MEGAWIN
MAKE YOU WIN

8051 内核-单片机

MA807
说明书

版本: **A1.0**

北京菱电科技有限公司
Tel:010-82674978

2014/02 version A1.0

功能

- 1-T 80C51 中央处理单元
- MA807 有 **14.5KB** 字节 Flash ROM
  - **13.5KB 为 AP 程序代码空间（0x0000 ~ 0x35FF）**
  - **1KB 为 IAP 数据 Flash 空间（0x3600 ~ 0x39FF）**
  - 为 Flash 存储器存取提供代码保护
- 片上 256 字节数据 RAM 和片上 256 字节扩展 RAM
- 双数据指针 DPTR
- 三个 16 位定时/计数器：Timer 0、 Timer 1 和 Timer 2.
  - T0CKO - P34，T1CKO - P35， T2CKO - P10
  - T0/T1/T2 可选择使能 X12 模式
- PWM-定时间器可以作 PWM 发生器或作普通 16 位定时器
  - 最大 8 路 PWM 输出 在 P2 端口
- 增强型 UART
  - 支持帧错误检测 硬件地址识别
  - 在 P16/P17 上功能交换
- 4 个外部中断输入： nINT0/nINT1/nINT2/nINT3
  - nINT0/nINT1 触发类型：低电平或下降沿
  - nINT2/nINT3 触发类型：低电平、下降沿、高电平、或上升沿
- 11 个中断源，四级优先级中断能力
- 内建模拟比较器和片上 VDD 参考输入
  - 比较器正输入有 4 路 I/O 选择
  - I/O 口可编程比较器负输入或 15 级片上 VDD 参考
- 15 位看门狗定时器 Watch-Dog-Timer.
  - 8 位分频
  - 通过 CPU 或上电一次性使能
- 41 个通用输入输出
  - P0, P1, P2, P3, P4 能被配置为准双向口（quasi-bidirectional）、推挽式输出（push-pull）、开漏集输出（open-drain）及仅输入（input only）四种模式
  - P6 只能作准双向口模式
  - P4.0 和 P4.1 能被配置为仅输入或准双向口模式（缺省）
- 电源控制：空闲模式（idle mode）和掉电模式（power-down mode）
  - 所有的中断和 16 个 GPIO 能唤醒空闲模式
  - 4 个外部中断和 16 个 GPIO 能唤醒掉电模式
- Brown-Out 侦察器 4.0V
  - 选择复位 CPU. 或产生 CPU 中断
- 工作电压： 4.5V~5.5V
- 工作频率：
  - 12 MHz 内部 RC 振荡器
    在室溫 @ 25℃下，频漂为+/- 1%,
    在工作溫度@ -20 ~ 50℃範圍內，频漂为+/- 2%,
    在工作溫度@ -40 ~ 85℃範圍內，频漂为+/- 4%,.
  - 内部 RC 振荡器通过 P6.0 输出
- 工作温度：
  - 工业级别 (-40℃ 至 +85℃)*
- 封装类型：

- LQFP44:：MA807AD44

*：样品测试

目录

1. 概述

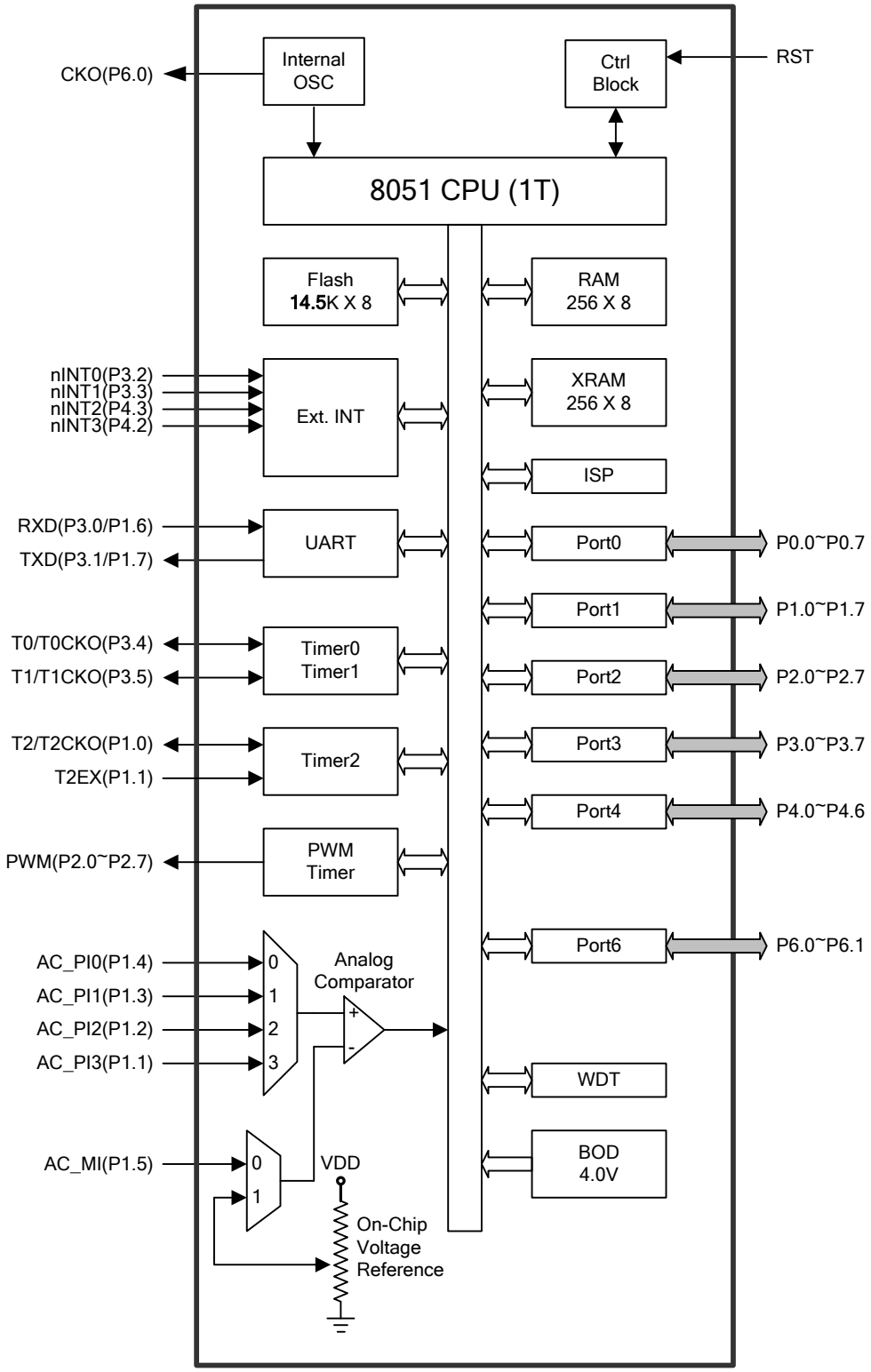
MA807 是基于80C51的高效1-T结构的单芯片微处理器，每条指令需要1~7个时钟信号 (比标准8051快6~7倍)，与8051指令集兼容。因此在与标准8051有同样的处理能力的情况下，MA807只需要非常低的运行速度，同时由此能很大程度的减少耗电量。

MA807 拥有**14.5**K字节的内置Flash存储器用于保存代码AP和数据IAP。Flash存储器可以通过在系统编程(ISP)，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。

MA807 保留了标准80C52的基本特色：256 字节的随机存储器、四个8位I/O口、两个外部中断、一个多源4级中断控制器、及三个定时/计数器。增加的有：MA807 有两个额外的I/O口（P4[6:0]、P6[1:0]），一个256字节的 XRAM，四个额外的能选择高电平或低电平触发的外部中断，一个PWM定时器，一个一次性使能的看门狗WDT，一个精确的多路输入和片上VDD参考电压的模拟比较器，一个Brown-out侦察器，一个高精度内部振荡器，一个为方便多处理器通讯和改进了速度（额外X2/X4模式）的增强型通用串行口（EUART）。

MA807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的内容被保存，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。

2. 方框图


CKO(P6.0)
Internal OSC
Ctrl Block
RST
8051 CPU (1T)
Flash 14.5K X 8
RAM 256 X 8
nINT0(P3.2)
nINT1(P3.3)
nINT2(P4.3)
nINT3(P4.2)
Ext. INT
XRAM 256 X 8
ISP
RXD(P3.0/P1.6)
TXD(P3.1/P1.7)
UART
Port0
P0.0~P0.7
Port1
P1.0~P1.7
T0/T0CKO(P3.4)
T1/T1CKO(P3.5)
Timer0 Timer1
Port2
P2.0~P2.7
T2/T2CKO(P1.0)
T2EX(P1.1)
Timer2
Port3
P3.0~P3.7
Port4
P4.0~P4.6
PWM(P2.0~P2.7)
PWM Timer
Port6
P6.0~P6.1
AC_PI0(P1.4)
AC_PI1(P1.3)
AC_PI2(P1.2)
AC_PI3(P1.1)
0
1
2
3
Analog Comparator
+
-
WDT
AC_MI(P1.5)
0
1
VDD
BOD 4.0V
On-Chip Voltage Reference

3. 特殊功能寄存器 SFR

3.1. SFR 映射表

	SFR 页	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8	0 F	---	--	CCAP0H	--	--	--	--	--
F0	0 F	B	--	--	--	--	--	--	--
E8	0 F	P4	--	CCAP0L	--	--	--	--	--
E0	0 F	ACC	WDTCR	IFD	IFADRH	IFADRL	IFMT	SCMD	ISPCR
D8	**0** **F**	**CCON** ---	CMOD	--	--	--	--	--	--
D0	0 F	PSW	--	--	--	--	--	GPWKPE	P1WKPE
C8	**0** **F**	**T2CON** **P6**	T2MOD	RCAP2L	RCAP2H	TL2	TH2	--	--
C0	0 F	XIFLG	XICON0	XICON1	--	--	--	--	CKCON0
B8	0 F	IP0L	SADEN	--	--	--	--	--	CKCON1
B0	0 F	P3	P3M0	P3M1	P4M0	P4M1	P5M0	P5M1	IP0H
A8	0 F	IE	SADDR	--	--	**SFRPI**	EIE1	EIP1L	EIP1H
A0	0 F	P2	--	AUXR1	AUXR2	--	--	--	--
98	0 F	SCON	SBUF	--	--	--	--	ACCON	ACMOD
90	0 F	P1	P1M0	P1M1	P0M0	P0M1	P2M0	P2M1	PCON1
88	0 F	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR0	
80	0 F	P0	SP	DPL	DPH	--	--	--	PCON0
		0/8	**1/9**	**2/A**	**3/B**	**4/C**	**5/D**	**6/E**	**7/F**

***: SFR 页需要正确设置相对应的 SFRPI=0x00, SFRPI=0x01 切换，才能正确 SFR 寄存器，**
（进入中断，SFRPI 需要保存当下值，并且设置中断使用的 SFR 页，并且中断返回前回存，MCU 不会自动切换）

3.2. SFR 位分配

符号	描述	地址	位地址和符号								复位值
			Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	
P0	端口 0	80H	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	11111111B
SP	堆栈指针	81H									00000111B
DPL	数据指针低	82H									00000000B
DPH	数据指针高	83H									00000000B
PCON0	电源控制 0	87H	SMOD1	SMOD0	GF	POF	GF1	GF0	PD	IDL	00010000B
TCON	定时器控制	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00000000B
TMOD	定时器模式	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00000000B
TL0	定时器 0 低字节	8AH									00000000B
TL1	定时器 1 低字节	8BH									00000000B
TH0	定时器 0 高字节	8CH									00000000B
TH1	定时器 1 高字节	8DH									00000000B
AUXR0	辅助寄存器 0	8EH	P60OC1	P60OC0	P60FD	P34FD	--	--	EXTRAM	--	0000xx0xB
P1	端口 1	90H	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	11111111B
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	00000000B
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	00000000B
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	00000000B
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	00000000B
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	00000000B
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	00000000B
PCON1	电源控制 1	97H	--	--	BORF	--	--	--	--	BOD	xxxxxxx0B
SCON	串口控制	98H	SM0/FE	SM1	SM2	REN	TB8	RB8	Ti	RI	00000000B
SBUF	串口数据缓冲器	99H									xxxxxxxxB
ACCON	模拟比较器控制	9EH	ACIDX	ACPDX	ACOUT	ACF	ACEN	ACM2	ACM1	ACM0	00x00000B
ACMOD	模式比较器模式	9FH	MVRS3	MVRS2	MVRS1	MVRS0	--	--	PIS1	PIS0	0000xx00B
P2	端口 2	A0H	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	11111111B
AUXR1	辅助寄存器 1	A2H	GPWKS1	GPWKS0	P5PWM	P1S0	GF2	--	--	DPS	00000xx0B
AUXR2	辅助寄存器 2	A3H	T0X12	T1X12	URM0X6	--	--	--	T1CKOE	T0CKOE	0000xx00B
IE	中断使能	A8H	EA	GF4	ET2	ES	ET1	EX1	ET0	EX0	00000000B
SADDR	从机地址	A9H									00000000B
SFRPI	SFR 页索引	ACH	--	--	--	--	IDX3	IDX2	IDX1	IDX0	xxxx0000B
EIE1	外部中断使能 1	ADH	--	--	--	--	--	--	EACI	EBOI	xxxxxx00B
EIP1L	外部中断优先级 1 低	AEH	--	--	--	0	0	PPTL	PACL	PBOL	xxx00000B
EIP1H	外部中断优先级 1 高	AFH	--	--	--	0	0	PPTH	PACH	PBOH	xxx00000B
P3	端口 3	B0H	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	11111111B
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	00000000B
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	00000000B
P4M0	端口 4 模式寄存器 0	B3H	--	P4M0.6	P4M0.5	P4M0.4	P4M0.3	P4M0.2	P4M0.1	P4M0.0	x00000xxB
P4M1	端口 4 模式寄存器 1	B4H	--	P4M1.6	P4M1.5	P4M1.4	P4M1.3	P4M1.2	P4M1.1	P4M1.0	x0000000B
		B5H									00000000B
		B6H									00000000B
IP0H	中断优先级 0 高	B7H	PX3H	PX2H	PT2H	PSH	PT1H	PX1H	PT0H	PX0H	00000000B
IP0L	中断优先级 0 低	B8H	**PX3L**	**PX2L**	PT2L	PSL	PT1L	PX1L	PT0L	PX0L	00000000B
SADEN	从机地址屏蔽	B9H									00000000B
CKCON1	时钟控制 1	BFH	OSCDR	--	--	XCKS4	XCKS3	XCKS2	XCKS1	XCKS0	xxx01010B
XIFLG	外部中断标志	C0H	--	--	--	--	0	0	IE3	IE2	xxxx0000B
XICON0	外部中断控制 0	C1H	--	INT3H	IT3	EX3	--	INT2H	IT2	EX2	x000x000B
XICON1	外部中断控制 1	C2H	--	0	IT5	0	--	0	0	0	x000x000B
CKCON0	时钟控制 0	C7H	--	--	--	--	--	SCKS2	SCKS1	SCKS0	xxxxx000B
T2CON	定时器 2 控制	C8H	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL	00000000B
P6	端口 6	C8H	--	--	--	--	--	--	P6.1	P6.0	xxxxxx11B
T2MOD	定时器 2 模式	C9H	--	--	--	T2X12	--	--	T2OE	DCEN	xxx0xx00B
RCAP2L	定时器 2 捕获 低	CAH									00000000B
RCAP2H	定时器 2 捕获 高	CBH									00000000B
TL2	定时器 2 低字节	CCH									00000000B
TH2	定时器 2 高字节	CDH									00000000B
PSW	程序状态字	D0H	CY	AC	F0	RS1	RS0	OV	F1	P	00000000B
GPWKPE	普通端口唤醒使能	D6H	0	GP6WE	0	GP4WE	GP3WE	GP2WE	GP1WE	GP0WE	00000000B
P1WKPE	端口 1 唤醒使能	D7H	P17WE	P16WE	P15WE	P14WE	P13WE	P12WE	P11WE	P10WE	00000000B
CCON	计数器控制寄存器.	D8H	CF	CR	-	-	PWMEN	-	-	-	00xx0xxxB

		D8H									11111111B
CMOD	计数器模式寄存器	D9H	CIDL	POS2	POS1	POS0	CPS2	CPS1	CPS0	ECF	00000000B
ACC	累加器	E0H	ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0	00000000B
WDTCR	看门狗定时器控制寄存器	E1H	WRF	--	ENW	CLW	WIDL	PS2	PS1	PS0	0x000000B
IFD	ISP Flash 数据	E2H									11111111B
IFADRH	ISP Flash 地址 高	E3H									00000000B
IFADRL	ISP Flash 地址 低	E4H									00000000B
IFMT	ISP 模式选择	E5H	--	--	--	--	MS3	MS2	MS1	MS0	xxxx0000B
IAPLB	IAP 低边界	注 1	IAPLB6	IAPLB5	IAPLB4	IAPLB3	IAPLB2	IAPLB1	IAPLB0	--	00110110B
SCMD	ISP 指令	E6H									xxxxxxxxB
ISPCR	ISP 控制寄存器	E7H	ISPEN	BS	SRST	CFAIL	--	--	--	--	0000xxxxB
P4	端口 4	E8H	--	P4.6	P4.5	P4.4	P4.3	P4.2	P4.1	P4.0	11111111B
CCAP0L	PWM 占空比 低	EAH									00000000B
B	B 寄存器	F0H	F7H	F6H	F5H	F4H	F3H	F2H	F1H	F0H	00000000B
		F8H									11111111B
CCAP0H	PWM 占空比 高	FAH									00000000B

注 1： 这个寄存器地址通过 IFMT 和 SCMD 来决定。详情请见 IFMT 寄存器描述。

3.3. SFR 内存分页

MA807 SFR 内存分页允许设备在 0x80 至 0xFF 存储器地址区域映射更多的 SFR 。SFR 存储器区域有 16 页。因此，从 0x80 至 0xFF 的每一页存储器能存取最大到 128 个 SFR。 MA807 利用两个 SFR 页：0 和 F。利用 SFR 页索引寄存器（SFRPI）去选择 SFR 页。读写 SFR 的流程如下：

1. 利用 SFRPI 寄存器选择 SFR 页号。
2. 用直接寻址方式读或写特殊功能寄存器 SFR（(MOV 指令)。

SFRPI: SFR 页索引寄存器

SFR 地址 = 0xAC

SFR 页 = 全部 复位值 = XXXX-0000

7	6	5	4	3	2	1	0
--	--	--	--	PIDX3	PIDX2	PIDX1	PIDX0
R	**R**	**R**	**R**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~4: 保留给测试。这些位必须写 “0” 。

Bit 3~0: SFR 页索引。可用到的页只有页 “0” 和 “F”。

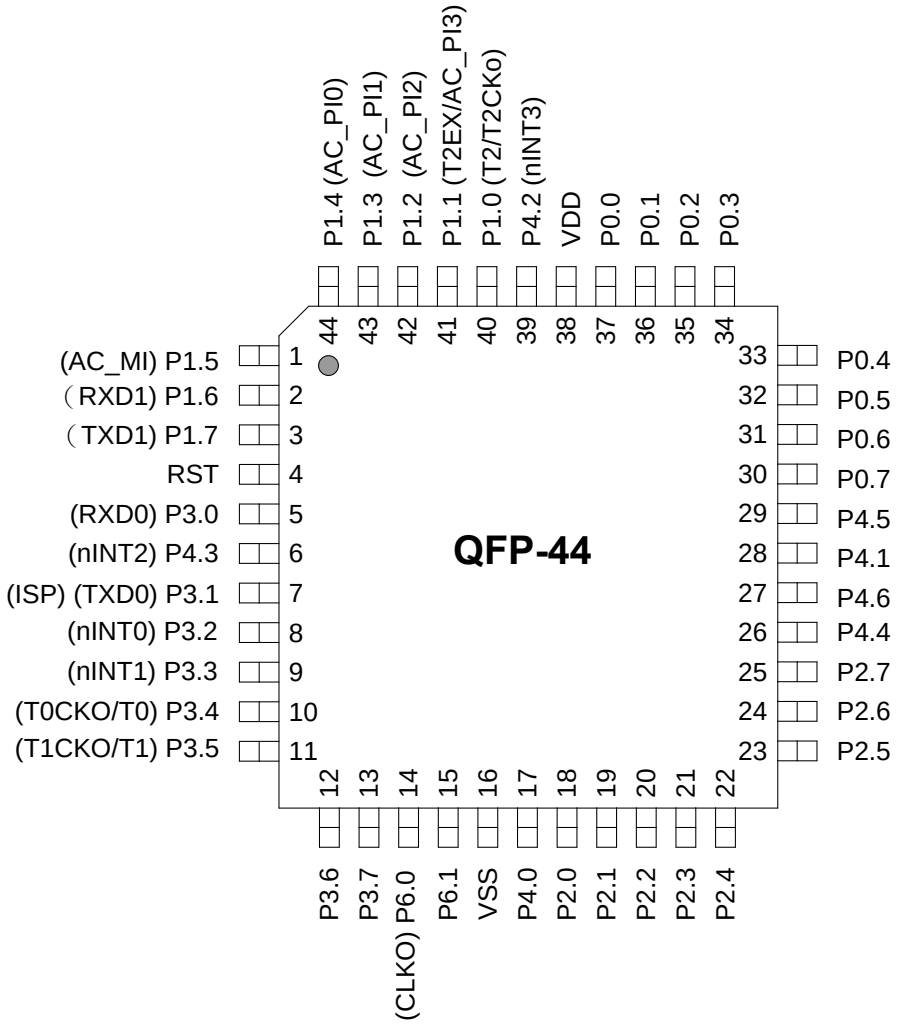
PIDX[3:0]	选择的页
0000	页 0
0001	页 1
0010	页 2
0011	页 3
……	……
……	……
……	……
1111	页 F

有两个寄存只在页 0 中：T2CON(C8H) 和 CCON(D8H)，及两个寄存只在页 F 中：P6(C8H) 。其它的寄存能在页 0 和页 F 两个页中存取。

4. 引脚

4.1. 封装

MA807 LQFP-44

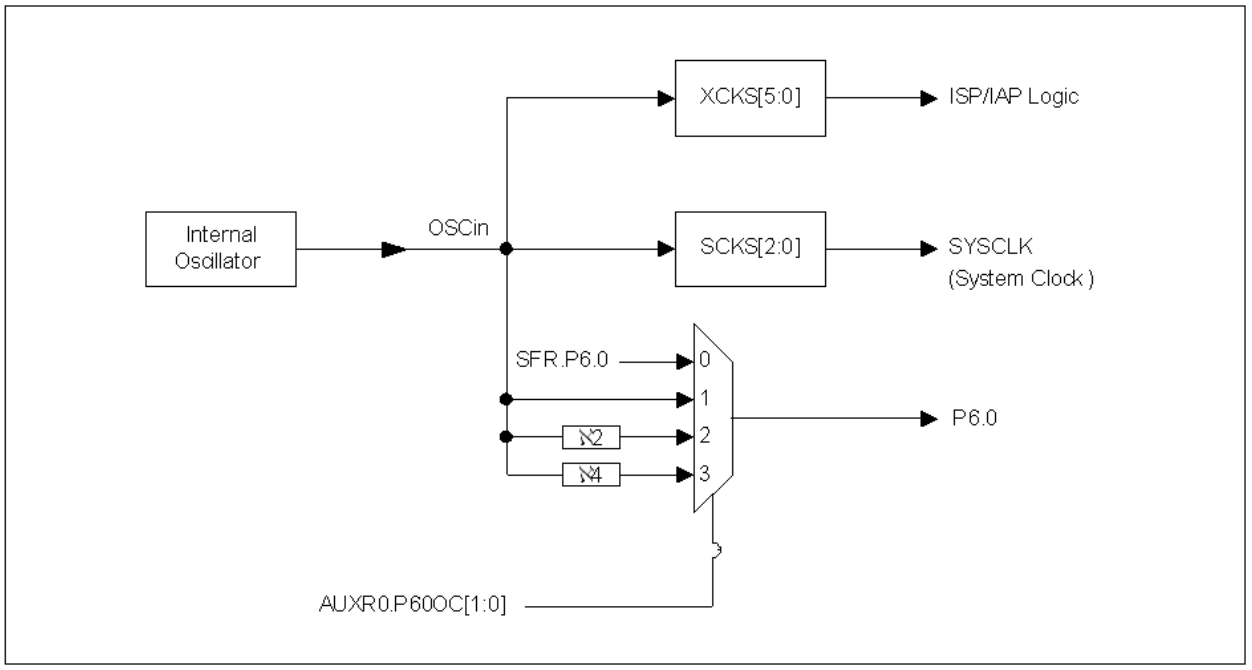


4.2. 引脚定义

助记符	Pin	类型		描述
P1.0 ~ P1.4 P1.5 ~ P1.7	40~44 1~3	I/O I/O		**端口 1**: 普通通用 I/O 口 1. P10 可作 T2 或 T2CKO. P11 可作 T2EX. P11 ~ P14 可作可编程比较器正输入.默认输入口是 P14. P15 可作比较器负输入 P16/P17 可通过固件配置为第二功能 RXD/TXD
P3.0 P3.1 ~ P3.7	5 7~13	I/O I/O		**端口 3**: 普通通用 I/O 口 3. P30 可作串口 RXD. P31 可作串口 TXD. P32 可作 nINT0. P33 可作 nINT1. P34 可作 T0 或 T0CKO. P35 可作 T1 或 T1CKO.
P0.7 ~ P0.0	30~37	I/O		**端口 0**: 普通通用 I/O 口 0.
P2.0 ~ P2.7	18~25	I/O		**端口 2**: 普通通用 I/O 口 2. 可以做 PWM 定时器的 PWM 输出
P4.0 P4.1 P4.2 P4.3 P4.4 P4.5 P4.6	17 28 39 6 26 29 27	I/O I/O I/O I/O I/O I/O I/O		**端口 4**: 普通通用 I/O 口 4. P40 和 P41 能被配置仅输入在默认状态 P42 可作 nINT3. P43 可作 nINT2.
RST	4	I		**RST**: 最少两个机器周期的高电平复位微控制器
P6.0 P6.1	14 15	I/O I/O		**端口 6**: 普通通用 I/O 口 6. P6.0 和内部时钟输出
VDD	38	P		**电源**
VSS	16	G		**地**

5. 系统时钟

5.1. 时钟结构图



5.2. 时钟寄存器

CKCON0: 时钟控制寄存器 0

SFR 地址 = 0xC7

SFR 页 = 全部 复位值 = xxxx-x000

7	6	5	4	3	2	1	0
-	-	-	-	-	SCKS2	SCKS1	SCKS0
R	**R**	**R**	**R**	**R**	**R/W**	**R/W**	**R/W**

Bit 7~3: 保留.

Bit 2~0: SCKS2 ~ SCKS0, 可编程系统时钟选择

SCKS[2:0]	系统时钟 (Fosc)
0 0 0	CLKin
0 0 1	CLKin /2
0 1 0	CLKin /4
0 1 1	CLKin /8
1 0 0	CLKin /16
1 0 1	CLKin /32
1 1 0	CLKin /64
1 1 1	CLKin /128

CKCON1: 时钟控制寄存器 1

SFR 地址 = 0xBF

SFR 页 = 全部 复位值= xxx0-1010

7	6	5	4	3	2	1	0
OSCDR	--	--	XCKS4	XCKS3	XCKS2	XCKS1	XCKS0
R/W	**R**	**R**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: OSCDR, OSC 驱动控制寄存器。
0: 足够驱动能力,驱动晶体振荡器。
1: 减少晶体振荡器的驱动能力。可以帮助减少 EMI。

Bit 6~5: 保留.

Bit 4~0: 设置振荡频率值去定义 ISP/IAP 操作的基本时间。根据 OSCin 填充一个适当的值，如下表：

[XCKS4~XCKS0] = OSCin – 1，当 *OSCin=1~25 (MHz)时。*

例如：
(1) 如果 OSCin=12MHz，则填充 [XCKS4~XCKS0]为 11，即：001011B.
(2) 如果 OSCin=6MHz，则填充 [XCKS4~XCKS0]为 5，即：000101B.

OSCin	XCKS[4:0]
1MHz	5'b00000
2MHz	5'b00001
3MHz	5'b00010
4MHz	5'b00011
……	……
22MHz	5'b10101
23MHz	5'b10110
24MHz	5'b10111
25MHz	5'b11000

默认值 XCKS= 5'b01010

AUXR0: 辅助寄存器 0
SFR 地址 = 0x8E
SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P60OC1	P60OC0	P60FD	P34FD	--	--	--	--
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6: P60 输出配置控制位 1 和 0。P60 支持下面的选择为 GPIO 或时钟源发生器。当 P60OC[1:0] 索引为非 P60 功能时，XTAL2 将驱动片上 RC 振荡器输出为其它设备提供时钟源。

P60OC[1:0]	XTAL2 功能	I/O 模式
00	P60	准双向口
01	INTOSC	推挽输出
10	INTOSC/2	推挽输出
11	INTOSC/4	推挽输出

Bit 5: P60FD, P6.0 快速驱动
0: P6.0 默认驱动输出
1: P6.0 快速驱动输出使能。如果 P6.0 被配置作时钟输出。

5.3. 时钟示例代码

(1). 功能需求: 在空闲模式下 SYSCLK 切换到 OSCin/128 (缺省值是 OSCin/1)

汇编语言代码范例:

```
CKS0            EQU         01h
CKS1            EQU         02h
CKS2            EQU         04h

    ORL         PCON2,#( CKS2 + CKS1 + CKS0)    ; 设置 CKS[2:0] = “111” 选择  OSCin/128
```

C 汇编语言代码范例:

```
#define     CKS0                0x01
#define     CKS1                0x02
#define     CKS2                0x04

    PCON2 &= (CKS2 | CKS1 | CKS0);          // 系统时钟分频  /128
                                            // CKS[2:0], 系统时钟分频
                                            //      0       | OSCin/1
                                            //      1       | OSCin/2
                                            //      2       | OSCin/4
                                            //      3       | OSCin/8
                                            //      4       | OSCin/16
                                            //      5       | OSCin/32
                                            //      6       | OSCin/64
                                            //      7       | OSCin/128
```

6. 8051 CPU 功能描述

6.1. CPU 寄存器

PSW: 程序状态字

SFR 地址 = 0xD0

SFR 页 = 全部 复位值 = 0000-0000

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

CY：进位标志。有进位/借位时 CY=1，否则 CY=0。

AC：半进位标志。当 D3 位向 D4 位产生进位/借位时 AC=1，否则 AC=0，常用于十进制调整运算中。

F0：用户可设定的标志位 0，可置位/复位，也可供测试。

RS1、RS0：四个通用寄存器组的选择位，该两位的四种组合状态用来选择 0~3 寄存器组。见下表：

RS1	RS0	工作寄存器组
0	0	0 组（00-07H）
0	1	1 组（08-0FH）
1	0	2 组（10-17H）
1	1	3 组（18-1FH）

OV：溢出标志。当带符号数运算结果超出 -128~+127 范围时 OV=1，否则 OV=0。当无符号数乘法结果超过 255 时，或当无符号数除法的除数为 0 时，OV=1，否则 OV=0。

F1：用户可设定的标志 1，可置位/复位，也可供测试。

P：奇偶校验标志。每条指令执行完，若 A 中 1 的个数为奇数时 P=1，否则 P=0，即奇偶校验方式。

SP: 堆栈指针

SFR 地址 = 0x81

SFR 页 = 全部 复位值 = 0000-0111

7	6	5	4	3	2	1	0
SP[7]	SP[6]	SP[5]	SP[4]	SP[3]	SP[2]	SP[1]	SP[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

DPL: 数据指针低字节

SFR 地址 = 0x82

SFR 页 = 全部 复位值 = 0000-0000

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

DPH: 数据指针高字节

SFR 地址 = 0x83

SFR 页 = 全部 复位值 = 0000-0000

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

B: B 寄存器

SFR 地址 = 0xF0

SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
B[7]	B[6]	B[5]	B[4]	B[3]	B[2]	B[1]	B[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

6.2. CPU 时序

MA807 是一个基于1-T结构80C51 CPU的单芯片维处理器，指令完全兼容8051，执行一条指令只要 1~7时钟周期(是标准8051的 6~7倍)。它通过管道结构加快了指令执行速度超过标准8051结构。指令时序不同于标准8051。

多数8051执行指令的差别是基于机器周期和时钟周期的，机器周期为2到12个时钟周期。然而，1T-80C51是基于时钟周期时序的。 所有指令时序被指定为时钟周期期间。关于1T-80C51指令的更多详情（包括每条指令的助记符、字节数、时钟周期）请参考“指令集”。

6.3. CPU 寻址方式

直接寻址

在指令中直接给出操作数地址的就属于直接寻址。此时，指令中的操作数部分就是操作数的地址。例如指令：

```
MOV     A,4FH                   ;(A)←(4FH)
```

可用于直接寻址的空间是，内部数据RAM的低128字节及特殊功能寄存器SFRs。

寄存器间接寻址

由指令中指出某一个寄存器的内容作为操作数的地址。内部RAM和外部RAM都能通过间接寻址方式进行访问。使用当前工作寄存器组中的R0或R1存放操作数单元的地址指针（8位地址），在执行PUSH（压栈）和POP（出栈）指令时采用堆栈指针SP作寄存间接寻址。而如果地址是16位时就只能使用DPTR数据指针作间接寻址了。例如指令：

```
MOV     A,@R0                   ;(A) ←((R0))
MOVX    A,@R1                   ;(A) ←((R1))
MOVX    A,@DPTR                 ;(A) ←((DPTR))
```

寄存器寻址(REG)

寄存器寻址就是以通过寄存器的内容作为操作数。在指令的助记符号中直接以寄存器的名字来表示操作数的地址。例如指令：

```
MOV     A,R0            ;(A) ←(R0)
ADD     A,R0            ;(A) ←(Acc)+(R0)
```

能用于这种寻址方式的寄存器还有ACC、B、DPTR、AB(双字节)和CY(位累加器)。

变址寻址

以某个寄存器的内容作为基本地址，然后在这个基本地址基础上加上地址偏移量才是真正的操作数地址。例如指令：

```
MOVC    A,@A+DPTR               ;(A) ←((A)+(DPTR))
```

不论用DPTR或是PC作为基址指针，变址寻址方式都只适用于8051的程序存储器，通常用于读取数据表。

立即寻址

指令中地址码部分给出的就是操作数。即取出指令的同时立即得到了操作数。例如指令：

```
MOV     A,#4FH                  ;(A) ←6FH
```

相对寻址

相对寻址时，由程序计数器PC提供的基地址与指令中提供的偏移量rel相加，得到操作数的地址。这时指出的地址是操作数与现行指令的相对位置。例如指令：

```
SJMP    rel                     ;PC←(PC)+2+rel
```

位寻址

操作数是二进制数的某一位，其位地址出现在指令中，例如指令：

```
SETB    bit                     ;(bit) ←1
```

7. 存储器结构

像所有的 80C51一样， MA807 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(AP)最大13.5K字节。在MA807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。

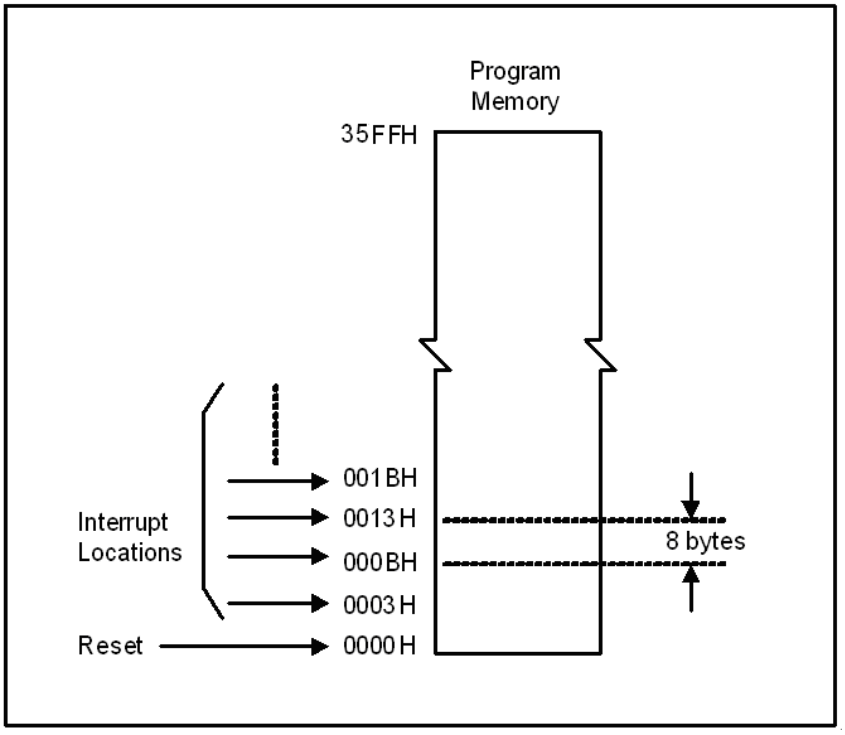
数据存储器使用与程序存储器不同的地址空间。MA807 有256字节的内部RAM和256字节片上扩展存储器 (XRAM)。

7.1. 片上程序存储器

程序存储器用来保存让CPU进行处理的程序代码，如图 8-1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。

图 8-1 程序存储器



7.2. 片上数据存储器 RAM

图 8-2 向MA807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。

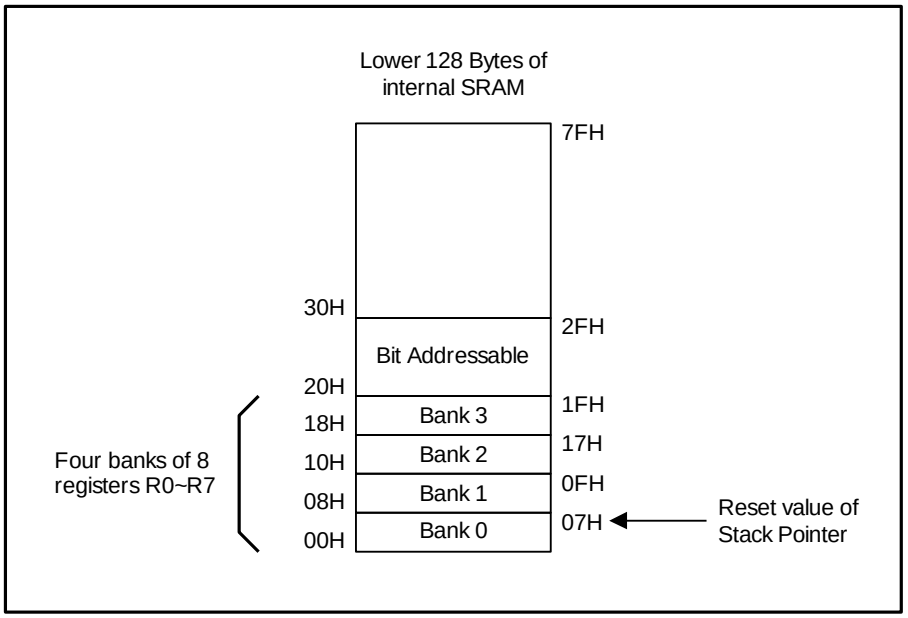
图 8-2 所示，低128字节RAM与所有80C51 一样。最低的32字节被划分为4组每组8字节的寄存器组。指令中称这些寄存器为R0到R7。程序状态字 (PSW) 中的两位用于选择哪组寄存器被使用。这使得程序空间能够被更有效的使用，因为对寄存器访问的指令比使用直接地址的指令短。 接下来的16字节是可以位寻址的存储器空间。80C51的指令集包含一个位操作指令集，这区域中的128位可以被这些指令直接使用。位地址从00H开始到7FH结束。

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。

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。

存取片上扩展 RAM (XRAM)，256 字节的 XRAM (0000H ~ 00FFH) 通过指令 MOVX 间接寻址。存取 XRAM 没有任何关于地址，地址锁存器和读/写选通信号的输出。

图 8-2 低 128 字节内部 RAM



7.3. 片上扩展 RAM (XRAM)

存取片上 RAM (XRAM)， 256 字节的 XRAM (0000H ~ 00FFH) 通过指令“MOVX @Ri” 和 “MOVX @DPTR”间接寻址。KEIL-C51 编译器中，分配变量定位 XRAM，应使用 “pdata” 或 “xdata” 去定义。编译后，这变量被分别编译为通过“MOVX @Ri” 和 “MOVX @DPTR”寻址。从而 MA807 硬体能正确存取它们。

7.4. 关于 C51 编译器的声明识别符

C51编译器的声明识别符与 MA807 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。MA807有 256 字节的片上 xdata 存储空间。

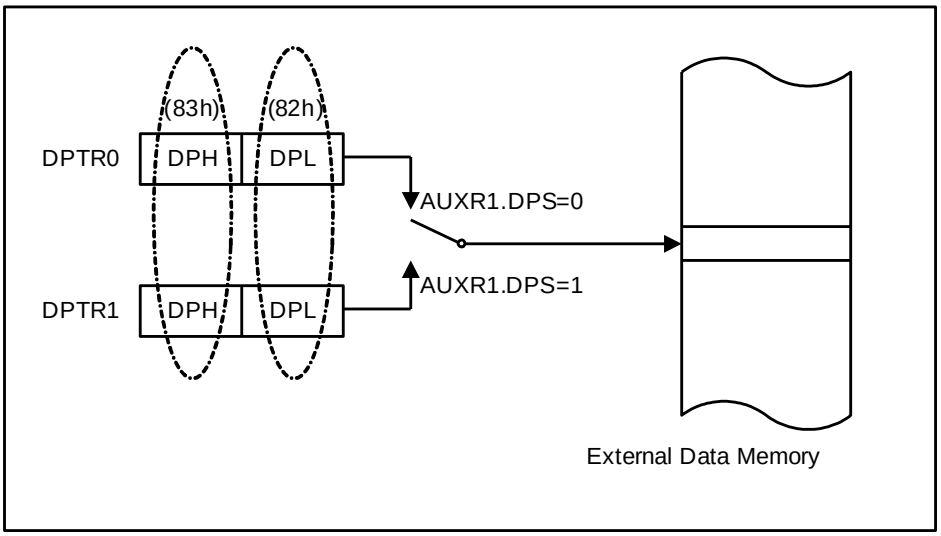
pdata
分页的外部数据(256 字节) 或片上的扩展RAM；通过 “MOVX @Ri” 指令访问标准80C51的256字节存储空间。MA807有片上256 字节 pdata 存储空间，它与xdata共享。

code
8K程序存储空间；通过“MOVC @A+DTPR”访问，作为程序的一部分被读取。MA807有14K字节的片上 code存储器.

8. 双数据指针寄存器 (DPTR)

通过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 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: 辅助控制积存器 1

SFR 地址 = 0xA2

SFR 页 = 全部 复位值 = 0000-0xx0

7	6	5	4	3	2	1	0
GPWKS1	GPWKS0	P5PWM	P1S0	GF2	GF	GF	DPS
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 0: DPTR 选择位，用来在 DPTR0 和 DPTR1 之间切换。

0: 选择 DPTR0

1: 选择 DPTR1

DPS	选择 DPTR
0	DPTR0
1	DPTR1

9. I/O 结构

MA807 有下列 I/O 端口：P0.0~P0.7, P1.0~P1.7, P2.0~P2.7, P3.0~P3.7, P4.0~P4.6, P6.0~P6.1。I/O 口数目，见下表：

表 10-1 可用到的 I/O 口数目

I/O Pins	Number of I/O ports
P0.0~P0.7, P1.0~P1.7, P2.0~P2.7, P3.0~P3.7, P4.0~P4.6, P6.0, P6.1	41

9.1. IO 配置

除 P6.0~P6.1 外，所有 I/O 端口引脚可以配置为四种模式中的其中一种。这四种类型有：准双向口(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)。P6.0 和 P6.1 只能作准双向口模式

下面描述这四种 I/O 口模式。

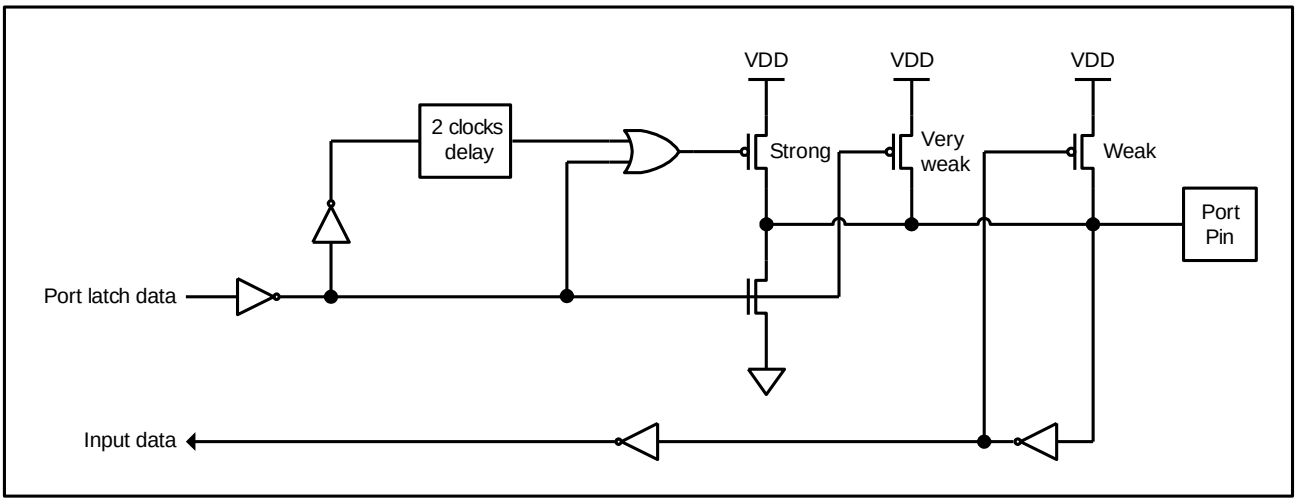
9.1.1. 准双向口

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。

准双向端口配置如下图所示。

图 10-1 准双向 I/O 口

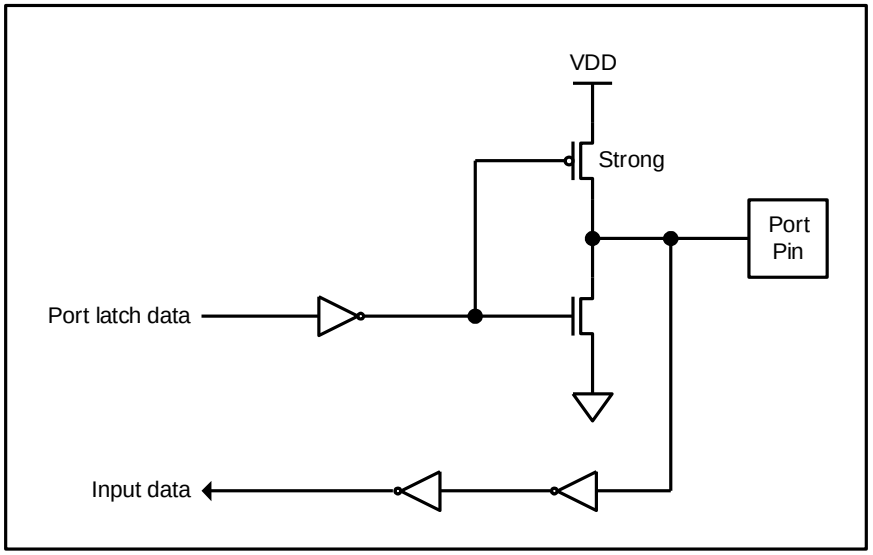


9.1.2. 推挽输出

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。

推挽输出配置如下图所示。

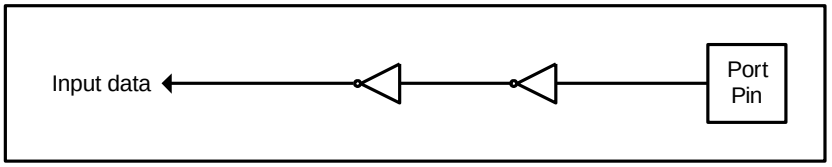
图 10-2 推挽输出



9.1.3. 仅输入(高阻输入)

仅输入模式在引脚上没有任何上拉电阻，如下图所示。

图 10-3 仅输入

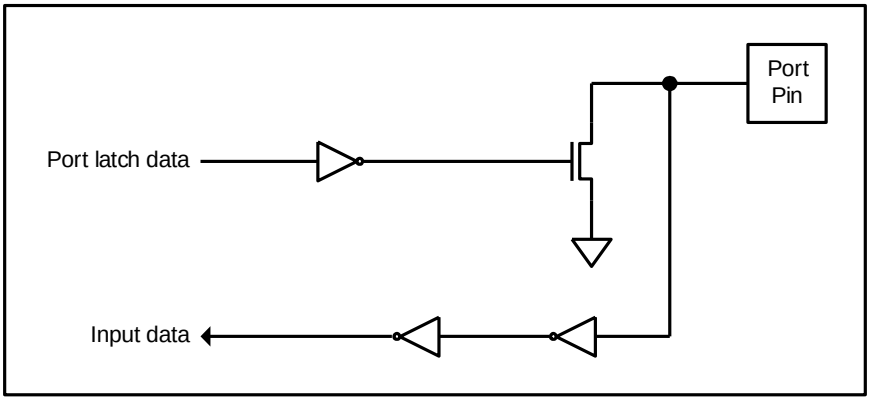


9.1.4. 开漏集输出

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。

图 10-4 开漏集输出



9.2. I/O 端口寄存器

MA807 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，如下表所示。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各个端口引脚的类型。

表 10-2 端口配置设定

PxM0.y	PxM1.y	端口模式
0	0	准双向端口
0	1	推挽式输出
1	0	仅输入(高阻抗输入)
1	1	开漏集输出

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

9.2.1. 端口 0

P0: 端口 0 寄存器

SFR 地址 = 0x80

SFR 页 = 全部 复位值 = 1111-1111

7	6	5	4	3	2	1	0
P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~0: P0.7~P0.0 通过 CPU 置 1/清 0。

P0M0: 端口 0 模式寄存器 0

SFR 地址 = 0x93

SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P0M0.7	P0M0.6	P0M0.5	P0M0.4	P0M0.3	P0M0.2	P0M0.1	P0M0.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

P0M1: 端口 0 模式寄存器 1

SFR 地址 = 0x94

SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P0M1.7	P0M1.6	P0M1.5	P0M1.4	P0M1.3	P0M1.2	P0M1.1	P0M1.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

9.2.2. 端口 1

P1: 端口 1 寄存器

SFR 地址 = 0x90

SFR 页 = 全部 复位值 = 1111-1111

7	6	5	4	3	2	1	0
P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~0: P1.7~P1.0 通过 CPU 置 1/清 0

P1M0: 端口 1 模式寄存器 0

SFR 地址 = 0x91

SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P1M0.7	P1M0.6	P1M0.5	P1M0.4	P1M0.3	P1M0.2	P1M0.1	P1M0.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

P1M1: 端口 1 模式寄存器 1

SFR 地址 = 0x92

SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P1M1.7	P1M1.6	P1M1.5	P1M1.4	P1M1.3	P1M1.2	P1M1.1	P1M1.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

如果 P1.1 ~ P1.5 任何一个口设置为比较器模式输入，则必须配置这个 I/O 口模式为仅输入模式为比较器得到模拟信号。

9.2.3. 端口 2

P2: 端口 2 寄存器

SFR 地址 = 0xA0

SFR 页 = 全部 复位值 = 1111-1111

7	6	5	4	3	2	1	0
P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~0: P2.7~P2.0 通过 CPU 置 1/清 0。 或在 PWM 模式时作 PWM 定时器下溢出事件的通道。

P2M0: 端口 2 模式寄存器 0

SFR 地址 = 0x95

SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P2M0.7	P2M0.6	P2M0.5	P2M0.4	P2M0.3	P2M0.2	P2M0.1	P2M0.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

P2M1: 端口 2 模式寄存器 1

SFR 地址 = 0x96

SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P2M1.7	P2M1.6	P2M1.5	P2M1.4	P2M1.3	P2M1.2	P2M1.1	P2M1.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

9.2.4. 端口 3

P3: 端口 3 寄存器

SFR 地址 = 0xB0

SFR 页 = 全部 复位值 = 1111-1111

7	6	5	4	3	2	1	0
P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~0: P3.7~P3.0 通过 CPU 置 1/清 0。

P3M0: 端口 3 模式寄存器 0

SFR 地址 = 0xB1

SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P3M0.7	P3M0.6	P3M0.5	P3M0.4	P3M0.3	P3M0.2	P3M0.1	P3M0.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

P3M1: 端口 3 模式寄存器 1

SFR 地址 = 0xB2

SFR 页 = 全部 复位值 = 0000-0000

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

9.2.5. 端口 4

P4: 端口 4 寄存器

SFR 地址 = 0xE8

SFR 页 = 全部 复位值 = x111-1111

7	6	5	4	3	2	1	0
--	P4.6	P4.5	P4.4	P4.3	P4.2	P4.1	P4.0
R	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 6~0: P4.6~P4.0 通过 CPU 置 1/清 0。

P4M0: 端口 4 模式寄存器 0

SFR 地址 = 0xB3

SFR 页 = 全部 复位值 = x000-00xx

7	6	5	4	3	2	1	0
--	P4M0.6	P4M0.5	P4M0.4	P4M0.3	P4M0.2	P4M0.1	P4M0.0
R	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

P4M0.1 和 P4M0.0 的复位值由硬件选项 P41IOE 和 P40IOE 决定。如果硬件选项 P41IOE 和 P40IOE 被打开，则 P4M0.1 和 P4M0.0 的复位值被置“1”配置 P4.1/P4.0 作仅输入模式。否则 P4M0.1/P4M0.0 的复位值被清“0”配置 P4.1/P4.0 作准双向口。

P4M1: 端口 4 模式寄存器 1

SFR 地址 = 0xB4

SFR 页 = 全部 复位值 = x000-0000

7	6	5	4	3	2	1	0
--	P4M1.6	P4M1.5	P4M1.4	P4M1.3	P4M1.2	P4M1.1	P4M1.0
R	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

9.2.6. 端口 6

P6: 端口 6 寄存器

SFR 地址 = 0xC8

SFR 页 = F 复位值 = xxxx-xx11

7	6	5	4	3	2	1	0
--	--	--	--	--	--	P6.1	P6.0
R	**R**	**R**	**R**	**R**	**R**	**R/W**	**R/W**

Bit 7~2: 保留.

Bit 1~0: P6.1~P6.0 通过 CPU 置 1/清 0。它们只支持准双向口模式。此寄存器只能通过 SFR 页 F 存取。

9.3. 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
```

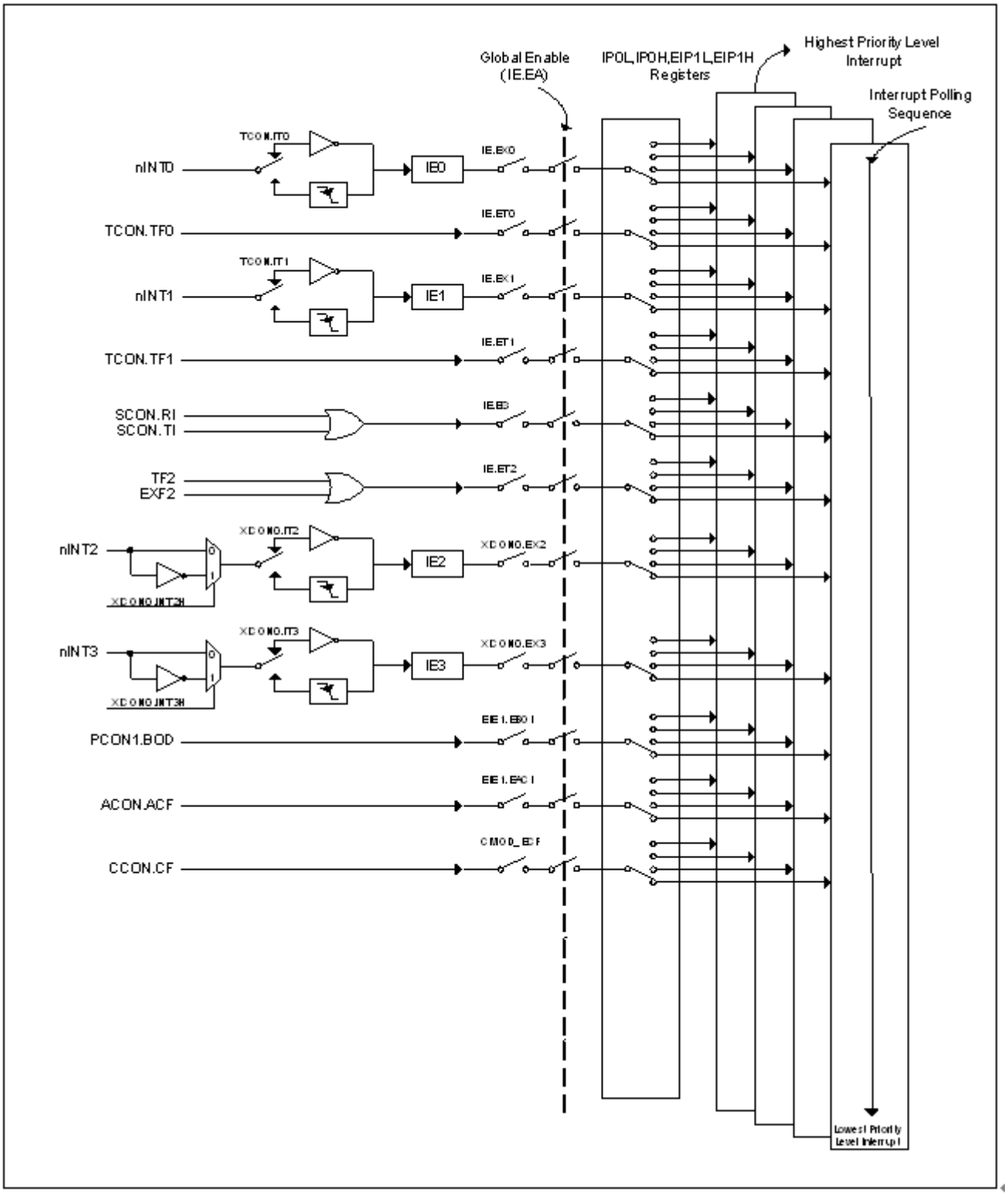
C 语言代码范例:

```
#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
```

10. 中断

10.1. 中断结构



10.2. 中断寄存器

IE: 中断使能寄存器

SFR 地址 = 0xE8

SFR 页 = 全部 复位值 = 0X00-0000

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

Bit 7: EA, 全局中断使能位
0: 禁止所有中断
1: 使能全局中断

Bit 6: 保留.

Bit 5: ET2, 定时器 2 中断使能位
0: 禁止定时器 2 中断
1: 使能定时器 2 中断

Bit 4: ES, 串口中断使能位
0: 禁止串口中断
1: 使能串口中断

Bit 3: ET1, 定时器 1 中断使能位
0: 禁止定时器 1 中断
1: 使能定时器 1 中断

Bit 2: EX1, 外部中断 1 使能位
0: 禁止外部中断 1
1: 使能外部中断 1

Bit 1: ET0, 定时器 0 中断使能位
0: 禁止定时器 0 中断
1: 使能定时器 0 中断

Bit 0: EX0, 外部中断 0 使能位
0: 禁止外部中断 0
1: 使能外部中断 0

XIFLG: 外部中断标志寄存器

SFR 地址 = 0xC0

SFR 页 = 全部 复位值 = XXXX-0000

7	6	5	4	3	2	1	0
--	--	--	--	0	0	IE3	IE2
R	**R**	**R**	**R**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~4: 保留.
Bit 3~2: 保留 0.

Bit 1: IE3, 外部中断 3 请求标志
0: 当进入中断服务程序时硬件清零。也可以通过指令清零。
1: 当侦察到外部中断时硬件置 1。也可以通过软件置 1。

Bit 0: IE2, 外部中断 2 请求标志

0: 当进入中断服务程序时硬件清零。也可以通过指令清零。
1: 当侦察到外部中断时硬件置 1。也可以通过软件置 1。

XICON0: 外部中断控制寄存器 0

SFR 地址 = 0xC1
SFR 页 = 全部 复位值 = X000-X000

7	6	5	4	3	2	1	0
--	INT3H	IT3	EX3	--	INT2H	IT2	EX2
R	**R/W**	**R/W**	**R/W**	**R**	**R/W**	**R/W**	**R/W**

Bit 7: 保留.

Bit 6: INT3H, nINT3 高/低触发使能
0: nINT3 低电平或下降沿触发（P4.2）。
1: nINT3 高电平或上升沿触发（P4.2）。

Bit 5: IT3, 外部中断 3 类型控制位
0: 电平触发
1: 边沿触发

Bit 4: EX3, 外部中断 3 使能位
0: 禁止外部中断 3
1: 使能外部中断 3

Bit 3: 保留.

Bit 2: INT2H, nINT2 高/低触发使能
0: nINT2 低电平或下降沿触发（P4.3）。
1: nINT2 高电平或上升沿触发（P4.3）。

Bit 1: IT2, 外部中断 2 类型控制位
0: 电平触发
1: 边沿触发

Bit 0: EX2, 外部中断 2 使能位
0: 禁止外部中断 2
1: 使能外部中断 2

EIE1: 扩展中断使能寄存器

SFR 地址 = 0xAD
SFR 页 = 全部 复位值 = XXXX-XX00

7	6	5	4	3	2	1	0
--	--	--	--	--	--	EACI	EBOI
R	**R**	**R**	**R**	**R**	**R**	**R/W**	**R/W**

Bit 7~2: 保留.

Bit 1: EACI, 模拟比较器中断使能位
0: 模拟比较器中当 ACCON.ACF =1 时使能比较器中断
1: 模拟比较器中当 ACCON.ACF =1 时禁止比较器中断

Bit 0: EBOI, BOD 中断使能
0: 电源控制模块中当 PCON1.BOD=1 时使能 BOD 中断
1: 电源控制模块中当 PCON1.BOD=1 时禁止 BOD 中断

IP0L: 中断优先级寄存器 0 低

SFR 地址 = 0xB8

SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
PX3L	PX2L	PT2L	PSL	PT1L	PX1L	PT0L	PX0L
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: PX3L, 外部中断 3 优先级 低位
Bit 6: PX2L, 外部中断 2 优先级 低位
Bit 5: PT2L, 定时器 2 中断优先级 低位
Bit 4: PSL, 串行口中断优先级 低位
Bit 3: PT1L, 定时器 1 中断优先级 低位
Bit 2: PX1L, 外部中断 1 优先级 低位
Bit 1: PT0L, 定时器 0 中断优先级 低位
Bit 0: PX0L, 定时器 0 中断优先级 低位

IP0H: 中断优先级寄存器 0 高

SFR 地址 = 0xB7

SFR 页 = 全部 复位值 = 0000-0000

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

Bit 7: PX3H, 外部中断 3 优先级 高位
Bit 6: PX2H, 外部中断 2 优先级 高位
Bit 5: PT2H, 定时器 2 中断优先级 高位
Bit 4: PSH, 串行口中断优先级 高位
Bit 3: PT1H, 定时器 1 中断优先级 高位
Bit 2: PX1H, 外部中断 1 优先级 高位
Bit 1: PT0H, 定时器 0 中断优先级 高位
Bit 0: PX0H, 定时器 0 中断优先级 高位

EIP1L: 扩展中断优先级寄存器 1 低

SFR 地址 = 0xAE

SFR 页 = 全部 复位值 = XXX0-0000

7	6	5	4	3	2	1	0
--	--	--	0	0	PPTL	PACL	PBOL
R	**R**	**R**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~5: 保留.
Bit 4~3: 保留 0.
Bit 2: PPTL, PWM 定时器 中断优先级 低位
Bit 1: PACL, 模拟比较器中断优先级 低位
Bit 0: PBOL, BOD 中断优先级 低位

EIP1H: 扩展中断优先级寄存器 1 高

SFR 地址 = 0xAF

SFR 页 = 全部 复位值 = XXX0-0000

7	6	5	4	3	2	1	0
--	--	--	0	0	PPTH	PACH	PBOH
R	**R**	**R**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~5: 保留.
Bit 4~3: 保留 0.

Bit 2: PPTH, PWM 定时器 中断优先级 高位
Bit 1: PACH, 模拟比较器中断优先级 高位
Bit 0: PBOH, BOD 中断优先级 高位

IP0L, IP0H, EIP1L 和 EIP1H 组合成 4 级优先级中断，见下表：

{IPH.x , IPL.x}	优先级
11	1 (最高)
10	2
01	3
00	4

MA807 有 11 个中断源。通过置位或清零寄存器 IE、EIE1、 XICON0、 XICON1 中对应使能位可以单独的使能或禁止每一个中断源。也包含一个全局禁止位 EA，清零 EA 则禁止所有中断。

每个中断源有两个对应的中断优先级寄存器位去设置它的优先级。一个在 IpxH 寄存器中，另一个在案 IpxL 寄存器中。高优先级的中断不会被低优先级的中断请求打断。两个优先不同的中断请求同时发生则高优先级的中断优先被处理，若优先级相同的中断请求同时发生则由内部查询顺序决定处理哪个中断。两级优先级中断结构在控制这些中断事物中是非常灵活的。下表展示了相同级别中断的内部查询顺序和中断向量。

中断源	向量地址	优先级别
外部中断 0	0003H	1 (高)
定时器 0	000BH	2
外部中断 1	0013H	3
定时器 1	001BH	4
串行口中断	0023H	5
定时器 2	002BH	6
外部中断 2	0033H	7
外部中断 3	003BH	8
BOD	0043H	9
模拟比较器	004BH	10
PWM 定时器	0053H	11

外部中断/INT0、/INT1、/INT2、/INT3 分别通过 TCON 的 IT0、IT1，XICON0 的 IT2、IT3，可以设置成电平触发或边沿触发。实际产生的中断标志位是 TCON 的 IE0、IE1，XIFLG 的 IE2、IE3。产生外部中断时，如果是边沿触发，进入中断服务程序后由硬件清除中断标志位，如果中断是电平触发，由外部请求源而不是由片内硬件控制请求标志。

定时器 0 和定时器 1 中断由 TF0 和 TF1(分别由各自的定时/计数寄存器控制)产生。当产生定时器中断时，进入中断服务程序后由片内硬件清除标志位。

串口中断由 RI 和 TI 的逻辑或产生。进入中断服务程序后，这些标志均不能被硬件清除。实际上，中断服务程序通常需要确定是由 R I 还是 TI 产生的中断，然后由软件清除中断标志。

定时器 2 中断由 TF2 和 EXF2 的逻辑或产生。进入中断服务程序后，这些标志均不能被硬件清除。实际上，中断服务程序通常需要确定是由 TF2 还是 EXF2 产生的中断，然后由软件清除中断标志。

BOD 中断由 PCON1.BOD 产生，当片上的 Brownout-Detector 检测到对应的低电平事件时置此位中断请求标志。进入中断服务程序后由片内硬件清除标志位。

模拟比较器中断由 ACCON .ACF 产生。进入中断服务程序后由片内硬件清除标志位。

PWM 定时器中断由 CCON.CF 产生。进入中断服务程序后由片内硬件清除标志位。i

所有这些产生中断的位都可通过软件置位或清零,与通过硬件置位或清零的效果相同。简而言之，中断可由软件产生，推迟或取消。

硬件如何响应中断

每一个机器周期的 S5P2 时间采样中断请求标志，采样被保持到下一个 S5P2 时期。如果一个中断请求标志在第一个 S5P2 时间被置位，在第二个 S5P2 时间（采样周期）将被硬件发现，这时若没有下面任何一个阻止条件则中断系统将产生一个硬件调用 LCALL 转去执行相应的中断服务程序。
阻止条件如下：
1. CPU 正在处理同级或更高级的中断；
2. 现行机器周期不是所执行的最后一个机器周期；
3. 正在执行的是 RETI 或是访问 IE 或 IP 的指令。

上叙任何一个条件都会阻止硬件去响应发生的中断请求。条件 2 保证了在进入其它中断服务程序前执行完当前指令。条件 3 保证了执行完 RETI 指令或是访问 IE 或 IP 的指令后在进入其它中断向量前能执行至少一条或更多条的指令。
每个机器周期硬件都会重复查询中断请求标志，查询到的值都是前一个 S5P2 时间的状态。值得注意的是如果一个被触发的中断标志由于上面的阻止条件发生了而没得到响应，并且此标志没有持续到阻止条件消失，这样的中断请求将得不到响应。也就是说，中断处理本身不能锁存中断，譬如外部电平中断若在电平出现时被屏蔽，而在中断识别之前电平消失，它被完全忽略。也就是每个查询都是新状态。

10.3. 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);
```

11. 定时器/计数器

MA807 有四个定时/计数器：定时器 0、定时器 1、定时器 2 和 PWM 定时器 。定时器 0/1/2 能被配置作为定时器或事件计数器。PWM 定时器能配置为定时器 或 PWM 发生器。

定时器功能，TLx 寄存器每 12 个时钟周期或 1 个时钟周期加 1，通过软件设置 AUXR2.T0X12、 AUXR2.T1X12 和 T2MOD.T2X12 位来选择 。每 12 个时钟周期加一，计数速率达 1/12 的晶振频率。

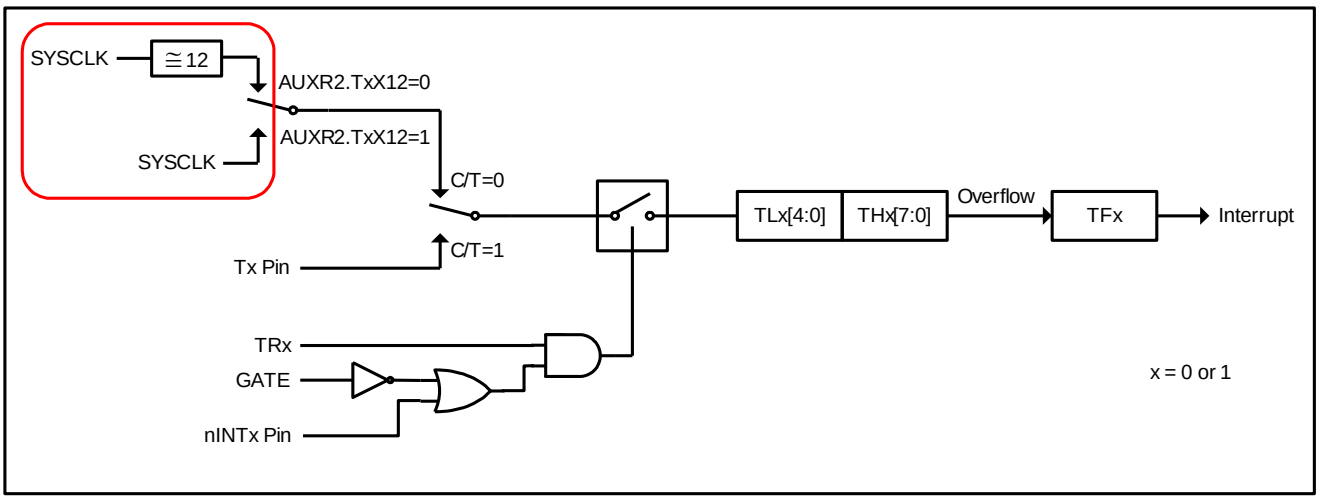
计数器功能，根据对应的外部输入引脚的下降沿 T0、T1 或 T2 寄存器加 1。在这些功能中，每个时钟周期对外部输入信号(T0、T1 和 T2 引脚)进行采样，每 12 个时钟周期对 T2 引脚采样。当采样信号出现一个高电平接着一个低电平，计数加 1。当检测到跳变时新计数值出现在寄存器中。对定时器 0 和定时器 1 来说，需要用两个时钟周期来识别下降沿跳变，最大的计数速率为 1/2 的晶振频率；对于定时器 2，需要用 24 个时钟周期来识别下降沿跳变，最大计数速率为 1/24 的晶振频率。外部输入信号没有严格的周期限制，但是要确保在电平改变前至少有一次采样，对定时器 0 和定时器 1 来说信号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

11.1. 定时器 0 和定时器 1

11.1.1. 模式 0

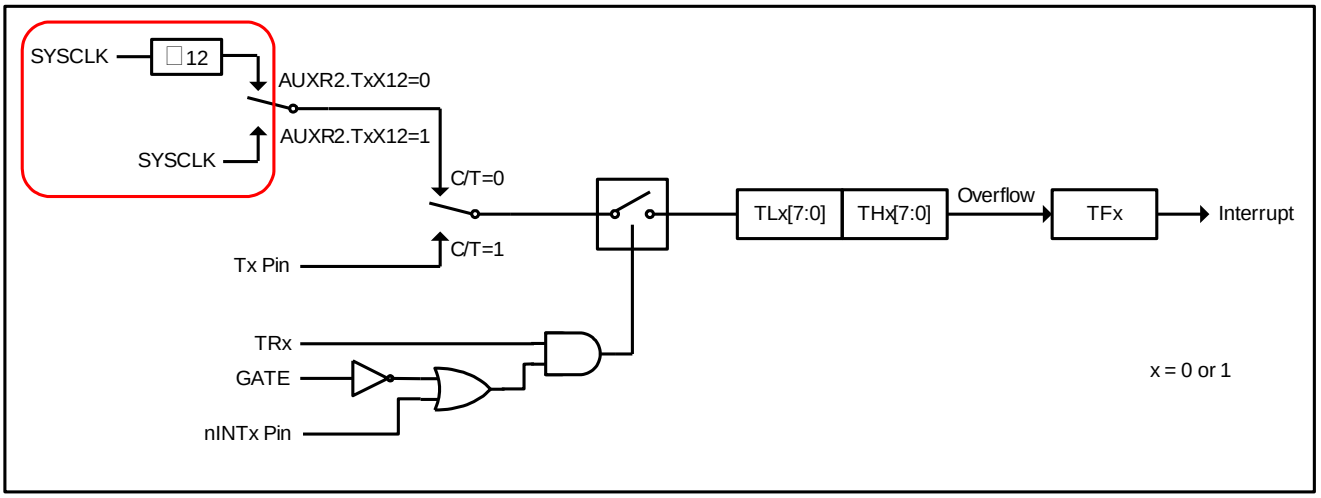
在这个模式，定时器寄存器配置为一个 13 位寄存器。计数器所有位从全 1 翻转到全 0，置位定时器中断标志位 TFx。当 TRx=1 且 GATE=0 或/INTx=1，定时器使能输入计数。(置 GATE=1 时通过外部输入/INTx 控制定时器，以便脉冲宽度测量)。TRx 和 TFx 控制位在专用寄存器 TCON。GATE 位在 TMOD。有两个不同的 GATE 位，一个是定时器 0(TMOD.7)另一个是定时器 0(TMOD.3)。

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。



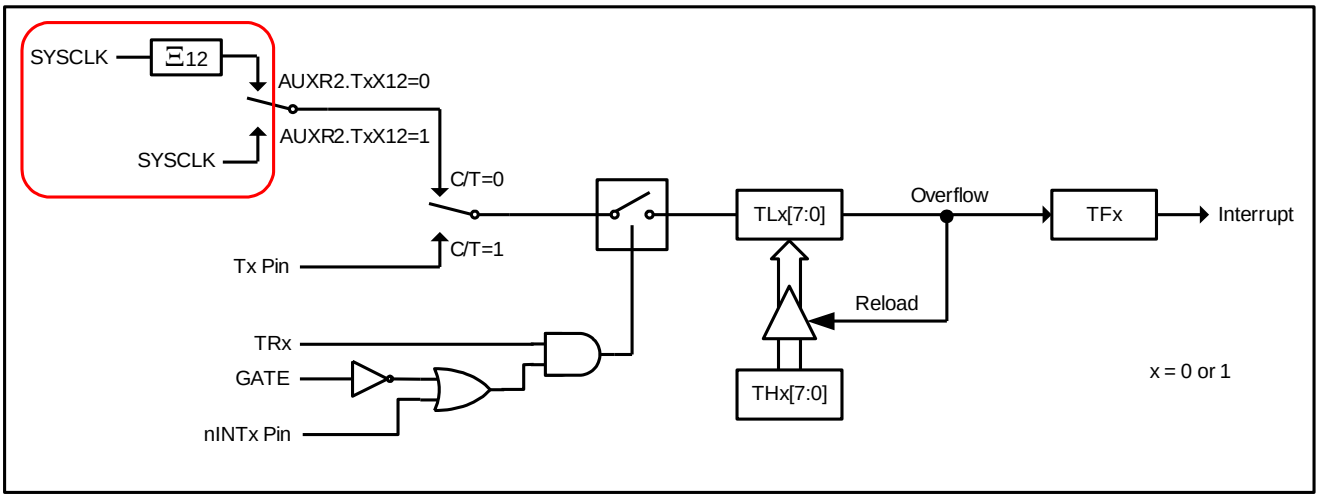
11.1.2. 模式 1

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。在这个模式，THx 和 TLx 串联，没有预分频。



11.1.3. 模式 2

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)。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。

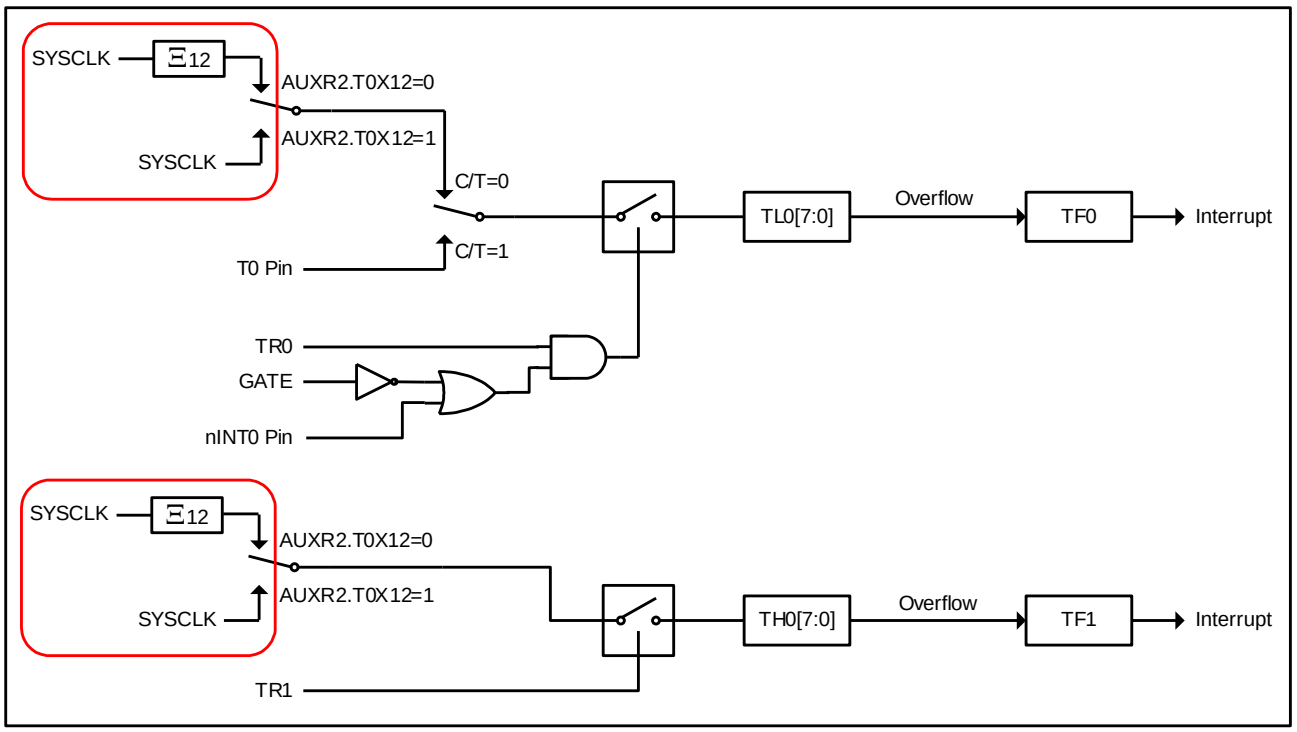


11.1.4. 模式 3

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。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。

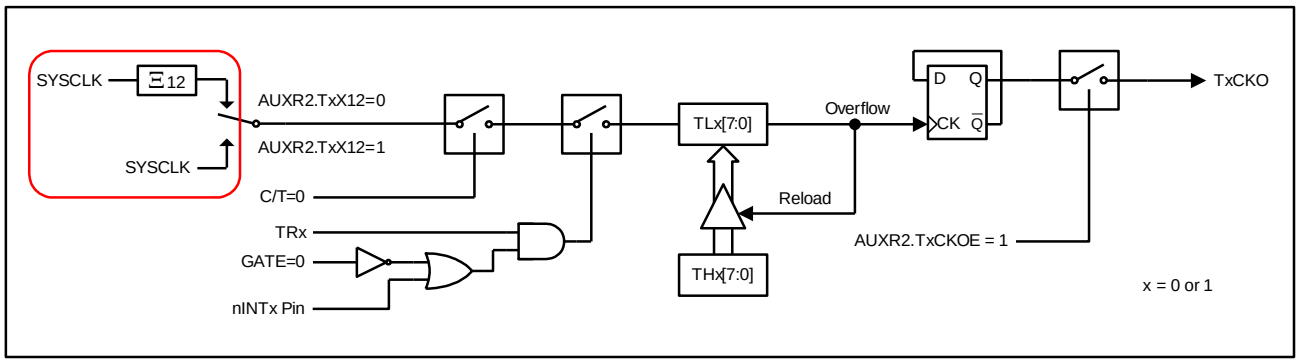
模式 3 提供当有额外的需求应用时的一个 8 位时器或计数器时。当定时器 0 在模式 3 时，定时器 1 可打开或关闭并切换到脱离，进入到自己的模式 3，或仍然可用作为串行口的波特率发生器，或者不需要中断的其它应用。



11.1.5. 定时器时钟输出

定时器的可编程时钟输出模式，则从引脚 TxCK0 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 THx 寄存器的值决定，公式如下所示。时钟输出模式编程步骤如下：

1. 在 AUXR2 寄存器置位 TxCKOE。
2. 在 TMOD 寄存器清除定时器 0/1 的 C/T 位。



$$\text{T0/T1 Clock-out Frequency} = \frac{\text{SYSCLK Frequency}}{n \times (256 - \text{THx})}$$

; n=24, if TxX12=0
; n=2, if TxX12=1
; x = 0 or 1 & C/T = 0

11.1.6. 定时器 0/1 寄存器

TMOD: 定时器/计数器模式寄存器

SFR 地址 = 0x89

SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
GATE	C/T	M1	M0	GATE	C/T	M1	M0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**
|←---------- Timer1 ----------→|				|←----------Timer0 ----------→|		

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)。

M1	M0	工作模式
0	0	13 位定时器/计数器。
0	1	16 位定时器/计数器。THx 与 TLx 串联，没有分频器
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: 定时器/计数器控制寄存器

SFR 地址 = 0x88

SFR 页 = 全部 复位值 = 0000-0000

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

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。
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.

TL0: 定时器 0 低

SFR 地址 = 0x8A

SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
TL0[7]	TL0[6]	TL0[5]	TL0[4]	TL0[3]	TL0[2]	TL0[1]	TL0[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

TH0: 定时器 0 高

SFR 地址 = 0x8C

SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
TH0[7]	TH0[6]	TH0[5]	TH0[4]	TH0[3]	TH0[2]	TH0[1]	TH0[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

TL1: 定时器 1 低

SFR 地址 = 0x8B
SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
TL1[7]	TL1[6]	TL1[5]	TL1[4]	TL1[3]	TL1[2]	TL1[1]	TL1[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

TH1: 定时器 1 高
SFR 地址 = 0x8D
SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
TH1[7]	TH1[6]	TH1[5]	TH1[4]	TH1[3]	TH1[2]	TH1[1]	TH1[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

AUXR2: 辅助寄存器 2
SFR 地址 = 0xA3
SFR 页 = 全部 复位值 = 000X-XX00

7	6	5	4	3	2	1	0
T0X12	T1X12	URM0X6	--	--	--	T1CKOE	T0CKOE
R/W	**R/W**	**R/W**	**R**	**R**	**R**	**R/W**	**R/W**

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。
T1CKOE: 置位/清零来使能/禁止从 P3.5 输出定时器 1 时钟。
T0CKOE: 置位/清零来使能/禁止从 P3.4 输出定时器 0 时钟。

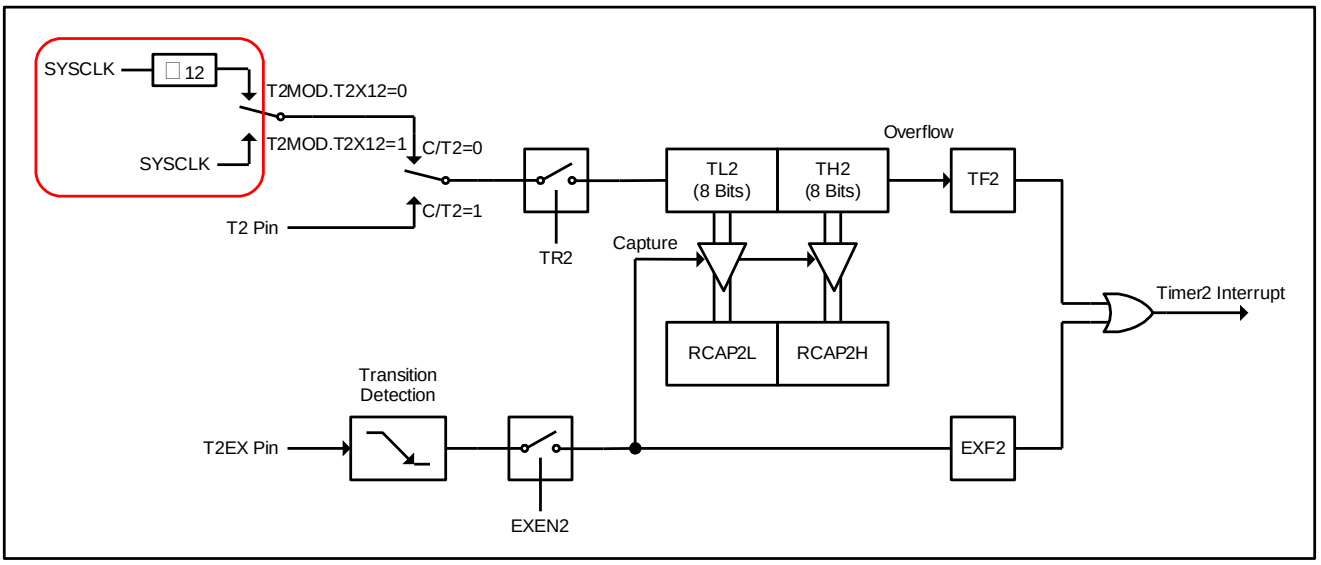
11.2. 定时器 2

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。

11.2.1. 捕获模式 (CP)

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)。捕获模式图解如下图。(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 12-5 定时器 2 捕获模式



11.2.2. 自动加载模式(AR)

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 12-6 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 12-6 定时器 2 自动加载模式（CEN=0）

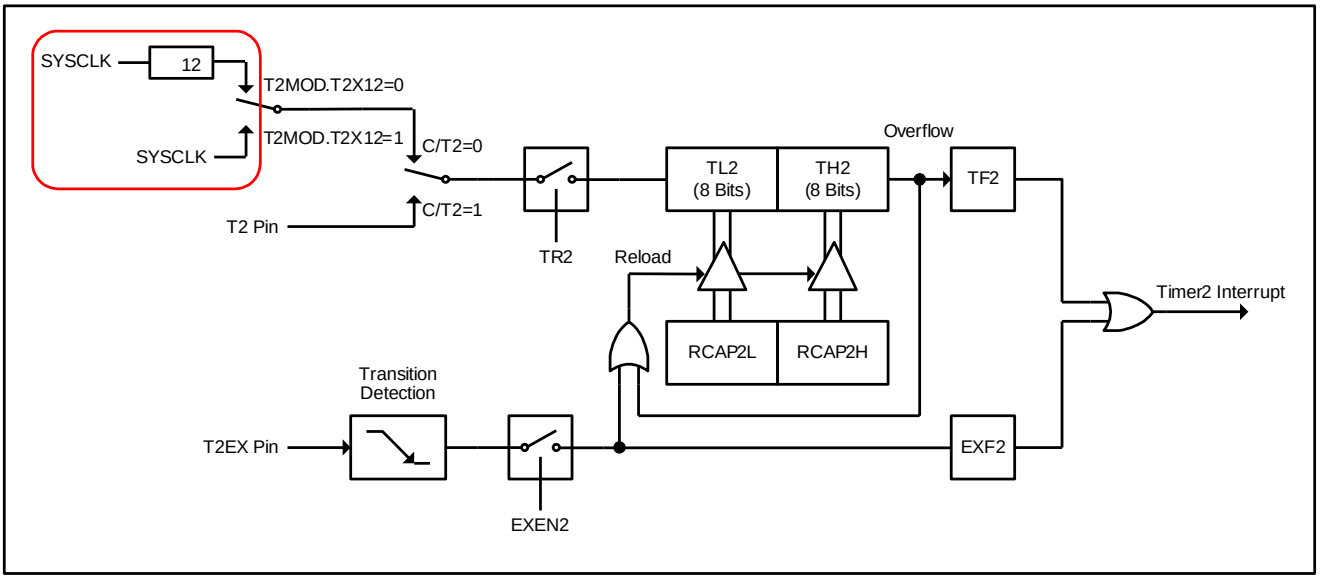
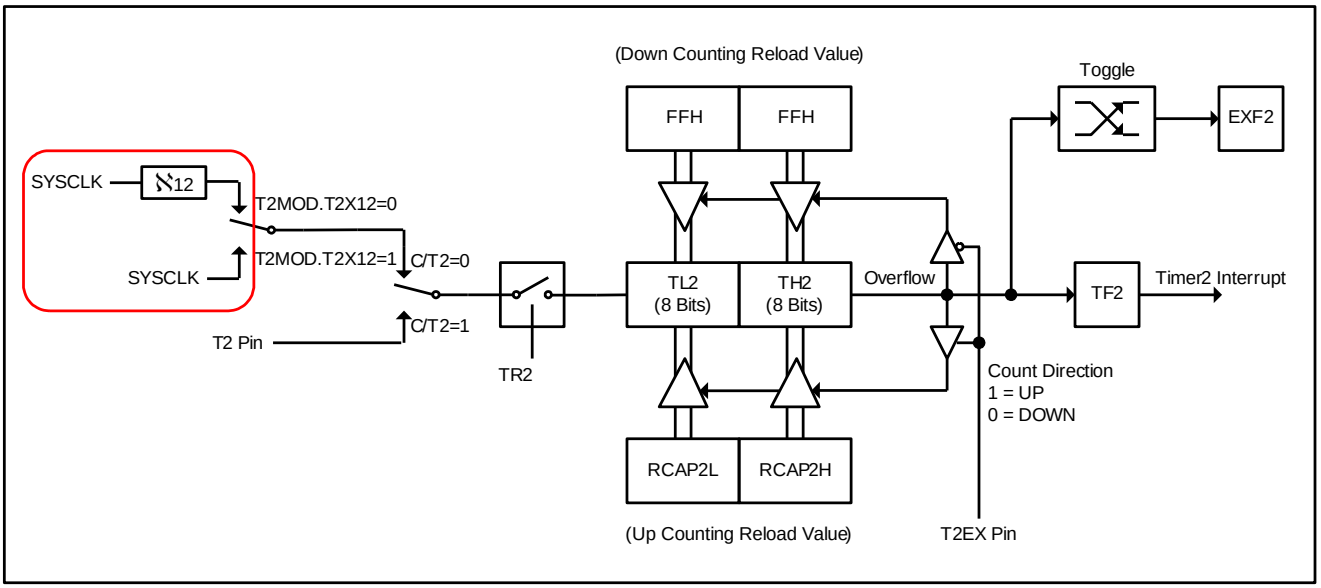


图 12-7 示 DCEN=1，使能定时器 2 向上或向下计数。这种模式下允许 T2EX 引脚控制计数方向。当 T2EX 的引脚为逻辑 1 时定时器 2 向上计数。定时器 2 在 0FFFFH 时溢出并置位 TF2 标志位，如果中断使能将产生中断。溢出也将引起 RCAP2L 和 RCAP2H 的 16 位值加载到定时器的寄存器 TL2 和 TH2。当 T2EX 的引脚为逻辑 0 时定时器 2 向下计数。当 TL2 和 TH2 和存储在 RCAP2L 和 RCAP2H 的值相等时将产生下溢。下溢将置位 TF2 标志位并将 0FFFFH 加载到定时器的寄存器 TL2 和 TH2。

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 12-7 定时器 2 自动加载模式（CEN=1）



11.2.3. 波特率发生器模式 (BRG)

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 12-8 所示定时器 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：

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

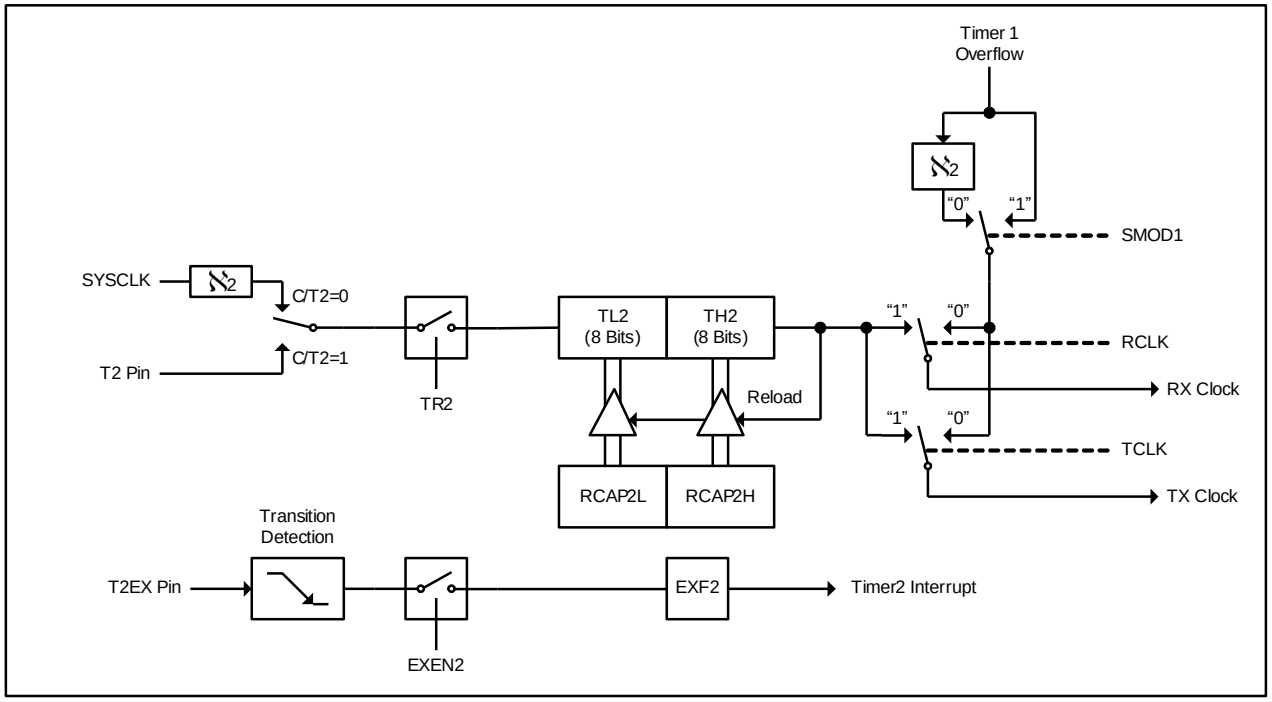
这里：Fosc 是系统时钟。RCAP2H，RCAP2L 的内容为一个 16 位的无符号数，可由如下计算出：

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

定时器 2 作为一个波特率发生器模式如图 12-8 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 12-8 定时器 2 波特率发生器模式



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(SYSCLK)和在 RCAP2H 和 RCAP2L 寄存器的加载值来决定，如下公式：

$$\text{T2 Clock-out Frequency} = \frac{\text{SYSCLK Frequency}}{4 \times (65536 - (\text{RCAP2H, RCAP2L}))}$$

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：

1. 置位 T2MOD 寄存器的 T2OE 位。
2. 清除 T2CON 寄存器的 C/T2 位。
3. 从公式计算出 16 位加载值并输入到 RCAP2H 和 RCAP2L 寄存器。
4. 在 TH2 和 TL2 输入一个 16 位初始值。可以和重载值相等。
5. 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。

注：

(1) 此模式定时器 2 溢出标志 TF2 从来不会被置 1。

(2) SYSCLK=12MHz 时，定时器 2 可编程输出频率范围为 45.7Hz 到 3MHz。

11.2.5. 定时器 2 寄存器

T2MOD: 定时器/计数器 2 模式寄存器

SFR 地址 = 0xC9

SFR 页 = 全部 复位值= XXX0-XX00

7	6	5	4	3	2	1	0
--	--	--	T2X12	--	--	T2OE	DCEN
R	**R**	**R**	**R/W**	**R**	**R**	**R/W**	**R/W**

T2X12：定时器 2 时钟源选择

0：选择 SYSCLK/12 作为时钟源

1：选择 SYSCLK 作为时钟源

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。

T2CON: 定时器/计数器 2 控制寄存器

SFR 地址 = 0xC8

SFR 页 = **0** 复位值 = 0000-0000

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

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。

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

TL2：定时器 2 寄存器 低

SFR 地址 = 0xCC
SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
TL2[7]	TL2[6]	TL2[5]	TL2[4]	TL2[3]	TL2[2]	TL2[1]	TL2[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

TH2：定时器 2 寄存器 高

SFR 地址 = 0xCD
SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
TH2[7]	TH2[6]	TH2[5]	TH2[4]	TH2[3]	TH2[2]	TH2[1]	TH2[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

RCAP2L：定时器 2 捕获寄存器 低

SFR 地址 = 0xCA
SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
RCAP2L[7]	RCAP2L[6]	RCAP2L[5]	RCAP2L[4]	RCAP2L[3]	RCAP2L[2]	RCAP2L[1]	RCAP2L[1]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

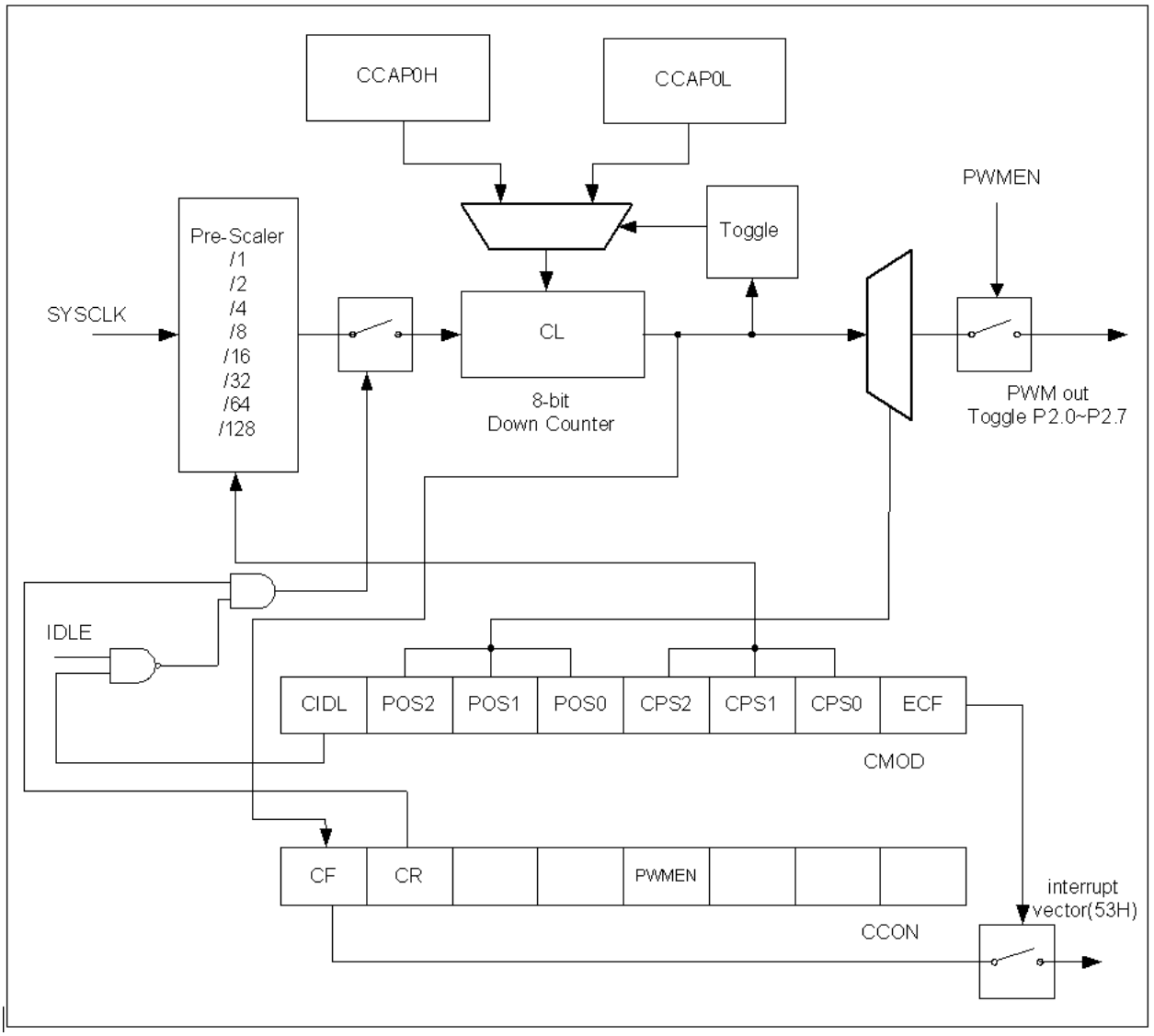
RCAP2H：定时器 2 捕获寄存器 高

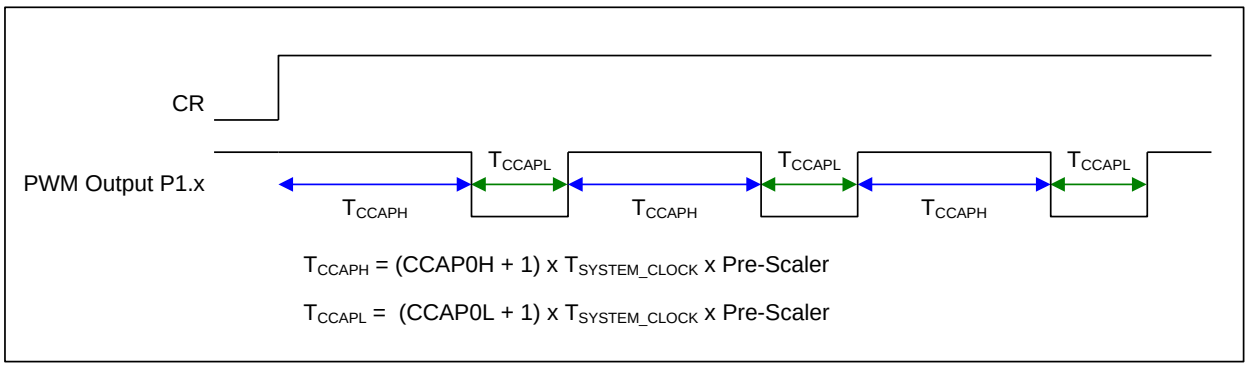
SFR 地址 = 0xCB
SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
RCAP2H[7]	RCAP2H[6]	RCAP2H[5]	RCAP2H[4]	RCAP2H[3]	RCAP2H[2]	RCAP2H[1]	RCAP2H[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

11.3. PWM 定时器

11.3.1. PWM 定时器结构





11.3.2. PWM 定时器寄存器

CMOD: PWM 定时器模式寄存器

SFR 地址 = 0xD9

SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
CIDL	POS2	POS1	POS0	CPS2	CPS1	CPS0	ECF
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: CIDL, CPU 空闲模式（IDLE）计数控制
0: IDLE 时 PWM 定时器继续计数
1: IDLE 时 PWM 定时器停止计数

Bit 6~4: POS[2:0], PWM 输出口选择

POS[2:0]	PWMEN	PWM 输出口 (AUXR1.P5PWM=0)
0 0 0	1	P2.0
0 0 1	1	P2.1
0 1 0	1	P2.2
0 1 1	1	P2.3
1 0 0	1	P2.4
1 0 1	1	P2.5
1 1 0	1	P2.6
1 1 1	1	P2.7
X X X	0	Disabled

Bit 3~1: CPS[2:0], PWM 计数器分频选择

CPS[2:0]	分频
0 0 0	1
0 0 1	2
0 1 0	4
0 1 1	8
1 0 0	16
1 0 1	32
1 1 0	64
1 1 1	128

Bit 0: ECF, 使能 PWM 定时器向下溢出时中断
0: 禁止 CCON.CF 位产生中断
1: 使能 CCON.CF 位产生中断

CCON: PWM 定时器控制寄存器

SFR 地址 = 0xD8

SFR 页 = **0** 复位值 = 00XX-0XXX

7	6	5	4	3	2	1	0
CF	CR	-	-	PWMEN	-	-	-
R/W	**R/W**	**R**	**R**	**R/W**	**R**	**R**	**R**

Bit 7: CF, PWM 定时器下溢出标志
0: 只能通过软件清 0
1: 当 PWM 计数器向下溢出时硬件置 1。若 EMOD.ECF=1 时，将产生一个中断请求。可以由硬件或软件置 1。

Bit 6: CR, PWM 定时器启动控制位
0: 停止
1: 启动

Bit 5~4: 保留.

Bit 3: PWMEN, PWM 输出使能
0: 禁止在选定的 I/O 口上输出 PWM
1: 使能在选定的 I/O 口上输出 PWM

Bit 2~0: 保留.

CACP0L: PWM 占空比寄存器 低

SFR 地址 = 0xEA
SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
CACP0L[7]	CACP0L[6]	CACP0L[5]	CACP0L[4]	CACP0L[4]	CACP0L[2]	CACP0L[1]	CACP0L[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

CACP0H: PWM 占空比寄存器 高

SFR 地址 = 0xFA
SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
CACP0H[7]	CACP0H[6]	CACP0H[5]	CACP0H[4]	CACP0H[4]	CACP0H[2]	CACP0H[1]	CACP0H[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

11.3.3. 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 (UART)

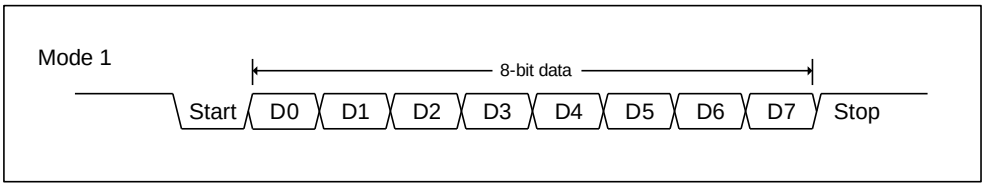
MA807 支持一个全双工的串行口，意思是可以同时发送和接收数据。它有一个接收缓冲，意味着在前一个接收到的字节没有从寄存器读出前，就可以开始接收第二个字节。但是，如果第一个字节在第二个字节接收完成前仍然没有被读出，则其中的一个字节将会丢失。串行口的接收和发送寄存器都通过特殊寄存器 SBUF 来访问。写到 SBUF 加载到传送寄存器，当从 SBUF 读时是一个物理上独立分离的接收寄存器。

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(P3.0)传送和接收。TXD(P3.1)总是作为输出移位时钟。波特率可通过 AUXR2.URM0X6 选择为系统时钟频率的 1/12 或 1/2。

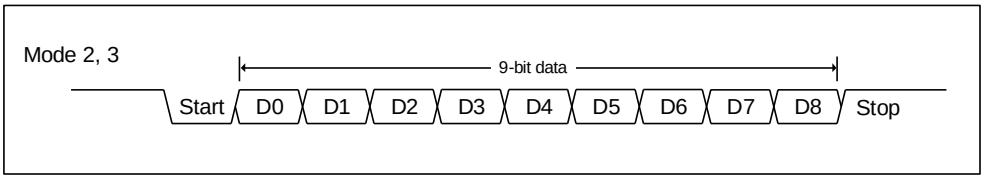
模式 1：10 位通过 TXD 传送或通过 RXD 接收，数据帧包括一个起始位(0)，8 个数据位(低位优先)，和一个停止位(1)。在接收时，停止位进入到专用寄存器(SCON)的 RB8。波特率是可变的。

图 123-1 模式 1 数据帧



模式 2：11 位通过 TXD 传送或通过 RXD 接收，数据帧包括一个起始位(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。

图 123-2 模式 2 数据帧



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，收到起始位时启动接收。

除了标准操作外，UART 还能具有侦察丢失停止位的帧错误和自动地址识别的功能。

12.1. 模式 0 详述

串行数据通过 RXD 读入和输出，TXD 输出移位时钟。接收和发送 8 位数据：8 个数据位(低位优先)。波特率可选择为系统时钟的 1/12 或 1/2。图 133-3 显示了串口模式 0 的简化功能框图。

使用 SBUF 作为一个目的寄存器可通过任何指令来启动传输。“写到 SBUF”信号触发 UART 引擎开始发送。SBUF 里面的数据在 TXD（P3.1）脚的每一个上升沿移出到 RXD（P3.0）脚。八个上升沿移位时钟过后，硬件置 TI 为 1 标志发送完成。见图 133-4。

当 REN=1 和 RI=0 时接收启动。在下一个指令周期，RX 控制单元写 11111110 到接收移位寄存器，且在下一个时钟相位激活接收。

接收使能移位时钟选择输出功能 P3.1 引脚。当接收激活时，在移位时钟的下降沿采样 RXD（P3.0）脚并移到寄存中。八个下降沿移位时钟过后，硬件置 RI 为 1 标志接收完成。见图 133-5。

图 123-3 串口模式 0

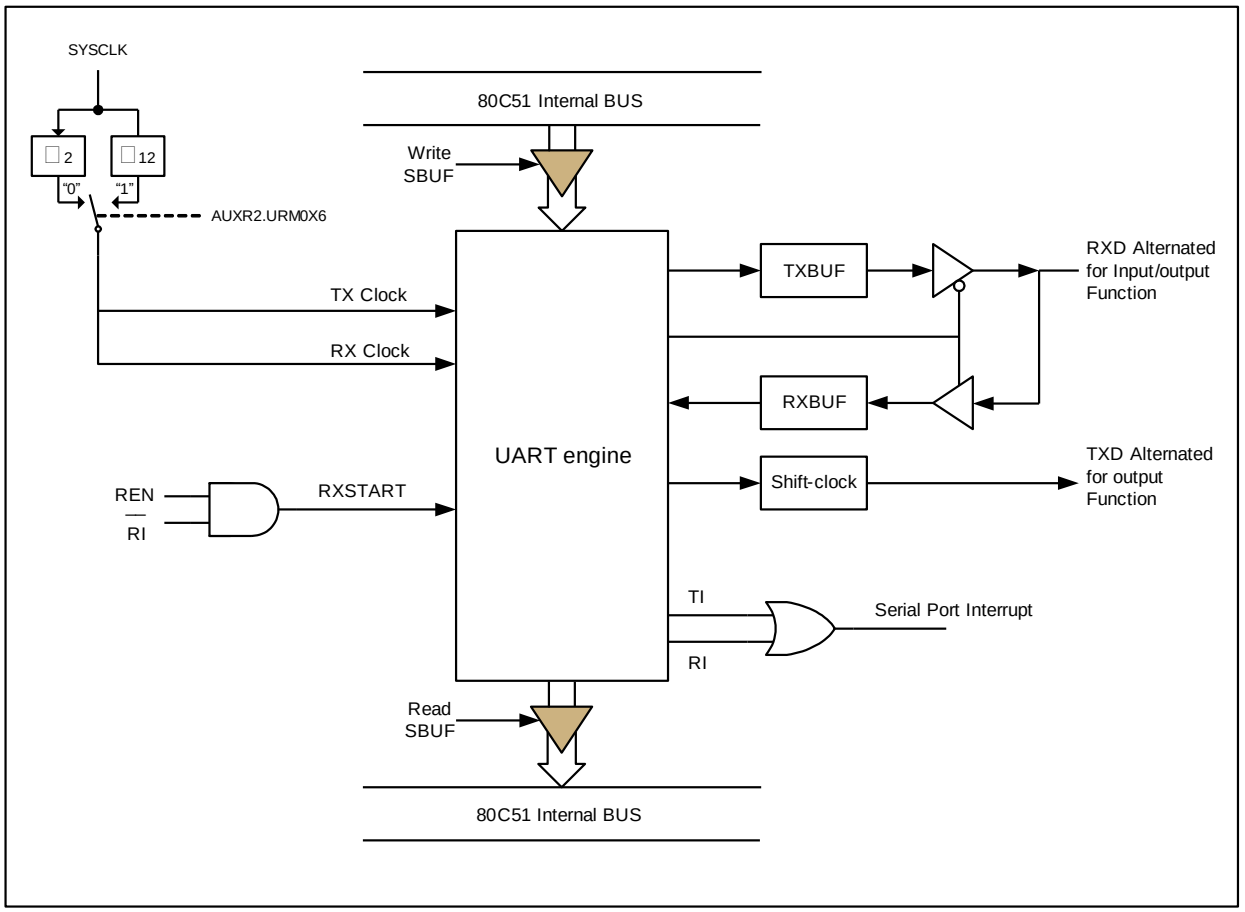


图 123-4 模式 0 发送波型

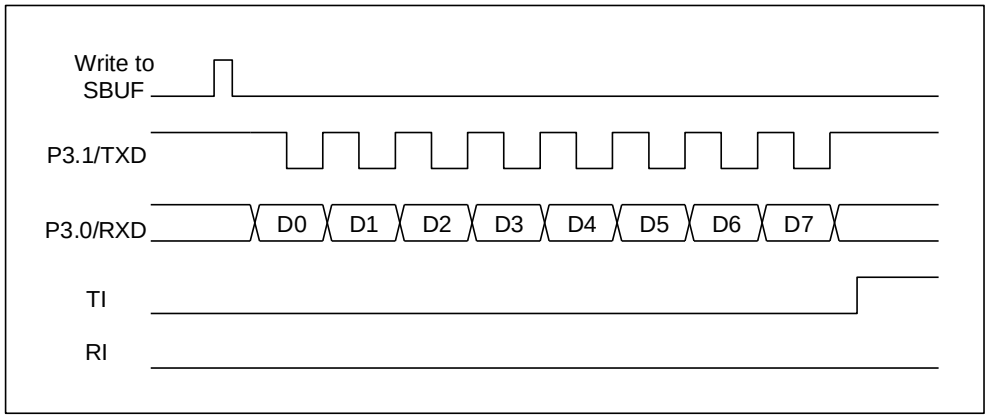
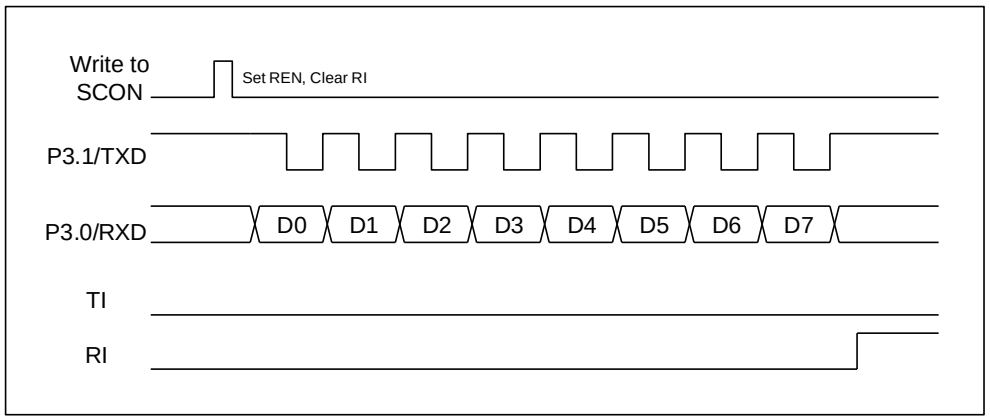


图 123-5 模式 0 接收波型


Write to
SCON
Set REN, Clear RI
P3.1/TXD
P3.0/RXD
D0
D1
D2
D3
D4
D5
D6
D7
TI
RI

12.2. 模式 1 详述

通过 TXD 发送 10 位数据或通过 RXD 接收 10 位数据：一个起始位(0)，8 个数据位(低位先出)，和一个停止位(1)。在接收时，停止位进入 SCON 的 RB8，波特率由定时器 1 或定时器 2 的溢出速率来决定。见图 133-2 和图 133-6。

使用 SBUF 作为目的寄存器可以使用任何指令来启动传输。写到 SBUF 信号也加载一个 1 到移位寄存器的第 9 位和发送控制单元的标志位，以发出一个传输请求。起始位在 TXD 时，传输开始被激活发送。一位时序后，数据激活，使能输出传送移位寄存器到 TXD。第一个移位脉冲在一位时间之后。

数据位从右移出，0 从左边移入。当数据位的最高位在移位寄存器的输出位置时，接着 1 加载到最高位的左边第个位置，且左边的其它位置为 0，移位控制器移出这最后一位后结束传输并置 TI 为 1。

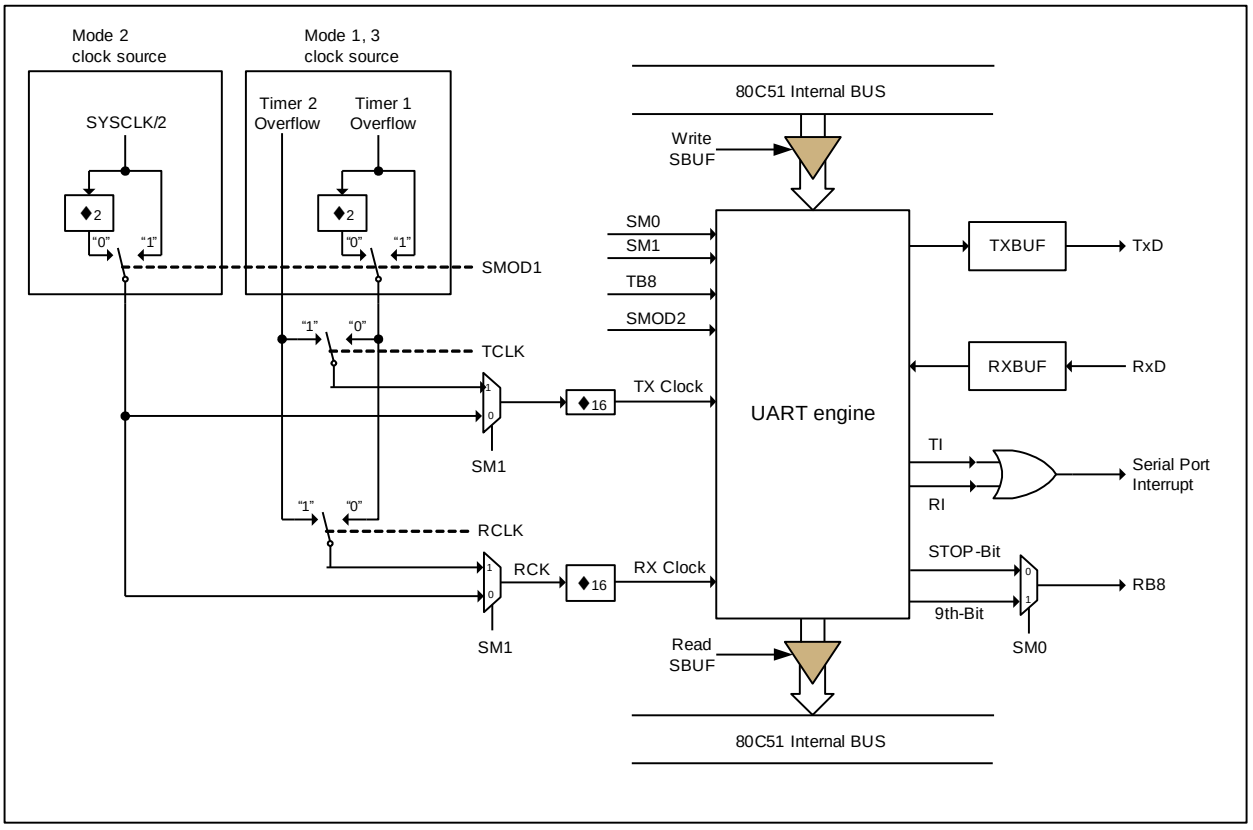
当检测到在 RXD 有 1 到 0 跳变的起始位时接收开始。数据位从右边进入，1 位从左边移出。当起始位到达移位寄存器的最左边位置时(在模式 1 是一个 9 位寄存器)，标志着 RX 接收器移完最后一位将把信息加载到 SBUF 和 RB8，并置位 RI。

在最后一个移位脉冲产生时，还必须同时满足下面两个条件，这次接收才被真正确认：

1. RI=0；
2. SM2=0，或接收数据位的第 9 位等于 1。

如果有一个条件不满足，接收帧将彻底丢失，且 RI 不会置位。如果所有条件都满足，接收的第 9 个数据位进入 RB8，且前 8 个数据位进入到 SBUF。一个位时序后，不论上述条件是否满足，接收单元继续在 RXD 输入引脚检测到 0 转换的起始位。

图 123-6 串口模式 1, 2, 3



12.3. 模式 2、3 详述

通过 TXD 传送 11 位或通过 RXD 接收 11 位：一个起始位(0)，8 个数据位(低位在先)，一个可编程的第 9 个数据位和一个停止位(1)。在传送时，数据的第 9 位(TB8)可分配为 0 或 1。在接收时，数据的第 9 位将进入到 SCON 的 RB8。在模式 2 波特率可编程为 1/16，1/32 或 1/64 的系统时钟频率。模式 3 可以产生可以从定时器 1 或定时器 2 产生可变的波特率。

接收部分和模式 1 相同。与模式 1 传送部分不同的只有传送寄存器的第 9 位。见图 133-2 和图 133-6。

使用 SBUF 作为目的寄存器时可通过任何指令来启动传送。“写到 SBUF”信号也加载 TB8 到传送移位寄存器的第 9 位，且 TX 控制单元的标志请求一个传送。激活发送开始传送，并将起始位放到 TXD。一位时序后，数据激活，使能传送移位寄存器输出到 TXD。第一个移位脉冲发生在第一个移位脉冲之后。第 1 个移位时钟 1(停止位)进入到移位寄存器的第 9 位。此后，只有 0 移入。因此， 数据位向右移出，0 随着时钟从左边移入。当 TB8 到达移位寄存器的输出位置时，接着停止位 1 被移到 TB8 的左边，且停止位左边的位都为零了。

在 RXD 引脚检测到下降沿时接收开始。当检测到一个传送时，启动接收，数据位从右边进入，1 位从左边移出。当起始位到达移位寄存器的最左边位置时(在模式 2 和模式 3 是一个 9 位寄存器)，则标志着 RX 接收器移完最后一位将把信息加载到 SBUF 和 RB8，并置位 RI。

在最后一个移位脉冲产生时，还必须同时满足下面两个条件，这次接收才被真正确认：

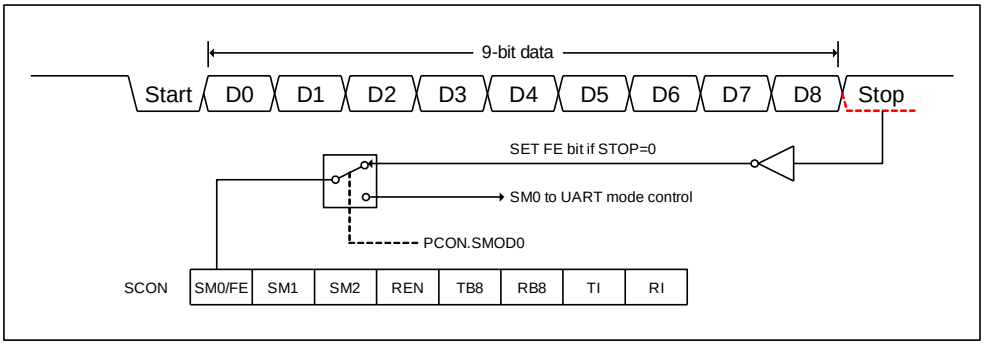
1. RI=0；
2. SM2=0，或接收数据位的第 9 位等于 1。

如果有一个条件不满足，接收帧将彻底丢失，且 RI 不会置位。如果所有条件都满足，接收的第 9 个数据位进入 RB8，且前 8 个数据位进入到 SBUF。一个位时序后，不论上述条件是否满足，接收单元继续在 RXD 输入引脚检测 1 到 0 转换的起始位。

12.4. 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。参考下图。

图 123-7 UART 帧错误检测



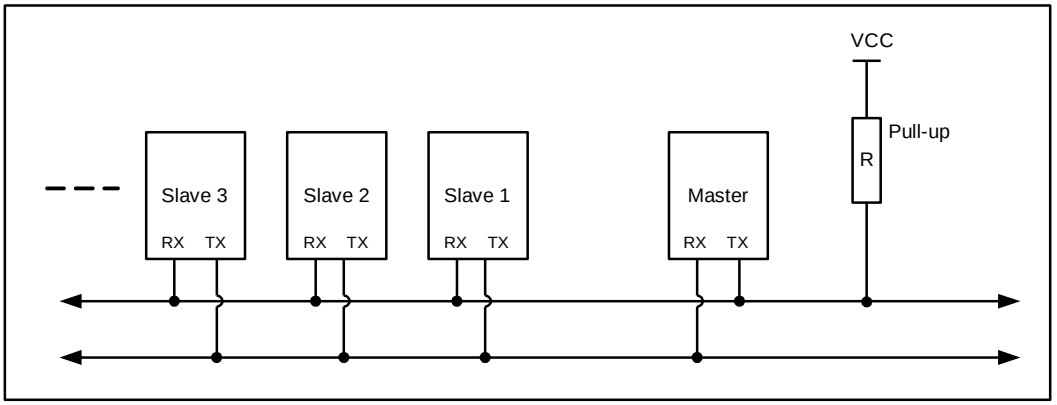
12.5. 多处理器通讯

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，除非收到一个有效的停止位否则接收中断不会被激活。

图 123-8 UART 多处理器通讯



12.6. 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：

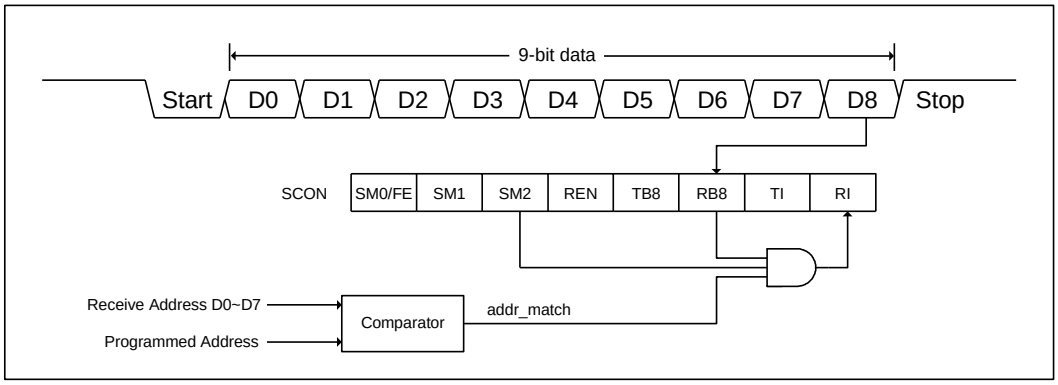
从机 0	从机 1	从机 2
SADDR = 1110 0000	SADDR = 1110 0000	SADDR = 1110 0000
SADEN = 1111 1001	SADEN = 1111 1010	SADEN = 1111 1100
地址 = 1110 0XX0	地址 = 1110 0X0X	地址 = 1110 00XX

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 123-9 自动地址识别



注: (1)收到匹配地址后(addr_match=1),清 SM2 以接收数据字节
(2) 收完全部数据字节后,置 SM2 为1 以等待下一个地址

12.7. 波特率设置

12.7.1. 模式 0 波特率

$$\text{Mode 0 Baud Rate} = \frac{\text{SYSCLK Frequency}}{n}$$

; n=12, if URM0X6=0
; n=2, if URM0X6=1

注:
如果 URM0X6=0,波特率公式同标准 8051.

12.7.2. 模式 2 波特率

12.7.3. 模式 1 & 3 波特率

使用定时器 1 作波特率发生器

使用定时器 2 作波特率发生器

12.8. 串口寄存器

四个操作模式除了波特率设置不同外其它都与标准 8051 相同。PCON, AUXR 和 AUXR2 三个寄存器与波特率设置有关。

SCON: 串口控制寄存器

SFR 地址 = 0x98
SFR 页 = 全部 复位值 = 0000-0000

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

Bit 7: 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。
Bit 7: SM0,串行口模式位 0(SMOD0 必须为 0 来访问 SM0 位)
Bit 6: 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	SYSCLK/12 或 /2
0	1	1	8 位 UART	可变的
1	0	2	9 位 UART	SYSCLK/64, /32, /16 或 /8
1	1	3	9 位 UART	可变的

Bit 5: 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，并且接收到的字节是本机地址或者是一个广播地址；在模式 1，如果 SM2=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本机地址或者是一个广播地址；在模式 0，SM2 可以为 0。

Bit 4: 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。

Bit 3: 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。

Bit 2: RB8, 在模式 2 和 3 时收到的第 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。

Bit 1: 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。其它模式中，在发送停止位之初由硬件置位，任何模式中必须由软件清除。

Bit 0: 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。其它模式中，在接收停止位的中间时刻由硬件置位。任何模式中必须由软件清除。

SBUF: 串口缓冲寄存器
SFR 地址 = 0x99
SFR 页 = 全部 复位值 = XXXX-XXXX

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

Bit 7~0: 在发送和接收时作缓冲寄存器

SADDR: 从机地址寄存器
SFR 地址 = 0xA9
SFR 页 = 全部 复位值 = 0000-0000

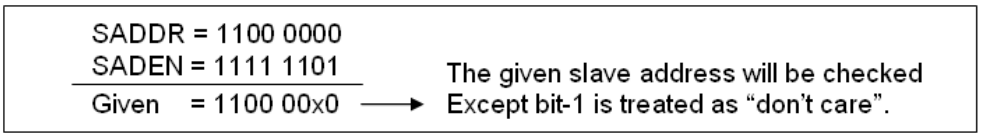
7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

SADEN: 从机地址屏蔽寄存器
SFR 地址 = 0xB9
SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

当地址自动识别功能启用后，可用 SADDR 和 SADEN 来预置地址，事实上，SADEN 是 SADDR 的“屏蔽”寄存器，如下图所示



每个从对象的广播地址为 SADDR 和 SADEN 进行逻辑“或”的结果，结果中为“0”的位将被忽略。在系统复位后，SADDR 和 SADEN 都被初始化为 0，从而忽略“Given”地址的全部地址位和“广播”地址的全部地址位而导致自动地址识别功能无效。

PCON0: 电源控制寄存器 0

SFR 地址 = 0x87

SFR 页 = 全部 复位值 = 00X1-0000

7	6	5	4	3	2	1	0
SMOD1	SMOD0	--	POF	GF1	GF0	PD	IDL
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W

Bit 7: SMOD1, 双倍波特率控制位
0: 禁止 UART 双倍波特率
1: 使能 UART 双倍波特率(模式 1, 2,或 3.)

Bit 6: SMOD0, 帧错误选则
0: SCON.7 作 SM0 功能
1: SCON.7 作 FE 功能。

AUXR2: 辅助寄存器 2

SFR 地址 = 0xA3

SFR 页 = 全部 复位值 = 0000-XX00

7	6	5	4	3	2	1	0
T0X12	T1X12	URM0X6	--	--	--	T1CKOE	T0CKOE
R/W	R/W	R/W	R	R	R	R/W	R/W

Bit 6: T1X12, 当 C/T=0 时，定时器 1 时钟源选择
0: 选择 SYSCLK/12.
1: 选择 SYSCLK 作时钟源

Bit 5: URM0X6, 串口模式 0 波特率选择
0: 选择 SYSCLK/12 作 UART 模式 0 波特率
1: 选择 SYSCLK/2 作 UART 模式 0 波特率

12.9. 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
}
```

13. 模拟比较器

MA807 提供用户一个模拟比较器，当正向输入端 AIN0 电压大于等于负向输入端 AIN1 时比较器输出逻辑 1 否则输出逻辑 0。有 AC_PI0, AC_PI1, AC_PI2 和 AC_PI3 四个模拟信号正向输入脚可供选择，它门通过多路器 PIS[1:0] 切换。MVRS[3:0] 可选择 16 个 AIN1（包括 AC_MI 和 15 段 VDD 参考）。CPU 通过读 ACCON.ACOUT 得到比较器的输出。

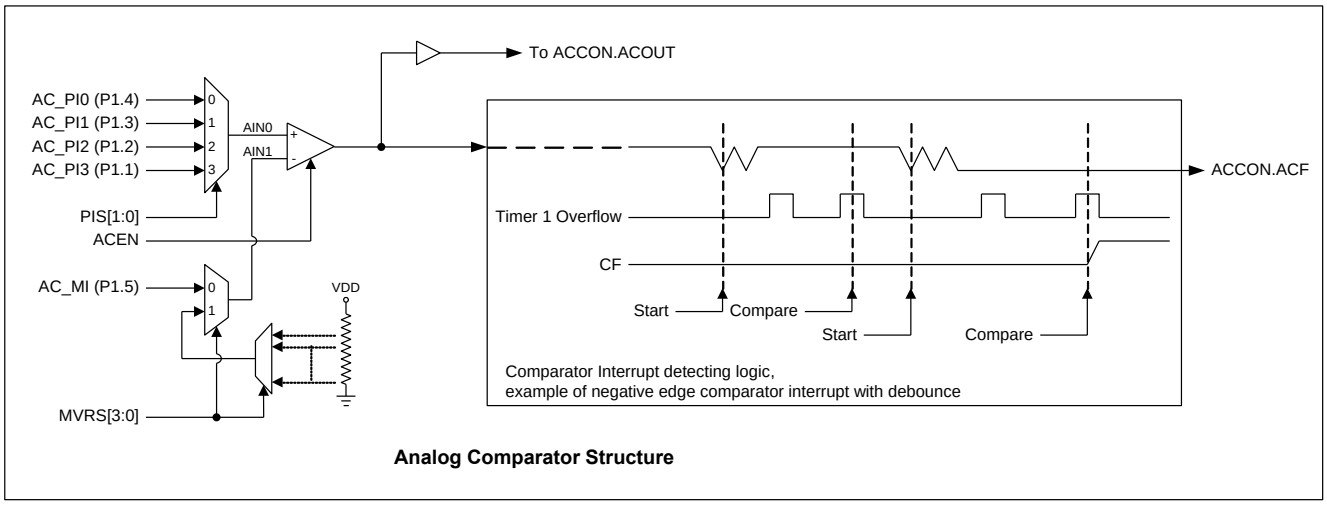
AC_MI(P1.5), AC_PI0(P1.4), AC_PI1(P1.3), AC_PI2(P1.2) 和 AC_PI3(P1.1)默认状态是带有上拉电阻的准双向口 。若作模拟比较器，软件必须配置所选择的比较器输入通道为仅输入模式以排除上拉电阻的影响。当进入掉电模式没有比较器操作时，必须设置 I/O 口为准双向口、推挽输出或开漏集输出以减少 I/O 口的消耗电流。

设置 ACCON. ACEN 使能比较器，第一次使能比较器，它的输出和中断标志在 10 毫秒内是不稳定的，所以在这段时间不要使能比较器中断，并且在使能比较器中断前清除比较器中断标志。

通过 ACCON 的 ACM 可以方便的设定比较器的中断模式。当 ACM 所设定的中断条件发生时比较器中断标志 ACCON.ACF 会被置 1，中断标志 ACF 可以被软件查询，可以用来去产生一个中断，但只能用软件清除。模拟比较器在 CPU 空闲模式 IDLE 或掉电模式下无法工作。

MA807 的模拟比较器支持在空闲模式 IDLE 或掉电模式下唤醒 CPU。这个应用，在空闲模式 IDLE 或掉电模式下通过 ACCON 寄存器中的位 ACIDX 和位 ACPDX 是有条件的使能比较器。如果比较器和它的中断被使能，则比较器能唤醒闲模式 IDLE 或掉电模式。详细功能见 ACCON 寄存器的说明。

13.1. 模拟比较器结构



Analog Comparator Structure

每个时钟周期比较器采样输出，这样的比较器非常适合处理慢速的模拟信号。三种延迟模式可以达到滤波的作用。延迟模式下，定时器 1 产生采样时间，当相应的转换事件发生时，比较器再一次采样时需等待两个定时器溢出。如果两次采样的结果相同则把 ACF 置 1，否则忽略它。滤波器可以通过定时器的溢出进行调整。因为定时器 1 一直不停的在运行，而延迟须等到两个溢出以保证采样点至少有一个溢出周期，所以在初始化边沿比较中断事件后，中断将发生在以后的第 1 个或第 2 个溢出周期之间。

13.2. 模拟比较器寄存器

ACCON: 模拟比较器控制寄存器

SFR 地址 = 0x9E

SFR 页 = 全部 复位值 = 00X0-0000

7	6	5	4	3	2	1	0
ACIDX	ACPDX	ACOUT	ACF	ACEN	ACM2	ACM1	ACM0

R/W	R/W	R	R/W	R/W	R/W	R/W	R/W

Bit 7: ACIDX, CPU 空闲模式（IDLE）比较器开关
0: IDLE 时比较器停止工作
1: IDLE 时比较器继续工作

Bit 6: ACPDX, 掉电模式模拟比较器控制位
0: 掉电模式时比较器停止工作
1: 掉电模式时比较器继续工作
如果 ACEN, ACPDX 和 EACI 被置 1，掉电模式中比较器仅能在低电平或高电平唤醒 CPU。

Bit 5: ACOUT,此位为比较器输出，只读。

Bit 4: ACF. 模拟比较器中断标志
0: 只能软件清 0
1: ACEN=1 且 ACM[2:0]所设定的比较中断发生时自动置 1。中断能被 IE 的 Bit6[EAC]使能或禁止。

Bit 3: ACEN. 模拟比较器使能
0: 禁止。清 0 此位将使比较器强制输出低电平并且防止 ACF 产生中断事件
1: 使能。

Bit 2~0: ACM2 ~ ACM1, 比较器中断模式选择

ACM[2:0]	Interrupt Mode
0 0 0	Negative (Low) level
0 0 1	Positive edge
0 1 0	Toggle with debounce
0 1 1	Positive edge with debounce
1 0 0	Negative edge
1 0 1	Toggle
1 1 0	Negative edge with debounce
1 1 1	Positive (High) level

ACMOD: 模拟比较器模式寄存器
SFR 地址 = 0x9F
SFR 页 = 全部 复位值 = 0000-XX00

7	6	5	4	3	2	1	0
MVRS3	MVRS2	MVRS1	MVRS0	--	--	PIS1	PIS0
R/W	R/W	R/W	R/W	R	R	R/W	R/W

Bit 7~4: MVRS[3:0], 模拟比较器负电压参考选择。这四位决定模拟比较器的负向输入端（V-）的输入源，如下：

MVRS[3:0]	(V-) 输入	MVRS[3:0]	(V-) 输入
0000	AC_MI(P1.5)	1000	8/16 VDD
0001	1/16 VDD	1001	9/16 VDD
0010	2/16 VDD	1010	10/16 VDD
0011	3/16 VDD	1011	11/16 VDD
0100	4/16 VDD	1100	12/16 VDD
0101	5/16 VDD	1101	13/16 VDD
0110	6/16 VDD	1110	14/16 VDD
0111	7/16 VDD	1111	15/16 VDD

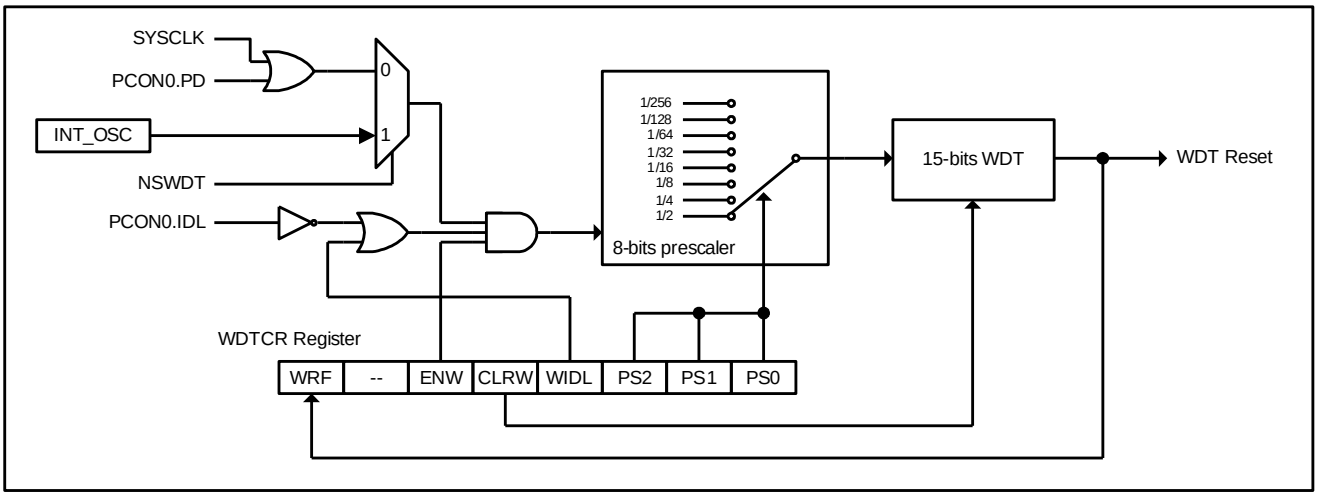
Bit 3~2: 保留.

Bit 1~0: PIS[3:0], 模拟比较器的正电压参考输入。这两位决定模拟比较器的正向输入端（V+）的输入源，如下：

PIS[2:0]	(V+) 输入选择
00	AC_PI0(P1.4)
01	AC_PI1(P1.3)
10	AC_PI2(P1.2)
11	AC_PI3(P1.1)

14. 看门狗 (WDT)

14.1. 看门狗结构



14.2. 看门狗寄存器

WDTCR: 看门狗控制寄存器

SFR 地址 = 0xE1

SFR 页 = 全部 复位值 = 0X00-XXXX

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

Bit 7: WRF, WDT 复位标志。WDT 溢出时，这一位被硬件置位，应当软件清零。

Bit 6: 保留.

Bit 5: 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。)*

Bit 4: 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。

Bit 3: WIDL. WDT 在空闲模式的运行。置位该位会让 WDT 在空闲模式下继续计数。

Bit 2~0: PS2 ~ PS0, 预分频设置，见下表：

PS[2:0]	预分频值
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

14.3. WDT 硬件选项

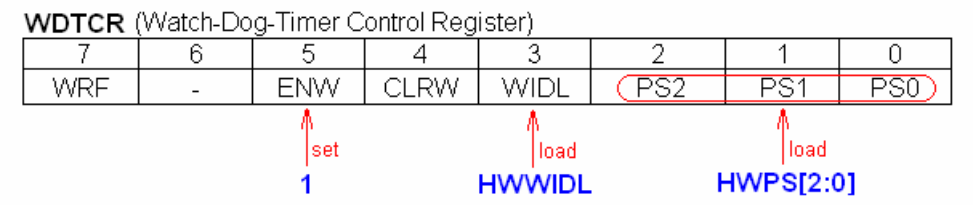
WDTCR 除了软件可以初始化外，也可以使用硬件选项 HWENW, HWWIDL 和 HWPS[2:0]在系统上电时由硬件自动初始化，这些硬件选项可以使用通用编程器或烧写器进行编程。（参考 MCU 的硬件选项）

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
(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



14.4. 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
```

15. 电源控制模块

MA807 支持一个电源监测模块， Brown-Out 侦察器、和两个电源节能模式：空闲模式（IDLE）和掉电模式（Power-Down）。通过 PCON0 和 PCON1 寄存器来访问这些模式。

15.1. 节能模式

15.1.1. 空闲模式（Idle）

可以通过软件的方式置 PCON.IDL 位，使设备进入空闲模式。在空闲模式下，系统不会给 CPU 提供时钟，CPU 状态、RAM、SP、PC、PSW、ACC 被保护起来。I/O 端口也保持当前逻辑。有两种方式是设备从空闲模式唤醒，首先，将“复位”脚连接到高电平来产生一个内部硬件复位可以唤醒空闲模式中的设备，其次任何处于激活状态的中断源都将会清除 PCON.0 而使设备终止空闲模式，并同时进入中断服务程序，只有在中断返回后才会开始执行进入空闲模式指令之后的程序。空闲模式下定时器 0、定时器 1、定时器 2、PWM 定时器和串口仍然处于工作状态，模拟比较器和看门狗有条件的设置在 IDLE 时是否唤醒 CPU。

另外一种使 IDLE 退出的方法是使能唤醒 GPIO 而不需要应用中断能力。

当在 IDLE 模式或掉电模式时，比较器若不用，则作为模拟比较器用的 I/O 口应该设置输出” 0” 或配置为准双向口。

15.1.2. 掉电模式(Power-down)

可以通过软件的方式置 PCON.PD 位，使设备进入掉电模式。在掉电模式下，振荡器被停止，Flash 存储器掉电以节约电能，只有上电电路继续刷新电源，在减少 VDD 的时候 RAM 的内容仍然会被保持，但特殊功能寄存器 SFR 的内容就不一定能保持住。外部复位、上电复位、外部中断或使能的唤醒口（通用 IO 口 GPIO）或使能的没有停止的看门狗定时器能使系统退出掉电模式。

系统复位或刚退出掉电模式后至少要等 4 微秒后才能进入或再次进入掉电模式。

15.1.3. 中断唤醒

外部中断 nINT0 (P3.2), nINT1 (P3.3), nINT2 (P4.3), nINT3 (P4.2), 四个外部中断可以使系统退出掉电模式，但这些中断必须在进入掉电模式前使能并配置成低电平敏感。

唤醒由内部时钟控制，中断口的下降沿系统退出掉电模式，振荡器开始振荡，内部时钟开始计数，但 CPU 要等到内部时钟计数满时才开始执行指令，计数溢出后，中断服务程序开始工作。为了避免中断被重复触发，中断服务程序在返回前应该被禁止，中断口低电平应保持足够长的时间以等系统稳定

15.1.4. 复位唤醒

外部复位脚 RST 唤醒有点类似于中断，复位脚 RST 有上升沿电平时系统退出掉电模式，振荡器开始振荡，内部时钟开始计数，但 CPU 要等到内部时钟计数满时才开始执行指令。复位脚 RST 必须保持足够长时间的高电平以保证系统完全复位，复位脚 RST 变成低电平时便开始执行程序。

值得指出的是当 IDLE 模式被硬件复位唤醒时，前两个机器周期（内部复位没有取得控制权）程序正常从进入 IDLE 模式的后一条指令执行。这时内部硬件是禁止访问内部 RAM 的，但访问 I/O 端口没有被禁止，为了保证不可预料的写 I/O 口，在进入 IDLE 指令后不要放置写 I/O 口或外部存储器的指令（最好加两到三个 NOP 指令）。

15.1.5. 普通 I/O (GPIO) 唤醒

MA807 中如果 P1WKPE 和 GPWKPE 寄存器中设置好了，P1 和普通端口(GPWKS[1:0]设置)具有唤醒的能力。P3.2/nINT0, P3.3/nINT1, P3.2, P3.3, 的中断被禁止但设置了 GPWKPE 寄存器相应位则它们仍然具有唤醒能力。但是 P4.2/nINT3 和 P4.3/nINT2 只能使能了中断才能唤醒系统。

一个使能的 GPIO 脚唤醒掉电模式有点类似于中断，GPIO 脚电平下降沿时系统被唤醒，振荡器开始振荡，内部时钟开始计数，但 CPU 要等到内部时钟计数满时才开始执行指令，计数溢出后，不发生中断，CPU 开始从使其进入掉电模式的下一条指令开始执行程序。也就是说，能唤醒的 GPIO 口只有唤醒功能而不会产生中断。

使能 GPIO 唤醒也具有从 IDLE 唤醒功能。它将恢复 CPU 开始从使其进入掉电模式的下一条指令开始执行程序。

15.2. Brown-Out 侦察器

15.3. 电源控制寄存器

PCON0: 电源控制寄存器 0

SFR 地址 = 0x87

SFR 页 = 全部 复位值 = 0001-0000

7	6	5	4	3	2	1	0
SMOD1	SMOD0	--	POF	GF1	GF0	PD	IDL
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 1: PD, 掉电（Power-Down）控制位
0: CPU 清 0 或任何一个退出 Power-down 事件发生时自动清 0
1: 置 1 则进入掉电模式

Bit 0: IDL, 空闲模式（IDLE 模式）控制位
0: CPU 清 0 或任何一个退出 IDLE 事件发生时自动清 0
1: 置 1 则进入 IDLE 模式

PCON1: 电源控制寄存器 1

SFR 地址 = 0x97

SFR 页 = 全部 复位值 = XXXX-XXX0

7	6	5	4	3	2	1	0
--	--	--	--	--	--	--	BOD
R	**R**	**R**	**R**	**R**	**R**	**R**	**R/W**

Bit 7~1: 保留.

Bit 0: BOD, Brown-Out 侦察标志
0: 必须由软件清 0
1: 若 Brown-Out 侦察器侦察到匹配的工作电压则置位

P1WKPE: 端口 1 唤醒使能控制寄存器

SFR 地址 = 0xD7

SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P17WKP	P16WKP	P15WKP	P14WKP	P13WKP	P12WKP	P11WKP	P10WKP
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~0: 对应的 I/O 脚唤醒使能
0: 禁止端口唤醒功能
1: 使能端口唤醒功能，掉电模式和空闲模式时当端口出现下降沿则唤醒

GPWKPE: 普通 I/O 端口唤醒使能控制寄存器

SFR 地址 = 0xD6

SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	GP6WKP	0	GP4WKP	GP3WKP	GP2WKP	GP1WKP	GP0WKP
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: 保留 0.
Bit 5: 保留 0.

Bit 6, 4~0: 所选择的唤醒端口对应的引脚使能控制位。通过 AUXR1.GPWKS[1:0]选择端口
0: 禁止引脚唤醒功能
1: 使能引脚唤醒功能，掉电模式和空闲模式时当引脚出现下降沿则唤醒

AUXR1: 辅助控制寄存器 1

SFR 地址 = 0xA2
SFR 页 = 全部 复位值 = 0000-0XX0

7	6	5	4	3	2	1	0
GPWKS1	GPWKS0	0	P1S0	GF2	--	--	DPS
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R**	**R**	**R/W**

Bit 7~6: GPWKS[1:0], 跟 GPWKPE 设置唤醒端口

GPWKS[1:0]	选择的端口
0 0	P0
0 1	P2
1 0	P3

15.4. 电源控制示例代码

(1) 功能需求: 选择系统时钟分频为 OSCin/128 的空闲模式 (默认为 OSCin/1)

汇编语言代码范例:

```
CKS0        EQU     01h
CKS1        EQU     02h
CKS2        EQU     04h

    ORL     PCON2,#( CKS2 + CKS1 + CKS0)    ; 设置 CKS[2:0] = “111” 选择 OSCin/128
```

C 语言代码范例:

```
#define     CKS0            0x01
#define     CKS1            0x02
#define     CKS2            0x04

    PCON2 |= (CKS2 | CKS1 | CKS0);          // 系统时钟分频 /128
                                            // CKS[2:0], 系统时钟分频
                                            //     0    | OSCin/1
                                            //     1    | OSCin/2
                                            //     2    | OSCin/4
                                            //     3    | OSCin/8
                                            //     4    | OSCin/16
                                            //     5    | OSCin/32
                                            //     6    | OSCin/64
                                            //     7    | OSCin/128
```

16. 在系统编程 (ISP)

IFD: ISP/IAP Flash 数据寄存器

SFR 地址 = 0xE2

SFR 页 = 全部 复位值 = 1111-1111

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

IFD 为 ISP/IAP 操作的数据寄存器，ISP/IAP 进行读写操作时，IFD 作为数据缓冲区。当用于访问 IAPLB 时，IFD 为 IAPLB 的值。

IFADRH: ISP/IAP 地址高

SFR 地址 = 0xE3

SFR 页 = 全部 复位值 = 0000-0000

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

IFADRH 存放 ISP/IAP 操作的目标地址的高位。

IFADRL: ISP/IAP 地址低

SFR 地址 = 0xE4

SFR 页 = 全部 复位值 = 0000-0000

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

IFADRL 存放 ISP/IAP 操作的目标地址的低位，在进行页擦除时，IFADRL 的值被忽略。

IFMT: ISP/IAP Flash 模式表寄存器

SFR 地址 = 0xE5

SFR 页 = 全部 复位值 = XXXX-0000

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

Bit 7~4: 保留

Bit 3~0: ISP/IAP 操作模式选择

Bit[3:0]	模式
0 0 0 0	空闲状态
0 0 0 1	读 Flash 数据
0 0 1 0	写 Flash 数据
0 0 1 1	擦除 Flash 数据页
0 1 0 0	设置 IAPLB
0 1 0 1	读取 IAPLB
其它	为测试而保留

IFMT 用于选则 ISP/IAP 操作功能

IAPLB: IAP 低边界

SFR 地址=间接地址　　复位值 = 0011-0110

7	6	5	4	3	2	1	0
IAPLB							0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

IAPLB 用于定义 IAP 空间的低边界限，由于 Flash 的页面大小为 512 字节，所以 IAPLB 的值必须为偶数

读 IAPLB 的方法：

```
IFMT = 0x05;
ISPCR = 0x80;
SCMD = 0x46;
SCMD = 0xB9;
    //此时 IFD 中保存的即为 IAPLB 的值
```

设置 IAPLB 的方法：

```
IFD = ??;   //将 IAPLB 的预设值写入 IFD 中
IFMT = 0x04;
ISPCR = 0x80;
SCMD = 0x46;
SCMD = 0xB9;
```

IAP 区域由 IAPLB 和 ISP 起始地址共同决定

IAP 低边界 = IAPLB * 256

IAP 高边界 = ISP 起始地址 – 1

例如：如果 IAPLB=0x36，则 IAP 存储器的范围位于 0x3600~0x39FF 。

需要注意的是 IAPLB 的值不能大于 0x3A00 的起始地址

SCMD: ISP 顺序命令寄存器 / RDID (读 DID 寄存器)

SFR 地址 = 0xE6

SFR 页　　= 全部　　复位值 = xxxx-xxxx

7	6	5	4	3	2	1	0
SCMD							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

ISP/IAP/IAPLB 的操作都需要用 SCMD 寄存器来触发，当 ISPCR.7 为“1”且 SCMD 顺序写入命令“0x46 0xB9”时，ISP 操作被触发。

ISPCR: ISP 控制寄存器

SFR 地址 = 0xE5

SFR 页　　= 全部　　复位值 = 0000-xxxx

7	6	5	4	3	2	1	0
ISPEN	SWBS	SWRST	CFAIL	--	--	--	--
R/W	R/W	R/W	R/W	R	R	R	R

Bit 7: ISPEN, ISP/IAP 使能位
0: 全局禁用 ISP/IAP 编程/擦除/读功能
1: 使能 ISP/IAP 编程/擦除/读功能

Bit 6: SWBS, 软件引导选择位
0: 复位后从 AP 区域启动

1: 复位后从 ISP 区域启动

Bit 5: SWRST, 软件复位触发控制位
0: 无操作
1: 产生软件复位，启动后硬件自动清除它

Bit 4: CFAIL, ISP/IAP 命令是否执行失败
0: ISP/IAP 操作成功
1: ISP/IAP 操作失败

Bit 3~0 : 保留

ISP 详细描述

MA807 不用 IDLE 模式去完成 ISP 操作，而是建立一个 CPU 等待以释放 Flash 存储器供给 ISP 控制电路使用。一旦 ISP 执行完成，CPU 将被唤醒并开始执行调用 ISP 指令的下一条指令，整个 ISP 操作期间中断服务程序被封锁。

ISP 控制电路有一个内部定时器产生时序去控制，可以通过 CKCON2.XCKS[4:0]去设定以得到精确的擦除/编程时序。

17. 在应用程序编程 (IAP)

MA807 可编程程序存储器（AP-memory）+ 可编程数据存储器（IAP-memory）大小被限制为 **14.5**K。

当 MCU 从应用程序区 AP 启动时，应用程序只能存取 AP 区域以及 IAP 区域。

17.1. ISP/IAP 示例代码

(1). 功能需求: ISP/IAP Flash 读的子程序

汇编语言代码范例:

```
IxP_Flash_Read    EQU        01h
ISPEN             EQU        80h

_ixp_read:
ixp_read:

    MOV     ISPCR,#ISPEN                    ; 功能使能
    MOV     IFMT,# IxP_Flash_Read           ; ixp_read=0x01

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

    MOV     SCMD,#046h                      ;
    MOV     SCMD,#0B9h                      ;

    MOV     A,IFD                           ; 现在读出来的数据存在于 IFD 中

    MOV     IFMT,#000h                      ; Flash_Standby=0x00
    ANL     ISPCR,#(0FFh – ISPEN)           ; 禁止功能

    RET
```

C 语言代码范例:

```
#define     Flash_Standby           0x00
#define     IxP_Flash_Read          0x01
#define     ISPEN                   0x80


unsigned char ixp_read (void)
{
    unsigned char arg;
    ISPCR = ISPEN;                      //功能使能
    IFMT = IxP_Flash_Read;              // IxP_Read=0x01
```

```
    IFADRH = ??
    IFADRL = ??

    SCMD = 0x46;                                    //
    SCMD = 0xB9;                                    //

    arg = IFD;

    IFMT = Flash_Standby;                   // Flash_Standby=0x00
    ISPCR &= ~ISPEN;

    return arg;
}
```

(2). 功能需求: ISP/IAP Flash 檫除的子程序

汇编语言代码范例:

```
IxP_Flash_ Erase    EQU             03h
ISPEN               EQU             80h

_ixp_erase:
ixp_erase:

    MOV     ISPCR,#ISPEN                        ; 功能使能
    MOV     IFMT,# IxP_Flash_Erase          ; ixp_erase=0x03

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

    MOV     SCMD,#046h                  ;
    MOV     SCMD,#0B9h                  ;

    MOV     IFMT,#000h                          ; Flash_Standby=0x00
    ANL     ISPCR,#(0FFh – ISPEN)           ; 禁止功能

    RET
```

C 语言代码范例:

```
#define     Flash_Standby           0x00
#define     IxP_Flash_Erase         0x03
#define     ISPEN                   0x80


void ixp_erase (unsigned char Addr_H, unsigned char Addr_L)
{
    ISPCR = ISPEN;                          //功能使能
    IFMT = IxP_Flash_Erase;                 // IxP_Erase=0x03

    IFADRH = Addr_H;
    IFADRL = Addr_L;

    SCMD = 0x46;                            //
```

```
    SCMD = 0xB9;                                        //


    IFMT = Flash_Standby;                       // Flash_Standby=0x00
    ISPCR &= ~ISPEN;
}
```

(3). 功能需求: ISP/IAP Flash 写的子程序

汇编语言代码范例:

```
IxP_Flash_Program EQU     02h
ISPEN             EQU     80h

_ixp_program:
ixp_program:

    MOV     ISPCR,#ISPEN                    ; 功能使能
    MOV     IFMT,# IxP_Flash_Program        ; ixp_program=0x03

    MOV     IFADRH,??                       ; 填写[IFADRH,IFADRL] 字节地址
    MOV     IFADRL,??
    MOV     IFD, A                          ; 现在,要写的数据存在于 A 累加器中

    MOV     SCMD,#046h                      ;
    MOV     SCMD,#0B9h                      ;

    MOV     IFMT,#000h                      ; Flash_Standby=0x00
    ANL     ISPCR,#(0FFh – ISPEN)           ; 禁止功能

    RET
```

C 语言代码范例:

```
#define     Flash_Standby           0x00
#define     IxP_Flash_Program       0x02
#define     ISPEN                   0x80

void ixp_program(unsigned char Addr_H, unsigned char Addr_L, unsigned char dta)
{
    ISPCR = ISPEN;                          //功能使能
    IFMT = IxP_Flash_Program;               // IxP_Program=0x02

    IFADRH = Addr_H;
    IFADRL = Addr_L;
    IFD = dta;
```

```
    SCMD = 0x46;                                    //
    SCMD = 0xB9;                                    //

    IFMT = Flash_Standby;                           // Flash_Standby=0x00
    ISPCR &= ~ISPEN;


}
```

18. 辅助特殊功能寄存器 SFR

AUXR0: 辅助控制寄存器 0

SFR 地址 = 0x8E

SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P60OC1	P60OC0	P60FD	P34FD	--	--	EXTRAM	GF
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6: P60 输出配置控制位 1 和 0。P60 支持下面的选择为 GPIO 或时钟源发生器。当 P60OC[1:0] 索引为非 P60 功能时，XTAL2 将驱动片上 RC 振荡器输出为其它设备提供时钟源。

P60OC[1:0]	XTAL2 功能	I/O 模式
00	P60	准双向口
01	INTOSC	推挽输出
10	INTOSC/2	推挽输出
11	INTOSC/4	推挽输出

Bit 5: P60FD, P6.0 快速驱动
0: P6.0 默认驱动输出
1: P6.0 快速驱动输出使能。如果 P6.0 被配置作时钟输出，使能这一位。

Bit 4: P34FD, P3.4 快速驱动
0: P3.4 默认驱动输出
1: P3.4 快速驱动输出使能。

Bit 1: EXTRAM, 外部数据 RAM 使能
0: 使能片上扩展数据 RAM (XRAM 256 字节)
1: 禁止片上扩展数据 RAM

Bit 0: 保留

AUXR1: 辅助控制寄存器 1

SFR 地址 = 0xA2

SFR 页 = 全部 复位值 = 0000-0000

7	6	5	4	3	2	1	0
GPWKS1	GPWKS0	0	P1S0	GF2	--	--	DPS
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6: GPWKS[1:0], 跟 GPWKPE 设置唤醒端口

GPWKS[1:0]	选择的端口
0 0	P0
0 1	P2
1 0	P3
1 1	P5

Bit 5: 保留 0

Bit 4: P1S0, 串口 P1.7/P1.6
0: 保持 P3.0/P3.1 作串口脚 RXD/TXD
1: 切换 P1.6/P1.7 作串口脚 RXD/TXD

Bit 3: GF2，通用标志 2

Bit 2~1: 保留

Bit 0: DPTR 选择位，用来在 DPTR0 和 DPTR1 之间切换。
0: 选择 DPTR0
1: 选择 DPTR1

AUXR2: 辅助寄存器 2

SFR 地址 = 0xA3

SFR 页 = 全部 复位值 = 000X-XX00

7	6	5	4	3	2	1	0
T0X12	T1X12	URM0X6	--	--	--	T1CKOE	T0CKOE
R/W	**R/W**	**R/W**	**R**	**R**	**R**	**R/W**	**R/W**

Bit 7: T0X12，当 C/T=0 时，定时器 0 的时钟源选择。
0: 选择 SYSCLK/12 作时钟源
1: 选择 SYSCLK 作时钟源

Bit 6: T1X12，当 C/T=0 时，定时器 1 的时钟源选择。
0: 选择 SYSCLK/12 作时钟源
1: 选择 SYSCLK 作时钟源

Bit 5: URM0X6，串口模式 0 波特率选择
0: 选择 SYSCLK/12 作 UART 模式 0 波特率
1: 选择 SYSCLK/2 作 UART 模式 0 波特率

Bit 3~2:保留

Bit 1: T1CKOE，定时器 1 时钟输出使能
0: 禁止定时器 1 时钟从 P3.5 输出
1: 使能定时器 1 时钟从 P3.5 输出

Bit 0: T0CKOE，定时器 0 时钟输出使能
0: 禁止定时器 0 时钟从 P3.4 输出
1: 使能定时器 0 时钟从 P3.4 输出

19. 指令集

Rn 暂存器 R0~R7

direct 8 位内部存储器，包括

1. 内部存储器(00~7F)的地址
2. 特殊功能寄存器(80~FF)的地址，如 P0,PSW,TMOD,…等

@Ri 由寄存器 R0 或 R1 所索引的内部 RAM 数据

#data 8 位常数

#data16 16 位常数

Addr16 16 位的目的地址，可使跳转指令跳转 64K

Addr11 11 位的目的地址，可使跳转指令跳转 2K

rel 有正负号的 8 位地址偏移量，用于相对地址的跳转

bit 1 个 bit: 指所有可以位寻址的的位元

A 累加器 Acc

C 或 CY 进位标志

AC 辅助进位标志

Bb 指定位元 B0~B7

D 半位元组(4bit)

F0 旗号 0

I 中断

PC 程序计数器

SP 堆栈

B 寄存器 B

DPTR 程序数据地址寄存器

@ 间接寻址符号

$ 程序计数器当前的值

reg 寄存器

数据传送			
助记符	描述	字节数	指令周期
MOV A, Rn	Acc ← Rn	1	1
MOV A, direct	Acc ← direct	2	2
MOV A, @Ri	Acc ← Ri	1	2
MOV A, #data	Acc ← data	2	2
MOV Rn,A	Rn ← Acc	1	2
MOV Rn,direct	Rn ← direct	2	4
MOV Rn,#data	Rn ← data	2	2
MOV direct,A	direct ← Acc	2	3
MOV direct,Rn	direct ← Rn	2	3

MOV direct,direct	direct ← direct	3	4
MOV direct,@Ri	direct ← Ri	2	4
MOV direct,#data	direct ← data	3	3
MOV @Ri,A	Ri ← Acc	1	3
MOV @Ri,direct	Ri ← direct	2	3
MOV @Ri,#data	Ri ← data	2	3
MOV DPTR,#data16	DPTR ← 16bit data	3	3
MOVC A,@A+DPTR	Acc ← (A+DPTR)地址所指的数据	1	4
MOVC A,@A+PC	Acc ← (A+PC)地址所指的数据	1	4
PUSH direct	堆栈 ← direct	2	4
POP direct	direct ← 堆栈	2	3
XCH A,Rn	A 和 Rn 互换	1	3
XCH A,direct	A 和 direct 互换	2	4
XCH A,@Ri	A 和 Ri 互换	1	4
XCHD A,@Ri	A 和 Ri 的低四互换	1	4
算术运算			
ADD A,Rn	Acc ← Acc+Rn	1	2
ADD A,direct	Acc ← Acc+direct	2	3
ADD A,@Ri	Acc ← Acc+Ri	1	3
ADD A,#data	Acc ← Acc+data	2	2
ADDC A,Rn	Acc ← Acc+Rn+C	1	2
ADDC A,direct	Acc ← Acc+direct+C	2	3
ADDC A,@Ri	Acc ← Acc+Ri+C	1	3
ADDC A,#data	Acc ← Acc+data+C	2	2
SUBB A,Rn	Acc ← Acc-Rn-C	1	2
SUBB A,direct	Acc ← Acc-direct-C	2	3
SUBB A,@Ri	Acc ← Acc-Ri-C	1	3
SUBB A,#data	Acc ← Acc-data-C	2	2
INC A	Acc ← Acc +1	1	2
INC Rn	Rn ← Rn +1	1	3
INC direct	direct ← direct +1	2	4
INC @Ri	Ri ← Ri +1	1	4
INC DPTR	DPTR ← DPTR +1	1	1
DEC A	Acc ← Acc - 1	1	2
DEC Rn	Rn ← Rn -1	1	3
DEC direct	direct ← direct -1	2	4
DEC @Ri	Ri ← Ri -1	1	4

MUL AB	两数相乘，结果高八位存入 B,低八位存入 A	1	4
DIV AB	Acc 除以 B, 商存入 Acc,余数存入 B	1	5
DA A	Acc 作十进制调整	1	4
逻辑运算			
ANL A,Rn	Acc ← Acc and Rn	1	2
ANL A,direct	Acc ← Acc and direct	2	3
ANL A,@Ri	Acc ← Acc and Ri	1	3
ANL A,#data	Acc ← Acc and data	2	2
ANL direct,A	Direct ← direct and Acc	2	4
ANL direct,#data	Direct ← direct and data	3	4
ORL A,Rn	Acc ← Acc or Rn	1	2
ORL A,direct	Acc ← Acc or direct	2	3
ORL A,@Ri	Acc ← Acc or Ri	1	3
ORL A,#data	Acc ← Acc or data	2	2
ORL direct,A	Direct ← direct or Acc	2	4
ORL direct,#data	Direct ← direct or data	3	4
XRL A,Rn	Acc ← Acc xor Rn	1	2
XRL A,direct	Acc ← Acc xor direct	2	3
XRL A,@Ri	Acc ← Acc xor Ri	1	3
XRL A,#data	Acc ← Acc xor data	2	2
XRL direct,A	Direct ← direct xor Acc	2	4
XRL direct,#data	Direct ← direct xor data	3	4
CLR A	清除累加器 Acc	1	1
CPL A	累加器反相	1	2
RL A	累加器向左旋转	1	1
RLC A	累加器和 C 左旋	1	1
RR A	累加器向右旋转	1	1
RRC A	累加器和 C 右旋	1	1
SWAP A	累加器的高低四位互换	1	1
位逻辑运算			
CLR C	清除进位标记	1	1
CLR bit	清除直接位元	2	4
SETB C	设定进位标记	1	1
SETB bit	设定直接位元	2	4
CPL C	进位标记取反	1	1
CPL bit	直接位元取反	2	4
ANL C,bit	C ← C and bit	2	3

ANL C,/bit	C ← C and bit(反相)	2	3
ORL C,bit	C ← C or bit	2	3
ORL C,/bit	C ← C or bit(反相)	2	3
MOV C,bit	C ← bit	2	3
MOV bit,C	bit ← bit	2	4
位逻辑跳转			
JC rel	如果 C=1 跳转到 rel	2	3
JNC rel	如果 C=0 跳转到 rel	2	3
JB bit,rel	如果 bit=1 跳转到 rel	3	4
JNB bit,rel	如果 bit=0 跳转到 rel	3	4
JBC bit,rel	如果 bit=1 跳转到 rel,并且清除 bit	3	5
程序跳转			
ACALL addr11	绝对式子程序调用	2	6
LCALL addr16	远程子程序调用	3	6
RET	从子程序返回	1	4
RETI	从中断返回	1	4
AJMP addr11	绝对式跳转	2	3
LJMP addr16	远程跳转	3	4
SJMP rel	短程跳转	2	3
JMP @A+DPTR	间接跳转	1	3
JZ rel	如果 Acc=0 则跳到 rel	2	3
JNZ rel	如果 Acc≠0 则跳到 rel	2	3
CJNE A,direct,rel	如果 Acc≠direct 则跳到 rel	3	5
CJNE A,#data,rel	如果 Acc≠data 则跳到 rel	3	4
CJNE Rn,#data,rel	如果 Rn≠data 则跳到 rel	3	4
CJNE @Ri,#data,rel	如果 Ri≠data 则跳到 rel	3	5
DJNZ Rn,rel	如果(Rn-1)≠0 则跳到 rel	2	4
DJNZ direct,rel	如果(direct-1)≠0 则跳到 rel	3	5
NOP	无动作	1	1

20. 最大绝对额定参数

MA807

参数	额定值	单位
环境温度偏差	-40 ~ +85	°C
存储温度	-65 ~ + 150	°C
IO 口和复位脚的对地电压	-0.5 ~ VDD + 0.5	V
VDD 脚的对地电压	-0.5 ~ +6.0	V
芯片总电流	400	mA
IO 口的最大吸收电流	40	mA

*注意：实际参数超过上述各项“绝对最大额定值”可能会对设备造成永久性损坏。这些参数是一个设备进行正常功能操作的最大额定值，任何超过上述各项的条件都不被建议，否则可能会影响设备运行的稳定性。

21. 电气特性

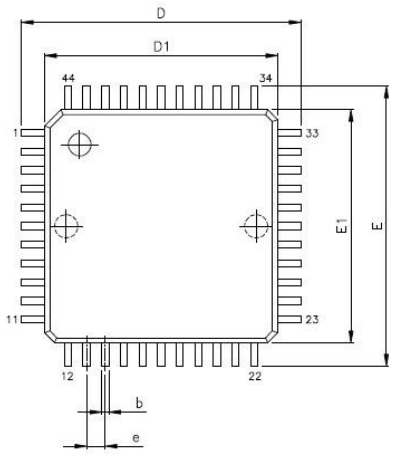
21.1. 直流特性

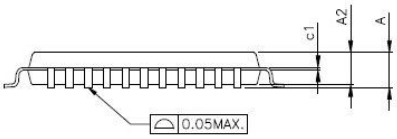
VSS = 0V, TA = 25 ℃, VDD = 5.0V

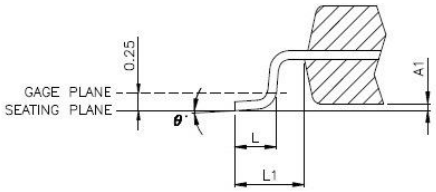
符号	参数	条件	范围			单位
			最小	标称	最大	
V_{IH1}	输入高电压 P0/P1/P2/P3/P4/P6 (Quasi, Input-only or Open-drain)		2.0			V
V_{IH2}	输入高电压, RST		3.5			V
V_{IL1}	输入低电压, P0/P1/P2/P3/P4/P6 (Quasi, Input-only or Open-drain)				0.8	V
V_{IL2}	输入低电压, RST				1.6	V
I_{IH1}	输入高的漏电流 P0/P1/P2/P3/P4/P6 (Quasi, Input-only or Open-drain)	V_{PIN} = VDD		0	10	uA
I_{IL1}	输入低的电流 P0/P1/P2/P3/P4/P6 (Quasi-bidirectional)	V_{PIN} = 0.4V		20	50	uA
I_{IL2}	输入低的电流 P0/P1/P2/P3/P4/P6 (Input-only or Open-drain)	V_{PIN} = 0.4V		0	10	uA
I_{H2L}	输入下降沿的跳变电流 P0/P1/P2/P3/P4/P6 (Quasi-bidirectional)	V_{PIN} =1.8V		250	500	uA
I_{OH1}	输出高的电流 P0/P1/P2/P3/P4/P6 (Quasi-bidirectional)	V_{PIN} =2.4V	-150	-220		uA
I_{OH2}	输出高的电流 P0/P1/P2/P3/P4/P6 (Push-pull output)	V_{PIN} =2.4V	-12			mA
I_{OL1}	输出低的电流 P0/P1/P2/P3/P4/P6 (Quasi, Open-drain or Push-pull output)	V_{PIN} =0.4V	12			mA
I_{OP}	工作电流	F_{OSC} = 12MHz		14	20	mA
I_{IDLE}	空闲模式电流	F_{OSC} = 12MHz		6	10	mA
I_{PD}	掉电模式电流			0.1	10	uA
R_{RST}	复位脚上的内部下拉电阻			100		Kohm
V_{CM}	Comparator input common mode voltage		0		VDD	V
V_{OS}	Comparator input offset voltage			10	30	mV
V_{BOD}	Brown-out detection voltage	F_{OSC} = 12MHz	3.95	4.0	4.05	V

22. 封装尺寸

LQFP-44







VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

SYMBOLS	MIN.	NOM.	MAX.
A	—	—	1.60
A1	0.05	—	0.15
A2	1.35	1.40	1.45
c1	0.09	—	0.16
D		12.00 BSC	
D1		10.00 BSC	
E		12.00 BSC	
E1		10.00 BSC	
e		0.80 BSC	
⚠ b (w/o plating)	0.25	0.30	0.35
L	0.45	0.60	0.75
L1		1.00 REF	
θ°	0°	3.5°	7°

NOTES:
1. JEDEC OUTLINE:MS-026 BCB
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.ALLOWABLE 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., Ltd.

3RD ANGLE SYS.	比例 SCALE	材質 MTRL	製程 FINISH	數量 QTY.
繪圖: CHK. 陳曉玲 日期: DATE: 2/20/08	圖名: TITLE: LOW PROFILE PLASTIC QUAD FLAT PACKAGE DATA SHEET 44 LEADS			
審核: CHK. Erik 日期: DATE: 2/20/08	圖號: DWG# MW-AD44-001			
核准: APPL. Rex 日期: DATE: 11/10/11	檔案: FILE: MW-AD44-001-03		版別: REV: 03	張數: QTY: 1

23. 修订历史

版本	日期	页	描述
V0.03	2009/Nov./13		初稿
V0.04	2010/Jan./04		AP IAP 空间容量修正
V0.05	2010/01/07		修改存储器地址文字错误
V0.06	2010/07/07	74 6	修改环境温度偏差 -40～+85 增加内部晶振频漂范围
A1.0	2014/02/17		编排修改，增加例程

免责声明

在此，笙泉（Megawin）代表“*Megawin Technology Co., Ltd.*”

生命支援

此产品并不是为医疗、救生或维持生命而设计的，并且当设备系统出现故障时，并不能合理地预示是否会对人身造成伤害。因此，当客户使用或出售用于上述应用的产品时，需要客户自己承担这样做的风险，笙泉公司并不会对不当地使用或出售我公司的产品而造成的任何损害进行赔偿。

更改权

笙泉保留产品的如下更改权，其中包括电路、标准单元、与/或软件 - 在此为提高设计的与/或性能的描述或内容。当产品在大批量生产时，有关变动将通过工程变更通知（ECN）进行通知。