Nuvoton 1T 8051 内核微控制器

N76E616

规格书

目　录

1. 概述

N76E616是一个内嵌FLASH、8位高性能1T 基于8051核的微控制器。指令集完全兼容标准和增强型的的80C51。

N76E616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N76E616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。

N76E616 提供丰富的外设包括256字节的SRAM和256字节的外部RAM(XRAM)，最大到46个IO口，两个16位定时器/计数器 0/1，一个16位定时器2带4个自动重载定时器或PWM定时器。一个看门狗定时器，一个自我唤醒定时器(WKT)，一个16位自动重载定时器3用于通用目的或波特率发生器。两个UART带有帧错误检测和自动地址识别，一个I^2C，8通道共享引脚中断用于所有I/O，一个10位的ADC，和最大到180段的LCD驱动。外设配备17个中断源带4级中断优先级。

N76E616配备5个时钟源和支持通过软件切换，四个时钟源包括2 MHz 到 16 MHz外部高速晶振，32.768 kHz外部低速晶振，外部时钟输入，10 kHz内部震荡器，和一个11.059 MHz内部精确振荡器，它在常温下达到±1%的精度。N76E616提供额外电源监测，例如上电复位和4级欠压检测，有稳定的电源通断性能用于高可靠的系统设计。

N76E616微控制器操作功耗非常低，有两种经济的电源模式用来减少电源的消耗---空闲模式和掉电模式。软件可以选择空闲模式关闭CPU时钟，但是允许外设继续工作。掉电模式停止整个系统时钟用于最小的功耗。N76E616的系统时钟可以通过时钟分频器放慢，实现在性能和功耗之间的平衡。

带有高性能的CPU核和丰富的外设，N76E616可以通用的需求，家庭应用或马达控制等。

2. 特性

- CPU:
  - 全静态设计8位 高性能 1T 基于8051 CMOS 微控制器.
  - 指令与 MCS-51完全兼容
  - 4-级中断优先级
  - 双数据指针 (DPTRs).
- 工作:
  - 宽电压范围 2.4V 至 5.5V.
  - 宽电压工作频率，最大至 16MHz.
  - 工业级温度等级 -40℃ 至 +105℃.
- 内存:
  - 最大到 18K 字节 APROM 用于用户代码
  - 可配置 4K/3K/2K/1K 字节 LDROM, 用于启动ISP代码
  - FLASH 内存每256字节为一页
  - 内嵌在应用编程 (IAP).
  - FLASH 内存 20,000次擦写寿命.
  - 代码安全加密.
  - 256 字节片上 RAM.
  - 额外 256 字节片上辅助 RAM (XRAM) ，通过 MOVX 指令访问.
  - UID (唯一码).
- 时钟源
  - 11.059 MHz 高速内部振荡器常温下精度达 ±1%,，全温全压范围精度 ±5%
  - 10 kHz 低速内部振荡器
  - 2 MHz 到 16 MHz 高速外部晶振.
  - 32.768 kHz 低速外部晶振.
  - 外部时钟输入.
  - 通过软件切换时钟源.
  - 可编程系统时钟分频，最大到1/512.

- 外设:
  - 最大 45 个通用 I/O 引脚和一个仅输入引脚，四个引脚提供正常电流或大电流驱动选择
  - 标准中断引脚 $\overline{INT0}$ 和 $\overline{INT1}$.
  - 两个 16位 定时器/计数器 0 和 1 ，兼容标准的 8051
  - 一个 16位 定时器 2 带四组定时器 2A/2B/2C/2D, 支持自动重载模式和PWM模式
  - 一个 16位 自动重载 定时器 3, 它可以是UART波特率时钟源
  - 一个可编程看门狗定时器 (WDT) ，时钟源为内部10 kHz，可以作为超时复位定时器或通用定时器
  - 一个专门的自我唤醒定时器用于省电模式的自我定时唤醒
  - 两个全双工 UART ，支持帧错误检测和自动地址识别
  - 一个 I2C总线，带主机从机模式，最大到 400kbps 数据率
  - 八通道引脚中断，与所有I/O端口共享。边沿/电平侦测的可变配置
  - 一个10位ADC，最大到300ksps采样率，带硬件转换结果比较
  - 4COM / 32SEG 或 6COM / 30SEG 配置的LCD 驱动
- 电源管理:
  - 两种省电模式：空闲模式和掉电模式
- 电源监测:
  - 欠压侦测 (BOD) ， 4-级电压选择，中断或复位可选
  - 上电复位 (POR)
- 强 抗ESD 和 EFT 性能
- 开发工具:
  - 新唐Nu-Link 片上调试 (OCD) ，用 KEIL™ 开发环境
  - 新唐Nu-Link 在电路编程 (ICP)
  - 新唐在系统编程(ISP)， 通过 UART 升级
- 部分型号和封装:

部分型号	APROM	LDROM	封装
N76E616AL48	18K 字节与 LDROM共享	最大到 4K 字节	LQFP48
N76E616AF44	18K 字节与 LDROM共享	最大到 4K 字节	PQFP44
N76E616AM44	18K 字节与 LDROM共享	最大到 4K 字节	LQFP44

3. 结构框图

图 3.1 所示 N76E616功能框图，用户可以在框图中找到设备的所有外设功能。

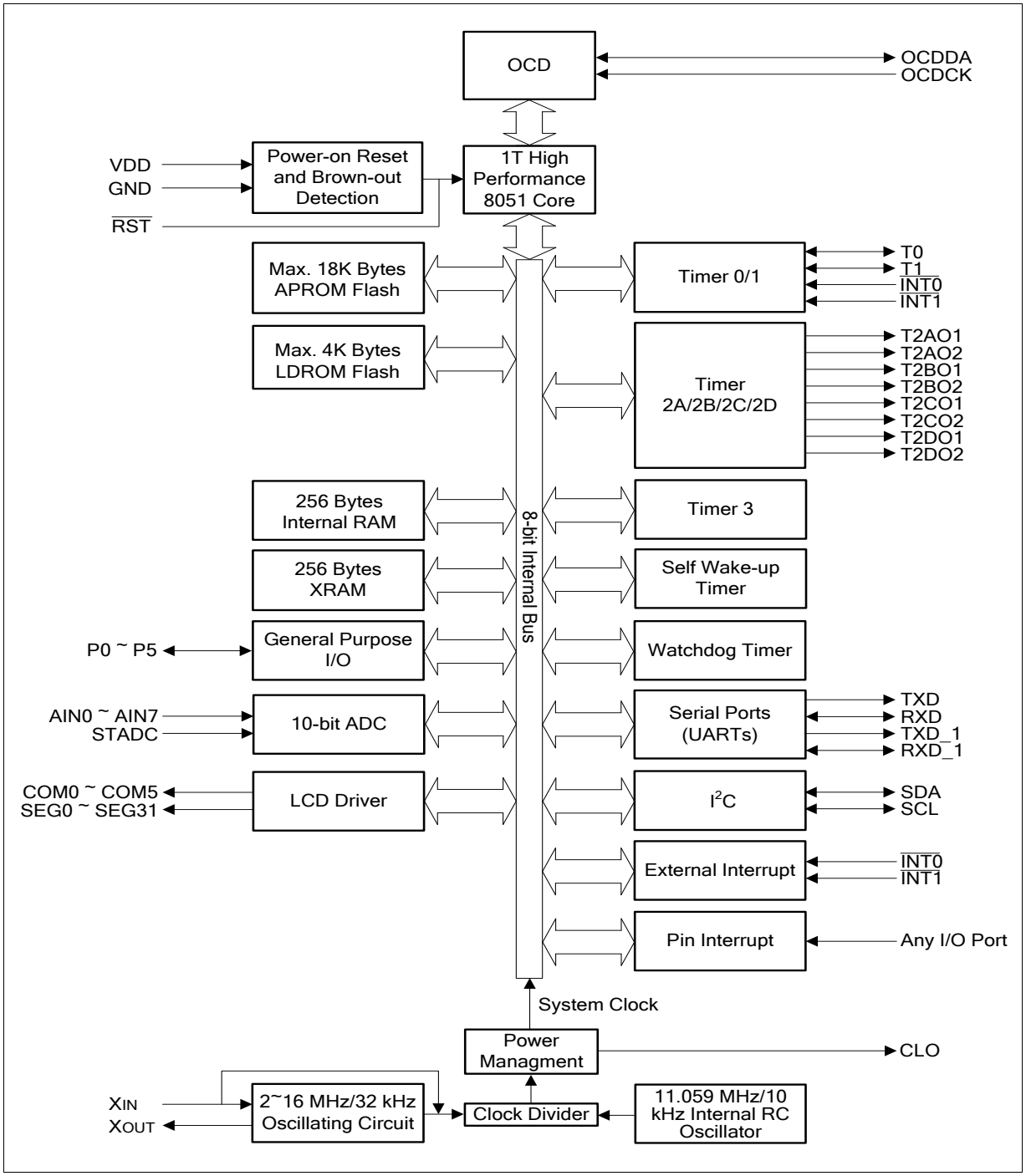


图 3.1. 结构框图

4. 管脚分布图

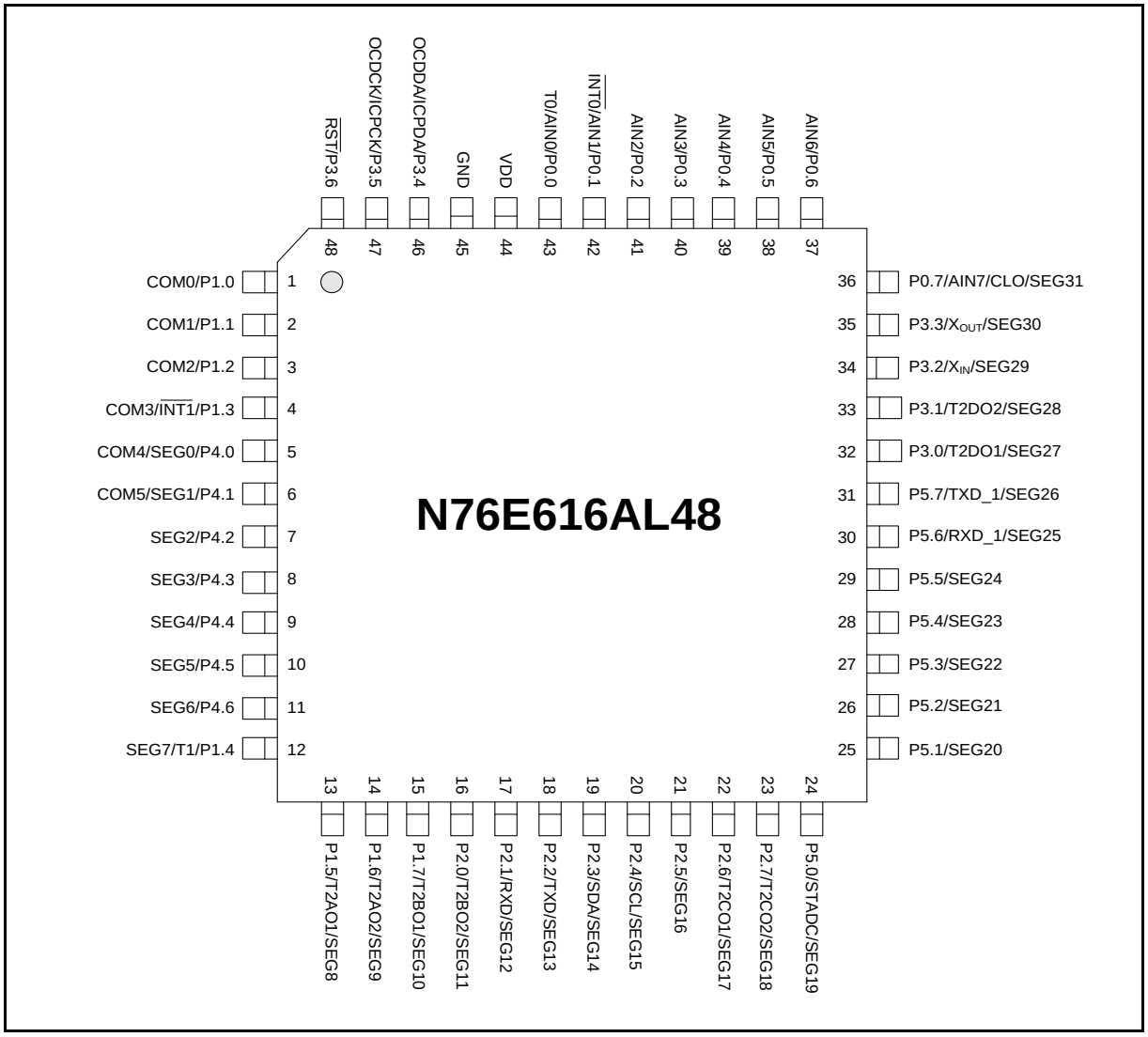


图 4-1 LQFP48 封装管脚分布图

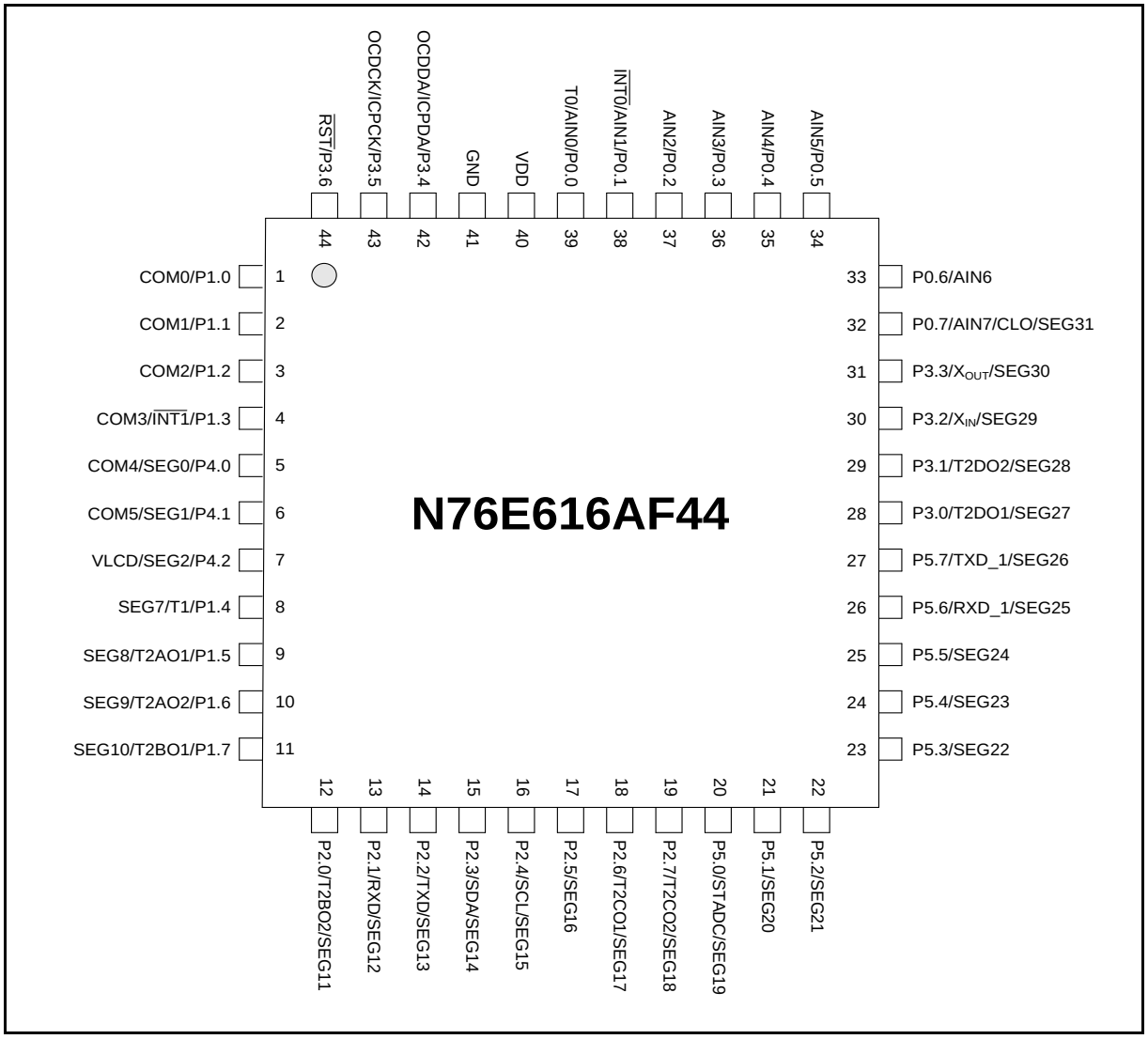


图 4-2 PQFP44 封装管脚分布图

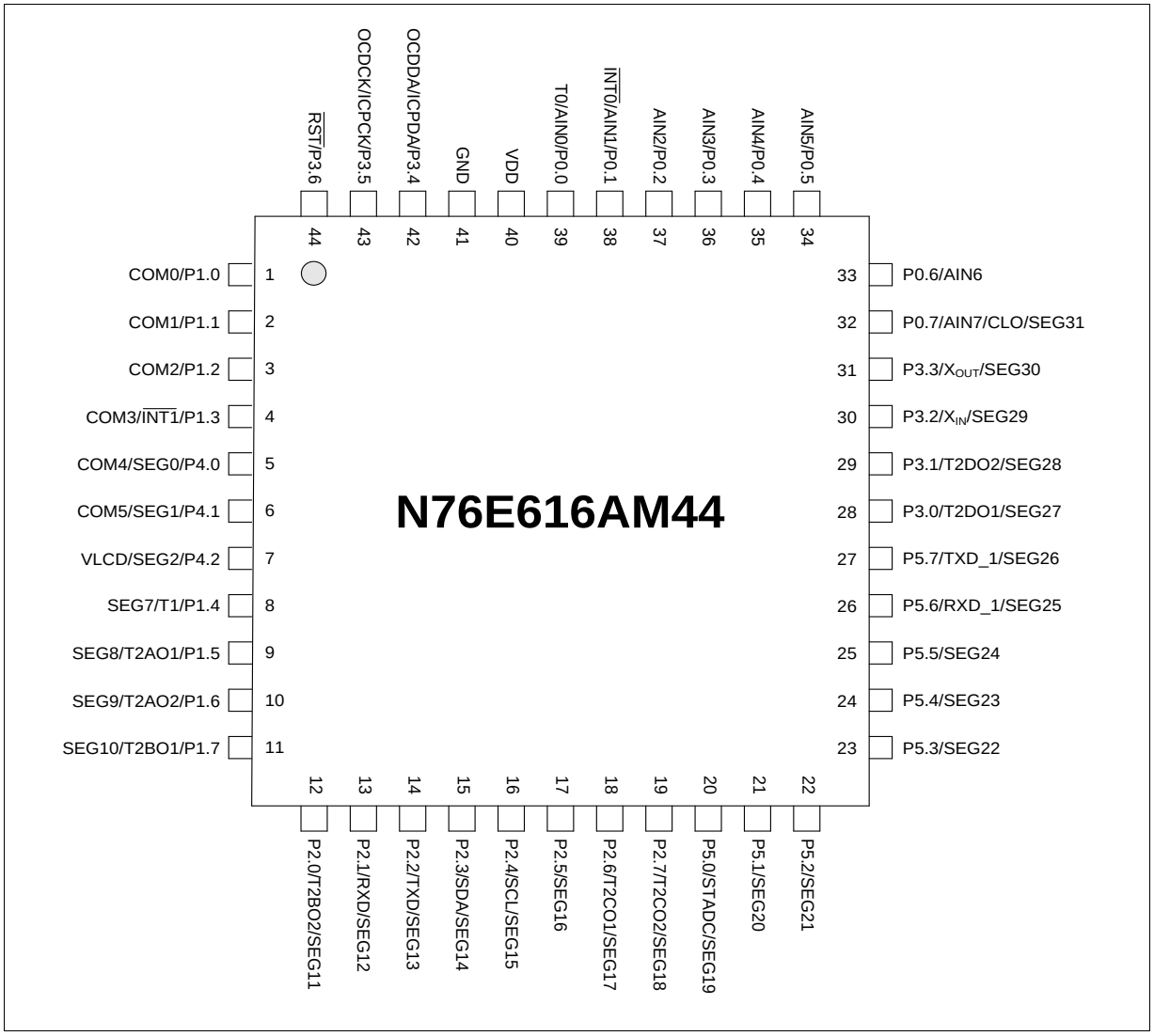


图 4-3 LQFP44 封装管脚分布图

表 4-1 管脚描述

管脚号码		符号	复合功能描述[1]
PQFP44 LQFP44	LQFP48		
40	44	VDD	**电源提供:** 提供 V_{DD} 电压
41	45	GND	**地:** 地电压
P0[7:0]			**P0:** 端口0可位操作， 复位后所有管脚为输入模式
39	43	P0.0/AIN0/T0	**P0.0:** 端口 0 管脚 0.
			AIN0: ADC 输入通道 0.
			T0: 外部计数输入到定时器/计数器0，或是输出
38	42	P0.1/AIN1/ $\overline{INT0}$	**P0.1:** 端口0管脚1.
			AIN1: ADC输入通道1.
			$\overline{INT0}$: 外部中断输入 0.
37	41	P0.2/AIN2	**P0.2:** 端口0管脚2.
			AIN2: ADC 输入通道2.
36	40	P0.3/AIN3	**P0.3:** 端口0管脚3.
			AIN3: ADC输入通道3.
35	39	P0.4/AIN4	**P0.4:** 端口0管脚4.
			AIN4: ADC 输入通道4.
34	38	P0.5/AIN5	**P0.5:** 端口0管脚5.
			AIN5: ADC输入通道5.
33	37	P0.6/AIN6	**P0.6:** 端口0管脚6.
			AIN6: ADC 输入通道 6.
32	36	P0.7/AIN7/ CLO/SEG31	**P0.7:** 端口0管脚7.
			AIN7: ADC输入通道7.
			CLO: 系统时钟输出
			SEG31: LCD segment 31 输出.
P1[7:0]			**P1:** 端口1可位操作， 复位后所有管脚为输入模式
1	1	P1.0/COM0	**P1.0:** 端口1管脚0.
			COM0: LCD common 0输出.
2	2	P1.1/COM1	**P1.1:** 端口1管脚1.
			COM1: LCD common 1输出.
3	3	P1.2/COM2	**P1.2:** 端口1管脚2.
			COM2: LCD common 2输出 .
4	4	P1.3/$\overline{INT1}$/ COM3	**P1.3:** 端口1管脚3.
			$\overline{INT1}$: 外部中断 1 输入.
			COM3: LCD common 3输出.
8	12	P1.4/T1/SEG7	**P1.4:** 端口1 管脚4.
			T1: 外部计数输入到定时器/计数器1，或是输出.
			SEG7: LCD segment 7输出.
9	13	P1.5/T2AO1/ SEG8	**P1.5:** Port 1管脚5.
			T2AO1: 定时器 2A 输出1.
			SEG8: LCD segment 8输出.
10	14	P1.6/T2AO2/ SEG9	**P1.6:** 端口1管脚6.
			T2AO2: 定时器 2A 输出 2.
			SEG9: LCD segment 9输出.
11	15	P1.7/T2BO1/ SEG10	**P1.7:** 端口1管脚7.
			T2BO1: 定时器 2B 输出 1.

表 4-1 管脚描述

管脚号码		符号	复合功能描述[1]
PQFP44 LQFP44	LQFP48		
			SEG10: LCD segment 10输出.

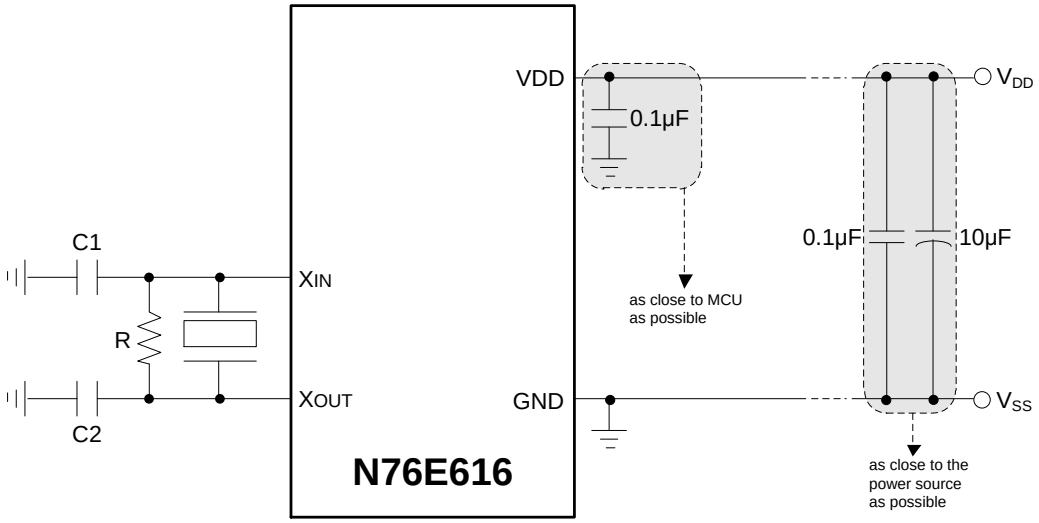
表 4-1 管脚描述

管脚号码		符号	复合功能描述[1]
PQFP44 LQFP44	LQFP48		
P2[7:0]			**P2:** 端口2可位操作， 复位后所有管脚为输入模式
12	16	P2.0/T2BO2/ SEG11	**P2.0:** 端口2 管脚 0.
			T2BO2: 定时器 2B 输出 2.
			SEG11: LCD segment 11 输出.
13	17	P2.1/RXD/ SEG12	**P2.1:** 端口2 管脚 1.
			RXD: 串行端口0 接收输入
			SEG12: LCD segment 12输出.
14	18	P2.2/TXD/ SEG13	**P2.2:** 端口2 管脚 2.
			TXD: 串行端口0 发送数据输出
			SEG13: LCD segment 13输出.
15	19	P2.3/SDA/ SEG14	**P2.3:** 端口2 管脚 3.
			SDA: I^2C 数据.
			SEG14: LCD segment 14输出.
16	20	P2.4/SCL/ SEG15	**P2.4:** 端口2 管脚 4.
			SCL: I^2C 时钟.
			SEG15: LCD segment 15输出.
17	21	P2.5/SEG16	**P2.5:** 端口2 管脚 5.
			SEG16: LCD segment 16输出.
18	22	P2.6/T2CO1/ SEG17	**P2.6:** 端口2 管脚 6.
			T2CO1: 定时器 2C输出1.
			SEG17: LCD segment 17输出.
19	23	P2.7/T2CO2/ SEG18	**P2.7:** 端口2 管脚 7.
			T2CO2: 定时器 2C输出2.
			SEG18: LCD segment 18输出.
P3[6:0]			**P3:** 端口2可位操作， 复位后所有管脚为输入模式。P3.6如果定义为普通I/O ， 始终为输入模式。
28	32	P3.0/T2DO1/ SEG27	**P3.0:** 端口3 管脚 0.
			T2DO1: 定时器 2D输出1.
			SEG27: LCD segment 27 输出.
29	33	P3.1/T2DO2/ SEG28	**P3.1:** 端口3 管脚 1.
			T2DO1: 定时器 2D输出2.
			SEG28: LCD segment 28输出.
30	34	P3.2/XIN/ SEG29	**P3.2:** 当HXT 或 LXT 没用到，端口3 管脚 2 可用。
			XIN: 如果 HXT 或 LXT 被使用, XIN 是输入管脚到内部反向放大器。如果ECLK 模式使能，XIN 是外部时钟输入引脚。
			SEG29: LCD segment 29 输出.
35	35	P3.3/XOUT/ SEG30	**P3.3:** 当HXT 或 LXT 没用到，端口3 bit 3 可用。
			XOUT: 如果 HXT 或 LXT 被使用, XOUT是输出管脚来自内部反向放大器。如果ECLK 模式使能，它发出XIN的反向信号。
			SEG30: LCD segment 30输出.
42	46	P3.4/ICPDA/ OCDDA	**P3.4:** 端口3 管脚 4.
			ICPDA: ICP 数据输入或输出.
			OCDDA: OCD数据输入或输出.

表 4-1 管脚描述

管脚号码		符号	复合功能描述[1]
PQFP44 LQFP44	LQFP48		
43	47	P3.5/ICPCK/ OCDCK	**P3.5:** 端口 3 管脚 5. **ICPCK:** ICP 时钟输入 **OCDCK:** OCD时钟输入
44	48	P3.6/$\overline{\text{RST}}$	**P3.6:** 当 RPD (CONFIG0.2) 配置为0，端口3 bit 6 输入引脚可用 。 **$\overline{\text{RST}}$:** $\overline{\text{RST}}$ 引脚是施密特触发输入引脚，用于硬件复位。该引脚的一个低电平复位芯片。$\overline{\text{RST}}$ 引脚有一个内部上拉电阻，通过简单的连一个电容到地，可以实现上电复位。
P4[6:0]			**P4:** 端口4仅可字节操作， 复位后所有管脚为输入模式
5	5	P4.0/SEG0/ COM4	**P4.0:** 端口4 管脚 0. **SEG0:** LCD segment 0 输出. **COM4:** LCD common 4输出.
6	6	P4.1/SEG1/ COM5	**P4.1:** 端口4 管脚 1. **SEG1:** LCD segment 1输出. **COM5:** LCD common 5输出.
7	7	P4.2/SEG2	**P4.2:** 端口4 管脚 2. **SEG2:** LCD segment 2输出.
-	8	P4.3/SEG3	**P4.3:** 端口4 管脚 3. **SEG3:** LCD segment 3输出.
-	9	P4.4/SEG4	**P4.4:** 端口4 管脚 4. **SEG4:** LCD segment 4输出.
-	10	P4.5/SEG5	**P4.5:** 端口4 管脚 5. **SEG5:** LCD segment 5输出.
-	11	P4.6/SEG6	**P4.6:** 端口4 管脚 6. **SEG6:** LCD segment 6输出.
P5[7:0]			**P5:** 端口5可位操作， 复位后所有管脚为输入模式
20	24	P5.0/STADC/ SEG19	**P5.0:** 端口5 管脚 0. **STADC:** 外部触发ADC信号. **SEG19:** LCD segment 19输出.
21	25	P5.1/SEG20	**P5.1:** 端口5 管脚 1. **SEG20:** LCD segment 20输出.
22	26	P5.2/SEG21	**P5.2:** 端口5 管脚 2. **SEG21:** LCD segment 21输出.
23	27	P5.3/SEG22	**P5.3:** 端口5 管脚 3. **SEG22:** LCD segment 22输出.
24	28	P5.4/SEG23	**P5.4:** 端口5 管脚 4. **SEG23:** LCD segment 23输出.
25	29	P5.5/SEG24	**P5.5:** 端口5 管脚 5. **SEG24:** LCD segment 24输出.
26	30	P5.6/RXD_1/ SEG25	**P5.6:** 端口5 管脚 6. **RXD_1:** 串行端口 1 接收输入 **SEG25:** LCD segment 25输出.
27	31	P5.7/TXD_1/ SEG26	**P5.7:** 端口5 管脚 7. **TXD_1:** 串行端口 1 发送数据输出. **SEG26:** LCD segment 26 输出.

[1] 所有 I/O 引脚可以配置成中断引脚，这个特征没有在多功能描述里面列出来。详见 章节**錯誤! 找不到參照來源。**“**錯誤! 找不到參照來源。**”



晶振频率	R	C1	C2
2 MHz 至 16 MHz	无需	根据晶振规格书要求	
32.768 kHz	无需		

图 4.1. 外部晶振应用线路

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N76E616中，有256字节内部RAM。对于需要更多内部RAM的许多应用，N76E616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，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256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

编程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。

N76E616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。N76E616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**錯誤! 找不到參照來源。**“**錯誤! 找不到參照來源。**”。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 大小选择** 该域用于选择 LDROM的大小. 111 = 无 LDROM. APROM 是 18K 字节. 110 = LDROM是1K字节. APROM是17K字节. 101 = LDROM是2K字节. APROM是16K字节. 100 = LDROM是3K字节. APROM是15K字节. 0xx = LDROM是4K字节. APROM是14K字节.

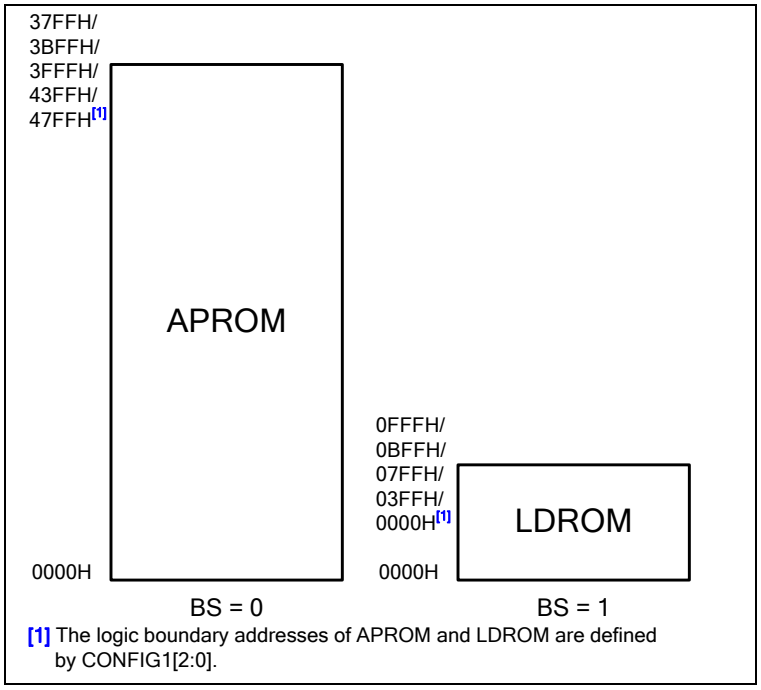


图 5.1. N76E616程序存储映像

5.2 数据存储器

图 5.2所示N76E616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。

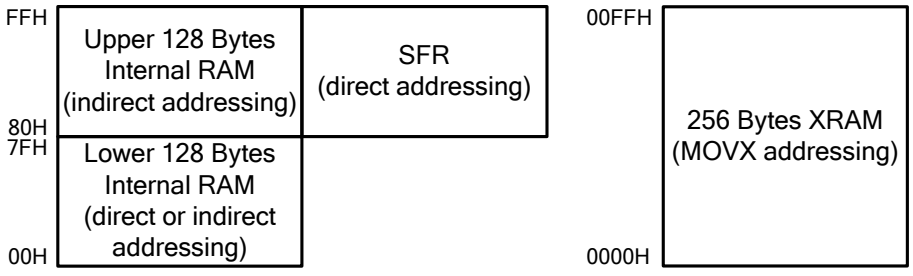


图 5.2.数据存储映射

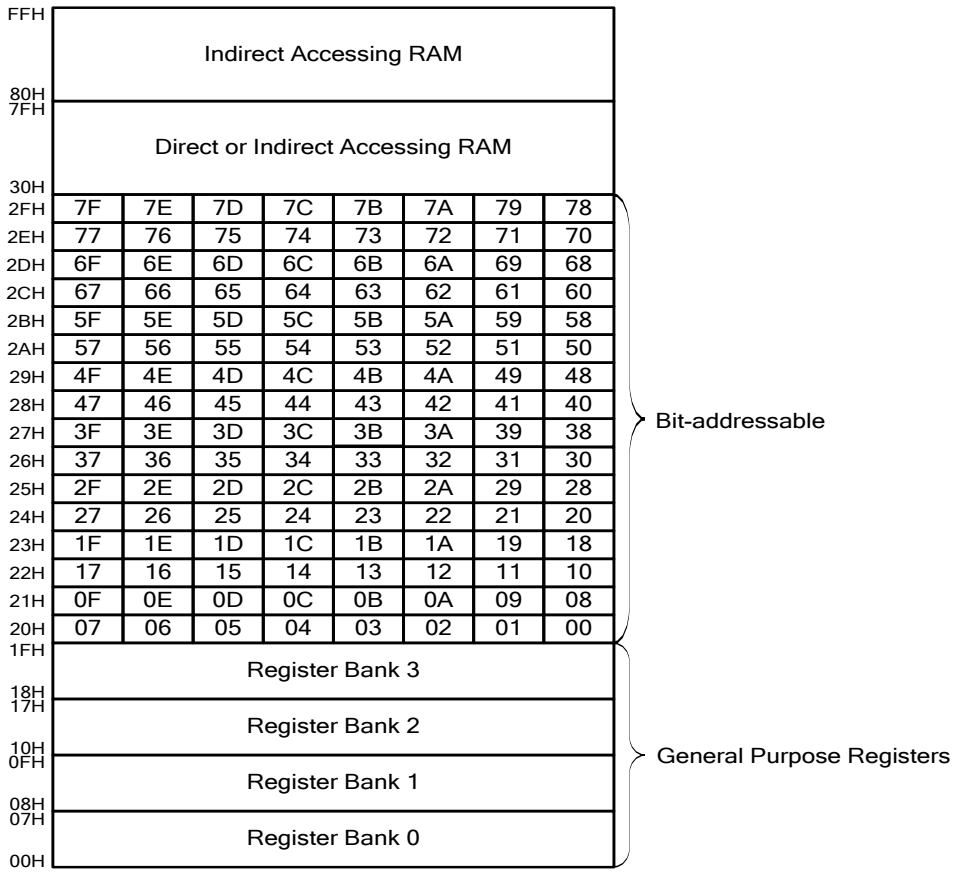

FFH
Indirect Accessing RAM
80H
7FH
Direct or Indirect Accessing RAM
30H
2FH 7F 7E 7D 7C 7B 7A 79 78
2EH 77 76 75 74 73 72 71 70
2DH 6F 6E 6D 6C 6B 6A 69 68
2CH 67 66 65 64 63 62 61 60
2BH 5F 5E 5D 5C 5B 5A 59 58
2AH 57 56 55 54 53 52 51 50
29H 4F 4E 4D 4C 4B 4A 49 48
28H 47 46 45 44 43 42 41 40
27H 3F 3E 3D 3C 3B 3A 39 38
26H 37 36 35 34 33 32 31 30
25H 2F 2E 2D 2C 2B 2A 29 28
24H 27 26 25 24 23 22 21 20
23H 1F 1E 1D 1C 1B 1A 19 18
22H 17 16 15 14 13 12 11 10
21H 0F 0E 0D 0C 0B 0A 09 08
20H 07 06 05 04 03 02 01 00
Bit-addressable
1FH
Register Bank 3
18H
17H
Register Bank 2
10H
0FH
Register Bank 1
08H
07H
Register Bank 0
00H
General Purpose Registers

图 5.3. 内部256字节RAM地址

5.3 片上 XRAM

N76E616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 示例代码：

```
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易失性数据存储。IAP的细节，请看章节 21.”錯誤! 找不到參照來源。”

6. 特殊功能寄存器 (SFR)

N76E616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。N76E616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 页的示例代码:

```
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存储映射

页	地址	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
0 1	F8	SCON_1	LCDCON	LCDCLK	LCDPTR	LCDDAT -	-	EIP1 -	EIPH1 -
0 1	F0	B	-	ADCAQT	-	-	-	P0DIDS	EIPH
0 1	E8	ADCCON0	PICON	PINEN	PIPEN	PIF	PITYP	LCDSEG3	EIP
0 1	E0	ACC	ADCCON1	ADCCON2	ADCMPL	ADCMPH	LCDSEG0	LCDSEG1	LCDSEG2
0 1	D8	P5	P4	P4M1 P4S	P4M2 -	P5M1 P5S	P5M2 -	-	-
0 1	D0	PSW	-	-	-	R2CL	R2CH	R2DL	R2DH
0 1	C8	T2CON	T2MOD0	T2MOD1	T2OE	R2AL	R2AH	R2BL	R2BH
0 1	C0	I2CON	I2ADDR	ADCRL	ADCRH	T3CON	R3L	R3H	TA
0 1	B8	IP	SADEN	SADEN_1	SADDR_1	I2DAT	I2STAT	I2CLK	I2TOC
0 1	B0	P3	P0M1 P0S	P0M2 -	P1M1 P1S	P1M2 P1OS	P2M1 P2S	P2M2 -	IPH
0 1	A8	IE	SADDR	WDCON	BODCON1	P3M1 P3S	P3M2 -	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	-	-	-	CKDIV	CKSWT	CKEN
0 1	88	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON	WKCON
0 1	80	P0	SP	DPL	DPH	-	-	RWK	PCON

SFR空间中未使用地址用‘-‘ 标记，访问改写这些地址字节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
LCDDAT	LCD 数据寄存器	FCH/0	-	-	LCDDAT[5:0]						0000 0000b
LCDPTR	LCD 数据指针	FBH	-	-	-	LCDPTR[4:0]					0000 0000b
LCDCLK	LCD 时钟控制寄存器	FAH	-	-	LCDCKS[1:0]		-	LCDDIV[2:0]			0000 0000b
LCDCON	LCD 控制寄存器	F9H	LCDEN	VLCDADJ	-	BIAS	DUTY[1:0]		RSEL[1:0]		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	PT2DH	-	PT2CH	PWDTH	PT2BH	PT2AH	PPIH	PI2CH	0000 0000b
P0DIDS	P0 数字输出关闭	F6H	P07DIDS	P06DIDS	P05DIDS	P04DIDS	P03DIDS	P02DIDS	P01DIDS	P00DIDS	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	PT2D	-	PT2C	PWDT	PT2B	PT2A	PPI	PI2C	0000 0000b
LCDSEG3	输入捕获2数据高位字节	EEH	SEG31EN	SEG30EN	SEG29EN	SEG28EN	SEG27EN	SEG26EN	SEG25EN	SEG24EN	0000 0000b
PITYP	输入捕获2数据低位字节	EDH	PIT7	PIT6	PIT5	PIT4	PIT3	PIT2	PIT1	PIT0	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	-	-	-	-	-	PIPS[2:0]			0000 0000b
ADCCON0	ADC 控制寄存器 0	E8H	(EF) ADCF	(EE) ADCS	(ED) -	(EC) -	(EB) ADCHS3	(EA) ADCHS2	(E9) ADCHS1	(E8) ADCHS0	0000 0000b
LCDSEG2	LCD segment 2	E7H	SEG23EN	SEG22EN	SEG21EN	SEG20EN	SEG19EN	SEG18EN	SEG17EN	SEG16EN	0000 0000b
LCDSEG1	LCD segment 1	E6H	SEG15EN	SEG14EN	SEG13EN	SEG12EN	SEG11EN	SEG10EN	SEG9EN	SEG8EN	0000 0000b
LCDSEG0	LCD segment 0	E5H	SEG7EN	SEG6EN	SEG5EN	SEG4EN	SEG3EN	SEG2EN	SEG1EN	SEG0EN	0000 0000b
ADCMPH	ADC 比较数据高字节	E4H	ADCMP[9:2]								0000 0000b
ADCMPL	ADC比较数据低字节	E3H	-	-	-	-	-	-	ADCMP[1:0]		0000 0000b
ADCCON2	ADC 控制寄存器 2	E2H	-	ADCMPOP	ADCMPEN	ADCMPO	-	-	-	-	0000 0000b
ADCCON1	ADC 控制寄存器 1	E1H	-	ADCDIV[2:0]			-	-	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
P5M2	P5 模式选择寄存器2	DDH/0	P5M2.7	P5M2.6	P5M2.5	P5M2.4	P5M2.3	P5M2.2	P5M2.1	P5M2.0	0000 0000b
P5S	P5 施密特触发输入	DCH/1	P5S.7	P5S.6	P5S.5	P5S.4	P5S.3	P5S.2	P5S.1	P5S.0	0000 0000b
P5M1	P5 模式选择1	DCH/0	P5M1.7	P5M1.6	P5M1.5	P5M1.4	P5M1.3	P5M1.2	P5M1.1	P5M1.0	1111 1111b
P4M2	P4 模式选择2	DBH/0	-	P4M2.6	P4M2.5	P4M2.4	P4M2.3	P4M2.2	P4M2.1	P4M2.0	0000 0000b
P4S	P4施密特触发输入	DAH/1	-	P4S.6	P4S.5	P4S.4	P4S.3	P4S.2	P4S.1	P4S.0	0000 0000b
P4M1	P4模式选择1	DAH/0	-	P4M1.6	P4M1.5	P4M1.4	P4M1.3	P4M1.2	P4M1.1	P4M1.0	0111 1111b
P4	端口 4	D9H	0	P4.6	P4.5	P4.4	P4.3	P4.2	P4.1	P4.0	Output latch, 0111 1111b Input, 0XXX XXXXb[3]
P5	端口 5	D8H	(DF) P5.7	(DE) P5.6	(DD) P5.5	(DC) P5.4	(DB) P5.3	(DA) P5.2	(D9) P5.1	(D8) P5.0	Output latch, 1111 1111b Input, XXXX XXXXb[3]
R2DH	定时器 2D重载数据高字节	D7H	R2DH[7:0]								0000 0000b
R2DL	定时器 2D重载数据低字节	D6H	R2DL[7:0]								0000 0000b
R2CH	定时器 2C重载数据高字节	D5H	R2CH[7:0]								0000 0000b
R2CL	定时器 2C重载数据低字节	D4H	R2CL[7:0]								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
R2BH	定时器 2B重载数据高字节	CFH	R2BH[7:0]								0000 0000b
R2BL	定时器 2B重载数据低字节	CEH	R2BL[7:0]								0000 0000b
R2AH	定时器 2A重载数据高字节	CDH	R2AH[7:0]								0000 0000b
R2AL	定时器 2A重载数据低字节	CCH	R2AL[7:0]								0000 0000b
T2OE	定时器 2 输出使能	CBH	T2DOE2	T2DOE1	T2COE2	T2COE1	T2BOE2	T2BOE1	T2AOE2	T2AOE1	0000 0000b
T2MOD1	定时器2 模式 1	CAH	T2DM	T2DPS[2:0]			T2CM	T2CPS[2:0]			0000 0000b
T2MOD0	定时器2模式0	C9H	T2BM	T2BPS[2:0]			T2AM	T2APS[2:0]			0000 0000b
T2CON	定时器2 控制寄存器	C8H	(CF) TF2D	(CE) TF2C	(CD) TF2B	(CC) TF2A	(CB) TR2D	(CA) TR2C	(C9) TR2B	(C8) TR2A	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
R3H	定时器3自动重载高字节	C6H	R3H[7:0]								0000 0000b
R3L	定时器3自动重载低字节	C5H	R3L[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	I²C 从机地址	C1H	I2ADDR[7:1]							GC	0000 0000b
I2CON	I²C 控制寄存器	C0H	(C7) -	(C6) I2CEN	(C4) STA	(C4) STO	(C3) SI	(C2) AA	(C1) -	(C0) -	0000 0000b
I2TOC	I²C定时计数器	BFH	-	-	-	-	-	I2TOCEN	DIV	I2TOF	0000 0000b
I2CLK	I²C 时钟	BEH	I2CLK[7:0]								0000 1110b
I2STAT	I²C 状态	BDH	I2STAT[7:3]					0	0	0	1111 1000b
I2DAT	I²C 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
P2M2	P2 模式选择2	B6H/0	P2M2.7	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.7	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	P2M1.7	P2M1.6	P2M1.5	P2M1.4	P2M1.3	P2M1.2	P2M1.1	P2M1.0	1111 1111b
P1OS	P1 输出增强	B4H/1	-	-	-	-	P1OS.3	P1OS.3	P1OS.1	P1OS.0	0000 0000b
P1M2	P1 模式选择2	B4H/0	P1M2.7	P1M2.6	P1M2.5	P1M2.4	P1M2.3	P1M2.2	P1M2.1	P1M2.0	0000 0000b
P1S	P1 施密特触发输入	B3H/1	P1S.7	P1S.6	P1S.5	P1S.4	P1S.3	P1S.2	P1S.1	P1S.0	0000 0000b
P1M1	P1 模式选择1	B3H/0	P1M1.7	P1M1.6	P1M1.5	P1M1.4	P1M1.3	P1M1.2	P1M1.1	P1M1.0	1111 1111b
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 施密特触发输入	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	(DF) 0	(DE) P3.6	(DD) P3.5	(DC) P3.4	(DB) P3.3	(DA) P3.2	(D9) P3.1	(D8) P3.0	Output latch, 0011 1111b Input, 0XXX XXXXb[3]
IAPCN	IAP 控制	AFH	IAPB[1:0]		FOEN	FCEN	FCTRL[3:0]				0011 0000b
IAPFD	IAP 数据	AEH	IAPFD[7:0]								0000 0000b
P3M2	P3 模式选择 2	ADH/0	CLOEN	P36UP	P3M2.5	P3M2.4	P3M2.3	P3M2.2	P3M2.1	P3M2.0	0000 0000b
P3S	P3 施密特输入	ACH/1	-	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	T1OE	T0OE	P3M1.5	P3M1.4	P3M1.3	P3M1.2	P3M1.1	P3M1.0	0011 1111b
BODCON1[4]	欠压检测控制1	ABH	-	-	-	-	-	LPBOD[1:0]		BODFLT	POR, 0000 0001b Others, 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 Others, 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
BODCON0[4]	欠压侦测控制0	A3H	BODEN[5]	-	BOV[1:0][5]		BOF[6]	BORST[5]	BORF	BOS[7]	POR, C0CC XC0Xb BOD, U0UU XU1Xb Others, U0UU XUUXb

表 6–2. SFR 定义及复位值

符号	定义	地址/页	MSB							LSB[1]	复位值[2]
AUXR1	辅助寄存器 1	A2H	SWRF	RSTPINF	T1LXTM	T0LXTM	GF2	-	0	DPS	POR, 0000 0000b Software, 1U00 0000b $\overline{RST}$ pin, U100 0000b Others, UU00 0000b
P2	端口 2	A0H	(A7) P2.7	(A6) P2.6	(A5) P2.5	(A4) P2.4	(A3) P2.3	(A2) P2.2	(A1) P2.1	(A0) P2.0	Output latch, 1111 1111b Input, XXXX 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	ET2D	-	ET2C	EWDT	ET2B	ET2A	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	Serial port 0 control	98H	(9F) SM0/FE	(9E) SM1	(9D) SM2	(9C) REN	(9B) TB8	(9A) RB8	(99) TI	(98) RI	0000 0000b
CKEN[4]	Clock enable	97H	EXTEN[1:0]		HIRCEN	-	-	-	-	CKSWTF	0011 0000b
CKSWT[4]	Clock switch	96H	HXTST	LXTST	HIRCST	-	ECLKST	OSC[2:0]			0011 0000b
CKDIV	Clock divider	95H	CKDIV[7:0]								0000 0000b
SFRS[4]	SFR page selection	91H	-	-	-	-	-	-	-	SFRPSEL	0000 0000b
P1	端口 1	90H	(97) P1.7	(96) P1.6	(95) P1.5	(94) P1.4	(93) P1.3	(92) P1.2	(91) P1.1	(90) P1.0	Output latch, 1111 1111b Input, XXXX XXXXb[3]
WKCON	自唤醒定时器控制	8FH	-	-	WKTCKS	WKTF	WKTR	WKPS[2:0]			0000 0000b
CKCON	时钟控制	8EH	-	-	-	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: 看 **[5]**; X: 看 **[3]**, **[6]**, 和 **[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)保持未编程1，那么读回P3.6引脚总是0。

[4] 这些 SFR有 TA 写保护。 看 章节 **錯誤! 找不到參照來源。**"**錯誤! 找不到參照來源。**"

[5] 在特定的复位之后，这些 SFR 根据CONFIG。细节看 章节 **錯誤! 找不到參照來源。**"**錯誤! 找不到參照來源。**"

[6] BOF 复位值取决于CONFIG2 不同的设置和V_{DD} 电压值。请查看

[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，访问改写这些位可能导致不可预知的后果。

7. 通用80C51 系统控制

A or 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** A 或 ACC 寄存器是标准的80C51 累加器，用于算术操作。

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** B 寄存器是另外一个标准80C51的累加器，它用于MUL 和 DIV 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 – 程序状态字 (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**

位	名称	描述
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.

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.
2	GF0	**通用标志0** 通用标志可由用户置位或清零.

8. I/O 端口结构和操作

N76E616最多支持26个可位寻址的通用I/O引脚，分成6组P0到P5 。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P3.6）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. 管脚I/O模式配置

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。默认为TTL电平输入模式。

有四个I/O引脚支持大的输出或灌电流，包括P1.0 到 P1.3。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P1OS寄存器的对应位，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，P3.6被配置为输入引脚。同时P3.6将永远在输入和施密特触发模式，且有一个内部上拉电阻，通过P36UP(P3M2.6)使能。如果RPD未编程，P3.6作为外部复位引脚，P3.6无效。

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，若为外部悬空状态，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I_{TL}）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。

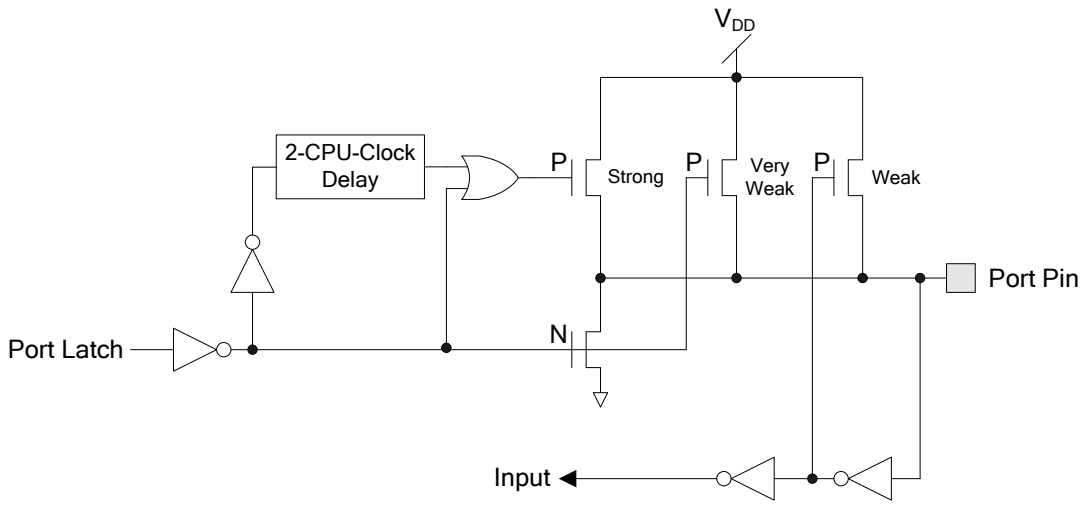


图 8.1. 准双向模式结构

8.2 推挽输出模式

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。

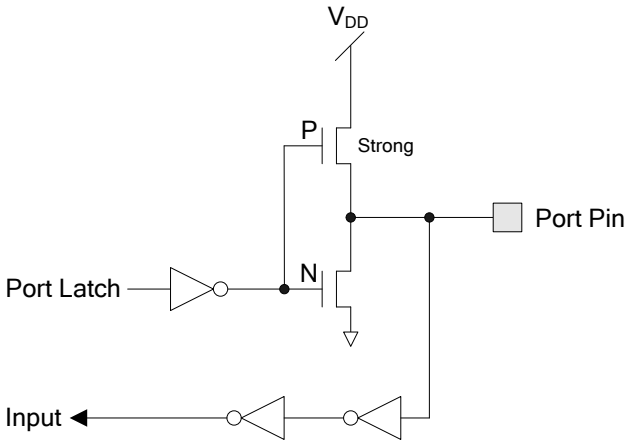


图 8.2. 推挽输出模式结构

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。

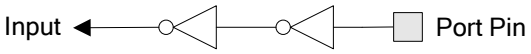


图 8.3. 输入高阻模式结构图

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I^2C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。

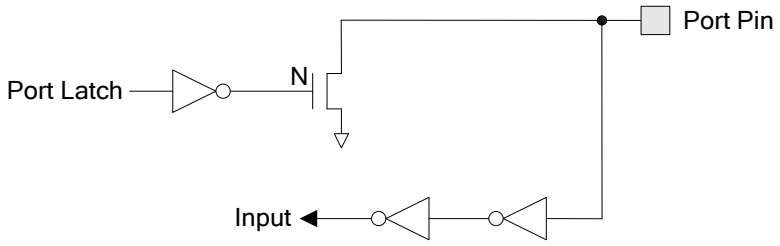


图 8.4. 开漏模式结构

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 I/O 端口的控制寄存器

N76E616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位 通用 I/O 端口.

P1 – 端口 1 (可位寻址)

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

地址: 90H 复位值: 1111 1111b

位	名称	描述
7:0	P1[7:0]	**端口 1** 端口 1 是 8位 通用 I/O 端口.

P2 – 端口 2 (可位寻址)

7	6	5	4	3	2	1	0
P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
读/写	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	P2[7:0]	**端口 2** 端口2 是 8位 通用 I/O 端口.

P3 – 端口 3 (可位寻址)

7	6	5	4	3	2	1	0
0	P3.6	P3.5	P3.4	P3.3	P3.2	P31	P3.0
R	R	读/写	读/写	读/写	读/写	读/写	读/写

地址: B0H 复位值: 0011 1111b

位	名称	描述
7	0	**保留** 该位读总是 0.
6	P3.6	**端口 3 第6位** 当RPD (CONFIG0.2) 配置为0，P3.6 是输入引脚。当RPD未配置P3.6读总是0。
5	P3.5	**端口 3 第5位** P3.5 是通用 I/O 引脚, 与多功能引脚 ICPCK 和 OCDCK共享
4	P3.4	**端口 3 第4位** P3.4 是通用 I/O 引脚, 与多功能引脚ICPDA和OCDDA共享
3	P3.3	**端口 3 第3位** 当 HXT 或 LXT 没有使用，P3.3可用。这种情况下P3.3 功能是通用 I/O。 如果 HXT 或 LXT 使用， P3.3 引脚功能作为 XOUT，写 P3.3 无效，读P3.3 总是0。
2	P3.2	**端口 3 第2位** 仅当 HXT, LXT, 或 ECLK 没有使用，P3.2可用，这种情况下P3.2 功能是通用 I/O。 如果 HXT, LXT, 或 ECLK使用， P3.2 引脚功能作为XIN。 写 P3.2无效，读P3.2 总是0。
1	P3.1	**端口 3 第1位** P3.1 是通用 I/O 引脚, 与多功能引脚T2DO2和SEG28共享
0	P3.0	**端口 3 第0位** P3.0 是通用 I/O 引脚, 与多功能引脚T2DO1和SEG27共享

P4 – 端口 4

7	6	5	4	3	2	1	0
0	P4.6	P4.5	P4.4	P4.3	P4.2	P4.1	P4.0
R	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: D9H 复位值: 0111 1111b

位	名称	描述
7	-	**保留** 该位读总是 0.
6:0	P4[6:0]	**端口 4** 端口 4 是 7位 通用 I/O 端口.

P5 – 端口 5 (可位寻址)

7	6	5	4	3	2	1	0
P5.7	P5.6	P5.5	P5.4	P5.3	P5.2	P5.1	P5.0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: D8H 复位值: 1111 1111b

位	名称	描述
7:0	P5[7:0]	**端口 5** 端口 5 是 8位 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P3.6引脚，在P3M6.2中有一个上拉控制。

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，详见表 8–1..

P1M1 – 端口 1 模式选择1[2]

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

地址: B3H, 页: 0　　复位值: 1111 1111b

位	名称	描述
7:0	P1M1[7:0]	**端口 1 模式选择 1**

P1M2 – 端口 1 模式选择 2[2]

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

地址: B4H, 页: 0　　复位值: 0000 0000b

位	名称	描述
7:0	P1M2[7:0]	**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，详见表 8–1..

P2M1 – 端口 2 模式选择 1[3]

7	6	5	4	3	2	1	0
P2M1.7	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	读/写

地址: B5H, 页: 0　　复位值: 0111 1111b

位	名称	描述
7:0	P2M1[7:0]	**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
P2M2.7	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	读/写

地址: B6H, 页: 0　　复位值: 0000 0000b

位	名称	描述
7:0	P2M2[7:0]	**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，详见表 8–1..

P3M1 – 端口 3 模式选择 1

7	6	5	4	3	2	1	0
T1OE	T0OE	P3M1.5[4]	P3M1.4[4]	P3M1.3[4]	P3M1.2[4]	P3M1.1[4]	P3M1.0[4]
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: ACH, 页: 0　　复位值: 0000 0011b

位	名称	描述
5:0	P3M1[5:0]	**端口 3 模式选择 1**

P3M2 – 端口 3 模式选择 2

7	6	5	4	3	2	1	0
CLOEN	P36UP	P3M2.5[4]	P3M2.4[4]	P3M2.3[4]	P3M2.2[4]	P3M2.1[4]	P3M2.0[4]
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
6	P36UP	**P3.6 上拉使能** 0 = P3.6 上拉禁止. 1 = P3.6 上拉使能 仅当RPD (CONFIG0.2) 被配置为 0.该位有效。当选择作为 $\overline{\text{RST}}$ 引脚上拉总是使能的。
5:0	P3M2[5:0]	**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，详见表 8–1..

P4M1 – 端口 4 模式选择 1[5]

7	6	5	4	3	2	1	0
-	P4M1.6	P4M1.5	P4M1.4	P4M1.3	P4M1.2	P4M1.1	P4M1.0
-	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: DAH, 页: 0　　复位值: 1111 1111b

位	名称	描述
6:0	P4M1[6:0]	**端口 4 模式选择 1**

P4M2 – 端口 4 模式选择 2[5]

7	6	5	4	3	2	1	0
-	P4M2.6	P4M2.5	P4M2.4	P4M2.3	P4M2.2	P4M2.1	P4M2.0
-	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: DBH, 页: 0　　复位值: 0000 0000b

位	名称	描述
6:0	P4M2[6:0]	**端口 4 模式选择 2**

[5] P4M1 和 P4M2结合用于决定P4每个引脚的I/O模式，详见表 8–1..

P5M1 – 端口 5 模式选择 1[6]

7	6	5	4	3	2	1	0
P5M1.7	P5M1.6	P5M1.5	P5M1.4	P5M1.3	P5M1.2	P5M1.1	P5M1.0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: DCH, 页: 0　　复位值: 1111 1111b

位	名称	描述
7:0	P5M1[7:0]	**端口 5 模式选择 1**

P5M2 – 端口 5 模式选择 2[6]

7	6	5	4	3	2	1	0
P5M2.7	P5M2.6	P5M2.5	P5M2.4	P5M2.3	P5M2.2	P5M2.1	P5M2.0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: DDH, 页: 0　　复位值: 0000 0000b

位	名称	描述
7:0	P5M2[7:0]	**端口 5 模式选择 2**

[6] P5M1 和 P5M2结合用于决定P5每个引脚的I/O模式，详见表 8–1..

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。P1OS[3:0] 位用于P1.0到P1.3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 和 P1OS 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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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** 0 = P0.n TTL电平输入 1 = P0.n.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
P1S.7	P1S.6	P1S.5	P1S.4	P1S.3	P1S.2	P1S.1	P1S.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

地址: B3H, 页: 1　　复位值: 0000 0000b

位	名称	描述
n	P1S.n	**P1.n施密特触发输入** 0 = P1.n TTL电平输入 1 = P1.n.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.7	P2S.6	P2S.5	P2S.4	P2S.3	P2S.2	P2S.1	P2S.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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** 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.6	P3S.5	P3S.4	P3S.3	P3S.2	P3S.1	P3S.0
-	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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** 0 = P3.n.TTL电平输入 1 = P3.n施密特触发输入

P4S –端口4施密特触发输入

7	6	5	4	3	2	1	0
-	P4S.6	P4S.5	P4S.4	P4S.3	P4S.2	P4S.1	P4S.0
-	R/W	R/W	R/W	R/W	R/W	R/W	R/W

地址: DAH, 页: 1　　复位值: 0000 0000b

位	名称	描述
n	P4S.n	**P4.n施密特触发输入** 0 = P4.n.TTL电平输入 1 = P4.n施密特触发输入

P5S –端口5施密特触发输入

7	6	5	4	3	2	1	0
P5S.7	P5S.6	P5S.5	P5S.4	P5S.3	P5S.2	P5S.1	P5S.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

地址: DCH, 页: 1　　复位值: 0000 0000b

位	名称	描述
n	P5S.n	**P5.n施密特触发输入** 0 = P5.n.TTL电平输入 1 = P5.n施密特触发输入

P1OS –端口1 输出能力控制

7	6	5	4	3	2	1	0
-	-	-	-	P1OS.3	P1OS.2	P1OS.1	P1OS.0
-	-	-	-	R/W	R/W	R/W	R/W

地址: B4H, 页: 1　　复位值: 0000 0000b

位	名称	描述
7:4	-	**保留**
n	P1OS.n	**P1.n 输出能力选择** 0 = P1.n 正常的输出能力 1 = P1.n 大电流输出能力. 注意：仅当I/O配置为推挽输出模式的时候，该配置才有效。

9. 定时器/计数器 0 和 1

N76E616的定时器/计数器 0 和 1分别为两个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。

定时器或计数器功能通过TMOD的C/$\overline{T}$位来选择。每一个定时器/计数器有它自己的选择位。TMOD.2用于定时器/计数器0功能选择，TMOD.6用于定时器/计数器1功能选择。

将它们设置作为定时器模式下，定时器将对时钟周期计数。T0M (CKCON.3) 位用于定时器0的时钟选择，时钟源可以是系统时钟的12分频或是直接系统时钟(F_{SYS})。T1M (CKCON.4) 用于定时器1时钟选择。 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。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，那么就会在T0或T1引脚识别到一个电平由高到低的跳变。当T0LXTM (T1LXTM) 置位，N76E616支持LXT输入模式。无论系统时钟怎么切换，它提供一个固定的溢出速率。此外每一个定时器/计数器可以配置为四种可能的模式中的任何一种。通过TMOD中的M0 和 M1 来选择模式。

定时器0和1可以配置成当定时器溢出发生的时候自动取反一个端口输出。T0 和 T1既用来做定时器的计数输入也用来做取反输出。这个功能通过P1M1寄存器的T0OE 和 T1OE控制位来使能。分别应用于定时器0和定时器1。当这种模式打开，定时器第一次溢出是先输出逻辑1。为了实现该模式的功能，C/$\overline{T}$位应该清零选择系统时钟作为定时器的时钟源。

注意：TH0 (TH1) 和 TL0 (TL1)是分别被访问的，强烈建议在模式0或模式1，在读或写TH0 (TH1) 和TL0 (TL1).之前，用户应该通过清零TR0 (TR1) 位暂停定时器。在自由计数下读或写会引起不可预知的结果。

TMOD – 定时器 0 及 1 模式

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 门控制** 0 = 当TR1=1时，定时器 1 时钟运行不管INT1的逻辑电平 1 = 当TR1=1和INT1为逻辑1，定时器1 运行

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 4	M1 M0	**定时器 1 模式选择** M1 M0 定时器1 模式 0 0 模式 0: 13位 定时器/计数器 0 1 模式 1: 16位 定时器/计数器 1 0 模式 2: 8位 带从TH1自动重载功能的定时器/计数器 1 1 模式 3: 定时器 1 停止
3	GATE	**定时器 0 门控制** 0 = 当TR0=1时，定时器 0 时钟运行不管INT0的逻辑电平 1 = 当TR0=1和INT0为逻辑1，定时器0 运行
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T0的下降沿递增
1 0	M1 M0	**定时器 0 模式选择** M1 M0 定时器 0 模式 0 0 模式0: 13位 定时器/计数器 0 1 模式1: 16位 定时器/计数器 1 0 模式2: 8位 带从TH0自动重载功能的定时器/计数器 1 1 模式3: TL0 作为一个 8位 定时器/计数器， TH0 作为一个 8位 定时器

TCON – 定时器 0 及 1 控制 (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

位	名称	描述
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 数据低字节

7	6	5	4	3	2	1	0
TL0[7:0]							
读/写							

地址: 8AH 复位值: 0000 0000b

位	名称	描述
7:0	TL0[7:0]	**定时器 0 数据低字节** 寄存器TL0是定时器0的16位计数数据寄存器低8位字节 .

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位计数寄存器的高字节 .

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位计数寄存器的低字节 .

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位计数寄存器的高字节

CKCON – 时钟控制

7	6	5	4	3	2	1	0
-	-	-	T1M	T0M	-	-	-
-	-	-	R/W	R/W	-	-	-

地址: 8EH 复位值: 0000 0000b

位	名称	描述
4	T1M	**定时器 1 时钟模式选择** 0 =定时器 1 的时钟源是系统时钟的12除频，它保留标准8051的特性。. 1 =定时器 1 的时钟源直接是系统时钟
3	T0M	**定时器 0 时钟模式选择** 0 = 定时器 0 的时钟源是系统时钟的12除频，它保留标准8051的特性。. 1 = 定时器 0 的时钟源直接是系统时钟.

AUXR1 – 辅助寄存器 1

7	6	5	4	3	2	1	0
SWRF	RSTPINF	T1LXTM	T0LXTM	GF2	-	0	DPS
R/W	R/W	R/W	R/W	R/W	-	R	R/W

地址: A2H 复位值 见 表 6–2.

位	名称	描述
5	T1LXTM	**定时器 1 LXT 输入模式** 0 = 定时器 1 计数时钟通过C/$\overline{T}$ (TMOD.6)和T1M (CKCON.4)选择 1 = 定时器1 计数 LXT 时钟.
4	T0LXTM	**定时器0 LXT 输入模式** 0 = 定时器 0 计数时钟通过C/$\overline{T}$ (TMOD.2) 和T0M (CKCON.3).选择 1 = 定时器 0 计数 LXT 时钟.

P3M1 – 端口 3 模式选择 1

7	6	5	4	3	2	1	0
T1OE	T0OE	P3M1.5	P3M1.4	P3M1.3	P3M1.2	P3M1.1	P3M1.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

地址: ACH, 页: 0　　复位值: 0011 1111b

位	名称	描述
7	T1OE	**定时器 1 输出使能** 0 = 定时器 1 输出禁止 1 = 定时器 1 从T1引脚输出使能 注意：仅当操作在定时器模式，定时器1输出应该使能
6	T0OE	**定时器 0 输出使能** 0 = 定时器 0 输出禁止 1 = 定时器 0 从T0引脚输出使能 注意：仅当操作在定时器模式，定时器0输出应该使能

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。Gate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。.

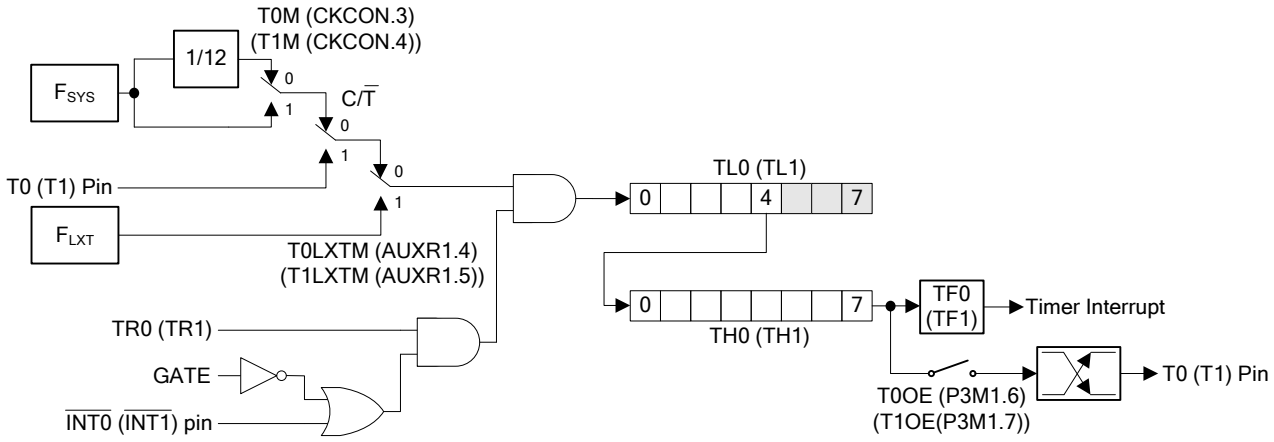


图 9.1. 定时器/计数器0及1模式0

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。

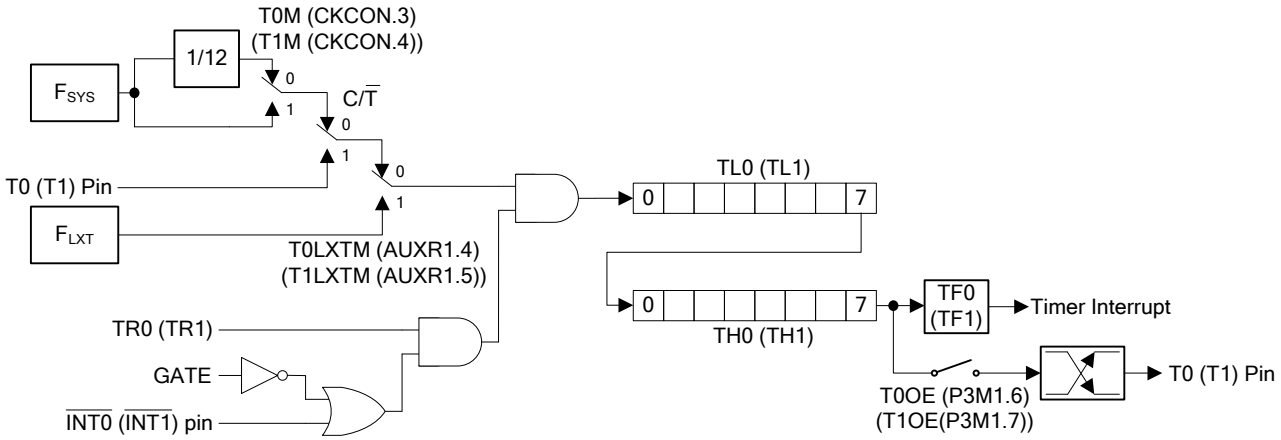


图 9.2. 定时器/计数器0及1模式1

9.3 模式 2 (8位自动重装载定时器)

模式2下定时器/计数器为自动重装载模式。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。当TRx位置1，GATE位及$\overline{INT0}$ ($\overline{INT1}$)配置正确时，计数器开始计数。GATE位及$\overline{INT0}$ ($\overline{INT1}$) 引脚的功能与模式0和1相同。

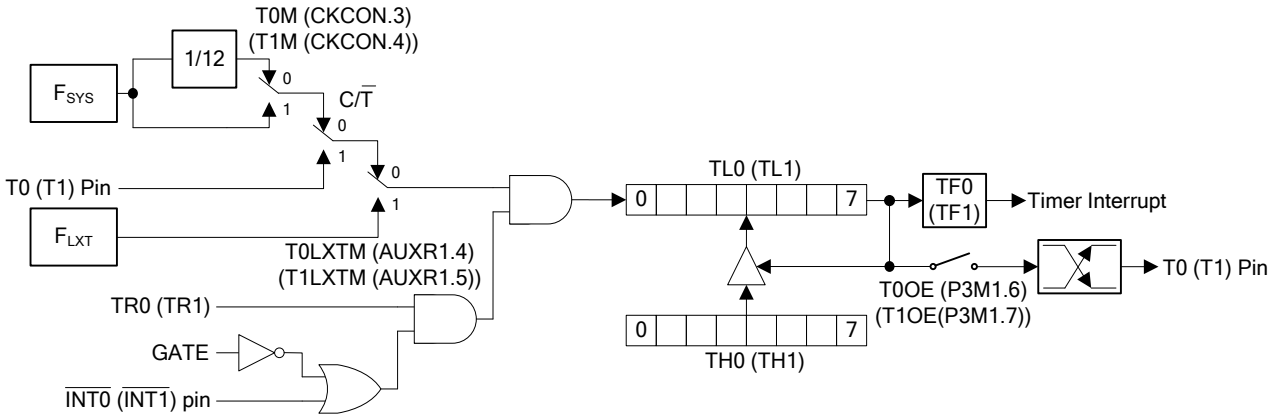


图 9.3. 定时器/计数器0及1模式2

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、$\overline{\text{INT0}}$和TF0。TL0也可以用来对T0 脚上的1到0 跳变计数，由C/$\overline{\text{T}}$ (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$\overline{\text{INT1}}$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。

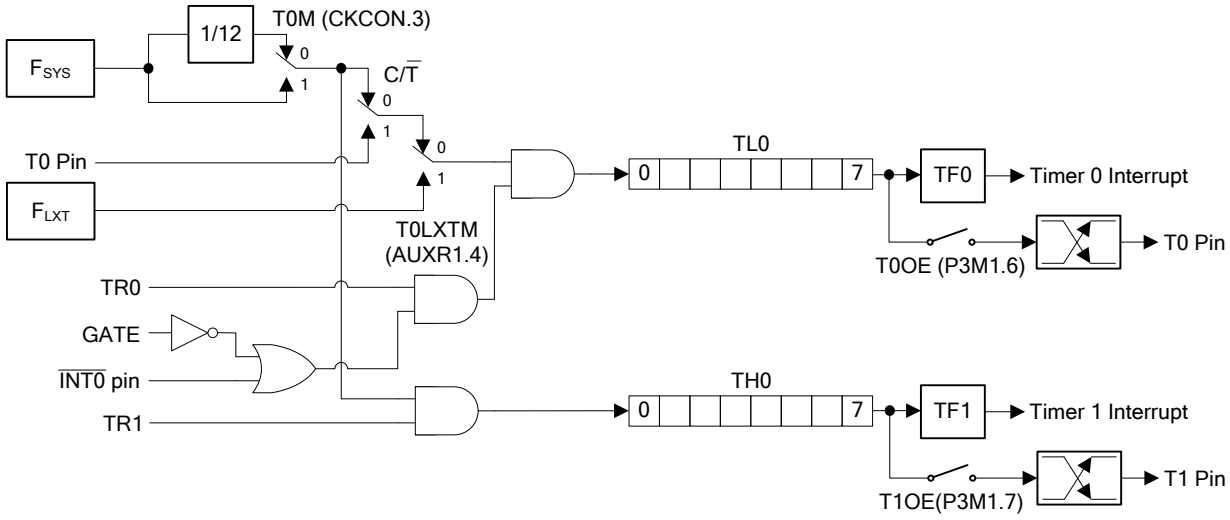


图 9.4. 定时器/计数器0模式3

10. 定时器 2A/2B/2C/2D

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。

10.1 自动重载模式

在自动重载模式，R2AH 和 R2AL的内容结合为一个16位的重载值，决定定时器2A的下溢速率。在决定预分频值和填合适值到R2AH和R2AL之后，用户可以设置TR2A (T2CON.0)来开始计数。在TR2A置位之后，R2AH 和 R2AL被加载到内部16位计数器中，并开始向下计数。当计数器溢出，TF2A (T2CON.4)由硬件置位为1，并使R2AH 和 R2AL得内容再一次被加载到内部16位计数器中。如果ET2A (EIE.2)置位为1，定时器2A中断服务程序会被执行。进入中断程序之后，TF2A由硬件自动清零。有两个互补输出引脚可用，在定时器下溢的时候会取反引脚。该功能通过T2OE寄存器中的T2AOE1 和T2AOE2控制位来使能。在第一次下溢的时候，T2AO1 是逻辑1优先。向下计数溢出的周期公式如下：

$$\text{周期} = \frac{256 \times R2AH + R2AL + 1}{F_{SYS} / \text{预分频}}.$$

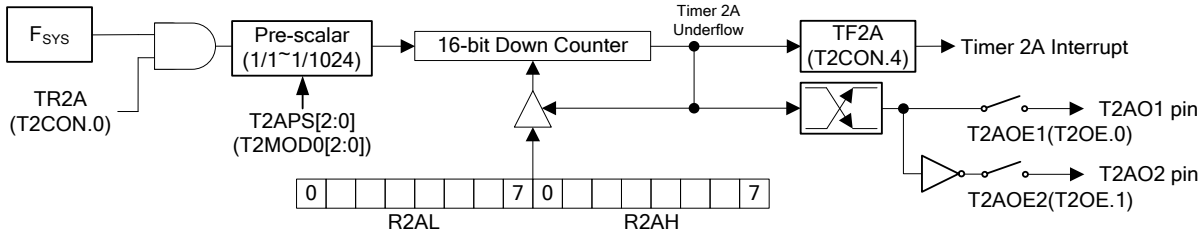


图 10.1. 定时器 2A 自动重装载模式框图

10.2 PWM 模式

在PWM模式，下数计数器配置为8位宽度。当下数到0时，R2AH 和 R2AL被交替地重载到计数器。R2AH数据定义PWM占空比高电平宽度，R2AL定义占空比低电平宽度。T2MOD0 和 T2MOD1中设置合适的预分频值之后，用户可以设置TR2A (T2CON.0)来开始PWM输出。同时R2AH加载到内部8位计数器，且计数器开始向下计数。在下次计数到0之前，R2AH重新载入且T2AO1回复逻辑1，一旦数到0，

R2AL被加载，且T2AO1取反输出逻辑0。定时器这样连续地运行产生连续的PWM波形。另外一个输出引脚T2AO2的状态和T2AO1相反。每次数到0的事件发生，TF2A会置1来请求定时器2A中断。在PWM模式，TF2A需要通过软件清除。PWM输出占空比和周期公式如下：

PWM 占空比高电平: $\frac{R2AH}{R2AH + R2AL}$.

PWM 占空比低电平: $\frac{R2AL}{R2AH + R2AL}$.

PWM 周期: $\frac{R2AH + R2AL}{F_{SYS} / \text{预分频}}$.

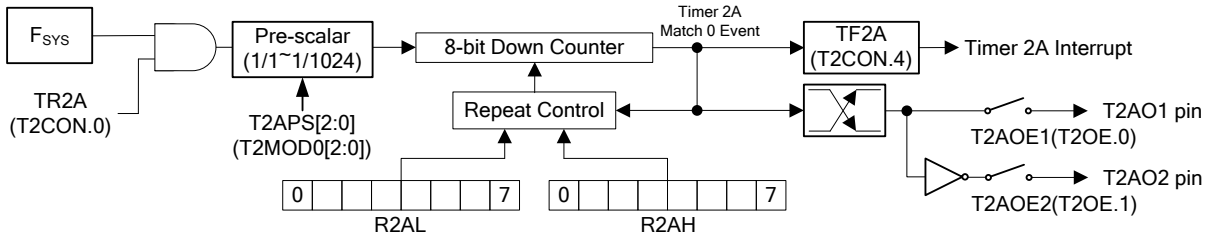


图 10.2. 定时器 2A PWM 模块框图

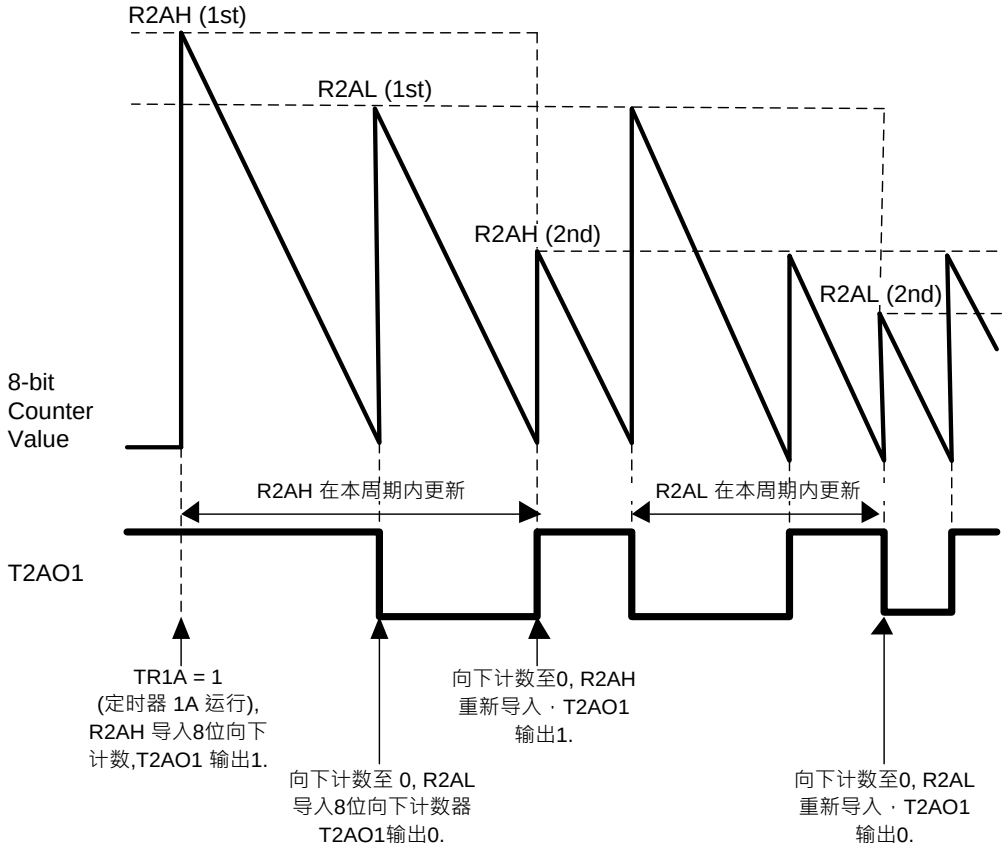


图 10.3. 定时器 2A PWM 波形

10.3 定时器2控制寄存器

T2CON – 定时器 2 控制寄存器 (可位寻址)

7	6	5	4	3	2	1	0
TF2D	TF2C	TF2B	TF2A	TR2D	TR2C	TR2B	TR2A
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: C8H　　复位值: 0000 0000b

位	名称	描述
7	TF2D	**定时器 2D 标志** 当定时器2D向下计数溢出时，该位置1。在自动重载模式，当程序执行定时器2D中断服务程序，TF2D由硬件自动清除。该位可以通过软件置位或清零。
6	TF2C	**定时器 2C 标志** 当定时器2C向下计数溢出时，该位置1。在自动重载模式，当程序执行定时器2C中断服务程序，TF2C由硬件自动清除。该位可以通过软件置位或清零。
5	TF2B	**定时器 2B 标志** 当定时器2B向下计数溢出时，该位置1。在自动重载模式，当程序执行定时器2B中断服务程序，TF2B由硬件自动清除。该位可以通过软件置位或清零。

位	名称	描述
4	TF2A	**定时器 2A 标志** 当定时器2A向下计数溢出时，该位置1。在自动重载模式，当程序执行定时器2A中断服务程序，TF2A由硬件自动清除。该位可以通过软件置1或清零。
3	TR2D	**定时器 2D 运行控制** 0 = 定时器2D停止并复位 1 = 定时器2D开始运行 注意在自动重载模式，重载寄存器R2DH 和 R2DL仅当定时器2D停止(TR2D= 0)才可以被写。如果在TR2D为1时写R2DH 或 R2DL，结果是不可预知的。
2	TR2C	**定时器 2C 运行控制** 0 = 定时器2C停止并复位 1 = 定时器2C开始运行 注意在自动重载模式，重载寄存器R2CH 和 R2CL仅当定时器2C停止(TR2C = 0)才可以被写。如果在TR2C为1时写R2CH 或 R2CL，结果是不可预知的。
1	TR2B	**定时器 2B 运行控制** 0 = 定时器2B停止并复位 1 = 定时器2B开始运行 注意在自动重载模式，重载寄存器R2BH 和 R2BL仅当定时器2B停止(TR2B= 0) 才可以被写。如果在TR2B为1时写R2BH 或 R2BL，结果是不可预知的。
0	TR2A	**定时器 2A 运行控制** 0 = 定时器2A停止和复位 1 = 定时器2A开始运行 注意在自动重载模式，重载寄存器R2AH 和 R2AL仅当定时器2A停止(TR2A = 0)才可以被写。如果在TR2A为1时写R2AH 或 R2AL，结果是不可预知的。

T2MOD0 – 定时器 2 模式 0

7	6	5	4	3	2	1	0
T2BM	T2BPS[2:0]			T2AM	T2APS[2:0]		
读/写	读/写			读/写	读/写		

地址: C9H　　复位值: 0000 0000b

位	名称	描述
7	T2BM	**定时器 2B 模式** 该位选择定时器2B的操作模式 0 = 自动重载模式. 1 = PWM 模式. 注意当TR2B为1时改变该位可能会引起不可预知的结果。

位	名称	描述
6:4	T2BPS[2:0]	**定时器 2B 预分频** 这些位决定定时器2B的时钟分频范围。 000 = 1/1. 001 = 1/2. 010 = 1/8. 011 = 1/16. 100 = 1/64. 101 = 1/128. 110 = 1/512. 111 = 1/1024. 注意当TR2B为1时，改变这些位可能导致不可预知的结果。
3	T2AM	**定时器 2A 模式** 该位选择定时器2A的操作模式 0 = 自动重载模式. 1 = PWM 模式. 注意当TR2A为1时改变该位可能会引起不可预知的结果。
2:0	T2APS[2:0]	**定时器 2A 预分频** 这些位决定定时器2A的时钟分频范围。 000 = 1/1. 001 = 1/2. 010 = 1/8. 011 = 1/16. 100 = 1/64. 101 = 1/128. 110 = 1/512. 111 = 1/1024. 注意当TR2A为1时，改变这些位可能导致不可预知的结果。

T2MOD1 – 定时器 2 模式 1

7	6	5	4	3	2	1	0
T2DM	T2DPS[2:0]			T2CM	T2CPS[2:0]		
读/写	读/写			读/写	读/写		

地址: CAH　　复位值: 0000 0000b

位	名称	描述
7	T2DM	**定时器 2D 模式** 该位选择定时器2D的操作模式 0 = 自动重载模式. 1 = PWM 模式. 注意当TR2D为1时改变该位可能会引起不可预知的结果。

位	名称	描述
6:4	T2DPS[2:0]	**定时器 2D 预分频** 这些位决定定时器2D的时钟分频范围。 000 = 1/1. 001 = 1/2. 010 = 1/8. 011 = 1/16. 100 = 1/64. 101 = 1/128. 110 = 1/512. 111 = 1/1024. 注意当TR2D为1时，改变这些位可能导致不可预知的结果。
3	T2CM	**定时器 2C 模式** 该位选择定时器2C的操作模式 0 = 自动重载模式. 1 = PWM 模式. 注意当TR2C为1时改变该位可能会引起不可预知的结果。
2:0	T2CPS[2:0]	**定时器 2C 预分频** 这些位决定定时器2C的时钟分频范围。 000 = 1/1. 001 = 1/2. 010 = 1/8. 011 = 1/16. 100 = 1/64. 101 = 1/128. 110 = 1/512. 111 = 1/1024. 注意当TR2C为1时，改变这些位可能导致不可预知的结果。

T2OE – 定时器 2 输出使能

7	6	5	4	3	2	1	0
T2DOE2	T2DOE1	T2COE2	T2COE1	T2BOE2	T2BOE1	T2AOE2	T2AOE1
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: CBH　　复位值: 0000 0000b

位	名称	描述
7	T2DOE2	**定时器 2D 输出使能 2** 0 = T2DO2 输出禁止. 1 = T2DO2 输出使能
6	T2DOE1	**定时器 2D 输出使能 1** 0 = T2DO1 输出禁止. 1 = T2DO1 输出使能
5	T2COE2	**定时器 2C 输出使能 2** 0 = T2CO2 输出禁止. 1 = T2CO2 输出使能
4	T2COE1	**定时器 2C 输出使能 1** 0 = T2CO1 输出禁止. 1 = T2CO1 输出使能

位	名称	描述
3	T2BOE2	**定时器 2B 输出使能 2** 0 = T2BO2 输出禁止. 1 = T2BO2 输出使能
2	T2BOE1	**定时器 2B 输出使能 1** 0 = T2BO1 输出禁止. 1 = T2BO1 输出使能
1	T2AOE2	**定时器 2A 输出使能 2** 0 = T2AO2 输出禁止. 1 = T2AO2 输出使能
0	T2AOE1	**定时器 2A 输出使能 1** 0 = T2AO1 输出禁止. 1 = T2AO1 输出使能

R2AL – 定时器 2A 自动重装载数据低字节

7	6	5	4	3	2	1	0
R2AL[7:0]							
读/写							

地址: CCH　　复位值: 0000 0000b

位	名称	描述
7:0	R2AL[7:0]	**定时器 2A 重载低字节** 在自动重载模式，它保持定时器2A重载值的低字节。在PWM模式，它保持低电平的占空比值

R2AH – 定时器 2A 自动重装载高字节

7	6	5	4	3	2	1	0
R2AH[7:0]							
读/写							

地址: CDH　　复位值: 0000 0000b

位	名称	描述
7:0	R2AH[7:0]	**定时器 2A 重载高字节** 在自动重载模式，它保持定时器2A重载值的高字节。在PWM模式，它保持高电平的占空比值

R2BL – 定时器2B自动重装载数据低字节

7	6	5	4	3	2	1	0
R2BL[7:0]							
读/写							

地址: CEH　　复位值: 0000 0000b

位	名称	描述
7:0	R2BL[7:0]	**定时器 2B 重载低字节** 在自动重载模式，它保持定时器2B重载值的低字节。在PWM模式，它保持低电平的占空比值

R2BH – 定时器 2B自动重装载高字节

7	6	5	4	3	2	1	0
R2BH[7:0]							
读/写							

地址: CFH　　复位值: 0000 0000b

位	名称	描述
7:0	R2BH[7:0]	**定时器 2B 重载低字节** 在自动重载模式，它保持定时器2B重载值的高字节。在PWM模式，它保持高电平的占空比值

R2CL –定时器 2C重载低字节

7	6	5	4	3	2	1	0
R2CL[7:0]							
读/写							

地址: D4H　　复位值: 0000 0000b

位	名称	描述
7:0	R2CL[7:0]	**定时器 2C 重载低字节** 在自动重载模式，它保持定时器2C重载值的低字节。在PWM模式，它保持低电平的占空比值

R2CH –定时器 2C 重载高字节

7	6	5	4	3	2	1	0
R2CH[7:0]							
读/写							

地址: D5H　　复位值: 0000 0000b

位	名称	描述
7:0	R2CH[7:0]	**定时器 2C 重载低字节** 在自动重载模式，它保持定时器2C重载值的高字节。在PWM模式，它保持高电平的占空比值

R2DL –定时器 2D 重载低字节

7	6	5	4	3	2	1	0
R2DL[7:0]							
读/写							

地址: D6H　　复位值: 0000 0000b

位	名称	描述
7:0	R2DL[7:0]	**定时器 2D 重载低字节** 在自动重载模式，它保持定时器2D重载值的低字节。在PWM模式，它保持低电平的占空比值

R2DH – 定时器 2D 自动重装载高字节

7	6	5	4	3	2	1	0
R2DH[7:0]							
读/写							

地址: D7H　　复位值: 0000 0000b

位	名称	描述
7:0	R2DH[7:0]	**定时器 2D 重载低字节** 在自动重载模式，它保持定时器2D重载值的高字节。在PWM模式，它保持高电平的占空比值

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 也是UART波特率的时钟源。细节请看 章节 錯誤! 找不到參照來源。"錯誤! 找不到參照來源。".

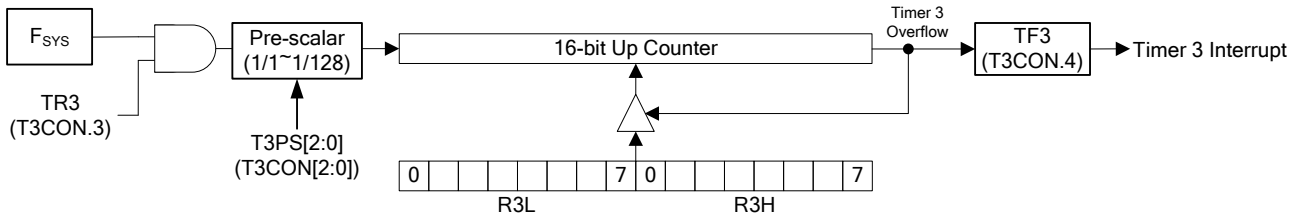


图 11.1. 定时器 3结构框图

T3CON – 定时器 3 控制

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.

R3L – 定时器 3 重载低字节

7	6	5	4	3	2	1	0
R3L[7:0]							
R/W							

地址: C5H　　复位值: 0000 0000b

位	名称	描述
7:0	R3L[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

R3H – 定时器 3 重载高字节

7	6	5	4	3	2	1	0
R3H[7:0]							
R/W							

地址: C6H　　复位值: 0000 0000b

位	名称	描述
7:0	R3H[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. 看门狗定时器 (WDT)

N76E616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。

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]	**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 (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: 见表 6–2.

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。

看门狗超时间隔有公式 $\frac{1}{F_{LIRC} \times clockdividerscalar} \times 64$ 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. 在不同分频下的看门狗超时间隔

WDPS.2	WDPS.1	WDPS.0	时钟预分频	看门狗定时器定时间隔 (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。.

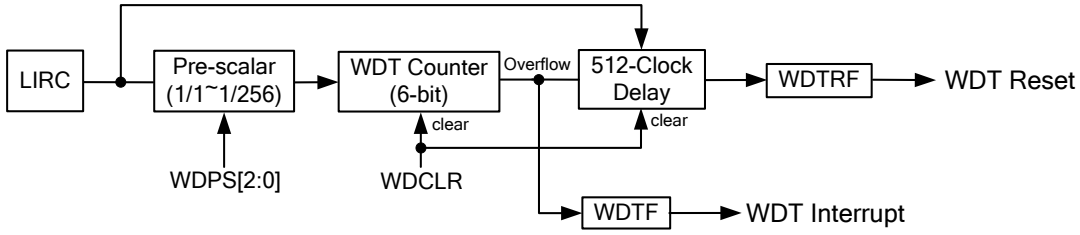


图 12.1. WDT作为超时复位定时器

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。

注： WDT 计数器在如下状态中，计数值和预分频值会自动清0：

（1） 进入睡眠模式或掉电模式时，以及从睡眠模式或掉电模式中唤醒时。

（2） 任何复位发生后。

12.2 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。

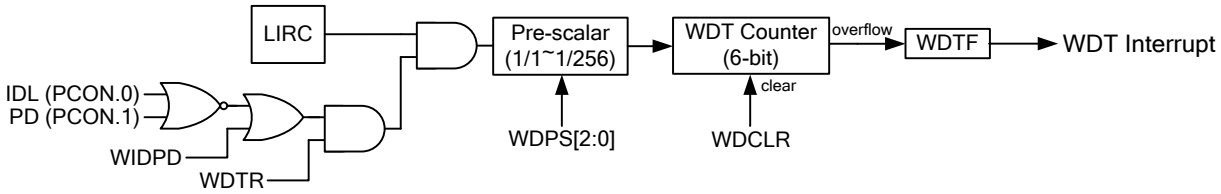


图 12.2. 看门狗定时器结构框图

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mA级，为了达到系统需要保持在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。N76E616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。

实现这一功能的示例代码如下：

```
        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)

N76E616有一个专门的自唤醒定时器（WKT），它用于低功耗模式下的周期唤醒或通用的定时器。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。

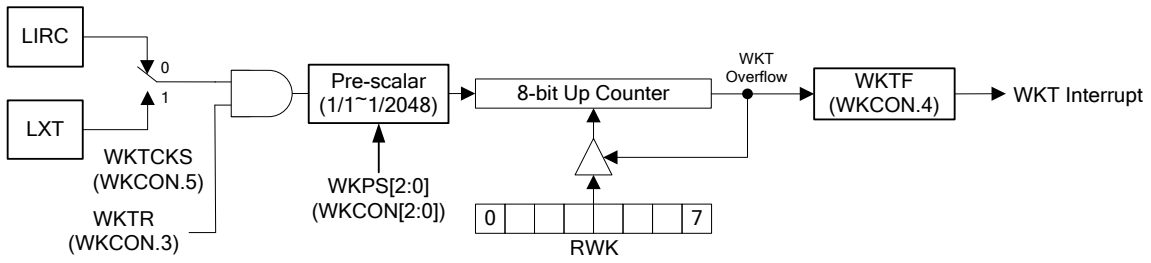


图 13.1. 自唤醒定时器结构框图

WKCON – 自唤醒定时器控制寄存器

7	6	5	4	3	2	1	0
-	-	WKTCKS	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	WKTCKS	**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** 用以存储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)

N76E616 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。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。这两个缓存共享一个寄存器地址 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P2.1 及 P2.2 (RXD 及 TXD) 或者 P5.6 及 P5.7 (RXD_1 及 TXD_1) 必须先送1。

SCON – 串口控制寄存器 (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	**模式选择** SMOD0 (PCON.6) = 0: 详细描述见 表 14–1.. SMOD0 (PCON.6) = 1: SM0/FE 帧错，软件清0. 0 = 无帧错. 1 = 检测到帧错.
6	SM1	
5	SM2	**多机通信模式** 此位功能依赖串口0的模式 . 模式0: 此位选择波特律是 F_{SYS}/12 或 F_{SYS}/2. 0 = 波特率是 F_{SYS}/12 标准8051兼容模式. 1 = 波特率是 F_{SYS}/2. 模式1:. 0 =总是接收数据，不管停止位是什么电平 1 = 只有接收到有效停止位，并且地址匹配时(包括广播地址)，才接收数据. 模式2 或 3: 多机通信功能. 0 = 无论第9位是0是1，总是接收 1 = 只有第9位是1, 且地址匹配时才接收(包括广播地址).
4	REN	**接收使能** 0 = 串口0禁接收. 1 = 工作模式为1,2,3时，使能接收. 工作模式为0时，在条件 REN = 1 和 RI = 0时接收被初始化.

位	名称	描述
3	TB8	**发送的第9位** 串口0在模式2，3时，此位是发送的第9位，模式0,1不用此位.
2	RB8	**收到的第9位** 串口0模式2,3时，此位是收到的第9位。模式 1 时, RB8 是停止位电平。模式0不用此位.
1	TI	**发送中断标志** 串口0，模式0发完第8位，其它模式发完最后一位时，硬件置1此位。SM2 被限制的例外。若使能了中断，此位置1会去执行中断代码。此位必须软件清0。
0	RI	**接收中断标志** 串口0，模式0收完第8位，其它模式收到停止位时，硬件置1此位。SM2 被限制的例外。若使能了中断，此位置1会去执行中断代码。此位必须软件清0。

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**
6	SM1_1	SMOD0_1 (T3CON.6) = 0: 详见 表 14–2. SMOD0_1 (T3CON.6) = 1: SM0_1/FE_1 帧错，软件清0. 0 = 无帧错. 1 = 检测到帧错.
5	SM2_1	**多机通信模式** 此位功能依赖串口0的模式 . 模式0: 无效 模式1:. 0 =总是接收数据，不管停止位是什么电平 1 = 只有接收到有效停止位，并且地址匹配时(包括广播地址)，才接收数据. 模式2 或 3: 多机通信功能. 0 = 无论第9位是0是1，总是接收 1 = 只有第9位是1, 且地址匹配时才接收(包括广播地址).
4	REN_1	**接收使能位** 0 = 接收禁止. 1 = 串口1,对于模式1,2,3来说是接收使能。对于模式0， REN_1 = 1 和 RI_1 = 0 时初始化
3	TB8_1	**发送的第9位** 串口1模式2,3发送的第9位，模式0,1不用此位.
2	RB8_1	**收到的第9位** 串口0模式2,3时，此位是收到的第9位。模式 1 时, RB8 是停止位电平。SM2_1 被限制的例外。模式0不用此位.
1	TI_1	**发送中断标志** 串口0，模式0发完第8位，其它模式发完最后一位时，硬件置1此位。若使能了中断，此位置1会去执行中断代码。此位必须软件清0。
0	RI_1	**接收中断标志** 串口1，模式0收完第8位，其它模式收到停止位时，硬件置1此位。SM2 _1被限制的例外。 若使能了中断，此位置1会去执行中断代码。此位必须软件清0。.

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: 见表 6–2.

位	名称	描述
7	SMOD	**串口0双倍波特律使能位** 串口0 模式2，或模式1,3以 定时器 1 溢出做波特律 时钟时，此位置1，波特律加倍。详见 表 14–1..
6	SMOD0	**串口0帧错标志访问使能** 0 = SCON.7 是 SM0 位. 1 = SCON.7 是 FE 位.

T3CON – 定时器 3 控制

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	**串口1双倍波特律使能位** 串口1 在模式2时，此位置1波特和律加倍。详见 表 14–2..
6	SMOD0_1	**串口1帧错标志访问使能** 0 = SCON_1.7 是 SM0_1 位. 1 = SCON_1.7 是 FE_1 位.

表 14–1. 串口0模式描述

模式	SM0	SM1	描述	位长	波特率
0	0	0	同步	8	F_{SYS} 除以 12 或 2[1]
1	0	1	异步	10	定时器 1/定时器3 溢出率除以 32 或16[2]
2	1	0	异步	11	F_{SYS} 除以32 或 64[2]
3	1	1	异步	11	定时器1/定时器3 溢出率除以 32 or 或 16[2]

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

表 14–2. 串口1模式描述

Mode	SM0	SM1	描述	位长	波特率
0	0	0	同步	8	F_{SYS} 除以 12 或 2[1]
1	0	1	异步	10	定时器3溢出率除以 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 除以 16

[1] 当 SM2_1 (SCON_1.5) 为 1 时.

[2] 当 SMOD_1 (T3CON.7)为 1 时.

SBUF – 串口0数据缓存

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

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

7	6	5	4	3	2	1	0
SBUF_1[7:0]							
R/W							

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

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 为 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

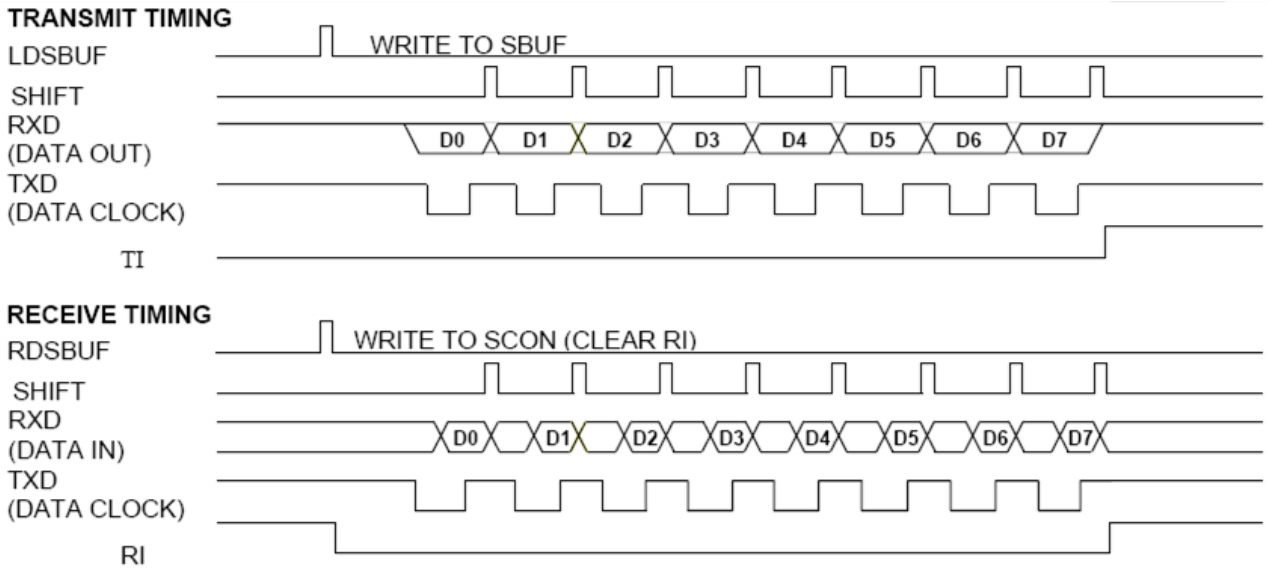


图 14.1. 串口 模式 0 时序图

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为全双工异步的方式工作。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.

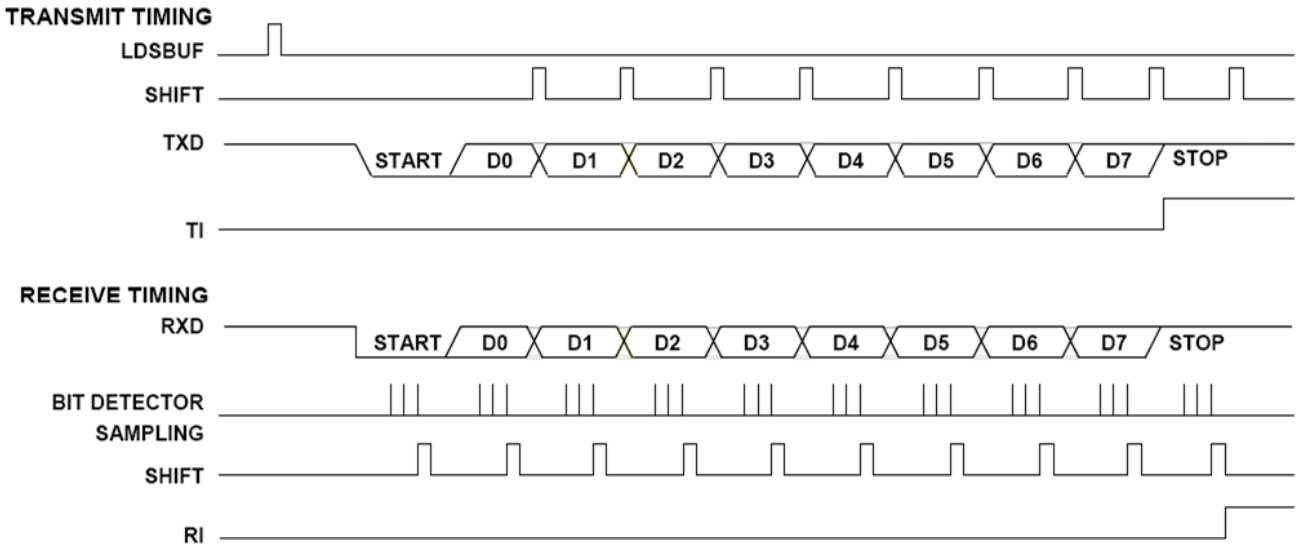


图 14.2. 串口 模式 1时序图

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. SM2 (SCON.5) = 0，或者SM2 = 1 时，收到的数据与配置的从机地址或广播地址一致且收到停止位为1. (详见 章节錯誤! 找不到參照來源。錯誤! 找不到參照來源。 和 章节 錯誤! 找不到參照來源。錯誤! 找不到參照來源。)

从上条件若满足，就把收到的数据存入SBUF，停止位电平写入RB8 (SCON.2)，并且RI 置1。若不不满足，数据不写入SBUF，RI 保持 0。接下来再次检测RXD的下降沿跳变，准备下一次接收。

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。

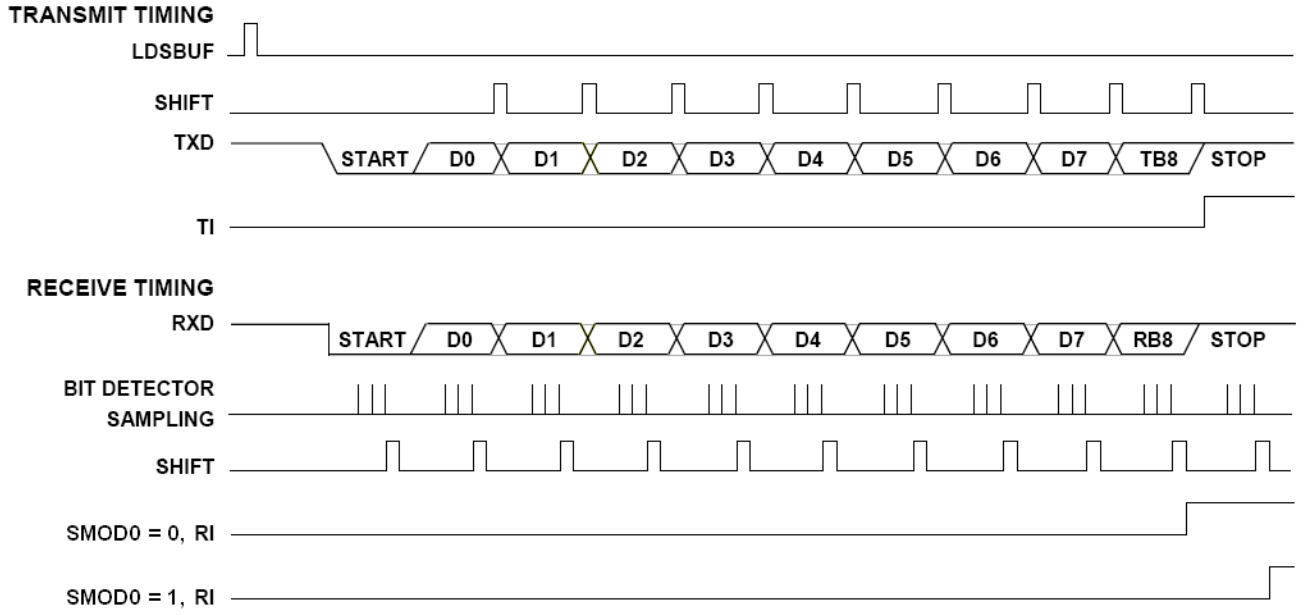


图 14.3. 串口 模式 2和3 时序图

写 SBUF开始发送，数据从TXD输出。先输出起始位，再出8位数据，接下来是TB8，然后是停止位。最后TI 置1表明发送结束。

REN 置1，接收准备，RXD的下沿开始接收，配置的波特率采样数据，在停止位中间，下列条件满足就把数据写入SBUF：

1. RI (SCON.0) = 0,

2. SM2 (SCON.5) = 0，或者 SM2 = 1时收到第九位为1并且地址匹配. (详见 章节**錯誤! 找不到參照來源。錯誤! 找不到參照來源。** 和 章节 **錯誤! 找不到參照來源。錯誤! 找不到參照來源。**)

若以上条件满足，收到的数据写入 SBUF ，第9位写入 RB8(SCON.2) 并且置1 RI。若条件不满足丢弃数据， RI 保持0。 然后检测RXD下沿，准备下次接收。

14.4 模式 3

模式3与模式2相同，只是模式3使用定时器1 溢出率做为波特率。时序见 图 14.3

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

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

地址: C4H　　复位值: 0000 0000b

位	名称	描述
5	BRCK	**串口0时钟源选择** 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. UART 波特率公式

串口模式	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 或 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]
	定时器 3 (仅供 UART0)	$\frac{2^{SMOD}}{32}\times\frac{F_{SYS}}{Pre-scale\times(65536-\{R3H,R3L\})}$ [5]
	定时器 3 (仅供 UART1)	$\frac{1}{16}\times\frac{F_{SYS}}{Pre-scale\times(65536-\{R3H,R3L\})}$ [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×RH3+RL3 . 当SMOD 为 1及与分频 1/1, {RH3,RL3} 不能设为 FFFFH.

表 14–4列举了由定时器1 产生的常用波特率。此种方式定时器1 配成重装载模式：SMOD (PCON.7) =0 且 T1M (CKCON.4) = 0. 表 14–5 列举了 串口0 用定时器3产生波特率的值. 若 SMOD = 0。 对于串口1 来说，相同的值，波特率会加倍。

表 14–4. 定时器1产生常用波特率

波特率 (bps)	振荡频率 (MHz)			
	3.6864	7.3728	11.0592	14.7456
	TH1 重载值			
57600	-	-	-	-
38400	-	-	-	FFH
19200	-	FFH	-	FEH
9600	FFH	FEH	FDH	FCH
4800	FEH	FCH	FAH	F8H
2400	FCH	F8H	F4H	F0H
1200	F8H	F0H	E8H	E0H
300	E0H	C0H	A0H	80H

表 14–5. 定时器3产生常用波特率

波特率 (bps)	振荡频率 (MHz)							
	3.6864	4	7.3728	8	11.0592	12	14.7456	16
	{R3H,R3L} 重载值							
115200	FFFFH	-	FFFEH	-	FFFDH	-	FFFCH	-
57600	FFFEH	-	FFFCH	-	FFFAH	-	FFF8H	-
38400	FFFDH	-	FFFAH	-	FFF7H	-	FFF4H	FFF3H
19200	FFFAH	-	FFF4H	FFF3H	FFEEH	-	FFE8H	FFE6H
9600	FFF4H	FFF3H	FFE8H	FFE6H	FFDCH	FFD9H	FFD0H	FFCCH
4800	FFE8H	FFE6H	FFD0H	FFCCH	FFB8H	FFB2H	FFA0H	FF98H
2400	FFD0H	FFCCH	FFA0H	FF98H	FF70H	FF64H	FF40H	FF30H
1200	FFA0H	FF98H	FF40H	FF30H	FEE0H	FEC8H	FE80H	FE5FH
300	FE80H	FE5FH	FD00H	FCBFH	FB80H	FB1EH	FA00H	F97DH

14.6 帧错检测

异步模式(模式 1, 2, 或 3.)有帧错检测功能。 检不到有效的停止位，就产生帧错。

帧错 FE 位在 SCON.7。正常情况下此位为 SM0。当使能了帧错功能 SMOD0 (PCON.6) = 1, 此位就是帧错 FE标志位了，实际上 SM0 和 FE 在不同的物理寄存器中。

通信若出现帧错，FE将被置1，后续数据无帧错 FE 位不会回零，一旦出现帧错 FE置1， FE位必须软件清0。注意，读写 FE，SMOD0 必须是置1的。

14.7 多机通信

N76E616 的串口模式2或3可以在一主多从的系统中发送多帧数据给一从机，且不打扰其它从机。使用这个功能必须配置 SM2 (SCON.5) = 1 以便收到一个字节时，若第9位为1就产生中断 (模式2第9位是停止位)。SM2 bit = 1时，第9位若是0就不产生中断。

主机给某一从机发数据时，先发一个地址字节——地址字节第9位是1，数据字节第9位是0。地址字节会让所有的从机都中断并检查是否与自己的从机地址一致，若一致就让SM2 =0并准备接收后续字节。地址不一致的从机 SM2 保持1，不会接收后续数据字节。

多机通信流程如下:

1. 主从机 串口都配置为模式 2 或 3.
2. 所有的从机 SM2 = 1.
3. 主机发送时序为:

    首字节：地址(第9位=1).

    后续字节: 数据(第9位=0).

4. 所有的从机接收到第9位为1的地址字节都会中断，并比较是否与自己的配置地址一致，若一致就让SM2=0，后续数据来了也会产生中断。地址字节不匹配的从机，不理会后续数据字节。
5. 地址匹配的从机，接收完数据后，再让SM2=1。

模式 0通信，SM2无效。模式1，若 SM2 = 1，有效的停止位才会让串口中断。

14.8 地址自动识别

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 –从机地址0

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

地址: A9H　　复位值: 0000 0000b

位	名称	描述
7:0	SADDR[7:0]	**从机地址0** 多机通讯时用作存储串口0的从机地址

SADEN – 从机地址掩码0

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

地址: B9H　　复位值: 0000 0000b

位	名称	描述
7:0	SADEN[7:0]	**从机地址0的掩码** 将该寄存器各位与从机地址0做与运算。这个寄存器为0对应的地址位，被忽略。

SADDR_1 – 从机地址1

7	6	5	4	3	2	1	0
SADDR_1[7:0]							
R/W							

地址: BBH 复位值: 0000 0000b

位	名称	描述
7:0	SADDR_1[7:0]	**从机地址1** 多机通讯时用作存储串口1的从机地址

SADEN_1 – 从机地址1的掩码

7	6	5	4	3	2	1	0
SADEN_1[7:0]							
R/W							

地址: BAH 复位值: 0000 0000b

位	名称	描述
7:0	SADEN_1[7:0]	**从机地址1的掩码** 与从机地址1做与运算，这个寄存器为0对应的地址位，被忽略。.

以下是从机地址运算示例.

从机0:

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

从机1:

```
地址 = 11000000b
掩码 = 11111110b
结果 = 1100000Xb
```

上面例子 SADDR 相同， SADEN 不同，形成不同地址匹配。从机 0 忽略地 址位1，从机1忽略地址位0。发往地址 1100 0010 的数据从机0会接收，从机1拒绝。发往地址11000001 的数据从机0拒绝而从机1接收。发往地址 11000000的数据，从机0从机1都接收。

下例显示选择从机1及从机2,忽略从机0范例

从机e 0:

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

从机 1:

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

从机 2:

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

上面例子，地址低三位区分不同匹配。从机 0 拥有唯一地址 11100110b。从机 1 拥有唯一地址 11100101b。 从机 2 拥有唯一地址 11100011b。若把数据发给从机0，1排除从机2，就发往地址 11100100b。

从机的广播地址可以通过 SADDR 与 SADEN 的或运算求得，结果为0的位，是忽略位。

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

采用”无关位”(don’t care bit) 可以使得广播地址呼叫更灵活，在通常状态下，全部定义为“无关位”即可，所以广播地址一般定义为 FFH.

复位后，SADDR 和 SADEN 的值为 00H。这样“Given”地址全部采用“无关位”广播地址成为 XXXXXXXXb (全部 “don’t care” bits)。这样可以确保串口响应全部地址，同时也与标准传统8051微处理器相同，不支持自动地址识别。

15. 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) 。

15.1 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：I²C总线上有一个节点输出0，总线上就是0电平，只有所有节点全输出1，总线上才是高电平，即通过外接上拉电阻把电平拉高。N76E616, 在设置I2CEN (I2CON.6)使能I2C功能之前，必须把 P2.3及 P2.4的输出锁存在逻辑1的状态。

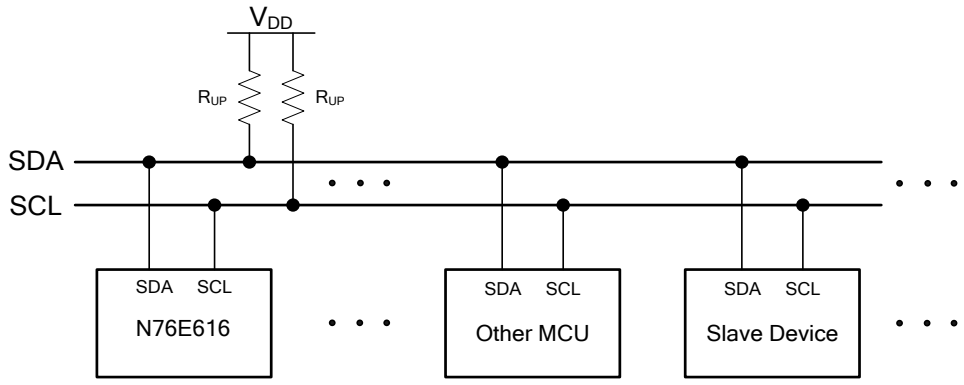


图 15.1. I²C 总线互联

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。

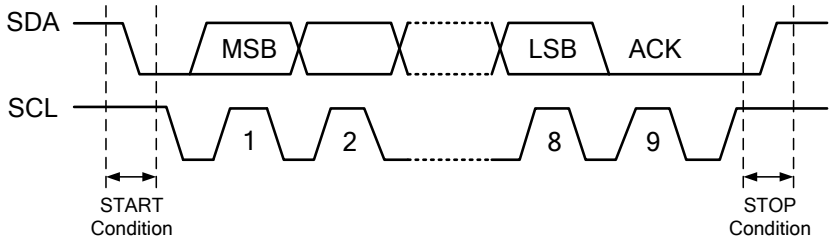


图 15.2. I²C Bus Protocol

15.1.1 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。

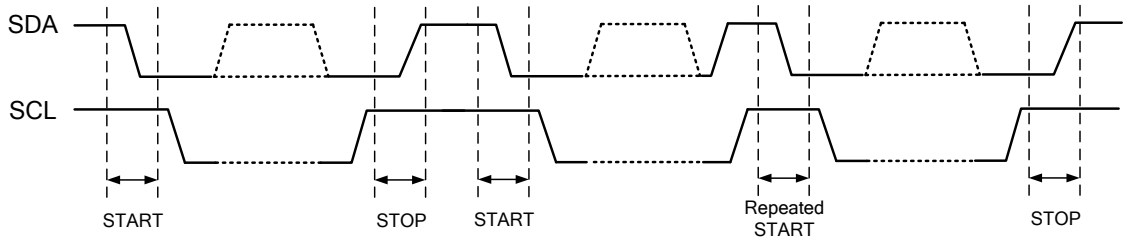


图 15.3. 起始(START)信号, 重复起始信号以及停止(STOP)条件

15.1.2 7位地址和数据格式

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。

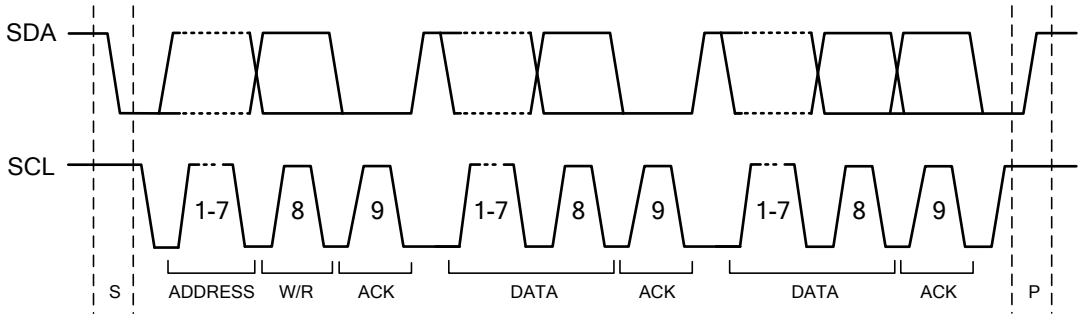


图 15.4. 一组 I^2C 传输数据格式

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。

15.1.3 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，将自身切换到未寻址从机模式，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寻址从机模式，并释放SDA线，以便主机直接发停止位STOP 或重起始位START。

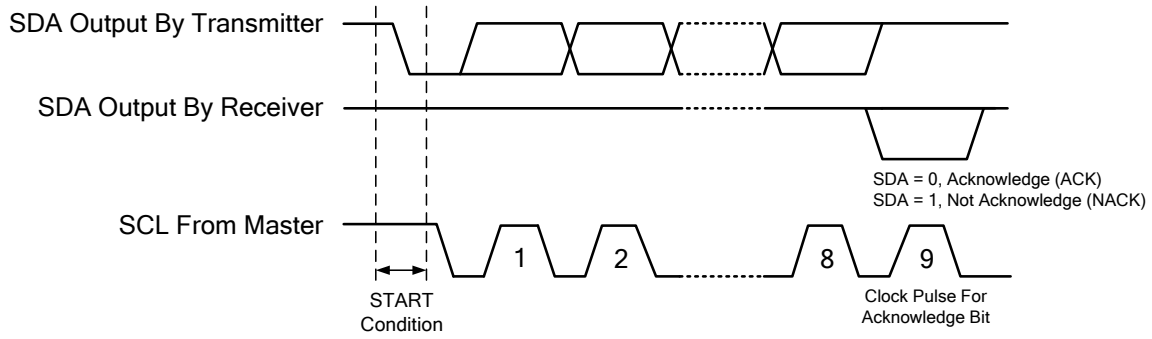


图 15.5. 应答位

15.1.4 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。

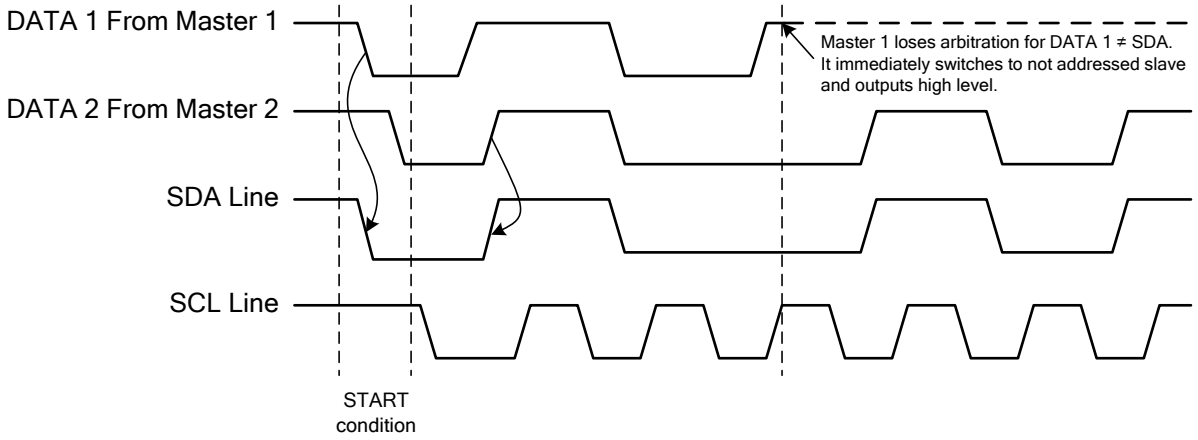


图 15.6. 图 15.7. 两台主机仲裁过程

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。

15.2 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
7	-	**保留**
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，且重新被主机寻址。 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。
1:0	-	**保留位**

I2STAT – I^2C 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]	**I^2C 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 – I^2C 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]	**I^2C 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。主机需要寻址该从机需在START之后写入该值。如果AA为1，该从机响应主机，成为被寻址从机。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.

15.3 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。

15.3.1 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。

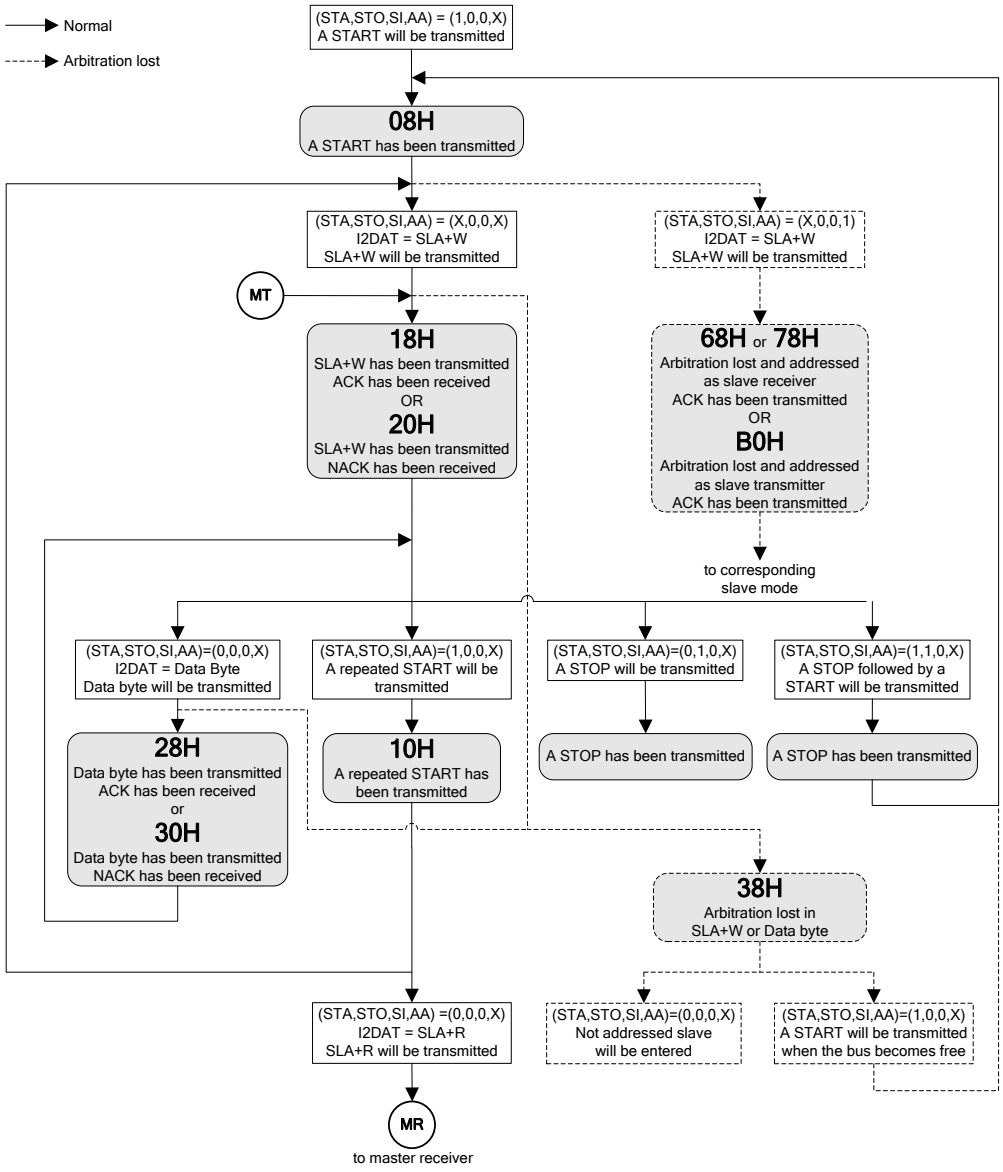


图 15.8. 主机发送模式流程图

15.3.2 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。

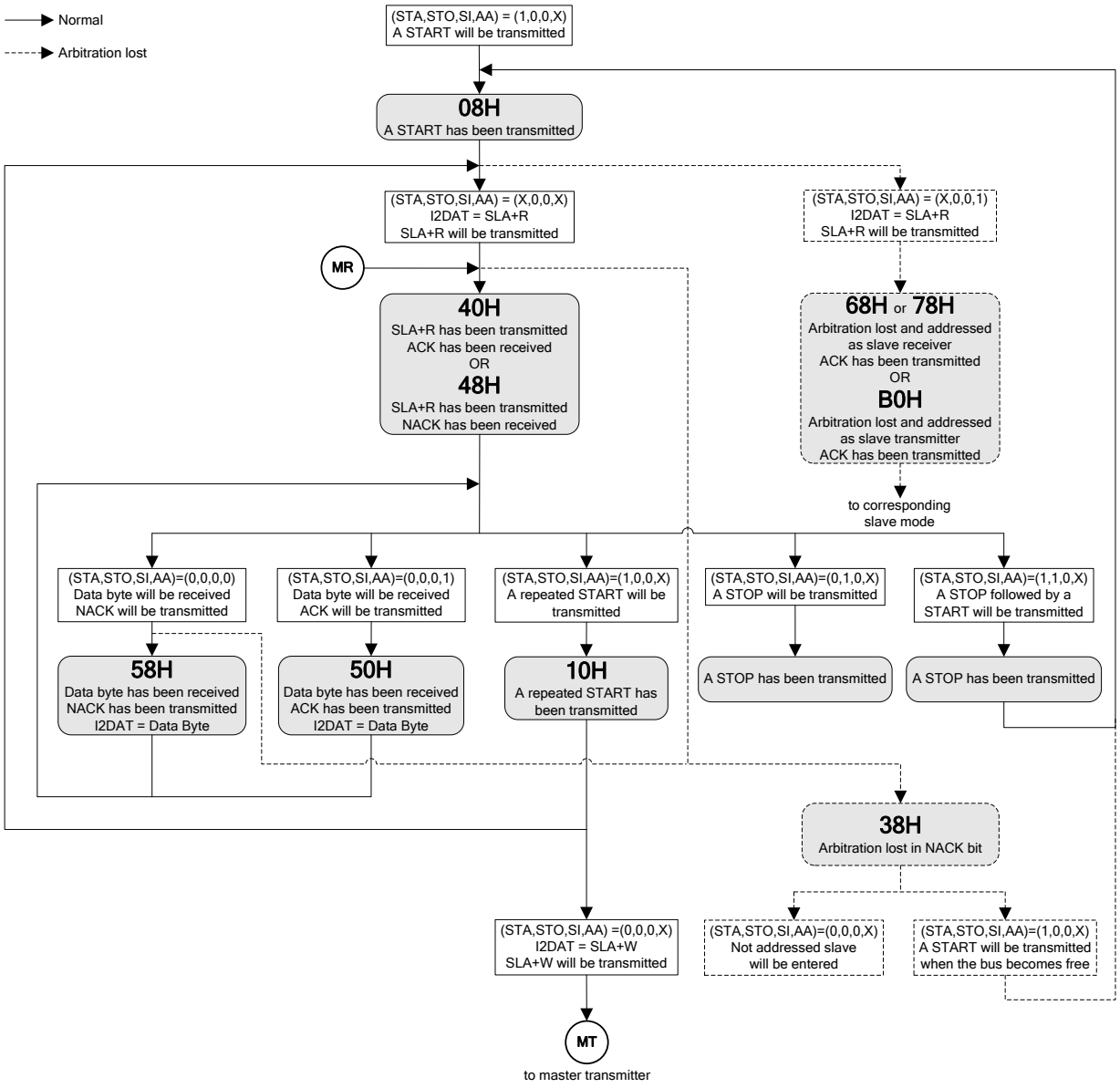


图 15.9. 主机接收模式流程图

15.3.3 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寻址。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，从机也将转为未寻址从机，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。

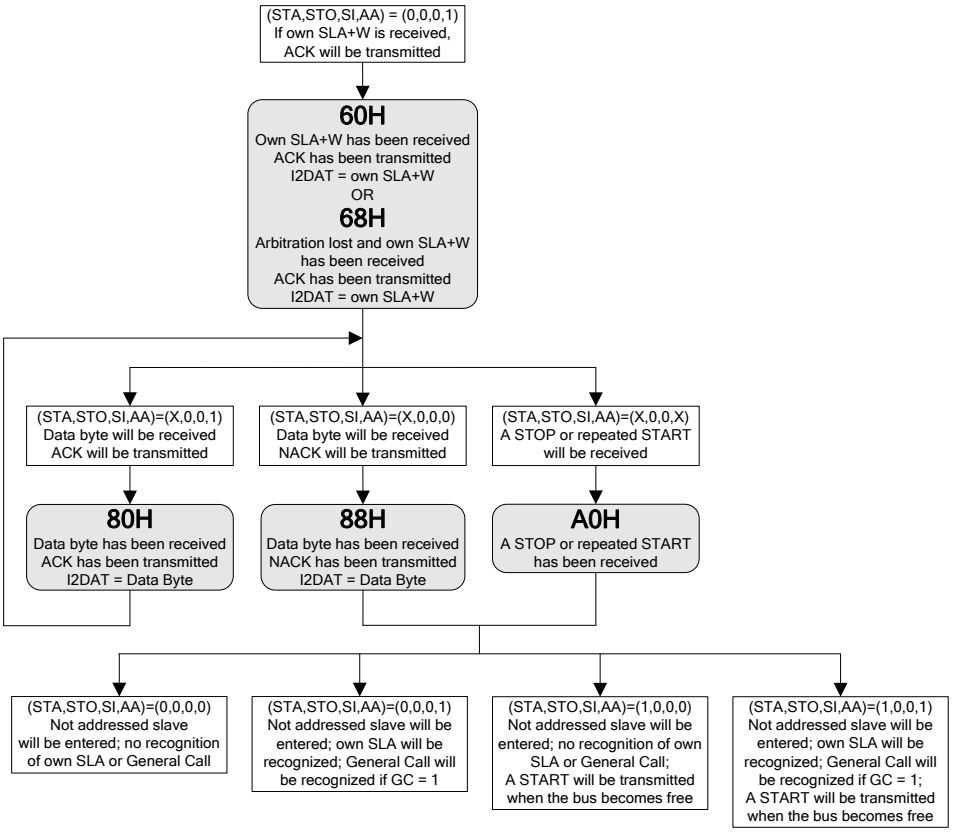


图 15.10. 从机接收模式流程图

15.3.4 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)寻址。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寻址，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。并变为未寻址从机。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，并将自身转为未寻址从机。

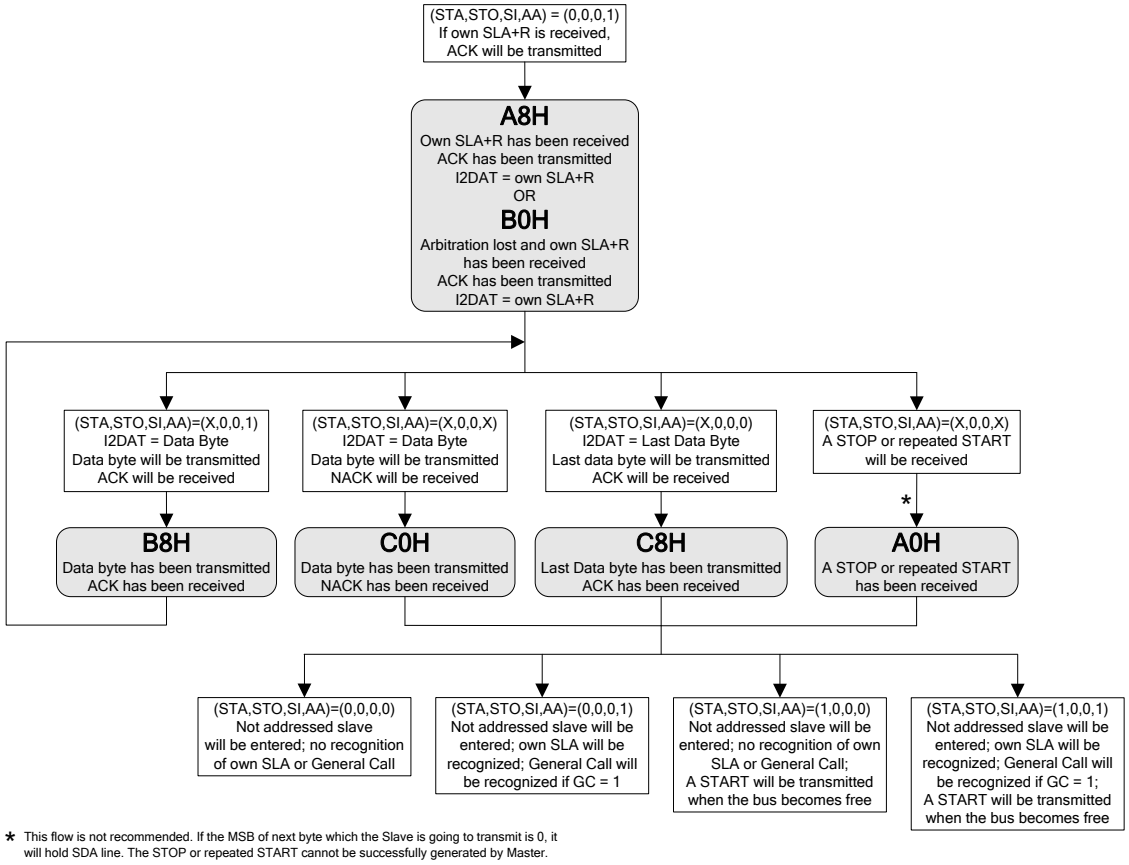


图 15.11. 从机发送模式

15.3.5 广播呼叫模式

广播呼叫模式是从机接收的一种特殊模式，寻址方式为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。

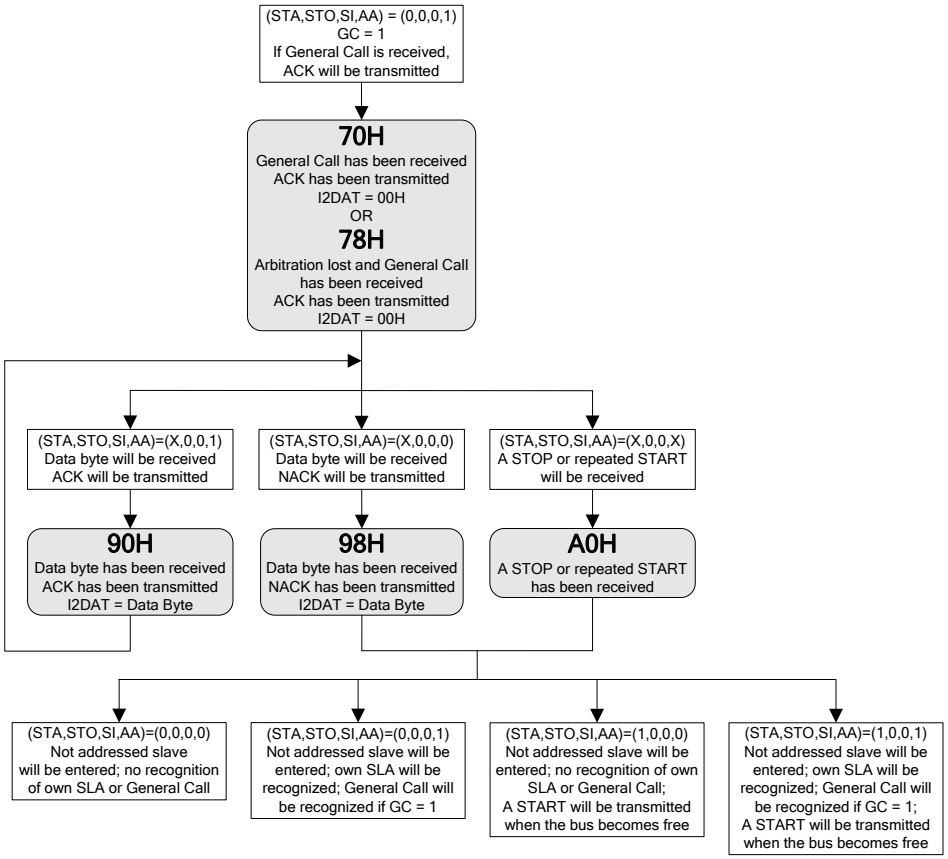


图 15.12. 广播呼叫模式流程图

15.3.6 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寻址从机模式，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。

15.4 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，请确认该状态不会发生，避免无法响应。

```
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
```

15.5 I²C 超时

N76E616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。

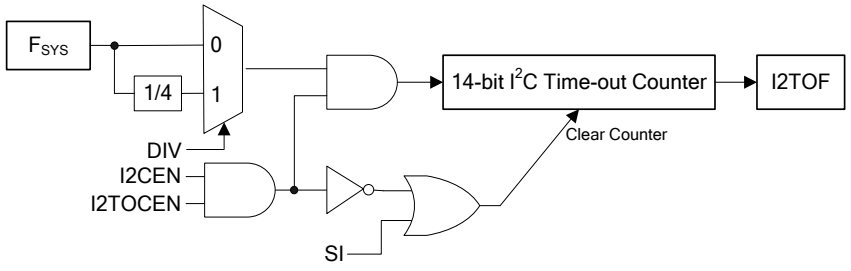


图 15.13. I²C 超时计数器

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.

15.6 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。

16. 引脚中断

N76E616 第一个引脚都有电平中断，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。PITYP 定义中断类型：电平中断，或跳变中断。 寄存器 PIF 中可以定义8个中断源，执行完中断代码，软件必须将其清0。

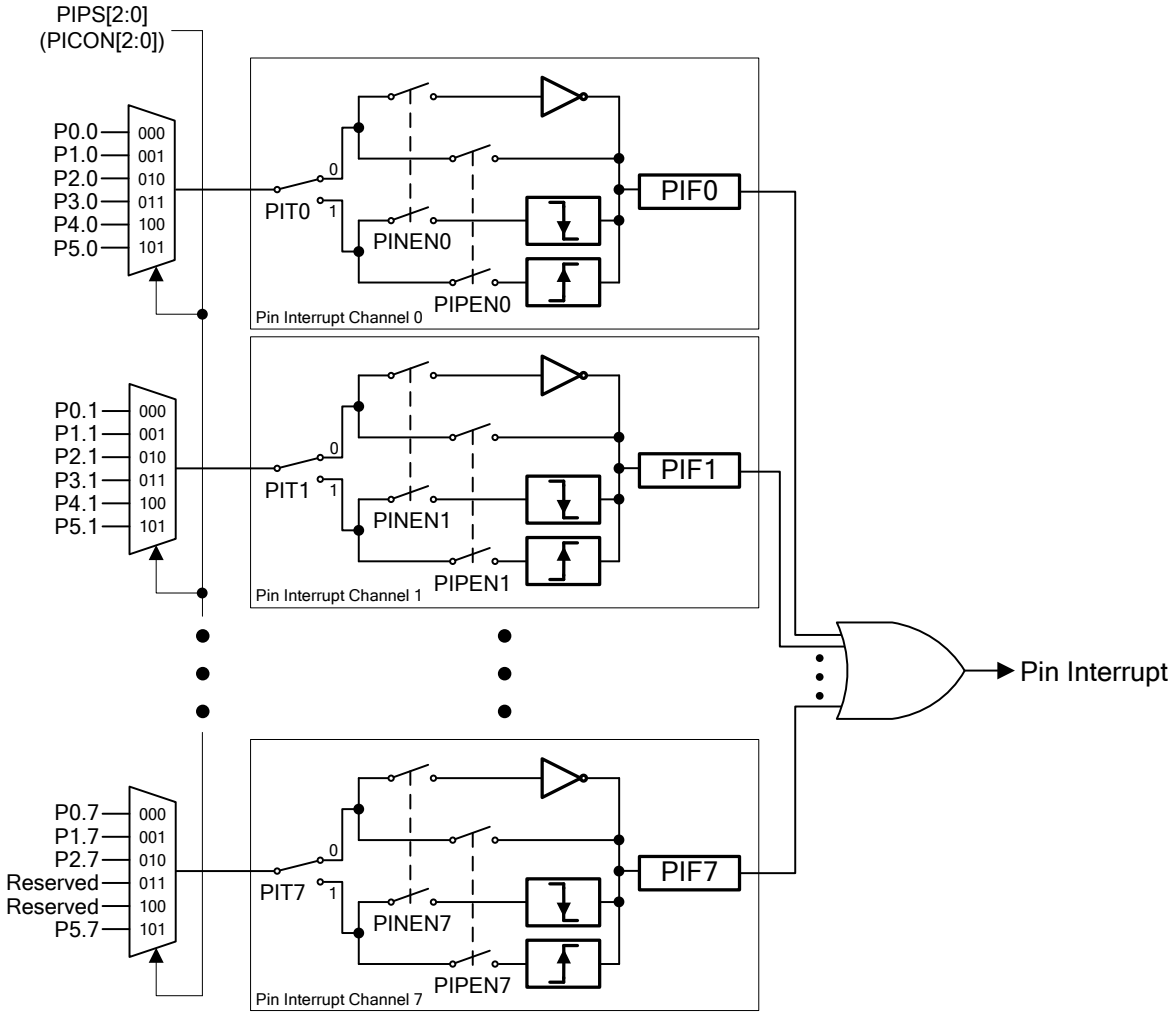


图 16.1. 引脚中断结构框图

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 – 引脚中断控制

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

地址: E9H 复位值: 0000 0000b

位	名称	描述
2:0	PIPS[2:0]	**中断引脚选 择** 选择8个中断源的输入引脚. 000 =端口 0. 001 =端口1. 010 =端口2. 011 =端口3. 100 =端口4. 101 =端口5. 其它: 保留.

PITYP –引脚中断类型选择

7	6	5	4	3	2	1	0
PIT7	PIT6	PIT5	PIT4	PIT3	PIT2	PIT1	PIT0
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: EDH 复位值: 0000 0000b

位	名称	描述
n	PITn	**引脚中断类型选择** 0 = 电平中断. 1 = 跳变中断.

PINEN –引脚中断反相特性选择使能位

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	**引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 –引脚中断正相特性使能

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	**引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 – 引脚中断标志位

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	**引脚中断通道标志位** 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。

17. 10位模数转换- (ADC)

N76E616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。N76E616 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。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。

17.1 功能描述

17.1.1 ADC 工作方式

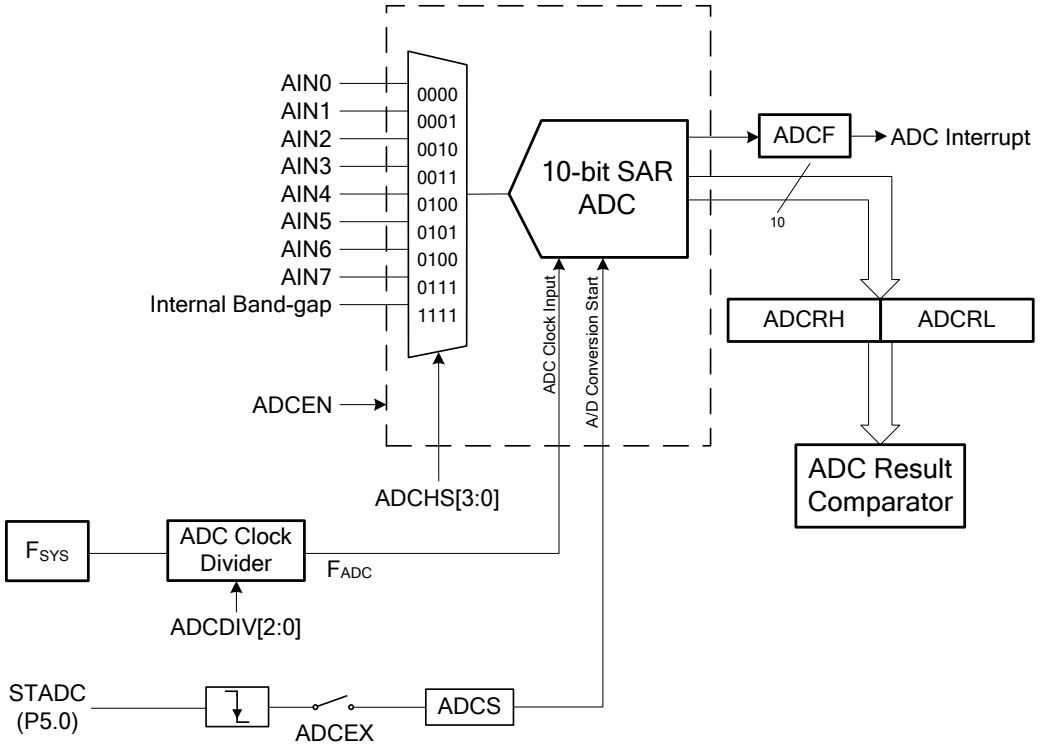


图 17.1. 10位 ADC 模块结构图

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31-9。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1 (1 + 0) 至 256 (1 + 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。

17.1.2 ADC转换结果比较器

N76E616 ADC 有个数字比较器： ACMPH 和 ACMPL。位 ADCMPEN (ADCCON2.5) 置1，使能每次转换与之比较功能。ADCMPO (ADCCON2.4) 是比较结果，ADCMPOP (ADCCON2.6)配置比较结果的极性，具体见下图。

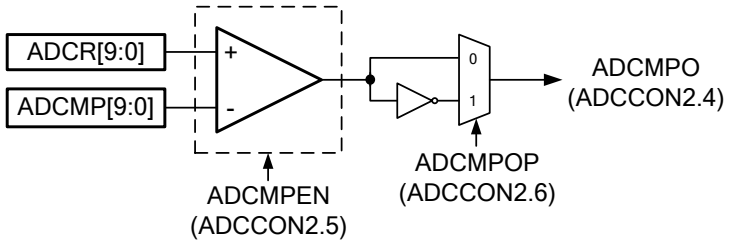


图 17.2. ADC 转换结果比较器

17.2 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	-	-	ADCHS3	ADCHS2	ADCHS1	ADCHS0
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	-	**保留位**

位	名称	描述
3:0	ADCHS[3:0]	**A/D 转换通道选择** 若 ADCEN= 0，所有通道都未连接。 0000 = AIN0. 0001 = AIN1. 0010 = AIN2. 0011 = AIN3. 0100 = AIN4. 0101 = AIN5. 0110 = AIN6. 0111 = AIN7. 1111 =接内部带隙基准源. 其它 = 保留

ADCCON1 – ADC 控制寄存器 1

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

地址: E1H　　复位值: 0010 0000b

位	名称	描述
7	-	**保留**
6:4	ADCDIV[2:0]	**ADC时钟分频** 000 = F_{ADC}为F_{SYS}/1. 001 = F_{ADC}为F_{SYS}/2. 010 = F_{ADC}为F_{SYS}/4. (缺省值) 011 = F_{ADC}为F_{SYS}/8. 100 = F_{ADC}为F_{SYS}/16. 101 = F_{ADC}为F_{SYS}/32. 110 = F_{ADC}为F_{SYS}/64. 111 = F_{ADC}为F_{SYS}/128.
3:2	-	**保留**
1	ADCEX	**ADC 外触发选择** 0 = A/D 转换只能由置1 ADCS 启动。 1 = 置1 ADCS或 引脚STADC的下沿都可启动转换。ADCS =1 触发引脚的信号将被忽略。
0	ADCEN	**ADC 使能** 0 = ADC 关闭。 1 = ADC 使能。

ADCCON2 – ADC 控制寄存器 2

7	6	5	4	3	2	1	0
-	ADCMPOP	ADCMPEN	ADCMPO	-	-	-	-
-	读/写	读/写	R	-	-	-	-

地址: E2H 复位值: 0000 0000b

位	名称	描述
7	-	**保留**
6	ADCMPOP	**ADC 比较输出的极性** 0 = 若ADCR[9:0]大于 ADCMP[9:0]，ADCMPO = 1 1 = 若ADCR[9:0] 小于 ADCMP[9:0]，ADCMPO= 1
5	ADCMPEN	**ADC 比较使能** 0 = ADC 关比较功能 1 = ADC 使能比较功能。
4	ADCMPO	**ADC 比较输出值** 每次ADC转换结束后更新此位的值。
3:0	-	**保留**

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 = $\frac{1+\text{ADCAQT}}{F_{ADC}}$. 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.

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

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

18. LCD 驱动

LCD显示面板在应用中很常见。N76E616内部集成了LCD驱动，可以用来直接驱动4个COM引脚(COM0至COM3) 32个SEG引脚(SEG0 to SEG31)或者是6个COM引脚(COM0 至COM5) 30个SEG引脚(SEG2 to SEG31)的LCD面板。LCD驱动支持1/4占空比、1/3占空比或1/6占空比。驱动电压支持1/3偏压或1/2偏压，驱动波形是类型A。当LCD时钟源选择LXT或LIRC时，LCD在掉电模式下可以继续工作。

18.1 功能描述

设置LCDEN (LCDCON.7)为1打开LCD电路。如果LCDEN位已被设置，则SEG引脚和COM引脚驱动信号依据内部显示寄存器来驱动LCD面板显示。占空比和偏压可以通过DUTY[1:0] (LCDCON[3:2]) 和BIAS (LCDCON.4)位段分别设置。可以通过设置VLCDADJ (LCDCON.6)位选择V_{LCD}驱动LCD面板电压是V_{DD}还是0.9V_{DD}。这个特别适用于5.0V电源系统，LCD面板却是4.5V供电的应用。注意用户在开启LCD驱动后不能再改变DUTY和BIAS，否则输出波形不可预知可能在一个LCD帧内产生一个直流分量。

LCD驱动偏压类型是R类型。偏压是由内部梯形电阻产生的。通过设置RSEL[1:0] (LCDCON[1:0])，梯形电阻总阻值可在600kΩ、300kΩ和150kΩ之中选择。150kΩ的电阻梯可以使LCD产生最佳显示效果，但是这会消耗很大 电流。对于低功耗应用，最好选择300kΩ 或 600kΩ。这一特征让用户在最佳显示效果和电源功耗之间可以灵活的选择。

LCD时钟源可以通过LCDCKS[1:0] (LCDCLK[5:4])选择，它的频率通过LCDDIV[2:0] (LCDCLK[2:0])进行配置。选择正确的帧速率对于LCD显示非常重要。正常帧速率推荐设置在30 Hz 至 100 Hz范围内。帧率过低会产生闪烁，帧率过高会产生鬼影甚至会消耗更多的电流。想要获得好的显示效果依据LCD面板选择恰当的帧率。下面是LCD频率计算公式：

$F_{LCD} = \frac{F_{SYS}}{2^{12} \times Divider}$, 当 LCDCKS[1:0] = [0,0] 时(LCD时钟源来自系统时钟);

$F_{LCD} = \frac{F_{LXT}}{2^{4} \times Divider}$, 当 LCDCKS[1:0] = [0,1] 时 (LCD时钟源来自LXT);

$F_{LCD} = \frac{F_{LIRC}}{2^{4} \times Divider}$, 当 LCDCKS[1:0] = [1,0] 时 (LCD时钟源来自LIRC).

COM引脚的选择依据一帧周期里的占空比，1/4占空比时COM0 至 COM3产生驱动信号。然而1/3占空比时COM0 至 COM2产生驱动信号，COM3不产生信号但是可以作为普通I/O用。当选择1/6占空比时，SEG0 和 SEG1 作为 COM4 和 COM5。原来的SEG0 和 SEG1的控制位不起作用。

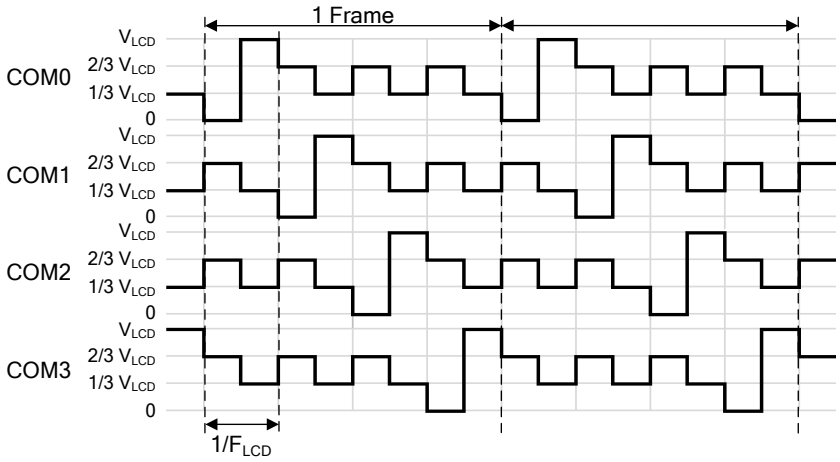


图 18.1. 1/4 占空比，1/3 偏压COM 信号

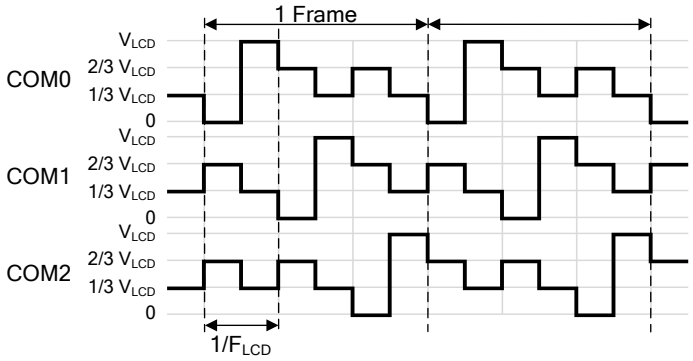


图 18.2. 1/3 占空比，1/3 偏压COM 信号

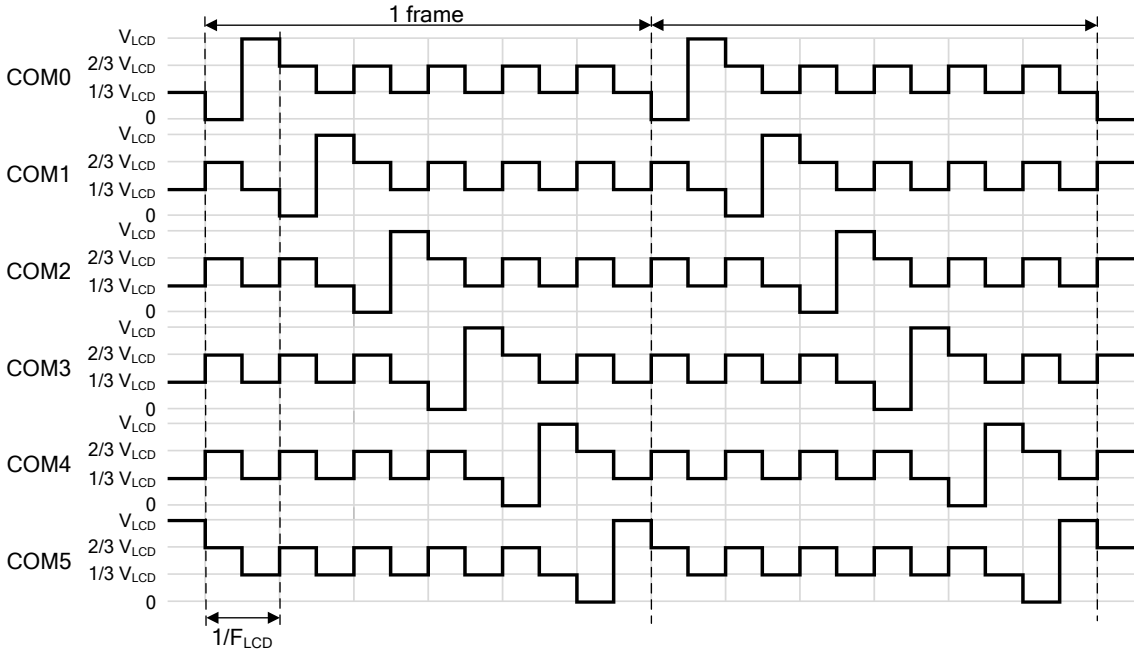


图 18.3. 1/6 占空比，1/3 偏压COM 信号

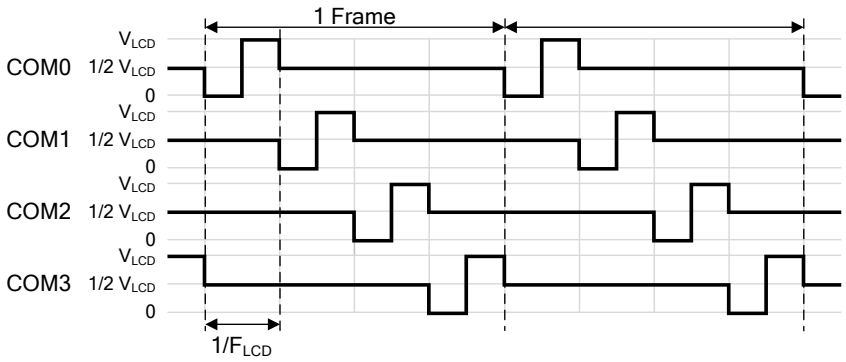


图 18.4. 1/4 占空比，1/2 偏压COM 信号

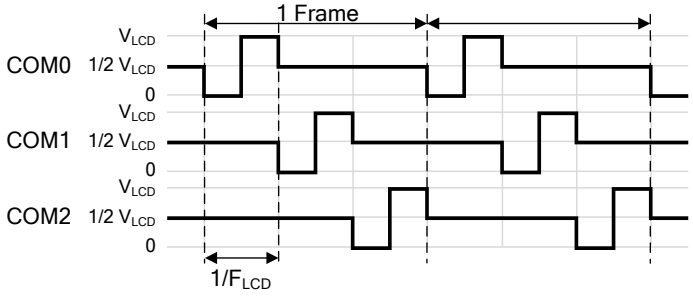


图 18.5. 1/3 占空比，1/2 偏压COM 信号

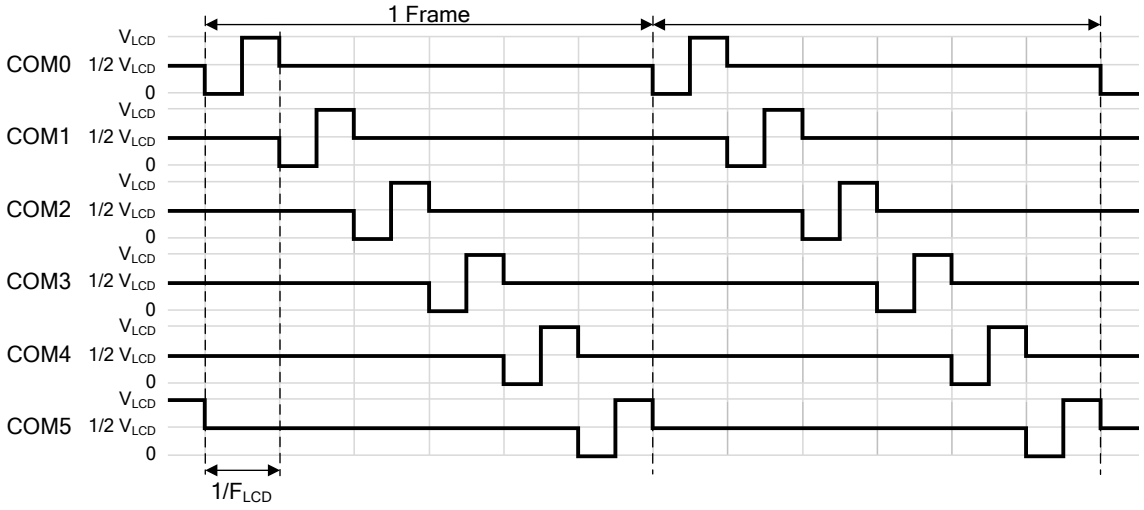


图 18.6. 1/6 占空比，1/2 偏压COM 信号

每一个SEG引脚信号依据LCD显示数据寄存器。存储在显示寄存器里的LCD数据在F_{LCD}同步转换下，依次自动产生LCD驱动信号到每个SEG引脚上，不需要过程控制。当显示数据寄存器一个位为1，相应的LCD的一个点会变暗。反之为0，像素点是透明的。

32个内部寄存器用来存储LCD显示数据。LCDPTR决定哪个寄存器可以被软件访问。每一个字节控制一个SEG引脚交错着的不同COM引脚。在写入LCDPTR后，依据LCD的显示内容需要往LCDDAT写入恰当的数据。注意那些没有用到的位和寄存器可以作为普通目的使用。

每一个SEG引脚拥有自己的使能位SEGnEN，这些位位于LCDSEG0 至 LCDSEG3 寄存器内。这些位控制着SEG引脚的信号输出。请注意SEGnEN位的一些特殊情况：

1.只有当LCDEN (LCDCON.7)为1时这些位才有效。当LCDEN 为 0，SEG引脚作为普通I/O和其它特殊功能而不是LCD显示功能。

2.当选择1/6占空比时，SEG0EN 和 SEG1EN是无效的。

3. HXT/LXT 和 AIN7 引脚与SEG共享。如果在启用LCD时需要使用HXT/LXT 或 AIN7，用户应该清除相应的SEGnEN位。否则SEG引脚输出的信号让HXT/LXT 和 AIN7工作异常

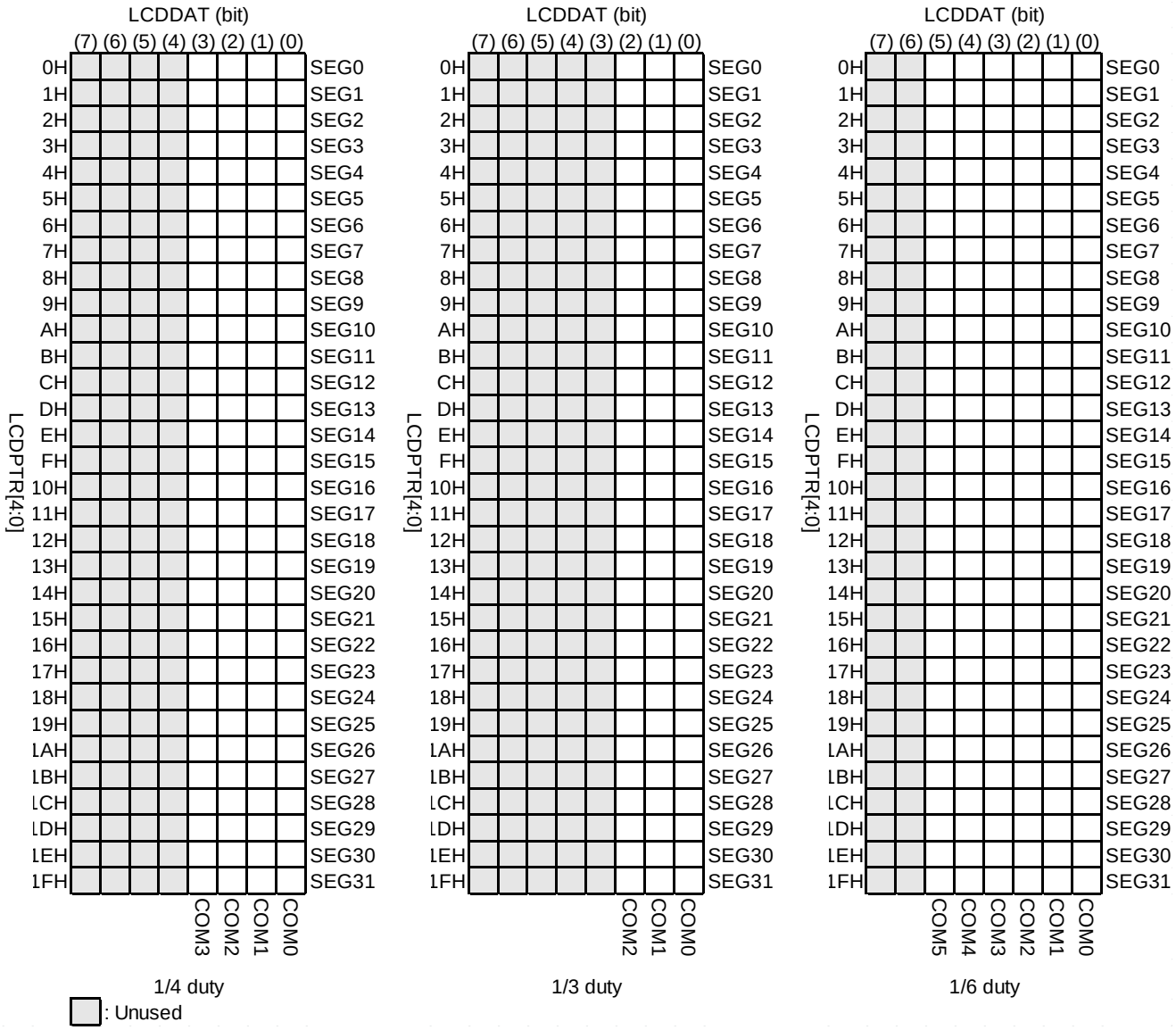


图 18.7. LCD 寄存器映射

以LCD特性，施加在像素上的电压必须是交流的，如果直接施以直流电，会造成永久性损伤。为避免这种情况，采用差分电压，确保每个段segment的平均值为0，这样实际施加到每个像素上的电压均方值，等于每个共极COM减去SEG的均方值。该差分均方值必须大于LCD灰度像素电压阀值，且小于清除电压阀值。

下图显示微处理器产生不同偏压下的显示结果。默认为1/4占空比的波形。

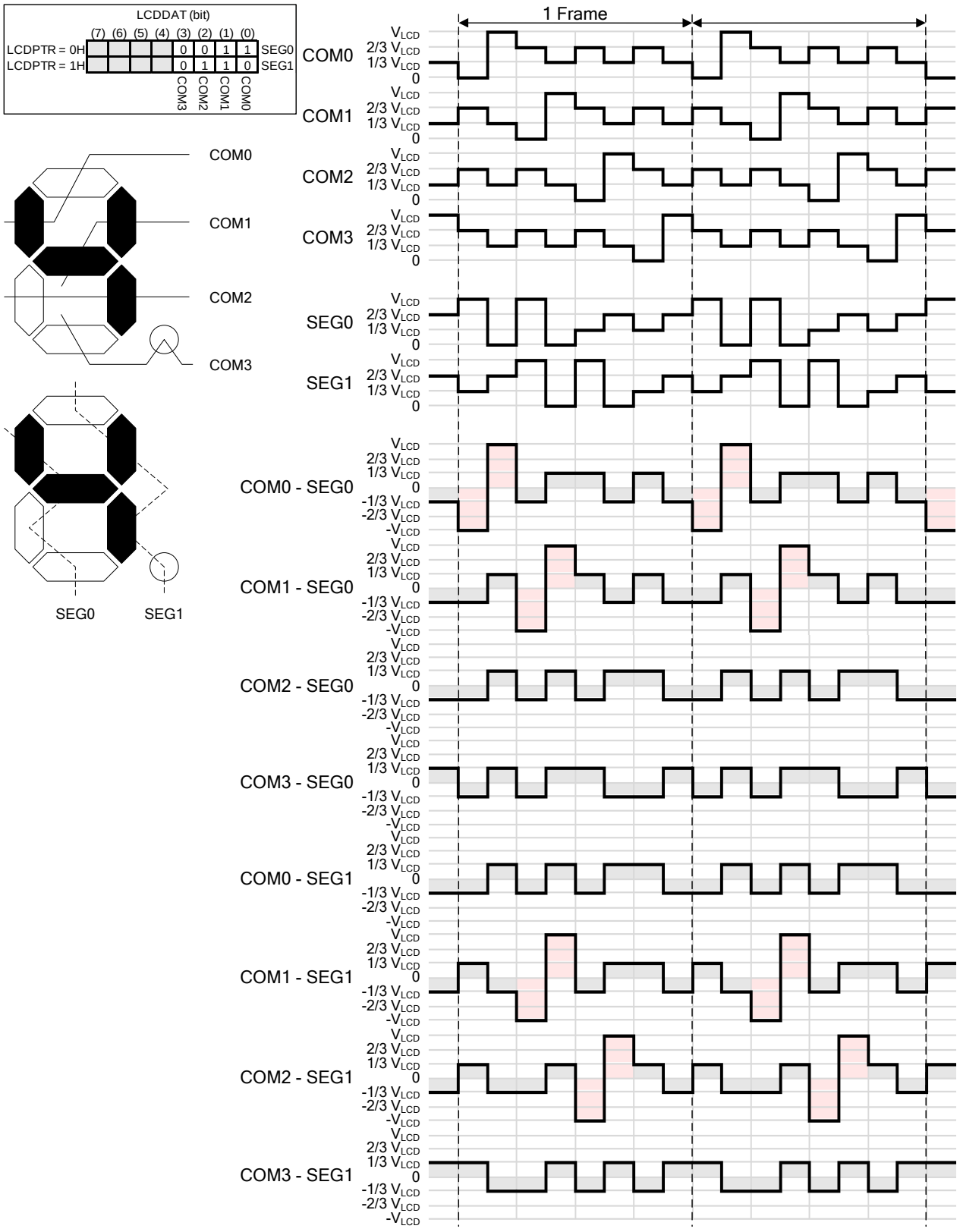


图 18.8. 1/3 偏压下COM及SEG驱动信号

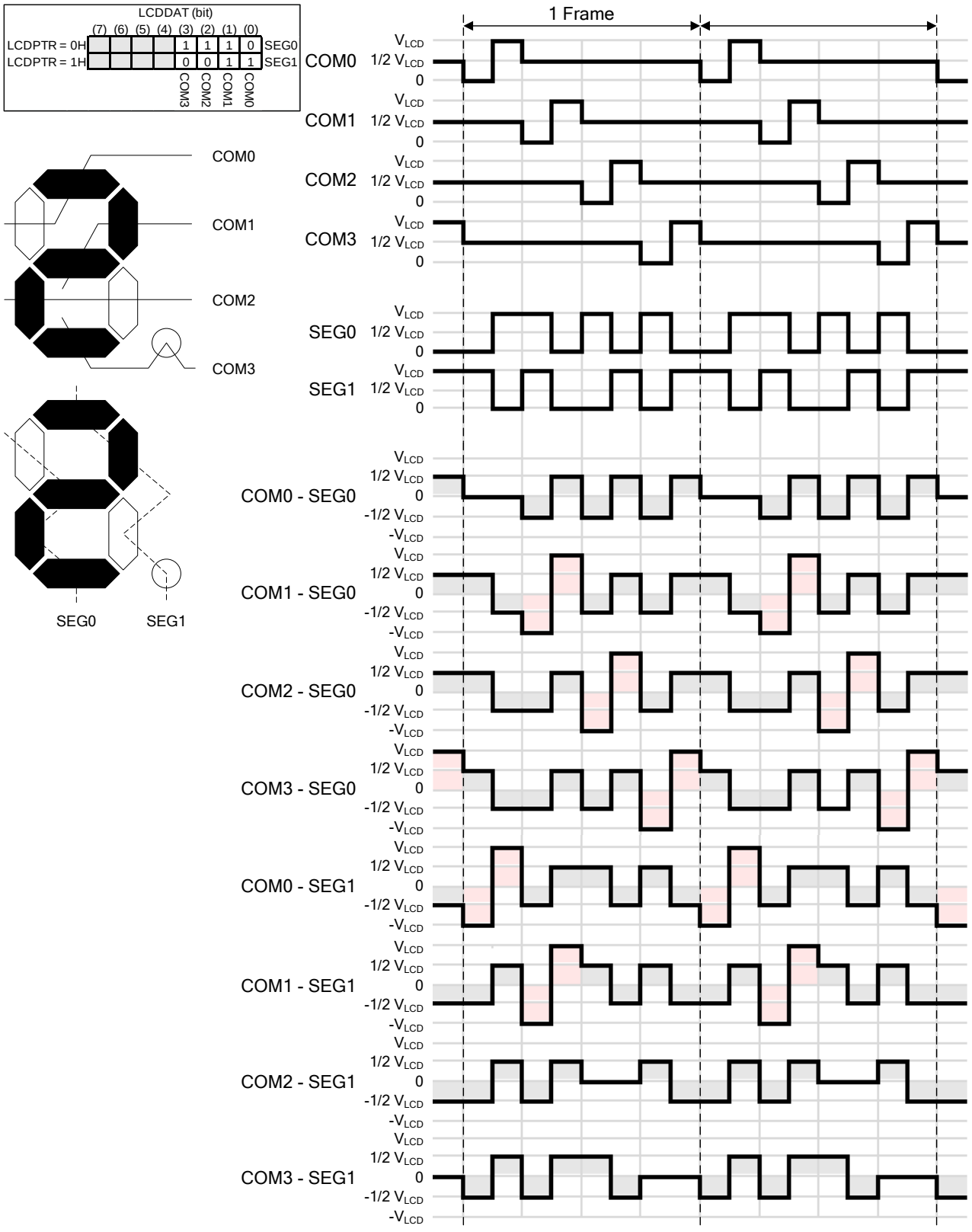


图 18.9. 1/2 偏压下COM及SEG驱动信号

18.2 LCD控制寄存器

LCDCON – LCD 控制寄存器

7	6	5	4	3	2	1	0
LCDEN	VLCDADJ	-	BIAS	DUTY[1:0]		RSEL[1:0]	
读/写	读/写	-	读/写	读/写		读/写	

地址: F9H 复位值: 0000 0000b

位	名称	描述
7	LCDEN	**LCD 使能** 0 = 关闭LCD电路。每一个SEG引脚和COM引脚作为普通I/O和其它特殊功能 1 = 开启LCD电路。COM引脚和使能的SEG引脚产生驱动波形
6	VLCDADJ	**V_{LCD} 调整** 0 = V_{LCD} 为 V_{DD}. 1 = V_{LCD} 为0.9V_{DD}.
5	-	**保留**
4	BIAS	**LCD 偏压** 0 = 1/3 偏压 1 = 1/2 偏压
3:2	DUTY[1:0]	**LCD 占空比** 00 = 1/4 占空比 01 = 1/3 占空比 10 = 1/6 占空比 11 = 保留 注意当选择1/3占空比时，仅仅使用COM0 至 COM2驱动LCD，COM3不被使用可以作为普通I/O。当选择1/6占空比时，SEG0 和 SEG1不可用，这些引脚被用作COM4和COM5
1:0	RSEL[1:0]	**LCD 电阻选择** 该位段可以选择偏压电阻梯的总阻值。值越小驱动能力越强。 00 = 150 kΩ. 01 = 300 kΩ. 10 = 600 kΩ. 11 = 保留.

LCDCLK – LCD 时钟控制

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

地址: FAH　　复位值: 0000 0000b

位	名称	描述
5:4	LCDCKS[1:0]	**LCD 时钟源选择** 00 = $F_{SYS}/2^{12}$ 01 = $LXT/2^4$ 10 = $LIRC/2^4$ 11 = 保留 注意当LCD时钟源选择LXT或LIRC时，LCD在掉电模式下还能继续工作。如果想要选择LXT作为LCD时钟源，用户必须在使用前用软件打开LXT。同时，由于占用I/O管脚，LCD的segment将减少2根。
2:0	LCDDIV[1:0]	**LCD 时钟除频** 000 = 1/1. 001 = 1/2. 010 = 1/4. 011 = 1/8. 100 = 1/16. 101 = 1/32. 其它 = 保留.

LCDPTR – LCD 数据指针

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

地址: FBH　　复位值: 0000 0000b

位	名称	描述
4:0	LCDPTR[4:0]	**LCD 数据指针** 该位段决定哪个显示数据寄存器可以通过LCDDAT被访问。在访问LCDDAT前，用户必须写入目标值到到LCDPTR里

LCDDAT – LCD 数据

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

地址: FCH 复位值: 0000 0000b

位	名称	描述
5:0	LCDDAT[5:0]	**LCD 数据** 通过LCDPTR写入相应的值到该寄存器可以让LCD的SEG和COM引脚有对应显示。 0 = LCD 像素点无显示 1 = LCD 像素点加深

LCDSEG0 – LCD Segment 0

7	6	5	4	3	2	1	0
SEG7EN	SEG6EN	SEG5EN	SEG4EN	SEG3EN	SEG2EN	SEG1EN	SEG0EN
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: E5H 复位值: 0000 0000b

LCDSEG1 – LCD Segment 1

7	6	5	4	3	2	1	0
SEG15EN	SEG14EN	SEG13EN	SEG12EN	SEG11EN	SEG10EN	SEG9EN	SEG8EN
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: E6H 复位值: 0000 0000b

LCDSEG2 – LCD Segment 2

7	6	5	4	3	2	1	0
SEG23EN	SEG22EN	SEG21EN	SEG20EN	SEG19EN	SEG18EN	SEG17EN	SEG16EN
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: E7H 复位值: 0000 0000b

LCDSEG3 – LCD Segment 3

7	6	5	4	3	2	1	0
SEG31EN	SEG30EN	SEG29EN	SEG28EN	SEG27EN	SEG26EN	SEG25EN	SEG24EN
读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写

地址: EEH 复位值: 0000 0000b

位	名称	描述
7:0	SEGnEN	**LCD SEGn 引脚使能** 0 = 禁用SEGn 引脚 1 =使能 SEGn 引脚 注意在该位为1同时LCDEN也为1时，从该位对应的I/O数据寄存器读回来的值总为0。写会改变I/O锁存器的值

18.3 LCD 程序流程

在使能LCD驱动前必须考虑好那些与被当作COM和SEG引脚的状态。所有用到的COM和SEG引脚都会输出波形，因此所有相关的I/O输出模式必须通过PxMn寄存器配置成只输入模式（高阻态）。这样可切断那些被用作COM 和 SEG引脚与数字电路的连接，I/O电路就不会影响到LCD波形输出。

配置完I/O模式后，用户依据LCD目标面板设置V_{LCD}、占空比和偏压。选择合适的帧速率（正常范围在 30 Hz 至 100 Hz）去设置LCDCLK寄存器的值。如果选择LXT作为LCD的时钟源，用户用软件打开LXT。

上述步骤过后，用户通过设置相应的SEGnEN位使能用到的SEG引脚。最后一步设置LCDEN (LCDCON.7)为1，打开LCD驱动。这些步骤过后所有被用到的COM和SEG引脚会产生驱动波形。后续就通过设置LCDPTR 和 LCDDAT决定LCD面板上像素点哪些变暗哪些不显示。

以下是完整的LCD驱动参考程序：

```
;*******************************************************************************
;       This code illustrates how to enable LCD driver. A 3V, 1/4 duty, 1/3
;       bias with 4*COM and 16*SEG panel is used. The system VDD is 3.3V. LCD
;       uses LXT 32.768 kHz as its clock source. COM0~COM3 and SEG0~SEG15 are
;       used to drive the panel to shows "12345678" eight digits by the same
;       typology of 图 18.8.
;*******************************************************************************
NUM_1  EQU    01100000b
NUM_2  EQU    00111110b
NUM_3  EQU    01111010b
NUM_4  EQU    01100011b
NUM_5  EQU    01011011b
NUM_6  EQU    01011111b
NUM_7  EQU    01110001b
NUM_8  EQU    01111111b

       ORG    0000h

;*******************************************************************************
;             Confeature I/O output mode
;*******************************************************************************
       ORL    P1M1,#11111111b            ;Con图 COM0~COM3, SEG0~SEG15 as
       ANL    P1M2,#00000000b            ;input only mode
       ORL    P2M1,#00011111b
       ANL    P2M2,#11100000b
       ORL    P4M1,#01111111b
       ANL    P4M2,#10000000b

;*******************************************************************************
;             LCD setting
;*******************************************************************************
       MOV    LCDCON,#01000000b          ;VLCD = 0.9VDD (VDD = 3.3V, VLCD =
                                         ;3.0V)
                                         ;1/4 duty, 1/3 bias, 150kΩ resistor
```

```
                                        ;ladder

;*******************************************************************************
;             LCD clock setting
;*******************************************************************************
       MOV    TA,#0AAh                  ;TA protection
       MOV    TA,#55h                   ;
       ORL    CKEN,#01000000b           ;Enable the LXT

Polling_LXT_stable:                     ;Waiting for the LXT stable
       MOV    A,CKSWT
       JNB    ACC.6, Polling_LXT_stable

       MOV    LCDCLK,#00010011b         ;Select LXT as LCD clock source
                                        ;Frame rate = (32768/16)/8/4 = 64

;*******************************************************************************
;             Enable SEG pins and enable LCD
;*******************************************************************************
       MOV    LCDSEG0,#0FFh             ;Enable SEG0~SEG15
       MOV    LCDSEG1,#0FFh

       ORL    LCDCON,#80h

;*******************************************************************************
;             Write LCD data to display "12345678"
;*******************************************************************************
       MOV    LCDPTR,#0
       MOV    LCDDAT,#(NUM_8 & 0x0F)
       MOV    LCDPTR,#1
       MOV    LCDDAT,#(NUM_8 >> 4)

       MOV    LCDPTR,#2
       MOV    LCDDAT,#(NUM_7 & 0x0F)
       MOV    LCDPTR,#3
       MOV    LCDDAT,#(NUM_7 >> 4)

       MOV    LCDPTR,#4
       MOV    LCDDAT,#(NUM_6 & 0x0F)
       MOV    LCDPTR,#5
       MOV    LCDDAT,#(NUM_6 >> 4)

       MOV    LCDPTR,#6
       MOV    LCDDAT,#(NUM_5 & 0x0F)
       MOV    LCDPTR,#7
       MOV    LCDDAT,#(NUM_5 >> 4)

       MOV    LCDPTR,#8
       MOV    LCDDAT,#(NUM_4 & 0x0F)
       MOV    LCDPTR,#9
       MOV    LCDDAT,#(NUM_4 >> 4)

       MOV    LCDPTR,#10
       MOV    LCDDAT,#(NUM_3 & 0x0F)
       MOV    LCDPTR,#11
       MOV    LCDDAT,#(NUM_3 >> 4)

       MOV    LCDPTR,#12
       MOV    LCDDAT,#(NUM_2 & 0x0F)
       MOV    LCDPTR,#13
       MOV    LCDDAT,#(NUM_2 >> 4)
```

```
MOV     LCDPTR,#14
MOV     LCDDAT,#(NUM_1 & 0x0F)
MOV     LCDPTR,#15
MOV     LCDDAT,#(NUM_1 >> 4)
```

19. 时控保护 (TA)

N76E616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，N76E616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 – 时控访问保护

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。

对于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 protect 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：

Example 1,

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

Example 2,

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

Example 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
```

Example 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。

20. 中断系统

20.1 中断概述

中断的目的是使软件处理非常规或异步事件。N76E616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 20–1..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 20–1. 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	-	-	-	-
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2B下溢出/匹配 0	0063H	12
定时器1 溢出	001BH	3	定时器2C下溢出/匹配 0	006BH	13
串口 0 中断	0023H	4	定时器2D下溢出/匹配 0	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	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			

20.2 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 (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	**所有中断使能** 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	**ADC中断使能** 0 = 禁用ADC中断 1 = 使能由ADCF (ADCCON0.7) 产生中断
5	EBOD	**BOD中断使能** 0 = 禁用BOD中断 1 =使能由BOF (BODCON0.3)产生中断
4	ES	**串口0中断使能** 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	**定时器1中断使能** 0 = 禁用定时器1中断 1 =使能由TF1 (TCON.7)产生中断
2	EX1	**外部中断1使能** 0 = 禁用外部中断1 1 =使能由$\overline{\text{INT1}}$产生中断
1	ET0	**定时器0中断使能** 0 = 禁用定时器1中断 1 =使能由TF0 (TCON.5)产生中断
0	EX0	**外部中断0使能** 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

7	6	5	4	3	2	1	0
ET2D	-	ET2C	EWDT	ET2B	ET2A	EPI	EI2C
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	-	**保留**
7	ET2D	**定时器2D中断使能** 0 = 禁用定时器2D中断 1 =使能由TF2D (T2CON0.7)产生中断

位	名称	描述
6	-	**保留**
5	ET2C	**定时器2C中断使能** 0 = 禁用定时器2C中断 1 =使能由TF2C (T2CON0.6)产生中断
4	EWDT	**WDT中断使能** 0 = 禁用WDT中断 1 =使能由WDTF (WDCON.5)产生中断
3	ET2B	**定时器2B中断使能** 0 = 禁用定时器2B中断 1 =使能由TF2B (T2CON0.5)产生中断
2	ET2A	**定时器2A中断使能** 0 = 禁用定时器2A中断 1 =使能由TF2A (T2CON0.4)产生中断
1	EPI	**引脚中断使能** 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** 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 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	**看门狗中断使能** 0 = 禁用WKT中断 1 =使能由WKTF (WKCON.4)产生中断
1	ET3	**定时器3中断使能** 0 = 禁用定时器3中断 1 =使能由TF3 (T3CON.4)产生中断
0	ES_1	**串口1中断使能** 0 = 禁用串口1中断 1 =使能由TI_1 (SCON_1.1) 或 RI_1 (SCON_1.0)产生中断

20.3 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 20–2 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 20–3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**錯誤! 找不到參照來源。錯誤! 找不到參照來源。**”。

表 20–2. 中断优先级设定

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

表 20–3. 中断源汇总表

中断源	向量地址	中断标志位	使能位	默认优先级	优先级控制位	可否从掉电唤醒
复位	0000H	-	总是使能	最高级	-	是
外部中断 0	0003H	IE0[1]	EX0	1	PX0, PX0H	是
掉电	0043H	BOF (BODCON0.3)	EBOD	2	PBOD, PBODH	是
看门狗定时器	0053H	WDTF (WDCON.5)	EWDT	3	PWDT, PWDTH	是
定时器 0	000BH	TF0[2]	ET0	4	PT0, PT0H	否
I^2C状态/超时	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	是
引脚中断	003BH	PIF0 to PIF7 (PIF)[3]	EPI	8	PPI, PPIH	是
定时器1	001BH	TF1[2]	ET1	9	PT1, PT1H	否
串口 0	0023H	RI + TI	ES	10	PS, PSH	否
定时器2A	002Bh	TF2A[2]	ET2A	11	PT2A, PT2AH	否
定时器2B	0063H	TF2B[2]	ET2B	13	PT2B, PT2BH	否
定时器2C	006BH	TF2C[2]	ET2C	14	PT2C, PT2CH	否
定时器2D	0073H	TF2D[2]	ET2D	15	PT2D, PT2DH	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	是

[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 – 中断优先级地位 (可位寻址)[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
7	-	**保留位**
6	PADC	**ADC中断优先级低位**
5	PBOD	**BOD检测中断优先级低位**

位	名称	描述
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的。见表 20–2.

IPH – 中断优先级高位[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
7	-	**保留**
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的。见表 20–2.

EIP – 扩展中断优先级 [3]

7	6	5	4	3	2	1	0
PT2D	-	PT2C	PWDT	PT2B	PT2A	PPI	PI2C
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	PT2D	**定时器2D中断优先级低位**
6	-	**保留**
5	PT2C	**定时器2C中断优先级低位**
4	PWDT	**WDT中断优先级低位**
3	PT2B	**定时器2B中断优先级低位**
2	PT2A	**定时器2A中断优先级低位**

位	名称	描述
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。见表 20–2.

EIPH –扩展中断优先级高字节[4]

7	6	5	4	3	2	1	0
PT2DH	-	PT2CH	PWDTH	PT2BH	PT2AH	PPIH	PI2CH
R/W	-	R/W	R/W	R/W	R/W	R/W	R/W

地址: F7H 复位值: 0000 0000b

位	名称	描述
7	PT2DH	**定时器2D 中断优先级高位**
6	-	**保留**
5	PT2CH	**定时器2C 中断优先级高位**
4	PWDTH	**WDT 中断优先级高位**
3	PT2BH	**定时器2B 中断优先级高位**
2	PT2AH	**定时器2A 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。见表 20–2.

EIP1 –扩展中断优先级1[5]

7	6	5	4	3	2	1	0
-	-	-	-	-	PWKT	PT3	PS_1
-	-	-	-	-	R/W	R/W	R/W

地址: FEH, 页数: 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。见表 20–2.

EIPH1 –扩展中断优先级高字节1[6]

7	6	5	4	3	2	1	0
-	-	-	-	-	PWKTH	PT3H	PSH_1
-	-	-	-	-	R/W	R/W	R/W

地址: FFH, 页数: 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。见表 20–2.

20.4 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。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，都不能产生LCALL。在每一个指令周期都会检测中断标志。如果上述条件有一个不满足，虽然标志位置'1'，也不能响应中断。当所有的条件都满足了，中断标志已经消失，该中断也不能再被回应。

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。使得控制器认为中断服务仍在进行。

20.5 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

20.6 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 和 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
R/W	R/W	R/W	R/W	R (level) R/W (edge)	R/W	R (level) R/W (edge)	R/W

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$\overline{\text{INT1}}$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$\overline{\text{INT1}}$的中断触发类型是下降沿还是低电平. 0 = $\overline{\text{INT1}}$ 为低电平触发 1 = $\overline{\text{INT1}}$ 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$\overline{\text{INT0}}$输入信号逻辑电平的反转。软件不可控制
0	IT0	**外部中断0类型选择** 该位选择$\overline{\text{INT0}}$的中断触发类型是下降沿还是低电平. 0 = $\overline{\text{INT0}}$ 为低电平触发 1 = $\overline{\text{INT0}}$ 为下降沿触发

21. 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。N76E616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在应用编程方法。

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: 见表 6–2.

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 21–1..
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)`

21.1 IAP 命令

N76E616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 21–1. 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	DID 低字节: 0000H DID 高字节: 0001H	DID 低字节: 50H DID 高字节: 2FH
读96位UID	XX	0	0	0100	0000H to 000BH	数据读出
APROM 页擦除	00	1	0	0010	地址输入[2]	FFH
LDROM 页擦除	01	1	0	0010	地址输入[2]	FFH
APROM 字节编程	00	1	0	0001	地址输入	数据写入
LDROM字节编程	01	1	0	0001	地址输入	数据写入
APROM 字节读	00	0	0	0000	地址输入	数据读出
LDROM 字节读	01	0	0	0000	地址输入	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
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 256字节，所以地址应该是目标页的地址

21.2 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

21.3 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。N76E616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20,000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保护
       MOV    TA,#55h
       ORL    CHPCON,#00000001b         ;IAPEN = 1, enable IAP mode

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

       MOV    IAPCN,#PAGE_ERASE_AP      ;Erase page 200h to 27Fh
       MOV    IAPAH,#02h
       MOV    IAPAL,#00h
       MOV    IAPFD,#0FFh
       MOV    TA,#0AAh                  ;IAPTRG is TA保护
```

```
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:

```
//***********************************************************************************
//     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
  256-element array in “code” area from absolute address 0x0200                */

volatile unsigned char code Data_Flash[256] _at_ 0x0200;

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

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

       IAPCN = PAGE_ERASE_AP;            //Erase page 200h to 27Fh
       IAPAH = 0x02;
       IAPAL = 0x00;
       IAPFD = 0xFF;
       TA = 0xAA;                        //IAPTRG is TA protect
       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 protect
       TA = 0x55;
       IAPUEN &= ~0x01;                  //APUEN = 0, disable APROM update

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

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

       while(1);
}
```

21.4 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N76E616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 con图 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 protect
        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 protect
        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 protect
        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,0B5h, 00h               ;OPCODEs of "MOV    P2M1,#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
```

22. 电源管理

N76E616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P3.6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P36UP (P3M2.6)打开内部上拉。

PCON – 电源控制

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

地址: 87H 复位值: 见表 6–2.

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。

P3M2 – 端口 3 模式选择 2

7	6	5	4	3	2	1	0
CLOEN	P36UP	P3M2.5	P3M2.4	P3M2.3	P3M2.2	P3M2.1	P3M2.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

地址: ADH, 页: 0 复位值: 0000 0000b

位	名称	描述
6	P36UP	**P3.6 上拉使能** 0 = 禁用P3.6上拉 1 =使能 P3.6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

22.1 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

22.2 掉电模式

掉电模式是N76E616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。

N76E616有两种方法可以退出掉电模式。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关闭ADC模块并使能欠压检测。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N76E616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$\overline{INT0}$ 和 $\overline{INT1}$、引脚中断、WDT中断、WTK中断和欠压中断。

23. 时钟系统

N76E616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。N76E616内嵌2个内部RC振荡器一个10 kHz低速和一个11.059 MHz高速RC振荡器，高速11.059 MHz误差在出厂时校准到±5%（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2 MHz 至16 MHz，低速为32.768 kHz。CKDIV除频器可以让N76E616在高效率与低功耗之间有个灵活的选择。

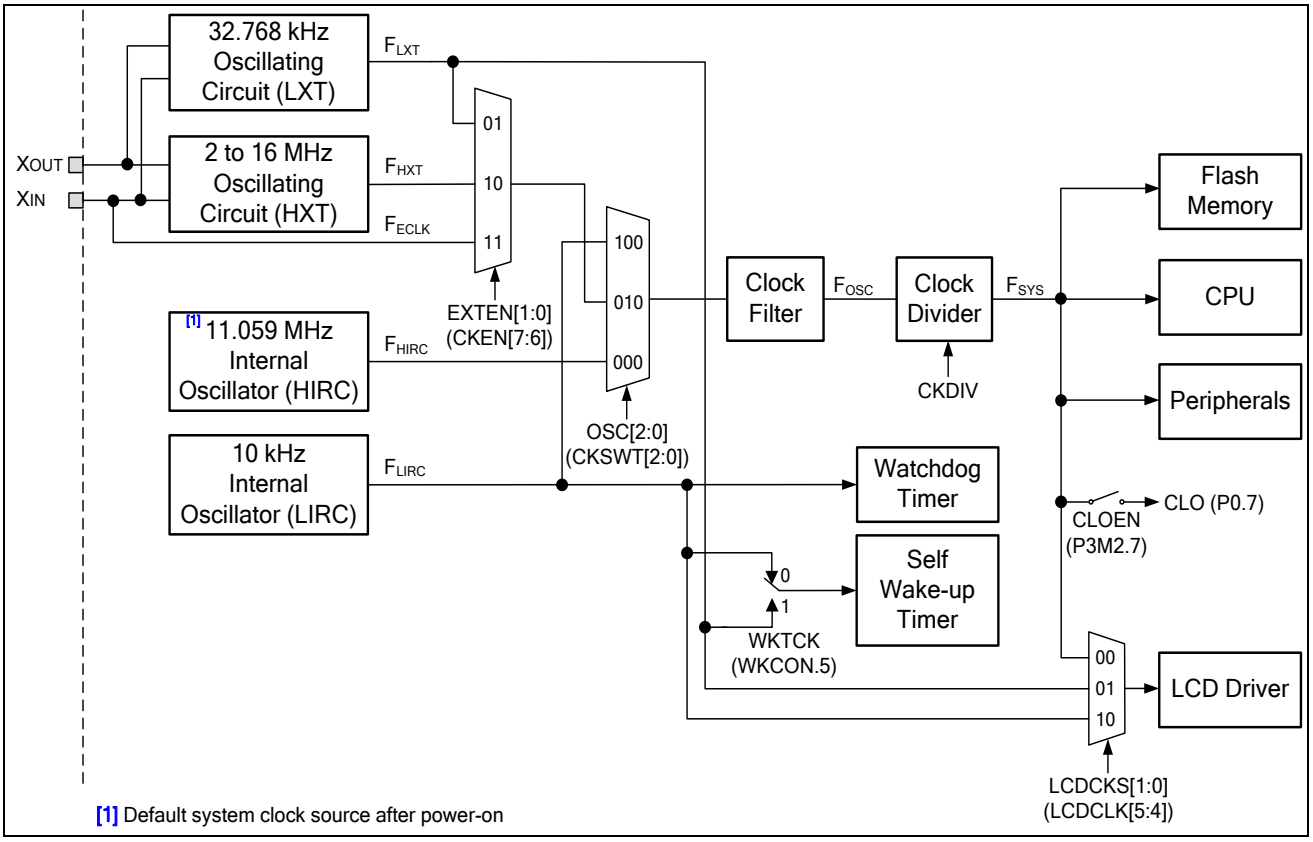


图 23.1时钟系统框图

23.1 时钟源

N76E616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N76E616的系统时钟。开启不同的时钟源可能会影响到多功能引脚P3.2/XIN 和 P3.3/XOUT。

23.1.1 内部振荡器

N76E616内部有两个RC振荡器，一个高速11.059 MHz（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[2:0] (CKSWT [2:0]) 为 [0,0,0]选择HIRC作为系统时钟，设置OSC[2:0] (CKSWT [2:0]) 为 [1,0,0] 选择LIRC作为系统时钟。注意N76E616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P3M1和P3M2寄存器来选择P3.2和P3.3的输出模式。

23.1.2 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，2MHz 到 16MHz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 P3.3 。可以通过配置P3M1和P3M2寄存器来选择P3.3的输出模式。

23.2 系统时钟切换

N76E616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 – 时钟切换 (TA 保护)

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

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

位	名称	描述
2:0	OSC[2:0]	**振荡器选择位段** 该位段是用来选择系统时钟源 000 = HIRC. 010 =外部时钟源，实际依据EXTEN[1:0]来确定 100 = LIRC. Others = 保留. 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 (TA 保护)

7	6	5	4	3	2	1	0
EXTEN[1:0]		HIRCEN	-	-	-	-	CKSWTF
R/W		R/W	-	-	-	-	R

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[2:0] 为 [0,1,0]。 00 = I/O.禁用外部时钟。P3.2 和 P3.3作为普通I/O使用 01 =使能 LXT 10 =使能 HXT 11 = 使能ECLK
5	HIRCEN	**HIRC 使能** 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** 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。

23.3 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]							
R/W							

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，$F_{SYS} = \frac{F_{OSC}}{2 \times CKDIV}$

23.4 系统时钟输出

N76E616提供一个CLO(P0.7)引脚可以输出系统时钟，该频率等同F_{SYS}。通过设置CLOEN (P3M2.7)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。

P3M2 – 端口 3 模式选择 2

7	6	5	4	3	2	1	0
CLOEN	P36UP	P3M2.5	P3M2.4	P3M2.3	P3M2.2	P3M2.1	P3M2.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

地址: ADH, 页: 0 复位值: 0000 0000b

位	名称	描述
7	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(P0.7)输出

24. 电源管理

为防止电源上电阶段或是系统板上电源不稳状况造成芯片不稳定运行，N76E616提供2种功能，上电侦测及欠压侦测来检测电源检测。

24.1 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

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.

位	名称	描述
4	POF	**上电复位标志** 上电复位后该标志位被置1。这表明是冷复位且复位周期已完成。该位在其它复位产生后保持不变。强力建议通过软件将其清除

24.2 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。四个电平2.2V, 2.7V, 3.8V, 和 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。

N76E616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。

对于噪声敏感的系统，N76E616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 24–2.

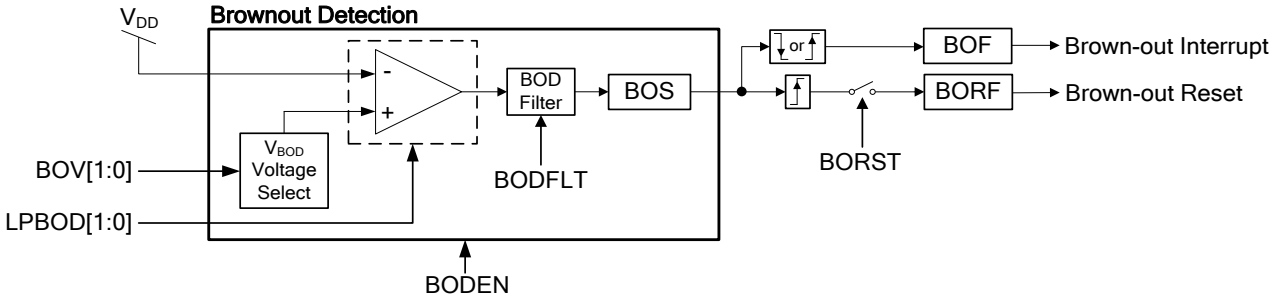


图 24.1.欠压检测框图

CONFIG2

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

出厂默认值: 1111 1111b

位	名称	描述
7	CBODEN	**配置欠压检测使能** 1 =使能欠压检测电路 0 =禁用欠压检测电路
6	-	**保留**
5:4	CBOV[1:0]	**配置欠压检测电压选择** 11 = V_{BOD}为 2.2V 10 = V_{BOD}为2.7V 01 = V_{BOD}为3.8V 00 = V_{BOD}为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 0 (TA 保护)

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

地址: A3H　　复位值: 见表 6–2.

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	-	**保留**
5:4	BOV[1:0]	**配置欠压检测电压选择** 11 = V_{BOD}为 2.2V 10 = V_{BOD}为2.7V 01 = V_{BOD}为3.8V 00 = V_{BOD}为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 bit 7、 bit 5~ 4 和 bit 2决定的
[2] BOF复位后的值依据CONFIG2的设置和V_{DD}的电平，请见表24-1

表 24–1. 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 1 (TA 保护)

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

地址: ABH　　复位值: 见表 6–2.

位	名称	描述
7:3	-	**保留**
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 24–2. 最小欠压检测脉冲宽度

BODFLT (BODCON1.1)	BOD 工作模式	系统时钟源	最小欠压检测检测脉冲宽度
0	正常模式 (LPBOD[1:0] = [0,0])	任何时钟源	Typ. 1μs
	低功率模式 1 (LPBOD[1:0] = [0,1])	任何时钟源	16 ($1/F_{LIRC}$)
	低功率模式2 (LPBOD[1:0] = [1,0])	任何时钟源	64 ($1/F_{LIRC}$)
	低功率模式3 (LPBOD[1:0] = [1,1])	任何时钟源	256 ($1/F_{LIRC}$)
1	正常模式 (LPBOD[1:0] = [0,0])	HIRC/HXT/ECLK	正常工作模式: 32 ($1/F_{SYS}$) 空闲: 32 ($1/F_{SYS}$) 掉电模式: 2 ($1/F_{LIRC}$)
		LIRC/LXT	2 ($1/F_{LIRC}$)
	低功率模式1 (LPBOD[1:0] = [0,1])	任何时钟源	18 ($1/F_{LIRC}$)
	低功率模式2 (LPBOD[1:0] = [1,0])	任何时钟源	66 ($1/F_{LIRC}$)
	低功率模式3 (LPBOD[1:0] = [1,1])	任何时钟源	258 ($1/F_{LIRC}$)

25. 复位

N76E616有几个选项让设备进入复位状态。通常，大多数特殊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。用户可以用软件读回这些标志来确定是哪个复位源引起的复位。有5种方法使芯片进入复位状态。它们是上电复位、外部复位引脚复位、软件复位、看门狗定时器复位、以及欠压复位。

25.1 上电复位

N76E616内部包含上电复位电路。在上电过程中，当V_{DD}低于参考电压门限值，参考电压保持MCU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

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

地址: 87H　　复位值: 见表 6–2.

位	名称	描述
4	POF	**上电复位标志** 上电复位后该位置1。表示一次冷复位，上电复位完成。其它复位后该位保持不变，建议软件清该标志。

25.2 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})，如果BORST (BODCON0.2)置1，欠压检测逻辑将复位MCU。当发生欠压复位后，BORF (BODCON0.1)位被硬件值1。BORF不会被其它复位改变除了上电复位和自身的欠压复位。建议通过软件清除该位。

BODCON0 –欠压检测控制 0 (TA 保护)

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

地址: A3H 复位值: 见表 6–2.

位	名称	描述
1	BORF	**欠压复位标志** 当MCU发生欠压复位后，该位被硬件值1。建议通过软件清除该位。

25.3 外部复位引脚复位

$\overline{\text{RST}}$复位引脚为史密特触发输入脚。将$\overline{\text{RST}}$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产生外部复位。

一旦芯片进入复位，只要$\overline{\text{RST}}$引脚上电平为0就会一直保持复位状态。在$\overline{\text{RST}}$撤除低电平之后，CPU将会退出复位状态，并从0000H开始执行代码。如果CPU在掉电模式下，有$\overline{\text{RST}}$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，复位信号将异步引起系统时钟恢复。在系统时钟稳定后，MCU 将进入复位状态。

有一个标志位RSTPINF (AUXR1.6)可以指示外部复位是否发生。在外部复位发生后，该位会被硬件置1. RSTPINF不会被其它复位改变除了上电复位和自身的欠压复位。建议通过软件清除该位。

AUXR1 – 辅助寄存器 1

7	6	5	4	3	2	1	0
SWRF	RSTPINF	T1LXTM	T0LXTM	GF2	-	0	DPS
R/W	R/W	R/W	R/W	R/W	-	R	R/W

地址: A2H 复位值: 见表 6–2.

位	名称	描述
6	RSTPINF	**外部复位标志** 当MCU发生外部复位后，该位被硬件值1。建议通过软件清除该位。

25.4 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但是软件没有及时去清狗，看门狗定时器将直接复位系统，CPU将从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 (WDCON.3)将置位。除上电复位和自身看门狗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 (TA 保护)

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

地址: AAH　　复位值: 见表 6–2.

位	名称	描述
3	WDTRF	**看门狗复位标志** 当MCU发生看门狗定时器复位后，该位被硬件值1。建议通过软件清除该位。

25.5 软件复位

N76E616提供了软件复位功能，这样允许应用程序以软件方式复位整个系统，就像外部复位一样初始化MCU。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。写1 到SWRST (CHPCON.7) 触发软件复位。注该位为TA 保护。在芯片复位前芯片执行的最后一条指令就是设置SWRST位，见下面参考例程。

当发生软件复位时，SWRF (AUXR.7)会被硬件自动置位。用户可检查该位作为复位源指示。SWRF不会被其它复位改变除了上电复位和自身的软件复位。也可以通过软件清除该位。

CHPCON – 芯片控制 (TA 保护)

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

地址: 9FH　　复位值: 见表 6–2.

位	名称	描述
7	SWRST	**软件复位** 该位置1将会产生一个软件复位。软件复位完成后将会由硬件自动清除

AUXR1 – 辅助寄存器 1

7	6	5	4	3	2	1	0
SWRF	RSTPINF	T1LXTM	T0LXTM	GF2	-	0	DPS
R/W	R/W	R/W	R/W	R/W	-	R	R/W

地址: A2H 复位值: 见表 6–2.

位	名称	描述
7	SWRF	**软件复位标志** 当MCU发生软件复位后，该位被硬件值1。建议通过软件清除该位。

软件参考你程如下：

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

25.6 启动选项

N76E616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 还是LDROM 中启动。复位后，如果BS = 0，CPU从APPROM 地址0000H处启动。反之CPU从LDROM 地址0000H处启动。注意除软件复位外所有复位过后，都会加载CONFIG0.7的值再取反后赋给BS位。

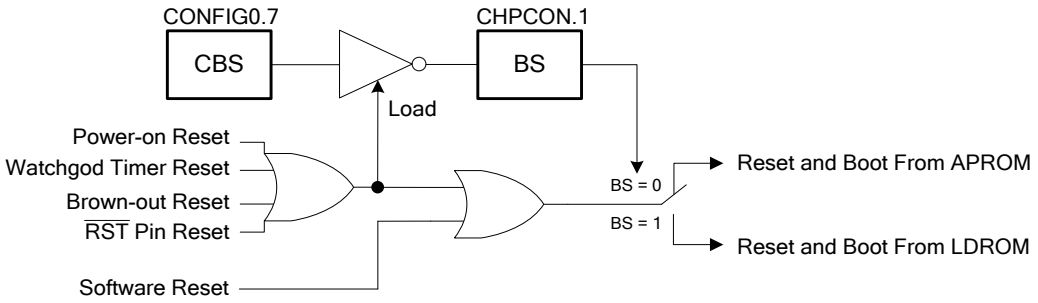


图 25.1. 启动选项框图

CONFIG0

7	6	5	4	3	2	1	0
CBS	-	-	OCDEN	-	RPD	LOCK	-
R/W	-	-	R/W	-	R/W	R/W	-

出厂默认值: 1111 1111b

位	名称	描述
7	CBS	**配置启动选项** 该位决定复位后MCU从哪个区块启动（除软件复位外） 1 = 复位后MCU由APROM启动（除软件复位外） 0 = 复位后MCU由LDROM启动（除软件复位外）

CHPCON – 芯片控制 (TA 保护)

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

地址: 9FH 复位值: 见表 6–2.

位	名称	描述
1	BS	**启动选项** 该位决定所有复位后MCU由哪块启动 0 =复位后MCU由APROM启动 1 =复位后MCU由LDROM启动

[1] 注该位由复位后（除软件复位外）CONFIG0.7的CBS位取反值初始化。软件复位后保持不变。

注MCU从所有复位状态释放后，硬件将检查 BS位以决定是由APROM还是LDROM启动。

25.7 复位状态

除了上电复位其它复位都不会影响片上RAM。复位期间，RAM中数据保持不变。上电复位后RAM的内容是不确定的。

复位后，除了一些受不同复位事件影响的位，大多数特殊寄存器恢复到初始值（详见表 6–2.）。当芯片一直处在复位状态时程序计数器会被强制保持为0000H。注堆栈指针复位为07H，因此，在复位期间，即使RAM内容不被修改，堆中内容也可能丢失。

复位后，所有外设和中断都被禁用。I/O口寄存器写入FFH，这些端口处于高阻输入模式。

26. 辅助功能

26.1 双DPTR

传统8051只有一个DPRT（数据指针），用单个DPTR将数据从一个地址移到另外一个地址需要很多代码并且执行效率又低。N76E616提供了两个数据指针这样在进行数据块移动时，软件可以同时加载源地址和目的地址。一次加载，软件只需简单设置DPS (AUXR1.0)位来切换DPRT和DPTR1数据指针。

下面是一个64字节块移动例程。将源地址和目的地址赋值给数据指针然后执行数据块搬移要比单个DPTR简单有效率的多。切换DPTR只需执行2个字节的最短指令INC AUXR1并不需要执行ORL或ANL。AUXR1.1硬件上始终保持为0，所以通过自加AUXR1改变DPS位并不会影响到该寄存器的其它位。

```
        MOV     R0,#64              ;number of bytes to move
        MOV     DPTR,#D_Addr        ;load destination address
        INC     AUXR1               ;change active DPTR
        MOV     DPTR,#S_Addr        ;load source address
LOOP:
        MOVX    A,@DPTR             ;read source data byte
        INC     AUXR1               ;change DPTR to destination
        MOVX    @DPTR,A             ;write data to destination
        INC     DPTR                ;next destination address
        INC     AUXR1               ;change DPTR to source
        INC     DPTR                ;next source address
        DJNZ    R0,LOOP
        INC     AUXR1               ;(optional) restore DPS
```

AUXR1也包含一个普通标志位GF2，这个位可以通过软件置位或是清除

DPL – 数据指针低字节

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

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** 16位数据指针低字节。DPL和DPH组成16数据指针DPTR用以寻址非易失内存或程序内存。DPS (DPS.0)位决定DPTR或DPTR1哪个被使用。

DPH –数据指针高字节

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

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** 16位数据指针高字节。DPH和DPL组成16数据指针DPTR用以寻址非易失内存或程序内存。DPS (DPS.0)位决定DPTR或DPTR1哪个被使用。

AUXR1 – 辅助寄存器 1

7	6	5	4	3	2	1	0
SWRF	RSTPINF	T1LXTM	T0LXTM	GF2	-	0	DPS
R/W	R/W	R/W	R/W	R/W	-	R	R/W

地址: A2H 复位值: 见表 6–2.

位	名称	描述
3	GF2	**普通标志位 2** 该位可以通过软件置位或是清除
1	0	**保留** 该位读总为0
0	DPS	**数据指针选择** 0 =数据指针0（DPTR）被使用（复位默认） 1 =数据指针1（DPTR1）被使用 在改变DPS后，之前未被使用的数据指针的值将保持不变

26.2 96位唯一识别码（UID）

在出厂前，每一颗N76E616都预烧录了96位的UID，这样可以保证其唯一性。用户只能通过IAP命令读取UID，详见 章节21.1 IAP 命令

27. 在片上调试 (OCD)

27.1 功能描述

N76E616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N76E616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。P3.6/$\overline{\text{RST}}$引脚也是必不可少的，它是用来控制OCD模式进入和退出的。N76E616的OCD和ICP功能是共享这3个引脚。

N76E616使用OCDDA、OCDCK、 和 P3.6/$\overline{\text{RST}}$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. P3.6/$\overline{\text{RST}}$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. P3.6/$\overline{\text{RST}}$配置成输入引脚时，必须和外部输入源断开

3.所有与OCDDA和OCDCK连接的外围器件必须断开

27.2 OCD限制条件

由于N76E616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P3.6/$\overline{\text{RST}}$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P3.4共享一个引脚，因此该引脚I/O功能或是其他功能都不能使用

3. OCDCK与P3.5共享一个引脚，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。

N76E616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	-	-	OCDEN	-	RPD	LOCK	-
R/W	-	-	R/W	-	R/W	R/W	-

出厂默认值: 1111 1111b

位	名称	描述
4	OCDEN	**OCD 使能** 1 = 禁用OCD 0 = 使能OCD

28. 在电路编程 (ICP)

Flash存储器可以通过在电路编程（ICP）进行编程。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。ICP可以在不用将控制器从系统中卸下对控制器再编程。

ICP需要三根信号线：$\overline{\text{RST}}$、ICPDA和 ICPCK。$\overline{\text{RST}}$被用作进入或退出ICP模式。ICPDA为数据输入和输出引脚。ICPCK为时钟输入引脚。用户想用ICP编程除了需要在电路板上预留这三根引脚外还要加上VDD和GND引脚。

新唐提供N76E616 ICP工具可以让用户很方便的通过新唐ICP工具进行编程。该工具依据MCU电气特性已经过新唐优化。在生产过程中保证了其稳定性，效率也会有所提高。更多细节内容，请访问新唐8位微控制器网站：

新唐 80C51 微控制器技术支持

29. 配置字（CONFIG ）

N76E616具有硬件配置字（CONFIG），设定这些配置字可用于安全位，系统时钟位等等。这些硬件配置位可通过编程器、ICP烧录器或IAP来配置。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，需要加载这些配置位到相应的寄存器位。这些加载发生在复位之后。这些寄存器位可以持续通过用户的软件控制。

注配置位标记为"-"的位应该保证不被编程器改写。

CONFIG0

7	6	5	4	3	2	1	0
CBS	-	-	OCDEN	-	RPD	LOCK	-
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。
4	OCDEN	**OCD 使能** 1 = 禁用OCD 0 = 使能OCD
2	RPD	**禁用复位脚** 1 =使能P3.6/$\overline{\text{RST}}$引脚复位功能。P3.6/$\overline{\text{RST}}$引脚作为外部复位引脚 0 =禁用P3.6/$\overline{\text{RST}}$引脚复位功能。P3.6/$\overline{\text{RST}}$引脚作为只输入引脚
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。

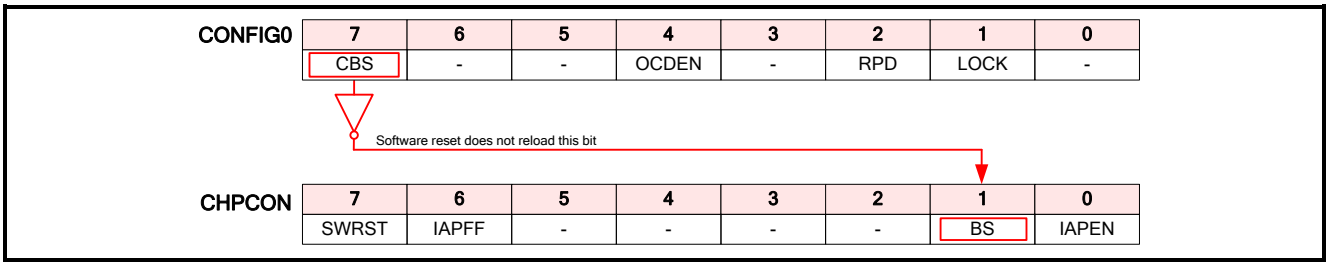


图 29.1. CONFIG0 复位后自动重载

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 大小选择** 选择 LDROM大小 111 = LDROM为0KB. APROM为 18KB 110 = LDROM为1KB. APROM为 17KB 101 = LDROM为2KB. APROM为 16KB 100 = LDROM为3KB. APROM为 15KB 0xx = LDROM为4KB. APROM为 14KB

CONFIG2

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

出厂默认值: 1111 1111b

位	名称	描述
7	CBODEN	**配置欠压检测使能** 1 =使能欠压检测 0 = 关闭欠压检测
5:4	CBOV[1:0]	**配置欠压检测电压选择** 11 = V_{BOD}为 2.2V. 10 = V_{BOD}为2.7V. 01 = V_{BOD}为3.8V. 00 = V_{BOD}为4.3V.
3	BOIAP	**欠压禁用IAP** 该位决定在欠压情况下是否禁用IAP擦除或是编程功能 1=当V_{DD}低于V_{BOD}时,禁用IAP擦除或编程功能 0= V_{DD}处在正常工作电压范围内都可以使用IAP擦除或编程功能
2	CBORST	**配置欠压检测复位使能** 该位决定欠压检测事件后是否发生欠压检测复位 1 =当V_{DD}下降到低于V_{BOD}，使能欠压检测复位。 0 =当V_{DD}下降到低于V_{BOD}，禁止欠压检测复位。

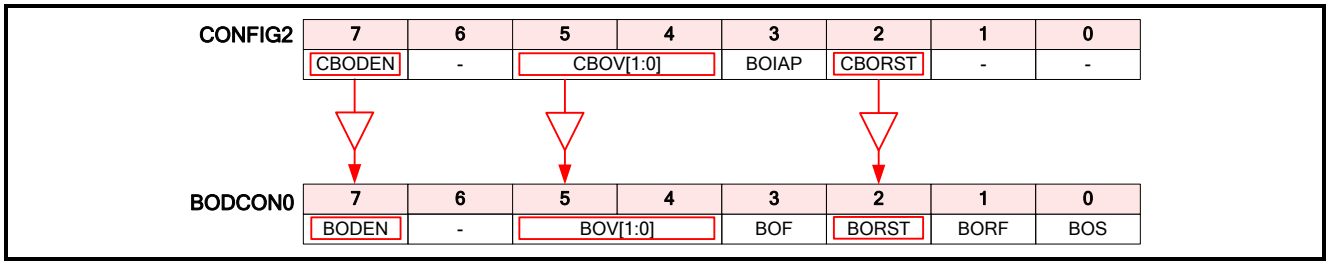


图 29.2. CONFIG2 上电复位重载位

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。 其它 = 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

30. 指令集

N76E616执行所有标准8051的指令集。然而,每条指令的时间是不同的，因为它使用的是高性能1T 8051的内核。本架构消除了冗余的总线状态并且并行执行取址、译码和执行阶段。N76E616每个机器周期只需一个时钟周期。使得在相对于传统12T 8051芯片工作在同样的时钟频率下性能要提高到8.1倍（在MIPS下）。然而，想要在系统中见到真实改善的速度需要依据系统执行的指令组合。

所有指令代码为8位机器码。单字节来源于程序存储器。机器码通过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 30–1列出所有指令，指令设定和寻址如下

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(0~127) 或SFR (I/O, 控制寄存器, 状态寄存器等(128~255)。
@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64k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SFR直接寻址

表 30–1. N76E616指令集

指令		机器码	占用字节	时钟周期	N76E616 与传统 80C51 速率对比
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

表 30–1. N76E616指令集

指令		机器码	占用字节	时钟周期	N76E616 与传统 80C51 速率对比
XRL	A, #data	64	2	2	6
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

表 30–1. N76E616指令集

指令		机器码	占用字节	时钟周期	N76E616 与传统 80C51 速率对比
SETB	C	D3	1	1	12
SETB	bit	D2	2	4	3
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] N76E616没有外部存储器总线，MOVX指令被用到访问内部XRAM。
[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]。

31. 电气特性

31.1 绝对最大额定值

参数	范围	单位
工作电压	-40 to +105	°C
存储温度	-55 to +150	°C
VDD 到 GND 电压	-0.3 to +6.3	V
其它引脚到GND电压	-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。

31.2 直流电气特性

表 31–1.直流电气特性表

符号	参数	条件	最小值	典型值	最大值	单位
工作电压						
V_{DD}	工作电压	F = 0 to 16 MHz	2.4	-	5.5	V
I/O						
V_{IL}	输入低电压 (标准端口TTL输入)		V_{DD}-0.3	-	0.2V_{DD}-0.1	V
V_{IL1}	输入低电压 (标准端口施密特输入和XIN)		V_{DD}-0.3	-	0.3V_{DD}	V
V_{IL2}	输入低电压(RST)		V_{DD}-0.3	-	0.2V_{DD}	V
V_{IH}	输入高电压 (标准端口TTL输入)		0.2V_{DD}+0.9	-	V_{DD}+0.3	V
V_{IH1}	输入高电压 (标准端口施密特输入，XIN和RST)		0.7V_{DD}	-	V_{DD}+0.3	V
V_{OL}	输出低电压[1] (标准输出能力，除输入高阻模式外所有模式)	V_{DD} = 4.5V, I_{OL} = 10mA	-	-	0.4	V
		V_{DD} = 3.0V, I_{OL} = 7mA	-	-	0.4	
		V_{DD} = 2.4V, I_{OL} = 5mA	-	-	0.4	
V_{OL1}	输出低电压[1] (P1.0 到 P1.3强输出能力，除输入高阻模式外所有模式)	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} = 18mA	-	-	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} = -360μA	2.4	-	-	V
		V_{DD} = 3.0V, I_{OH} = -90μA	2.4	-	-	
		V_{DD} = 2.4V, I_{OH} = -50μA	2.0	-	-	
V_{OH1}	输出高电压 (标准输出能力，推挽输出模式)	V_{DD} = 4.5V, I_{OH} = -20mA	2.4	-	-	V
		V_{DD} = 3.0V, I_{OH} = -5.5mA	2.4	-	-	
		V_{DD} = 2.4V, I_{OH} = -3mA	2.0	-	-	
V_{OH2}	输出高电压 (P1.0 到 P1.3强输出能力，推挽输出模式)	V_{DD} = 4.5V, I_{OH} = -32mA	2.4	-	-	V
		V_{DD} = 3.0V, I_{OH} = -8mA	2.4	-	-	
		V_{DD} = 2.4V, I_{OH} = -4mA	2.0	-	-	
I_{IL}	逻辑0输入电流 (准双向模式)	V_{DD} = 5.5V, V_{IN} = 0.4V	-	-	-75	μA
I_{TL}	逻辑1向0转换电流[2] (准双向模式)	V_{DD} = 5.5V	--	-500	-650	μA
I_{LI}	输入漏电流 (开漏模式或只输入模式)		-	1	±10	μA
R_{RST}	$\overline{RST}$复位脚内部上拉电阻		50	-	600	kΩ
工作电流						
I_{DD}	正常工作电流[3]	HXT, V_{DD} = 5.0V	-	5.1	-	mA
		HXT, V_{DD} = 3.3V	-	3.9	-	
		HIRC, V_{DD} = 5.0V	-	2.9	-	mA
		HIRC, V_{DD} = 3.3V	-	2.8	-	
		LXT, V_{DD} = 5.0V	-	180	-	μA
		LXT, V_{DD} = 3.3V	-	165	-	
		LIRC, V_{DD} = 5.0V	-	175	-	μA
		LIRC, V_{DD} = 3.3V	-	160	-	
I_{IDL}	空闲模式电流	HXT, V_{DD} = 5.0V	-	3.5	-	mA
		HXT, V_{DD} = 3.3V	-	2.2	-	
		HIRC, V_{DD} = 5.0V	-	1.8	-	mA
		HIRC, V_{DD} = 3.3V	-	1.7	-	
		LXT, V_{DD} = 5.0V	-	180	-	μA
		LXT, V_{DD} = 3.3V	-	165	-	
		LIRC, V_{DD} = 5.0V	-	175	-	μA
		LIRC, V_{DD} = 3.3V	-	160	-	
I_{PD}	掉电模式电流 (BOD 关闭, LXT 关闭)	T_A = 25℃	-	6.5	8	μA
		T_A = -40℃ to +105℃	-	40	80	μA

符号	参数	条件	最小值	典型值	最大值	单位
I_{PD1}	掉电模式电流 (BOD 关闭, LXT 开启, XTGS[1:0] = [0,1])	T_A = 25℃	-	8.5	11	μA
		T_A = -40℃ to +105℃	-	44	85	μA

[1] 基于稳定（非暂态）状态条件下，I_{OL}外部限制如下：
每个管脚最大I_{OL}: 40mA
所有管脚I_{OL} 总和输出: 120mA

[2]所有管脚在准双向模式下当被外部从1拉到0时，在V_{IN}接近2V时转换电流达到最大值

[3] MCU执行“SJMP $”循环指令下测得，所有管脚配置成准双向模式

31.3 交流电气特性

表 31–2. 系统时钟交流电气特性

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	-	16	MHz
	高速晶体振荡器频率(HXT)	2	-	16	
	低速晶体振荡器频率(LXT)	-	32.768	-	kHz
t_{CHCX}	外部时钟输入高电平时间	30	-	-	ns
t_{CLCX}	外部时钟输入低电平时间	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

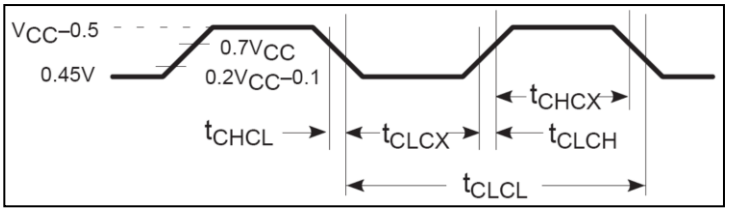


图 31.1. 外部时钟输入时序

表 31–3.内部振荡器交流电气 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}	高速 11.059 MHz 振荡频率 (HIRC)	V_{DD} = 5.0V, T_A = 25℃	±1%	10.948	11.059	11.170	MHz
		V_{DD} = 2.4V to 5.5V, T_A = -40℃ to +105℃	±5%	10.506		11.612	
F_{LIRC}	低速 10 kHz 振荡频率 (LIRC)	V_{DD} = 2.4V to 5.5V, T_A = -40℃ to +105℃	50%	5	10	15	kHz

表 31–4. 掉电唤醒交流电气特性

符号	参数	条件	最小值	典型值	最大值	单位
T_{PDWK}	掉电唤醒时间	HIRC	-	40	-	μs
		HXT, F = 16 MHz	-	600	-	

表 31–5. 外部复位脚交流电气特性

符号	参数	条件	最小值	典型值	最大值	单位
T_{RST}	$\overline{RST}$ 复位脚侦测脉冲宽度		-	24/F_{SYS}	450	μs

31.4 模拟电气特性

表 31–6. POR上电复位电气特性

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		1.6	2.0	2.4	V
T_{PORRD}	上电复位释放延时		-	5	-	ms

表 31–7. BOD欠压电气特性

符号	参数	条件	最小值	典型值	最大值	单位
V_{BOD0}	欠压阈值 4.3V	BOV[1:0] = [0,0]	4.1	4.3	4.5	V
V_{BOD1}	欠压阈值3.8V	BOV[1:0] = [0,1]	3.5	3.7	3.9	V
V_{BOD2}	欠压阈值2.7V	BOV[1:0] = [1,0]	2.55	2.7	2.85	V
V_{BOD3}	欠压阈值2.2V	BOV[1:0] = [1,1]	2.05	2.2	2.35	V
V_{BODHYS}	欠压滞后性	LPBOD[1:0] = [0,0] only	30	-	200	mV
I_{BOD}	欠压静态电流	V_{DD} = 5V, LPBOD[1:0] = [0,0]	-	180	260	μA
		V_{DD} = 5V, LPBOD[1:0] = [0,1]	-	25	35	
		V_{DD} = 5V, LPBOD[1:0] = [1,0]	-	6	9	
		V_{DD} = 5V, LPBOD[1:0] = [1,1]	-	2	3.5	
T_{BOD}	欠压 侦测脉冲宽度		详见 表 24–2			-
T_{BODEN}	欠压启动时间		2	-	3	$1/F_{LIRC}$

表 31–8. 带隙基准电压电气特性

符号	参数	条件	最小值	典型值	最大值	单位
V_{BG}	带隙基准电压		1.15	1.21	1.27	V
T_{BGEN}	带隙基准电压启动时间		1	-	2	$1/F_{LIRC}$

表31–9. 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_{DD} = 5V	-	500	650	μA
V_{AIN}	模拟输入电压	-	0	-	V_{DD}	V
N_R	分辨率	V_{DD} = 2.4V to 5.5V		10		位
DNL	微分非线性误差	V_{DD} = 5V, T_A = 25℃, 采样率 = 300 ksps	-	+3.5	-	LSB

符号	参数	条件	最小值	典型值	最大值	单位
		V_{DD} = 2.4V to 5.5V, T_A = -40℃ to +105℃, 采样率 = 300 ksps	-	-	+6	LSB
		V_{DD} = 2.4V to 5.5V, T_A = -40℃ to +105℃, 采样率 = 150 ksps	-	-	+2	LSB
INL	积分非线性误差	V_{DD} = 5V, T_A = 25℃, 采样率 = 300 ksps	-	+2	-	LSB
		V_{DD} = 2.4V to 5.5V, T_A = -40℃ to +105℃, 采样率 = 300 ksps	-4.5	-	+2.5	LSB
		V_{DD} = 2.4V to 5.5V, T_A = -40℃ to +105℃, 采样率 = 150 ksps	-1.5	-	+1.5	LSB
OE	偏移误差	V_{DD} = 2.4V to 5.5V, T_A = -40℃ to +105℃, 采样率 = 300 ksps	-1	-	+3	LSB
		V_{DD} = 2.4V to 5.5V, T_A = -40℃ to +105℃, 采样率= 150 ksps	-1	-	+2.5	LSB
FE	满量程误差	V_{DD} = 2.4V to 5.5V, T_A = -40℃ to +105℃, 采样率 = 300 ksps	-3	-	-0.5	LSB
TUE	总不可调整误差	V_{DD} = 5V, T_A = 25℃, 采样率 = 300 ksps	-	+5	-	LSB
		V_{DD} = 2.4V to 5.5V, T_A = -40℃ to +105℃, 采样率 = 300 ksps	-	-	+8	LSB
		V_{DD} = 5V, T_A = 25℃, T_A = -40℃ to +105℃, 采样率 = 150 ksps	-	+2.5	-	LSB
		V_{DD} = 2.4V to 5.5V, T_A = -40℃ to +105℃, 采样率 = 150 ksps	-	-	+4	LSB
-	一致性	V_{DD} = 2.4V to 5.5V	保证			-
F_{ADC}	ADC 时钟频率	V_{DD} = 2.4V to 5.5V	0.01	-	4	MHz
T_S	采样时间(软件可调)	V_{DD} = 2.4V to 5.5V	6	-	261	$1/F_{ADC}$
T_{CONV}	ADC总转换时间	V_{DD} = 2.4V to 5.5V	T_S + 12			$1/F_{ADC}$
T_{ADCEN}	ADC 输入等效电阻	V_{DD} = 2.4V to 5.5V	-	-	10	μs
R_{IN}	ADC 输入等效电容	V_{DD} = 2.4V to 5.5V	-	2.5	-	kΩ

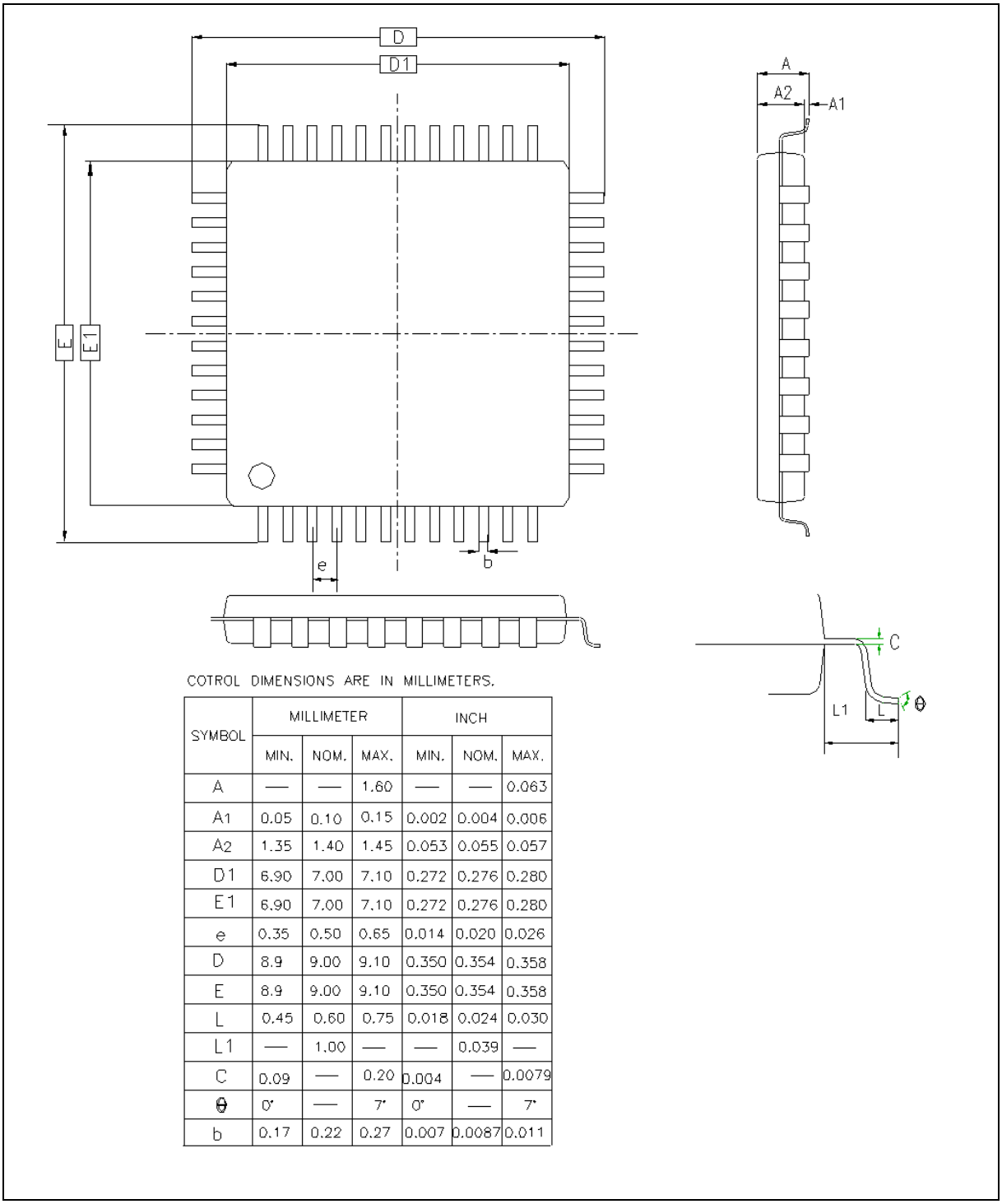
符号	参数	条件	最小值	典型值	最大值	单位
C_{IN}	ADC 工作电压	V_{DD} = 2.4V to 5.5V	-	3.6	-	pF

表 31–10. LCD 驱动电路电气特性

符号	参数	条件	最小值	典型值	最大值	单位
V_{LCD}	LCD 工作电压		3.0	-	5.5	V
I_{LCD}	LCD 工作电流	V_{LCD} = 5V, RSEL[1:0] = [0,0], 帧频 = 64 Hz, 1/4 占空比, 1/3 偏压, 所有SEG管脚使能，显示断开	-	35	40	μA
		V_{LCD} = 5V, RSEL[1:0] = [0,1], 帧频= 64 Hz, 1/4占空比, 1/3偏压, 所有SEG管脚使能，显示断开	-	18	20	
		V_{LCD} = 5V, RSEL[1:0] = [1,0], 帧频= 64 Hz, 1/4占空比, 1/3偏压, 所有SEG管脚使能，显示断开	-	10	11	

32. 封装信息

32.1 48脚 LQFP 7x7x1.4mm

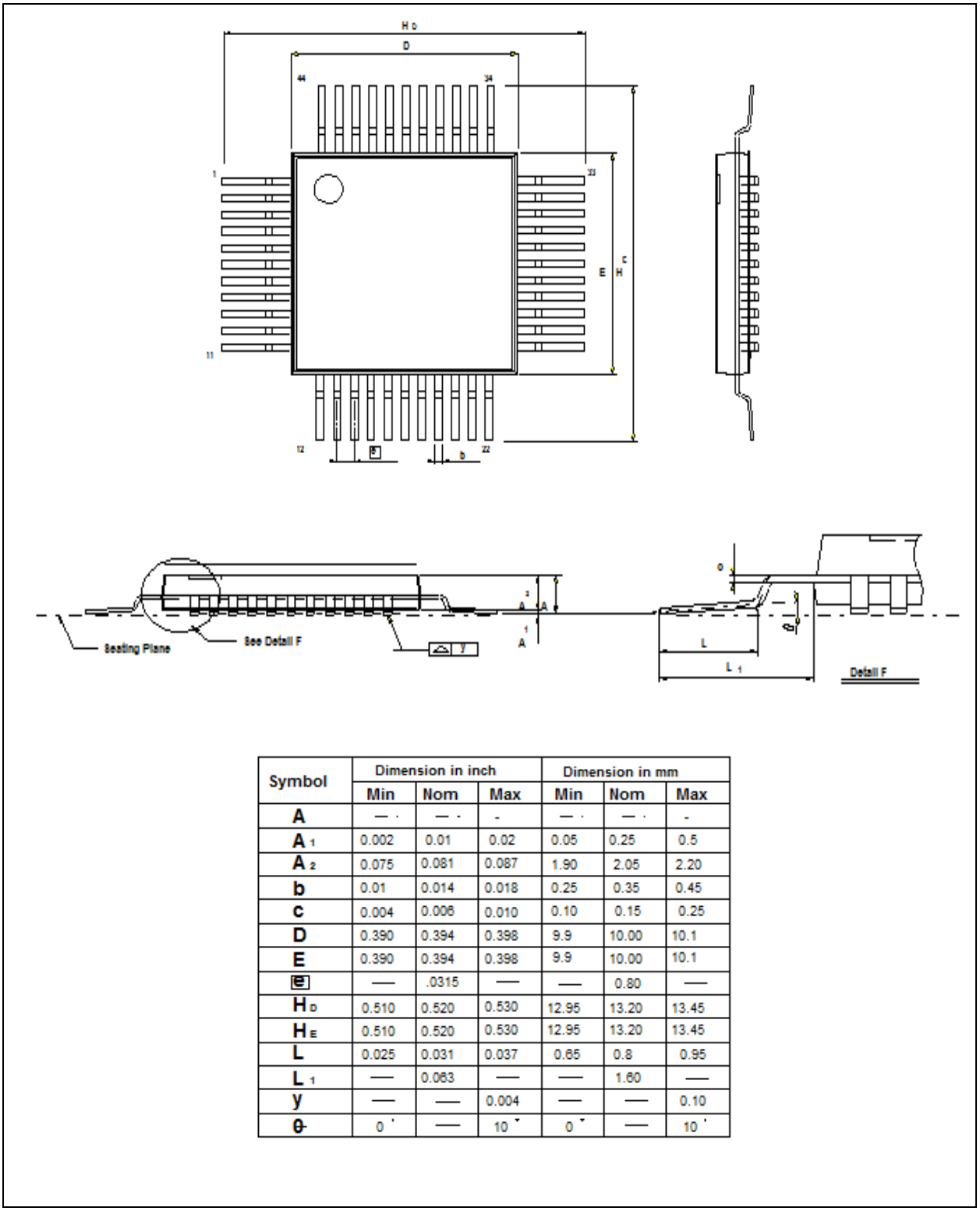


COTROL DIMENSIONS ARE IN MILLIMETERS.

SYMBOL	MILLIMETER			INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	—	—	1.60	—	—	0.063
A1	0.05	0.10	0.15	0.002	0.004	0.006
A2	1.35	1.40	1.45	0.053	0.055	0.057
D1	6.90	7.00	7.10	0.272	0.276	0.280
E1	6.90	7.00	7.10	0.272	0.276	0.280
e	0.35	0.50	0.65	0.014	0.020	0.026
D	8.9	9.00	9.10	0.350	0.354	0.358
E	8.9	9.00	9.10	0.350	0.354	0.358
L	0.45	0.60	0.75	0.018	0.024	0.030
L1	—	1.00	—	—	0.039	—
C	0.09	—	0.20	0.004	—	0.0079
θ	0°	—	7°	0°	—	7°
b	0.17	0.22	0.27	0.007	0.0087	0.011

图 32-1 LQFP-48 (7x7x1.4mm footprint 2.0mm) 封装信息

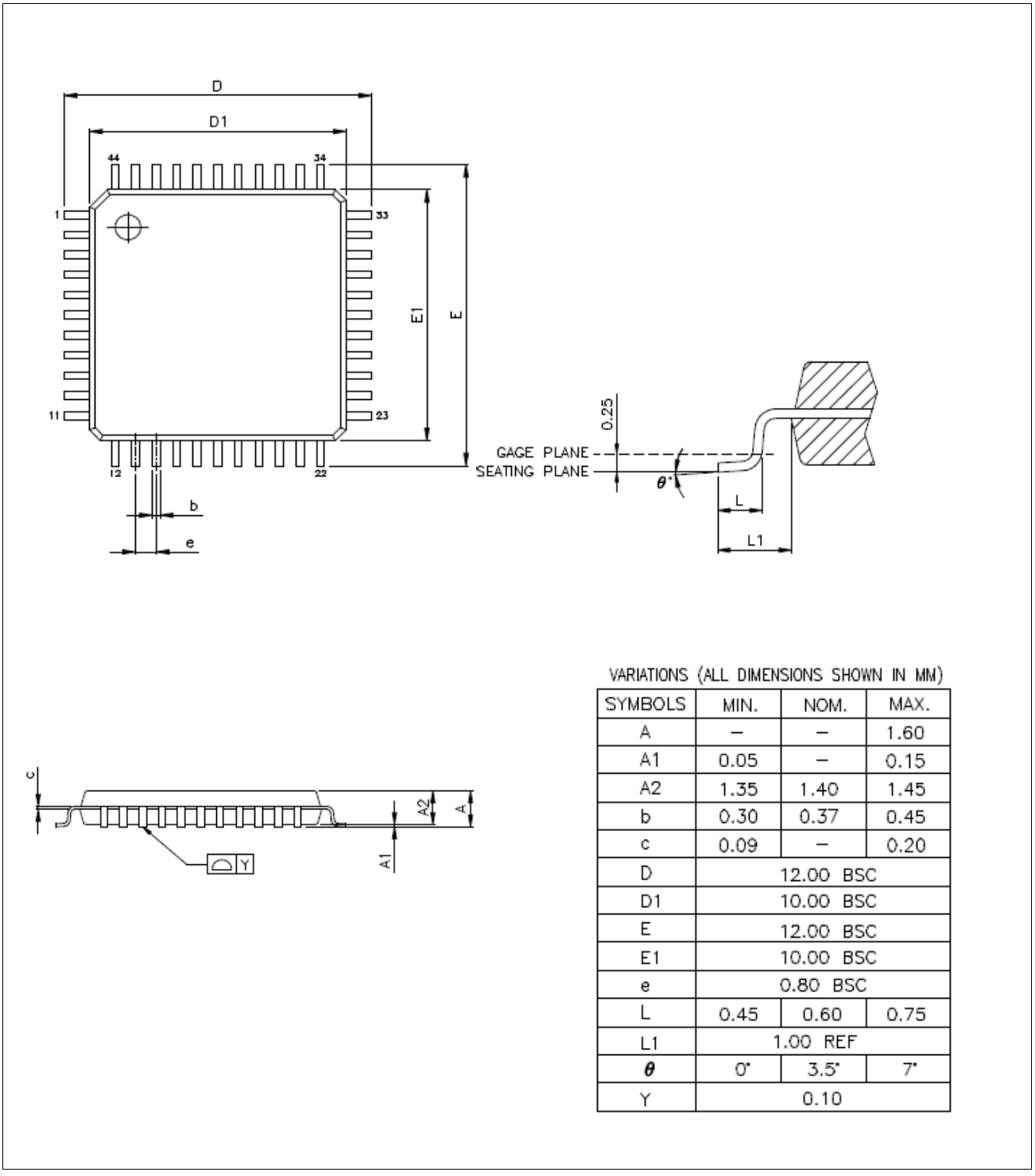
32.2 44脚 PQFP 10x10x2.0mm



Symbol	Dimension in inch			Dimension in mm		
	Min	Nom	Max	Min	Nom	Max
A	—	—	-	—	—	-
A_1	0.002	0.01	0.02	0.05	0.25	0.5
A_2	0.075	0.081	0.087	1.90	2.05	2.20
b	0.01	0.014	0.018	0.25	0.35	0.45
c	0.004	0.006	0.010	0.10	0.15	0.25
D	0.390	0.394	0.398	9.9	10.00	10.1
E	0.390	0.394	0.398	9.9	10.00	10.1
e	—	.0315	—	—	0.80	—
H_D	0.510	0.520	0.530	12.95	13.20	13.45
H_E	0.510	0.520	0.530	12.95	13.20	13.45
L	0.025	0.031	0.037	0.65	0.8	0.95
L_1	—	0.063	—	—	1.60	—
y	—	—	0.004	—	—	0.10
θ	0 °	—	10 °	0 °	—	10 °

图 32-2 PQFP44 (10x10x2.0mm, footprint 3.2mm) 封装信息

32.3 44脚 LQFP 10x10x1.4mm



VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

SYMBOLS	MIN.	NOM.	MAX.
A	–	–	1.60
A1	0.05	–	0.15
A2	1.35	1.40	1.45
b	0.30	0.37	0.45
c	0.09	–	0.20
D	12.00 BSC		
D1	10.00 BSC		
E	12.00 BSC		
E1	10.00 BSC		
e	0.80 BSC		
L	0.45	0.60	0.75
L1	1.00 REF		
θ	0°	3.5°	7°
Y	0.10		

图 32-3 LFP44 (10x10x1.4 mm, Footprint 2.0 mm)封装信息

33. 文件版本

版本	日期	描述
V1.00	2015/8/10	初次发布
V1.01	2016/2/20	修改管脚分布芯片名称
V1.02	2016/10/25	1. 增加 PQFP44 封装信息，管脚分布。 2. 章节 12.1 增加 WDT 清零条件说明。 3. 章节 17.1.1 修正 ADCAQT 数值描述。 4. 章节 23 修正 HIRC 全温全压偏差至 5%。 5. 章节 21.3 修正 FLASH 写次数为 20,000 次。
V1.03	2018/1/5	修正N76E616AL48 管脚描述管脚编号。
V1.04	2018/3/2	章节2、4及32 增加LQFP44封装相关信息