MEGAWIN
MAKE YOU WIN

8051 内核-单片机

MPC82G516
规格书

北京菱电科技有限公司
TEL:010-82674978

版本: A1.0

功能

- ⌘ 一般8051功能
  - 8051兼容的指令集
  - 256字节随机存取储存器
  - 64K外部的数据存储器空间
  - 四个8个位双向I/O口
  - 三个16个位定时器/计数器
  - 全双工 UART
  - 14中断源，4个优先级
  - 节能模式：空闲模式，掉电模式
- ⌘ 高速 1-T 结构 80C51 内核
- ⌘ 片上 64KB Flash, 60KB程序存储器, 3KB IAP 資料存储器, 1KB ISP 引導程序空間
- ⌘ 片上 1024 字节扩展RAM (XRAM)
- ⌘ 额外可位寻址的 I/O 口, P4
- ⌘ I/O口结构类型
  - 准双向输出
  - 开漏输出
  - 仅输入
  - 推挽式的输出
- ⌘ 额外的外部中断 /INT2 & /INT3
- ⌘ Timer2减计数能力
- ⌘ 增强 UART 功能
  - 帧错误侦测
  - 自动地址匹配
- ⌘ 第二个 UART和配套的波特率产生器
- ⌘ 6单元PCA (可编程计数器阵列)
  - 捕捉模式
  - 16位软件定时器模式
  - 高速输出模式
  - PWM (脉冲宽度调变器) 模式
- ⌘ SPI 接口 (主/从模式)
- ⌘ 10位8通道ADC转换器
- ⌘ 8输入辅助键盘中断
- ⌘ 外部中断唤醒掉电模式
- ⌘ 3个可编程时钟输出
- ⌘ 看门狗定时器
- ⌘ 双数据指针
- ⌘ 低速外部存储器的MOVX时间延展
- ⌘ 可配置系统时钟减少耗电量
- ⌘ 电源监视功能：掉电检测和上电标志
- ⌘ ISP (在系统编程) 更新程序存储器: 1KB ISP 空間
- ⌘ IAP (在应用编程) 为应用程序写非易失性数据: 3KB IAP 空間
- ⌘ Flash 寿命: 20,000次擦写循环
- ⌘ 时钟频率: 最高 24MHz
- ⌘ 电源: 2.4V~3.6V (3.3V 系统), or 2.7V~5.5V (5V or 宽电压范围系统)

- 温度等级: -40 to +85 °C
- 封装: PDIP40, PLCC44, PQFP44, LQFP48

1 目录

功能 3
1 目录 5
2 综述 8
3 方框图 9
4 引脚 10
4.1 引脚结构 10
4.2 引脚定义 14
4.3 引脚功能重映射 16
5 存储器组织 17
5.1 程序存储器 17
5.2 数据存储器 19
5.3 关于 C51 编译器的声明识别符 23
6 特殊功能寄存器 (SFRs) 24
6.1 SFR 映射位置 24
6.2 SFR 描述 25
7 片上扩展 RAM (XRAM) 30
7.1 在软件中使用 XRAM 30
8 外部数据存储器的存取 32
8.1 配置 ALE 引脚 32
8.2 低速存储器的存取时间延展 32
9 双数据指针寄存器(DPTR) 35
10 I/O 结构 36
10.1 配置 I/O 36
10.2 I/O 口用作 ADC 功能 38
10.3 I/O 口注意事项 38
10.4 GPIO 示例代码 39
11 定时器/计数器 40
11.1 定时器 0 和定时器 1 40
11.2 定时器 2 44
11.3 定时器 0/1 示例代码 50
12 串行口 52
12.1 标准 UART 52
12.2 扩展的 UART 功能 55
13 第 2 个 UART (UART2) 58
13.1 UART2 配置寄存器 58
13.2 UART2 波特率 59
13.3 标准 UART 使用 UART2 的波特率发生器 60
13.4 UART2 波特率发生器的可编程时钟输出 60
13.5 串行口示例代码 62
14 可编程计数器阵列(PCA) 64
14.1 PCA 概述 64
14.2 PCA 定时/计数器 64
14.3 比较/捕获模块 66
14.4 PCA 模式设置 67
14.5 PCA 示例代码 70
15 串行外设接口(SPI) 72
15.1 典型 SPI 配置 73
15.2 SPI 配置 75

15.3 从机注意事项 76
15.4 主机注意事项 76
15.5 /SS 引脚的模式改变 76
15.6 数据冲突 76
15.7 SPI 时钟频率选择 76
15.8 数据模式 77
15.9 SPI 示例代码 79
16 模数（A/D）转换器 83
16.1 ADC 控制寄存器 83
16.2 ADC 操作 84
16.3 ADC 注意事项 85
16.5 ADC 示例代码 86
17 键盘中断 88
17.1 键盘中断示例代码 89
18 看门狗定时器（WDT） 90
18.1 WDT 控制寄存器 90
18.2 WDT 操作 90
18.3 WDT 示例代码 91
18.4 掉电和待机模式下的 WDT 91
18.5 WDT 硬件初始化 92
18.6 WDT 示例代码 93
19 中断系统 94
19.1 中断源 94
19.2 与中断相关的寄存器 96
19.3 中断使能 97
19.4 中断优先级 98
19.5 中断响应 98
19.6 外部中断 99
19.7 单步运行 99
19.8 中断示例代码 100
20 ISP & IAP 101
20.1 Flash 存储器 102
20.2 ISP 操作 103
20.3 IAP 操作 111
21 节能模式 114
21.1 空闲模式 114
21.2 休眠模式 115
21.3 时钟降速 115
22 系统时钟 116
22.1 内置振荡器 116
23 电源监测功能 117
23.1 上电监测 117
23.2 掉电监测 118
24 复位源 119
24.1 上电复位 119
24.2 RST 引脚硬件复位 119
24.3 看门狗复位 119
24.4 软件复位 120
24.5 掉电复位 120
25 硬件熔丝位选项 121
26 指令集 122
26.1 算术运算指令 123

26.2 逻辑操作指令 124
26.3 数据传送指令 125
26.4 布尔操作指令 126
26.5 控制和转移指令 127
27 应用事项 128
27.1 V30 供电 128
27.2 复位电路 128
27.3 晶振电路 129
28 片上调试功能 130
29 极限参数 131
30 直流特性 132
31 订货信息 136
32 封装尺寸 137
33 免责声明 141
34 版本历史 142

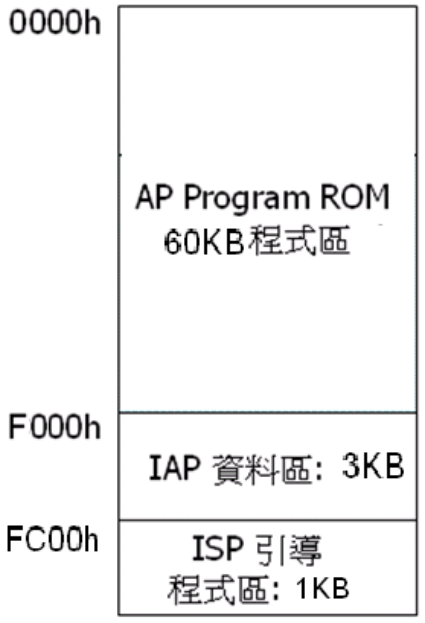
2 综述

MPC82G516 是基于80C51的高效1-T结构的单芯片微处理器，每条指令需要1~7个时钟信号 (比标准8051快6~7倍)，与8051指令集兼容。因此在与标准8051有同样的处理能力的情况下，MPC82G516只需要非常低的运行速度，同时由此能很大程度的减少耗电量。

MPC82G516拥 有 64K字节的内置Flash存储器用于保存代码和数据。Flash存储器可以通过并行模式编程，也拥有通过在系统编程(ISP)进行编程的能力。同时，也提供在应用编程(IAP)的能力。ISP让使用者无需从产品中取下微控制器就可以下载新的代码；IAP意味着应用程序正在运行时，微控制器能够在Flash中写入非易失数据。这些功能都由内建的电荷泵提供编程用的高压。

除了8051 MCU的标准功能(例如 256 字节的随机存储器，四个8位I/O口，三个定时/计数器，全双工的串口和 一个多源4级中断控制)外，许多系统级的功能已经集成到MPC82G516内部。这些功能有1024字节的扩展随机存储器(XRAM)， 一个额外的 I/O口 (P4)，10位的模/数转换器，PCA，SPI，第二个UART接口，辅助键盘中断， 一个看门狗定时器等等。这些功能能够有效地减少电路板面积和系统成本，而且这些功能使得 MPC82G516 在广泛的应用领域内成为一种强有力的微控制器。

MPC82G516有两种节能模式和8位的系统时钟分频器，以减少耗电量。在空闲模式下，CPU被冻结而外围模块和中断系统依然活动。在掉电模式下，随机存储器RAM和特殊功能寄存器SFR的能容被保存，而其他所有功能被终止。最重要的是，在掉电模式下的微控制器可以被外部中断唤醒。同时使用者可以通过8位的系统时钟分频器减慢系统速度以减少耗电量。

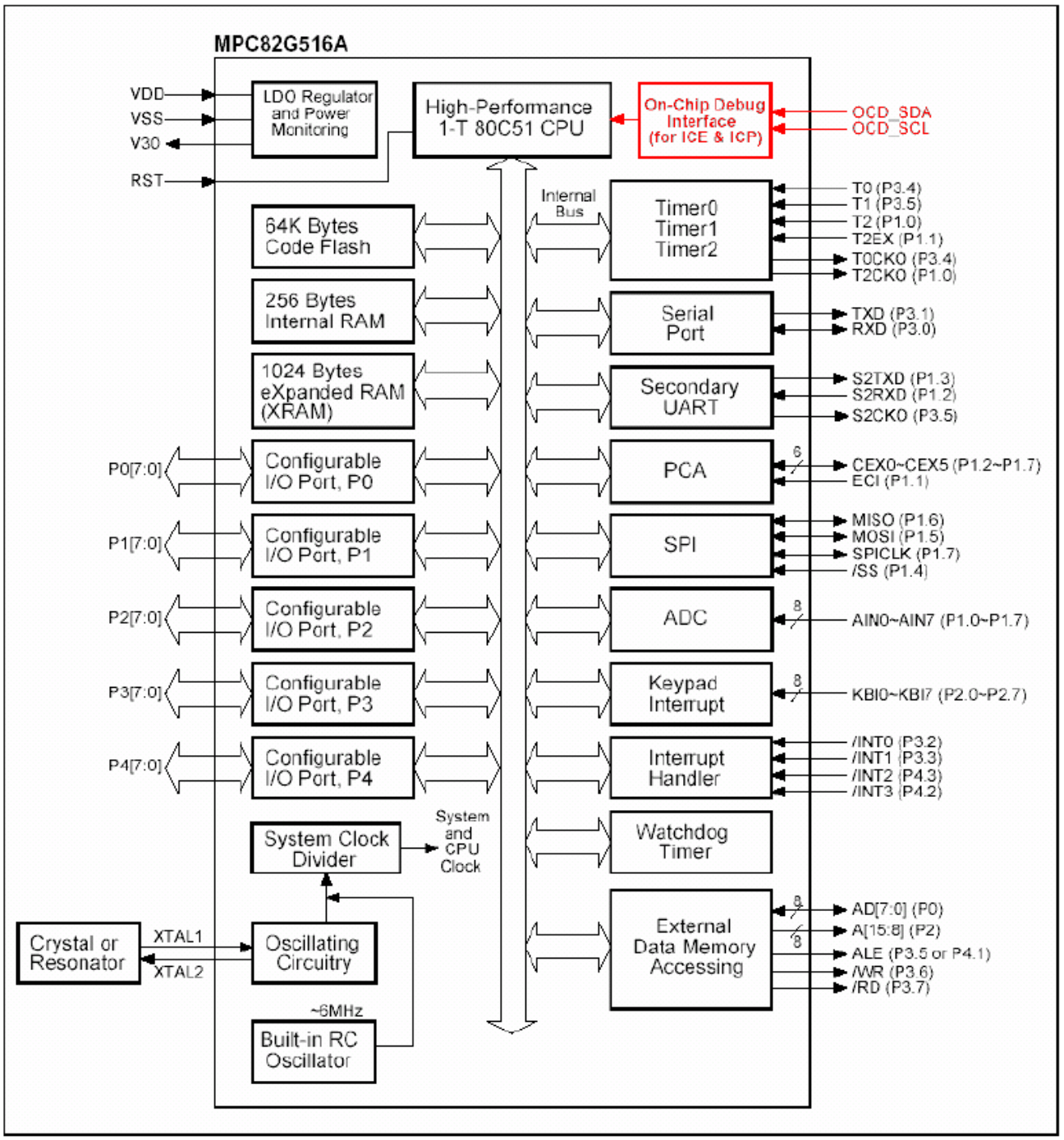


MPC82G516程式區與
IAP, ISP 區域分配說明

3 方框图

Figure 3-1 MPC82G516功能方框图。

Figure 3-1. 方框图



4 引脚

4.1 引脚结构

Figure 4-1. 引脚结构: 40-Pin PDIP

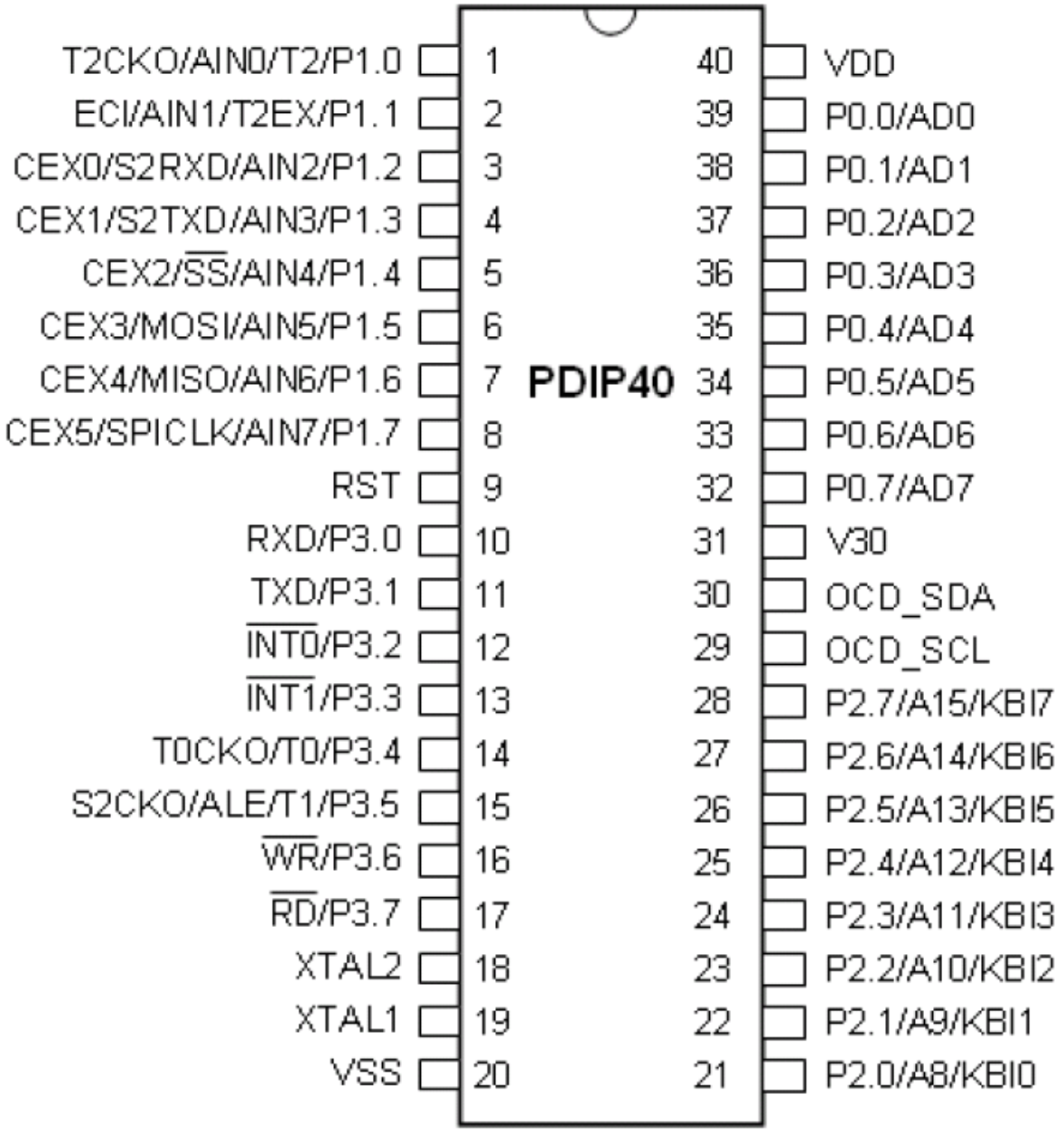


Figure 4-2. 引脚结构: 44-Pin PLCC

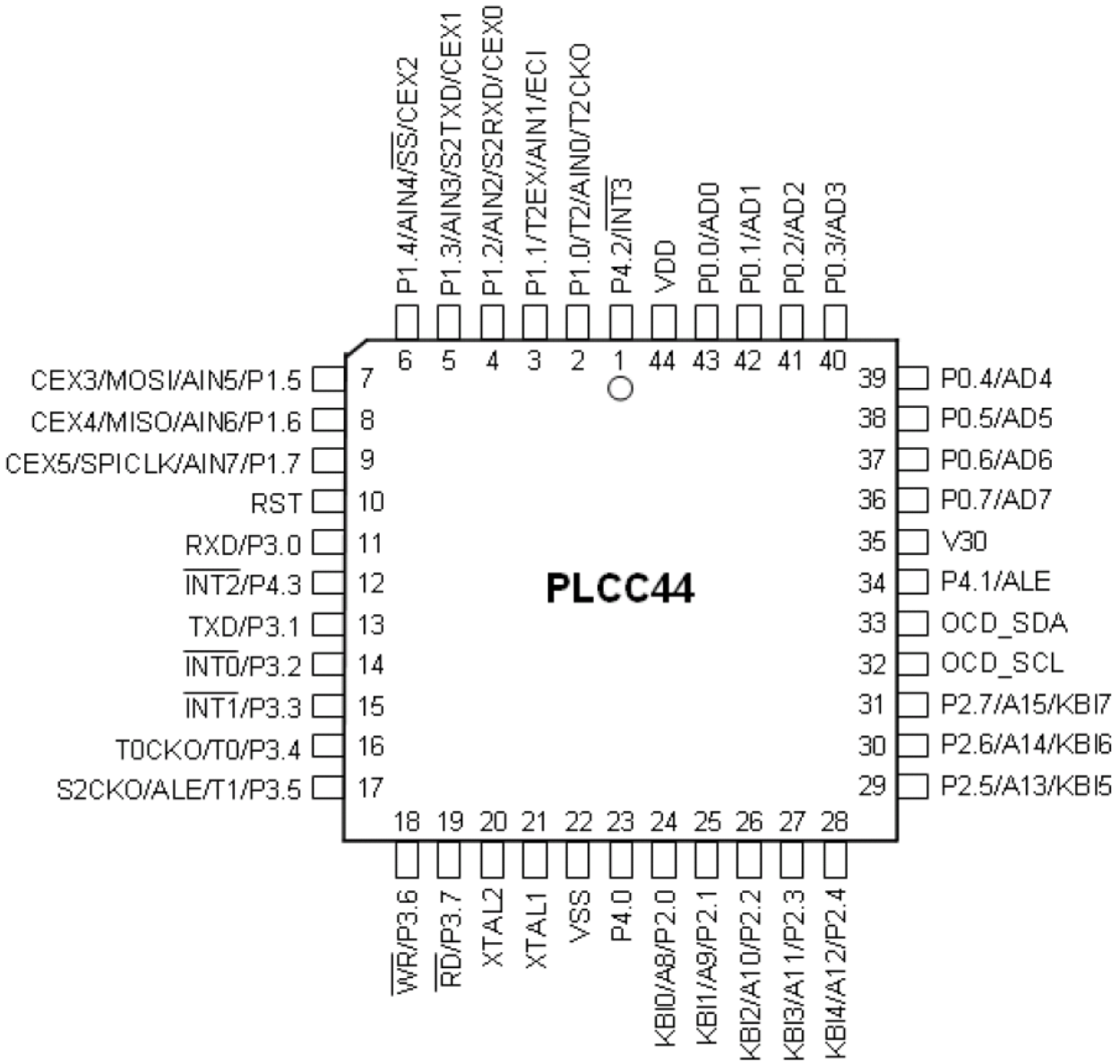


Figure 4-3. 引脚结构: 44-Pin PQFP

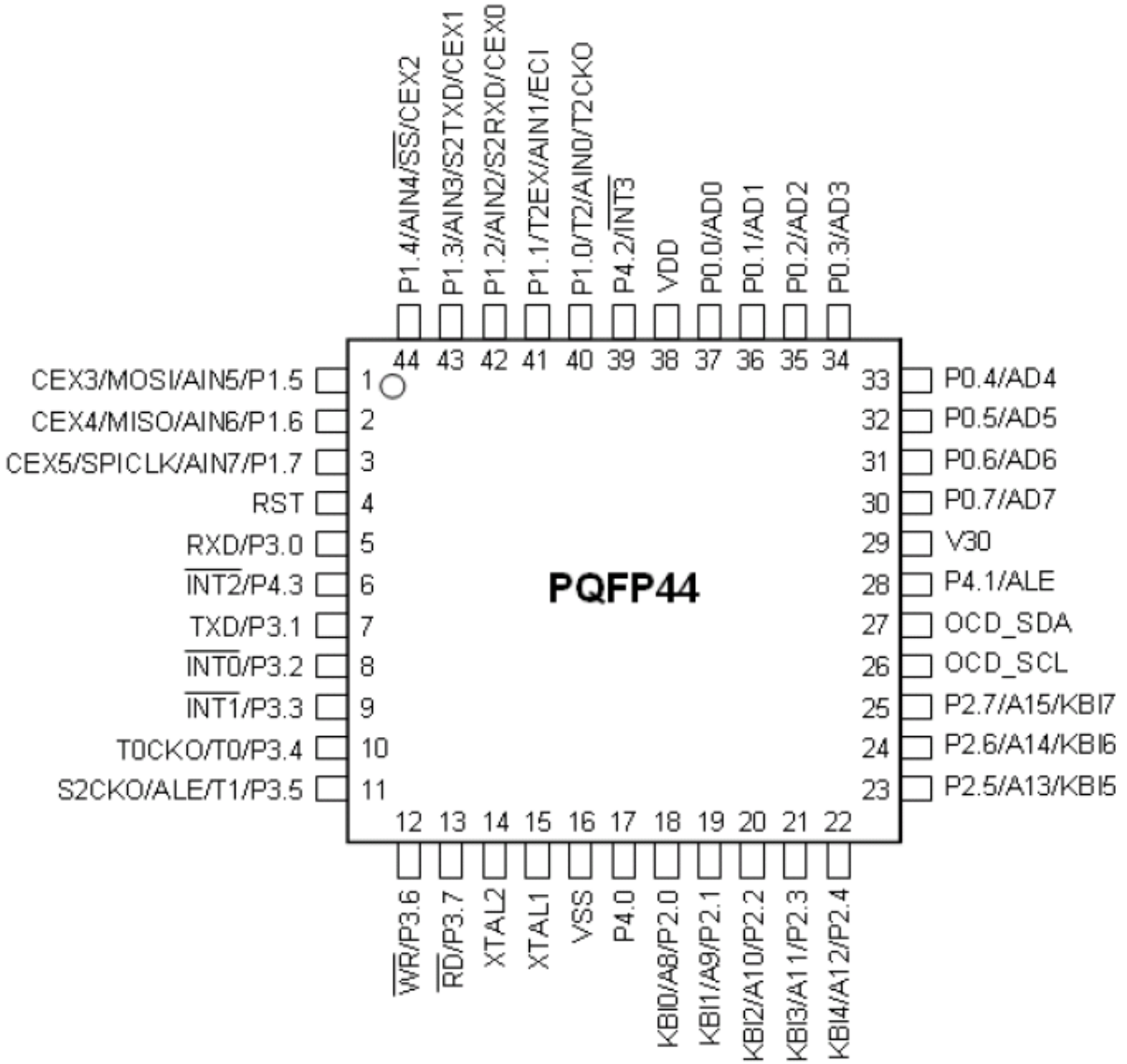
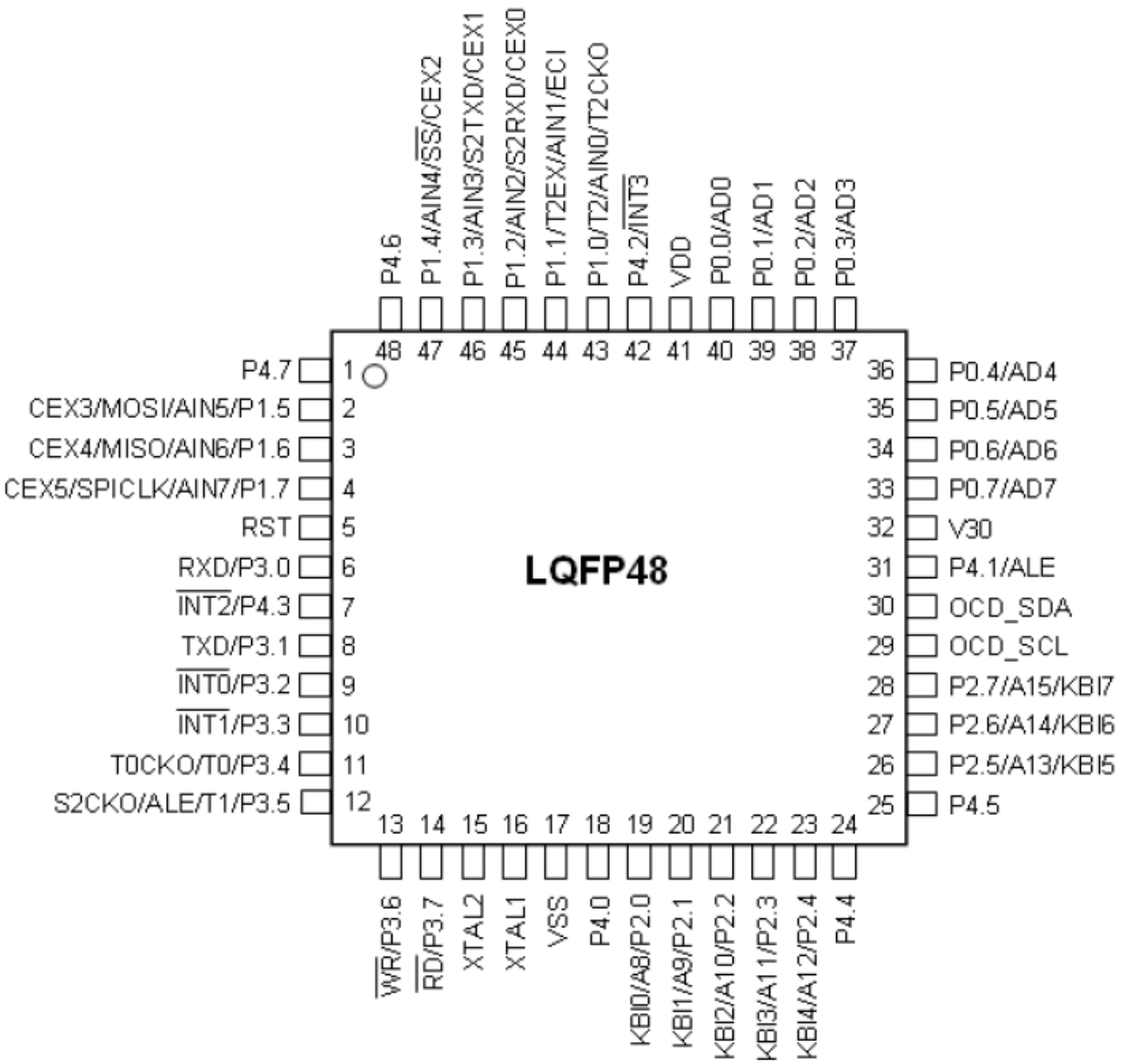


Figure 4-4. 引脚结构: 48-Pin LQFP



4.2 引脚定义

Table 4-1. 引脚定义

助记符	引脚号					I/O 类型	描述
	40-Pin DIP	44-Pin PLCC	44-Pin PQFP	48-Pin LQFP			
P0.0 (附加功能) AD0	39	43	37	40		I/O I/O	* 端口0 位0. * AD0: 读写外部数据存储器A0/D0复用
P0.1 (附加功能) AD1	38	42	36	39		I/O I/O	* 端口0 位-1. * AD1: 读写外部数据存储器A1/D1复用
P0.2 (附加功能) AD2	37	41	35	38		I/O I/O	* 端口 0 位-2. * AD2: 读写外部数据存储器A2/D2复用.
P0.3 (附加功能) AD3	36	40	34	37		I/O I/O	* 端口 0 位-3. * AD3: 读写外部数据存储器A3/D3复用
P0.4 (附加功能) AD4	35	39	33	36		I/O I/O	* 端口 0 位-4. * AD4: 读写外部数据存储器A4/D4复用
P0.5 (附加功能) AD5	34	38	32	35		I/O I/O	* 端口 0 位-5. * AD5: 读写外部数据存储器A5/D5复用
P0.6 (附加功能) AD6	33	37	31	34		I/O I/O	* 端口 0 位-6. * AD6: 读写外部数据存储器A6/D6复用
P0.7 (附加功能) AD7	32	36	30	33		I/O I/O	* 端口 0 位-7. * AD7: 读写外部数据存储器A7/D7复用.
P1.0 (附加功能) T2 (附加功能) AIN0 (附加功能) T2CKO	1	2	40	43		I/O I I O	* 端口 1 位-0. * T2: 定时/计数器2的外部输入. * AIN0: ADC 模拟量输入通道0. * T2CKO: 定时器2的可编程时钟输出.
P1.1 (附加功能) T2EX (附加功能) AIN1 (附加功能) ECI	2	3	41	44		I/O I I I	* 端口 1 位-1. * T2EX: 定时/计数器2重装入/捕获/方向控制. * AIN1: ADC模拟量输入通道1.. * ECI: PCA 外部时钟输入.
P1.2 (附加功能) AIN2 (附加功能) S2RXD (附加功能) CEX0	3	4	42	45		I/O I I I/O	* 端口 1 位-2. * AIN2: ADC模拟量输入通道2. * S2RXD: 第二 UART 串行输入. * CEX0: PCA 单元0外部I/O.
P1.3 (附加功能) AIN3 (附加功能) S2TXD (附加功能) CEX1	4	5	43	46		I/O I O I/O	* 端口 1 位-3. * AIN3: ADC模拟量输入通道3 * S2TXD: 第二 UART 串行输出 * CEX1: PCA单元1外部I/O..
P1.4 (附加功能) AIN4 (附加功能) /SS (附加功能) CEX2	5	6	44	47		I/O I I I/O	* 端口 1 位-4. * AIN4: ADC模拟量输入通道4. * /SS: SPI 从机选择 * CEX2: PCA单元2外部I/O..
P1.5 (附加功能) AIN5 (附加功能) MOSI (附加功能) CEX3	6	7	1	2		I/O I I/O I/O	* Port 1 位-5. * AIN5: ADC模拟量输入通道5 * MOSI: SPI 主机输出或从机输入. * CEX3: PCA单元3外部I/O.
P1.6 (附加功能) AIN6 (附加功能) MISO (附加功能) CEX4	7	8	2	3	-	I/O I I/O I/O	* 端口 1 位-6. * AIN6: ADC模拟量输入通道6. * MISO: SPI主机输入或从机输出. * CEX4: PCA单元4外部I/O..
P1.7 (附加功能) AIN7 (附加功能) SPICLK (附加功能) CEX5	8	9	3	4	-	I/O I I/O I/O	* 端口1 位-7. * AIN7: ADC模拟量输入通道7 * SPICLK: SPI 时钟，主机输出从机输入 * CEX5: PCA单元5外部I/O..

(续上表)

MNEMONIC	PIN NUMBER					I/O TYPE	DESCRIPTION
	40-Pin DIP	44-Pin PLCC	44-Pin PQFP	48-Pin LQFP			
P2.0 (附加功能) A8 (附加功能) KBI0	21	24	18	19		I/O O I	* 端口 2 位-0. * A8: 读写外部数据存储器输出A8 * KBI0: 辅助键盘输入0.
P2.1 (附加功能) A9 (附加功能) KBI1	22	25	19	20		I/O O I	* 端口 2 位-1. * A9: 读写外部数据存储器输出A9. * KBI1: 辅助键盘输入1.
P2.2 (附加功能) A10 (附加功能) KBI2	23	26	20	21		I/O O I	* 端口 2 位-2. * A10: 读写外部数据存储器输出A10. * KBI2: 辅助键盘输入2.
P2.3 (附加功能) A11 (附加功能) KBI3	24	27	21	22		I/O O I	* 端口 2 位-3. * A11: 读写外部数据存储器输出A11 * KBI3: 辅助键盘输入3.
P2.4 (附加功能) A12 (附加功能) KBI4	25	28	22	23		I/O O I	* 端口 2 位-4. * A12: 读写外部数据存储器输出A12 * KBI4: 辅助键盘输入4.
P2.5 (附加功能) A13 (附加功能) KBI5	26	29	23	26		I/O O I	* 端口 2 位-5. * A13: 读写外部数据存储器输出A13 * KBI5: 辅助键盘输入5.
P2.6 (附加功能) A14 (附加功能) KBI6	27	30	24	27		I/O O I	* 端口 2 位-6. * A14: 读写外部数据存储器输出A14 * KBI6: 辅助键盘输入6.
P2.7 (附加功能) A15 (附加功能) KBI7	28	31	25	28		I/O O I	* 端口 2 位-7. * A15: 读写外部数据存储器输出A15 * KBI7: 辅助键盘输入7.
P3.0 (附加功能) RXD	10	11	5	6		I/O I/O	* 端口 3 位-0. * RXD: 串行输入, 模式0的数据I/O.
P3.1 (附加功能) TXD	11	13	7	8		I/O O	* 端口 3 位-1. * TXD: 串行输出.
P3.2 (附加功能) /INT0	12	14	8	9		I/O I	* 端口 3 位-2. * /INT0: 外部中断0输入.
P3.3 (附加功能) /INT1	13	15	9	10		I/O I	* 端口 3 位-3. * /INT1: 外部中断1输入.
P3.4 (附加功能) T0 (附加功能) T0CKO	14	16	10	11		I/O I O	* 端口 3 位-4. * T0: 定时/计数器0外部输入. * T0CKO: 定时器0的可编程时钟输出
P3.5 (附加功能) T1 (附加功能) ALE (附加功能) S2CKO	15	17	11	12		I/O I O O	* 端口 3 位-5. * T1: 定时/计数器1外部输入. * ALE:地址锁存信号， 在外部数据存储器访问期间锁存地址低8位. * S2CKO: 定时器S2BRT.可编程时钟输出
P3.6 (附加功能) /WR	16	18	12	13		I/O O	* 端口 3 位-6. * /WR: 外部数据存储器写信号.
P3.7 (附加功能) /RD	17	19	13	14		I/O O	* 端口 3 位-7. * /RD: 外部数据存储器读信号.

(续上表)

MNEMONIC	PIN NUMBER					I/O TYPE	DESCRIPTION
	40-Pin DIP	44-Pin PLCC	44-Pin PQFP	48-Pin LQFP			
P4.0	-	23	17	18		I/O	* 端口 4 位-0.
P4.1 (附加功能) ALE	-	34	28	31		I/O O	* 端口 4 位-1. * ALE: 地址锁存信号， 在外部数据存储器访问期间锁存地址低8位
P4.2 (附加功能) /INT3	-	1	39	42		I/O I	* 端口 4 位-2. * /INT3: 外部中断3输入
P4.3 (附加功能) /INT2	-	12	6	7		I/O I	* 端口 4 位-3. * /INT2: 外部中断2输入
P4.4	-	-	-	24		I/O	* 端口 4 位-4.
P4.5	-	-	-	25		I/O	* 端口 4 位-5.
P4.6	-	-	-	48		I/O	* 端口 4 位-6.
P4.7	-	-	-	1		I/O	* 端口 4 位-7.
OCD_SDA	30	33	27	30		I/O	片上调试 OCD_SDA接口
OCD_SCL	29	32	26	29		I	片上调试 OCD_SCL接口
XTAL1	19	21	15	16		I	晶体1: 反向振荡放大器输入和内部时钟输入
XTAL2	18	20	14	15		O	晶体 2: 反向振荡放大器输出
RST	9	10	4	5		I	24个时钟周期的高电平复位微控制器，内置复位下拉电阻
V30	31	35	29	32		O	内部 LDO输出：外接一个外部电容（1μF)接地.
VDD	40	44	38	41		I	空闲、掉电和正常模式的电源正极
VSS	20	22	16	17		I	地，0V参考电压

Note: "(Alt. Fun.)" means the Alternate Function of this pin.

4.3 引脚功能重映射

许多I/O引脚，除了正常的I/O功能之外，也有其他复用功能。默认情况下，P2和P1被辅助键盘中断, PCA, SPI 和 UART2复用。但是，使用者可以通过设定AUXR1寄存器的P4KB，P4PCA，P4SPI 和 P4S2控制位使上面的那些功能重新映射到P4上。.当软件所需要的引脚数多于40个的时候，此功能尤其有用。注意，任何时候这四个控制位只能有一个被置位。

AUXR1 (地址=A2H，辅助寄存器1，复位值=0000,0000B)

7	6	5	4	3	2	1	0
P4KB	P4PCA	P4SPI	P4S2	GF2	-	-	DPS

P4KB: 被置位时，键盘接口被映射到P4
'KBI7' P2.7的功能被映射到 P4.7
'KBI6' P2.6的功能被映射到 P4.6
'KBI5' P2.5的功能被映射到 P4.5
'KBI4' P2.4的功能被映射到 P4.4
'KBI3' P2.3的功能被映射到 P4.3
'KBI2' P2.2的功能被映射到 P4.2
'KBI1' P2.1的功能被映射到 P4.1
'KBI0' P2.0的功能被映射到 P4.0

P4PCA: 被置位时,PCA 接口被映射到 P4
'ECI' P1.1 的功能被映射到P4.1
'CEX0' P1.2 的功能被映射到P4.2

‘CEX1’ P1.3 的功能被映射到P4.3
‘CEX2’ P1.4 的功能被映射到P4.4
‘CEX3’ P1.5 的功能被映射到P4.5
‘CEX4’ P1.6 的功能被映射到P4.6
‘CEX5’ P1.7 的功能被映射到P4.7

P4SPI: 被置位时, SPI 接口被映射到P4
‘/SS’ P1.4的功能被映射到 P4.4
‘MOSI’ P1.5的功能被映射到 P4.5
‘MISO’ P1.6的功能被映射到 P4.6
‘SPICLK’ P1.7的功能被映射到 P4.7

P4S2: 被置位时, UART2接口被映射到P4
‘S2RXD’ P1.2的功能被映射到P4.2
‘S2TXD’ P1.3的功能被映射到P4.3

5 存储器组织

像所有的 80C51一样， MPC82G516的程序存储器和数据存储器的地址空间是分开的，这样8位微处理器可以通过一个8位的地址快速而有效的访问数据存储器。

程序存储器(ROM)只能读取，不能写入。最大可以达到64K字节。在MPC82G516中，所有的程序存储器都是片上Flash存储器。因为没有设计外部程序使能 (/EA)和编程使能 (/PSEN) 信号，所以不允许外接程序存储器

数据存储器使用与程序存储器不同的地址空间。MPC82G516有256字节的内部RAM，使用外部数据存储器最多可以有64K字节的地址空间。CPU 通过使用一个16位的地址（通过DPTR）和读、写操作信号 (/RD和/WR) 访问外部数据存储器。由于一些应用程序需要多一点的内部 RAM, 所以 MPC82G516在片上集成了1024字节的外部存储器(XRAM).

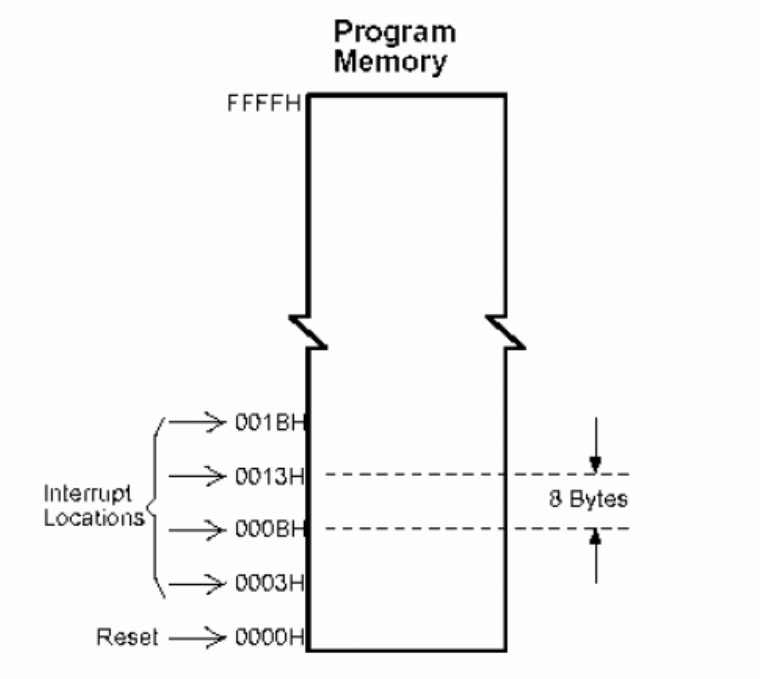
5.1 程序存储器

程序存储器用来保存让CPU进行处理的程序代码，如Figure 5-1所示。复位后，CPU从地址为0000H的地方开始运行，用户应用代码的起始部分应该放在这里。为了响应中断，中断服务位置(被称为中断矢量)应该位于程序存储器。每个中断在程序存储器中有一个固定的起始地址，中断使CPU跳到这个地址运行中断服务程序。举例来说， 外部中断0被指定到地址0003H，如果使用外部中断0，那么它的中断服务程序一定是从0003H开始的。如果中断未被使用，那么这些地址就可以被一般的程序使用。

中断服务程序的起始地址之间有8字节的地址间隔：外部中断0，*0003H*；定时器0，*000BH*；外部中断1, *0013H*；定时器1，*001BH*等等。如果中断服务程序足够短，它完全可以放在这8字节的空间中。如果其他的中断也被使用的话，较长的中断服务程序可以通过一条跳转指令越过后面的中断服务起始地址。

注意，MPC82G516不能外接程序存储器，所有应用代码都保存在片上的Flash存储器中。因此 /EA 和 /PSEN 信号因为不再需要而被省略了。用户应注意这一点。

Figure 5-1. 程序存储器



5.2 数据存储器

Figure 5-2 向MPC82G516使用者展示了内部和外部数据存储器的空间划分。内部数据存储器被划分为三部分，通常被称为低128字节 RAM ，高128字节 RAM 和128字节 SFR 空间。内部数据存储器的地址线只有8位宽，因此地址空间只有256字节。 SFR 空间的地址高于7FH，用直接地址访问；而用间接访问的方法访问高128字节的RAM。这样虽然SFR和高128字节RAM占用相同的地址空间，但他们实际上是分开的

如Figure 5-3所示，低128字节RAM与所有80C51 一样。最低的32字节被划分为4组每组8字节的寄存器组。指令中称这些寄存器为R0到R7。.程序状态字 (PSW) 中的两位用于选择哪组寄存器被使用。这使得程序空间能够被更有效的使用，因为对寄存器访问的指令比使用直接地址的指令短。 接下来的16字节是可以位寻址的存储器空间。80C51的指令集包含一个位操作指令集，这区域中的128位可以被这些指令直接使用。位地址从00H开始到7FH结束。

所有的低128字节RAM都可以用直接或间接地址访问，而高128字节RAM只能用间接地址访问。

Figure 5-4 给出了特殊功能寄存器 (SFR) 的概览。SFR包括端口寄存器，定时器和外围器件控制器，这些寄存器只能用直接地址访问。SFR 空间中有16个地址同时支持位寻址和直接寻址。可以位寻址的 SFR 的地址末位是0H或8H。

为了访问外部数据存储器，EXTRAM位应该被设为1。访问外部数据存储器可以使用一个16位地址 (使用‘MOVX @DPTR’)或一个8位地址 (使用 ‘MOVX @Ri’)。下面详细说明。

用8位地址访问

8位地址通常使用1根或更多的I/O口标明RAM的页数。如果使用8位地址，在访问外部存储器的周期中，P2寄存器保存P2引脚的状态。这将保证页的访问。 Figure 5-5 展示了一个2K字节外部数据存储器的硬件配置。P0口作为地址和数据总线复用，而P2口的三根线用于标明RAM的页数。.处理器产生/RD和/WR (P3.7和P3.6附加功能)信号控制存储器。当然也可以使用其他的I/O口而非P2口来标明RAM的页数。

用16位地址访问

16位地址通常用于访问64K字节的外部数据存储器。Figure 5-6展示了一个64K字节外部数据存储器的硬件配置。 当使用16位地址的时候，除了P0, /RD and /WR,的动作以外，地址的高字节通过P2口输出，并且在读写周期中是被锁定的。

无论如何，地址的低字节和数据字节在P0口是时分复用的。ALE (地址锁存使能) 被用来使地址字节被外部锁存器锁存，地址字节在 ALE负跳变时有效。在写周期中，数据在/WR有效之前在P0口出现，直到/WR无效的时候消失。在读周期中，数据在/RD信号无效之前被P0口接受。在任何外部存储器访问期间，CPU向P0口锁存器(特殊功能寄存器)写0FFH，以消除任何可能被锁存的数据。

访问片上扩展存储器 (XRAM)， EXTRAM 位应该被设为0。Figure 5-2，这1024字节的XRAM (0000H to 03FFH)外部访问指令MOVX间接存取。 对XRAM的访问没有任何的地址输出、地址锁存信号和读写控制。这意味着P0, P2, ALE, P3.6 (/WR) 和 P3.7 (/RD) 在访问XRAM期间保持不变。

Figure 5-2. 数据存储器

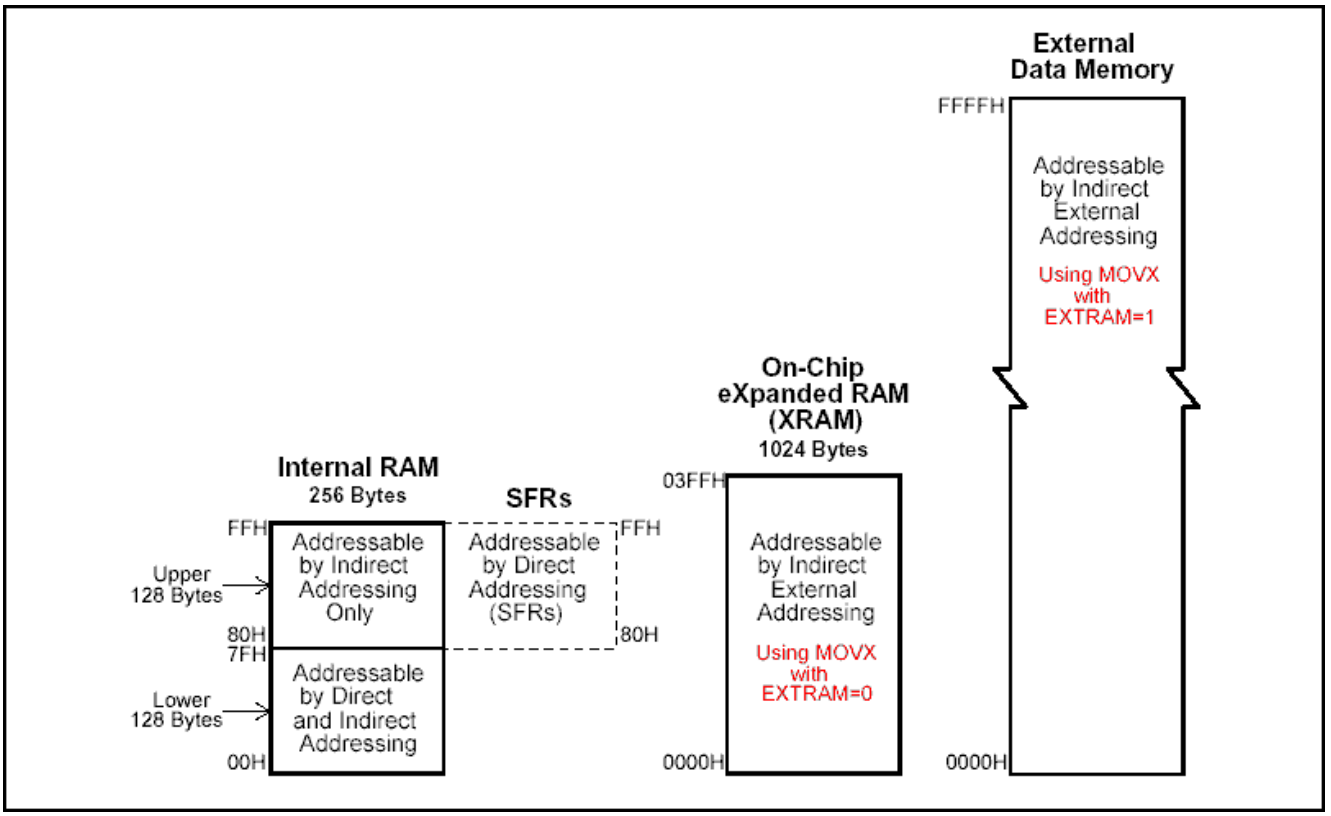


Figure 5-3. 内部RAM的低128字节

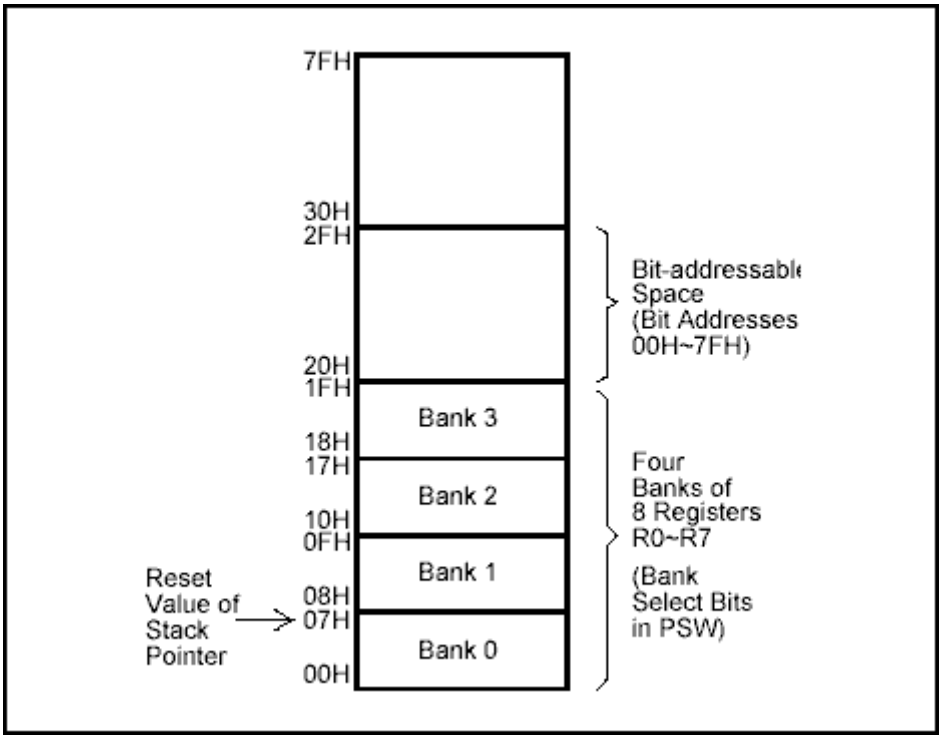
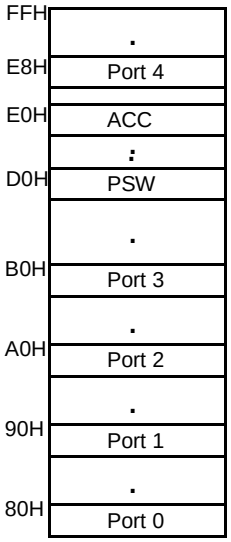
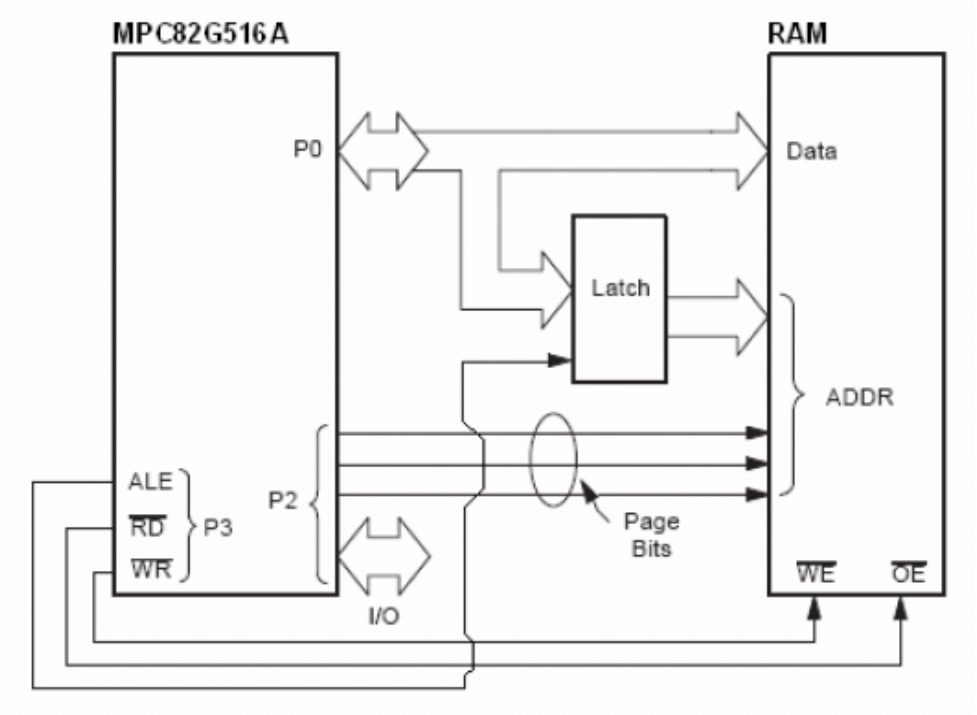


Figure 5-4. SFR 空间



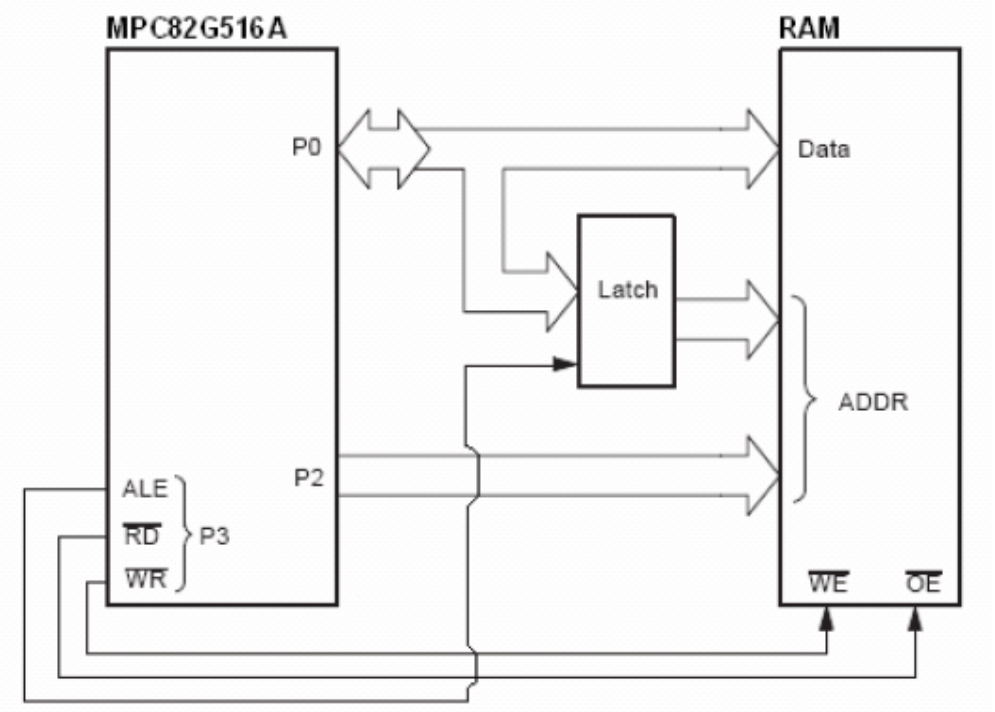
1. I/O 是寄存器的映射
2. 地址结尾是0H或8H的寄存器可被位寻址.
   - 端口
   - 累加器
   - PSW
   (*Etc*.)

Figure 5-5. 通过8位地址访问外部RAM (使用 ‘MOVX @ Ri’ 和页选择)



在这种情况下，P2口的其他位可做一般I/O口使用.

Figure 5-6. 通过16位地址访问外部RAM (使用 ‘MOVX @ DPTR’)



5.3 关于 C51 编译器的声明识别符

C51编译器的声明识别符与 MPC82G516 存储空间的对应关系。:

data
128字节的内部数据存储空间 (00h~7Fh)；使用除MOVX和MOVC以外的指令，可以直接或间接的访问。 全部或部分的堆栈可能保存在此区域中。

idata
间接数据；256字节的内部数据存储空间 (00h~FFh) 使用除MOVX和 MOVC以外的指令间接访问。全部或部分的堆栈可能保存在此区域中。此区域包括 ***data区*** 和data区以上的128字节。

sfr
特殊功能寄存器； CPU寄存器和外围部件控制/状态寄存器，只能通过直接地址访问。.

xdata
外部数据或片上的扩展RAM (XRAM)；通过 “MOVX @DPTR” 指令访问标准80C51的64K存储空间。MPC82G516 有 1024 字节的片上 ***xdata*** 存储空间.

pdata
分页的外部数据(256 字节) 或片上的扩展 RAM；通过 “MOVX @Ri” 指令访问标准 80C51 的 256 字节存储空间。MPC82G516 有片上 1024 字节中的最低位 256 字节当作 ***pdata*** 存储空间.

code
64K程序存储空间；通过“MOVC @A+DTPR”访问，作为程序的一部分被读取。MPC82G516 有 64K 字节的片上 ***code*** 存储器.

6 特殊功能寄存器 (SFRs)

6.1 SFR 映射位置

特殊功能寄存器空间的内部记忆区域的一个映像叫做“SFR 存储器映射表”。如Table 6-1 所示，在SFR 存储器映射表中，不是所有的地址都被使用。空闲的地址没有被实现或设计用来进行测试。读取这些地址将返回随机的数据，而向其中写入数据，将导致不可预知的硬件动作。使用者的软件最好不要访问空闲的地址。

.

Table 6-1. SFR 存储器映射表

	8 BYTES								
F8H	-	CH	CCAP0H	CCAP1H	CCAP2H	CCAP3H	CCAP4H	CCAP5H	FFH
F0H	B	-	PCAPWM0	PCAPWM1	PCAPWM2	PCAPWM3	PCAPWM4	PCAPWM5	F7H
E8H	P4	CL	CCAP0L	CCAP1L	CCAP2L	CCAP3L	CCAP4L	CCAP5L	EFH
E0H	ACC	WDTCR	IFD	IFADRH	IFADRL	IFMT	SCMD	ISPCR	E7H
D8H	CCON	CMOD	CCAPM0	CCAPM1	CCAPM2	CCAPM3	CCAPM4	CCAPM5	DFH
D0H	PSW	-	-	-	-	KBPATN	KBCON	KBMASK	D7H
C8H	T2CON	T2MOD	RCAP2L	RCAP2H	TL2	TH2	-	-	CFH
C0H	XICON	-	-	-	-	ADCTL	ADCH	PCON2	C7H
B8H	IP	SADEN	S2BRT	-	-	-	ADCL	-	BFH
B0H	P3	P3M0	P3M1	P4M0	P4M1			IPH	B7H
A8H	IE	SADDR	S2CON	-	-	AUXIE	AUXIP	AUXIPH	AFH
A0H	P2		AUXR1	-	-	-	AUXR2	-	A7H
98H	SCON	SBUF	S2BUF	-	-	-	-	-	9FH
90H	P1	P1M0	P1M1	P0M0	P0M1	P2M0	P2M1	EVRCR	97H
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	STRETCH	8FH
80H	P0	SP	DPL	DPH	SPSTAT	SPCTL	SPDAT	PCON	87H

↑

可位寻址的 SFRs

6.2 SFR 描述

6.2.1 标准 80C51的 SFRs

标准80C51的SFR如Table 6-2所示。其中，C51核心寄存器的功能在下面被概略说明。更多的关于标准SFR的使用信息将在外围器件中介绍。.

C51 核心寄存器

累加器： ACC是累加寄存器，这是给累加器的特定助记符，但是只提及累加器内容时被标记位A.

B 寄存器: B 被用在乘或除运算中，对于其他的指令可以当做一般寄存器使用。

堆栈指针：堆栈指针寄存器宽度是8位，它指向堆栈的顶端最后被使用的数据。虽然低位字节通常是用来作为工作寄存器，但是使用者通过设置堆栈指针，可以把堆栈放到内部RAM的任何位置。复位后，堆栈指针的初值为 07H， 这样堆栈从08H开始。

数据指针：数据指针(DPTR) 有一个高字节(DPH) 和一个低字节(DPL). 它的功能是为MOVX 指令保存一个16位的存储器地址。这个地址可以指向片上或片外的程序/数据存储器，或者外围设备的存储器映射地址。它可以被当做16位寄存器或者两个独立的8位寄存器。

程序状态字： PSW 寄存器包含如下列各项所详述的程序状态信息.

PSW (地址=D0H，程序状态字，复位值=0000,0000B)

7	6	5	4	3	2	1	0
CY	AC	F0	RS1	RS2	OV	-	P

CY: 进位标志.
当最后一个算数运算有进位（加）或借位（减）的时候，该位被置位。
其他的算术运算将它清除为逻辑0。

AC: 辅助进位标志. (对于BCD 运算)
当最后一个算数运算向高四位有进位（加）或借位（减）的时候，该位被置位。
其他的算术运算将它清除为逻辑0.

F0: 标志 0.
可位寻址，通常作为用户使用的软件控制标志位

RS1: 寄存器组选择位1

RS0: 寄存器组选择位0.

(RS1, RS0)	工作寄存器组和地址
(0, 0)	Bank 0 (00H~07H)
(0, 1)	Bank 1 (08H~0FH)
(1, 0)	Bank 2 (10H~17H)
(1, 1)	Bank 3 (18H~1FH)

OV: 溢出标志.
这位在下列的环境之下被设定成1:
• ADD, ADDC, SUBB 指令引起的数据的溢出.
• MUL 指令的结果引起的溢出 (结果超过 255).
• DIV 指令除数为零.
ADD, ADDC, SUBB, MUL, DIV 指令的其它结果将该位清0.

P: 奇偶标志.
每个指令周期由硬件置1或清0，用来指示累加器中“1”为奇数个或偶数个。

(注意：PSW 寄存器可位寻址，所有的被释放的位能被软件设定或清除。.)

Table 6-2. 标准 80C51的 SFRs

符号	描 述	地址	位地址 & 符号								复位值
			Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	
ACC*	累加器	E0H	E7H ACC.7	E6H ACC.6	E5H ACC.5	E4H ACC.4	E3H ACC.3	E2H ACC.2	E1H ACC.1	E0H ACC.0	00H
B*	B 寄存器	F0H	F7H B.7	F6H B.6	F5H B.5	F4H B.4	F3H B.3	F2H B.2	F1H B.1	F0H B.0	00H
PSW*	程序状态字	D0H	D7H CY	D6H AC	D5H F0	D4H RS1	D3H RS0	D2H OV	D1H -	D0H P	00H
SP	堆栈指针	81H									07H
DPH	数据指针高	83H									00H
DPL	数据指针低	82H									00H
P0*	端口 0	80H	87H P0.7	86H P0.6	85H P0.5	84H P0.4	83H P0.3	82H P0.2	81H P0.1	80H P0.0	FFH
P1*	端口 1	90H	97H P1.7	96H P1.6	95H P1.5	94H P1.4	93H P1.3	92H P1.2	91H P1.1	90H P1.0	FFH
P2*	端口 2	A0H	A7H P2.7	A6H P2.6	A5H P2.5	A4H P2.4	A3H P2.3	A2H P2.2	A1H P2.1	A0H P2.0	FFH
P3*	端口 3	B0H	B7H P3.7	B6H P3.6	B5H P3.5	B4H P3.4	B3H P3.3	B2H P3.2	B1H P3.1	B0H P3.0	FFH
IP*	中断优先级	B8H	BFH -	BEH -	BDH PT2	BCH PS	BBH PT1	BAH PX1	B9H PT0	B8H PX0	00H
IE*	中断允许	A8H	AFH EA	AEH -	ADH ET2	ACH ES	ABH ET1	AAH EX1	A9H ET0	A8H EX0	00H
TMOD	定时器模式	89H	GATE	C/-T	M1	M0	GATE	C/-T	M1	M0	00H
TCON*	定时器控制	88H	8FH TF1	8EH TR1	8DH TF0	8CH TR0	8BH IE1	8AH IT1	89H IE0	88H IT0	00H
T2CON*	定时器2 控制	C8H	CFH TF2	CEH EXF2	CDH RCLK	CCH TCLK	CBH EXEN2	CAH TR2	C9H C/-T2	C8H CP/-RL2	00H
TH0	定时器0 高字节	8CH									00H
TL0	定时器0 低字节	8AH									00H
TH1	定时器1 高字节	8DH									00H
TL1	定时器1 低字节	8BH									00H
TH2	定时器2 高字节	CDH									00H
TL2	定时器2 低字节	CCH									00H
RCAP2H	定时器2 捕获, 高	CBH									00H
RCAP2L	定时器2 捕获, 低	CAH									00H
SCON*	串口控制	98H	9FH SM0/FE	9EH SM1	9DH SM2	9CH REN	9BH TB8	9AH RB8	99H TI	98H RI	00H
SBUF	串口数据缓冲区	99H									xxH
PCON	电源控制	87H	SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL	10H#

注:

*: 可位寻址
-: 保留位
#: 复位值依赖于复位源

6.2.2 新加入的 SFRs

新加入的 SFRs 如 Table 6-3所示。更多的关于新加入的SFR的使用信息将在外围器件中介绍。

Table 6-3. 新加入的 SFRs

符号	描 述	地址	位地址 & 符号								复位值
			Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	
Interrupt											
XICON*	外部中断控制	C0H	C7H PX3	C6H EX3	C5H IE3	C4H IT3	C3H PX2	C2H EX2	C1H IE2	C0H IT2	00H
IPH	中断优先级高	B7H	PX3H	PX2H	PT2H	PSH	PT1H	PX1H	PT0H	PX0H	00H
AUXIE	辅助中断允许	ADH	-	-	EKB	ES2	EBD	EPCA	EADC	ESPI	00H
AUXIP	辅助中断优先级	AEH	-	-	PKB	PS2	PBD	PPCA	PADC	PSPI	00H
AUXIPH	辅助中断优先级高	AFH	-	-	PKBH	PS2H	PBDH	PPCAH	PADCH	PSPIH	00H
I/O Port											
P4*	端口4	E8H	EFH P4.7	EEH P4.6	EDH P4.5	ECH P4.4	EBH P4.3	EAH P4.2	E9H P4.1	E8H P4.0	FFH
P0M0	端口0 模式寄存器0	93H	P0M0.7	P0M0.6	P0M0.5	P0M0.4	P0M0.3	P0M0.2	P0M0.1	P0M0.0	00H
P0M1	端口0 模式寄存器1	94H	P0M1.7	P0M1.6	P0M1.5	P0M1.4	P0M1.3	P0M1.2	P0M1.1	P0M1.0	00H
P1M0	端口1 模式寄存器0	91H	P1M0.7	P1M0.6	P1M0.5	P1M0.4	P1M0.3	P1M0.2	P1M0.1	P1M0.0	00H
P1M1	端口1 模式寄存器1	92H	P1M1.7	P1M1.6	P1M1.5	P1M1.4	P1M1.3	P1M1.2	P1M1.1	P1M1.0	00H
P2M0	端口2 模式寄存器0	95H	P2M0.7	P2M0.6	P2M0.5	P2M0.4	P2M0.3	P2M0.2	P2M0.1	P2M0.0	00H
P2M1	端口2 模式寄存器1	96H	P2M1.7	P2M1.6	P2M1.5	P2M1.4	P2M1.3	P2M1.2	P2M1.1	P2M1.0	00H
P3M0	端口3 模式寄存器0	B1H	P3M0.7	P3M0.6	P3M0.5	P3M0.4	P3M0.3	P3M0.2	P3M0.1	P3M0.0	00H
P3M1	端口3 模式寄存器1	B2H	P3M1.7	P3M1.6	P3M1.5	P3M1.4	P3M1.3	P3M1.2	P3M1.1	P3M1.0	00H
P4M0	端口4 模式寄存器0	B3H	P4M0.7	P4M0.6	P4M0.5	P4M0.4	P4M0.3	P4M0.2	P4M0.1	P4M0.0	00H
P4M1	端口4 模式寄存器1	B4H	P4M1.7	P4M1.6	P4M1.5	P4M1.4	P4M1.3	P4M1.2	P4M1.1	P4M1.0	00H
Keypad Interrupt											
KBCON	键盘控制	D6H	-	-	-	-	-	-	PATNS	KBIF	00H
KBPATN	键盘模式	D5H	KBPATN.7	KBPATN.6	KBPATN.5	KBPATN.4	KBPATN.3	KBPATN.2	KBPATN.1	KBPATN.0	FFH
KBMASK	键盘中断过滤	D7H	KBMASK.7	KBMASK.6	KBMASK.5	KBMASK.4	KBMASK.3	KBMASK.2	KBMASK.1	KBMASK.0	00H
Serial Port											
SADEN	从机地址过滤	B9H	SADEN.7	SADEN.6	SADEN.5	SADEN.4	SADEN.3	SADEN.2	SADEN.1	SADEN.0	00H
SADDR	从机地址	A9H	SADDR.7	SADDR.6	SADDR.5	SADDR.4	SADDR.3	SADDR.2	SADDR.1	SADDR.0	00H
S2CON	UART2 控制	AAH	S2SM0	S2SM1	S2SM2	S2REN	S2TB8	S2RB8	S2TI	S2RI	00H
S2BRT	UART2波特率定时器	BAH									00H
S2BUF	UART2 串口缓冲器	9AH									xxH
ADC											
ADCTL	ADC 控制寄存器	C5H	ADCON	SPEED1	SPEED0	ADCI	ADCS	CHS2	CHS1	CHS0	00H
ADCH	ADC 结果，高字节	C6H									xxH
ADCL	ADC 结果，低字节	BEH									xxH
SPI											
SPCTL	SPI 控制寄存器	85H	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	04H
SPSTAT	SPI 状态寄存器	84H	SPIF	WCOL	-	-	-	-	-	-	00H
SPDAT	SPI 数据寄存器	86H									00H

PCA											
CCON*	PCA 计数器控制	D8H	DFH CF	DEH CR	DDH CCF5	DCH CCF4	DBH CCF3	DAH CCF2	D9H CCF1	D8H CCF0	00H
CMOD	PCA 计数器模式	D9H	CIDL	-	-	-	-	CPS2	CPS1	ECF	00H
CH	PCA 计数器.高	F9H									00H
CL	PCA 计数器.低	E9H									00H
CCAPM0	PCA 通道0 比较/捕获寄存器	DAH	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	00H
CCAPM1	PCA 通道1 比较/捕获寄存器	DBH	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	00H
CCAPM2	PCA 通道2 比较/捕获寄存器	DCH	-	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2	00H
CCAPM3	PCA 通道3 比较/捕获寄存器	DDH	-	ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3	00H
CCAPM4	PCA 通道4 比较/捕获寄存器	DEH	-	ECOM4	CAPP4	CAPN4	MAT4	TOG4	PWM4	ECCF4	00H
CCAPM5	PCA 通道5 比较/捕获寄存器	DFH	-	ECOM5	CAPP5	CAPN5	MAT5	TOG5	PWM5	ECCF5	00H
CCAP0H	PCA 通道0 捕获寄存器，高字节	FAH									00H
CCAP0L	PCA 通道0 捕获寄存器，低字节	EAH									00H
CCAP1H	PCA 通道1 捕获寄存器，高字节	FBH									00H
CCAP1L	PCA 通道1 捕获寄存器，低字节	EBH									00H
CCAP2H	PCA 通道2 捕获寄存器，高字节	FCH									00H
CCAP2L	PCA 通道2 捕获寄存器，低字节	ECH									00H
CCAP3H	PCA 通道3 捕获寄存器，高字节	FDH									00H
CCAP3L	PCA 通道3 捕获寄存器，低字节	EDH									00H
CCAP4H	PCA 通道4 捕获寄存器，高字节	FEH									00H
CCAP4L	PCA 通道4 捕获寄存器，低字节	EEH									00H
CCAP5H	PCA 通道5 捕获寄存器，高字节	FFH									00H
CCAP5L	PCA 通道5 捕获寄存器，低字节	EFH									00H
PCAPWM0	PCA PWM 模式辅助寄存器 0	F2H	-	-	-	-	-	-	ECAP0H	ECAP0L	00H
PCAPWM1	PCA PWM 模式辅助寄存器 1	F3H	-	-	-	-	-	-	ECAP1H	ECAP1L	00H
PCAPWM2	PCA PWM 模式辅助寄存器 2	F4H	-	-	-	-	-	-	ECAP2H	ECAP2L	00H
PCAPWM3	PCA PWM 模式辅助寄存器 3	F5H	-	-	-	-	-	-	ECAP3H	ECAP3L	00H
PCAPWM4	PCA PWM 模式辅助寄存器 4	F6H	-	-	-	-	-	-	ECAP4H	ECAP4L	00H
PCAPWM5	PCA PWM 模式辅助寄存器 5	F7H	-	-	-	-	-	-	ECAP5H	ECAP5L	00H
Others											
AUXR	辅助寄存器	8EH	URTS	ADRJ	P41ALE	P35ALE	-	-	EXTRAM	-	00H
AUXR1	辅助寄存器 1	A2H	P4KB	P4PCA	P4SPI	P4S2	GF2	-	-	DPS	00H
AUXR2	辅助寄存器 2	A6H	T0X12	T1X12	URM0X6	S2TR	S2SMOD	S2TX12	S2CKOE	T0CKOE	00H
T2MOD	定时器 2 模式控制	C9H	-	-	-	-	-	-	T2OE	DCEN	00H
STRETCH	外部访问扩展	8FH	-	-	ALES1	ALES0	-	RWS2	RWS1	RWS0	23H
PCON2	电源控制 2	C7H	-	-	-	-	-	SCKD2	SCKD1	SCKD0	00H
WDTCR	看门狗定时器	E1H	WRF	-	ENW	CLRW	WIDL	PS2	PS1	PS0	00H#
EVRCR	EVR 控制寄存器	97H	EOPFI	ECPFI	OPF	CPF	PMUOFF	-	-	-	30H#

ISP										
ISPCR	ISP 控制寄存器	E7H	ISPEN	SWBS	SWRST	-	-	CKS2	CKS1	CKS0	00H
IFMT	ISP 模式选择	E5H	-	-	-	-	-	-	MS1	MS0	63H
IFADRH	ISP Flash地址高字节	E3H									00H
IFADRL	ISP Flash地址低字节	E4H									00H
IFD	ISP Flash 数据	E2H									FFH
SCMD	ISP 指令	E6H									xxH

注:

*: 可位寻址

-: 保留位

#: 复位值依赖于复位源.

7 片上扩展 RAM (XRAM)

访问片上扩展RAM (XRAM), EXTRAM 位应该被清零。这1024字节的 XRAM (地址从 0000H 到 03FFH) 被外部访问指令 “MOVX @DPTR”.间接访问
对XRAM的访问没有任何的地址输出、地址锁存信号和读写控制。这意味着P0, P2, P3.5/P4.1(ALE), P3.6 (/WR) 和 P3.7 (/RD) 在访问XRAM期间保持不变。但是如果地址超过了0x03FF,访问会被自动转到外部数据存储器。.

AUXR (地址=8EH, 辅助寄存器, 复位值=0000,xx0xB)

7	6	5	4	3	2	1	0
URTS	ADRJ	-	-	-	-	EXTRAM	-

EXTRAM:

0: 当地址小于 0x0400的时候禁止访问外部数据存储器;
访问地址 0x0000~0x03FF时，自动装到片上 XRAM.

1: 可以访问地址 0x0000~0xFFFF的全部外部数据存储器; 访问片上 XRAM 被禁止。.

7.1 在软件中使用 XRAM

Keil-C51 编译器中，将变量分配到XRAM中，需要使用“xdata” 声明.
编译后，被声明位 “xdata”的变量将通过 “MOVX @DPTR”指令进行存取. 使用者可以通过 “*Keil Software — Cx51 Compiler User’s Guide*”.获得更多的信息。

Table 7-1. XRAM 存储类型声明

Explicitly Declared Memory Types

You may specify where variables are stored by including a memory type specifier in the variable declaration.

The following table summarizes the available memory type specifiers.

Memory Type	Description
code	Program memory (64 KBytes); accessed by opcode MOVC @A+DPTR.
data	Directly addressable internal data memory; fastest access to variables (128 bytes).
idata	Indirectly addressable internal data memory; accessed across the full internal address space (256 bytes).
bdata	Bit-addressable internal data memory; supports mixed bit and byte access (16 bytes).
xdata	External data memory (64 KBytes); accessed by opcode MOVX @DPTR.
far	Extended RAM and ROM memory spaces (up to 16MB); accessed by user defined routines or specific chip extensions (Philips 80C51MX, Dallas 390).
pdata	Paged (256 bytes) external data memory; accessed by opcode MOVX @R*n*.

As with the **signed** and **unsigned** attributes, you may include memory type specifiers in the variable declaration.

Example:

```
char data var1;
char code text[] = "ENTER PARAMETER:";
unsigned long xdata array[100];
float idata x,y,z;
unsigned int pdata dimension;
unsigned char xdata vector[10][4][4];
char bdata flags;
```

8 外部数据存储器的存取

和 5.2 章描述的一样，访问外部数据存储器需要将 EXTRAM 位置 1。访问外部数据存储器既可用 16 位地址(使用“MOVX @DPTR)，也可以使用 8 位地址(使用“MOVX @Ri”)，如下所述。

通过 8 位地址访问

8 位地址线经常使用 1 个或多个 I/O 线结合来访问 RAM 页面。如果使用 8 位地址，在外部存储器读写周期，专用寄存器 P2 口的内容始终保持在 P2 引脚。使得访问页面非常容易。图 5-5 示一个访问 2K 的外部 RAM 的硬件配置。P0 用作一个地址/数据分时复用总线到 RAM，P2 口的三线用来访问 RAM 页。CPU 产生/RD 和/WR(P3.7 和 P3.6 可选择功能)来选通存储器。当然，用户可以使用其它 I/O 线来代替 P2 到 RAM 页面。

通过 16 位地址访问

16 位地址线经常用来访问 64K 的外部数据存储器。图 5-6 示硬件配置来访问外部 64K 的 RAM。只要使用 16 位地址，除了 P0 口、/RD 和/WR 之外，P2 口的高位地址在读写周期一直保持。

在任何情况下，P0 口的低位地址线和数据线为分时复用。ALE(地址锁存使能)用来将地址字节捕获到外部锁存器。在 ALE 的负跳变时地址字节有效。接着，在写周期，在/WR 激活之前，要写的数据出现在 P0 口，并一直保持直到/WR 信号释放。在一个读周期，信号字节从 P0 口接收在读选通信号释放前。在访问外部存储器时，CPU 写 0FFH 到端口 0 锁存(专用寄存器)，专用寄存器 P0 保持的信息被擦除。

8.1 配置 ALE 引脚

对 MPC82G516 来说，ALE 信号有专门的引脚。80C51 的单片机在没有外部访问时仍然输出 ALE 信号，除了访问访问外部数据存储器(EXTRAM=1)时，器件不会输出 ALE 信号。

8.2 低速存储器的存取时间延展

为了访问低速的外部数据存储器，设计了时序延长机制来控制“MOVX”指令的访问时序。延长(STRETCH)寄存器的位 ALES1 和 ALES0，控制延长设置时间和保持时间并保持到 ALE 的下降沿。另外，位 RWS2、RWS1 和 RWS0 控制延长读写脉冲宽度。用户可以通过适当的配置 STRETCH 寄存器，以适应外部数据存储器读写的需求。

STRETCH(地址=8FH，延长寄存器，复位值=0000,0011B)

7	6	5	4	3	2	1	0
-	-	ALES1	ALES0	-	RWS2	RWS1	RWS0

{ALES1，ALES0}：
00：没有延长，P0 口地址的设置/保持时间随着 ALE 的下降沿在一个时钟周期；
01：1 个时钟周期的延长，P0 口地址设置/保持时间随着 ALE 的下降沿在二个时钟周期；
10：2 个时钟周期的延长，P0 口地址设置/保持时间随着 ALE 的下降沿在三个时钟周期；
11：3 个时钟周期的延长，P0 口地址设置/保持时间随着 ALE 的下降沿在四个时钟周期；
{RWS2，RWS1，RWS0}：
000：没有延长，MOVX 读写脉冲为一个时钟周期；
001：1 个时钟延长，MOVX 读写脉冲为 2 个时钟周期；
010：2 个时钟延长，MOVX 读写脉冲为 3 个时钟周期；
011：3 个时钟延长，MOVX 读写脉冲为 4 个时钟周期；
100：4 个时钟延长，MOVX 读写脉冲为 5 个时钟周期；
101：5 个时钟延长，MOVX 读写脉冲为 6 个时钟周期；
110：6 个时钟延长，MOVX 读写脉冲为 7 个时钟周期；
111：7 个时钟延长，MOVX 读写脉冲为 8 个时钟周期；
看如下时序波形的演示。

图 8-1. “MOVX @DPTR,A” 没有延迟

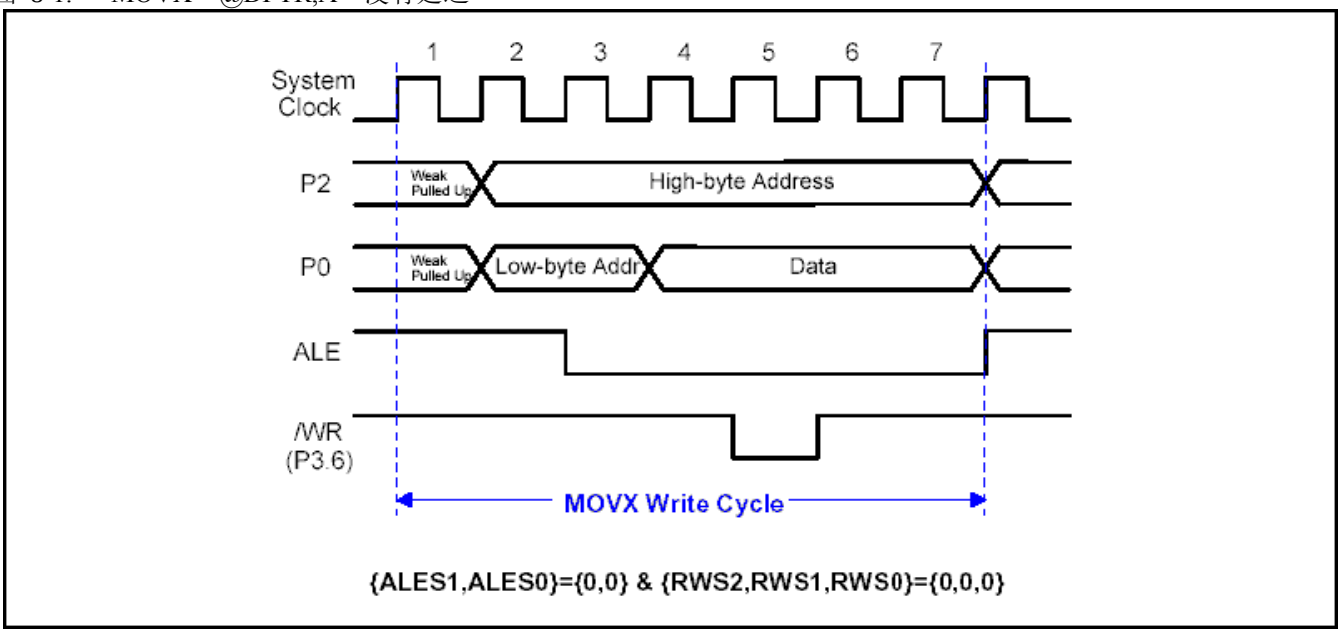


图 8-2 “MOVX @DPTR，A”有延迟

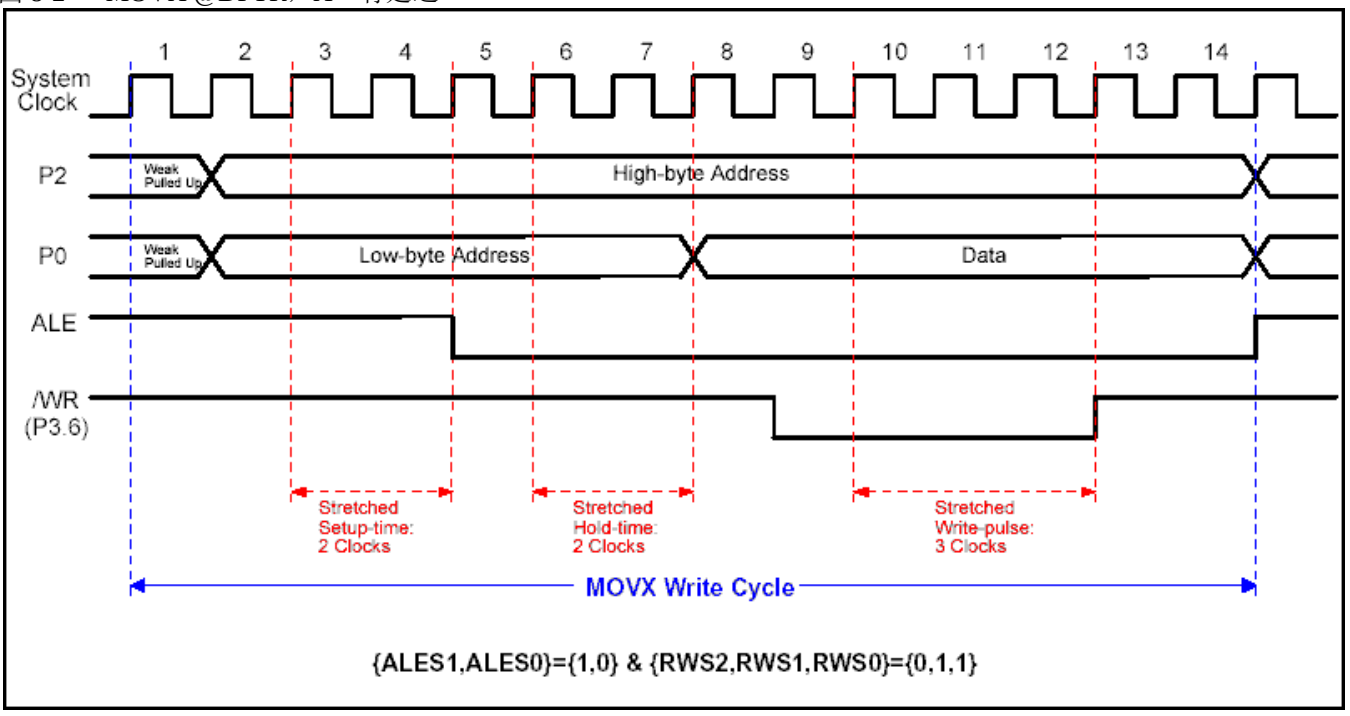


图 8-3 “MOVX A，@DPTR”没有延迟

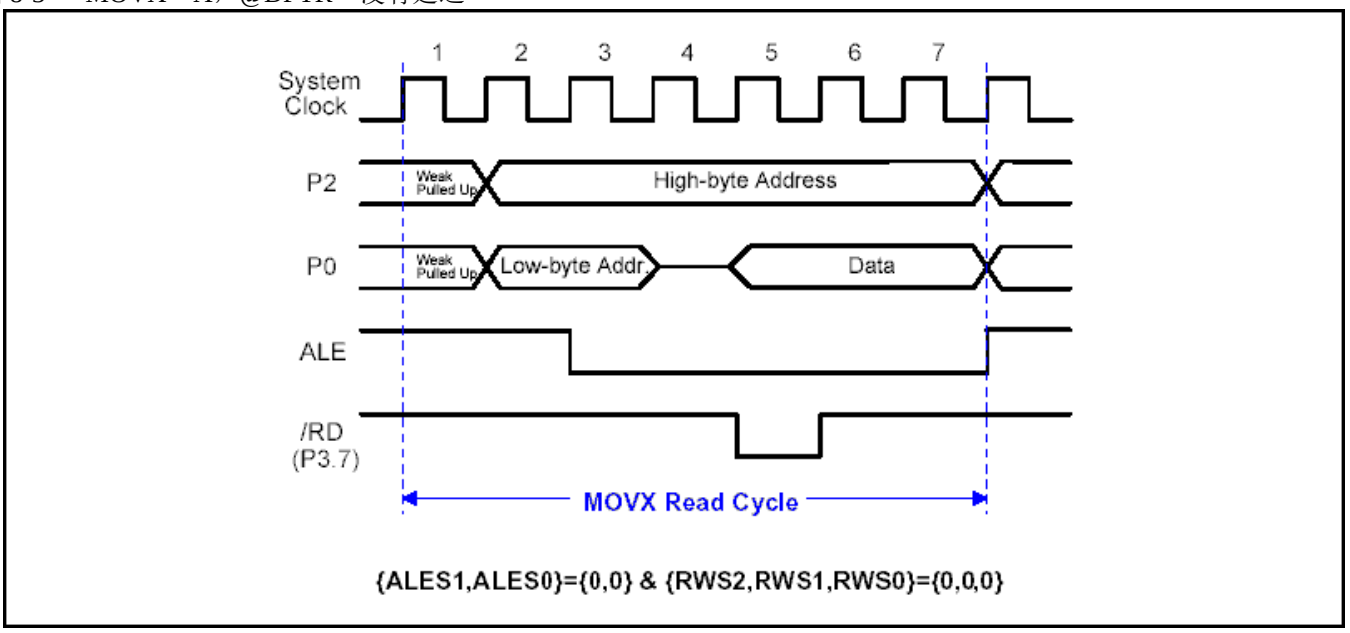
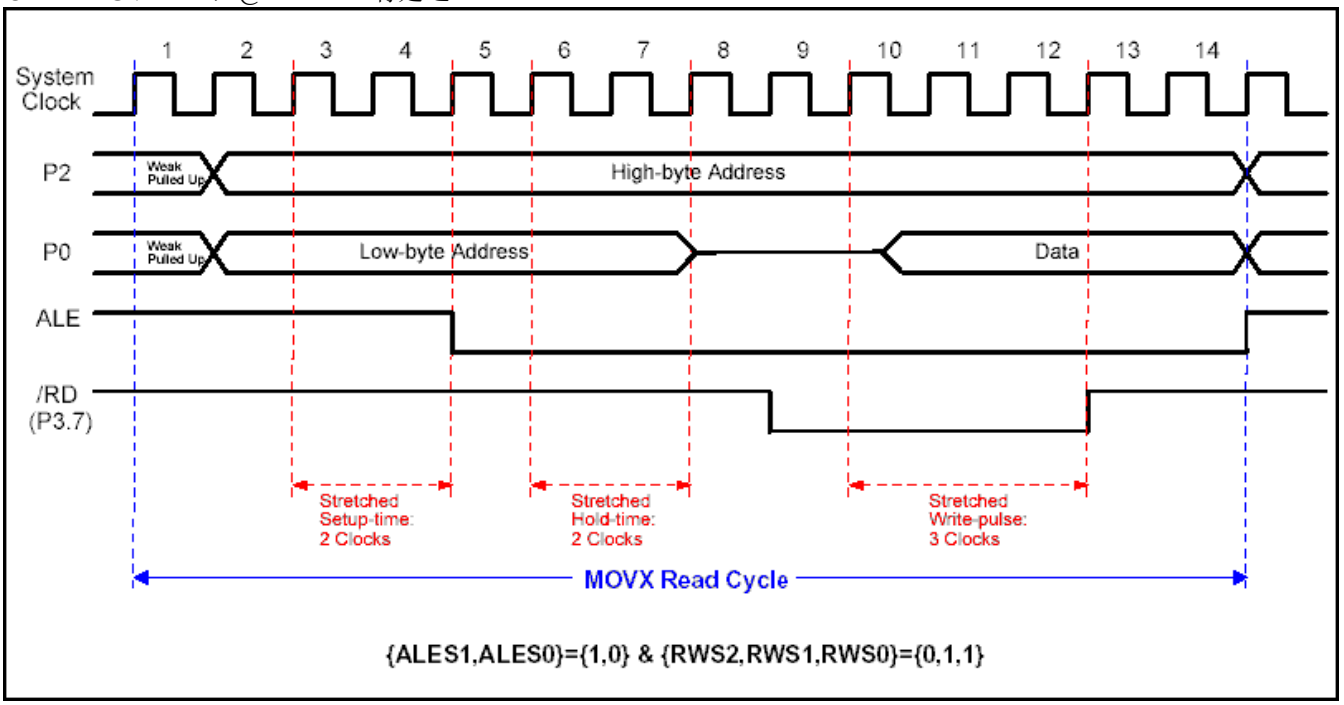


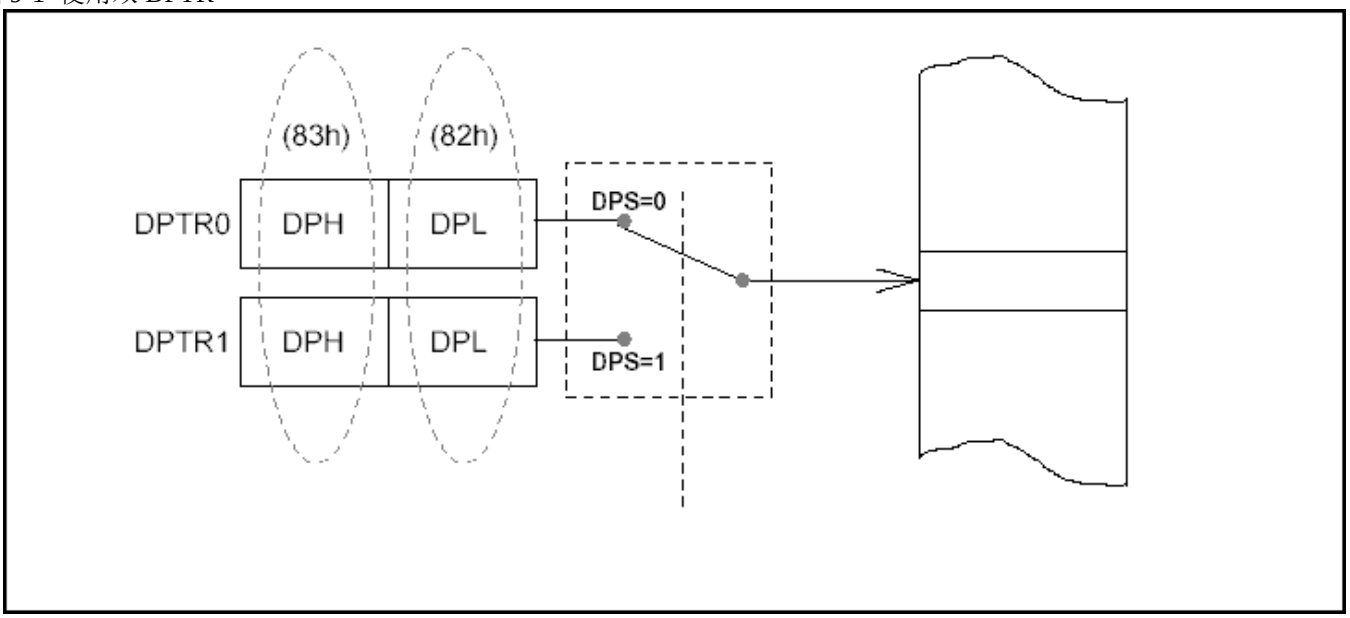
图 8-4 “MOVX A，@DPTR” 有延迟



9 双数据指针寄存器(DPTR)

传统的数据指针用来加速代码执行和减少代码尺寸。双 DPTR 结构是一种方法，芯片将指定外部数据存储器的定位地址。外部存储器有两个 16 位 DPTR 寄存器，和一个控制位称作为 DPS(AUXR1.0)，允许在程序代码和外部存储器之间的切换。

图 9-1 使用双 DPTR



DPTR 指令

使用 DPS 位的六条指令参考 DPTR 的当前选择，如下：

```
INC       DPTR              ；数据指针加 1
MOV       DPTR,#data16      ；DPTR 加载 16 位常量
MOVC      A，@A+DPTR        ；将代码字节移动到 ACC
MOVX      A，@DPTR          ；  移动外部 RAM(16 位地址)到 ACC
MOVX      @DPTR，A          ；移动 ACC 到外部 RAM(16 位地址)
JMP       @A+DPTR           ；直接跳转到 DPTR
```

AUXR1 (地址=A2H，辅助寄存器 1，复位值=0000,0000B)

7	6	5	4	3	2	1	0
P4KB	P4PCA	P4SPI	P4S2	GF2	-	-	DPS

DPS：DPTR 选择位，用来在 DPTR0 和 DPTR1 之间切换

在 DPTR0 和 DPTR1 之间切换时，可通过软件来保存 DPS 位状态

DPS	DPTR 选择
0	DPTR0
1	DPTR1

10 I/O 结构

MPC82G516 有五个 I/O 端口：端口 0、端口 1、端口 2、端口 3 和端口 4。所有的端口都为 8 位端口。准确的可用 I/O 引脚数量由封装类型决定。见表 10-1。

表 10-1 可用 I/O 引脚数量

封装类型	I/O引脚 脚	引脚数量
40-pin DIP	P0, P1, P2, P3	32
44-pin PLCC	P0, P1, P2, P3, P4.0~P4.3	36
44-pin PQFP	P0, P1, P2, P3, P4.0~P4.3	36
48-pin LQFP	P0, P1, P2, P3, P4	40

10.1 配置 I/O

MPC82G516 的所有端口可通过软件个别的、独立的配置为四种之中的一种类型，基于位位基础，如表 10-2 所示。这四种类型有：准双向(标准 8051 的 I/O 端口)、上拉输出、集电极开路输出和输入(高阻抗输入)。每个端口有两个模式寄存器来选择每个端口引脚的输出类型。

表 10-2 端口配置设定

PxM0.y	PxM1.y	端 口
0	0	准双向端口
0	1	推挽式的输出
1	0	输入(高阻抗)
1	1	集电极开路输出

这里 x=0~4(端口号)，y=0~7(端口引脚号)。寄存器 PxM0 和 PxM1 列表如下。

P0M0 (地址=93H，端口 0 模式寄存器 0，复位值=0000,0000B)

7	6	5	4	3	2	1	0
P0M0.7	P0M0.6	P0M0.5	P0M0.4	P0M0.3	P0M0.2	P0M0.1	P0M0.0

P0M1 (地址=94H，端口 0 模式寄存器 1，复位值=0000,0000B)

7	6	5	4	3	2	1	0
P0M1.7	P0M1.6	P0M1.5	P0M1.4	P0M1.3	P0M1.2	P0M1.1	P0M1.0

P1M0 (地址=91H，端口 1 模式寄存器 0，复位值=0000,0000B)

7	6	5	4	3	2	1	0
P1M0.7	P1M0.6	P1M0.5	P1M0.4	P1M0.3	P1M0.2	P1M0.1	P1M0.0

P1M1 (地址=92H，端口 1 模式寄存器 1，复位值=0000,0000B)

7	6	5	4	3	2	1	0
P1M1.7	P1M1.6	P1M1.5	P1M1.4	P1M1.3	P1M1.2	P1M1.1	P1M1.0

P2M0 (地址=95H，端口 2 模式寄存器 0，复位值=0000,0000B)

7	6	5	4	3	2	1	0
P2M0.7	P2M0.6	P2M0.5	P2M0.4	P2M0.3	P2M0.2	P2M0.1	P2M0.0

P2M1 (地址=96H，端口 2 模式寄存器 1，复位值=0000,0000B)

7	6	5	4	3	2	1	0
P2M1.7	P2M1.6	P2M1.5	P2M1.4	P2M1.3	P2M1.2	P2M1.1	P2M1.0

P3M0 (地址=B1H，端口 3 模式寄存器 0，复位值=0000,0000B)

7	6	5	4	3	2	1	0
P3M0.7	P3M0.6	P3M0.5	P3M0.4	P3M0.3	P3M0.2	P3M0.1	P3M0.0

P3M1 (地址=B2H，端口 3 模式寄存器 1，复位值=0000,0000B)

7	6	5	4	3	2	1	0
P3M1.7	P3M1.6	P3M1.5	P3M1.4	P3M1.3	P3M1.2	P3M1.1	P3M1.0

P4M0 (地址=B3H，端口 4 模式寄存器 0，复位值=0000,0000B)

7	6	5	4	3	2	1	0
P4M0.7	P4M0.6	P4M0.5	P4M0.4	P4M0.3	P4M0.2	P4M0.1	P4M0.0

P4M1 (地址=B4H，端口 4 模式寄存器 1，复位值=0000,0000B)

7	6	5	4	3	2	1	0
P4M1.7	P4M1.6	P4M1.5	P4M1.4	P4M1.3	P4M1.2	P4M1.1	P4M1.0

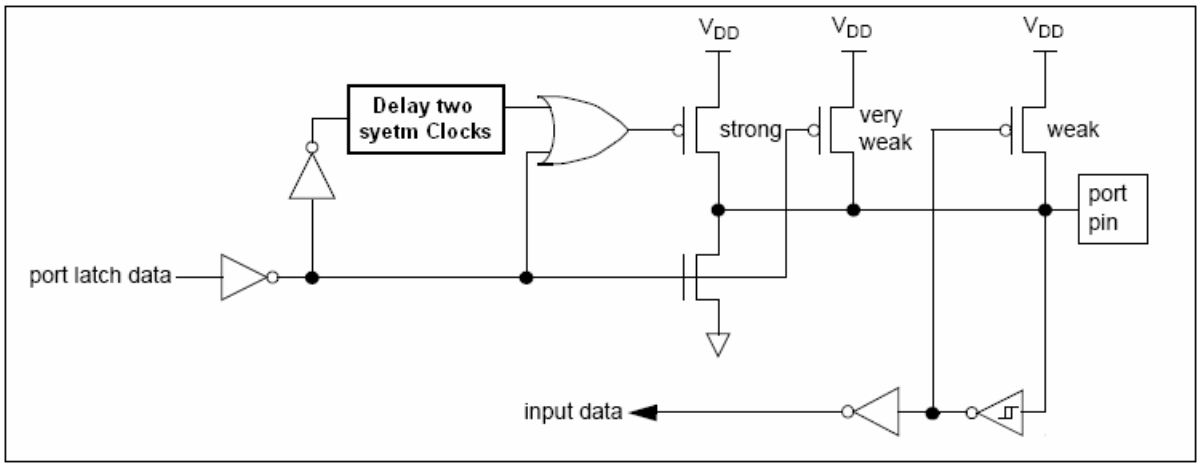
10.1.1 准双向 I/O

端口引脚工作在准双向模式时与标准 8051 端口引脚类似。一个准双向端口用作输入和输出时不需要对端口重新配置。这种可能是因为端口输出逻辑高时，弱上拉，允许外部器件拉低引脚。当输出低时，强的驱动能力可吸收大电流。在准双向输出时有三个上拉晶体管用于不同的目的。

其中的一种上拉，称为微上拉，只要端口寄存器的引脚包含逻辑 1 则打开。如果引脚悬空，则这种非常弱上拉提供一个非常小的电流将引脚拉高。第二种上拉称为“弱上拉”，端口寄存器的引脚包含逻辑 1 时且引脚自身也在逻辑电平时打开。这种上拉对准双向引脚提供主要的电流源输出为 1。如果引脚被外部器件拉低，这个弱上拉关闭，只剩一个微上拉。为了在这种条件下将引脚拉低，外部器件不得不吸收超过弱上拉功率的电流，且拉低引脚在输入的极限电压之下。第三种上拉称为“强”上拉。这种上拉用于加速准双向端口的上升沿跳变，当端口寄存器从逻辑 0 到逻辑 1 时。当这发生时，强上拉打开两个 CPU 时钟，快速将端口引脚拉高。

准双向端口配置如图 10-1 所示。准双向端口有施密特触发器来抑制输入噪音。

图 10-1 准双向 I/O

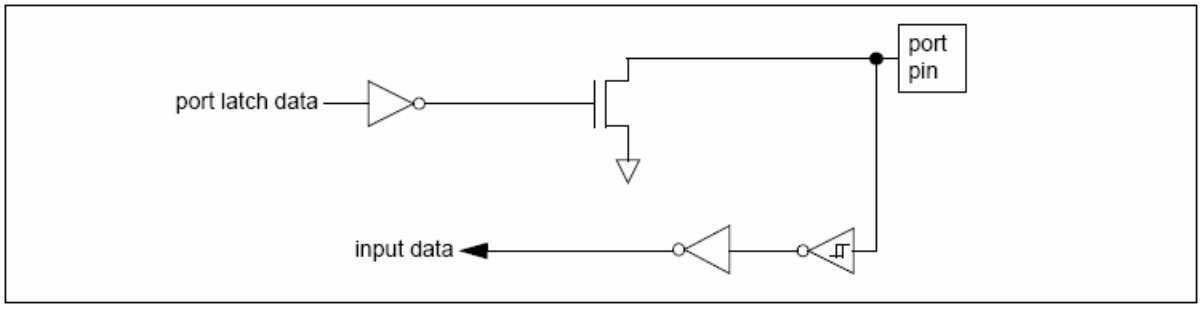


10.1.2 开漏输出

配置为开漏输出时，当端口寄存器包含逻辑 0 时，关闭所有上拉，只有端口引脚的下拉晶体管。使用这个功能配置应用，端口引脚必须有外部上拉，典型的将电阻接到 VDD。这个模式的下拉和准双向端口的模式相同。另外，在这种配置下的端口输入引脚的输入路径和准双向模式相同。

开漏输出端口配置如图 10-2 所示。开漏输入也有一个施密特触发器用来抑制噪音。

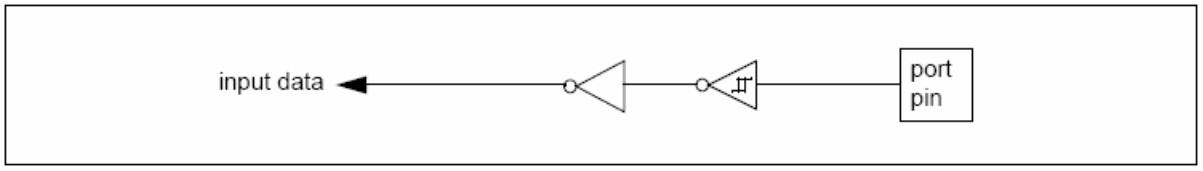
图 10-2 开漏输出



10.1.3 输入口(高阻输入)

输入配置一个施密特触发器但是在引脚上没有任何上拉电阻，如下图 10-3 所示。

图 10-3 输入

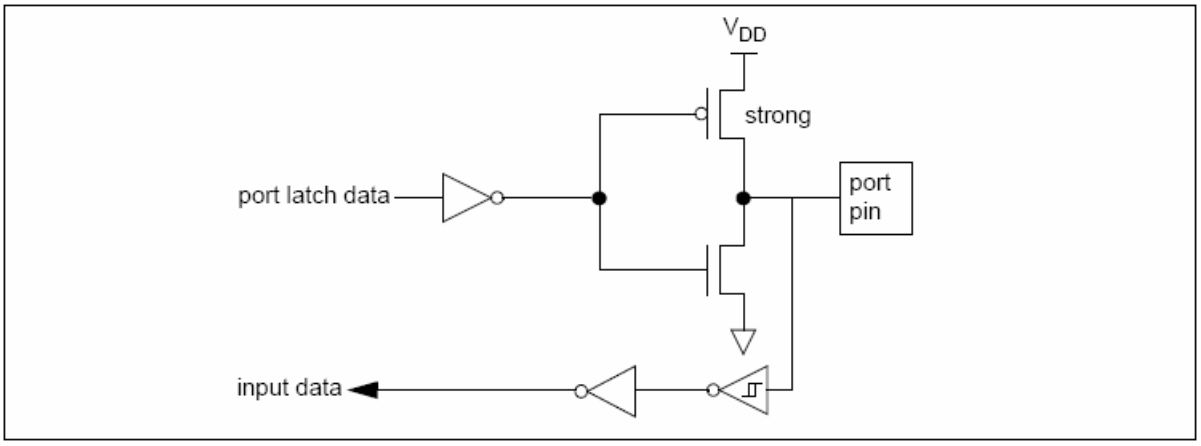


10.1.4 推挽输出

推挽输出配置有下拉，和开漏输出、准双向输出模式有着相同的结构，当端口寄存器包含逻辑 1 时提供一个连续的强上拉。当一个端口输出需要更大的电流时可配置为推挽输出模式。另外，在这种配置下的端口输入引脚和输入路径的准双向模式的配置相同。

上拉端口配置如图 10-4 所示。上拉端口引脚也包含一个输入施密特触发器用来降低噪音。

图 10-4 上拉输出



10.2 I/O 口用作 ADC 功能

端口 1 用作于可选择功能的模拟输入。为了获得最好的模拟性能，引脚用作 ADC 时应将数字输出禁止。这可能通过将引脚配置为输入模式来实现。

10.3 I/O 口注意事项

MPC82G516 的每个输出都设计有吸收典型 LED 的驱动电流能力，然而，所有端口的总输出最大电流不能超过极限值。请参考 29 章：绝对最大值

10.4 GPIO 示例代码

(1). 功能需求: 设置P1.0 为仅输入模式

汇编语言代码范例:

```
P1Mn0       EQU         01h

    ORL     P1M0, #P1Mn0
    ANL     P1M1, #(0FFh + P1Mn0)       ; 配置 P1.0 为仅输入模式
    SETB    P1.0                        ; 设置 P1.0 数据为“1”而使能输入模式
```

C 语言代码范例:

```
#define   P1Mn0       0x01

    P1M0 |= P1Mn0;
    P1M1 &= ~P1Mn0;                     //配置 P1.0 为仅输入模式
    P10 = 1;                            //设置 P1.0 数据为“1”而使能输入模式
```

(2). 功能需求: 设置 P1.0 为推挽输出

汇编语言代码范例:

```
P1Mn0       EQU         01h

    ANL     P1M0, #(0FFh - P1Mn0)
    ORL     P1M1, #P1Mn0                ; /配置 P1.0 为推挽输出
    SETB    P1.0
```

C 语言代码范例:

```
#define   P1Mn0       0x01

    P1M0 &= ~P1Mn0;
    P1M1 |= P1Mn0;                      //配置 P1.0 为推挽输出
    P10 = 1;                            // P10 输出高电平
```

(3). 功能需求: 设置 P1.0 为漏极开路输出模式

汇编语言代码范例:

```
P1Mn0       EQU         01h

    ORL     P1M0, #P1Mn0
    ORL     P1M1, #P1Mn0                ; 配置 P1.0 为漏极开路输出
    SETB    P1.0                        ; 设置 P1.0 数据为“1”而使能漏极开路输出模式
```

C 语言代码范例:

```
#define   P1Mn0       0x01

    P1M0 |= P1Mn0;
    P1M1 |= P1Mn0;                      //配置 P1.0 为漏极开路输出
    P10 = 1;                            //设置 P1.0 数据为“1”而使能漏极开路输出模式
```

11 定时器/计数器

MPC82G516 有三个 16 位定时器/计数器：定时器 0、定时器 1 和定时器 2。每一个包含两个 8 位寄存器，THx 和 TLx(这里，x=0、1 或 2)。所有这些操作既可配置为定时器或事件记数器。

定时器功能，TLx 寄存器每 12 个时钟周期或 1 个周期加 1，通过软件来选择 。因此可认为为计数器时钟周期。每记 12 个时钟周期，计数速率达 1/12 的晶振频率。

计数器功能，下降沿时寄存器加 1，根据外部输入引脚 T0、T1 或 T2。在这些功能中，每个时钟周期对外部输入信号(T0 引脚和 T1 引脚)进行采样，每 12 个时钟周期对 T2 引脚采样。当采样信号出现一个高电平接着一个低电平，计数加 1。当检测到跳变时新计数值出现在寄存器中。对定时器 0 和定时器 1 来说，需要用两个时钟周期来识别下降沿跳变，最大的计数速率为 1/2 的晶振频率；对于定时器 2，需要用 24 个时钟周期来识别下降沿跳变，最大计数速率为 1/24 的晶振频率。外部输入信号没有严格的周期限制，但是要确保在电平改变前至少有一次采样，对定时器 0 和定时器 1 来说信号应该至少保持一个时钟，定时器 2 需要 12 个时钟周期。

对定时器 0 和定时器 2 来说，除了标准 8051 定时器的功能之外，添加了一些新的特征。下面的子章节将详细描述这些定时器/计数器。

11.1 定时器 0 和定时器 1

定时器或计数器功能通过专用寄存器 TMOD 的控制位 C/T 来选择，如下所示。这两个定时器/计数器有四种工作模式，通过 TMOD 的位对(M1，M0)来选择。这两个定时器/计数器的模式 0、1 和模式 2 是相同的，模式 3 是不同的。除了 TMOD 之外，其它专用寄存器 TCON 和 AUXR2 包含几个控制位和状态位与这两个定时也相关，也如下所示。

TMOD (地址=89H,定时器/计数器模式控制位，复位值=0000,0000B)

定时器1				定时器0			
7	6	5	4	3	2	1	0
GATE	C/-T	M1	M0	GATE	C/-T	M1	M0

GATE：当门控位置位时，只有在/INT0 或/INT1 引脚是高电平且 TR0 或 TR1 控制位置位时，定时器/计数器 0 或 1 使能。
当门控制位清零时，只要 TR0 或 TR1 置 1 定时器 0 或 1 使能。

C/T： 定时器或计数器选择器。清零为定时器功能(从内部系统时钟输入)。置位为计数器功能(从 T0 或 T1 引脚输入)。

M0	M1	工作模式
0	0	8 位定时器/计数器。THx 与 TLx 作为 5 位预分频器
0	1	16 位定时器/计数器。THx 与 TLx 串联；没有分步频器
1	0	8 位自动重载定时器/计数器。THx 保持一个值，并在每次溢出时加载到 TLx
1	1	(定时器 0)TL0 是一个 8 位定时器/计数器并通过标准定时器 0 的控制位控制。TH0 仅仅是一个 8 位定时器通过定时器 1 的控制位控制
1	1	(定时器 1)定时器/计数器停止

TCON (地址=88H,定时器/计数器控制位,复位值=0000,0000B)

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1：定时器 1 溢出标志位。定时器/计数器溢出时由硬件置位。处理器进入中断向量程序由硬件清零。
TR1：定时器 1 运行控制位。通过软件置位/清零开启或关闭定时器/计数器 1。
TF0：定时器 0 溢出标志位。定时器/计数器溢出时由硬件置位。处理器进入中断向量程序由硬件清零。
TR0：定时器 0 运行控制位。通过软件置位/清零开启或关闭定时器/计数器 0。

AUXR2 (地址=A6H,辅助寄存器2,复位值=0000,0000B)

7	6	5	4	3	2	1	0

T0X12	T1X12	URM0X6	S2TR	S2SMOD	S2TX12	S2CKOE	T0CKOE

T0X12：当 C/T=0 时，定时器 0 的时钟源选择。

置位选择 Fosc 作为系统时钟源，清零选择 Fosc/12 作为时钟源。

T1X12：当 C/T=0 时，定时器 1 的时钟源选择。

置位选择 Fosc 作为系统时钟源，清零选择 Fosc/12 作为时钟源。

T0CKOE：置位/清零来使能/禁止从 P3.4 输出定时器 0 时钟。

四种工作模式在以下的文本中描述。

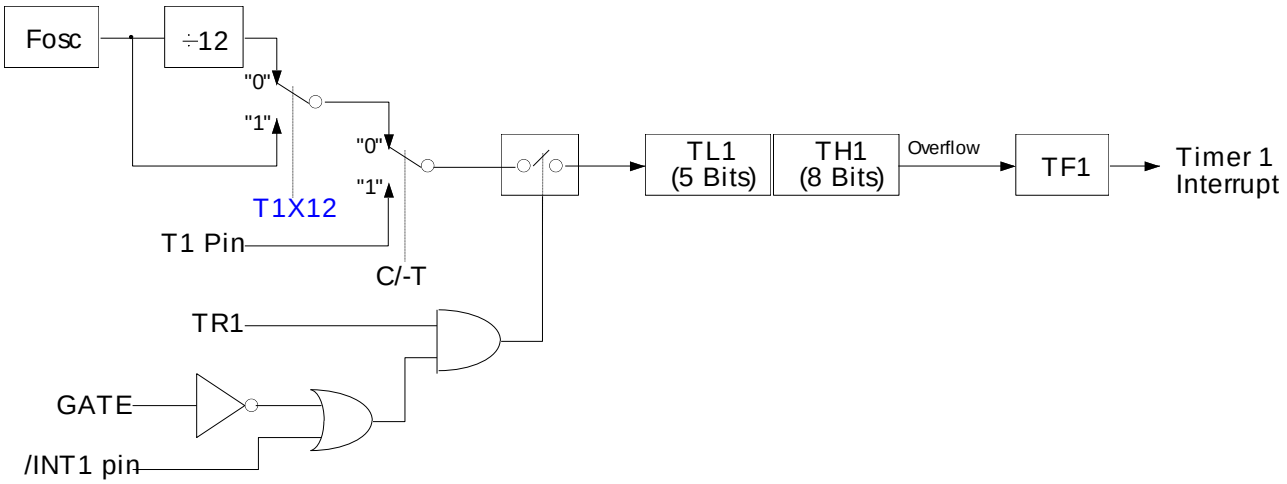
11.1.1 模式 0：13 位定时/计数器

定时器 0 和定时器 1 的模式 0 看起来像一个 32 预分频的 8 位计数器。且这两个定时器运行模式 0 是相同的。图 11-1 展示模式 0 的运行。

在这个模式，定时器寄存器配置为一个 13 位寄存器。计数器所有位从全 1 翻转到全 0，置位定时器中断标志位 TFx。当 TRx=1 且 GATE=0 或/INTx=1，定时器使能输入计数。(置 GATE=1 时通过外部输入/INTx 控制定时器，以便脉冲宽度测量)。TRx 和 TFx 控制位在专用寄存器 TCON。GATE 位在 TMOD。有两个不同的 GATE 位，一个是定时器 0(TMOD.7)另一个是定时器 0(TMOD.3)。

13 位寄存器包含 THx 的所有 8 位和 TLx 的低 5 位。TLx 的高 3 位是不确定的可以忽略。置位运行标志(TRx)不会清除寄存器。意思是说用户在开始计数前应对 THx 和 TLx 进行初始化。

图11-1. 定时器1工作在模式0: 13位定时器/计数器

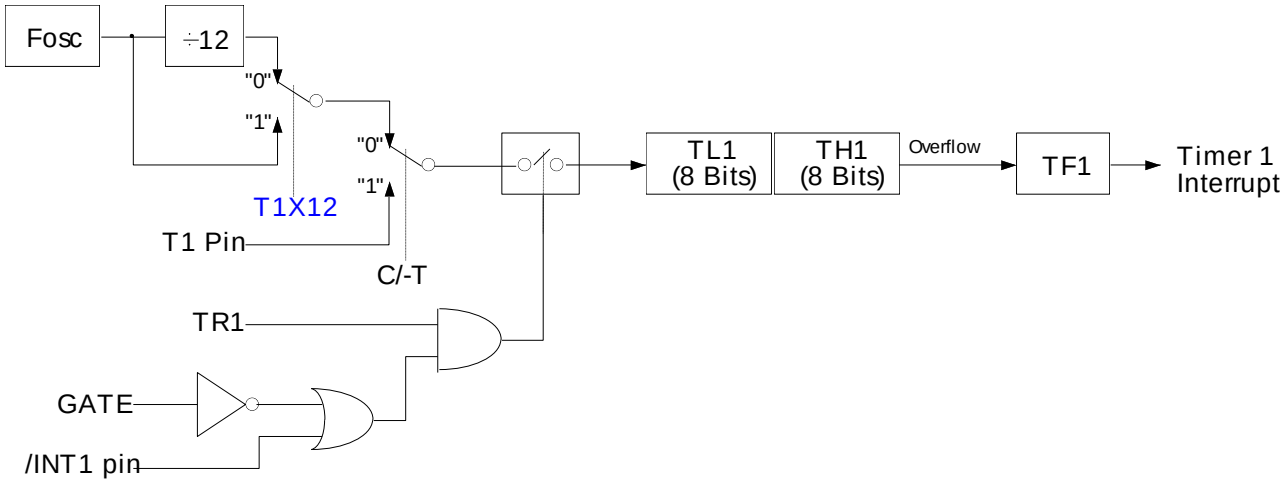


** Fosc is the system clock.*

11.1.2 模式1：16位定时/计数器

除了定时器的寄存器使用全部16位外，模式1和模式0是相同的。参考图11-2。在这个模式，THx和TLx串联，没有预分频。

图 11-2. 定时器1的模式1：16位定时器/计数器

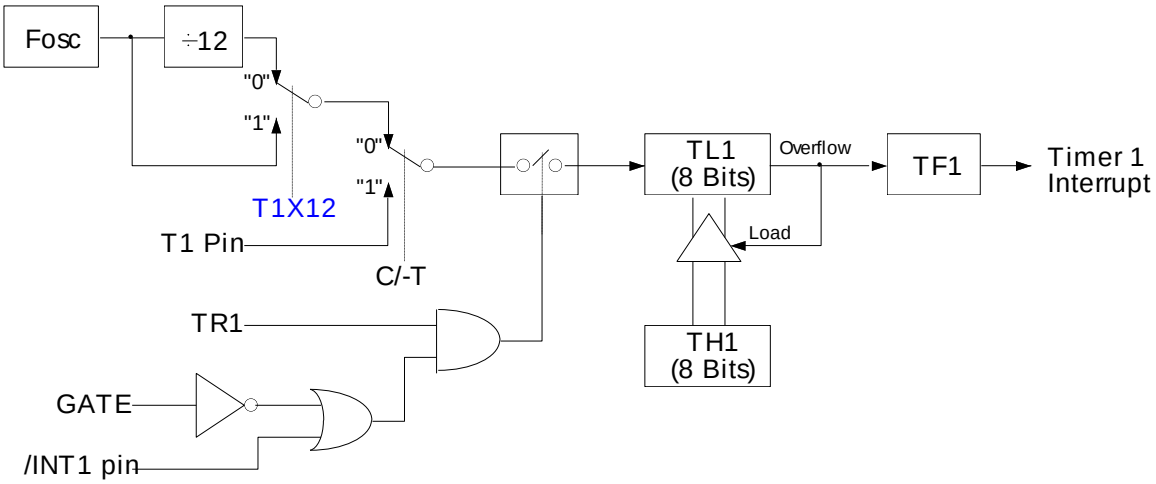


** Fosc is the system clock.*

11.1.3 模式 2：8 位自动加载

模式 2 配置定时器寄存器为一个自动加载的 8 位计数器(TLx)，如图 11-3 所示。TLx 溢出不仅置位 TFx，而且也将 THx 的内容加载到 TLx，THx 内容由软件预置，加载不会改变 THx 的值。

图 11-3. 定时器 1 的模式 2：8 位自动加载



** Fosc is the system clock.*

11.1.4 模式 3：两个 8 位定时/计数器

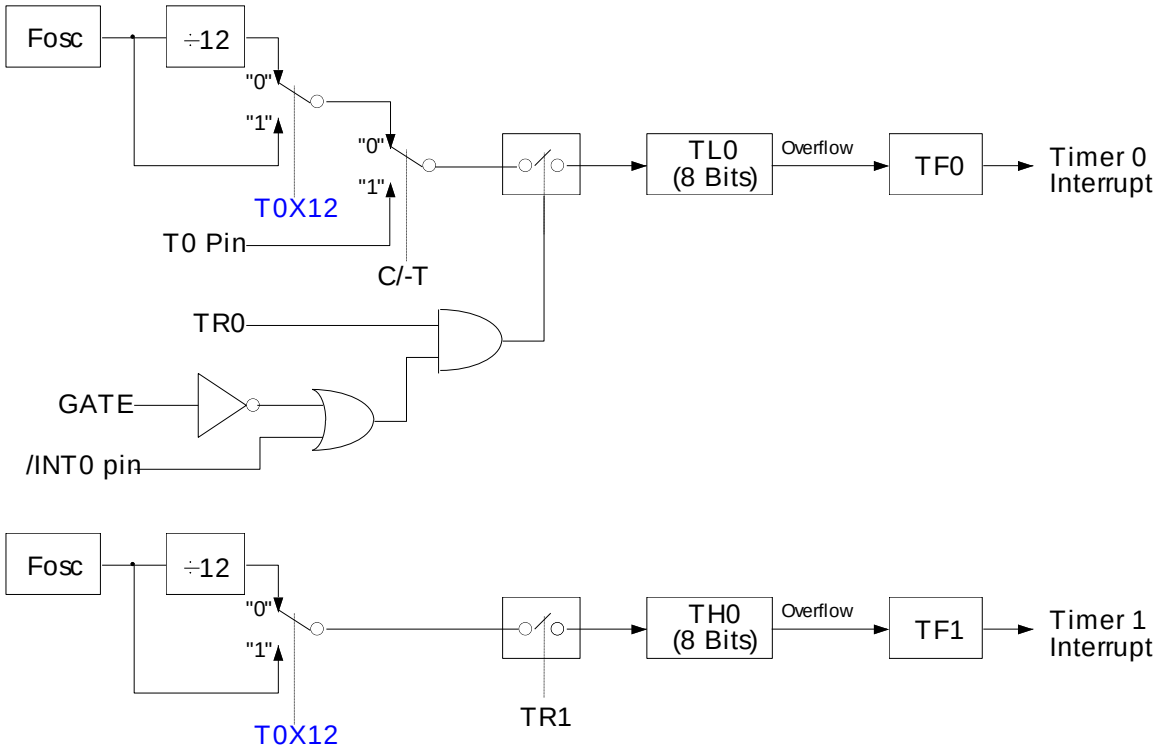
定时器 1 在模式 3 保持计数值。效果和设置 TR1=0 一样。

定时器 0 在模式 3 建立 TL0 和 TH0 两个独立的计数器。定时器 0 的模式 3 的逻辑图如图 11-4 所示。TL0 使用定

时器 0 控制位：C/T、GATE、TR0、/INT0 和 TF0。TH0 锁定为定时器功能(每个机器周期计数)且接替定时器 1 来使用 TR1 和 TF1，因从 TH0 控制定时器 1 中断。

模式 3 提供当有额外的需求应用时的一个 8 位时器或计数器时。当定时器 0 在模式 3 时，定时器 1 可打开或关闭并切换到脱离，进入到自己的模式 3，或仍然可用作为串行口的波特率发生器，或者不需要中断的其它应用。

图 11-4 定时器 0 工作在模式 3：两个 8 位定时/计数器



** Fosc is the system clock.*

11.1.5 定时器 0 的可编程时钟输出模式

使用定时器 0 的可编程时钟输出模式，则从引脚 T0CK0(P3.4)输出占空比为 50%的时钟周期。输出频率根据系统时钟频率(Fosc)和加载值到 TH0 寄存器，公式如下所示。

$$\text{Clock-Out Frequency} = \frac{Fosc}{n \times (256 - TH0)}$$

Where,

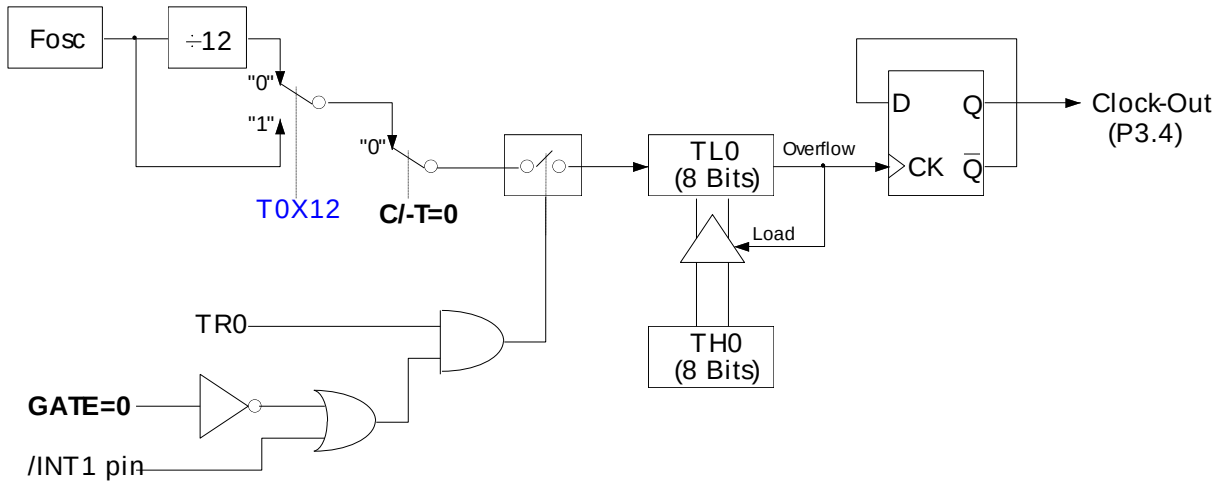
n=24 if T0X12=0,

n=2 if T0X12=1.

定时器 0 的可编程时钟输出模式编程步骤如下：

- 在 AUXR2 寄存器置位 T0CKOE。
- 清除定时器 0 的 C/T 位在 TMOD 寄存器。
- 清除定时器 0 的 GATE 位在 TMOD 寄存器。
- 从公式计算出 8 位自动加载值并输入到 TH0 寄存器。
- 在 TL0 寄存器输入一个 8 位初始值。可以和自动加载值相同。
- 通过设置 TCON 寄存器的 TR0 位启动定时器。

图 11-5. 定时器 0 的可编程时钟输出模式



** Fosc is the system clock.*

11.2 定时器 2

定时器 2 是一个 16 位定时器/计数器，既可作为一个定时器也可以作为一个事件计数器，通过专用寄存器 T2CON 的 C/T2 位来选择。定时器 2 有四种工作模式：捕获、自动加载(向上或向下计数)、波特率发生器和可编程时钟输出，通过专用寄存器 T2CON 和 T2MOD 来选择，如下所示。

T2CON (地址=C8H, 定时器/计数器2控制寄存器,复位值=0000,0000B)

7	6	5	4	3	2	1	0
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/-T2	CP/-RL2

TF2：定时器 2 溢出标志位，定时器 2 溢出置位且必须通过软件清零。当 RCLK=1 或 TCLK=1 时，TF2 不会被置位。
EXF2：定时器 2 外部标志位，在 EXEN2=1 时，且在 T2EX 上有负跳变时加载或捕获将引起置位。当定时器 2 中断使能时，EXF2=1 时将引起 CPU 进入定时器 2 中断向量程序。EXF2 必须通过软件清零。EXF2 在向上/向下计数器模式不会产生中断。
RCLK：接收时钟控制位。置位时，串行口使用定时器 2 溢出脉冲来接收，在模式 1 和模式 3 时。RCLK=0 使用定时器 1 溢出脉冲来产生接收时钟。
TCLK：传送时钟控制位。置位时，串行口使用定时器 2 溢出脉冲来发送，在模式 1 和模式 3 时。TCLK=0 使用定时器 1 溢出脉冲来产生发送时钟。
EXEN2：定时器 2 外部使能位。置位时，如果定时器 2 没有用作串行口时钟，在 T2EX 的负跳变时捕获或加载并作为结果。
TR2：定时器 2 的启始和停止位。逻辑 1 时启动定时器。
C/T2：定时器或计数器选择。清零时，选择内部定时器。置位时，选择外部事件计数器(下降沿触发)。
CP/RL2：捕获/加载控制位。置位时，如果 EXEN2=1，在 T2EX 的负跳变时将产生捕获。清零时，如果 EXEN2=1，定时器 2 溢出或 T2EX 上有负跳变时将产生自动加载。当 RCLK=1 或 TCLK=1 时，这一位被忽略并强制加载在定时器 2 溢出时。

T2MOD (地址=C9H,定时器/计数器 2 控制寄存器,复位值=xxxx,xx00B)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	T2OE	DCEN

T2OE：定时器 2 时钟输出使能位，置位使能清零禁止。
DCEN：定时器 2 向下计数使能位，置位使能清零禁止。

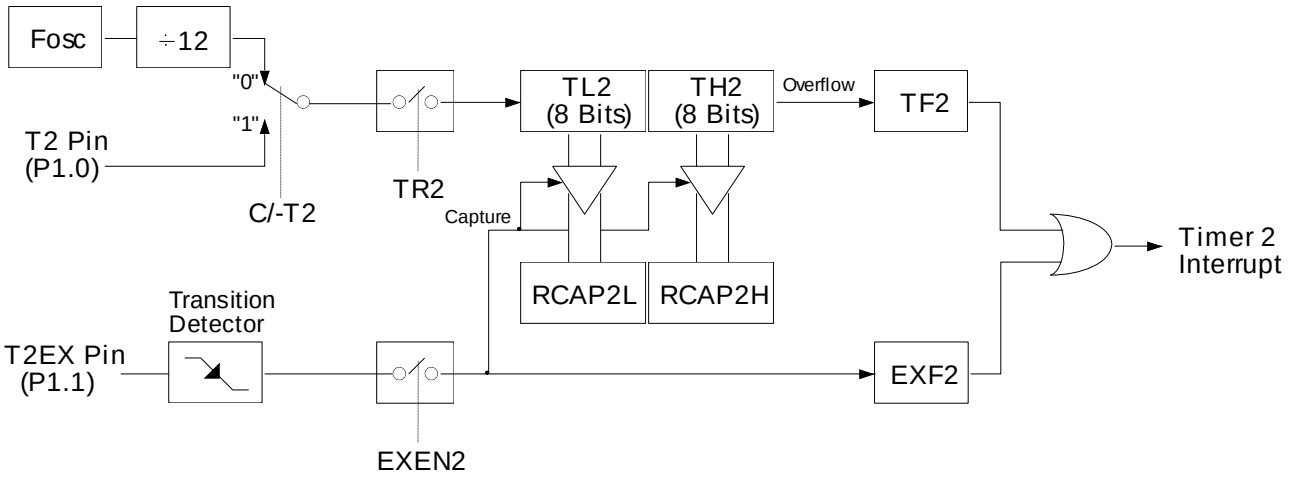
表 11-1. 定时器 2 运行模式

RCLK + TCLK	CP/-RL2	TR2	DCEN	T2OE	Mode
x	x	0	x	0	定时器关闭
1	x	1	0	0	波特率发生器
0	1	1	0	0	16位捕获
0	0	1	0	0	16位自动加载 (仅向上计数)
0	0	1	**1**	0	16位自动加载(向上计数或向下计数)
0	0	1	0	**1**	可编程时钟输出

11.2.1 捕获模式

在捕获模式，有两个选项通过 T2CON 中的 EXEN2 位来选择。如果 EXEN2=0，定时器 2 做为一个 16 位的定时器或计数器，向上溢出，定时器 2 溢出时 TF2 置位。这位可以用来产生中断(通过使能 IE 寄存器中的定时器 2 中断位)。如果 EXEN2=1，定时器 2 仍然向上， 当外部输入信号 T2EX 由下降沿跳变时引起定时器 2 的寄存器 TH2 和 TL2 分别对应的捕获到 RCAP2H 和 RCAP2L。另外，T2EX 的跳变引起 T2CON 的 EXF2 置位，且 EXF2 位(象 TF2)将产生一个中断(中断向量的位置和定时器 2 溢出中断位置相同)。捕获模式图解如图 11-6。(在这个模式 TL2 和 TH2 没有加载值。直到从 T2EX 捕获事件发生，在 T2EX 引脚跳变或 Fosc/12 的脉冲产生时计数器仍然保持计数)。

图 11-6. 定时器2捕获模式



** Fosc is the system clock.*

11.2.2 自动加载模式 (加计数或减计数)

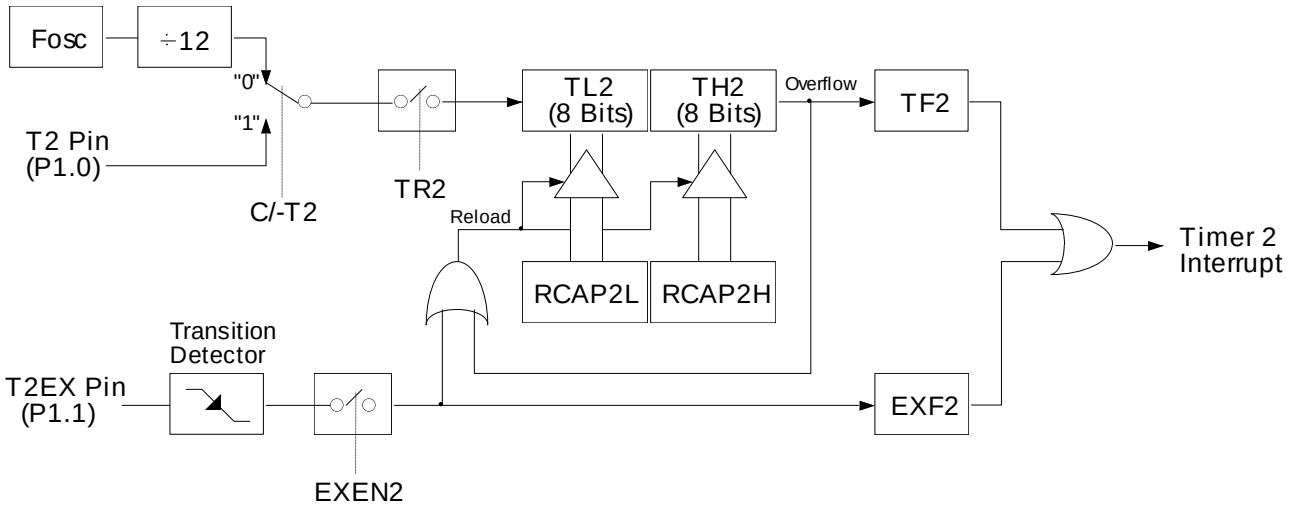
在 16 位自动加载模式，定时器既可配置成定时器也可以配置成计数器(C/T2 在 T2CON 寄存器)，接着编程向上或向下计数。计数方向由 T2MOD 寄存器的 DCEN 位来决定(向下计数使能)。在复位之后，DCEN=0 意思是默认为定时器 2 向上计数。如果 DCEN 置位，定时器 2 向上或向下计数由 T2EX 引脚的值来决定。

图 11-7 示 DCEN=0，自动使能定时器 2 向上计数。这个模式有两个选项可以通过 T2CON 寄存器的 EXEN2 位来选择。 如果 EXEN2=0，定时器向上计数 0XFFFF 接着计数将置位 TF2(溢出标志位)。这将引起定时器 2 的寄存器将 RCAP2L 和 RCAP2H 的值加载。RCAP2L 和 RCAP2H 的值由软件预置。如果 EXEN2=1，一个溢出或在输入 T2EX 的一个负跳变将触发加载 16 位值。跳变将置位 EXF2 位。当 TF2 或 EXF2 置 1 时，如果定时器 2 中断使能，将产生中断。

图 11-8 示 DCEN=1，使能定时器 2 向上或向下计数。这种模式下允许 T2EX 引脚控制计数方向。当 T2EX 的引脚为逻辑 1 时定时器 2 向上计数。定时器 2 在 0FFFFH 时溢出并置位 TF2 标志位，如果中断使能将产生中断。溢出也将引起 RCAP2L 和 RCAP2H 的 16 位值加载到定时器的寄存器 TL2 和 TH2。当 T2EX 的引脚为逻辑 0 时定时器 2 向下计数。当 TL2 和 TH2 和存储在 RCAP2L 和 RCAP2H 的值相等时将产生下溢。下溢将置位 TF2 标志位并将 0FFFFH 加载到定时器的寄存器 TL2 和 TH2。

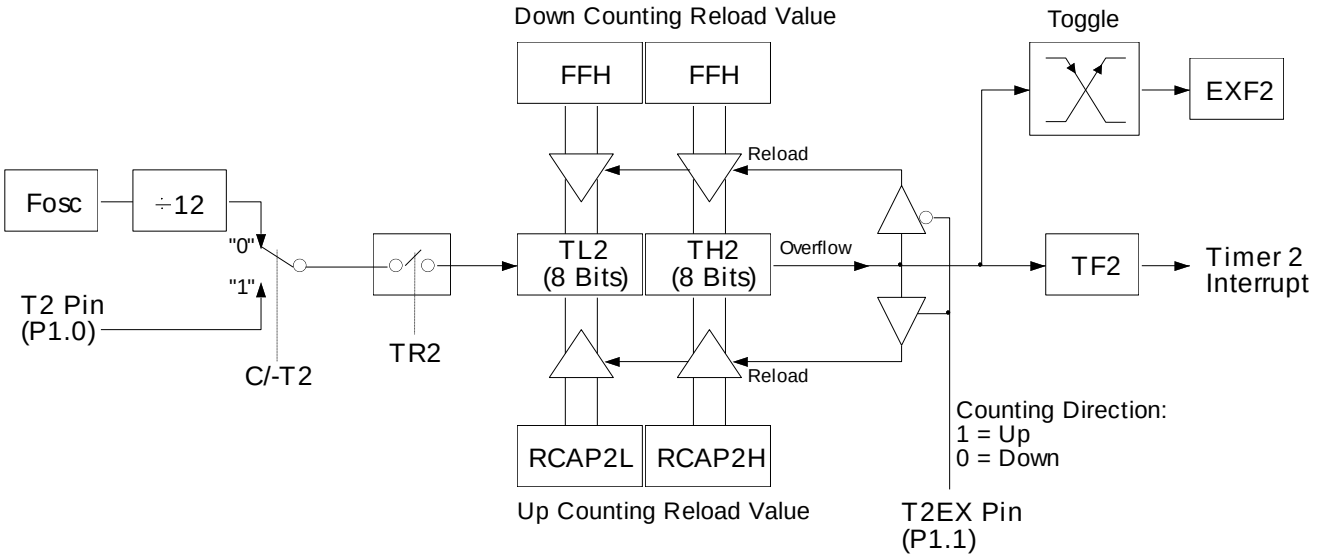
当定时器 2 下溢或上溢时外部标志位 EXF2 将被触发。如果需要 EXF2 可作为 17 位分辨率。EXF2 标志位在这个模式下不会产生中断。

图11-7. 定时器2自动加载模式 (DCEN=0)



** Fosc is the system clock.*

图 11-8. 定时器2自动加载模式(DCEN=1)



** Fosc is the system clock.*

11.2.3 波特率发生器模式

T2CON 寄存器的 RCLK 和或 RCLK 位允许串行口发送和接收波特率既可源自定时器 1 或定时器 2。当 TCLK=0 时，定时器 1 作为串行口传送波特率发生器。当 TCLK=1，定时器 2 作为串行口传送波特率发生器。RCLK 对串行口接收波特率有相同的功能。有了这两位，串行口可能有不同的接收和发送波特率，一个通过定时器 1 来产生，另一个通过定时器 2 来产生。

图 11-9 示定时器 2 在波特率发生器模式。波特率发生器模式像自动加载模式，翻转时将把寄存器 RCAP2H 和 RCAP2L 的值加载到定时器 2 的寄存器，RCAP2H 和 RCAP2L 的值由软件预置。

模式 1 和 3 的波特率由定时器 2 的溢出速率决定

$$模式1和模式3波特率 = \frac{定时器2溢出速率}{16}$$

定时器既可配置为“定时器”或“计数器”工作方式。在许多应用场合，配置成“定时器”工作方式(C/T2=0)。当定时器 2 作为波特率发生器时定时器操作是不同的。

通常，作为一个定时器将在 1/12 的系统时钟频率加 1。作为一个波特率发生器，系统时钟频率的 1/2 加 1。波特率计算公式如下：

$$模式1和模式3的波特率 = \frac{Fosc}{2x(65536-[RCAP2H,RCAP2L]} x\frac{1}{16}$$

这里：Fosc 是系统时钟。RCAP2H，RCAP2L 的内容为一个 16 位的无符号数，可由如下计算出：

$$[RCAP2H,RCAP2L] = 65536 - \frac{Fosc}{32x波特率}$$

定时器 2 作为一个波特率发生器模式如图 11-9 所示，只有在 T2CON 寄存器的位 RCLK 和/或 TCLK=1 为 1 时有效。注意 TH2 翻转不会置位 TF2，也不会产生中断。因而，当定进器 2 在波特率发生器模式时定时器中断不需要禁止。如果 EXEN2(T2 外部中断使能位)置位，T2EX(定时器/计数器 2 触发输入)的负跳变将置位 EXF2(T2 外部标志位)，但是不会引起从(RCAP2H，RCAP2L)到(TH2，TL2 的)重载。因此，当定时器 2 作为波特率发生器时，如果需要的话，T2EX 也可以作为传统的外部中断。

当定时器 2 在波特率发生器模式时，不能试着去读 TH2 和 TL2。作为一个波特率发生器，定时器 2 在 1/2 的系统时钟频率或从 T2 引脚的异步时增 1；在这些条件下，读写操作将会不正确。寄存器 RCAP2 可以读，但是不可以写，因为写和重载重叠并引起写和/或加载错误。在进入定时器 2 或 RCAP2 寄存器时定时器不可以关闭(清零 TR2)。

图11-9.　定时器2的波特率发生器模式

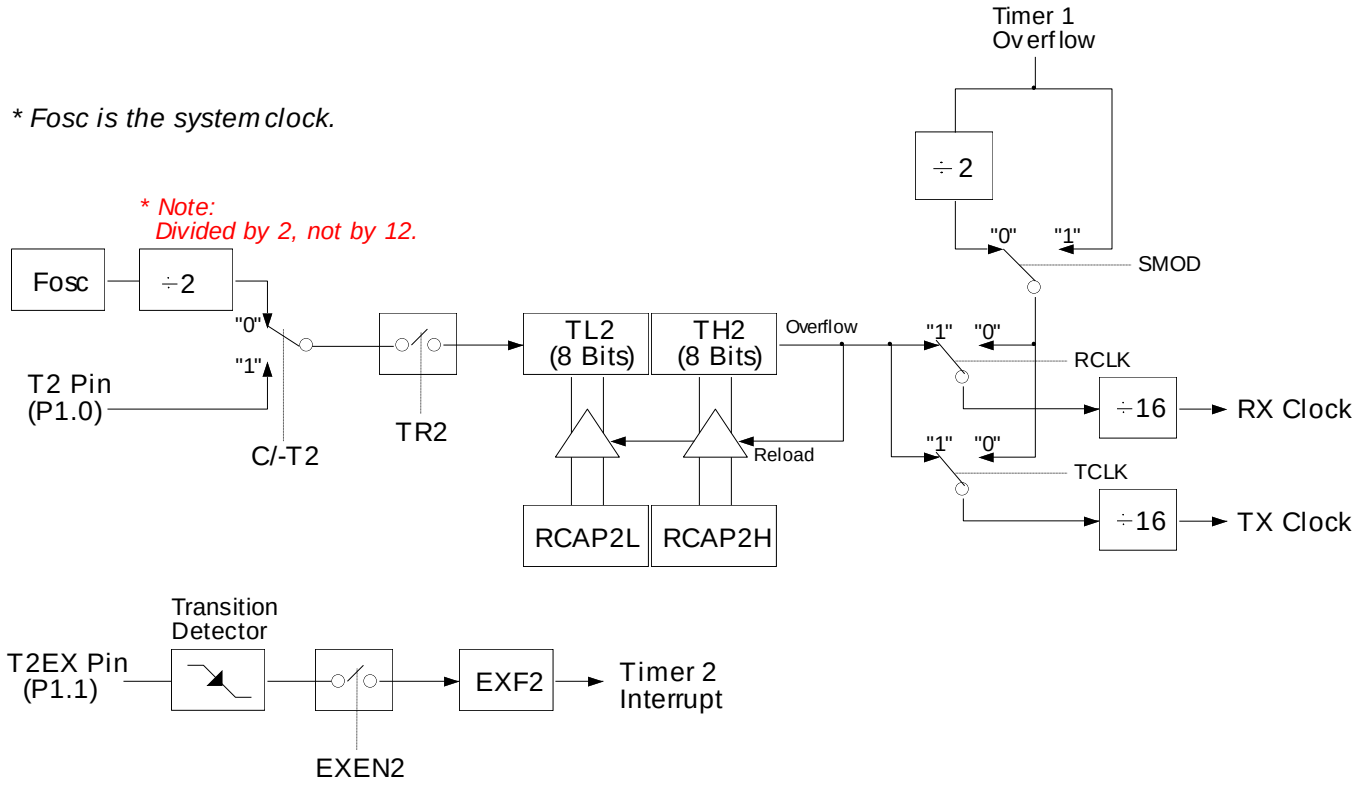


表 11-2和表 11-3 列出不同的经常使用的波特率和如何从定时器2获得。

表11-2. 定时器2产生常用的波特率 @ Fosc=11.0592MHz

Baud Rate	Timer 2 in Baud Rate Generator Mode		
	[RCAP2H, RCAP2L]	RCAP2H	RCAP2L
3	64384	FBH	80H
6	64960	FDH	C0H
12	65152	FEH	80H
18	65248	FEH	E0H
24	65392	FFH	70H
48	65440	FFH	A0H
72	65464	FFH	B8H
96	65488	FFH	D0H
144	65500	FFH	DCH
192	65518	FFH	EEH
384	65524	FFH	F4H
576	65530	FFH	FAH
115200	65533	FFH	FDH

表11-3. 定时器2产生常用的波特率 @ Fosc=22.1184MHz

波特率	定时器2在波特率发生器模式		
	[RCAP2H, RCAP2L]	RCAP2H	RCAP2L
300	63232	F7H	00H
600	64384	FBH	80H
1200	64960	FDH	C0H
1800	65152	FEH	80H
2400	65248	FEH	E0H
4800	65392	FFH	70H
7200	65440	FFH	A0H
9600	65464	FFH	B8H
14400	65488	FFH	D0H
19200	65500	FFH	DCH
38400	65518	FFH	EEH
57600	65524	FFH	F4H
115200	65530	FFH	FAH

11.2.4 定时器2的可编程时钟输出模式

使用定时器2的可编程时钟输出模式，则从引脚T2CKO(P1.0)输出占空比为50%的时钟周期。
输出频率由系统时钟频率(Fosc)和在RCAP2H和RCAP2L寄存器的重载值来决定，如下公式：

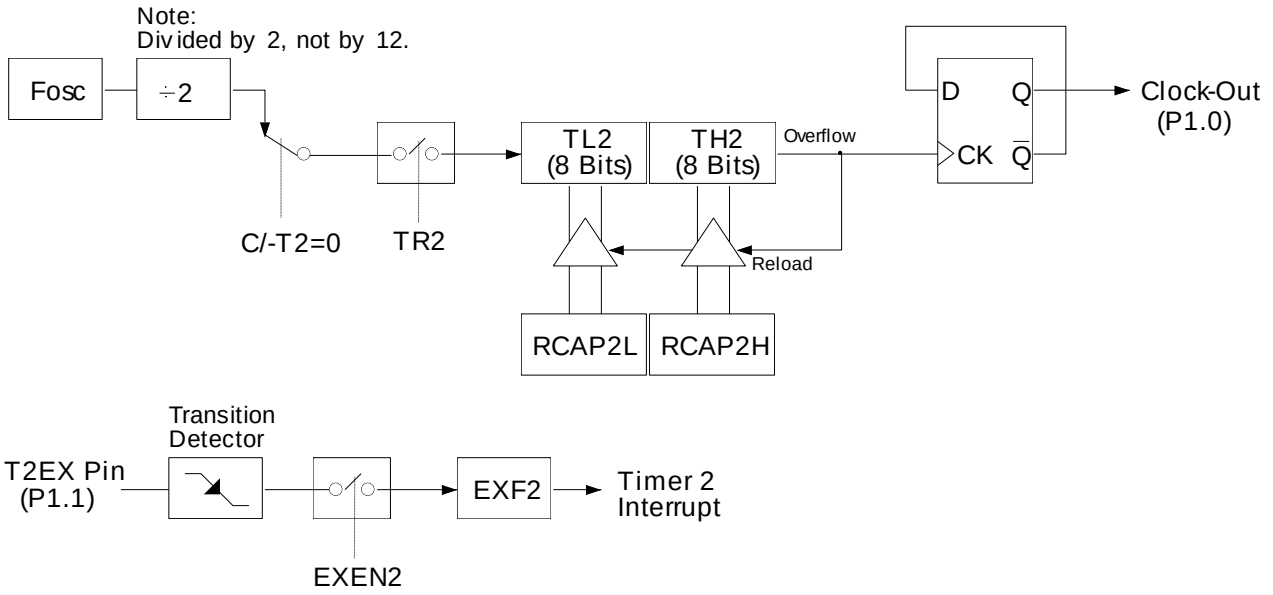
$$\text{时钟输出频率} = \frac{Fosc}{4x(65536 - [RCAP2H, RCAP2L]}$$

这里[RCAP2H,RCAP2L]=RCAP2H和RCAP2L内容产生的一个16位无符号数。
定时器2 的可编程时钟输出模式编程步骤如下：

- 置位T2MOD寄存器的T2OE位。
- 清除T2CON寄存器的C/T2位。
- 从公式计算出16位加载值并输入到RCAP2H和RCAP2L寄存器。
- 在TH2和TL2输入一个16位初始值。可以和重载值相等。
- 设置T2CON的TR2控制位开启动定时器。

在时钟输出模式，定时器2翻转不会产生中断，这和用作波特率发生器时相似。可同时使用定时器2作为一个波特率发生器和时钟发生器。注意，波特率和时钟输出都由定时器2的溢出速率来决定。

图 11-10. 定时器2 的可编程时钟输出模式



** Fosc is the system clock.*

11.3 定时器 0/1 示例代码

(1). 功能需求: 定时器 T0 以 10KHz 的频率唤醒空闲模式，SYSCLK = 12MHz 晶振

汇编语言代码范例:

```
T0M0        EQU     01h
T0M1        EQU     02h
PT0     EQU     02h
PT0H        EQU     02h
IDL     EQU     01h

    ORG     0000h
    JMP     main

    ORG     0000Bh
time0_isr:
    to do…
    RETI

main:                                   ;
    MOV     TH0,#(256-100)          ; 设置定时器 0 溢出率为 = SYSCLK x 100
    MOV     TL0,#(256-100)          ;
    ANL     TMOD,#0F0h              ; 设置定时器为模式 2
    ORL     TMOD,#T0M1              ;
    CLR     TF0                     ; 清定时器 0 标志位

    ORL     IP,#PT0                 ; 选择定时器 0 中断优先级
    ORL     IPH,#PT0H               ;

    SETB    ET0                     ; 使能定时器 0 中断
    SETB    EA                      ; 使能全局中断

    SETB    TR0                     ; 启动定时器 0 运行

    ORL     PCON,#IDL               ; 设置 MCU 进入空闲模式
    JMP     $
```

C 语言代码范例:

```
#define     T0M0        0x01
#define     T0M1        0x02
#define     PT0     0x02
#define     PT0H        0x02
#define     IDL     0x01

void time0_isr(void) interrupt 1
{
    To do…
}
void main(void)
{
    TH0 = TL0 = (256-100);          //设置定时器 0 溢出率为 = SYSCLK x 100
    TMOD &= 0xF0;                   // S 设置定时器为模式 2
    TMOD |= T0M1;
    TF0 = 0;                        //清定时器 0 标志位

    IP |= PT0;                      //选择定时器 0 中断优先级
    IPH |= PT0H;
    ET0 = 1;                        //使能定时器 0 中断
    EA = 1;                         //使能全局中断

    TR0 = 1;                        //启动定时器 0 运行
    PCON =IDL;                      //设置 MCU 进入空闲模式
    while(1);
}
```

(2). 功能需求: 选择定时器 0 时钟源为 SYSCLK (使能 T0X12)

汇编语言代码范例:

```
T0M0        EQU         01h
T0M1        EQU         02h
PT0     EQU         02h
PT0H        EQU         02h
T0X12       EQU         80h

    ORG     0000h
    JMP     main

    ORG     0000Bh
time0_isr:
    to do…
    RETI

main:
    ORL     AUXR, #T0X12                ; 选择定时器 0 时钟源为 SYSCLK
    CLR     TF0                         ; 清定时器 0 标志位

    ORL     IP,#PT0                     ; 选择定时器 0 中断优先级
    ORL     IPH,#PT0H                   ;

    SETB    ET0                         ; 使能定时器 0 中断
    SETB    EA                          ; 使能全局中断

    MOV     TH0, #(256 - 240)           ;中断间隔 20us
    MOV     TL0, #(256 - 240)           ;

    ANL     TMOD,#0F0h                  ; 设置定时器 0 为模式 2
    ORL     TMOD,#T0M1                  ;


    SETB    TR0                         ; 启动定时器 0
    JMP     $
```

C 语言代码范例:

```
#define     T0M0        0x01
#define     T0M1        0x02
#define     PT0     0x02
#define     PT0H        0x02
#define     T0X12       0x80

    AUXR |= T0X12                       //选择定时器 0 时钟源为 SYSCLK
    TF0 = 0;

    IP |= PT0;                          //选择定时器 0 中断优先级
    IPH |= PT0H;

    ET0 = 1;                            //使能定时器 0 中断
    EA = 1;                             //使能全局中断

    TH0 = TL0 = (256 - 240);

    TMOD &= 0xF0;                       //设置定时器 0 为模式 2
    TMOD |= T0M1;

    TR0 = 1;                            //启动定时器 0
```

12 串行口

12.1 标准 UART

一个全双工的串行口，意思是可以同时发送和接收数据。它也有一个接收缓冲，意味着在前一个接收到的字节没有从寄存器读出前，就可以开始接收第二个字节。(然而，如果第一个字节在第二个字节接收完成前仍然没有被读出，则其中的一个字节将会丢失)。串行口的接收和发送寄存器都通过特殊寄存器 SBUF 来访问。写到 SBUF 加载到传送寄存器，当从 SBUF 读时是通过一个物理上独立分离的接收寄存器。

串行口可以工作在四种模式：模式 0 提供同步通讯同时模式 1、2 和模式 3 提供异步通讯。异步通讯作为一个全双工的通用异步收发器(UART)，可以同时发送和接收并使用不同的波特率。

模式 0：8 位数据(低位先出)通过 RXD 传送和接收。TXD 总是作为输出移位时钟。波特率固定为系统时钟频率的十二分之一，也就是 Fosc/12。

模式 1：10 位通过 TXD 传送或通过 RXD 接收，一个起始位(0)，8 个数据位(低位优先)，和一个停止位(1)。在接收时，停止位进入到专用寄存器(SCON)的 RB8。波特率是可变的。

模式 2：11 位通过 TXD 传送或通过 RXD 接收，起始位(0)，8 个数据位(低位优先)，一个可编程的第九个数据位和一个停止位(1)。在传送时，第 9 个数据位(TB8 在 SCON 寄存器)可以分配为 0 或者 1。例如，奇偶检验位(P，在 PSW 寄存器)可以移到 TB8 中。在接收时，第九个数据位到 SCON 寄存器中的 RB8，同时忽略停止位。波特率可以配置为 1/32 或 1/64 的系统时钟频率。也就是 Fosc/64 或 Fosc/32。

模式 3：11 位通过 TXD 传送或通过 RXD 接收，起始位(0)，8 个数据位(低位优先)，一个可编程的第九个数据位和一个停止位(1)。实际上，模式 3 和模式 2 除了波特率不相同之外其它的都相同。模式 3 的波特率是可变的。

在四种模式中，使用 SBUF 作为一个目的寄存器，可以通过任何指令发起传输。在模式 0，当 RI=0 且 REN=1 时启动接收。在其它模式，在 REN=1 时，通过收到起始位启动接收。

12.1.1 多处理器通讯

模式 2 和 3 在用作多处理器通讯时有特殊的规定。在这两种模式，接收 9 个数据位。第 9 个数据位存入 RB8，接着进来一个停止位。端口可以编程为在 RB8=1 时，当收到停止位后，串口中断将激活。这种特征通过设置 SM2 位(在 SCON 寄存器中)来使能。这种方式用于多处理器系统如下：

当主处理器想传送一个数据块到多个从机中的某一个时，首先传送想要传送的目标地址标识符的地址。地址字节与数据字节的区别在于，在地址字节中第 9 位为 1，数据字节中为 0。当 SM2=1 时，收到一个数据字节将不会产生中断。然而一个地址字节将引发所有从机中断。因而所有的从机可以检测收到的字节是否是自己的地址。从机地址将清除 SM2 位并准备好接收即将进来的所有数据。从机地址不匹配的将保持 SM2 置位，并继续他们的工作，忽略进来的数据字节。

SM2 在模式 0 和模式 1 没有影响，但是可以用来检测停止位的有效性。在接收模式 1 中，如果 SM2=1，除非一个收到一个有效的停止位否则接收中断不会被激活。

12.1.2 串行口（UART）配置寄存器

串行口的控制位和状态位在专用寄存器 SCON。这个寄存器不仅包含模式选择位，也包含用来传送和接收(TB8 和 RB8)的第 9 个数据位，和串行中断位(TI 和 RI)。

SCON (地址=98H,串行口控制寄存器，复位值=0000,0000B)

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

FE：帧错误位。当接收器检测到一个无效的停止位时这位置 1。当收到有效的帧时 FE 不会自动清除，但是可以用软件清除。SMOD0 位(在 PCON 寄存器)必须置 1 来使能访问 FE 位。

SM0：串行口模式位 0(SMOD0 必须为 0 来访问 SM0 位)

SM1：串行口模式位 1。

SM0	SM1	模式	描述	波特率
0	0	0	移位寄存器	Fosc/12 或 Fosc/2
0	1	1	8 位 UART	可变的
1	0	2	9 位 UART	Fosc/64 或 Fosc/32
1	1	3	9 位 UART	可变的

这里，Fosc 是系统时钟频率。

SM2：在模式 2 和 3 时使能地址自动识别。如果 SM2=1 那么 RI 将不能设置，除非接收到的第 9 位数据(RB8)为 1，指示是一个地址，并且一个接收字节给出定的或者是一个广播地址。在模式 1，如果 SM=1 那么 RI 将不能被激活除非收到一个有效的停止位，并且接收到的字节是给定的或者是一个广播地址。在模式 0，SM 可以为 0。

REN：允许接收位。通过软件置 1 接收使能，软件清零将禁止接收。

TB8：在模式 2 和 3 时第 9 位数据被传送，根据需要通过软件置位或清零。

RB8：在模式 2 和 3 时收到第 9 位数据。在模式 1，如果 SM2=0，RB8 是收到数据的停止位。在模式 0，RB8 没有使用。

TI：发送中断标志。在模式 0 时，在第 8 位个数据位时序后由硬件置位。或者在其它模式下，在发送停止位前。在任何串行传输前，必须由软件清除。

RI：接收中断标志。在模式 0 时，在第 8 位个数据位时序后由硬件置位。或者在其它模式下，在传送的中途改变。在任何串行传输前，必须由软件清除。

PCON (地址=87H,电源控制寄存器,复位值=00xx,0000B (or 00x1,0000B 在上电复位后))

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

SMOD0：清零时 SCON.7 的功能为“SM0”，置 1 时 SCON.7 的功能为“FE”。

AUXR2 (地址=A6H,辅助寄存器 2, 复位值=00x0,0000B)

7	6	5	4	3	2	1	0
T0X12	T1X12	URM0X6	S2TR	S2SMOD	S2TX12	S2CKOE	T0CKOE

T1X12：

当 C/-T=0 时，选择定时器 1 为时钟源。

置位选择 Fosc 作为时钟源，清零时选择 Fosc/12 作为时钟源。

URM0X6：

置位时选择 Fosc/2 作为 UART 模式 0 的波特率。

清零时选择 Fosc/12 作为 UART 模式 0 的波特率。

12.1.3 波特率

在模式 0 的波特率可以为 Fosc/12 或 Fosc/2，根据控制位 URM0X6(在 AUXR2 寄存器)。这里，Fosc 是系统时钟频率。模式 1 和 3 的波特率由定时器 1 和定时器 2 的溢出速率来决定的。模式 2 的波特率由 PCON 寄存器的 SMOD 位来决定。如果 SMOD=0(复位值)，波特率是 Fosc/64；如果 SMOD=1，波特率是 Fosc/32，如下所示。

$$模式2波特率 = \frac{2^{SMOD}}{64} xFosc$$

12.1.4 使用定时器 1 产生波特率

定时器 1 作为波特率发生器(T2CON.RCLK=0，T2CON.TCLK=0)时，模式 1 和模式 3 的波特率由定时器 1 溢中速率和 SMOD 的值来决定，如下所示：

$$模式1，模式3波特率 = \frac{2^{SMOD}}{32} x(定时器1溢出速率)$$

在这个应用中可以禁止定时器 1 中断。定时器自身可配置为“定时器”或配置为“计数器”，或在三种运行模式中的一种。在多数典型应该中，被配置为“定时器”运行，并且在自动加载模式(TMOD 的高四位等于 0010B)。在那种情况下波特率由下面的公式给出：

$$模式1，模式3波特率 = \frac{2^{SMOD}}{32} x \frac{Fosc}{nx(256 - TH1)}$$

使用定时器 1 中断使能可实现非常低的波特率，配置定时器作为一个 16 位定时器(TMOD 的高 4 位为 0001B)，使

用定时器 1 中断并且用软件加载。

表 12-1 和表 12-2 列出不同的经常使用的波特率和如何从定时器 1 获得 8 位自动加载模式。

表 12-1 定时器 1 产生普通使用的波特率@Fosc=11.0592MHz

波特率	TH1，自动加载值			
	T1X12=0		T1X12=1	
	SMOD=0	SMOD=1	SMOD=0	SMOD=1
300	160	64		
600	208	160		
1200	232	208		
1800	240	224	64	
2400	244	232	112	
4800	250	244	184	112
7200	252	248	208	160
9600	253	250	220	184
14400	254	252	232	208
19200		253	238	220
38400			247	238
57600		255	250	244
115200			253	250

表 12-2. 定时器 1 产生通用的波特率 @ Fosc=22.1184MHz

波特率	TH1，自动加载值			
	T1X12=0		T1X12=1	
	SMOD=0	SMOD=1	SMOD=0	SMOD=1
300	64	-	-	-
600	160	64	-	-
1200	208	160	-	-
1800	224	192	-	-
2400	232	208	-	-
4800	244	232	112	-
7200	248	240	160	64
9600	250	244	184	112
14400	252	248	208	160
19200		250	220	184
38400		253	238	220
57600		254	244	232
115200		255	250	244

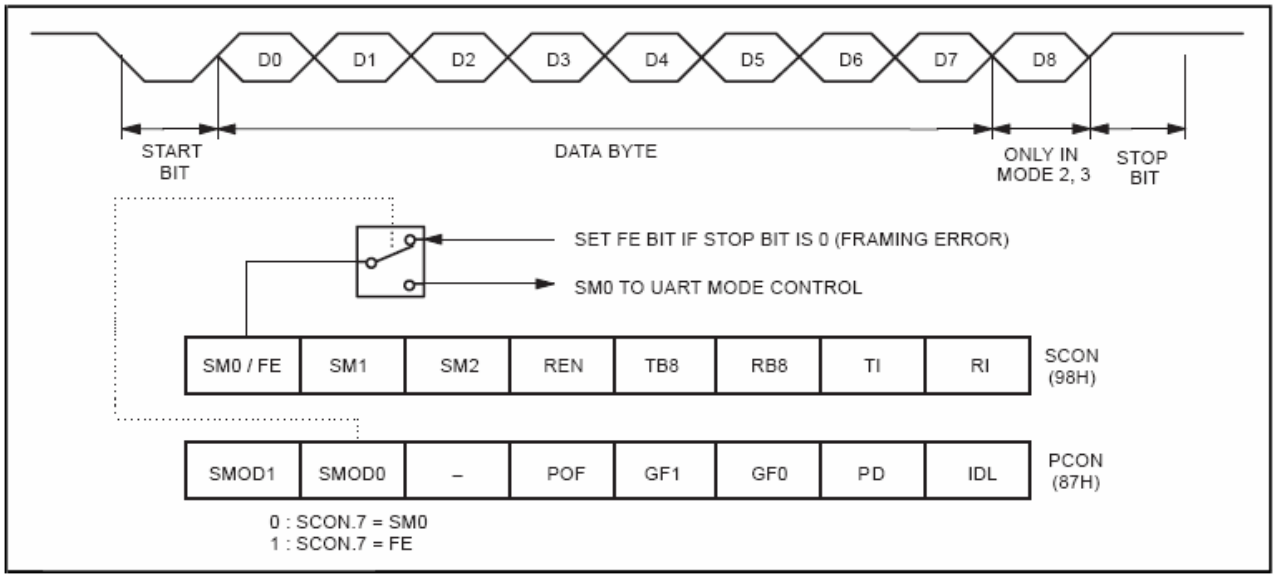
12.2 扩展的 UART 功能

除了支持标准操作外，该UART还支持通过检查是否丢失停止位来进行帧错误检测，和自动地址识别功能。

12.2.1 帧错误检测

开启帧错误检测功能后，UART 会在通讯中检测是否丢失停止位，如果丢失一个停止位，就设置SCON寄存器的FE标志位。FE标志位和SM0标志位共享SCON.7，SMOD0标志位(PCON.6)决定SCON.7究竟代表哪个标志，如果SMOD0位（PCON.6）置位则SCON.7就是FE标志，SMOD0位清零则SCON.7就是SM0标志。当SCON.7代表FE时，只能软件清零。请参考图12-5.

图12-5. UART帧错误检测



12.2.2 自动地址识别

自动地址识别通过硬件比较可以让UART识别串行码流中的地址部分，该功能免去了使用软件识别时需要大量代码的麻烦。该功能通过设定SCON的SM2位来开启。

在9位数据UART模式下，即模式2和模式3，收到特定地址或广播地址时自动置位接收中断(RI)标志，9位模式的第9位信息为1表明接收的是一个地址而不是数据。自动地址识别功能请参考图12-6。在8位模式，即模式1下，如果SM2置位并且在8位地址与给定地址或广播地址核对一致后收到有效停止位则RI置位。 模式0是移位寄存器模式，SM2被忽略。

使用自动地址识别功能可以让一个主机选择性的同一个或多个从机进行通讯，所有从机可以使用广播地址接收信息。增加了SADDR从机地址寄存器和SADEN地址掩码寄存器。

SADEN用来定义SADDR中的那些位是“无关紧要”的，SADEN掩码和SADDR寄存器进行逻辑与来定义供主机寻址从机的“给定” 地址，该地址让多个从机进行排他性的识别。

下面的实例帮助理解这个方案的通用性：

```
从机 0                      从机 1
    SADDR = 1100 0000           SADDR = 1100 0000
    SADEN = 1111 1101           SADEN = 1111 1110
    地址   = 1100 00X0          地址   = 1100 000X
```

上面的例子中SADDR是相同的值，而使用SADEN数据来区分两个从机。
从机0要求第0位必须为0，并忽略第1位的值；从机1要求第1位必须为0，并忽略第0位的值。从机0的唯一地址是1100 0010，而从机1的唯一地址是1100 0001，地址1100 0000是可以同时寻找到从机0和从机1的。

下面一个更为复杂的系统可以寻址到从机1和从机2，而不会寻址到从机0：

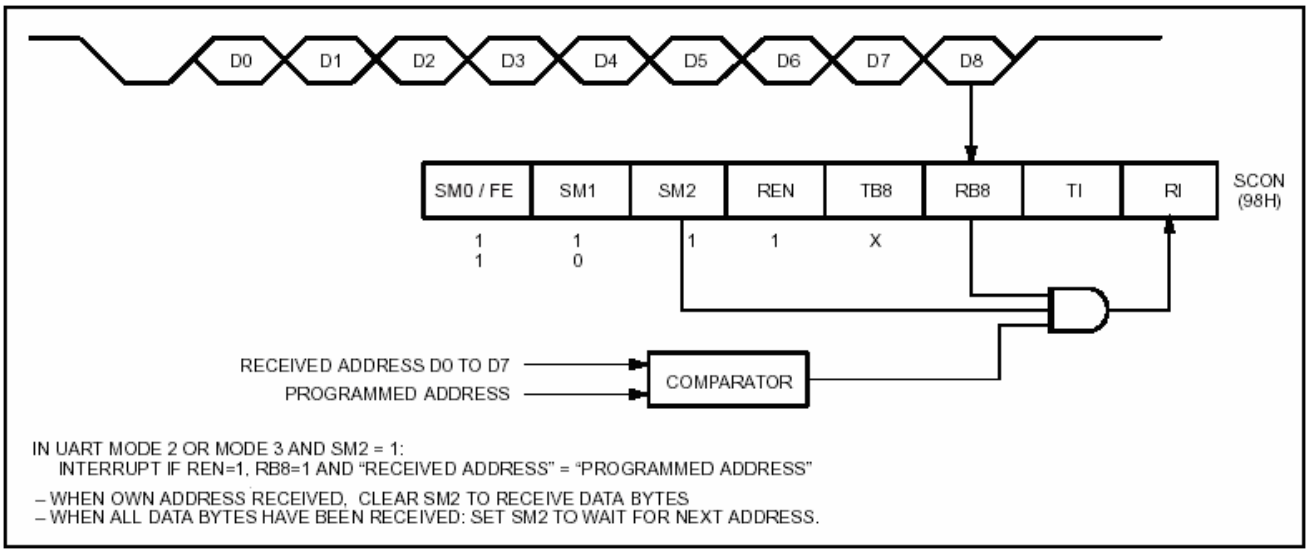
```
从机 0                        从机 1                从机 2
    SADDR = 1100 0000           SADDR = 1110 0000       SADDR = 1110 0000
    SADEN = 1111 1001           SADEN = 1111 1010       SADEN = 1111 1100
    地址   = 1100 0XX0          地址   = 1110 0X0X      地址   = 1110 00XX
```

上面的例子中，3个从机的低3位地址不一样，从机0要求第0位必须为0，1110 0110可以唯一寻址从机0；从机1要求第1位必须为0，1110 0101可以唯一寻址从机1；从机2要求第2位必须为0，它的唯一地址是1110 0011。为了寻址到从机0和从机1而不会寻址到从机2，可以使用地址1110 0100，因为这个地址第2位是1。

每个从机的广播地址SADDR和SADEN的逻辑或，0按不需关心处理。大部分情况下，使用FF作为广播地址。

复位后，SADDR（SFR地址0A9H）和SADEN（SFR地址0B9H）值均为0，这样可以接收所有地址的信息，也就有效的禁用了自动地址识别模式，从而使该处理器运行于标准80C51的UART下。

图 12-6. UART多处理器通讯，自动地址识别



13 第 2 个 UART (UART2)

MPC82G516带有第2个UART（以后就称作UART2），和第1个UART一样，也有4种运行模式，两个UART的区别如下：

(1) UART2没有增强功能：帧错误检测和自动地址识别。
(2) UART2使用特定的波特率定时器作为其波特率发生器。
(3) UART2使用端口P1.3 (S2TXD) 和 P1.2 (S2RXD) 分别作为接收和发送端口。

两个UART可以不同或相同模式、不同或相同通讯速率同时工作。

13.1 UART2 配置寄存器

以下特殊功能寄存器是和UART2相关的:

S2CON (地址=AAH, UART2控制寄存器，复位值=0000,0000B)

7	6	5	4	3	2	1	0
S2SM0	S2SM1	S2SM2	S2REN	S2TB8	S2RB8	S2TI	S2RI

S2SM0: UART2 模式选择位0.
S2SM1: UART2 模式选择位1.

S2SM0	S2SM1	模式	功能描述	波特率
0	0	0	移位寄存器	Fosc/12
0	1	1	8位 UART	可变
1	0	2	9位 UART	Fosc/64或Fosc/32
1	1	3	9位 UART	可变

表中 Fosc 是系统时钟频率。

S2SM2:使能模式2和模式3下多机通讯功能，若SM2=1，除非接收到的第9位（RB8）为1，表明接收到的是地址且和设定地址或广播地址相同，否则RI不会置位。模式1下，若SM2=1，除非收到给定地址或广播地址，且收到有效停止位，否则RI不会置位。模式0下，SM2应设为0。

S2REN:使能串行接收。软件设置使能接收，软件清零则关闭接收功能。

S2TB8: 模式2和3下，要发送的第9位数据，软件清零或设置。

S2RB8: 模式2和3下，收到的第9位数据。模式1下，若SM2=0，RB8是收到的停止位。模式0下， RB8没有用。

S2TI: 发送中断标志。硬件置位，必须软件清零。模式0下在发送完第8位数据后，或其他串行发送模式下发送停止位时，由硬件自动置位。
S2RI:接收终端标志。硬件置位，必须软件清零。模式0下接收到第8位数据后，或其他串行发送模式下接收到停止位的过程中，由硬件自动置位（例外情况请参见SM2）。

S2BUF (地址=9AH, UART2串行数据缓冲器，复位值=xxH)

7	6	5	4	3	2	1	0
(D7)	(D6)	(D5)	(D4)	(D3)	(D2)	(D1)	(D0)

S2BRT (地址=BAH, UART2波特率定时器重装寄存器，复位值=00H)

7	6	5	4	3	2	1	0
(Baud Rate Timer Reload Value)							

AUXR2 (地址=A6H, 辅助寄存器2，复位值=0000,0000B)

7	6	5	4	3	2	1	0
T0X12	T1X12	URM0X6	S2TR	S2SMOD	S2TX12	S2CKOE	T0CKOE

S2TR: UART2波特率定时器控制位。设置/清零进行打开/关闭。
S2SMOD: UART2二倍波特率使能位。设置即表示进行波特率二倍频。

S2TX12: UART2波特率定时器时钟源选择。设置选择Fosc，清零选择Fosc/12。

S2CKOE: 设置时打开UART2波特率定时器在P3.5引脚的输出功能。

AUXR (地址=8EH, 辅助寄存器, 复位值=0000, xx0xB)

7	6	5	4	3	2	1	0
URTS	ADRJ	P41ALE	P35ALE	-	-	EXTRAM	-

URTS:

0: 定时器1或定时器2可以用作模式1和模式3下的波特率发生器。

1: 当定时器1作为第一个UART的模式1或模式3下的波特率发生器时，定时器1的溢出信号被UART2波特率定时器溢出信号替代。（请参考13-3节）

13.2 UART2 波特率

13.2.1 模式 0

若 RM0X6=0,

$$\text{模式 0 波特率} = \frac{Fosc}{12}$$

若URM0X6=1,

$$\text{模式 0 波特率} = \frac{Fosc}{2}$$

13.2.2 模式1 和模式3

$$\text{模式 1, 3 波特率} = \frac{2^{S2SMOD}}{32} \times \frac{Fosc}{n \times (256-S2BRT)}$$

其中,

若S2X12=0，则n=12,
若S2X12=1，则n=1

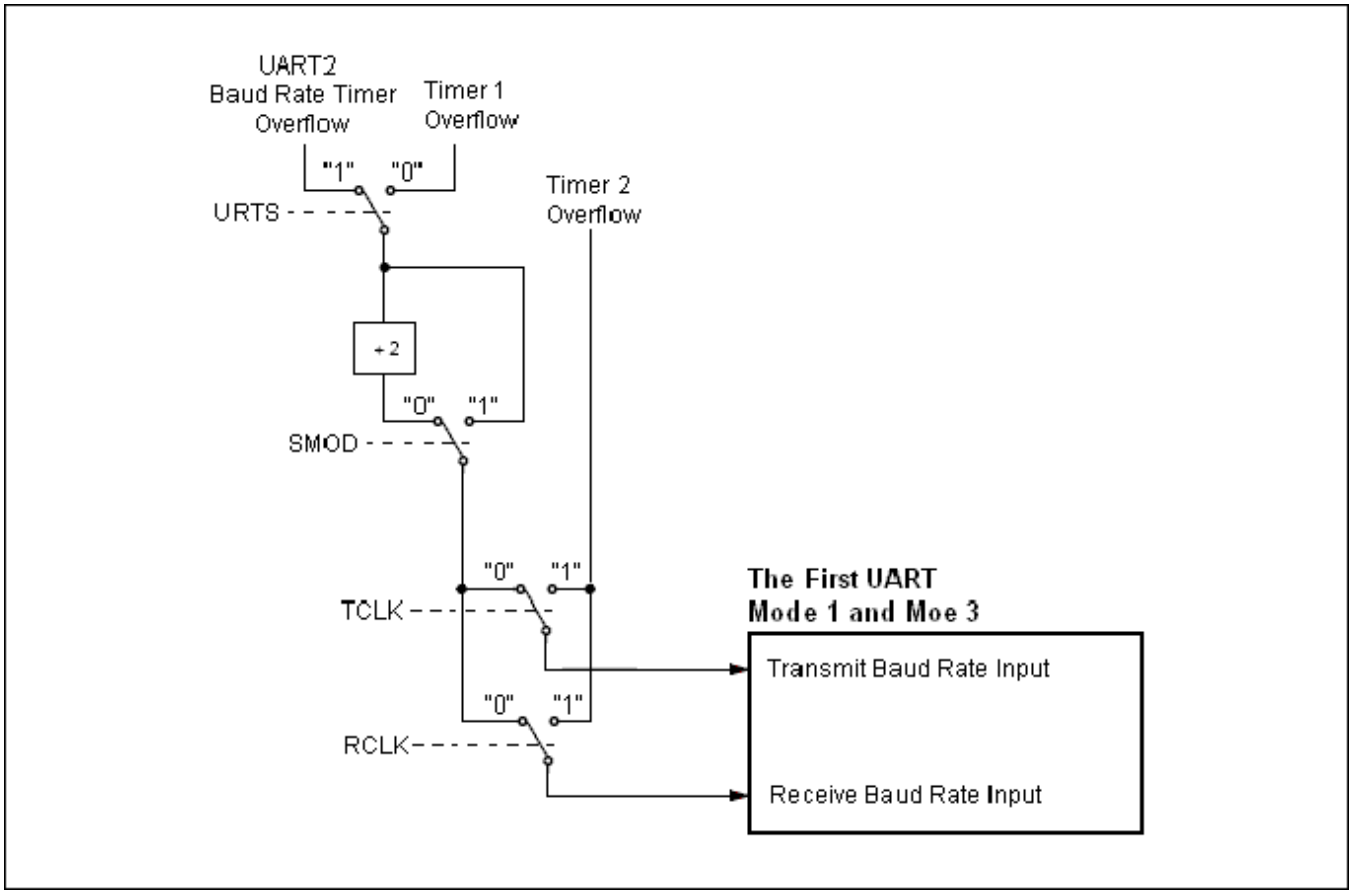
13.2.3 模式 2

$$\text{模式2 波特率} = \frac{2^{S2SMOD}}{64} \times Fosc$$

13.3 标准 UART 使用 UART2 的波特率发生器

在标准UART的模式1和模式3下，用户可以通过清除T2CON寄存器的TCLK和RCLK位选择使用定时器1作为波特率发生器。在这种情况下，若URTS位（AUXR寄存器）设为1，定时器1的溢出信号被UART2波特率定时器的溢出信号代替，换句话说，用户可以将UART2的波特率发生器作为标准UART的模式1或3下的波特率发生器，只要RCLK=0, TCLK=0 且 URTS=1。在这种条件下，定时器1可以作其他用途，当然，若UART2（模式1或3下）此时也在工作，则这两个UART运行在相同的波特率。

图 13-1. 标准UART的新波特率源



13.4 UART2 波特率发生器的可编程时钟输出

使用UART2波特率定时器，可以从引脚S2CKO（P3.5）输出一个50%占空比的时钟信号。输出的时钟信号频率取决于时钟频率（Fosc）和S2BRT寄存器中的重装值，如下所示公式计算：

$$\text{Clock-Out Frequency} = \frac{Fosc}{n \times (256 - S2BRT)}$$

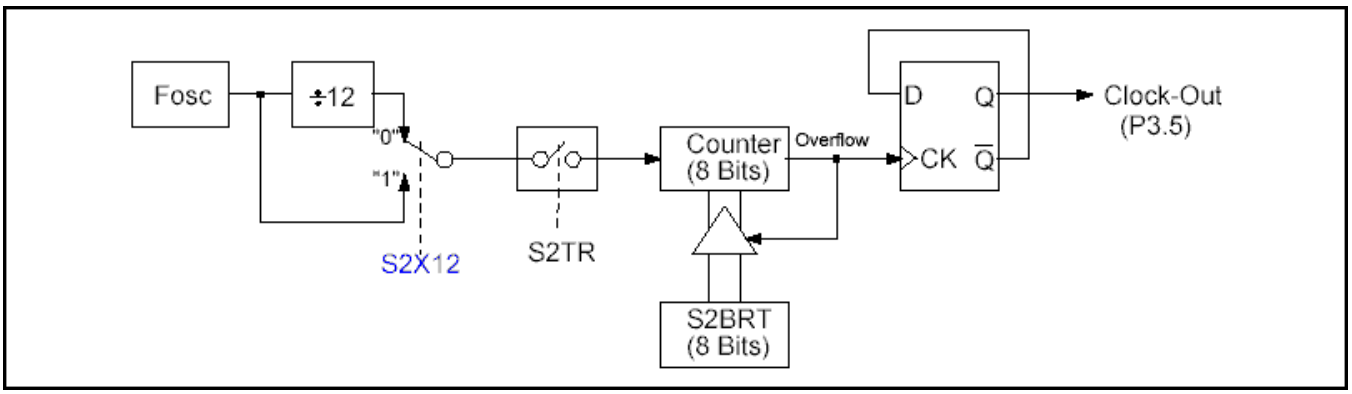
Where,
n=24 if S2X12=0,
n=2 if S2X12=1.

UART2波特率定时器的可编程时钟输出模式编程如下：

- 设置AUXR2寄存器的S2CKOE位。

- 由公式确定8位重载值并写入S2BRT寄存器。
- 设置AUXR2中的运行控制位来启动UART2波特率定时器

图 13-2. UART2波特率定时器的可编程时钟输出



13.5 串行口示例代码

(1). 功能需求: 串行口输入 RI 唤醒空闲模式

汇编语言代码范例:

```
PS              EQU             10h
PSH             EQU             10h


        ORG     00023h
uart_ri_idle_isr:
        JB      RI,RI_ISR                       ; 判断是否串行输入中断
        JB              TI,TI_ISR                                       ; 判断是否串行发送中断
RETI                                            ; 中断返回

RI_ISR:
;       Process
        CLR     RI                                      ; 清除 RI 标志
        RETI                                            ; 中断返回

TI_ISR:
;       Process
        CLR     TI                                      ; 清除 TI 标志
        RETI                                            ; 中断返回

main:
        CLR     TI                                      ; 清除 TI 标志
        CLR     RI                                      ; 清除 RI 标志
        SETB    SM1                                     ;
        SETB    REN                                     ; 8 位的模式 2，接收使能

        CALL    UART_Baud_Rate_Setting          ;参考 "錯誤! 找不到參照來源。 ~ 錯誤! 找不到參照來源。" 获得更多信息

        MOV     IP,#PSL                         ; 选择串行口中断优先级
        MOV     IPH,#PSH                        ;

        SETB    ES                                      ; 使能串行口中断
        SETB    EA                                      ; 使能全局中断

        ORL     PCON,#IDL;                      ; 设置 MCU 进入空闲模式
```

C 语言代码范例:

```
#define     PS                  0x10
#define     PSH                 0x10

void uart_ri_idle_isr(void) interrupt 4
{       if(RI)
        {
                RI=0;
                // to do ...
        }

        if(TI)
        {
                TI=0;
                // to do ...
        }
}

void main(void)
{
        TI = RI = 0;
        SM1 = REN = 1;                                  // 8 位的模式 2，接收使能
```

```
    UART_Baud_Rate_Setting()                    //参考 "錯誤! 找不到參照來源。 ~ 錯誤! 找不到參照來源。" 获得更多信息

    IP = PSL;                                   //选择串行口中断优先级
    IPH = PSH;                                  //

    ES = 1;                                     // 使能串行口中断
    EA = 1;                                     //使能全局中断

    PCON |= IDL;                                //设置 MCU 进入空闲模式
}
```

14 可编程计数器阵列(PCA)

MPC82G516带有一个可编程计数器阵列(PCA)，该功能与标准定时/计数器相比以更少的CPU占用提供了更多的定时能力。它的优点包括减少了软件复杂度并提高了精度。

14.1 PCA 概述

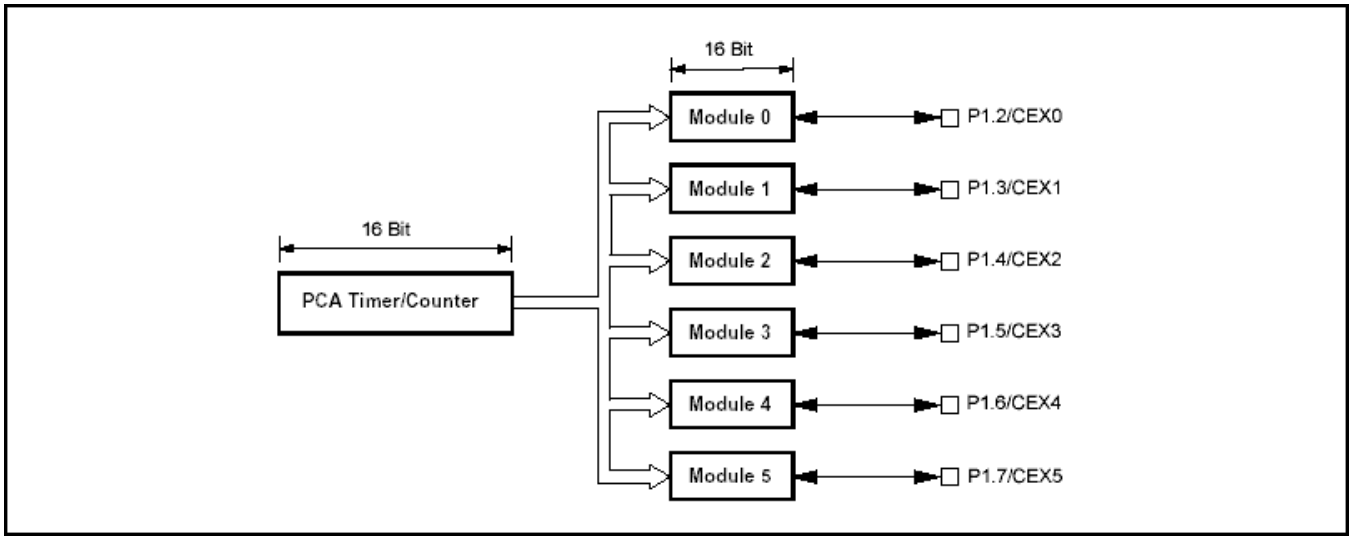
PCA由一个特定定时/计数器作为一个6比较/捕获模块的时基，图 14-1 显示了PCA的功能方框图。需要注意的是PCA定时器和模块都是16位的。如果一个外部事件同一个模块关联，那末该功能就和相应的端口1引脚共享。若某模块没有使用端口引脚，这个引脚还可以用作标准I/O。

6比较/捕获模块中的每一个都可以编程为如下任意模式：

- 上升和/或下降沿捕获
- 软件定时器
- 高速输出
- 脉宽调制输出

所有这些模式将在后面的章节进行详细讨论。这里，让我们先看看如何设置PCA时钟和模块。

图14-1. PCA 方框图



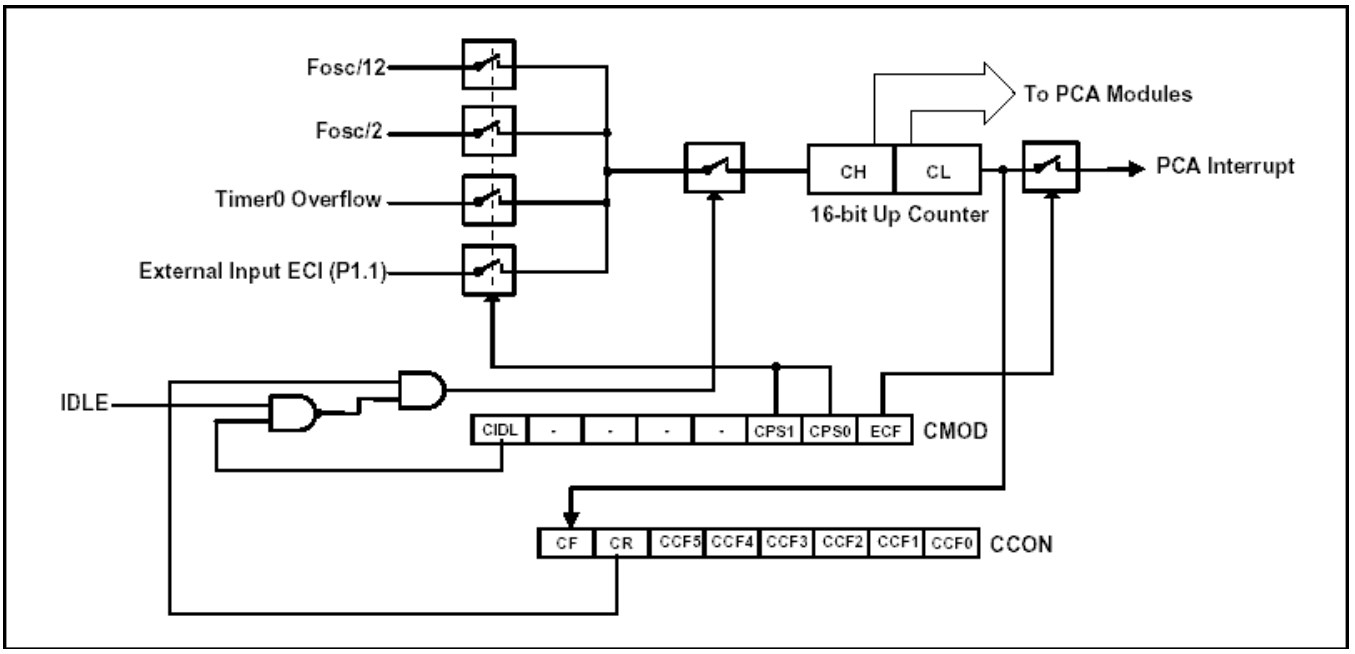
14.2 PCA 定时/计数器

PCA的定时/计数器由一个可以自由运行的16位定时器组成，如图14-2所示分为CH和CL高低两部分，它是所有模块的共有时基，它的时钟输入可以从以下来源选取：

- 1/12 系统时钟频率,
- 1/2 系统时钟频率,
- 定时器0溢出，可以让低频时钟源输入到PCA定时器.
- 外部时钟输入，ECI（P1.1）引脚的1-0反转.

特殊功能寄存器 CMOD 包含了计数脉冲选择位 (CPS1 和 CPS0) 来指定PCA定时器时钟源。这个寄存器也包括了ECF位来使能计数器溢出中断。此外，用户可以在待机模式下设置计数器待机位（CIDL），来关闭PCA定时器，这样可以进一步降低待机模式下的功耗。

图 14-2. PCA 定时/计数器



图中，Fosc是系统时钟.

CMOD (地址=D9H, PCA计数器模式寄存器)

7	6	5	4	3	2	1	0
CIDL	-	-	-	-	CPS1	CPS0	ECF

CIDL: PCA 计数器待机控制.
CIDL=0 让PCA计数器在待机模式下继续运行。
CIDL=1 待机模式下关闭PCA计数器。

CPS1-CPS0: PCA计数器时钟源选择位.

0	0	内部时钟, Fosc/12 (*Fosc代表系统时钟.)*
0	1	内部时钟, Fosc/2
1	0	定时器 0 溢出
1	1	ECI引脚输入的外部时钟源.

ECF: 使能PCA计数器溢出中断.
ECF=1 当CF位（CCON寄存器中）置位时使能中断。

如下所示的CCON寄存器包含PCA运行控制位和PCA定时器与每个模块的标志。要运行PCA，CR为（CCON.6）必须软件置位，要关闭PCA，可以清除该位。PCA计数器溢出时，CF (CCON.7)置位，并且若CMOD寄存器的ECF为置位，还会产生一个中断，CF位只能软件清零。CCF0到CCF5是模块0到模块5的相应中断标志位，当发生一个匹配或捕获事件时，硬件置位，这些位也必须软件清零。PCA中断系统如图 14-3所示。

CCON (地址=D8H, PCA控制寄存器)

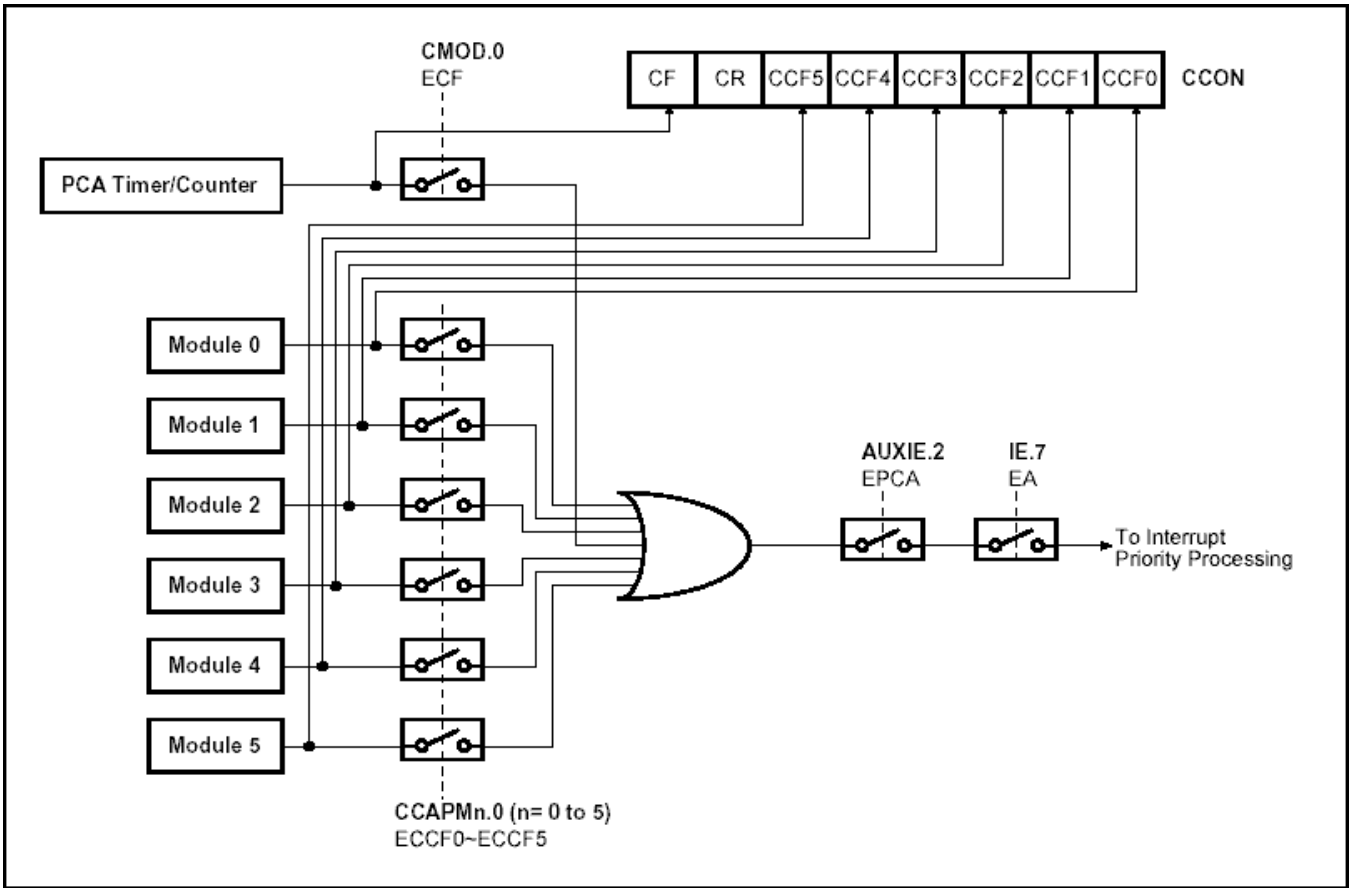
7	6	5	4	3	2	1	0
CF	CR	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0

CF: PCA 计数溢出标志。溢出时硬件置位，CF标志在CMOD寄存器的ECF位置位时会产生一个中断，CF可以硬件或软件置位，但只能软件清零。

CR: PCA 计数控制位。
软件设置来启停PCA计数器。1：启动PCA计数器，0：关闭PCA计数器。

CCF0~CCF5: PCA模块0到模块5中断标志。
发生一个匹配或捕获时硬件置位，必须软件清零。

图 14-3. PCA 中断系统



14.3 比较/捕获模块

6比较/捕获模块中的每一个都有一个模式寄存器，叫做CCAPMn（n代表0，1，2，3，4，5），来选择其工作模式。ECCFn位控制当中断标志置位时每个模块的中断开启/关闭。

CCAPMn, n=0~5 (地址=DAH~DFH, PCA模块比较/捕获寄存器)

7	6	5	4	3	2	1	0
-	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn

ECOMn: 比较器使能位。 ECOMn=1 使能比较器功能。

CAPPn: 捕获上升沿。CAPPn=1 使能上升沿捕获

CAPNn: 捕获下降沿。CAPNn=1 使能下降沿捕获

MATn: 匹配控制。MATn=1，PCA计数器同相应模块的比较/捕获寄存器匹配时设置CCON寄存器的CCFn位。

TOGn: 翻转控制。TOGn=1，PCA计数器同相应模块的比较/捕获寄存器匹配时CEXn引脚电平翻转一次。

PWMn: PWM控制。PWMn=1使能CEXn引脚用作脉宽调制输出。

ECCFn: 使能CCFn中断。使能CCON寄存器中的比较/捕获标志CCFn中断。

注:CAPNn (CCAPMn.4)位和CAPPn (CCAPMn.5)位决定了捕获发生时信号脉冲沿，若这两位同时设置，则上下降沿都会发生捕获。

每一个模块都有一对8位比较/捕获寄存器（CCAPnH, CCAPnL）与其相关联。这些寄存器用来保存捕获事件发生时或比较事件发生时的值。当某个模块用作PWM模式时，除了上面两个寄存器外，还有一个扩展的寄存器PCAPWMn被用来扩展输出占空比的范围，扩展的范围从0%到100%，步距是1/256。

PCAPWMn, n=0~5 (地址=F2H~F7H, PWM 模式辅助寄存器)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	ECAPnH	ECAPnL

ECAPnH: 扩展的第9位（最高位扩展），用在PWM模式下，与CCAPnH联用并成为其第9位。
ECAPnL: 扩展的第9位（最高位扩展），用在PWM模式下，与CCAPnL联用并成为其第9位。

14.4 PCA 模式设置

表14-1显示了不同PCA功能对应的CCAPMn寄存器设置。

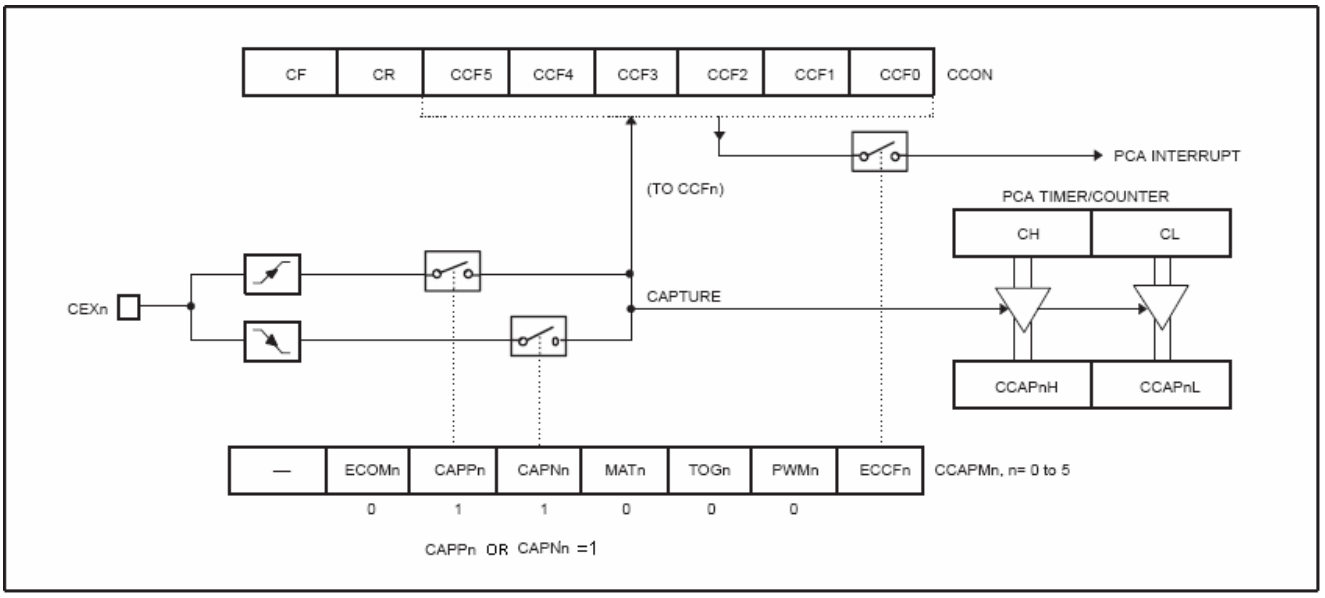
表14-1. PCA模块工作模式

ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn	模块功能
0	0	0	0	0	0	0	无操作
X	1	0	0	0	0	X	16位CEXn引脚上升沿触发捕获模式
X	0	1	0	0	0	X	16位CEXn引脚下降沿触发捕获模式
X	1	1	0	0	0	X	16位CEXn引脚跳变触发捕获模式
1	0	0	1	0	0	X	16位软件定时器
1	0	0	1	1	0	X	16位高速输出
1	0	0	0	0	1	0	8位脉宽调制器(PWM)

14.4.1 捕获模式

要让某一PCA模块工作与捕获模式，相应的CAPN和/或CAPP位必须置位。外部CEX输入会在每次跳变是采样，当有效跳变发生时，PCA硬件会将PCA计数器寄存器值装入模块的捕获寄存器（CH放入CCAPnH，CL放入CCAPnL）。若模块的CCFn和ECCFn标志置位，会产生一个中断。

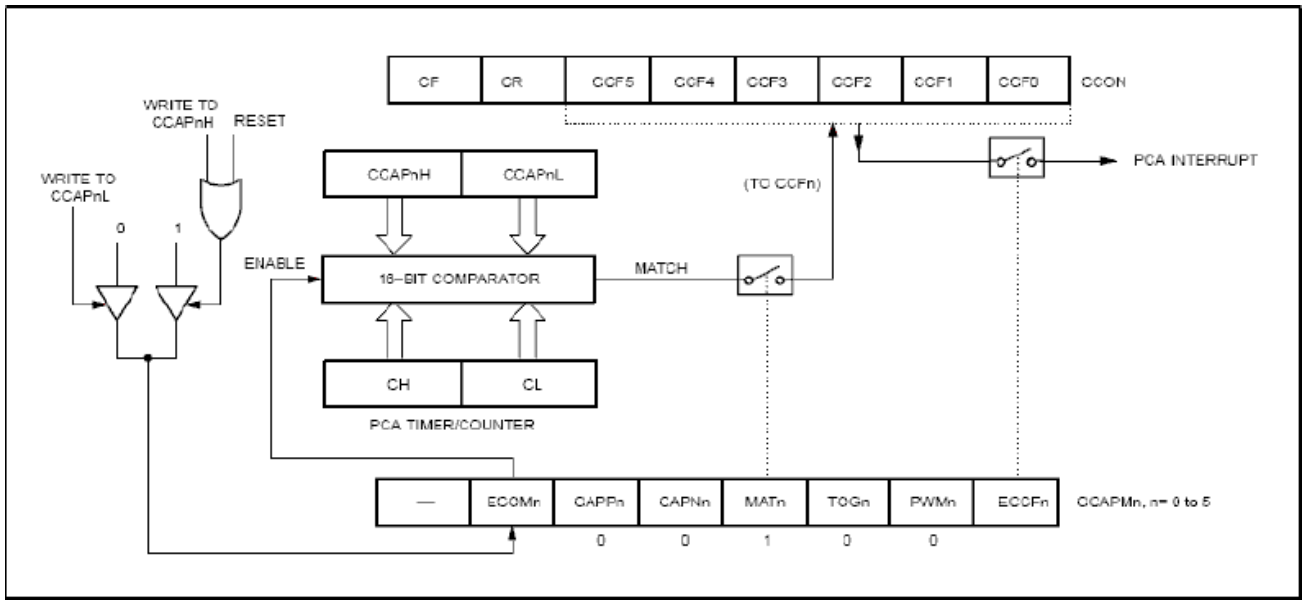
图 14-4. PCA捕获模式



14.4.2 16位软件定时器模式

PCA模块可以通过设置CCAPMn寄存器的ECOM位和MAT位来作为一个软件定时器使用，PCA定时器与模块的捕获寄存器值进行比较，若相等则当CCFn和ECCFn位设置时会产生一个中断信号。

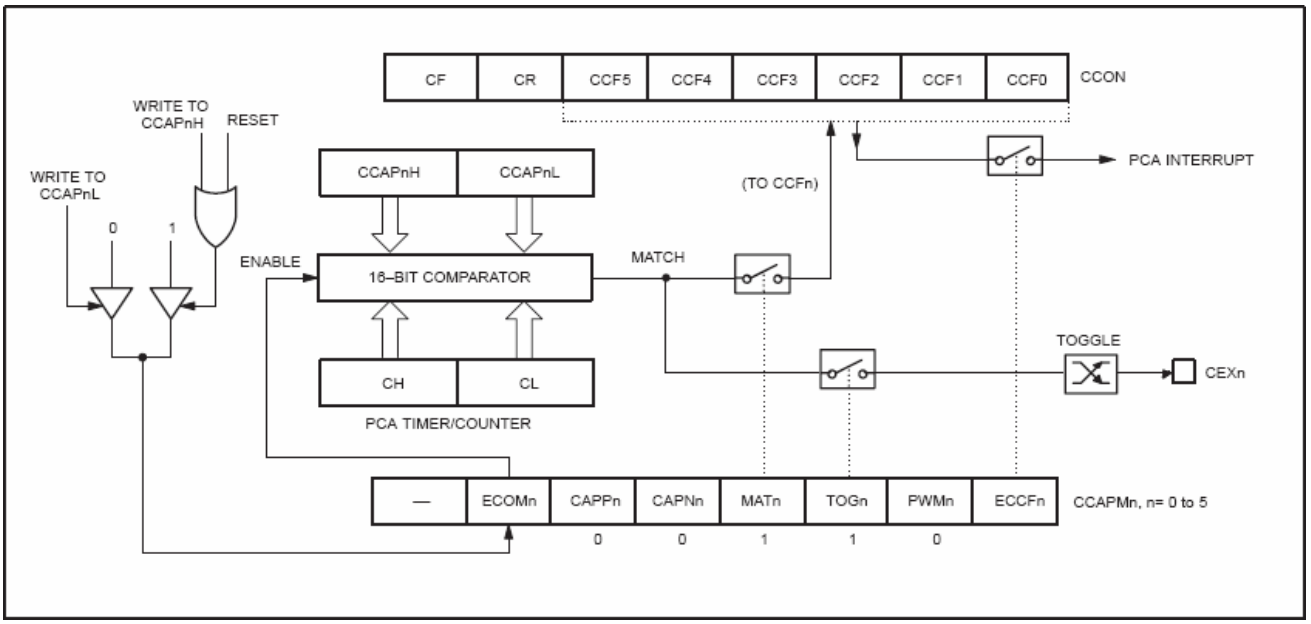
图 14-5. PCA 软件定时器模式



14.4.3 高速输出模式

这种模式下，每当PCA计数器与模块捕获寄存器值相等时，CEX的输出就翻转一次。为激活这种模式，CCAPMn 寄存器的TOG, MAT 和 ECOM 位必须都置为1。

图 14-6. PCA 高速输出模式



14.4.4 PWM 模式

所有PCA模块都可用作PWM输出，输出频率取决于PCA定时器的时钟源，所有的模块都有相同的输出频率，因为它们共享PCA定时器。

占空比取决于模块捕获寄存器CCAPnL 与扩展的第9位ECAPnL的值。当9位数据{0,[CL]}值小于{ ECAPnL,

[CCAPnL] }组成9位数据时，输出低电平，相等或大于时输出高电平。

当CL从0xFF到0x00溢出时，{ ECAPnL, [CCAPnL] } 的值使用{ ECAPnH,[CCAPnH] }的值重载，这样可以允许无异常脉冲的更新PWM。 模块的CCAPMn 寄存器PWMn 和 ECOMn 位必须置位以使能PWM模式。

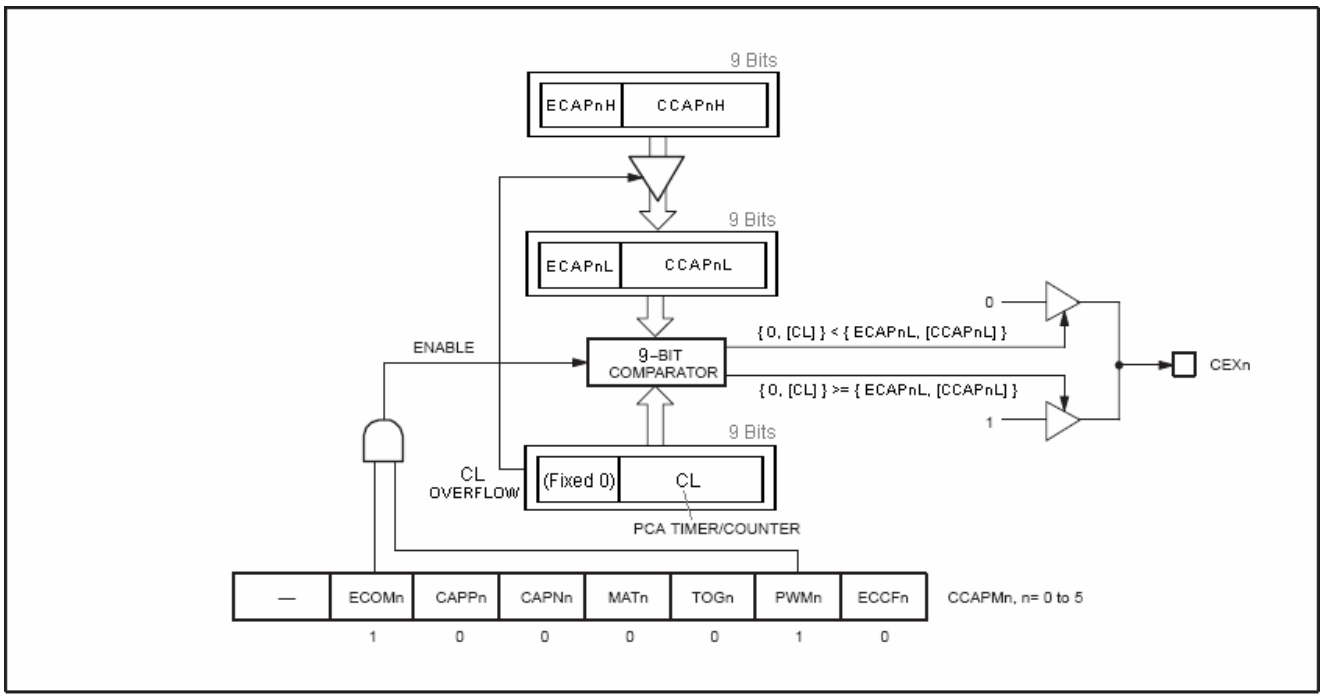
使用9位比较，输出的占空比可以真正实现从0%到100%可调。占空比计算公式如下：

占空比 = 1 – { ECAPnH, [CCAPnH] } / 256.

这里, *[CCAPnH]* 是CCAPnH 寄存器的值， *ECAPnH* (PCAPWMn 寄存器的第1位) 是1位值。所以， *{ ECAPnH, [CCAPnH] }* 组成了9位比较器用的9位值。

例如,
a. 若 ECAPnH=0 且 CCAPnH=0x00 (即9位值, 0x000), 占空比是100%.
b. 若 ECAPnH=0 且 CCAPnH=0x40 (即9位值, 0x040), 占空比是75%.
c. 若 ECAPnH=0 且 CCAPnH=0xC0 (即9位值, 0x0C0), 占空比是25%.
d. 若 ECAPnH=1 且 CCAPnH=0x00 (即9位值, 0x100), 占空比是0%.

图 14-7. PCA PWM 模式



14.5 PCA 示例代码

(1). 功能需求: 设置 PWM2/PWM3 输出占空比 25% 和 75% 的波形

汇编语言代码范例:

```
PWM2        EQU         02h
ECOM2       EQU         40h
PWM3        EQU         02h
ECOM3       EQU         40h


PWM2_PWM3:
    MOV         CCON,#00H               ; 停止 CR
    MOV         CMOD,#02H                   ; PCA 时钟源 = 系统时钟 / 2

    MOV         CCAPM2, #(ECOM2 + PWM2)  ; 使能 PCA 模块 2 (PWM 模式)
    MOV         CCAP2H,#0C0H                ; 占空比为 25%

    MOV         CCAPM3, #(ECOM3 + PWM3)  ; 使能 PCA 模块 3 (PWM 模式)
    MOV         CCAP3H,#40H                 ; 占空比为 75%
    ;
    SETB        CR                              ; 启动 PCA
```

C 语言代码范例:

```
#define     PWM2            EQU     0x02
#define     ECOM2           EQU     0x40
#define     PWM3            EQU     0x02
#define     ECOM3           EQU     0x40

void main(void)
{
    // set PCA
    CCON = 0x00;                                        //禁止 PCA & 清 CCF0, CCF1, CCF2, CCF3,CF 标志
    CMOD = 0x02;                                            // PCA 时钟源 = 系统时钟 / 2

    CCAPM2 |= (ECOM2 | PWM2);                   //使能 PCA 模块 2 (PWM 模式)
    CCAP2H = 0xC0;                          //占空比为 25%

    CCAPM3 |= (ECOM3 | PWM3);                   //使能 PCA 模块 3 (PWM 模式)
    CCAP3H = 0x40;                          //占空比为 75 %
    //---------------------------------------------
    CR = 1;                                                 //启动 PCA

    while (1);
}
```

(2). 功能需求: 设置模块 0 为 CEX0 上升沿捕捉模式，模块 1 为 PWM 输出占空比为 25%的波形

汇编语言代码范例:

```
PWM1        EQU         02h
CAPP0       EQU     20h
ECOM1       EQU         40h

PWM2_PWM3:
    MOV         CCON,#00H           ; 停止 CR
    MOV         CMOD,#02H               ; PCA 时钟源 = 系统时钟 / 2

    ORL     CCAPM0, #CAPP0      ; 模块 0 为捕捉 CEX0 上升沿

    ORL     CCAPM1, #(ECOM1 + PWM1)  ; 模块 1 为 PWM 输出
    MOV         CCAP3H, #0C0h           ; 占空比 25 %
    SETB        CR              ; 启动 PCA
```

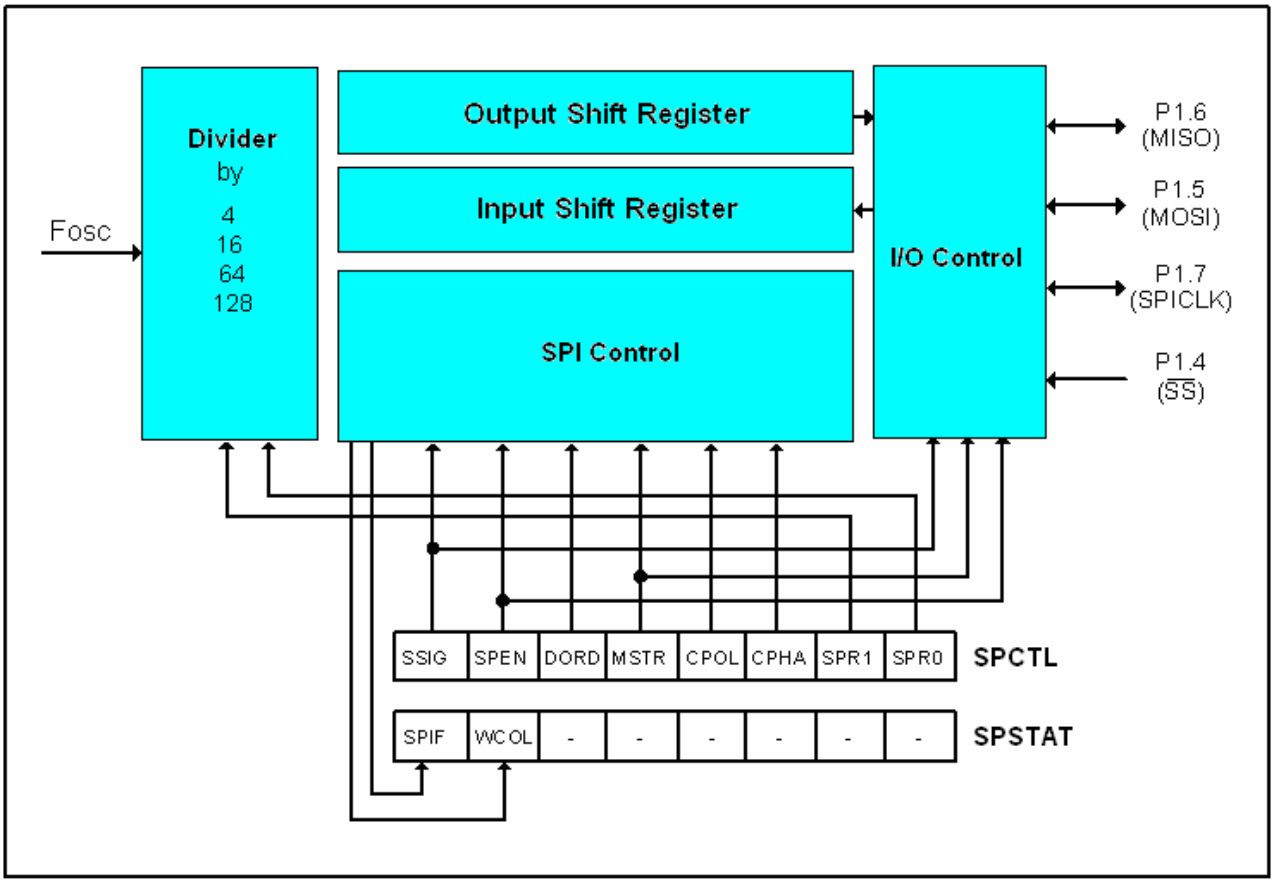
C 语言代码范例:

```
#define     PWM1            EQU     0x02
#define     CAPP0           EQU             0x20
#define     ECOM1           EQU     0x40


void main(void)
{
    // set PCA
    CCON = 0x00;                                    //禁止 PCA & 清 CCF0, CCF1, CCF2,CCF3,CF 标志
    CMOD = 0x02;                                        // PCA 时钟源 = 系统时钟 / 2

            CCAPM0 |= CAPP0;                    //模块 0 为捕捉 CEX0 上升沿

    CCAPM1 |= (ECOM1 | PWM1);               //模块 1 为 PWM 输出
    CCAP3H = 0xC0;                  //占空比 25 %

    CR = 1;                                             //启动 PCA

    while (1);
}
```

15 串行外设接口(SPI)

MPC82G516提供了一个高速串行通讯接口（SPI）。SPI接口是一种全双工、高速同步通讯总线，有两种操作模式：主机模式和从机模式。无论哪种模式，12MHz系统时钟时支持高达3Mbps的通讯速度。MPC82G516的SPI状态寄存器（SPSTAT）有一个传送完成标志（SPIF）和写冲突标志（WCOL）。

图 15-1. SPI 框图



SPI接口有4个引脚： MISO (P1.6), MOSI (P1.5), SPICLK (P1.7) 和/SS (P1.4):

• SPICLK, MOSI 和 MISO 通常将两个或多个SPI设备连接在一起。数据从主机到从机使用MOSI 引脚 (Master Out / Slave In主出从入) ，从从机到主机使用MISO 引脚 (Master In / Slave Out主入从出)。 SPICLK 信号在主机模式时输出，从机模式时输入。若SPI接口禁用，即 SPEN (SPCTL.6) = 0，这些引脚可以作为普通I/O口使用。

• /SS是从机选择端。典型配置中，SPI主机可以使用其某个端口选择某一个SPI设备作为当前从机，一个SPI 从机设备使用它的/SS引脚确定自己是否被选中。下面条件下/SS被忽略:

- 若 SPI系统被禁用，即 SPEN (SPCTL.6) = 0 (复位值).
- 若SPI作为主机运行，即 MSTR (SPCTL.4) = 1, 且 P1.4 (/SS) 被配置成输出。
- 若/SS被设置成忽略，即 SSIG (SPCTL.7) = 1, 这个端口作为普通I/O使用。

注意，即使SPI被配置成主机运行(MSTR=1)，它仍然可以被/SS引脚的低电平拉成从机(若 SSIG=0)，一旦发生这种情况， SPIF 位(SPSTAT.7)置位。 (参见 15.5节：/SS引脚的模式改变)

下面是SPI操作相关寄存器:

SPCTL (地址=85H, SPI控制寄存器，复位值=0000,0100B)

7	6	5	4	3	2	1	0
SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0

SSIG: 忽略/SS
若SSIG=1, MSTR位决定该设备是主机还是从机。
若SSIG=0, /SS引脚决定该设备是主机还是从机。

SPEN: SPI使能
若SPEN=1, SPI功能打开。
若SPEN=0, SPI接口禁用，所有SPI引脚可作为通用I/O口使用。

DORD: SPI 数据顺序
1： 传送数据时先传数据字节最低位。
0： 传送数据时先传数据字节最高位。

MSTR: 主机/从机模式选择

CPOL: SPI时钟极性选择
1: SPICLK待机是为高电平，SPICLK时钟脉冲前沿是下降沿，而后沿是上升沿。
0: SPICLK待机是为低电平，SPICLK时钟脉冲前沿是上升沿，而后沿是下降沿。

CPHA: SPI时钟相位选择
1: SPICLK脉冲前沿放数据，后沿采样。
0: /SS引脚低电平 (SSIG=0)开始放数据并在SPICLK后沿改变数据. 数据在SPICLK的前沿采样。
(注： 若 SSIG=1, CPHA必须不为1，否则就是为定义操作.)

SPR1-SPR0: SPI 时钟速率选择 (主机模式)
00 : Fosc/4
01 : Fosc/16
10 : Fosc/64
11 : Fosc/128 (*Fosc是系统时钟.)*

SPSTAT (地址=84H, SPI状态寄存器, 复位值=00xx,xxxxB)

7	6	5	4	3	2	1	0
SPIF	WCOL	-	-	-	-	-	-

SPIF: SPI 传输完成标志
当一次串行传输完成时，SPIF位置位，同时若SPI中断允许，会产生一个中断。若 /SS 引脚在主机模式下被拉低且SSIG=0，SPIF位也会置位以表明“模式改变”。SPIF标志通过软件在该位写入“1”来清零。

WCOL: SPI 写冲突标志
SPI数据寄存器SPDAT在数据传输过程中被写入时，WCOL置位。(参见 15.6节:写冲突). WCOL标志通过软件在该位写入“1”来清零。

SPDAT (地址=86H, SPI数据寄存器, 复位值=0000,0000B)

7	6	5	4	3	2	1	0
(MSB)							(LSB)

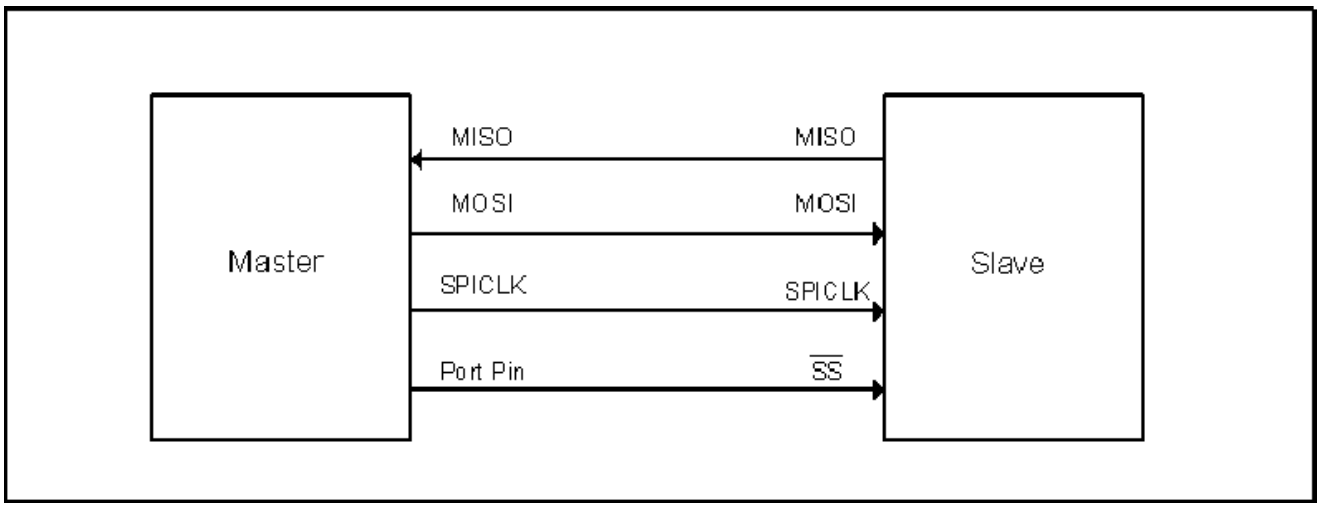
SPDAT有两个物理缓冲器供传输过程中写入和读取，一个写入缓冲器，一个读取缓冲器。

15.1 典型 SPI 配置

15.1.1 单主机和单从机

对于主机: 任何端口，包括P1.4 (/SS)，都可以用来控制从机的/SS片选引脚。
对于从机: SSIG 为 ‘0’， /SS引脚决定该设备是否被选中。

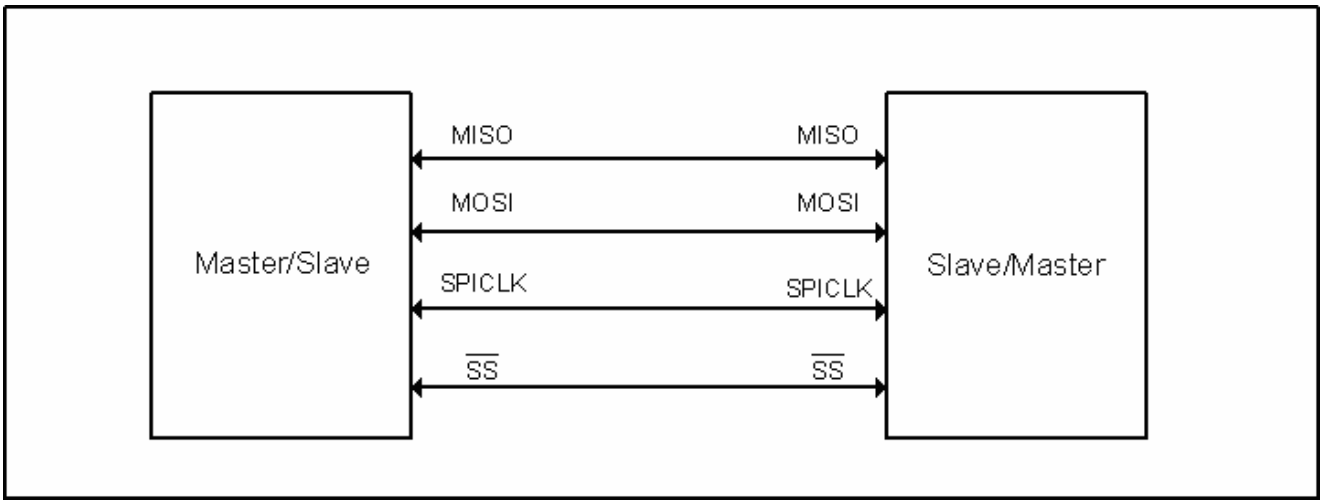
图 15-2. SPI 单主从机配置



15.1.2 双驱动器,可以是主机或从机

两个彼此连接的设备，均可成为主机或从机，没有SPI操作时，都可以被通过设置MSTR=1， SSIG=0, P1.4 (/SS)双向口配置成主机。任何一方要发起传输，它可以配置P1.4位输出并强行拉低，使另一个设备发生“被改成从机模式”事件。(参见 15.5节：/SS引脚模式改变)

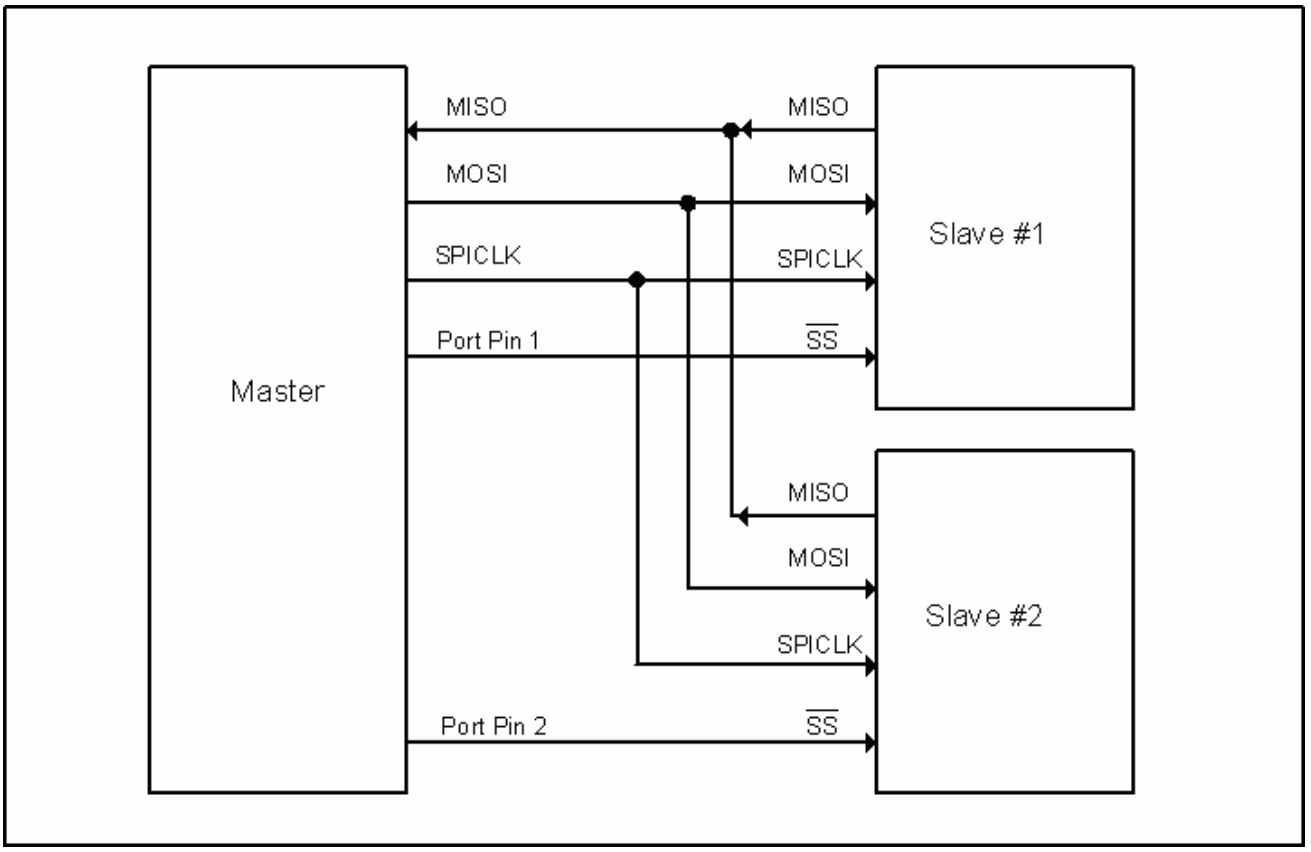
15-3. SPI 双驱动器，可以是主机或从机配置



15.1.3 单主机和多从机

对于主机: 任何端口，包括P1.4 (/SS)，都可以用来控制从机的/SS片选引脚。
对于所有从机: SSIG 为 ‘0’， /SS引脚决定该设备是否被选中。

图 15-4. SPI 单主机和多从机配置



15.2 SPI 配置

表15-1 显示了主/从机模式配置及使用方法和传输方向。

表 15-1. SPI 主从机选择

SPEN (SPCTL.6)	SSIG (SPCTL.7)	/SS 引脚	MSTR (SPCTL.4)	模式	MISO 引脚	MOSI 引脚	SPICLK 引脚	备注
0	X	X	X	SPI 禁用	输入	输入	输入	P1.4~P1.7用作通用I/O
1	0	0	0	从机 (被选中)	输出	输入	输入	被选择为从机
1	0	1	0	从机 (未被选中)	高阻	输入	输入	未被选中
1	0	0	1 → 0	从机 (通过模式改变)	输出	输入	输入	若/SS被拉低，MSTR被硬件自动清‘0’，模式被改为从机
1	0	1	1	主机 (待机)	输入	高阻	高阻	MOSI和SPICLK在主机待机时被置为高阻，以防止总线冲突。
				主机 (活动)		输出	输出	MOSI和SPICLK在主机活动时被上拉。
1	1	X	0	从机	输出	输入	输入	
1	1	X	1	主机	input	输出	输出	

“X” 表示”无需关心”.

15.3 从机注意事项

当CPHA = 0时, SSIG必须为 0 且 /SS 引脚必须在每次串行字节传输前负跳变，传输结束恢复正常高电平。注意 SPDAT寄存器不能在/SS引脚低电平时写入；CPHA = 0, SSIG=1的操作是未定义的。

当CPHA =1时， SSIG可以为0或1。若SSIG=0, /SS引脚可以在每次成功传输之间保持低电平（可以一直拉低）， 这种格式有时非常适合单固定主从机配置应用。

15.4 主机注意事项

SPI通讯中，传输总是有主机发起 。若SPI使能(SPEN=1)并作为主机运行，写入SPI数据寄存器(SPDAT) 数据即可开始SPI时钟生成器和数据传输器，大约半个到1个SPI位时间后写入SPDAT的数据开始出现在MOSI线上。

在开始传输之前，主机通过拉低相应/SS引脚选择一个从机作为当前从机。写入SPDAT寄存器数据从主机MOSI引脚移出，同时从从机MISO移入主机MISO的数据也写入到主机的SPDAT寄存器中。

移出1字节后，SPI时钟发生器停止，置传输完成标志SPIF，若SPI中断使能则生成一个中断。主机CPU和从机CPU中的两个移位寄存器可以看成一个分开的16位环形移位寄存器，数据从主机移到从机同时数据也从从机移到主机。这意味着，在一次传输过程中，主从机数据进行了交换。

15.5 /SS 引脚的模式改变

若 SPEN=1, SSIG=0, MSTR=1 且 /SS pin=1, SPI使能在主机模式，这种情况下，其他主机可以将/SS引脚拉低来选择该设备为从机并开始发送数据过来。为避免总线冲突，该SPI设备成为一个从机，MOSI 和SPICLK引脚被强制为输入端口，MISO成为输出端口，SPSTAT中SPIF标志置位，若此时SPI中断使能，则还会产生一个SPI中断。用户软件必须经常去检查MSTR位，若该位被从机选择清零而用户又想要继续保持该SPI主机模式，用户必须再次设置MSTR位，否则，将处于从机模式。

15.6 数据冲突

SPI在发送方向是单缓冲的，而在接收方向是双缓冲的。发送数据直到上一次数据发送完成后才能写入移位寄存器，数据发送过程中写入数据寄存器就会使WCOL(SPSTAT.6) 置位来表明数据冲突。这种情况下，正在发送的数据继续发送，而刚写入数据寄存器造成冲突的数据就会丢失。

写冲突对于主从机都有可能发生，对于主机，这种现象并不多见，因为主机控制着数据的传送；然而对于从机，由于没有控制权，因此很可能会发生。

对于数据接收，接收的数据被传输到一个并行读数据缓冲器中，以便于移位寄存器再能接收新的字节。然而，接收的数据必须在下个字节完全移入前从数据寄存器SPDAT读出，否则前一个数据就会丢失。

WCOL使用软件向其位写入'1'来清零。

15.7 SPI 时钟频率选择

SPI时钟频率选择（主机模式）使用SPCTL寄存器的SPR1 和SPR0 位来设置，如表15-2所示

表15-2. SPI 串行时钟速率

SPR1	SPR0	SPI时钟速率 @ Fosc=12MHz	Fosc 分频
0	0	3 MHz	4
0	1	750 KHz	16
1	0	187.5 KHz	64
1	1	93.75 KHz	128

这里, Fosc是系统时钟.

15.8 数据模式

时钟相位(CPHA) 位可以让用户设定数据采样和改变时的时钟沿。时钟极性位,CPOL, 可以让用户设定时钟极性。下面图例显示了不同时钟相位、极性设置下SPI通讯时序。

图 15-5. SPI从机传送,CPHA=0

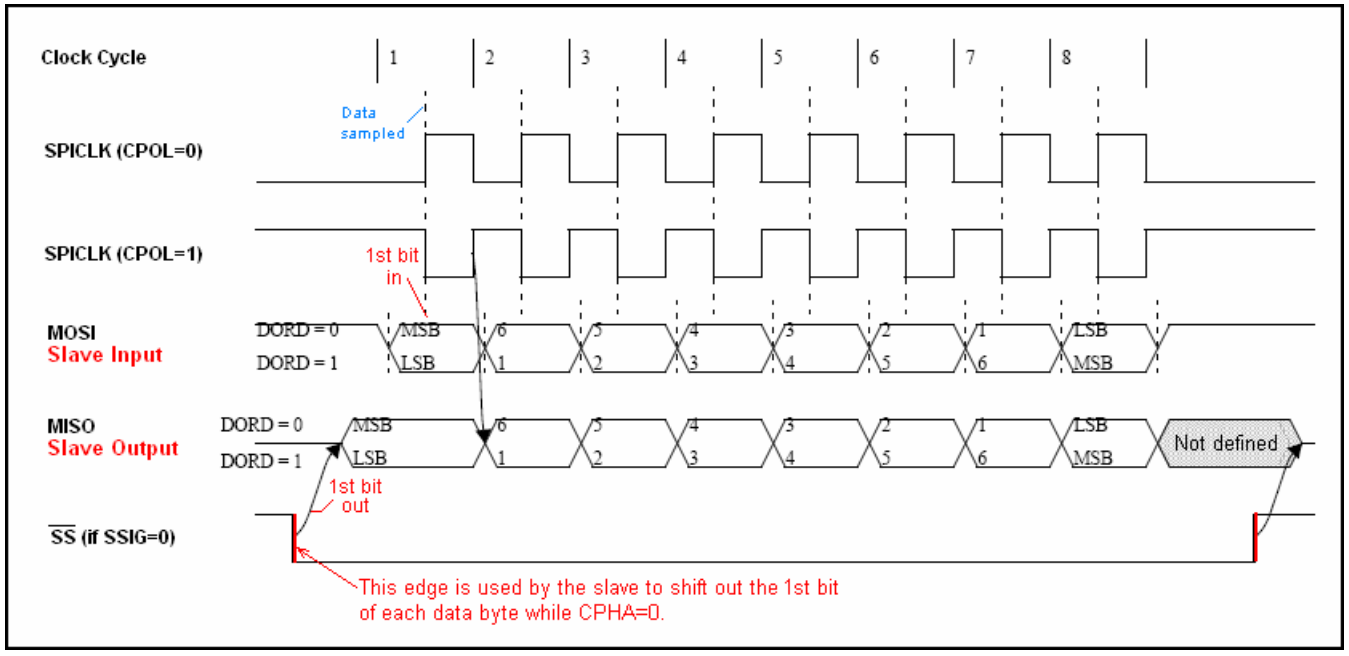


图 15-6. SPI从机传送,CPHA=1

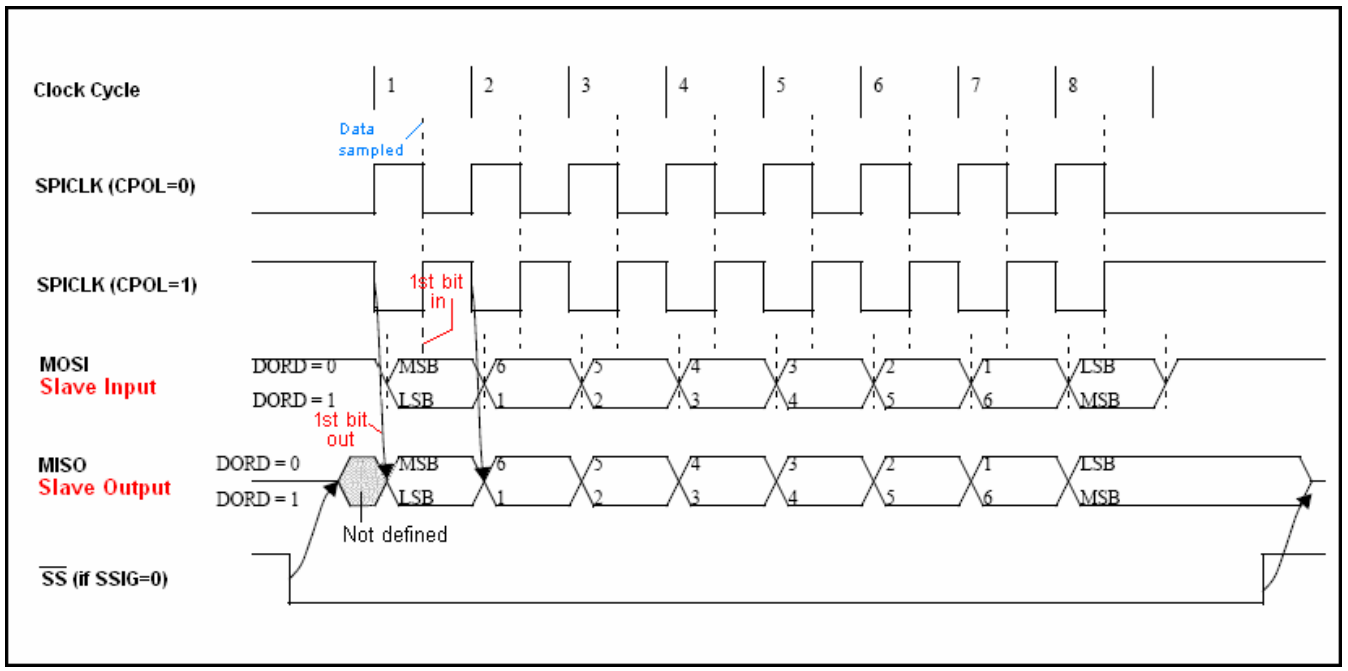


图 15-7. SPI主机传送, CPHA=0

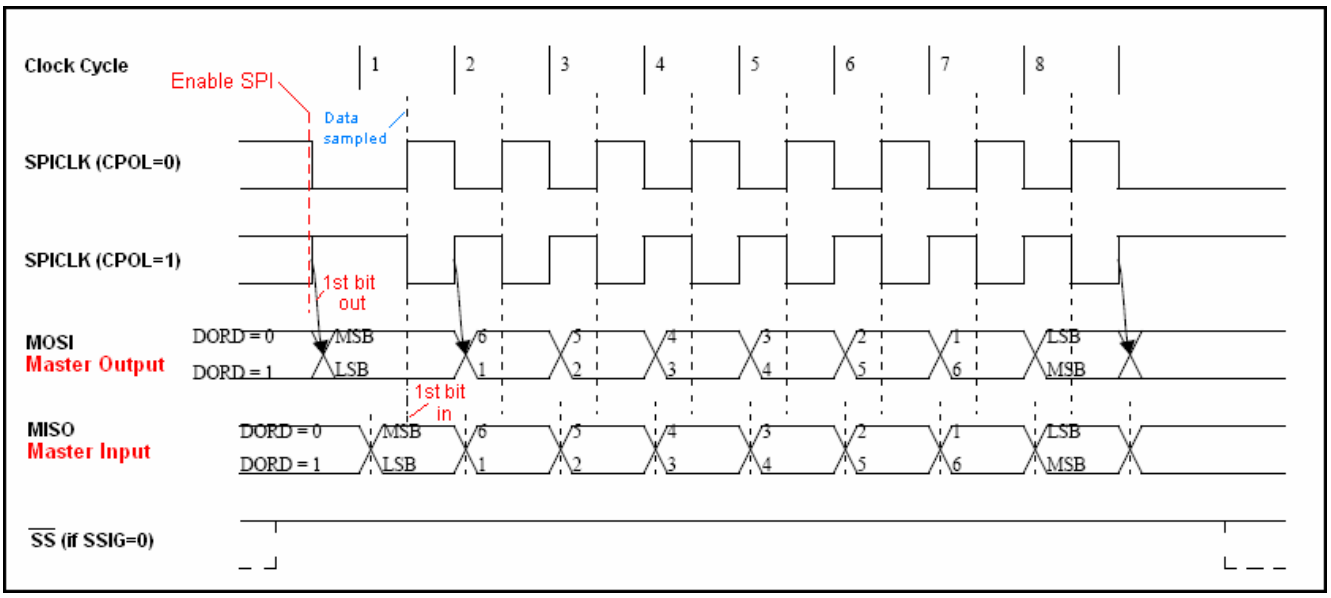
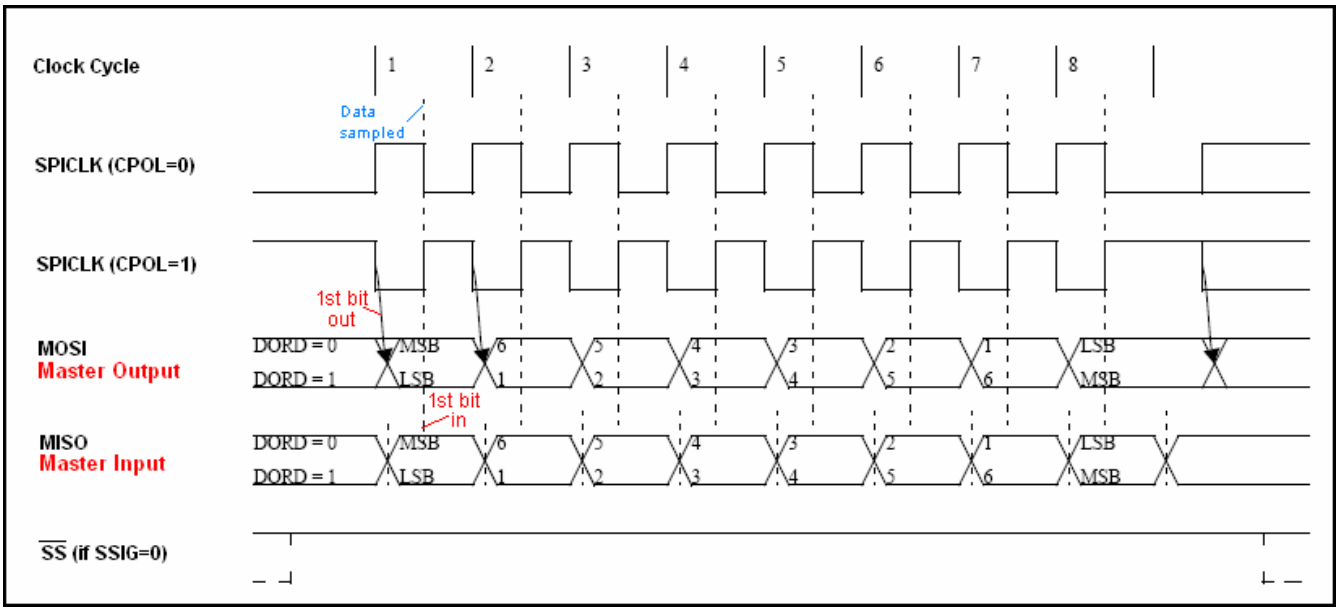


图 15-8. SPI主机传送, CPHA=1



15.9 SPI 示例代码

(1). 功能需求: SPI 主机读和写，采样数据在上升沿并且时钟前沿是上升沿。

汇编语言范例:

```
CPHA        EQU         04h
CPOL        EQU         08h
MSTR        EQU         10h
SPEN        EQU                     40h
SSIG        EQU                     80h
SPIF    EQU         80h

Initial_SPI:                                ;初始化 SPI
    ORL     SPICTL, #(SSIG + SPEN + MSTR)               ;使能 SPI 主模式
    RET

SPI_Write:
    MOV     SPIDAT, R7                      ;写自变量 R7
  wait_write:
    MOV     A, SPISTAT
    JNB     ACC.7, wait_write               ;等待传送完成
    ANL     SPISTAT, #(0FFh - SPIF)             ;清除 SPI 中断标志
    RET

SPI_Read:
    MOV     SPIDAT, #0FFh                       ;触发 SPI 读
  wait_read:
    MOV     A, SPISTAT
    JNB     ACC.7, wait_read                ;等待读完成
    ANL     SPISTAT, #(0FFh - SPIF)                     ;清楚 SPI 中断标志
    MOV     A, SPIDAT                       ;移动读的数据到 A
    RET
```

C 语言范例：

```
#define     CPHA            0x04
#define     CPOL            0x08
#define     MSTR            0x10
#define     SPEN            0x40
#define     SSIG        0x80
#define     SPIF        0x80

void Initial_SPI(void)
{
    SPICTL |= (SSIG | SPEN | MSTR);             // 使能 SPI 主机模式
}

void SPI_Write(unsigned char arg)
{
    SPIDAT = arg;                               //写自变量
    while(!(SPISTAT & SPIF));                   //等待传送完成
    SPISTAT &= ~SPIF;                           //清楚 SPI 中断标志
}

unsigned char SPI_Read(void)
{
    SPIDAT = 0xFF;                          //触发 SPI 读
    while(!SPISTAT & SPIF);                     //等待传送完成
    SPISTAT &= ~SPIF;                           //清除 SPI 中断标志
    return SPIDAT;
}
```

(2). 功能需求： SPI 主机读和写，采样数据在上升沿和时钟前沿为下降沿。

汇编语言范例:

```
CPHA        EQU         04h
CPOL        EQU         08h
MSTR        EQU         10h
SPEN        EQU                 40h
SSIG        EQU             80h
SPIF    EQU         80h

Initial_SPI:                                ;初始化 SPI
    ORL     SPICTL, #(SSIG + SPEN + MSTR + CPOL)                    ;使能 SPI 主机模式
    RET

SPI_Write:
    MOV     SPIDAT, R7                      ;写自变量 R7
  wait_write:
    MOV     A, SPISTAT
    JNB     ACC.7, wait_write               ;等待传送完成
    ANL     SPISTAT, #(0FFh - SPIF)             ;清除 SPI 中断标志
    RET

SPI_Read:
    MOV     SPIDAT, #0FFh                       ;触发 SPI 读
  wait_read:
    MOV     A, SPISTAT
    JNB     ACC.7, wait_read                ;等待读完成
    ANL     SPISTAT, #(0FFh - SPIF)                 ;清除 SPI 中断标志
    MOV     A, SPIDAT                       ;转移读的数据到 A
    RET
```

C 语言范例:

```
#define     CPHA            0x04
#define     CPOL            0x08
#define     MSTR            0x10
#define     SPEN            0x40
#define     SSIG        0x80
#define     SPIF        0x80

void Initial_SPI(void)
{
    SPICTL |= (SSIG | SPEN | MSTR | CPOL);          // 使能 SPI 主机模式
}

void SPI_Write(unsigned char arg)
{
    SPIDAT = arg;                           //写自变量
    while(!(SPISTAT & SPIF));               //等待传送完成
    SPISTAT &= ~SPIF;                       //清除 SPI 中断标志
}

unsigned char SPI_Read(void)
{
    SPIDAT = 0xFF;                      //触发 SPI 读
    while(!SPISTAT & SPIF);                 //等待读完成
    SPISTAT &= ~SPIF;                       //清除 SPI 中断标志
    return SPIDAT;
}
```

(3). 功能需求: SPI 主机读和写,采样数据在下降沿并且时钟前言是上升沿。

汇编语言代码范例:

```
CPHA            EQU             04h
CPOL            EQU             08h
MSTR            EQU             10h
SPEN            EQU                     40h
SSIG            EQU                 80h
SPIF        EQU             80h

Initial_SPI:                                    ; 初始化 SPI
    ORL     SPICTL, #(SSIG + SPEN + MSTR + CPHA)                ; 使能 SPI 主机模式
    RET

SPI_Write:
    MOV     SPIDAT, R7                          ; 写自变量 R7
  wait_write:
    MOV     A, SPISTAT
    JNB     ACC.7, wait_write                   ; 等待传送完成
    ANL      SPISTAT, #(0FFh - SPIF)                ; 清除 SPI 中断标志
    RET

SPI_Read:
    MOV     SPIDAT, #0FFh                           ; 触发 SPI 读
  wait_read:
    MOV     A, SPISTAT
    JNB     ACC.7, wait_read                    ; 等待读完成
    ANL     SPISTAT, #(0FFh - SPIF)                         ; 清除 SPI 中断标志
    MOV      A, SPIDAT                          ;转移读的数据到 A
    RET
```

C 语言代码范例:

```
#define     CPHA            0x04
#define     CPOL            0x08
#define     MSTR            0x10
#define     SPEN            0x40
#define     SSIG        0x80
#define     SPIF        0x80

void Initial_SPI(void)
{
    SPICTL |= (SSIG | SPEN | MSTR | CPHA);          //使能 SPI 主机模式
}

void SPI_Write(unsigned char arg)
{
    SPIDAT = arg;                                   //写自变量
    while(!(SPISTAT & SPIF));                       //等待传送完成
    SPISTAT &= ~SPIF;                               //清除 SPI 中断标志
}

unsigned char SPI_Read(void)
{
    SPIDAT = 0xFF;                              //触发 SPI 读
    while(!SPISTAT & SPIF);                         //等待读完成
    SPISTAT &= ~SPIF;                               //清除 SPI 中断标志
    return SPIDAT;
}
```

(4). 功能需求: SPI 主机读和写,采样数据在下降沿并且时钟前言是下降沿。

汇编语言代码范例:

```
CPHA            EQU             04h
CPOL            EQU             08h
MSTR            EQU             10h
SPEN            EQU                 40h
SSIG            EQU                 80h
SPIF        EQU             80h

Initial_SPI:                                    ;初始化 SPI
    ORL     SPICTL, #(SSIG + SPEN + MSTR + CPOL + CPHA)     ;使能 SPI 主机模式
    RET

SPI_Write:
    MOV     SPIDAT, R7                          ;写自变量 R7
  wait_write:
    MOV     A, SPISTAT
    JNB     ACC.7, wait_write                   ; 等待传送完成
    ANL     SPISTAT, #(0FFh - SPIF)                 ; 清除 SPI 中断标志
    RET

SPI_Read:
    MOV     SPIDAT, #0FFh                           ;触发 SPI 读
  wait_read:
    MOV     A, SPISTAT
    JNB     ACC.7, wait_read                    ; 等待读完成
    ANL     SPISTAT, #(0FFh - SPIF)                             ; 清除 SPI 中断标志
    MOV     A, SPIDAT                           ; 转移读的数据到 A
    RET
```

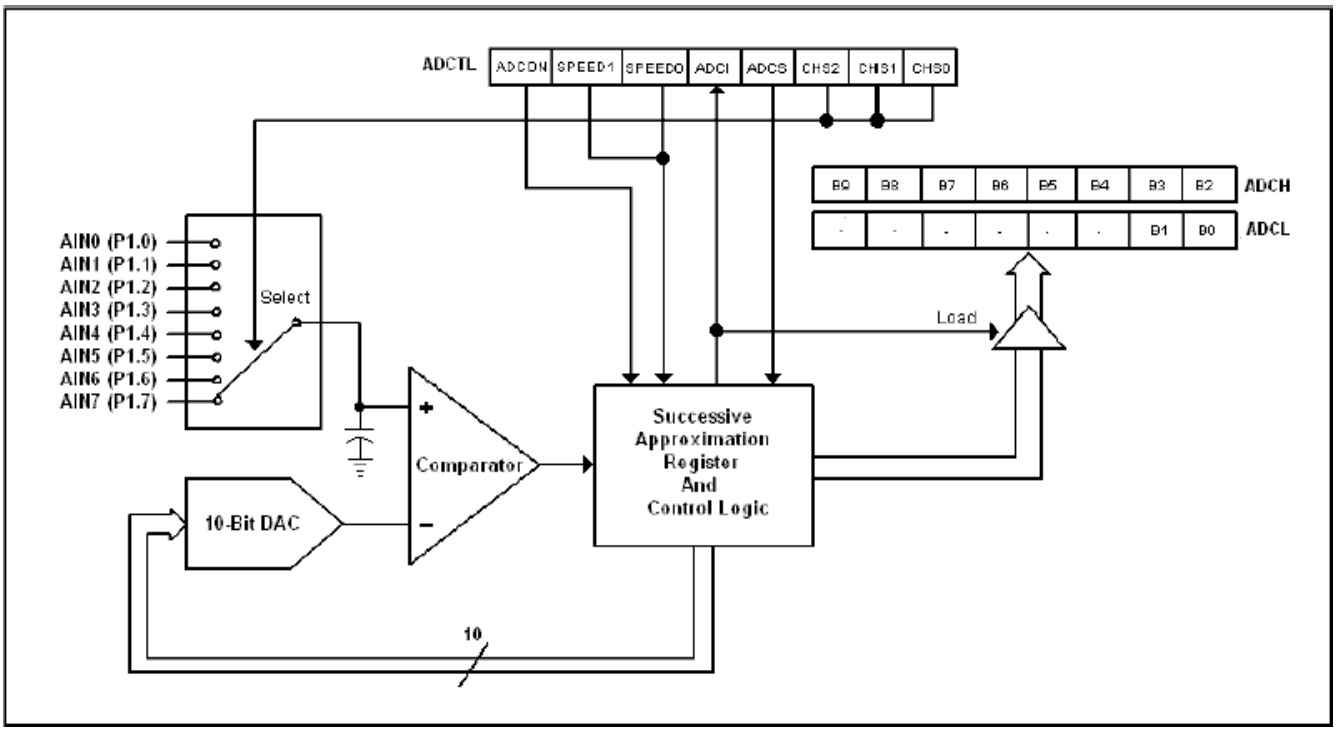
C 语言代码范例:

```
#define     CPHA            0x04
#define     CPOL            0x08
#define     MSTR            0x10
#define     SPEN            0x40
#define     SSIG        0x80
#define     SPIF        0x80

void Initial_SPI(void)
{
    SPICTL |= (SSIG | SPEN | MSTR | CPOL | CPHA);                  //使能 SPI 主机模式
}

void SPI_Write(unsigned char arg)
{
    SPIDAT = arg;                               //写自变量
    while(!(SPISTAT & SPIF));                   //等待传送完成
    SPISTAT &= ~SPIF;                           //清除 SPI 中断标志
}

unsigned char SPI_Read(void)
{
    SPIDAT = 0xFF;                          //触发 SPI 读
    while(!SPISTAT & SPIF);                     //等待读完成
    SPISTAT &= ~SPIF;                           //清除 SPI 中断标志
    return SPIDAT;
}
```

16 模数（A/D）转换器

MPC82G516带有一个10位、8通道逐次逼近型(SAR)模数转换器. 图 16-1 显示了A/D转换器部分框图。 八路模拟输入同P1口共享端口，多路输入带有一个采样保持电路将模拟电压输入到比较器的输入端，比较器的输出连接到SAR进行逐次逼近操作。

图 16-1. ADC功能框图



16.1 ADC 控制寄存器

ADCTL (地址=C5H, ADC控制寄存器, 复位值=0000,0000B)

7	6	5	4	3	2	1	0
ADCON	SPEED1	SPEED0	ADCI	ADCS	CHS2	CHS1	CHS0

ADCON: 0：关闭ADC模块；1：开始ADC模块.

SPEED1 and SPEED0: 模数转换速度选择位.
(0,0): 一次转换需要1080个时钟周期.
(0,1): 一次转换需要540个时钟周期.
(1,0): 一次转换需要360个时钟周期.
(1,1): 一次转换需要270个时钟周期.

注：1时钟周期等于1/Fosc.

ADCS: ADC 启动位.
软件置此位启动A/D转换，转换完成，ADC硬件会自动清除ADCS并设置ADCI.。ADCS不能被软件清零。 ADCS或ADCI为1时将不会开始新的A/D转换。

ADCI: ADC 中断标志.
一次A/D转换完成时该标志置1，若中断允许则还会产生一个中断。该标志必须软件清零。

CHS2, CHS1 and CHS0: 输入通道选择位
(0,0,0): 选择AIN0 (P1.0) 作为模拟输入

(0,0,1): 选择AIN1 (P1.1) 作为模拟输入
(0,1,0): 选择AIN2 (P1.2) 作为模拟输入
(0,1,1): 选择AIN3 (P1.3) 作为模拟输入

(1,0,0): 选择AIN4 (P1.4) 作为模拟输入
(1,0,1): 选择AIN5 (P1.5) 作为模拟输入
(1,1,0): 选择AIN6 (P1.6) 作为模拟输入
(1,1,1): 选择AIN7 (P1.7) 作为模拟输入

AUXR (地址=8EH, 辅助寄存器, 复位值=0000,xx0xB)

7	6	5	4	3	2	1	0
URTS	ADRJ	-	-	-	-	EXTRAM	-

ADRJ:
0: 转换结果高8位存入ADCH[7:0], 低2位存入ADCL[1:0].
1: 转换结果高2位存入ADCH[1:0], 低8位存入ADCL[7:0].

若 ADRJ=0

ADCH (地址=C6H, ADC结果高字节寄存器, 复位值=xxH)

7	6	5	4	3	2	1	0
(B9)	(B8)	(B7)	(B6)	(B5)	(B4)	(B3)	(B2)

ADCL (地址=BEH, ADC结果低字节寄存器, 复位值=xxH)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	(B1)	(B0)

若 ADRJ=1

ADCH (地址=C6H, ADC结果高字节寄存器, 复位值=xxH)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	(B9)	(B8)

ADCL (地址=BEH, ADC结果低字节寄存器, 复位值=xxH)

7	6	5	4	3	2	1	0
(B7)	(B6)	(B5)	(B4)	(B3)	(B2)	(B1)	(B0)

16.2 ADC 操作

基于引脚兼容标准 8051 MCU的考虑, ADC硬件没有独立的内部正参考电压（Vref+）和负参考电压（Vref-）输入引脚. Vref+ 和Vref- 输入芯片内部分别接至VDD和地。所以，满量程Vref+ – Vref-就是VDD.

A/D转换结果可以由下面公式计算:

$$\text{ADC 结果} = 1024 \times \frac{\text{Vin} - \text{Vref-}}{\text{Vref+} - \text{Vref-}} = \frac{\text{AIN}x \text{ 模拟输入电压}}{\text{VDD 电压}}$$

这里, Vin是模拟输入电压，***x*** = 0~7 (AIN0~AIN7的任何引脚).

输入的模拟电压应在 Vref+ 和 Vref-之间, 即VDD 和地之间。对于输入电压介于*Vref- 和 Vref- + 1/2 LSB之间的*, 转换结果是 00,0000,0000B = 000H. 对于输入电压介于 *Vref+ – 3/2 LSB* and *Vref+之间的*, 转换结果是 11,1111,1111B = 3FFH. 这里：

$$1\ \text{LSB} = \frac{\text{Vref+} - \text{Vref-}}{1024} = \frac{\text{VDD}}{1024}$$

在使用ACD功能之前，用户应:

1) 设置ADCON 位启动ADC硬件,
2) 设置SPEED1 和 SPEED0位设定转换速度,
3) 设置CHS2、 CHS1 和CHS0选择输入通道,
4) 设置P1M0和P1M1寄存器将所选引脚设定成只输入模式,
5) 设置ADRJ 位配置ADC转换结果输出形式.

现在，用户就可以置位ADCS来启动AD转换了。转换时间取决于SPEED1 和 SPEED0位的设置。一旦转换结束，硬件自动清除ADCS位，设置中断标志ADCI，并将转换结果按照ADRJ的设置存入ADCH和ADCL。

如上所述，中断标志ADCI，由硬件设置以表明一次转换完成。因此，有两种方法检测AD转换是否完成：
(1) 软件检测ADCI中断标志;
(2) 设置AUXIE寄存器EADC位和IE寄存器EA位使能ADC中断。这样，转换结束就会跳入中断服务进程。
无论(1) 或 (2), ADCI标志都必须在下次转换前用软件清零。

16.3 ADC 注意事项

ADC注意事项如下所列 .

16.4.1 A/D 转换时间

用户可以根据输入的模拟信号频率选择合适的转换速度。例如，若 Fosc=10MHz ，转换速度设为270个时钟周期，则输入的模拟信号频率不应超过37KHz，以保证转换精度。(转换时间 = 1/10MHz x270 = 27us, 所以转换速率 = 1/27us = 37KHz.)

16.4.2 I/O 口用于ADC 转换

用作A/D转换的模拟输入引脚也可以保持其数字I/O输入输出功能。为了获得最佳转换效果，用作ADC的引脚应当禁止其数字输出和输入，可以按照引脚配置一节中的描述将引脚设为只输入模式。

16.4.3 待机和掉电模式

在待机和掉电模式下，ADC将无法使用，若A/D功能打开，它将消耗一部分的电流。因此，为了降低待机和掉电模式下的功耗，可以在进入掉电和待机模式前关闭ADC硬件 (ADCON=0)。

16.4.4 VDD 供电要求

如前所述， Vref+ 和 Vref- 内部分别连接到了 VDD和地，VDD可加载的电压为2.7V~5.5V (5V系统) 或 2.4V~3.6V (3.3V系统)，所以满量程电压Vref+ – Vref- = VDD 就并不固定。然而，计算公式保持不变，这造成同样的Vin却得到不同的转换结果。因此，VDD必须保持不变，这是用户必须要注意的。

既然VDD用作正参考电压Vref+, 用户也必须保证VDD尽可能的纯净，以期得到最佳的ADC性能。

16.5 ADC 示例代码

(1). 功能需求: ADC 示例代码为系统时钟 SYSCLK=24MHz， 模拟输入是 P1.0/P1.1/P1.2 且 SPEED[1:0]=SYSCLK/270 为 88.9KHz 转换率。

汇编语言代码范例:

```
CHS0            EQU             01h
CHS1            EQU             02h
ADCS            EQU             08h
ADCI            EQU             10h
SPEED0          EQU             20h
SPEED1          EQU             40h
ADCON           EQU             80h

INITIAL_ADC_PIN:
        ORL     P1M0, #00000111B                ; P1.0, P1.1, P1.2 =  仅输入模式
        ANL     P1M1,#11111000B

        MOV      ADCTL,#ADCON                           ;  使能  ADC 模块
        ; delay 5us
        ; call ....

Get_P10:
        MOV     ADCTL, #(ADCON + SPEED1 + SPEED0)
                                        ;  使能 ADC 模块和启动转换
                                                                ;  转换速度  114.3k @ 24MHz,  选择  P1.0  为
ADC  输出引脚
        CALL    delay_5us
        ORL     ADCTL, #ADCS

        MOV      A, ADCTL                               ;  检测是否转换完成?
        JNB     ACC.4,$-3
        ANL      ADCTL,#(0FFh - ADCI - ADCS)                    ; clear ADCI & ADCS
        MOV      AIN0_data_V,ADCV                  ;  保存 P1.0 ADC  数据
        ; to do ...

Get_P11:
        MOV      ADCTL,#(ADCON + SPEED1 + SPEED0 + CHS0)   ;  选择 P1.1
        CALL    delay_5us
        ORL     ADCTL, #ADCS

        MOV      A, ADCTL                               ;  检测是否转换完成?
        JNB     ACC.4,$-3
        ANL      ADCTL,# (0FFh - ADCI - ADCS)           ;  清除 ADCI & ADCS
        MOV      AIN1_data_V,ADCV
        ; to do ...

Get_P12:
        MOV      ADCTL,#(ADCON + SPEED1 + SPEED0 + CHS1)   ;  选择 P1.2
        CALL    delay_5us
        ORL     ADCTL, #ADCS

        MOV      ACC,ADCTL                              ;  检测是否转换完成?
        JNB     ACC.4,$-3
        ANL      ADCTL,# (0FFh - ADCI - ADCS)           ;  清除 ADCI & ADCS
        MOV      AIN2_data_V,ADCV

        ; to do ...

        RET
```

C 语言代码范例:

```
#define     CHS0                0x01
```

```
#define     CHS1                0x02
#define     ADCS                0x08
#define     ADCI                0x10
#define     SPEED0              0x20
#define     SPEED1              0x40
#define     ADCON               0x80


void main(void)
{
    unsigned char AIN0_data_V, AIN1_data_V, AIN2_data_V;

    P1M0 |= 0x07;                                   // P1.0, P1.1, P1.2 =  仅输入模式
    P1M1 &= ~0x07;

    ADCTL = ADCON;                          //使能 ADC 模块
    // delay 5us
    // ...

    // select P1.0
    ADCTL = (ADCON | SPEED1 | SPEED0);
                                            //使能 ADC 模块和启动转换
                                            //  转换速度为 114.3k @ 24MHz,  选择  P1.0  为  ADC  输入脚
     Delay_5us();
     ADCTL |= ADCS;

    while ((ADCTL & ADCI) == 0x00);                 //等待完成
    ADCTL &= ~(ADCI | ADCS);
    AIN0_data_V = ADCV;

    // to do ...

    // select P1.1
    ADCTL = (ADCON | SPEED1 | SPEED0 | CHS0);   //  选择 P1.1
     Delay_5us();
     ADCTL |= ADCS;


    while ((ADCTL & ADCI) == 0x00);                 //等待完成
    ADCTL &= ~(ADCI | ADCS);
    AIN1_data_V = ADCV;

    // to do ...

    // select P1.2
    ADCTL = (ADCON | SPEED1 | SPEED0 | CHS1);  //  选择 P1.2
     Delay_5us();
     ADCTL |= ADCS;

    while ((ADCTL & ADCI) == 0x00);                 //等待完成
    ADCTL &= ~(ADCI | ADCS);
    AIN2_data_V = ADCV;

    // to do ...

    while (1);

}
```

17 键盘中断

键盘中断功能主要用于当P2口等于或不等于某个值时产生一个中断，这个功能可以用作总线地址识别或键盘键码识别。

有3个特殊功能寄存器与此功能相关。键盘中断掩码寄存器(KBMASK) 用来定义P2口哪些引脚可以产生中断；键盘模式寄存器(KBPATN)用来定义与P2口值进行比较的值，比较匹配时硬件置键盘中断控制寄存器(KBCON)中的键盘中断标志(KBIF) ，若AUXIE中的EKBI中断允许且EA=1，则还会产生一个中断。键盘中断控制寄存器(KBCON)中的PATN_SEL位用来定义比较是“相等”还是“不等”匹配。

为了使用键盘中断作为“键盘”中断，用户需要设置 KBPATN=0xFF 和 PATN_SEL=0 (不相等)，然后将任意按键连接到KBMASK 寄存器定义的相应P2口，按下时就可以设置中断标志KBIF，并当中断使能时产生中断。这个中断可以将CPU从待机模式或掉电模式下唤醒。这个功能在手持设备，电池供电系统等要求低功耗而且易用的设备上特别有用。

下面是键盘中断操作相关特殊功能寄存器：

KBPATN (地址=D5H，键盘模式寄存器，复位值=1111,1111B)

7	6	5	4	3	2	1	0
KBPATN.7	KBPATN.6	KBPATN.5	KBPATN.4	KBPATN.3	KBPATN.2	KBPATN.1	KBPATN.0

KBPATN.7~0: 键盘模式.

KBCON (地址=D6H，键盘控制寄存器，复位值=xxxx,xx00B)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	PATN_SEL	KBIF

PATN_SEL: 模式匹配极性选择.
1: 键盘输入等于KBPATN中用户定义模式时产生中断。
0: 键盘输入不等于KBPATN中用户定义模式时产生中断。

KBIF:
键盘中断标志.P2端口值匹配KBPATN，KBMASK，PATN_SEL设置条件时置位。需要软件写入‘0’来清零.

KBMASK (地址=D7H，键盘中断掩码寄存器，复位值=0000,0000B)

7	6	5	4	3	2	1	0
KBMASK.7	KBMASK.6	KBMASK.5	KBMASK.4	KBMASK.3	KBMASK.2	KBMASK.1	KBMASK.0

KBMASK.7: 置位时，使能P2.7作为键盘中断源 (KBI7).
KBMASK.6: 置位时，使能P2.6作为键盘中断源 (KBI6).
KBMASK.5: 置位时，使能P2.5作为键盘中断源 (KBI5).
KBMASK.4: 置位时，使能P2.4作为键盘中断源 (KBI4).
KBMASK.3: 置位时，使能P2.3作为键盘中断源 (KBI3).
KBMASK.2: 置位时，使能P2.2作为键盘中断源 (KBI2).
KBMASK.1: 置位时，使能P2.1作为键盘中断源 (KBI1).
KBMASK.0: 置位时，使能P2.0作为键盘中断源 (KBI0).

17.1 键盘中断示例代码

(1). 功能需求: 键盘中断源*输入是 P2.3~P2.0*

汇编语言代码范例:

```
      Jmp       main;
   ORG       0006Bh
KBI_ISR:                                    ;
      PUSH      ACC                                ;

      To do     KBI key check……

      ANL       KBCON,#(0FFh - KBIF)                    ; 清除  KBI  中断标志  (write “0”)
      POP       ACC                              ;
      RETI                                        ;

main:
      ANL       P2M1,#0F0h                        ; P2.3~P2.0 =  仅输入模式
      ORL       P2M0,#00Fh                         ;

      MOV       KBMASK,#00Fh                      ; 使能 P2.3~P2.0 KBI  键盘中断源

      ANL       KBCON,# (0FFh - KBIF)          ; 清除  KBI  中断标志  (write “0”)

      ORL       AUXIE,#EKBI                       ; 使能 KBI  键盘中断
      SETB      EA                                    ; 使能全局中断

       Jmp      $;
```

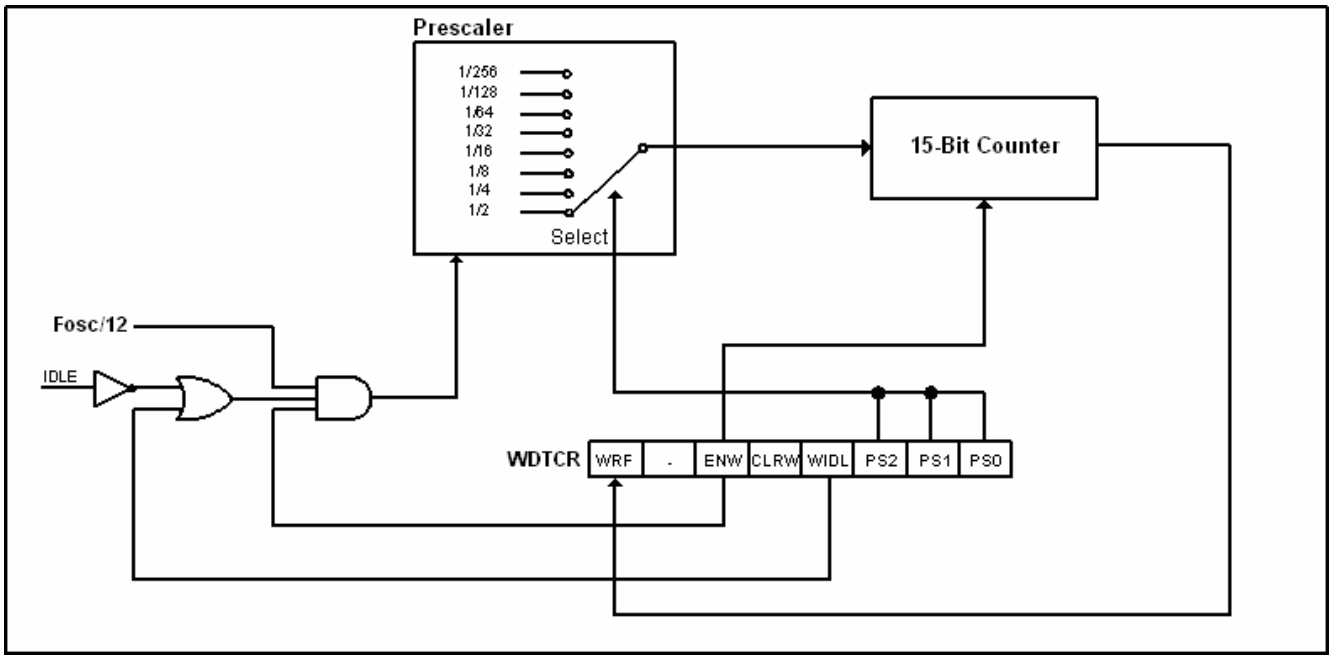
C 语言代码范例：

```
void KBI_ISR(void) interrupt 13
{
      // Do KBI key check
      KBCON &= ~KBIF;                          //清除  KBI  中断标志  (write “0”)
    }
}

void main(void)
{
      P2M1 &= 0xF0;                             // P2.3~P2.0 =  仅输入模式
      P2M0 |= 0x0F;                            //


      KBMASK = 0x0F;                             //使能 P2.3~P2.0 KBI  键盘中断源

      KBCON &= ~KBIF;                            //清除  KBI  中断标志  (write “0”)

      AUXIE |= EKBI;                            //使能 KBI  键盘中断
      EA = 1;                                    //使能全局中断

   While(1);
}
```

18 看门狗定时器（WDT）

看门狗定时器 (WDT) 用来使程序从跑飞或死机状态恢复的一个手段。软件跑飞时，WDT使用系统复位来防止系统执行错误的代码。WDT由一个15位独立定时器、一个8分频器和一个控制寄存器组成，图18-1显示了WDT功能框图。

图 18-1. WDT 功能框图



18.1 WDT 控制寄存器

WDTCR (地址=E1H, WDT控制寄存器，上电复位值=0x00,0000B)

7	6	5	4	3	2	1	0
WRF	-	ENW	CLRW	WIDL	PS2	PS1	PS0

WRF: WDT复位标志。WDT溢出时，这一位被硬件置位，应当软件清零.

ENW: WDT使能位。设置打开WDT。 ***(注:一旦设置，该位就只能上电复位清零了.)***

CLRW: WDT清零位. 该位写“1”会清除15位WDT计数器为0000H. 注意该位本身不需写‘0’清除。

WIDL: WDT在待机模式的运行。置位该位会让WDT在待机模式下继续计数。

PS2~PS0: 分频系数设置.

PS2	PS1	PS0	分频值
0	0	0	2
0	0	1	4
0	1	0	8
0	1	1	16
1	0	0	32
1	0	1	64
1	1	0	128
1	1	1	256

18.2 WDT 操作

WDT上电复位后默认是禁用的。要使能WDT，用户必须设置ENW 位。当WDT使能时，用户需要不断设置CLW位来清除WDT计数器以防止其溢出。15位 WDT 计数器计数到32767 (7FFFH)时溢出，发生溢出会复位器件。WDT使能时，每12个时钟周期加1，这意味着，用户必须至少每32767 x12 系统时钟周期清零WDT计数器一次。

WDT是一次性使能的，所谓“一次性使能”指：一旦通过设置ENW位使能WDT，就没有其它办法来禁用，除非上电复位。WDTCR寄存器也将始终保持最先编程的值，即使在任何复位（硬件复位、软件复位、WDT复位）后，除了上电复位。 例如，若WDTCR值是 0x2D，WDTCR将保持0x2D，而不是每次复位后都是0x00，只有上电复位才能将该值初始成0x00。换句话说，WDT只能上电复位禁用，这就是所谓“一次性使能”WDT。

WDT复位周期由下面公式决定：

2^{15} x 预分频器值 x (12 / Fosc)

表 18-1 显示了运行在6/12/24MHz 下的MCU的WDT溢出周期，这些周期都是用户清除WDT防止复位的最大时间间隔。

表 18-1. WDT 溢出周期

PS2	PS1	PS0	预分频器值	Fosc=6MHz	Fosc=12MHz	Fosc=24MHz
0	0	0	2	131.072 ms	65.536 ms	32.768 ms
0	0	1	4	262.144 ms	131.072 ms	65.536 ms
0	1	0	8	524.288 ms	262.144 ms	131.072 ms
0	1	1	16	1.048 s	524.288 ms	262.144 ms
1	0	0	32	2.097 s	1.048 s	524.288 ms
1	0	1	64	4.194 s	2.097 s	1.048 s
1	1	0	128	8.389 s	4.194 s	2.097 s
1	1	1	256	16.778 s	8.389 s	4.194 s

18.3 WDT 示例代码

条件: WDT 溢出周期= 1.048 秒 @Fosc=12MHz

```
WDTCR_buf DATA  30h                ;声明WDTCR寄存器缓冲
                                   ;(由于 WDTCR是一个只写寄存器)

start:;...
      MOV    WDTCR_buf,#00h  ;初始化WDTCR缓冲
      ORL    WDTCR_buf,#04h  ;PS2=1
      ANL    WDTCR_buf,#0FCh ;PS1=0,PS0=0
      MOV    WDTCR,WDTCR_buf ;写 WDTCR → (PS2,PS1,PS0)=(1,0,0)，预分频器=32
      ORL    WDTCR_buf,#20h   ;ENW=1
      MOV    WDTCR,WDTCR_buf ;写 WDTCR → 使能 WDT
main_loop:
      ORL    WDTCR_buf,#10h     ;CLRW=0
      MOV    WDTCR,WDTCR_buf ;写 WDTCR → 清WDT计数器
      ;...
      ;...
      JMP    main_loop
```

18.4 掉电和待机模式下的 WDT

掉电模式下，晶振停止，WDT也停止了，所以掉电模式下，用户不需要在喂狗了。有3种从掉电模式下恢复的方法：硬件复位，外部中断 (/INT0~/INT3) 或键盘中断，需要在进入掉电模式前确定中断优先级。上电复位，器件所有部分都复位了，WDT也不例外。使用外部中断或键盘中断退出掉电模式，为了防止WDT复位器件，建议在这些中断服务进程中清WDT计数器。

当然，为了确保退出掉电模式后不会立即被WDT复位，更好的办法是进入掉电模式前喂一次狗。

待机模式下，晶振仍然工作着，用户可以设置WIDL位让WDT继续工作或清零WIDL位使WDT在待机模式停止工作。对于待机模式下继续工作的情况，为了防止待机模式下WDT溢出，用户应当使用一个定时器周期性退出待机模式喂狗，喂狗后再进入待机模式。

18.5 WDT 硬件初始化

WDTCR除了软件可以初始化外，也可以使用硬件选项**HWENW**, **HWWIDL** 和 **HWPS[2:0]**在系统上电时由硬件自动初始化，这些硬件选项可以使用通用编程器或烧写器进行编程。（参考 25节：MCU的硬件选项）

若 **HWENW** 编程使能，硬件将会在上电时自动做以下WDTCR初始化动作：

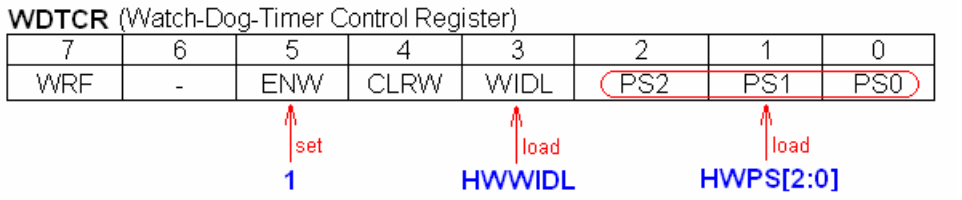
(1) 设置ENW 位.

(2) 装 **HWWIDL** 到WIDL位，并

(3) 装 **HWPS[2:0]** 到 PS[2:0] 位.

WDTCR (Watch-Dog-Timer Control Register)

7	6	5	4	3	2	1	0
WRF	-	ENW	CLRW	WIDL	PS2	PS1	PS0



18.6 WDT 示例代码

(1)功能需求: 使能 WDT 并且选择 WDT 预分频为 1/32

汇编语言代码范例:

```
PS0             EQU     01h
PS1             EQU     02h
PS2             EQU     04h
WIDL                EQU     08h
CLRW                        EQU     10h
ENW             EQU     20h
WRF             EQU     80h

    ANL     WDTCR,#(0FFh - WRF)                 ; 清除 WRF 标志(写“0”)
    MOV     WDTCR,#(ENW + CLRW + PS2)   ; 使能 WDT 并且选择 WDT 预分频为 1/32
```

C 语言代码范例:

```
#define     PS0     0x01
#define     PS1     0x02
#define     PS2     0x04
#define     WIDL        0x08
#define     CLRW                0x10
#define     ENW     0x20
#define     WRF     0x80

    WDTCR &= ~WRF;                                  //清除 WRF 标志(写“0”)
    WDTCR = (ENW | CLRW | PS2);             //使能 WDT 并且选择 WDT 预分频为 1/32
                                                    // PS[2:0]   | WDT  预分频器选项
                                                    //      0      | 1/2
                                                    //      1      | 1/4
                                                    //      2      | 1/8
                                                    //      3      | 1/16
                                                    //      4      | 1/32
                                                    //      5      | 1/64
                                                    //      6      | 1/128
                                                    //      7      | 1/256
```

19 中断系统

MPC82G516 有四级中断优先级的14个中断源. 通过几个寄存器来设置4级中断。 通过 IE, IP, IPH, AUXIE, AUXIP, AUXIPH, XICON和TCON. 使用 IPH (中断优先级高位)和 AUXIPH (辅助中断优先级高位) 寄存器来设置四级中断优先级. 四级中断优先级结构为中断源的应用提供了很大的便利.

19.1 中断源

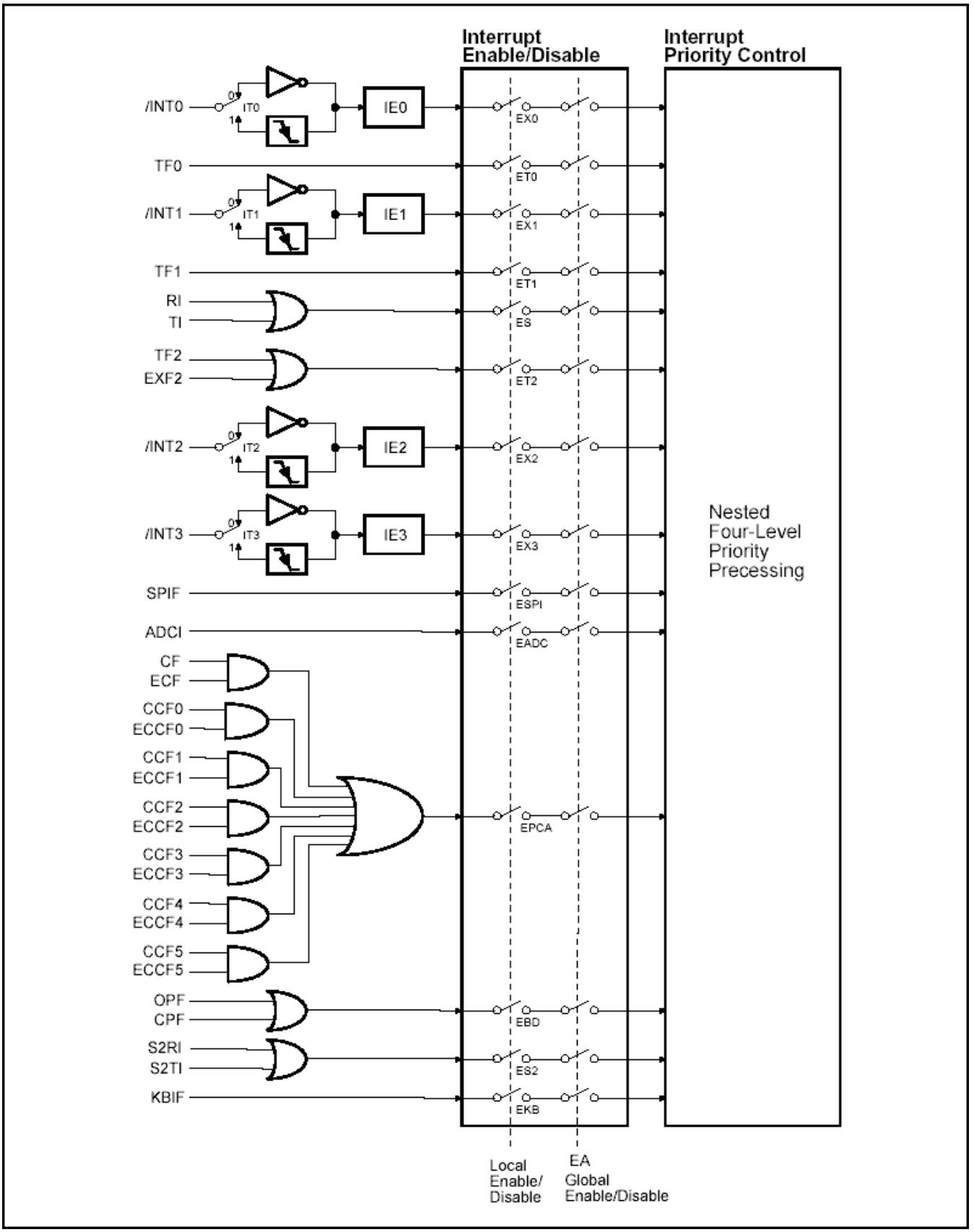
表 19-1列出了所有的中断源. 使能位被允许，中断请求时硬件会产生一个中断请求标志. 当然,总中断使能位 EA (IE 寄存器)必须使能. 中断请求位能由软件置1或清零，这和硬件置1或清零结果相同. 同理，中断可以由软件产生或取消. 中断优先级位决定每个中断产生的优先级.多个中断同时产生时依照中断优先级顺序处理.中断向量地址表示中断服务程序的入口地址.

图 19-1 对所有中断系统进行综述.

表 19-1. 中断源

序号	名称	使能	中断请求	优先位	优先级	地址
#1	外部中断 INT0	EX0	IE0	PX0H, PX0	(高优先级)	0003H
#2	Timer 0	ET0	TF0	PT0H, PT0	.	000BH
#3	外部中断, INT1	EX1	IE1	PX1H, PX1	.	0013H
#4	Timer 1	ET1	TF1	PT1H, PT1	.	001BH
#5	串口中断	ES	RI, TI	PSH, PS	.	0023H
#6	Timer 2	ET2	TF2, EXF2	PT2H, PT2	.	002BH
#7	外部中断, INT2	EX2	IE2	PX2H, PX2	.	0033H
#8	外部中断, INT3	EX3	IE3	PX3H, PX3	.	003BH
#9	SPI	ESPI	SPIF	PSPIH, PSPI	.	0043H
#10	ADC	EADC	ADCI	PADCH, PADC	.	004BH
#11	PCA	EPCA	CF, CCFn (n=0~5)	PPCAH, PPCA	.	0053H
#12	欠压检测	EBD	OPF, CPF	PBDH, PBD	.	005BH
#13	UART2	ES2	S2RI, S2TI	PS2H, PS2	.	0063H
#14	键盘中断KBI	EKB	KBIF	PKBH, PKB	(低优先级)	006BH

图 19-1. 中断系统


Interrupt
Enable/Disable
Interrupt
Priority Control
/INT0
IT0
IE0
EX0
TF0
ET0
/INT1
IT1
IE1
EX1
TF1
ET1
RI
TI
ES
TF2
EXF2
ET2
/INT2
IT2
IE2
EX2
/INT3
IT3
IE3
EX3
Nested
Four-Level
Priority
Precessing
SPIF
ESPI
ADCI
EADC
CF
ECF
CCF0
ECCF0
CCF1
ECCF1
CCF2
ECCF2
CCF3
ECCF3
CCF4
ECCF4
CCF5
ECCF5
EPCA
OPF
CPF
EBD
S2RI
S2TI
ES2
KBIF
EKB
Local
Enable/
Disable
EA
Global
Enable/Disable

19.2 与中断相关的寄存器

下面是中断过程中的相关特殊功能寄存器

IE (地址A8H，中断使能寄存器，复位值=0x00,0000B)

7	6	5	4	3	2	1	0
EA	-	ET2	ES	ET1	EX1	ET0	EX0

EA: 总中断使能位. EA = 0，禁止所有中断. EA = 1，使能所有中断

ET2: 定时器2 中断使能.

ES: 串口 中断使能

ET1: 定时器1 中断使能

EX1: 外部中断1 使能.

ET0: 定时器0 中断使能

EX0: 外部中断0 使能.

IP (地址B8H,中断优先级寄存器，复位值=xx00,0000B)

7	6	5	4	3	2	1	0
-	-	PT2	PS	PT1	PX1	PT0	PX0

PT2: 定时器2 中断优先级位.

PS: 串口 中断优先级位.

PT1: 定时器1 中断优先级位.

PX1: 外部中断1 优先级位.

PT0: 定时器0 中断优先级位.

PX0: 外部中断0 优先级位.

IPH (地址B7H，中断 优先级高位寄存器，复位值=0000,0000B)

7	6	5	4	3	2	1	0
PX3H	PX2H	PT2H	PSH	PT1H	PX1H	PT0H	PX0H

PX3H: 外部中断3 优先级位, H.

PX2H: 外部中断2 优先级位, H.

PT2H: 定时器2 中断优先级位, H.

PSH: 串口 中断优先级位, H.

PT1H: 定时器1 中断优先级位, H.

PX1H: 外部中断1 优先级位, H.

PT0H: 定时器0 中断优先级位, H.

PX0H: 外部中断0 优先级位, H.

AUXIE (地址ADH，辅助中断使能寄存器，复位值=xx00,0000B)

7	6	5	4	3	2	1	0
-	-	EKB	ES2	EBD	EPCA	EADC	ESPI

EKB: 键盘 中断使能位.

ES2: UART2 中断使能位.

EBD: 欠压检测 中断使能位.

EPCA: PCA 中断使能位.

EADC: ADC 中断使能位.

ESPI: SPI 中断使能位.

AUXIP (地址AEH,辅助中断优先级寄存器，复位值=xx00,0000B)

7	6	5	4	3	2	1	0
-	-	PKB	PS2	PBD	PPCA	PADC	PSPI

PKB: 键盘 中断优先级位.

PS2: UART2 中断优先级位.

PBD: 欠压检测 中断优先级位.

PPCA: PCA 中断优先级位.

PADC: ADC 中断优先级位.

PSPI: SPI 中断优先级位.

AUXIPH (地址AFH，辅助中断优先级高位寄存器，复位值=xx00,0000B)

7	6	5	4	3	2	1	0
-	-	PKBH	PS2H	PBDH	PPCAH	PADCH	PSPIH

PKBH: 键盘 中断优先级位, H

PS2H: UART2 中断优先级位,H

PBDH: 欠压监测 中断优先级位,H.

PPCAH: PCA 中断1 优先级位, H.

PADCH: ADC 中断 优先级位,H.

PSPIH: SPI 中断0 优先级位, H.

XICON (地址C0H，外部中断 控制寄存器，复位值=0000,0000B)

7	6	5	4	3	2	1	0
PX3	EX3	IE3	IT3	PX2	EX2	IE2	IT2

PX3: 外部中断3 优先级位.

EX3: 外部中断3 使能位.

IE3: 外部中断3 中断标志.

IT3: 外部中断3 类型控制位. 1: 边沿触发; 0: 电平触发.

PX2: 外部中断2 优先级位.

EX2: 外部中断2 使能位.

IE2: 外部中断2 中断标志.

IT2: 外部中断2 类型控制位. 1: 边沿触发; 0: 电平触发.

TCON (地址88H，定时/计数器 控制寄存器，复位值=0000,0000B)

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

IE1: 外部中断1 请求标志. 外部中断1 由边沿或电平触发（由IT1设置）硬件置标志.

IT1: 外部中断1 类型控制位. 软件选择下降沿/低电平触发外部中断1.

IE0: 外部中断0 请求标志. 外部中断0 由边沿或电平触发（由IT0设置）硬件置标志.

IT0: 外部中断0 类型控制位. 软件选择下降沿/低电平触发外部中断0.

19.3 中断使能

通过设置寄存器 IE, AUXIE 和 XICON能对每个中断进行使能和禁止操作.需注意IE中有个总中断允许位EA.EA 置‘1’所有中断的使能和禁止由单独的设置位决定. EA被清‘0’, 所有中断被禁止.

19.4 中断优先级

中断优先级设置和80C51相同, 除了有四级中断优先级比80C51多两级.优先级位(参见 表 19-1) 决定每个中断的优先级.
表 19-2, 外部中断0 中断优先级设置, 通过不同的设置组合决定中断优先级.

表 19-2. 外部中断0 的四级优先级

优先级位		优先级
PX0H	PX0	
0	0	级别0 (低)
0	1	级别1
1	0	级别2
1	1	级别3 (高)

如果两个优先级的中断同时产生，高优先级的中断总是优先处理.低优先级的中断在处理过程中可以被高优先级的中断打断，等高优先级的中断处理完后低优先级的中断才能从被打断处继续执行。

同等优先级的中断同时产生，执行顺序由中断向量决定顺序执行，中断向量越小的优先级越高.

19.5 中断响应

每个机器周期都会采样中断标志位. 如果没有下列阻止条件. 前一个指令周期中产生中断标志位置位, 接下来的指令周期中将以LCALL调用中断服务程序,下列几种情况将LCALL指令锁定:

1. 另一个高优先级的中断正在处理.
2. 当前机器周期不是正在执行的指令的最后一个机器周期.
3. 正在执行指令RETI 或正在写和中断相关的寄存器(IE或IP寄存器).

上述三种情况将锁定LCALL指令使之暂时不能访问中断服务程序. 第二种情况在引导任何中断服务程序之前必须执行完. 第三种情况保证RETI的执行或写中断相关的寄存器能顺利完成, 在中断进入引导程序之前至少运行一个以上的指令周期.

每个指令周期执行一次采样, 采样结果在下一个指令周期被执行.如果电平触发的外部中断的中断标志被置位，在下一个中断来临之前未处理，则处理当前中断，前一中断被忽略,每个采样周期都采样的当前的中断，如果不及时处理则被忽略.

处理器硬件产生LCALL来调用中断服务程序来响应一个中断. 中断在下一次同种中断来临之前必须被处理，否则将会被覆盖，但不会被其他中断影响.定时器2, 串口, PCA, 电源监测和UART2产生的中断标志必须由软件来清除. 清除中断标志 (IE0, IE1, IE2 or IE3) 后中断源被再次激活. 硬件执行LCALL将当前PC值推入堆栈 (但是不保存PSW状态)将中断服务程序的向量地址装载到PC中, 如表 19-1所示.

中断处理程序以RETI指令结束. RETI提示处理器中断服务程序已经结束, 然后从堆栈中将断点的地址装载到程序计数器(PC)中. 从断点处继续执行.

注意 指令RET也可以结束中断, 虽然可以从中断返回但中断控制系统会认为中断还在继续.这意味着在多次中断后堆栈将溢出.注意:中断向量开始的空间只有8字节.这意味着如果一个中断服务程序超过7个字节以上,就必须使用跳转到其他空间以便不覆盖其他中断的入口地址.

19.6 外部中断

外部中断源包括 /INT0, /INT1, /INT2 和 /INT3, 外部中断能被任意的低电平触发或下降沿触发, 通过设置寄存器 TCON 和 XICON中的位 IT0, IT1, IT2和 IT3. ITx= 0, INTx管脚被拉低将触发中断. ITx = 1,INTx管脚上的下降沿将触发中断. 中断被触发后将置位TCON、XICON中的IE0、IE1、IE2 、IE3. 如果中断被激活这些标志在进入中断服务程序后被硬件清除. 如果中断是电平触发, 外中断源必须有效,直到中断被响应,在执行完中断服务程序前,此有效中断必须被清除.

每个指令周期都采样外部中断引脚, 输入的高或低将保持至少一个振荡周期以确保能被采样到. 如果外部中断有效, 外部中断引脚上的信号高必须保持至少一个周期, 接下来的低也必须至少保持一个周期, 以确保中断能响应置标志IEx.执行中断服务程序后IEx标志将被CPU硬件清除.

如果外部中断是电平触发, 外中断源必须有效,直到中断被响应,在执行完中断服务程序前,此有效中断必须被清除. 否则将产生下一次中断.

19.7 单步运行

80C51的中断结构允许使用很少的软件开销来实现单步运行.如前所述,在正在执行相同或更高优先级的中断的时候中断请求是不会被响应的,直到运行完当前中断服务程序后至少再执行一条以上的指令.因而, 如果一个中断服务程序正在执行,这不能被其他中断重入, 直到当前中断服务程序被执行完后至少再执行一条以上的指令. 下面是一个电平触发的外部中断(e.g., INT0)的使用实例.代码如下:

```
JNB    P3.2,$   ;等待直到INT0变高
JB     P3.2,$   ;等待直到INT0 变低
RETI            ;服务程序结束并执行一条指令
```

当前INT0引脚(P3.2)保持低状态,CPU进入外部中断0服务程序等待INT0的引脚上有脉冲产生(从低到高再到低).然后执行RETI指令, 跳回被中断的程序执行一条指令,然后再次进入外部中断 0 中断服务程序等待引脚P3.2上再次产生一个脉冲.这样就是在引脚P3.2上每产生一个脉冲程序步进一条指令,这样就实现了单步操作.

19.8 中断示例代码

(1). 功能需求: 在掉电模式下设置 INT0 高电平唤醒 MCU

汇编语言代码范例:

```
PX0             EQU     01h
PX0H            EQU     01h
PD              EQU     02h

        ORG     0000h
        JMP     main

        ORG     00003h
ext_int0_isr:
        to do.....
        RETI

main:

        SETB    P3.2                    ;

        ORL     IP,#PX0                 ; 选择 INT0 中断优先级
        ORL     IPH,#PX0H               ;

        JNB     P3.2, $                 ; 确认 P3.2 输入高

        SETB    EX0                     ; 使能 INT0 中断
        CLR     IE0                     ; 清除 INT0 标志
        SETB    EA                      ; 使能全局中断

        ORL     PCON,#PD                ; 设置 MCU 进入掉电模式

        JMP     $
```

C 语言代码范例:

```
#define     PX0             0x01
#define     PX0H            0x01
#define     PD              0x02

void ext_int0_isr(void) interrupt 0
{
    To do……
}

void main(void)
{
    P32 = 1;

    IP |= PX0;                  //选择 INT0 中断优先级
    IPH |= PX0H;

    while(!P32);                //确认 P3.2 输入高

    EX0 = 1;                    //使能 INT0 中断
    IE0 = 0;                    //清除 INT0 标
    EA = 1;                     //使能全局中断

    PCON |= PD;                 //设置 MCU 进入掉电模式

    while(1);
```

20 ISP & IAP

MPC82G516可以使用下面的方法对内部FLASH进行编程。

(1) 传统的并行编程方法: 通用编程器 (此处不再累述).
(2) 在系统编程方式(ISP): 使用内部引导程序来编程.

如图 20-1所示MPC82G516的Flash结构,MPC82G516的FLASH 被划分成 AP-部分, IAP-部分和ISP-部分. AP-部分装载用户服务程序; IAP- 部分装载非遗失数据,ISP-空间装载在系统编程的引导程序.

ISP programmer 能编程FLASH的一些确切的空间; 例如 AP-空间和IAP-空间.
表 20-1 罗列了上述操作方法的不同之处.
表 20-1. 不同的编程模式对照

分类	并行编程模式	ISP	IAP
擦除/编程/校验	Yes	Yes	Yes
编程 区域	所有Flash & MCU's 硬件选项	AP-部分 & IAP-部分	IAP-部分
硬件或软件限制?	硬件限制	软件限制	软件限制
编程介面	并口	串行方式使用 DTA (P3.1)	None
编程前的预处 理	无	预编程引导程序 & HWBS enabled	None
编程 工具	通用编程器或 "Megawin 8051 Writer"	"Megawin ISP 编程器"	None

为什么要ISP?

ISP使用户无需将MCU从应用系统中取下修改程序(AP-空间)和非易失性数据(IAP-空间)成为可能. 这个功能非常实用是在应用现场改写应用程序成为可能. (注意 ISP 功能引导预编程在ISP-空间中的程序)

为什么要IAP?

IAP-空间能保存应用系统中需要保存的实时数据,掉电后不易失,所以可以替代串行存储器比如93C46或24C01等.

20.1 Flash 存储器

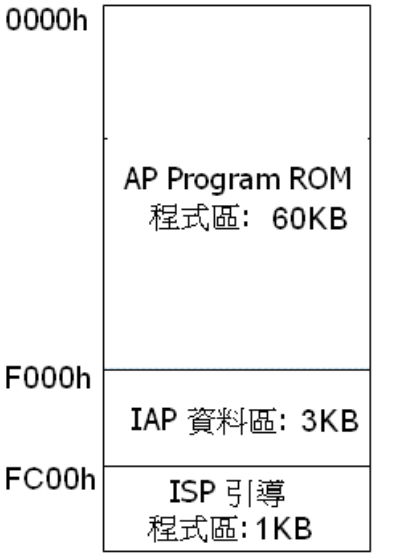
20.1.1 Flash 特点

1. Flash只能页擦除，不能字节擦除.并且页擦除后数据全部变成0xFF.
2. 字节 0xFF能被编程成非0xFF字节．任何非-0xFF字节不能编程成0xFF字节除非使用页擦除.
3. 每页有512字节，页地址总是位于0x0200*N, N (=0,1,2,3,..) 意思为第N个页.
4. 擦写次数: 20,000 擦写次数.

20.1.2 Flash 结构

图 20-1 显示MPC82G516的FLASH结构. Flash被划分成AP-区域, IAP-区域和ISP-区域. AP-空间用来存储用户应用程序; IAP-空间用于保存非易失性数据; ISP-空间用来保存在系统编程的引导程序. Flash 总容量为 64K 字节, IAP-空间為3KB, ISP空间為1KB.

图 20-1. Flash 结构



MPC82G516 程式區與
IAP, ISP 區域分配說明

20.2 ISP 操作

一般来说，用户无需知道 ISP的具体操作因为 Megawin 提供了ISP标准工具(参见20.2.5节)．用户如果想设计自己的ISP 操作,本节对 ISP操作的具体寄存器做详细解说．

20.2.1 ISP 寄存器

以下是有关 ISP 操作的特殊寄存器．用户程序可以访问这些寄存器．

ISPCR (地址E7H, ISP 控制寄存器，复位值=000x,x000B)

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
ISPEN	SWBS	SWRST	-	-	CKS2	CKS1	CKS0

ISPEN: 使能ISP 功能．

SWBS: 软件引导选择．设置此位在软件复位后选择是从ISP-空间/AP-空间引导．

SWRST: 对此位写‘1’触发软件复位．

CKS2~CKS0: ISP 操作中的延时依照不同的振荡频率而不同，参见表 20-2.

表 20-2. ISP 延时设置

CKS2	CKS1	CKS0	振荡器频率 (MHz)
0	0	0	> 24
0	0	1	20 ~ 24
0	1	0	12 ~ 20
0	1	1	6~ 12
1	0	0	3 ~ 6
1	0	1	2 ~ 3
1	1	0	1 ~ 2
1	1	1	< 1

IFMT (地址E5H, ISP 模式寄存器，复位值=xxxx,x000B)

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
-	-	-	-	-	MS2	MS1	MS0

MS2, MS1 and MS0: ISP 模式选择位，见表 20-3.

表 20-3. ISP 模式选择

MS2	MS1	MS0	ISP模式
0	0	0	备用
0	0	1	读
0	1	0	编程
0	1	1	页擦除

备用模式: 保持ISP 硬件处于无效状态

页擦除模式: 擦除一页 (512字节) 页地址由寄存器IFADRH,IFADRL选择

编程模式:编程地址由寄存器IFADRH,IFADRL选择

读模式: 读地址由寄存器IFADRH,IFADRL选择

IFADRH (地址E3H, ISP Flash地址高寄存器，复位值=0000,0000B)

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
(高位地址, A15~A8)							

IFADRL (地址E4H, ISP Flash地址低寄存器，复位值=0000,0000B)

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
(低位地址, A7~A0)							

IFD (地址E2H, ISP Flash数据寄存器，复位值=0000,0000B)

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
(读或编程数据字节)							

SCMD (地址E6H, ISP Sequential Command 寄存器，复位值=0000,0000B)

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
(ISP触发命令)							

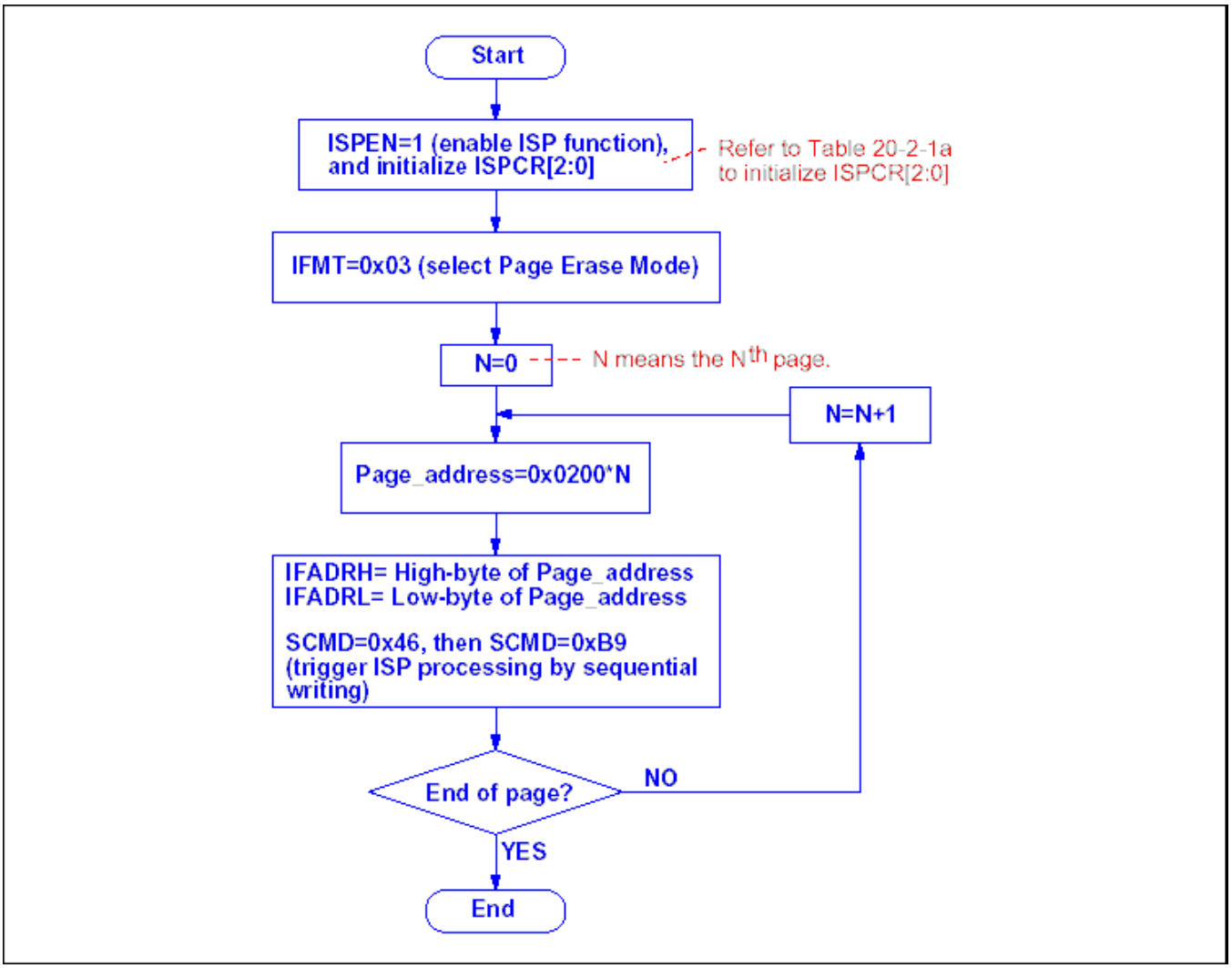
顺序写0x46 和0xB9到这寄存器能触发ISP操作.

20.2.2 ISP 模式说明

ISP 模式是用来引导程序来编程 AP-空间 和IAP-空间. 同样也能在*用户的应用程序* 对IAP-空间进行编程. 下面显示 ISP 模式下的不同操作.

20.2.2.1 Flash页擦除模式

图 20-2. “Flash 页擦除流程图”

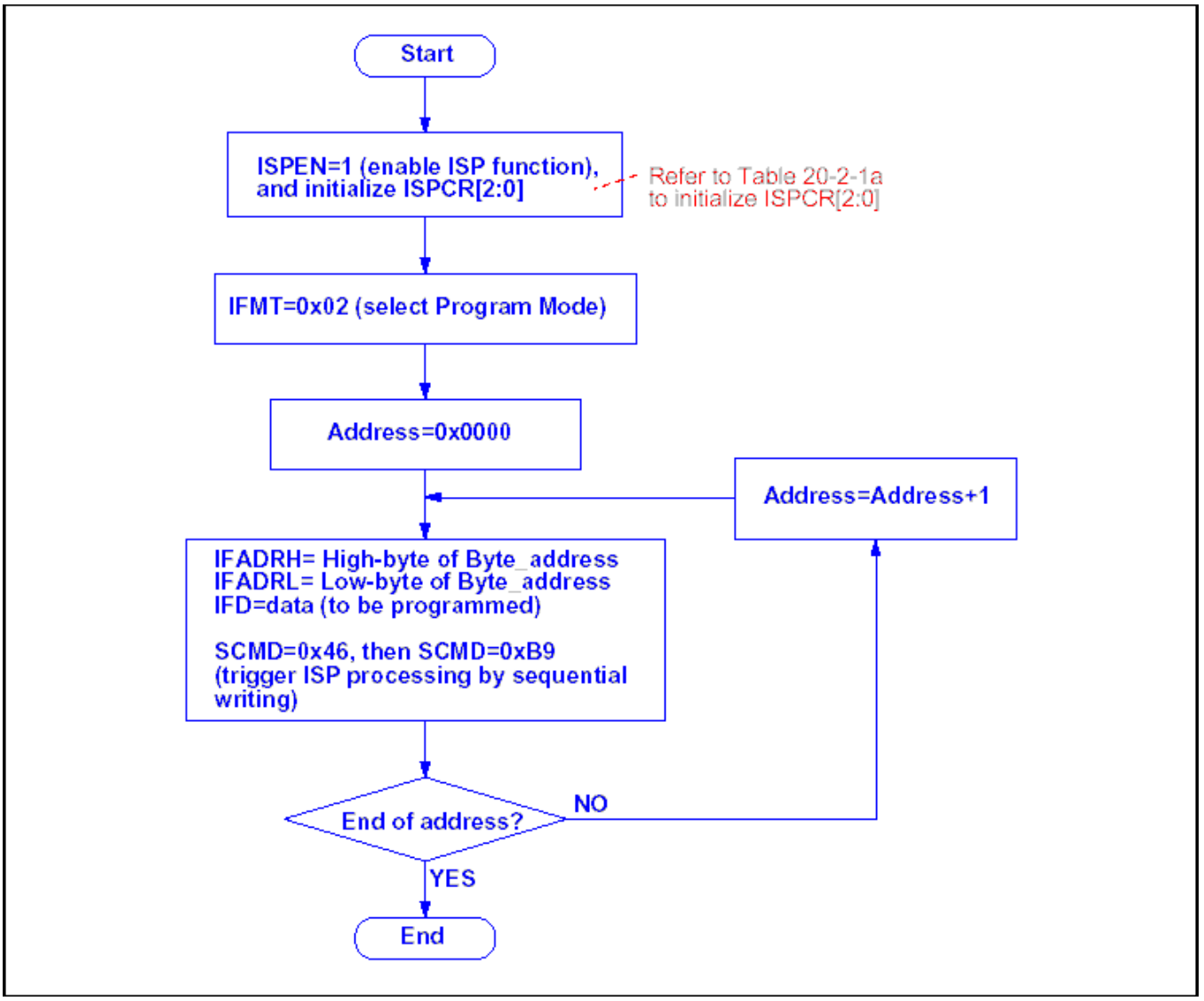


“页擦除模式”操作示例

```
MOV     ISPCR,#10000011b ;ISPCR.7=1, 使能 ISP
                         ;ISPCR[2:0]=011, 假定 MPC82-series 在 @11.0592MHz运行

MOV     IFMT,#03h        ;选择页擦除模式

MOV     IFADRH,??        ;使用 [IFADRH,IFADRL] 装载页地址
MOV     IFADRL,??        ;

MOV     SCMD,#46h        ;触发 ISP 处理
MOV     SCMD,#0B9h       ;

; MCU 停止直到操作完全结束
```

20.2.2.2 Flash编程模式

图 20-3. “Flash 编程流程图”



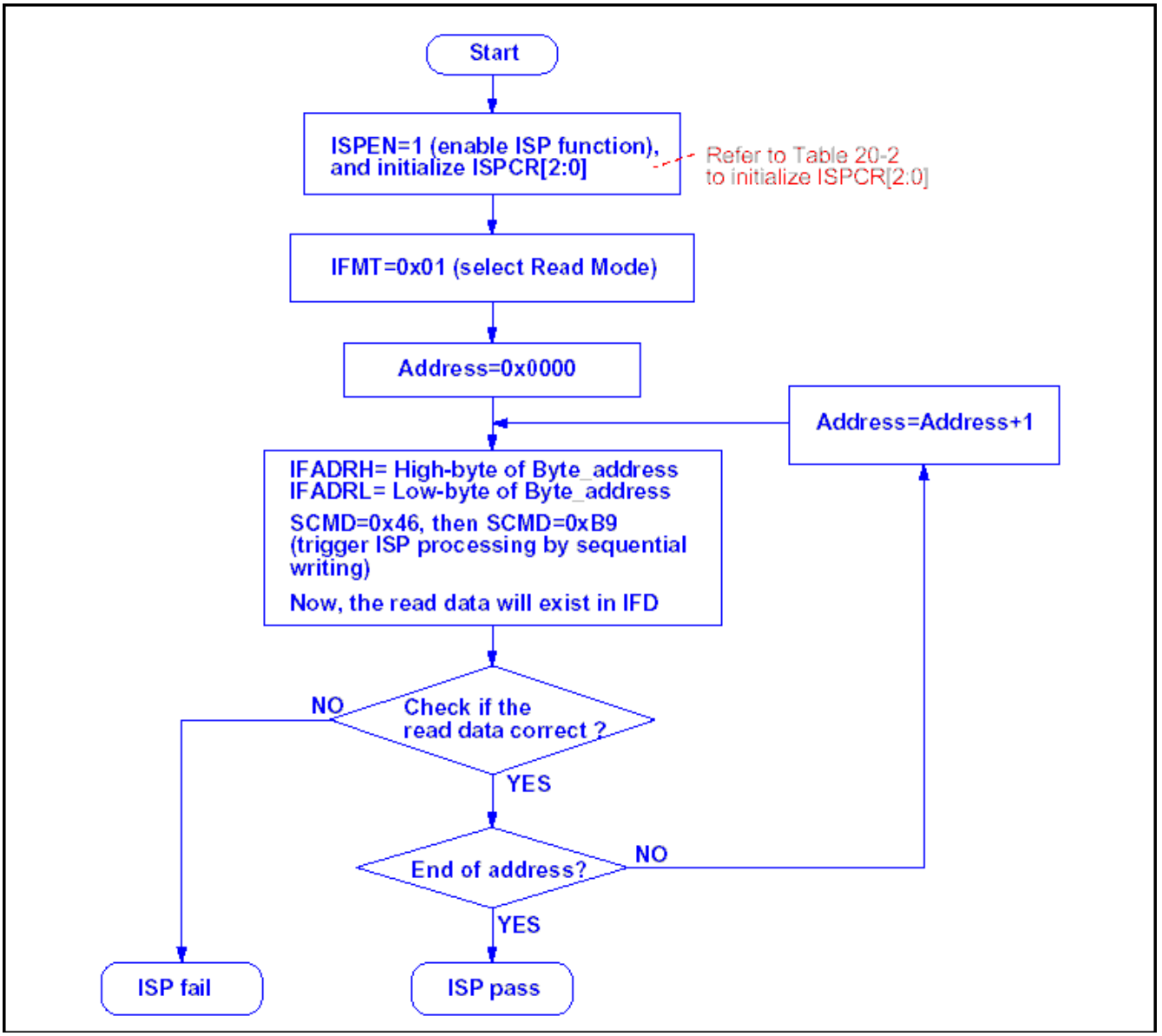
“编程模式”操作示例代码

```
MOV     ISPCR,#10000011b ;ISPCR.7=1, 使能 ISP
                         ;ISPCR[2:0]=011, 假设 MPC82-series 运行在 @11.0592MHz
```

```
MOV     IFMT,#02h          ;选择编程模式

MOV     IFADRH,??          ;通过 [IFADRH,IFADRL] 装载编程字节地址
MOV     IFADRL,??          ;
MOV     IFD,??             ;通过 IFD 装载编程数据

MOV     SCMD,#46h          ;触发 ISP 处理
MOV     SCMD,#0B9h         ;

; MCU等待直到全部处理结束
```

20.2.2.3 Flash 读模式

图 20-4. “Flash 读操作流程”



“读模式”操作示例程序

```
    MOV     ISPCR,#10000011b ;ISPCR.7=1, 使能ISP
                             ;ISPCR[2:0]=011, 假定 MPC82-series 运行在 @11.0592MHz
    MOV     IFMT,#01h        ;选择读模式
    MOV     IFADRH,??        ;fill [IFADRH,IFADRL] with byte address
    MOV     IFADRL,??        ;
    MOV     SCMD,#46h        ;trigger ISP processing
    MOV     SCMD,#0B9h       ;
    ;Now, MCU will halt here until processing completed
    MOV     A,IFD            ;now, the read data exists in IFD
    CJNE    A,??,isp_error   ;and, the user can check if the data is correct
    ;...
isp_error:
    JMP     $
```

20.2.3 实现在系统编程（ISP）

在使用ISP 功能之前，用户将使用通用编程器, “Megawin 8051 烧录器” (参见20.4.1节) 进行以下处理:

(1) 适合 ISP-空间 大小的 ‘引导程序’.

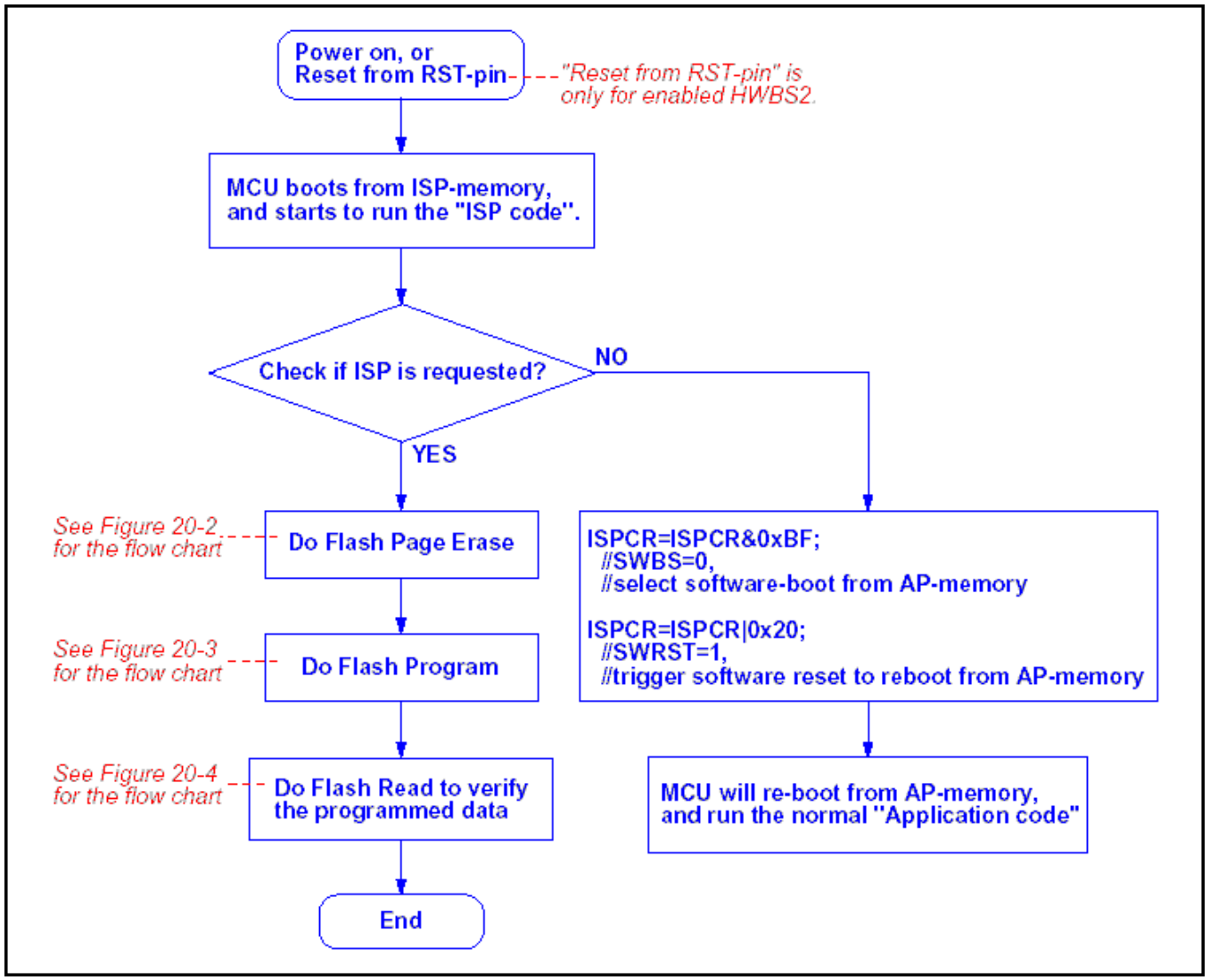
(2) 将 ‘引导程序’ (以下称 ‘**ISP 代码**’) 下载到ISP-空间.

我们已经知道, ISP 代码的作用是用来编程AP-空间 和 IAP-空间. 因而 ***MCU 必须从 ISP-空间 来引导用来执行 ISP 代码***. 从ISP-空间引导在系统编程有两种执行方法

方式 1: MCU 上电复位后由ISP地址开始引导

上电后MCU 直接从ISP-空间引导, MCU的硬件选项*HWBS*或*HWBS2*必须被使能. 一旦 *HWBS* 或*HWBS2*被使能,MCU上电复位后总是从ISP-空间引导引导代码. ISP 代码启动后将监测是否有ISP请求,如果没有 ISP请求, the ISP代码将触发软件复位,MCU将重新从AP-空间引导用户应用程序. 参见下面流程图.

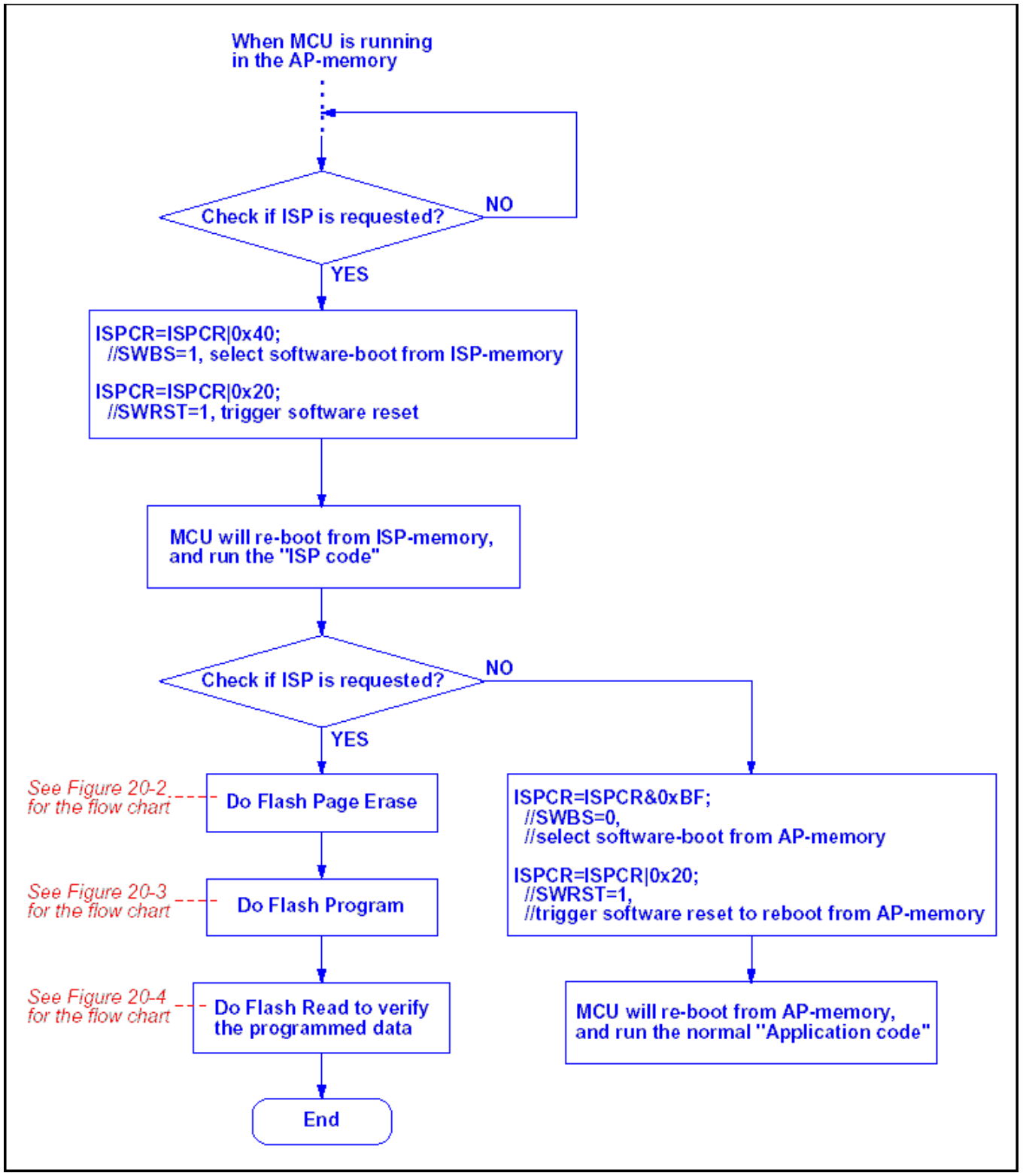
图 20-5. 从ISP-部分直接引导 (HWBS 或 HWBS2 已被激活)



方式 2: MCU 通过AP-空间来重新从ISP-空间引导

当MCU在AP-空间运行时通过触发一次软件复位重新选择从ISP-空间引导.在这个方案中，HWBS 和HWBS2都不必使能. MCU只需在AP-空间运行时触发一次软件复位重新选择从ISP-空间引导. 参见下列流程图.

图 20-6. 在AP-空间重新从ISP-空间引导



20.2.4 ISP 注意事项

编写ISP 代码的注意事项

ISP代码从MCU的FLASH中分配的*ISP 开始地址*开始存放(参见图 20-1),无需在用户代码中计算代码的偏移地址 (= *ISP 开始地址*). 代码的偏移地址由硬件自动分配. 用户只需象在AP-空间编写用户代码一样编写ISP代码.

ISP过程中的中断

在激活 ISP后, MCU 将停止所有任务直到 ISP 处理结束. 在此期间,已被使能的中断将被挂起不执行. 直到ISP处理结束, MPU 将依照中断标志位依次执行已产生的中断. 用户必须了解以下情况:

(1) MCU 正在执行ISP过程时将忽略任何中断.

(2) 低优先级的中断,外部中断, /INTx, 会保持激活状态直到 ISP过程结束. 或这些将被忽略.

ISP操作目标

如前所述, ISP 用来对AP-空间和 IAP-空间进行编程. 一旦访问的地址超过了 IAP-空间, 硬件会自动触发产生ISP处理. 这种 ISP 触发没有任何意义,硬件在做空操作.

ISP操作FLASH寿命

Flash能进行 20,000次擦写操作, 这就是说, 擦-写周期不超过20,000次. 因而用户可以随时对 AP-空间 和 IAP-空间进行修改.

20.2.5 Megawin提供的 ISP工具

虽然用户可以自己设计ISP程序, Megawin为用户提供了两种ISP工具; 一种 *ISP编程器*, PC机通过USB口来连接目标MCU;另一种通过计算机的COM 端口, 除了RS232收发器外无需其他的硬件. 下面简单描述这些工具. 用户可以申请Megawin提供详细的说明.

20.2.5.1 “Megawin 8051 ISP 编程器”

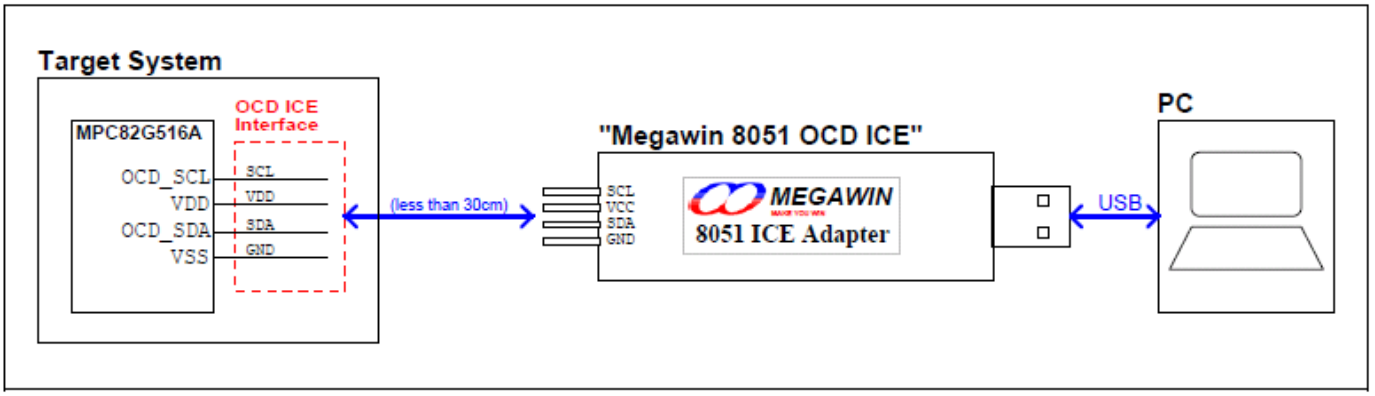
简介

1. 出厂前预编程了‘ISP代码’.
2. ISP介面只使用一个I/O端口 (P3.1).
3. 操作时无需振荡器工作.
4. 无需主机支持就能操作.

以上特点使ISP 编程器有很好的易用性. 显然, 这能单机下载程序. 这点在现场没有PC机的情况下非常有用. 以下是 “Megawin 8051 ISP 编程器”的图片. ISP 接口只需要三根引脚: 编程器通过***DTA***线将编程输出传递给目标MCU; ***VCC***和 ***GND***为ISP编程器提供电源. PC机通过USB 接口将编程数据下载进 ISP 编程器.

图 20-7. “8051 ISP 编程器”

图 20-8. ISP 操作系统连接图



20.3 IAP 操作

编程IAP非易失存储区叫做IAP编程，与ISP编程相比，除了以下两点不同，其它都是一样的。

(1) IAP只能编程IAP存储区,ISP能编程AP和IAP存储区
(2) IAP程序代码在AP存储区执行, ISP程序代码在ISP存储区执行.

所有ISP方法(见 20.2.2节)都可以应用到IAP操作，使用IAP功能之前，必须存在IAP存储区，用户可以用“Megawin 8051编程器”并行编程器配置MPC82G516 的IAP存储区 (见 20.4.1节).

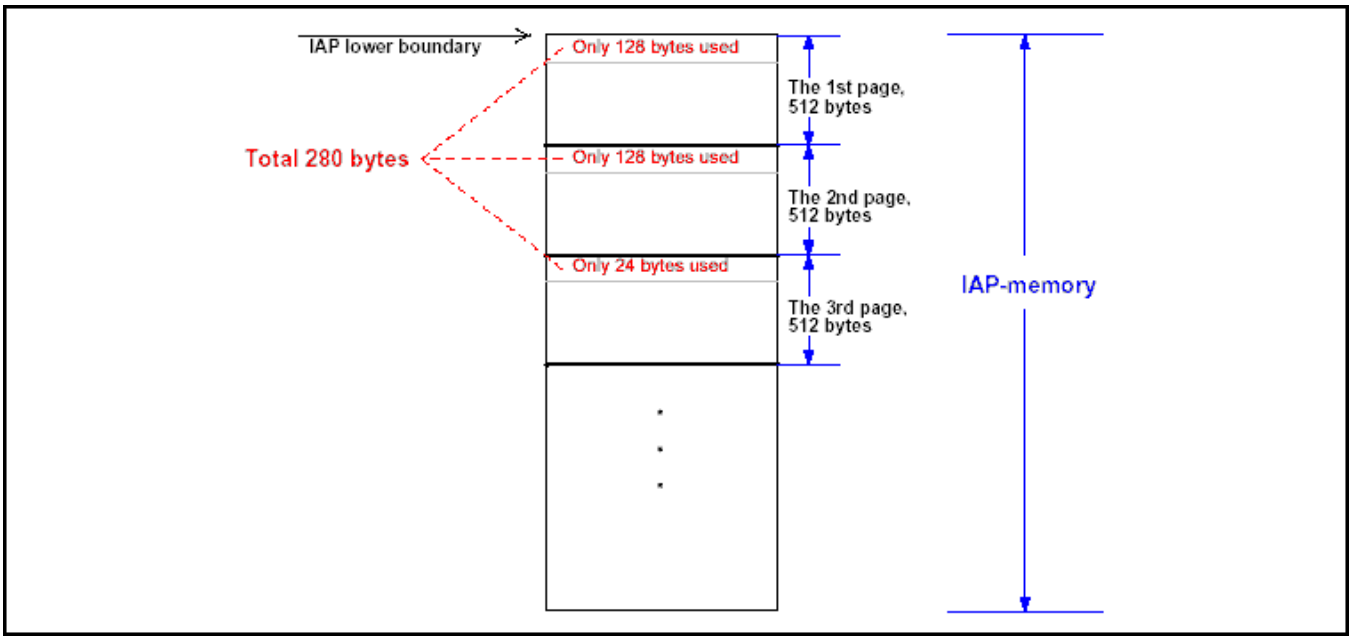
20.3.1 使用IAP功能更新程序

由于闪存只能执行页擦除，并且0xFF字节的编程为一个非0xFF字节。因此，如果一些字节（在同一页）必须改变为非0xFF ，用户应遵守以下步骤：

第一步) 读出所有数据保存到”**页面缓冲区**”
第二步) 擦除该页
第三步) 把需要的数据写入”**页面缓冲区**”
第四步) 把更新的”**页面缓冲区**”重新写入该页

在这里，用户可能会问：**页面缓冲区**在哪里？**页面缓冲区**可能存在于传统的256字节的暂存RAM或片上扩展内存(即XRAM ，访问用 ‘ MOVX '指令)通常情况下， 页面缓冲区大小应等于的闪存页面大小（即512字节），然而，由于有限空间的RAM或XRAM有时不可能的满足。下面是一个例子，显示如何解决这个问题，假设280字节的非易失性存储空间需要读写，但只有128字节的RAM可用于"**页面缓冲区**"
在这种情况下，我们可以用3页提供280字节的非易失存储空间:第一页128字节，第二页128字节，第三页24字节.如下图20-10所示。请注意，我们别无选择，只能采取这种用法！

图 20-10. 页面缓冲少于512字节时的用法



20.3.2 IAP 示例代码

如上所述，所有的ISP模式也可应用于IAP操作，这些模式的示范代码如下显示

触发“页面擦除模式” 的演示代码

```
MOV     ISPCR,#10000011b ; ISPCR.7=1, 启用ISP
                         ; ISPCR[2:0]=011, 假设MPC82系列运行在11.0592MHz
MOV     IFMT,#03h        ; 选择页擦除模式
MOV     IFADRH,??        ; 填写 [IFADRH,IFADRL] 为页地址
MOV     IFADRL,??        ; ！这个页面必须在IAP存储区
MOV     SCMD,#46h        ; 触发的ISP处理

MOV SCMD,#0B9h

;现在单片机将会停止运行，直到处理完成
```

触发”编程模式”的演示代码

```
MOV     ISPCR,#10000011b ; ISPCR.7=1, 启用 ISP
                         ; ISPCR[2:0]=011, 假设MPC82系列运行在11.0592MHz

MOV     IFMT,#02h        ; 选择 编程 模式

MOV     IFADRH,??        ; 填写 [IFADRH,IFADRL] 为字节地址
MOV     IFADRL,??        ; ！这个字节地址必须在IAP存储区
MOV     IFD,??           ; 填写IFD的数据进行编程
MOV     SCMD,#46h        ; 触发的ISP处理
MOV     SCMD,#0B9h       ;

; 现在单片机将会停止运行，直到处理完成
```

触发”**读模式**”的演示代码

```
MOV     ISPCR,#10000011b ; ISPCR.7=1, 启用 ISP
                         ; ISPCR[2:0]=011, 假设MPC82系列运行在11.0592MHz

MOV     IFMT,#01h        ; 选择 读 模式

MOV     IFADRH,??        ; 填写 [IFADRH,IFADRL] 为字节地址
```

```
MOV     IFADRL,??          ; ！这个字节地址必须在IAP存储区
MOV     SCMD,#46h          ;  触发的ISP处理
MOV     SCMD,#0B9h

; 现在单片机将会停止运行，直到处理完成

MOV     A,IFD              ;现在，读出的数据在IFD寄存器了
...
...
```

20.3.3 IAP注意事项

IAP过程中的中断

IAP触发ISP处理后，单片机将为ISP处理停止了一段时间的内部处理，直至处理完成， 此时，如果先前中断是使能的,那么中断将轮候提供服务。一旦处理完成后,单片机继续运行,并且继续处理中断队列中的仍然有效的中断.用户需要注意以下问题：

(1) ISP处理中，单片机停止运行,任何中断不能及时提供服务。

(2) /INTx 等低级别的外部中断,应该一直保持有效，直到ISP处理完成，否则 /INTx 将会被忽略。

IAP的存储目标

如前所述, IAP是用来编程IAP存储区，一旦IAP用于编程非IAP存储区，硬件将自动忽略触发的ISP处理,这样ISP触发非法并且硬件不做任何处理.

另一种读取IAP数据区的方法

若要读取Flash中的的IAP存储区,除了使用Flash读取方法，另一种方法是使用指令“MOVC A,@A+DPTR”, DPTR和ACC 分别是你想要地址和偏移地址，另外，访问的范围必须属于IAP存储区，否则读取数据将是不确定的，请注意，使用"MOVC"指令是速度远远超过使用IAP Flash读模式

IAP Flash寿命

内置Flash的寿命是 20,000 擦写,也就是说擦除后再写的次数不要超过20,000次,因此用户在应用中应该注意经常需要频繁更新的IAP存储区.

21 节能模式

MPC82G516 有两种降低功耗的模式和一种通过设置8位系统时钟来降低功耗. 空闲模式下CPU停止操作外部设备内部中断系统继续运行.掉电模式只保持RAM 和 SFR的内容,其他功能均冻结;重要的是外部中断能唤醒掉电模式. 可以通过降低工作频率来降低待机功耗.

寄存器PCON 和 PCON2 与节能模式有关,如下所示.

PCON (地址87H, Power Control 寄存器, 复位值=00xx,0000B (或00x1,0000B 上电复位后))

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

GF1: 通用多功能控制位1.

GF0: 通用多功能控制位0.

PD: 节能模式设置位. 设置此位可以激活节能模式.

IDL: 空闲模式设置位. 设置此位可以激活空闲模式.

PCON2 (地址C7H, Power Control 寄存器 2, 复位值=00x0,0000B)

7	6	5	4	3	2	1	0
-	-	-	-	-	SCKD2	SCKD1	SCKD0

SCKD2~SCKD0: 系统时钟分频控制位.

SCKD2	SCKD1	SCKD0	Fosc (系统时钟)
0	0	0	晶振频率
0	0	1	晶振频率 /2
0	1	0	晶振频率/4
0	1	1	晶振频率/8
1	0	0	晶振频率/16
1	0	1	晶振频率/32
1	1	0	晶振频率/64
1	1	1	晶振频率/128

(参见22节: 系统时钟.)

21.1 空闲模式

设置 IDL 位 (PCON.0) 将使MCU进入空闲模式, CPU的内部时钟源停止工作外部设备继续工作 , 例如中断, 定时器, 串口功能 and so forth. The CPU contents,片内RAM, 所有的特殊寄存器在空闲状态时保持不变.I/O口保持空闲模式前的逻辑状态.

有两种办法终止空闲模式. 激活中断使能例如外部中断, 定时器, 串口中断和键盘中断能硬件清除PCON.0来终止空闲模式. 进入中断服务程序知道执行到指令 RETI, 执行下一条指令表示空闲模式已经执行完了.另一个结束空闲模式的方法是RST引脚上产生一个上电复位. 晶振将继续运行,硬件复位需要保持 24 时钟周期确保复位完成.

标志位 GF0 和GF1能指示出中断是在非空闲操作还是空闲时产生的. 举例说明, 激活空闲模式的指令也能设置一个或两个标志位. 空闲状态被中断打断时,中断服务程序能通过标志位判断是普通操作还是空闲操作.

21.2 休眠模式

软件调用休眠模式能进一步降低功耗.通过指令设置PD 位 (PCON.1) 可以使MCU进入休眠模式.在休眠模式下,片内振荡停止工作.时钟被冻结, 所有功能均被冻结, 片内RAM 和所有的工作寄存器保持进入休眠前的状态.端口输出依照各自的工作寄存器保持状态.

任何RST引脚引起的硬件复位或外部中断 (INT0~INT3) 与键盘中断都能将MCU从休眠状态唤醒. 复位将初始化工作寄存器但不改变片内RAM的值.外部中断 与键盘中断唤醒将保持工作寄存器和片内RAM的值; 然后将进入中断服务程序, 从中断服务程序返回后再执行一条指令将使系统结束休眠状态.

21.2.1 自休眠模式中断唤醒

休眠模式能被外部中断或键盘中断唤醒, 必须在调用休眠模式指令之后插入至少一个NOP指令. 这个NOP指令将避免从中断服务程序返回时产生意外的错误.

示例: 使用/INT0 唤醒的例子.

```
;*************************************************************************************
; 用/INT0 中断来唤醒休眠
;*************************************************************************************

 INT0     BIT    0B2H        ; P3.2
 EA       BIT    0AFH        ; IE.7
 EX0      BIT    0A8H        ; IE.0
          CSEG   AT  0000h
          JMP    start

          CSEG   AT  0003h   ; /INT0 中断向量, 地址0003h
          JMP    IE0_isr

 IE0_isr

          CLR    EX0
          ;... 嵌入应用程序
          ;...
          RETI

;
 start:
          ;....

          SETB   INT0        ; 拉高 P3.2

          CLR    IE0         ; 清除 /INT0 中断标志
          SETB   IT0         ; may select falling-edge/low-level triggered
          SETB   EA          ; 允许中断
          SETB   EX0         ; 使能 /INT0 中断

          ORL    PCON,#02h   ; 使MCU进入休眠模式
          NOP                ; ! 注意: 这必须有一个 NOP

 操作摘要:
          ;If /INT0是下降沿触发, MCU被唤醒, 进入"IE0_isr",
          ;然后回到这里继续运行 !

          ;...
          ;...
 ;
```

21.3 时钟降速

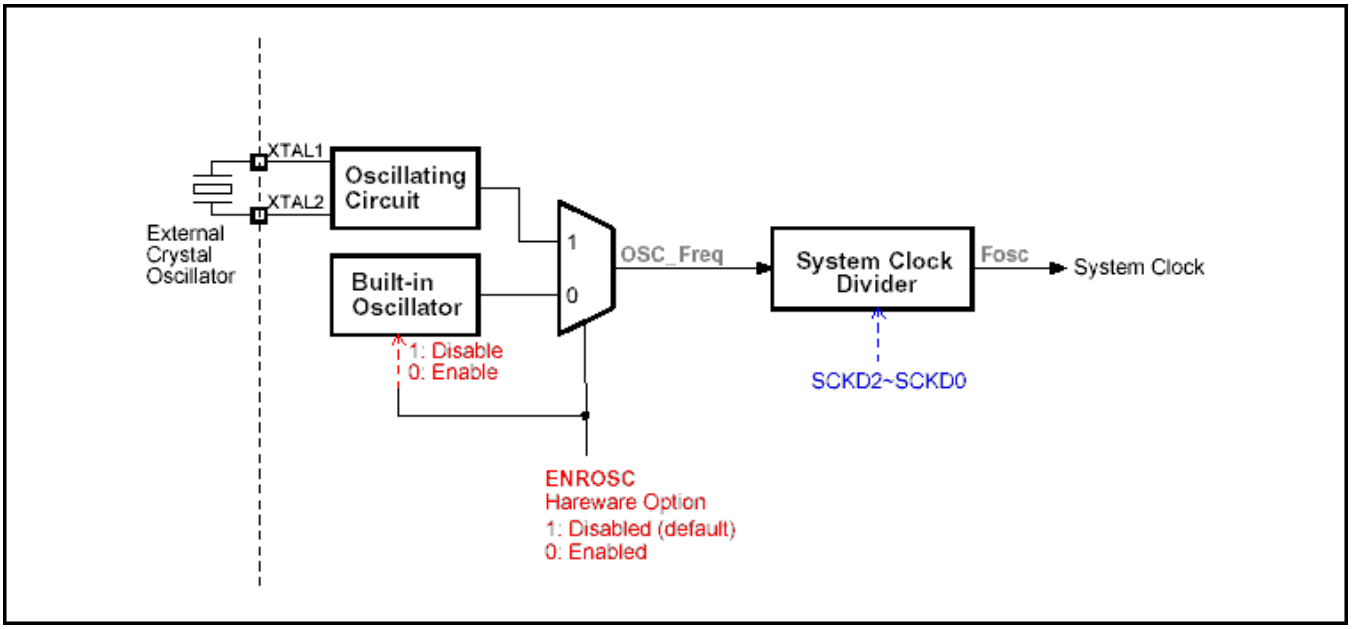
对SCKD2~SCKD0 位 (PCON2 寄存器, 参见22节)进行编程成非-0/0/0的数将使MCU的运行速度减慢来降低功耗. 用户可以对照列表来选择不同的运行速度. 原则上速度选项不影响系统的正常操作,可以随时通过编程来恢复到常速运行.

22 系统时钟

系统时钟有两个时钟源: 外部时钟和内置振荡. 系统时钟, Fosc, 可以通过设置来选择两种时钟源,如图 22-1所示. 用户能通过设置SCKD2~SCKD0 位 (PCON2 寄存器)来获得理性的时钟.

内置振荡由硬件选项ENROSC来使能. 参见25节: MCU的硬件选项.

图 22-1. 系统时钟示意图



PCON2 (地址C7H, 电源控制寄存器 2, 复位值=00x0,0000B)

7	6	5	4	3	2	1	0
-	-	-	-	-	SCKD2	SCKD1	SCKD0

SCKD2~SCKD0: 系统时钟分频器选择位.

SCKD2	SCKD1	SCKD0	Fosc (System Clock)
0	0	0	晶振频率
0	0	1	晶振频率/2
0	1	0	晶振频率/4
0	1	1	晶振频率/8
1	0	0	晶振频率/16
1	0	1	晶振频率/32
1	1	0	晶振频率/64
1	1	1	晶振频率/128

22.1 内置振荡器

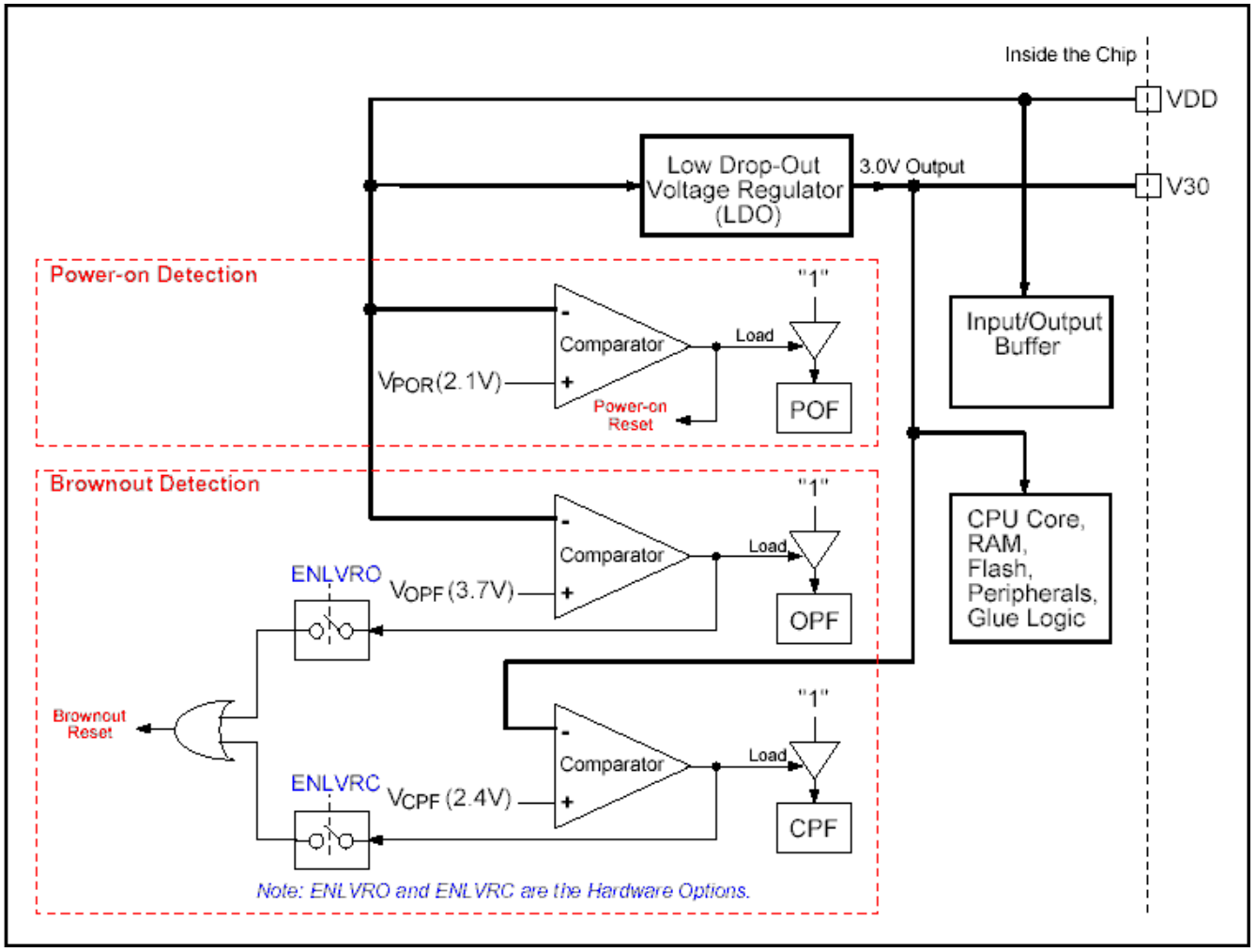
The MPC82G516 有一个内置的6MHz振荡.如果不需要精确的外部时钟时.可以使能内部振荡,可以使用通用编程器对ENROSC硬件选项进行编程来对此进行设置.

特点是6MHz 室温 (25˚C). 误差 ±30% 在 –40˚C 到 +85˚C (–40˚C误差+30%, +85˚C误差-30%). *所以只能在不需要精准的振荡时钟的时候选择内部振荡。*

23 电源监测功能

MPC82G516 具有电源监测功能预防在上电掉电过程中或电源不稳定时引发错误操作.这由两种硬件功能来保证: 上电监测和掉电监测. 图 23-1 电源监测原理图.

图 23-1. 电源监测结构图表



23.1 上电监测

POF标志(PCON.4)由硬件设置用来指示上电过程.标志由硬件设置由软件来清除，能帮助用户了教MCU的启动条件是什么,*冷启动* (即上电过程)或热启动 (如由RST引脚输入的硬件复位，软件复位或看门狗复位). 除上电时，POF位同样在VDD 降到V_{POR}电压之下时也由硬件进行设置.

PCON (地址87H，电源控制寄存器，复位值=00xx,0000B (或00x1,0000B上电复位时))

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

POF: Power-ON标志. 上电复位标志.能被软件设置. 这只能被软件清除

23.2 掉电监测

电源监测功能能监测电源是否已经跌到正常工作电压阀值═以下. 电源监测功能在电源异常时将置位电源异常标志位,如果电源监测被使能这将产生一个电源监测(#12 in 表 19-1). 以下简述两种电源监测模式 .

有两种电源监测方式:

(1) VDD 电源异常选项: 当VDD引脚电压降到 V_{OPF} (3.7V)以下, OPF标志被硬件置位表明VDD 电压异常. 此选项适用使用在5V电压系统.

(2) LDO 电源异常选项: 当LDO 输出电压降到 V_{CPF} (2.4V)以下, CPF标志被硬件置位表明 LDO 电压异常,此选项适用于3.3V电压系统.

注: 参见27.1 节3.3V, 5V或宽电压系统.

注意在上电复位过程中, 电压异常标志 OPF和 CPF在电压高于3.7V 和 2.4V将分别被硬件置位, 用户可以对此进行软件清除. 标志 OPF 和 CPF可以触发电源监测中断 ,如果 EA被置位和 EOPFI 或 EOPCI 被置位 参见19节: 中断系统).

硬件选项ENLVRO 或 ENLVRC如果被使能, 电源监测触发一次内部复位.
(参见25节: MCU的硬件选项).

EVRCR 寄存器 电源监测控制及相关标志位.

EVRCR (地址97H, EVR Control 寄存器, 复位值=00xx,0000B (或 0011,0000B 在上电复位后))

7	6	5	4	3	2	1	0
EOPFI	ECPFI	OPF	CPF	PMUOFF	(保留)	(保留)	(保留)

EOPFI: 使能/禁止 中断OPF=1时.

ECPFI: 使能/禁止 中断CPF=1时.

OPF: VDD电压异常标志. VDD 引脚上的电源电压跌倒3.7V以下时, 硬件置位此位,只能被软件清除.

CPF: LDO 电压异常标志.LDO电压跌倒2.4V以下时, 硬件置位此位,只能被软件清除.

PMUOFF:设置此位将不使用电源监测功能来记录电源状态.

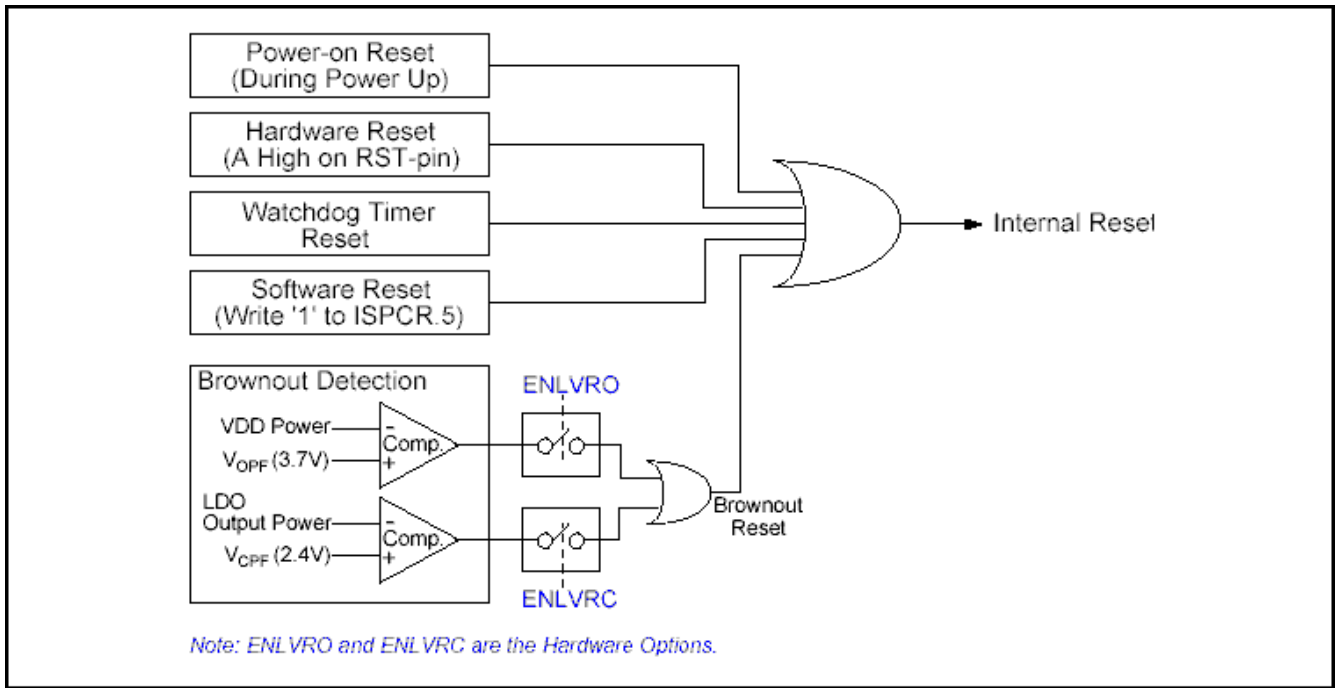
寄存器的保留位必须是'0'.

24 复位源

复位能通过下列的复位源触发 (参见 图 24-1):

- 上电复位
- 通过RST引脚硬件复位
- 看门狗定时器复位
- 软件复位
- 电源监测的欠压复位

图 24-1. 复位结构图表



24.1 上电复位

上电复位 (POR)用于在电源上电过程中产生一个复位信号. 微控制器在VDD电压上升到V_{POR} (POR开始电压)电压之前将保持复位状态. VDD电压降到 V_{POR}之下后微控制器将再次进入复位状态. 在一个电源周期中, 如果需要再产生一次上电复位 VDD 必须降到 V_{POR}之下. 参见30节: 直流特性 相关 V_{POR}部分.

24.2 RST 引脚硬件复位

保持复位引脚 RST至少 24个振荡周期的高电平,将产生一个上电复位信号,为确保微控制器的正常工作,必须在RST引脚上连接可靠的硬件复位电路.

24.3 看门狗复位

看门狗定时器被使能,它将以系统时钟的12分频(12/Fosc)进行增量计数,正常使用过程中必须不断清零,如果溢出将会产生中断复位信号.

24.4 软件复位

向位 SWRST 写'1'将触发软件复位,然后通过设置SWBS寄存器来选择从AP-空间或ISP-空间来重新引导程序. 下面是ISPCR 寄存器详细说明.

ISPCR (地址E7H, ISP Control 寄存器, 复位值=000x,x000B)

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
ISPEN	SWBS	SWRST	-	-	CKS2	CKS1	CKS0

SWBS: 软件引导选择. 1/0用来选择软件复位后从ISP-空间/AP-空间引导.

SWRST: 写 '1' 触发软件复位.

24.5 掉电复位

硬件选项 ENLVRO 或 ENLVRC 被使能, 欠压时触发中断复位.
(参见25节: 微控制器的硬件选项).

25 硬件熔丝位选项

单片机的熔丝位选项决定了器件的性能，且不能够被编程或者软件控制。熔丝位选项只能用通用编程器，如“Megawin 8051 烧录器”进行程式化。当全片擦除后，所有熔丝位选项均为“disabled”状态，并且没有ISP内存和IAP内存设定。MPC82G516有如下熔丝位选项：

ENLVRC:
[enabled]: 欠压复位当LDO输出低于V_{CPF} (2.4V)时。
[disabled]: 无欠压复位当LDO输出低于V_{CPF} (2.4V)时。

OSCDN:
[enabled]: 振荡增益衰减，减少EMI干扰。
[disabled]: 正常振荡增益。

ENLVRO:
[enabled]: 欠压复位，当VDD低于V_{OPF} (3.7V).
[disabled]: 无欠压复位，当VDD低于V_{OPF} (3.7V).

ENROSC:
[enabled]: 使能内置RC振荡器.
[disabled]: 禁用内置RC振荡器.

WDSFWP:
[enabled]: 特殊功能寄存器WDTCR软件写保护，除CLRW位外。
[disabled]: 特殊功能寄存器WDTCR可软件写。

HWENW (带有变量 **HWWIDL** 和 **HWPS[2:0]**):
[enabled]:上电时自动硬件使能看门狗定时器。

也就是说:
在WDTCR寄存器中，硬件自动：
(1) 设置 *ENW* 位,
(2) 载入 **HWWIDL** 进 *WIDL* 位中，并且
(3) 载入 **HWPS[2:0]** 进 *PS[2:0]* 位中.

例如：
如果 **HWWIDL** 和 **HWPS[2:0]** 分别被设定为1 和 5，那么 **WDTCR** 将被初始化为 0x2D 当上电时，如下：

WDTCR (Watch-Dog-Timer Control Register)

7	6	5	4	3	2	1	0
WRF	-	*ENW*	CLRW	*WIDL*	*PS2*	*PS1*	*PS0*
		set		load		load	
		1		HWWIDL		HWPS[2:0]	

[disabled]: 上电时看门狗定时器无改变。

26 指令集

80C51指令集专门为8位控制应用程序最优化，它规定了多种快速寻址模式访问内部RAM，减轻小型数据结构的字节运算。这种指令集提供广泛的类似于分散数据类型支持单个位变量，允许当需要布尔运算时的直接位操作控制和逻辑运算。

MPC82G516指令集除了执行时间外，是完全兼容80C51指令的。例如，执行一条指令的时钟周期数，最短只需要一个时钟周期，最长需要7个时钟周期。

寻址模式

80C51指令集的寻址模式如下：

Direct Addressing 直接寻址
直接寻址时操作数用指令中一个8位地址的区域表示，只有内部数据存储器和特殊功能寄存器可以直接寻址。

Indirect Addressing 间接寻址
间接寻址时指令用一个包含操作数地址的寄存器表示，内部和外部存储器均可间接寻址。8位地址的地址寄存器可以是选中区的R0或R1，16位地址的地址寄存器只能是16位的“数据指针”寄存器，DPTR。

Register Instructions 寄存器操作（寻址）
包含从R0到R7的寄存器区可以被某些指令存取，这些指令的操作码中用3位寄存器说明。存取寄存器的指令有更高的代码效率，因为这种模式减少了一个地址字节。当指令被执行时，其中被选取的区一个8位寄存器被存取。执行时，用PSW寄存器中两位区选择位来选择四分之一区。

Register-Specific Instructions 特殊寄存器寻址（寄存器间接寻址）
一些指令具有一个特定的寄存器，例如，一些指令常用于累加器，或数据指针等等，所以没有需要指向它的地址字节。操作码本身就行了。有关累加器的指令A就是累加器的特殊操作码。

Immediate Constants 立即寻址
常量的数值可以在程序存储器中跟随操作码。例如，“MOV A, #100”将十进制数100装入累加器，该数表示为十六进制是64H。

Indexed Addressing 索引寻址
索引寻址只能访问程序存储器，且只读。这种寻址模式用查表法读取程序存储器。一个16位基址寄存器（数据指针DPTR或程序计数器PC）指向表的基地址，累加器提供偏移量。程序存储器中表项目地址由基地址加上累加器数据后形成。另一种索引寻址方式是利用“case jump”指令。跳转指令中的目标地址是基地址加上累加器数据后的值。

介绍指令集之前，需注意的：

Rn	当前选中寄存器区的工作寄存器R0-R7
direct	128个内部RAM地址，包括任意I/O口、控制或状态寄存器
@Ri	用R0或R1间接寻址内部RAM地址
#data	指令中的8位常量
#data16	指令中的16位常量
addr16	用于LCALL和LJMP指令中的16位目标地址，可以是64K字节程序存储器地址空间的任何位置
addr11	用于ACALL和AJMP指令中的11位目标地址，只能是从下一条指令的第一个字节开始计算，同一个程序存储器2K字节页面内的位置
rel	有符号的8位偏移字节，用SJMP和所有的条件跳转，范围在以下一条指令的第一个字节开始的－128到＋127字节内
bit	内部RAM、I/O、控制位或状态位的128位直接位寻址位

26.1 算术运算指令

助记符	描述	字节	占用时钟周期
算术运算			
ADD A,Rn	将寄存器Rn中的内容加到累加器中	1	2
ADD A,direct	直接地址单元中的内容加到累加器中	2	3
ADD A,@Ri	寄存器工作寄存器Ri指向的地址单元中的内容加到累加器中	1	3
ADD A,#data	立即数加到累加器中	2	2
ADDC A,Rn	累加器与工作寄存器Rn中的内容、连同进位位相加，结果存在累加器中	1	2
ADDC A,direct	累加器与直接地址单元的内容、连同进位位相加，结果存在累加器中	2	3
ADDC A,@Ri	累加器与工作寄存器Ri指向的地址单元中的内容、连同进位位相加，结果存在累加器中	1	3
ADDC A,#data	累加器与立即数、连同进位位相加，结果存在累加器中	2	2
SUBB A,Rn	累加器与工作寄存器中的内容、连同借位位相减，结果存在累加器中	1	2
SUBB A,direct	累加器与直接地址单元中的内容、连同借位位相减，结果存在累加器中	2	3
SUBB A,@Ri	累加器与工作寄存器Ri指向的地址单元中内容、连同借位位相减，结果存在累加器中	1	3
SUBB A,#data	累加器与立即数、连同借位位相减，结果存在累加器中	2	2
INC A	累加器中的内容加1	1	2
INC Rn	寄存器Rn的内容加1	1	3
INC direct	直接地址单元中的内容加1	2	4
INC @Ri	工作寄存器Ri指向的地址单元中的内容加1	1	4

INC	**DPTR**	数据指针DPTR的内容加1	1	1
DEC	**A**	累加器中的内容减1	1	2
DEC	**Rn**	寄存器Rn中的内容减1	1	3
DEC	**direct**	直接地址单元中的内容减1	2	4
DEC	**@Ri**	工作寄存器Ri指向的地址单元中的内容减1	1	4
MUL	**AB**	ACC中内容与寄存器B中内容相乘，其结果低位存在ACC中、高位存在寄存器B中	1	4
DIV	**AB**	ACC中内容除以寄存器B中内容，商存在ACC，而余数存在寄存器B中	1	5
DA	**A**	ACC十进制调整	1	4

26.2 逻辑操作指令

助记符		描述	字节	占用时钟周期
逻辑运算				
ANL	**A,Rn**	累加器和寄存器Rn中的内容相“与”	1	2
ANL	**A,direct**	累加器和直接地址单元中的内容相“与”	2	3
ANL	**A,@Ri**	累加器和工作寄存器Ri指向的地址单元中的内容相“与”	1	3
ANL	**A,#data**	累加器和立即数相“与”	2	2
ANL	**direct,A**	直接地址单元中的内容和累加器相“与”	2	4
ANL	**direct,#data**	直接地址单元中的内容和立即数相“与”	3	4
ORL	**A,Rn**	累加器和寄存器Rn中的内容相“或”	1	2
ORL	**A,direct**	累加器和直接地址单元中的内容相“或”	2	3
ORL	**A,@Ri**	累加器和工作寄存器Ri指向的地址单元中的内容相“或”	1	3
ORL	**A,#data**	累加器和立即数相“或”	2	2
ORL	**direct,A**	直接地址单元中的内容和累加器相“或”	2	4
ORL	**direct,#data**	直接地址单元中的内容和立即数相“或”	3	4
XRL	**A,Rn**	累加器和寄存器Rn中的内容相“异或”	1	2
XRL	**A,direct**	累加器和直接地址单元中的内容相“异或”	2	3
XRL	**A,@Ri**	累加器和工作寄存器Ri指向的地址单元中的内容相“异或”	1	3
XRL	**A,#data**	累加器和立即数相“异或”	2	2
XRL	**direct,A**	直接地址单元中的内容和累加器相“异或”	2	4
XRL	**direct,#data**	直接地址单元中的内容和立即数相“异或”	3	4
CLR	**A**	累加器内容清“0”	1	1
CPL	**A**	累加器按位取反	1	2
RL	**A**	累加器循环左移一位	1	1
RLC	**A**	累加器连同进位位CY循环左移一位	1	1
RR	**A**	累加器循环右移一位	1	1
RRC	**A**	累加器连同进位位CY循环右移一位	1	1
SWAP	**A**	累加器高低半字节互换	1	1

26.3 数据传送指令

助记符	描述	字节	占用时钟周期
数据传送			
MOV A,Rn	寄存器Rn中的内容送到累加器中	1	1
MOV A,direct	直接地址单元中的内容送到累加器中	2	2
MOV A,@Ri	工作寄存器Ri指向的地址单元中的内容送到累加器中	1	2
MOV A,#data	立即数送到累加器中	2	2
MOV Rn,A	累加器中内容送到寄存器Rn中	1	2
MOV Rn,direct	直接寻址单元中的内容送到寄存器Rn中	2	4
MOV Rn,#data	立即数直接送到寄存器Rn中	2	2
MOV direct,A	累加器送到直接地址单元	2	3
MOV direct,Rn	寄存器Rn中的内容送到直接地址单元	2	3
MOV direct,direct	直接地址单元中的内容送到另一个直接地址单元	3	4
MOV direct,@Ri	工作寄存器Ri指向的地址单元中的内容送到直接地址单元	2	4
MOV direct,#data	立即数送到直接地址单元	3	3
MOV @Ri,A	累加器送到以工作寄存器Ri指向的地址单元中	1	3
MOV @Ri,direct	直接地址单元中内容送到以工作寄存器Ri指向的地址单元中	2	3
MOV @Ri,#data	立即数送到以工作寄存器Ri指向的地址单元中	2	3
MOV DPTR,#data16	16位常数的高8位送到DPH，低8位送到DPL	3	3
MOVC A,@A+DPTR	以DPTR为基地址变址寻址单元中的内容送到累加器中	1	4
MOVC A,@A+PC	以PC为基地址变址寻址单元中的内容送到累加器中	1	4
MOVX A,@Ri 注1	寄存器Ri指向扩展RAM地址(8位地址)中的内容送到ACC中	1	3
MOVX A,@DPTR 注1	数据指针指向扩展RAM地址(16位地址)中的内容送到ACC中	1	3
MOVX @Ri,A 注1	累加器中的内容送到寄存器Ri指向的扩展RAM地址（8位地址）中	1	4
MOVX @DPTR,A 注1	累加器中的内容送到寄存器Ri指向的扩展RAM地址（16位地址）中	1	3
MOVX A,@Ri 注2	寄存器Ri指向片外RAM地址中的内容送到ACC中	1	7 Note3
MOVX A,@DPTR 注2	数据指针指向片外RAM地址（16位地址）中内容送到ACC中	1	7 Note3
MOVX @Ri,A 注2	累加器中的内容送到寄存器Ri指向片外RAM地址（8位地址）中	1	7 Note3
MOVX @DPTR,A 注2	累加器中的内容送到数据指针指向片外RAM地址（16位地址）中	1	7 Note3
PUSH direct	直接地址单元中的数据压入堆栈中	2	4
POP direct	出栈数据送到直接地址单元中	2	3
XCH A,Rn	累加器与寄存器Rn中的内容互换	1	3
XCH A,direct	累加器与直接地址单元中的内容互换	2	4
XCH A,@Ri	累加器与工作寄存器Ri指向的地址单元中内容互换	1	4
XCHD A,@Ri	累加器与工作寄存器Ri指向的地址单元中内容低半字节互换	1	4

注1：当控制位 EXTRAM=0，所有“MOVX”指令均指向片内扩展XRAM区

注2：当控制位 EXTRAM=1，所有“MOVX”指令均指向外部数据存储区

注3：访问外部数据存储区的机器周期时间计算方法是: ***7 + 2 x (ALE延长时钟) + (RW延长时钟)***

26.4 布尔操作指令

助记符	描述	字节	占用时钟周期
布尔变量操作			
CLR C	清“0”进位位	1	1
CLR bit	清“0”直接地址位	2	4
SETB C	置“1”进位位	1	1
SETB bit	置“1”直接地址位	2	4
CPL C	进位位求反	1	1
CPL bit	直接地址位求反	2	4
ANL C,bit	进位位和直接地址位相“与”	2	3
ANL C,/bit	进位位和直接地址位的反码相“与”	2	3
ORL C,bit	进位位和直接地址位相“或”	2	3
ORL C,/bit	进位位和直接地址位的反码相“或”	2	3
MOV C,bit	直接地址位数据送入进位位	2	3
MOV bit,C	进位位数据送入直接地址位	2	4

26.5 控制和转移指令

助记符	描述	字节	占用时钟周期
PROAGRAM AND MACHINE CONTROL			
ACALL addr11	绝对短调用子程序，2K字节（页内）空间限制	2	6
LCALL addr16	绝对长调用子程序，64K字节空间限制	3	6
RET	子程序返回	1	4
RETI	中断子程序返回	1	4
AJMP addr11	绝对短转移，2K字节（页内）空间限制	2	3
LJMP addr16	绝对长转移，64K字节空间限制	3	4
SJMP rel	相对转移	2	3
JMP @A+DPTR	转移到DPTR加ACC所指间接地址	1	3
JZ rel	累加器为“0”则转移	2	3
JNZ rel	累加器不为“0”则转移	2	3
JC rel	进位位为“1”则转移	2	3
JNC rel	进位位为“0”则转移	2	3
JB bit,rel	直接地址位为“1”则转移	3	4
JNB bit,rel	直接地址位为“0”则转移	3	4
JBC bit,rel	直接地址位为“1”则转移，且清“0”该位	3	5
CJNE A,direct,rel	累加器中的内容不等于直接地址单元的内容，则转移到偏移量所指向的地址，否则程序往下执行	3	5
CJNE A,#data,rel	累加器中的内容不等于立即数，则转移到偏移量所指向的地址，否则程序往下执行	3	4
CJNE Rn,#data,rel	寄存器Rn中的内容不等于立即数，则转移到偏移量所指向的地址，否则程序往下执行	3	4
CJNE @Ri,#data,rel	工作寄存器Ri指向的地址单元中的内容不等于立即数，则转移到偏移量所指向的地址，否则程序往下执行	3	5
DJNZ Rn,rel	寄存器Rn中的内容减1，如不等于0，则转移到偏移量所指向的地址，否则程序往下执行	2	4
DJNZ direct,rel	直接地址单元中的内容减1，如不等于0，则转移到偏移量所指向的地址，否则程序往下执行	3	5
NOP	空操作指令	1	1

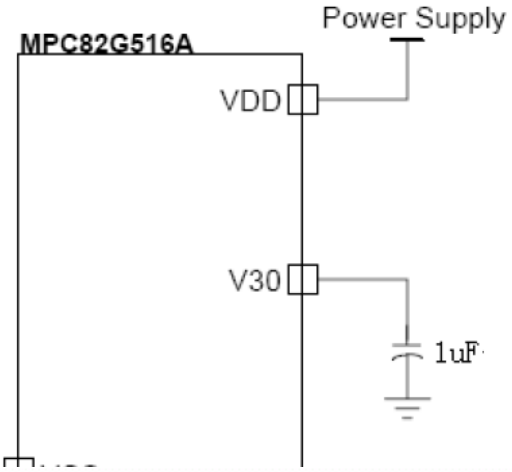
27 应用事项

27.1 V30 供电

MPC82G516的V30管脚，系统在工作状态下，必须要接上一个1uF的电容，使其内部LDO电源工作正常，并且可以增强芯片抗干扰。

MPC82G516在系统中，且需要一个滤波电容，如图27-3所示。

图27-3. V30 接脚



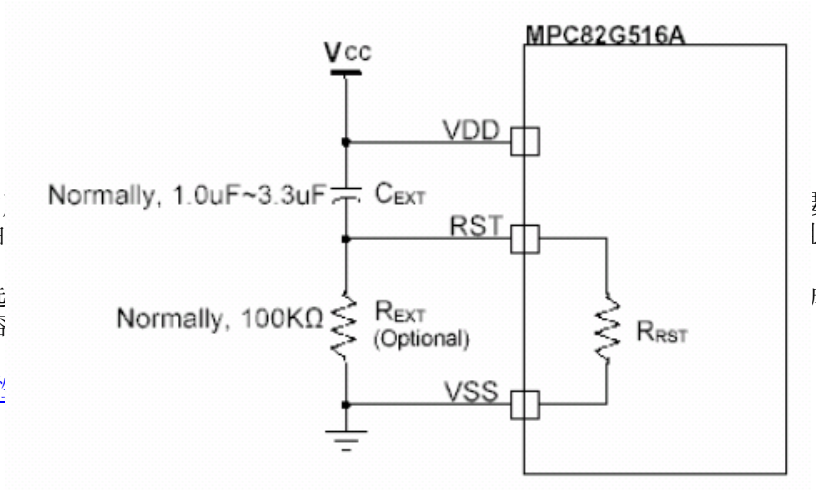
27.2 复位电路

通常，上电时能够成功 要有外部复位。图27-4 示意了外部复位电路，由 R_{EXT} 所组成。

一般来说，R_{EXT} 是可选 成电阻允许上电时只用 一个接到VDD的外部电容

参看 章节 30: 直流特性

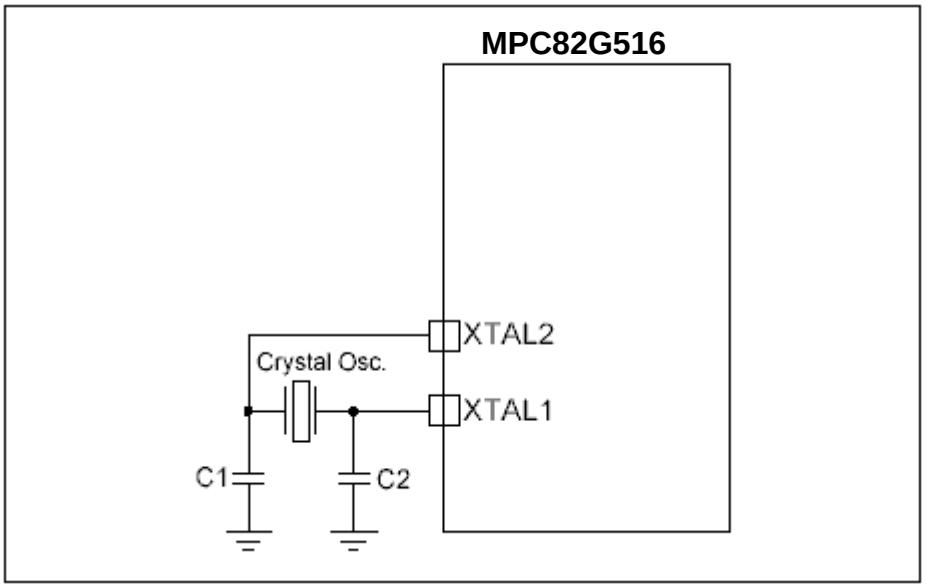
图 27-4. 复位电路



27.3 晶振电路

为了成功获得精确的振荡（至24MHz），电容C1和C2是必需的，无论硬件选项OSCDN的状态（enabled 或 disabled）。通常，C1和C2可以选取相同的值，大约20pF～150pF范围内。

图 27-5. 晶振电路



28 片上调试功能

MPC82G516A为在线仿真（In-Circuit Emulator，ICE）设计有专门的片上调试（On-Chip Debug，OCD）接口。OCD接口提供片上和在系统的无干扰调试，不占用任何目标系统资源。ICE的常用必备操作均支持，如复位，运行，停止，单步，运行到光标，断点设置。

Megawin提供使用OCD技术的在线调试工具“Megawin 8051 OCD ICE”给用户，如图28-1。

1.用户开发时不必准备任何开发板或者传统的ICE探针插座转接器。用户所需的仅仅是在系统上为OCD接口保留一个4引脚的接插件：*VCC*, *OCD_SDA*, *OCD_SCL* 和*GND*。图28-2展示了OCD ICE的系统框图。

另外，最强大的特点是它能直接用*Keil 8051 IDESoftware*中的*dScope-Debugger*功能选项连接到用户的目标系统进行调试。当然，你得要用*Keil 8051 IDE*软件来调试。

注:*“Keil”* 是 *“Keil Elektronik GmbH and Keil Software, Inc.”*的注册商标，*“Keil 8051 IDE software”* 是*8051*嵌入式系统开发中最常用的 *C51* 编译器。

特点

Megawin 专利 OCD (On-Chip-Debug) 技术

单芯片在系统实时调试

2引脚专用串行OCD接口，不占用目标系统资源

直接连接到*Keil 8051 IDE Software*的调试功能

USB连接目标板和上位机(PC)

常用调试功能：复位，运行，停止，单步和运行到光标

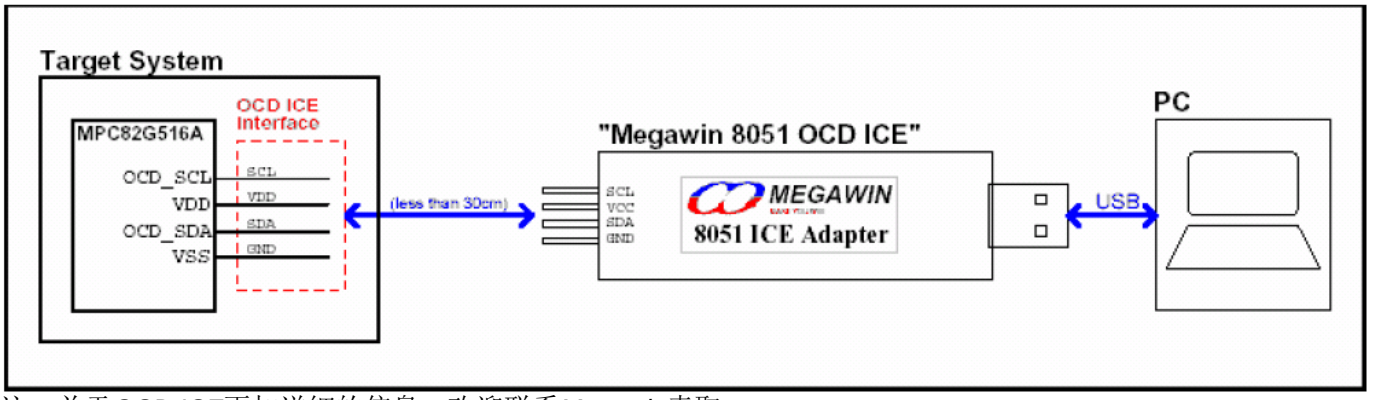
可编程断点，最多可同时设置4个断点

多种常用调试窗口：寄存器、反汇编、Watch、内存窗口

源级（汇编或C语言）调试能力

图 28-1. “8051 ICE Adapter”的图片

图 28-2. ICE功能的结构框图



注：关于*OCD ICE*更加详细的信息，欢迎联系*Megawin*索取。

29 极限参数

参数	范围	单位
工作温度 *注4	-40 ~ +85	℃
存储温度	-55 ~ +125	℃
VDD 对 VSS 电压	-0.5 ~ +6.5	V
其他引脚对 VSS 电压	-0.5 ~ VDD+0.5	V
单输出最大 I_{OL}/I_{OH} *注5	20	mA
所有输出最大 I_{OL}/I_{OH} *注5	100	mA
消耗功率 *注6	1.5	W

注:

1. 器件超过“极限参数范围”可能导致永久性损坏。工作或者存储超出这个范围是不推荐的，且可能影响器件的可靠性。
2. 本产品包括专门为保护其内部设备不被静电损坏的电路。虽然如此，还是建议应采取常规的预防措施，以避免应用时超过额定最大值。
3. 参数是有效的工作温度范围内，另有指明的除外。
4. 样品测试
5. 在稳态（非瞬态）条件，I_{OL}/I_{OH} 必须外部限制。.
6. 基于封装的传热限制，不是器件的能量消耗

30 直流特性

[条件1] 3.3V系统

F_{OSC}=12MHz, T_{amb}=-40℃~+85℃, V_{DD}=2.4V~3.6V, 除非另外说明

符号	参数	测试条件	Min	Typ*[1]	Max	Unit
I_{IL1}	逻辑 0 输入电流, P0/P1/P2/P3/P4 (准双向)	V_{DD}=3.6V and V_{IN}=0.4V	-	-	-10	⌈A
I_{IL2}	逻辑 0 输入电流, P0/P1/P2/P3/P4 (*仅输入*)	V_{DD}=3.6V and V_{IN}=0.4V	-	0	-	⌈A
I_{TL} *2	逻辑 1至0 过度电流, P0/P1/P2/P3/P4 (准双向)	V_{DD}=3.6V and V_{IN}=1.5V	-	-	-120	⌈A
V_{IH1}	输入高电压, P0/P1/P2/P3/P4 (*准双向或仅输入*)		0.3V_{DD} +0.5	-	V_{DD} +0.5	V
V_{IH2}	输入高电压, RST		0.25V_{DD} +0.5	-	V_{DD} +0.5	V
V_{IH3}	输入高电压, XTAL1		0.4V_{DD}	-	V_{DD} +0.5	V
V_{IL1}	输入低电压, P0/P1/P2/P3/P4 (*准双向或仅输入*)		-0.5	-	0.3V_{DD}	V
V_{IL2}	输入低电压, RST		-0.5	-	0.3V_{DD}	V
V_{IL3}	输入低电压, XTAL1		-0.5	-	0.35V_{DD}	V
V_{OH1} *3	输出大电流, P0/P1/P2/P3/P4 (*准双向*)	V_{DD}=2.4V and I_{OH}=-17⌈A V_{DD}=3.6V and I_{OH}=-70⌈A	2.0 2.4	-	-	V
V_{OH2} *3	输出大电流, P0/P1/P2/P3/P4 (*推挽输出*)	V_{DD}=2.4V and I_{OH}=-2.1mA V_{DD}=3.6V and I_{OH}=-8.5mA	2.0 2.4	-	-	V
V_{OH3} *3	输出大电流, XTAL2	V_{DD}=2.4V and I_{OH}=-0.9mA V_{DD}=3.6V and I_{OH}=-3.2mA	2.0 2.4	-	-	V
V_{OL1} *3	输出小电流, P0/P1/P2/P3/P4 (*准双向*)	V_{DD}=2.4V and I_{OL}=+7.0mA V_{DD}=3.6V and I_{OL}=+10.2mA	-	-	0.4	V
V_{OL2} *3	输出小电流, P0/P1/P2/P3/P4 (*推挽输出*)	V_{DD}=2.4V and I_{OL}=+7.0mA V_{DD}=3.6V and I_{OL}=+10.2mA	-	-	0.4	V
V_{OL3} *3	输出小电流, P0/P1/P2/P3/P4 (*开漏输出*)	V_{DD}=2.4V and I_{OL}=+7.0mA V_{DD}=3.6V and I_{OL}=+10.2mA	-	-	0.4	V
V_{OL4} *3	输出小电流, XTAL2	V_{DD}=2.4V and I_{OL}=+1.4mA V_{DD}=3.6V and I_{OL}=+1.6mA	-	-	0.4	V
R_{RST}	内部复位下拉电阻		180	-	320	KΩ
V_{CPF}	欠压阀值, LDO 输出		-	2.4	-	V
V_{RAM} *4	RAM 保持电压		1.2	-	-	V

(接上页)

符号	参数	测试条件	Min	Typ*[1]	Max	Unit
V_{DD} *5,*8	电源电压	F_{osc}=6MHz, T_{amb}= -40℃ T_{amb}= +25℃ T_{amb}= +85℃	 2.4 2.1 1.8	 -	 3.6	V
		F_{osc}=12MHz, T_{amb}= -40℃ T_{amb}= +25℃ T_{amb}= +85℃	 2.4 2.1 1.8	 -	 3.6	
		F_{osc}=24MHz, T_{amb}= -40℃ T_{amb}= +25℃ T_{amb}= +85℃	 2.4 2.3 2.3	 -	 3.6	
V_{POR}	上电复位电压阀值	T_{amb}= -40℃ T_{amb}= +25℃ T_{amb}= +85℃	-	2.4 2.1 1.8	-	V
I_{DD} *6	工作时，电源电流*7	V_{DD}=2.4V F_{osc}=6MHz F_{osc}=12MHz F_{osc}=24MHz	-	4.0 6.2 11.0	5.0 7.5 13.5	mA
		V_{DD}=3.6V F_{osc}=6MHz F_{osc}=12MHz F_{osc}=24MHz	-	9.1 13.0 20.8	11.0 15.5 25.0	
	闲置时，电源电流	V_{DD}=2.4V F_{osc}=6MHz F_{osc}=12MHz F_{osc}=24MHz	-	1.7 2.6 4.5	2.5 3.5 5.5	mA
		V_{DD}=3.6V F_{osc}=6MHz F_{osc}=12MHz F_{osc}=24MHz	-	3.1 4.8 8.2	4.0 6.0 10.0	
	关断时，电源电流	F_{osc}=24MHz, V_{DD}=2.4V~3.6V	-	1	10	⌈A

注:

*1: 典型值是根据数目有限的样品，并没有保证。所列的值是在室温下，另有指明的除外。
*2: 用准双向模式，当外部驱动从逻辑1到逻辑0时端口引脚通过的电流。当V_{IN} 大约为1.5V时电流值最大。
*3: 参考 章节 28: 极限参数范围 中稳态（非瞬态）的I_{OH} 和I_{OL}的限定
如果 I_{OH} 超出测试条件, V_{OH} 将会比列出的数值低.
如果 I_{OL} 超出测试条件, V_{OL} 将会比列出的数值高
*4: RAM数据保存时的电源电压.
*5: 硬件选项 ENLVRO & ENLVRC disabled.
*6: 硬件选项 OSCDN enabled.
*7: 当CPU运行一条空指令循环，如下.

```
Loop: NOP
      JMP   Loop
```

*8: 列出的最低限度电源电压供给仅包括低于正常工作下的逻辑功能，但不包括闪存擦除和写入。2.4V~3.6V范围是整个芯片功能正常工作的电源电压。

[条件2]　5V或宽电压范围

F_{OSC}=12MHz, T_{amb}=-40℃~+85℃, V_{DD}=2.7V~5.5V, 除非特别说明

符号	参数	测试条件	Min	Typ*[1]	Max	Unit
I_{IL1}	逻辑 0 输入电流, P0/P1/P2/P3/P4 (准双向)	V_{DD}=5.5V and V_{IN}=0.4V	-	-	-30	⌈A
I_{IL2}	逻辑 0 输入电流, P0/P1/P2/P3/P4 (*仅输入*)	V_{DD}=5.5V and V_{IN}=0.4V	-	0	-	⌈A
I_{TL} *2	逻辑 1至0 过度电流, P0/P1/P2/P3/P4 (准双向)	V_{DD}=5.5V and V_{IN}=2.0V	-	-	-260	⌈A
V_{IH1}	输入高电压, P0/P1/P2/P3/P4 (*准双向或仅输入*)		0.25V_{DD} +0.7	-	V_{DD} +0.5	V
V_{IH2}	输入高电压, RST		0.25V_{DD} +0.5	-	V_{DD} +0.5	V
V_{IH3}	输入高电压, XTAL1		0.5V_{DD} -0.3	-	V_{DD} +0.5	V
V_{IL1}	输入低电压, P0/P1/P2/P3/P4 (*准双向或仅输入*)		-0.5	-	0.25V_{DD} +0.1	V
V_{IL2}	输入低电压, RST		-0.5	-	0.25V_{DD} +0.2	V
V_{IL3}	输入低电压, XTAL1		-0.5	-	0.35V_{DD} -0.1	V
V_{OH1} *3	输出大电流, P0/P1/P2/P3/P4 (*准双向*)	V_{DD}=2.7V and I_{OH}=-17⌈A V_{DD}=5.5V and I_{OH}=-210⌈A	2.4	-	-	V
V_{OH2} *3	输出大电流, P0/P1/P2/P3/P4 (*推挽输出*)	V_{DD}=2.7V and I_{OH}=-2.1mA V_{DD}=5.5V and I_{OH}=-25.0mA	2.4	-	-	V
V_{OH3} *3	输出大电流, XTAL2	V_{DD}=2.7V and I_{OH}=-0.9mA V_{DD}=5.5V and I_{OH}=-9.4mA	2.4	-	-	V
V_{OL1} *3	输出小电流, P0/P1/P2/P3/P4 (*准双向*)	V_{DD}=2.7V and I_{OL}=+11.8mA V_{DD}=5.5V and I_{OL}=+17.6mA	-	-	0.4	V
V_{OL2} *3	输出小电流, P0/P1/P2/P3/P4 (*推挽输出*)	V_{DD}=2.7V and I_{OL}=+11.8mA V_{DD}=5.5V and I_{OL}=+17.6mA	-	-	0.4	V
V_{OL3} *3	输出小电流, P0/P1/P2/P3/P4 (*开漏输出*)	V_{DD}=2.7V and I_{OL}=+11.8mA V_{DD}=5.5V and I_{OL}=+17.6mA	-	-	0.4	V
V_{OL4} *3	输出小电流, XTAL2	V_{DD}=2.7V and I_{OL}=+1.4mA V_{DD}=5.5V and I_{OL}=+1.6mA	-	-	0.4	V
R_{RST}	内部复位下拉电阻		110	-	280	KΩ
V_{OPF}	欠压阀值, LDO 输出		-	3.7	-	V
V_{RAM} *4	RAM 保持电压		1.5	-	-	V

(接上页)

Symbol	Parameter	Test Conditions	Min	Typ*[1]	Max	Unit
V_{DD} *5, *8	电源电压	F_{osc}=6MHz, T_{amb}= -40℃ T_{amb}= +25℃ T_{amb}= +85℃	 2.4 2.1 1.8	 -	 5.5	V
		F_{osc}=12MHz, T_{amb}= -40℃ T_{amb}= +25℃ T_{amb}= +85℃	 2.4 2.2 1.9	 -	 5.5	
		F_{osc}=24MHz, T_{amb}= -40℃ T_{amb}= +25℃	 2.4 2.4	 -	 5.5	
V_{POR}	上电复位电压阀值	T_{amb}= -40℃ T_{amb}= +25℃ T_{amb}= +85℃	-	2.4 2.1 1.8	-	V
I_{DD} *6	工作时，电源电流*7	V_{DD}=2.7V F_{osc}=6MHz F_{osc}=12MHz F_{osc}=24MHz	-	 4.7 6.9 11.0	 6.0 8.5 13.5	mA
		V_{DD}=5.5V F_{osc}=6MHz F_{osc}=12MHz F_{osc}=24MHz	-	 8.9 11.6 18.1	 11.0 14.0 22.0	
	闲置时，电源电流	V_{DD}=2.7V F_{osc}=6MHz F_{osc}=12MHz F_{osc}=24MHz	-	 1.9 2.9 5.0	 2.5 3.5 6.0	mA
		V_{DD}=5.5V F_{osc}=6MHz F_{osc}=12MHz F_{osc}=24MHz	-	 4.1 5.1 8.1	 5.0 6.5 10.0	
	关断时，电源电流	F_{osc}=24MHz, V_{DD}=2.7V~5.5V	-	1	10	⌈A

注:

*1: 典型值是根据数目有限的样品，并没有保证。所列的值是在室温下，另有指明的除外。
*2: 用准双向模式，当外部驱动从逻辑1到逻辑0时端口引脚通过的电流。当V_{IN} 大约为2.0V时电流值最大。
*3: 参考 章节 28: 极限参数范围 中稳态（非瞬态）的I_{OH} 和I_{OL}的限定
如果 I_{OH} 超出测试条件, V_{OH} 将会比列出的数值低.
如果 I_{OL} 超出测试条件, V_{OL} 将会比列出的数值高
*4: RAM数据保存时的电源电压.
*5: 硬件选项 ENLVRO & ENLVRC disabled.
*6: 硬件选项 OSCDN enabled.
*7: 当CPU运行一条空指令循环，如下.

```
Loop: NOP
      JMP   Loop
```

*8: 列出的最低限度电源电压供给仅包括低于正常工作下的逻辑功能，但不包括闪存擦除和写入。2.7V~5.5V范围是整个芯片功能正常工作的电源电压。

31 订货信息

零 件 号 码	封　　装		包 装
	名称	描述	
MPC82G516AE	PDIP-40	塑料双列直插封装 40 引脚 (600 mil)	Tube管
MPC82G516AP	PLCC-44	Plastic Leaded Chip Carrier; 44 引脚	Tube管
MPC82G516AF	PQFP-44	塑料四面扁平封装 44 引脚 外形 10x10x2.0 mm	Tray 盘
MPC82G516AD	LQFP-48	塑料超低四面扁平封装48 引脚 外形 7x7x1.4 mm	Tray 盘

零件号码示意

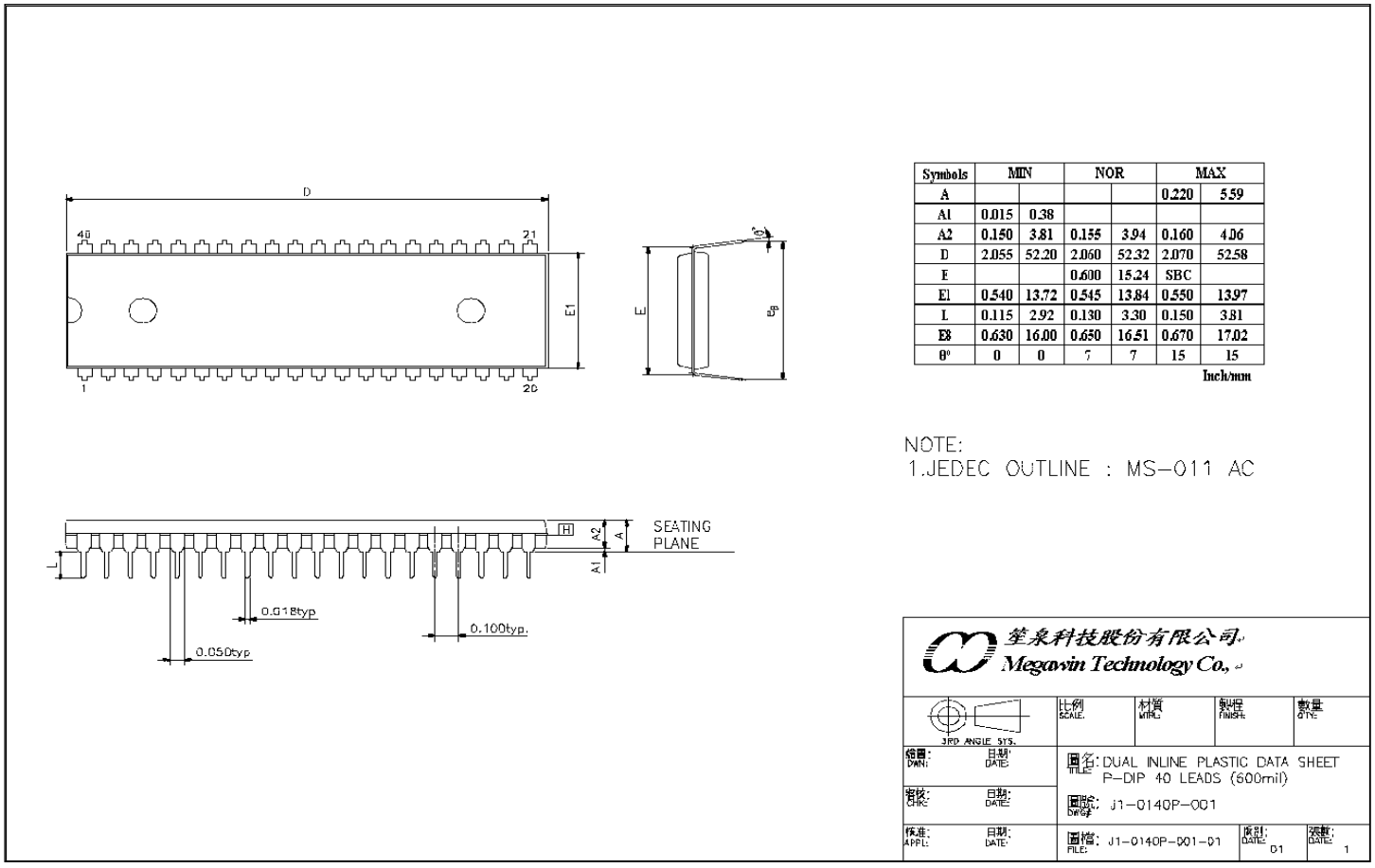
MPC82G516AE

Main Number

封装方式
E: PDIP-40
P: PLCC-44
F: PQFP-44
D: LQFP-48

32 封装尺寸

40-pin PDIP 封装



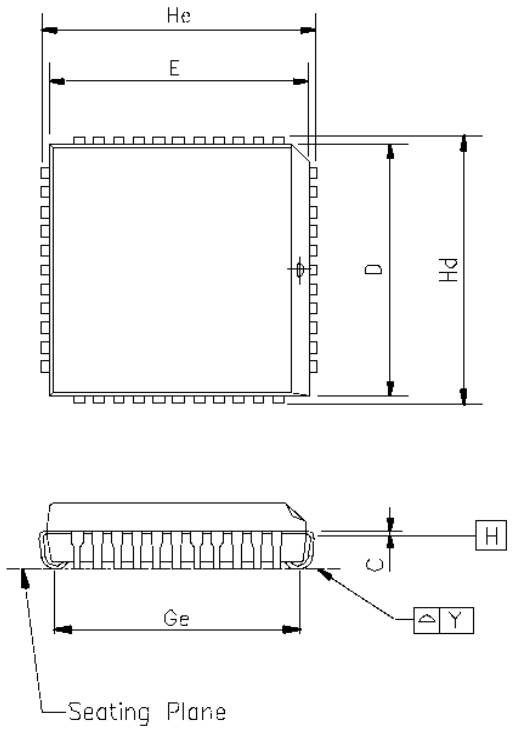
Symbols	MIN		NOR		MAX	
A					0.220	5.59
A1	0.015	0.38				
A2	0.150	3.81	0.155	3.94	0.160	4.06
D	2.055	52.20	2.060	52.32	2.070	52.58
E			0.600	15.24	SBC	
E1	0.540	13.72	0.545	13.84	0.550	13.97
L	0.115	2.92	0.130	3.30	0.150	3.81
eB	0.630	16.00	0.650	16.51	0.670	17.02
θ°	0	0	7	7	15	15

Inch/mm

NOTE:
1.JEDEC OUTLINE : MS-011 AC

3RD ANGLE SYS.	比例 SCALE:	材質 MTRL:	製作 FINISH:	數量 QTY:
繪圖: DWN: 日期: DATE:	圖名: TITLE: DUAL INLINE PLASTIC DATA SHEET P-DIP 40 LEADS (600mil)			
審核: CHK: 日期: DATE:	圖號: DWG#: J1-0140P-001			
核准: APPL: 日期: DATE:	檔名: FILE: J1-0140P-001-D1	版別: DATE: D1	張數: DATE: 1	

44-Pin PLCC Package



Symbols	MIN		NOR		MAX	
A					0.185	4.70
A1	0.020	0.51				
A2	0.145	3.68	0.150	3.81	0.155	3.94
b1	0.026	0.66	0.028	0.71	0.032	0.81
b	0.016	0.41	0.018	0.46	0.022	0.56
c	0.007	0.18	0.010	0.25	0.013	0.33
D	0.648	16.46	0.653	16.59	0.658	16.71
E	0.648	16.46	0.653	16.59	0.658	16.71
e			0.050	1.27	BSC	
Gd	0.590	14.99	0.610	15.49	0.630	16.00
Ge	0.590	14.99	0.610	15.49	0.630	16.00
Hd	0.680	17.27	0.690	17.53	0.700	17.78
He	0.680	17.27	0.690	17.53	0.700	17.78
L	0.090	2.29	0.100	2.54	0.110	2.79
Y					0.004	0.10

Inch/mm

* *NOTE:*

1.JEDEC OUTLINE : MO-047 AC

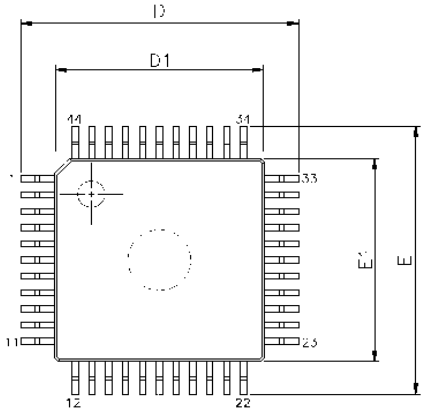
2.DATUM PLANE H IS LOCATED AT THE BOTTOM OF THE MOLD PARTING LINE COINCIDENT WITH WHERE THE LEAD EXITS THE BODY

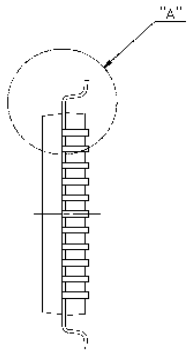
3.DIMENSIONS E AND D DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 10 MIL PER SIDE. DIMENSIONS E AND D DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE H.

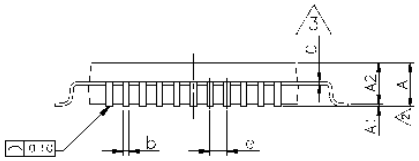
4.DIMENSION b1 DOES NOT INCLUDE DAMBAR PROTRUSION .

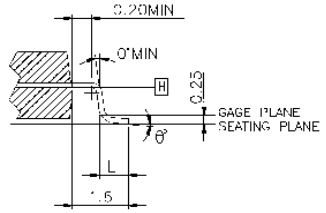
笙泉科技股份有限公司 Megawin Technology Co.,			
3RD ANGLE SYS	比例 SCALE	材質 MTRL	製程 FINISH	數量 QTY
設計 DWN 日期 DATE	圖名: TITLE: PLASTIC CHIP CARRIER DATA SHEET 44 LEADS 0.05" LEAD SPACING SQUARE			
審核: CHK 日期: DATE	圖號: DWG#: J1-05044-001			
核准: APPD 日期 DATE	檔名: FILE: J-044P-1	版次: REV: 02	張次: QTY: 1	

44-Pin PQFP 封装









DETAIL A

Symbols	MIN		NOR		MAX	
A					0.106	2.70
A1	0.010	0.25	0.012	0.30	0.014	0.35
A2	0.075	1.90	0.079	2.00	0.087	2.20
b			0.012	0.30	(TYP.)	
D	0.512	13.00	0.520	13.20	0.528	13.4
D1	0.390	9.90	0.394	10.00	0.398	10.10
E	0.512	13.00	0.520	13.20	0.528	13.40
E1	0.390	9.90	0.394	10.00	0.398	10.10
L	0.029	0.73	0.035	0.88	0.037	0.93
e			0.031	0.80	(TYP.)	
θ°	0	0			7	7
C	0.004	0.10	0.006	0.15	0.008	0.2

Inch/mm

NOTES:

1.JEDEC OUTLINE:MO-108 AA-1

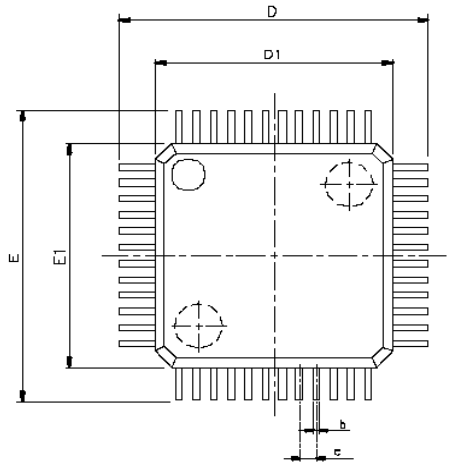
2.DATUM PLANE H IS LOCATED AT THE BOTTOM OF THE MOLD PARTING LINE COINCIDENT WITH WHERE THE LEAD EXITS THE BODY.

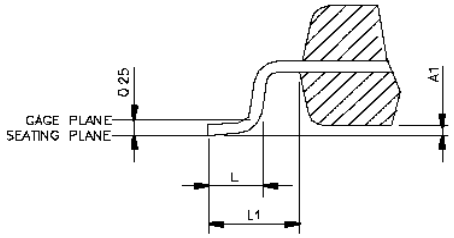
3.DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 mm PER SIDE. DIMENSIONS D1 AND E1 DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE H.

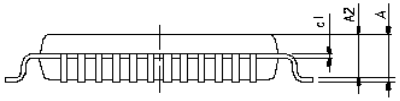
4.DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION.

笙泉科技股份有限公司 Megawin Technology Co.,			
比例 SCALE	材質 MTRL	製程 FINISH	數量 QTY
繪圖: DWN: 日期: DATE:	圖名: TITLE: METRIC PLASTIC QUAD FLAT PACKAGE OUTLINE 10X10MM BODY 44L DATA SHEET		
審核: CHK: 日期: DATE:	圖號: DWG#: J1-0344Q-001		
核定: APPR: 日期: DATE:	圖檔: FILE: J-QFP44	版別: REV: 03	張數: QTY: 1

48-Pin LQFP Package







SYMBOL	MIN		NOR		MAX
A					1.6
A1	0.002	0.05		0.006	0.15
A2	0.053	1.35		0.057	1.45
c1	0.004	0.09		0.006	0.16
D			9.00 BSS		
D1			7.00 BSC		
E			9.00 BSD		
E1			7.00 BSC		
e			0.5 BSC		
b	0.007	0.17		0.011	0.27
L	0.018	0.45		0.030	0.75
L1					

Inch/mm

NOTES:

1. JEDEC OUTLINE: MS-026 BBC
2. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25mm PER SIDE. D1 AND E1 ARE MAXIMUM PLASTIC BODY SIZE DIMENSIONS IMCLUDING MOLD MISMATCH.
3. DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION. ALLDWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED THE MAXIMUM b DIMENSION BY MORE THAN 0.08mm.

笙泉科技股份有限公司
Megawin Technology Co.,

3RD ANGLE SYS.	比例 SCALE	材質 MTRL	製程 FINISH	數量 QTY
審閱 CHK'D: 日期: DATE:	圖名: TITLE: THIN QUAD FLAT PACKAGE OUTLINE 48PIN 1.4MM THICK BODY DATA SHEET			
繪製: DWN: 日期: DATE:	圖號: DWG#: J1-DB48F-D01			
核准: APPL: 日期: DATE:	檔名: FILE: JQFP48-1	版別: REV: 02	張數: QTY: 1	

33 免责声明

此中， Megawin 表示 “***Megawin Technology Co., Ltd.***”

生命维持系统 — 本产品不能设计于医疗，生命救护或生命维持系统的应用，或者系统故障时将导致可预料的人身伤害。客户使用或销售应用于此类应用时的产品，自担风险，且同意完全赔偿 Megawin 任何由于此类不适当的使用或销售导致的损害赔偿。

内容更新 — Megawin 保留修改产品的权利－包括电路，标准单元，以及软件（叙述或包含此中改进设计及性能）。当产品量产时，相关修改将通过工程变更通知（ECN）传达。

34 版本历史

修订	描　　述	日期
V1.00	重新編排版本	2011/03
V1.01	AUXR1 寄存器错误	2011/03
V1.02	补上ICE接口描述	2011/07
V1.03	ICE接口描述遗漏以及修正错误	2011/11
A1.0	重新編排版本	2014/03