Nuvoton 1T- 8051内核微控制器

N76E885

规格书

目　　录

1. 概述

N76E885 为带有 FLASH的增强型8位51内核微控制器 (1T工作模式)，指令集与标准 8051完全兼容并具备更高效能。

N76E885内嵌18K的FLASH存储区，通常称作APROM，用于存放用户程序代码。该存储区支持IAP烧写功能，即可通过片内固件更新程序代码，IAP功能同时提供用户可自行配置加密程序区或数据存储区，也可通过IAP指令或MOVC指令读取任一区域内数据。另外N76E885还配置额外具有一存储区称作LDROM，该区域可存放用于执行ISP的引导代码（boot code），LDROM区域从18K的APROM区域内分割出来，通过配置位CONFIG配置大小，最多可配置到 4K 字节。整个18K FLASH区域还支持ICP编程方式，即通过片外I/O由总线方式烧写片内FLASH数据，该方式不占用片内代码空间，且可通过加密位对FLASH完全加密，保障程序代码无法读出。

N76E885提供丰富的特性功能模块包括， 256 字节 片内RAM, 256字节片外RAM (XRAM。最多可达26个标准管脚。2组标准16位定时器/计数器：定时器0及1，一组带有3路管脚输出捕捉模式的的16位定时器：定时器2，一组看门狗定时器 (WDT)，一组自唤醒定时器 (WKT)，一组带自动重装载功能，可用于产生标准波特率的定时器：定时器3。相对应的2组标准串行口(UART)，这两组串行口具有帧错误侦测及自动地址识别功能。一组SPI，一组 I^2C，四对增强型PWM输出，8路10位ADC。上述功能对应产生18个中断源，具有4级中断优先级配置。

N76E885 支持五组时钟源输入，所有时钟源支持软件切换立即生效(on-the-fly)功能。 5组时钟源包括2MHz ~ 25 MHz 高速外部晶振, 32.768 kHz 低速外部晶振，外部时钟输入，10 kHz内部低速振荡器及22.118 MHz内部高精度振荡器，该振荡器精度达到室温条件下±1%。N76E885 提供额外的电源监控管理模块，包含上电复位模块POR，以及8级低电压侦测模块用于保障芯片在上电及掉电时系统稳定工作。

N76E885 可运行在两种低功耗模式——空闲模式及掉电模式下。空闲模式时，芯片主时钟关闭，但部分功能模块仍然运行。掉电模式下芯片全部时钟关闭确保芯片功耗达到最低。在正常工作模式下，也可选择主时钟除频工作，用以确保低功耗的运行效果。

高效能、丰富的功能模块及配置，N76E885可灵活适用于各种应用场合，甚至是马达控制等高端需求控制系统。

2. 特性

- **CPU:**
  - 全静态8位 1T 8051内核 CMOS 微控制器.
  - 指令集全兼容 MCS-51.
  - 4级优先级中断配置
  - 双数据指针（DPTRs）
- **工作条件:**
  - 宽电压工作范围 2.4V 至 5.5V.
  - 宽工作频率最高至 25MHz.
  - 工业级工作温度 -40℃ 至 +105℃.
- **存储器:**
  - 最高至 18K 字节 APROM 用户程序代码区
  - 可配置4K/3K/2K/1K/0K 字节 LDROM 引导代码区，用户可灵活配置用途
  - 所有FLASH 区域分隔为128字节一页
  - 内建IAP编程功能
  - FLASH存储区正常100,000次改写次数，超过10年数据保存时间
  - 代码加密功能.
  - 256 字节片内直接存取RAM.
  - 额外256字节片内间接存取RAM (XRAM) 通过MOVX 指令读写.
- **时钟源**
  - 22.118 MHz 高速内部振荡器，电源5.0V条件下±1%精度等级。全工作条件范围±2%精度等级。
  - 10 kHz 低速内部振荡器
  - 支持2 MHz 至 25 MHz 高速外部晶振输入
  - 支持32.768 kHz 低速外部晶振输入
  - 支持外部时钟输入
  - 支持系统时钟即时软件切换(On-the-fly)功能
  - 支持软件配置时钟除频最高至1/512.

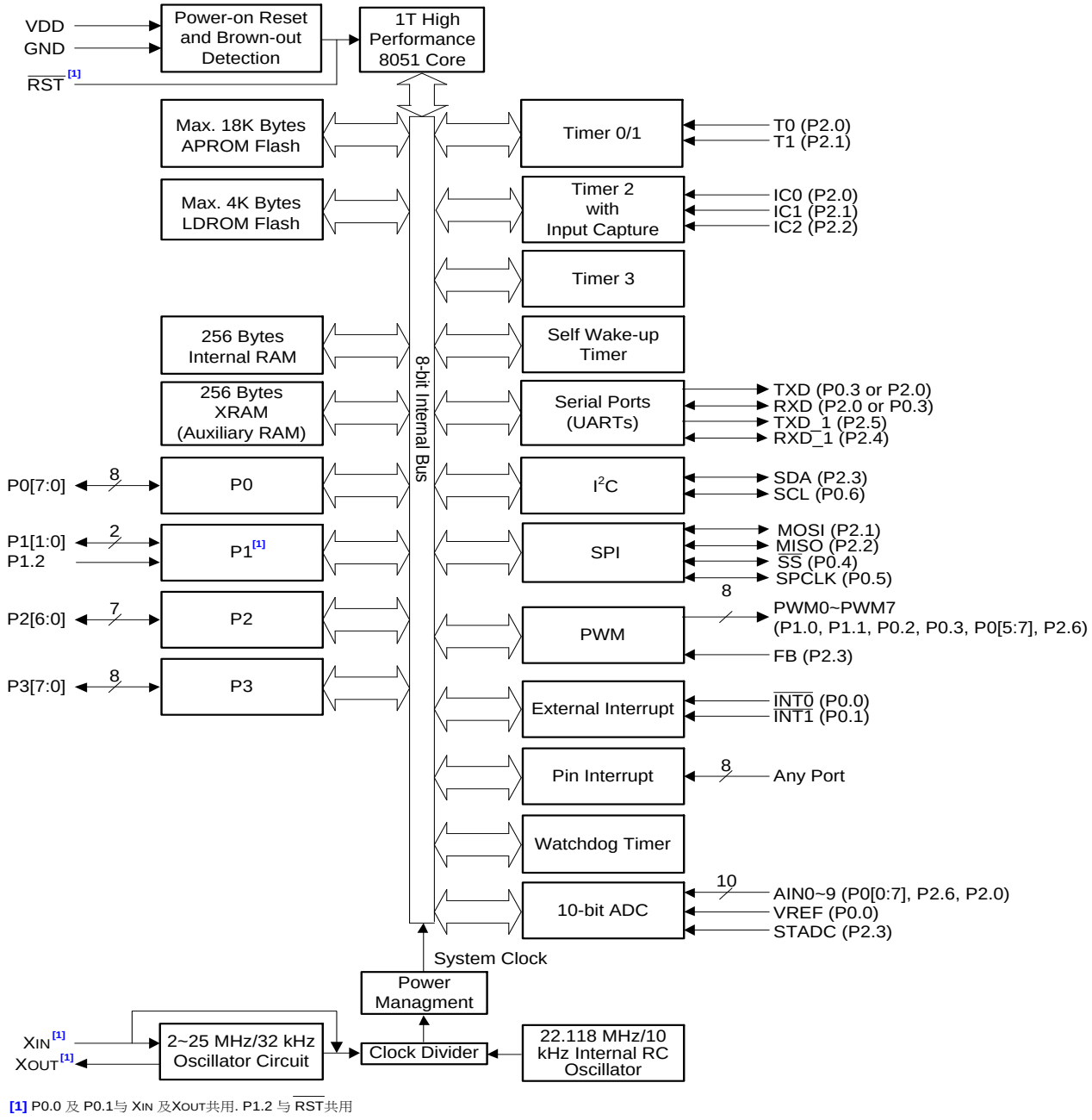
- **功能:**
  - 最多25个标准通用管脚，1个仅输入管脚。所有输出管脚可通过软配置两种输出斜率(slew rate)，5个管脚可提供标准/高灌入电流功能.
  - 标准外部中断脚 $\overline{\text{INT0}}$及$\overline{\text{INT1}}$
  - 两组16位定时器/计数器0及1，与标准8051兼容
  - 一组16位定时器2带有3路输入捕获功能
  - 一组16位自动重装载功能定时器3，可用于配置串行口波特
  - 一组看门狗 (WDT)，由内部 10kHz 独立时钟作为时钟源
  - 一组自唤醒功能定时器 (WKT)
  - 两组全双工串口，带有帧错误侦测及自动地址辨识功能。最高波特率可设置到 781.25kbps，串口0 (UART0) 的 TXD 及 RXD 脚可通过软件更换管脚位置
  - 一组SPI总线，主机及从机模式最高传输速率皆可达到 6.25Mbps
  - 一组I2C总线，主机及从机模式最高传输速率皆可达到 400kbps.
  - 四对，8通道脉宽调制器(PWM)，12位精度，4种模式并带有故障刹车(Fault Brake)功能
  - 8通道管脚中断功能，可通过软件配置边沿或电平触发
  - 一组10位 ADC 带外部参考电压V_{REF}输入功能，最高 300ksps转换速率，硬件启动及转换结果匹配控制马达PWM功能
- 电源管理模块
  - 两种省电模式: 空闲模式及掉电模式
- 电源监控:
  - 欠压检测 (BOD)用于侦测系统供电低电压，8级电压选择，可配置位中断或复位响应。
  - 上电复位 (POR) 电压管理
- 强效ESD 及EFT能力
- 开发工具：
  - 基于 KEIL™ 开发环境的新唐片上调试(OCD仿真)
  - 新唐在电路编程 (ICP编程)
  - 通过串口烧写芯片的新唐在系统编程 (ISP编程)

- 编号及封装:

编号	APROM	LDROM	封装
N76E885AT28	18K 字节 与LDROM共享	最高至 4K 字节	TSSOP-28
N76E885AT20			TSSOP-20

3. 功能方块图

图 3-1 显示 N76E885 所有功能模块及外接端口配置



[1] P0.0 及 P0.1与 XIN 及XOUT共用. P1.2 与 RST共用

图 3-1. 功能方块图

4. 管脚配置

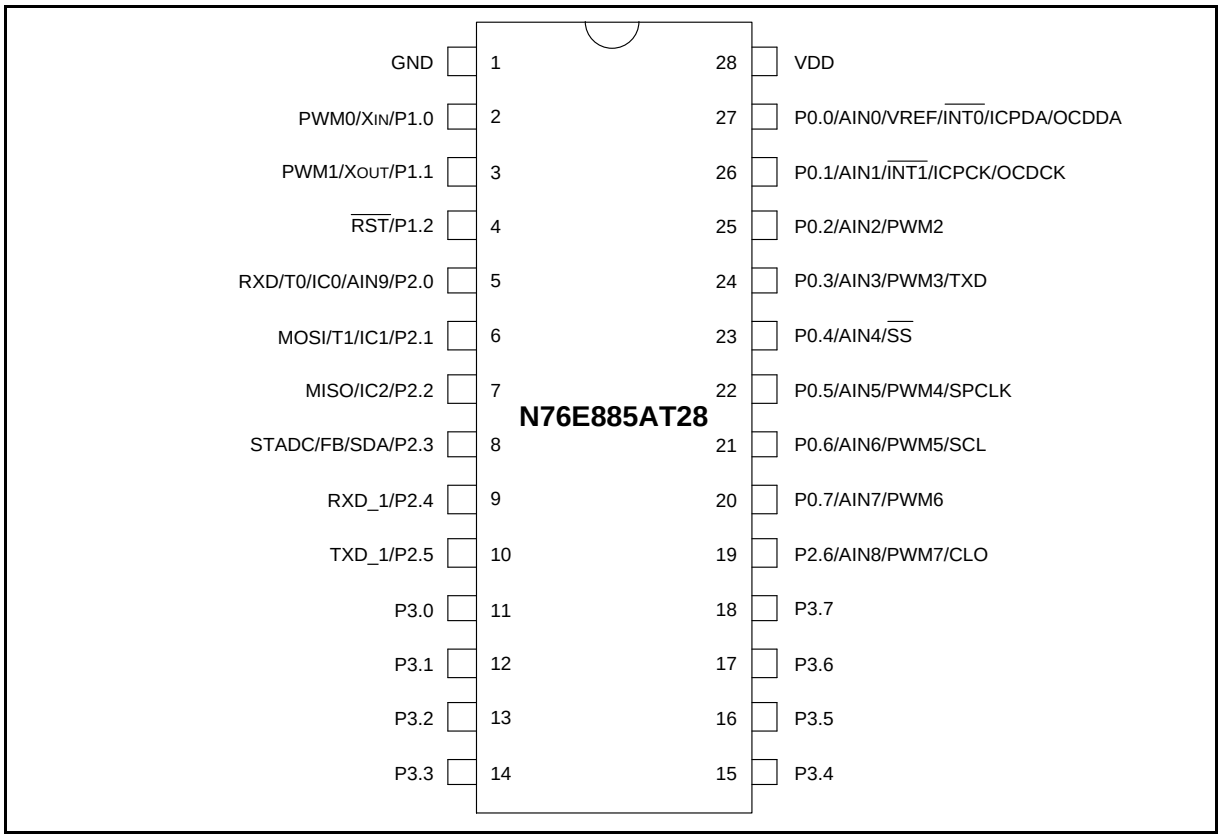


图 4-1. TSSOP-28 封装管脚配置

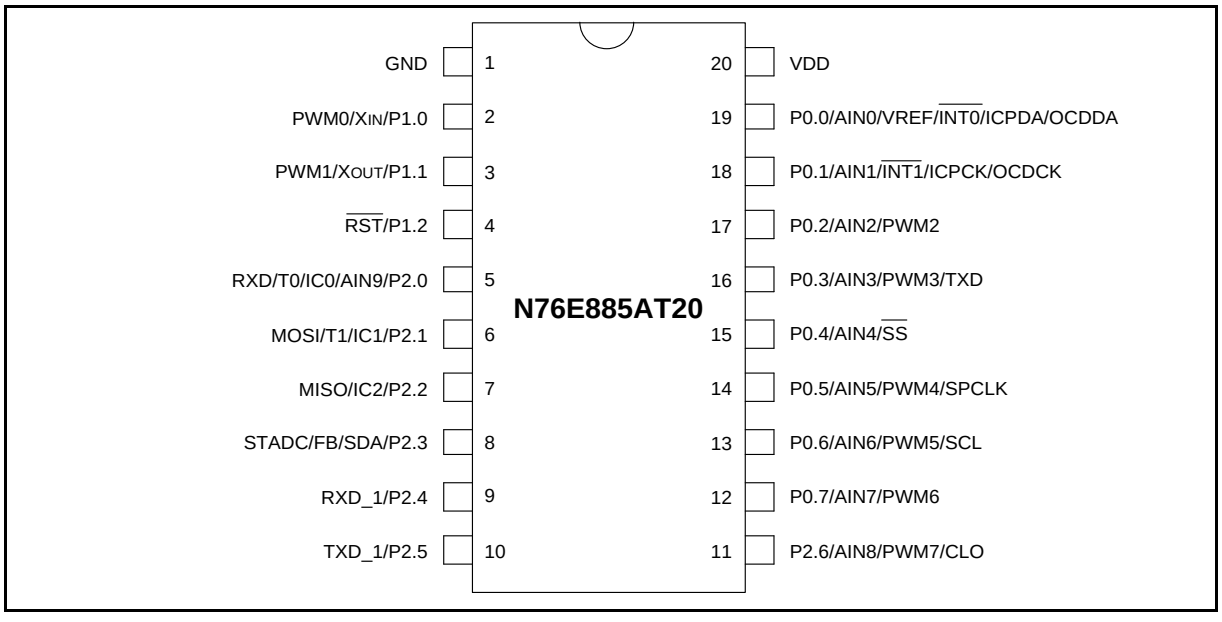


图 4-2. TSSOP-20 封装管脚配置

表 4–1. 管脚描述

管脚编号		符号	功能描述 [1]
TSSOP20	TSSOP28		
20	28	VDD	**电源:** 电源正端.
1	1	GND	**电源地:** 电源负端
P0[7:0]			**端口0:** 端口 0可位寻址，总共8个输入输出管脚。复位后，所有管脚为输入高阻模式。
19	27	P0.0/AIN0/VREF/$\overline{INT0}$/ICPDA/ OCDDA	**P0.0:** 端口0管脚0.
			AIN0: ADC输入通道 0.
			VREF: ADC V_{REF} 参考电压输入
			$\overline{INT0}$: 外部中断0输入
			ICPDA: ICP 编程数据输入输出脚
			OCDDA: OCD仿真数据输入输出脚
18	26	P0.1/AIN1/$\overline{INT1}$/ICPCK/ OCDCK	**P0.1:** 端口 0 管脚 1.
			AIN1: ADC 输入通道 1.
			$\overline{INT1}$: 外部中断1输入
			ICPCK: ICP编程时钟输入脚.
			OCDCK: OCD仿真时钟输入脚
17	25	P0.2/AIN2/PWM2	**P0.2:** 端口0管脚2..
			AIN2: ADC输入通道 2.
			PWM2: PWM 输出通道2
16	24	P0.3/AIN3/PWM3/TXD	**P0.3:** 端口 0 管脚 3.
			AIN3: ADC 输入通道3.
			PWM3: PWM 输出通道3
			TXD[2]**:** 串口0数据发送脚
15	23	P0.4/AIN4/$\overline{SS}$	**P0.4:** 端口0管脚4.
			AIN4: ADC输入通道4.
			$\overline{SS}$: SPI 从机选择输入脚
14	22	P0.5/AIN5/PWM4/SPCLK	**P0.5:** 端口0管脚5.
			AIN5: ADC 输入通道 5.
			PWM4: PWM 输出通道 4.
			SPCLK: SPI 时钟脚.
13	21	P0.6/AIN6/PWM5/SCL	**P0.6:** 端口 0 管脚 6.
			AIN6: ADC输入通道6.
			PWM5: PWM输出通道5
			SCL: I^2C 时钟脚.
12	20	P0.7/AIN7/PWM6	**P0.7:** 端口 0 管脚 7.
			AIN7: ADC输入通道7
			PWM6: PWM输出通道6.
P1[2:0]			**端口1:** 端口 1可位寻址，总共3个输入输出管脚。P1.2 始终为输入脚。
2	2	P1.0/X_{IN}/PWM0	**P1.0:** 端口1管脚0。使用内部晶振时可用
			X_{IN}: 当系统时钟采用 HXT 或 LXT 时，X_{IN} 为外部晶振输入脚。如果使用外部时钟输出模式， X_{IN} 为外部时钟输入脚。
			PWM0: PWM 输出通道 0.
3	3	P1.1/X_{OUT}/PWM1	**P1.1:** 端口1管脚1。使用内部晶振时可用

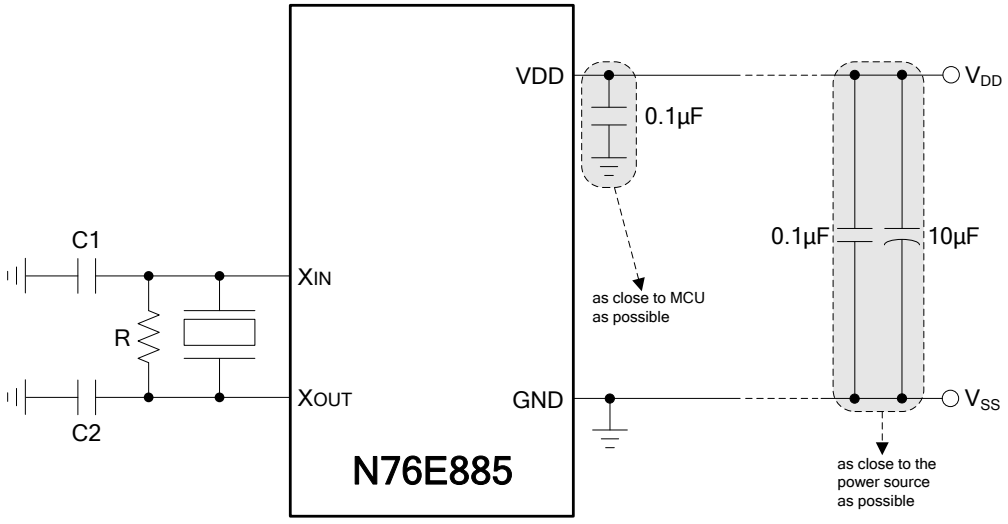
表 4–1. 管脚描述

管脚编号		符号	功能描述 [1]
TSSOP20	TSSOP28		
			XOUT: 当系统时钟采用 HXT 或 LXT 时，XOUT 为外部晶振输出脚。输出与XIN 相反的信号。
			PWM1: PWM 输出通道 1.
4	4	P1.2/$\overline{\text{RST}}$	**P1.2:** 端口1管脚2. RPD (CONFIG0.2) 配置为0
			$\overline{\text{RST}}$: $\overline{\text{RST}}$ 复位脚为施密特触发输入，用以外部复位信号复位芯片。 $\overline{\text{RST}}$ 内部带上拉电阻，外部只需接下拉电容，即可稳定工作。
P2[6:0]			**端口2:** 端口2可位寻址，总共7个输入输出管脚。复位状态下，所有管脚为输入高阻模式。
5	5	P2.0/AIN9/IC0/T0/RXD	**P2.0:** 端口2管脚0
			AIN9: ADC输入通道9
			IC0: 定时器输入捕获通道0
			T0: 定时器/计数器0,外部计数输入脚
			RXD[2]**:** 串口0数据接收脚
6	6	P2.1/IC1/T1/MOSI	**P2.1:** 端口2管脚1.
			IC1: 定时器输入捕获通道1
			T1: 定时器/计数器1,外部计数输入脚
			MOSI: SPI 主机输出从机输入脚
7	7	P2.2/IC2/MISO	**P2.2:** 端口2管脚2.
			IC2: 定时器输入捕获通道2
			MISO: SPI 主机输入从机输出脚
8	8	P2.3/SDA/FB/STADC	**P2.3:** 端口2管脚3.
			SDA: I^2C 数据脚.
			FB: 故障刹车输入脚.
			STADC: 外部启动ADC触发脚
9	9	P2.4/RXD_1	**P2.4:** 端口 2 管脚 4.
			RXD_1: 串口1数据输入脚
10	10	P2.5/TXD_1	**P2.5:** 端口2管脚5
			TXD_1: 串口1数据发送脚
11	19	P2.6/AIN8/PWM7/CLO	**P2.6:** 端口2管脚6
			AIN8: ADC输入通道8
			PWM7: PWM 输出通道7.
			CLO: 系统时钟输出脚.

表 4–1. 管脚描述

管脚编号		符号	功能描述 [1]
TSSOP20	TSSOP28		
P3[7:0]			**PORT3:** 端口 0可位寻址，总共8个输入输出管脚。复位后，所有管脚为输入高阻模式。
-	11	P3.0	**P3.0:** 端口 3 管脚 0.
-	12	P3.1	**P3.1:** 端口 3 管脚 1.
-	13	P3.2	**P3.2:** 端口 3 管脚 2.
-	14	P3.3	**P3.3:** 端口 3 管脚 3.
-	15	P3.4	**P3.4:** 端口 3 管脚 4.
-	16	P3.5	**P3.5:** 端口 3 管脚 5..
-	17	P3.6	**P3.6:** 端口 3 管脚 6.
-	18	P3.7	**P3.7:** 端口 3 管脚 7.

[1] 所有管脚都可以配置为外部中断输入脚，该功能未列入管脚描述列表，详见章节 17. “管脚中断”.
[2] 串口0的 TXD 及 RXD 可通过寄存器UART0PX (AUXR1.2).配置位置



晶振频率	R	C1	C2
32.768 kHz	不需要	根据晶振规格书配置	
2 MHz 至 25 MHz	不需要		

图 4-3. 外部晶振线路图

5. 内存架构

标准的基于80C51微控制器将内存分成两个不同的部分，编程内存和数据内存。编程内存用来存储指令代码。而数据内存用来存储编程执行过程中的数据或变量。

数据内存占用的地址空间独立于编程内存。在N76E885中，有256字节内部RAM。对于需要更多内部RAM的许多应用，N76E885提供另外片上256字节RAM，叫做XRAM，通过MOVX指令访问。

整个嵌入的FLASH闪存，作为编程内存的功能，被分成三块。应用ROM（APROM）正常情况下为用户代码，加载ROM（LDROM）通常为启动代码，CONFIG字节用于硬件初始化。事实上，APROM 和LDROM功能上相似，但是大小不一样，每一块由一页一页组成，每页大小是128字节。FLASH控制单元支持擦除、编程、和读模式。外部烧写器通过指定的I/O口烧写。在应用编程（IAP）或在系统编程（ISP）都可以执行这些模式。

5.1 程序内存

程序内存存储编程代码用于执行，如图 5–1.所示。在任何复位之后CPU从地址0000H开始执行 。

用于服务中断，中断服务位置（叫做中断向量）应该位于编程内存。每一个中断被分配一个固定的编程内存地址。中断引起CPU跳到中断服务子程序（ISR）开始执行的地方。例如外部中断0被分配到地址0003H。如果外部中断0打算使用，它的服务子程序应该从地址0003H开始。如果中断不使用，该地址可以作为通用的编程内存。

中断服务位置间隔为八个字节：0003H用于外部中断0，000BH用于定时器0，0013H用于外部中断1，001BH用于定时器1 等。如果一个中断子程序足够短，可以完整地放在这8个字节间隔中。而长的中断服务程序，如果其他的中断有使用，需要用JMP指令跳过后面的中断地址。

N76E885提供两个内部编程内存块APROM和LDROM。虽然他们都和标准8051编程内存一样，但是根据他们ROM的大小不一样，扮演着不同的角色。N76E885的APROM可以最大到18K字节。用户代码通常放在这里面。CPU从这里获取指令来执行。MOVC指令也可以从这个区域读取。

另外单独的编程块叫做LDROM，它的通常功能是存储启动代码用于ISP。它可以更新APROM空间和CONFIG字节。APROM中的代码也可以重新编程LDROM。对于APROM和LDROM的ISP的细节和配置位，请看章节 22.4 “在系统编程(ISP)”。 注意APROM 和 LDROM 是硬件独立模块，因此如果CPU从LDROM启动，CPU会自动重映射PC指针到0000H到LDROM开始的地址。因此CPU认为LDROM是单独的编程内存且所有中断向量独立于APROM。

CONFIG1

7	6	5	4	3	2	1	0
-	-	-	-	-	LDSIZE[2:0]		
-	-	-	-	-	读/写		

出厂默认值: 1111 1111b

位	名称	描述
2:0	LDSIZE[2:0]	**LDROM 容量选择** 111 = 无 LDROM. APROM 为 18K 字节. 110 = LDROM 为 1K 字节. APROM 为 17K 字节. 101 = LDROM 为 2K 字节. APROM 为16K 字节. 100 = LDROM 为 3K 字节. APROM 为15K 字节. 0xx = LDROM is 4K 字节. APROM 为14K 字节.

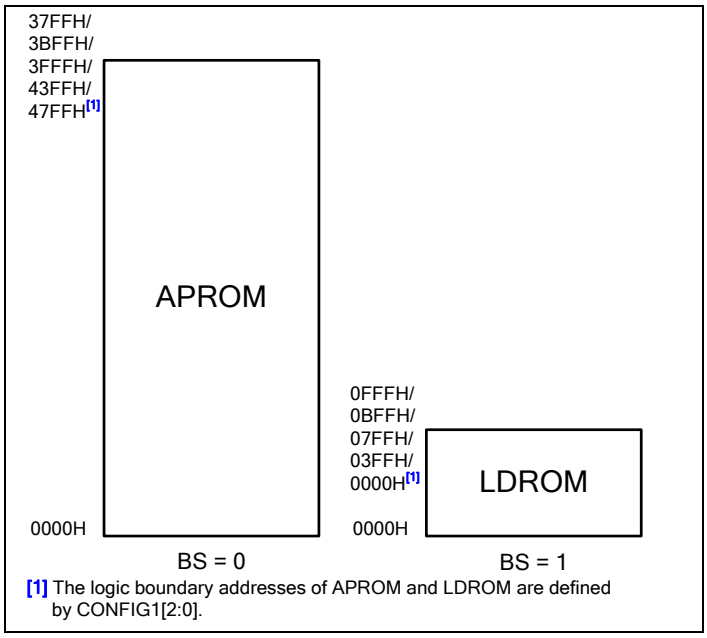


图 5–1. N76E885 程序内存分配图

5.2 数据内存

图 5-2所示N76E885中可用内部数据内存。内部数据内存占用一个独立于编程内存的地址空间。内部数据内存可以分割成三块。分别是RAM的低128字节，RAM的高128字节，和SFR空间的128字节。内部数据内存的地址总是8位宽度的，可用于256字节的地址空间。直接寻址高于7FH的地址会访问特殊功能寄存器（SFR）,间接寻址高于7FH的地址会访问RAM的高128字节。虽然SFR地址空间和RAM高128字节共享相同的逻辑地址80H 到 FFH，事实上他们是物理独立的实体。直接寻址区别于RAM的高128字节，仅可以访问SFR。SFR空间中的16个地址既可以字节寻址也可以位寻址。这些位寻址的SFR分布在地址以0H或8H结尾的地方。

内部RAM的低128字节在所有的80C51设备上都有的。最低的32字节作为通用寄存器分成四组8个寄存器，程序指令调用这些寄存器作为R0到R8。程序状态字(PSW[3:4])的两个位RS0 和 RS1用于选择哪个寄存器组被使用。这使代码空间更有效率，因为寄存器指令比其他直接寻址的指令更短。接下来16个字节(字节地址 20H 到 2FH)是可位寻址的内存空间(位地址 00H 到 7FH)。80C51指令集包括单位指令的广阔选择。这个域的128个位可以通过这些指令直接寻址。该域的位地址从00H 到 7FH。

所有低128字节空间，采用直接或间接寻址效果相同，是同一块区域。高128字节必须采用间接寻址，否则读写的是SFR。

对于整个256字节的内部RAM，另外一个应用是用于堆栈。这个区域通过堆栈指针(SP)来选择，SP存储堆栈顶的地址。当CALL、JMP或中断被调用，返回的地址就存在堆栈里面。没有限制堆栈从RAM的什么地方开始。默认情况下，在复位后堆栈指针为07H。用户可以改变该地址为任何想要的值。SP会指向最后使用的值。因此SP会增加，然后地址保存到堆栈中。当堆栈的内容出栈，SP会递减。

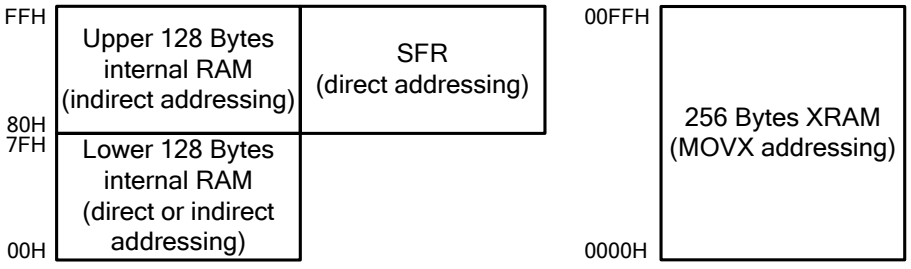


图 5-2. 数据内存分配图

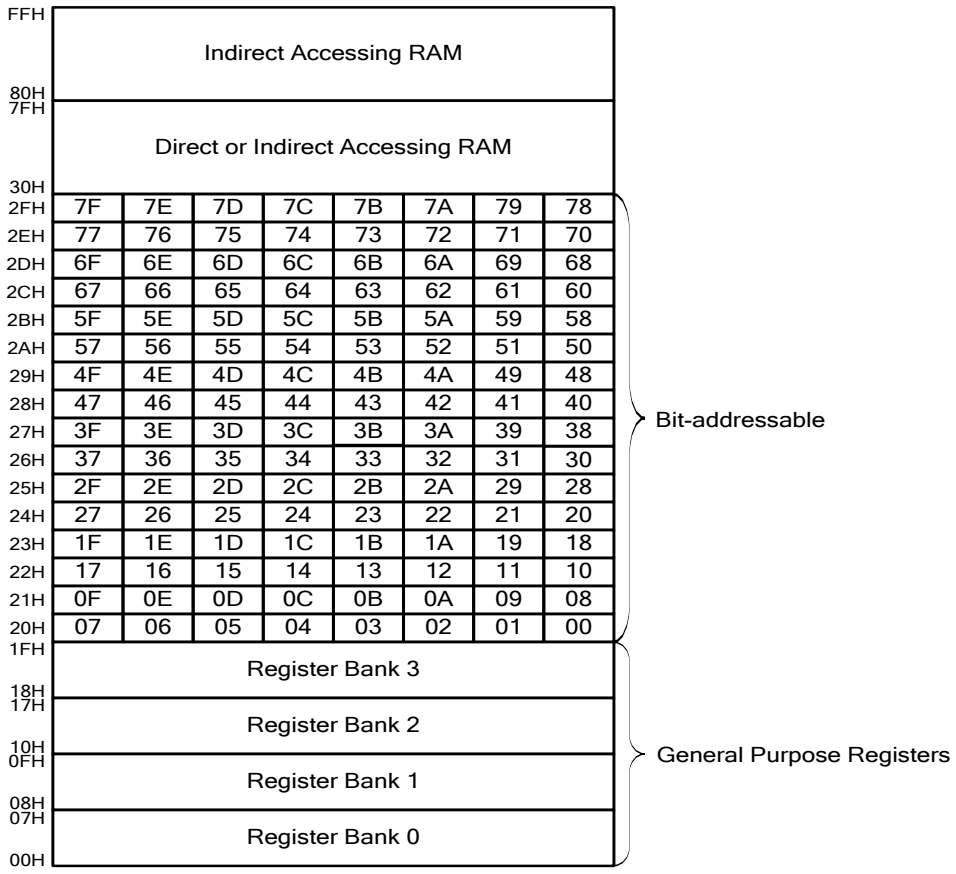


图 5-3. 内部 256 字节 RAM 地址

5.3 片上XRAM

N76E885提供额外的片上256字节的额外RAM 叫做XRAM来扩大RAM的空间。它占用地址空间从00H到FFH。这256字节的XRAM通过MOVE外部指令MOVX @DPTR 或 MOVX @Ri. (看下面示例代码)。注意堆栈指针不能位于XRAM的任何区域。

XRAM 读写汇编范例:

```
MOV    R0,#23H              ;write #5AH to XRAM with address @23H
MOV    A,#5AH
MOVX   @R0,A
MOV    R1,#23H              ;read from XRAM with address @23H
MOVX   A,@R1
MOV    DPTR,#0023H          ;write #5BH to XRAM with address @0023H
MOV    A,#5BH
MOVX   @DPTR,A
MOV    DPTR,#0023H          ;read from XRAM with address @0023H
MOVX   A,@DPTR
```

5.4 非易失性数据存储

通过使用IAP，APROM 或 LDROM任何页都可以用来作为非可变数据存储。IAP的细节，详见章节 22. “在应用编程 (IAP)” 。

6. 特殊功能寄存器(SFR)

N76E885用特殊功能寄存器（SFR）来控制或监视外设和其他模块。SFR位于地址80到FFH地址空间，仅可以通过直接寻址访问。那些地址以0H或8H结尾的SFR是可以位寻址的。当用户需要修改某一位而不改变其他位的情况下，这是非常有用的。其他所有的SFR仅可以字节寻址。N76E885包含标准8051中出现的所有SFR，然而一些额外的SFR也包含在内。因此在原始8051中一些没有使用的字节被给予了新的功能。SFR如下所列：

为了在地址0x80 到 0xFF之间提供多于128字节的SFR，补充了SFR页。默认情况下，所有SFR访问目的都是SFR页0。在设备初始化过程中位于SFR页1的地址可能需要去访问。寄存器SFRS用来切换SFR地址页。注意这个寄存器是有TA些保护的，大部分可用的SFR都在SFR页0和页1。

SFRS – SFR页选择(TA 保护)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	SFRPAGE
-	-	-	-	-	-	-	读/写

地址: 91H　　复位值: 0000 0000b

位	名称	描述
0	SFRPAGE	**SFR 页选择** 0 = 指令访问SFR 页 0. 1 = 指令访问SFR 页 1.

切换 SFR 页的例程:

```
MOV     TA,#0AAH            ;switch to SFR page 1
MOV     TA,#55H
ORL     SFRS,#01H

MOV     TA,#0AAH            ;switch to SFR page 0
MOV     TA,#55H
ANL     SFRS,#0FEH
```

表 6–1. SFR 内存分布

SFR Page	Addr	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
0 1	F8	SCON_1	PDTEN	PDTCNT	PMEN	PMD	-	EIP1 -	EIPH1 -
0 1	F0	B	-	ADCAQT	SPCR	SPSR	SPDR	P0DIDS	EIPH
0 1	E8	ADCCON0	PICON	PINEN	PIPEN	PIF	C2L	C2H	EIP
0 1	E0	ACC	ADCCON1	ADCCON2	ADCDLY	C0L	C0H	C1L	C1H
0 1	D8	PWMCON0	PWMPL	PWM01L	PWM23L	PWM67L	PWM45L	PIO	PWMCON1
0 1	D0	PSW	PWMPH	PWM01H	PWM23H	PWM67H	PWM45H	PNP	FBD
0 1	C8	T2CON	T2MOD	RCMP2L	RCMP2H	TL2	TH2	ADCMPL	ADCMPH
0 1	C0	I2CON	I2ADDR	ADCRL	ADCRH	T3CON	RL3	RH3	TA
0 1	B8	IP	SADEN	SADEN_1	SADDR_1	I2DAT	I2STAT	I2CLK	I2TOC
0 1	B0	P3	P0M1 P0S	P0M2 P0SR	P1M1 P1S	P1M2 P1SR	P2M1 P2S	P2M2 P2SR	IPH
0 1	A8	IE	SADDR	WDCON	BODCON1	P3M1 P3S	P3M2 P3SR	IAPFD	IAPCN
0 1	A0	P2	-	AUXR1	BODCON0	IAPTRG	IAPUEN	IAPAL	IAPAH
0 1	98	SCON	SBUF	SBUF_1	EIE	EIE1	-	-	CHPCON
0 1	90	P1	SFRS	CAPCON0	CAPCON1	CAPCON2	CKDIV	CKSWT	CKEN
0 1	88	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON	WKCON
0 1	80	P0	SP	DPL	DPH	-	-	RWK	PCON

SFR空间中没有占用的地址用‘-‘ 标记，保留将来使用。访问这些地址会有不可预知的影响，请避免这种情况。

表 6–2. SFR定义及复位值

符号	定义	地址/(页)	MSB							LSB[1]	复位值[2]
EIPH1	扩展中断优先级高位1	FFH/(0)	-	-	-	-	-	PWKTH	PT3H	PSH_1	0000 0000b
EIP1	扩展中断优先级1	FEH/(0)	-	-	-	-	-	PWKT	PT3	PS_1	0000 0000b
PMD	PWM 掩码数据	FCH	PMD7	PMD6	PMD5	PMD4	PMD3	PMD2	PMD1	PMD0	0000 0000b
PMEN	PWM 掩码使能	FBH	PMEN7	PMEN6	PMEN5	PMEN4	PMEN3	PMEN2	PMEN1	PMEN0	0000 0000b
PDTCNT[4]	PWM 死区时间计数	FAH	PDTCNT[7:0]								0000 0000b
PDTEN[4]	PWM 死区时间使能位	F9H	-	-	-	PDTCNT.8	PDT67EN	PDT45EN	PDT23EN	PDT01EN	0000 0000b
SCON_1	串口1控制寄存器	F8H	(FF) SM0_1/ FE_1	(FE) SM1_1	(FD) SM2_1	(FC) REN_1	(FB) TB8_1	(FA) RB8_1	(F9) TI_1	(F8) RI_1	0000 0000b
EIPH	扩展中断优先级高位	F7H	PT2H	PSPIH	PFBH	PWDTH	PPWMH	PCAPH	PPIH	PI2CH	0000 0000b
P0DIDS	P0 数字输出关闭	F6H	P07DIDS	P06DIDS	P05DIDS	P04DIDS	P03DIDS	P02DIDS	P01DIDS	P00DIDS	0000 0000b
SPDR	SPI 数据寄存器	F5H	SPDR[7:0]								0000 0000b
SPSR	SPI 状态寄存器	F4H	SPIF	WCOL	SPIOVF	MODF	DISMODF	-	-	-	0000 0000b
SPCR	SPI 控制寄存器	F3H	SSOE	SPIEN	LSBFE	MSTR	CPOL	CPHA	SPR[1:0]		0000 0000b
ADCAQT	ADC 采样时间	F2H	ADCAQT[7:0]								0000 0000b
B	B 寄存器	F0H	(F7) B.7	(F6) B.6	(F5) B.5	(F4) B.4	(F3) B.3	(F2) B.2	(F1) B.1	(F0) B.0	0000 0000b
EIP	扩展中断优先级	EFH	PT2	PSPI	PFB	PWDT	PPWM	PCAP	PPI	PI2C	0000 0000b
C2H	输入捕获2数据高位字节	EEH	C2H[7:0]								0000 0000b
C2L	输入捕获2数据低位字节	EDH	C2L[7:0]								0000 0000b
PIF	管脚中断标志	ECH	PIF7	PIF6	PIF5	PIF4	PIF3	PIF2	PIF1	PIF0	0000 0000b
PIPEN	管脚中断高电平/上升沿	EBH	PIPEN7	PIPEN6	PIPEN5	PIPEN4	PIPEN3	PIPEN2	PIPEN1	PIPEN0	0000 0000b
PINEN	管脚中断低电平/下降沿	EAH	PINEN7	PINEN6	PINEN5	PINEN4	PINEN3	PINEN2	PINEN1	PINEN0	0000 0000b
PICON	管脚中断控制	E9H	PIT67	PIT45	PIT3	PIT2	PIT1	PIT0	PIPS[1:0]		0000 0000b
ADCCON0	ADC 控制寄存器 0	E8H	(EF) ADCF	(EE) ADCS	(ED) ETGSEL1	(EC) ETGSEL0	(EB) ADCHS3	(EA) ADCHS2	(E9) ADCHS1	(E8) ADCHS0	0000 0000b
C1H	输入捕获1数据高位字节	E7H	C1H[7:0]								0000 0000b
C1L	输入捕获1数据低位字节	E6H	C1L[7:0]								0000 0000b
C0H	输入捕获0数据高位字节	E5H	C0H[7:0]								0000 0000b
C0L	输入捕获0数据低位字节	E4H	C0L[7:0]								0000 0000b
ADCDLY	ADC 触发延时	E3H	ADCDLY[7:0]								0000 0000b
ADCCON2	ADC 控制寄存器 2	E2H	ADFBEN	ADCMPOP	ADCMPEN	ADCMPO	P26DIDS	P20DIDS	-	ADCDLY.8	0000 0000b
ADCCON1	ADC 控制寄存器1	E1H	VREFSEL	ADCDIV[2:0]			ETGTYP[1:0]		ADCEX	ADCEN	0010 0000b
ACC	累加器	E0H	(E7) ACC.7	(E6) ACC.6	(E5) ACC.5	(E4) ACC.4	(E3) ACC.3	(E2) ACC.2	(E1) ACC.1	(E0) ACC.0	0000 0000b
PWMCON1	PWM 控制寄存器 1	DFH	PWMMOD[1:0]		GP	PWMTYP	FBINEN	PWMDIV[2:0]			0000 0000b
PIO	PWM I/O 切换	DEH	PIO7	PIO6	PIO5	PIO4	PIO3	PIO2	PIO1	PIO0	0000 0000b
PWM45L	PWM4/5 占空比低字节	DDH	PWM45[7:0]								0000 0000b
PWM67L	PWM6/7占空比低字节	DCH	PWM67[7:0]								0000 0000b
PWM23L	PWM2/3占空比低字节	DBH	PWM23[7:0]								0000 0000b
PWM01L	PWM0/1占空比低字节	DAH	PWM01[7:0]								0000 0000b
PWMPL	PWM 周期低字节	D9H	PWMP[7:0]								0000 0000b
PWMCON0	PWM 控制寄存器 0	D8H	(DF) PWMRUN	(DE) LOAD	(DD) PWMF	(DC) CLRPWM	(DB) INTTYP1	(DA) INTTYP0	(D9) INTSEL1	(D8) INTSEL0	0000 0000b
FBD	故障刹车数据	D7H	FBF	FBINLS	FBD5	FBD4	FBD3	FBD2	FBD1	FBD0	0000 0000b
PNP	PWM 负极性	D6H	PNP7	PNP6	PNP5	PNP4	PNP3	PNP2	PNP1	PNP0	0000 0000b
PWM45H	PWM4/5占空比高字节	D5H	-	-	-	-	PWM45[11:8]				0000 0000b
PWM67H	PWM6/7占空比高字节	D4H	-	-	-	-	PWM67[11:8]				0000 0000b
PWM23H	PWM2/3占空比高字节	D3H	-	-	-	-	PWM23[11:8]				0000 0000b
PWM01H	PWM0/1占空比高字节	D2H	-	-	-	-	PWM01[11:8]				0000 0000b
PWMPH	PWM 周期高字节	D1H	-	-	-	-	PWMP[11:8]				0000 0000b
PSW	程序状态字	D0H	(D7) CY	(D6) AC	(D5) F0	(D4) RS1	(D3) RS0	(D2) OV	(D1) -	(D0) P	0000 0000b
ADCMPH	ADC 比较高字节	CFH	ADCMP[9:2]								0000 0000b
ADCMPL	ADC 比较低字节	CEH	-	-	-	-	-	-	ADCMP[1:0]		0000 0000b
TH2	定时器 2 高字节	CDH	TH2[7:0]								0000 0000b
TL2	定时器 2 低字节	CCH	TL2[7:0]								0000 0000b
RCMP2H	定时器 2 比较高字节	CBH	RCMP2H[7:0]								0000 0000b
RCMP2L	定时器 2 比较低字节	CAH	RCMP2L[7:0]								0000 0000b
T2MOD	定时器 2 模式	C9H	LDEN	T2DIV[2:0]			CAPCR	CMPCR	LDTS[1:0]		0000 0000b
T2CON	定时器 2 控制寄存器	C8H	(CF) TF2	(CE) -	(CD) -	(CC) -	(CB) -	(CA) TR2	(C9) -	(C8) CM/RL2	0000 0000b

表 6–2. SFR定义及复位值

符号	定义	地址/(页)	MSB							LSB[1]	复位值[2]
TA	时控访问保护	C7H	TA[7:0]								0000 0000b
RH3	定时器3自动重载高字节	C6H	RH3[7:0]								0000 0000b
RL3	定时器3自动重载低字节	C5H	RL3[7:0]								0000 0000b
T3CON	定时器 3 控制寄存器	C4H	SMOD_1	SMOD0_1	BRCK	TF3	TR3	T3PS[2:0]			0000 0000b
ADCRH	ADC 结果高字节	C3H	ADCR[9:2]								0000 0000b
ADCRL	ADC 结果低字节	C2H	-	-	-	-	-	-	ADCR[1:0]		0000 0000b
I2ADDR	I^2C 从机地址	C1H	I2ADDR[7:1]							GC	0000 0000b
I2CON	I^2C 控制寄存器	C0H	(C7) -	(C6) I2CEN	(C4) STA	(C4) STO	(C3) SI	(C2) AA	(C1) -	(C0) -	0000 0000b
I2TOC	I^2C定时计数器	BFH	-	-	-	-	-	I2TOCEN	DIV	I2TOF	0000 0000b
I2CLK	I^2C 时钟	BEH	I2CLK[7:0]								0000 1110b
I2STAT	I^2C 状态	BDH	I2STAT[7:3]					0	0	0	1111 1000b
I2DAT	I^2C 数据	BCH	I2DAT[7:0]								0000 0000b
SADDR_1	从机 1地址	BBH	SADDR_1[7:0]								0000 0000b
SADEN_1	从机 1地址掩码	BAH	SADEN_1[7:0]								0000 0000b
SADEN	从机 0 地址掩码	B9H	SADEN[7:0]								0000 0000b
IP	中断优先级	B8H	(BF) -	(BE) PADC	(BD) PBOD	(BC) PS	(BB) PT1	(BA) PX1	(B9) PT0	(B8) PX0	0000 0000b
IPH	中断优先级高	B7H	-	PADCH	PBODH	PSH	PT1H	PX1H	PT0H	PX0H	0000 0000b
P2SR	P2 斜率控制	B6H/(1)	-	P2SR.6	P2SR.5	P2SR.4	P2SR.3	P2SR.2	P2SR.1	P2SR.0	0000 0000b
P2M2	P2 模式选择2	B6H/(0)	-	P2M2.6	P2M2.5	P2M2.4	P2M2.3	P2M2.2	P2M2.1	P2M2.0	0000 0000b
P2S	P2 施密特触发输入	B5H/(1)	-	P2S.6	P2S.5	P2S.4	P2S.3	P2S.2	P2S.1	P2S.0	0000 0000b
P2M1	P2 模式选择1	B5H/(0)	-	P2M1.6	P2M1.5	P2M1.4	P2M1.3	P2M1.2	P2M1.1	P2M1.0	0111 1111b
P1SR	P1 斜率控制	B4H/(1)	-	-	-	-	-	-	P1SR.1	P1SR.0	0000 0000b
P1M2	P1 模式选择2	B4H/(0)	-	-	-	-	CLOEN	P12UP	P1M2.1	P1M2.0	0000 0000b
P1S	P1 施密特输入	B3H/(1)	P21SNK	P20SNK	P03SNK	P02SNK	P01SNK	P1S.2	P1S.1	P1S.0	0000 0000b
P1M1	P1 模式选择1	B3H/(0)	-	-	-	-	T1OE	T0OE	P1M1.1	P1M1.0	0000 0011b
P0SR	P0 斜率控制	B2H/(1)	P0SR.7	P0SR.6	P0SR.5	P0SR.4	P0SR.3	P0SR.2	P0SR.1	P0SR.0	0000 0000b
P0M2	P0 模式选择2	B2H/(0)	P0M2.7	P0M2.6	P0M2.5	P0M2.4	P0M2.3	P0M2.2	P0M2.1	P0M2.0	0000 0000b
P0S	P0 施密特输入	B1H/(1)	P0S.7	P0S.6	P0S.5	P0S.4	P0S.3	P0S.2	P0S.1	P0S.0	0000 0000b
P0M1	P0 模式选择1	B1H/(0)	P0M1.7	P0M1.6	P0M1.5	P0M1.4	P0M1.3	P0M1.2	P0M1.1	P0M1.0	1111 1111b
P3	端口 3	B0H	(B7) P3.7	(B6) P3.6	(B5) P3.5	(B4) P3.4	(B3) P3.3	(B2) P3.2	(B1) P3.1	(B0) P3.0	输出 1111 1111b 输入, XXXX XXXXb[3]
IAPCN	IAP 控制	AFH	IAPA[17:16]		FOEN	FCEN	FCTRL[3:0]				0011 0000b
IAPFD	IAP 数据	AEH	IAPFD[7:0]								0000 0000b
P3SR	P3 斜率控制	ADH/(1)	P3SR.7	P3SR.6	P3SR.5	P3SR.4	P3SR.3	P3SR.2	P3SR.1	P3SR.0	0000 0000b
P3M2	P3 模式选择 2	ADH/(0)	P3M2.7	P3M2.6	P3M2.5	P3M2.4	P3M2.3	P3M2.2	P3M2.1	P3M2.0	0000 0000b
P3S	P3 施密特输入	ACH/(1)	P3S.7	P3S.6	P3S.5	P3S.4	P3S.3	P3S.2	P3S.1	P3S.0	0000 0000b
P3M1	P3 模式选择 1	ACH/(0)	P3M1.7	P3M1.6	P3M1.5	P3M1.4	P3M1.3	P3M1.2	P3M1.1	P3M1.0	1111 1111b
BODCON1[4]	欠压检测控制1	ABH	-	-	-	-	-	LPBOD[1:0]		BODFLT	POR, 0000 0001b 其他, 0000 0UUUb
WDCON[4]	看门狗定时器控制	AAH	WDTEN	WDCLR	WDTF	WIDPD	WDTRF	WDPS[2:0]			POR, 0000 0111b WDT, 0000 1UUUb 其他, 0000 UUUUb
SADDR	从机 0 地址	A9H	SADDR[7:0]								0000 0000b
IE	中断使能	A8H	(AF) EA	(AE) EADC	(AD) EBOD	(AC) ES	(AB) ET1	(AA) EX1	(A9) ET0	(A8) EX0	0000 0000b
IAPAH	IAP 地址高字节	A7H	IAPA[15:8]								0000 0000b
IAPAL	IAP地址低字节	A6H	IAPA[7:0]								0000 0000b
IAPUEN[4]	IAP 更新使能	A5H	-	-	-	-	-	CFUEN	LDUEN	APUEN	0000 0000b
IAPTRG[4]	IAP 执行	A4H	-	-	-	-	-	-	-	IAPGO	0000 0000b

表 6–2. SFR定义及复位值

符号	定义	地址/(页)	MSB							LSB[1]	复位值[2]
BODCON0[4]	欠压侦测控制0	A3H	BODEN[5]	BOV[2:0][5]			BOF[6]	BORST[5]	BORF	BOS[7]	POR, CCCC XC0Xb BOD, UUUU XU1Xb Others, UUUU XUUXb
AUXR1	辅助寄存器 1	A2H	SWRF	RSTPINF	T1LXTM	T0LXTM	GF2	UART0PX	0	DPS	POR, 0000 0000b Software, 1U00 0000b $\overline{RST}$ pin, U100 0000b Others, UU00 0000b
P2	端口 2	A0H	(A7) 0	(A6) P2.6	(A5) P2.5	(A4) P2.4	(A3) P2.3	(A2) P2.2	(A1) P2.1	(A0) P2.0	Output latch, 0111 1111b Input, 0XXX XXXXb[3]
CHPCON[4]	芯片控制	9FH	SWRST	IAPFF	-	-	-	-	BS[5]	IAPEN	Software, 0000 00U0b Others, 0000 00C0b
EIE1	扩展中断使能1	9CH	-	-	-	-	-	EWKT	ET3	ES_1	0000 0000b
EIE	扩展中断使能	9BH	ET2	ESPI	EFB	EWDT	EPWM	ECAP	EPI	EI2C	0000 0000b
SBUF_1	串口1数据缓存	9AH	SBUF_1[7:0]								0000 0000b
SBUF	串口0数据缓存	99H	SBUF[7:0]								0000 0000b
SCON	串口0控制寄存器	98H	(9F) SM0/FE	(9E) SM1	(9D) SM2	(9C) REN	(9B) TB8	(9A) RB8	(99) TI	(98) RI	0000 0000b
CKEN[4]	时钟使能	97H	EXTEN[1:0]		HIRCEN	-	-	-	-	CKSWTF	0011 0000b
CKSWT[4]	时钟切换	96H	HXTST	LXTST	HIRCST	-	ECLKST	OSC[1:0]		-	0011 0000b
CKDIV	时钟除频	95H	CKDIV[7:0]								0000 0000b
CAPCON2	输入捕获控制寄存器2	94H	-	ENF2	ENF1	ENF0	-	-	-	-	0000 0000b
CAPCON1	输入捕获控制寄存器1	93H	-	-	CAP2LS[1:0]		CAP1LS[1:0]		CAP0LS[1:0]		0000 0000b
CAPCON0	输入捕获控制寄存器0	92H	-	CAPEN2	CAPEN1	CAPEN0	-	CAPF2	CAPF1	CAPF0	0000 0000b
SFRS[4]	SFR 页选择	91H	-	-	-	-	-	-	-	SFRPSEL	0000 0000b
P1	端口 1	90H	(97) 0	(96) 0	(95) 0	(94) 0	(93) 0	(92) P1.2	(91) P1.1	(90) P1.0	Output latch, 0000 0111b Input, 0000 0XXXb[3]
WKCON	自唤醒定时器控制	8FH	-	-	WKTCK	WKTF	WKTR	WKPS[2:0]			0000 0000b
CKCON	时钟控制	8EH	-	PWMCKS	-	T1M	T0M	-	-	-	0000 0000b
TH1	定时器1高字节	8DH	TH1[7:0]								0000 0000b
TH0	定时器0高字节	8CH	TH0[7:0]								0000 0000b
TL1	定时器1低字节	8BH	TL1[7:0]								0000 0000b
TL0	定时器0高字节	8AH	TL0[7:0]								0000 0000b
TMOD	定时器0 及 1 模式	89H	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0	0000 0000b
TCON	定时器 0 及1 控制	88H	(8F) TF1	(8E) TR1	(8D) TF0	(8C) TR0	(8B) IE1	(8A) IT1	(89) IE0	(88) IT0	0000 0000b
PCON	电源控制	87H	SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL	POR, 0001 0000b Others, 000U 0000b
RWK	自唤醒定时器重载数据	86H	RWK[7:0]								0000 0000b
DPH	数据指针高字节	83H	DPTR[15:8]								0000 0000b
DPL	数据指针低字节	82H	DPTR[7:0]								0000 0000b
SP	堆栈指针	81H	SP[7:0]								0000 0111b
P0	端口 0	80H	(87) P0.7	(86) P0.6	(85) P0.5	(84) P0.4	(83) P0.3	(82) P0.2	(81) P0.1	(80) P0.0	Output latch, 1111 1111b Input, XXXX XXXXb[3]

[1] ( ) 项意思是可位寻址 SFR

[2] 复位值符号描述. 0: 逻辑 0; 1: 逻辑 1; U: 不变; C: 详见**[5]**; X: 详见 **[3]**, **[6]**, **[7]**.

[3] 复位之后所有I/O引脚为默认悬浮输入模式。如果RPD (CONFIG0.2) =1（未编程），那么读回P1.2引脚总是0.

[4] 这些 SFRS 需要时控保护开启。详见 章节20. “时控保护”

[5] 在特定的复位之后，这些 SFR 根据CONFIG。详见 章节 29. “配置字”.

[6] BOF 复位值取决于CONFIG2 不同的设置和V_{DD} 电压值。详见表 25–1
[7] 当欠压侦测使能，BOS 是只读标志，由V_{DD} 值来决定。

标记 '-' 的位保留将来使用。他们必须保持在初始状态，访问这些位可能导致不可预知的后果。

7. 通用 80C51 系统控制

A 或 ACC – 累加器 (可位寻址)

7	6	5	4	3	2	1	0
ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: E0H 复位值: 0000 0000b

位	名称	描述
7:0	ACC[7:0]	**累加器** 标准80C51 累加器

B – B 寄存器 (可位寻址)

7	6	5	4	3	2	1	0
B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: F0H 复位值: 0000 0000b

位	名称	描述
7:0	B[7:0]	**B 寄存器** 标准80C51 B累加器，常用语乘法除法指令

SP – 堆栈指针

7	6	5	4	3	2	1	0
SP[7:0]							
读/写							

地址: 81H 复位值: 0000 0111b

位	名称	描述
7:0	SP[7:0]	**堆栈指针** 堆栈指针存储的是RAM的地址，该地址是堆栈的起始地址。在执行PUSH 或 CALL 指令的时候，在数据被存储之前，堆栈指针递增。注意：SP的默认值是07H。因此堆栈起始位置在08H。

DPL –数据指针低字节

7	6	5	4	3	2	1	0
DPL[7:0]							
读/写							

地址: 82H　　复位值: 0000 0000b

位	名称	描述
7:0	DPL[7:0]	**数据指针低字节** 这是16位数据指针的低字节， DPL 结合DPH作为16位的数据指针DPTR访问想要访问的RAM或编程内存地址。DPS (AUXR1.0) 位决定哪一个数据指针DPTR 或DPTR1激活。

DPH – 数据指针高字节

7	6	5	4	3	2	1	0
DPH[7:0]							
读/写							

地址: 83H　　复位值: 0000 0000b

位	名称	描述
7:0	DPH[7:0]	**数据指针高字节** 这是16位数据指针的高字节， DPH结合DPL作为16位的数据指针DPTR访问想要访问的RAM或编程内存地址。DPS (AUXR1.0) 位决定哪一个数据指针DPTR 或DPTR1激活。

PSW –编程状态字 (可位寻址)

7	6	5	4	3	2	1	0
CY	AC	F0	RS1	RS0	OV	F1	P
读/写	读/写	读/写	读/写	读/写	读/写	读/写	R

地址: D0H　　复位值: 0000 0000b

位	名称	描述
7	CY	**高位进位标志** 进行加法或减法操作时，当前运算需要向高位进位或借位时，CY将置位，否则清零。 在进行MUL 或 DIV运算时，CY始终为0。 CY受DA A指令影响，用来表示是否初始BCD数大于100。 在CJNE指令中，如果第一个无符号数的值小于第二个，则CY置1，否则清0。
6	AC	**辅助进位标志** 当前运算导致从半字节的低序第4位进位或借位，该位置位，否则清零。
5	F0	**用户标志0.** 可由用户置位或清零的通用标志。
4	RS1	**寄存器页选择**

3	RS0	这两位用来选择R0到R7位于四页中的哪一页：. RS1 RS0 寄存器页 RAM 地址 0 0 0 00H 到 07H 0 1 1 08H 到 0FH 1 0 2 10H 到 17H 1 1 3 18H 到 1FH
2	OV	**溢出标志** OV用于标示发生溢出。对于加法指令 ADD或ADDC指令中，如果位6有进位而位7没进位，或者位7有进位而位6没有进位，则溢出标志置“1”，反之清“0”。 OV 也用于标示有符号数累加结果，当两个正数相加，或两个负数相加结果为负数时OV为1。对于减法指令SUBB，当位6发生借位而位7没有，或者位7发生借位而位6没有借位，则溢出标志置“1”，反之清“0”。 OV也用于标示两个数相减时，当一个正数加一负数结果为负，或两个负数相减结果为负时。 对于MUL乘法指令，当结果大于255 (00FFH)时，OV置1。反之清0。 对于DIV除法指令，通常情况下OV为0。 除非当B设定值为00H，则A和B的返回值为随机值，同时OV 置1。
1	F1	**用户标志1** 可由用户置位或清零的通用标志。
0	P	**奇偶标志** 当累加结果为奇数时，该标志置1，偶数时清0。其执行奇偶校验。

表 7–1. 指令对标志位的影响

指令	CY	OV	AC	指令	CY	OV	AC
ADD	X[1]	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C, bit	X		
MUL	0	X		ANL C, /bit	X		
DIV	0	X		ORL C, bit	X		
DA A	X			ORL C, /bit	X		
RRC A	X			MOV C, bit	X		
RLC A	X			CJNE	X		
SETB C	1						

[1] X表示根据指令的结果变化.

PCON – 电源控制

7	6	5	4	3	2	1	0
SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL
读/写	读/写	-	读/写	读/写	读/写	读/写	读/写

地址: 87H 复位值: 详见 表 6–2. SFR定义及复位值

位	名称	描述
3	GF1	**通用标志1** 通用标志可由用户置位或清零.

位	名称	描述
2	GF0	**通用标志0** 通用标志可由用户置位或清零.

8. I/O 端口结构及工作模式

N76E885最大支持26个可位寻址的通用I/O引脚，分成4组 P0 到 P3 。每一个端口有它的端口控制寄存器(Px)。端口控制寄存器的写和读有不同的意思。写端口控制寄存器设置输出锁存逻辑值，读获取端口引脚的逻辑状态。所有I/O引脚（除P1.2）可以被软件独立配置成四中I/O模式中的一种。这四种模式是准双向模式（标准8051端口结构）、推挽输出、输入和开漏模式。每一个端口通过两个特殊功能寄存器 PxM1 和 PxM2来选择端口Px的I/O模式。下表指示如何选择Px.n的I/O模式。注意任何复位之后，默认的配置是高阻输入模式。

表 8–1. 配置不同的I/O 模式

PxM1.n	PxM2.n	I/O 类型
0	0	准双向
0	1	推挽输出
1	0	输入 (高阻)
1	1	开漏

所有I/O引脚可以通过PxS寄存器里对应的位选择为TTL电平输入或施密特触发输入 。施密特触发输入有更好的抗干扰能力。

有四个I/O引脚支持大的输出或灌电流，包括P0.1, P0.2, P0.3, P2.0, 及 P2.1.。默认情况下，它们和其他I/O引脚的输出能力一样。通过设置P1S寄存器的PxnSNK位，它们可以被独立地配置成高输出电流能力。这适合用于驱动LED或不带额外BJT设备的大负载。

当配置RPD (CONFIG0.2) 为0，P1.2 被配置为输入引脚。同时P1.2将永远在输入和施密特触发模式，通过P12UP （P1M2.2)，使能内部上拉电阻。如果RPD未编程，P1.2作为外部复位引脚，P1.2作为管脚功能无效，由于作为复位脚，内部上拉电阻始终有效，此种状态下读取P1.2的值始终为0.

8.1 准双向模式

准双向模式作为标准8051的I/O结构，可以同时用作输入和输出。当端口输出逻辑高时，驱动能力较弱，同时允许外部器件将电平拉低。当引脚被拉低时有强驱动能力，会吸收大电流。在准双向I/O 结构中，有三个上拉三极，适应不同的应用。其中一个上拉叫做特弱上拉，当端口锁定在逻辑1时，打开特弱上拉，特弱上拉有很小电流将引脚拉高。

第二种上拉为“弱上拉”，当外部端口引脚自身处于逻辑1电平时打开。这种上拉提供源电流以使准双向引脚输出1。如果引脚为逻辑1，被外部器件拉低， “弱上拉”关闭， 仅有“特弱上拉”打开。 此时要将引脚拉

低，外部器件要有足够的灌电流 (大于ITL)以克服“弱上拉”，并使端口的电压低于输入门限电压 (低于 V_{IL})。

第三种上拉为“强上拉”。这种上拉用于在准双向口引脚上，加速端口电平由逻辑0转为逻辑1的转换速度。 当这种情况发生时，强上拉打开两个总线时钟的时间以快速将端口引脚拉高。 然后就关闭，弱上拉和特弱上拉继续保持该端口引脚为高。 准双向端口结构如下所示。

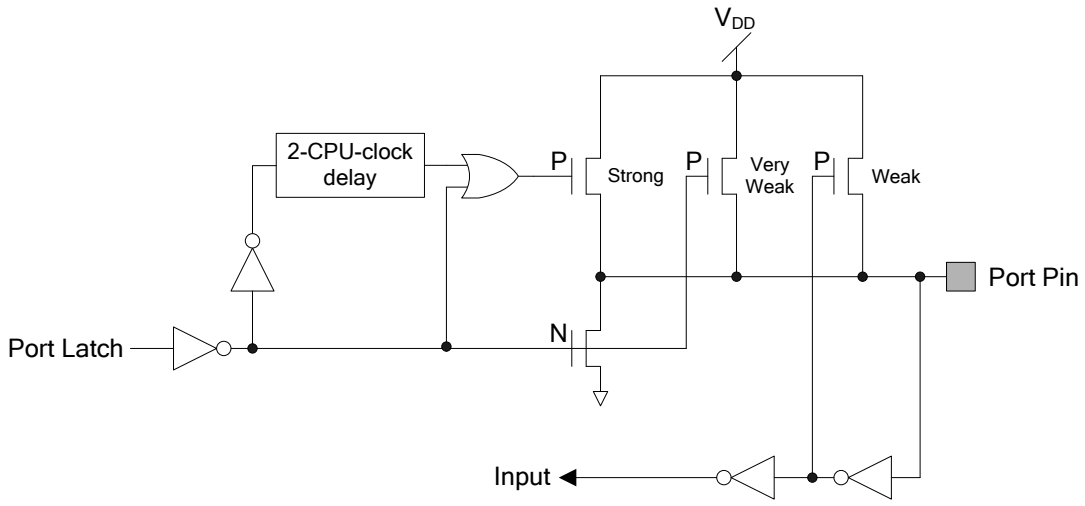


图 8-1. 准双效模式结构

8.2 推挽模式

推挽输出配置与开漏和准双向输出模式有相同的下拉结构。当端口锁定为1时，提供持续的强上拉。推挽输出模式用于需要从端口输出大电流从时的应用。

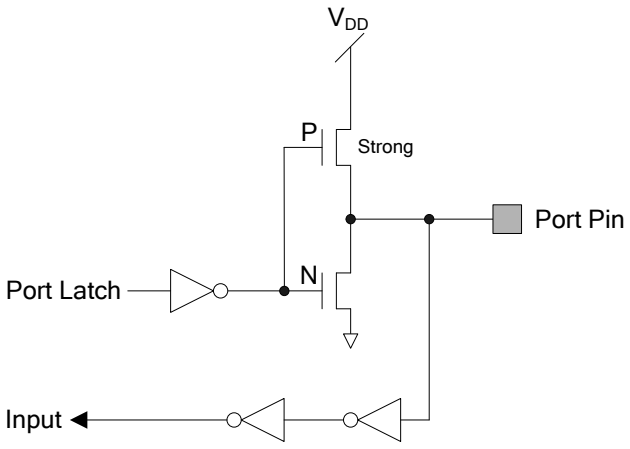


图 8-2. 推挽模式结构图

8.3 输入高阻模式

输入模式提供真实的高阻输入路径。虽然准双向模式也可以作为输入引脚，但是它需要相对强的输入源。输入模式的好处是减少在逻辑0时电流的消耗，如果是准双向模式，逻辑0时总是消耗来自V_{DD} 的电流。用户需要注意的是，输入模式应该由外部设备或电阻提供一个确定的电平。悬浮的引脚在掉电状态下会引起漏电。

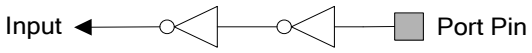


图 8-3. 输入高阻模式结构图

8.4 开漏模式

开漏输出配置关闭所有内部上拉，当端口锁定为逻辑0时，仅打开驱动端口的下拉晶体管。当单口锁存为逻辑1时，它就和输入模式一样。通常用于I2C输出线上，开漏引脚需要加一个外部上拉电阻，典型连一个电阻到VDD 。 用户需要注意的是，开漏模式输出逻辑1的时候，应该由外部设备或电阻提供一个确定的电平。悬浮的引脚在掉电状态下会引起漏电。

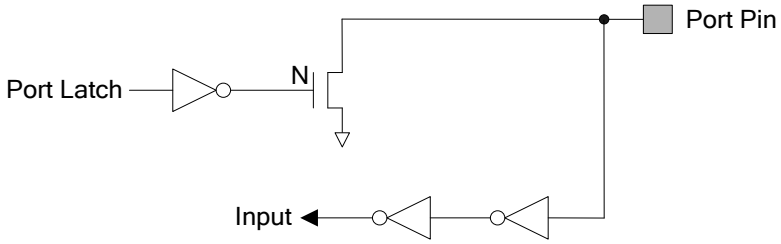


图 8-4. 开漏模式结构图

8.5 读-修改-写 指令

从SFR或内部RAM读一个字节，修改它，并重新写回去的指令，叫做读-修改-写指令。当目的是一个I/O端口或一个端口位，这些指令读内部输出锁存而不是外部引脚的状态，这种指令读端口SFR的值，修改它并写回到SFR端口。所有读-修改-写的指令如下所列：

指令	描述
ANL	逻辑 与. (ANL direct, A 和 ANL direct, #data)
ORL	逻辑 或. (ORL direct, A 和 ORL direct, #data)
XRL	逻辑异或 OR. (XRL direct, A 和 XRL direct, #data)

JBC		为1转跳指令并清除. (JBC bit, rel)
CPL		位取反. (CPL bit)
INC		加一指令. (INC direct)
DEC		减一指令. (DEC direct)
DJNZ		减一不为零转跳指令. (DJNZ direct, rel)
MOV	bit, C	移进位标志到位. (MOV bit, C)
CLR	bit	清位. (CLR bit)
SETB	bit	置位. (SETB bit)

最后三条指令看似不是明显的读-修改-写指令，实际也是读-修改-写指令。 可以读整个端口锁定值，修改改变位，写入新的值。

8.6 管脚控制寄存器

N76E885有许多I/O控制寄存器提供灵活的各种应用。和I/O相关的SFR可以分类成三组：输入输出控制，输出模式控制、输入类型和灌电流控制。所有SFR如下所列：

8.6.1 输入输出数据控制

这些寄存器是I/O输入输出数据缓存。读获取I/O输入的数据。写驱动数据输出，所有这些寄存器都是可位寻址的。..

P0 – 端口 0 (可位寻址)

7	6	5	4	3	2	1	0
P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: 80H 复位值: 1111 1111b

位	名称	描述
7:0	P0[7:0]	**端口 0** 端口 0 是 8-bit 通用 I/O 端口.

P1 – 端口 1 (可位寻址)

7	6	5	4	3	2	1	0
0	0	0	0	0	P1.2	P1.1	P1.0
R	R	R	R	R	R	读/写	读/写

地址: 90H 复位值: 1111 1111b

位	名称	描述
7:3	0	**保留位** 始终为0
2	P1.2	**端口 1 第 2 位** 当RPD (CONFIG0.2)=0，P1.2为输入高阻管脚。默认不更改，P1.2读取始终为0
1	P1.1	**端口 1 第 1 位** 当采用内部晶振或外部时钟输入作为系统时钟时，P1.1可用作普通管脚使用。当采用外部晶振输入方式，P1.1脚用作 XOUT 此时对P1.1写入数值无效，且时钟读取值为0
0	P1.0	**端口 1 第 0 位** 当采用内部晶振或外部时钟输入作为系统时钟时，P1.0可用作普通管脚使用。当采用外部晶振输入方式，P1.0脚用作 XIN 此时对P1.0写入数值无效，且时钟读取值为0

P2 – 端口 2 (可位寻址)

7	6	5	4	3	2	1	0
0	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
R	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: A0H 复位值: 1111 1111b

位	名称	描述
7	0	**保留位** 读取始终为0
6:0	P2[7:0]	**端口 2** Port 2 最大7位通用管脚。

P3 – 端口 3 (可位寻址)

7	6	5	4	3	2	1	0
P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: B0H 复位值: 1111 1111b

位	名称	描述
7:0	P3[7:0]	**端口 3** 端口 3 是 8-bit 通用 I/O 端口.

8.6.2 输出模式控制

这些寄存器控制输出模式。可以配置为四种模式：输入模式、准双向模式、推挽或开漏模式。每一个引脚可以独立地配置。对P1.2引脚，在P1M2.2时钟有一个上拉电阻电阻选择位。

P0M1 – 端口 0模式选择1[1]

7	6	5	4	3	2	1	0
P0M1.7	P0M1.6	P0M1.5	P0M1.4	P0M1.3	P0M1.2	P0M1.1	P0M1.0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: B1H, 页: 0　　复位值: 1111 1111b

位	名称	描述
7:0	P0M1[7:0]	**端口 0 模式选择1**

P0M2 – 端口 0 模式选择 2 [1]

7	6	5	4	3	2	1	0
P0M2.7	P0M2.6	P0M2.5	P0M2.4	P0M2.3	P0M2.2	P0M2.1	P0M2.0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: B2H, 页: 0　　复位值: 0000 0000b

位	名称	描述
7:0	P0M2[7:0]	**端口 0 模式选择 2**

[1] P0M1 和 P0M2 结合用于决定P0每个引脚的I/O模式，见表 8–1. 配置不同的I/O .

P1M1 – 端口 1 模式选择 1

7	6	5	4	3	2	1	0
-	-	-	-	T1OE	T0OE	P1M1.1[2]	P1M1.0[2]
-	-	-	-	读/写	读/写	读/写	读/写

地址: B3H, 页: 0　　复位值: 0000 0011b

位	名称	描述
1:0	P1M1[1:0]	**端口 1 模式选择 1**

P1M2 –端口 1 模式选择2

7	6	5	4	3	2	1	0
-	-	-	-	CLOEN	P12UP	P1M2.1[2]	P1M2.0[2]
-	-	-	-	读/写	读/写	读/写	读/写

地址: B4H, 页: 0　　复位值: 0000 0000b

位	名称	描述
2	P12UP	**P1.2 上拉电阻使能位** 0 = P1.2 上拉功能关闭 1 = P1.2 上拉功能打开. 仅当 RPD (CONFIG0.2) =0 时，该位有效。当配置位$\overline{\text{RST}}$复位脚时，内部上拉电阻始终有效
1:0	P1M2[1:0]	**端口 1 模式选择2**

[2] P1M1 和 P1M2结合用于决定P1每个引脚的I/O模式，见表 8–1. 配置不同的I/O .

P2M1 – 端口2 模式选择 1 [3]

7	6	5	4	3	2	1	0
-	P2M1.6	P2M1.5	P2M1.4	P2M1.3	P2M1.2	P2M1.1	P2M1.0
-	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: B5H, 页: 0　　复位值: 0111 1111b

位	名称	描述
6:0	P2M1[6:0]	**端口2 模式选择 1**

P2M2 – 端口2 模式选择 2 [3]

7	6	5	4	3	2	1	0
-	P2M2.6	P2M2.5	P2M2.4	P2M2.3	P2M2.2	P2M2.1	P2M2.0
-	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: B6H, 页: 0　　复位值: 0000 0000b

位	名称	描述
6:0	P2M2[6:0]	**端口2 模式选择 2**

[3] P2M1 和 P2M2结合用于决定P2每个引脚的I/O模式，见表 8–1. 配置不同的I/O .

P3M1 – 端口 3 模式选择 1 [4]

7	6	5	4	3	2	1	0
P3M1.7	P3M1.6	P3M1.5	P3M1.4	P3M1.3	P3M1.2	P3M1.1	P3M1.0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: ACH, 页: 0　　复位值: 1111 1111b

位	名称	描述
7:0	P3M1[7:0]	**端口 3 模式选择 1**

P3M2 – 端口3 模式选择 2 [4]

7	6	5	4	3	2	1	0
P3M2.7	P3M2.6	P3M2.5	P3M2.4	P3M2.3	P3M2.2	P3M2.1	P3M2.0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: ADH, 页: 0　　复位值: 0000 0000b

位	名称	描述
7:0	P3M2[7:0]	**端口3 模式选择 2**

[4] P3M1 和 P3M2结合用于决定P3每个引脚的I/O模式，见表 8–1. 配置不同的I/O .

8.6.3 输入类型和输出能力控制

每一个 I/O 引脚可以独立地配置成TTL输入或施密特触发输入。P1S[7:3]位用于P0.1, P0.2, P0.3, P2.0, 及 P2.1的输出控制。 这四个引脚支持大灌入电流和输出电流。注意所有PxS 寄存器通过切换SFR页到页1来访问。

P0S – 端口 0 施密特触发输入

7	6	5	4	3	2	1	0
P0S.7	P0S.6	P0S.5	P0S.4	P0S.3	P0S.2	P0S.1	P0S.0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: B1H, 页: 1　　复位值: 0000 0000b

位	名称	描述
n	P0S.n	**P0.n 施密特触发输入使能位** 0 = TTL 电平输入. 1 = 施密特触发输入

P1S – 端口 1施密特触发输入

7	6	5	4	3	2	1	0
P21SNK	P20SNK	P03SNK	P02SNK	P01SNK	P1S.2	P1S.1	P1S.0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: B3H, 页: 1　　复位值: 0000 0000b

位	名称	描述
7	P21SNK	**P2.1 大电流输出功能选择** 0 = P2.1 正常输出能力 1 = P2.1 大电流输出功能打开
6	P20SNK	**P2.0 大电流输出功能选择** 0 = P2.0 正常输出能力. 1 = P2.0 大电流输出功能打开
5	P03SNK	**P0.3 大电流输出功能选择** 0 = P0.3 正常输出能力 1 = P0.3 大电流输出功能打开.

位	名称	描述
4	P02SNK	**P0.2 大电流输出功能选择** 0 = P0.2 正常输出能力 1 = P0.2 大电流输出功能打开
3	P01SNK	**P0.1 大电流输出功能选择** 0 = P0.1 正常输出能力 1 = P0.1 大电流输出功能打开
2	P1S.2	**P1.2 施密特触发输入功能** 0 = P1.2 TTL输入电平. 1 = P1. 2 施密特触发输入
1	P1S.1	**P1.1 施密特触发输入功能** 0 = P1.1 TTL输入电平. 1 = P1.1. 施密特触发输入
0	P1S.0	**P1.0 施密特触发输入功能** 0 = P.0 TTL输入电平 1 = P1.0. 施密特触发输入

P2S – 端口 2 施密特触发输入

7	6	5	4	3	2	1	0
-	P2S.6	P2S.5	P2S.4	P2S.3	P2S.2	P2S.1	P2S.0
-	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: B5H, 页: 1　　复位值: 0000 0000b

位	名称	描述
n	P2S.n	**P2.n 施密特触发输入选择** 0 = P2.n. TTL电平输入 1 = P2.n. 施密特触发输入

P3S – 端口 3施密特触发输入

7	6	5	4	3	2	1	0
P3S.7	P3S.6	P3S.5	P3S.4	P3S.3	P3S.2	P3S.1	P3S.0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: ACH, 页: 1　　复位值: 0000 0000b

位	名称	描述
n	P3S.n	**P3.n 施密特触发输入选择** 0 = P3.n. TTL电平输入 1 = P3.n. 施密特触发输入

8.6.4 输出斜率控制

N76E885可单独控制管脚输出斜率。默认情况下，管脚采用普通斜率模式。当用户切换到高速斜率模式时，每个管脚斜率可看到显著变化。注更改PxSR寄存器需要将SFR切换到页1

P0SR – 端口 0 斜率控制

7	6	5	4	3	2	1	0
P0SR.7	P0SR.6	P0SR.5	P0SR.4	P0SR.3	P0SR.2	P0SR.1	P0SR.0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: B2H, 页: 1　　复位值: 0000 0000b

位	名称	描述
n	P0SR.n	**P0.n 斜率控制** 0 = P0.n 普通输出 1 = P0.n 高速输出

P1SR – 端口 1 斜率控制

7	6	5	4	3	2	1	0
-	-	-	-	-	-	P1SR.1	P1SR.0
-	-	-	-	-	-	读/写	读/写

地址: B4H, 页: 1　　复位值: 0000 0000b

位	名称	描述
n	P1SR.n	**P1.n 斜率控制** 0 = P1.n 普通输出 1 = P1.n 高速输出

P2SR – 端口 2 斜率控制

7	6	5	4	3	2	1	0
-	P2SR.6	P2SR.5	P2SR.4	P2SR.3	P2SR.2	P2SR.1	P2SR.0
-	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: B6H, 页: 1　　复位值: 0000 0000b

位	名称	描述
n	P2SR.n	**P2.n 斜率控制** 0 = P2.n 普通输出 1 = P2.n 高速输出

P3SR – 端口 3 斜率控制

7	6	5	4	3	2	1	0
P3SR.7	P3SR.6	P3SR.5	P3SR.4	P3SR.3	P3SR.2	P3SR.1	P3SR.0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: ADH, 页: 1　　复位值: 0000 0000b

位	名称	描述
n	P3SR.n	**P3.n 斜率控制** 0 = P3.n 普通输出. 1 = P3.n 高速输出

9. 定时器 / 计数器 0 及 1

N76E885系列的定时器/计数器0 和 1 是两个16位定时器/计数器。 每个都由两个8位的寄存器组成16计数寄存器.对于定时器/计数器0，为高8位寄存器TH0、 低8位寄存器TL0。 同样定时器/计数器1也有两个8位寄存器， TH1 和TL1。 TCON 和 TMOD 可以配置定时器/计数器0和1的模式。

与常见的定时器/计数器相比，还有一个定时器0或定时器1的溢出端口电位翻转输出使能. 当定时器发生溢出时，端口电位翻转输出可配置成根据T0 或 T1引脚自动翻转输出。

将它们设置为定时器后，定时器将对时钟周期计数。时钟源可以是系统时钟的12分频或是系统时钟的4分频。在计数器模式下，每当检测到外部计数输入脚上的负电平跳变（T0针对定时器0，T1针对定时器1），计数寄存器的内容就会加一。T0和T1上的电平在每个机器周期的C4态被采样，如果在一个机器周期采样到高电平，在下一个机器周期采样到低电平，那么就会确认一个电平由高到低的跳变，计数器寄存器指针加一。由于需要2个机器周期来确认管脚上的电平负跳变，因此外部输入信号的最大频率是主频的24分之一。无论是定时器还是计数器，计数寄存器都在机器周期的C3态加一。因此在定时器模式下，在T0 和T1 脚上检测到的电平负跳变会在紧跟着检测到该电平跳变后的那个机器周期中使计数器加1。

TMOD寄存器中的“C/$\overline{T}$”位决定工作在定时器模式还是计数器模式。每个定时器/计数器都有它自己的模式选择位；TMOD中用第2位选择定时器/计数器0 的功能、第6位来选择定时器/计数器1的功能。此外每个定时器/计数器都可以选定4种运行方式中的一种来运行。由TMOD中的M0 和M1位来选择定时器的工作模式.

N76E885系列可以像标准8051/52家族一样，计数速率为时钟的1/12，或进行快速模式, 计数速率为时钟的1/4。 速率由CKCON的 T0M 和 T1M 位控制，在使用标准8051/52速率时，默认值为0。

TMOD – 定时器 0 及 1 模式寄存器

7	6	5	4	3	2	1	0
GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: 89H　　复位值: 0000 0000b

位	名称	描述
7	GATE	**定时器1门选择** 0 = 无论$\overline{INT1}$ 的逻辑为何，当TR1为1时，开始计数 1 = 仅当$\overline{INT1}$ 为1时， TR1同时为1，才开始计数

位	名称	描述
6	C/$\overline{T}$	**定时器 1 计数器/定时器选择.** 0 = 定时器1随内部时钟而递增. 1 = 定时器1 随外部引脚T1的下降沿递增
5	M1	**定时器1模式选择** **M1 M0 定时器1 模式** 0 0 模式0: 8位定时器/计数器带5位预分频 TL1[4:0]) 0 1 模式1: 16位定时器/计数器 1 0 模式2: 8位定时器/计数 带自动从TH1重载模式 1 1 模式3: 定时器1停止
4	M0	
3	GATE	**定时器0门选择** 0 = 无论$\overline{INT0}$ 的逻辑为何，当TR0为1时，开始计数 1 = 仅当$\overline{INT0}$ 为1时， TR0同时为1，才开始计数
2	C/$\overline{T}$	**定时器 0 计数器/定时器选择.** 0 = 定时器0随内部时钟而递增. 1 = 定时器0 随外部引脚T1的下降沿递增
1	M1	**定时器0模式选择** **M1 M0 定时器0 模式** 0 模式0: 8位定时器/计数器带5位预分频 (TL1[4:0]) 0 1 模式1: 16位定时器/计数器 1 0 模式2: 8位定时器/计数器带自动从TH0重载模式在 1 1 Mode 3: TL0 / TH1 分别单独作为一个8位定时器/计数器
0	M0	

TCON – 定时器 0 和 1 控制位 (可位寻址)

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
读/写	读/写	读/写	读/写	R (电平) 读/写 (边沿)	读/写	R (电平) 读/写 (边沿)	读/写

地址: 88H 复位值: 0000 0000b

位	名称	描述
7	TF1	**定时器 1 溢出标志.** 在定时器1溢出时该位置1。当程序响应定时器1中断执行相应的中断服务程序时，该位自动清0。软件也可对其写1或写0
6	TR1	**定时器 1 启动控制.** 0 = 定时器1 中止. 清该位将中止定时器1和当前计数将保存在TH1 和 TL1. 1 = 使能定时器1.
5	TF0	**定时器 0溢出标志.** 在定时器0溢出时该位置1。当程序响应定时器0中断执行相应的中断服务程序时，该位自动清0。软件也可对其写1或写0
4	TR0	**定时器 0 启动控制.** 0 = 定时器0 中止. 清该位将中止定时器0和当前计数将保存在TH0和 TL0. 1 = 使能定时器0.

TL0 –定时器0 低字节

7	6	5	4	3	2	1	0
TL0[7:0]							
r/w							

地址： 8AH 复位值：0000 0000B

位	名称	描述
7:0	TL0[7:0]	**定时器 0 低字节** 寄存器TL0是定时器0的16位数值的低字节

TH0 –定时器0高字节

7	6	5	4	3	2	1	0
TH0[7:0]							
r/w							

地址： 8CH 复位值：0000 0000B

位	名称	描述
7:0	TH0[7:0]	**定时器 0 高字节** 寄存器TH0是定时器0的16位数值的高字节

TL1 – 定时器1低字节

7	6	5	4	3	2	1	0
TL1[7:0]							
r/w							

地址： 8BH 复位值：0000 0000B

位	名称	描述
7:0	TL1[7:0]	**定时器 1 低字节** 寄存器TL1是定时器1的16位数值的低字节

TH1 –定时器1高字节

7	6	5	4	3	2	1	0
TH1[7:0]							
r/w							

地址： 8DH 复位值：0000 0000B

位	名称	描述
7:0	TH1[7:0]	**定时器 1 高字节** 寄存器TH1是定时器1的16位数值的高字节

CKCON – 时钟控制寄存器

7	6	5	4	3	2	1	0
-	PWMCKS	-	T1M	T0M	-	-	-
-	读/写	-	读/写	读/写	-	-	-

地址: 8EH 复位值: 0000 0000b

位	名称	描述
4	T1M	**定时器 1 时钟选择:** 0: 定时器 1 的时钟选择为 1/12 系统时钟. 1: 定时器1的时钟选择为1/4系统时钟.
3	T0M	**定时器 0 时钟选择:** 0: 定时器 0 的时钟选择为 1/12 系统时钟. 1: 定时器0的时钟选择为1/4系统时钟

AUXR1 – 辅助功能寄存器

7	6	5	4	3	2	1	0
SWRF	RSTPINF	T1LXTM	T0LXTM	GF2	UART0PX	0	DPS
读/写	读/写	读/写	读/写	读/写	读/写	R	读/写

地址: A2H 复位值: 详见 表 6–2. SFR定义及复位值

位	名称	描述
5	T1LXTM	**定时器 1 LXT 输入模式** 0 = 定时器 1时钟源由C/$\overline{T}$ (TMOD.6) 及 T1M (CKCON.4)配置 1 = 定时器 1时钟源为LXT.
4	T0LXTM	**定时器 0 LXT输入模式** 0 = 定时器 0 时钟源由 C/$\overline{T}$ (TMOD.2) 及 T0M (CKCON.3)配置 1 = 定时器 0 时钟源为 LXT.

P1M1 – 端口 1 模式选择1

7	6	5	4	3	2	1	0
-	-	-	-	T1OE	T0OE	P1M1.1	P1M1.0
-	-	-	-	读/写	读/写	读/写	读/写

地址: B3H 复位值: 0000 0011b

位	名称	描述
3	T1OE	**定时器 1 输出使能** 0 = 定时器1输出关闭. 1 = 定时器1 从T1脚输出. 定时器1,只有配置位定时器模式才可由T1输出。.
2	T0OE	**定时器 0输出使能** 0 = 定时器 0 输出关闭. 1 = 定时器 0 从T0脚输出. 定时器1,只有配置位定时器模式才可由T0输出

9.1 模式 0 (13位定时器)

在模式 0, 定时器/计数器是13位的计数器。13位的计数器由TH0 (TH1) 和 TL0 (TL1)的低五位组成。TL0 (TL1)的高三位被忽略。当TR0 (TR1)置位且GATE是0或$\overline{INT0}$ ($\overline{INT1}$) 是1定时器/计数器使能。门设置为1允许定时器通过外部输入引脚$\overline{INT0}$ ($\overline{INT1}$)计算脉冲的宽度。当13位的定时器计数值变为1FFFH后，下一次计数会使其变为0000H。定时器溢出标志TF0 (TF1) 置位 ，如果中断打开，此时还会产生一个定时器中断。.

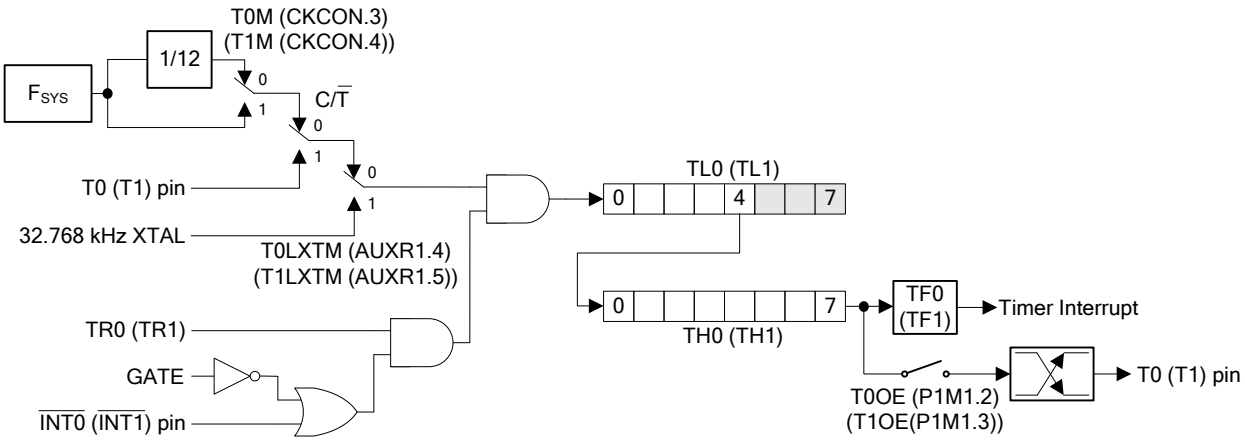


图 9-1. 定时器/计数器0和1 模式 0

9.2 模式 1 (16位定时器)

模式1与模式0 非常相似，只是模式1下定时器/计数器为16位的，而非13位。就是说是用THx和TLx的全部16位来计数。当计数值由FFFFH向0000H翻转后，相应的溢出标志TFx置1，如果中断使能产生中断。

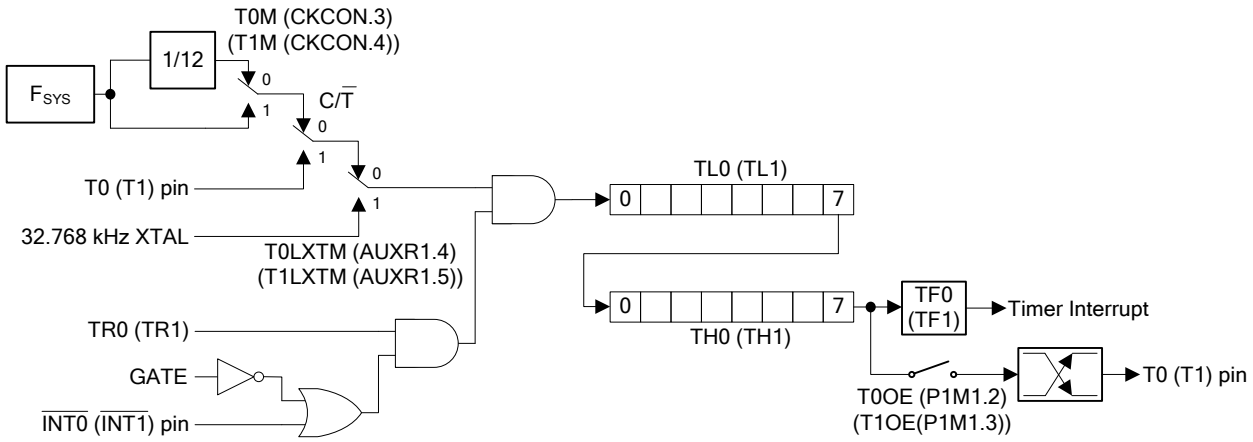


图 9-2. 定时器/计数器0和1 模式1

9.3 模式 2 (8位自动重装载定时器)

模式2下定时器/计数器为自动重装模式。此模式下TLx是一个8位的计数器，THx保存重装计数值。当TLx由FFH向00H溢出后，TCON中的TFx标志置位THx中内容重装至TLx，继续计数过程。重装过程中THx内的值保持不变.该特征最好地适用于UART波特率发生器，不需要连续软件介入. 注：仅有定时器1可以用作UART的波特率源.正确设置GATE和INTx 引脚及TRx位，使能计数. GATE 和INTx 引脚的功能与模式0和1相同。

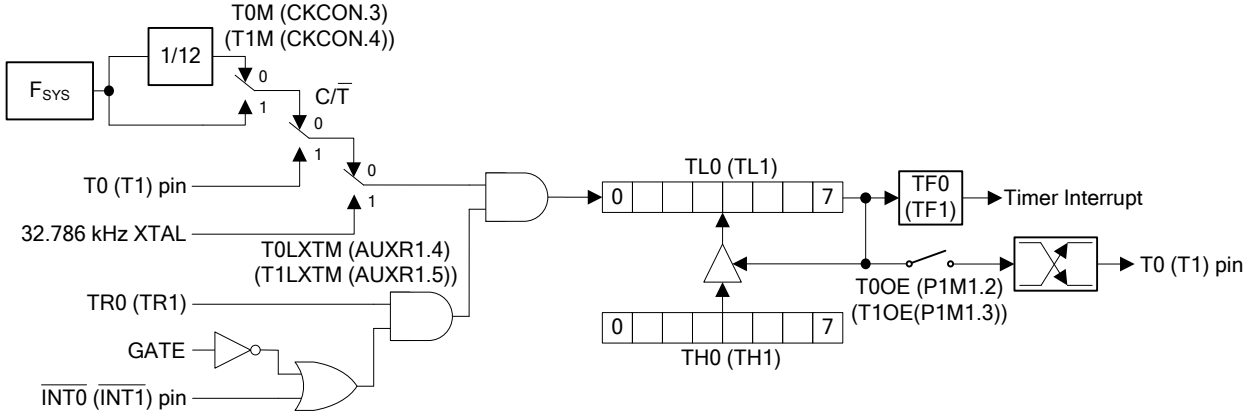


图 9-3. 定时器/计数器0和1 模式2

9.4 模式 3 (两组独立8位定时器)

模式3 有着不同的工作方式。对定时器/计数器1 来说模式3 会将其停止；对定时器/计数器0来说模式3下TL0 和TH0是2个独立的8位计数寄存器。模式3下TL0用定时器0的控制位：如C/T、GATE、 TR0、

INT0和TF0。TL0也可以用来对T0 脚上的1到0 跳变计数，由C/$\overline{T}$ (TMOD.2).来决定。TH0 只能对内部时钟源计数，并使用定时器/计数器1的控制位（TR1和TF1）。当需要额外的8位定时器时可以使用模式3 。 当定时器0处于模式3时，定时器1可以通过将其放入或离开模式3的方式来打开或关闭它。定时器1依然可以工作在模式0、1、2下，但它的灵活性受到限制。虽然基本功能得以维持，但已不能对TF1和TR1进行控制。此时定时器1依然可以使用GATE及INT1脚、T1M, 和 T1LXTM。它同样可以用作串行口的波特率发生器或其他不需要中断的应用。

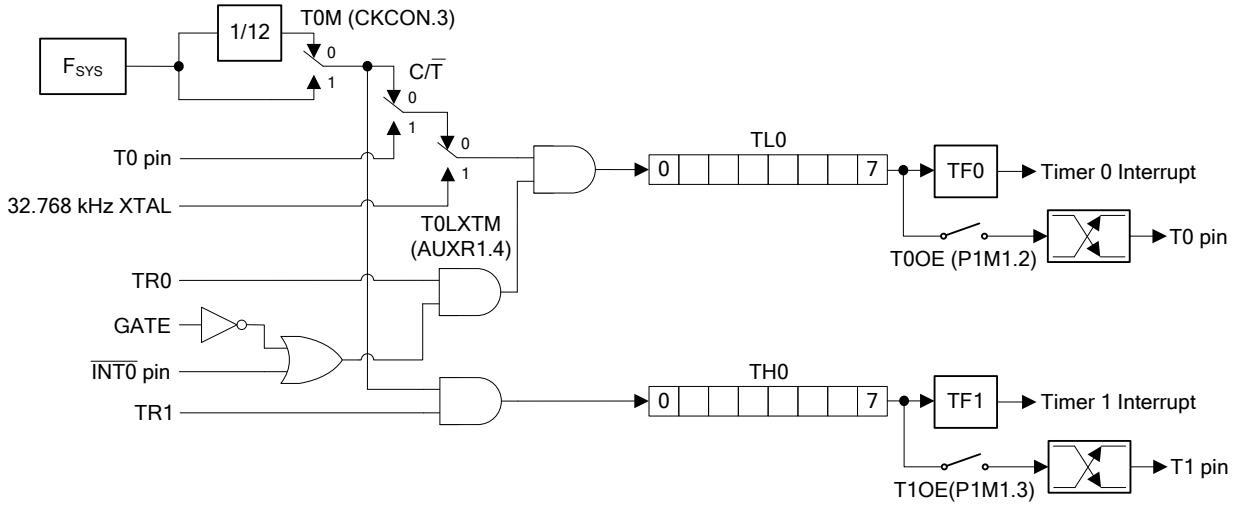


图 9-4. 定时器/计数器0和1 模式3

10. 定时器2及输入捕获

定时器2模块分成四个一样的定时器：定时器2A, 定时器2B, 定时器2C, 和定时器2D。每一个定时器通过简单的自动重载、下数的计数器实现。注意每一个定时器有相同的结构和功能。他们中的每一个可以通过它们的控制位单独的控制。每一个定时器支持两种操作模式：自动重载模式和PWM模式。通过T2MOD0 或 T2MOD1寄存器的 T2xM来选择。用户可以通过T2xPS[2:0]选择预分频值 。有两个输出引脚产生互补的50%占空比周期或PWM波形。在下面描述和图形中，用定时器2A来说明，其他三个定时器和2A一样。

定时器2为一个16位计数器通过高位数据寄存器TH2和地位数据寄存器TL2,组成。同时搭配寄存器RCMP2H 及 RCMP2L，配置CM/$\overline{RL2}$ (T2CON.0)位后，定时器2可工作在比较模式及自动重装载模式下。定时器2具有3通道输入捕获模式，可测量脉冲宽度。捕获结果存放在寄存器C0H 及 C0L, C1H 及 C1L, C2H 及 C2L中。定时器2的时钟源为系统时钟除频，总共具有8级分频，适用于更多应用需求。当TR2 (T2CON.2) 置 1，定时器使能。下列寄存器用于控制定时器2。

T2CON – 定时器 2 控制寄存器

7	6	5	4	3	2	1	0
TF2	-	-	-	-	TR2	-	CM/$\overline{RL2}$
读/写	-	-	-	-	读/写	-	读/写

地址: C8H　　复位值: 0000 0000b

位	名称	描述
7	TF2	**定时器 2 溢出标志** 当定时器2溢出或者比较数据符合，该位置1.。如果已使能定时器2中断，该位置1会让芯片进入定时器2中断运行。该位硬件置1，但需要软件清0。
2	TR2	**定时器2 运行控制** 0 = 定时器计数关闭。清除该位关闭定时器计数，目前已计数值会保留在TH2及TL2中 1 = 定时器2计数使能.
0	CM/$\overline{RL2}$	**定时器 2 比较及自动重装载功能选择** 0 = 自动重装载模式 1 = 比较器模式.

T2MOD – 定时器2模式选择

7	6	5	4	3	2	1	0
LDEN	T2DIV[2:0]			CAPCR	CMPCR	LDTS[1:0]	
读/写	读/写			读/写	读/写	读/写	

地址: C9H 复位值: 0000 0000b

位	名称	描述
7	LDEN	**自动重装载使能位** 0 = 自动重装载功能关闭 1 = 将RCMP2H及RCMP2L自动重装载至TH2及TL2功能使能
6:4	T2DIV[2:0]	**定时器 2时钟除频** 000 = 1/1. 001 = 1/4. 010 = 1/16. 011 = 1/32. 100 = 1/64. 101 = 1/128. 110 = 1/256. 111 = 1/512.
3	CAPCR	**捕获模式自动清除** 该位仅当定时器设定位自动重装载模式下有效。该位使能，当捕获完成，TH2及TL2内数据移入RCMP2H 及 RCMP2L后，自动清除TH2及TL2计数寄存器功能。 0 = 捕获完成后定时器2计数按之前计数值继续累加。 1 = 捕获完成后定时2数据自动清0
2	CMPCR	**比较完成自动清除** 该位仅当定时器2比较功能模式下有效。当比较符合后，自动清除TH2及TL2计数器值. 0 = 比较符合之后，定时器2内数据按之前继续计数。 1 = 比较符合之后，定时器2内数据清0。
1:0	LDTS[1:0]	**自动重装载触发选择** 00 = 当定时器2溢出自动重装载 01 = 当捕获0通道事件完成，自动重装载 10 = 当捕获1通道事件完成，自动重装载 11 = 当捕获2通道事件完成，自动重装载

RCMP2L – 定时器 2 重装载 / 比较数据低字节

7	6	5	4	3	2	1	0
RCMP2L[7:0]							
读/写							

地址: CAH 复位值: 0000 0000b

位	名称	描述
7:0	RCMP2L[7:0]	**定时器 2 重装载/比较器低字节** 当定时器2设定位比较模式，预存放低字节待比较数据。 当设定位自动重装载，与存放低字节数据。

RCMP2H –定时器 2 重装载 / 比较数据高字节

7	6	5	4	3	2	1	0
RCMP2H[7:0]							
读/写							

地址: CBH 复位值: 0000 0000b

位	名称	描述
7:0	RCMP2H[7:0]	**定时器 2 重装载/比较器高字节** 当定时器2设定位比较模式，预存放高字节待比较数据。 当设定位自动重装载，与存放高字节数据。

TL2 – 定时器2低字节

7	6	5	4	3	2	1	0
TL2[7:0]							
读/写							

地址: CCH 复位值: 0000 0000b

位	名称	描述
7:0	TL2[7:0]	**定时器 2 低字节数据** 该寄存器存放16位定时器2，实际计数的低8位字节数据。

TH2 – 定时器2高字节

7	6	5	4	3	2	1	0
TH2[7:0]							
读/写							

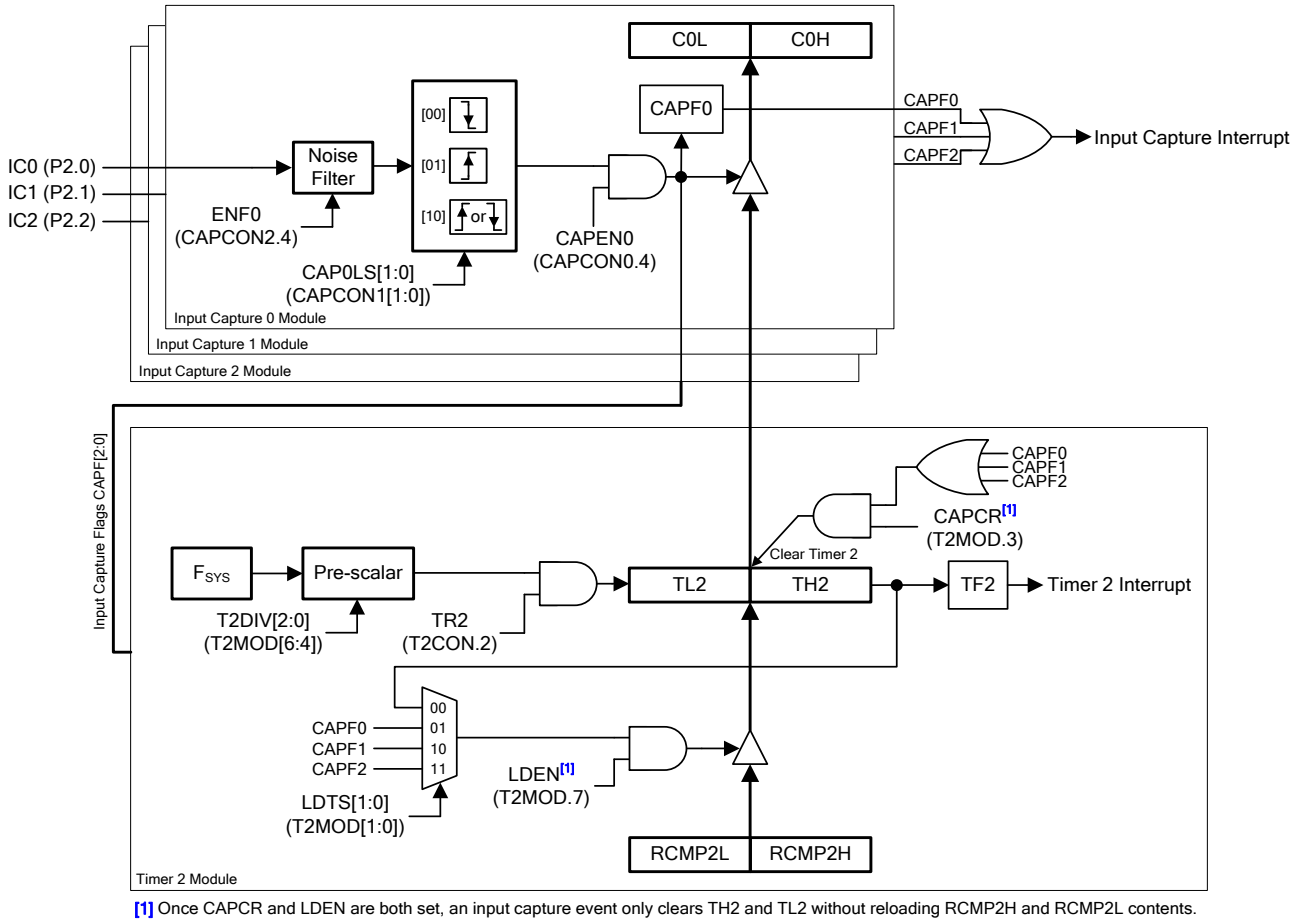
地址: CDH 复位值: 0000 0000b

位	名称	描述
7:0	TH2[7:0]	**定时器 2 高字节数据** 该寄存器存放16位定时器2，实际计数的高8位字节数据。

由于TH2和TL2分在两个寄存器内，强烈建议软件在停止定时器2工作之后再更改这两个寄存器的值。如果还在运行过程中对寄存器直接改动，可能引发无法预测的结果。

10.1 自动重装载功能

当CM/$\overline{RL2}$ 清0，定时器2配置为自动重装载模式 。在该模式下，当LDEN使能后，每次计数完成，自动将 RCMP2H 及 RCMP2L 寄存器内的值，写入TH2 及 TL2 中。根据LDTS[1:0] (T2MOD[1:0])配置，触发可以是所选通道定时器2溢出或是捕获完成。注，一旦CAPCR (T2MOD.3) 置1，捕获完成后，仅清除 TH2 及 TL2 内的值，不会将 RCMP2H 及 RCMP2L的值载入.



[1] Once CAPCR and LDEN are both set, an input capture event only clears TH2 and TL2 without reloading RCMP2H and RCMP2L contents.

图 10-1. 定时器2自动重装载机及捕获模式功能模块图

10.2 比较模式

当CM/$\overline{RL2}$ 置1，定时器2配置为比较器模式。在该模式下，RCMP2H 及 RCMP2L 预存待比较数据。由于定时器2向上计数，一旦TH2 及 TL2 符合 RCMP2H 及 RCMP2L设定值, TF2 (T2CON.7) 将会硬件置1，用以标示两组数据已相符。

对 CMPCR (T2MOD.2) 置1，当比较结果相符后，会直接清0 定时器2 计数器。

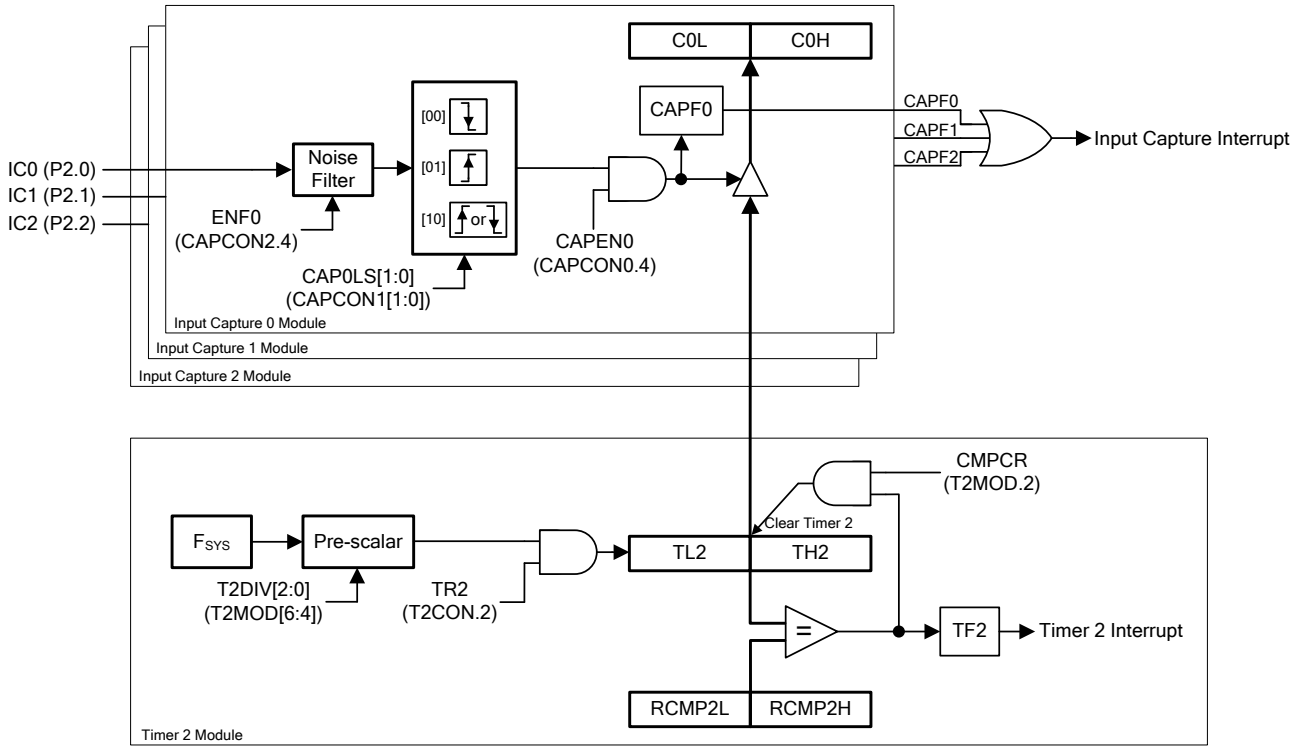


图 10-2. 定时器2比较模式输入与捕获模式结构功能图

10.3 输入捕获模式

定时器2的输入捕获模式. 定时器2通过 CP/$\overline{RL2}$和 LDEN位清零进入输入捕获模式。 输入捕获模块通过寄存器CAPCON0~2配置，输入捕获模块支持3通道输入(IC0至IC2管脚)，与普通端口共用的 P2.0至P2.2 每个输入通道组成自己的史密特触发输入。 每个通道的噪声滤波器通过设置ENF0~2 (CAPCON2[6:4])使能， 可滤除小于4个CPU时钟的输入毛刺。输入捕获0~2有独立的边沿检测与定时器2的触发边沿共享(通过对CAPCON1配置)支持正边沿捕获，负边沿捕获，或双边沿捕获。 每个输入捕获通道有自己的使能位CAPEN0~2 (CAPCON0[6:4])。

当使能任何输入捕获通道和所选择的边沿触发发生时，定时器2的计数值TH2和TL2将被捕获、传输并存储到捕获寄存器CnH 和 CnL。 边沿触发也可硬件方式使CAPFn (CAPCON0.n)置1。 如果ECPTF (EIE.7)和EA都打开，将产生中断。 三个输入捕获标志共享中断向量, 用户应该检查CAPFn以确定哪个通道有输入捕获。 这些标志必须由软件清零。

CAPCR (T2MOD.3)用于周期计算。设置CAPCR为1，可以硬件上达到TH2与TL2的值已经被捕获后自动清定时器2为0000H功能，可以省去软件循环清除标志的步骤。

CAPCON0 – 输入捕获控制寄存器0

7	6	5	4	3	2	1	0
-	CAPEN2	CAPEN1	CAPEN0	-	CAPF2	CAPF1	CAPF0
-	读/写	读/写	读/写	-	读/写	读/写	读/写

地址: 92H　　复位值: 0000 0000b

位	名称	描述
6	CAPEN2	**使能输入捕获通道2使能位** 0 = 关闭输入捕获通道2. 1 = 打开输入捕获通道2.
5	CAPEN1	**使能输入捕获通道1使能位** 0 = 关闭输入捕获通道1. 1 = 打开输入捕获通道1.
4	CAPEN0	**使能输入捕获通道0使能位** 0 = 关闭输入捕获通道0. 1 = 打开输入捕获通道0
2	CAPF2	**输入捕获通道2标志位** 如果输入捕获2边沿发生，该位由硬件置位，由软件清零
1	CAPF1	**输入捕获通道1标志位** 如果输入捕获1边沿发生，该位由硬件置位，由软件清零
0	CAPF0	**输入捕获通道0标志位** 如果输入捕获0边沿发生，该位由硬件置位，由软件清零

CAPCON1 – 输入捕获控制寄存器1

7	6	5	4	3	2	1	0
-	-	CAP2LS[1:0]		CAP1LS[1:0]		CAP0LS[1:0]	
-	-	读/写		读/写		读/写	

地址: 93H　　复位值: 0000 0000b

位	名称	描述
5:4	CAP2LS[1:0]	**输入捕获通道2条件选择.** 00 = 下降沿. 01 = 上升沿. 10 = 上升沿或下降沿. 11 = 保留.
3:2	CAP1LS[1:0]	**输入捕获1电平条件选择** 00 = 下降沿. 01 = 上升沿. 10 = 上升沿或下降沿. 11 = 保留
1:0	CAP0LS[1:0]	**输入捕获通道0条件选择** 00 = 下降沿. 01 = 上升沿. 10 = 上升沿或下降沿. 11 = 保留.

CAPCON2 –输入捕获控制寄存器2

7	6	5	4	3	2	1	0
-	ENF2	ENF1	ENF0	-	-	-	-
-	读/写	读/写	读/写	-	-	-	-

地址: 94H　　复位值: 0000 0000b

位	名称	描述
6	ENF2	**输入捕获通道2 噪声滤波使能位** 0 = 关闭输入捕获通道2的噪声滤波 1 = 打开输入捕获通道2的噪声滤波
5	ENF1	**输入捕获通道1 噪声滤波使能位** 0 = 关闭输入捕获通道1的噪声滤波 1 = 打开输入捕获通道1的噪声滤波
4	ENF0	**输入捕获通道0 噪声滤波使能位** 0 = 关闭输入捕获通道0的噪声滤波 1 = 打开输入捕获通道0的噪声滤波

C0L – 捕获通道0低字节

7	6	5	4	3	2	1	0
C0L[7:0]							
读/写							

地址: E4H　　复位值: 0000 0000b

位	名称	描述
7:0	C0L[7:0]	**捕获通道0 输入结果低字节** 寄存器C0L是16位捕获通道0输入结果的低字节值

C0H – 捕获通道0高字节

7	6	5	4	3	2	1	0
C0H[7:0]							
读/写							

地址: E5H　　复位值: 0000 0000b

位	名称	描述
7:0	C0H[7:0]	**捕获通道0 输入结果高字节** 寄存器C0H是16位捕获通道0输入结果的高字节值

C1L – 捕获通道1低字节

7	6	5	4	3	2	1	0
C1L[7:0]							
读/写							

地址: E6H 复位值: 0000 0000b

位	名称	描述
7:0	C1L[7:0]	**捕获通道1 输入结果低字节** 寄存器C1L是16位捕获通道0输入结果的低字节值

C1H – 捕获通道1高字节

7	6	5	4	3	2	1	0
C1H[7:0]							
读/写							

地址: E7H 复位值: 0000 0000b

位	名称	描述
7:0	C1H[7:0]	**捕获通道1 输入结果高字节** 寄存器C1H是16位捕获通道0输入结果的高字节值

C2L – 捕获通道2低字节

7	6	5	4	3	2	1	0
C2L[7:0]							
读/写							

地址: EDH 复位值: 0000 0000b

位	名称	描述
7:0	C2L[7:0]	**捕获通道2 输入结果低字节** 寄存器C2L是16位捕获通道0输入结果的低字节值

C2H – 捕获通道2高字节

7	6	5	4	3	2	1	0
C2H[7:0]							
读/写							

地址: EEH 复位值: 0000 0000b

位	名称	描述
7:0	C2H[7:0]	**捕获通道2 输入结果高字节** 寄存器C2H是16位捕获通道0输入结果的高字节值

11. 定时器 3

定时器 3是一个补充的16位自动重载上数定时器。用户可以通过T3PS[2:0] (T3CON[2:0])选择预分频，并填重载值到R3H 和 R3L寄存器来决定它的溢出速率。用户可以设置TR3 (T3CON.3)来开始计数。当计数跨过FFFFH，TF3 (T3CON.4)置为1，且R3H 和 R3L寄存器的内容重载到内部16位计数器。如果ET3 (EIE1.1)置为1，定时器3中断服务 程序被执行。当进入中断服务程序，TF3会被硬件自动清零。

定时器3同时也用作串口波特率产生定时器，详细内容请参考章节14.5 ”波特率”。

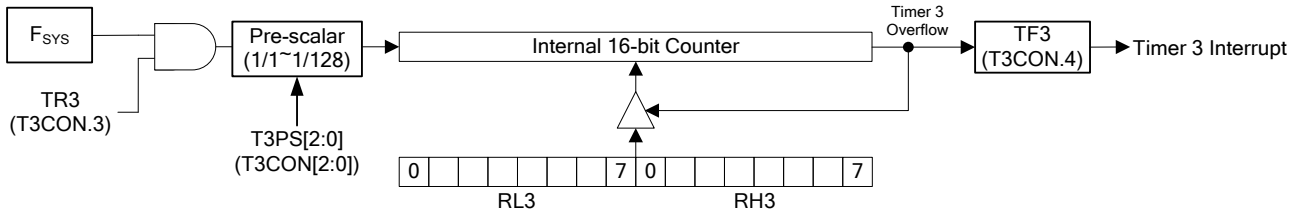


图 11-1. 定时器3结构图

T3CON – 定时器3控制寄存器

7	6	5	4	3	2	1	0
SMOD_1	SMOD0_1	BRCK	TF3	TR3	T3PS[2:0]		
读/写	读/写	读/写	读/写	读/写	读/写		

地址: C4H　　复位值: 0000 0000b

位	名称	描述
4	TF3	**定时器 3 溢出标志** 当定时器3溢出，该位置位。当程序执行定时器3的中断服务程序，该位由硬件自动清零。该位可以通过软件置位或清零。
3	TR3	**定时器 3 运行控制** 0 = 定时器 3 停止. 1 = 定时器 3 开始计数 注意重载寄存器R3H 和 R3L仅在定时器3停止(TR3 为 0)的时候才可以被写。如果TR3 位1写，结果是不可预知的。
2:0	T3PS[2:0]	**定时器 3 预分频** 这些位决定定时器3的时钟分频。 000 = 1/1. 001 = 1/2. 010 = 1/4. 011 = 1/8. 100 = 1/16. 101 = 1/32. 110 = 1/64. 111 = 1/128.

RL3 – 定时器3自动重装载寄存器低字节

7	6	5	4	3	2	1	0
RL3[7:0]							
读/写							

地址: C5H 复位值: 0000 0000b

位	名称	描述
7:0	RL3[7:0]	**定时器 3 重载低字节** 它保持着定时器3重载值的低字节

RH3 –定时器3自动重装载寄存器高字节

7	6	5	4	3	2	1	0
RH3[7:0]							
读/写							

地址: C6H 复位值: 0000 0000b

位	名称	描述
7:0	RH3[7:0]	**定时器 3 重载高字节** 它保持着定时器3重载值的高字节

12. 看门狗定时 (WDT)

N76E885提供一组看门狗定时器(WDT)，它可以配置成一个超时复位定时器用于复位整个设备。一旦由于外界干扰设备进入非正常状态或挂起，看门狗可以复位恢复系统。 用于监测系统以提高系统可靠性。对于容易受到噪声，电源干扰或静电放电干扰的系统，是十分有用的工具。看门狗也可以配置成通用定时器，它的周期中断服务作为事件定时器或连续系统监测，可以工作在空闲模式或掉电模式。WDTEN[3:0] (CONFIG4[7:4])初始化WDT工作在超时复位定时器或通用定时器模式。

配置位4

7	6	5	4	3	2	1	0
WDTEN[3:0]				-	-	-	-
读/写				-	-	-	-

出厂默认值: 1111 1111b

位	名称	描述
7:4	WDTEN[3:0]	**WDT 使能** 该域配置MCU执行后，WDT的动作。 1111 = WDT 禁止， WDT 可以通过软件控制用于通用定时器。 0101 = WDT 使能，作为一个超时复位定时器，且在空闲或掉电模式会停止运行。 其他 = WDT 使能，作为一个超时复位定时器，且在空闲或掉电模式会保持运行。

WDT带一个分频器用于分频10K LIRC时钟。分频器的输出可选，决定了超时间隔。当超时间隔完成，会从空闲或掉电模式唤醒系统，且如果WDT中断使能会产生一个中断事件。如果WDT初始化为一个超时复位定时器，如果没有任何软件动作，在一个延时周期后会产生系统复位。

WDCON –看门狗定时器控制寄存器 (TA 保护)

7	6	5	4	3	2	1	0
WDTR	WDCLR	WDTF	WIDPD	WDTRF[1]	WDPS[2:0][2]		
读/写	读/写	读/写	读/写	读/写	读/写		

地址: AAH 复位值: 详见 表 6–2. SFR定义及复位值

位	名称	描述
7	WDTR	**WDT 运行** 该位仅当控制位WDTEN[3:0] (CONFIG4[7:4])全为1时有效。这时WDT工作在通用定时器模式。 0 = WDT 禁止. 1 = WDT 使能. WDT 计数器开始运行。

位	名称	描述
6	WDCLR	**WDT 清除** 设置该位复位WDT计数到00H。它使计数器到一个已知的状态，防止系统出现不可预知的复位。写和读WDCLR位意思是不一样的。 写: 0 = 无影响 1 = 清 WDT 计数器. 读: 0 = WDT 计数器完全清零 1 = WDT 计数器还没有清零.
5	WDTF	**WDT 超时标志** 该位表示WDT计数器的溢出。该标志应该通过软件清零。.
4	WIDPD	**WDT 工作在空闲或掉电模式** 该位仅当控制位WDTEN[3:0] (CONFIG4[7:4])全为1时有效。它决定WDT作为通用定时器在空闲或掉电模式下是否保持工作。 0 = WDT 在空闲或掉电模式下停止工作 1 = WDT 在空闲或掉电模式下保持工作.
3	WDTRF	**WDT 复位标志** 当CPU通过WDT超时事件复位，该位会由硬件置位。该位建议复位之后通过软件清零。
2:0	WDPS[2:0]	**WDT 时钟预分频选择** 这些位决定了WDT时钟的预分频，从1/1 到 1/256。见 表 12–1. 默认是最大分频值。

[1] WDTRF 在上电复位之后会被清零，在WDT复位之后置位，在其他任何复位之后保持不变。
[2] WDPS[2:0] 在上电复位之后全部置位，在其他任何复位之后保持不变。

看门狗定时器溢出时间间隔算式为 $\frac{1}{F_{LIRC} \times clock\,divider\,scalar} \times 64$,决定， F_{LIRC} 是内部震荡10 kHz的频率。下表所示不同的超时间隔的例子：

表 12–1.看门狗定时器溢出及分频列表

WDPS.2	WDPS.1	WDPS.0	时钟除频	看门狗溢出时间 (F_{LIRC} ~= 10 kHz)
0	0	0	1/1	6.40 ms
0	0	1	1/4	25.60 ms
0	1	0	1/8	51.20 ms
0	1	1	1/16	102.40 ms
1	0	0	1/32	204.80 ms
1	0	1	1/64	409.60 ms
1	1	0	1/128	819.20 ms
1	1	1	1/256	1.638 s

12.1 超时复位定时器

当 CONFIG 位 WDTEN[3:0] (CONFIG4[7:4]) 不是 FH,，WDT是初始化为一个超时复位定时器。如果 WDTEN[3:0] 不是 5H， WDT在系统进入空闲或掉电模式后允许继续运行。注意当WDT初始化为超时复位定时器，WDTR 和 WIDPD 没有功能。.

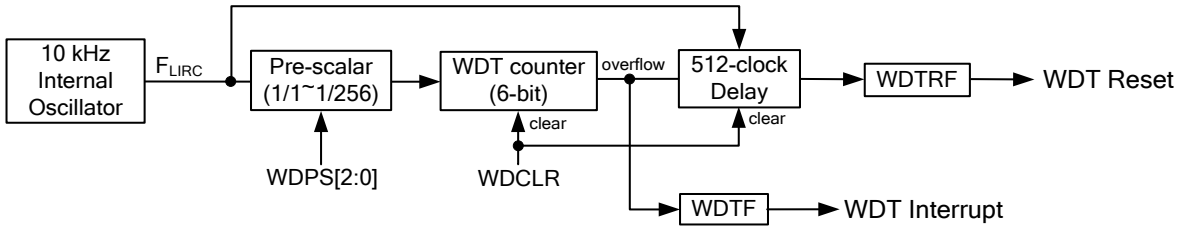


图 12-1. WDT 超时复位定时器

在设备上电后，开始执行软件代码。WDT开始计数。超时间隔通过WDPS[2:0] (WDCON[2:0])选择。当选择的超时发生，WDT会置位中断标志WDTF (WDCON.5)。如果WDT中断使能位EWDT (EIE.4)和全局中断使能EA都置位，WDT中断程序被执行。同时一个额外的512个LIRC时钟延时用于计数器的清零来避免系统被WDT复位。如果在这512个时钟内没有写1到WDCLR，WDT复位会发生。置位WDCLR位用来清零WDT计数器。该位是自清零的用于用户监视它。一旦WDT复位发生。WDT复位标志WDTRF (WDCON.3)会被置位。除了上电复位之外的其他任何复位后，该位保持不变。用户可以通过软件清零WDTRF。注意WDCON的所有位需要时序访问写。

看门狗定时器复位的主要应用是系统监测， 这对于实时控制很重要，适用于电磁干扰等避免发生程序跑飞等场合，或在未知状态发生时保护用户的代码。 使用看门狗定时器 用户可选择理想的看门狗复位看

门狗定时时间。设定 WCLR, 可使代码继续运行而无看门狗定时器复位。如果代码运行在错误的状态下，无法及时清看门狗定时器，将引起芯片复位，使系统从错误的状态恢复过来。

12.2 用作通用定时器

看门狗定时器的另一个应用是用作简单的定时器。当CONFIG 位 WDTEN[3:0] (CONFIG4[7:4]) 是 FH。WDT初始化为通用定时器。在这种模式下WDTR 和 WIDPD 是完全可以通过软件访问的。

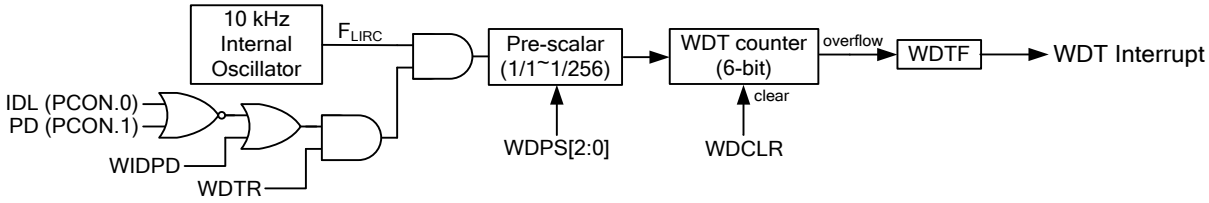


图 12-2. 看门狗定时器结构图

WDT通过设置WDTR为1开始运行，通过清零WDTR停止。当WDT选择的时间间隔到后，WDTF标志会置位。软件查询WDTF标志来侦测超时。如果单独的中断EWDT (EIE.4)和全局中断EA置位，WDT会产生中断。WDT会继续计数，用户应该清零WDTF并等待下一次溢出，通过查询WDTF标志或等待中断发生。

在一些功耗的应用中，CPU常在没有处理事件时处于空闲模式或掉电模式，需要定时唤醒察看是否需要响应，而定时器0到3打开后耗电量将达到毫安(mA)级，为了达到系统需要保持在微安(μA)级的要求，当没有事情需要做的时候，CPU应该停留在掉电模式。并且可以通过可编程的时候间隔唤醒。N76E885配备了很有用的WDT唤醒功能，由于基于内部10kHz的RC时钟源，看门狗定时器功耗非常低，它能够在掉电模式下计数并唤醒CPU。例程如下：

```
        ORG    0000H
        LJMP   START

        ORG    0053H
        LJMP   WDT_ISR

        ORG    0100H
;*******************************************************************
;WDT interrupt service routine
;*******************************************************************
WDT_ISR:
        CLR    EA
        MOV    TA,#0AAH
        MOV    TA,#55H
        ANL    WDCON,#11011111B    ;clear WDT interrupt flag
        SETB   EA
        RETI

;*******************************************************************
;Start here
;*******************************************************************
START:
        MOV    TA,#0AAH
        MOV    TA,#55H
        ORL    WDCON,#00010111B    ;choose  interval  length  and  enable  WDT  running
during                                  ;Power-down
        SETB   EWDT                ;enable WDT interrupt
        SETB   EA

        MOV    TA,#0AAH
        MOV    TA,#55H
        ORL    WDCON,#10000000B    ; WDT run

;*******************************************************************
;Enter Power-down mode
;*******************************************************************
LOOP:
        ORL    PCON,#02H
        LJMP   LOOP
```

13. 自唤醒定时器(WKT)

N76E885 有一个专用的自唤醒定时器（WKT），用于低功耗模式下的周期唤醒芯片，也可用作通用定时器。WKT保持计数在空闲或掉电模式。当WKT用于唤醒定时器，WKT要在进入掉电之前开启。WKT有两个时钟源，LIRC或LXT，由WKTCKS (WKCON.5)位决定。注意系统时钟频率必须大于WKT时钟两倍以上。如果WKT开始计数，在设备进入空闲或掉电模式下，选择的时钟源会保持工作。注意选择的WKT时钟源不会连同WKT的配置自动使能。用户应该手动使能选择的时钟源并等待它稳定确保操作的成功。

WKT配备一个简单的8位自动重载上数定时器。它的预分频从1/1 到 1/512，通过WKPS[2:0] (WKCON[2:0]).来选择。用户填重载值到RWK寄存器来决定它的溢出速率。WKTR (WKCON.3)置位开始计数。当计数器溢出，WKTF (WKCON.4)置位为1，并重载RWK寄存器的值到内部8为计数器。如果EWKT (EIE1.2)置为1，WKT中断服务程序被执行。

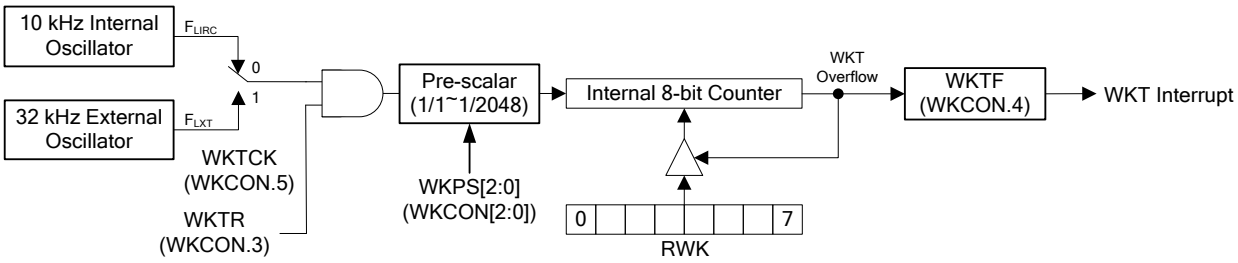


图 13-1. 自唤醒定时器结构图

WKCON – 自唤醒定时器控制寄存器

7	6	5	4	3	2	1	0
-	-	WKTCK	WKTF	WKTR	WKPS[2:0]		
-	-	读/写	读/写	读/写	读/写		

地址: 8FH　　复位值: 0000 0000b

位	名称	描述
5	WKTCK	**WKT 时钟源选择** 0 = LIRC. 1 = LXT. 注意当WKT在运行中的时候，该位不能切换。它需要在WKTR设置为1之前选择。
4	WKTF	**WKT 溢出标志** 当WKT溢出，该位置位。如果WKT中断和全局中断使能，置位该位会使CPU执行WKT中断服务程序。该位不会被硬件自动清零，应该通过软件清零。

位	名称	描述
3	WKTR	**WKT 运行控制** 0 = WKT 停止. 1 = WKT 开始运行 注意重载寄存器RWK仅在WKT停止(WKTR 为 0).的时候可以写入。否则结果是不可预知的。
2:0	WKPS[2:0]	**WKT 预分频** 这些位决定 WKT 时钟的预分频. 000 = 1/1. 001 = 1/4. 010 = 1/16. 011 = 1/64. 100 = 1/256. 101 = 1/512. 110 = 1/1024. 111 = 1/2048.

RWK – 自唤醒定时器重装载数据寄存器

7	6	5	4	3	2	1	0
RWK[7:0]							
读/写							

地址: 86H　　复位值: 0000 0000b

位	名称	描述
7:0	RWK[7:0]	**WKT 重载字节** 用以保存WKT的8位重载值。注意如果预分频是1/1，RWK限制不能是FFH。

14. 串口 (UART)

N76E885有两个具备地址自动识别和帧错功能的全双工串口。两个串口的功能是一样的，为了区分两个串口控制位，串口1的控制位以“_1”结尾（例如SCON_1）。下述详例以串口0为例。

每个串口都有一种同步工作模式：模式0。三种全双工异步模式：模式1，2，和3——这意味着收发可以同时连续进行。 接收是双缓存的，收到的一个字节在读走前，可以开始接收下一个字节。这两个缓存共用一个寄存器地址 SBUF。对SBUF写入数据用于发送，接收数据也从寄存器SBUF中读取。串口共有4种模式，任何一种模式都由SBUF发起指令。注意，在使用串口功能前，串口所用管脚P2.0 及 P0.3 (RXD 及 TXD) 或者 P2.4 及 P2.5 (RXD_1 及 TXD_1) 必须先送1。N76E855提供更灵活的管脚配置，可将串口0的TXD及RXD通过UART0PX (AUXR1.2).更改位置。

SCON – 串口0控制寄存器 (可位寻址)

7	6	5	4	3	2	1	0
SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: 98H　　复位值: 0000 0000b

位	名称	描述
7	SM0/FE	**串口0模式选择位** SMOD0 (PCON.6) = 0: 详见表 14–1. 串口0 模式描述 SMOD0 (PCON.6) = 1: SM0/FE 位用作帧错误 (FE) 状态标志. 0 = 没有帧错误 (FE) 1 = 检测到帧错误 (FE)
6	SM1	
5	SM2	**串口0处理通讯模式选择** 该位功能取决于串口0模式 模式0: 该位选择波特率F_{SYS}/12 或 F_{SYS}/4. 0 = 时钟运行在F_{SYS}/12 波特率.与标准8051兼容。 1 = 时钟运行在F_{SYS}/4 波特率，获得更高传输速度 模式1: 该位检查有效停止位. 0 = 接收有效不管停止位是否有逻辑电平. 1 = 接收仅在接收停止位为逻辑1,同时接收数据与GIVEN或BROADCAST地址匹配时有效 模式2 或 3: 对于多机通信. 0 =接收有效不管第9位是否有逻辑电平. 1 =接收仅在第9位为逻辑1和接收数据与GIVEN或BROADCAST地址匹配时有效.

位	名称	描述
4	REN	**串口0接收使能** 0 = 关闭串口0接收功能. 1 = 打开串口0在模式1，2或3模式下的接收功能。接收完成后，该位不会被硬件清除。所以用户必须在每字节接收完毕后，软件清除该位，并等待接收下一字节.
3	TB8	**串口0第9位发送位** 串口0在模式2和3中要被发送的第九位数据。在模式0和1中，不支持该功能.
2	RB8	**串口0第9位接收位** 串口0在模式2和3中接收到的第九位数据。模式1下，若SM2=0则RB8是接收到的停止位。模式0下该位无意义.
1	TI	**串口0发送中断标志位** 发送中断标志：模式0下该标志由硬件在发送完8位数据后置1，而在其它模式下在串行发送到停止位的开始时置位。当该位中断使能，发生中断后会转至中断子程。该位必须由软件来清除。
0	RI	**串口0接收中断标志** 模式0下该标志由硬件置位。在模式0中，接收到第8位或第9位；模式1中接收到停止位(stop bit)；模式2和3中接收到第9位，使该位置位。当SM2被限制的情况例外。当串口0中断使能，该位置位会转跳到中断子程运行。该位必须由软件来清除.

SCON_1 – 串口1控制寄存器 (可位寻址)

7	6	5	4	3	2	1	0
SM0_1/FE_1	SM1_1	SM2_1	REN_1	TB8_1	RB8_1	TI_1	RI_1
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: F8H 复位值: 0000 0000b

位	名称	描述
7	SM0_1/FE_1	**串口1模式选择** SMOD0_1 (T3CON.6) = 0: 详见 表 14–2. 串口1模式描述 SMOD0_1 (T3CON.6) = 1: SM0_1/FE_1 用于标示帧错误 (FE) 状态位。由软件清除. 0 = 没有帧错误 (FE) 1 = 检测到帧错误 (FE)
6	SM1_1	
5	SM2_1	**串口1处理通讯模式选择** 该位功能取决于串口1模式 模式0: 无效果 模式1: 该位检查有效停止位. 0 = 接收有效不管停止位是否有逻辑电平. 1 = 接收仅在接收停止位为逻辑1,同时接收数据与GIVEN或BROADCAST地址匹配时有效 模式2 或 3: 对于多机通信. 0 =接收有效不管第9位是否有逻辑电平. 1 =接收仅在第9位为逻辑1和接收数据与GIVEN或BROADCAST地址匹配时有效.
4	REN_1	**串口1接收使能** 0 = 关闭串口1接收功能. 1 = 打开串口1在模式1，2或3模式下的接收功能。 模式0下，打开接收，需配置REN_1=1及RI_1=0
3	TB8_1	**串口1第9位发送位** 串口0在模式2和3中要被发送的第九位数据。在模式0和1中，不支持该功能.
2	RB8_1	**串口1第9位接收位** 串口0在模式2和3中接收到的第九位数据。模式1下，若SM2=0则RB8是接收到的停止位。模式0下该位无意义.
1	TI_1	**串口1发送中断标志位** 发送中断标志：模式0下该标志由硬件在发送完8位数据后置1，而在其它模式下在串行发送到停止位的开始时置位。当该位中断使能，发生中断后会转至中断子程。该位必须由软件来清除。
0	RI_1	**串口0接收中断标志** 模式0下该标志由硬件置位，当串口发生帧错误接收到第8位；模式1，2或3中接收到停止位(stop bit)；使该位置位。当SM2_1被限制的情况例外。当串口1中断使能，该位置位会转跳到中断子程运行。该位必须由软件来清除.

PCON – 电源控制寄存器

7	6	5	4	3	2	1	0
SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL
读/写	读/写	-	读/写	读/写	读/写	读/写	读/写

地址: 87H 复位值: 详见 表 6–2. SFR定义及复位值

位	名称	描述
7	SMOD	**串口0波特率加倍使能.** 串口0在模式2, 或模式1或3于定时器1溢出作为波特率时钟源时。详见表 14–1. 串口0 模式描述
6	SMOD0	**串口0帧错误标志使能位** 0 = 关闭帧错误检测功能。和标准8051相同作为SM0用 1 = 使能帧错误检测. SM0/FE位用于帧错误 (FE) 状态标志

T3CON – 定时器 3 控制位

7	6	5	4	3	2	1	0
SMOD_1	SMOD0_1	BRCK	TF3	TR3	T3PS[2:0]		
读/写	读/写	读/写	读/写	读/写	读/写		

地址: C4H 复位值: 0000 0000b

位	名称	描述
7	SMOD_1	**串行口1波特率加倍使能.** 当串口1在模式2下，时钟溢出波特率加倍使能。详见 表 14–2. 串口1模式描述
6	SMOD0_1	**串口1帧错误使能位** 0 =关闭帧错误检测功能。作为SM0_1用. 1 =使能帧错误检测，用作帧错误 (FE) 状态标志FE_1

表 14–1. 串口0 模式描述

Mode	SM0	SM1	描述	帧位数	波特率
0	0	0	同步	8	F_{SYS} 除以12 或 除以 4[1]
1	0	1	异步	10	定时器1溢出时间 除以32 或 除以16 [2]
2	1	0	异步	11	F_{SYS} 除以64 或 除以32 [2]
3	1	1	异步	11	定时器1溢出时间 除以32 或 除以16 [2]

[1] 当 SM2 (SCON.5) 设为 1.
[2] 当 SMOD (PCON.7) 设为 1.

表 14–2. 串口1模式描述

Mode	SM0	SM1	描述	帧位数	波特率
0	0	0	同步	8	F_{SYS} 除以23或除以2[1]
1	0	1	异步	10	定时器3溢出除以16
2	1	0	异步	11	F_{SYS} 除以32或64[2]
3	1	1	异步	11	定时器3溢出时间除以16

[1] 当 SM2_1 (SCON_1.5) 设为 1.
[2] 当 SMOD_1 (T3CON.7) 设为 1.

SBUF – 串口0数据缓存寄存器

7	6	5	4	3	2	1	0
SBUF[7:0]							
读/写							

地址: 99H　　复位值: 0000 0000b

位	名称	描述
7:0	SBUF[7:0]	**串口0数据缓存** 串口0接收或发送的数据都放在这个寄存器中。实际上该地址上有2个独立的8位寄存器。一个用于接收数据，一个用于发送数据。对它进行读操作将会接收串行数据，对它进行写操作则发送串行数据. 每次向SBUF写入一字节数据，启动一次发送

SBUF_1 – 串口1数据缓存寄存器

7	6	5	4	3	2	1	0
SBUF_1[7:0]							
读/写							

地址: 9AH　　复位值: 0000 0000b

位	名称	描述
7:0	SBUF_1[7:0]	**串口1数据缓存** 串口1接收或发送的数据都放在这个寄存器中。实际上该地址上有2个独立的8位寄存器。一个用于接收数据，一个用于发送数据。对它进行读操作将会接收串行数据，对它进行写操作则发送串行数据. 每次向SBUF_1写入一字节数据，启动一次发送

AUXR1 – 辅助寄存器1

7	6	5	4	3	2	1	0
SWRF	RSTPINF	T1LXTM	T0LXTM	GF2	UART0PX	0	DPS
读/写	读/写	读/写	读/写	读/写	读/写	R	读/写

地址: A2H　　复位值: 详见 表 6–2. SFR定义及复位值

位	名称	描述
2	UART0PX	**串口0管脚位置配置** 0 = RXD 为 P2.0， TXD 为 P0.3 （默认值）. 1 = RXD 为 P0.3， TXD 为 P2.0. 注：更改此位后TXD及RXD的配置更改立即生效。软件需确保切换过程中没有进行串口传输，否则可能引发无法预测的结果。

14.1 模式0

模式 0与外部设备进行同步通信的方式。在该模式下，串行数据由RXD脚进行收发，而TXD 脚用于产生移位时钟。这种方式下是以半双工的形式进行通信，每帧接收或发送8位数据。数据的最低位被最先发送或接收，波特率F_{SYS}/12(SM2 (SCON.5) 为 0) 或 F_{SYS}/2 （SM2 = 1）. 无论传输或接受 串行时钟将一直产生.因此串口模式 0 为主机模式。图 14-1 显示串口模式0传输时序图

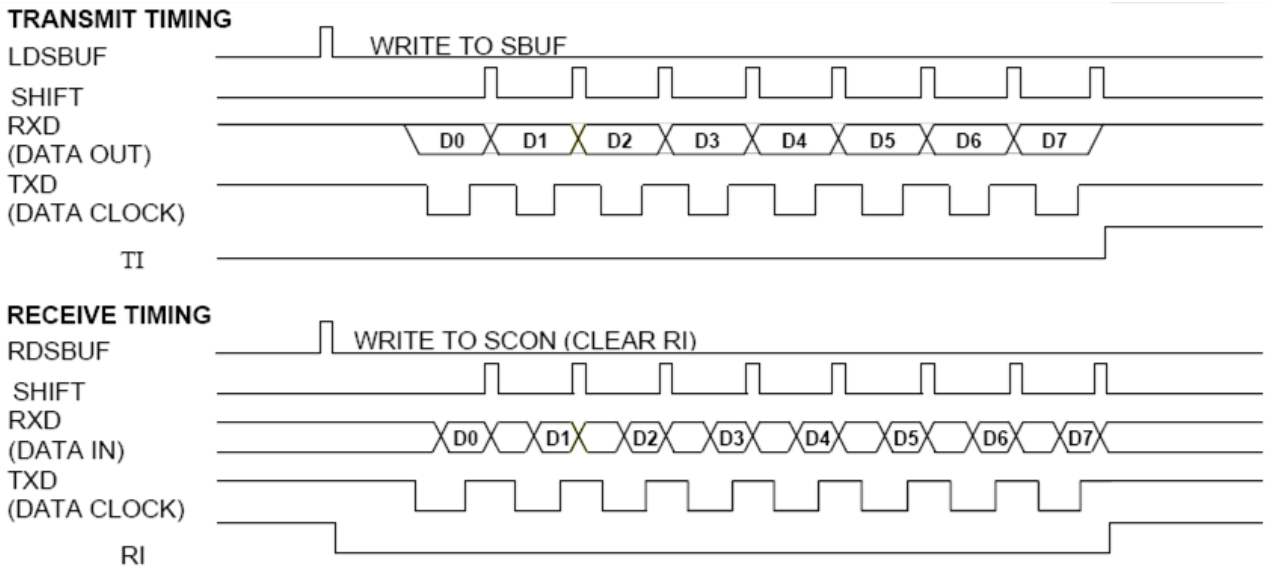


图 14-1. 串口模式0时序图

如图所示，数据由双向RXD线进行收发。位移时钟TXD线用来输出移位时钟，移位时钟用来和其它设备串行接收/发送数据。数据最低位用于移入移出数据，波特率根据TXD时钟同步。

向SBUF的写入数据将会开启发送，此时移位时钟启动数据从RXD脚串行移出，直至送完8位数据传输。传输标志位TI (SCON.1) 置 1表示 1 字节传输完成。

当REN (SCON.4)=1 且 RI(SCON.0)=0 时串行口接收数据。移位时钟被激活，串行口会在移位时钟的上升沿锁定数据。外部设备要在移位时钟的下降沿处送出数据。这个过程持续到8位数据全部发送完毕。RI会在TXD的最后一个下降沿处置1，这时接收动作结束，注REN不由硬件清零，用户应该首先清零RI，清REN，并再次通过软件置位REN，以触发下一字节的接收。

14.2 模式 1

模式1为全双工的方式工作。串行通信的数据贞由10位数据组成，在RXD和TXD脚上进行收发。10位数据组成如下：起始位（位0），8位数据（最低位在前），终止位（1）。波特率由定时器1决定，SMOD (PCON.7) 设置为1使波特率加倍(定时器1为波特率发生源). 图 14-2为模式1的时序图.

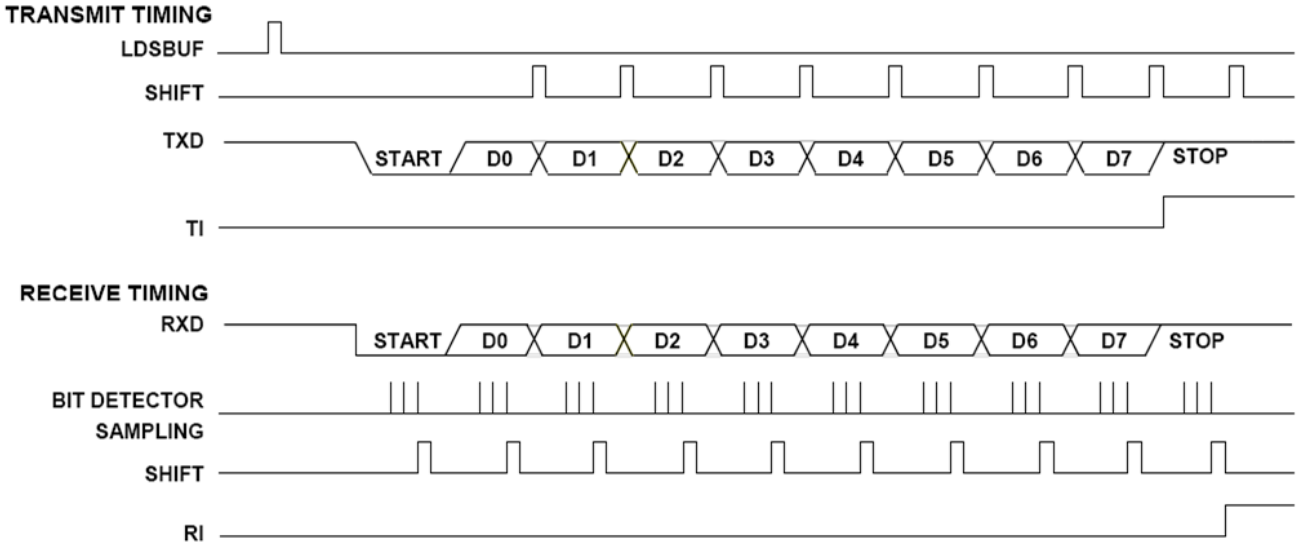


图 14-2. 串口模式1时序图

向SBUF写入指令开始传输，传输发生在TXD引脚上。首先是开始位，随后是8位数据位，最后是停止位，停止位出现后，TI（SCON.1）置1 表示一个字节传输完成，所有位的传输速度取决于波特率。

当波特率发生器打开且REN(SCON.4) =1 时系统进行接收操作，RXD脚上接收到1-0跳变就启动接收器接收。数据根据波特率的时钟频率接收，停止位必须符合一定的条件，才能从SBUF读到的数据：

1. RI (SCON.0) = 0

2. 任何SM2 (SCON.5) = 0, 或当SM2 = 1.时接收停止位= 1。或是被定址“Given”或符合广播地址（Broadcast address）详见14.7 “多机通讯” 和 14.8 “自动地址识别”.

如果上述条件满足，SBUF将加载接收数据，RB8（SCON.2）停止位，和RI将被置1， 如果条件不满足，RI保持为0，没有数据加载。完成接收过程后，串口控制等待RXD脚上的另一个1-0传输以开始新的数据接收。

14.3 模式 2

模式2为全双工异步通信，与模式1不同的是，模式2是11位收发： 数据由起始位位（逻辑0），8位数据（最低位在前），第9位数据（TB8）和停止位组成。第9位做奇偶校验或用来区分数据和地址，数据接收至RB8。波特率是时钟频率的1/32 或1/64，由 SMOD位来选择。图 14-3 模式2的传输时序。

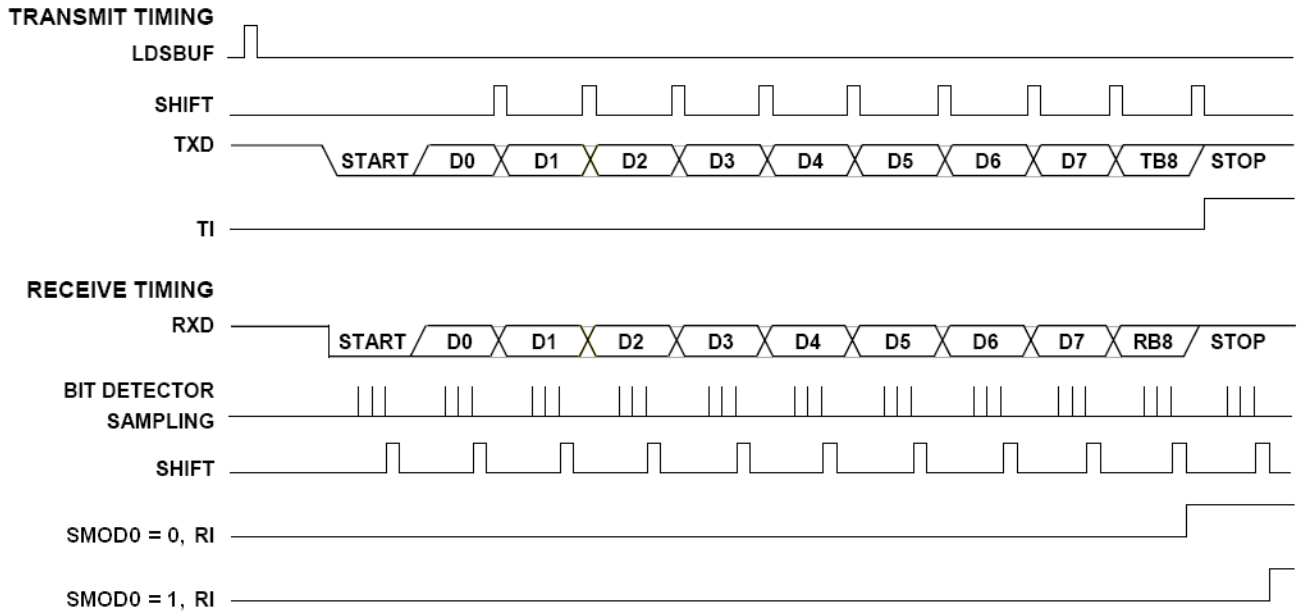


图 14-3. 串口模式2及模式3 时序图

向SBUF中写入数据启动一次发送，首先是开始位，8位数据和TB8（SCON.3），最后是停止位，停止位出现时，TI将置位以示传输完成

当REN=1 时系统进行接收操作，RXD上的下降沿表示接收过程开始，数据根据所配置波特率进行采样，并在选定的波特率转移。在第9位的情况下，必须符合一定的条件，装载SBUF才接收到的数据：

1. RI (SCON.0) = 0,

2. 任何 SM2(SCON.5) = 0, 或接受 9^{th} 位 = 1 当 SM2 = 1。被定址“Given”或符合广播地址（Broadcast address）详见14.7 “多机通讯” 和 14.8 “自动地址识别”.。

如果上述条件满足，则停止位进入RB8(SCON.2)，8位数据进入SBUF，RI置位，否则丢弃接收到的贞数据。在停止位的中间，接收器重启，开始新的一次接收。

14.4 模式 3

除波特率外 模式 3 与模式 2相同。模式3采用定时器1的溢出作为波特率时钟。

14.5 波特率

不同的波特率时钟源，以及不同模式，以及所对应的串口不同，产生的结果也不同。详见表 14–3. 用于设定不同的波特率。

在模式1或模式3，串口0的波特率时钟源可通过BRCK (T3CON.5)选择定时器1或定时器3。对于串口1，只有采用定时器3作为时钟源。

T3CON – 定时器3控制寄存器

7	6	5	4	3	2	1	0
SMOD_1	SMOD0_1	BRCK	TF3	TR3	T3PS[2:0]		
读/写	读/写	读/写	读/写	读/写	读/写		

地址: C4H　　复位值: 0000 0000b

位	名称	描述
5	BRCK	**串口0波特率时钟源选择** 该位用于设置串口0 在模式1或模式3下，所使用的定时器 0 = 定时器 1. 1 = 定时器 3.

当采用定时器1作为波特率发生器，需要关闭定时器1中断。定时器1设置为计数器或是定时器都可。通常会设定位定时器自动重装载模式（定时器模式2）。如果采用定时器3，同样也需要关闭定时器3中断。

表 14–3. 串口波特率算式

UART 模式	波特率时钟源	波特率
0	系统时钟	$F_{SYS}/12$ or $F_{SYS}/2$ [1]
2	系统时钟	$F_{SYS}/64$ or $F_{SYS}/32$ [2]
1 or 3	定时器 1 (仅供UART0)[3]	$\frac{2^{SMOD}}{32}\times\frac{F_{SYS}}{12\times(256-TH1)}$ or $\frac{2^{SMOD}}{32}\times\frac{F_{SYS}}{256-TH1}$ [4]

UART 模式	波特率时钟源	波特率
	定时器 3 (仅供 UART0)	$\frac{2^{SMOD}}{32} \times \frac{F_{SYS}}{Pre\text{-}scale \times (65536 - \{RH3,RL3\})}$ [5]
	定时器 3 (仅供 UART1)	$\frac{1}{16} \times \frac{F_{SYS}}{Pre\text{-}scale \times (65536 - \{RH3,RL3\})}$ [5]

[1] SM2 (SCON.5) 或 SM2_1(SCON_1.5) 设为1.
[2] SMOD (PCON.7) 或 SMOD_1(T3CON.7) 设为 1.
[3] 定时器 1 配置为定时方式自动重装载模式（模式2）
[4] T1M (CKCON.4) 设为。当 SMOD 为 1, TH1 不能设为 FFH.
[5] {RH3,RL3} 在算式中 = $256 \times RH3 + RL3$. 当SMOD 为 1及与分频 1/1, {RH3,RL3} 不能设为 FFFFH.

表 14–4 列出各种常用的波特率及如何配置定时器1的数据。在此表中定时器设定为自动重装载模式 SMOD (PCON.7) = 0 及 T1M (CKCON.4) = 0. 表 14–5 设定串口0 定时器3，若 SMOD = 0，波特率加倍。

表 14–4. 常用波特率表（使用定时器1）

波特率 (bps)	主振频率(MHz)				
	3.6864	11.0592	14.7456	18.432	22.1184
	TH1 填入值				
57600	-	-	-	-	FFH
38400	-	-	FFH	-	-
19200	-	-	FEH	-	FDH
9600	FFH	FDH	FCH	FBH	FAH
4800	FEH	FAH	F8H	F6H	F4H
2400	FCH	F4H	F0H	ECH	E8H
1200	F8H	E8H	E0H	D8H	D0H
300	E0H	A0H	80H	60H	40H

表 14–5. 常用波特率表（使用定时器3）

波特率 (bps)	主振频率(MHz)				
	3.6864	11.0592	14.7456	18.432	22.1184
	{RH3,RL3} 自动重装载值				
115200	FFFFH	FFFDH	FFFCH	FFFBH	FFFAH
57600	FFFEH	FFFAH	FFF8H	FFF6H	FFF4H
38400	FFFDH	FFF7H	FFF4H	FFF1H	FFEEH
19200	FFFAH	FFEEH	FFE8H	FFE2H	FFDCH

波特率 (bps)	主振频率(MHz)				
	3.6864	11.0592	14.7456	18.432	22.1184
	{RH3,RL3} 自动重装载值				
9600	FFF4H	FFDCH	FFD0H	FFC4H	FFB8H
4800	FFE8H	FFB8H	FFA0H	FF88H	FF70H
2400	FFD0H	FF70H	FF40H	FF10H	FEE0H
1200	FFA0H	FEE0H	FE80H	FE20H	FDC0H
300	FE80H	FB80H	FA00H	F880H	F700H

14.6 帧错误检测

帧错误检测用于异步模式 (模式 1, 2 和 3.)。 当有总线由干扰或连接不上，导致没有检测到停止位时，发生帧错误.串口可以检测帧错误，并通过软件标志提示出错。

SCON.7是FE标志（帧错误标志）（FE_1） 。在标准8051种该位是SM0 ，但当SMOD0 (PCON.6)置1，帧错误检测功能打开，她作为FE标志，但在 N76E885系列中它有附加功能称为SM0/FE。他们其实是相互独立的标志位

FE标志由硬件置位且必须由软件清0。注意在对FE标志位进行读写时，SMOD0必须为1。如果FE置位，那么下次接收到的正确数据帧不会将其清除。对该位的清除必须由软件来完成。

14.7 多机通讯

N76E885串口多机通讯，可让一个主机（master device） 向多个从机 （slave device） 发送多帧序列信息, 在同一串行线上使用该功能过程中不需要中断其它从机设备。该功能只能在模式2或模式3下进行。当第9位数据接收到后，第9位数据内容存入RB8 (SCON.2)。当接收到停止位stop bit后，用户可通过设定SM2（SCON.5）为1使能该功能。只有RB8为1时，才能产生中断。当SM2位 为1，接收到的第9位数据为0，不会引发中断。在上例中，第9位能简单得从地址中分离。

当主机需要向多个从机中的一个发送数据时，首先需要发送目标从机的地址。注，地址字节与数据字节是不同的：在地址字节中，第9位为1。而数据字节中第9位为0。地址字节会触发所有从机，而每台从机检查接收到的地址是否与自身匹配。未被寻址到的从机的SM2 必须保持，从而系统会持续工作。.

配置多机通信步骤如下：

1. 设置所有设备(主机与从机)为串口模式2或3.

2. 所有从机 SM2 位置为1

3. 主机传输协议:

– 开始位： 地址(第9位 = 1)，定义目标从机

– 下 1 字节 : 数据(第9位 = 0).

4. 当目标从机接受到开始位, 所以从机将中断 因为第9位数据为 1。 目标从机比较自身地址 并且清 SM2 位 接受下列数据， 其它从机继续运行。

5. 接收到所有数据后， 置 SM2 为 1 等待下一地址.

SM2 在模式 0 下无效。若 SM2 置 1，模式1可用于检测有效的停止位，同时将无法产生中断除非有效停止位已经接收。

14.8 自动地址识别

自动地址识别是这样一种特性，它允许串口在数据流传输过程中，通过硬件识别比较，确认特定数据为地址。该功能可以节省软件识别地址而所占用的程序空间，仅当串口识别到自身地址时，接收器置位RI位并请求中断。 当多机通信特征使能时（SM2置位），就使能自动地址识别。

如果需要，用户可以在模式1下使能自动地址识别特征。在这种配置下，停止位取代第九位的数据位。仅当接收命令的帧地址与器件地址匹配和有效的停止位中止时，RI置位。

使用自动地址识别，允许一个主机选择与一个或多个从机通信，通过“Given”从机地址. 所有从机可以通过“广播”地址联系。有两个特殊功能寄存用于定义从机地址 SADDR和从机地址掩码SADEN。 SADEN用于定义SADDR的哪位被用，哪位不必关心. SADEN掩码可以与SADDR以逻辑与得方式以创建每个从机的“Given” 地址。 使用 “Given”地址允许多从机被识别。

SADDR – 从机0地址

7	6	5	4	3	2	1	0
SADDR[7:0]							
读/写							

地址: A9H 复位值: 0000 0000b

位	名称	描述
7:0	SADDR[7:0]	**从机0地址** 该字节定义微控器自身的从机地址以用于串口0多机通信..

SADEN – 从机 0地址掩码

7	6	5	4	3	2	1	0
SADEN[7:0]							
读/写							

地址: B9H 复位值: 0000 0000b

位	名称	描述
7:0	SADEN[7:0]	**从机0地址掩码.** 该字节为掩码存储"Given"地址的"无关"位。无关位可使更多从机得以灵活运用。

SADDR_1 – 从机 1地址

7	6	5	4	3	2	1	0
SADDR_1[7:0]							
读/写							

地址: BBH 复位值: 0000 0000b

位	名称	描述
7:0	SADDR_1[7:0]	**从机1地址** 该字节定义微控器自身的从机地址以用于串口1多机通信..

SADEN_1 – 从机1地址掩码

7	6	5	4	3	2	1	0
SADEN_1[7:0]							
读/写							

地址: BAH 复位值: 0000 0000b

位	名称	描述
7:0	SADEN_1[7:0]	**从机1地址掩码.** 该字节为掩码存储"Given"地址的"无关"位。无关位可使更多从机得以灵活运用。

下列范例用以说明该功能的灵活应用。

范例 1, 从机 0:

```
SADDR = 11000000b
SADEN = 11111101b
Given = 110000X0b
```

范例 2, 从机 1:

```
SADDR = 11000000b
SADEN = 11111110b
Given = 1100000Xb
```

在上面的例子中SADDR是相同的，SADEN的数据用于区分两个从机。从机0要求位0为”0”而忽略位1，从机1要求位1为”0”而位0被忽略。一个从机0唯一的地址11000010B，由于从机1要求位1为0。一个从机1唯一的地址将自1位011000001b将排除从机0。这两个从机可以选择在同一时间，地址位0 = 0（从机0）和第1位= 0（从机1）。因此，使用广播地址(Boadcast address) 就可以解决同时地址位为11000000b的问题。

更复杂应用可用于排除从机0之后，选择从机1或2：

范例 1, 从机 0:

```
SADDR = 11000000b
SADEN = 11111001b
Given = 11000XX0b
```

范例 2, 从机 1:

```
SADDR = 11100000b
SADEN = 11111010b
Given = 11100X0Xb
```

范例 3, 从机 2:

```
SADDR = 11000000b
SADEN = 11111100b
Given = 110000XXb
```

在上面的例子中，3个从机的分别是在地址的低3位。从机0要求位0 = 0，它可用11100110b解决。从机1要求位1= 0，它可用11100101b识别。从机2要求位2= 0，其独立的地址是11100011b。要选择从机0和1，去除从机2，可使用地址11100100b，因为它是必要的第2位= 1来排除从机2。

为每个从站的地址是“广播”创建逻辑或SADDR和SADEN。使用“无关”地址功能时结果为零。在大多数情况下，解释“无关”的广播地址将是FFh。例如：

```
SADDR     = 01010110b
SADEN     = 11111100b
Broadcast = 1111111Xb
```

使用“无关”位可在广播模式下，提供更灵活的应用。不过在大部分应用条件下，广播地址全部使用FFH。

复位后，SADDR和SADEN初始化为00H。这将对于所有所有“无关”地址产生一个“Given”地址，以及一个“广播”地址对应所有XXXXXXXXb地址（所有“无关”位）。这样有效地禁止了自动寻址模式，允许微控制器保持标准串口模式而不使用这个功能。

15. SPI 总线

N76E885系列 有支持高速串行通信的SPI模块。SPI 为全双工、高速、同步传输总线在微芯片外设 EEPROM, LCD 驱动, D/A 转换之间，提供8种主机从机模式传输，速度可达时钟频率$F_{SYS}/4$，支持传输完成标志位“写”冲突标志位。 在多主机系统中，SPI 支持主机模式故障报错用以防止主机冲突。

15.1 功能描述

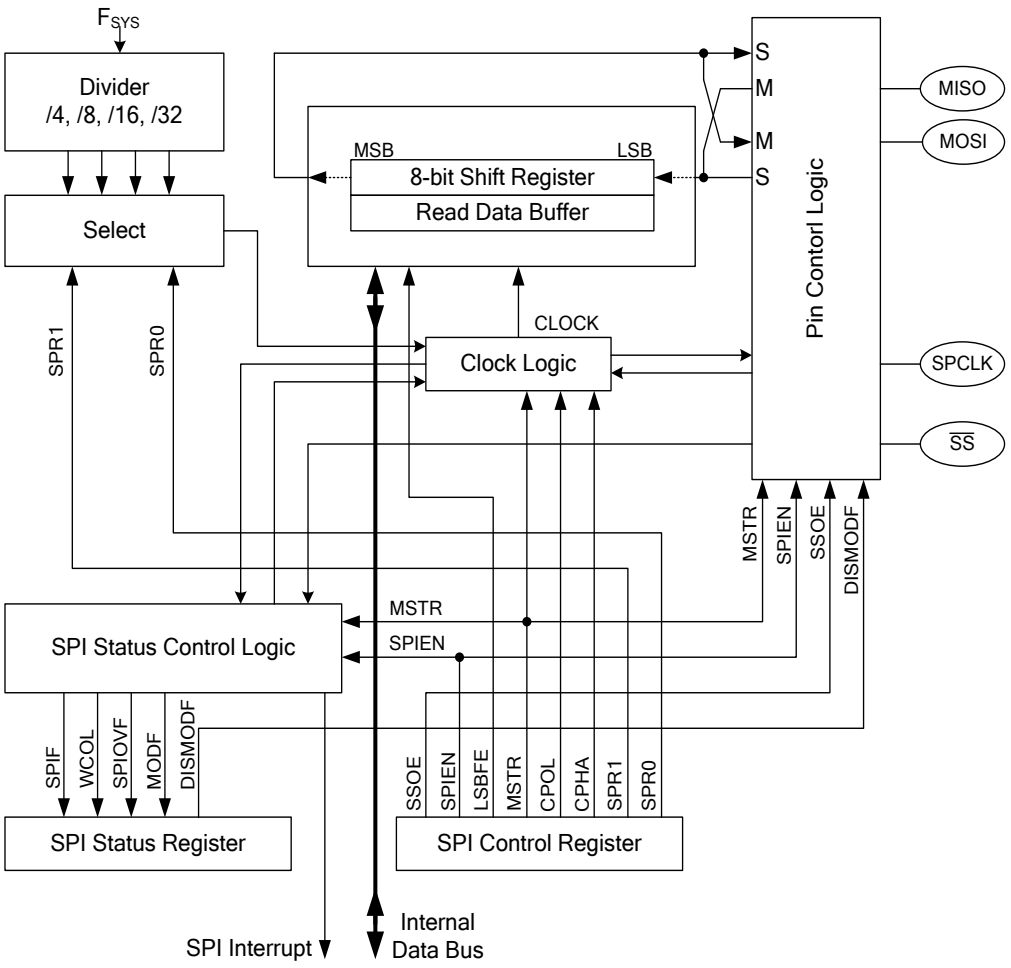


图 15-1. SPI 功能模块

表 15–1 为SPI框图，展示了SPI的体系结构。SPI寄存器板块是SPI模块的主要组成部分，包括逻辑控制，波特率控制和管脚逻辑控制，SPI 包括移位寄存器和读出数据缓冲器，传送数据是单缓冲器，接收数据是双缓冲器。在传送完成之前传送的数据不能写入移位装置。

SPI 界面需要四个管脚 主进/从出(MISO)，主出/从进(MOSI)，移位时钟(SPCLK), 和从机选择($\overline{SS}$)。MOSI脚用于传输主机到从机的8位数据，因些，MOSI是一个主机设备的输出引脚，从机设备的输入引脚。相应的，MISO用于接收从机到主机的串行数据。

SPCLK引脚为主机模式下的时钟输出，从机模式的时钟输入。移位时钟用于MOSI和MISO脚之间数据传输的时钟同步。位移时钟由主机输出，一组SPI传输系统上只能有一个主机以避免设备冲突。建议该管脚设置为史密特触发输入模式。

每路从机外设通过设定从机选择脚 ($\overline{SS}$)使能。当需要读取任何从机时，该信号脚必须保持低。当 $\overline{SS}$ 为高，从机读写将被禁止。若为多从机模式， 在同一时刻必须只有一个从机保持此低电位，对于主机 $\overline{SS}$ 脚不做任何用途，可配置为普通端口另做他用。 $\overline{SS}$ 可用于多主机模式下错误侦测功能(详见 章节 15.5 “模式故障侦测”)。N76E885 也提供通过 $\overline{SS}$ 锁存字节传输数据功能。

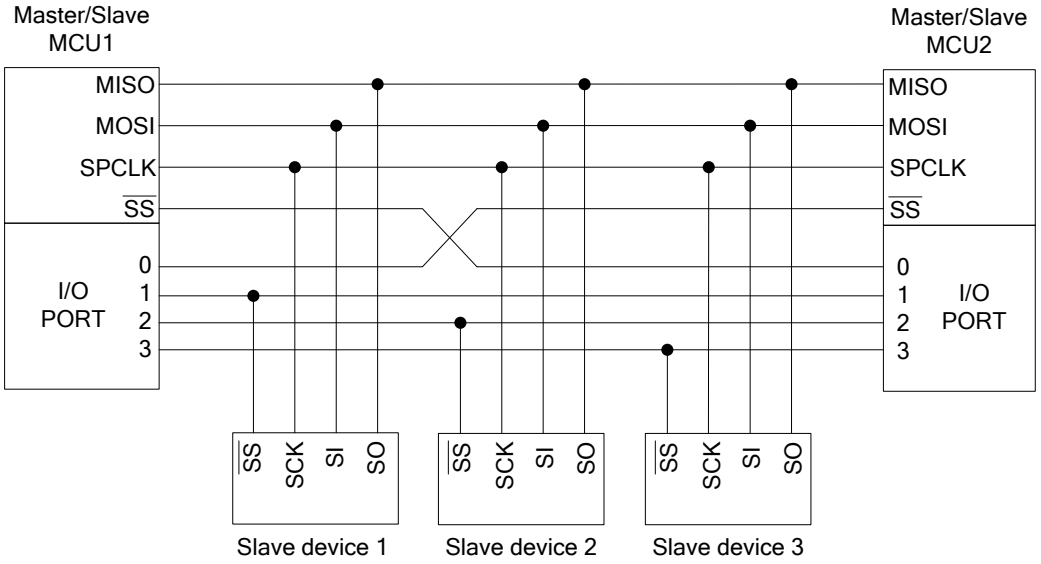


图 15-2. SPI 多主机，多从机连接界面

图 15-2为典型的SPI设备通信总线通常为 3 信号线相连, MOSI ~ MOSI, MISO ~ MISO, 和 SPCLK ~ SPCLK. 主机通过四线并行连接的方式，每根SS线分别控制每个从机。MCU1 和 MCU2 可以任意定义为主/从机模式. $\overline{SS}$ 需配置为主机模式侦测功能 避免多主机冲突。

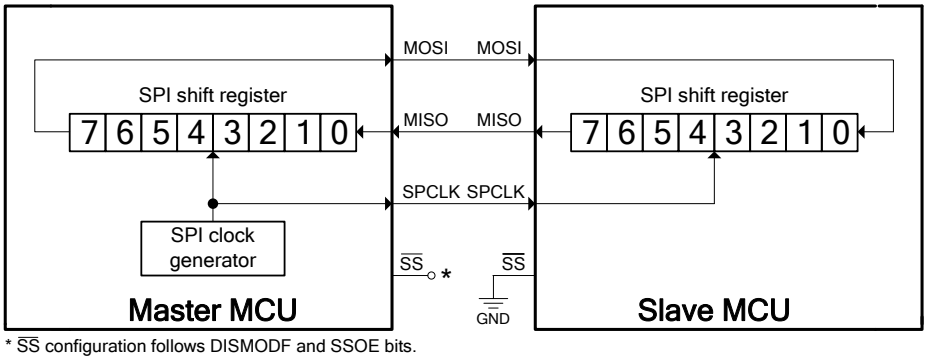


图 15-3. SPI 单主机，单从机连接界面

图 15-3表示SPI 模块单主机/从机互连简图。在传输时，主机通过MOSI线向从机发送数据。同时， 主机也通过MISO线由从机接收数据。此时主机和从机的两个数据寄存器可被视为一个16位的循环位移寄存器。因此，当主机向从机某地址送数据时，从机内该地址内的数据同时也由从机推向主机。传输进行了交换的动作。

默认情况下，SPI先发送MSB 。当LSBFE (SPCR.5) 置1，SPI首先发送LSB，该位不会影响寄存器内MSB/LSB的排列顺序。注，下述全部基于 LSBFE为0的情况，MSB 首先被发送。

控制寄存器 (SPCR), SPI 状态寄存器 (SPSR), SPI 数据寄存器 (SPDR) 这三个寄存器用于SPI传输。这些寄存器提供控制，状态检测，数据存储以及时钟发生设置。

SPCR – SPI控制寄存器

7	6	5	4	3	2	1	0
SSOE	SPIEN	LSBFE	MSTR	CPOL	CPHA	SPR1	SPR0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: F3H　　复位值: 0000 0000b

位	名称	描述
7	SSOE	**从机选择输出使能位** 该位搭配DISMODF (SPSR.3) 用于定义管脚，如表 15–1.，该位仅在MSTR=1和DISMODF=1的条件下有效。 注，仅当MSTR = 1 及DISMODF = 1的条件下该位有效。 0 = $\overline{SS}$ 作为普通 I/O. 1 = $\overline{SS}$ 选择外部从机驱动 自动拉低 ，总线进入空闲模式时变高
6	SPIEN	**SPI 使能** 0 = 关闭SPI功能. 1 = 打开SPI功能.
5	LSBFE	**LSB 优先使能** 0 = SPI 优先传输最高位MSB数据 1 = SPI 优先传输最低位LSB数据

位	名称	描述
4	MSTR	**使能主机模式** 该位用于切换SPI工作于主机与从机模式. 0 = SPI 配置为从机模式. 1 = SPI配置为主机模式.
3	CPOL	**SPI 时钟极性选择** CPOL位定义在SPI总线在空闲模式时时钟脚的电平状态。详见 图 15-4. SPI 时钟格式. 0 = SPI时钟在空闲模式时低电平. 1 = SPI时钟在空闲模式时高电平.
2	CPHA	**SPI 时钟相位选择** CPHA 位定义在采样所用时钟边沿。详见 图 15-4. SPI 时钟格式 0 = SPI在时钟第一个边沿采样数据。 1 = SPI在时钟第二个边沿采样数据。
1:0	SPR[1:0]	**SPI 时钟速率选择** 这两位搭配确定SPI时钟分频 下述以 F_{SYS} = 24 MHz 条件计算. SPR1 SPR0 除频 SPI 时钟速率 0 0 4 6M bit/s 0 1 8 3M bit/s 1 0 16 1.5M bit/s 1 1 32 750k bit/s SPR[1:0]只在主机模式有效 (MSTR = 1)。在从机模式下，时钟脚自动与主机同步最高速率为 F_{SYS}/4

表 15–1. 从机选择脚定义

DISMODF	SSOE	主机模式 (MSTR = 1)	从机模式 (MSTR = 0)
0	X	$\overline{SS}$作为模式错误输入脚	$\overline{SS}$ 作为从机选择输入脚
1	0	普通I/O	
1	1	自动 $\overline{SS}$ 输出	

SPSR – SPI 状态寄存器

7	6	5	4	3	2	1	0
SPIF	WCOL	SPIOVF	MODF	DISMODF	-	-	-
读/写	读/写	读/写	读/写	读/写	-	-	-

地址: F4H 复位值: 0000 0000b

位	名称	描述
7	SPIF	**SPI传输完成标志** 在SPI数据传输完成或接收到的数据移入到SPI读缓冲时，该位通过硬件设置为1. 如果使能 ESPI (EIE .6) 和 EA, SPI中断请求. 该位必须由软件清零. 如果SPIF置1，禁止向SPDR写入
6	WCOL	**写冲突位** 该位表示写冲突事件. 一旦发生写冲突事件，该位被置位，必须通过软件清零.
5	SPIOVF	**SPI 溢出标志** 该位表示溢出事件，一旦发生溢出，该位置位，如果使能ESPI 和 EA, SPI请求中断. 该位必须由软件清零
4	MODF	**模式错误中断状态标志** 该位表示模式错误事件. 如果 $\overline{SS}$ 配置成模式错误输入(MSTR=1且DISMODF=0)和 $\overline{SS}$ 被外部器件拉低, 产生模式错误. MODF将被置1. 如果使能 ESPI 和 EA, SPI中断请求. 该位必须由软件清零
3	DISMODF	**禁止模式错误检测.** 该位结合SSOE (SPCR.7) 位用于决定$\overline{SS}$ 的特征. DISMODF 仅在主机模式下有效(MSTR = 1)详见 表 15–1. 0 = 使能模式错误检测. $\overline{SS}$ 为模式错误检测的输入脚，不管SSOE 1 = 禁止模式错误检测. $\overline{SS}$ 的特征依赖SSOE 位

SPDR – SPI 数据寄存器

7	6	5	4	3	2	1	0
SPDR[7:0]							
读/写							

地址: F5H 复位值: 0000 0000b

位	名称	描述
7:0	SPDR[7:0]	**串行外设数据寄存器** 该字节为SPI总线上传输或接收的数据。这该字节写入执行对移位寄存器写操作。读取这个字节，实际上是一个缓冲区读取数据读取。在主机模式，写该寄存器同时初始化传输和一个字节接收。

15.2 工作模式

15.2.1 主机模式

对MSTR (SPCR.4)位置1，芯片作为主机模式开始SPI传输模块开始工作。整个SPI系统中只允许一个主机启动传输。每次传输总是由主机发起，对主机SPDR寄存器的写开始传送。在SPCLK控制下在MOSI管脚传送数据。8位数据传输完毕，SPIF (SPSR.7)由硬件自动置位以示完成一个字节数据传输， 同时由从机接收到的数据传送到SPDR。从SPDR读出数据后，用户才可以清除SPIF。

15.2.2 从机模式

设定MSTR为0，SPI将工作在从机模式。当作为从机模式时，SPCLK管脚变为输入脚，它将被另外一个主机的SPI设备控制， $\overline{SS}$ 管脚也变为输入，同样，在数据传输完成前保持低电平状态。如果 $\overline{SS}$ 变为高电平，SPI将被迫进入闲置状态。如果 $\overline{SS}$ 管脚在传输的过程被置高，那么传输将被取消，同时接受数据的缓存区也将进入闲置状态。

在从机模式下，数据在MOSI管脚从主机向从机流动，在MISO管脚从从机向主机流动。根据SPCLK的时钟控制数据由主机位移传入，每次一个字节传输完成SPIF置1，此时读取SPDR寄存器即为该字节内容。对SPDR的读实际上就是对缓冲器的读。为了防止缓冲器溢出和由于溢出导致的数据丢失，SPIF必须在数据第二次从移位寄存器向读缓冲器传送前清零。

15.3 时钟格式与数据传输

为了适应各种各样的同步串行外设，SPI提供时钟极性位CPOL（SPCR.3）和时钟相位位CPHA（SPCR.2）寄存器用以控制。如图 15-4. SPI 时钟格式所示，CPOL和CPHA组和出四种不同的时钟格式。 CPOL 位表示空闲状态时SPCLK脚电位。 CPHA位表示是由MOSI或由MISO上那条线的边缘采

样。在同一系统上的主从设备中，CPOL和CPHA的应该是相同的，传输不同的数据格式，将产生随机错误结果。

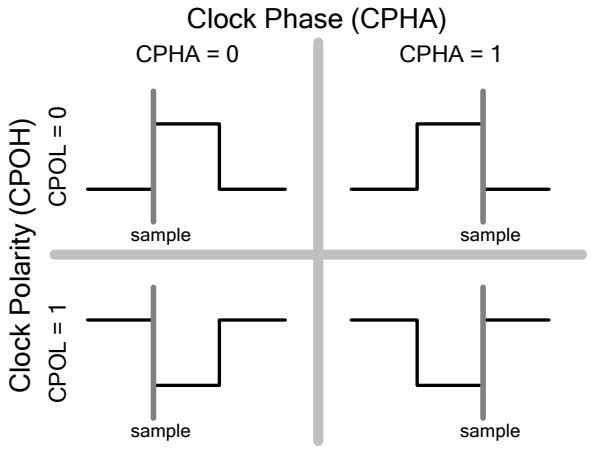


图 15-4. SPI 时钟格式

在SPI传输中，总是由主机启动传输。如果SPI被选定作为主模机式（MSTR = 1）并且打开传输（SPIEN =1），对主机的SPI数据寄存器（SPDR）写入内容将启动SPI时钟和数据传输。传出一个字节的同时会接受一个字节的内容，此后SPI时钟停止，主机和从机的SPIF（SPSR.7）同时被置1 。如果SPI中断使能位ESPI（EIE.6）设置为1，全局中断使能（EA= 1），芯片将执行SPI（ISR）的中断服务程序。

关于从机模式下，$\overline{SS}$信号需要注意。如图 15-4. SPI 时钟格式所示，CPHA=0时，第一个SPCLK边沿为MSB的采样点（LSBFE= 0，MSB优先发送为例）。因此，从机必须在SPCLK第一个采样边沿出现之前先把MSB传出。$\overline{SS}$的下降沿可用于准备MISO的MSB。因此，每次成功串行传输一个字节后，该引脚必须切换先高然后低，每个成功逐次串行字节之间。此外，如果从将数据写入SPI数据寄存器（SPDR）时，如果$\overline{SS}$为低电位，则会发生写冲突错

当CPHA = 1，采样边沿位于SPCLK时钟的第二个边沿。从机使用的第一个SPCLK时钟转移的MSB，而不是$\overline{SS}$的下降沿。因此，在每次成功传输时$\overline{SS}$可以始终保持低电位保持低之间的转移。此格式更适合单主机单从机的结构使用。从机的$\overline{SS}$可以不连接在SPI系统中，直接接地。

在SPI传输使能(SPIEN = 1)前，必须先对SPI传输进行配置，否则传输过程中对LSBFE, MSTR, CPOL, CPHA 及 SPR[1:0] 的任一更改，将会停止SPI传输并强迫总线进入空闲模式。所以需要对周期或状态的更改前，请先关闭SPIEN使能位。

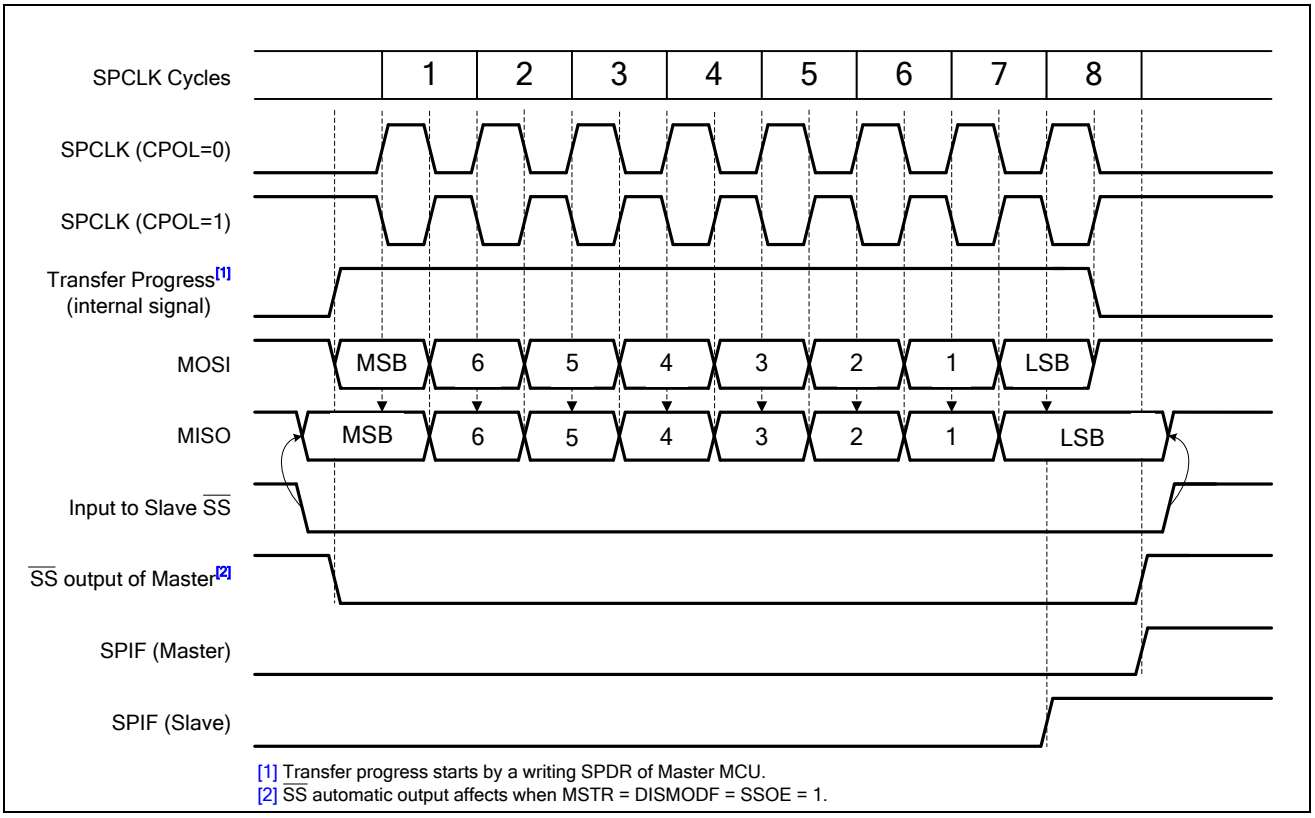


图 15-5. CPHA = 0 时SPI时钟与数据格式

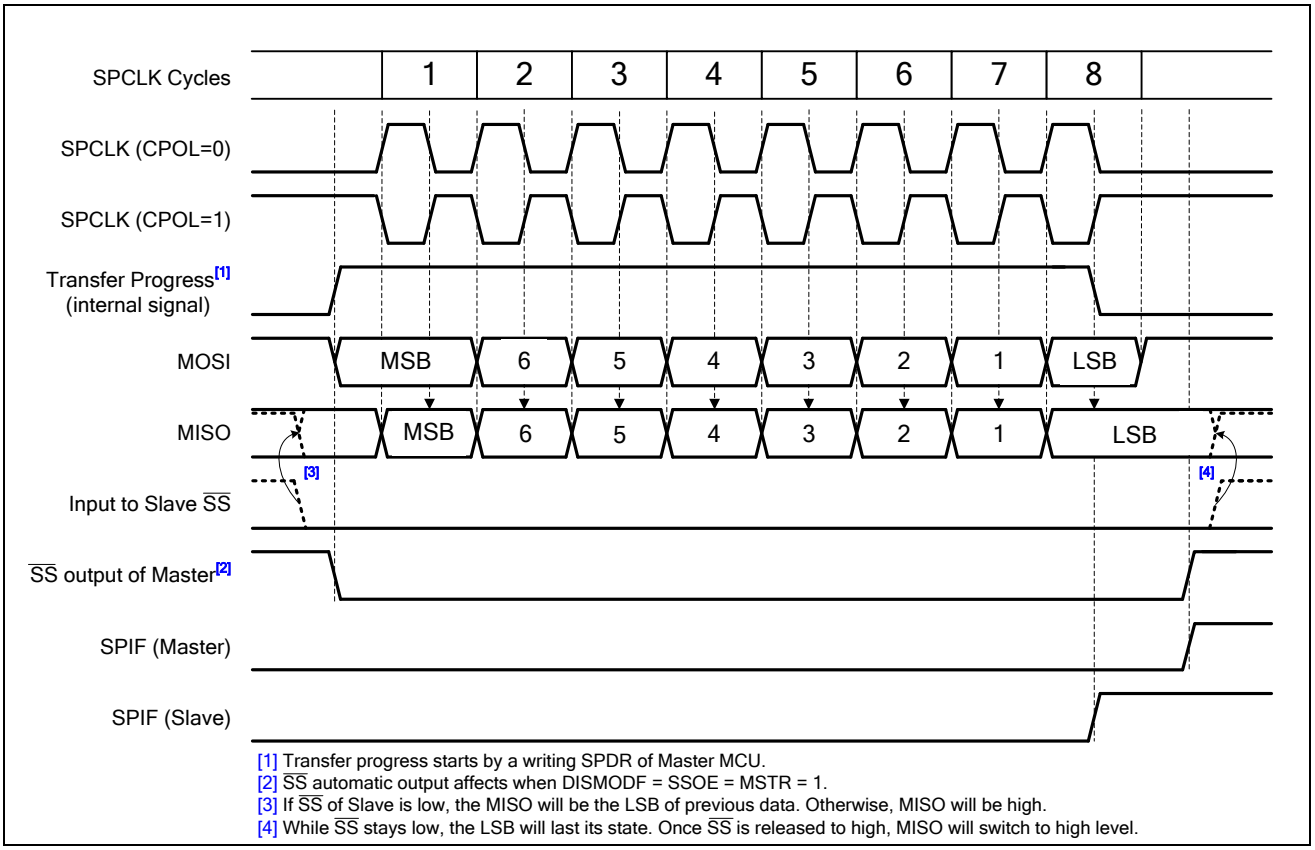


图 15-6. CPHA = 1时SPI 时钟与数据格式

15.4 从机选择SS脚配置

N76E885 SPI 提供灵活的 SS配置用于不同系统。当作为从机时， SS时钟定一位从机选择输入脚，当作为主机时， SS有三种不同的功能定义，可以通过DISMODF (SPSR.3) 和SSOE (SPCR.7)配置。默认情况DISMODF=0，故障侦测功能打开， SS配置为输入脚并检测是否发生故障。反之，如果DISMODF=1，故障侦测功能关闭，SSOE寄存器定义控制SS管脚。当SSOE=1，从机选择信号自动生成，主机的SS管脚直接与从机的SS脚连接，当选择外部从机进行传输时SS自动拉低，当进入闲置状态或者没有选择从机时，自动拉高。当SSOE=0且DISMODF=1时， SS不再用作SPI管脚，而完全配置为普通端口状态。

15.5 模式故障侦测

在一个SPI网络中，当不止一个设备有可能成为主机时，为减少数据传输错误，模式故障侦测功能是非常有用的。模式故障侦测发现SS由其它设备拉低，配置详见表 15–1.，说明系统上有一个从机试图寻找

主机地址并把注主机认为成从机。此时，硬件回自动将SPCR 的 MSTR 和SPIEN清除，从而SPI功能关闭，莫使错误侦测标志MODF (SPSR.4)置1 ，如果之前已打开中断ESPI (EIE .6) 和EA置1，则会进入中断向量。

15.6 写冲突错误

写冲突检测显示当正在进行一次传送时，设备正在试图写数据到SPDR。若之前一笔数据尚未移出，下一笔数据无发写入SPDR中。在传输过程中，对SPDR的直接写，将发生一个写冲突错误(WCOL(SPSR.6)会被置1)。由于SPDR不是双缓冲寄存器，直接写入就会将数据移入SPDR，总线上的错误就会产生。如果转移连续稳定没有受到干扰，那么导致错误的写数据是不会写进移位装置。一次写冲突通常是一个从机错误，原因是当主机开始一次传送时主机知道传送正在进行，所以主机没有理由产生写冲突错误，尽管SPI逻辑可以在主机和从机之间进行写冲突检测。WCOL标志用软件清除。

15.7 移出错误

对于接收数据，SPI是双向缓冲的。接收数据时，数据移入一个并行的数据缓冲去中，位移器同时清空以接收下一个数据。因此，在下一个数据传入之间，必须确保从SPDR中读取出当前数据。当第一个字节从缓冲区内被读取，并且SPIF被清零后，模块准备接收下一个数据，这样的传输不会产生溢出错误。反之，产生溢出错误时，第二个字节的数据没有正常传入，缓冲区内仍保留有第一个字节的数据。当发生溢出错误时，SPIOVF (SPSR.5)会被硬件置1。如果中断打开，会进入中断向量。图 15-7. SPI 溢出波形 表示接收数据与溢出错误之间的关系。

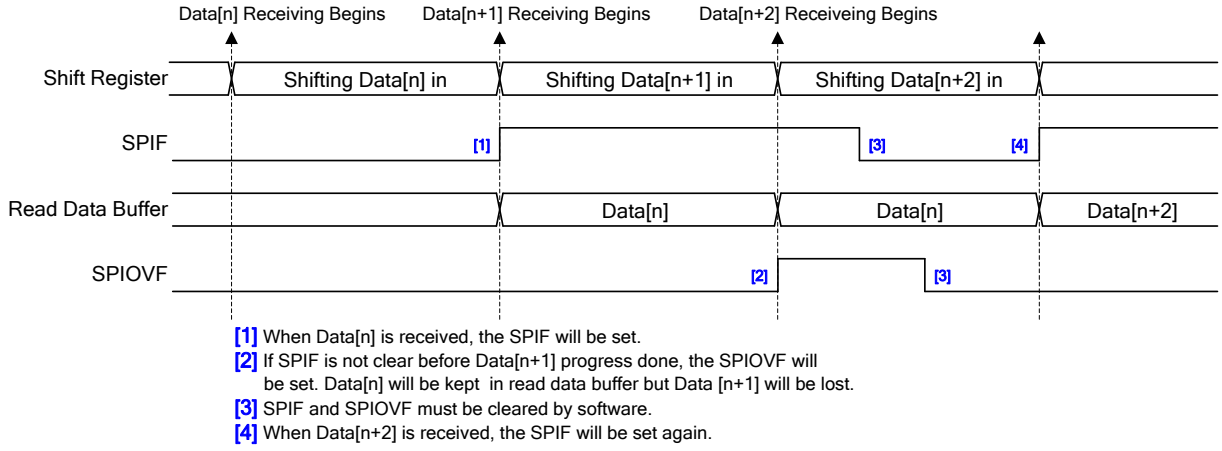


图 15-7. SPI 溢出波形

15.8 SPI 中断

SPI中断状态标志包括SPIF、MODF 和 SPIOVF 用于产生SPI中断需求。当有外部数据传入SPDR或自身完成数据传输后，这些位于SPSR和SPIF内的标志位会被置起。MODF置1时，表示 $\overline{SS}$ 进入模式错误状态，SPIOVF表示接收发生溢出错误。当SPI中断打开时（ESPI (EIE.6) 和EA置1），当这3个标志中的任意一个置1，CPU会执行SPI中断服务程序。用户若需要了解是由何种标志引起中断，必须检查相应的标志位。这三个标志必须由用户软件清除。

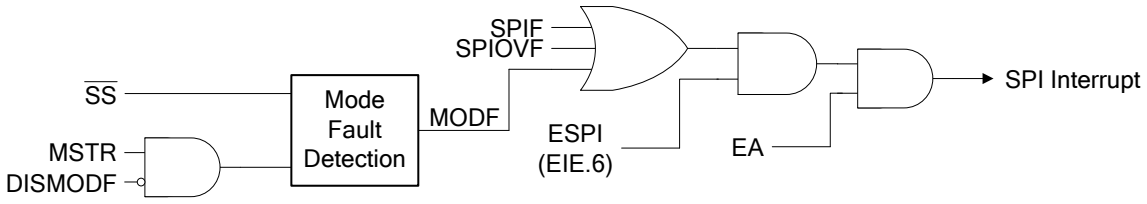


图 15-8. SPI 中断请求

16. I²C 总线

I²C 总线在 MCU 与 EEPROM，LCD模块，温度传感器等等之间，提供了一种串行通信方式。I²C 用两条线 (数据线SDA 和时钟线 SCL) 在设备间传输数据。

I²C 总线的数据线在主机与从机之间双向传输。可以用于多主机系统，支持无中央主机及多主机系统，主机与主机之间的总线仲载传输，同步时钟SCL的存在，允许设备间多种不同波特率的数据传输。 支持四种传输模式：主发，主收，从发，从收。I²C 总线仅支持 7位地址。支持广播呼叫，支持标准速率传输 (100kbps) 和快速传输 (400k bps) 。

16.1 功能描述

对于双向传输操作，SDA 及SCL 引脚必须配置成开漏配置，形成逻辑线与功能： 当有一个器件输出0，总线上就是0电平，所有器件全输出1，总线上才是高电平，即通过外接上拉电阻把电平拉高。N76E885, 在设置I2CEN (I2CON.6)使能I2C功能之前，必须把 P2.3， P0.6的输出锁存在逻辑1的状态。

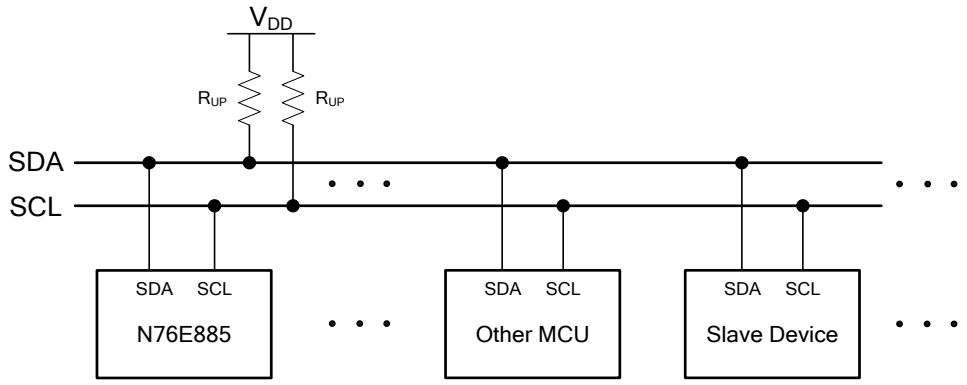


图 16-1. I²C 总线连接

I²C 空闲时，两条线都为高。这时任一设备都可以做为主机发个起始位 START 开始数据传输，在停止位 STOP 出现之前，总线被认为处于忙状态。 主机产生时钟以及起始位和停止位。 如果总线上没有START起始信号，则所有总线设备被认为未被寻址从机，硬件自动匹配自己的从机地址或广播呼叫地址，(广播地址可由 GC (I2ADDR.0)使能或禁止.)。若地址匹配，就产生中断。

I2C总线上传输的每个字节都包含8个数据位和一个应答位，共9位。但每次传输的字节个数没有明确界定(起始位 START 和停止位 STOP之间的字节个数) 。 主机产生8个时钟脉冲，以传输8位数据后，在第8个时钟SCL下沿，由SDA脚输出数据，并转为输入模式以读取检测第9位应答位。在第9个时钟脉冲后，数据接收端若没准备好接收下一个字节，可以拉住时钟线保持低，让传输挂起。接收端释放时钟线SCL以后，传输继续。

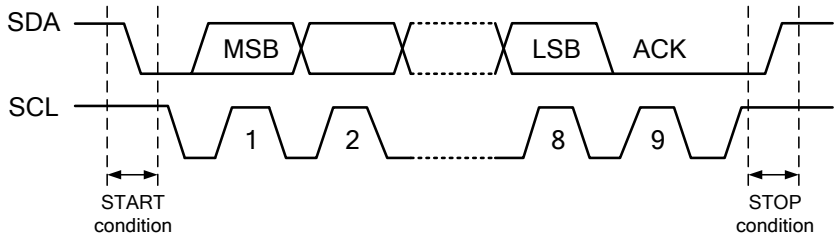


图 16-2. I²C 总线协议

16.1.1 起始START及结束 STOP条件

I²C 总线时序定义了起始START (S)和结束STOP (P)的条件。时钟SCL为高时，数据线SDA由高电平至低电平的跳变被认为是起始标志。时钟SCL为高时，数据线SDA由低电平至高电平的跳变被认为是结束标志。起始和结束都由主机产生，起始和结束之间被认为是总线忙状态。当成功判定结束条件以后，主机释放总线，所有设备都回到监听总线起始位状态，之前被呼叫从机也转为未寻址从机。I²C总线进入空闲状态等待下一个起始START信号，开始下一次传输。

主机若发出停止位STOP，传输就停止了，然而，这个主机可以不发停止位，而是再次发出起始START信号（Sr）继续和上个地址通信，或者换个地址继续通信。

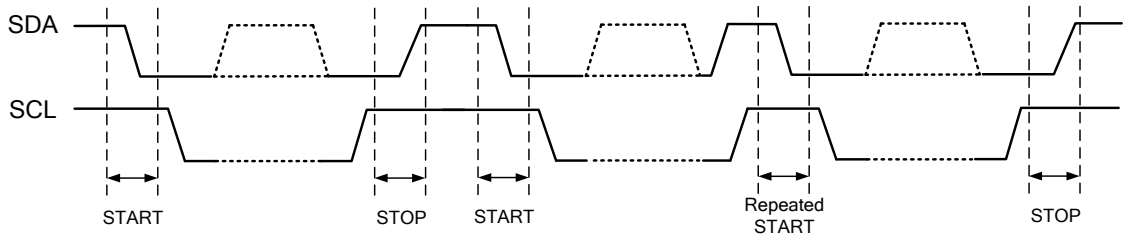


图 16-3. 起始START信号, 重复起始信号以及停止STOP条件

16.1.2 7位地址数据格式

起始位 START 之后, 第一个字节必须是“7位地址SLA+第8位读写方向位(R/W)”，用以定义目标从机地址以及写入或读出数据。若第8位是0，即SLA+R，表示下个字节开始主机向从机写数据；若是1，即SLA+W，就表示下字节开始，主机由从机读数据。所以，一个数据传输包含起始位 START，从机地址+读写位 SLA+W/R，一个或多个字节数据，最后是停止位 STOP。当第一字节已定义读写方向，随后的8位数据就跟随之前的设定进行传输。

I2C总线还有一种特殊寻址方式，广播呼叫寻址。在该模式下，发送的首字节数据为0。广播呼叫模式应用于主机希望向所有从机传输相同数据。当此寻址方式启用。收到广播要不要发应答由软件决定。若某

个从机发了应答，这个从机就收发后续数据，和标准主从收发方式相同。注意：地址0x00默认用于广播呼叫方式，不能用于普通从机地址。因此理论上，总共7位地址I^2C 总线，共可以连接127个设备，地址由1至127。

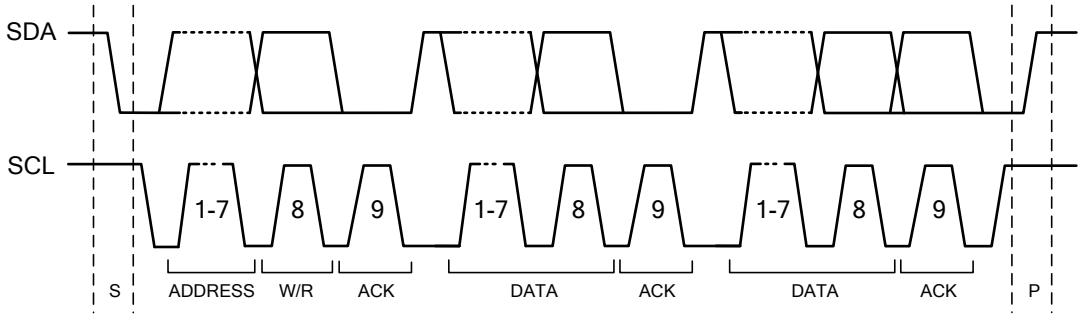


图 16-4.一组 I^2C 传输数据格式

在数据传输过程中，在时钟高电平时，SDA需要保持数据内容不能更改。只有在SCL为低时，SDA内容可以改变。

16.1.3 应答ACK

每字节传输SCL第9个脉冲用于传输应答位 (ACK)。用于传输器件之间，通过将SDA拉低，接收端（无论主机或是从机）回应发送端（无论主机或从机）所用。 应答位时钟由主机产生，发送端设备在应答位时钟高电平周期内，需放弃对SDA的控制。ACK 为一个低电平信号。在这个时钟周期的高电平时，SDA保持低电平用以表示接收端已成功接收到发送端的数据。通常被寻址的从机在整个传输过程中每字节都需要回复应答位。当该从机无应答（NACK），将SDA线保持高以便主机产生停止(STOP)或发送重复开始(START)信号。

若从机接收应答从机地址后，将自身切换到未定址从机模式，从而无法接收更多数据字节，并将SDA线拉高，此时主机应发送停止STOP信号或重复起始(repeated START)信号。

若是主机接收，主机控制着收发字节个数，主机在最后一个字节收发结束后不发应答，从机发送端将切换至未定址从机模式，并释放SDA线，以便主机直接发停止位STOP 或重起始位START。

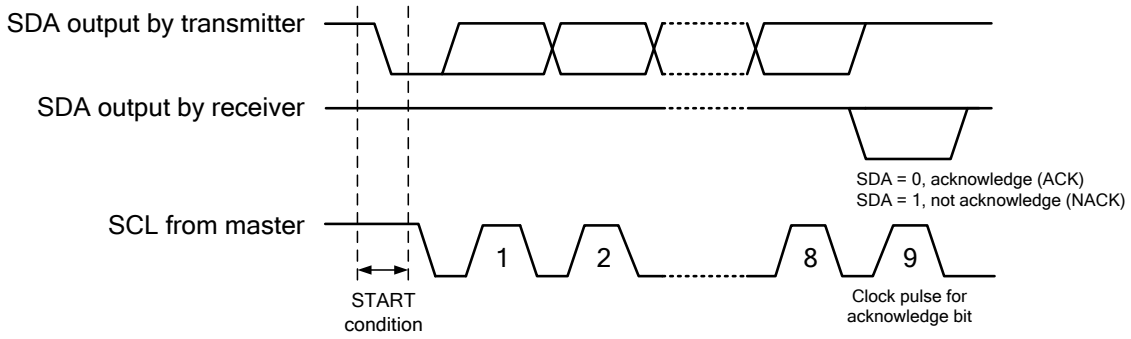


图 16-5. 应答位

16.1.4 仲裁

主机仅可在总线空闲是发起传输。可能有多个器件同时发起始位START试图发起数据传输，这时就会出现总线仲裁。在该状态下，当SCL为高时，SDA上呈现仲裁信号。在仲裁过程中，第一发起主机对SDA线置1（高电平）而另一个主机发送0（低电平），发送后主机会对SDA线上信号与自己发出的信号进行比较，由于“线与”的原因，时钟SCL为高时，发送0的主机会成功，而发送1的主机会失败。发送失败的主机立刻切换自身到未被寻址的从机状态，以确保自身能被仲裁胜利的主机寻址到。同时也释放数据线，并回到地址侦测状态，仲载失败的主机，仍会发送时钟，直到当前字节结束。

仲裁机制让每个主机发送数据时，都会同时比较总线上的数据是否与自己发送的一致。注：如果其它主机发送0，发送1的主机会在仲中失败仲裁，当仲裁会持续到总线上只有一个主机。如两个主机同时向一个从机发数据时，地址相同，仲裁会在第二个字节持续。

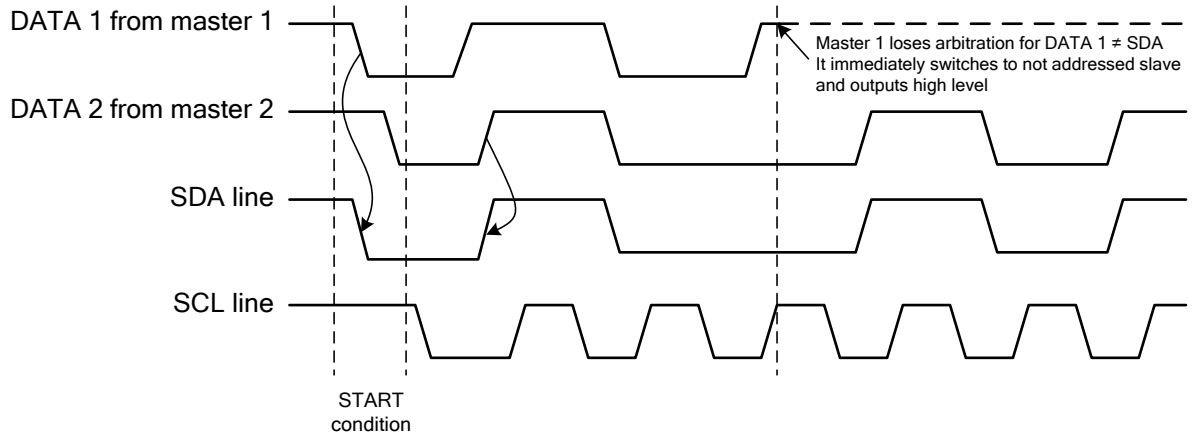


图 16-6. 两台主机仲裁过程

I^2C 总线的这种仲裁机制，让总线上的设备可以有多个主机，而且没有优先等级。

从机不介入仲裁。

16.2 I²C控制寄存器

I²C共有五个控制寄存器： I2CON, I2STAT, I2DAT, I2ADDR, 和 I2CLK. 这些寄存器用以提供协议控制，状态显示，数据传输、接收以及时钟速率控制。以下详述：

I2CON – I²C 控制寄存器 (可位寻址)

7	6	5	4	3	2	1	0
-	I2CEN	STA	STO	SI	AA	-	-
-	读/写	读/写	读/写	读/写	读/写	-	-

地址: C0H 复位值: 0000 0000b

位	名称	描述
6	I2CEN	**I²C 总线使能** 0 = I²C 禁止. 1 = I²C 使能. 使能 I²C之前, P2.3 和 P0.6 必须配置为输出1
5	STA	**起始标志START** 当STA置1，如果总线空闲，I²C产生START信号，如果总线忙，I²C等待停止条件STOP，然后产生START信号 如果总线已经在总线模式且已发送一个或多个字节，此时再设定STA，I²C 总线将产生重复开始信号repeated START 注：STA可在任何时间置1，包括从机模式。但硬件不会在发送START或repeat START信号后自动清0。用户需软件清除。
4	STO	**停止标志STOP** I²C 总线在主机模式下设定STO为1，将会向总线发送停止信号STOP。一旦总线上停止条件完成，STO由硬件自动清0。 当总线上产生错误状态(I2STAT 为 00H) STO 也会置1。这种情况下总线不会发送停止位。 如果STA和STO同时置1，且在主机模式下，I²C总线在发送START后马上发送STOP。如果在从机模式下，应避免STA及STO同时置1，以避免引发帧错误。
3	SI	**I²C 中断标志** I²C所有26种状态中出现一种，硬件就会置1此位 (F8H 除外)，此时软件根据读取I2STAT内值，决定下一步骤。 SI由软件清0.在SI被清0之前，SCL低电平周期延长，传输暂停，该状态对于从机处理接收到的数据非常有用，可以确保准确处理再接收下一笔数据。 SI 位被软件清0后，I²C外设才会继续下一步：根据软件控制位设定，继续发数据，或起始位，或停止位等，因此在清除SI位之前，软件需确认适当的状态。

位	名称	描述
2	AA	**应答响应标志** 若AA = 1，接收数据时，会在第9位发出应答ACK——拉低数据线。 若AA = 0，接收数据时，将向总线发送不应答（NACK），第9位时间不拉低数据线，若器件自我清除AA标志位，则会清除其从机地址或广播呼叫，SI会被清0，中断不予产生。所以MCU将不响应任何数据，包括从机地址。 从机接收时若AA=0不返回应答，或从机发送时没收到应答，从机传输就结束了。 注： 若已被寻址的从机，在从机接收模式下未回复应答位或在从机发送模式下未接收到应答位，该从机将变为未寻址从机，无法接收数据直到其AA被置1，且重新被主机定址。 特殊情况： 注意：从机发送时，状态码若为 C8H，从机发送最后一个字节之前，让 AA= 0，发送完最后一个字节，不再回应答，传输结束。主机若再从总线上读数据，将得到FFH。

I2STAT – I²C 状态寄存器

7	6	5	4	3	2	1	0
I2STAT[7:3]					0	0	0
R					R	R	R

地址: BDH　　复位值: 1111 1000b

位	名称	描述
7:3	I2STAT[7:3]	**I²C 状态字** 高5位为状态码，共有27种值。 I2STAT = F8H 时， 表示空闲，SI 将保持为0。 其它26种状态，都会让SI置1，且产生中断请求。
2:0	0	**保留位**

I2DAT – I²C 数据寄存器

7	6	5	4	3	2	1	0
I2DAT[7:0]							
读/写							

地址: BCH　　复位值: 0000 0000b

位	名称	描述
7:0	I2DAT[7:0]	**I²C 数据寄存器** 该寄存器存放准备发送的，或接收到的数据。只要SI = 1，此数据就有效。 数据发送时，数据移位到总线上，同时总线上的数据会接收回来，所以总线仲裁失败时，再读这个数据，可能与之前写入的值不一样。

I2ADDR – I²C 从机地址

7	6	5	4	3	2	1	0
I2ADDR[7:1]							GC
读/写							读/写

地址: C1H　　复位值: 0000 0000b

位	名称	描述
7:1	I2ADDR[7:1]	**I²C 从机地址** 主机模式: 无效 从机模式: 存放7位从机地址。主机需要定址该从机需在START之后写入该值。如果AA为1，该从机响应主机，成为被定址从机。否则主机呼叫地址会被忽略。 注：I2ADDR[7:1] 不能写为全0，因为0x00为广播呼叫方式寻址专用。
6	GC	**广播呼叫位** 主机模式: 无效 从机模式: 0 = 广播呼叫模式忽略，不响应。 1 = 如果AA置1，参与广播呼叫模式，若AA清0，忽略广播呼叫。

I2CLK – I²C 时钟寄存器

7	6	5	4	3	2	1	0
I2CLK[7:0]							
读/写							

地址: BEH　　复位值: 0000 1110b

位	名称	描述
7:0	I2CLK[7:0]	**I²C 时钟设定** 主机模式: 该寄存器设定作主机时I²C 总线时钟速率。算式如下： $\frac{F_{SYS}}{4\times(I2CLK+1)}$. 默认状态下，时钟频率为 400k bps (系统频率 24 MHz)。注I2CLK 值写入00H 及01H 无效。 从机模式: 该字节无效，从机自动跟随主机时钟，最高 400k bps.

16.3 工作模式

I²C 协议四种模式：主发送，主接收，从发送，从接收。还有一种特殊模式广播——与主发送类似。

16.3.1 主机发送模式

主机发送多个字节到从机，主机产生时钟，故需要在I2CLK内填入设定值。 主机发送模式需要将STA (I2CON.5) 置1。此时，一旦检测到总线空闲，主机就会发出一个起始位START，若成功, SI (I2CON.3) 将被置1，状态码 I2STAT 置为 08H。 接下来应把从机地址和写位(SLA+W)写入 I2DAT ，然后清0位 SI，总线上发出 SLA+W。

主机发出 SLA+W 收到从机应答位ACK后， SI 被置1，状态码 I2STAT = 18H。接下来将按照用户定义格式发送数据。所有的数据发送完以后，位 STO (I2CON.4) 置1， 并清0 SI位以发出停止信号STOP，或者也可以发送重复起始信号repeat START，而不发送STOP，直接开始新一轮数据传输。

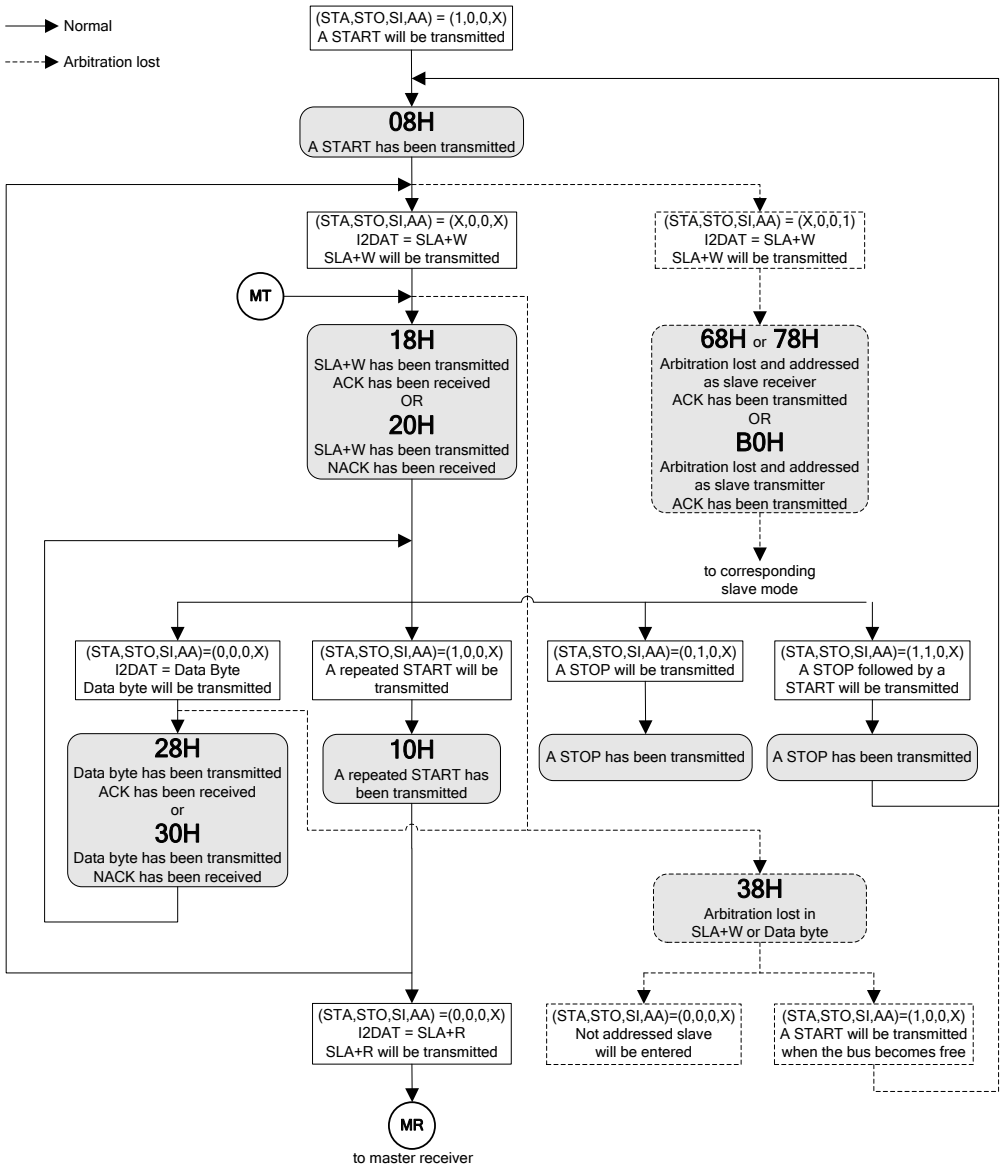


图 16-7. 主机发送模式流程图

16.3.2 主机接收模式

主机接收模式，由从机接收数据。初始化设置与主机发送模式相同，主机发送起始位以后， I2DAT 应写入从机地址和“读位” (SLA+R)。收到从机应答位ACK后 SI 被置1且状态码 I2STAT= 40H。 SI 清0后开始接收从机数据，若 AA 位 (I2CON.2) =1，主机收到数据后回应答位；若 AA =0主机收到数据后不回应答NACK。然后主机可以发停止位或重起始开始新一轮传输。

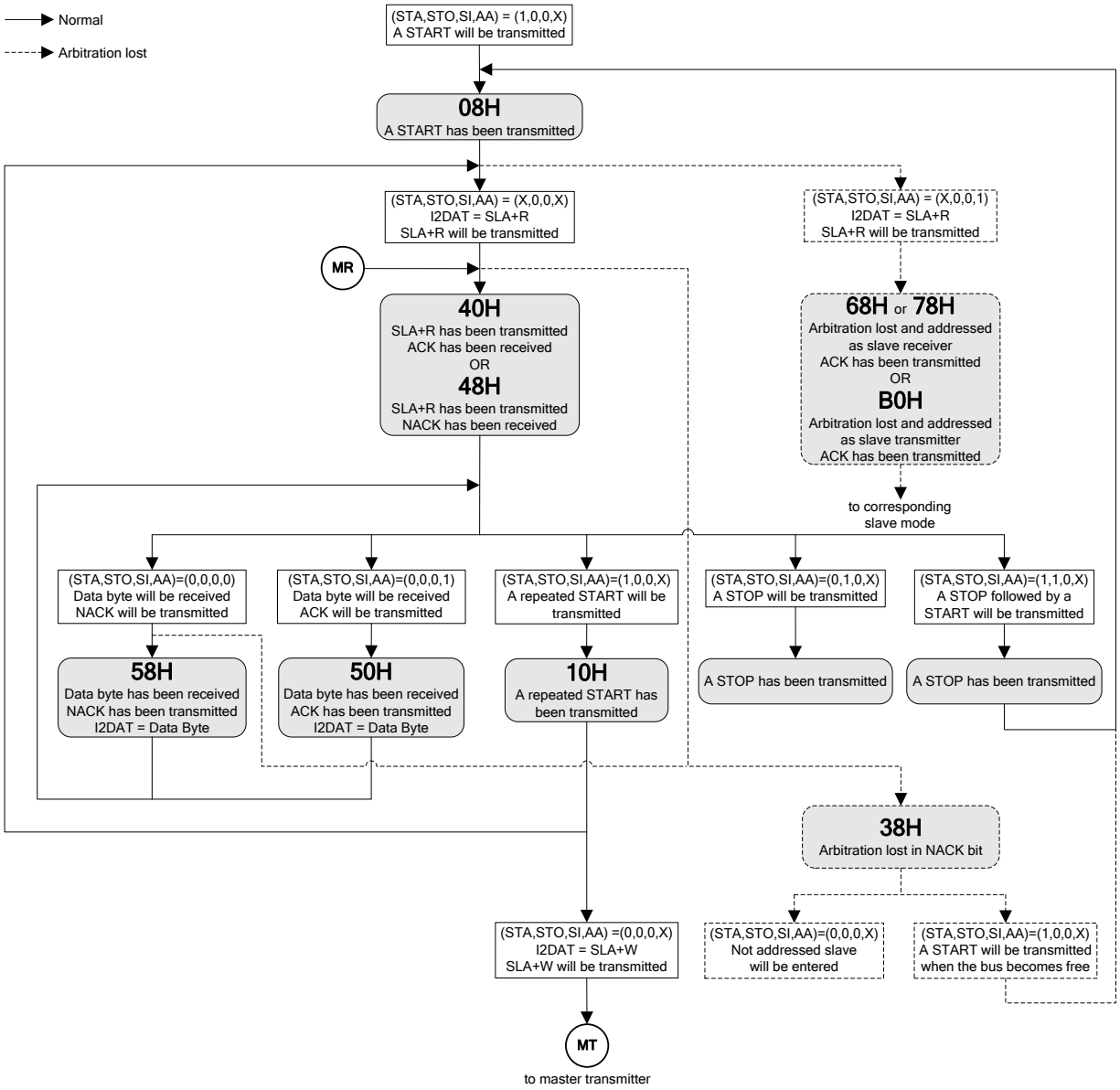


图 16-8. 主机接收模式流程图

16.3.3 从机接收模式

在从机接收模式下，从机接收主机发来的数据。在传输开始前，I2ADDR 应写入从机地址，I2CLK内容无效，AA置1用以应答对自身地址的定址。上述初始化后，从机进入空闲模式等待“写”信号（SLA+W）。若主机仲裁失败，也会直接进入从机接收模式。

当从机被“写”信号SLA+W寻址到后，需要清0 SI位，以便从主机接收数据。如果在传输过程中AA=0，从机将在下一字节返回无应答位NACK，从机也将转为未定址从机，与主机联系终止，不再接收数据，且I2DAT保持之前接收到的数据。

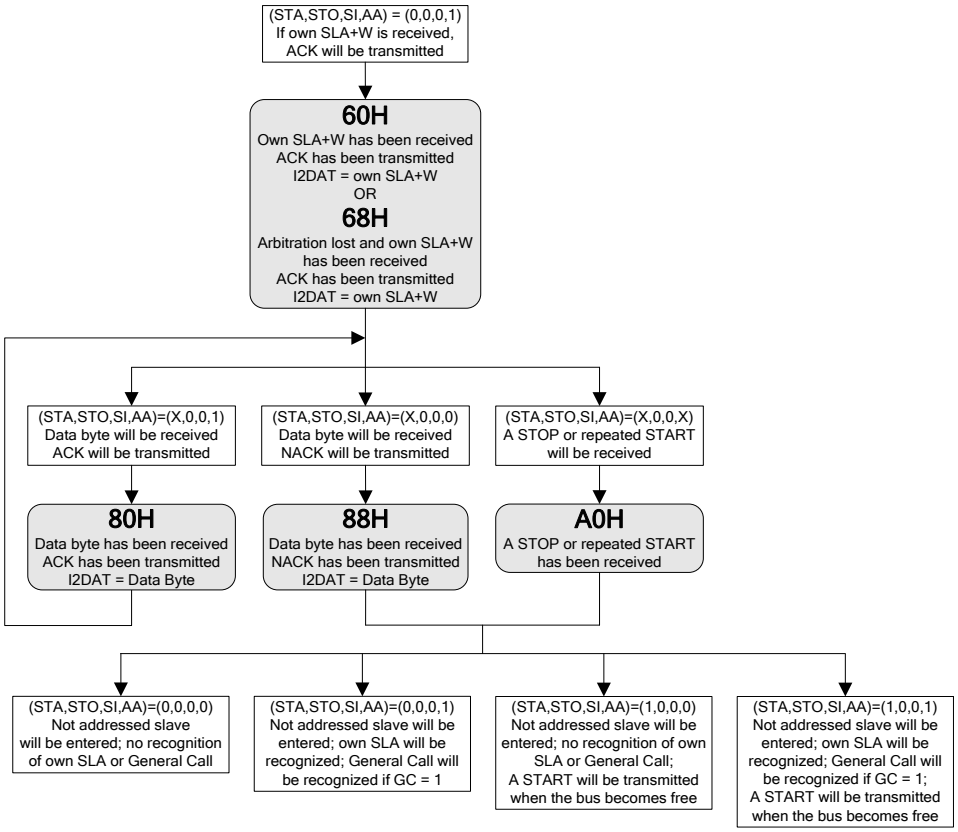


图 16-9. 从机接收模式流程图

16.3.4 从机发送模式

从机发送模式，数据由从机相助及放松。当给定I2ADDR及I2CON值后，器件等待直到自身地址被“读“信号(SLA+R)定址。若主机仲裁失败，也可进入从机发送模式。

当从机被“写”信号SLA+R定址，需要将SI信号清0用以向主机发送数据。通常主机接收每字节数据后会返回应答位，如果没有接收到应答位，接下去的传输中，从机将发送全1数据。并变为未定址从机。

如果传输过程中AA清0 ， 从机将发送最后一个字节数据，并在接下去的传输中发送全1数据，并将自身转为未定址从机。

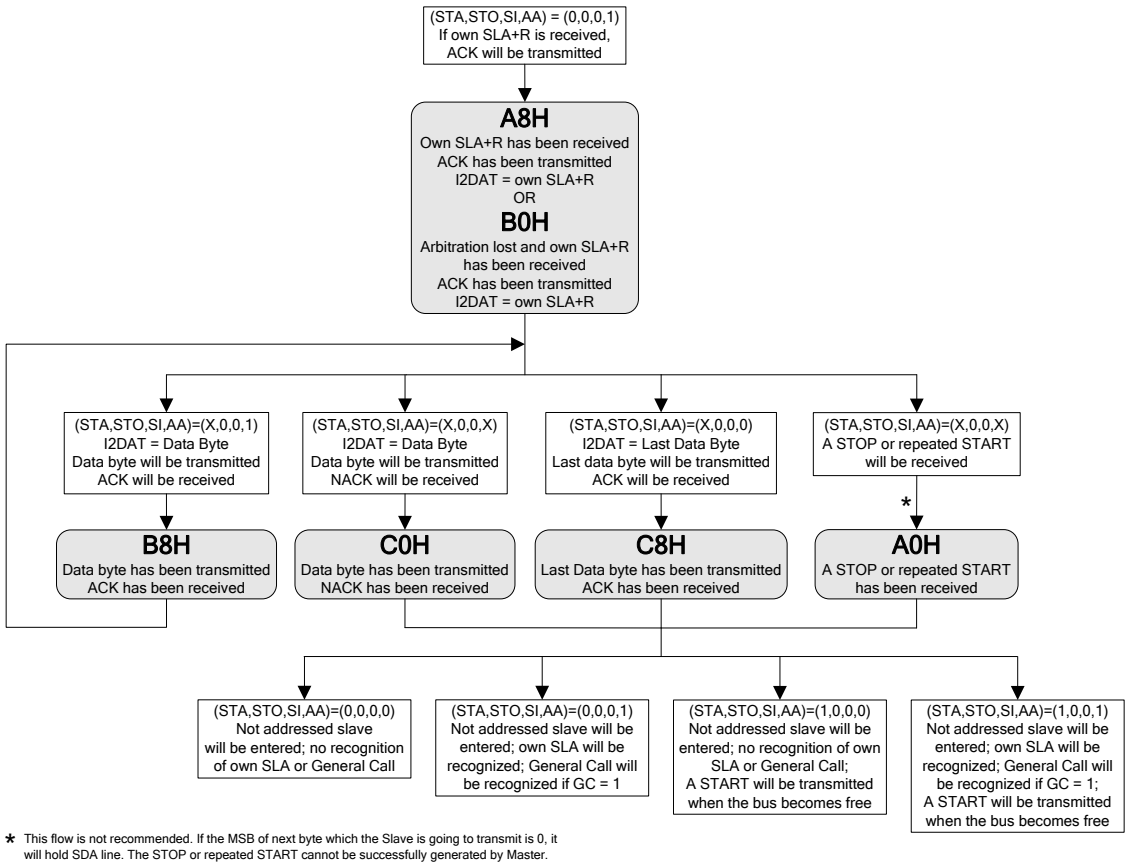


图 16-10. 从机发送模式

16.3.5 广播呼叫模式

广播呼叫模式是从机接收的一种特殊模式，定址方式为0x00。GC (I2ADDR.0) 位及AA 位置1，接收广播数据。在该模式下从机内I2STAT值与普通从机接收模式不同。仲裁失败也可能被误认为是广播数据。

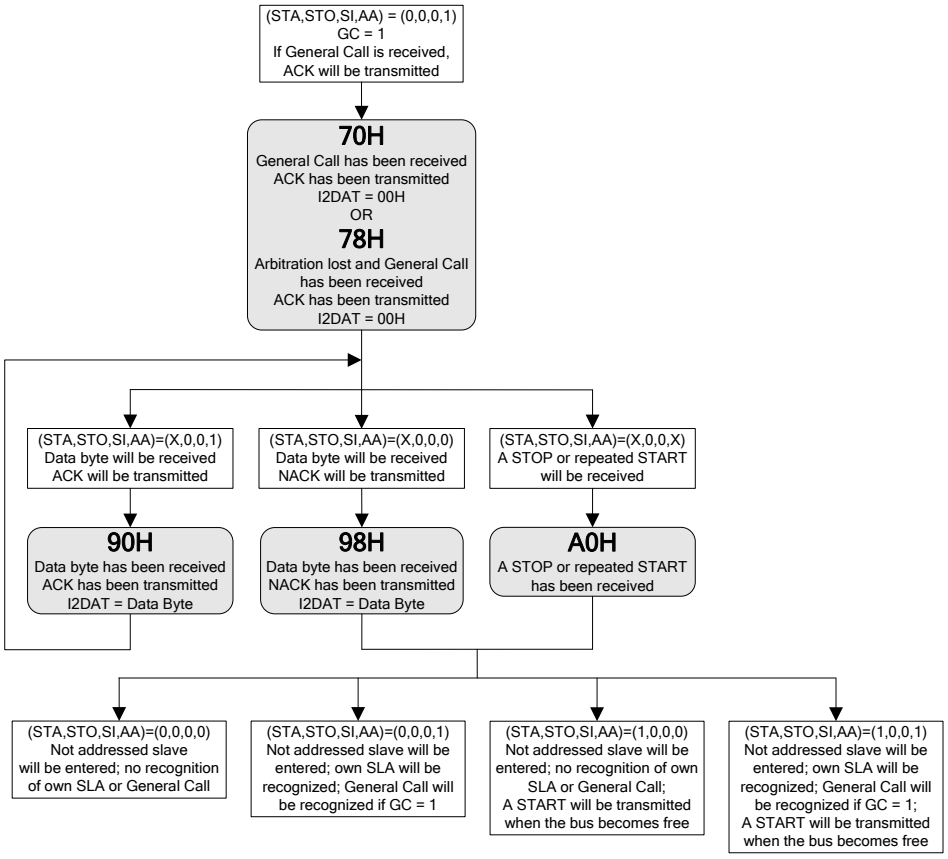


图 16-11. 广播呼叫模式流程图

16.3.6 状态字

I2STAT状态寄存器中,有两种状态字不归属25种之前所述状态：F8H 及 00H。

F8H 用以标示在之前的传输中没有有效信息，意味着，SI标志位为0且无I^2C 中断产生。

00H 标示在传输过程中，总线发生错误。总线错误是指START 或 STOP在传输的过程中出现在错误的位置，例如在地址或数据的第二位。当总线发生错误，SI标志马上会被置1，工作中的节点设备切换到未被定址从机模式，释放SDA 和SCL，并将I2STAT寄存器 清0。要回复总线状态，需要置1 STO位，清除SI位，然后STO会由硬件清0，在不需要STOP信号的条件下，释放总线回复正常空闲状态。

由一种特殊情况，从机失去同步，SDA线被强制拉低，START信号或重复起始(repeat START)信号无法成功产生。解决方法是在SCL线上额外多送一个时钟脉冲。通过将STA位置1，总线上产生额外的时钟脉冲，由于SDA始终拉低，SDA线上不产生START信号。一旦SDA线被释放，正常的START信号送出，状态寄存器上会显示08H，串行传输继续。相同的状况，如果需要发送重复开始 (repeated START)

信号受阻，也可以采用上述方式。在上述方式完成传输后，状态寄存器显示10H，而不是显示08H。

注：软件无法介入上述总线问题传输。

16.4 I²C 中断服务程序范例

下例为在KEIL™ C51 中I²C 中断服务程序范例，包含26中状态子程处理。用户可根据实际应用选取所需状态处理。删除状态时，请确认该状态不会产生。

```
void I2C_ISR (void) interrupt 6
{
       switch (I2STAT)
       {
              //============================================
              //Bus Error, always put in ISR for noise handling
              //============================================
              case 0x00:                          /*00H, bus error occurs*/
                     STO = 1;                     //recover from bus error
                     break;
              //===========
              //Master Mode
              //===========
              case 0x08:                          /*08H, a START transmitted*/
                     STA = 0;                     //STA  bit  should  be  cleared  by
software
                     I2DAT = SLA_ADDR1;           //load SLA+W/R
                     break;
              case 0x10:                          /*10H, a repeated START transmitted*/
                     STA = 0;
                     I2DAT = SLA_ADDR2;
                     break;
              //=======================
              //Master Transmitter Mode
              //=======================
              case 0x18:                          /*18H,   SLA+W   transmitted,   ACK
received*/
                     I2DAT = NEXT_SEND_DATA1;     //load DATA
                     break;
              case 0x20:                          /*20H,   SLA+W   transmitted,   NACK
received*/
                     STO = 1;                     //transmit STOP
                     AA = 1;                      //ready  for  ACK  own  SLA+W/R  or
General Call
                     break;
              case 0x28:                          /*28H,   DATA   transmitted,   ACK
received*/
                     if (Conti_TX_Data)           //if continuing to send DATA
                            I2DAT = NEXT_SEND_DATA2;
                     else                         //if no DATA to be sent
                     {
                            STO = 1;
                            AA = 1;
                     }
                     break;
```

```
                case 0x30:                          /*30H,    DATA    transmitted,    NACK
received*/
                      STO = 1;
                      AA = 1;
                      break;
                //===========
                //Master Mode
                //===========
                case 0x38:                          /*38H, arbitration lost*/
                      STA = 1;                      //retry to transmit START if bus free
                      break;
                //====================
                //Master Receiver Mode
                //====================
                case 0x40:                          /*40H,    SLA+R    transmitted,    ACK
received*/
                      AA = 1;                       //ACK next received DATA
                      break;
                case 0x48:                          /*48H,    SLA+R    transmitted,    NACK
received*/
                      STO = 1;
                      AA = 1;
                      break;
                case 0x50:                          /*50H,     DATA     received,     ACK
transmitted*/
                      DATA_RECEIVED1 = I2DAT;       //store received DATA
                      if (To_RX_Last_Data1)         //if last DATA will be received
                            AA = 0;                 //not ACK next received DATA
                      else                          //if continuing receiving DATA
                            AA = 1;
                      break;
                case 0x58:                          /*58H,     DATA     received,     NACK
transmitted*/
                      DATA_RECEIVED_LAST1 = I2DAT;
                      STO = 1;
                      AA = 1;
                      break;
                //====================================
                //Slave Receiver and General Call Mode
                //====================================
                case 0x60:                          /*60H,   own   SLA+W   received,   ACK
returned*/
                      AA = 1;
                      break;
                case 0x68:                          /*68H, arbitration lost in SLA+W/R
                                                      own SLA+W received, ACK returned */
                      AA = 0;                       //not ACK next received DATA after
                                                    //arbitration lost
                      STA = 1;                      //retry to transmit START if bus free
                      break;
                case 0x70:                          /*70H,   General   Call   received,   ACK
returned */
                      AA = 1;
                      break;
                case 0x78:                          /*78H, arbitration lost in SLA+W/R
                                                      General    Call    received,    ACK
returned*/
                      AA = 0;
                      STA = 1;
                      break;
```

```
                case 0x80:                          /*80H,   previous   own   SLA+W,   DATA
received,
                                                      ACK returned*/
                      DATA_RECEIVED2 = I2DAT;
                      if (To_RX_Last_Data2)
                            AA = 0;
                      else
                            AA = 1;
                      break;
                case 0x88:                          /*88H,   previous   own   SLA+W,   DATA
received,
                                                      NACK returned, not addressed SLAVE
mode
                                                      entered*/
                      DATA_RECEIVED_LAST2 = I2DAT;
                      AA = 1;                       //wait for ACK next Master addressing
                      break;
                case 0x90:                          /*90H,  previous  General  Call,  DATA
received,
                                                      ACK returned*/
                      DATA_RECEIVED3 = I2DAT;
                      if (To_RX_Last_Data3)
                            AA = 0;
                      else
                            AA = 1;
                      break;
                case 0x98:                          /*98H,  previous  General  Call,  DATA
received,
                                                      NACK returned, not addressed SLAVE
mode
                                                      entered*/
                      DATA_RECEIVED_LAST3 = I2DAT;
                      AA = 1;
                      break;
                //==========
                //Slave Mode
                //==========
                case 0xA0:                          /*A0H,   STOP   or   repeated   START
received while
                                                      still addressed SLAVE mode*/
                      AA = 1;
                      break;
                //======================
                //Slave Transmitter Mode
                //======================
                case 0xA8:                          /*A8H,   own   SLA+R   received,   ACK
returned*/
                      I2DAT = NEXT_SEND_DATA3;
                      AA = 1;                       //when AA is "1", not last data to be
                                                    //transmitted
                      break;
                case 0xB0:                          /*B0H, arbitration lost in SLA+W/R
                                                      own SLA+R received, ACK returned */
                      I2DAT = DUMMY_DATA;
                      AA = 0;                       //when AA is "0", last data to be
                                                    //transmitted
                      STA = 1;                      //retry to transmit START if bus free
                      break;
```

```
                case 0xB8:                          /*B8H,   previous   own   SLA+R,   DATA
transmitted,
                                                      ACK received*/
                      I2DAT = NEXT_SEND_DATA4;
                      if (To_TX_Last_Data)          //if last DATA will be transmitted
                            AA = 0;
                      else
                            AA = 1;
                      break;
                case 0xC0:                          /*C0H,   previous   own   SLA+R,   DATA
transmitted,
                                                      NACK received, not addressed SLAVE
mode
                                                      entered*/
                      AA = 1;
                      break;
                case 0xC8:                          /*C8H, previous own SLA+R, last DATA
trans-
                                                      mitted, ACK received, not addressed
SLAVE
                      AA = 1;                         mode entered*/
                      break;
          }//end of switch (I2STAT)

          SI = 0;                                   //SI  should  be  the  last  command  of
I2C ISR
          while(STO);                               //wait  for  STOP  transmitted  or  bus
error
                                                    //free, STO is cleared by hardware
}//end of I2C_ISR
```

16.5 I²C 超时

N76E885带一组14位超时计数器，已防止I²C总线无法释放。一旦使能了超时定时器，计数器开始计数直至溢出，即如果开启中断，I2TOF位会被置1。使能计数器，SI置1启动计数， SI清0停止计数。若I²C总线出现故障，SI 位长时间不能清0，超时定时器就会溢出，并进入中断。

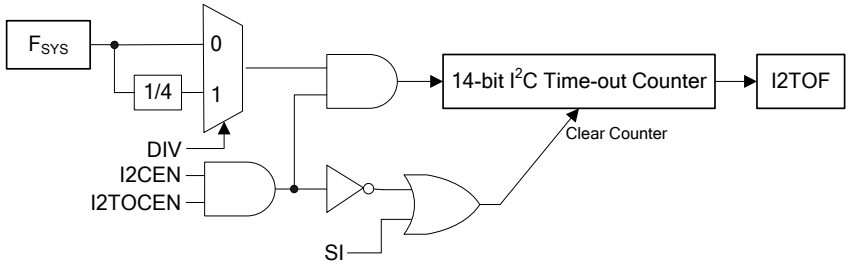


图 16-12. I²C 超时计数器

I2TOC – I²C 超时计数器

7	6	5	4	3	2	1	0
-	-	-	-	-	I2TOCEN	DIV	I2TOF
-	-	-	-	-	读/写	读/写	读/写

地址: BFH　　复位值: 0000 0000b

位	名称	描述
2	I2TOCEN	**I²C 超时计数器使能位** 0 = 关闭 1 = I²C 超时计数器使能.
1	DIV	**I²C 超时计数器计时除频** 0 = I²C 超时计数器除频 F_{SYS}/1. 1 = I²C 超时计数器除频 F_{SYS}/4.
0	I2TOF	**I²C 超时标志** 一旦超时计数器溢出，该位置1。用户软件清0.

16.6 I²C 中断

I²C的两个标志位：SI 和 I2TOF 置1，都会引起中断。如果 EI2C (EIE.0) = 1且 EA = 1, CPU 就会去执行中断代码。用户可以读取这两个标志位，来确定中断产生的原因。这两个标志需软件清0。

17. 管脚中断

N76E885 第一个管脚都有电平中断，或跳变中断功能，并且可以唤醒休眠状态的CPU。但最多可以配置8个引脚中断。

每个引脚中断的使能和极性都可以由 PIPEN 和 PINEN 单独控制。PICON 选择中断引脚。同时也用来定义中断类型：电平中断，或边沿侦测中断。 每条通路都有自己的中断标志，总共8个中断标志，存放在寄存器 PIF 中，进入中断后判断该寄存器以确定中断发生具体管脚。PIF由硬件置1，通过软件清0。

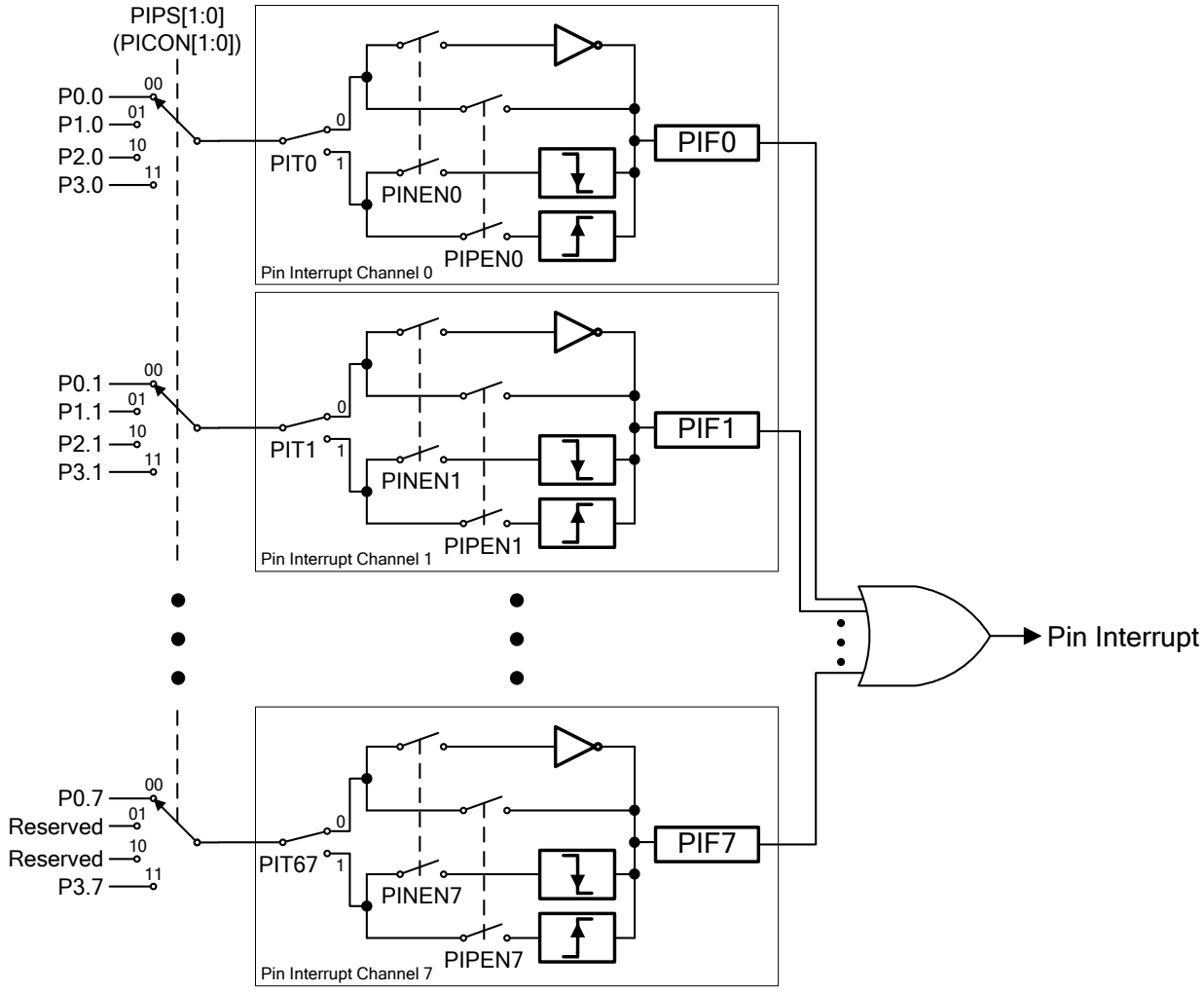


图 17-1. 管脚中断结构框图

引脚中断可唤醒芯片。通常用于在省电应用，例如芯片长时间进入空闲模式或掉电模式中，外部按键触发，以节省整体功耗。

PICON – 管脚中断控制寄存器

7	6	5	4	3	2	1	0
PIT67	PIT45	PIT3	PIT2	PIT1	PIT0	PIPS[1:0]	
读/写	读/写	读/写	读/写	读/写	读/写	读/写	

地址: E9H　　复位值: 0000 0000b

位	名称	描述
7	PIT67	**管脚中断通道6 及 7 类型选择** 该位用以配置管脚中断6和7产生中断的信号类型 0 = 电平触发 1 = 边沿触发
6	PIT45	**管脚中断通道4 及 5 类型选择** 该位用以配置管脚中断4和5产生中断的信号类型 0 = 电平触发 1 = 边沿触发
5	PIT3	**管脚中断通道3 类型选择** 该位用以配置管脚中断3产生中断的信号类型 0 = 电平触发 1 = 边沿触发
4	PIT2	**管脚中断通道2 类型选择** 该位用以配置管脚中断2产生中断的信号类型 0 = 电平触发 1 = 边沿触发
3	PIT1	**管脚中断通道1 类型选择** 该位用以配置管脚中断1产生中断的信号类型 0 = 电平触发 1 = 边沿触发
2	PIT0	**管脚中断通道0 类型选择** 该位用以配置管脚中断0产生中断的信号类型 0 = 电平触发 1 = 边沿触发
1:0	PIPS[:0]	**管脚中断端口选择** 该位段选择管脚中断所用8位端口 00 = 端口 0. 01 = 端口1. 10 = 端口2. 11 = 端口3.

PINEN – 管脚中断反相特性使能

7	6	5	4	3	2	1	0
PINEN7	PINEN6	PINEN5	PINEN4	PINEN3	PINEN2	PINEN1	PINEN0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: EAH 复位值: 0000 0000b

位	名称	描述
n	PINENn	**管脚中断反相特性选择使能位** 该位用以使能低电平/下降沿触发中断。至于是电平还是边沿，由PICON.寄存器的PITn位决定 0 = 关闭中断 1 = 低电平/ 下降沿触发.

PIPEN – 管脚触发正相特性使能

7	6	5	4	3	2	1	0
PIPEN7	PIPEN6	PIPEN5	PIPEN4	PIPEN3	PIPEN2	PIPEN1	PIPEN0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: EBH 复位值: 0000 0000b

位	名称	描述
n	PIPENn	**管脚中断正相特性选择使能位** 该位用以使能高电平/上升沿触发中断。至于是电平还是边沿，由PICON.寄存器的PITn位决定 0 = 关闭中断 1 = 高电平/ 上升沿触发.

PIF – 管脚中断标志寄存器

7	6	5	4	3	2	1	0
PIF7	PIF6	PIF5	PIF4	PIF3	PIF2	PIF1	PIF0
只读 (电平) 读/写 (边沿)	只读(电平) 读/写 (边沿)	只读(电平) 读/写 (边沿)	只读(电平) 读/写 (边沿)	只读(电平) 读/写 (边沿)	只读(电平) 读/写 (边沿)	只读(电平) 读/写 (边沿)	只读(电平) 读/写 (边沿)

地址: ECH 复位值: 0000 0000b

位	名称	描述
n	PIFn	**管脚中断通道标志位** 如果选择边沿触发有效，当通道的产生边沿跳变信号进入中断，该标志置1，需要由软件清除。 如果选择电平触发有效，该标志根据管脚上对应的电平变换，软件无法控制该位。

18. 脉宽调制电路(PWM)

PWM (脉宽调制电路) 在系统应用领域用途十分广泛。可用于电机驱动，风扇控制，背光调节，LED 光源调光或用作简单的低通滤波电路数模转换模块。

N76E885 PWM 模块特别设计，非常适用于电机控制，可产生四对，最高12位精度的PWM输出。该模块架构适用于驱动单相或三相无刷直流电机(BLDC)，或是三相交流感应电机。每对PWM输出可配置为独立输出模式、互补模式或是同步模式。当设定为互补模式时，还可以填入可配置的死区时间，用以对MOS管连续开关时进行保护。PWM波形可配置边沿对齐或中心对齐来选择中断响应位置。

18.1 功能描述

18.1.1 PWM 发生器

PWM 发生器时钟由系统时钟或定时器1计数溢出产生。可通过PWM时钟预分频调整1/1~1/128除频。PWM周期由12位寄存器 {PWMPH, PWMPL} 组合预先设定。该寄存器决定所有PWM的计数周期指。每对PWM的占空比由寄存器 {PWM01H, PWM01L}, {PWM23H, PWM23L}, {PWM45H, PWM45L}, 及 {PWM67H, PWM67L}设定。由此四对PWM可产生独立的占空比信号。设定反向寄存器，可使PWM产生与设定值周期及占空比完全相同，但信号电平相反的PWM信号。

为了更好适用于三相电机控制，可适用GP (PWMCON1.5)位来产生组群模式，这样 {PWM01H, PWM01L} 占空比寄存器决定三相PWM输出。在三相电机控制应用中，三对PWM输出产生完全相同的占空比信号，一旦组群模式启用 {PWM23H, PWM23L} 及 {PWM45H, PWM45L} 寄存器失效

注：启用PWM输出，芯片不会自动配置管脚为“输出模式”，用户需要通过软件配置。

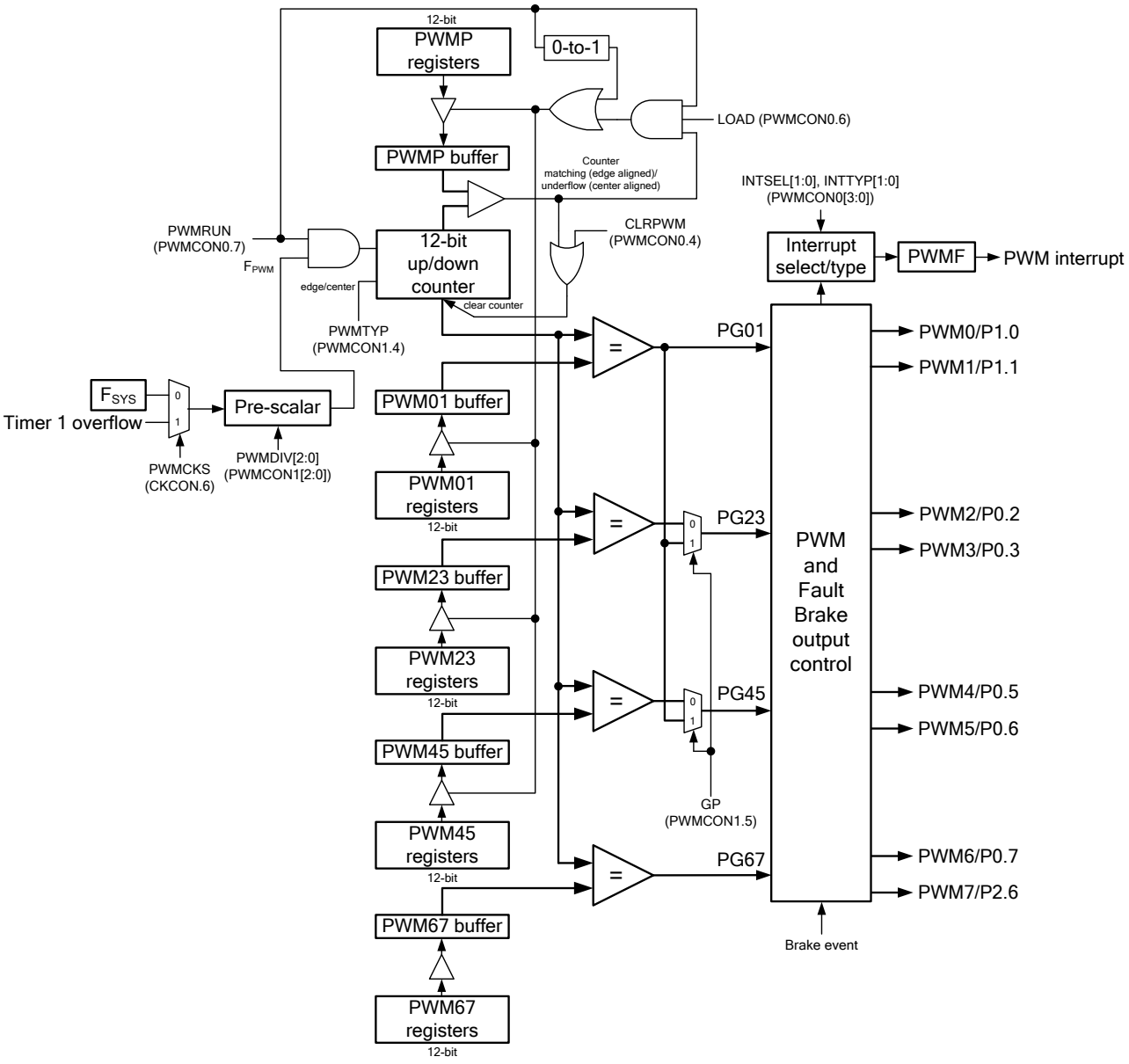


图 18-1. PWM 模块框图

PWM 计数器产生四个PWM信号，称为PG01, PG23, PG45, 及 PG67。该信号通过PWM产生及故障刹车输出控制电路直至输出管脚 。输出控制电路决定输出PWM信号的模式、死区时间、输出掩码、故障刹车以及PWM的极性。最后一个特性是通过管脚特性配置来设定。用户需要通过PIOn位来决定需要输出的PWM信号极性。

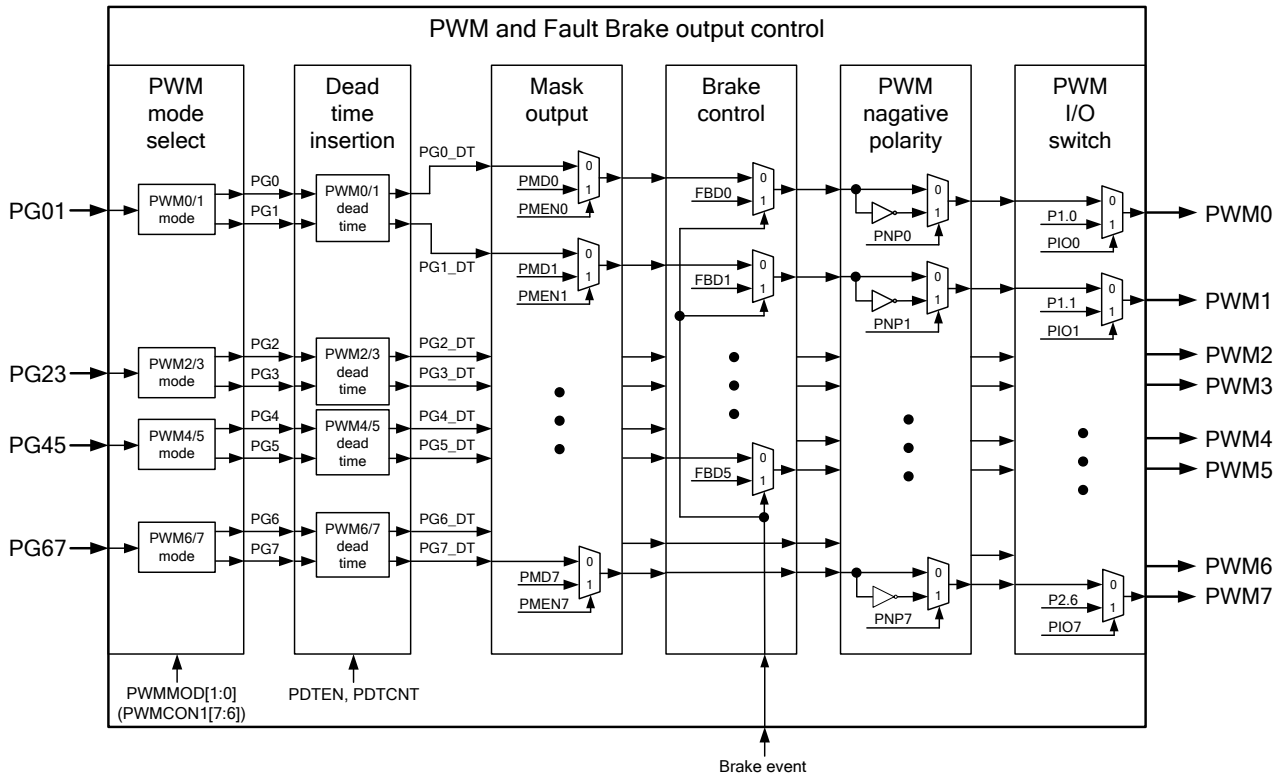


图 18-2. PWM 以及故障刹车控制模块图

请按照如下初始化步骤来产生PWM信号。第一步，设定CLRPWM (PWMCON0.4)位，清零计数器以确保12位向上计数器计数正确。然后设定{PWMPH, PWMPL} 及所有 {PWMnH, PWMnL}寄存器。对PWMRUN (PWMCON0.7) 置1，开始12位向上计数器计数。PWM信号开始产生，输出管脚输出PWM信号。所有的周期及占空比寄存器，具有硬件双缓存设计，因此 {PWMPH, PWMPL} 及 {PWMnH, PWMnL} 寄存器可随时被改写，但不会马上响应更改内容更新输出，而是于寄存器 LOAD (PWMCON0.6) 置1后，开始输出更改后的PWM信号。用于防止产生非完整周期的PWM波形。

采用LOAD更新PWM周期及占空比寄存器值的过程必须特别注意，等待LOAD信号由硬件清零再执行其他更改。任何当LOAD还在保持1的时候，对周期或占空比寄存器内容的更改，可能引发无法预测的结果。

PWMCON0 – PWM 控制寄存器 0 (可位寻址)

7	6	5	4	3	2	1	0
PWMRUN	LOAD	PWMF	CLRPWM	INTTYP1	INTTYP0	INTSEL1	INTSEL0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: D8H　　复位值: 0000 0000b

位	名称	描述
7	PWMRUN	**PWM 运行使能位** 0 = PWM 模块空闲. 1 = PWM 开始运行
6	LOAD	**PWM 载入新周期及占空比** 该位用于载入周期及占空比所新设定的值。当前一个PWM周期输出结束，载入动作才会开始。更改的周期及占空比值将在下一个周期展现。当载入动作结束，硬件自动将LOAD位清0。这一特性会使得LOAD位写入及读出值可能不一致，意义也不相同。. 写: 0 = 不动作. 1 = 载入之前在缓存内存入的周期及占空比值. 读: 0 = 载入动作已完成 1 = 正在执行载入动作
5	PWMF	**PWM 标志位** 该位根据PWMCON1 的 INTSEL[1:0] 及 INTTYP[1:0] 位来设定。由软件清除。
4	CLRPWM	**清除 PWM计数器** 该位置1，会直接清零PWM12位计数器至000H。当清除计数器动作完成，硬件会自动将该位清0。这特性会使得CLRPWM位写入和读出值不一致，意义也不相同。 写: 0 = 无动作. 1 = 清除12位PWM计数器. 读: 0 = PWM12位计数器已清除。 1 = 12位计数器内还有数值，未清除。

PWMCON1 – PWM 控制寄存器 1

7	6	5	4	3	2	1	0
PWMMOD[1:0]		GP	PWMTYP	FBINEN	PWMDIV[2:0]		
读/写		读/写	读/写	读/写	读/写		

地址: DFH 复位值: 0000 0000b

位	名称	描述
5	GP	**群组模式使能位** 该位使能PWM群组模式。一旦使能，三对PWM的占空比由PWM01H 和 PWM01L决定，原本配置的数据失效。 0 = 未组成群组模式. 1 = 群组模式使能
2:0	PWMDIV[2:0]	**PWM 时钟除频** 该寄存器段用于配置PWM时钟频率预分频。 000 = 1/1. 001 = 1/2 010 = 1/4. 011 = 1/8. 100 = 1/16. 101 = 1/32. 110 = 1/64. 111 = 1/128.

CKCON – 时钟控制寄存器

7	6	5	4	3	2	1	0
-	PWMCKS	-	T1M	T0M	-	-	-
-	读/写	-	读/写	读/写	-	-	-

地址: 8EH 复位值: 0000 0000b

位	名称	描述
6	PWMCKS	**PWM 时钟源选择** 0 = PWM时钟源为系统时钟F_{SYS}. 1 = PWM时钟源为定时器1

PWMPL – PWM 周期寄存器低字节

7	6	5	4	3	2	1	0
PWMP[7:0]							
读/写							

地址: D9H 复位值: 0000 0000b

位	名称	描述
7:0	PWMP[7:0]	**PWM 周期值低字节** 该寄存器存储与PWMPH搭配存储PWM的周期值，该位段位低字节

PWMPH – PWM 周期寄存器高字节

7	6	5	4	3	2	1	0
-	-	-	-	PWMP[11:8]			
-	-	-	-	读/写			

地址: D1H　　复位值: 0000 0000b

位	名称	描述
3:0	PWMP[11:8]	**PWM周期寄存器高字节** 与PWMPL搭配组成PWM周期信号

PWM01L – PWM0/1 占空比寄存器低字节

7	6	5	4	3	2	1	0
PWM01[7:0]							
读/写							

地址: DAH　　复位值: 0000 0000b

位	名称	描述
7:0	PWM01[7:0]	**PWM0/1 占空比寄存器低字节** 该位于PWM01H 搭配，配置由PG01输出的占空比数据

PWM01H – PWM0/1 占空比寄存器高字节

7	6	5	4	3	2	1	0
-	-	-	-	PWM01[11:8]			
-	-	-	-	读/写			

地址: D2H　　复位值: 0000 0000b

位	名称	描述
3:0	PWM01[11:8]	**PWM0/1 占空比数据高字节** 该位于PWM01L搭配，配置由PG01输出的占空比数据

PWM23L – PWM2/3 占空比寄存器低字节

7	6	5	4	3	2	1	0
PWM23[7:0]							
读/写							

地址: DBH　　复位值: 0000 0000b

位	名称	描述
7:0	PWM23[7:0]	**PWM2/3 占空比低字节** 该位于PWM23H搭配，配置由PG23输出的占空比数据

PWM23H – PWM2/3占空比寄存器高字节

7	6	5	4	3	2	1	0
-	-	-	-	PWM23[11:8]			
-	-	-	-	读/写			

地址: D3H 复位值: 0000 0000b

位	名称	描述
3:0	PWM23[11:8]	**PWM2/3占空比数据高字节** 该位于PWM2/3L搭配，配置由PG23输出的占空比数据

PWM45L – PWM4/5占空比寄存器低字节

7	6	5	4	3	2	1	0
PWM45[7:0]							
读/写							

地址: DDH 复位值: 0000 0000b

位	名称	描述
7:0	PWM45[7:0]	**PWM4/5 占空比低字节** 该位于PWM45H搭配，配置由PG45输出的占空比数据

PWM45H – PWM4/5占空比寄存器高字节

7	6	5	4	3	2	1	0
-	-	-	-	PWM45[11:8]			
-	-	-	-	读/写			

地址: D5H 复位值: 0000 0000b

位	名称	描述
3:0	PWM45[11:8]	**PWM4/5占空比数据高字节** 该位于PWM4/5L搭配，配置由PG45输出的占空比数据

PWM67L – PWM6/7占空比寄存器低字节

7	6	5	4	3	2	1	0
PWM67[7:0]							
读/写							

地址: DCH 复位值: 0000 0000b

位	名称	描述
7:0	PWM67[7:0]	**PWM6/7占空比低字节** 该位于PWM67H搭配，配置由PG67输出的占空比数据

PWM67H – PWM6/7占空比寄存器高字节

7	6	5	4	3	2	1	0
-	-	-	-	PWM67[11:8]			
-	-	-	-	读/写			

地址: D4H 复位值: 0000 0000b

位	名称	描述
3:0	PWM67[11:8]	**PWM6/7占空比数据高字节** 该位于PWM6/7L搭配，配置由PG67输出的占空比数据

PIO – PWM 或 I/O 选择寄存器

7	6	5	4	3	2	1	0
PIO7	PIO6	PIO5	PIO4	PIO3	PIO2	PIO1	PIO0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: DEH 复位值: 0000 0000b

位	名称	描述
7	PIO7	**P2.6/PWM7管脚功能选择位** 0 = P2.6/PWM7管脚用作P2.6. 1 = P2.6/PWM7管脚用作PWM7输出.
6	PIO6	**P0.7/PWM6管脚功能选择位** 0 = P0.7/PWM6管脚用作P0.7. 1 = P0.7/PWM6管脚用作PWM6输出.
5	PIO5	**P0.6/PWM5管脚功能选择位** 0 = P0.6/PWM5管脚用作P0.6. 1 = P0.6/PWM5管脚用作PWM5输出.
4	PIO4	**P0.5/PWM4管脚功能选择位** 0 = P0.5/PWM4管脚用作P0.5. 1 = P0.5/PWM4管脚用作PWM4输出.
3	PIO3	**P0.3/PWM3管脚功能选择位** 0 = P0.3/PWM3管脚用作P0.3. 1 = P0.3/PWM3管脚用作PWM3输出.
2	PIO2	**P0.2/PWM2管脚功能选择位** 0 = P0.2/PWM2管脚用作P0.2. 1 = P0.2/PWM2管脚用作PWM2输出.
1	PIO1	**P1.1/PWM1管脚功能选择位** 0 = P1.1/PWM1管脚用作P1.1. 1 = P1.1/PWM1管脚用作PWM1输出.
0	PIO0	**P1.0/PWM0管脚功能选择位** 0 = P1.0/PWM0管脚用作P1.0. 1 = P1.0/PWM0管脚用作PWM0输出.

18.1.2 PWM 类型

PWM发生器可以配置位两种类型，边沿对齐或中心对齐，由PWMTYP (PWMCON1.4) 位决定。

PWMCON1 – PWM 控制寄存器 1

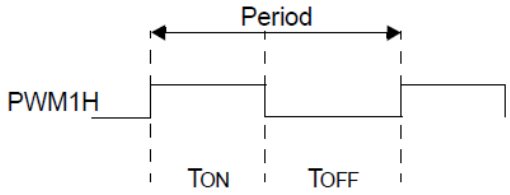
7	6	5	4	3	2	1	0
PWMMOD[1:0]		GP	PWMTYP	FBINEN	PWMDIV[2:0]		
读/写		读/写	读/写	读/写	读/写		

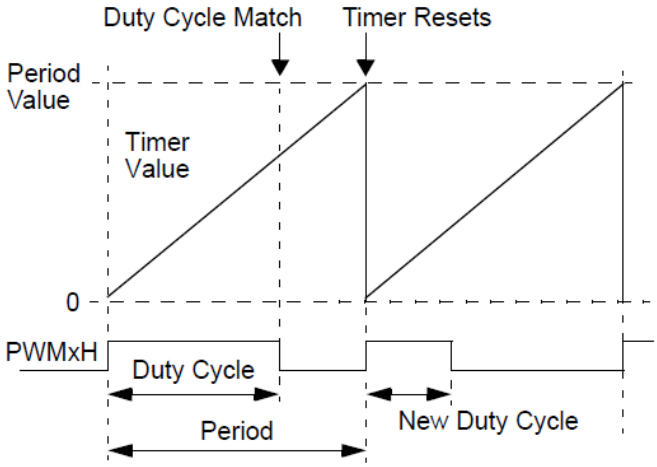
地址: DFH 复位值: 0000 0000b

位	名称	描述
4	PWMTYP	**PWM 类型选择** 0 = 边沿对齐型 PWM. 1 = 中心对齐型 PWM.

18.1.2.1 边沿对齐型

边沿对齐模式，12位计数器设定为单周期模式，从000H 至{PWMPH, PWMPL} 向上计数模式由000H开始计数。PWM计数器开始运行，PWM输出信号置1，（PGn不经过模块PWM故障刹车输出控制），当达到12位占空比计数器{PWMnH, PWMnL}设定值后PWM输出清0，同时占空比寄存器清0。PWM输出波形为左边沿对齐方式。





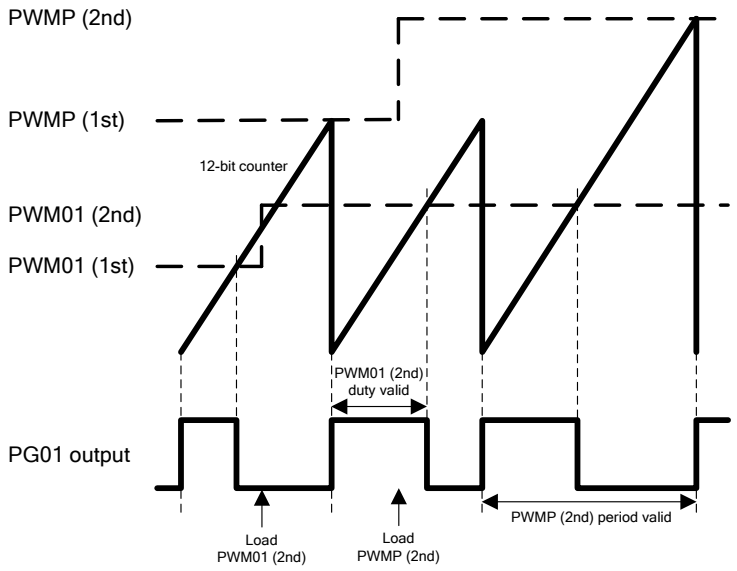


图 18-3. PWM 边沿对齐方式波形图

PWM边沿对齐方式输出频率及占空比算式如下：

$$\text{PWM 频率} = \frac{F_{PWM}}{\{PWMPH,PWMPL\}+1}$$ (F_{PWM} 为PWM时钟源除以PWMDIV).

$$\text{PWM 占空比高电平} = \frac{\{PWMnH,PWMnL\}}{\{PWMPH,PWMPL\}+1}.$$

18.1.2.2 中心对齐模式

中心对齐模式，12位计数器采用双周期模式，从000H开始向上计数至{PWMPH, PWMPL}然后由{PWMPH, PWMPL}向下计数至000H。当计数器计数至12位占空比寄存器设定值时，PGn输出信号清0，随后当向下计数至占空比寄存器设定值时置1。中心对齐型PWM用于产生非重叠波形。

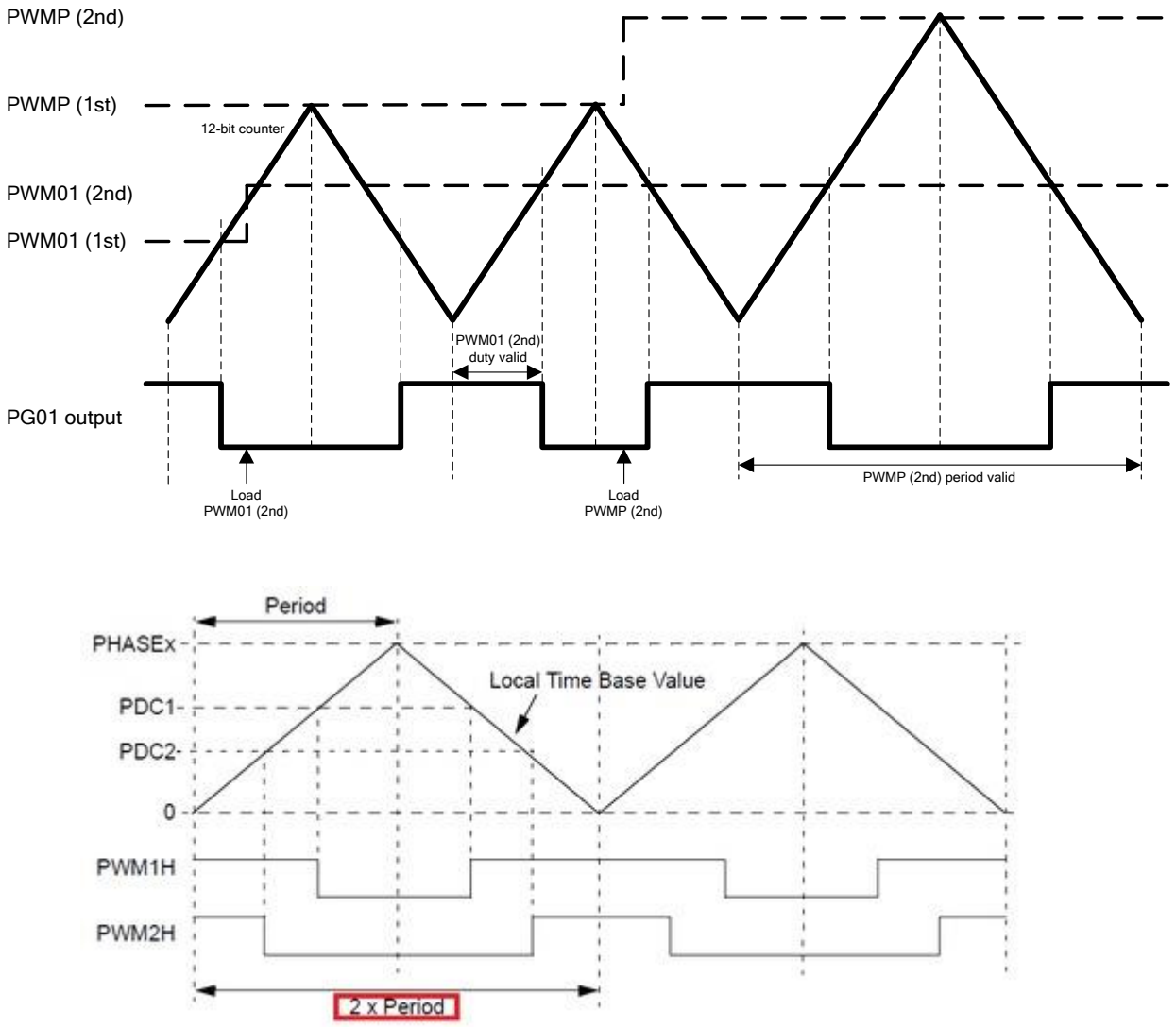


图 18-4. PWM中心对齐模式波形图

中心对齐模式输出频率及占空比算式如下：

$$\text{PWM 频率} = \frac{F_{PWM}}{2\times\{PWMPH,PWMPL\}}$$ (F_{PWM}为PWM时钟源除以PWMDIV).

$$\text{PWM 占空比高电平} = \frac{\{PWMnH,PWMnL\}}{\{PWMPH,PWMPL\}}.$$

18.1.3 工作模式

当 PGn 输出信号通过PWM故障刹车控制模块。PWM模式选择模块可产生不同类型的4组，8通道PWM输出模式PG0~PG7。支持独立输出模式，互补模式及同步模式。

PWMCON1 – PWM控制寄存器1

7	6	5	4	3	2	1	0
PWMMOD[1:0]		GP	PWMTYP	FBINEN	PWMDIV[2:0]		
读/写		读/写	读/写	读/写	读/写		

地址: DFH　　复位值: 0000 0000b

位	名称	描述
7:6	PWMMOD[1:0]	**PWM 模式选择** 00 = 独立输出模式 01 = 互补模式 10 = 同步模式 11 = 保留位

18.1.3.1 独立输出模式

当PWMMOD[1:0] (PWMCON1[7:6]) 设定为 [0:0]，PWM为独立输出模式。该模式为默认输出模式。PG0/2/4/6 输出PWM信号，PWM输出PG1/3/5/7保持高电平。

18.1.3.2 带死区控制的互补模式

当PWMMOD[1:0] = [0:1]，设定位互补模式。在该模式中PG0/2/4/6输出信号与独立输出模式相同，但PG1/3/5/7 输出PG0/2/4/6的反向信号。该模式可使PG0/PG1形成一对相对的PWM输出。同样PG2/PG3, PG4/PG5, 及PG6/PG7也是如此。

在实际的电机应用中，互补模式PWM输出需要加入“死区”控制用以防止电源切换时器件损伤，对于无法做到连续的开关电源器件，N76E885 每对PWM共享一组0位向下计数器PDTCNT，用以在每对两个PWM信号中加入一些关闭时间，同样在ODTCNT定时器溢出，电平0到1转换的边沿会加入一段延迟。下图标示加入相同死区时间的PG0/PG1、PG2/PG3, PG4/PG5,及 PG6/PG7信号。每对是否加入死区都可以通过PDTEN[3:0]寄存器来配置。

注： PDTCNT 及 PDTEN 寄存器都有时控保护 (TA)。仅当PWM配置位互补模式，死区功能才会生效。

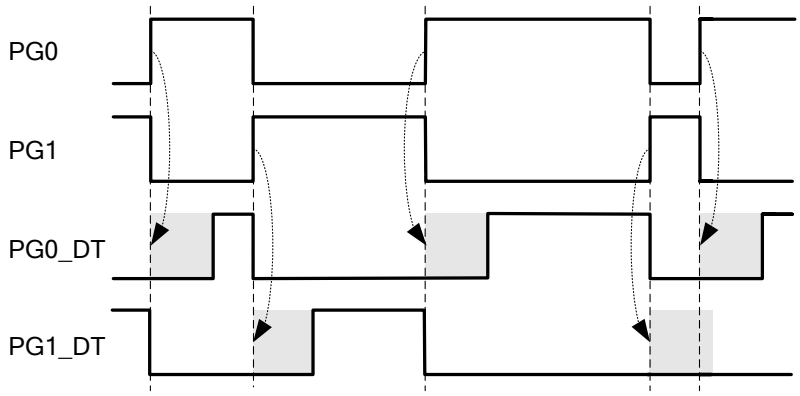


图 18-5. PWM 互补模式死区控制

PDTEN – PWM 死区使能寄存器(TA 保护)

7	6	5	4	3	2	1	0
-	-	-	PDTCNT.8	PDT67EN	PDT45EN	PDT23EN	PDT01EN
-	-	-	读/写	读/写	读/写	读/写	读/写

地址: F9H　　复位值: 0000 0000b

位	名称	描述
4	PDTCNT.8	**PWM 8位死区时间计数器高位** 详见 PDTCNT 寄存器.
3	PDT67EN	**PWM6/7 死区时间使能位** 仅当PWM6/7配置位互补模式，死区功能才会生效。 0 = GP6/GP7信号无延时 1 = 在GP6/GP7信号上升沿加入死区时间延时
2	PDT45EN	**PWM4/5 死区时间使能位** 仅当PWM4/5配置位互补模式，死区功能才会生效。 0 = GP4/GP5信号无延时 1 = 在GP4/GP5信号上升沿加入死区时间延时
1	PDT23EN	**PWM2/3 死区时间使能位** 仅当PWM2/3配置位互补模式，死区功能才会生效。 0 = GP2/GP3信号无延时 1 = 在GP2/GP3信号上升沿加入死区时间延时
0	PDT01EN	**PWM0/1 死区时间使能位** 仅当PWM0/1配置位互补模式，死区功能才会生效。 0 = GP0/GP1信号无延时 1 = 在GP0/GP1信号上升沿加入死区时间延时

PDTCNT – PWM 死区时间计数器(TA 保护)

7	6	5	4	3	2	1	0
PDTCNT[7:0]							
读/写							

地址: FAH　　复位值: 0000 0000b

位	名称	描述
7:0	PDTCNT[7:0]	**PWM 死区时间计数器低字节** 该8位寄存器与PDTEN.4 组成9位PWM死区时间计数器PDTCNT。该计数器仅当PWM设定位互补模式，且有效死区使能位已设置时有效。 PWM 死区时间= $\frac{PDTCNT+1}{F_{SYS}}$. 注在PWM运行过程中，请勿更改PDTCNT的值

18.1.3.3 同步模式

当PWMMOD[1:0] ，PWM设定位同步= [1:0]。在该模式下PG0/2/4/6信号输出与单独输出模式相同。PG1/3/5/7 输出信号与PG02/4/6也完全相同。

18.1.4 输出掩码控制

每路PWM信号都可通过软件方式停止输出。这个功能在电气控制换向器的电机，例如直流无刷电机BLDC应用中非常必要。PMEN寄存器包含6位控制PWM那一路需要掩码的控制位PMD 寄存器定义每路需掩码的电位值。PMEN的默认值为00H，即所有输出都不进行掩码。注：掩码电位按照PMD设定值决定，并不受PNP寄存器影响。

PMEN – PWM 输出掩码控制

7	6	5	4	3	2	1	0
PMEN7	PMEN6	PMEN5	PMEN4	PMEN3	PMEN2	PMEN1	PMEN0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: FBH　　复位值: 0000 0000b

位	名称	描述
n	PMENn	**PWMn 输出掩码使能位** 0 = PWMn 信号输出 1 = PWMn 根据PMDn设定的电位值掩码

PMD – PWM 掩码数据寄存器

7	6	5	4	3	2	1	0
PMD7	PMD6	PMD5	PMD4	PMD3	PMD2	PMD1	PMD0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: FCH　　复位值: 0000 0000b

位	名称	描述
n	PMDn	**PWMn 掩码数据寄存器** PWMn 信号输出掩码电准位值 0 = PWMn 掩码，输出低电平 1 = PWMn 掩码，输出高电平

18.1.5 故障刹车

故障刹车功能长用于提高PWM电路特性，配置为输入故障侦用以保护电机系统防止损坏。当FBINEN (PWMCON1.3)置1，故障刹车输入脚(FB) 生效。当故障发生，PWM相对应的独立管脚相对应的FBD值会被更改，PWMRUN (PWMCON0.7)位自动被硬件清除，PWM输出停止，PWM 12位计数器复位清0，指标位FBF 置1，标示发生了故障刹车中断。即便软件清除FBF值，FBD数据内容仍然保持不变。用户需要重新设定PWMRUN的值以重新启动输出PWM信号。此时，故障刹车状态会被释放，PWM按设定值正常输出。故障刹车具有极性控制位FBINLS (FBD.6) 。注，FB管脚内部有固定的8/F_{SYS}响应过滤结构，从FB脚信号发生到故障刹车响应需要超过8个系统时钟，以避免管脚干扰信号引发误操作。另一个触发故障刹车的方式是ADC结果比较，与FB脚输入效果相同。详见19.1.3 “ADC转换结果比较器 ”.。注， PWM6 及 PWM7 不具有故障刹车数据设定。PWM6 及 PWM7 输出状态保持故障发生时的状态。用户需注意输出状态值是否会影响硬件系统，并通过软件调整。

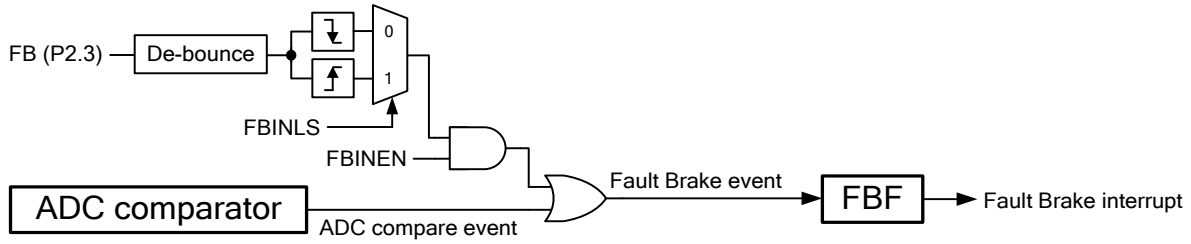


图 18-6. 故障刹车功能模块图

PWMCON1 – PWM控制寄存器 1

7	6	5	4	3	2	1	0
PWMMOD[1:0]		GP	PWMTYP	FBINEN	PWMDIV[2:0]		
读/写		读/写	读/写	读/写	读/写		

地址: DFH 复位值: 0000 0000b

位	名称	描述
3	FBINEN	**FB 管脚输入使能位** 0 = 关闭 1 = 根据FB脚输入，PWM故障刹车功能使能。当符合FBINLS (FBD.6)位设定的信号出现。PWM0~5输出根据FBD寄存器所设定电位值的的刹车信号，PWM6/7 保持刹车发生时的电位值。PWMRUN (PWMCON0.7) 位自动由硬件清除。当PWMRUN信号重新置1，PWM信号重新输出。

FBD – PWM 故障刹车数据

7	6	5	4	3	2	1	0
FBF	FBINLS	FBD5	FBD4	FBD3	FBD2	FBD1	FBD0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: D7H 复位值: 0000 0000b

位	名称	描述
7	FBF	**故障刹车标志位** 当FBINEN设置为1，FB管脚上侦测到符合FBINLS (FBD.6)设定的边沿信号后，该位置1。该位需要通过软件清0。当FBF清0后，故障刹车仍然不会释放PWM输出，需要重新输出，需要设置PWMRUN (PWMCON0.0)为1，重新启动PWM输出。
6	FBINLS	**FB 管脚输入选择** 0 = 下降沿. 1 = 上升沿.
n	FBDn	**PWMn 故障刹车数据** 0 = 当故障发生时PWMn 信号输出为0 1 = 当故障发生时PWMn 信号输出为1

18.1.6 极性控制

每路PWM带有独立的极性控制位PNP0~PNP7。默认正逻辑位高电平有效，即PWM输出高电平，电源切换开，低电平电源切换关。用户可通过设置PNP位来改变PWM输出极性。产生相反的信号。

PNP – PWM 负极性寄存器

7	6	5	4	3	2	1	0
PNP7	PNP6	PNP5	PNP4	PNP3	PNP2	PNP1	PNP0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: D6H 复位值: 0000 0000b

位	名称	描述
n	PNPn	**PWMn 负极性输出使能** 0 = PWMn 输出按照设定直接输出到PWMn管脚. 1 = PWMn 输出按照设定取反输出到PWMn管脚

18.2 PWM 中断

PWM模块带有标志位PWMF (PWMCON0.5) 用以标示当前PWM周期完成状态。响应条件根据INTSEL[1:0] and INTTYP[1:0] (PWMCON0[1:0] 及 [3:2])设定。注：中心点触发或边沿点触发仅适用于中心对齐模式。PWMF由软件清除。

PWMCON0 – PWM 控制寄存器 0 (可位寻址)

7	6	5	4	3	2	1	0
PWMRUN	LOAD	PWMF	CLRPWM	INTTYP1	INTTYP0	INTSEL1	INTSEL0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: D8H 复位值: 0000 0000b

位	名称	描述
3:2	INTTYP[1:0]	**PWM 中断类型选择** 00 = PWM0/2/4/6 脚下降沿. 01 = PWM0/2/4/6 脚上升沿. 10 = 每个PWM周期的中点 11 = 每个PWM周期的终点 注：中心点中断方式或终点中断方式仅适用于PWM中心对齐模式。
1:0	INTSEL[1:0]	**PWM 中断对选择** 该位段用以选择中断响应所相对的PWM脚。 00 = PWM0. 01 = PWM2. 10 = PWM4. 11 = PWM6.

PWM 中断波形如下

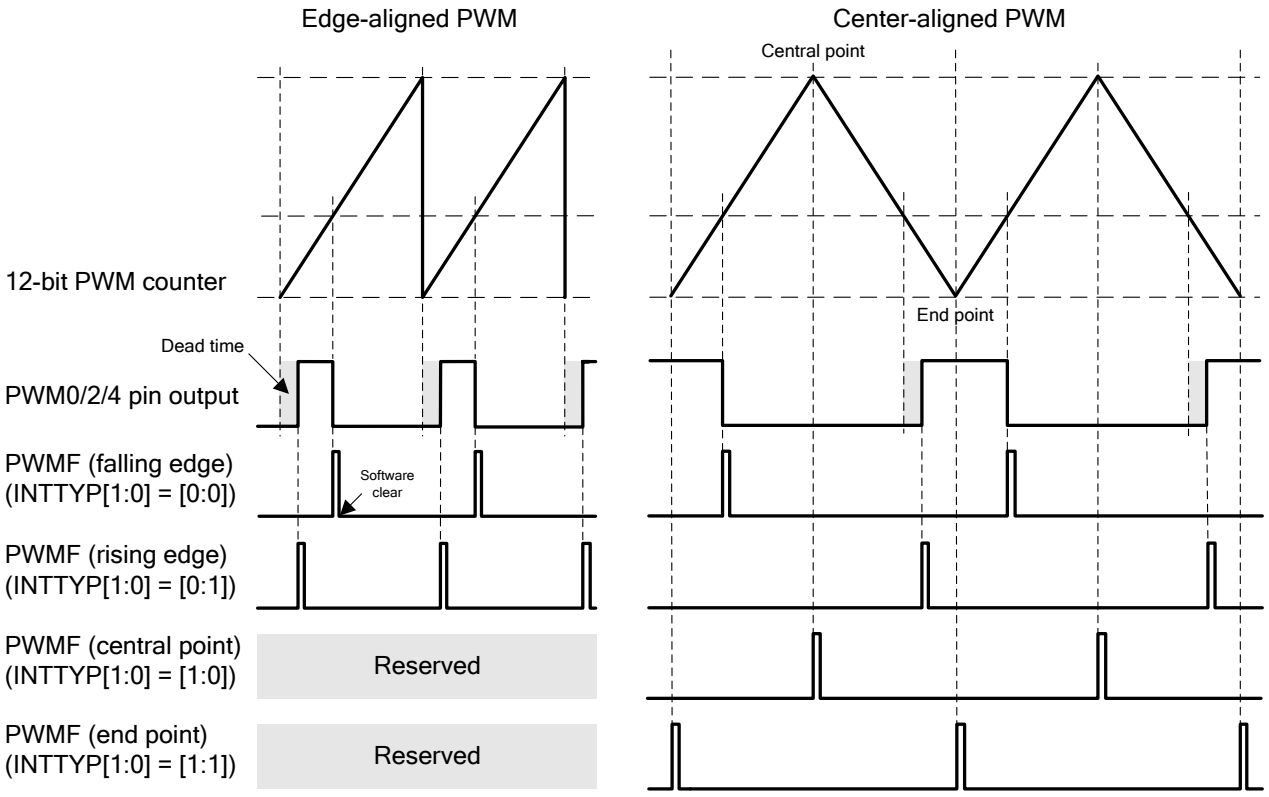


图 18-7. PWM 中断类型

故障刹车中断为另一组中断信号。具有独立的中断向量，不在PWM中断内响应。如果故障刹车中断EFB (EIE.5)使能位使能，无论故障刹车或是ADC比较输入中断，FBF(FBD.7)都会置1。FBF通过软件清0。

19. 10位模数转换- (ADC)

N76E885 内建10位逐次逼近型模数转换模块(SAR ADC)。模数转换模块负责将管脚上的模拟信号转换为10位2进制数据。N76E885 10路输入为单端模式。内部带隙电压(band-gap voltage)为1.22V，同时也可用作ADC输入端，（特别注意，由于设计要求，当使用带隙电压作为ADC输入时，必须先设定BODEN(BODCON0.7)位为1，比较好的方式是在编程器上对CONFIG的CBODEN位写1，这样每次复位后BODEN位就会自动配置为1）。所有模拟电路通过同一组采样电路， 及同一组采样保持电容。该组采样保持电容为转换电路的输入端。然后转换器通过逐次逼近的方式得到有效结果并存放在寄存器中。

19.1 功能描述

19.1.1 ADC 工作方式

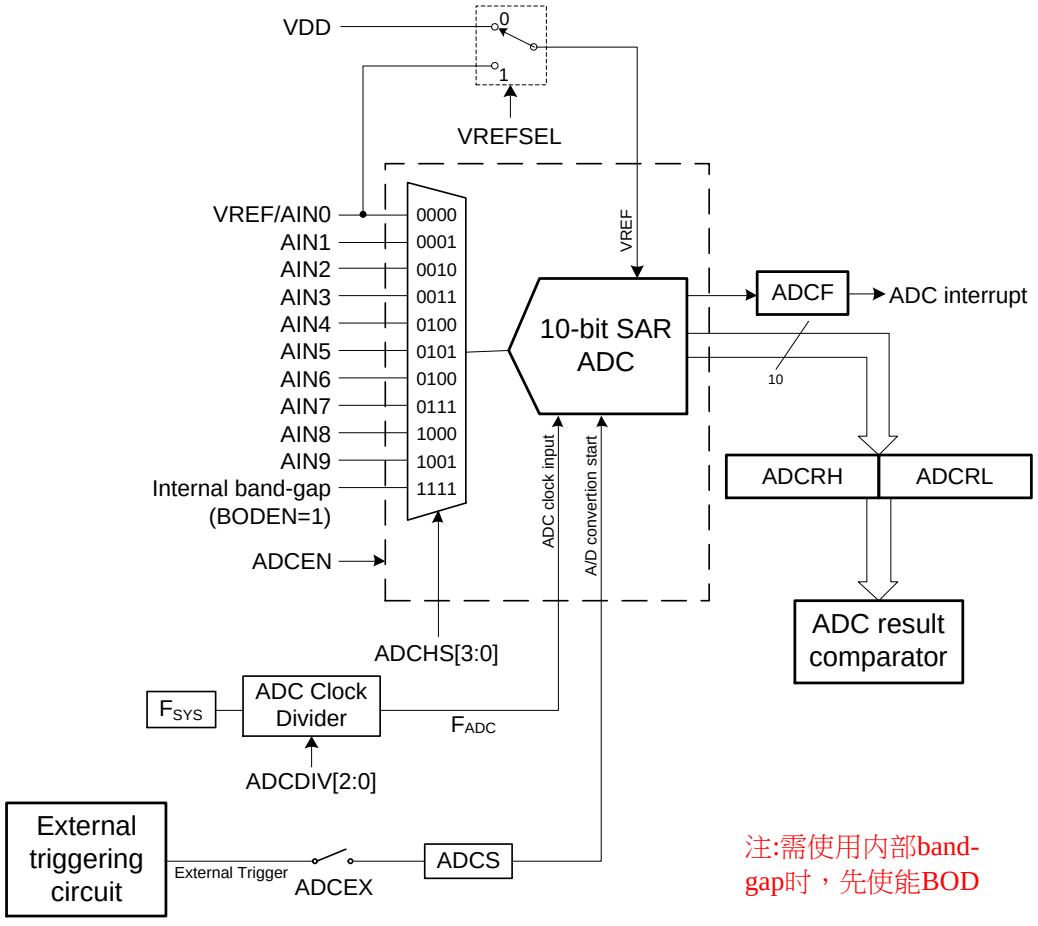


图 19-1. 10位 ADC 模块结构图

由于ADC模块需要额外功耗，在开始AD转换前，ADC模块需要通过ADCEN (ADCCON1.0)使能，从而激活ADC电路，一旦AD转换模块不再使用，建议关闭ACEN以节省芯片整体功耗。参考电压 VREF 输入可以通过VREFSEL (ADCCON1.7) 位设定为V_{DD}或外部参考电压输入脚AIN0/VREF。

ADC转换输入脚需要特别配置，用户可通过ADCHS[3:0] 来选择采样所需要的管脚。同时采样所用管脚需要配置PxMn寄存器将管脚配置为 “输入高阻模式” (input-only high impedance) 。配置后，管脚与数字电路部分将断开，但数字电路部分仍然可以工作，因此数字输入将可能产生漏电流。所以还需要通过配置P0DIDS 及 ADCCON2 寄存器相应位来关闭数字输入缓冲区。如上配置后，ADC输入脚将变成纯模拟输入电路。同样ADC采样时钟也需要认真考虑。ADC最高时钟频率参考表 32–10. 当采样时钟设定超过最大值时，采样结果数据为何不可预测。

开始AD转换的开关为设定ADCS位(ADCCON0.6)为1。当转换完成后，硬件会自动清除该位，同时置1 ADCF (ADCCON0.7)位，如果之前ADC中断已使能，则会进入中断。转换结果存放在ADCRH (高8位)及 ADCRL (低2位)中。10位转换结果值为 $1023\times\frac{V_{AIN}}{V_{REF}}$ 。

ADC 采样时间是可以配置的，通过配置寄存器ADCAQT，采样时间范围为6 (6 + 0) 至 261 (6 + 255) 个ADC时钟周期。当模拟输入脚的输入阻抗无法达到理想时，该功能相当有用。调整采样时间可以克服输入阻抗的影响。

内部及外部数字电路，可能干扰采样结果的准确度。所以如果需要高精准的转换结果，请参考如下设置，以减少噪音干扰。

1. 模拟输入脚尽量离芯片越近越好。避免管脚附近有高速数字电路经过，并离高速数字电路越远越好。

2. 在转换过程中，将芯片进入空闲模式。

3. 如果模拟输入脚AIN在系统中同时需要切换做数字管脚，请确保在转换过程中不要做数字/模拟切换动作。

19.1.2 外部触发ADC

除了通过软件启动外， N76E885 提供硬件触发方式启动AD转换。一旦ADCEX (ADCCON1.1) 置1，PWM边沿或周期，STADC的边沿将会自动触发AD转换启动信号 (由硬件设定ADCS信号) 。触发信号由ETGSEL (ADCCON0[5:4]) 及 ETGTYP (ADCCON1[3:2]) 配置。同时还可以设置外部信号至启动AD之间的延时计数。该功能将非常适用于高精度电机控制 。注意，在AD模块转换过程中(ADCS = 1)，任何软件或硬件触发信号都是无效的。

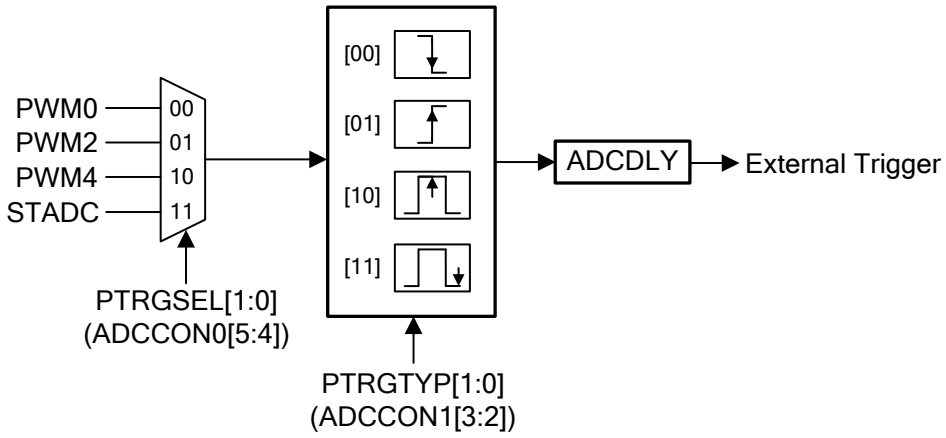


图 19-2. 外部触发ADC结构

19.1.3 ADC转换结果比较器

N76E885 ADC 提供一组数字比较器，用于比较AD 10位转换结果与预先填入寄存器ACMPH 及 ACMPL 的内容是否一致。ADC 比较器使能位为ADCMPEN (ADCCON2.5) 一旦设定，每次AD转换结束都会进行比较。ADCMPO (ADCCON2.4) 显示根据ADCMPOP (ADCCON2.6) 设定的比较结果. 当ADFBEN (ADCCON2.7)设置后，ADC比较结果可触发PWM故障刹车。详见 章节 18.1.5 “故障刹车”.

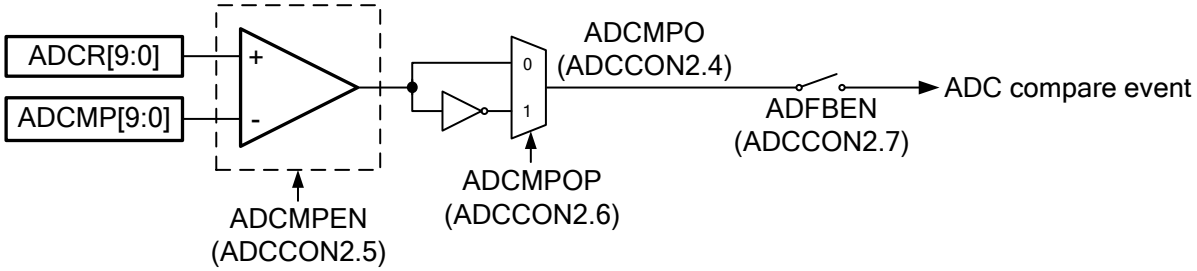


图 19-3. ADC转换结果比较

19.2 ADC控制寄存器

ADCCON0 – ADC控制寄存器0 (可位寻址)

7	6	5	4	3	2	1	0
ADCF	ADCS	ETGSEL1	ETGSEL0	ADCHS3	ADCHS2	ADCHS1	ADCHS0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: E8H　　复位值: 0000 0000b

位	名称	描述
7	ADCF	**ADC标志位** 当AD转换完成，该位置1。可读取到当前AD转换结果。该位为1时无法开始新一轮转换，需要软件清零。
6	ADCS	**A/D 转换软件启动位** 该位置1启动AD转换。在AD转换过程中该位保持为1，当转换结束硬件自动清0。这意味着写入ADCS的值和读出的不一定相符 写: 0 = 无动作. 1 = 开始AD转换 读: 0 = ADC 模块空闲状态 1 = ADC 模块工作中
5:4	ETGSEL[1:0]	**外部触发源选择** 当 ADCEX (ADCCON1.1) 为1，该位选择外部触发ADC的来源 00 = PWM0. 01 = PWM2. 10 = PWM4. 11 = STADC 脚.
3:0	ADCHS[3:0]	**A/D转换通道选择** 该位用于选择ADC转换通道。当ADCEN 为 0所有输入无效. 0000 = AIN0. 0001 = AIN1. 0010 = AIN2. 0011 = AIN3. 0100 = AIN4. 0101 = AIN5. 0110 = AIN6. 0111 = AIN7. 1000 = AIN8. 1001 = AIN9. 1111 = 内部带隙电压(band-gap)1.22V.（需BODEN=1，band-gap才会有效） Others = 保留.

ADCCON1 – ADC控制寄存器1

7	6	5	4	3	2	1	0
VREFSEL	ADCDIV[2:0]			ETGTYP[1:0]		ADCEX	ADCEN
读/写	读/写			读/写		读/写	读/写

地址: E1H 复位值: 0010 0000b

位	名称	描述
7	VREFSEL	**VREF源选择** 0 = 芯片 VDD. 1 = 外部 AIN0/VREF 引脚.
6:4	ADCDIV[2:0]	**ADC 时钟除频** 000 = F_{ADC}等于F_{SYS}/1. 001 = F_{ADC} 等于F_{SYS}/2. 010 = F_{ADC} 等于F_{SYS}/4. (默认值) 011 = F_{ADC} 等于F_{SYS}/8. 100 = F_{ADC} 等于F_{SYS}/16. 101 = F_{ADC} 等于F_{SYS}/32. 110 = F_{ADC} 等于F_{SYS}/64. 111 = F_{ADC} 等于F_{SYS}/128.
3:2	ETGTYP[1:0]	**外部触发信号类型选择** 当 ADCEX (ADCCON1.1) 置1，该位决定响应外部触发的类型。 00 = PWM0/2/4 或 STADC 脚的下降沿. 01 = PWM0/2/4 或 STADC 脚的上升沿. 10 = 一个PWM周期的中点. 11 = 一个PWM周期的终点. 注PWM周期中点或终点触发仅适用于中心对齐模式的PWM输出。
1	ADCEX	**ADC 触发启动信号选择位** 该位决定启动ADC的触发条件 0 = 当软件设定ADCS位，启动AD转换 1 = 当软件设定ADCS位后，还需要外部触发信号启动。外部触发信号条件由寄存器ETGSEL[1:0] 及 ETGTYP[1:0]决定。注，当ADCS为1时（正在转换），外部触发信号不会影响ADC直到ADC转换结束ADCS被硬件清0。
0	ADCEN	**ADC 使能位** 0 = ADC 转换电路关闭 1 = ADC 转换电路打开

ADCCON2 – ADC控制寄存器 2

7	6	5	4	3	2	1	0
ADFBEN	ADCMPOP	ADCMPEN	ADCMPO	P26DIDS	P20DIDS	-	ADCDLY.8
读/写	读/写	读/写	只读	读/写	读/写	-	读/写

地址: E2H 复位值: 0000 0000b

位	名称	描述
7	ADFBEN	**ADC比较结果响应故障刹使能寄存器** 0 = 关闭 1 = ADC 触发故障刹车功能打开 当比较结果 ADCMPO 为1触发故障刹车模块。即符合PWM故障刹车输出值后，硬件将清除 PWMRUN (PWMCON0.7)，并终止PWM输出。当PWMRUN置1，PWM重新输出。
6	ADCMPOP	**ADC比较器输出极性选择位** 0 = 若ADCR[9:0]大于或等于ADCMP[9:0].，ADCMPO 为 1 1 = 若ADCR[9:0]小于ADCMP[9:0]，ADCMPO 为 1
5	ADCMPEN	**ADC 结果比较使能位** 0 = ADC 结果比较功能关闭. 1 = ADC r结果比较功能打开.
4	ADCMPO	**ADC比较结果输出位** 该位输出ACMPOP设定比较输出的结果。每次AD转换结束都会更新输出。
3	P26DIDS	**P2.6 脚数字输入关闭** 0 = 数字输入功能打开 1 = 数字输入功能关闭。 P2.6读取始终为 0。
2	P20DIDS	**P2.0脚数字输入关闭** 0 = 数字输入功能打开 1 = 数字输入功能关闭。 P2.0读取始终为 0
0	ADCDLY.8	**ADC 外部触发延时计数器数值 第8位** 详见 ADCDLY 寄存器描述

ADCAQT – ADC采样时间

7	6	5	4	3	2	1	0
ADCAQT[7:0]							
读/写							

地址: F2H 复位值: 0000 0000b

位	名称	描述
7:0	ADCAQT[7:0]	**ADC 采样时间** 该8位寄存器决定ADC采样时间，算式如下： ADC 采样时间 = $\frac{6+ADCAQT}{F_{ADC}}$. 默认最小采样时间为 6个 ADC 时钟周期。注，在ADC转换过程中，不得更改该寄存器的值

P0DIDS – P0 数字输入功能关闭寄存器

7	6	5	4	3	2	1	0
P07DIDS	P06DIDS	P05DIDS	P04DIDS	P03DIDS	P02DIDS	P01DIDS	P00DIDS
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: F6H 复位值: 0000 0000b

位	名称	描述
n	P0nDIDS	**P0.n 数字输入关闭** 0 = P0.n 数字输入功能打开 1 = P0.n 数字输入功能关闭. P0.n读取始终为 0.

ADCDLY – ADC外部触发延迟计数器

7	6	5	4	3	2	1	0
ADCDLY[7:0]							
读/写							

地址: E3H 复位值: 0000 0000b

位	名称	描述
7:0	ADCDLY[7:0]	**ADC 外部触发启动延迟计数器低位** 该8位寄存器与ADCCON2.0组成 9位计数器，用于在外部触发启动ADC之前加入一段延迟。延迟计数结束在开始ADC转换 外部延迟时间 = $\frac{ADCDLY}{F_{ADC}}$. 注，该延迟仅当 ADCEX (ADCCON1.1) 置1时有效。如果启用PWM输出触发ADC功能，在PWM运行过程中不得更改ADCDLY计数值。

ADCRH – ADC转换结果高位寄存器

7	6	5	4	3	2	1	0
ADCR[9:2]							
R							

地址: C3H 复位值: 0000 0000b

位	名称	描述
7:0	ADCR[9:2]	**ADC转换结果高位** ADC转换结果高8位

ADCRL – ADC转换结果低位寄存器

7	6	5	4	3	2	1	0
-	-	-	-	-	-	ADCR[1:0]	
-	-	-	-	-	-	R	

地址: C2H 复位值: 0000 0000b

位	名称	描述
1:0	ADCR[1:0]	**ADC转换结果低位** ADC转换结果低2位

ADCMPH –ADC比较值高位寄存器

7	6	5	4	3	2	1	0
ADCMP[9:2]							
W/R							

地址: CFH 复位值: 0000 0000b

位	名称	描述
7:0	ADCMP[9:2]	**ADC比较值高字节** ADC比较值高字节8位内容

ADCMPL – ADC比较值低位寄存器

7	6	5	4	3	2	1	0
-	-	-	-	-	-	ADCMP[1:0]	
-	-	-	-	-	-	W/R	

地址: CEH 复位值: 0000 0000b

位	名称	描述
1:0	ADCMP[1:0]	**ADC 比较值低位** ADC比较值低2位

20. 时控保护 (TA)

N76E885有几个特殊功能，如看门狗定时器，欠压功能检测等。这些功能是系统正常运行的关键。如果没有对这些寄存器进行写保护，无关代码可能对其写入不确定的值，结果导致不正确的操作和控制失常。为了防止这种风险，N76E885的时控保护功能，可以限制对关键的SFR的写访问。此保障方式，是用一个定时器控制的访问。以下寄存器是相关的时控保护(TA)。

TA – 时控保护寄存器

7	6	5	4	3	2	1	0
TA[7:0]							
W							

地址: C7H　　复位值: 0000 0000b

位	名称	描述
7:0	TA[7:0]	**时控保护** TA寄存器控制着对被保护的SFRs的访问权限。当需要写特殊规定的寄存器时，必须先对TA寄存器写入AAH，接着是55H，当写完这两条后，才可以有4个时钟周期的时间对具有时控保护的寄存器写入数据。

对被保护的位的访问是受时间限制的。要对他进行写操作，那么时控窗口必须打开，否则写操作无效。当对TA写入AAH时，计数器计数3个时钟周期等待对TA写入55H。如果在写完AAH后的3个时钟周期内再写入55H则时控访问窗口被打开。该窗口保持4个时钟周期，4个时钟周期过后窗口自动关闭。一旦时控窗口关闭，那么要重复上述过程来访问被保护的位。注意受TA保护的SFRs只是写需要时控保护，读不受保护。用户读这些受TA保护的寄存器是不需要对TA写入AAH和55H的。下面列出对时控寄存器进行访问的推荐代码。

```
(CLR    EA)                 ;if any interrupt is enabled, disable temporally
 MOV    TA,#0AAH
 MOV    TA,#55H
(Instruction that writes a TA protected register)
(SETB   EA)                 ;resume interrupts enabled
```

在执行上述指令过程中，所有中断必须关闭，避免随中断产生的延时影响三条指令的有效时间。若没有打开任何中断，则 CLR EA 和 SETB EA 指令则可省略。.

以下为是写时控保护寄存器的正确和错误范例：

范例 1,

```
MOV     TA,#0AAH            ;3 clock cycles
MOV     TA,#55H             ;3 clock cycles
ORL     WDCON,#data         ;4 clock cycles
```

范例2,

```
MOV     TA,#0AAH            ;3 clock cycles
MOV     TA,#55H             ;3 clock cycles
NOP                         ;1 clock cycle
ANL     BODCON0,#data       ;4 clock cycles
```

范例3,

```
MOV     TA,#0AAH            ;3 clock cycles
MOV     TA,#55H             ;3 clock cycles
MOV     WDCON,#data1        ;3 clock cycles
ORL     BODCON0,#data2      ;4 clock cycles
```

范例4,

```
MOV     TA,#0AAH            ;3 clock cycles
NOP                         ;1 clock cycle
MOV     TA,#55H             ;3 clock cycles
ANL     BODCON0,#data       ;4 clock cycles
```

在第一个例子中，写保护位在三个时钟周期窗口关闭之前完成。然而，在例2中，BODCON0的写入并没有在时控保护打开时完成，操作完这些指令后，BODCON0的值不会有变化。示例3中，WDCON写入成功，但对BODCON0访问超过三个机器周期窗口，因此BODCON0值不会改变。例4，第二次写55H对应第一个AAH写入时间超过了3个机器周期，时控保护打开失败，所以后面的写入全部无效。

21. 中断系统

21.1 中断概述

中断的目的是使软件处理非常规或异步事件。N76E885有4个中断优先级和18个中断源。每个中断源都有独立的优先级位、标志、中断向量和使能位。另外，中断可被全局使能或关闭。当一个中断发生时，CPU将服务中断。这个服务被指定为一个中断服务例程(ISR)。如表 21–1. 中断向量.所示，ISR被分配到预先指定的地址中。当中断发生时且中断使能, CPU 将跳转到相应的中断向量地址. 执行此地址处的程序, 并停留在中断服务状态直到执行完ISR. 一旦ISR 开始执行, 仅能被更高优先级的中断抢占。 ISR通过指令RETI中止，该指令强迫CPU回到中断发生前所执行指令的下一条指令。

表 21–1. 中断向量

中断源	向量地址	向量号	中断源	向量地址	向量号
复位	0000H	-	SPI 中断	004BH	9
外部中断 0	0003H	0	WDT 中断	0053H	10
定时器 0 溢出	000BH	1	ADC 中断	005BH	11
外部中断 1	0013H	2	定时器输入捕获中断	0063H	12
定时器1 溢出	001BH	3	PWM 中断	006BH	13
串口 0 中断	0023H	4	PWM故障刹车中断	0073H	14
定时器 2A 下溢出/匹配 0	002BH	5	串口1中断	007BH	15
I^2C 状态/超时 中断	0033H	6	定时器3溢出中断	0083H	16
引脚中断	003BH	7	WKT自唤醒定时器中断	008BH	17
欠压检测中断	0043H	8			

21.2 中断使能

每一个中断源都可以通过各自的中断使能位开启或是关闭，这些位在IE和EIE特殊功能寄存器SFRs中。有一个全局中断EA(IE.7)位，清0该位将关闭所有中断，不管各个独自的中断是否使能。当EA为0时有中断请求，该中断会被挂起直到EA恢复为1才去执行该中断。所有中断标志位可以用软件置位故也可以用软件启动中断。

每一个中断产生时对应中断标志位都会被置1，不管是通过硬件还是软件。用户在中断服务程序里应该小心处理中断标志位，大多数中断标志位都是写0清除这样可以避免递归中断请求。

IE – 中断使能寄存器 (可位寻址)

7	6	5	4	3	2	1	0
EA	EADC	EBOD	ES	ET1	EX1	ET0	EX0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: A8H 复位值: 0000 0000b

位	名称	描述
7	EA	**使能所有中断** 该位全局使能/禁止所有中断. 可以禁止所有单个中断的设置. 0 = 禁止所有中断源. 1 = 使能每个中断, 如果单个中断使能, 将会产生中断.
6	EADC	**使能ADC中断** 0 = 禁用ADC中断 1 = 使能由ADCF (ADCCON0.7) 产生中断
5	EBOD	**使能BOD中断** 0 = 禁用BOD中断 1 =使能由BOF (BODCON0.3)产生中断
4	ES	**使能串口0中断** 0 = 禁用串口0中断 1 =使能由TI (SCON.1) 或 RI (SCON.0)产生中断
3	ET1	**使能定时器1中断** 0 = 禁用定时器1中断 1 =使能由TF1 (TCON.7)产生中断
2	EX1	**使能外部中断1** 0 = 禁用外部中断1 1 =使能由$\overline{\text{INT1}}$产生中断
1	ET0	**使能定时器0中断** 0 = 禁用定时器1中断 1 =使能由TF0 (TCON.5)产生中断
0	EX0	**使能外部中断0** 0 = 禁用外部中断0 1 =使能由$\overline{\text{INT0}}$产生中断

EIE – 扩展中断使能寄存器

7	6	5	4	3	2	1	0
ET2	ESPI	EFB	EWDT	EPWM	ECAP	EPI	EI2C
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: 9BH 复位值: 0000 0000b

位	名称	描述
7	ET2	**定时器2中断使能位** 0 = 禁用定时器2中断 1 = 当TF2 (T2CON.7)为1 产生中断.

位	名称	描述
6	ESPI	**SPI 中断使能位** 0 = 禁用SPI 中断 1 = 使能由SPIF (SPSR.7), SPIOVF (SPSR.5), 或 MODF (SPSR.4) 产生中断
5	EFB	**PWM故障刹车中断使能位** 0 = 禁用PWM故障刹车中断 1 =当FBF (FBD.7) 为1产生中断.
4	EWDT	**使能WDT中断** 0 = 禁用WDT中断 1 =使能由WDTF (WDCON.5)产生中断
3	EPWM	**PWM中断使能位** 0 = 禁用定时器2B中断 1 =使能由TF2B (T2CON0.5)产生中断
2	ECAP	**定时器输入捕获中断** 0 = 禁用定时器2A中断 1 =使能由TF2A (T2CON0.4)产生中断
1	EPI	**外部引脚中断使能位** 0 = 禁用引脚中断 1 =使能由PIF寄存器任意一个标志位产生中断
0	EI2C	**I^2C中断使能位** 0 = 禁用I^2C中断 1 =使能由SI (I2CON.3) 或 I2TOF (I2TOC.0)产生中断

EIE1 – 扩展中断使能寄存器1

7	6	5	4	3	2	1	0
-	-	-	-	-	EWKT	ET3	ES_1
-	-	-	-	-	读/写	读/写	读/写

地址: 9CH 复位值: 0000 0000b

位	名称	描述
2	EWKT	**WKT中断使能位** 0 = 禁用WKT中断 1 = 当WKTF (WKCON.4)为1产生中断
1	ET3	**定时器3中断使能位** 0 = 禁用定时器3中断 1 = 当TF3 (T3CON.4)为1产生中断
0	ES_1	**串口1中断使能位** 0 = 禁用串口1中断 1 = 当 TI_1 (SCON_1.1) 或 RI_1 (SCON_1.0)为1产生中断

21.3 中断优先级

对中断来说，系统为其提供4种优先级：最高（3级）、高（2级）、低（1级）、最低（0级）。中断源可以单独设置各自的优先级位来配置其优先级。表 21–2 列举了4种优先级配置。相对来说，低优先级中

断可以被高优先级中断打断，但是不能被同等优先级或是更低的优先级打断。默认优先级可以帮助中断控制器解决同等优先级同时请求中断的状况。

在多个中断时，遵循以下规则：

1.当一个低优先级中断正在运行，这时一个高优先级产生，该中断会被打断去执行高优先级中断。当高优先级中断执行完RETI后，低优先级中断恢复继续运行。当低优先级中断执行完RETI后，控制器把运行权利交给主程序。

2.如果一个高优先级中断正在运行,不能被任何其他中断源——即使这是一个高优先级中断在默认优先级中比正在运行的中断更高。

3.低优先级中断只有在其他中断没有执行的情况下才能被调用。然后同时，低优先级中断不能被另一个低优先级中断打断，即使这个低优先级中断在默认优先级中比正在运行的中断更高。

4.如果两个中断同时发生,优先级高的中断先执行。如果两个中断优先级相同，默认优先级高的中断先执行，这是符合默认优先级唯一的条件。

默认优先级如表 21–3所示。同时总结了中断源、标志位、向量地址、使能位、优先级位和允许CPU可以从芯片从掉电模式中唤醒。更多的将CPU从掉电模式中唤醒细节见 章节 23.2 “掉电模式”.

表 21–2.中断优先级设定

中断优先级控制位		中断优先级
IPH / EIPH / EIPH1	IP / EIP / EIP2	
0	0	等级 0 (最低)
0	1	等级 1
1	0	等级2
1	1	等级3 (最高)

表 21–3. 各级中断源特性表

中断源	向量地址	中断标志	使能位	默认优先级	优先级控制位	是否支持掉电模式唤醒
复位	0000H	-	始终打开	最高	-	支持
外部中断0	0003H	IE0[1]	EX0	1	PX0, PX0H	支持
欠压检测	0043H	BOF (BODCON0.3)	EBOD	2	PBOD, PBODH	支持
看门狗定时器	0053H	WDTF (WDCON.5)	EWDT	3	PWDT, PWDTH	支持
定时器 0	000BH	TF0[2]	ET0	4	PT0, PT0H	否

中断源	向量地址	中断标志	使能位	默认优先级	优先级控制位	是否支持掉电模式唤醒
I^2C 中断	0033h	SI + I2TOF (I2TOC.0)	EI2C	5	PI2C, PI2CH	否
ADC	005Bh	ADCF	EADC	6	PADC, PADCH	否
外部中断1	0013H	IE1[1]	EX1	7	PX1, PX1H	支持
外部管脚中断	003BH	PIF0 to PIF7 (PIF)[3]	EPI	8	PPI, PPIH	支持
定时器 1	001BH	TF1[2]	ET1	9	PT1, PT1H	否
串口 0	0023H	RI + TI	ES	10	PS, PSH	否
故障刹车	0073h	FBF (FBD.7)	EFB	11	PFB, PFBH	否
SPI	004Bh	SPIF (SPSR.7) + MODF (SPSR.4) + SPIOVF (SPSR.5)	ESPI	12	PSPI, PSPIH	否
定时器 2	002BH	TF2[2]	ET2	13	PT2, PT2H	否
定时器输入捕获	0063H	CAPF[2:0] (CAPCON0[2:0])	ECAP	14	PCAP, PCAPH	否
PWM	006BH	PWMF	EPWM	15	PPWM, PPWMH	否
串口 1	007BH	RI_1 + TI_1	ES_1	16	PS_1, PSH_1	否
定时器 3	0083H	TF3[2] (T3CON.4)	ET3	17	PT3, PT3H	否
自唤醒定时器	008BH	WKTF (WKCON.4)	EWKT	18	PWKT, PWKTH	支持

[1] 当外部中断引脚设置成边沿触发(ITx = 1)，在执行中断服务程序时中断标志位IEx 会被自动清除。当被设置成电平触发时(ITx = 0)，IEx会根随各自引脚反向电平状态变化一致，不能通过软件控制。

[2] 在执行中断服务程序时中断标志位TF0, TF1, 或TF3 会被自动清除。在定时器 2x工作在自动加载模式下，执行中断服务程序时TF2x也会被硬件自动清除

[3] 当引脚中断选择了电平触发，PIFn标志位反应各自通道的状态，软件无法控制

IP –中断优先级寄存器 (位寻址)[1]

7	6	5	4	3	2	1	0
-	PADC	PBOD	PS	PT1	PX1	PT0	PX0
-	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: B8H　　复位值: 0000 0000b

位	名称	描述
6	PADC	**ADC中断优先级低位**
5	PBOD	**BOD检测中断优先级低位**
4	PS	**串口0中断优先级低位**
3	PT1	**定时器1中断优先级低位**
2	PX1	**外部中断1中断优先级低位**
1	PT0	**定时器0中断优先级低位**
0	PX0	**外部中断0中断优先级低位**

[1] IP使用时结合IPH一起决定每个中断源的优先级。详见表 21–2.中断优先级设定

IPH – 中断优先级高位寄存器[2]

7	6	5	4	3	2	1	0
-	PADCH	PBODH	PSH	PT1H	PX1H	PT0H	PX0H
-	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: B7H 复位值: 0000 0000b

位	名称	描述
6	PADC	**ADC中断优先级高位**
5	PBOD	**BOD检测中断优先级高位**
4	PSH	**串口0中断优先级高位**
3	PT1H	**定时器1中断优先级高位**
2	PX1H	**外部中断1中断优先级高位**
1	PT0H	**定时器0中断优先级高位**
0	PX0H	**外部中断0中断优先级高位**

[2] IPH使用时结合IP一起决定每个中断源的优先级。详见表 21–2.中断优先级设定

EIP –扩展中断优先级寄存器[3]

7	6	5	4	3	2	1	0
PT2	PSPI	PFB	PWDT	PPWM	PCAP	PPI	PI2C
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: EFH 复位值: 0000 0000b

位	名称	描述
7	PT2	**定时器2中断优先级低位**
6	PSPI	**SPI中断优先级低位**
5	PFB	**故障刹车中断优先级低位**
4	PWDT	**WDT中断优先级低位**
3	PPWM	**PWM中断优先级低位**
2	PCAP	**定时器输入捕获中断优先级低位**
1	PPI	**引脚中断优先级低位**
0	PI2C	**I^2C中断优先级低位**

[3] EIP使用时结合EIPH一起决定每个中断源的优先级的。详见 表 21–2.中断优先级设定 确认中断优先级.

EIPH –扩展中断优先级高位寄存器[4]

7	6	5	4	3	2	1	0
PT2H	PSPIH	PFBH	PWDTH	PPWMH	PCAPH	PPIH	PI2CH
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: F7H 复位值: 0000 0000b

位	名称	描述
7	PT2H	**定时器2中断优先级高位**
6	PSPIH	**SPI中断优先级高位**
5	PFBH	**故障刹车中断优先级高位**
4	PWDTH	**WDT中断优先级高位**
3	PPWMH	**PWM中断优先级高位**
2	PCAPH	**定时器输入捕获中断优先级高位**
1	PPIH	**引脚中断优先级高位**
0	PI2CH	**I^2C中断优先级高位**

[4]] EIPH使用时结合EIP一起决定每个中断源的优先级的。详见表 21–2.中断优先级设定 确认中断优先级
.

EIP1 –扩展中断优先级寄存器1[5]

7	6	5	4	3	2	1	0
-	-	-	-	-	PWKT	PT3	PS_1
-	-	-	-	-	读/写	读/写	读/写

地址: FEH, 页: 0 复位值: 0000 0000b

位	名称	描述
2	PWKT	**WKT 中断优先级低位**
1	PT3	**定时器3 中断优先级低位**
0	PS_1	**串口1 中断优先级低位**

[5] EIP1使用时结合EIPH1一起决定每个中断源的优先级的。详见 表 21–2.中断优先级设定 确认中断优先级

EIPH1 –扩展中断优先级高位寄存器1 [6]

7	6	5	4	3	2	1	0
-	-	-	-	-	PWKTH	PT3H	PSH_1
-	-	-	-	-	读/写	读/写	读/写

地址: FFH, 页: 0 复位值: 0000 0000b

位	名称	描述
2	PWKTH	**WKT 中断优先级高位**
1	PT3H	**定时器3 中断优先级高位**
0	PSH_1	**串口1 中断优先级高位**

[6] EIPH1使用时结合EIP1一起决定每个中断源的优先级的。详见 表 21–2.中断优先级设定 确认中断优先级

21.4 中断服务

中断标志位和优先级在每个系统时钟周期都会被采样。在同一个周期内，被采集到的中断都会被存储及分级。如果满足特定的条件硬件将执行内部产生的LCALL指令，目标地址是中断向量地址。能产生LCALL条件如下：

1. 当前没有执行同等或是更高优先级中断

2.当前查询中断标志周期正好是当前执行指令的最后一个周期

3. 当前指令不能包含写任何中断使能位或是优先级设定位和也不能是RETI

如果以上任何一个条件不满足，就不能产生LCALL。在每一个指令周期都会重新检测中断标志。当某个中断标志被置起但没有满足上述条件故不会被响应，即便随后阻碍响应的条件已去除但仍然未及时响应中断，这个中断标志仍然会在下一个指令周期被去除。

处理器响应一个有效的中断是通过执行一个LCALL 指令将程序转移到中断入口地址。引起中断的中断标志依据不同的中断源在执行中断时可能被硬件清除也有可能不被清除。硬件LCALL与软件LCALL指令相同，该指令保存程序计数器内容到堆栈，但是不保存程序状态字PSW。当中断发生时PC被装入中断向量地址，这样可以产生LCALL。从向量地址继续执行直到执行RETI指令。在执行RETI指令时，处理器弹出堆栈，将栈顶内容加载到PC。用户必须注意堆栈是状态，如果堆栈的内容被修改，处理器不会被通知，将会从堆栈加载的地址继续执行。注RET指令与RETI指令表现相同，但它不会通知中断控制器中断服务已经完成，致使控制器认为中断服务仍在进行。

21.5 中断延迟

每一个中断源的响应时间取决于几个方面，如中断自身特点和指令的执行。在每个时钟周期每一个中断标志和优先级都会被检测。如果有一个中断请求满足以上3个条件，硬件将自动产生LCALL指令，执行该指令需要4个机器周期。这样从中断标志置位到执行中断服务程序最少只需要5个机器周期。

如果三个条件有一个不满足，很长的响应时间是可以预知的。如果高优先级和同等优先级中断正在执行，那么中断延迟时间很明显取决于正在执行的中断服务程序。如果查询周期不是指令执行的最后一个周期，那么需要等待指令执行完毕。最大的响应时间（如果没有其他中断正在执行或是也没有更高优先级中断产生）出现在正在执行RETI指令，然后执行一个最长6个时钟周期的指令作为下一个指令。从一个中断源被激活(没有发现),最长的反应时间是16个时钟周期。这些周期包括完成RETI指令的5个时钟周

期、完成最长指令的6个时钟周期、侦测中断1个时钟周期和完成硬件LCALL跳转到中断地址的4个时钟周期。

因此一个简单的中断系统响应时间总是大于5个时钟周期并且不超过16个时钟周期

21.6 外部中断

外部中断$\overline{INT0}$和$\overline{INT1}$被作为中断源。它们可以依据IT0 (TCON.0) 和 IT1 (TCON.2)选择是边沿触发还是电平触发中断。检测IE0 (TCON.1) 和 IE1 (TCON.3)用以产生中断。在边沿触发模式下，每个系统时钟周期都会采样$\overline{INT0}$ 和 $\overline{INT1}$输入状态。如果在一个周期中采样是高然后在下一个周期中为低，那么这个高到低的转换将会被检测到并且置位中断请求标志IE0或是IE1。由于每个系统时钟周期都去采样外部中断，所以高电平或是低电平至少保持一个系统时钟周期。当中断服务程序被执行时，IE0和IE1会被硬件自动清除。如果选择电平触发模式，那么必须保持引脚为低直到进入中断服务，在进入中断服务程序时IE0和IE1不会被硬件清除。在电平触发模式下，IE0和IE1状态与$\overline{INT0}$ 和 $\overline{INT1}$引脚电平相反。当中断服务程序结束后引脚依然保持低电平，处理器会响应另一个来自同一中断源的中断。$\overline{INT0}$ 和 $\overline{INT1}$均支持将芯片从掉电模式唤醒。

TCON – 定时器 0及 1控制寄存器 (可位寻址)

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
读/写	读/写	读/写	读/写	只读 (电平) 读/写 (边沿)	读/写	只读 (电平) 读/写 (边沿)	读/写

地址: 88H　　复位值: 0000 0000b

位	名称	描述
3	IE1	**外部中断1边沿标志** 当检测到边沿/电平类型时，该标志由硬件置位. 如果 IT1 = 1(下降沿触发), 该位将保持置1直到软件清零或在外部中断1服务程序中硬件清零。 如果 IT1 = 0(低电平触发), 该标志是$\overline{INT1}$输入信号逻辑电平的反转。软件不可控制
2	IT1	**外部中断1类型选择** 该位选择$\overline{INT1}$的中断触发类型是下降沿还是低电平. 0 = $\overline{INT1}$ 为低电平触发 1 = $\overline{INT1}$ 为下降沿触发
1	IE0	**外部中断0边沿标志** 当检测到边沿/电平类型时，该标志由硬件置位. 如果 IT0 = 1(下降沿触发), 该位将保持置1直到软件清零或在外部中断0服务程序中硬件清零。 如果 IT0 = 0(低电平触发), 该标志是$\overline{INT0}$输入信号逻辑电平的反转。软件不可控制

位	名称	描述
0	IT0	**外部中断0类型选择** 该位选择$\overline{INT0}$的中断触发类型是下降沿还是低电平. 0 = $\overline{INT0}$ 为低电平触发 1 = $\overline{INT0}$ 为下降沿触发

22. 在应用编程 (IAP)

修改FLASH数据通常需要很长时间，不像RAM那样可以实时操作。而且擦除、编程或是读取FLASH数据需要遵循相当复杂的时序步骤。N76E885提供方便FALSH编程方式可以帮助用户通过IAP方式重编程FLASH内容。IAP就是通过软件实现在电路进行电擦除和编程的方法。

通过设置IAPEN（CHPCON.0受TA保护）使能IAP并且置位IAPUEN相应位打开需要升级的目标区域后，用户可以将16位目标地址写入IAPAH和IAPAL，数据写入IAPFD，命令写入IAPCN，然后通过设置触发位IAPGO(IAPTRG.0)开始执行IAP（IAPTRG也受TA保护）。此时，CPU保持程序计数器，内建IAP自动控制内部充电泵提高电压，并控制信号时序。擦除和编程时间是内部控制的与工作电压和频率无关。通常页擦除时间是5 ms，字节编程时间是23.5μs。IAP动作完成后，程序计数器继续运行之后的指令。IAPGO位将自动清零。IAPFF (CHPCON.6)是IAP错误标志，可以用来检查之前IAP操作成功与否。通过这些纯软件的设置，可以很方便用户对FLASH存储器进行擦除、编程和校验。

下列寄存器与IAP指令相关

CONFIG2

7	6	5	4	3	2	1	0
CBODEN	CBOV[2:0]			BOIAP	CBORST	-	-
读/写	读/写			读/写	读/写	-	-

出厂默认值: 1111 1111b

位	名称	描述
3	BOIAP	**欠压禁止IAP位** 该位决定当系统电压低于欠压侦测设定值时，IAP擦除及编程功能是否禁止。该位仅当欠压侦测功能使能后有效。 1 = 当V_{DD} 低于V_{BOD}.设定值时，IAP 擦除或编程功能禁止 0 = V_{DD} 任何电压状态下，IAP擦除及编程功能都可执行

CHPCON – 芯片控制寄存器 (TA保护)

7	6	5	4	3	2	1	0
SWRST	IAPFF	-	-	-	-	BS	IAPEN
只写	读/写	-	-	-	-	读/写	读/写

地址: 9FH　　复位值: 详见 表 6–2. SFR定义及复位值

位	名称	描述
6	IAPFF	**IAP 错误标志** 满足以下任意条件，硬件将在IAPGO(ISPTRG.0)置位后置位该位： (1)访问地址超过其大小的区域 (2) IAPCN 命令无效 (3)擦除或编程没有使能的区域

位	名称	描述
		(4) 当BOIAP(CONFIG2.5)为1、 BODEN (BODCON0.7)为1以及 BORST(BODCON0.2) 为0时，擦除或编程工作在V_{BOD}下。 该位应该由软件清零
0	IAPEN	**IAP 使能** 0 =禁用IAP功能 1 = 使能IAP功能 一旦使能IAP功能，HIRC将会被打开作为时序控制。清IAPEN应该在IAP操作最后一条指令，这样可以停止内部振荡器以减少功耗

IAPUEN – IAP更新使能(TA 保护)

7	6	5	4	3	2	1	0
-	-	-	-	-	CFUEN	LDUEN	APUEN
-	-	-	-	-	读/写	读/写	读/写

地址: A5H 复位值: 0000 0000b

位	名称	描述
2	CFUEN	**CONFIG更新使能** 0 =.禁用IAP擦除或编程CONFIG 1 =使能IAP擦除或编程CONFIG
1	LDUEN	**LDROM 更新使能** 0 =禁用IAP擦除或编程LDROM 1 = 使能IAP擦除或编程LDROM
0	APUEN	**APROM 更新使能** 0 = 禁用IAP擦除或编程APROM 1 = 使能IAP擦除或编程APROM

IAPCN – IAP 控制寄存器

7	6	5	4	3	2	1	0
IAPB[1:0]		FOEN	FCEN	FCTRL[3:0]			
读/写		读/写	读/写	读/写			

地址: AFH 复位值: 0011 0000b

位	名称	描述
7:6	IAPB[1:0]	**IAP 控制** 该字节是IAP控制命令。表 22–1. IAP模式与命令代码.
5	FOEN	
4	FCEN	
3:0	FCTRL[3:0]	

IAPAH – IAP地址高字节

7	6	5	4	3	2	1	0
IAPA[15:8]							
读/写							

地址: A7H 复位值: 0000 0000b

位	名称	描述
7:0	IAPA[15:8]	**IAP 地址高字节** IAPAH 包含地址 IAPA[15:8]

IAPAL – IAP地址低字节

7	6	5	4	3	2	1	0
IAPA[7:0]							
读/写							

地址: A6H 复位值: 0000 0000b

位	名称	描述
7:0	IAPA[7:0]	**IAP 地址低字节** IAPAL 包含地址 IAPA[7:0]

IAPFD – IAP内存数据

7	6	5	4	3	2	1	0
IAPFD[7:0]							
读/写							

地址: AEH 复位值: 0000 0000b

位	名称	描述
7:0	IAPFD[7:0]	**IAP 内存数据** 该字节包含将要读或写进内存空间的数据。编程模式下，用户需要在触发IAP之前写数据到IAPFD里，读/校验模式下，在IAP完成后从IAPFD读出数据。

IAPTRG – IAP 触发 (TA 保护)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	IAPGO
-	-	-	-	-	-	-	W

地址: A4H 复位值: 0000 0000b

位	名称	描述

0	IAPGO	**IAP 执行**
		设置该位为1开始执行IAP。该指令后，CPU保持程序计数器(PC)，IAP硬件自动管理控制该过程。IAP完成后，程序计数器继续执行。IAPGO位自动清零，保持为0。
		在触发IAP动作前，如果中断打开应该临时关闭，程序过程如下：
		`CLR EA`
		`MOV TA,#0AAH`
		`MOV TA,#55H`
		`ORL IAPTRG,#01H`
		`(SETB EA)`

22.1 IAP 命令

N76E885 通过IAP可编写 APROM, LDROM, 或 CONFIG。IAP运作模式和编程区域目标地址是IAPCN决定的。

表 22–1. IAP模式与命令代码

IAP 模式	IAPCN				IAPA[15:0] {IAPAH, IAPAL}	IAPFD[7:0]
	IAPB[1:0]	FOEN	FCEN	FCTRL[3:0]		
读公司 ID	XX[1]	0	0	1011	X	DAH
读器件 ID	XX	0	0	1100	低字节 DID: 0000H 高字节 DID: 0001H	低字节DID: 50H 高字节DID: 21H
读96位UID	XX	0	0	0100	0000H to 000BH	数据读出
APROM 页擦除	00	1	0	0010	地址写入[2]	FFH
LDROM 页擦除	01	1	0	0010	地址写入[2]	FFH
APROM 字节编程	00	1	0	0001	地址写入	数据写入
LDROM字节编程	01	1	0	0001	地址写入	数据写入
APROM 字节读	00	0	0	0000	地址写入	数据读出
LDROM 字节读	01	0	0	0000	地址写入	数据读出
擦除所有CONFIG	11	1	0	0010	0000H	FFH
CONFIG 字节编程	11	1	0	0001	CONFIG0: 0000H CONFIG1: 0001H CONFIG2: 0002H CONFIG4: 0004H	数据写入

IAP 模式	IAPCN				IAPA[15:0] {IAPAH, IAPAL}	IAPFD[7:0]
	IAPB[1:0]	FOEN	FCEN	FCTRL[3:0]		
CONFIG 字节读	11	0	0	0000	CONFIG0: 0000H CONFIG1: 0001H CONFIG2: 0002H CONFIG4: 0004H	数据读出

[1] 'X' 表示无关项
[2] 每一页是 128字节，所以地址应该是目标页的地址

22.2 IAP 用户指南

IAP可以方便用户更新FLASH内容，不过，用户必须遵循一定限制以确保IAP正确执行，否则可能引起不确定的结果，甚至损坏器件。此外下文对于正确执行IAP有很好建议。

(1)没有IAP操作时，用户必须清IAPEN (CHPCON.0)为0。可以防止系统意外触发IAP。此外，IAP需要使用内部HI RC振荡器。如果选择外部时钟源，禁止ISP将停止内部HIRC，可以达到省电的目的。注写IAPEN受TA保护。

(2)当LOCK位(CONFIG0.1) 被激活，IAP读，写或擦除仍然有效。

在进行IAP时，如果中断打开应该临时清除EA位

擦除或编程的页不能是当前代码执行的页。否则会出现不可预计程序动作，甚至破坏存储的数据

22.3 使用Flash存储器作为数据存储

在一般应用中，有时需要一些数据在断电情况下不能丢失，以便用户读回或是更新，作为系统控制的参数。N76E885支持IAP功能并且存储在flash中的字节都可以用MOVC指令读取，所有很适合作为非易失数据存储器。Flash写次数为100,000次，以下参考应用代码：

汇编例程如下:

```
;*****************************************************************************
;      This code illustrates how to use IAP to make APROM 201h as a byte of
;      Data Flash when user code is executed in APROM.
;*****************************************************************************
PAGE_ERASE_AP              EQU          00100010b
BYTE_PROGRAM_AP            EQU          00100001b

       ORG    0000h

       MOV    TA,#0AAh                  ;CHPCON is TA protected
       MOV    TA,#55h
       ORL    CHPCON,#00000001b         ;IAPEN = 1, enable IAP mode

       MOV    TA,#0AAh                  ;IAPUEN is TA protected
       MOV    TA,#55h
       ORL    IAPUEN,#00000001b         ;APUEN = 1, enable APROM update
```

```
MOV     IAPCN,#PAGE_ERASE_AP        ;Erase page 200h~27Fh
MOV     IAPAH,#02h
MOV     IAPAL,#00h
MOV     IAPFD,#0FFh
MOV     TA,#0AAh                    ;IAPTRG is TA protected
MOV     TA,#55h
ORL     IAPTRG,#00000001b           ;write '1' to IAPGO to trigger IAP process
MOV     IAPCN,#BYTE_PROGRAM_AP      ;Program 201h with 55h
MOV     IAPAH,#02h
MOV     IAPAL,#01h
MOV     IAPFD,#55h
MOV     TA,#0AAh
MOV     TA,#55h
ORL     IAPTRG,#00000001b

MOV     TA,#0AAh
MOV     TA,#55h
ANL     IAPUEN,#11111110b           ;APUEN = 0, disable APROM update

MOV     TA,#0AAh
MOV     TA,#55h
ANL     CHPCON,#11111110b           ;IAPEN = 0, disable IAP mode

MOV     DPTR,#201h
CLR     A
MOVC    A,@A+DPTR                   ;Read content of address 201h
MOV     P0,A

SJMP    $
```

C 语言例程如下:

```
//***********************************************************************************
//     This code illustrates how to use IAP to make APROM 201h as a byte of
//     Data Flash when user code is executed in APROM.
//***********************************************************************************
#define      PAGE_ERASE_AP       0x22
#define      BYTE_PROGRAM_AP     0x21

/*Data Flash, as part of APROM, is read by MOVC. Data Flash can be defined as
  128-element array in "code" area from absolute address 0x0200                */

volatile unsigned char code Data_Flash[128] _at_ 0x0200;

Main (void)
{
       TA = 0xAA;                        //CHPCON is TA protected
       TA = 0x55;
       CHPCON |= 0x01;                   //IAPEN = 1, enable IAP mode

       TA = 0xAA;                        //IAPUEN is TA protected
       TA = 0x55;
       IAPUEN |= 0x01;                   //APUEN = 1, enable APROM update

       IAPCN = PAGE_ERASE_AP;            //Erase page 200h~27Fh
       IAPAH = 0x02;
       IAPAL = 0x00;
       IAPFD = 0xFF;
       TA = 0xAA;                        //IAPTRG is TA protected
       TA = 0x55;
       IAPTRG |= 0x01;                   //write '1' to IAPGO to trigger IAP process

       IAPCN = BYTE_PROGRAM_AP;          // Program 201h with 55h
       IAPAH = 0x02;
       IAPAL = 0x01;
       IAPFD = 0x55;
       TA = 0xAA;
       TA = 0x55;
       IAPTRG |= 0x01;                   //write '1' to IAPGO to trigger IAP process

       TA = 0xAA;                        //IAPUEN is TA protected
       TA = 0x55;
       IAPUEN &= ~0x01;                  //APUEN = 0, disable APROM update

       TA = 0xAA;                        //CHPCON is TA protected
       TA = 0x55;
       CHPCON &= ~0x01;                  //IAPEN = 0, disable IAP mode

       P0 = Data_Flash[1];               //Read content of address 200h+1

       while(1);
}
```

22.4 在系统编程(ISP)

Flash存储器支持硬件编程和在应用编程（IAP）。硬件编程是在产品进入批量生产阶段，采用编程器可以节省费用和时间。然而，如果产品在研发阶段或产品需要更新软固件时，硬件编程就显得不太方便，

采用在系统编程（ISP）方式，可使这一过程变得方便。执行ISP不需要将控制器从系统板上拆下来。通过软件控制可以使设备被重新编程。因此这使得更新应用程序固件得到广泛的应用。

用户可以开发自己的引导代码放在LDROM中。LDROM最大为4KB。用户开发的引导代码可以通过并行烧录器或是在电路编程器（ICP）下载到LDROM中去。

一般来说，ISP是PC与MCU之间进行通讯。PC通过串口传输给MCU新的用户代码。然后引导代码接收这些数据，将这些数据通过IAP命令编程到用户代码区域。新唐针对N76E885提供ISP固件和PC端软件，这样可以很容易实现ISP通过UAT升级代码。更多信息请访问新唐8位微控制器网站：新唐80C51微控制器技术支持.

以下是简单ISP参考代码

```
;*****************************************************************************
;      This code illustrates how to do APROM and CONFIG IAP from LDROM.
;      APROM are re-programmed by the code to output P1 as 55h and P2 as AAh.
;      The CONFIG2 is also updated to disable BOD reset.
;      User needs to configure CONFIG0 = 0x7F, CONFIG1 = 0xFE, CONFIG2 = 0xFF.
;*****************************************************************************
PAGE_ERASE_AP               EQU         00100010b
BYTE_PROGRAM_AP             EQU         00100001b
BYTE_READ_AP                EQU         00000000b
ALL_ERASE_CONFIG            EQU         11100010b
BYTE_PROGRAM_CONFIG         EQU         11100001b
BYTE_READ_CONFIG            EQU         11000000b

       ORG    0000h

       CLR    EA                        ;disable all interrupts
       CALL   Enable_IAP

       CALL   Enable_AP_Update
       CALL   Erase_AP                  ;erase AP data
       CALL   Program_AP                ;programming AP data
       CALL   Disable_AP_Update
       CALL   Program_AP_Verify         ;verify Programmed AP data

       CALL   Read_CONFIG               ;read back CONFIG2
       CALL   Enable_CONFIG_Update
       CALL   Erase_CONFIG              ;erase CONFIG bytes
       CALL   Program_CONFIG            ;programming CONFIG2 with new data
       CALL   Disable_CONFIG_Update
       CALL   Program_CONFIG_Verify     ;verify Programmed CONFIG2

       CALL   Disable_IAP
       MOV    TA,#0AAh                  ;TA protection
       MOV    TA,#55h                   ;
       ANL    CHPCON,#11111101b         ;BS = 0, reset to APROM
       MOV    TA,#0AAh
       MOV    TA,#55h
       ORL    CHPCON,#80h               ;software reset and reboot from APROM

       SJMP   $
```

```
;*****************************************************************
;             IAP Subroutine
;*****************************************************************
Enable_IAP:
       MOV    TA,#0AAh                    ;CHPCON is TA protected
       MOV    TA,#55h
       ORL    CHPCON,#00000001b           ;IAPEN = 1, enable IAP mode
       RET

Disable_IAP:
       MOV    TA,#0AAh
       MOV    TA,#55h
       ANL    CHPCON,#11111110b           ;IAPEN = 0, disable IAP mode
       RET

Enable_AP_Update:
       MOV    TA,#0AAh                    ;IAPUEN is TA protected
       MOV    TA,#55h
       ORL    IAPUEN,#00000001b           ;APUEN = 1, enable APROM update
       RET

Disable_AP_Update:
       MOV    TA,#0AAh
       MOV    TA,#55h
       ANL    IAPUEN,#11111110b           ;APUEN = 0, disable APROM update
       RET

Enable_CONFIG_Update:
       MOV    TA,#0AAh
       MOV    TA,#55h
       ORL    IAPUEN,#00000100b           ;CFUEN = 1, enable CONFIG update
       RET

Disable_CONFIG_Update:
       MOV    TA,#0AAh
       MOV    TA,#55h
       ANL    IAPUEN,#11111011b           ;CFUEN = 0, disable CONFIG update
       RET

Trigger_IAP:
       MOV    TA,#0AAh                    ;IAPTRG is TA protected
       MOV    TA,#55h
       ORL    IAPTRG,#00000001b           ;write '1' to IAPGO to trigger IAP process
       RET

;*****************************************************************
;             IAP APROM Function
;*****************************************************************
Erase_AP:
       MOV    IAPCN,#PAGE_ERASE_AP
       MOV    IAPFD,#0FFh
       MOV    R0,#00h
Erase_AP_Loop:
       MOV    IAPAH,R0
       MOV    IAPAL,#00h
       CALL   Trigger_IAP
       MOV    IAPAL,#80h
       CALL   Trigger_IAP
       INC    R0
       CJNE   R0,#44h,Erase_AP_Loop
```

```
        RET

Program_AP:
        MOV     IAPCN,#BYTE_PROGRAM_AP
        MOV     IAPAH,#00h
        MOV     IAPAL,#00h
        MOV     DPTR,#AP_code
Program_AP_Loop:
        CLR     A
        MOVC    A,@A+DPTR
        MOV     IAPFD,A
        CALL    Trigger_IAP
        INC     DPTR
        INC     IAPAL
        MOV     A,IAPAL
        CJNE    A,#14,Program_AP_Loop
        RET

Program_AP_Verify:
        MOV     IAPCN,#BYTE_READ_AP
        MOV     IAPAH,#00h
        MOV     IAPAL,#00h
        MOV     DPTR,#AP_code
Program_AP_Verify_Loop:
        CALL    Trigger_IAP
        CLR     A
        MOVC    A,@A+DPTR
        MOV     B,A
        MOV     A,IAPFD
        CJNE    A,B,Program_AP_Verify_Error
        INC     DPTR
        INC     IAPAL
        MOV     A,IAPAL
        CJNE    A,#14,Program_AP_Verify_Loop
        RET

Program_AP_Verify_Error:
        CALL    Disable_IAP
        MOV     P0,#00h
        SJMP    $

;*****************************************************************
;             IAP CONFIG Function
;*****************************************************************
Erase_CONFIG:
        MOV     IAPCN,#ALL_ERASE_CONFIG
        MOV     IAPAH,#00h
        MOV     IAPAL,#00h
        MOV     IAPFD,#0FFh
        CALL    Trigger_IAP
        RET


Read_CONFIG:
        MOV     IAPCN,#BYTE_READ_CONFIG
        MOV     IAPAH,#00h
        MOV     IAPAL,#02h
        CALL    Trigger_IAP
        MOV     R7,IAPFD
        RET
```

```
Program_CONFIG:
       MOV    IAPCN,#BYTE_PROGRAM_CONFIG
       MOV    IAPAH,#00h
       MOV    IAPAL,#02h
       MOV    A,R7
       ANL    A,#11111011b
       MOV    IAPFD,A                    ;disable BOD reset
       MOV    R6,A                       ;temp data
       CALL   Trigger_IAP
       RET

Program_CONFIG_Verify:
       MOV    IAPCN,#BYTE_READ_CONFIG
       MOV    IAPAH,#00h
       MOV    IAPAL,#02h
       CALL   Trigger_IAP
       MOV    B,R6
       MOV    A,IAPFD
       CJNE   A,B,Program_CONFIG_Verify_Error
       RET

Program_CONFIG_Verify_Error:
       CALL   Disable_IAP
       MOV    P0,#00h
       SJMP   $

;*********************************************************************
;             APROM code
;*********************************************************************
AP_code:
       DB     75h,0B1h, 00h              ;OPCODEs of "MOV    P0M1,#0"
       DB     75h,0ACh, 00h              ;OPCODEs of "MOV    P3M1,#0"
       DB     75h, 90h, 55h              ;OPCODEs of "MOV    P1,#55h"
       DB     75h,0A0h,0AAh              ;OPCODEs of "MOV    P2,#0AAh"
       DB     80h,0FEh                   ;OPCODEs of "SJMP   $"

       END
```

23. 电源管理

N76E885有几个途径可以帮助用户控制设备功耗。省电的途径有掉电模式和空闲模式。为了稳定的电流消耗，必须处理好每个引脚的模式和状态。每个引脚状态需和外部上下拉一致，比如上拉就要输出1，下拉就要输出0。如果引脚是悬浮的，建议用户配置端口为准双向模式。如果P1.2配置为只输入引脚，必须外接上拉或是下拉电阻或者通过设置P12UP (P1M2.2)打开内部上拉。

PCON – 电源管理寄存器

7	6	5	4	3	2	1	0
SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL
读/写	读/写	-	读/写	读/写	读/写	读/写	读/写

地址: 87H　　复位值: 详见 表 6–2. SFR定义及复位值

位	名称	描述
1	PD	**掉电模式** 设置该位使MCU进入掉电模式。在此模式下，CPU和外设时钟停止，程序计数器（PC）挂起。此时为最小功耗。CPU从掉电模式下唤醒后，该位自动由硬件清零，且在系统唤醒之前程序继续执行中断服务程序（ISR）。从ISR返回后，设备继续执行系统进入掉电模式时所处的指令。 注如果IDL位和PD位同时置位，MCU进入掉电模式。从掉电模式退出后不会进入空闲模式。
0	IDL	**空闲模式** 设置该位使MCU进入空闲模式。在此模式下，CPU时钟停止，且程序计数器（PC）挂起，但是所有外设继续工作。CPU从空闲模式唤醒后，该位自动由硬件清零，且在系统唤醒之前程序继续执行中断服务程序（ISR）。从ISR返回后，设备继续执行系统进入掉电模式时所处的指令。

P1M2 – 端口1模式选择 2

7	6	5	4	3	2	1	0
-	-	-	-	CLOEN	P12UP	P1M2.1	P1M2.0
-	-	-	-	读/写	读/写	读/写	读/写

地址: B4H　　复位值: 0000 0000b

位	名称	描述
2	P12UP	**P1.2 上拉使能** 0 = 禁用P1.2上拉 1 =使能 P1.2上拉 该位仅在RPD (CONFIG0.2)设置为0有效。当选择作为复位脚时，上拉总是打开的

23.1 空闲模式

空闲模式下通过保持程序计数器使CPU挂起。在空闲模式下没有程序代码的取指和运行。这迫使CPU处于待机状态。程序计数器(PC)，堆栈指针(SP)，程序状态字(PSW)，累加器(ACC)，和其他寄存器在空

闲模式下保持其值不变。端口引脚保持进入空闲前的逻辑状态。通常空闲模式下的功耗约为工作状态下的一半。

既然在空闲模式下，外设电路如定时器和串口的时钟依然工作，则可以通过相应使能的中断唤醒CPU。用户可通过向IDL (PCON.0)写1进入空闲模式。这条指令是系统进入空闲模式前的最后一条指令。

有两种方法可以中止空闲模式，方法一，任何使能的中断可以使系统退出空闲模式。这将自动清零IDL位，中止空闲模式，且将执行中断服务程序(ISR)。在使用指令RETI跳出ISR，所执行的将是进入空闲模式指令后的程序。第二种方法是除软件复位外的所有复位。如果看门狗复位用来退出空闲模式，WIDPD (WDCON.4)需要设置为1，让WDT在空闲模式下继续运行

23.2 掉电模式

掉电模式是N76E885进入的最低功耗状态，保持功耗在“微安uA” 级。此时停止系统时钟源。CPU和外设如定时器或UART都待机，Flash 内存也停止，所有动作完全停止，功耗降至最低。可以通过向PD (PCON.1)写1进入掉电模式。这条指令是系统进入掉电模式前的最后一条指令。在掉电模式下，RAM保存其内容，端口引脚的值也保持不变。

N76E885有多种方法可以退出掉电模式。

方法一，除软件复位外的所有复位..。欠压检测复位将把CPU由掉电模式唤醒，在系统进入掉电模式之前要确保使能欠压检测。即使为了降低功耗，我们还是建议在掉电模式下开启BOD欠压检测功能。当然RST引脚的复位或上电复位也可以唤醒掉电状态。RST引脚复位或上电复位后，CPU初始化，并开始执行程序。

方法二，可以通过外部中断将N76E885从掉电模式唤醒。在中断条件设定的情况下，外部中断触发信号会异步重启系统时钟。在振荡器稳定后，设备执行相应的外部中断的中断服务例程（ISR）。从ISR返回后，设备继续执行系统进入掉电模式时所处的指令。可以将芯片从掉电模式唤醒的中断有：外部中断$\overline{\text{INT0}}$ 和$\overline{\text{INT1}}$、引脚中断、WDT中断、WTK中断和欠压中断。

24. 时钟系统

N76E885拥有多种时钟源可供选择，这样在应用中可以有多种选择使系统性能发挥到最佳同时功耗降到最低。共有5种系统时钟源可供选择其包括:内部高速/低速振荡器、外部高速/低速晶体振荡器和XIN引脚外部时钟源，可以通过软件选择。N76E885内嵌2个内部RC振荡器一个10 kHz低速和一个 22.1184MHz高速RC振荡器，高速22.1184MHz误差在出厂时校准到±2%（全温度全电压状态下）。如果时钟源选择了外部晶体振荡器，那么将有两种频率可供选择：高速 2MHz 至 25MHz，低速为32.768 kHz。CKDIV除频器可以让N76E885在高效率与低功耗之间有个灵活的选择。

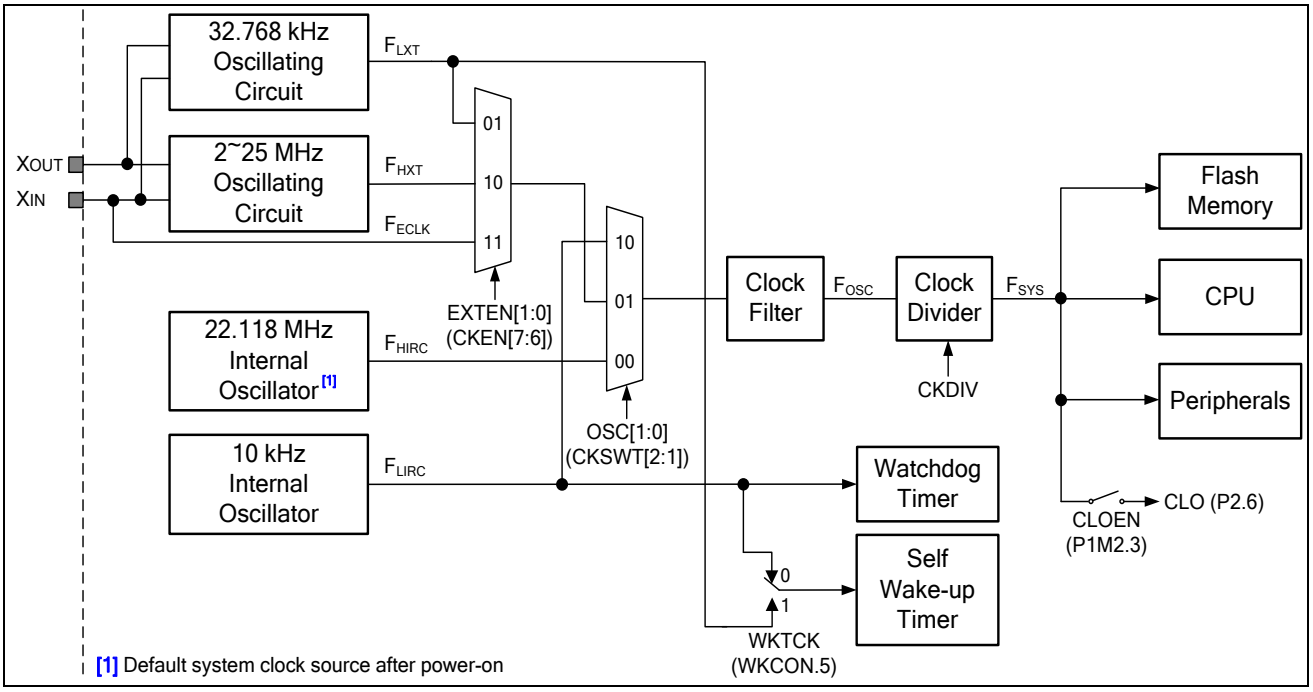


图 24-1 时钟系统框图

24.1 系统时钟源

N76E885共有5种系统时钟源其中包括: 内部高速/低速振荡器、外部高速/低速晶体振荡器和XIN引脚外部时钟源。它们每一个都可以作为N76E885的系统时钟。开启不同的时钟源可能会影响到多功能引脚P1.0/XIN 和 P1.1/XOUT。

24.2 内部振荡器

N76E885内部有两个RC振荡器，一个高速22.1184MHz（HIRC）和一个低速10 kHz（LIRC）。它们都可被选择作为系统时钟。通过设置HIRCEN (CKEN.5)位使能HIRC。用户可设置OSC[1:0] (CKSWT

[2:1]) 为[1,1] 或 [0,0]选择HIRC作为系统时钟，设置OSC[1:0] (CKSWT [2:1]) 为 [1,0] 选择LIRC作为系统时钟。注意N76E885上电后HIRC 和 LIRC都默认开启，并且系统默认HIRC为系统时钟。当系统使用内部振荡器作为时钟源时，XIN 和 XOUT自动作为普通I/O P3.2和P3.3 ，用以扩展普通I/O的数量。可以通过配置P1M2寄存器来选择P1.0 和 P1.1的输出模式。

24.3 外部晶体振荡器或时钟输入

外部时钟源有三个可选时钟源，2 MHz 到 25MHz高速晶体振荡器（HXT）、32.768 kHz低速晶体振荡器（LXT）和通过XIN引脚输入的外部时钟(ECLK)。通过设置EXTEN[1:0] (CKEN[7:6])的值打开相应的时钟源。用户可设置OSC[2:0] (CKSWT [2:0]) 为 [0,1,0]选择外部时钟作为系统时钟。当HXT或LXT作为系统时钟时，XIN 和 XOUT分别作为时钟源的输入和输出。晶体振荡器应该连接到XIN 和 XOUT上。当使能并选择ECLK作为系统时钟时，外部时钟经过XIN给系统提供时钟。一般应用中驱动XIN是通过有源振荡器或是来自另外一个主机的时钟。当系统时钟选择ECLK时，XOUT引脚自动作为普通I/O P1.1。可以通过配置P3M1和P3M2寄存器来选择P1.1的输出模式。用户请特别注意，选择外部时钟ECLK模式时，输入XIN的时钟信号电压不能超过1.8V，否则会损坏该功能模块。

24.4 系统时钟切换

N76E885可以通过软件设置CKSWT 和 CKEN寄存器动态的切换时钟源。这给应用带来了很大方便。注意写这些特殊寄存器会有TA时序保护。在这些可控的时钟下，系统时钟可以在外部时钟和内部时钟甚至是内部高速与低速之间自由的切换。然而在切换时钟源时，必须保证待切换的时钟源已稳定。因此，用户需要遵循以下设置步骤才能成功完成时钟切换。用户首先要通过配置CKEN寄存器打开目标时钟源，再通过查询CKSWT寄存器中对应的标志位来确定时钟源是否稳定，最后写OSC[2:0]切换到目标时钟。这些步骤过后，将会成功的切换时钟。如果用户关心功耗的话可以将原先的时钟源关闭了。如果不遵守以上步骤，硬件将会采取以下一些措施来应对这些违规的操作。

1.如果用户试图改变CKEN的值来关闭当前时钟源，设备将忽略这个操作。系统时钟维持现状，CKEN值不变。

2.如果用户试图改变OSC[2:0]的值来切换系统时钟而待切换新时钟源未被打开，OSC[2:0]值将会立刻被更新。但是系统时钟保持不变，CKSWTF (CLKEN.0)会被硬件置位。

3.如果用户切换系统时钟源但是目标时钟源已经打开还没稳定，那么那么硬件会等待目标稳定后再切换过去。在等待期间，设备继续以原有时钟源工作并且CKSWTF会被置1。等到目标时钟源稳定标志位(见CKSWT[7:5] 和 CKSWT.3)被置位，时钟将会成功切换，CKSWTF会被硬件自动清0。

下面是系统时钟由HIRC切换到HXT的参考例程

```
       MOV    TA,#0AAh                  ;TA protection
       MOV    TA,#55h                   ;
       ORL    CKEN,#10000000b           ;Enable the HXT

;******Polling can be ignored if not disabling the original clock source******
Polling_HXT_stable:                     ;Waiting for the HXT stable
       MOV    A,CKSWT
       JNB    ACC.7, Polling_HXT_stable
;*******************************************************************************

       MOV    TA,#0AAh                  ;TA protection
       MOV    TA,#55h                   ;
       MOV    CKSWT,#02h                ;switch the clock source to the HXT

;******Disable the original HIRC clock source, for example******
       MOV    TA,#0AAh                  ;TA protection
       MOV    TA,#55h                   ;
       ANL    CKEN,#11011111b           ;Disable the IHRC
;*************************************************************
```

CKSWT – Clock Switch (TA 保护)

7	6	5	4	3	2	1	0
HXTST	LXTST	HIRCST	-	ECLKST	OSC[1:0]		-
只读	只读	只读	-	只读	只写		-

地址: 96H　　复位值: 0011 0000b

位	名称	描述
7	HXTST	**HXT 状态** 0 = HXT不稳定或是没有开启 1 = HXT 开启并稳定
6	LXTST	**LXT状态** 0 = LXT 不稳定或是没有开启 1 = LXT开启并稳定
5	HIRCST	**HIRC状态** 0 = HIRC 不稳定或是没有开启 1 = HIRC开启并稳定
3	ECLKST	**ECLK状态** 0 = ECLK 不稳定或是没有开启 1 = ECLK开启并稳定
2:1	OSC[1:0]	**振荡器选择位** 该位是用来选择系统时钟源 00 = 内部 22.118 MHz 晶振. 01 = 外部时钟控制，通过EXTEN[1:0] (CKEN[7:6]) 设置. 10 = 内部 10 kHz 晶振. 11 = 保留位. 注意该位段只写，读回来的值的可能与当前时钟源不一致

CKEN – 时钟使能寄存器 (TA 保护)

7	6	5	4	3	2	1	0
EXTEN[1:0]		HIRCEN	-	-	-	-	CKSWTF
读/写		读/写	-	-	-	-	只读

地址: 97H　　复位值: 0011 0000b

位	名称	描述
7:6	EXTEN[1:0]	**外部时钟源使能** 该位段用来开启外部时钟源。如果系统想要选择外部时钟源作为系统时钟同时也需要设置OSC[1:0] 为 [0,1]。 00 = I/O.禁用外部时钟。P1.0 和P1.1作为普通I/O使用 01 = 使能 LXT 32.768 kHz 10 = 使能 HXT 2 MHz 至 25 MHz 11 = 使能ECLK，通过XIN 输入外部时钟信号
5	HIRCEN	**HIRC 22.1184MHz使能位** 0 =关闭 HIRC 1 =使能 HIRC 注意一旦设置IAPEN (CHPCON.0)位开启IAP功能，HIRC将会自动使能，硬件也会设置HIRCEN 和 HIRCST位。IAPEN被清除后，HIRCEN 和 EHRCST位会恢复为原始值
0	CKSWTF	**时钟切换错误标志位** 0 =之前的系统时钟源切换成功 1 = 先前用户试图切换系统时钟的时钟源没有开启或是不稳定。如果待切换的时钟不稳定该位将一直保持为1，直到时钟源稳定并切换成功为止。

24.5 系统时钟除频

振荡频率(F_{OSC})通过配置除频寄存器CKDIV以整数倍（最大到1/510）除频后再供给系统作为系统时钟(F_{SYS})。这一特征可以临时让MCU跑在很低的速度下来降低功耗。通过系统除频，可以让MCU在正常工作模式下运行很低速度保证其能够相应一些事件而不只是中断事件（比如空闲模式只能通过中断事件退出）。这有可能比空闲模式还要省电。这样可以避开在掉电模式情况下需要等待振荡器重新起振的时间。CKDIV的值可以在任何时间被程序改变除了不能在中断里改变。

CKDIV – 时钟除频

7	6	5	4	3	2	1	0
CKDIV[7:0]							
读/写							

地址: 95H　　复位值: 0000 0000b

位	名称	描述
7:0	CKDIV[7:0]	**时钟除频** 下面是系统频率F_{SYS}计算公式 当 CKDIV = 00H时，$F_{SYS} = F_{OSC}$ 当 CKDIV = 01H ~ FFH时，$F_{SYS} = \frac{F_{OSC}}{2 \times CKDIV}$

24.6 系统时钟输出

N76E885提供一个CLO(P2.6)引脚可以输出系统时钟，该频率等同F_{SYS}。通过设置CLOEN (P1M2.3)位打开这个功能。在掉电模式下CLO输出也会停止，因为系统时钟已被关闭。注意当有干扰问题或是功耗问题时，用户最好关闭CLO输出。

P1M2 – 端口1模式选择寄存器 2

7	6	5	4	3	2	1	0
-	-	-	-	CLOEN	P12UP	P1M2.1	P1M2.0
-	-	-	-	读/写	读/写	读/写	读/写

地址: B4H　　复位值: 0000 0000b

位	名称	描述
3	CLOEN	**系统时钟输出使能** 0 =禁用系统时钟输出 1 =使能系统时钟输出，从CLO(P2.6)输出

25. 电源监控

为了防止上电和掉电时出现错误执行，N76E885提供两个电源监控功能，上电检测和欠压检测

25.1 上电复位 (POR)

上电检测功能用于检测电源上升到系统可以工作的电压。上电检测后，POF (PCON.4) 将置1表明为冷复位，上电复位完成。POF标志可由软件清零。

PCON – 电源控制寄存器

7	6	5	4	3	2	1	0
SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL
读/写	读/写	-	读/写	读/写	读/写	读/写	读/写

地址: 87H　　复位值: 详见 表 6–2. SFR定义及复位值

位	名称	描述
4	POF	**上电复位标志** 当上电后该位置1，用以标示当前冷复位，上电复位完成。其它任何复位不会影响该位，建议通过软件清0。

25.2 欠压检测(BOD)

另一个电源监控功能是欠压检测，欠压检测电路是用于监测执行期间V_{DD}电平。有四个可编程的欠压检测触发电平，以适用于宽电压应用。8阶电平1.7V, 2.0V, 2.2V, 2.4V, 2.7V, 3.0V, 3.7V, 及 4.3V可由CONFIG2的CBOV设置选择。当然在通电后也可以通过设置BOV[1:0] (BODCON0[5:4])来改变BOD电平。当V_{DD} 下降到所选择的欠压检测触发电平(V_{BOD})，欠压检测逻辑将复位CPU或请求欠压检测中断。用户可结合不同应用决定设备是欠压复位还是产生欠压中断。上电后也可以通过软件打开BOD，注意在软件打开BOD后需要等待2到3个LIRC时钟才能正常工作。

当V_{DD}下降到V_{BOD}下且BORST (BODCON0.2)为0时，BOD将会请求中断。此情况下，BOF (BODCON0.3)将被置1。用户清除该标志后，V_{DD}依然保持在V_{BOD}下，BOF不会被再次置1。BOF仅通知用户电源电压下降发生。当V_{DD}上升到高于V_{BOD}时，BOF将置1，以示电源恢复。BOD电路提供了一个很有用的状态位BOS (BODCON0.0)，可以用来指示当前是欠压还是电源已经恢复。设置BORST为1将开启欠压复位功能。欠压复位过后，BORF (BODCON0.1)将会被硬件置1。它不会被其它复位重置除上电复位外。该位可以通过软件清除。注意BODCON0所有位的写入都受时序访问TA保护。

N76E885支持低功率BOD模式，在为了节省电流消耗的同时最大的发挥BOD检测性能。通过设置LPBOD[1:0] (BODCON1[2:1])可以周期性的开启传感器的电源电压，通常每隔1.6 ms, 6.4 ms, 或 25.6

ms。这样可以节省很多电。注意在低功率BOD模式下，欠压检测迟滞特性将会消失。注，BODCON0带有TA保护。

对于噪声敏感的系统，N76E885有一个BOD的滤波器可以避免电源噪声无意识地触发BOD事件。BOD滤波器上电默认开启，如果用户想要一个快速反应的BOD系统可以通过清BODFLT (BODCON1.0)为0来关闭。最小欠压检测脉冲宽度见表 25–2..

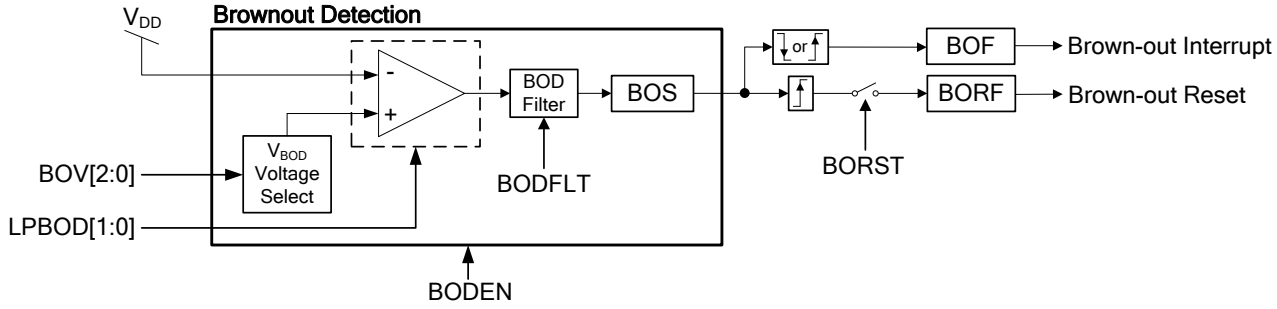


图 25-1. 欠压检测框图

CONFIG2

7	6	5	4	3	2	1	0
CBODEN	CBOV[2:0]			BOIAP	CBORST	-	-
读/写	读/写			读/写	读/写	-	-

出厂默认值: 1111 1111b

位	名称	描述
7	CBODEN	**配置欠压侦测使能位** 1 = 欠压侦测功能打开. 0 = 欠压侦测功能关闭
6:4	CBOV[1:0]	**配置欠压侦测电压选择位** 111 = V_{BOD} 生效电压 1.7V. 110 = V_{BOD} 生效电压2.0V. 101 = V_{BOD} 生效电压2.2V. 100 = V_{BOD} 生效电压 2.4V. 011 = V_{BOD} 生效电压 2.7V. 010 = V_{BOD} 生效电压 3.0V. 001 = V_{BOD} 生效电压 3.7V. 000 = V_{BOD} 生效电压 4.3V.
2	CBORST	**配置欠压检测复位使能** 该位决定在电源电压跌到以V_{BOD}下时是否产生欠压检测复位 1 =使能欠压检测复位 0 =禁用欠压检测复位

BODCON0 – 欠压检测控制寄存器0 (TA 保护)

7	6	5	4	3	2	1	0
BODEN[1]	BOV[2:0][1]			BOF[2]	BORST[1]	BORF	BOS
读/写	读/写			读/写	读/写	读/写	R

地址: A3H　　复位值: 详见 表 6–2. SFR定义及复位值

位	名称	描述
7	BODEN	**欠压检测使能** 0 =禁用欠压检测电路 1 =使能欠压检测电路 注意在开启该功能后需要2到3个LIRC时钟BOD才能正常工作
6:4	BOV[2:0]	**欠压侦测电压选择位** 111 = V_{BOD} 生效电压 1.7V. 110 = V_{BOD} 生效电压2.0V. 101 = V_{BOD} 生效电压2.2V. 100 = V_{BOD} 生效电压 2.4V. 011 = V_{BOD} 生效电压 2.7V. 010 = V_{BOD} 生效电压 3.0V. 001 = V_{BOD} 生效电压 3.7V. 000 = V_{BOD} 生效电压 4.3V.
3	BOF	**欠压中断标志** 当V_{DD}下降到V_{BOD}以下或V_{DD}上升到V_{BOD}以上时，该标志由硬件设置为逻辑1。如果EBOD (EIE.2)和EA (IE.7) 都置位，将请求欠压检测中断。该位必须由软件清零。
2	BORST	**欠压检测复位使能** 该位决定在电源电压跌到以V_{BOD}下时是否产生欠压检测复位 0 =禁用欠压检测复位 1 =使能欠压检测复位
1	BORF	**欠压复位标志** 当MCU发生欠压复位时，该位被硬件值1。建议通过软件清除该位。
0	BOS	**欠压状态标志** 在BOD电路开启时，该位反应V_{DD}与V_{BOD}比较情况。BOD电路关闭时保持为0. 0 = V_{DD}电压大于V_{BOD}或是BOD电路关闭 1 = V_{DD}电压小于V_{BOD} 注该位为只读位

[1] 所有复位后BODEN、 BOV[1:0]和 BORST初始化的值是直接通过加载CONFIG0 位 7、 位 5~ 4 和 位 2决定的
[2] BOF复位后的值依据CONFIG2的设置和V_{DD}的电平.

表 25–1. BOF 复位值

CBODEN (CONFIG2.7)	CBORST (CONFIG2.2)	V_{DD} 电平	BOF
1	1	总是> V_{BOD}	0
1	0	< V_{BOD}	1
1	0	> V_{BOD}	0
0	X	X	0

BODCON1 –欠压检测控制寄存器1 (TA 保护)

7	6	5	4	3	2	1	0
-	-	-	-	-	LPBOD[1:0]		BODFLT
-	-	-	-	-	读/写		读/写

地址: ABH　　复位值: 详见 表 6–2. SFR定义及复位值

位	名称	描述
2:1	LPBOD[1:0]	**低功率BOD使能** 00 = BOD正常模式，BOD电路总是开启 01 = BOD低功耗模式1，每隔1.6ms周期性开启BOD电路 10 = BOD低功耗模式2，每隔6.4ms周期性开启BOD电路 11 = BOD低功耗模式3，每隔25.6ms周期性开启BOD电路
0	BODFLT	**BOD滤波器控制** 当系统时钟选择HIRC、HXT或ECLK并且BOD没有工作在低功率模式下，BOD有一个滤波器计数32个系统时钟F_{SYS}来滤除电源噪声。其它情况下滤波器计数2个LIRC时钟 当CPU停在掉电模式时，BOD滤波计数一直是2个LIRC时钟 BOD滤波器有效地避免电源噪声误触发BOD时间发生。设置该位可以开启或是关闭BOD滤波功能 0 =禁用BOD滤波器 1 =使能BOD滤波器(上电默认开启)

表 25–2.欠压侦测脉宽最小值

BODFLT (BODCON1.1)	BOD 工作模式	系统时钟源	欠压侦测脉宽最小值
0	正常工作模式 (LPBOD[1:0] = [0,0])	任意时钟源	Typ. 1μs
	低电压工作模式1 (LPBOD[1:0] = [0,1])	任意时钟源	16 ($1/F_{LIRC}$)
	低电压工作模式2 (LPBOD[1:0] = [1,0])	任意时钟源	64 ($1/F_{LIRC}$)
	低电压工作模式3 (LPBOD[1:0] = [1,1])	任意时钟源	256 ($1/F_{LIRC}$)
1	正常工作模式 (LPBOD[1:0] = [0,0])	HIRC/HXT/ECLK	正常工作模式: 32 ($1/F_{SYS}$) 空闲模式: 32 ($1/F_{SYS}$) 掉电模式: 2 ($1/F_{LIRC}$)
		LIRC/LXT	2 ($1/F_{LIRC}$)
	低电压工作模式1 (LPBOD[1:0] = [0,1])	任意时钟源	18 ($1/F_{LIRC}$)
	低电压工作模式2 (LPBOD[1:0] = [1,0])	任意时钟源	66 ($1/F_{LIRC}$)
	低电压工作模式3 (LPBOD[1:0] = [1,1])	任意时钟源	258 ($1/F_{LIRC}$)

26. 复位

N76E885的复位条件有集中类型。 通常，大部分特殊功能寄存器复位后的值与复位条件无关，但是一些复位源的标志位的状态取决于复位源.有5种方法使芯片进入复位状态。 他们是上电复位,外部复位脚复位,，软件复位，看门狗定时器复位，以及欠压复位。

26.1 上电复位

N76E885包含内部上电参考电压。在上电过程中，当V_{DD}低于参考电压门限值，该参考电压保持CPU为复位模式。这种设计使CPU在V_{DD} 不满足执行读取存储器时，不访问程序存储器空间。如果从程序存储器读取并执行一个不确定的操作码，可能会使CPU甚至是整个系统进入错误状态。V_{DD} 上升到参考门限电压以上，系统工作，所选的振荡器起振，程序从0000H开始执行。同时，上电标志 POF (PCON.4) 置1表示冷复位，上电复位完成。注：上电后，内部RAM的内容不确定。建议用户初始化RAM。

建议通过软件清除POF为0，以检测在下一次复位是冷复位还是热复位。如果是由掉电或上电引起的冷复位，POF 将再次置1。如果是由其他复位源引起的热复位，POF将保持为0。用户可以检测复位标志位，处理热复位事件。

PCON – 电源控制寄存器

7	6	5	4	3	2	1	0
SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL
读/写	读/写	-	读/写	读/写	读/写	读/写	读/写

地址: 87H　　复位值: 详见 表 6–2. SFR定义及复位值

位	名称	描述
4	POF	**上电复位标志** 当上电后该位置1，用以标示当前冷复位，上电复位完成。其它任何复位不会影响该位，建议通过软件清0。

26.2 欠压复位

欠压检测电路于用监测系统运行时V_{DD}电平。当V_{DD}下降到所选的欠压触发电平 (V_{BOD})或上升到V_{BOD}，如果BORST (BODCON0.2) 置 1，欠压检测逻辑将复位CPU。发生欠压复位后，BORF (BODCON0.1)通过硬件你置1，除上电复位及欠压复位，该位不会置1，并通过软件清0。

BODCON0 – 欠压侦测控制寄存器0 (TA 保护)

7	6	5	4	3	2	1	0
BODEN	BOV[2:0]			BOF	BORST	BORF	BOS
读/写	读/写			读/写	读/写	读/写	R

地址: A3H　　复位值: 详见 表 6–2. SFR定义及复位值

位	名称	描述
1	BORF	**欠压复位标志位** 当MCU发生欠压复位，该位将被硬件置1，建议复位发生后软件清0.

26.3 外部复位脚复位

$\overline{RST}$复位引脚是硬件为史密特触发复位输入脚。外部$\overline{RST}$引脚保持在最少24个机器周期的低电平，以确保能检测到有效的硬件复位信号，就完成一次硬件复位动作。 复位电路同步于内部复位信号。因此复位是同步动作，要求时钟在此期间运行以引起外部复位。

一旦芯片满足复位条件，只要RST引脚上电平为0就会保持在复位状态。在$\overline{RST}$改为高电平之后，CPU将在两个机器周期内退出复位状态，并从0000H开始执行代码。如果CPU在掉电模式下，有RST引脚复位时，触发硬件复位的方法略有不同。因为掉电模式停止系统时钟，复位信号将同步引起系统时钟恢复。在系统时钟稳定后，CPU 将进入复位状态，然后退出并从0000H地址开始执行应用程序。

RSTPINF (AUXR1.6) 位为复位标志位，用于标示发生外部复位。当发生外部复位后，该位硬件置1。除上电复位及外部复位脚复位外，该位不会置1，并通过软件清0。

AUXR1 – 辅助寄存器1

7	6	5	4	3	2	1	0
SWRF	RSTPINF	T1LXTM	T0LXTM	GF2	UART0PX	0	DPS
读/写	读/写	读/写	读/写	读/写	读/写	R	读/写

地址: A2H　　复位值: 详见 表 6–2. SFR定义及复位值

位	名称	描述
6	RSTPINF	**外部复位标志位** 当MCU发生外部复位脚复位，该位将被硬件置1，建议复位发生后软件清0.

26.4 看门狗定时器复位

看门狗定时器是一个带可编程溢出时间间隔的自由运行的定时器。用户可以在任何时候清除看门狗定时器，使它重新开始计数。当选择的溢出时间间隔发生溢出后，看门狗定时器将直接复位系统。其复位条件通过硬件保持两个机器周期。复位完成后，芯片从0000H开始运行。

一旦由看门狗定时器引起复位，看门狗定时器复位标志WDTRF (WDCON0.3)将置位。除上电复位外该位保持不变，用户可以通过软件清 WDTRF。

WDCON – 看门狗定时器控制寄存器 (TA保护)

7	6	5	4	3	2	1	0
WDTEN	WDCLR	WDTF	WIDPD	WDTRF	WDPS[2:0]		
读/写	读/写	读/写	读/写	读/写	读/写		

地址: AAH　　复位值: 详见 表 6–2. SFR定义及复位值

位	名称	描述
3	WDTRF	**看门狗复位标志** WDT 复位标志。当MCU复位时，该位由硬件置位。该位由硬件清零。

26.5 软件复位

N76E885 增强了软件复位功能。允许应用程序以软件方式复位整个系统。这对于ISP动作结束后非常有用。例如，如果 LDROM更新APROM，ISP 完成且APROM 中代码已更新，软件复位可使CPU 从APROM 中启动以检查 APROM 中代码。写 1 到 SWRST (CHPCON.7) 触发软件复位。注该位为TA 保护。见下面例程。

一旦发生软件复位SWRF (AUXR1.7) 会被硬件自动设为1。用户可通过读取该位来确定复位发生原因。除上电复位或软件复位外SWRF不会被其它复位修改。该位通过软件清零。

CHPCON – 芯片控制(TA 保护)

7	6	5	4	3	2	1	0
SWRST	IAPFF	-	-	-	-	BS	IAPEN
W	读/写	-	-	-	-	读/写	读/写

地址: 9FH　　复位值: 详见 表 6–2. SFR定义及复位值

位	名称	描述
7	SWRST	**软件复位** 对该位写1，芯片执行软件复位，复位完成后该位自动清0.

AUXR1 – 辅助寄存器1

7	6	5	4	3	2	1	0
SWRF	RSTPINF	T1LXTM	T0LXTM	GF2	UART0PX	0	DPS
读/写	读/写	读/写	读/写	读/写	读/写	R	读/写

地址: A2H　　复位值: 详见 表 6–2. SFR定义及复位值

位	名称	描述
7	SWRF	**软件复位标志位** 当MCU发生软件复位后，该位硬件置1。建议程序中清0.

软件复位例程如下

```
ANL    AUXR1,#01111111b    ;software reset flag clear
...
...
CLR    EA
MOV    TA,#0AAh
MOV    TA,#55h
ORL    CHPCON,#10000000b   ;software reset
```

26.6 启动选择

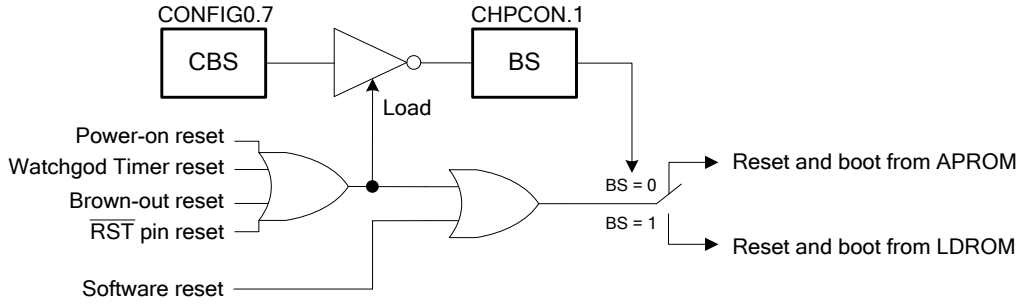


图 26-1. 启动选择结构图

N76E885提供给用户灵活的启动选择以适用于不同应用。CHPCON.1的BS位用于决定复位后CPU从APROM 或是LDROM 中启动。复位后，如果BS = 0，CPU从APPROM 中启动，由0000H开始运行.。反之CPU 从 LDROM启动，从0000H开始运行。注：每次复位后，BS由配置位的CONFIG0.7，CBS位读取数值写入BS。

CONFIG0

7	6	5	4	3	2	1	0
CBS	-	OCDPWM	OCDEN	-	RPD	LOCK	-
读/写	-	读/写	读/写	-	读/写	读/写	-

出厂默认值: 1111 1111b

位	名称	描述
7	CBS	**配置启动选项** 该位定义除软件复位外的所有复位后MCU从哪个区块启动。 1 = 除软件复位外的所有复位后MCU从APROM启动。 0 = 除软件复位外的所有复位后MCU从LDROM启动。

CHPCON – 芯片控制 (TA 保护)

7	6	5	4	3	2	1	0
SWRST	IAPFF	-	-	-	-	BS[1]	IAPEN
W	读/写	-	-	-	-	读/写	读/写

地址: 9FH　　复位值: 详见 表 6–2. SFR定义及复位值

位	名称	描述
1	BS	**启动选择** 定义复位后MCU由哪块启动。 0 =由APROM启动。 1 =由LDROM启动。

[1] 注该位由复位后（除软件复位外）读取CONFIG0.7的CBS位内容并写入。软件复位后保持不变.

注CPU从所有复位状态释放后，硬件将检查BS位（非CBS）以决定是由APROM还是LDROM启动。

26.7 复位状态

复位状态不会影响片上RAM。复位期间，RAM中的数据保留。注如果V_{DD}下降到近1.2V时，RAM中的数据可能会丢失。这是RAM保存所需的最小电压。因此，在上电复位后，在电源失效的状态下，RAM中的内容将不确定。如果电源下降到数据保存的最小电压以下，RAM的内容也将丢失。

复位后，除一些受不同复位事件影响的寄存器外，大多数寄存器回到初始值。见表 6–2. SFR定义及复位值 对于所有寄存器的初始状态，一些特殊功能寄存器初始值取决于不同复位源。一旦复位，程序指针强制切换至 0000H 并适用于全复位条件。备注:堆栈指针复位至07H 同时堆栈内的数据可能丢失，即便RAM内的数据未改变。

复位状态下，所有外设及中断关闭，所有管脚值为FFH，并切换至输入模式。

27. 辅助功能

27.1 双DPTR

传统8051架构仅有一组DPTR（数据指针）。在仅一组DPTR的结构中，当需要从某个绝对地址移动数据至另一个绝对地址时，程序非常冗长。N76E885 提供两组数据指针，这样程序可以分别同时定义源地址和目标地址，直接进行数据移动。程序通过DPS (AUXR1.0)位切换DPTR 及 DPTR1。

下例为64字节双DPTR。给定源地址和目标地址，然后采用循环指令可以简单移动整组数据，比单DPTR指令节省许多。相对ORL 或 ANL指令，使用INC AUXR1 指令是最简短方式(2 字节)。AUXR1.1具有硬件拉0 功能，所以每次执行加一指令即可，不会影响该寄存器其它控制位。

```
        MOV     R0,#64              ;赋值需移动字节数
        MOV     DPTR,#D_Addr        ;写入目标地址
        INC     AUXR1               ;更改有效DPTR指针
        MOV     DPTR,#S_Addr        ;写入源地址
LOOP:
        MOVX    A,@DPTR             ;读取源地址内数据入累加
        INC     AUXR1               ;更改有效DPTR
        MOVX    @DPTR,A             ;将累加器内源地址数据写入目标地址
        INC     DPTR                ;递增目标地址
        INC     AUXR1               ;更改有效DPTR
        INC     DPTR                ;递增源地址
        DJNZ    R0,LOOP
        INC     AUXR1               ;(可选) 重置DPS
```

AUXR1 第三位用于提供用户一个通用标志位，可以通过软件置1或清除

DPL – 数据指针低字节

7	6	5	4	3	2	1	0
DPL[7:0]							
读/写							

地址: 82H　　复位值: 0000 0000b

位	名称	描述
7:0	DPL[7:0]	**数据指针低字节** 该字节与DPH搭配组成16位数据指针DPTR定义指向非易失性存储空间或程序存储空间地址。DPS (DPS.0) 位定义指向 DPTR 或 DPTR1哪一个为当前有效。

DPH –数据指针高字节

7	6	5	4	3	2	1	0
DPH[7:0]							
读/写							

地址: 83H 复位值: 0000 0000b

位	名称	描述
7:0	DPH[7:0]	**数据指针高字节** 该字节与DPL搭配组成16位数据指针DPTR定义指向非易失性存储空间或程序存储空间地址。DPS (DPS.0) 位定义指向 DPTR 或 DPTR1哪一个为当前有效。

AUXR1 – 辅助寄存器1

7	6	5	4	3	2	1	0
SWRF	RSTPINF	T1LXTM	T0LXTM	GF2	UART0PX	0	DPS
读/写	读/写	读/写	读/写	读/写	读/写	R	读/写

地址: A2H 复位值: 详见 表 6–2. SFR定义及复位值

位	名称	描述
3	GF2	**通用标志位 2** 可通过软件置一或清除
1	0	**保留位** 该为始终为0
0	DPS	**数据指针选择** 0 = 数据指针0 (DPTR) 默认有效. 1 = 数据指针1 (DPTR1)有效. 当通过 DPS 切换当前有效DPTR后，前有效DPTR寄存器内的值保持不变。

27.2 96位序列号 (UID)

出厂前，每颗 N76E885都会预烧一个96位的序列号，用以确保该芯片的唯一性，这个唯一代码被称为序列号UID (Unique Code)。用户获得序列号的唯一方式是通过 IAP指令读取，详见章节 22.1 “IAP ”.

28. 片上调试(OCD)

28.1 功能描述

N76E885内嵌在片上调试功能（OCD），这为软件开发者提供了低成本调试方法，并且N76E885的每一种封装都适用。OCD具有完整的程序流程控制调试能力主要有8个硬件断点、单步运行、全速运行和非侵入命令的内存访问。OCD系统并不占用任何本地内存也不和片上外设共享资源。

当OCDEN (CONFIG0.4)配置为0，LOCK (CONFIG0.1)为1时，OCD才能有效。如果芯片已加密则OCD就不能工作。OCD系统使用两线串行接口，OCDDA 和 OCDCK，让目标设备和控制调试主机建立通讯。OCDDA是输入/输出引脚调试时作为数据传输口，OCDCK是输入口调试时作数据同步用。P1.2/$\overline{\text{RST}}$引脚也是必不可少的，它是用来控制OCD模式进入和退出的。N76E885的OCD和ICP功能是共用这3个引脚。

N76E885使用OCDDA、OCDCK、 和 P1.2/$\overline{\text{RST}}$引脚与OCD系统交互。在设计系统用到OCD时，必须考虑下面一些限制条件：

1. P1.2/$\overline{\text{RST}}$配置成外部复位引脚时，它不能直接连接到V_{DD}上并且要和所有外部复位设备断开

2. 如果 P1.2/$\overline{\text{RST}}$配置成输入引脚时，必须和外部输入源断开

3. 所有外部复位源必须断开

4.所有与OCDDA和OCDCK连接的外围器件必须断开

28.2 OCD限制条件

由于N76E885功能比较丰富而引脚比较有限，所以一个引脚上可能多个功能。使用OCD系统时肯定会牺牲一些功能，主要有以下一些限制条件：

1. OCD模式用到P1.2/$\overline{\text{RST}}$引脚，因此该引脚既不能作为输入也不能作为外部复位

2. OCDDA与P0.0共用一个引脚，因此该引脚I/O功能或是其他功能都不能使用

3. OCDCK与P0.1共用一个引脚，因此该引脚I/O功能或是其他功能都不能使用

4.当系统处在空闲或是掉电模式时，因为部分外设时钟已经停止所以任何访问可能会是无效的。读访问可能返回一个无用的数据，写访问可能不会成功。

5. 不能关闭HIRC，因为OCD需要这个时钟监视内部工作状态。在调试模式下，关闭HIRC的指令将不起作用，CPU进入掉电模式时HIRC会继续运行。

N76E885的OCD系统还有另一个限制就是正在运行用户程序时不能执行非侵入命令。非侵入命令可以用调试器访问MCU的存储单元、状态或是控制寄存器，一个读或是写控制寄存器必须在MCU停止的条件下进行，产生停止条件是在与硬件断点匹配后或是单步运行后。

CONFIG0

7	6	5	4	3	2	1	0
CBS	-	OCDPWM	OCDEN	-	RPD	LOCK	-
读/写	-	读/写	读/写	-	读/写	读/写	-

出厂默认值: 1111 1111b

位	名称	描述
5	OCDPWM	**片上调试OCD占用模式下PWM输出状态** 该为决定当偏上调试功能OCD占用CPU停止动作时，PWM输出的状态 1 = PWM 输出脚为三态模式（Tri-state） 0 = PWM 持续输出. 备注：该位仅当PWM管脚的PIO 位设置为1后有效 。
4	OCDEN	**OCD 使能** 1 = 禁用OCD 0 = 使能OCD

29. 配置字

N76E885具有硬件配置位，设定这些配置位可用于安全位，系统时钟位等等。这些硬件配置位可通过编程器/烧录器或ISP 来配置。N76E885具有四个配置位，配置位0～4。有些特定的配置位定义的功能也可以通过特定寄存器位重新配置。因此，需要加载这睦配置位到相应的寄存器位。这些加载发生在复位之后。软件复位会加载部分配置位至特殊功能寄存器相应的位，这些寄存器位也可以通过用户的软件控制及修改。其他复位将不改变这些寄存器位的值。

配置字中标注为“-“的位，在编程时请务必保持默认值，不要进行编写

CONFIG0

7	6	5	4	3	2	1	0
CBS	-	OCDPWM	OCDEN	-	RPD	LOCK	-
读/写	-	读/写	读/写	-	读/写	读/写	-

出厂默认值: 1111 1111b

位	名称	描述
7	CBS	**配置启动选项** 该位定义除软件复位外的所有复位后MCU从哪个区块启动。 1 = 除软件复位外的所有复位后MCU从APROM启动。 0 = 除软件复位外的所有复位后MCU从LDROM启动。
5	OCDPWM	**片上调试OCD占用模式下PWM输出状态** 该为决定当偏上调试功能OCD占用CPU停止动作时，PWM输出的状态 1 = PWM 输出脚为三态模式（Tri-state） 0 = PWM 持续输出. 备注：该位仅当PWM管脚的PIO 位设置为1后有效 。
4	OCDEN	**片上调试OCD 使能位** 1 = OCD 功能关闭. 0 = OCD 功能打开.
2	RPD	**复位脚禁能位** 1 = P1.2/$\overline{\text{RST}}$ 复位功能使能，管脚用作外部复位脚 0 = P1.2/$\overline{\text{RST}}$ 复位功能关闭，管脚用作输入脚 P1.2.
1	LOCK	**芯片加密使能位** 1 = 芯片不加密。 Flash存储器不加密，用户可通过硬件编程器/ ICP编程器读取FLASH的值. 0 = 芯片加密。全芯片FLASH区域加密，通过硬件编程器/ ICP编程器读取FALSH的值，读回全部位（FFH），对FLASH进行编程无效。 备注：配置字内容始终不加密，可以读出。当LOCK位配置为0对芯片加密后，配置字内容不能单独擦除或改写，解除芯片加密的唯一方式是执行全擦除动作(whole chip erase)，一但执行全擦除动作，FLASH内所有内容将被擦除且配置字内容也会被擦除。 芯片加密，不影响IAP功能。

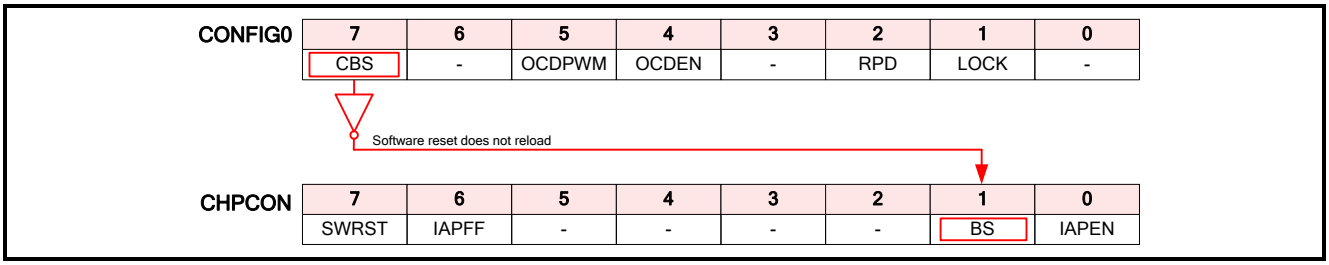


图 29-1. CONFIG0 复位后自动重载

CONFIG1

7	6	5	4	3	2	1	0
-	-	-	-	-	LDSIZE[2:0]		
-	-	-	-	-	读/写		

出厂默认值: 1111 1111b

位	名称	描述
2:0	LDSIZE[2:0]	**LDROM 区域选择** 该区域确定FLASH内LDROM区域大小 111 = 无 LDROM. APROM 为 18K 字节. 110 = LDROM 为 1K字节. APROM 为 17K 字节. 101 = LDROM 为 2K字节. APROM 为 16K 字节. 100 = LDROM 为 3K字节. APROM 为 15K 字节. 0xx = LDROM 为 4K字节. APROM 为 14K 字节.

CONFIG2

7	6	5	4	3	2	1	0
CBODEN	CBOV[2:0]			BOIAP	CBORST	-	-
读/写	读/写			读/写	读/写	-	-

出厂默认值: 1111 1111b

位	名称	描述
7	CBODEN	**配置欠压侦测使能位** 1 = 欠压侦测功能打开. 0 = 欠压侦测功能关闭
6:4	CBOV[1:0]	**配置欠压侦测电压选择位** 111 = V_{BOD} 生效电压 1.7V. 110 = V_{BOD} 生效电压2.0V. 101 = V_{BOD} 生效电压2.2V. 100 = V_{BOD} 生效电压 2.4V. 011 = V_{BOD} 生效电压 2.7V. 010 = V_{BOD} 生效电压 3.0V. 001 = V_{BOD} 生效电压 3.7V. 000 = V_{BOD} 生效电压 4.3V.

位	名称	描述
3	BOIAP	**欠压禁止IAP位** 该位决定当系统电压低于欠压侦测设定值时，IAP擦除及编程功能是否禁止。该位仅当欠压侦测功能使能后有效。 1 = 当V_{DD} 低于V_{BOD}.设定值时，IAP 擦除或编程功能禁止 0 = V_{DD} 任何电压状态下，IAP擦除及编程功能都可执行
2	CBORST	**CONFIG 欠压复位使能位** 该为决定当侦测到电压低于V_{BOD} 时，芯片是否复位 1 = 欠压复位功能使能 0 = 欠压复位功能关闭

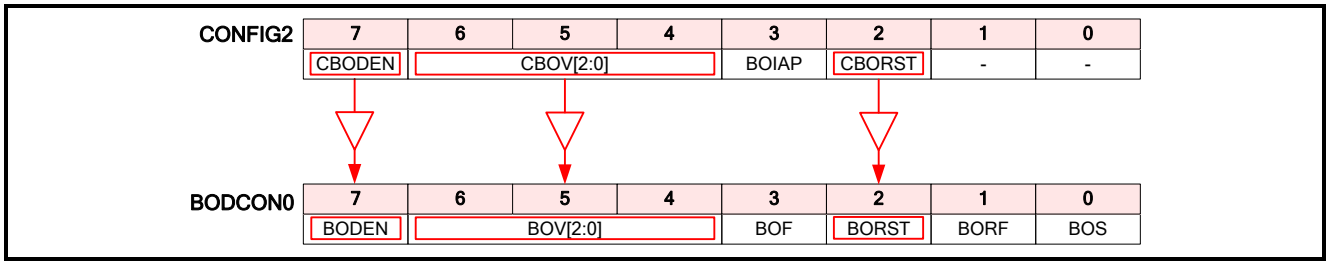


图 29-2. CONFIG2 上电复位重载位

CONFIG4

7	6	5	4	3	2	1	0
WDTEN[3:0]				-	-	-	-
读/写				-	-	-	-

出厂默认值: 1111 1111b

位	名称	描述
7:4	WDTEN[3:0]	**看门狗定时器WDT使能** 该为决定复位后看门狗定时器功能 1111 = 看门狗定时器关闭。WDT可用作普通定时器用。 0101 = WDT使能并具有定时复位功能，在空闲模式和掉电模式下不工作。 Others = WDT使能并具有定时复位芯片功能，在空闲和掉电模式下，定时器持续工作
3:0	-	**保留位**

30. 在电路编程(ICP)

ICP (在电路编程) 模式是另一种烧写存储器FLASH的方式。通常硬件编程模式采用gang-writer批量编程器，用于量产时节省时间及硬件，但在实验开发工程阶段，或者是系统板已经到终端客户手上时，需要拆卸芯片，采用硬件编程模式是不适用的。采用ICP编程模式就可以不需要解焊芯片。同时量产也适用于先将未烧写编程过的芯片打件在系统板上进行量产或在已量产完成的系统上更新程序使用。

仅需要3个引脚执行ICP功能，, $\overline{\text{RST}}$, ICPDA, 以及 ICPCK。$\overline{\text{RST}}$ 用于进入或退出ICP模式，ICPDA为数据输入输出脚。ICPCK为编程时钟输入脚。用户需要在系统板上预留VDD、GND以及这三个脚。

Nuvoton 提供N76E885 ICP 工具，用户可直接借由ICP编程器实现系统版上ICP 功能。如此高效稳定的编程方式具体内容请参考, Nuvoton 8位微处理器网页: Nuvoton 80C51 单片机技术支持.

31. 指令集

N76E885 执行所有标准8051的指令集与MCS-51全兼容，每条指令执行周期根据1T 8051内核标准执行。该架构消除了冗余总线状态，且实现了取，译码和运行时间并行同时执行。N76E885每个机器周期占用一个时钟周期，对比传统12T 80C81系统，同样的系统时钟频率，提升性能至8.1倍（基于MIPS）。当然实际的速率还是需要基于指令编译的结果。

所有指令代码为 8-位 OPCODE。单字节来源于程序存储器。OPCODE 通过CPU译码。决定系统的工作情况以及从存储器中运作数据。如果没有其它数据请求，为单字节指令。在一些应用中，需要更多的数据，使用2或3字节指令。

表 31–1 列出所有指令， 指令集及寻址模式注释如下。

Rn (n = 0~7)	寄存器 R0~R7 为当前选择寄存器区域。
	8-位 内部数据地址。可作为内部 RAM 地址 (0~127) 或 SFR (I/O, 控制寄存器, 状态寄存器等(128~255)。
Rn (n = 0~7)	寄存器 R0~R7 为当前选择寄存器区域
direct	8-位 内部数据地址。可作为内部 RAM 地址 (00H to 7FH) 或 SFR (80H to FFH)。
@Ri (i = 0, 1)	通过R0或R1间接寻址8-位内部RAM 区域 (0~255)。
#data	指令包括8-位常量。
#data16	指令包括16-位常量。
Addr16	16-位目的地址。使用LCALL和LJMP。分支可达16k字节程序空间任何位置。
Addr11	11-位目的地址。使用ACALL和AJMP。分支达2k字节程序内存。
Rel	带符号(2's 互补) 8-位偏移字节。使用SJMP和条件分支。范围为-128到+127字节。
Bit	对内部数据RAM或寄存器直接寻址。

表 31–1. 指令集

指令	OPCODE	字节	时钟周期	N76E885对比传统80C51速率
NOP	00	1	1	12
ADD A, Rn	28~2F	1	2	6
ADD A, direct	25	2	3	4
ADD A, @Ri	26, 27	1	4	3
ADD A, #data	24	2	2	6
ADDC A, Rn	38~3F	1	2	6
ADDC A, direct	35	2	3	4
ADDC A, @Ri	36, 37	1	4	3
ADDC A, #data	34	2	2	6
SUBB A, Rn	98~9F	1	2	6
SUBB A, direct	95	2	3	4
SUBB A, @Ri	96, 97	1	4	3
SUBB A, #data	94	2	2	6
INC A	04	1	1	12
INC Rn	08~0F	1	3	4
INC direct	05	2	4	3
INC @Ri	06, 07	1	5	2.4
INC DPTR	A3	1	1	24
DEC A	14	1	1	12
DEC Rn	18~1F	1	3	4
DEC direct	15	2	4	3
DEC @Ri	16, 17	1	5	2.4
MUL AB	A4	1	4	12
DIV AB	84	1	4	12
DA A	D4	1	1	12
ANL A, Rn	58~5F	1	2	6
ANL A, direct	55	2	3	4
ANL A, @Ri	56, 57	1	4	3
ANL A, #data	54	2	2	6
ANL direct, A	52	2	4	3
ANL direct, #data	53	3	4	6
ORL A, Rn	48~4F	1	2	6
ORL A, direct	45	2	3	4
ORL A, @Ri	46, 47	1	4	3
ORL A, #data	44	2	2	6
ORL direct, A	42	2	4	3
ORL direct, #data	43	3	4	6
XRL A, Rn	68~6F	1	2	6
XRL A, direct	65	2	3	4
XRL A, @Ri	66, 67	1	4	3
XRL A, #data	64	2	2	6

表 31–1. 指令集

指令		OPCODE	字节	时钟周期	N76E885对比传统80C51速率
XRL	direct, A	62	2	4	3
XRL	direct, #data	63	3	4	6
CLR	A	E4	1	1	12
CPL	A	F4	1	1	12
RL	A	23	1	1	12
RLC	A	33	1	1	12
RR	A	03	1	1	12
RRC	A	13	1	1	12
SWAP	A	C4	1	1	12
MOV	A, Rn	E8~EF	1	1	12
MOV	A, direct	E5	2	3	4
MOV	A, @Ri	E6, E7	1	4	3
MOV	A, #data	74	2	2	6
MOV	Rn, A	F8~FF	1	1	12
MOV	Rn, direct	A8~AF	2	4	6
MOV	Rn, #data	78~7F	2	2	6
MOV	direct, A	F5	2	2	6
MOV	direct, Rn	88~8F	2	3	8
MOV	direct, direct	85	3	4	6
MOV	direct, @Ri	86, 87	2	5	4.8
MOV	direct, #data	75	3	3	8
MOV	@Ri, A	F6, F7	1	3	4
MOV	@Ri, direct	A6, A7	2	4	6
MOV	@Ri, #data	76, 77	2	3	6
MOV	DPTR, #data16	90	3	3	8
MOVC	A, @A+DPTR	93	1	4	6
MOVC	A, @A+PC	83	1	4	6
MOVX	A, @Ri[1]	E2, E3	1	5	4.8
MOVX	A, @DPTR[1]	E0	1	4	6
MOVX	@Ri, A[1]	F2, F3	1	6	4
MOVX	@DPTR, A[1]	F0	1	5	4.8
PUSH	direct	C0	2	4	6
POP	direct	D0	2	3	8
XCH	A, Rn	C8~CF	1	2	6
XCH	A, direct	C5	2	3	4
XCH	A, @Ri	C6, C7	1	4	3
XCHD	A, @Ri	D6, D7	1	5	2.4
CLR	C	C3	1	1	12
CLR	bit	C2	2	4	3
SETB	C	D3	1	1	12
SETB	bit	D2	2	4	3

表 31–1. 指令集

指令		OPCODE	字节	时钟周期	N76E885对比传统80C51速率
CPL	C	B3	1	1	12
CPL	bit	B2	2	4	3
ANL	C, bit	82	2	3	8
ANL	C, /bit	B0	2	3	8
ORL	C, bit	72	2	3	8
ORL	C, /bit	A0	2	3	8
MOV	C, bit	A2	2	3	4
MOV	bit, C	92	2	4	6
ACALL	addr11	11, 31, 51, 71, 91, B1, D1, F1[2]	2	4	6
LCALL	addr16	12	3	4	6
RET		22	1	5	4.8
RETI		32	1	5	4.8
AJMP	addr11	01, 21, 41, 61, 81, A1, C1, E1[3]	2	3	8
LJMP	addr16	02	3	4	6
SJMP	rel	80	2	3	8
JMP	@A+DPTR	73	1	3	8
JZ	rel	60	2	3	8
JNZ	rel	70	2	3	8
JC	rel	40	2	3	8
JNC	rel	50	2	3	8
JB	bit, rel	20	3	5	4.8
JNB	bit, rel	30	3	5	4.8
JBC	bit, rel	10	3	5	4.8
CJNE	A, direct, rel	B5	3	5	4.8
CJNE	A, #data, rel	B4	3	4	6
CJNE	Rn, #data, rel	B8~BF	3	4	6
CJNE	@Ri, #data, rel	B6, B7	3	6	4
DJNZ	Rn, rel	D8~DF	2	4	6
DJNZ	direct, rel	D5	3	5	4.8

[1] N76E885没有外部存储器总线结构。MOVX指令仅用于读写内部XRAM用
[2] 11位地址[A10:A8]的最高三位决定ACALL hex 码。代码为[A10,A9,A8,1,0,0,0,1]。
[3] 11位地址[A10:A8]的最高三位决定AJMP hex 码。代码为 [A10,A9,A8,0,0,0,0,1]。

32. 电气特性

32.1 绝对最大额定值

参数	值	单位
工作温度(T_A)	-40 to +105	°C
储存温度	-55 to +150	°C
VDD 管脚与GND 管脚之间压差	-0.3 to +6.3	V
其余管脚至 GND 管脚之间压差	-0.3 to (V_{DD}+0.3)	V

用户于使用中达到或超过“绝对最大额定值”中所列值有可能会造成芯片永久性损坏。本规格仅采用芯片在该条件内测试完成，如超过本规范部分不做保证。芯片长期处于绝对最大额定值条件下，可能会影响器件的可靠性。

32.2 DC电气特性

表 32–1. DC电气特性表

符号	参数	条件	最小值	典型值	最大值	单位
电源电压						
V_{DD}	工作电压	F = 0 to 25 MHz	2.4	-	5.5	V
I/O						
V_{IL}	输入低电压 (I/O 为 TTL 输入模式)		V_{ss}-0.3	-	0.2V_{DD}-0.1	V
V_{IL1}	输入低电压 (I/O为 施密特触发输入模式， $\overline{RST}$, 及 X_{IN})		V_{ss}-0.3	-	0.3V_{DD}	V
V_{IH}	输入高电压 (I/O 为 TTL 输入模式)		0.2V_{DD}+0.9	-	V_{DD}+0.3	V
V_{IH1}	输入高电压 (I/O为 施密特触发输入模式 及 X_{IN})		0.7V_{DD}	-	V_{DD}+0.3	V
V_{IH2}	输入高电压 ($\overline{RST}$)		0.8V_{DD}	-	V_{DD}+0.3	V
V_{OL}	输出低电压[1] (正常灌电流强度，除输入模式外所有模式)	V_{DD} = 4.5V, I_{OL} = 16mA	-	-	0.4	V
		V_{DD} = 3.0V, I_{OL} = 13mA	-	-	0.4	
		V_{DD} = 2.4V, I_{OL} = 7mA	-	-	0.4	
V_{OL1}	输出低电压[1] (P0.1~P0.3, P2.0~P2.1带有强灌电流功能脚，除输入模式外所有模式)	V_{DD} = 4.5V, I_{OL} = 32mA	-	-	0.4	V
		V_{DD} = 3.0V, I_{OL} = 24mA	-	-	0.4	
		V_{DD} = 2.4V, I_{OL} = 11mA	-	-	0.4	

符号	参数	条件	最小值	典型值	最大值	单位
V_{OH}	输出高电压 (准双向模式)	V_{DD} = 4.5V, I_{OH} = -380μA	2.4	-	-	V
		V_{DD} = 3.0V, I_{OH} = -100μA	2.4	-	-	
		V_{DD} = 2.4V, I_{OH} = -40μA	2.0	-	-	
V_{OH1}	输出高电压 (强推挽模式)	V_{DD} = 4.5V, I_{OH} = -16mA	2.4	-	-	V
		V_{DD} = 3.0V, I_{OH} = -4.5 mA	2.4	-	-	
		V_{DD} = 2.4V, I_{OH} = -2mA	2.0	-	-	
I_{IL}	逻辑0输入电流 (准双向模式)	V_{DD} = 5.5V, V_{IN} = 0.4V	-	-	-50	μA
I_{TL}	逻辑1至0转换电流[2] (准双向模式)	V_{DD} = 5.5V	--	-	-650	μA
I_{LI}	输入漏电流 (开漏模式或输入模式)		-	1	±10	μA
R_{RST}	$\overline{RST}$ 管脚内部下拉电阻		50	-	600	kΩ
工作电流						
I_{DD}	正常工作电流[3]	HXT, XTGS[1:0] = [1,1]	-	0.12F+0.7	0.13F+0.9	mA
		HIRC	-	3.9	4.3	mA
		LXT, XTGS[1:0] = [0,1]	-	220	280	μA
		LIRC	-	190	250	μA
I_{IDL}	空闲模式工作电流	HXT, XTGS[1:0] = [1,1]	-	0.07F+0.5	0.07F+0.7	mA
		HIRC		2.6	2.8	mA
		LXT, XTGS[1:0] = [0,1]	-	180	250	μA
		LIRC	-	170	240	μA
I_{PD}	掉电模式工作电流 (BOD 关闭, LXT 关闭)	T_A = 25℃	-	1.3	2.5	μA
		T_A = -40℃ to +105℃	-	-	20	μA
I_{PD1}	掉电模式工作电流(BOD 关闭, LXT 打开, XTGS[1:0] = [0,1])	T_A = 25℃	-	2.3	4.0	μA
		T_A = -40℃ to +105℃	-	-	23	μA

[1] 在稳态(非瞬态)条件下, I_{OL} 必须受到如下限制,
每个管脚最大I_{OL}值: 40mA
所有管脚输出最大值 I_{OL} : 120mA

[2] 所有管脚准双向模式，当外部输入由1至0转换的转换电流，在V_{IN} 接近2V时达到最大值

[3] 测量条件，MCU运行死循环指令“SJMP $” 所有管脚配置位准双向模式。

32.3 AC电气特性

表 32–2. 系统时钟AC电气特性

符号	参数	最小值	典型值	最大值	单位
1/ t_{CLCL}	外部时钟输入频率 (ECLK)	0	-	25	MHz
	高速晶振/振荡器 输入频率(HXT)	2	-	25	
	低速晶振/振荡器 输入频率(LXT)	-	32.768	-	kHz
t_{CHCX}	外部时钟输入高电平保持时间	30	-	-	ns
t_{CLCX}	外部时钟输入低电平保持时间	30	-	-	ns
t_{CLCH}	外部时钟输入上升时间	-	-	10	ns
t_{CHCL}	外部时钟输入下降时间	-	-	10	ns

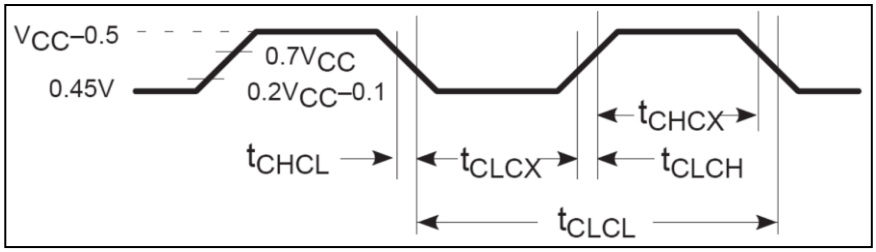


图 32-1. 外部时钟输入时序图

表 32–3. 管脚斜率控制AC电气特性

PxSR.n 位值	符号	参数	条件	最小值	典型值	最大值	单位
0	F_{OUT}	最大输出频率 [1]	V_{DD} = 5.0V, C_L = 30pF	-	34	-	MHz
			V_{DD} = 3.3V, C_L = 30pF	-	22.5	-	
			V_{DD} = 2.4V, C_L = 30pF	-	12.8	-	
	T_R	输出由低至高上升时间	V_{DD} = 5.0V, C_L = 30pF	-	7.4	-	ns
			V_{DD} = 3.3V, C_L = 30pF	-	11	-	
			V_{DD} = 2.4V, C_L = 30pF	-	18	-	
	T_F	输出由高至低下降时间	V_{DD} = 5.0V, C_L = 30pF	-	7.2	-	ns
			V_{DD} = 3.3V, C_L = 30pF	-	11.2	-	
			V_{DD} = 2.4V, C_L = 30pF	-	21	-	
1	F_{OUT}	最大输出频率[1]	V_{DD} = 5.0V, C_L = 30pF	-	39	-	MHz
			V_{DD} = 3.3V, C_L = 30pF	-	27.5	-	
			V_{DD} = 2.4V, C_L = 30pF	-	17	-	
	T_R	输出由高至低下降时间	V_{DD} = 5.0V, C_L = 30pF	-	7	-	ns

PxSR.n 位值	符号	参数	条件	最小值	典型值	最大值	单位
			V_{DD} = 3.3V, C_L = 30pF	-	10	-	
			V_{DD} = 2.4V, C_L = 30pF	-	16	-	
	T_F	输出由高至低下降时间	V_{DD} = 5.0V, C_L = 30pF	-	4.8	-	ns
			V_{DD} = 3.3V, C_L = 30pF	-	7	-	
			V_{DD} = 2.4V, C_L = 30pF	-	11.8	-	

[1] 已达到最大输出频率($(T_R + T_F) \le 1/2$) T 且占空比为 45% 至 55%. 见下图.

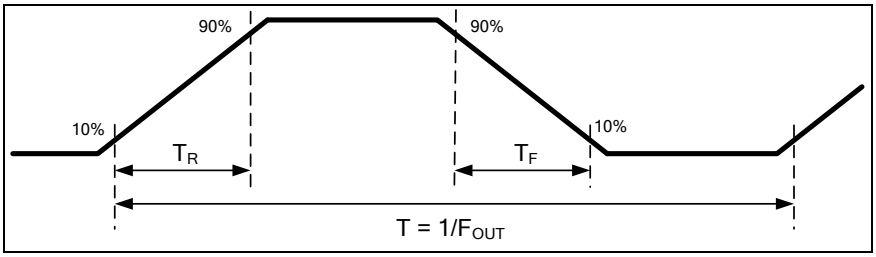


图 32-2. 管脚 AC特性定义图

表 32–4. 内部晶振AC电气特性

符号	参数	条件	频率误差	最小值	典型值	最大值	单位
F_{HIRC}	内部高速22.118 MHz 振荡频率(HIRC)	T_A = -10℃ to +70℃	1%	21.897	22.118	22.340	MHz
		T_A = -40℃ to +105℃	2%	21.676		22.560	
F_{LIRC}	内部低速 10 kHz 振荡频率(LIRC)		35%	6.5	10	13.5	kHz

表 32–5. 掉电唤醒电气特性

符号	参数	条件	最小值	典型值	最大值	单位
T_{PDWK}	掉电模式唤醒时间	HIRC	-	60	-	μs
		HXT, F = 25MHz	-	500	-	

表 32–6. 外部复位脚电气特性

符号	参数	条件	最小值	典型值	最大值	单位
T_{RST}	$\overline{RST}$ 复位脚侦测有效脉宽		-	24/F_{SYS}	450	μs

32.4 模拟电路电气特性

表 32–7. POR 电气特性

符号	参数	条件	最小值	典型值	最大值	单位

符号	参数	条件	最小值	典型值	最大值	单位
V_{POR}	上电复位电压		1.3	1.4	1.5	V
T_{PORRD}	上电复位释放时间		-	4	-	ms

表 32–8. BOD 电气特性

符号	参数	条件	最小值	典型值	最大值	单位
V_{BOD0}	欠压阀值4.3V	BOV[2:0] = [0,0,0]	4.15	4.3	4.45	V
V_{BOD1}	欠压阀值3.7V	BOV[2:0] = [0,0,1]	3.55	3.7	3.85	V
V_{BOD2}	欠压阀值3.0V	BOV[2:0] = [0,1,0]	2.85	3.0	3.15	V
V_{BOD3}	欠压阀值2.7V	BOV[2:0] = [0,1,1]	2.6	2.7	2.8	V
V_{BOD4}	欠压阀值2.4V	BOV[2:0] = [1,0,0]	2.3	2.4	2.5	V
V_{BOD5}	欠压阀值2.2V	BOV[2:0] = [1,0,1]	2.1	2.2	2.3	V
V_{BOD6}	欠压阀值2.0V	BOV[2:0] = [1,1,0]	1.9	2.0	2.1	V
V_{BOD7}	欠压阀值 1.7V	BOV[2:0] = [1,1,1]	1.6	1.7	1.8	V
V_{BODHYS}	欠压迟滞	LPBOD[1:0] = [0,0] only	50	65	80	mV
I_{BOD}	欠压静态电流	V_{DD} = 5V, LPBOD[1:0] = [0,0]	-	55	70	μA
		V_{DD} = 5V, LPBOD[1:0] = [0,1]	-	14	16	
		V_{DD} = 5V, LPBOD[1:0] = [1,0]	-	4	6	
		V_{DD} = 5V, LPBOD[1:0] = [1,1]	-	1.5	2.5	
T_{BOD}	欠压侦测脉宽		详见 表 25–2			-
T_{BODEN}	欠压检测使能时间		2	-	3	$1/F_{LIRC}$

表 32–9. 带隙电压(Band-gap)电气特性

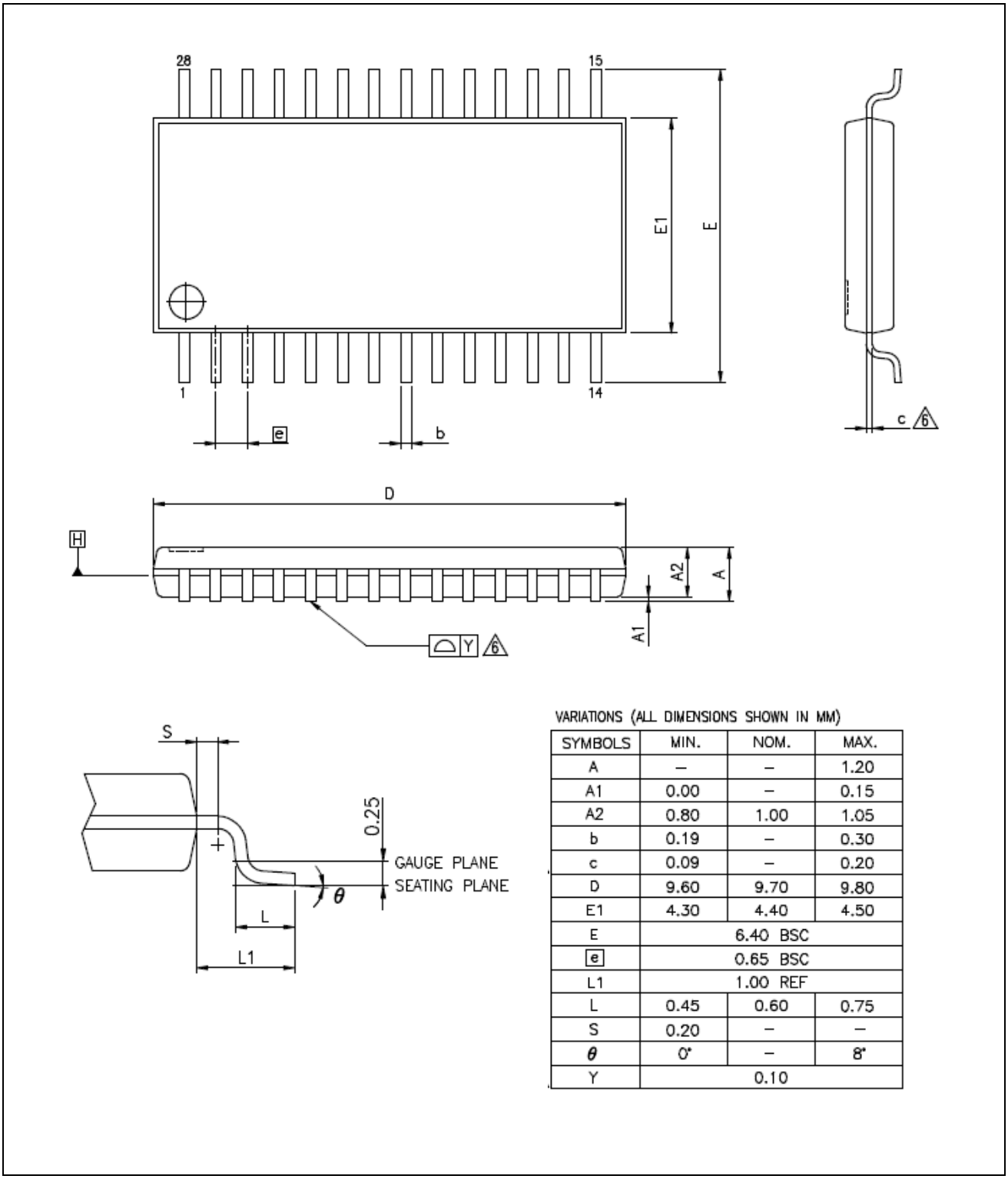
符号	参数	条件	最小值	典型值	最大值	单位
V_{BG}	带隙电压		1.16	1.22	1.28	V
T_{BGEN}	带隙电压使能时间		1	-	2	$1/F_{LIRC}$

表 32–10. ADC电气特性

符号	参数	条件	最小值	典型值	最大值	单位
V_{AVDD}	ADC 工作电压		2.4	-	5.5	V
I_{AVDD}	ADC 工作电流	V_{AVDD} = 5V, VREFSEL = 0	-	160	220	μA
V_{VREF}	模拟信号参考电压		1.8	-	V_{AVDD}	V

符号	参数	条件	最小值	典型值	最大值	单位
V_{AIN}	模拟信号输入电压		0	-	V_{VREF}	V
N_R	分辨率			10		bit
DNL	微分非线性误差		-	+1.5	+2	LSB
INL	积分非线性误差		-	±1	±2	LSB
OE	偏移误差		-	+2	+3	LSB
FE	满量程误差		-	+1.5	+2.5	LSB
TUE	总不可调整误差		-	+3.5	+4	LSB
-	一致性			保证		-
F_{ADC}	ADC 时钟频率	V_{VREF} = 3.0V to 5.5V	0.01	-	6	MHz
		V_{VREF} = 2.4V to 5.5V	0.01	-	3	
T_S	采样时间（软件可调）		6	-	261	$1/F_{ADC}$
T_{CONV}	总转换时间			T_S + 12		$1/F_{ADC}$
T_{ADCEN}	ADC 使能时间			32		$1/F_{ADC}$
R_{IN}	ADC输入等效电阻		-	-	7	kΩ
C_{IN}	ADC 输入等效电容			10	12	pF

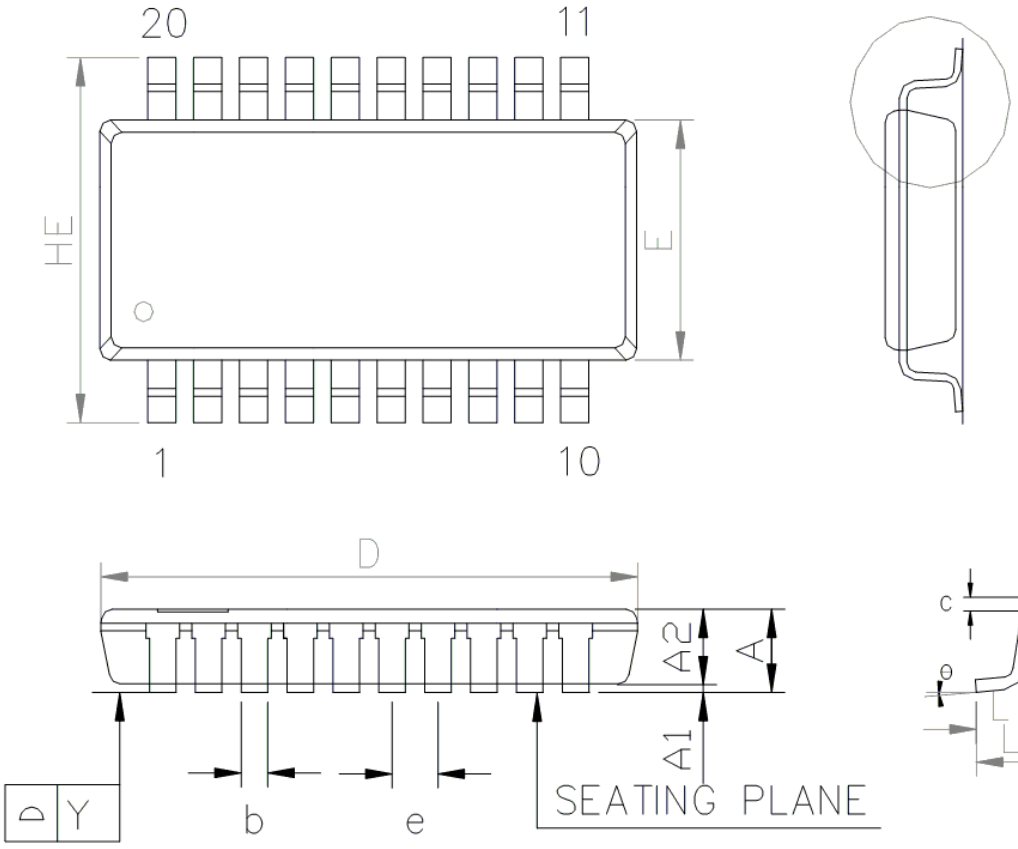
33. 封装信息



VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

SYMBOLS	MIN.	NOM.	MAX.
A	–	–	1.20
A1	0.00	–	0.15
A2	0.80	1.00	1.05
b	0.19	–	0.30
c	0.09	–	0.20
D	9.60	9.70	9.80
E1	4.30	4.40	4.50
E	6.40 BSC		
e	0.65 BSC		
L1	1.00 REF		
L	0.45	0.60	0.75
S	0.20	–	–
θ	0°	–	8°
Y	0.10		

图 33-1. TSSOP-28 封装信息



SYMBOL	DIMENSION (MM)			DIMENSION (INCH)		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	-	-	1,20	-	-	0.047
A1	0,05	-	0,15	0.002	-	0.006
A2	0.80	0.90	1,05	0.031	0.035	0.041
E	4.30	4.40	4.50	0.169	0.173	0.177
HE	6.40 BSC			0.252 BSC		
D	6.40	6.50	6.60	0.252	0.256	0.260
L	0.50	0.60	0.75	0.020	0.024	0.030
L1	1.00 REF			0.039 REF		
b	0.19	-	0.30	0.007	-	0.012
e	0.65 BSC			0.026 BSC		
c	0.09	-	0.20	0.004	-	0.008
θ	0°	-	8°	0°	-	8°
Y	0.10 BASIC			0.004 BASIC		

图 33-2. TSSOP-20 封装信息

34. 版本信息

版本	更新日期	描述
1.00_SC1	2015/9/15	初次发布
1.01	2015/12/7	修正ECLK 1.8v 章节1. 修正FLASH数据保存年限 章节6.去除AUXR寄存器描述 章节19.修正增加使用band gap需使能BODEN位 章节32.VIL 及 VIL1 最小值修正 章节33.修正TSSOP28封装Y数值

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure Usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.