，异步通信协议处理的时序

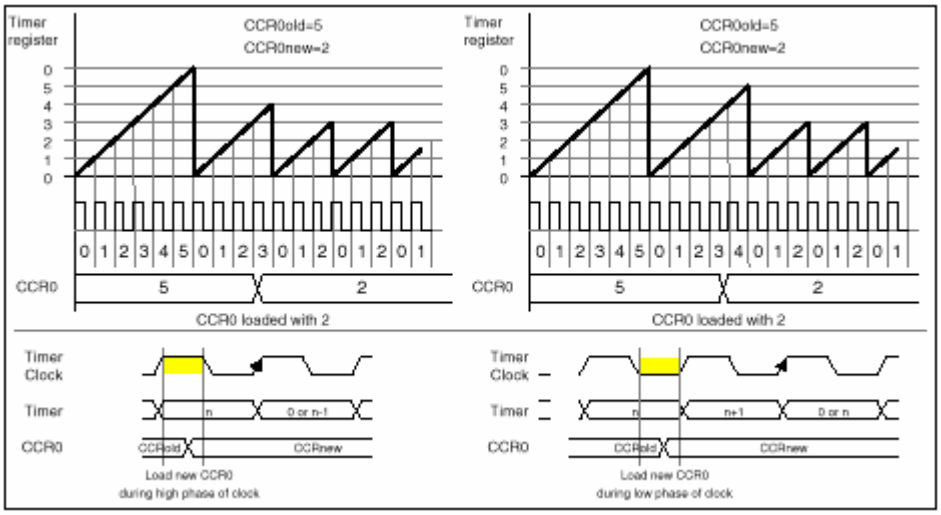
11.4 Timer_A 的特殊情况

本节讨论一些可能的特殊情况。所有定时器和比较功能的一条基本原则是，要执行选定的功能定时器寄存器必须(通过定时器时钟)增 1 或减 1。

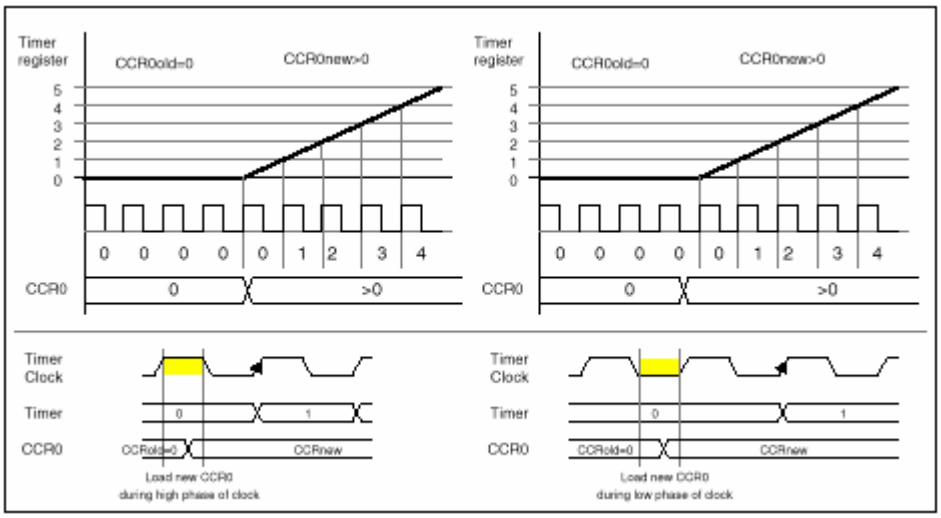


11.4.1 CCR0，用作周期寄存器

比较寄存器在定时器寄存器增计数前半周期与定时器寄存器匹配。如 CCR0 用作周期寄存器且新周期等于或大于旧周期，定时器将工作到新周期，无特殊情况需要注意。如 CCR0 用作周期寄存器且新周期小于旧周期，如果新值在定时器时钟的高电平时写入 CCR0，定时器在下一个时钟上升沿开始改变。定时器继续在下一个定时器时钟有效边沿加 1。如果新值在定时器时钟的低电平时写入 CCR0，定时器在定时器时钟的下一个上升沿加 1 并在第二个上升沿开始改变。

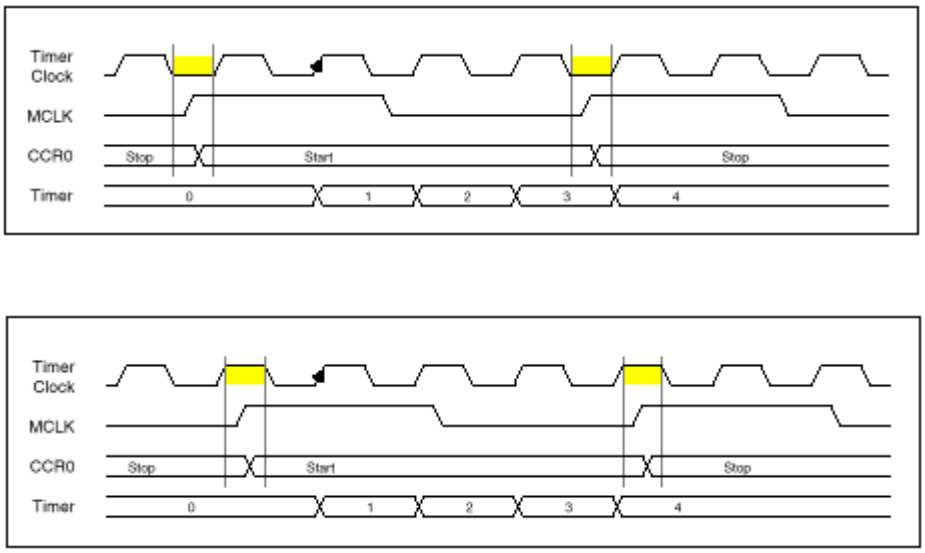


以上例子说明了增计数模式的不同情况。当定时器工作在增减计数模式，如果定时器工作在增计数时，响应相同。如果在减计数时改变 CCR0，定时器继续减计数到零。



11.4.2 定时器寄存器的启/停

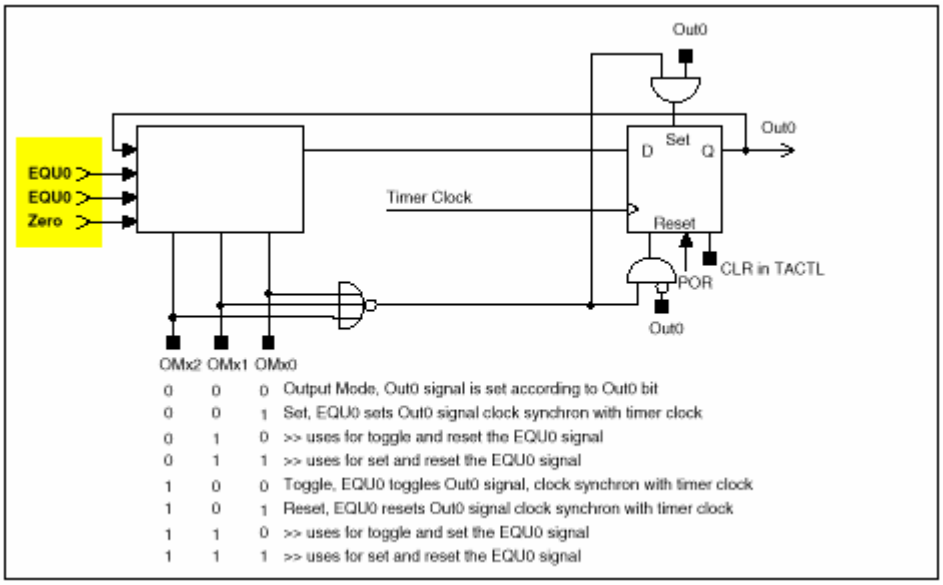
定时器寄存器的开始和停止遵循与 CCR0 周期寄存器相同的基本规则。



选定的计数模式在定时器时钟的下降沿时装入。如果 3 个运行模式之一选中，定时器寄存器在下一个上升沿加 1。如果定时器寄存器被停止，下一个时钟上升沿不再使定时器寄存器加 1。

11.4.3 输出单元 0

所有输出单元具有相同的结构。输入用不同的控制信号定义特殊操作。控制信号中的两个是：相关模块 x 中的“定时器比较相等”寄存器(CCRx)的比较器输出和模块 0 中的“定时器比较相等”寄存器(CCR0)的比较器输出。当模块 x 是输出单元 0，不是所有可能的工作状况都可以采用。见下图：



建议采用模式 0、1、4 和 5

通用同步异步收发模块 USART

本章说明了串行通信接口 USART。它实现了两种功能，使得串行通信可以以不同的模式工作。第一种功能是熟悉的异步串行通信协议 UART；第二种功能是串行外围模块接口功能 SPI，它也得到广泛应用。尽管对于这两种功能硬件是公用的，但仍要针对它们的最终选择加以说明，在应用环境中通常定义为 UART 或 SPI。经过适当的软硬件设计，这两种功能可以交替使用。由控制寄存器中的一位来定义模块以 UART 或 SPI 模式运行。

目录 页号

USART 外围接口

通用同步/异步接口是一串行通道，它允许 7 或 8 位串行位流以编程速率或外部时钟定义的速率移入、移出 MSP430。根据硬件结构，USART 外围接口支持两种不同的串行协议：通用异步协议（常简称为 RS232）和同步串行协议（即 SPI 协议）。

用控制寄存器 UCTL 中的控制位 SYNC 来选择所需的模式：

SYNC=0： 选择异步模式 UART

SYNC=1： 选择同步模式 SPI

USART 以字节外围模块与 CPU 相连。微控制器通过 3 或 4 个引脚与外部系统相连。

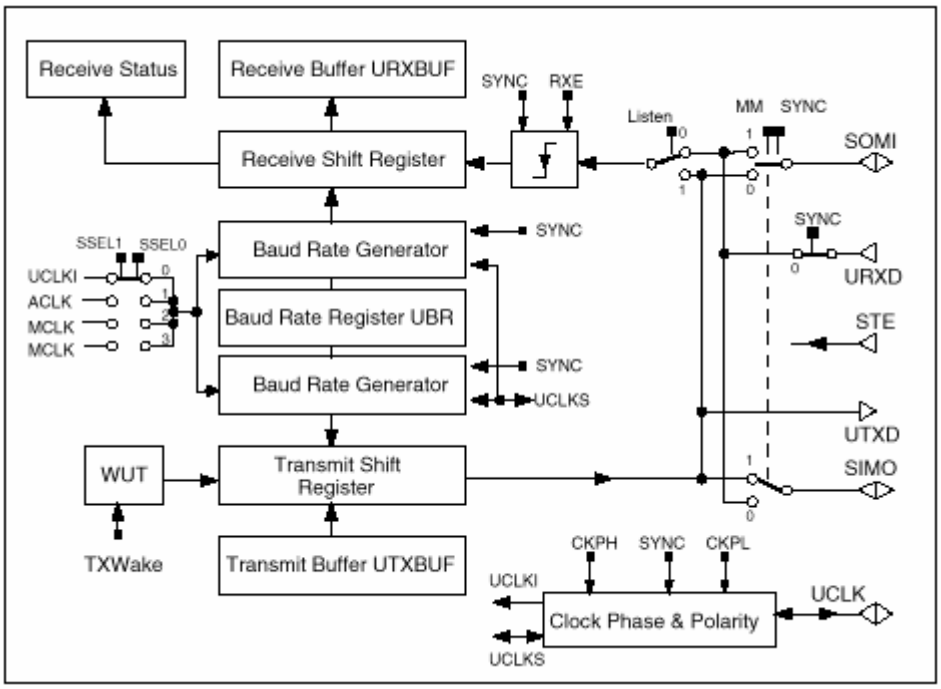


图 12.1：USART 功能框图

12．USART 外围接口，UART 模式

通用同步/异步接口是一串行通道，它允许 7 或 8 位串行位流以编程速率移入、移出 MSP430。USART 控制寄存器 UCTL 的控制位 SYNC 复位时选择异步通信模式。USART 作为字节外围与 CPU 相连。微控制器通过 3 个引脚与外部系统相连。

USART 的串行异步通信特性

- 异步模式，包括线路空闲/地址位通信协议
- 两个移位寄存器，串行数据移入 URXD，从 UTXD 移出
- 数据发送和接收都从最低位开始
- 可编程的发送/接收数据传输率
- 状态标志

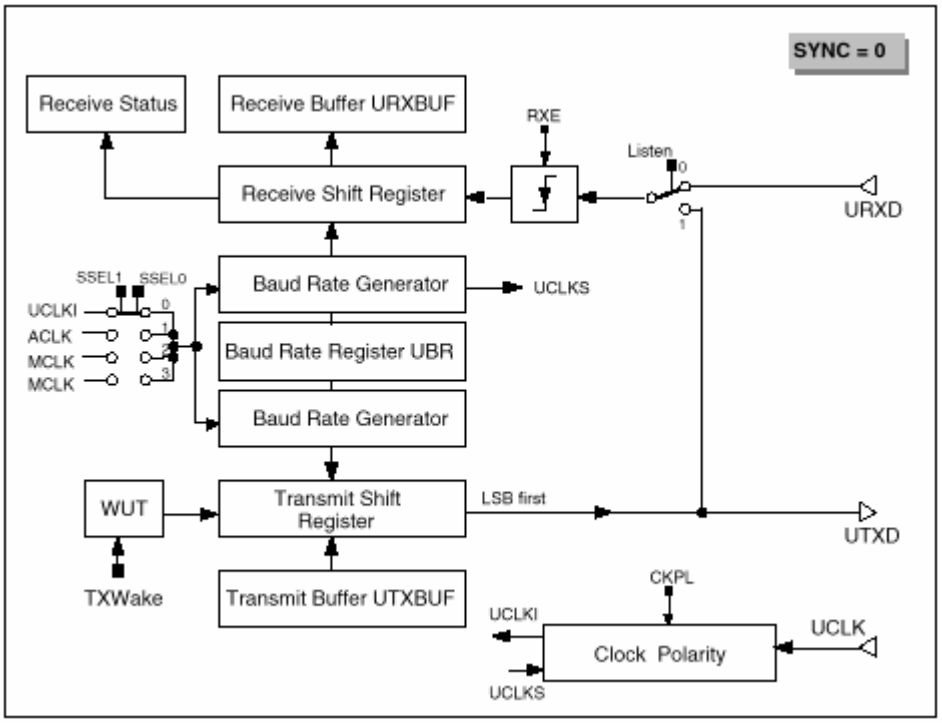


图 12.1：USART 功能框图：UART 模式

12.1 异步操作

在异步模式下，接收器实现自身与帧的同步，但外部发送和接收设备并不使用同一个时钟源；波特率的产生是在本地的。

12.1.1 异步帧格式

异步帧格式由 1 个起始位、7 或 8 个数据位、奇/偶/无校验位、1 个地址位（地址位模式）和 1 或 2 个停止位组成。通过选择时钟源和波特率寄存器的数据来确定位周期。

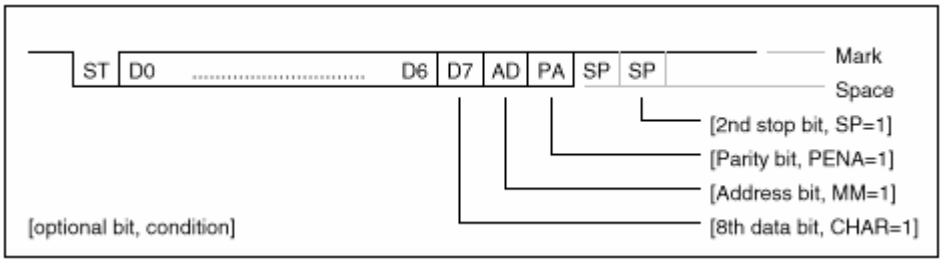


图 12.2：异步帧格式

以接收到有效的起始位来初始化接收操作。包括检测 URXD 端口的下降沿，然后以 3 次采样多数表决方法取值，其中 2 个必须为 0。采样发生在负边沿后 BRCLK 周期的 n/2-x、n/2、和 n/2+x 处。这一过程实现拒收错误起始位及帧中各位的中心定位功能，使各位在中心采用多数表决读数。其中 x 的值是 BRCLK 的 1/32 到 1/63，但是在最低的 BRCLK。X 的取值取决于波特率发生器的分频。

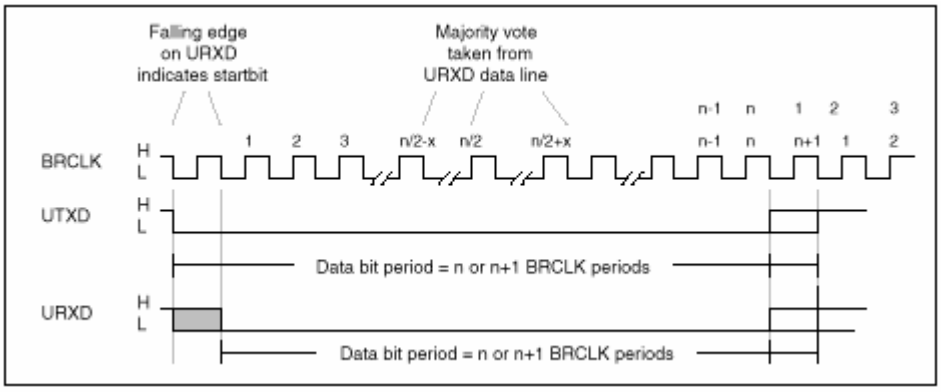


图 12.3：异步位格式，以 n 或 n+1 周期为例

12.1.2 异步通信的波特率发生器

MSP430 的波特率发生器与其它标准的串行通信接口适配器实现的方法不同。

标准的波特率发生

标准的实现方法是将一时钟源预分频，再加一固定分频器，通常是 16 分频。

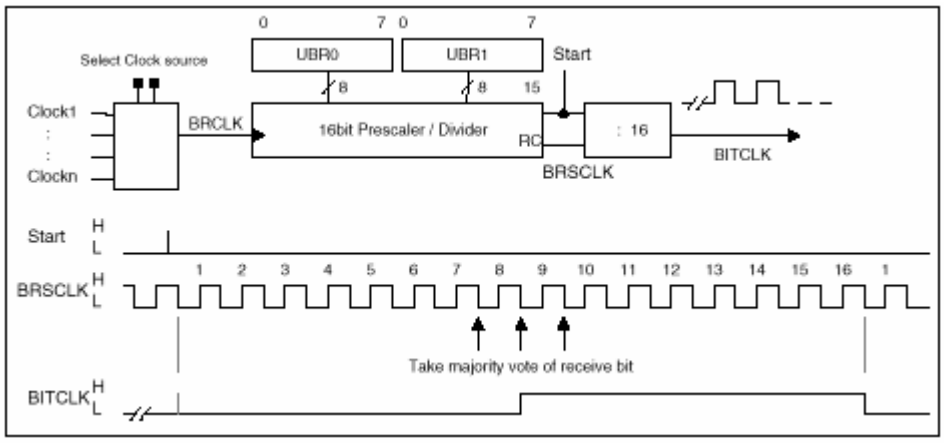


图 12.4：标准的波特率发生，非 MSP430 采用

波特率 = BRCLK / (n*16)

用这一常用的方法发生波特率无法得到接近预分频输入频率 BRCLK 的波特率。例如分频因子 N 不能为 18，同样非整数因子也无法使用，如 13.67。

例 1

设 BRCLK 信号频率为 32768Hz，波特率要求为 4800 波特，则分频因子为 6.83。但在标准波特率发生器里，最小的因子为 16，因此晶振频率和波特率发生不能满足要求。

例 2

设 BRCLK 信号的频率为 1.04MHz（32 x 32768Hz），波特率要求为 19200 波特，则分频因子为 54.61，但在标准波特率发生器里，较接近的分频因子为 48（3*16）和 64（4*16），因此晶振频率和波特率发生不能满足要求。晶振频率需要挑选以满足通信要求。而其它原则，诸如电流消耗、简单的实时钟功能或系统成本限制等无法仔细考虑。

MSP430 的波特率发生

MSP430 波特率发生器用了一个预分频/分频器和一个调整器。即使晶振频率不是所需波特率的整数倍，这一组合不仅能正常工作而且使通信协议可以工作在最大的波特率。采用这

一技术，即使是手表晶体（32768Hz），波特率达到 4800（9600）波特也是可能的。它到来的优点很明显，使得复杂的 MSP430 工作模式选择在低功耗成为可能。

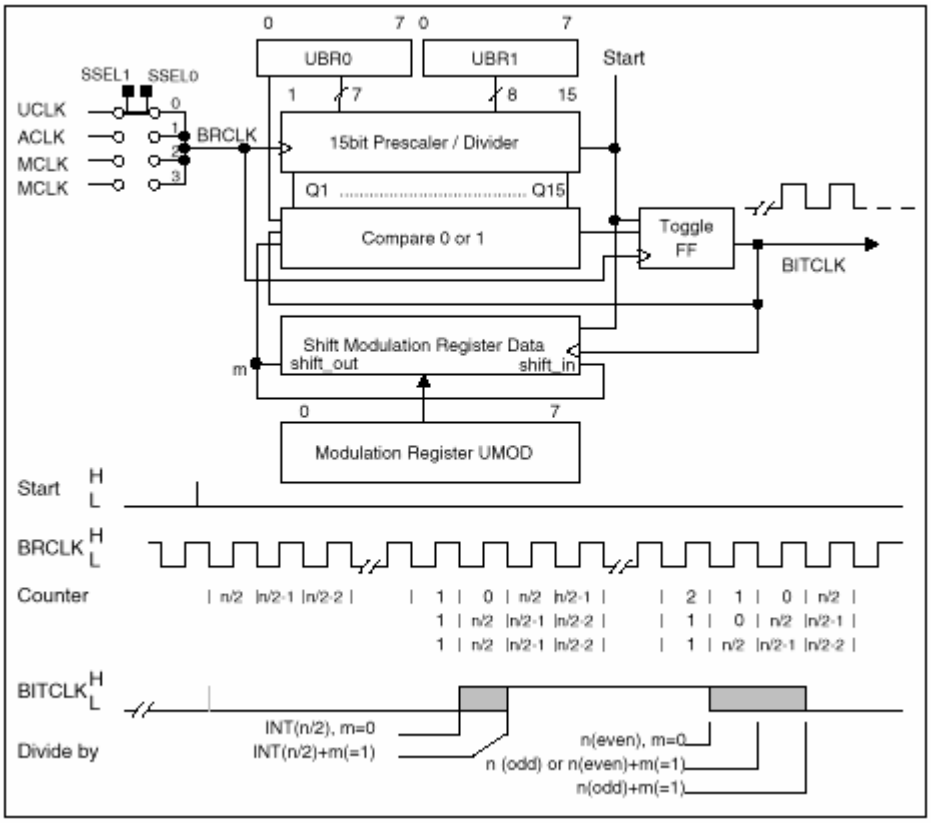


图 12.5：MSP430 波特率发生：n 或 n+1 周期举例

调整寄存器的 LSB 位（即最低位）最先用于调整，它由起始位开始。某调整位置位，将分频因子增加 1。

例 1：

假定 BRCLK 的频率为 32768Hz，波特率要求为 4800 波特，则分频因子为 6.83。MSP430 的 USART 采用分频因子 6 加上调整寄存器加载 6Fh（01101111）来产生波特率。即分频器按顺序 7，7，7，7，6，7，7，6，•••••• 来分频。在 8 位调整位都使用后，再重复这一顺序。

例 2：

假定 BRCLK 信号的频率为 1.04MHz，波特率要取为 19200 波特，则分频因子为 54.61。MSP430 的 USART 采用分频因子 54（36h）加上调整寄存器加载 D5h 来产生波特率，即分频器按顺序 55，54，55，54，55，54，55，55 •••••• 来分频。在 8 位调整位都使用后，再重复这一顺序。

下表列出了在手表晶体 32768Hz（ACLK）和 MCLK(设为 32 倍 ACLK)时的标准波特率所需波特率寄存器和调整寄存器数值。同时列出接收时的误差。此外，接收时的同步误差也应该考虑。

	分频		ACLK				MCLK (= 32 x ACLK)			
波特率	ACLK	MCLK	UBR1	UBR0	UMOD	最大误差%	UBR1	UBR0	UMOD	最大误差%
75	436.91	13981.00	1	B4	FF	-.1/.3	36	9D	FF	0/.1
110	297.89	9532.51	1	29	FF	0/.5	25	3C	FF	0/.1
150	218.45	6990.50	0	DA	55	0/.4	1B	4E	FF	0/.1
300	109.23	3495.25	0	6D	22	-.3/.7	0D	A7	00	-.1/0
600	54.61	1747.63	0	36	D5	-1/1	06	D3	FF	0/.3
1200	27.31	873.81	0	1B	03	-4/3	03	69	FF	0/.3
2400	13.65	436.91	0	0D	6B	-6/3	01	B4	FF	0/.3
4800	6.83	218.45	0	06	6F	-9/11	0	DA	55	0/.4
9600	3.41	109.23	0	03	4A	-21/12	0	6D	03	-.4/1
19200		54.61					0	36	6B	-.2/2
38400		27.31					0	1B	03	-4/3
76800		13.65					0	0D	6B	-6/3
115200		9.10					0	09	08	-5/7

表 12.1：常用波特率，波特率数据及误差

对接收模式和发送模式计算了最大误差。接收模式误差是针对每一位扫描在理想的中间位置的积累定时误差。发送模式误差是针对理想的位周期的积累定时误差。

MCLK 的最高频率见器件手册，它可以超过所举例子中的频率。

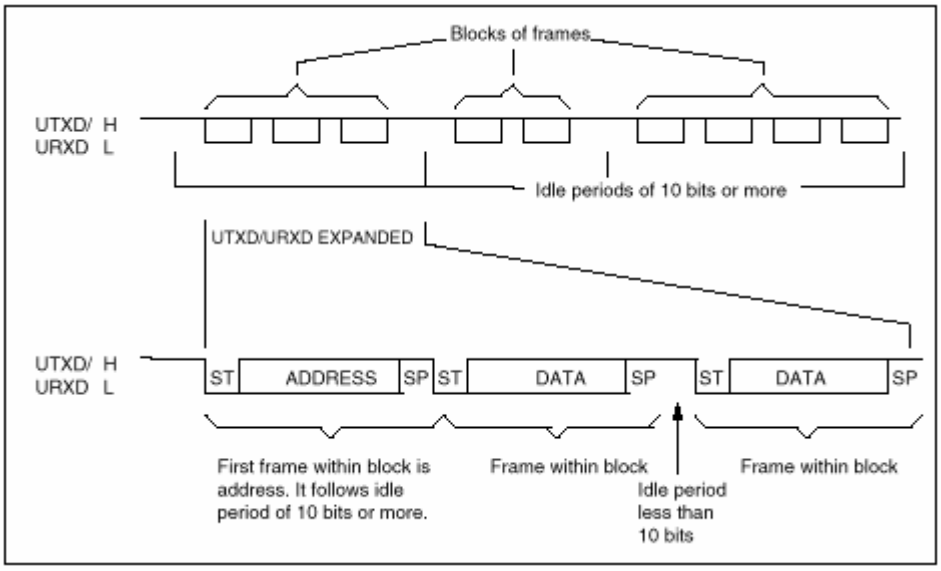
12.1.3 异步通信格式

当选择异步模式时，USART 模块支持 2 种多处理机通信模式。在同一个串行链路上，多个处理机之间可以用这些格式来交换信息。信息以一个多帧数据块，从一个指定的源送达一个或多个目的位置。USART 可识别数据块的起始，并能抑制接收端处理中断和状态信息，直至数据块起始被识别。在这两种多处理机模式下，USART 数据交换过程可以用数据查询方法也可用接收中断来实现。

这两种异步多处理机协议，即线路空闲和地址位多处理机模式，实现了在多处理机通信系统间的有效数据传输。它们也用于使系统的激活状态压缩至最低，以节省电流消耗或处理资源。控制寄存器的 MM 位用来确定是地址位模式还是线路空闲多处理机模式。这两种格式采用唤醒发送、地址特性（TXWake 位）和激活 RXWake 位等功能。URXWIE 和 URXIE 位控制这些模式的发送和接收。

12.1.4 线路空闲多处理机模式

在这种模式下，数据块被一段空闲时间分隔。在字符的第一个停止位之后接收到 10 个以上的“1”，则表示检测到接收线路空闲。



12.6：线路空闲多处理机协议

如果采用 2 位停止位，则第 2 个停止位看作空闲周期的第 1 个传号。空闲周期后的第一个字符是地址字符。RXWake 位可作为地址字符的标记。在线路空闲多处理机格式中，当接收字符是地址字符时 RXWake 置位，并送入接收缓存中。

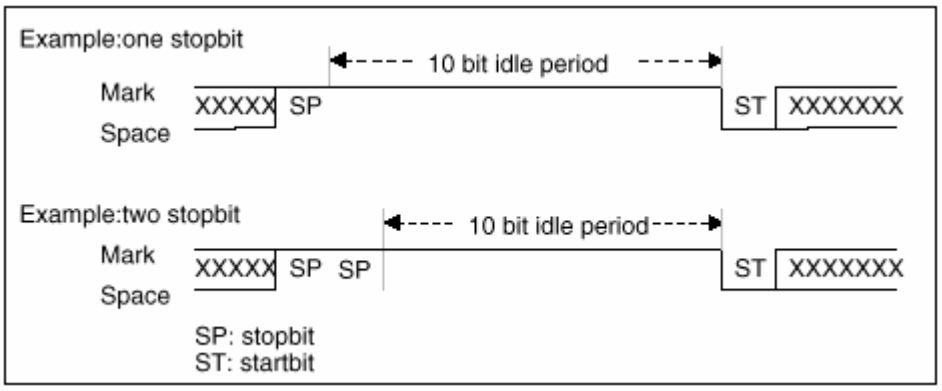


图 12.7：USART 接收空闲检测

正常情况下，如果 USART 接收控制寄存器的 URXWIE 置位，字符在接收器以通常方法拼装，但是不会将该字符送入接收缓存 URXBUF，也不会产生中断。只有当接收到地址字符时，接收器将暂时激活，字符送入 URXBUF，同时将中断标志 URXIFG 置位。相应的错误状态标志位也置位。应用软件可以验证接收到的地址。如果地址匹配，应用软件将处理后续字符并执

行适当的操作。如果地址不匹配，处理机等待下一个地址字符的到来。URXWIE 位不会自动修改的：必须由用户根据接收地址字符和非地址字符的需要去修改。

在线路空闲多处理机模式下，为了产生有效的地址字符识别，可以产生精确的空闲周期。与 TXWake 位相关的是临时唤醒标志（WUT）。WUT 是一个内部标志，与 TXWake 一起构成双缓存。当发送器从 UTXBUF 装入数据，WUT 也从 TXWake 装入，同时 TXWake 复位。

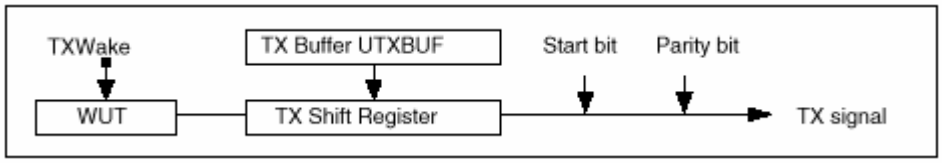


图 12.8：双缓存的 WUT 和 TX 移位寄存器

发送空闲帧以标识一个地址字符按下面步骤完成：

首先 TXWake 位应置位，将任意数据（内容无关）写入 UTXBUF（UTXIFG 应置位）。当发送移位寄存器空时（TXEPT 置位），则 UTXBUF 的内容被送入发送移位寄存器，同时 TXWake 的值移入 WUT。如果此时 WUT 置位，则要发送的起始位、数据位和校验位被抑制，发送一个正好 11 位的空闲周期。在地址字符识别空闲周期之后移出串行端口的下一个数据是 TXWake 置位后写入 UTXBUF 中的第二个字符。当地址识别被发送后，写入 UTXBUF 中的第一个字符被抑制并在以后被忽略。将任意内容的字符写入 UTXBUF 是必要的，只有这样才能使 TXWake 的值移入 WUT 中。

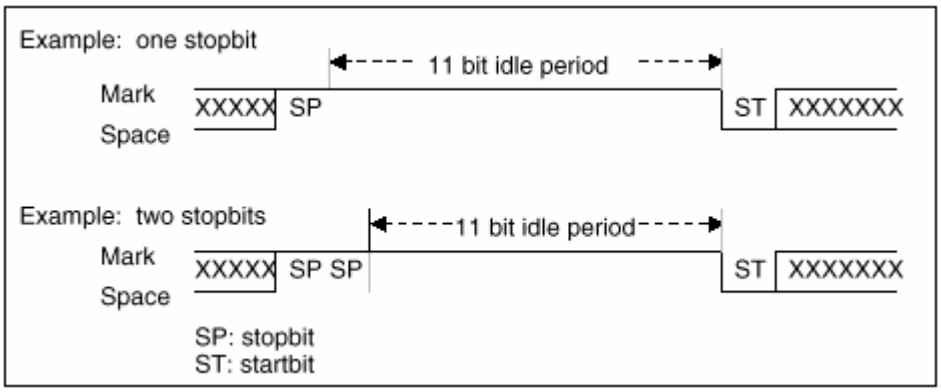


图 12.9：USART 发送空闲周期的产生

12.1.5 地址位格式

在这种模式下，字符包含一额外的位作为地址标识。数据块的第一个字符携带一个置位的地址位，以表明该字符是一地址。当接收字符是地址时 RXWake 位置位，并且被送入接收缓存 URXBUF（允许接收时）。

正常情况下，如果 USART 的 URXWIE 位置位，数据字符按通常方式在接收器拼装，但是

它们不会送入接收缓存 URXBUF，也不会产生中断。只有当接收到一个地址位置位的字符，接收器暂时被激活，字符送入 URXBUF，同时 URXIFG 置位。相应的错误状态标志将置位。应用软件按有效利用资源、降低功耗的原则作后续操作。应用软件可验证接收到的地址。如果匹配，处理机将读取数据块的后续数据；如果地址不匹配，处理机将等待下一地址字符的到来。

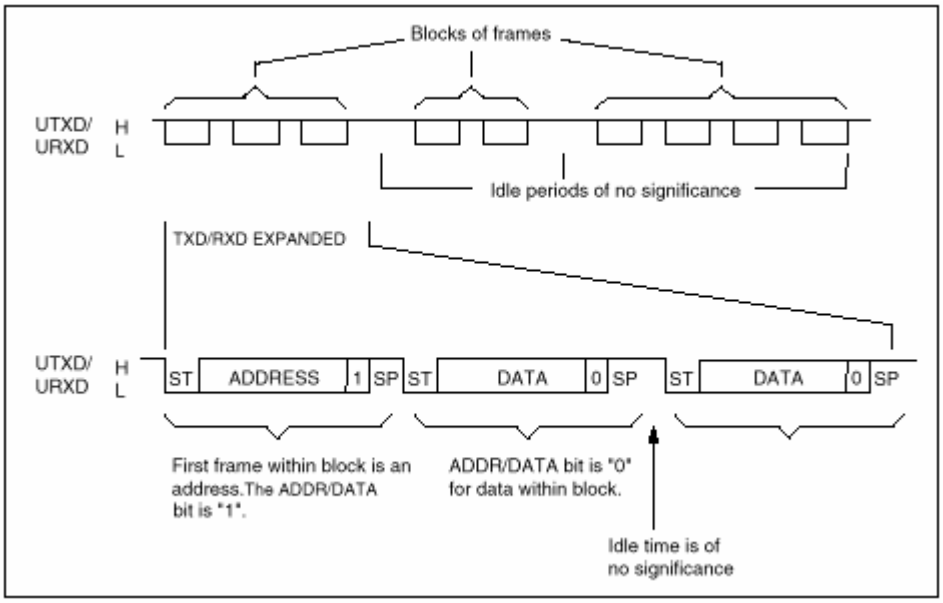


图 12.10：地址位多处理机协议

在地址位多处理机模式下，通过写 TXWake 位控制字符的地址位。每当字符从 UTXBUF 传送至发送器时，TXWake 位装入字符的地址位。然后 TXWake 位将被 USART 清除。

12.2 中断和控制功能

USART 外围模块有 2 个主要中断源，即发送和接收。2 个独立的中断向量，一个用于接收中断事件，另一个用于发送中断事件。

USART 中的控制位位于 SFR 地址中：

- 接收中断标志　　URXIFG　　初始状态复位（用 PUC/SWRST）
- 接收中断允许　　URXIE　　初始状态复位（用 PUC/SWRST）
- 接收允许　　URXE　　初始状态复位（用 PUC）
- 发送中断标志　　UTXIFG　　初始状态置位（用 PUC/SWRST）
- 发送中断允许　　UTXIE　　初始状态复位（用 PUC/SWRST）
- 发送允许　　UTXE　　初始状态复位（用 PUC）

接收器和发送器是完全独立操作的。但是使用同一个波特率发生器。发送和接收使用相同的波特率。

12.2.1 USART 接收允许

接收允许位 URXE 的置位或复位，允许或禁止接受器从 URXD 数据线路接收位流。禁止 USART 接收器，如已开始一次接收操作，在完成后停止接收操作；如无接收操作进行，将立即停止接收操作。起始位检测也同时禁止。

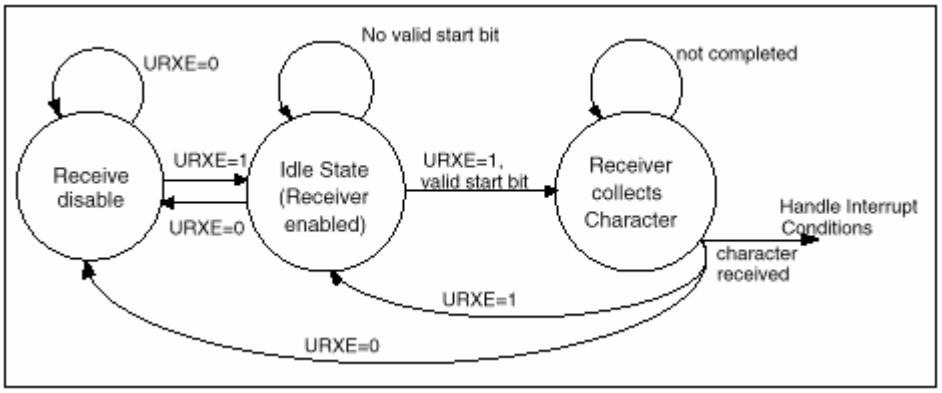


图 12.11：接收允许 URXE 状态图

注意：URXE 重允许，UART 模式
在接收器完全禁止时，接收器额定重允许与通信线路的数据流是异步的。在接受接收数据前查找线路空闲状态可以实现同步。

12.2.2 USART 发送允许

发送允许位 UTXE 的置位或复位，将允许或禁止串行数据线路上字符发送。UTXE 复位时，已激活的发送不会停止，在完成已写入发送缓存内的全部数据发送后发送才被禁止。如果发送已完毕，对发送缓存写入数据不会产生发送操作。

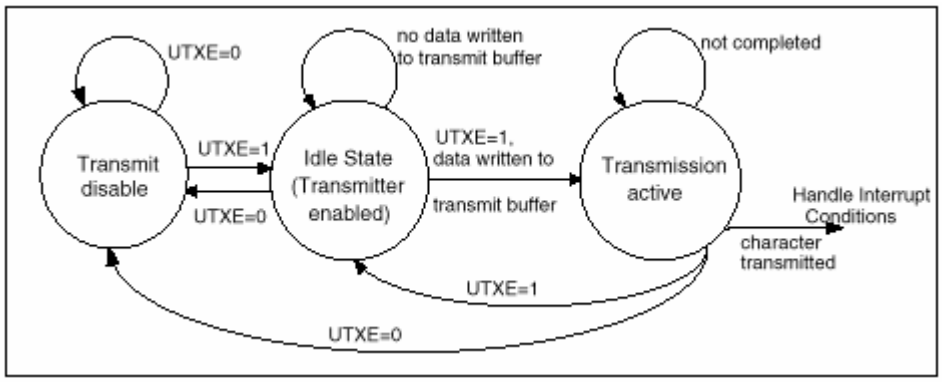


图 12.12：发送允许状态图

当 UTXE 复位时，发送缓存能照常写入，但不会启动发送。一旦 UTXE 置位，缓存内字符的发送立即开始。这一字符可以正确发送。

注意：写 UTXBUF，UART 模式

在 UTXBUF 未就绪而发送允许（UTXE 置位）时，不要对它写入数据。否则移出的字符可能是随机的。

12.2.3 USART 接收中断操作

每次接收字符并装入接收缓存时，接收中断标志 URXIFG 置位或保持不变。

- 当 URXEIE 复位，错误字符（校验、帧或打断错）不会使中断标志 URXIFG 置位，即 URXIFG 不被改变。
- 所有字符类型（URXWIE=0）或仅地址字符类型（URXWIE= 1 ）可使中断标志 URXIFG 置位，以 URXWIE 决定原因。如 URXEIE 也置位，错误字符将使中断标志 URXIFG 置位。

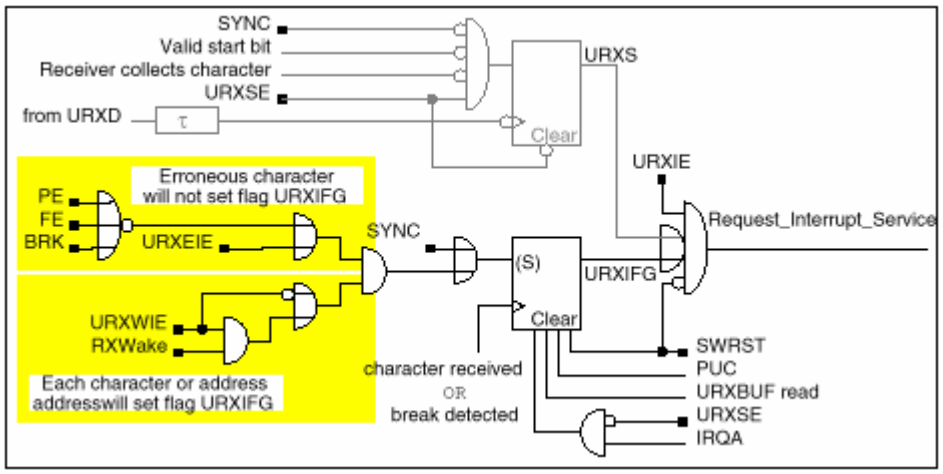


图 12.13：接收中断条件

在系统复位 PUC 或在软件复位 SWRST 时 URXIFG 复位。如执行中断服务（URXSE=0）或接收缓存 URXBUF 被读取，则 URXIFG 自动复位。接收中断标志 URXIFG 置位表示有等待服务的中断事件。如果接收中断允许位 URXIE 置位，允许处理等待的中断请求。接收中断标志位 URXIFG 和接收中断允许位 URXIE 在发生 PUC 和 SWRST 时复位。

URXIFG 可用软件访问，而 URXS 不能用软件访问。当两个中断事件，即接收起始检测和字符接收有效都被软件允许，URXIFG 指示请求中断服务的是字符接收而不是检测到起始位。这样做的原因是一旦中断软件处理接收起始位检测将使 URXSE 复位。这将清除 URXS 位，防止 URXS 又引起一次中断请求。这时 URXIFG 已经复位，因为这时 URXIFG 锁存，已无置位条件。

12.2.3 USART 发送中断操作

发送器使发送中断标志 UTXIFG 置位，表示发送缓存 UTXBUF 已准备好接收下一个字符。如果中断请求的服务软件已启动或对 UTXBUF 写入一个字符，则 URXIFG 被自动复位。这一标志在 UTXIE 和通用中断允许 GIE 置位时引起发送中断。在 PUC 或 SWRST 信号消除时 UTXIFG 置位。

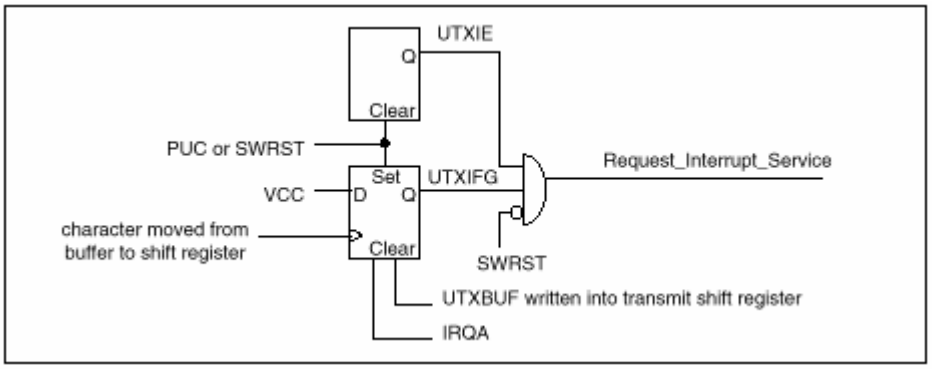


图 12.14：发送中断条件

发送中断允许位 UTXIE 控制 UTXIFG 请求中断的能力，但不阻止 UTXIFG 置位。UTXIE 在发生 PUC 或 SWRST 时复位。PUC 或 SWRST 使 UTXIFG 置位，但是使 UTXIE 复位，这样确保了完全的中断控制能力。

12.3 控制和状态寄存器

USART 模块的硬件是字节结构的，必须用字节指令访问（后缀“B”）。

寄存器	缩写	类型	地址	初始状态
● USART 控制寄存器	UCTL	读/写	070h	见位说明
● 发送控制寄存器	UTCTL	读/写	071h	见位说明
● 接收控制寄存器	URCTL	读/写	072h	见位说明
● 调整控制寄存器	UMCTL	读/写	073h	不变
● 波特率寄存器 0	UBR0	读/写	074h	不变
● 波特率寄存器 1	UBR1	读/写	075h	不变
● 接收缓存	URXBUF	读/写	076h	不变
● 发送缓存	UTXBUF	读	077h	不变

除非在功能描述中说明，发生 PUC 后各位的值都是随机的。

通过 PUC 或 SWRST 位使 USART 执行复位。PUC 后，SWRST 仍保持置位，USART 保持这一状态直到通过 SWRST 的复位来禁止 USART 复位。

SYNC 位决定 USART 模块处于异步还是同步模式。控制寄存器各位的功能在这两种模式下可能不同。本节的各位说明是指异步模式的，即 SYNCV=0。它们在同步模式的功能在 USART 的 SPI 部分说明。

12.3.1 USART 控制寄存器 UCTL

控制寄存器内的信息决定了 USART 的基本操作。如：选择通信协议、通信模式和校验位。在 SWRST 复位使 USART 复位操作禁止前，各位应根据选择的模式进行编程。

UCTL（070h）

7							0
PENA	PEV	SP	CHAR	listen	SYNC	MM	SWRST
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

图 12.15：USART 控制寄存器

位 0：如 SWRST 置位，USART 的状态机构和运行标志初始化成复位状态。所有受影响的逻辑保持在复位状态，直至 SWRST 复位。这意味着一次系统复位后，只有对 SWRST 复位，USART 才能重新被允许。接收和发送允许标志 URXE 和 UTXE 不会因 SWRST 而更改。

位 1：多处理机模式：（地址/线路空闲唤醒）

USART 模块支持两种多处理机协议：线路空闲和地址位。选择多处理机模式会影响自动地址解码功能的操作。

MM=0：　线路空闲多处理机协议

MM=1：　地址位多处理机协议

MM 复位将执行常规的异步协议。

位 2：USART 模块的模式和功能选择。

SYNC 位选择 USART 外围接口模块的功能。USART 的一些控制位在 UART 模式和 SPI 模式中有不同的功能。

SYNC=0：　UART 模式

SYNC=1： SPI 模式

位 3：Listen 位选择是否将发送数据内部反馈给接收器。

Listen=0： 无反馈

Listen=1： 发送信号由内部反馈给接收器。每个 MSP430 的 USART 发送的数据同时被 USART 的接受器接收，因此不再接收任何外部信号。

位 4：字符长度

选择字符以 7 或 8 位发送。7 位字符不用 URXBUF 和 UTXBUF 的最高位，这一位填“0”。

CHAR=0： 7 位

CHAR=1： 8 位

位 5：停止位数

决定发送时停止位数。但是接收器只检测 1 位停止位。

SP=0： 1 位停止位

SP=1： 2 位停止位

位 6：校验奇偶位

如 PENA 置位（校验允许），PEV 位按发送或接收字符、地址位（地址位多处理机模式）和校验位中的“1”的数量定义奇校验或偶校验。

PEV=0： 奇校验

PEV=1： 偶校验

位 7：校验允许位

如果禁止校验，发送时不会产生校验位，接收时也不期望收到这一位。因为校验位不是数据位之一，接收到的校验位不传送入 URXBUF 中。在地址位多处理机模式中，地址位包括在校验计算中。

PEN=0： 校验禁止

PEN=1： 校验允许

注意：传号、空号定义

传号电平与空闲状态信号电平相同。空号电平与传号电平相反：起始位总是空号。

12.3.2 发送控制寄存器 UTCTL

寄存器 UTCTL 控制与发送操作相关的 USART 硬件。

UTCTL（071h）

7							0
未用	CKPL	SSEL1	SSEL0	URXSE	TXWake	未用	TXEPT
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

图 12.16：USART 发送控制寄存器

位 0：发送器空标志 TXEPT 在发送移位寄存器和 UTXBUF 空时置位，当数据写入 UTXBUF 时复位。它在 SWRST 时置位。

位 1：未用

位 2：TXWake 位用于控制多处理机通信模式的发送特性。通过装入 UTXBUF 起动一次发送，用 TXWake 位状态初始化地址识别特性。它不必清除，一旦被传送至 WUT(“暂时唤醒”)，

USART 硬件自动将它清除；SWRST 也能它清除。

位 3：接收触发沿控制位置位请求接收中断服务。这时相应的允许位和 GIE 应置位。用这一位的优点是能在中断服务程序中启动微控制器时钟系统，包括 MCLK，还能通过调整模式控制位来保持运行。即使系统因关闭 MCLK 进入低功耗模式，USART 也能在选定的 MCLK 下工作。

位 4、5：时钟源选择 0 和 1。时钟源选择位确定用于波特率发生器的时钟源。

SSEL1、0　0　　选择外部时钟 UCLKI

1　　选择辅助时钟 ACLK

2、3　　选择系统主时钟 MCLK

位 6：时钟极性 CKPL。CKPL 位控制 UCLK1 信号的极性

CKPL=0：　UCLKI 信号与 UCLK 极性相同

CKPL=1：　UCLKI 信号与 UCLK 极性相反

位 7：未用

12.3.3 接收控制寄存器 URCTL

URCTL 控制与接收操作相关的 USART 硬件并保存由最新写入 URXBUF 的字符引起的出错状况和唤醒条件。一旦 FE、PE、OE、BRK、RXERR 或 RXWake 的任何一位置位，不能通过接收下一个字符来复位。它们的复位要通过访问接收缓存 URXBUF、USART 的软件复位 SWRST、系统复位 PUC 或用指令修改。

URCTL（072h）

7							0
FE	PE	OE	BRK	URXEIE	URXWIE	RXWake	RXERR
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

图 12.17：USART 接收控制寄存器

位 0：接收错误位 RXERR 置位表明有一个或多个出错标志（FE，PE，OE 或 BRK）置位。用指令清除出错位，它不会复位。

位 1：接收唤醒检测。RXWake 位

当接收的字符是一地址字符且被送入接收缓存时，置位。

地址位多处理机模式：　接收字符地址位置位时，置位。

线路空闲多处理机模式：接收字符前检测到 URXD 线路空闲（11 位传号），置位。

访问接收缓存 URXBUF、USART 软件复位 SWRST 或系统复位 PUC 都使 RXWake 复位。

位 2：接收唤醒中断允许位 URXWIE 选择设置中断标志位的字符类型。

URXWIE=0：接收到的每一个字符都将使 URXIFG 置位。

URXWIE=1：只有地址字符才使 URXIFG 置位，在两种多处理机模式中工作相同。

唤醒中断允许的特征依赖于接收出错字符的特征。见 URXEIE 位说明。

位 3：接收出错字符中断允许位 URXEIE 选择允许出错字符使 URXIFG 置位。

URXEIE=0：每一个接收的出错字符都不改变 URXIFG 位

URXEIE=1：根据 URXWIE 位的设置，所有字符都可使 URXIFG 置位。

URXEIE	URXWIE	字符出错	地址字符	接收字符后的 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

位 4：当一次打断发生并且 URXEIE 置位，打断检测位 BRK 置位。RXD 线路从丢失的第一个停止位开始连续出现至少 10 位低电平被识别为打断。在检测到打断后，通过接收字符是不会清除标志。只有通过 SWRST、系统复位和读取 URXBUF 来复位。

位 5：当一个字符写入 URXBUF 时前一字符还未被读出，则溢出标志 OE 置位。这时前一字符被覆盖而丢失。OE 通过 SWRST 位、系统复位和读取 URXBUF 来复位。

位 6：当一个接收字符中“1”的个数和它的校验位不符并被装入接收缓存时，则校验错标志 PE 置位。校验计算包括地址位（用于地址位多处理机模式）。校验发生和检测被禁止时标志无用。这时读出为“0”。它通过 SWRST、系统复位和读 URXBUF 来复位。

位 7：当一个接收字符的停止位为“0”并被装入接收缓存时，帧错标志被置位。即使用多停止位模式，也只有第一个停止位被检测。丢失停止位意味着从起始位开始的同步特性已丧失，字符的帧是错的。FE 用 SWRST、PUC 和读 URXBUF 来复位。

注意：接收状态控制位

接收状态控制位 FE、PE、OE、BRK 和 RXWake 由硬件根据接收字符的条件置位。它们一旦置位就会一直保持，直到用软件直接复位或读取接收缓存。未清除的出错位可能引起错误的字符解释或失去中断能力。

12.3.4 波特率选择和调制控制寄存器

波特率产生器利用波特率选择寄存器 UBR1 和 UBR0，以及调整控制寄存器，来产生串行数据流的位定时。

UBR0（074h）

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

UBR1（075h）

7							0
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
rw	rw	rw	rw	rw	rw	rw	rw

图 12.18：USART 波特率选择寄存器

波特率 = BRCLK /（UBR+(m7+m6+..+m0)/8）

波特率控制范围： 3 ≤ UBR ＜ 0FFFFh， UBR=[UBR1，UBR0]

即使晶振频率不是所需波特率的整数倍，调整控制寄存器与 UBR0/1 确保产生适当的定时时间。

UMCTL（073h）

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

图 12.19：USART 调整控制寄存器

如果位m_i置位，对应位的定时时间按波特率分频器的输入时钟扩展一个时钟周期。

每接收或发送一位，在调整控制寄存器的下一位被用来决定当前位的定时时间。协议的第一位（即起始位）的定时由UBR加上m_0 决定，下一位由UBR加上M_1 决定.....

调整顺序为：

m0 - m1 - m2 - m3 - m4 - m5 - m6 - m7 - m0 - m1 - m2 - .....

12.3.5 USART 接收数据缓存 URXBUF

接收缓存 URXBUF 含有此前从接收移位寄存器来的数据。读取 URXBUF 中数据，将使复位接收出错位、RXWake 位和中断标志 URXIFG 位复位。

URXBUF（076h）

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

图 12.20：USART 接收缓存

在 7 位字长模式，URXBUF 的最高位总是复位。

当接收和控制条件为真，接收缓存装入当前接收的字符。

URXEIE	URXWIE	装入 URXBUF	PE	PE	BRK
0	1	无差错地址字符	0	0	0
1	1	所有的地址字符	x	x	x
0	0	无差错字符	0	0	0
1	0	所有字符	x	x	x

12.3.6 USART 发送数据缓存 UTXBUF

发送缓存含有当前要由发送器发送的数据。

UTXBUF（077h）

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

图 12.21：USART 发送缓存

UTXIFG 标志表示 UTXBUF 已准备好接收下一个要发送的字符。

将数据写入 UTXBUF 初始化发送功能。如果发送移位寄存器为空或即将为空，数据的发送立即开始。

只有当 UTXBUF 为空时，数据才能写入缓存，否则可能发送不可预料的字符。

12.4 UART 模式，低功耗模式应用特性

MSP430 有许多的功能和操作特性支持实现基于 MSP430 结构的超低功耗系统：

- 利用对 UART 帧的敏感作为起动条件，系统可从任意的处理模式开始运行
- 用最低输入时钟频率来实现波特率
- 支持多处理机模式来减少 MSP430 资源的使用

12.4.1 由 UART 帧启动接收操作

当波特率发生用系统主时钟 MCLK 时，能最有效地在接收通道作起始检测，而整个系统可能并不需要 MCLK 工作。接收的起始条件 URXD 信号的下降沿。在 URXIE 和 GIE 允许时，每当它触发中断标志 URXS 时将请求一次中断服务。这样 MSP430 将返回到活动模式并具备在 MCLK 和 ACLK 活动状态下的全部系统性能。

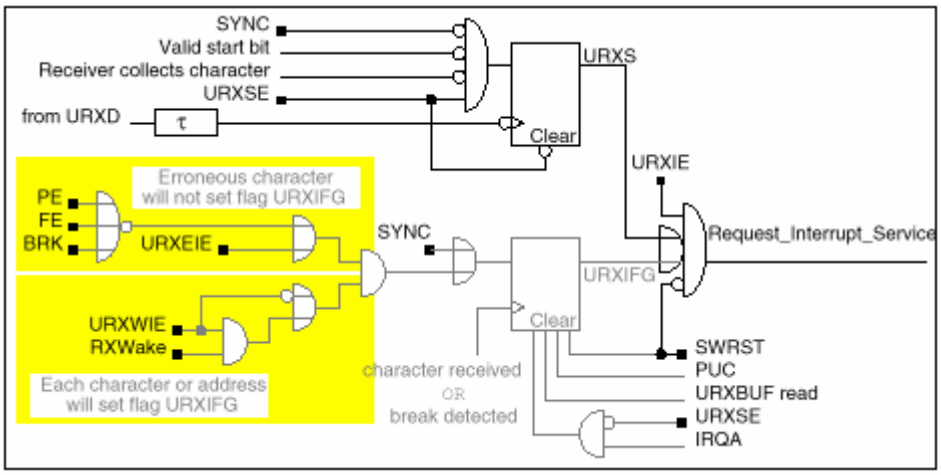


图 12.22：接收起动检测

有三种字符流不能使中断标志 URXIFG 置位：

- 错误字符（URXEIE=0）
- 地址字符（URXWIE=1）
- 检测到无效起始位

中断软件应处理这些情况。中断处理必须正确设置时钟系统环境；时钟系统将连续运行（即有电流消耗），直到通过软件修改配置。如果 CPU 工作在活动模式，且时钟系统正常工作，就不必使用起始条件检测。

起动条件

进入 USART 模块的 URXD 信号首先进入去毛刺电路，使毛刺不能触发接收起始位标志 URXS。这可防止 URXD 线路上的小毛刺起动通信模块。在噪声环境里，因为毛刺不会起动系统和 USART，电流消耗也会降低。

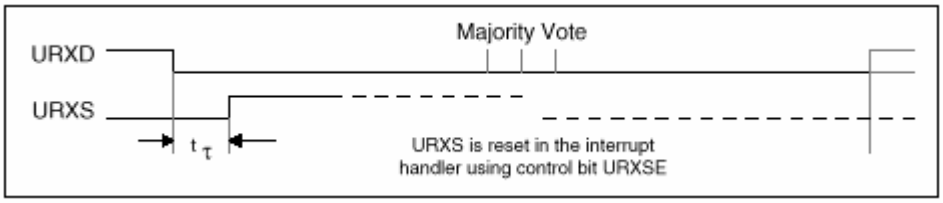


图 12.23：用 URXS 标志接收起始位时序，接受起始位

当URXD信号超过去毛刺时间t_τ,但是起始位的多数表决检测失败,UART停止字符的接收。软件必须处理这种情况使系统保持在一个适当的低功耗模式，中断标志URXIFG不会置位。

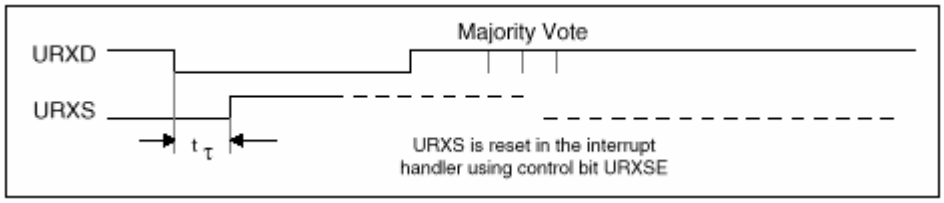


图 12.24：用 URXS 标志接收起始位时序，不接受起始位

MSP430 启动后，URXD线路上的毛刺被自动压制，不会影响系统。去毛刺时间t_τ的数据在相应的器件说明书中有说明。

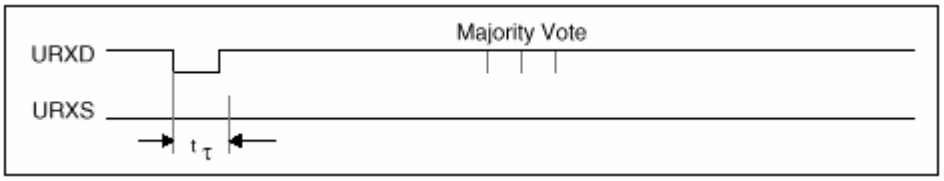


图 12.25：用 URXS 标志接收起始位时序，毛刺抑制

中断服务将使控制寄存器 UCTL 的 URXSE 复位，以防止 URXS 再次请求中断服务。同时允许接收中断标志 URXIFG 的功能。

```
*******************************************************************
*    INTERRUPT HANDLER FOR FRAME START CONDITION AND              *
*    CHARACTER RECEIVE                                            *
*******************************************************************
IFG2      .EQU      3              ; URXIFG AND UTXIFG IN ADDRESS 3
UTCTL     .EQU      71H            ;
```

```
UTXIFG    .EQU      0              ;
URXSE     .EQU      8              ;
                                   ;
URX_INT   BIT.B     #URXIFG,&IFG2  ; TEST URXIFG SIGNAL TO CHAECK
          JNE       ST_COND        ; IF FRAME START CONDITION
          .....
          .....
ST_COND   BIC.B     #URXSE,&UTCTL  ; CLEAR FF/SIGNAL URXS, STOP
                                   ; FURTHER INTERRUPT REQUESTS
          BIS.B     #URXSE,&UTCTL  ; PREPARE FF_URXS FOR NEXT FRAME
          .....                    ;.START CONDITION
          .....                    ; AND SET THE CONDITIONS TO RUN
          .....                    ; THE CLOCK NEEDED FOR UART RX
```

注意： UART 时钟暂停使检测打断位中止

如果 UART 工作在用起始位唤醒的状态下并且在完成一个字符的接收后关闭 UCLK，通信线路的打断状态不会被 UART 硬件自动检测到。打断检测需要波特率发生器输出的时钟信号 BRSCLK，但是因为失去 UCLK 它已停止了。

12.4.2 时钟频率的充分利用与 UART 的波特率

电流消耗与时钟频率呈线性关系。它应该在满足应用条件前提下保持最低频率。因为各种原因需要快速通信：如工业过程的校准和测试，关键应用场合的报警，人们寻求信息时的响应时间 .....

MSP430 的 USART 波特率发生器能产生的波特率可达时钟频率的 1/3。波特率时间的附加调整带来极大好处，因为帧的每一位定时都可调整。定时的逐位调整甚至可以不用整数分频。波特率能做到在 32768Hz 晶振时得到 4800 波特，最大误差为 11% 。而标准 UART 在同样条件下只能达到的波特率为 75，最大误差已经达到－14.6% 。

12.4.3 多处理机模式对节约 MSP430 资源的支持

多字符协议的通信系统可利用线路空闲或者地址位这两种多处理机模式。第一个字符可能是目标地址、信息标识或其它定义。这一字符由软件解释。如果对于应用有一定的意义，则采集其后继字符并进一步明确它的定义。毫无意义的首字符将停止处理设备的活动。这一特性的应用支持在接收操作时的中断唤醒特性和用发送来传递唤醒条件。避免无意义字符激活系统节约了 MSP430 资源的使用，使系统能维持在最有效的功率消耗状况下。

在多处理机模式中，错误字符的拒收避免中断处理这些字符。这点非常有用。处理机将在最有效的功率消耗模式下等待一个需要处理的字符到来。

12.5 波特率计算

MSP430 的波特率发生器使用一个分频器和一个调整器，分频因子 N 由给定的晶振频率和所需的波特率决定。

N = BRCLK/波特率

所需的分频因子 N 通常有一个整数部分和一个小数部分。波特率发生器的分频器实现了分频因子 N 的整数部分，调整器尽可能接近地满足小数部分。分频因子 N 定义为：

N = UBR+(m7+m6+..+m0)/8

其中： N 为目标分频因子
UBR 是以 16 位表示的寄存器 UBR1 和 UBR0
m_I 表示当前的调整数据

波特率 = BRCLK / N = BRCLK /（UBR+(m7+m6+..+m0)/8）

发送操作的位定时

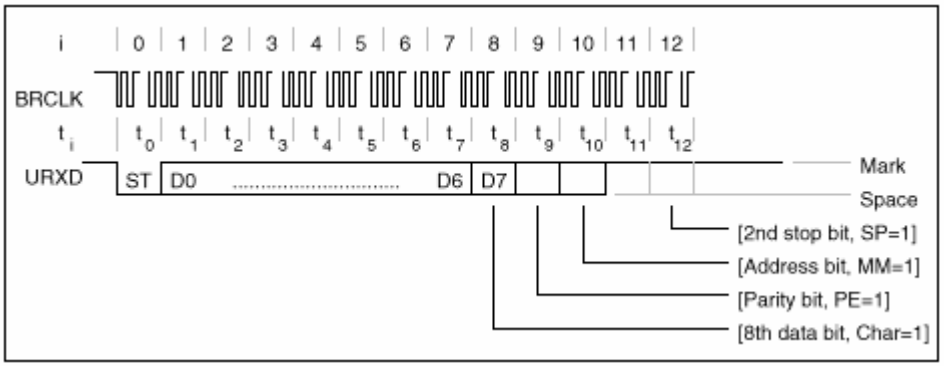


图 12.26：MSO430 发送位定时

一帧和一个字符中每一位的定时是实际位传输时间的总和。对于每一位，定时计算考虑了波特率发生与理想定时间的误差。相关的信息是相对于实际位的误差，而不是相对全体的误差。

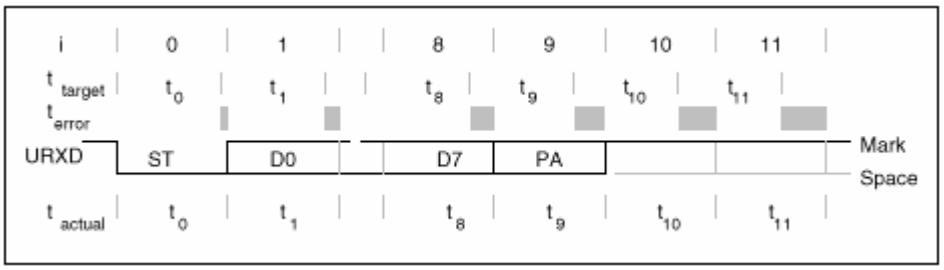


图 12.27：MSP430 发送位定时误差

即使每位误差很小，最终也会产生大的误差。必须考虑到它们是积累的而不是相对的。。每一位的误差可通过下面公式来计算：

$$\text{Error [\%]} = \frac{\sum_{i=0}^{n-1} t_{actual_i} - \sum_{i=0}^{n-1} t_{target_i}}{t_{baud\ rate}} \times 100\%$$

或

$$\text{Error [\%]} = \left(\left(\frac{\text{baud rate}}{\text{BRCLK}} \times ((i+1) \times \text{UBR} + \sum_{i=0}^{n-1} m_i) - (i+1)\right) \times 100\%\right.$$

其中， baudrate 是所需波特率
BRCLK 是输入频率，选自 UCLK、ACLK 或 MCLK
i=0 针对起始位、i=1 针对 D0 .....
UBR 是分频因子，在 UBR1 和 UBR0 中

例 1：

设： baudrate=2400 波特
BRCLK=32768Hz（ACLK）
UBR=13，因为理想的分频因子为 13.67
M=6Bh，m_7=0，m_6=1，m_5=1，m_4=0，m_3=1，m_2=0，m_1=1，m_0=1，最低位m_0最先使用。

起始 误差[%]=（ 波特率 / BRCLK *（（ 0 + 1）* UBR + 1）- 1）* 100% = 2.54%
D0 误差[%]=（ 波特率 / BRCLK *（（ 1 + 1）* UBR + 2）- 2）* 100% = 5.08%
D1 误差[%]=（ 波特率 / BRCLK *（（ 2 + 1）* UBR + 2）- 3）* 100% = 0.29%
D2 误差[%]=（ 波特率 / BRCLK *（（ 3 + 1）* UBR + 3）- 4）* 100% = 2.83%
D3 误差[%]=（ 波特率 / BRCLK *（（ 4 + 1）* UBR + 3）- 5）* 100% = -1.95%
D4 误差[%]=（ 波特率 / BRCLK *（（ 5 + 1）* UBR + 4）- 6）* 100% = 0.59%
D5 误差[%]=（ 波特率 / BRCLK *（（ 6 + 1）* UBR + 5）- 7）* 100% = 3.13%
D6 误差[%]=（ 波特率 / BRCLK *（（ 7 + 1）* UBR + 5）- 8）* 100% = -1.66%
D7 误差[%]=（ 波特率 / BRCLK *（（ 8 + 1）* UBR + 6）- 9）* 100% = 0.88%
校验 误差[%]=（ 波特率 / BRCLK *（（ 9 + 1）* UBR + 7）-10）* 100% = 3.42%
停止 1 误差[%]=（ 波特率 / BRCLK *（（10 + 1）* UBR + 7）-11）* 100% = -1.37%
停止 2 误差[%]=（ 波特率 / BRCLK *（（11 + 1）* UBR + 8）-12）* 100% = 1.17%

标准波特率所需要的波特率寄存器和调整寄存器列表如下，针对 32768Hz 手表晶振(ACLK)并假定 MCLK 是 ACLK 的 32 倍条件下。列出了发送和接收的计算误差。在接收时还应考虑同步误差。

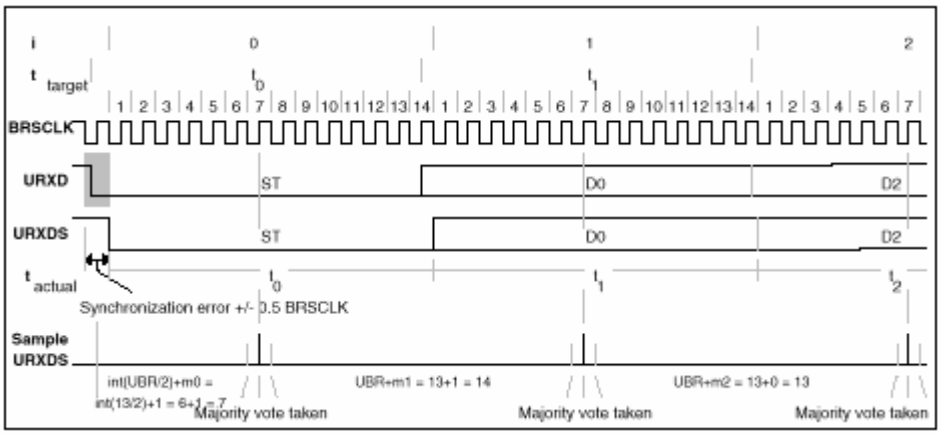
波特率	分频信号		ACLK (32768Hz)			max. TX 误差	max. RX 误差	同步 RX 误差	MCLK (1048576Hz)			max. TX 误差	max. RX 误差
	ACLK	MCLK	UBR1	UBR0	UMOD	%	%	%	UBR1	UBR0	UMOD	%	%
75	436.91	13981	1	B4	FF	-.1/.3	-.1/.3	+/-2	36	9D	FF	0/.1	+/-2

110	297.89	9532.51	1	29	FF	0/.5	0/.5	+/-3	25	3C	FF	0/.1	+/-3
150	218.45	6990.5	0	DA	55	0/.4	0/.4	+/-2	1B	4E	FF	0/.1	+/-2
300	109.23	3495.25	0	6D	22	-.3/.7	-.3/.7	+/-2	0D	A7	00	-.1/0	+/-2
600	54.61	1747.63	0	36	D5	-1/1	-1/1	+/-2	06	D3	FF	0/.3	+/-2
1200	27.31	873.81	0	1B	03	-4/3	-4/3	+/-2	03	69	FF	0/.3	+/-2
2400	13.65	436.91	0	0D	6B	6/3	-6/3	+/-4	01	B4	FF	0/.3	+/-2
4800	6.83	218.45	0	06	6F	-9/11	-9/11	+/-7	0	DA	55	0/.4	+/-2
9600	3.41	109.23	0	03	4A	-21/12	-21/12	+/-15	0	6D	03	-.4/1	+/-2
19200		54.61							0	36	6B	-.2/2	+/-2
38400		27.31							0	1B	03	-4/3	+/-2
76800		13.65							0	0D	6B	-6/3	+/-4
115200		9.10							0	09	08	-5/7	+/-7

表 12.2：常用波特率，波特率数据与误差

同步误差来源于 URXD 端口数据信号与内部时钟系统之间的异步定时。接收信号与 BRSCLK 时钟是同步的。BRSCLK 时钟频率是位定时的 16 至 31 倍。

BRSCLK=BRCLK	当		N	≤1F
BRSCLK=BRCLK/2	当	20h	≤N	≤3Fh
BRSCLK=BRCLK/4	当	40h	≤N	≤7Fh
BRSCLK=BRCLK/8	当	80h	≤N	≤FFh
BRSCLK=BRCLK/16	当	100h	≤N	≤1FF
BRSCLK=BRCLK/32	当	200h	≤N	≤3FFh
BRSCLK=BRCLK/64	当	400h	≤N	≤7FFh
BRSCLK=BRCLK/128	当	800h	≤N	≤FFFh
BRSCLK=BRCLK/256	当	1000h	≤N	≤1FFFh
BRSCLK=BRCLK/512	当	2000h	≤N	≤3FFFh
BRSCLK=BRCLK/1024	当	4000h	≤N	≤7FFFh
BRSCLK=BRCLK/2048	当	8000h	≤N	≤FFFFh



对于起始位检测的目标波特率定时$t_{target0}$是波特率定时$t_{baud\ rate}$的一半，因为位测试是在位周期的中间进行的。其它后续位的目标波特率定时$t_{targeti}$就是波特率$t_{baud\ rate}$。

$$Error\ [\%] = \frac{t_{actual0} + t_{target0}}{0.5 \times t_{target0}} + \frac{\sum_{i=1}^{n-1} t_{actuali} - \sum_{i=1}^{n-1} t_{targeti}}{t_{target}} \times 100\%$$

或

$$Error\ [\%] = \left(\frac{baud\ rate}{BRCLK} \times \{2 \times [m0 + int(UBR/2)] + (i \times UBR + \sum_{i=1}^{n-1} m_i)\} - 1 - i\right) \times 100\%$$

其中， baudrate 是所需波特率
BRCLK 是输入频率，选自 UCLK、ACLK 或 MCLK
i=0 针对起始位、i=1 针对 D0 ......
UBR 是分频因子，在 UBR1 和 UBR0 中

例 2

设： baudrate=2400 波特
BRCLK=32768Hz（ACLK）
UBR=13，因为理想的分频因子为 13.67
M=6Bh，m_7=0，m_6=1，m_5=1，m_4=0，m_3=1，m_2=0，m_1=1，m_0=1，最低位m_0最先使用。

起始	误差[%]=(波特率/BRCLK*(2*(1+6)+(0*UBR+0))-0- 1)*100% = 2.54%
D0	误差[%]=(波特率/BRCLK*(2*(1+6)+(1*UBR+1))-1- 1)*100% = 5.08%
D1	误差[%]=(波特率/BRCLK*(2*(1+6)+(2*UBR+1))-1- 2)*100% = 0.29%
D2	误差[%]=(波特率/BRCLK*(2*(1+6)+(3*UBR+2))-1- 3)*100% = 2.83%
D3	误差[%]=(波特率/BRCLK*(2*(1+6)+(4*UBR+2))-1- 4)*100% = -1.95%
D4	误差[%]=(波特率/BRCLK*(2*(1+6)+(5*UBR+3))-1- 5)*100% = 0.59%
D5	误差[%]=(波特率/BRCLK*(2*(1+6)+(6*UBR+4))-1- 6)*100% = 3.13%
D6	误差[%]=(波特率/BRCLK*(2*(1+6)+(7*UBR+4))-1- 7)*100% = -1.66%
D7	误差[%]=(波特率/BRCLK*(2*(1+6)+(8*UBR+5))-1- 8)*100% = 0.88%
校验	误差[%]=(波特率/BRCLK*(2*(1+6)+(9*UBR+6))-1- 9)*100% = 3.42%
停止 1	误差[%]=(波特率/BRCLK*(2*(1+6)+(10*UBR+6))-1-10)*100% = -1.37%
停止 2	误差[%]=(波特率/BRCLK*(2*(1+6)+(11*UBR+7))-1-11)*100% = 1.17%

波特率总结

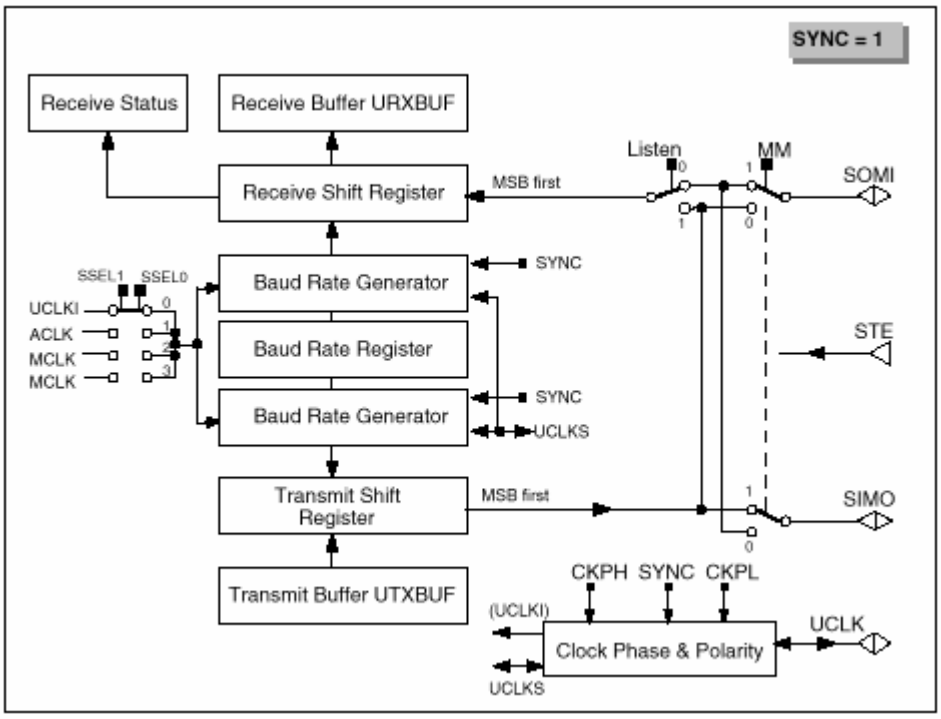
系统选择产生一合适的串行通信位流，它允许波特率接近 USART 的输入时钟。通过调整单独一位的定时能达到低的积累误差。实际上，如果误差在 20% 至 30% 以内，是可以进行正确的串行通信的。

13．USART 外围接口，SPI 模式

同步接口是一个串行通道，允许 7 位或 8 位数据流以外部或内部编程的速率移入或移出 MSP430。USART 以字节外围模块与 CPU 连接。它通过 3 或 4 个外部引脚使微控制器与外部系统相连。

USART 的串行同步通信特性：

- 控制寄存器中的控制位 SYNC 置位，选择同步通信模式。
- 经 SOM1、S1MO、UCLK 和 STE 引脚，支持 3 或 4 线 SPI 操作。
- 可选择主模式或从模式。
- 接收和发送有各自的移位寄存器（URXBUF 和 UTXBUF）。
- 接收和发送双缓存。
- 时钟极性和时钟相位可控。
- 主模式的时钟频率可控。
- 7 或 8 位字符长度。



3.1：USART 原理框图，SPI 模式

13.1 USART 同步操作

同步模式数据和时钟信号用于发送和接收串行数据。主机提供时钟和数据，从机利用这一时钟将串行信息移进或移出。4 线 SPI 模式用附加控制线，来允许从机数据的发送和接收。它由主机控制。

有 3 或 4 个信号用于数据交换：

- SIMO： 从进、主出
- SOMI： 从出、主进
- UCLK： USART 时钟，由主机驱动，从机用它发送和接收数据
- STE： 从机发送允许，用于 4 线模式中控制多主从系统中的多个从机

下图说明，USART 用同步模式与另一设备的串行通道互联时使用一个公共的发送接收移位寄存器，MSP430 可以是主机或从机。操作是相同的。主机通过发送 UCLK 信号来初始化发送。对于主机，数据在一个时钟沿移出发送移位寄存器，在另一个反向沿移入接收移位寄存器。对于从机，数据移位操相同，用公共的移位寄存器接收和发送数据。主机和从机的发送和接收是同时进行的。

由应用软件来决定数据是否有意义：

1. 主机发送数据，从机发送伪数据
2. 主机发送数据，从机发送数据
3. 主机发送伪数据，从机发送数据

主机可在任意时候初始化发送并控制 UCLK。由软件协议来决定主机知道何时从机广播数据。

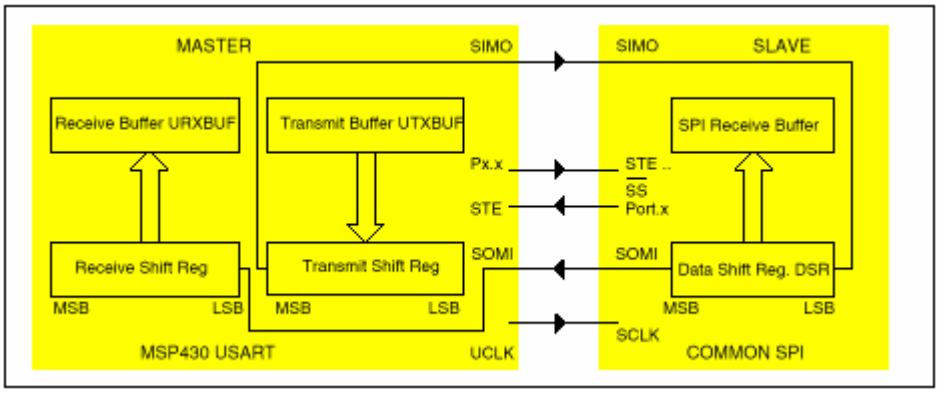
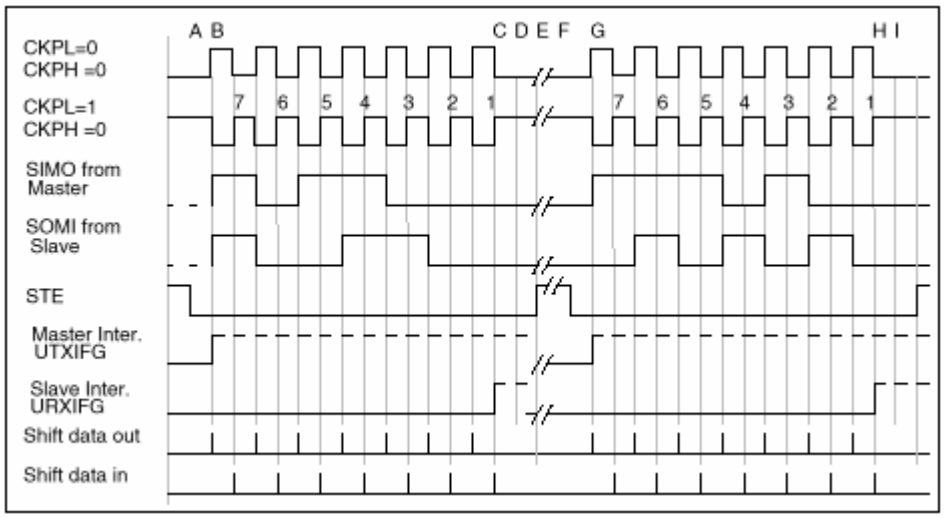
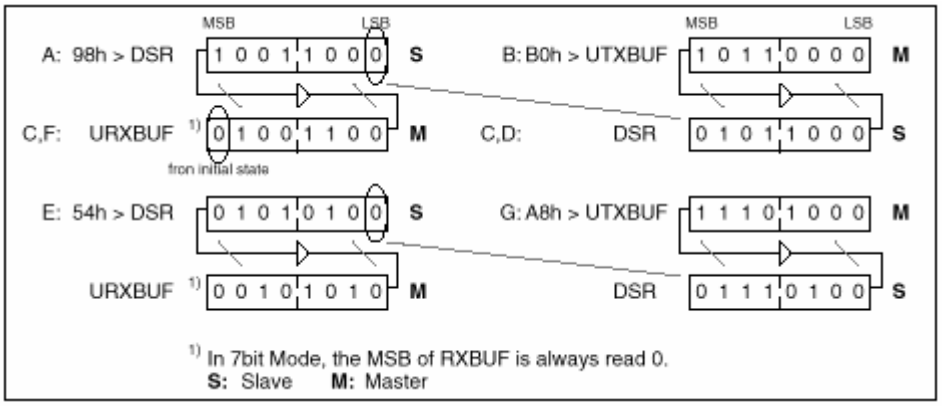


图 13.2：MSP430USART 作为主机，具有 SPI 的设备作为从机

以下是 7 位字长串行同步数据发送的例子。接收移位寄存器的初始内容是 00。



A. 从机写 98h 到 DSR 并等待主机将数据移出。
B. 主机写 B0h 到 UTXBUF，它立即送到发送移位寄存器，且开始发送。
C. 完成第一个字符，中断标志置位。
D. 从机从它的接收缓存读出 58h（右对齐）。
E. 从机写 54h 到 DSR 并等待主机将数据移出。
F. 主机从它的接收缓存 URXBUF 读出 4ch（右对齐）。
G. 主机写 E8h 到发送缓存 UTXBUF 并开始发送。
H. 完成第二个字符，中断标志置位。
I. 主机收到 2Ah，从机收到 74h（右对齐）。



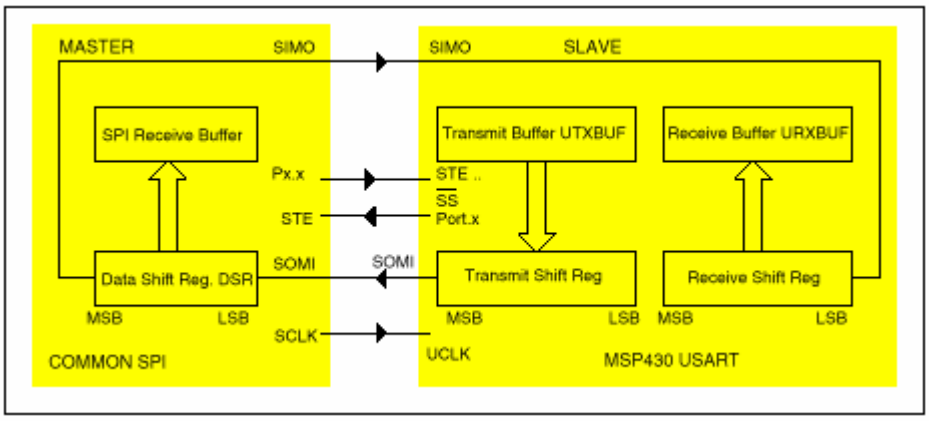


图 13.3：MSP430USART 作为 3 线或 4 线结构的从机

13.1.1 SPI 模式中的主模式，MM=1，SYNC=1

控制寄存器 UCTL 中的主模式位 MM 置位选择主机模式。USART 通过在 UCLK 引脚 UCLK 信号来控制串行通信网。在第一个 UCLK 周期，数据从 SIMO 引脚输出，并在相应的 UCLK 周期的中间，从 SOMI 引脚锁存数据。

一旦移位寄存器为空，已写入发送缓存 UTXBUF 的数据移入移位寄存器，并启动在 SIMO 引脚的数据发送，先发送最高有效位。同时，接收数据移入接收移位寄存器，当移完所选定的位数接收移位寄存器中的数据移入 URXBUF，并使中断标志 URXIFG 置位。最高有效位首先移入接收移位寄存器。以右对齐的方式存入接收缓存 URXBUF。如果这时前一数据未被读取，溢出错位 OE 置位。

注意：USART 同步主模式，接收初始化
为了接收一个字符，主机必须向发送缓存 UTXBUF 写入数据。当发送移位寄存器为空时接收开始，数据移入这一移位寄存器。接收和发送总是在时钟的两个反向沿处一起发生的。

用发送中断标志 UTXIFG 或接收中断标志 URXIFG 可以完成协议的控制。数据从移位寄存器发送给从机后，可立即用 UTXIFG 标志将数据从缓存中移入移位寄存器，开始一次发送。从机接收定时应确保能及时获取数据。URXIFG 标志指示数据移出移入完成的时间。主机可利用 URXIFG 确定从机已准备好接收新的数据。

任一标准的数字端口，包括 STE 在内，用标准数字端口功能可选择一个从机。从机因 STE 信号允许访问 SOMI 数据线，同时允许接收 UCLK 的时钟信号。

4 线 SPI 主机模式，SYNC=1，STC=0，MM=1

由激活的主机用 STE 信号防止与别的主机发生总线冲突。如果相应的 PnSEL 位选择模块功能，STE 引脚成为输入。主机在 STE 信号为高时正常操作。当 STE 置为低时，例如另一设备申请成为主机，这时当前的主机作出如下反应：

- 驱动 SPI 总线的 SIMO 和 UCLK 的引脚置为输入。

- 出错位 FE 和 URCTL 中的中断标志 URXIFG 置位。

总线冲突消除，即 SIMO 和 UCLK 不再驱动总线线路，同时出错标志通知软件，系统完整性被破坏。当 STE 为低电平，SIMO、UCLK 引脚强制变为输入，当 STE 返回高电平，系统将返回到由相应控制位定义状态。

在三线模式中，STE 输入信号与主机无关。

13.1.2 SPI 模式中的从模式，MM=0，SYNC=1

如选择为同步模式并且主机模式位 MM 复位，微控制器进入从机模式。

为接收外部主机提供的串行移位时钟，UCLK 引脚工作在输入状态。数据传输速率由此时钟信号决定，而不是内部的比特率发生器。在开始 UCLK 之前由 UTXBUF 装入到移位寄存器的数据，在主机提供的 UCLK 信号作用下通过 SOMI 引脚发送。同时，在 UCLK 时钟的反向沿 SIMO 引脚上的串行数据移入接收移位寄存器。

接收中断标志 URXIFG 置位，表示数据已被接收并已传送给接收缓存。当新数据写入接收缓存时前一个接收到的数据还未被读取，溢出标志将置位。

4 线 SPI 从机模式，MM=0，SYNC=1，STC=0

在 4 线 SPI 模式中，STE 信号被从机用作发送、接收允许信号。它由主机提供。当 STE 信号为高，禁止接收和发送，当 STE 信号为低，允许接收和发送。无论何时 STE 信号变高，任何已起动的接收操作都将暂停，直到 STE 信号再次变低时继续。STE 信号用于允许一个从机访问数据线路。当 STE 为高时 SOMI 成为输入状态。

13.2 中断与控制功能

USART 外围模块有发送和接收两个主中断源。两个独立的中断矢量，一个对应接收，一个对应发送。

USART 的控制位位于 SFR 地址：

- 接收中断标志　　URXIFG　　初始状态复位（用 PUC/SWRST）
- 接收中断允许　　URXIE　　初始状态复位（用 PUC/SWRST）
- 接收允许　　URXE　　初始状态复位（用 PUC）
- 发送中断标志　　UTXIFG　　初始状态置位（用 PUC/SWRST）
- 发送中断允许　　UTXIE　　初始状态复位（用 PUC/SWRST）
- 发送允许　　UTXE　　初始状态复位（用 PUC）

USART 的发送和接收是并行进行的，并且使用同步主机模式的同一个波特率发生器。在同步从机模式中，UCLK 引脚上的外部时钟用于接收和发送。

13.2.1 USART 接收允许

接收允许位 URXE 允许或禁止接收器从 URXD/SOMI 数据线进行位流的收集。禁止 USART 接收（URXE=0），在完成已开始的接收操作后接收停止，如果无接收操作发生立即停止。在同步模式中，这时 UCLK 不移动数据到接收移位寄存器中。

MSP430 作为主机时的接收

如果 MSP430 的 USART 选择 SPI 主机模式，3 线和 4 线模式的接收操作相同。

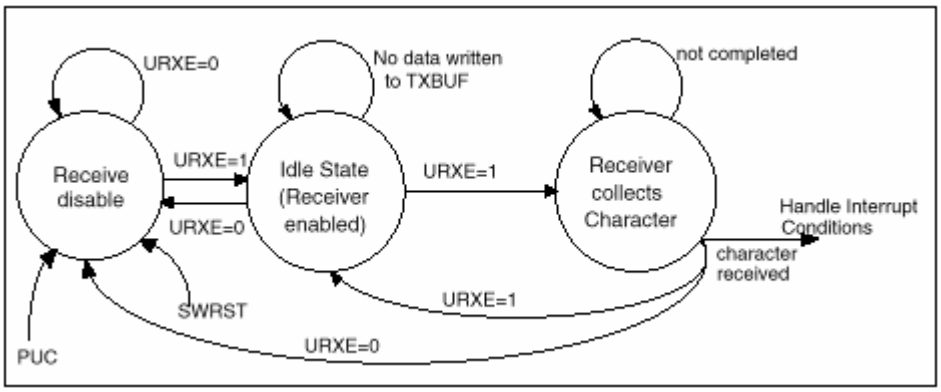


图 13.4：接收允许 URXE 的状态图，MSP430 为主机

MSP430 作为从机时的接收，3 线模式。

当 MSP430 的 USART 选择 SPI 的从机模式时，3 线和 4 线模式的接收操作是不同的。在 3 线模式中，接收操作一旦起动，没有外部 SPI 接收控制信号能使它停止。上电清除 PUC、软件复位 SWRST 或接收允许 URXE 能停止接收操作，并使 USART 复位。

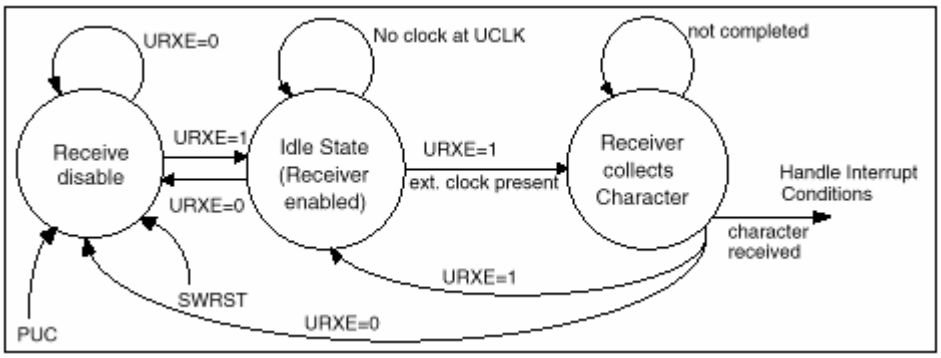


图 13.5：接收允许 URXE 的状态图，MSP430 为从机/3 线模式

注意：URXE 再允许，SPI 模式

由于接收器的完全禁止，接收器的再允许与通信线路的数据流是异步的。要实现与数据流的同步，必须象通常的 3 线 SPI 模式那样采用软件协议来实现。

MSP430 作为从机时的接收，4 线模式

在 4 线模式中，STE 引脚上的外部 SPI 接收控制信号可停止已经起动的接收操作。上电清除 PUC、软件复位 SWRST 或 URXE 能停止接收操作并使操作控制状态机复位。无论何时当 STE 为高，接收操作就暂停。

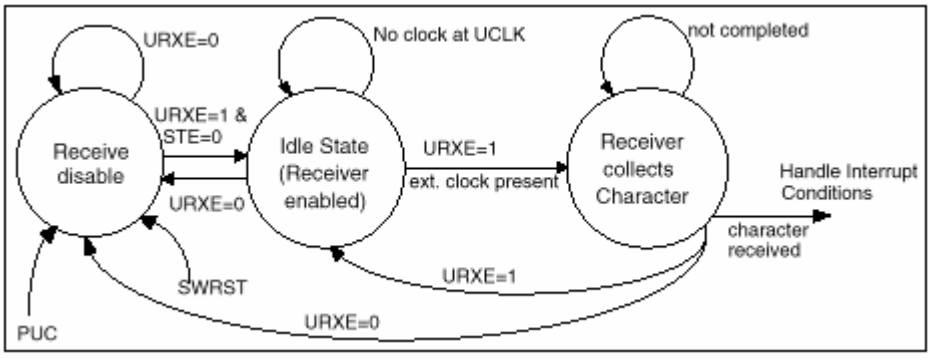


图 13.6：接收允许 URXE 的状态图，MSP430 为从机/4 线模式

13.2.2 USART 发送允许

发送允许位 UTXE 允许或禁止字符移位到串行数据线。它的复位禁止发送器，但是活动的发送并不暂停，直到已写入发送缓存的数据全部发送完毕。继续向发送缓存写入数据不会产生数据发送。如 UTXBUF 就绪，发送请求将一直保持，一旦 UTXE 置位并且发送器为空，立即起动数据发送。STE 的低电平信号将从总线上取消活动的主机（4 线模式）。STE 为低表明有另一主机请求成为活动的主机。

USART 发送允许，MSP430 是主机

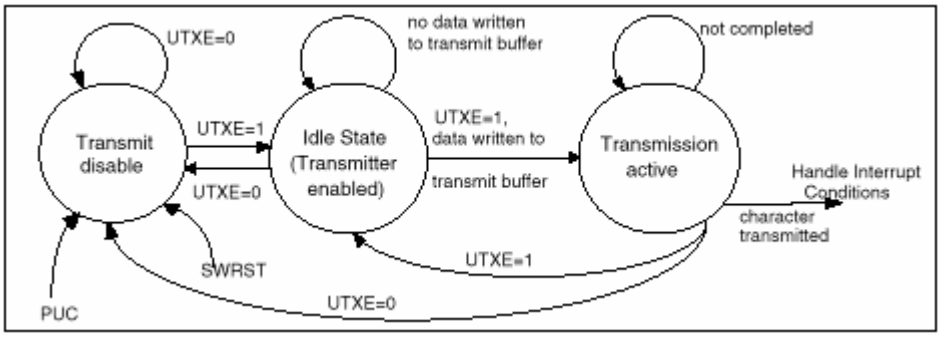


图 13.7：发送允许的状态图，MSP430 为主机

USART 发送允许，MSP430 是从机

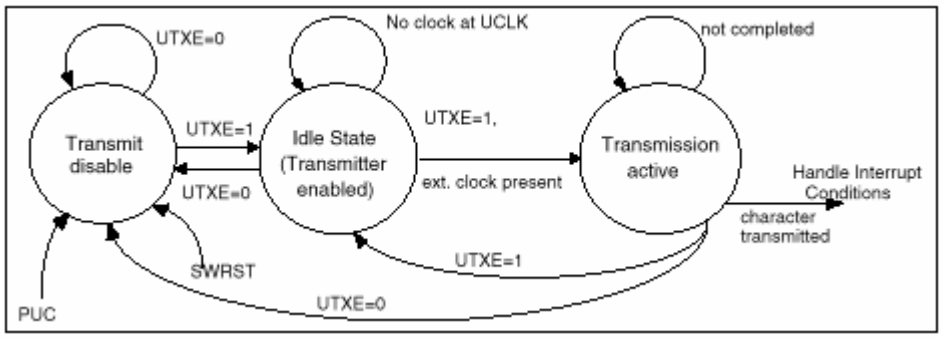


图 13.8：发送允许的状态图，MSP430 为从机

当 UTXE 复位，数据能正常写入发送缓存，但是不能起动发送操作。一旦 UTXE 置位，发送缓存中的数据立即装入发送移位寄存器，字符发送启动。

注意：写入 UTXBUF，SPI 模式
在 UTXBUF 未就绪（UTXIFG=0）且发送允许（UTXE=1）时，不能对它写入。否则移出的字符会是随机的。

13.2.3 USART 接收中断操作

接收中断标志 URXIFG 在每接收完一次字符并装入接收缓存时置位。异步通信时不使用这一方式。

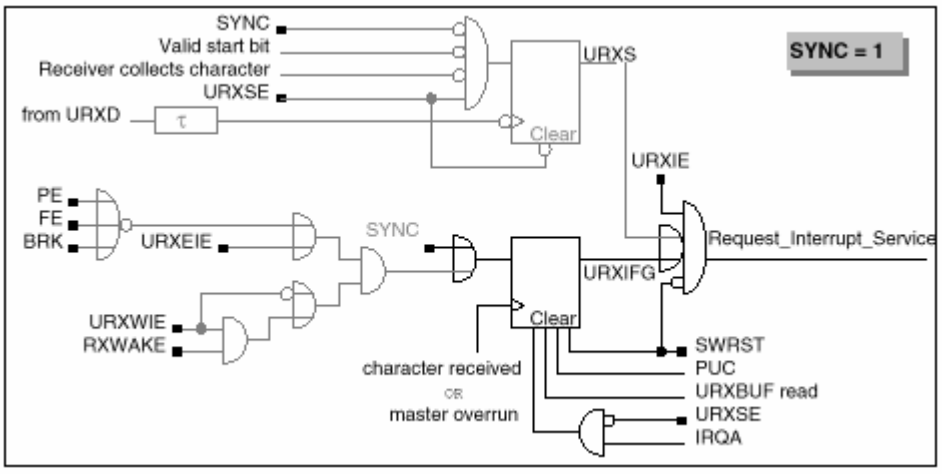


图 13.9：接收中断条件

在系统复位 PUC 和软件复位 SWRST 使 URXIFG 复位。如果执行中断服务或对 URXBUF 读操作使 URXIFG 自动复位。接收中断允许 URXIE 置位时允许中断请求并进行执行挂起中断请求服务。URXIFG 和 URXIE 都因 PUC 和 SWRST 而复位。

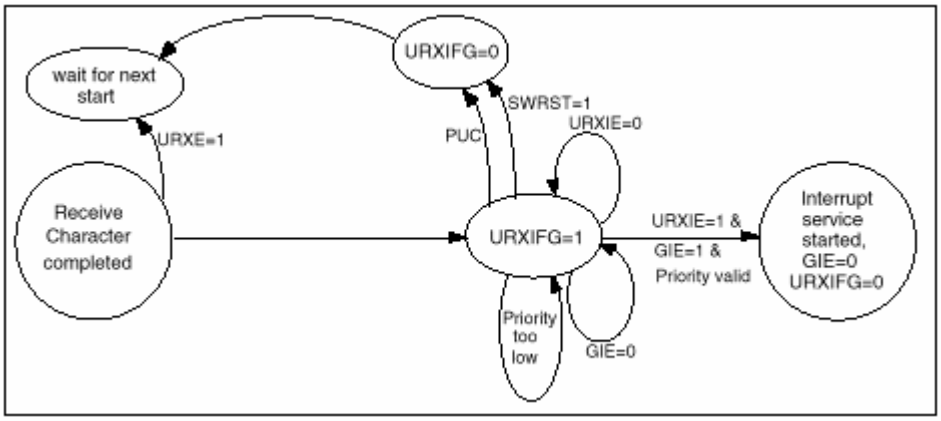


图 13.10：接收中断状态图

13.2.4 USART 发送中断操作

发送器对 UTXIFG 置位，表明发送缓存 UTXBUF 准备好接收下一个字符。如果执行中断请求服务或将字符写入 UTXBUF 中，UTXIFG 将自动复位。如果 UTXIE 和 GIE 中断允许位置位，UTXIFG 置位时导致一次发送中断。UTXIFG 在系统复位 PUC 后或 SWRST 消除后置位。

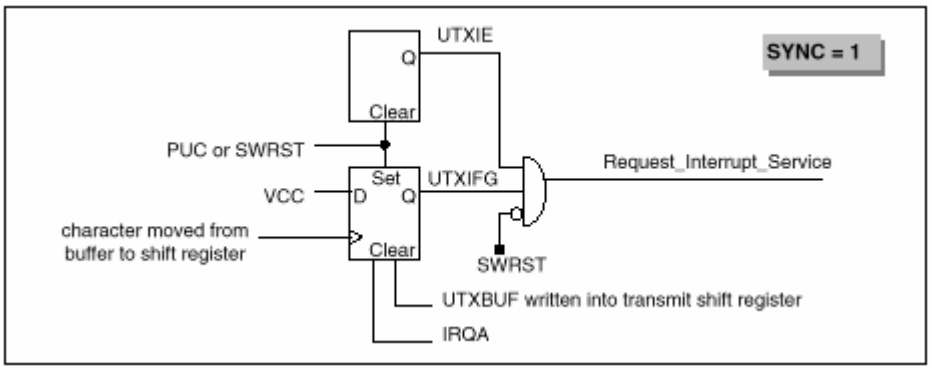


图 13.11：发送中断条件

发送中断允许位 UTXIE 控制 UTXIFG 请求中断的能力，但不能妨碍 UTXIFG 置位。UTXIE 因 PUC 或 SWRST 而复位。系统复位 PUC 或软件复位使 UTXIFG 置位，并使 UTXIE 复位保证了完整的中断控制能力。

13.3 控制与状态寄存器

USART 硬件是字节结构，因此必须以字节指令访问（后缀“B”）。

寄存器	缩写	寄存器类型	地址	初始状态
● USART 控制寄存器	UCTL	读写	070H	见各位说明
● 发送控制寄存器	UTCTL	读写	071H	
● 接收控制寄存器	URCTL	读写	072H	
● 调整控制寄存器	UMCTL	读写	073H	不变
● 波特率寄存器 0	UBR0	读写	074H	不变
● 波特率寄存器 1	UBR1	读写	075H	不变
● 接收缓存	URXBUF	读写	076H	不变
● 发送缓存	UTXBUF	读	077H	不变

除非在功能描述中说明，PUC 后所有位是随机的。

由 PUC 或 SWRST 位执行 USART 复位。PUC 后如 SWRST 保持置位，USART 将维持在复位状态，直到 SWRST 复位使 USART 复位结束。PUC 后禁止 SPI 模式。

由定义 SYNC 位确定 USART 模块工作在异步模式或同步模式。控制寄存器的各位在两种模式中功能上有差异。下面各位及其功能的描述都是在同步模式下，即 SYNC=。异步模式下的功能在 USART 串行接口 UART 模式部分中描述。

13.3.1 USART 控制寄存器

控制寄存器的信息决定了 USART 模块的基本操作。寄存器位选择通信模式和字符位数。所有位根据选择模式的编程都应在因 SWRST 复位使 USART 复位之前进行。

UCTL（070h）

7							0
未用	未用	未用	CHAR	listen	SYNC	MM	SWRST
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

图 13.12：USART 控制寄存器

位 0：如 SWRST 置位，USART 的状态机构和运行标志初始化成复位状态。所有受影响的逻辑保持在复位状态，直至 SWRST 复位。这意味着一次系统复位后，只有对 SWRST 复位，USART 才能重新被允许。

位 1：当 MM 位置位，选择主机模式，当 MM 位复位，选择从机模式。

位 2：USART 外围模块的模式选择。

SYNC 位选择 USART 外围接口模块的功能。USART 的一些控制位在 UART 模式和 SPI 模式中有不同的功能。

SYNC=0：　UART 模式

SYNC=1：　SPI 模式

位 3：Listen 位选择是否将发送数据内部反馈给接收器。

位 4：字符长度

选择字符以 7 或 8 位发送。7 位字符不用 URXBUF 和 UTXBUF 的最高位，这一位填“0”。

CHAR=0：　7 位

CHAR=1：　8 位

位 5：未用

位 6：未用

位 7：未用

13.3.2 发送控制寄存器 UTCTL

控制寄存器 UTCTL 控制与发送有关的 USART 硬件。

UTCTL（071h）

7							0
CKPH	CKPL	SSEL1	SSEL0	未用	未用	STC	TXEPT
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

图 13.13：USART 发送控制寄存器

位 0：发送移位寄存器和 UTXBUF 都为空时，发送器空标志 TXEPT 置位，并在数据写入 UTXBUF 时复位。它在 SWRST 置位。

位 1：从机发送控制位 STC 选择 STE 引脚信号在主机和从机中的使用

STC=0：选择 SPI 的 4 线模式，STE 信号用于使主机避免总线冲突，或用于在从机模式中控制发送和接收的允许。

STC=1：选择 SPI 的 3 线模式，STE 在主机、从机模式中不起作用。

位 2、3：未用

位 4、5：时钟源选择 0、1

只有当选择主机模式时，时钟源选择位定义用于波特率发生器的时钟源。

SSEL1、SSEL0	0	选择外部时钟 UCLK
	1	选择辅助时钟 ACLK
	2、3	选择主系统时钟 MCLK

主机模式（MM=1），不能选 UCLK 上的外部时钟，因为主机将 UCLK 提供给所有从机。

从机模式 SSEL0，SSEL0 是无关紧要的，这时总是用外部 UCLK 信号作为它的时钟。

位 6、7：时钟极性 CKPL 和时钟相位 CKPH，

CKPL 位控制 SPICLK 信号的极性。

CKPL=0：低电平为无效电平，数据在 UCLK 的上升沿输出，输入数据在下降沿锁存。

CKPL=1：高电平为无效电平，数据在 UCLK 的下降沿输出，输入数据在上升沿锁存。

CKPH 位控制 SPICLK 信号的相位。

CKPH=0：正常的 UCLK 时钟

CKPH=1：UCLK 被延迟半个周期。

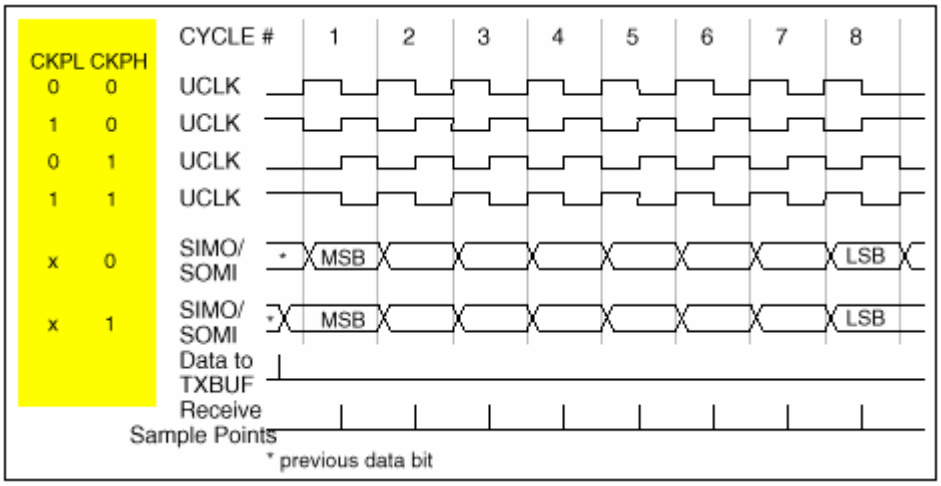


图 13.14：USART 的时钟相位和极性

当工作在 CKPH 置位时，同步模式的 USART 在发送移位寄存器被装入数据有效后和 UCLK 的第一个沿前准备好数据第一位。数据在 UCLK 的第一个沿锁存，在第二个沿发送。

13.3.3 接收控制寄存器 URCTL

URCTL 控制与接收有关的 USART 硬件，并保存出错信息。

URCTL（072h）

7							0
FE	未定义	OE	未定义	未用	未用	未定义	未定义
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

图 13.15：USART 接收控制寄存器

位 0：未定义，由 USART 硬件驱动
位 1：未定义，由 USART 硬件驱动
位 2：未用
位 3：未用
位 4：未定义，由 USART 硬件驱动
位 5：当一个字符写入 URXBUF 时前一字符还未被读出，则溢出标志 OE 置位。这时前一字符被覆盖而丢失。OE 通过 SWRST 位、系统复位、读取 URXBUF 和用指令来复位。
位 6：未定义，由 USART 硬件驱动
位 7：帧错。选择 4 线模式时，因总线冲突使有效主机停止并在 STE 引脚信号出现负跳变使 FE 位置位。FE 位通过 SWRST、系统复位、读 URXBUF 和用指令来复位。

13.3.4 波特率选择和调制控制寄存器

波特率放生器用波特率选择寄存器的 UBR1 和 UBR2 来产生串行数据流的位定时。最小的分频因子为 2。

UBR0（074h）

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

UBR1（075h）

7							0
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
rw	rw	rw	rw	rw	rw	rw	rw

图 13.16：USART 波特率选择寄存器

波特率 = BRCLK /（UBR+(m7+m6+..+m0)/8），　　UBR=[UBR1，UBR0]

可选择的最大发送波特率，主机模式为波特率发生器输入时钟的一半，从机模式取决于在 UCLK 上提供的外部时钟。

在串性同步通信时不需要用调整寄存器。建议将它复位（各位置为 0）。

UMCTL（073h）

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

图 13.17：USART 调整控制寄存器

13.3.5 USART 接收数据缓存 URXBUF

接收缓存 URXBUF 含有此前从接收移位寄存器来的数据。由 SWRST 或 PUC 清除。读取 URXBUF 中数据，将使复位接收出错位和中断标志 URXIFG 复位。

URXBUF（076h）

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

图 13.18：USART 接收缓存

在 7 位字长模式，URXBUF 的最高位总是复位。

13.3.6 USART 发送数据缓存 UTXBUF

发送缓存含有当前需要发送器发送的数据。

UTXBUF（077h）

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

图 13.19：USART 发送缓存

UTXIFG 标志表示 UTXBUF 已准备好接收下一个要发送的字符。

主机模式时，将数据写入 UTXBUF 初始化发送功能。如果发送移位寄存器为空或即将为空，数据的发送立即开始。

当选择每字符 7 位，数据移入发送缓存必须左对齐，因为最高有效位首先发送。

14. 液晶显示驱动

目录 页号

14.1 LCD 驱动基本原理

液晶显示是通过环境光来显示信息的，其本身并不主动发光。因此功耗很低。只要求液晶周围有足够的光强。

液晶必须由交流电压驱动。直流驱动会损坏液晶。交流驱动是其功耗的主要因素。从驱动电路看液晶等效为一个电容。它的一个电极叫背极或公共极，由 COMn 信号驱动，另一电极叫段极，由 SEGn 信号驱动。交流驱动的频率很低，在 30Hz 至 1000Hz 范围内。LCD 生产厂家的数据手册给出了确切的频率范围。

人们已经开发了多种驱动液晶显示的方法。当要显示的段数、所需的引脚数和显示的对比度、温度等要求不同时，采用的方法也不同。

多路寻址方法能有效地减少所需的引脚。

MSP430 系列的 LCD 模块支持的驱动方法有 4 种：

- 静态驱动
- 2MUX 或 1/2 占空比，1/2 偏压
- 3MUX 或 1/3 占空比，1/3 偏压
- 4MUX 或 1/4 占空比，1/3 偏压

静态方法只需一个引脚作公共极（COM0），每个段都需要一个引脚：

引脚数 = 1 + 段数

2MUX 方法需要 2 个引脚作公共极（COM0、COM1），每 2 段需要一个引脚：

引脚数 = 2 + 段数/2

3MUX 方法需要 3 个引脚作公共极（COM0、COM1、COM2），每 3 段需要一个引脚：

引脚数 = 3 + 段数/3

4MUX 方法需要 4 个引脚作公共极（COM0、COM1、COM2、COM3），每 4 段需要一个引脚：

引脚数 = 4 + 段数/4

增加多路转接数就能减少引脚数。如显示 80 段，各方法所需的引脚数如下：

静态驱动： 引脚数 = 1 + 80 = 81

2MUX： 引脚数 = 2 + 80/2 = 42

3MUX： 引脚数 = 3 + 80/3 = 30

4MUX： 引脚数 = 4 + 80/4 = 24

静态驱动方法

静态驱动方法，每根段线驱动 1 段。

下面的例子说明 LCD 显示一个“5”时，各引脚的连接和输出波形。

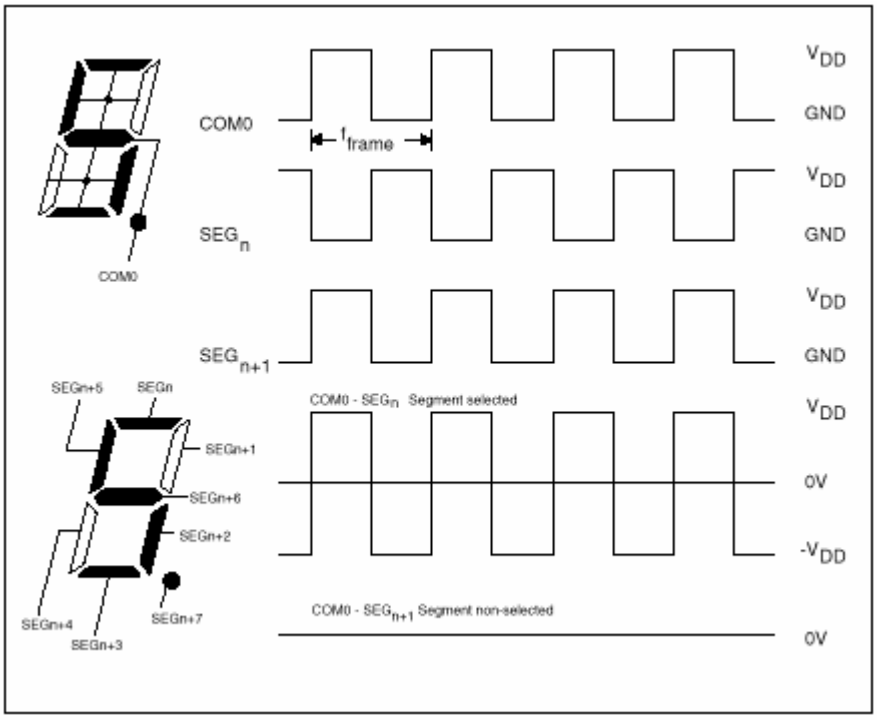


图 14.1：静态驱动波形实例

2MUX，1/2 偏压

2MUX 驱动方法，每根段线驱动 2 段。

下面的例子说明 LCD 显示一位“5”时，各引脚的驱动电压波形。

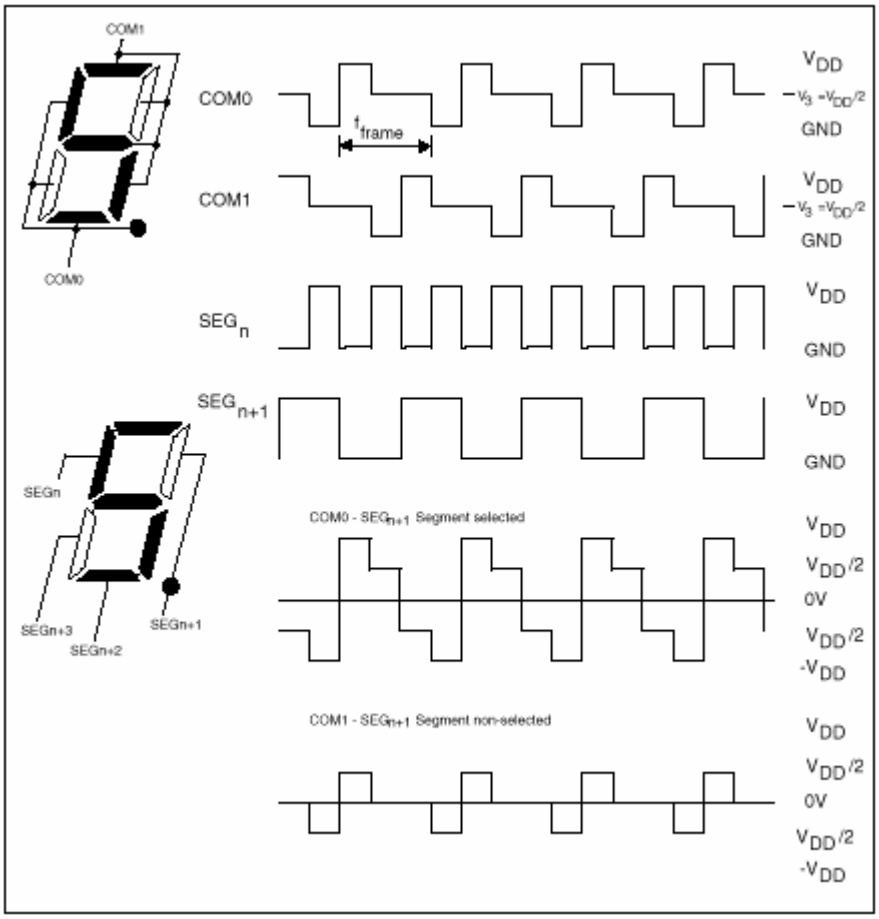


图 14.2：2MUX 驱动波形实例

3MUX，1/3 偏压

3MUX 驱动方法，每根段线驱动 3 段。

下面的例子说明 LCD 显示一位“5”时，各引脚的驱动电压波形。

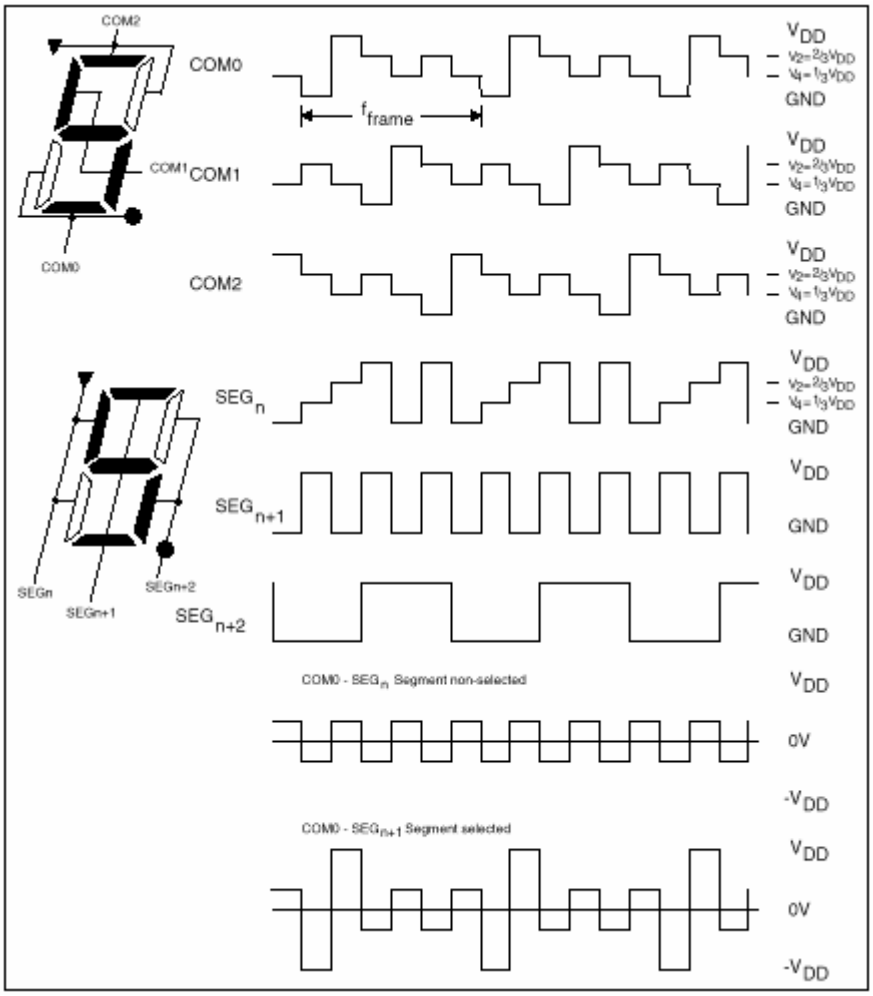


图 14.3：3MUX 驱动波形实例

4MUX，1/3 偏压

4MUX 驱动方法，每根段线驱动 4 段。

下面的例子说明 LCD 显示一位“5”时，各引脚的驱动电压波形。

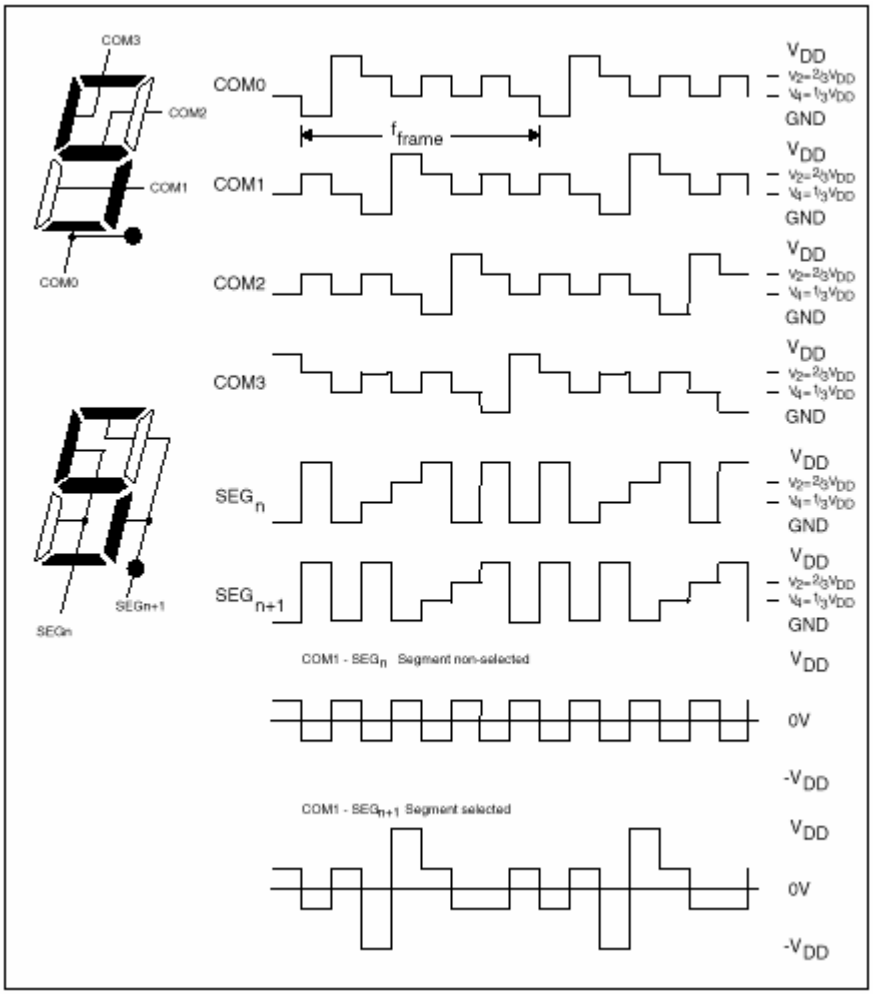


图 14.4：4MUX 驱动波形实例

14.2 LCD 控制器/驱动器

LCD 控制器/驱动器外围模块按显示内存的数据产生 SEG 信号和 COM 信号。它含有驱动外部直接连接的 LCD 的全部功能模块。LCD 外围模块的主要的功能模块有：

- 包含段信息的数据内存
- 定时发生器
- 模块总线接口
- LCD 模块：使用外加的模拟电压
- LCD+ 模块：使用内部产生的模拟电压，仅对此模块

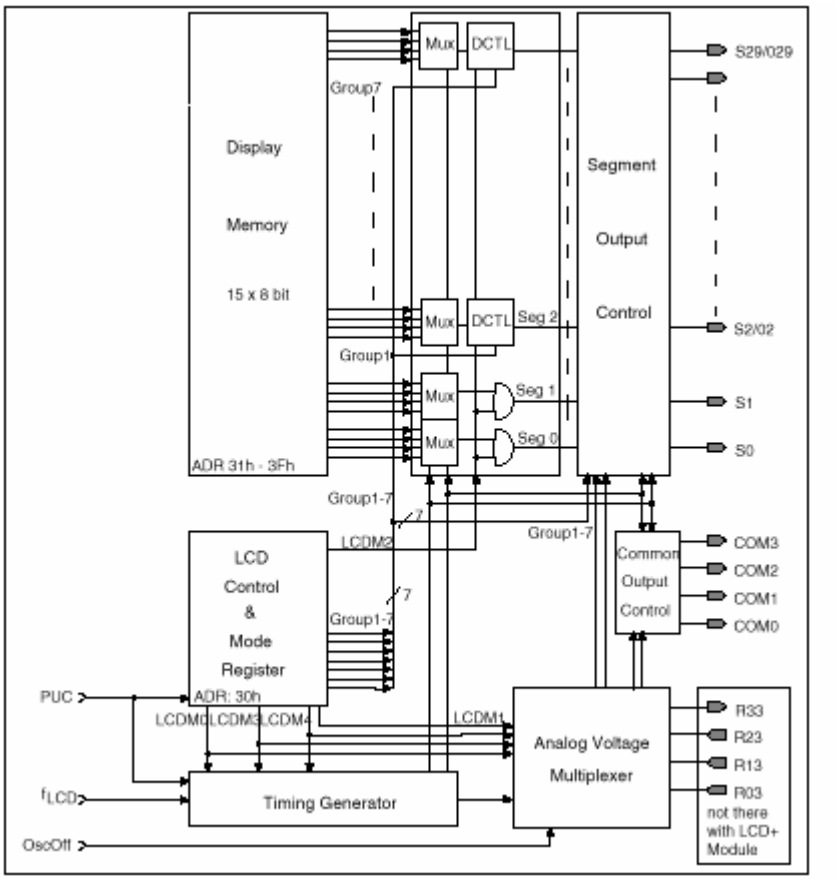


图 14.5：LCD 控制器/驱动器功能框图

LCD 模块和 LCD+ 模块的区别

	LCD 模式	LCD+模式
• 模拟电压	由外部产生 选项 • 2 输入端 R23、R13 V1=VCC、V5=VSS • 3 输入端 R23、R13、R03 V1=VCC • 4 输入 R133、R23、R13、R03	由内部产生
• 控制位 LCDM1	未用	选择权电阻网络阻抗
• 控制位 LCDM0	• 定时发生器停止	• 定时发生器停止 • 权电阻网络电流为 0

14.2.1 LCD 控制器/驱动器功能

LCD 驱动器的功能为：

- 自动从显示内存读取数据并产生 SEG、COM 信号

- 可选择 4 种不同的驱动模式
  静态模式
  2MUX，1/2 偏压
  3MUX，1/3 偏压
  4MUX，1/3 偏压
  BT(basic timer)中，有两位可用来选择 4 种不同的扫描频率。
- 段线输出端口可切换成普通输出端口
- 没有存储段信息的显示内存可用作普通内存
- 用辅助时钟（ACLK），经过 BT 来操作
- 仅针对 LCD+ 模块：
  用电阻网络为 LCD 驱动提供各种模拟电压。
  控制寄存器 LCDCTL 中的一位控制电阻网络和 V1 的通断。

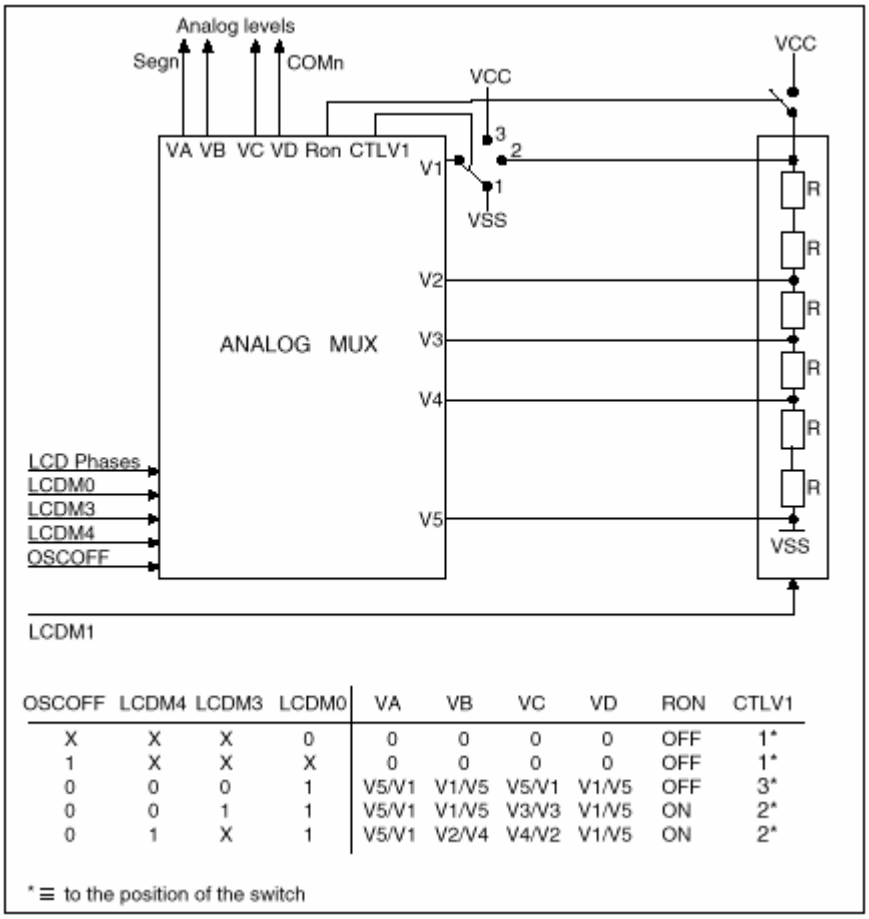
  LCD 线的帧频 f
- 静态方法　　$f = 1/2 * f_{LCD}$
- 2MUX　　$f = 1/4 * f_{LCD}$
- 3MUX　　$f = 1/6 * f_{LCD}$
- 4MUX　　$f = 1/8 * f_{LCD}$

LCD+ 模块：

由内部提供模拟电压。

当状态寄存器的 OSCOff 位置位，电阻网络电源被切断，而与 LCDM0 无关。

由于静态模式只用到 V1 和 V5，模拟电压发生器关闭。这样降低了供电电流消耗。



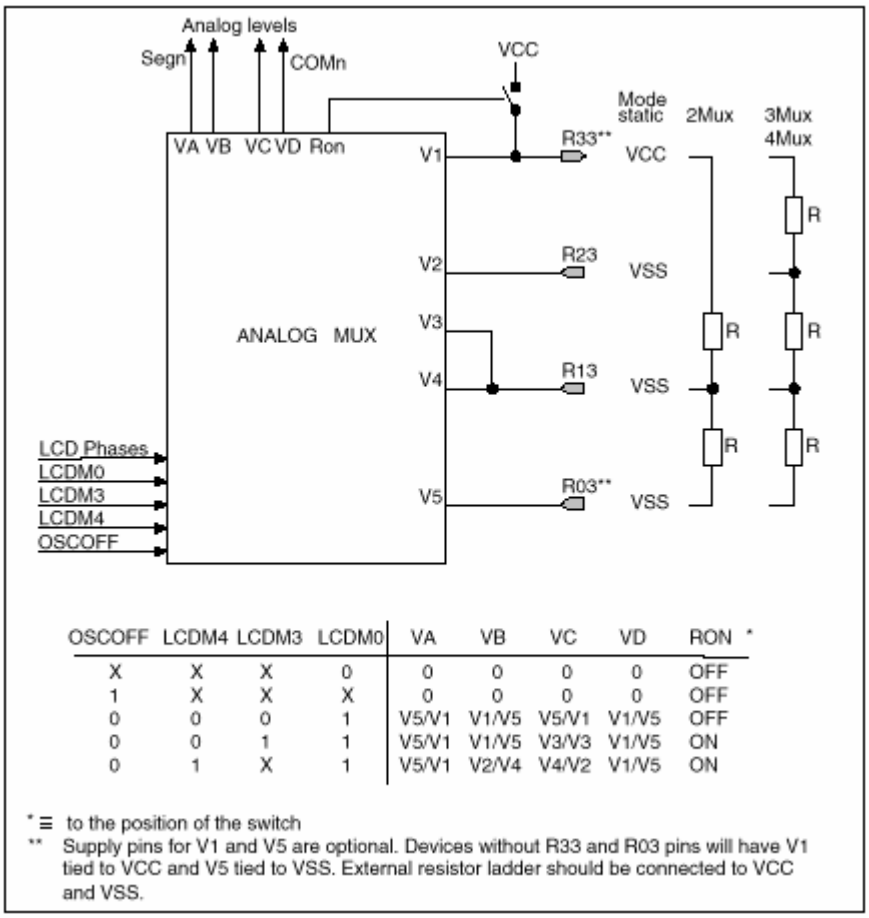
OSCOFF	LCDM4	LCDM3	LCDM0	VA	VB	VC	VD	RON	CTLV1
X	X	X	0	0	0	0	0	OFF	1*
1	X	X	X	0	0	0	0	OFF	1*
0	0	0	1	V5/V1	V1/V5	V5/V1	V1/V5	OFF	3*
0	0	1	1	V5/V1	V1/V5	V3/V3	V1/V5	ON	2*
0	1	X	1	V5/V1	V2/V4	V4/V2	V1/V5	ON	2*

* ≡ to the position of the switch

图 14.6：LCD+ 模块的内部模拟电压发生

LCD 模块

由外部提供模拟电压，加在 R33**、R23、R13、R03** 引脚上。



OSCOFF	LCDM4	LCDM3	LCDM0	VA	VB	VC	VD	RON *
X	X	X	0	0	0	0	0	OFF
1	X	X	X	0	0	0	0	OFF
0	0	0	1	V5/V1	V1/V5	V5/V1	V1/V5	OFF
0	0	1	1	V5/V1	V1/V5	V3/V3	V1/V5	ON
0	1	X	1	V5/V1	V2/V4	V4/V2	V1/V5	ON

* ≡ to the position of the switch
** Supply pins for V1 and V5 are optional. Devices without R33 and R03 pins will have V1 tied to VCC and V5 tied to VSS. External resistor ladder should be connected to VCC and VSS.

图 14.7：LCD 模块的外部模拟电压

注：**R33 和 R03 引脚的电压可选。请阅器件数据手册

14.2.2 LCD 控制和模式寄存器

LCD 控制和模式寄存器定义了不同的操作条件。LCD 模块是字节结构，必须用字节指令访问（后缀 B）

LCDCTL（030h）

7							0
LCDM7	LCDM6	LCDM5	LCDM4	LCDM3	LCDM2	LCDM1	LCDM0
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

LCDM0：LCDM0=0： 定时发生器关闭。
COM 和 SEG 线为低。
选作输出端口线的引脚不受影响。

LCD+ 模块：切断电阻网络的电源。

LCDM0=1：COM 线、SEG 线按显示内存的数据输出信号。

选作输出端口线的引脚不受影响。

LCD+ 模块：对电阻网络供电，只适用于 2MUX、3MUX、4MUX。

LCDM1：选择模拟电压发生器的内部电阻来选择 LCD 驱动电压幅度。只对 LCD+ 模块有效。

LCDM1=0：选择模拟发生器的高阻抗。

LCDM1=1：选择模拟发生器的低阻抗。

LCDM2、3、4：选择显示模式，并可将段输出切换到非选择电平。

LCDM4	LCDM3	LCDM2	显示模式	LCD+ 偏压	LCD 偏压
X	X	0	不影响，显示关闭，所有段信号是非选择电平，端口输出保持稳定		
0	0	1	静态	1/1	R33*，R03*
0	1	1	2MUX	1/2	R33*，R13，R03*
1	0	1	3MUX	1/3	R33*，R23，R13，R03*
1	1	1	4MUX	1/3	R33*，R23，R13，R03*

*：可选引脚

LCDM2 禁止（0）或允许（1）段输出。这是通过和每一段信息的“与”来实现的，它位于显存输出与段输出之间的并串转换模块中。保存在显存内的段信息不变。这个功能的主要目的是支持动态显示。

LCDM5、6、7：选择含有段信息或端口信息的一组输出。由显存的各位状态来决定段线选为端口输出功能，同时不再作为 LCD 段线的一部分。

LCDM7	LCDM6	LCDM5	0 组	1 组	2 组	3 组	4 组	5 组	6 组	7 组
0	0	0	S0-S1	02-05	06-09	010-013	014-017	018-021	022-025	026-029
0	0	1	S0-S1	S2-S5	06-09	010-013	014-017	018-021	022-025	026-029
0	1	0	S0-S1	S2-S5	S6-S9	010-013	014-017	018-021	022-025	026-029
0	1	1	S0-S1	S2-S5	S6-S9	S10-S13	014-017	018-021	022-025	026-029
1	0	0	S0-S1	S2-S5	S6-S9	S10-S13	S14-S17	018-021	022-025	026-029
1	0	1	S0-S1	S2-S5	S6-S9	S10-S13	S14-S17	S18-S21	022-025	026-029
1	1	0	S0-S1	S2-S5	S6-S9	S10-S13	S14-S17	S18-S21	S22-S25	026-029
1	1	1	S0-S1	S2-S5	S6-S9	S10-S13	S14-S17	S18-S21	S22-S25	S26-S29

注意：LCD 控制位

LCDM5、LCDM6、LCDM7 在发生 PUC 时复位。

段功能：　S××信号是显示驱动信号，按 COM 线的帧定时带有调制电压。

端口功能：　作为普通端口时信号是静态的，它按显存各位输出数字电平。0 至 3 位针对奇数 SEG 线（n=3,5,...），4 至 7 位针对偶数 SEG 线（n=2,4,...）。

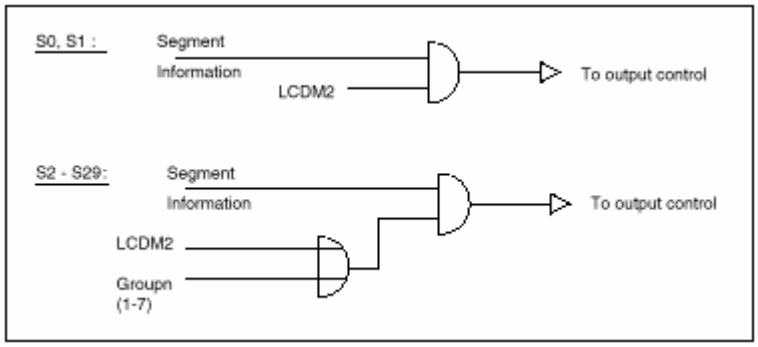


图 14.8：信息控制

14.2.3 LCD 显示内存

LCD 显示内存存储在所有运行及省电模式中待显示的信息。显存的各位与 LCD 显示段直接相关。LCD 显示图形由软件解码，从 BCD 或二进制码表示转换成各组 SEG/COM 显示信息。内存的位信息与一条 SEG 线和一条 COM 线对应。即如果置位，对应的段显示或相反。

一根段线控制 1-4 段的显示与否，取决于多路切换方式：

静态驱动 → 每条段线 1 段

2MUX 驱动 → 每条段线 2 段

3MUX 驱动 → 每条段线 3 段

4MUX 驱动 → 每条段线 4 段

由 LCD 控制器/驱动器的定时发生器驱动并串转换，将 LCD 显存的并行信息转换成段线所需要的串行信息。LCD 显存各位与 COM 线直接在硬件上相连。

静态驱动 → COM0：位 0 至 Sn，位 4 至 Sn+1

2MUX 驱动 → COM0：位 0 至 Sn，位 4 至 Sn+1，COM1：位 1 至 Sn，位 5 至 Sn+1

3MUX 驱动 → COM0：位 0 至 Sn，位 4 至 Sn+1，COM1：位 1 至 Sn，位 5 至 Sn+1
COM2：位 2 至 Sn，位 6 至 Sn+1

4MUX 驱动 → COM0：位 0 至 Sn，位 4 至 Sn+1，COM1：位 1 至 Sn，位 5 至 Sn+1
COM2：位 2 至 Sn，位 6 至 Sn+1，COM3：位 3 至 Sn，位 7 至 Sn+1

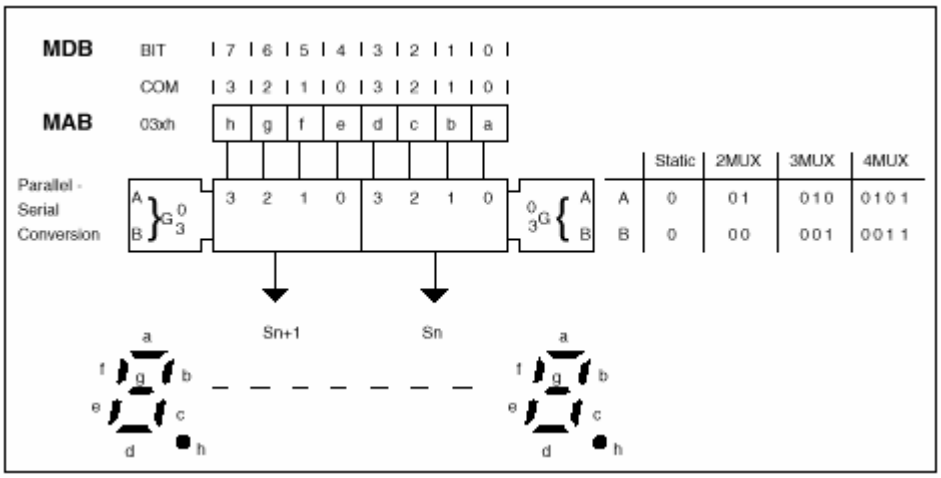


图 14.9：显存各位与段线关系

静态驱动方法的显存

静态驱动方法只使用一条 COM0 线。即 COM0。位 0 和位 4 存储段信息，其余各位可用作普通内存。

可显示的段数最多为 30。

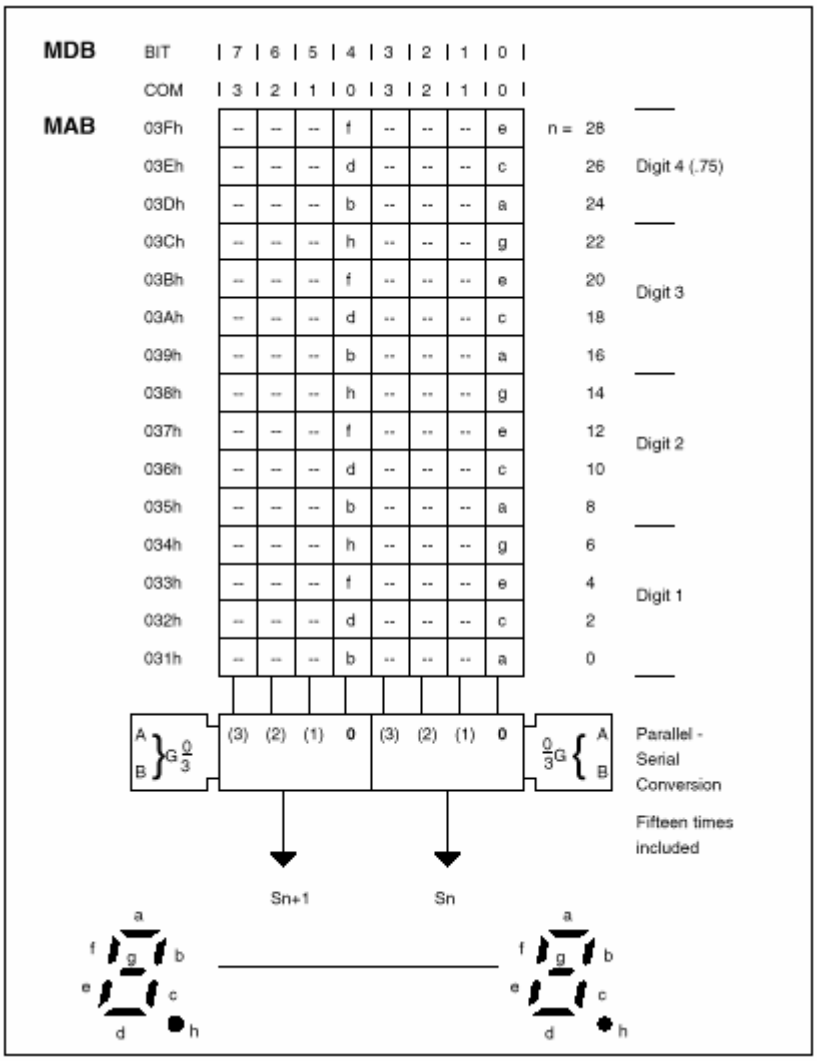


图 14.10：静态驱动时显存的使用

2MUX，1/2 偏压驱动方法的显存

2MUX 方法使用 2 条 COM 线，COM0、COM1。位 0、位 1、位 4 和位 5 存储段信息，其余各位可用作普通内存。

可显示的段数最多为 60。

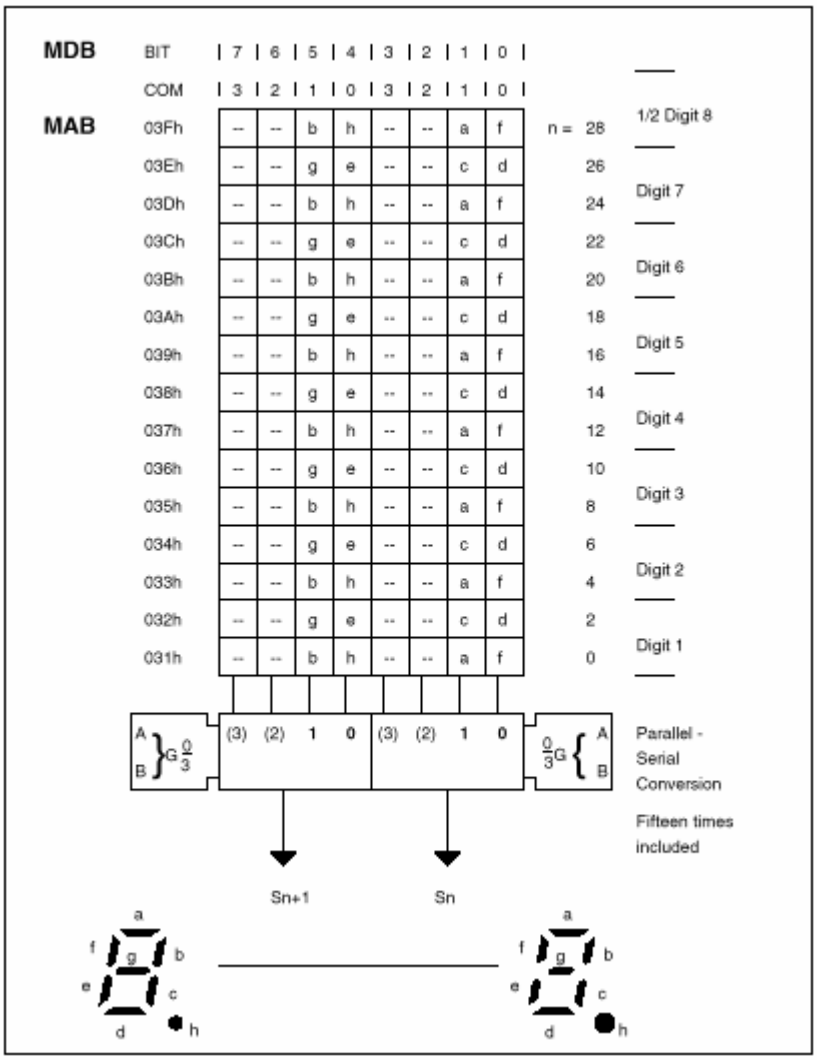


图 14.11：2MUX 方法驱动时显存的使用

3MUX，1/3 偏压驱动方法的显存

3MUX 方法使用 3 条 COM 线，COM0、COM1、COM2。位 0、位 1、位 2、位 4、位 5 和位 6 存储段信息，其余各位可用作普通内存。

可显示的段数最多为 90。

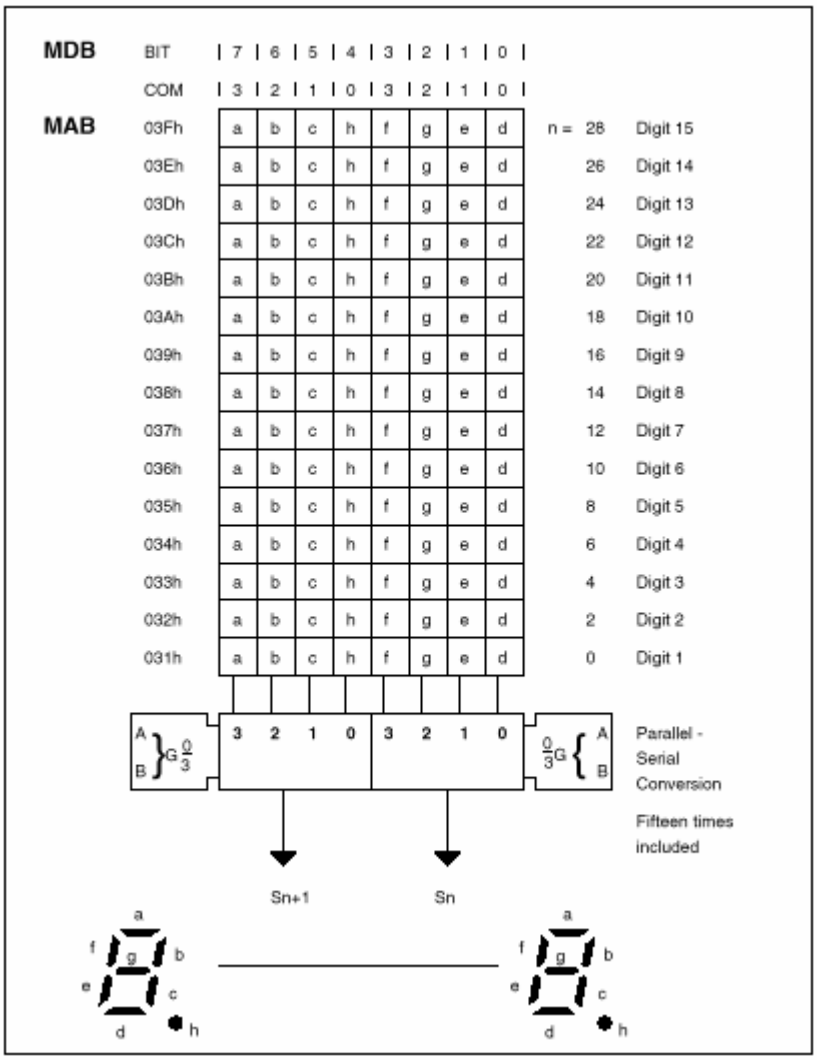


图 14.12：3MUX 方法驱动时显存的使用

4MUX，1/3 偏压驱动方法的显存

4MUX 方法使用 4 条 COM 线，COM0、COM1、COM2 和 COM3。位 0、位 1、位 2、位 3、位 4、位 5、位 6 和位 7 存储段信息。

可显示的段数最多为 120。

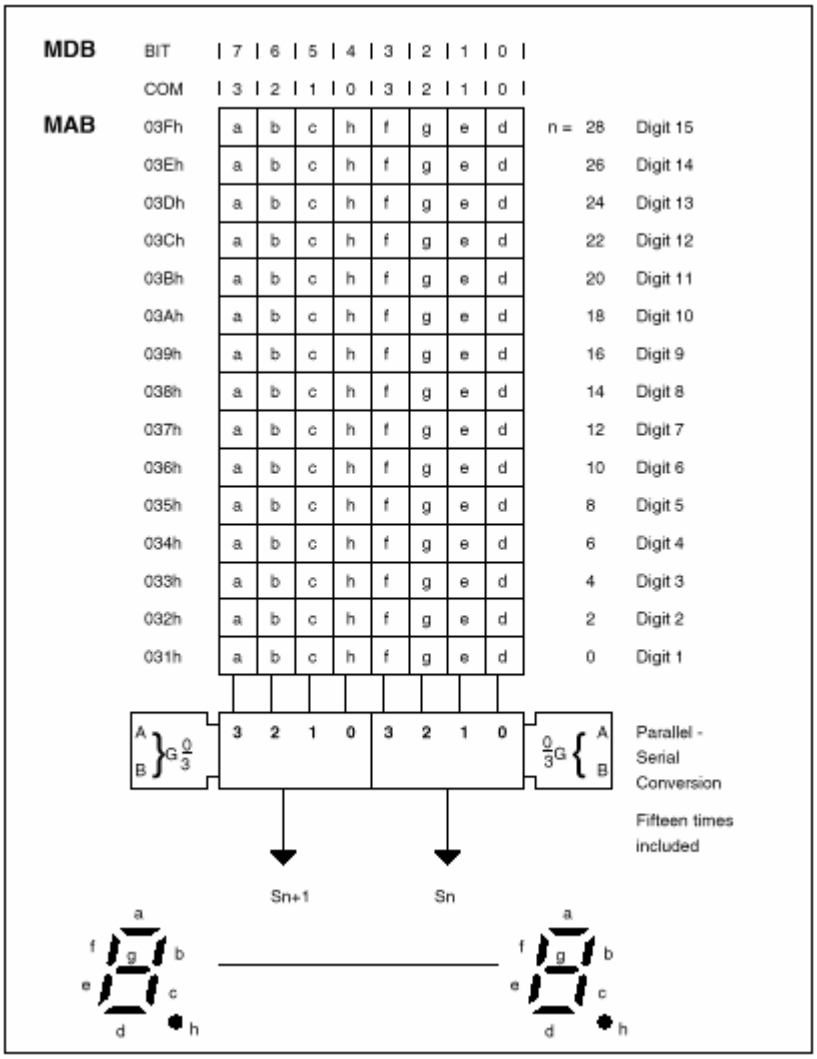


图 14.13：4MUX 方法驱动时显存的使用

14.2.4 LCD 操作软件例程

这一节中提供了 LCD 显示数字的例程。一个数字由标准用法的 7 段组成。

应用 4MUX，1/3 偏压的 LCD 例程

```
        .sect   "lcd4mux",0f000h
; The 4MUX rate is the most easy-to-handle display rate. All eight segments of
; a digit are located in one display memory byte
;
a       .EQU    080h
b       .EQU    040h
c       .EQU    020h
d       .EQU    001h
e       .EQU    002h
f       .EQU    008h
g       .EQU    004h
h       .EQU    010h
;
; The LSDigit of register Rx (000m) should be displayed.
; The Table represents the 'on'-segments according to the content of Rx.
;
        ...........
LCD1    .EQU    00031h          ; Address of LC Display Memory
        ...........
        ...........
LCD15   .EQU    0003Fh
        ...........
;       ...........
        ...........
        MOV.B   Table(Rx),&LCDn ; n = 1 ..... 15
                                ; all eight segments are written to the
                                ; display memory
        ...........
        ...........
;
Table   .BYTE   a+b+c+d+e+f     ; displays "0"
        .BYTE   b+c             ; displays "1"
        ...........
        ...........
        .BYTE   b+c+d+e+g       ; displays "d"
        .BYTE   a+d+e+f+g       ; displays "E"
        .BYTE   a+e+f+g         ; displays "F"
```

应用 3MUX，1/3 偏压的 LCD 例程

```
        .sect   "lcd3mux",0f000h
; The 3MUX rate supports nine segments instead of eight segments for each digit.
; The nine segments of a digit are located in 1 ½ display memory bytes.
a       .EQU    0040h
b       .EQU    0400h
c       .EQU    0200h
d       .EQU    0010h
e       .EQU    0001h
f       .EQU    0002h
g       .EQU    0020h
h       .EQU    0100h
Y       .EQU    0004h
; The LSDigit of register Rx (000m) should be displayed.
; The Table represents the 'on'-seg according to the LSDigit of register of Rx.
; The register Ry is used for temporary memory
;
LCD1    .EQU    00031h
        ...........
LCD15   .EQU    0003Fh
ODDDIG  RLA     Rx
        MOV     Table(Rx),Ry    ; Load segment information to temporary mem.
                                ; (Ry) = 0000 0bch 0agd 0Yfe
        MOV.B   Ry,&LCD n       ; write 'a, b, c, d, e, f' of Digit n (LowByte)
        SWPB    Ry              ; (Ry) = 0agd 0Yfe 0000 0bch
        BIC.B   #07h,&LCDn+1    ; write 'b, c, h' of Digit n (HighByte)
        BIS.B   Ry,&LCDn+1
        .....
EVNDIG  RLA     Rx
        MOV     Table(Rx),Ry    ; Load segment information to temporary mem.
                                ; (Ry) = 0000 0bch 0agd 0Yfe
        RLA     Ry              ; (Ry) = 0000 bch0 agd0 Yfe0
        RLA     Ry              ; (Ry) = 000b ch0a gd0Y fe00
        RLA     Ry              ; (Ry) = 00bc h0ag d0Yf e000
        RLA     Ry              ; (Ry) = 0bch 0agd 0Yfe 0000
        BIC.B   #070h,&LCDn+1
        BIS.B   Ry,&LCDn+1      ; write 'Y, f, e' of Digit n+1 (LowByte)
        SWPB    Ry              ; (Ry) = 0Yfe 0000 0bch 0agd
        MOV.B   Ry,&LCDn+2      ; write 'b, c, h, a, g, d' of Digit n+1 (HighByte)
        ...........
Table   .WORD   a+b+c+d+e+f     ; displays "0"
        .WORD   b+c             ; displays "1"
        ...........
        .WORD   a+e+f+g         ; displays "F"
```

应用 2MUX，1/2 偏压的 LCD 例程

```
        .sect   "lcd2mux",0f000h
; All eight segments of a digit are located in two display memory bytes with the
; 2MUX display rate
a       .EQU    002h
b       .EQU    020h
c       .EQU    008h
d       .EQU    004h
e       .EQU    040h
f       .EQU    001h
g       .EQU    080h
h       .EQU    010h
; The register content of Rx (000m) should be displayed.
; The Table represents the 'on'-segments according to the content of Rx.
        ...........
LCD1    .EQU    00031h
        ...........
        ...........
LCD15   .EQU    0003Fh
        ...........
        ...........
        MOV.B   Table(Rx),Ry    ; Load segment information to temporary mem.
        MOV.B   Ry,&LCD n       ; (Ry) = 0000 0000 gebh cdaf
                                ; Note:
                                ; All bits of an LCD memory byte are written
        RRA     Ry              ; (Ry) = 0000 0000 0geb hcda
        RRA     Ry              ; (Ry) = 0000 0000 00ge bhcd
        MOV.B   Ry,&LCD n+1     ; Note:
                                ; All bits of an LCD memory byte are written
        ...........
        ...........
;
Table   .BYTE   a+b+c+d+e+f     ; displays "0"
        ...........
        .BYTE   a+b+c+d+e+f+g+h ; displays "8"
        ...........
        ...........
        .BYTE
        ...........
;
```

应用静态驱动的 LCD 例程

```
        .sect   "lcd1mux",0f000h
; All eight segments of a digit are located in four display memory bytes with the
; static display method.
;
a       .EQU    001h
b       .EQU    010h
c       .EQU    002h
d       .EQU    020h
e       .EQU    004h
f       .EQU    040h
g       .EQU    008h
h       .EQU    080h
; The register content of Rx should be displayed.
: The Table represents the 'on'-segments according to the content of Rx.
;
        ...........
LCD1    .EQU    00031h
        ...........
LCD15   .EQU    0003Fh
        ...........
        ...........
        MOV.B   Table(Rx),Ry    ; Load segment information to temporary mem.
                                ; (Ry) = 0000 0000 hfdb geca
        MOV.B   Ry,&LCDn        ; Note:
                                ; All bits of an LCD memory byte are written
        RRA     Ry              ; (Ry) = 0000 0000 0hfd bgec
        MOV.B   Ry,&LCDn+1      ; Note:
                                ; All bits of an LCD memory byte are written
        RRA     Ry              ; (Ry) = 0000 0000 00hf dbge
        MOV.B   Ry,&LCDn+2      ; Note:
                                ; All bits of an LCD memory byte are written
        RRA     Ry              ; (Ry) = 0000 0000 000h fdbg
        MOV.B   Ry,&LCDn+3      ; Note:
                                ; All bits of an LCD memory byte are written
        ...........
Table   .BYTE   a+b+c+d+e+f     ; displays "0"
        .BYTE   b+c             ; displays "1"
        ...........
        .BYTE
        ...........
```

14.3 LCD 端口功能

LCD 的 COM 线和 SEG 线数量极大，封装后占用的引脚也多，这会限制集成度。为了充分利用段线，LCDM5 至 LCDM7 可选择 4 根段线一组的功能，是作为段线还是作为多用途的普通输出端口。显存的位信息定义了信号线的逻辑状态。输出信号为数字，即或者接地，或者接供电电压 VCC。

用 Sxx 表示段线，Oxx 表示普通端口线。同一引脚的 xx 号一样，S 或 O 表示功能状态。

LCDM7	LCDM6	LCDM5	0 组	1 组	2 组	3 组	4 组	5 组	6 组	7 组
0	0	0	S0-S1	02-05	06-09	010-013	014-017	018-021	022-025	026-029
0	0	1	S0-S1	S2-S5	06-09	010-013	014-017	018-021	022-025	026-029
0	1	0	S0-S1	S2-S5	S6-S9	010-013	014-017	018-021	022-025	026-029
0	1	1	S0-S1	S2-S5	S6-S9	S10-S13	014-017	018-021	022-025	026-029
1	0	0	S0-S1	S2-S5	S6-S9	S10-S13	S14-S17	018-021	022-025	026-029
1	0	1	S0-S1	S2-S5	S6-S9	S10-S13	S14-S17	S18-S21	022-025	026-029
1	1	0	S0-S1	S2-S5	S6-S9	S10-S13	S14-S17	S18-S21	S22-S25	026-029
1	1	1	S0-S1	S2-S5	S6-S9	S10-S13	S14-S17	S18-S21	S22-S25	S26-S29

图 14.14：段线和端口输出线的分组

注意：控制位
控制位 LCDM5...LCDM7 在 PUC 发生时复位。

段线信号 Sxx 是 LCD 显示驱动信号的一部分，按 COMn 线的帧频输出调制电压。

作为输出端口的输出信号 Oxx 是静态信号。它们按显存各位状态提供数字电平。位 0 至位 3 对应于奇数段线（n=3,5,...），位 4 至位 7* 对应于偶数段线（n=2,4,...）。

* 表示是否使用该位取决于 LCD 驱动方法。

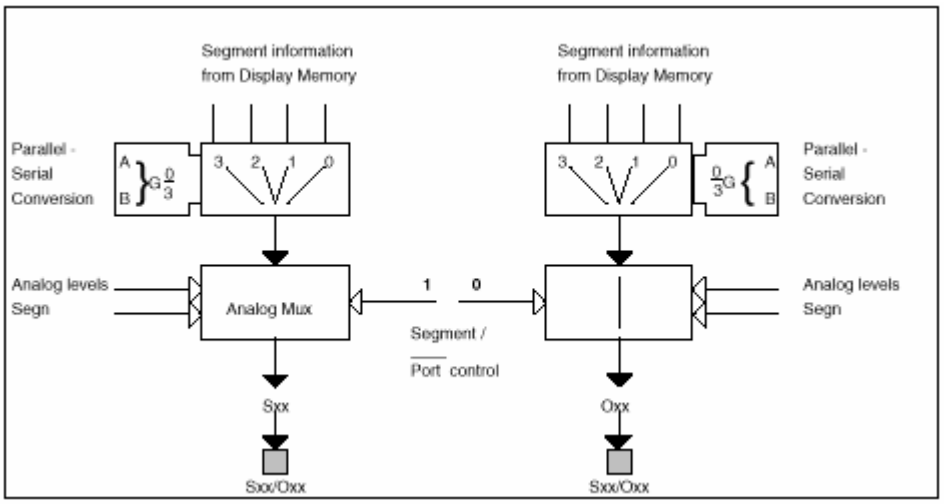


图 14.15　段线或端口输出线

输出信号 0xx 的逻辑状态取决于显存各位的状态。根据奇数或偶数线的选择，可能对应于位 0 至位 3 或位 4 至位 7。

- xx=2,4,...28：0xx 由位 0 至位 3 定义
- xx=3,5,...29：0xx 由位 4 至位 7 定义

14.4 LCD 与端口模式混合应用实例

下面的例子使用了混合模式：用 4MUX 方法驱动 LCD 显示 13 位数字，同时使用一个端口组，输出 4 位数字信号。

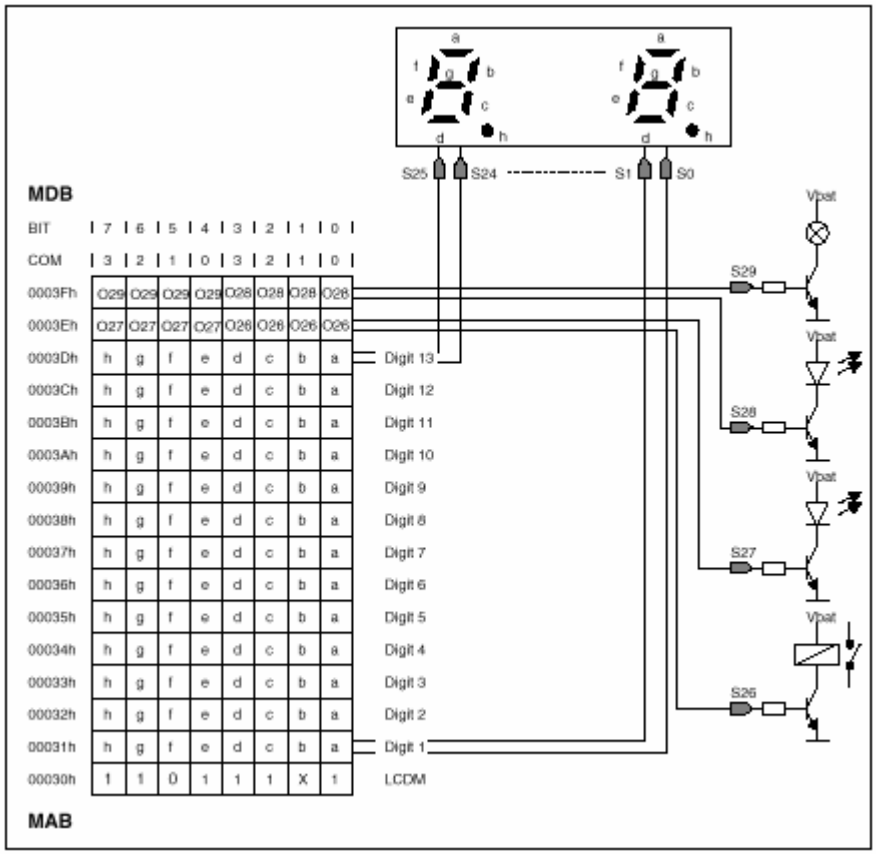


图 14.16：应用实例

注意：LCD 端口输出
任一 LCD 端口输出由 4 位定义。一组的 4 位应该是同一电平，否则输出是不稳定的。如 028 要输出高电平，位 0 至位 3 都应是高。

软件举例：设置 028、029 不变

```
LCD15       .EUQ    0003Fh
            BIS.B   00Fh,&LCD15
```

15. A/D 转换器

目录 页号

A/D 模块特性

- 8 个 A/D 输入通道
- 4 个模拟输入端可作为可编程电流源(经外接电阻 REXT)
- 可作比例测量或绝对值测量
- 内置的采样/保持电路
- 有转换结束中断标志(EOC)
- ADAT 寄存器可将转换结果保存到下一次转换开始
- 低功耗
- 独立完成转换，不需要 CPU 额外的处理开销
- 可编程为 12 或 14 位精度
- 4 个可编程量程可达到 14 位动态范围
- 快速的转换时间
- 大供电电压范围
- 单一覆盖整个 A/D 转换范围

15.1 概述

(12+2)位 A/D 转换器是字访问的外围模块。通过读取 ADAT 寄存器，转换结果出现在 16 位总线上。必须注意，当转换开始后逐位逼近寄存器(SAR)上出现转换的各位。它们立即进入 ADAT 寄存器，并且直到通过对 ACTL 寄存器中的开始转换位（SOC）置位初始化转换过程才被清除。由于 SAR 对 MDB 是透明的，转换进程可以通过对只读的 ADAT 寄存器读取来监视。SOC 置位为新数据清除 SAR 寄存器，同时为下一次转换启动 A/D 转换器时钟。

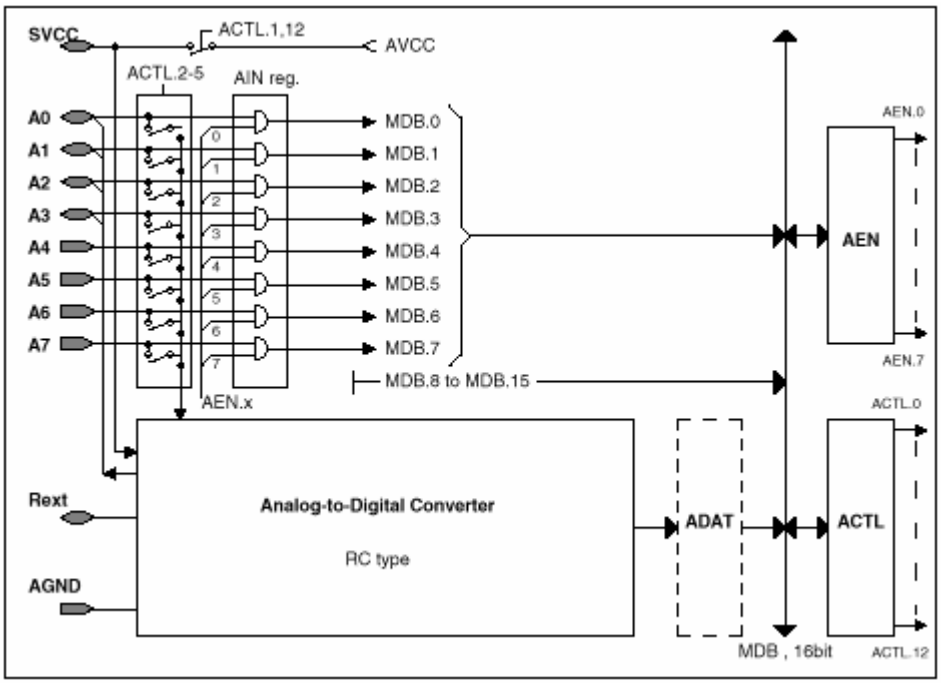


图 15.1：ADC 模块结构

模块有 8 个可选模拟输入通道，经多路切换进入转换器的输入电路，可以在任意时刻对任一通道进行转换。其中的 A0、A1、A2、A3 四个通道还可作为 4 个电流源输出，其值可以经外接电阻 Rext 设定。这些电流源可以开通输出（一次一个）驱动外接传感器来实现比例测量。将外部稳压源接于 SVCC 引脚，可进行精密的绝对测量。

8 个模拟通道也可作为数字输入。对 AEN 寄存器的相应位编程可以使数字信号在全部的 8 个通道或者所选择的通道上输入。通道上的数据可以从 AIN 寄存器上读出。当模拟信号转换正在进行中，邻近通道的数字信号变化会引起交互干扰，导致噪声和输出错码。

根据 ACTL 寄存器中的 b11 的状态，A/D 转换器有两种工作模式，或者 12 位，或者（12+2）位。当输入信号的电压范围已知，将 b11 复位后，ACTL 寄存器中的两位可以定义所要求的转换电压范围。A/D 转换器会在 4 个电压范围之一检测输入信号，然后转换成 12 位分辨率的数据。在这样的非自动模式下，实际上也使转换器的工作产生 14 位的动态范围。而在对 b11 置位所选择的自动模式下，电压范围是由转换器自动选择来达到 14 位效果的。输入信号采样两次，一次是确定选择电压范围的高 2 位，后一次是作转换确定余下的 12 位，以获得（12+2）的结果。在这两种模式中，当转换结束（EOC），中断标志会自动设置以通知微处理机一次转换已经完成。EOC 信号同时关闭 A/D 时钟直至下一次 SOC 位置位，以减少功耗。

注意：ADC，SOC
一次转换在启动后必须在下一次转换启动前完成，否则会产生不可预测的转换数据。

微处理机内核可经过内部系统总线与 A/D 通信，这需要提供适当的模块地址和对 ACTL 及 AEN 寄存器编程。微处理机从 ADAT 寄存器将转换数据读回。

在省电模式下，整个 A/D 转换器关闭以停止电流消耗，但是这只有在 SVCC 不是外部驱动时才正确。当发生转换启动信号或上电信号时，转换器被唤醒，为保证转换精度转换器要经过 6us 才能达到就绪状态。

15.2 A/D 转换操作

15.2.1 A/D 转换

上电后需对 ACTL 寄存器编程，以确定是作比例检测还是绝对检测，以及检测范围的选择是非自动的还是自动的。对于非自动模式，检测范围位一旦选定后在转换期间即不可改变，以免使转换结果失效。

对 ACTL 寄存器中的 SOC 位置位，将激活 A/D 时钟并开始一次新的转换。转换器的工作基于逐位逼近技术，先用一个电阻阵列分辨 M 位高位，然后用一个开关电容阵列分辨余下的 L 位低位。

由 2^M个等权值的电阻组成电阻阵列构成一个D/A。由L个电容器组成的电容阵列构成一个电荷分配型A/D。电容器是 2 进制权值，即它们的容量从最小值开始按 2 的幂增加。电容数

量对应转换器的检测电压范围或输出码的低L位。

当选择了感兴趣模拟通道，采样过程就开始了。模拟输入电压采样到电容阵列的上电极。采样后模拟开关与 A/D 断开，因此这时外部模拟信号可不必输入。

在电阻阵列上完成一次逐次逼近搜索以找到对应于输入电压的电压范围。从一个电阻上得到 VH 和 VL，同时得到高位。电容阵列根据压差（VH-VL）用一个类似的从最高位电容开始的逐位逼近搜索来得到低位。

电容阵列中从最大电容到最小电容连续作开关切换，因此初始存储的电荷在电容上不断地再分配。电阻阵列的开关连接和电容阵列下电极的开关连接改变上电极的电压，使它尽可能接近输入电压，开关的设置对应于 2 进制码（12 或（12+2）位），它可以表示为一个分式 VIN/VREF。

上电极电压由一个带偏移消除电路的比较器监视，以确定输入电压是高于或低于上电极电压，并产生决定逐位逼近搜索方向的数字控制信号。

至最低位电容开关确定后，仍然会存在一个最小电压差异，这就是转换器的分辨率，或者可表示为VREF/2^n，其中的n是数据位数。

转换过程完成后，上电极的电压差根据转换器的最低位分辨率已尽可能地接近于 0。转换结束信号（EOC）指示在 ADAT 寄存器中可以读出一个 12 位或（12+2）位转换数据以作进一步的处理。

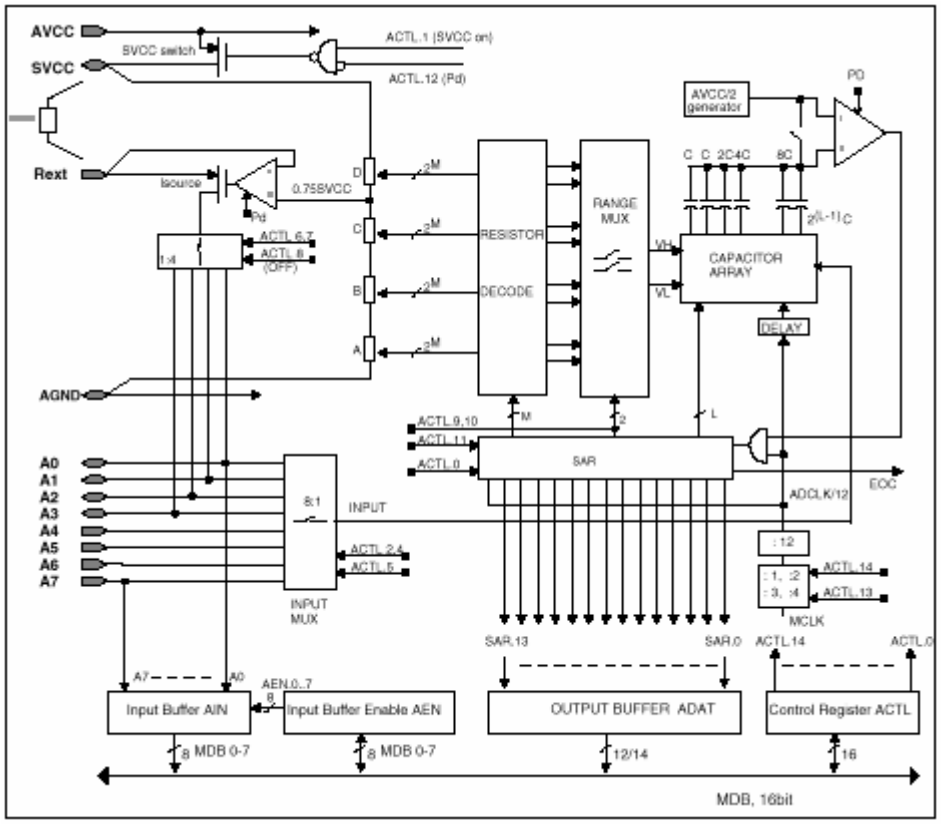


图 5.2：ADC 方框图：

A/D 转换时序

省电模式位的复位将激活 ADC 模块，为使内部偏压能准确地建立，激活过程至少要 6us.

A/D 转换工作在 ADCLK 的 1/12 的时钟频率下。ADCLK 频率的选择要适合电路特性。由于内部的采样和转换网络的时间常数，如果 ADCLK 频率太高将无法保证精确的 12 位转换。由于电容阵列的放电，ADCLK 频率太低也无法保证精度，即使输入信号在采集期间是保持有效和稳定的。用 ACTL 寄存器中的 2 位（ADCLK）可以选择准确的 ADCLK 频率，即对 MCLK 时钟作 1 至 4 分频。

模拟信号采样消耗 12 个 ADCLK 时钟脉冲，12 位的转换需要 7x12 个 ADCLK 周期。当 ACTL.11 复位，A/D 转换工作在预选测量范围的 12 位转换模式时，时间消耗即如上述。总的 12 位转换消耗 96 个 ADCLK 周期。

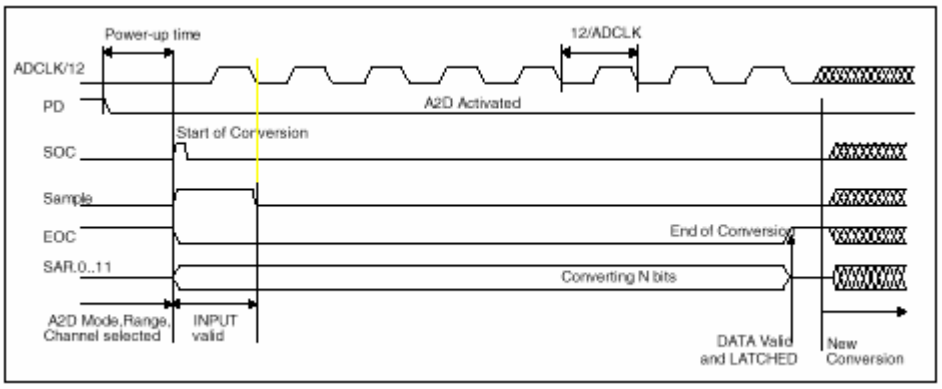


图 15.3：ADC 时序，12 位转换

当 ACTL.11 置位，A/D 工作在自动选择测量范围的（12+2）位转换模式。模拟输入信号要采样两次，每次消耗 12 个 ADCLK 周期。经第一次对输入信号的采样，完成测量范围的转换需花时 24 个 ADCLK 周期。在第二次对输入信号采样后，进行耗时 84 个 ADCLK 周期的 12 位转换。因此，总的（12+2）位转换需要 132 个 ADCLK 周期。

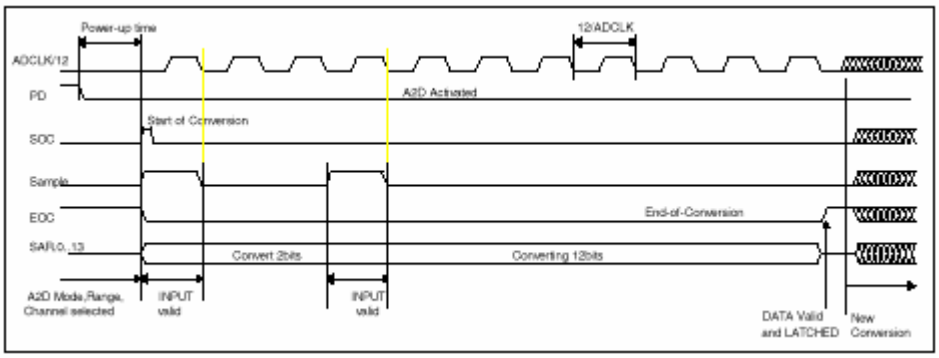


图 15.4：ADC 时序，（12+2）位转换

为了保证转换的精度，在采样期间输入信号必须保持有效和稳定。在整个转换期间相邻通道上也不应有活动的数字信号，以保证不应供电电源毛刺、接地电压变化和交互干扰等造成的误差引起信号品质下降。

A/D 转换器采用了电荷再分配方法，当输入端经内部开关切换到采样端时，因为开关的作用会引起转移电流流入或流出模拟输入端。

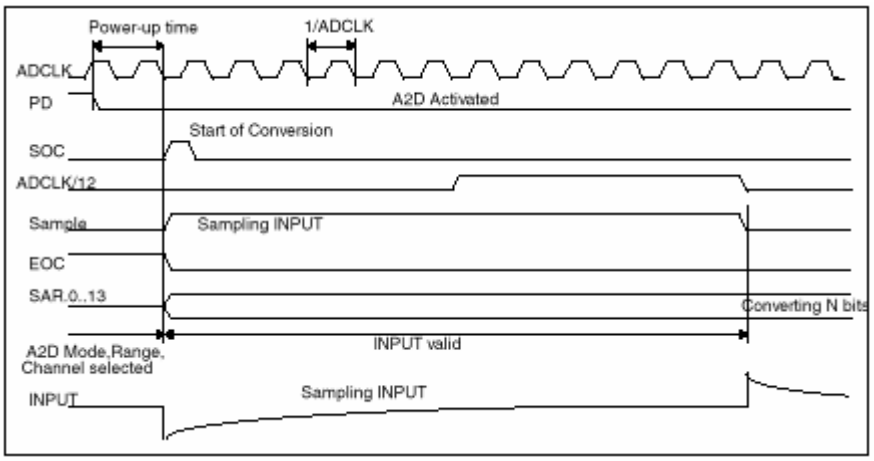


图 15.5：ADC，输入采样时序

这些电流跳变发生在内部采样脉冲的上升沿和下降沿，它们在引起问题之前会很快地衰减和消失，因为它们的时间常数比转换器内部的等效 RC 时间常数要小。模拟输入端可看作由 32pF（电容阵列）和 2kΩ（模拟开关接通）串接的等效 RC 电路。然而当外部的信号源的动态内阻很大，这些跳变在设定的采样时间内可能并未消失，从而难以保证 12 位或（12+2）位的精度。

15.2.2 A/D 中断

当 A/D 转换结束，EOC 信号变高，并用对中断标志位 ADIFG 置位来激活 A/D 中断电路，通知系统的各个部分 A/D 转换已完成。当中断允许位 ADIE 置位，即会发生中断请求。

15.2.3 A/D 量程

可以选择 4 个测量范围之一，以在一个指定的测量范围内产生用 12 位精度的结果。如果 ACTL.11 复位，可以得到等效的 14 位动态范围。测量范围的定义要在转换开始之前通过 ACTL.9 和 ACTL.10 来确定。如果 ACTL.11 置位，转换器在采样期间以对输入信号两次采样的方法找到适当的测量范围，一次用于测量范围选择，第二次实现 12 位转换，因此得到全量程的（12+2）位精度的结果。

测量范围是：

0.00xVREF ≤ VIN ＜ 0.25xVREF　　范围 A
0.25xVREF ≤ VIN ＜ 0.50xVREF　　范围 B
0.50xVREF ≤ VIN ＜ 0.75xVREF　　范围 C
0.75xVREF ≤ VIN ＜ 1.00xVREF　　范围 D

其中 VREF 是 SVCC 端的电压，或者由外部供给，或者连接到 AVCC 上。由 ACTL.12 控制 SVCC 开关来连接。

选定适当的测量范围后，通过相应控制位选择，输入信号被连接到了转换器的输入端。A/D 转换器对选定通道的输入信号作处理，软件可以通过 ADAT 寄存器对转换结果作访问。

在任一测量范围内的数码（10 进制）为：

Ntyp = INT | VIN x 2^{14}/VREF - 2^{13} x ACTL.10 - 2^{12} x ACTL.9 |

其中，ACTL.10 和 ACTL.9 分别是 ACTL 寄存器的第 10 位和第 9 位。

对于 12 位转换模式：

0000h ≤ N ≤ 0FFFh　　范围 A
0000h ≤ N ≤ 0FFFh　　范围 B
0000h ≤ N ≤ 0FFFh　　范围 C
0000h ≤ N ≤ 0FFFh　　范围 D

对于（12+2）位转换模式：

0000h ≤ N ≤ 3FFFh

注意：ADC 偏移电压
任何因对 SVCC 或 AGND 端的寄生阻抗引起在电阻阵列两端产生的电压变化，即偏移电压（Vio）都会对输出数码和转换关系造成失真。

15.2.4 A/D 电流源

有 4 个模拟 I/O 端可以作为电流源输出。电流源输出（Isource）可以经外部电阻 Rext 设定在引脚 A0、A1、A2、A3 上提供。其值为：

Isource=(0.25xSVCC)/Rext, 其中 SVCC 是 SVCC 端电压,Rext 是 SVCC,Rext 间外接电阻。

在作比例测量时，输入通道（仅对 A0,A1,A2,A3）上经阻抗元件产生的电压（Vin）为：

Vin = (0.25 x SVCC) x (Rsens / Rext)，其中 Rsens 是外接阻抗元件的电阻。

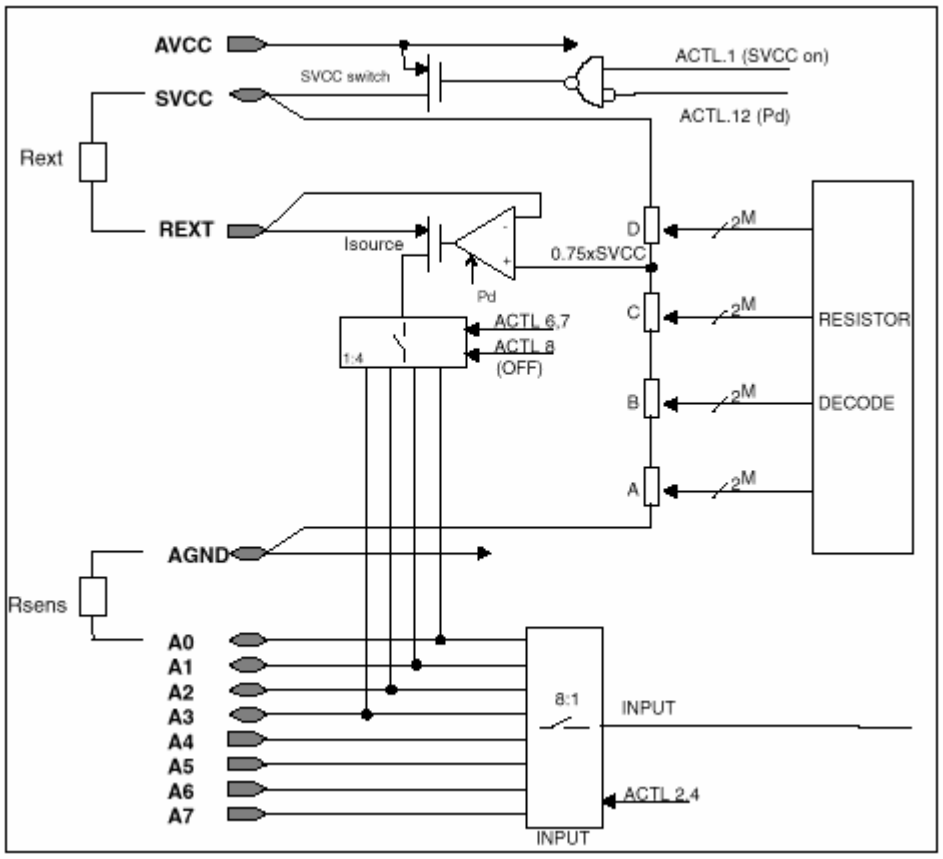


图 15.6：A/D 电流源

A/D 转换器用连接电阻元件来实现传感器应用所需要的精密恒流源，这样才能使输入信号有与电流源相同的与供电电压或参考电压的关系。

15.2.5 A/D 输入端与多路切换

模拟输入

模拟输入信号采样后加在内部电容上，并且在转换期间保持。电容上的电荷由信号源提供，充电时间规定为 12 个ADCLK周期的采样时间。因此外部信号源的电阻和动态阻抗需受限制，RC时间常数要足够短，为确保 12 位精度要在分配好的采样时间内完成输入信号的建立。时间常数的典型值为 $0.8/f_{ADCLK}$ 。高信号源阻抗对转换器精度有不利影响，不仅因为RC建立稳定的时间特性，也由于漏电流或DC输入平均电流（输入开关电流）引起的输入端电压降。对于 12 位转换器，因漏电流引起的以LSB表示的误差为：

Error(LSB) = 4 x 漏电流(uA) x 信号源阻抗(kΩ) / VREF（V)

例：50nA 漏电流、10kΩ信号源阻抗、3V VREF 产生 0.7LSB 的误差。

这一点同样适用于参考电压源的输出阻抗。应足够小以确保能在（0.2/ADCLK）秒内建立稳定，因产生漏电流引入的误差应远小于 1LSB。

模拟多路开关

模拟多路开关用 ACTL 中的控制位确定选择 8 个单端输入通道之一。它用 T 形开关使通道间的耦合减至最小，以避免对输入模拟信号的影响。工作时将未选中的通道与 A/D 绝缘并将中间节点接地 AGND，而分布电容的接地消除了交互干扰。

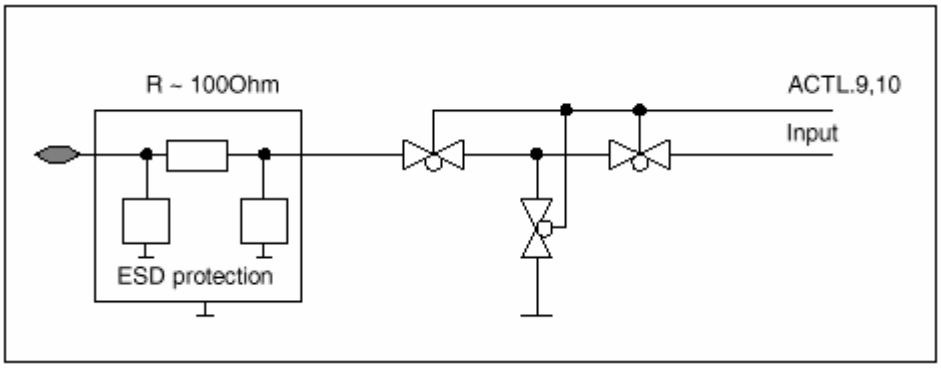


图 15.7：模拟多路开关

存在交互干扰的原因，是总是存在跨越开关的和开关之间的寄生电容。它可能有多种形式，如：一个关断开关的输入和输出间，一个关断的模拟输入通道与一个相邻的开通的输出通道，引起的误差会进入输出的数码中。为了实现高精度的转换，必须采用屏蔽和各种常见的 PCB 布线技术将各种交互干扰减至最小。

15.2.6 A/D 接地与降噪

对任何一个高分辨率转换（≥12 位）必须特别注意 PCB 布线和接地方案，以消除地电流环路、寄生元件参数效应和噪声。有许多标准技术可以在各种应用注意事项中找到。

地电流环路是这样形成的，A/D 的电阻分压器的回传电流流过了与其它模拟或数字电路公共的一段线路。如果不注意，这样的电流会产生一个小的偏移电压，叠加在 A/D 转换器的参考电压和输入电压上。避免地电流环路的方法之一是 AGND 采用星形连接方案。在这种方案中地电流或参考电流不会流过公共的输入回路，从而消除了引起误差的电压。

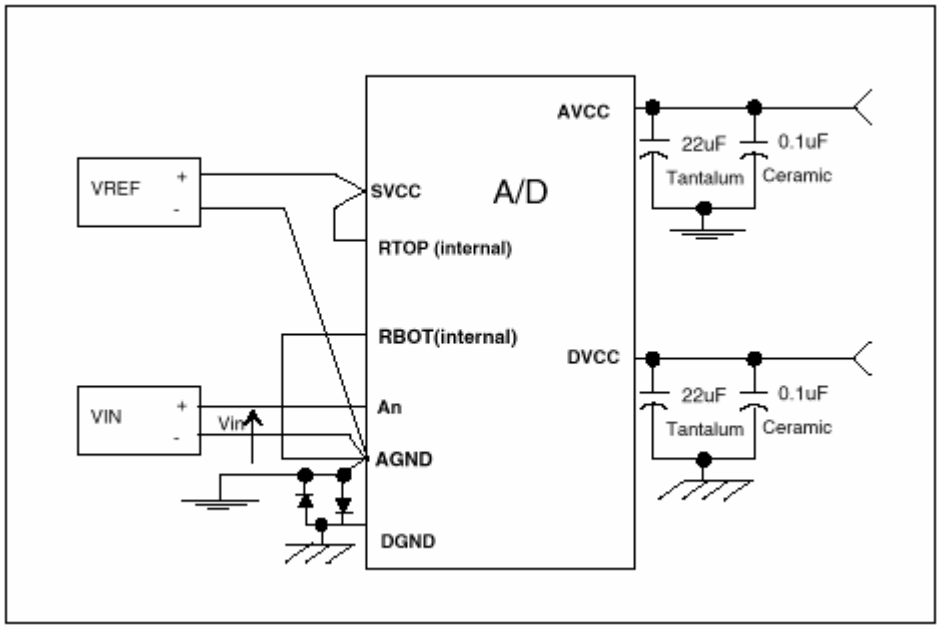


图 15.8：A/D 的接地和消除噪声

数字地DGND与模拟地AGND可以采用星形方案连接在一起。但如果采用分离的供电方案，用两个反向连接的二极管可以限制低于±700mV 的电压差。

此外，因数字开关电路和开关电源造成叠加在供电线路上的纹波和噪声很让人伤脑筋。

通常内部噪声很小，折算成输入信号远小于一个 LSB，因此输出代码非常稳定。然而噪声经过供电电路及接地线耦合到设备上，噪声的作用加强了，信号会引起不稳定。这可以通过多次采样求平均的方法改善。另一方面，如果 SVCC 或 VREF 的电压降低，相当于 LSB 对应的绝对数值降低，这样也造成噪声的作用加强。因此，为达到所要求的精度，干净而无噪声的电路是最重要的。

仔细地安排旁路电容，建立回到各自的地的回路，对稳定供电电流和降低噪声是有帮助的。

15.2.7 A/D 输入与输出引脚

输入引脚

有两种不同类型的输入信号，即模拟输入端 A0、A1、A2、A3、A4、A5、A6、A7 和 REXT、SVCC。

从通道 A0 到 A7 输入的信号，可以作为 A/D 转换器处理的模拟信号，也可以作为读入到处理单元的数字信号。REXT 和 SVCC 间的外接电阻决定了被激活的电流源工作时的总电流。SVCC 端可以作为输入或输出，当内部 SVCC 开关关断时，SVCC 作为输入，VREF 由外部提供。当内部 SVCC 开关开通时，SVCC 作为输出。

输出引脚

有两种不同类型的输出信号，即作为输出的 A0、A1、A2、A3 和 SVCC。

当选择电流源功能时，模拟端 A0、A1、A2、A3 之一可以输出电流。当 SVCC 开关开通时，SVCC 端有与 AVCC 相同的电压。

电源引脚

有 4 个供电引脚，分成数字和模拟电流通道：

AVCC，DVCC，AGND，DGND

部分 MSP430 型号有全部 4 个引脚，以达到高模拟分辨率。而其它的型号只有针对模拟和数字的 VCC 和 GND，引脚已在内部相连。

15.3 A/D 控制寄存器

有 4 个控制寄存器：

寄存器	简称	寄存器类型	地址	初始状态
输入寄存器	AIN	只读	0110h	...
输入允许寄存器	AEN	读写	0112h	复位
ADC 控制寄存器	ACTL	读写	0114h	见图
保留			0116h	
ADC 数据寄存器	ADAT	只读	0118h	...

所有寄存器可以用任意指令以寄存器读写模式访问。

输入寄存器 AIN

A0 到 A7 上的信号可以作为模拟信号或数字信号。数字值可以通过访问输入寄存器读出。作为数字信号读入由输入允许寄存器选择。

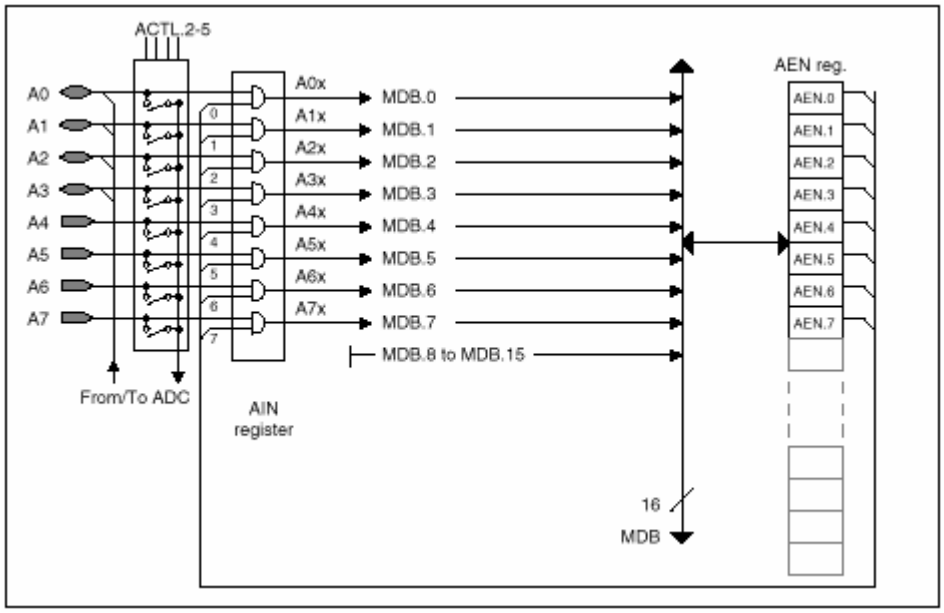


图 15.9：ADC 输入寄存器，输入允许寄存器

输入寄存器 AIN 作为只读寄存器连接于 16 位的 MDB。寄存器的低字节有效，MDB0 到 MDB7 对应于 A0 到 A7。寄存器的高字节读出总是 00h。

AIN（110h）

MDB.15							MDB.8	MDB.7							MDB.0
0	0	0	0	0	0	0	0	AIN.7	AIN.6	AIN.5	AIN.4	AIN.3	AIN.2	AIN.1	AIN.0
r0	r0	r0	r0	r0	r0	r0	r0	r	r	r	r	r	r	r	r
								↑	↑	↑	↑	↑	↑	↑	↑
								A7x	A6x	A5x	A4x	A3x	A2x	A1x	A0x

输入信号的各位由相应位的允许信号作为门控，即 Ax .AND. AENx。未选中（不允许）的各位读出为“0”。

输入允许寄存器 AEN

输入允许寄存器 AEN 作为读写寄存器连接于 16 位的 MDB。寄存器的低字节有效，MDB0 到 MDB7 对应于 A0 到 A7。寄存器的高字节读出总是 00h。

AEN（112h）

MDB.15							MDB.8	MDB.7							MDB.0
0	0	0	0	0	0	0	0	AEN.7	AEN.6	AEN.5	AEN.4	AEN.3	AEN.2	AEN.1	AEN.0
r0	r0	r0	r0	r0	r0	r0	r0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

输入允许寄存器是按位控制的。

AEN.x = 0　模拟输入，对 AIN 寄存器访问时，这一位读出为 0。

AEN.x = 1　数字输入，对 AIN 寄存器访问时，这一位读出为相应引脚的逻辑电平。

初始状态各位为 0。

ADC 数据寄存器 ADAT

ADC 数据寄存器保持 A/D 转换结果。转换数据在转换结束时进入寄存器，并保持到由对 SOC 位置位启动的下一次转换。

ADAT（118h） ACTL.11=0

MDB.15															MDB.0
0	0	0	0	MSB											LSB
r0	r0	r0	r0	r	r	r	r	r	r	r	r	r	r	r	r

ADAT（118h） ACTL.11=0

MDB.15															MDB.0
0	0	RA1	RA0	MSB											LSB
r0	r0	r	r	r	r	r	r	r	r	r	r	r	r	r	r

ADC 控制寄存器 ACTL

ACTL（114h）

MDB.15															MDB.0
0	ADCLK		Pd	范围选择			电流源			A/D 输入选择				Vref	CS
r0	rw-0	rw-0	rw-1	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	(w)r0

ACTL.0 转换启动

ACTL.1 VREF 来源

ACTL.1=0，SVCC 开关关断，SVCC 与 VCC 不连，VREF 由外部提供

ACTL.1=1，SVCC 开关开通，SVCC 与 VCC 相连，VREF 不必由外部提供

ACTL.2-5 AD 输入选择

这些位选择转换的通道。通道的改变只能在转换完成后进行。在转换过程中改变通道将使转换过程失效。

ACTL.5	ACTL.4	ACTL.3	ACTL.2	通道
0	0	0	0	A0
0	0	0	1	A1
0	0	1	0	A2
0	0	1	1	A3
0	1	0	0	A4
0	1	0	1	A5
0	1	1	0	A6
0	1	1	1	A7
1	X	X	X	无

ACTL.6-8 AD 电流源输出控制

这些位用于选择用作电流源输出的通道。通道的改变只能在转换完成后进行。在转换过程中改变通道将使转换过程失效。

ACTL.8	ACTL.7	ACTL.6	通道
0	0	0	A0
0	0	1	A1
0	1	0	A2

0	1	1	A3
1	X	X	无

ACTL.9-11　测量范围选择。

这些位在转换启动后必须保持不变。在转换进行中对它们处理会使错误的转换数据进入 ADAT。

ACTL.11	ACTL.10	ACTL.9	范围
0	0	0	A
0	0	1	B
0	1	0	C
0	1	1	D
1	X	X	自动

ACTL.11　测量范围选择模式。

ACTL.11=0，由 ACTL.9 和 ACTL.10 实现非自动的测量范围选择。

ACTL.11=1，激活自动选择测量范围的（12+2）位转换模式。此时不必关心 ACTL.9 与 ACTL.10 的状态

ACTL.12　省电模式（Pd）

ACTL.12=1，SVCC 开关关断，
比较器进入省电模式，
电流源关断。

ACTL.13-14　ADCLK 选择

ADC 的时钟频率应调整到器件参数说明的最高频率。

ACTL.14	ACTL.13	ADCLK
0	0	MCLK
0	1	MCLK/2
1	0	MCLK/3
1	1	MCLK/4

ACTL.15　保留。

软件测试不足与溢出

因为 ADAT 寄存器是 16 位外围模块，以字模式处理数据，不能检测溢出和不足。它可以简化为程序中的几条简单的指令。

```
; Bit11 of the ACTL register is reset, a 12-bit conversion was active
;
        CMP     #0,&ADAT        ; test for 12-bit ADC underflow
        JEQ     UndFlow         ; Yes, continue with underflow handling
        CMP     #0FFFh,&ADAT    ; test for 12-bit ADC overflow
;
; The MSBits, not implemented in the converter's hardware are read as 0s
        JEQ     OverFlow        ; Yes, continue with overflow handling
; Bit11 of the ACTL register is set, a (12+2)-bit conversion was active
; The conversion range should be limited to Range A to C
;
        CMP     #0,&ADAT        ; test for (12+2)-bit ADC underflow
        JEQ     UndFlow         ; Yes, continue with underflow handling
        CMP     #2FFFh,&ADAT    ; test only for Range A to C overflow
;
; The MSBits, not implemented in the converter's hardware are read as 0s
        JHS     OverFlow        ; Yes, continue with overflow handling
```

16. 其它模块

目录 页号

16.1 晶体振荡器

晶振运行的所有元件都集成在 MSP430 内，不需别的外部器件。晶振电路是按超低功耗要求设计的。因此印制电路设计应使晶体和 MSP430 器件间的连接距离要短。

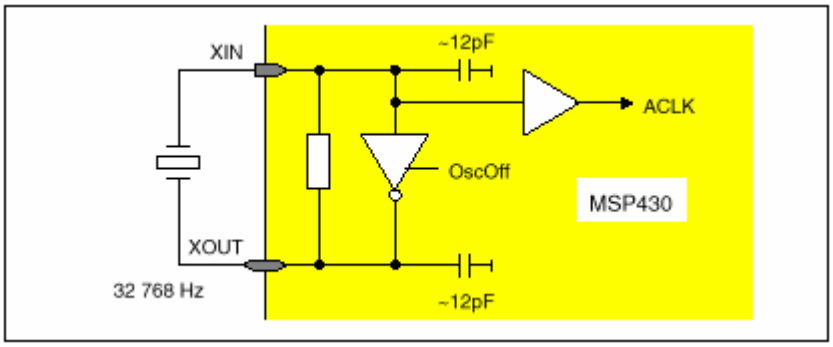


图 16.1：晶振原理图

在 OSCOff 模式下，ACLK 信号保持为高电平。

16.2 上电电路

上电电路是系统复位电路的一部分，它由两部分组成：上电复位检测和上电复位延迟。为了给系统提供复位条件，POR 的输出延迟送到 POR 锁存器和 PUC 锁存器，以保证对这两个锁存器置位。

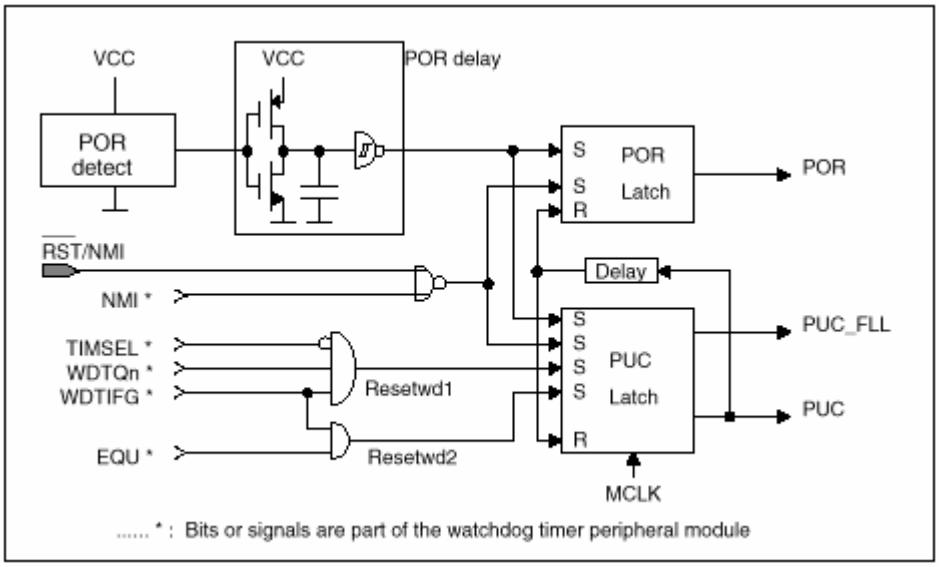


图 16.2：上电复位和上电清除电原理图

VCC 的上升时间较快时，POR 延迟足够长的时间，以保证 POR 信号在上电后能正确地对电路作初始化。

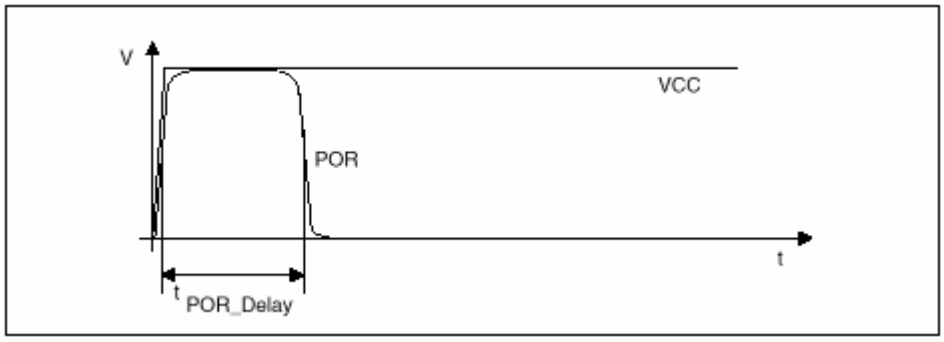


图 16.3：VCC 上电时间较快时的上电复位

VCC 的上升时间较慢时，POR 通过检测 VCC 来确定 POR 信号，以保证 POR 信号在上电后能正确地对电路初始化。

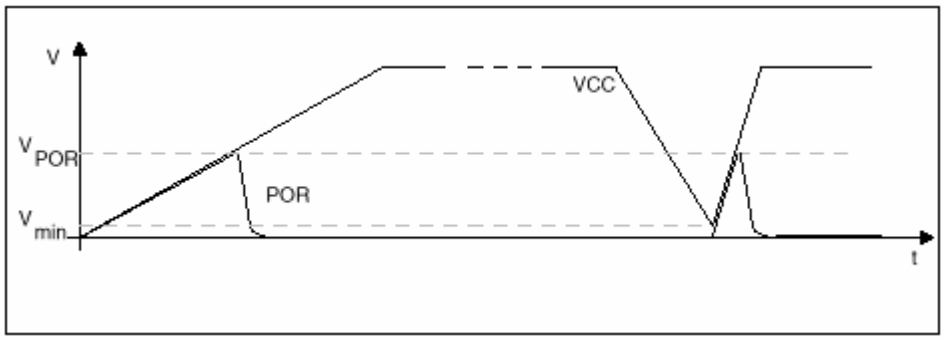


图 16.4：VCC 上电时间较慢时的上电复位

图中供电电压 VCC 跌落到 Vmin 以下，以确保第二个 POR 信号在 VCC 电压又一次上升时发生。如果 VCC 不能跌落到小于 Vmin，POR 信号不会产生，上电条件不能建立。

16.3 晶振缓冲输出

由控制寄存器 CBCTL 选择晶体缓冲输出的频率。

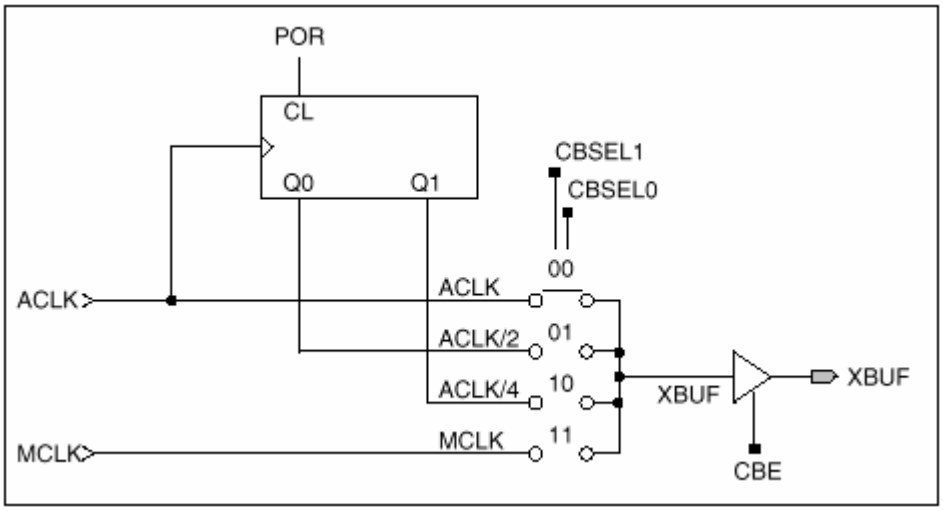


图 16.5：晶振缓冲原理图

时钟缓冲输出外围模块的控制寄存器控制 XBUF 引脚输出的频率及输出缓冲的三态。

分频器运行在正确操作必需的最小值。例如，选择 ACLK 或 MCLK，或者 CBE 置位时，它将暂停工作。

控制寄存器的 3 位 CBSEL1、CBSECO、CBE 用 POR 信号复位。在 VCC 上电过程中，或者在 RST*/NMI 选择复位功能时接地，POR 信号有效。

CBCTL（053h）

7							0
---	---	---	---	---	CBSEL1	CBSEL0	CBE
					w-(0)	w-(0)	w-(0)

位 0：CBE 位控制输出缓冲的三态。

CBE=1：输出缓冲允许

CBE=0：输出缓冲禁止

在上电复位（POR）时，输出缓冲总是禁止的。外部元件不会得到所选的频率。

位 1、2：CBSEL1 和 CBSEL0 位选择在 XBUF 引脚上提供的频率信号。

CBSEL1	CBSEL0	XBUF
0	0	ACLK（POR 后状态）
0	1	ACLK/2
1	0	ACLK/4
1	1	MCLK

附录 A. 外围模块地址分配

本附录集中了外围模块和控制位信息供参考。

每一外围模块的寄存器表示为一行含寄存器的控制位或状态位的表格。此寄存器的符号和外围模块的 16 进制地址在每一行的左边。

各位的可访问性与/和硬件条件用以下定义符号标明在每一位的下方：

- rw: 读/写
- r: 只读
- r0: 读出为“0”
- r1: 读出为“1”
- w: 只写
- w0: 写入“0”
- w1: 写入“1”
- (w): 无寄存器位，写入“1”将产生一个脉冲，读出总为“0”
- h0: 由硬件复位
- h1: 由硬件置位
- -0，-1: PUC 信号活动（Reset 或 WDT）后的状态
- -(0)，-(1): POR 信号活动（Reset）后的状态

SFR，字节访问

Bit # -	7	6	5	4	3	2	1	0
000Fh								
Module enable 2, ME2 0005h							UTXE rw-0	URXE rw-0
Module enable 1, ME1 0004h								
Interrupt flag 2, IFG2 0003h	BTIFG rw					ADIFG rw-0	UTXIFG rw-0	URXIFG rw-0
Interrupt flag 1, IFG1 0002h				NMIIFG rw-0	P0IFG.1 rw-0	P0IFG.0 rw-0	OFIFG rw-1	WDTIFG rw 1)
Interrupt enable 2,IE2 0001h	BTIE rw-0				3) rw-0	2) rw-0	UTXIE rw-0	URXIE rw-0
Interrupt enable 1, IE1 0000h					P0IE.1 rw-0	P0IE.0 rw-0	OFIE rw-0	WDTIE rw-0

1）WDTIFG 由 POR 信号复位，由 WDT 溢出或口令不符置位

2）对 320 型：　　ADIE，针对 12+2b ADC 模块（类型：rw-0）

　　对 310 型：　　TPIE，针对定时器/端口模块（类型：rw-0）

3）对 320、330 型：　TPIE，针对定时器/端口模块（类型：rw-0）

数字 I/O 组，字节访问

Bit # -	7	6	5	4	3	2	1	0
Direction reg., P4SEL 001Fh	P4SEL.7 rw-0	P4SEL.6 rw-0	P4SEL.5 rw-0	P4SEL.4 rw-0	P4SEL.3 rw-0	P4SEL.2 rw-0	P4SEL.1 rw-0	P4SEL.0 rw-0
Direction reg., P4DIR 001Eh	P4DIR.7 rw-0	P4DIR.6 rw-0	P4DIR.5 rw-0	P4DIR.4 rw-0	P4DIR.3 rw-0	P4DIR.2 rw-0	P4DIR.1 rw-0	P4DIR.0 rw-0
Output reg., P4OUT 001Dh	P4OUT.7 rw	P4OUT.6 rw	P4OUT.5 rw	P4OUT.4 rw	P4OUT.3 rw	P4OUT.2 rw	P4OUT.1 rw	P4OUT.0 rw
Input register, P4IN 001Ch	P4IN.7 r	P4IN.6 r	P4IN.5 r	P4IN.4 r	P4IN.3 r	P4IN.2 r	P4IN.1 r	P4IN.0 r
Direction reg., P3SEL 001Bh	P3SEL.7 rw-0	P3SEL.6 rw-0	P3SEL.5 rw-0	P3SEL.4 rw-0	P3SEL.3 rw-0	P3SEL.2 rw-0	P3SEL.1 rw-0	P3SEL.0 rw-0
Direction reg., P3DIR 001Ah	P3DIR.7 rw-0	P3DIR.6 rw-0	P3DIR.5 rw-0	P3DIR.4 rw-0	P3DIR.3 rw-0	P3DIR.2 rw-0	P3DIR.1 rw-0	P3DIR.0 rw-0
Output reg., P3OUT 0019h	P3OUT.7 rw	P3OUT.6 rw	P3OUT.5 rw	P3OUT.4 rw	P3OUT.3 rw	P3OUT.2 rw	P3OUT.1 rw	P3OUT.0 rw
Input register, P3IN 0018h	P3IN.7 r	P3IN.6 r	P3IN.5 r	P3IN.4 r	P3IN.3 r	P3IN.2 r	P3IN.1 r	P3IN.0 r
0017h								
0016h								
Interrupt Enable, P0IE 0015h	P0IE.7 rw-0	P0IE.6 rw-0	P0IE.5 rw-0	P0IE.4 rw-0	P0IE.3 rw-0	P0IE.2 rw-0	*) r0	*) r0
Int. Edge Sel., P0IES 0014h	P0IES.7 rw	P0IES.6 rw	P0IES.5 rw	P0IES.4 rw	P0IES.3 rw	P0IES.2 rw	P0IES.1 rw	P0IES.0 rw
Interrupt Flags, P0IFG 0013h	P0IFG.7 rw-0	P0IFG.6 rw-0	P0IFG.5 rw-0	P0IFG.4 rw-0	P0IFG.3 rw-0	P0IFG.2 rw-0	*) r0	*) r0
Direction reg., P0DIR 0012h	P0DIR.7 rw-0	P0DIR.6 rw-0	P0DIR.5 rw-0	P0DIR.4 rw-0	P0DIR.3 rw-0	P0DIR.2 rw-0	P0DIR.1 rw-0	P0DIR.0 rw-0
Output reg., P0OUT 0011h	P0OUT.7 rw	P0OUT.6 rw	P0OUT.5 rw	P0OUT.4 rw	P0OUT.3 rw	P0OUT.2 rw	P0OUT.1 rw	P0OUT.0 rw
Input register, P0IN 0010h	P0IN.7 r	P0IN.6 r	P0IN.5 r	P0IN.4 r	P0IN.3 r	P0IN.2 r	P0IN.1 r	P0IN.0 r

*) 中断允许位和中断标志位包含在 SFR 组中。

数字 I/O 组，字节访问

Bit # -	7	6	5	4	3	2	1	0
002Fh								
Direction reg., P2SEL 002Eh	P2SEL.7 rw-0	P2SEL.6 rw-0	P2SEL.5 rw-0	P2SEL.4 rw-0	P2SEL.3 rw-0	P2SEL.2 rw-0	P2SEL.1 rw-0	P2SEL.0 rw-0
nterrupt Enable, P2IE 002Dh	P0IE.7 rw-0	P2IE.6 rw-0	P2IE.5 rw-0	P2IE.4 rw-0	P2IE.3 rw-0	P2IE.2 rw-0	P2IE.1 rw-0	P2IE.0 rw-0
Int. Edge Sel., P2IES 002Ch	P2IES.7 rw	P2IES.6 rw	P2IES.5 rw	P2IES.4 rw	P2IES.3 rw	P2IES.2 rw	P2IES.1 rw	P2IES.0 rw
nterrupt Flags, P2IFG 002Bh	P2IFG.7 rw-0	P2IFG.6 rw-0	P2IFG.5 rw-0	P2IFG.4 rw-0	P2IFG.3 rw-0	P2IFG.2 rw-0	P2IFG.1 rw-0	P2IFG.0 rw-0
Direction reg., P2DIR 002Ah	P2DIR.7 rw-0	P2DIR.6 rw-0	P2DIR.5 rw-0	P2DIR.4 rw-0	P2DIR.3 rw-0	P2DIR.2 rw-0	P2DIR.1 rw-0	P2DIR.0 rw-0
Output reg., P2OUT 0029h	P2OUT.7 rw	P2OUT.6 rw	P2OUT.5 rw	P2OUT.4 rw	P2OUT.3 rw	P2OUT.2 rw	P2OUT.1 rw	P2OUT.0 rw
Input register, P2IN 0028h	P2IN.7 r	P2IN.6 r	P2IN.5 r	P2IN.4 r	P2IN.3 r	P2IN.2 r	P2IN.1 r	P2IN.0 r
0027h								
Direction reg., P1SEL 0026h	P1SEL.7 rw-0	P1SEL.6 rw-0	P1SEL.5 rw-0	P1SEL.4 rw-0	P1SEL.3 rw-0	P1SEL.2 rw-0	P1SEL.1 rw-0	P1SEL.0 rw-0
nterrupt Enable, P1IE 0025h	P0IE.7 rw-0	P1IE.6 rw-0	P1IE.5 rw-0	P1IE.4 rw-0	P1IE.3 rw-0	P1IE.2 rw-0	P1IE.1 rw-0	P1IE.0 rw-0
Int. Edge Sel., P1IES 0024h	P1IES.7 rw	P1IES.6 rw	P1IES.5 rw	P1IES.4 rw	P1IES.3 rw	P1IES.2 rw	P1IES.1 rw	P1IES.0 rw
nterrupt Flags, P1IFG 0023h	P1IFG.7 rw-0	P1IFG.6 rw-0	P1IFG.5 rw-0	P1IFG.4 rw-0	P1IFG.3 rw-0	P1IFG.2 rw-0	P1IFG.1 rw-0	P1IFG.0 rw-0
Direction reg., P1DIR 0022h	P1DIR.7 rw-0	P1DIR.6 rw-0	P1DIR.5 rw-0	P1DIR.4 rw-0	P1DIR.3 rw-0	P1DIR.2 rw-0	P1DIR.1 rw-0	P1DIR.0 rw-0
Output reg., P1OUT 0021h	P1OUT.7 rw	P1OUT.6 rw	P1OUT.5 rw	P1OUT.4 rw	P1OUT.3 rw	P1OUT.2 rw	P1OUT.1 rw	P1OUT.0 rw
Input register, P1IN 0020h	P1IN.7 r	P1IN.6 r	P1IN.5 r	P1IN.4 r	P1IN.3 r	P1IN.2 r	P1IN.1 r	P1IN.0 r

LCD 寄存器组，字节访问

Bit # -	7	6	5	4	3	2	1	0
LCD Memory 15 003Fh	S29C3 rw	S29C2 rw	S29C1 rw	S29C0 rw	S28C3 rw	S28C2 rw	S28C1 rw	S28C0 rw
LCD Memory 14 003Eh	S27C3 rw	S27C2 rw	S27C1 rw	S27C0 rw	S26C3 rw	S26C2 rw	S26C1 rw	S26C0 rw
LCD Memory 13 003Dh	S25C3 rw	S25C2 rw	S25C1 rw	S25C0 rw	S24C3 rw	S24C2 rw	S24C1 rw	S24C0 rw
LCD Memory 12 003Ch	S23C3 rw	S23C2 rw	S23C1 rw	S23C0 rw	S22C3 rw	S22C2 rw	S22C1 rw	S22C0 rw
LCD Memory 11 003Bh	S21C3 rw	S21C2 rw	S21C1 rw	S21C0 rw	S20C3 rw	S20C2 rw	S20C1 rw	S20C0 rw
LCD Memory 10 003Ah	S19C3 rw	S19C2 rw	S19C1 rw	S19C0 rw	S18C3 rw	S18C2 rw	S18C1 rw	S18C0 rw
LCD Memory 9 0039h	S17C3 rw	S17C2 rw	S17C1 rw	S17C0 rw	S16C3 rw	S16C2 rw	S16C1 rw	S16C0 rw
LCD Memory 8 0038h	S15C3 rw	S15C2 rw	S15C1 rw	S15C0 rw	S14C3 rw	S14C2 rw	S14C1 rw	S14C0 rw
LCD Memory 7 0037h	S13C3 rw	S13C2 rw	S13C1 rw	S13C0 rw	S12C3 rw	S12C2 rw	S12C1 rw	S12C0 rw
LCD Memory 6 0036h	S11C3 rw	S11C2 rw	S11C1 rw	S11C0 rw	S10C3 rw	S10C2 rw	S10C1 rw	S10C0 rw
LCD Memory 5 0035h	S9C3 rw	S9C2 rw	S9C1 rw	S9C0 rw	S8C3 rw	S8C2 rw	S8C1 rw	S8C0 rw
LCD Memory 4 0034h	S7C3 rw	S7C2 rw	S7C1 rw	S7C0 rw	S6C3 rw	S6C2 rw	S6C1 rw	S6C0 rw
LCD Memory 3 0033h	S5C3 rw	S5C2 rw	S5C1 rw	S5C0 rw	S4C3 rw	S4C2 rw	S4C1 rw	S4C0 rw
LCD Memory 2 0032h	S3C3 rw	S3C2 rw	S3C1 rw	S3C0 rw	S2C3 rw	S2C2 rw	S2C1 rw	S2C0 rw
LCD Memory 1 0031h	S1C3 rw	S1C2 rw	S1C1 rw	S1C0 rw	S0C3 rw	S0C2 rw	S0C1 rw	S0C0 rw
CD Cntl&Mode, LCDC 0030h	LCDM7 rw-0	LCDM6 rw-0	LCDM5 rw-0	LCDM4 rw-0	LCDM3 rw-0	LCDM2 rw-0	LCDM1 rw-0	LCDM0 rw-0

注意：LCD 存储器位按 MSP430 规则命名。各位名称的第一部分是相应的 SEG 线，第二部分是相应的 COM 线。

例如，对于用了 SEG4 和 COM3 的段的命名： S4C3。

8 位定时器/计数器组，Basic Timer 组，定时器/端口组，字节访问

Bit # -	7	6	5	4	3	2	1	0
Timer/Port Enable reg., TPE 04Fh	TPSSEL3 rw-0	TPSSEL2 rw-0	TPE.5 rw-0	TPE.4 rw-0	TPE.3 rw-0	TPE.2 rw-0	TPE.1 rw-0	TPE.0 rw-0
Timer/Port Data reg., TPD 04Eh	B16 rw-0	CPON rw-0	TPD.5 rw-0	TPD.4 rw-0	TPD.3 rw-0	TPD.2 rw-0	TPD.1 rw-0	TPD.0 rw-0
Timer/Port Counter1, TPCNT2 04Dh	2^7 rw	2^6 rw	2^5 rw	2^4 rw	2^3 rw	2^2 rw	2^1 rw	2^0 rw
Timer/Port Counter1, TPCNT1 04Ch	2^7 rw	2^6 rw	2^5 rw	2^4 rw	2^3 rw	2^2 rw	2^1 rw	2^0 rw
Timer/Port control reg., TPCTL 04Bh	TPSSEL1 rw-0	TPSSEL0 rw-0	ENB rw-0	ENA rw-0	EN1 r-0	RC2FG rw-0	RC1FG rw-0	EN1FG rw-0
Counter Data, 8bit Basic Timer, BTCNT2 0047h	2^7 rw	2^6 rw	2^5 rw	2^4 rw	2^3 rw	2^2 rw	2^1 rw	2^0 rw
Counter Data, 8bit Basic Timer, BTCNT1 0046h	2^7 rw	2^6 rw	2^5 rw	2^4 rw	2^3 rw	2^2 rw	2^1 rw	2^0 rw
045h								
Counter Data, 8bit Timer/Counter, TCDAT 0044h	TCDAT.7 rw	TCDAT.6 rw	TCDAT.5 rw	TCDAT.4 rw	TCDAT.3 rw	TCDAT.2 rw	TCDAT.1 rw	TCDAT.0 rw
Pre-load Register, 8bit Timer/Counter, TCPLD 0043h	TCPLD.7 rw	TCPLD.6 rw	TCPLD.5 rw	TCPLD.4 rw	TCPLD.3 rw	TCPLD.2 rw	TCPLD.1 rw	TCPLD.0 rw
Control Register, 8bit Timer/Counter, TCCTL 0042h	SSEL1 rw-0	SSEL0 rw-0	ISCTL rw-0	TXEN rw-0	ENCNT rw-0	RXACT rw-0	TXD rw-0	RXD r(-1)
0041h								
Basic Timer, BTCTL 0040h	SSEL rw	Reset * Hold ** rw	DIV rw	FRFQ1 rw	FRFQ0 rw	IP2 rw	IP1 rw	IP0 rw

PWM 定时器，EPROM 控制寄存器和系统时钟发生器组，字节访问

Bit # -	7	6	5	4	3	2	1	0
PWM timer counter PWMCNT.2 005Fh	2^7 rw-0	2^6 rw-0	2^5 rw-0	2^4 rw-0	2^3 rw-0	2^2 rw-0	2^1 rw-0	2^0 rw-0
PWM duty register PWMDTR.2 005Eh	2^7 rw-1	2^6 rw-1	2^5 rw-1	2^4 rw-1	2^3 rw-1	2^2 rw-1	2^1 rw-1	2^0 rw-1
PWM duty buffer PWMDTB.2 005Dh	2^7 rw-0	2^6 rw-0	2^5 rw-0	2^4 rw-0	2^3 rw-0	2^2 rw-0	21 rw-0	2^0 rw-0
PWM timer control register PWMCTL.2 005Ch	---- rw-0	SSEL2 rw-0	SSEL1 rw-0	SSEL0 rw-0	CMPM r	---- rw-0	OS rw-0	OE rw-0
PWM timer counter PWMCNT.1 005Bh	2^7 rw-0	2^6 rw-0	2^5 rw-0	2^4 rw-0	2^3 rw-0	2^2 rw-0	2^1 rw-0	2^0 rw-0
PWM duty register PWMDTR.1 005Ah	2^7 rw-1	2^6 rw-1	2^5 rw-1	2^4 rw-1	2^3 rw-1	2^2 rw-1	2^1 rw-1	2^0 rw-1
PWM duty buffer PWMDTB.1 0059h	2^7 rw-0	2^6 rw-0	2^5 rw-0	2^4 rw-0	2^3 rw-0	2^2 rw-0	2^1 rw-0	2^0 rw-0
PWM timer control register PWMCTL.1 0058h	---- rw-0	SSEL2 rw-0	SSEL1 rw-0	SSEL0 rw-0	CMPM r	---- rw-0	OS rw-0	OE rw-0
'ROM control register EPCTL 0054h	r-0	r-0	r-0	r-0	r-0	r-0	VPPS rw-0	EXE rw-0
Crystal Buffer ctl. reg. *) CBCTL 053h						CBSEL1 w-(0)	CBSEL0 w-(0)	CBE w-(0)
System Clock Gen., req. Cntl. SCFQCTL 0052h	M rw-0	2^6 rw-0	2^5 rw-0	2^4 rw-1	2^3 rw-1	2^2 rw-1	2^1 rw-1	2^0 rw-1
System Clock Gen., req. Integrator SCFI1 0051h	2^9 rw-0	2^8 rw-0	2^7 rw-0	2^6 rw-0	2^5 rw-0	2^4 rw-0	2^3 rw-0	2^2 rw-0
System Clock Gen., req. Integrator SCFI0 0050h	0 r	0 r	0 r	FN_4 rw-0	FN_3 rw-0	FN_2 rw-0	2^1 rw-0	2^0 rw-0

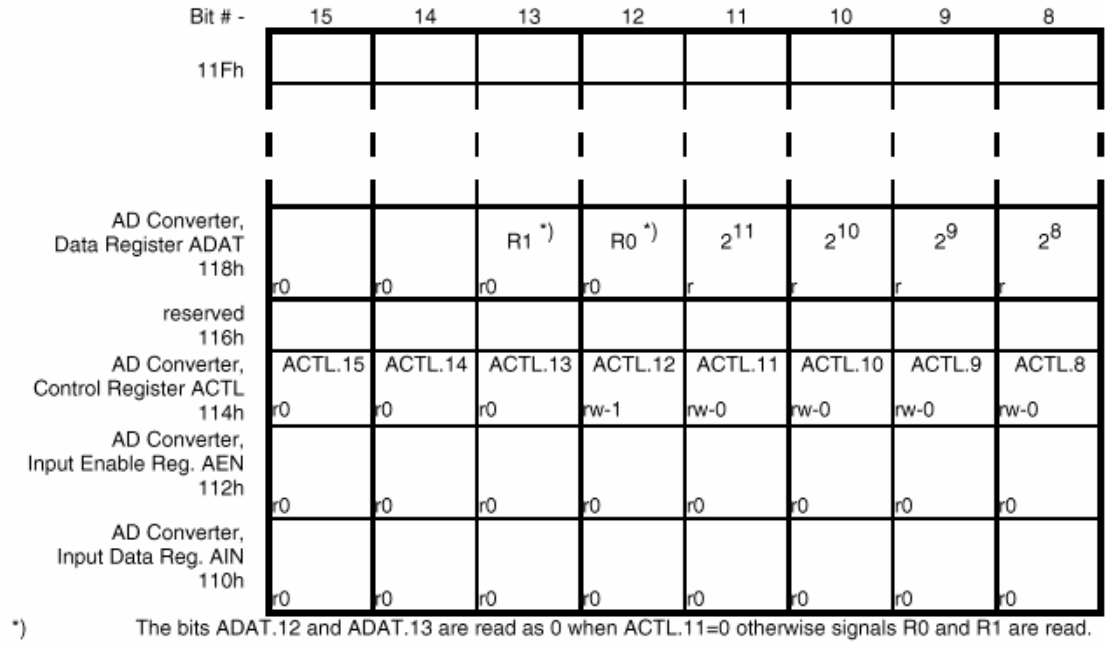
*) CBSel1, CBSEL0 and CBE bit are reset with POR signal.

USART 组，选择 UART 模式：SYNC 位= 0，字节访问

Bit # -	7	6	5	4	3	2	1	0
07Fh								
Transmit Buffer TXBUF 077h	2^7 rw	2^6 rw	2^5 rw	2^4 rw	2^3 rw	2^2 rw	2^1 rw	2^0 rw
Receive Buffer RXBUF 076h	2^7 r	2^6 r	2^5 r	2^4 r	2^3 r	2^2 r	2^1 r	2^0 r
Baud Rate UBR1 075h	2^{15} rw	2^{14} rw	2^{13} rw	2^{12} rw	2^{11} rw	2^{10} rw	2^9 rw	2^8 rw
Baud Rate UBR0 074h	2^7 rw	2^6 rw	2^5 rw	2^4 rw	2^3 rw	2^2 rw	2^1 rw	2^0 rw
Modulation Control UMCTL 073h	m7 rw	m6 rw	m5 rw	m4 rw	m3 rw	m2 rw	m1 rw	m0 rw
Receive Control URCTL 072h	FE rw-0	undef. rw-0	OE rw-0	undef. rw-0	unused rw-0	unused rw-0	undef. rw-0	undef. rw-0
Transmit Control UTCTL 071h	CKPH rw-0	CKPL rw-0	SSEL1 rw-0	SSEL0 rw-0	unused rw-0	unused rw-0	STC rw-0	TXEPT rw-1
USART Control UCTL 070h	unused rw-0	unused rw-0	unused rw-0	CHAR rw-0	Listen rw-0	SYNC rw-0	MM rw-0	SWRST rw-1

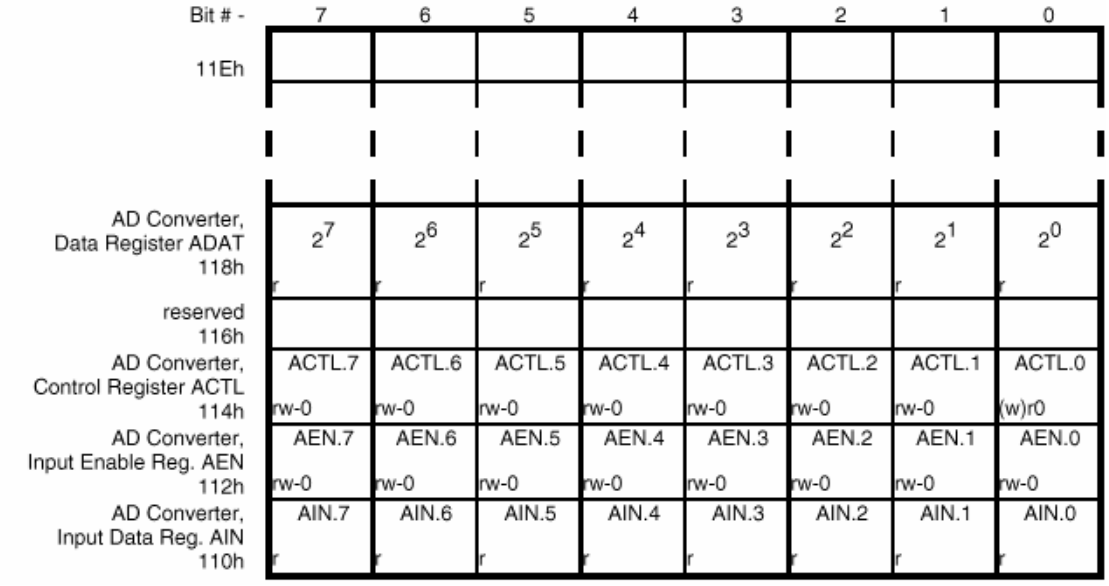
Bit # -	7	6	5	4	3	2	1	0
07Fh								
Transmit Buffer TXBUF 077h	2^7 rw	2^6 rw	2^5 rw	2^4 rw	2^3 rw	2^2 rw	2^1 rw	2^0 rw
Receive Buffer RXBUF 076h	2^7 r	2^6 r	2^5 r	2^4 r	2^3 r	2^2 r	2^1 r	2^0 r
Baud Rate UBR1 075h	2^{15} rw	2^{14} rw	2^{13} rw	2^{12} rw	2^{11} rw	2^{10} rw	2^9 rw	2^8 rw
Baud Rate UBR0 074h	2^7 rw	2^6 rw	2^5 rw	2^4 rw	2^3 rw	2^2 rw	2^1 rw	2^0 rw
Modulation Control UMCTL 073h	m7 rw	m6 rw	m5 rw	m4 rw	m3 rw	m2 rw	m1 rw	m0 rw
Receive Control URCTL 072h	FE rw-0	undef. rw-0	OE rw-0	undef. rw-0	unused rw-0	unused rw-0	undef. rw-0	undef. rw-0
Transmit Control UTCTL 071h	CKPH rw-0	CKPL rw-0	SSEL1 rw-0	SSEL0 rw-0	unused rw-0	unused rw-0	STC rw-0	TXEPT rw-1
USART Control UCTL 070h	unused rw-0	unused rw-0	unused rw-0	CHAR rw-0	Listen rw-0	SYNC rw-0	MM rw-0	SWRST rw-1

A/D 转换器寄存器组，字访问



Bit # -	15	14	13	12	11	10	9	8
11Fh								
AD Converter, Data Register ADAT 118h			R1 *)	R0 *)	2^{11}	2^{10}	2^{9}	2^{8}
	r0	r0	r0	r0	r	r	r	r
reserved 116h								
AD Converter, Control Register ACTL 114h	ACTL.15	ACTL.14	ACTL.13	ACTL.12	ACTL.11	ACTL.10	ACTL.9	ACTL.8
	r0	r0	r0	rw-1	rw-0	rw-0	rw-0	rw-0
AD Converter, Input Enable Reg. AEN 112h								
	r0	r0	r0	r0	r0	r0	r0	r0
AD Converter, Input Data Reg. AIN 110h								
	r0	r0	r0	r0	r0	r0	r0	r0

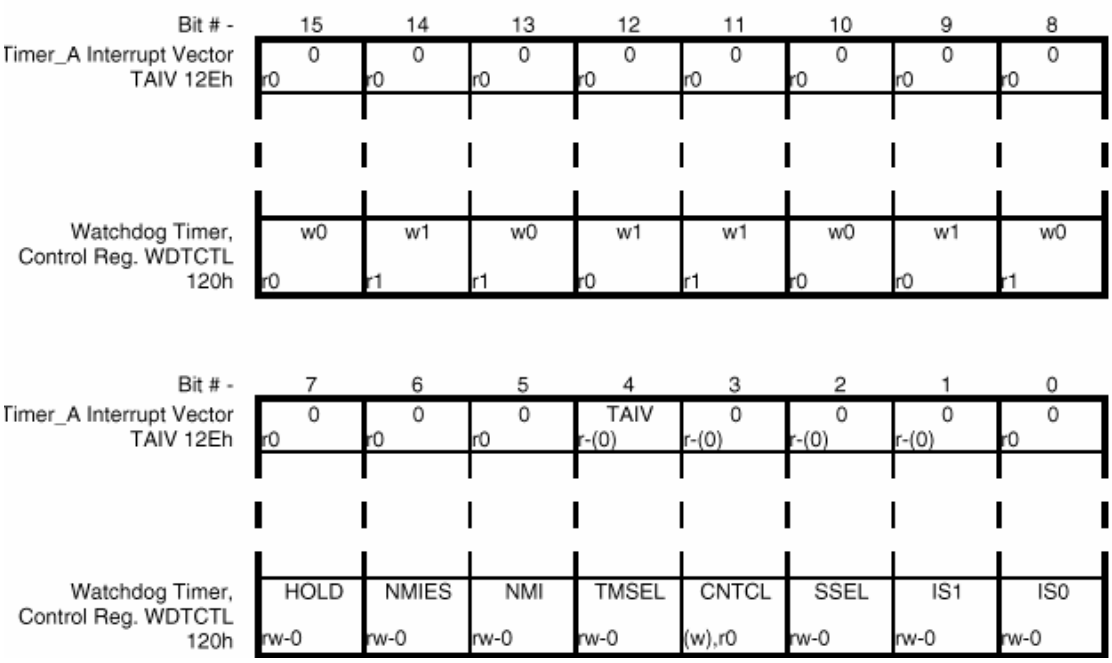
*) The bits ADAT.12 and ADAT.13 are read as 0 when ACTL.11=0 otherwise signals R0 and R1 are read.



Bit # -	7	6	5	4	3	2	1	0
11Eh								
AD Converter, Data Register ADAT 118h	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
	r	r	r	r	r	r	r	r
reserved 116h								
AD Converter, Control Register ACTL 114h	ACTL.7	ACTL.6	ACTL.5	ACTL.4	ACTL.3	ACTL.2	ACTL.1	ACTL.0
	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	(w)r0
AD Converter, Input Enable Reg. AEN 112h	AEN.7	AEN.6	AEN.5	AEN.4	AEN.3	AEN.2	AEN.1	AEN.0
	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
AD Converter, Input Data Reg. AIN 110h	AIN.7	AIN.6	AIN.5	AIN.4	AIN.3	AIN.2	AIN.1	AIN.0
	r	r	r	r	r	r	r	r

*）当 ACTL.11=0，ADAT.12 和 ADAT.13 读出为 0；ACTL.11=1，R0 和 R1 可读。

看门狗/定时器寄存器和 Timer_A 中断向量寄存器组，字访问



乘法器寄存器组，字访问

Bit # -	15	14	13	12	11	10	9	8
Sum Extend, SumExt 013Eh	*) r	*) r	*) r	*) r	*) r	*) r	*) r	*) r
Result High Word ResHi 013Ch	2^{15} rw	2^{14} rw	2^{13} rw	2^{12} rw	2^{11} rw	2^{10} rw	2^{9} rw	2^{8} rw
Result Low Word ResLo 013Ah	2^{15} rw	2^{14} rw	2^{13} rw	2^{12} rw	2^{11} rw	2^{10} rw	2^{9} rw	2^{8} rw
Second Operand OP2 0138h	2^{15} rw	2^{14} rw	2^{13} rw	2^{12} rw	2^{11} rw	2^{10} rw	2^{9} rw	2^{8} rw
0136h								
MPY+ACC MAC 0134h	2^{15} rw	2^{14} rw	2^{13} rw	2^{12} rw	2^{11} rw	2^{10} rw	2^{9} rw	2^{8} rw
Multiply signed MPYS 0132h	2^{15} rw	2^{14} rw	2^{13} rw	2^{12} rw	2^{11} rw	2^{10} rw	2^{9} rw	2^{8} rw
Multiply unsigned MPY 0130h	2^{15} rw	2^{14} rw	2^{13} rw	2^{12} rw	2^{11} rw	2^{10} rw	2^{9} rw	2^{8} rw

Bit # -	7	6	5	4	3	2	1	0
Sum Extend, SumExt 013Eh	*) r	*) r	*) r	*) r	*) r	*) r	*) r	*) r
Result High Word ResHi 013Ch	2^{7} rw	2^{6} rw	2^{5} rw	2^{4} rw	2^{3} rw	2^{2} rw	2^{1} rw	2^{0} rw
Result Low Word ResLo 013Ah	2^{7} rw	2^{6} rw	2^{5} rw	2^{4} rw	2^{3} rw	2^{2} rw	2^{1} rw	2^{0} rw
Second Operand OP2 0138h	2^{7} rw	2^{6} rw	2^{5} rw	2^{4} rw	2^{3} rw	2^{2} rw	2^{1} rw	2^{0} rw
0136h								
MPY+ACC MAC 0134h	2^{7} rw	2^{6} rw	2^{5} rw	2^{4} rw	2^{3} rw	2^{2} rw	2^{1} rw	2^{0} rw
Multiply signed MPYS 0132h	2^{7} rw	2^{6} rw	2^{5} rw	2^{4} rw	2^{3} rw	2^{2} rw	2^{1} rw	2^{0} rw
Multiply unsigned MPY 0130h	2^{7} rw	2^{6} rw	2^{5} rw	2^{4} rw	2^{3} rw	2^{2} rw	2^{1} rw	2^{0} rw

*) 扩展寄存器 SumExt 保存 16x16 位乘积的符号(MPYS)，或者乘加运算的溢出部分(MAC)。SumExt 寄存器为：

- 0FFFFh 当乘法运算结果为负
- 0h 当乘法运算结果为正
- 0h 当乘加运算无溢出
- 1h 当乘加运算有溢出

Timer_A 寄存器组(I)，字访问

Bit # -	15	14	13	12	11	10	9	8
017Eh								
017Ch								
Cap/Com register CCR4 017Ah	2^{15} rw-(0)	2^{14} rw-(0)	2^{13} rw-(0)	2^{12} rw-(0)	2^{11} rw-(0)	2^{10} rw-(0)	2^{9} rw-(0)	2^{8} rw-(0)
Cap/Com register CCR3 0178h	2^{15} rw-(0)	2^{14} rw-(0)	2^{13} rw-(0)	2^{12} rw-(0)	2^{11} rw-(0)	2^{10} rw-(0)	2^{9} rw-(0)	2^{8} rw-(0)
Cap/Com register CCR2 0176h	2^{15} rw-(0)	2^{14} rw-(0)	2^{13} rw-(0)	2^{12} rw-(0)	2^{11} rw-(0)	2^{10} rw-(0)	2^{9} rw-(0)	2^{8} rw-(0)
Cap/Com register CCR1 0174h	2^{15} rw-(0)	2^{14} rw-(0)	2^{13} rw-(0)	2^{12} rw-(0)	2^{11} rw-(0)	2^{10} rw-(0)	2^{9} rw-(0)	2^{8} rw-(0)
Cap/Com register CCR0 0172h	2^{15} rw-(0)	2^{14} rw-(0)	2^{13} rw-(0)	2^{12} rw-(0)	2^{11} rw-(0)	2^{10} rw-(0)	2^{9} rw-(0)	2^{8} rw-(0)
Timer_A register TAR 0170h	2^{15} rw-(0)	2^{14} rw-(0)	2^{13} rw-(0)	2^{12} rw-(0)	2^{11} rw-(0)	2^{10} rw-(0)	2^{9} rw-(0)	2^{8} rw-(0)
016Eh								
016Ch								
Cap/Com Control CCTL4, 0164h	CM41 rw-(0)	CM40 rw-(0)	CCIS41 rw-(0)	CCIS40 rw-(0)	SCS4 rw-(0)	SCCI4 rw-(0)	unused r0	CAP4 rw-(0)
Cap/Com Control CCTL3, 0164h	CM31 rw-(0)	CM30 rw-(0)	CCIS31 rw-(0)	CCIS30 rw-(0)	SCS3 rw-(0)	SCCI3 rw-(0)	unused r0	CAP3 rw-(0)
Cap/Com Control CCTL2, 0164h	CM21 rw-(0)	CM20 rw-(0)	CCIS21 rw-(0)	CCIS20 rw-(0)	SCS2 rw-(0)	SCCI2 rw-(0)	unused r0	CAP2 rw-(0)
Cap/Com Control CCTL1, 0164h	CM11 rw-(0)	CM10 rw-(0)	CCIS11 rw-(0)	CCIS10 rw-(0)	SCS1 rw-(0)	SCCI1 rw-(0)	unused r0	CAP1 rw-(0)
Cap/Com Control CCTL0, 0162h	CM01 rw-(0)	CM00 rw-(0)	CCIS01 rw-(0)	CCIS00 rw-(0)	SCS0 rw-(0)	SCCI0 rw-(0)	unused r0	CAP0 rw-(0)
Timer_A Control TACTL 0160h	unused rw-(0)	unused rw-(0)	unused rw-(0)	unused rw-(0)	unused rw-(0)	SSEL2 rw-(0)	SSEL1 rw-(0)	SSEL0 rw-(0)

Timer_A 寄存器组(II)，字访问

Bit # -	7	6	5	4	3	2	1	0
017Eh								
017Ch								
Cap/Com register CCR4 017Ah	2^7 rw-(0)	2^6 rw-(0)	2^5 rw-(0)	2^4 rw-(0)	2^3 rw-(0)	2^2 rw-(0)	2^1 rw-(0)	2^0 rw-(0)
Cap/Com register CCR3 0178h	2^7 rw-(0)	2^6 rw-(0)	2^5 rw-(0)	2^4 rw-(0)	2^3 rw-(0)	2^2 rw-(0)	2^1 rw-(0)	2^0 rw-(0)
Cap/Com register CCR2 0176h	2^7 rw-(0)	2^6 rw-(0)	2^5 rw-(0)	2^4 rw-(0)	2^3 rw-(0)	2^2 rw-(0)	2^1 rw-(0)	2^0 rw-(0)
Cap/Com register CCR1 0174h	2^7 rw-(0)	2^6 rw-(0)	2^5 rw-(0)	2^4 rw-(0)	2^3 rw-(0)	2^2 rw-(0)	2^1 rw-(0)	2^0 rw-(0)
Cap/Com register CCR0 0172h	2^7 rw-(0)	2^6 rw-(0)	2^5 rw-(0)	2^4 rw-(0)	2^3 rw-(0)	2^2 rw-(0)	2^1 rw-(0)	2^0 rw-(0)
Timer_A register TAR 0170h	2^7 rw-(0)	2^6 rw-(0)	2^5 rw-(0)	2^4 rw-(0)	2^3 rw-(0)	2^2 rw-(0)	2^1 rw-(0)	2^0 rw-(0)
016Eh								
016Ch								
Cap/Com Control CCTL4, 016Ah	OutMod42 rw-(0)	OutMod41 rw-(0)	OutMod40 rw-(0)	CCIE4 rw-(0)	CCI4 r	OUT4 rw-(0)	COV4 rw-(0)	CCIFG4 rw-(0)
Cap/Com Control CCTL3, 0168h	OutMod32 rw-(0)	OutMod31 rw-(0)	OutMod30 rw-(0)	CCIE3 rw-(0)	CCI3 r	OUT3 rw-(0)	COV3 rw-(0)	CCIFG3 rw-(0)
Cap/Com Control CCTL2, 0166h	OutMod22 rw-(0)	OutMod21 rw-(0)	OutMod20 rw-(0)	CCIE2 rw-(0)	CCI2 r	OUT2 rw-(0)	COV2 rw-(0)	CCIFG2 rw-(0)
Cap/Com Control CCTL1, 0164h	OutMod12 rw-(0)	OutMod11 rw-(0)	OutMod10 rw-(0)	CCIE1 rw-(0)	CCI1 r	OUT1 rw-(0)	COV1 rw-(0)	CCIFG1 rw-(0)
Cap/Com Control CCTL0, 0162h	OutMod02 rw-(0)	OutMod01 rw-(0)	OutMod00 rw-(0)	CCIE0 rw-(0)	CCI0 r	OUT0 rw-(0)	COV0 rw-(0)	CCIFG0 rw-(0)
Timer_A Control TACTL 0160h	ID1 rw-(0)	ID0 rw-(0)	MC1 rw-(0)	MC0 rw-(0)	unused rw-(0)	CLR rw-(0)	TAIE rw-(0)	TAIFG rw-(0)

附录 B. 指令组说明

（略）

附录 C. EPROM 编程

本附录描述 MSP430 的 EPROM 模块。EPROM 为紫外光擦除和电编程。含有 EPROM 模块的 MSP430 器件提供一个多次编程的窗口封装或一次性编程的 OTP 封装。

C.1 EPROM 操作

CPU 可从 EPROM 读取数据和指令。当 TDI/VPP 引脚加上编程电压后，CPU 也可向 EPROM 模块写入。读 EPROM 和读其它内部外围模块一样。编程和读取可以在字节或字边界进行。

擦除

编程前应先擦除整个 EPROM。只要让透明窗受到紫外光的照射就可以了。

注意：环境光对 EPROM 的影响
普通环境光包含擦除 EPROM 所需波长。对有窗器件编程时，窗口必须用不透光的封条封住。紫外光照射 EPROM 时，如果 EEPROM 模块也在同一芯片上也会被擦除。EEPROM 中的有用数据应该重新编程。

数据可以通过内部集成的“JTAG”特性串行编程，或通过应用软件的一部分编程。“JTAG”实现了内部的安全机制。一旦“安全熔丝”激活，由于“JTAG”的作用便不能访问器件。“JTAG”永久性地切换成旁路模式。

编程

编程时必须在 TDF/VPP 引脚外加电压，为编程提供所需的电压和电流。在器件手册的电特性中标明了编程所需的最短时间。

加了外部电压后，由 EPROM 控制寄存器 EPCTL 控制 EPROM 的编程。EPROM 经擦除后各位为“1”，编程后对应的位将变为“0”。

EPROM 模块的编程操作既可以按字节或字，也可按不同长度的数据块，甚至是整个模块。所有的位在编程前必需先擦除，使它们最终都为“1”。编程操作可对单一器件进行，甚至可在器件安装在系统时进行。提供的外部编程电压应该在器件手册所要求的范围。“JTAG”引脚的电平在器件手册中定义，通常是 CMOS 电平。

MSP430 字格式编程

	.....
xxxAh	DEF0
xxx8h	9ABC
xxx6h	5678
xxx4h	1234
	.....

MSP430 字节格式编程

	.....
xxxBh	DE
xxxAh	F0
xxx9h	9A
xxx8h	BC
xxx7h	56
xxx6h	78
xxx5h	12
xxx4h	34
	.....

EPROM 控制寄存器 EPCTL

EPCTL（054h）

7							0
---	---	---	---	---	---	VPPS	EXE
r-0	r-0	r-0	r-0	r-0	r-0	rw-0	rw-0

位 0：执行位 EXE 控制初始化和终止对 EPROM 模块的编程。在 EXE 位置位前必须在 TDI/VPP 引脚加外部电压。定时条件在器件手册中说明。

位 1：VPPS 位置位，使外部编程电压与 EPROM 模块相连。VPPS 位应在 EXE 位置位前置位。它可与 EXE 位一起复位。在编程操作时 VPPS 位不能复位。

EPROM 保护

当“安全熔丝”激活，通过串行测试和编程接口“JTAG”对 EPROM 的访问被禁止。激活安全熔丝是通过串行指令移入“JTAG”实现的。激活安全熔丝是不可逆的，任何对内部系统的访问都被中断。按标准 IEEE1149.1 说明的旁路功能成为有效。

C.2 快速编程算法

快速编程模式通常用于向 EPROM 写入数据。已编程为逻辑“0”只能通过紫外光擦除。

快速编程使用两种类型脉冲：初期脉冲和最终脉冲。典型的初期脉冲宽度为 100us（见最近的器件手册）。每次基本脉冲后对编程数据作校验。如果连续失败 25 次，此次编程操作失效。如果读取了正确的数据，使用最终脉冲编程，它是已用初期脉冲数的 3 倍。

C.3 通过串行数据链路应用“JTAG”特性的EPROM模块编程

“JTAG”的硬件连接是通过4个引脚外加GND和VCC实现的，即TMS、TCK、TDI（VPP）、TDO/TDI。

C.4 通过微控制器软件实现对 EPROM 模块编程

对 EPROM 模块编程所要求的硬件很简单：TDI/VPP 引脚加编程电压，运行适当的软件即可。控制 EPROM 编程的软件不能在要编程的 EPROM 模块上运行。因为不可能同时从 EPROM 读指令和往 EPROM 写数据。软件需要在另一个存储器上运行，如 ROM、RAM 或其它的 EPROM 模块。

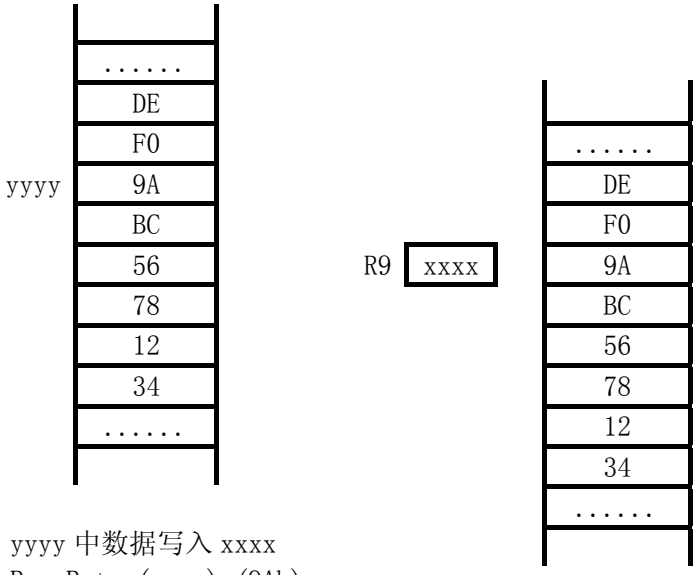
注*： TMS 和 TCK 引脚有内部上拉电阻

注：** MSP430 的 ROM 型在 TDI/VPP 引脚有上拉电阻，而 MSP430Pxxx 和 MSP430Exxx 型没有。为防止输入引脚的浮空，要外加一个上拉或下拉电阻。

注*：** 为防止 TDO/TDI 在用 TDI 功能时输入引脚的浮空，要外加一个上拉或下拉电阻。

使用控制器的软件对 EPROM 编程的例子

以下软件实例是用快速编程算法向 EPROM 写一个字节。代码写入与位置无关，但是要在使用前先行加载（例如装入 RAM）。编程软件运行在 RAM 中，避免了与写入 EPROM 的冲突。要写的数据（字节）放在 RAM 中的“BurnByte”地址，EPROM 的目标地址由 SET 指令定义的寄存器指针确定。定时调整为 1us 一个周期。如果选择其它的循环时间/处理机频率，应该按照不同的操作条件调整软件。



例：yyyy 中数据写入 xxxx
BurnByte=(yyyy)=(9Ah)
R9=xxxx

当有数据往 EPROM 写入时，不能同时执行程序代码。在以下实例中，用一个子程序把程序代码移到另一个存储器中，如装入 RAM。

```
;-------------------------------------------------------------------------
; Definitions used in Subroutine :
; Move programming code sequence into RAM (load_burn_routine)
; Burn a byte into the EPROM area (RELOC_Burn_EPROM)
;-------------------------------------------------------------------------
EPCTL       .set    054h            ; EPROM Control Register
VPPS        .set    2               ; Program Voltage bit
EXE         .set    1               ; Execution bit
BurnByte    .set    0220h           ; address of data to be written
Burn_orig   .set    0222h           ; Start address of burn program in the RAM
loops       .set    25
r_timer     .set    r8              ; 1us = 1 cycle
pointer     .set    r9              ; pointer to the EPROM address
; r9 is saved in the main routine before subroutine call is executed
r_count     .set    r10
lp          .set    3               ; dec r_timer : 1 cycle : loop_t100
                                    ; jnz : 2 cycles : loop_t100
ov          .set    2               ; mov #(100-ov)/lp,r_timer : 2 cycles
; Load EPROM programming sequence to another location e.g. RAM, Subroutine
;--- Burn subroutine: position independent code!
RAM_Burn_EPROM
            .set    Burn_orig
load_burn_routine
            push    r9
            push    r10
            mov     #Burn_EPROM,R9          ; load pointer source
            mov     #RAM_Burn_EPROM,R10     ; load pointer dest.
load_burn1  mov     @R9,0(R10)              ; move a word
            incd    R10                     ; dest. pointer + 2
            incd    R9                      ; source pointer + 2
            cmp     #Burn_end,R9            ; compare to end_of_table
            jne     load_burn1
            pop     r9
            pop     r10
            ret
; Program one byte into EPROM, Subroutine
Burn_EPROM  dint                            ; ensure correct burn
timing      mov.b   #VPPS,&EPCTL            ; VPPS on
            push    r_timer                 ; save registers
            push    r_count                 ; programming subroutine
            mov     #loops,r_count          ; 2 cycles = 2 us
Repeat_Burn mov.b   &BurnByte,0(pointer)    ; write to data to EPROM
                                            ; 6 cycles = 6 us
            bis.b   #EXE,&EPCTL             ; EXE on
```

```
                                        ; 4 cycles = 4 us
                                        ; total cycles VPPon to EXE
                                        ; 12 cycles = 12 us (min.)
            mov     #(100-ov)/lp,r_timer    ;:programming pulse of
100us
wait_100                                ;:starts, actual time
102us       dec     r_timer             ;:
            jnz     wait_100            ;:
            bic.b   #EXE,&EPCTL         ;:EXE / prog. puls off
            mov     #4,r_timer          ;:wait min. 10 us
wait_10                                 ;:before verifying
            dec     r_timer             ;:programmed EPROM
            jnz     wait_10             ;:location, actual 13+ us
            cmp.b   &BurnByte,0(pointer)    ; verify data = burned data
            jne     Burn_EPROM_bad          ; data ≠ burned data > jump
; Continue here when data correctly burned into EPROM location
            mov.b   &BurnByte,0(pointer)    ; write to EPROM again
            bis.b   #EXE,&EPCTL             ; EXE on
            add     #(0ffffh-loop),r_count  ; Number of loops for
                                            ; successful programming
final_puls  mov     #(300-ov)/lp,r_timer    ;:programming pulse of
wait_300                                    ;:3*100us*N starts
            dec     r_timer             ;:
            jnz     wait_300            ;:
            inc     r_count             ;:
            jn      final_puls          ;:
            clr.b   &EPCTL              ;:EXE off / VPPS off
            jmp     Burn_EPROM_end
Burn_EPROM_bad
            dec     r_count             ; not ok : decrement loop counter
            jnz     Repeat_Burn         ; loop not ended : do another trial
            inv.b   &BurnByte           ; return the inverted data to flag
                                        ; failing the programming attempt
                                        ; the EPROM address is unchanged
Burn_EPROM_end
            pop     r_timer
            pop     r_count
            eint
            ret
Burn_end
```