MEGAWIN
MAKE YOU WIN

8051 内核-单片机

MA86E/L104

说明书

版本: **A1.3**

2015/05 version A1.3

功能

- 1-T 80C51 微处理器
- **MA86E/L104** 4K 的程序存储器空间
  - ISP 程序空间可以选择 0.5KB/1KB/1.5KB……3.5KB
  - 软件灵活的定义 IAP 空间大小
  - 密码保护程序区访问
  - Flash 写/擦 次数： 2000 次@ IAP 空间 0.5KB, 1000 次@ IAP 空间 1KB, …
  - Flash 数据保留时间： 100 年 25℃
  - MA82E/L104 出厂默认空间设置
    - AP 程序空间（0000h~07FFh）
    - IAP 数据空间（0800h~0BFFh）
    - ISP 引导码空间（0C00h~0FFFh）（保留在线烧录用，用户程序无法更动）
- 内部 256 字节数据存储器
- 中断控制
  - 6 中断源, 4 个优先级
  - 二个外部中断， nINT0 和 nINT1
  - 所有的外部中断支持高/低或上升/下降沿触发
- 两个 16-位 定时/计数, Timer 0 和 Timer 1.
  - T0CKO 在 P34 和 T1CKO 在 P35.
  - T0/T1 可以选择 X12 模式
- 增强型 UART (S0)
  - 帧误差侦测
  - 自动地址识别
  - 速度增强机制(X2/X4 模式)
  - 4 通道 UART 重复机构
- 所有 I/O 具有键盘中断功能.
- 可程序控制的看门狗定时器，时钟来源为 ILRCO
  - 通过 CPU 或上电复位一次性使能
  - 看门狗溢出会中断或复位 CPU
  - 掉电模式支持看门狗功能(Wath Mode)
- 20 脚的封装片最多 18 个 I/O
  - P3 可以设置准双向口模式，推挽输出模式，开漏集输出模式，仅输入模式。
  - P1, P4.0 和 P4.1 可以设置为推挽输出模式，开漏集输出模式。
  - P4.0, P4.1 和 P3.6 公用 XTAL2, XTAL1 和 RST.
  - 所有的 I/O 有唤醒功能。
- 多种省电模式：掉电模式， 空闲模式，慢频模式，副频模式，watch 模式和 monitor 模式。
  - 所有的中断能唤醒空闲模式。
  - 6 中源能唤醒掉电模式。

- 慢频模式和副频模式支持低速 MCU 运转
- Watch 模式在掉电模式下能复位 CPU
- Monitor 模式在掉电模式下支持 BOD0 复位 CPU (仅仅 L-系列)

- 低电压检测： VDD 4.0V E-系列 和 VDD 2.6V L-系列
  - 中断 CPU 或 复位 CPU
  - 在掉电模式下唤醒 CPU (仅仅 L-系列)
- 工作电压：
  - MA86E104: 4.2V~5.5V，要求 Flash 写操作（ISP/IAP/ICP）最小电压 4.5V
  - MA86L104: 2.4V~3.6V.，要求 Flash 写操作（ISP/IAP/ICP）最小电压 2.7V
- 工作频率范围：25MHz(最大)
  - MA86E104: 0 – 12MHz @ 4.2V – 5.5V 和 0 – 25MHz @ 4.5V – 5.5V
  - MA86L104: 0 – 12MHz @ 2.4V – 3.6V 和 0 – 25MHz @ 2.7V – 3.6V
- 时钟源种类：
  - 内部震荡 22.118MHz/24MHz (IHRCO)：典型的，工厂校对到±1%。
  - 外部晶振模式
  - 内部低功耗 64KHz RC 振荡(ILRCO)
  - 外部时钟输入(ECKI) XTAL2/P4.0
  - 内部震荡输出 XTAL2/P4.0
- 工作温度：
  - 工业级 (-40℃ to +85℃)*
- 封装类型：
  - SOP20: MA86E/L104AS20
  - SOP16: MA86E/L104AS16
  - SOP8: MA86E/L104AS8

*:抽样检测

目录

1. 概述

MA86E/L104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1~6 时钟信号 (比标准的8051快 6~7倍)，与标准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，**MA86E/L104**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。

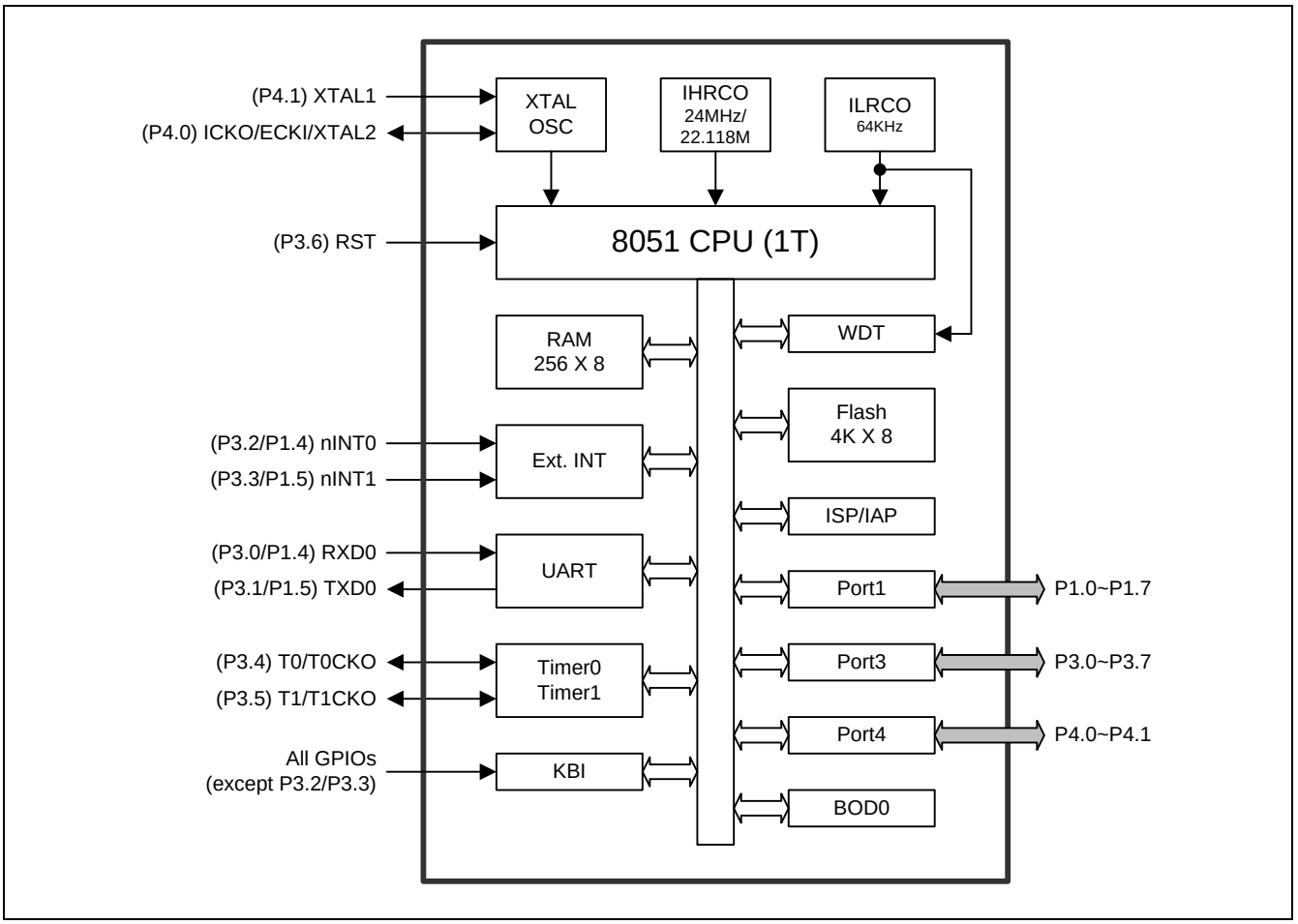
MA86E/L104有 4K字节的内置Flash存储器用于保存代码和数据。Flash存储器可以通过串行模式编程 (ICP, 在电路编程) 或者 ISP模式 (在系统编程)进行编程的能力。同时，也提供在应用编程(IAP)的能力。ISP和ICP让使用者无需从产品中取下微控制器就可以下载新的代码；IAP意味着应用程序正在运行时，微控制器能够在Flash中写入非易失数据。这些功能都由内建的电荷泵提供编程用的高压。

MA86E/L104除了80C52 MCU的标准功能（例如 256 字节的随机存储器，二个8位I/O口，二个带有高/低触发选项的外部中断，一个多源4级中断控制和二个16位定时/计数器)外， **MA86E/L104** 有二个额外的 I/O 口 (P4.0 和 P4.1)，键盘中断，一次性使能的看门狗定时器。一个低电压检测器，一个外部的晶振，一个外部的晶振 (与 P4.0 和 P4.1共用)， 一个高精度的内部振荡(IHRCO)，一个低速的内部 RC 振荡器 (ILRCO) 和一个多功能的增型的串口(EUART) 和一个低速增强设备 (X2/X4 模式).

MA86E/L104 有多种工作模式可以减少耗电量：空闲，掉电模式,慢频模式，副频模式，watch 模式和 monitor 模式。 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的值被保存，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多种中断或复位唤醒。在慢频模式， 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。选择副频模式系统时钟来自内部低速振荡器CPU 用一个特别慢的速度在运行。watch 模式, 在掉电模式或空闲模式下保持WDT正常运行来唤醒CPU。Monitor 模式，在掉电模式检测电压，当电压特别低的时候会复位。

2. 方框图

图 2–1. 方框图


(P4.1) XTAL1
(P4.0) ICKO/ECKI/XTAL2
XTAL OSC
IHRCO 24MHz/ 22.118M
ILRCO 64KHz
(P3.6) RST
8051 CPU (1T)
RAM 256 X 8
WDT
Flash 4K X 8
(P3.2/P1.4) nINT0
(P3.3/P1.5) nINT1
Ext. INT
ISP/IAP
(P3.0/P1.4) RXD0
(P3.1/P1.5) TXD0
UART
Port1
P1.0~P1.7
(P3.4) T0/T0CKO
(P3.5) T1/T1CKO
Timer0 Timer1
Port3
P3.0~P3.7
Port4
P4.0~P4.1
All GPIOs (except P3.2/P3.3)
KBI
BOD0

3. 特殊功能寄存器

3.1. SFR 图

表 3–1. SFR 图

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8	--	--	--	--	--	--	--	--
F0	B	--	--	--	--	--	--	--
E8	P4	--	--	--	--	--	--	--
E0	ACC	WDTCR	IFD	IFADRH	IFADRL	IFMT	SCMD	ISPCR
D8	--	--	--	--	--	--	--	--
D0	PSW	--	--	--	--	--	P3KBIE	P1KBIE
C8	--	--	--	--	--	--	--	--
C0	--	--	--	--	--	--	--	CKCON0
B8	IP0L	SADEN	--	--	--	--	--	--
B0	P3	P3M0	P3M1	P4M0	PUCON0	--	--	IP0H
A8	IE	SADDR	--	--		EIE1	EIP1L	EIP1H
A0	--	AUXR0	AUXR1	AUXR2	--	--	--	--
98	SCON	SBUF	--	--	--	--	--	--
90	P1	P1M0	--	--	--	--	--	PCON1
88	TCON	TMOD	TL0	TL1	TH0	TH1	SFIE	--
80	--	SP	DPL	DPH	--	--	--	PCON0
	0/8	**1/9**	**2/A**	**3/B**	**4/C**	**5/D**	**6/E**	**7/F**

3.2. SFR 位分配

表 3–2. SFR 位分配

标号	描述	地址	位地址和标号								复位值
			Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	
SP	Stack Pointer	81H									00000111B
DPL	Data Pointer Low	82H									00000000B
DPH	Data Pointer High	83H									00000000B
PCON0	Power Control 0	87H	SMOD1	SMOD0	GF	POF	GF1	GF0	PD	IDL	00010000B
TCON	*Timer Control*	*88H*	*TF1*	*TR1*	*TF0*	*TR0*	*IE1*	*IT1*	*IE0*	*IT0*	*00000000B*
TMOD	Timer Mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00000000B
TL0	Timer Low 0	8AH									00000000B
TL1	Timer Low 1	8BH									00000000B
TH0	Timer High 0	8CH									00000000B
TH1	Timer High 1	8DH									00000000B
SFIE	System Flag INT En.	8EH	UTIE	--	--	--	KBIFIE	--	BOF0IE	WDTFIE	0xxx0x00B
P1	*Port 1*	*90H*	*P1.7*	*P1.6*	*P1.5*	*P1.4*	*P1.3*	*P1.2*	*P1.1*	*P1.0*	*11111111B*
P1M0	P1 Mode Register 0	91H	P1M0.7	P1M0.6	P1M0.5	P1M0.4	P1M0.3	P1M0.2	P1M0.1	P1M0.0	00000000B
PCON1	Power Control 1	97H	SWRF	EXRF	--	--	KBIF	--	BOF0	WDTF	00xx0x00B
SCON	*Serial Control*	*98H*	*SM0/FE*	*SM1*	*SM2*	*REN*	*TB8*	*RB8*	*Ti*	*RI*	*00000000B*
SBUF	Serial Buffer	99H									xxxxxxxxB
AUXR0	Auxiliary Register 0	A1H	P40OC1	P40OC0	P40FD	GF	P1FS1	P1FS0	INT1H	INT0H	00000000B
AUXR1	Auxiliary Register 1	A2H	RTX3E	RTX2E	RTX1E	RTX0E	GF	GF	RXCS1	RXCS0	00000000B
AUXR2	Auxiliary Register 2	A3H	**URXR**	**BTI**	URM0X6	SMOD2	**T1X12**	**T0X12**	T1CKOE	T0CKOE	00000000B
IE	*Interrupt Enable*	*A8H*	*EA*	--	--	*ES*	*ET1*	*EX1*	*ET0*	*EX0*	*00000000B*
SADDR	Slave Address	A9H									00000000B
EIE1	Extended INT Enable 1	ADH	--	--	--	--	ESF	--	--	--	xxxx0xxxB
EIP1L	Ext. INT Priority 1 Low	AEH	--	--	--	--	PSFL	--	--	--	xxxx0xxxB
EIP1H	Ext. INT Priority 1 High	AFH	--	--	--	--	PSFH	--	--	--	xxxx0xxxB
P3	*Port 3*	*B0H*	*P3.7*	*P3.6*	*P3.5*	*P3.4*	*P3.3*	*P3.2*	*P3.1*	*P3.0*	*11111111B*
P3M0	P3 Mode Register 0	B1H	P3M0.7	P3M0.6	P3M0.5	P3M0.4	P3M0.3	P3M0.2	P3M0.1	P3M0.0	00000000B
P3M1	P3 Mode Register 1	B2H	P3M1.7	P3M1.6	P3M1.5	P3M1.4	P3M1.3	P3M1.2	P3M1.1	P3M1.0	00000000B
P4M0	P4 Mode Register 0	B3H	--	--	--	--	--	--	P4M0.1	P4M0.0	xxxxxx00B
PUCON0	Pull-Up Control 0	92H	--	PU40	--	--	PU11	PU10	--	--	x0xx00xxB
IP0H	Interrupt Priority 0 High	B7H	--	--	--	PSH	PT1H	PX1H	PT0H	PX0H	00000000B
IP0L	*Interrupt Priority Low*	*B8H*	--	--	--	*PSL*	*PT1L*	*PX1L*	*PT0L*	*PX0L*	*00000000B*
SADEN	Slave Address Mask	B9H									00000000B
CKCON0	Clock Control 0	C7H	AFS	--	--	--	--	SCKS2	SCKS1	SCKS0	0xxxx000B
PSW	*Program Status Word*	*D0H*	*CY*	*AC*	*F0*	*RS1*	*RS0*	*OV*	*F1*	*P*	*00000000B*
P3KBIE	P3 KBI Enable	D6H	P37KBIE	P36KBIE	P35KBIE	P34KBIE	P41KBIE	P40KBIE	P31KBIE	P30KBIE	00000000B
P1KBIE	P1 KBI Enable	D7H	P17KBIE	P16KBIE	P15KBIE	P14KBIE	P13KBIE	P12KBIE	P11KBIE	P10KBIE	00000000B
ACC	*Accumulator*	*E0H*	*ACC.7*	*ACC.6*	*ACC.5*	*ACC.4*	*ACC.3*	*ACC.2*	*ACC.1*	*ACC.0*	*00000000B*
WDTCR	Watch-dog-timer Control register	E1H	WREN	NSW	ENW	CLW	WIDL	PS2	PS1	PS0	0x000000B
IFD	ISP Flash data	E2H									11111111B
IFADRH	ISP Flash address High	E3H									00000000B
IFADRL	ISP Flash Address Low	E4H									00000000B
IFMT	ISP Mode Table	E5H	--	--	--	--	--	MS2	MS1	MS0	xxxxx000B
SCMD	ISP Serial Command	E6H									xxxxxxxxB
ISPCR	ISP Control Register	E7H	ISPEN	BS	SRST	CFAIL	--	--	--	--	0000xxxxB
P4	*Port 4*	*E8H*	--	--	--	--	--	--	*P4.1*	*P4.0*	*xxxxxx11B*
B	*B Register*	*F0H*	*B.7*	*B.6*	*B.5*	*B.4*	*B.3*	*B.2*	*B.1*	*B.0*	*00000000B*

3.3. 辅助 SFR 图 (Page P)

MA86E/L104 特殊功能寄存器（SFR）有一个辅助索引 P 页，它写的方法跟标准的 8051 特殊功能寄存器的不一样。象访问 ISP/IAP 一样通过设置 IFMT 和 SCMD 来访问这个辅助的特殊功能寄存器。P 页有 256 字节有用到的为 5 个物理字节地址和 4 个逻辑字节地址。5 个物理字节地址包括 IAPLB, CKCON2, PCON2, SPCON0 和 DCON0。4 个逻辑字节地址包括 PCON0, PCON1, CKCON0 和 WDTCR。在 0~F 写这 4 个逻辑地址会得到相同的特殊功能（SFR）值。 更多的细节请参考 P 页访问章节。

表 3–3. 辅助 SFR 图 (Page P)

		0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8		--	--	--	--	--	--	--	--
F0		--	--	--	--	--	--	--	--
E8		--	--	--	--	--	--	--	--
E0		--	WDTCR	--	--	--	--	--	--
D8		--	--	--	--	--	--	--	--
D0		--	--	--	--	--	--	--	--
C8		--	--	--	--	--	--	--	--
C0		--	--	--	--	--	--	--	CKCON0
B8		--	--	--	--	--	--	--	--
B0		--	--	--	--	--	--	--	--
A8		--	--	--	--	--	--	--	--
A0		--	--	--	--	--	--	--	--
98		--	--	--	--	--	--	--	--
90		--	--	--	--	--	--	--	PCON1
88		--	--	--	--	--	--	--	--
80		--	--	--	--	--	--	--	PCON0
78		--	--	--	--	--	--	--	--
70		--	--	--	--	--	--	--	--
68		--	--	--	--	--	--	--	--
60		--	--	--	--	--	--	--	--
58		--	--	--	--	--	--	--	--
50		--	--	--	--	--	--	--	--
48		SPCON0	--	--	--	DCON0	--	--	--
40		CKCON2	--	--	--	PCON2	--	--	--
38		--	--	--	--	--	--	--	--
30		--	--	--	--	--	--	--	--
28		--	--	--	--	--	--	--	--
20		--	--	--	--	--	--	--	--
18		--	--	--	--	--	--	--	--
10		--	--	--	--	--	--	--	--
08		--	--	--	--	--	--	--	--
00		--	--	--	IAPLB	--	--	--	--
		0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F

3.4. 辅助 SFR 位分配 (Page P)

表 3–4. 辅助 SFR 位分配 (Page P)

S 标号	描述	地址	位地址和标号								复位值
			Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	
Physical Bytes											
IAPLB	IAP Low Boundary	03H	IAPLB6	IAPLB5	IAPLB4	IAPLB3	IAPLB2	IAPLB1	IAPLB0	--	00001000B
CKCON2	Clock Control 2	40H	0	0	XTALE	IHRCOE	0	0	OSCS1	OSCS0	00010000B
PCON2	Power Control 2	44H	0	AWBOD0	0	0	0	0	BO0RE	**1**	00000001B
SPCON0	SFR Page Control 0	48H	0	0	0	WRCTL	0	CKCTL0	PWCTL1	PWCTL0	00000000B
DCON0	Device Control 0	4CH	HSE	IAPO	0	0	0	0	RSTIO	0	10000010B
Logical Bytes											
PCON0	Power Control 0	87H	SMOD1	SMOD0	GF	POF	GF1	GF0	PD	IDL	00010000B
PCON1	Power Control 1	97H	SWRF	EXRF	--	**--**	KBIF	--	BOF0	WDTF	00xx0x00B
CKCON0	Clock Control 0	C7H	AFS	--	--	--	--	SCKS2	SCKS1	SCKS0	0xxxx001B
WDTCR	Watch-dog-timer Control register	E1H	WREN	NSW	ENW	CLW	WIDL	PS2	PS1	PS0	00000000B xxx0xxxxB

写入 P 页 SFR 例程：

```
IFADRH = 0x00;
ISPCR = ISPEN;                  //使能 IAP/ISP 功能
IFMT = MS2;                     // P 页写，IFMT =0x04
IFADRL = SPCON0;                //相对应的 P 页 SFR 设置 P 页 SFR 地址
IFD |= CKCTL0;                  // 设置 CKCTL0
SCMD = 0x46;                    //
SCMD = 0xB9;                    //
IFMT = Flash_Standby;           // IAP/ISP 备用模式，IFMT =0x00
ISPCR &= ~ISPEN;
```

4. 引脚结构

4.1. 封装指南

图 4–1. MA86E/L104AS20 顶视图

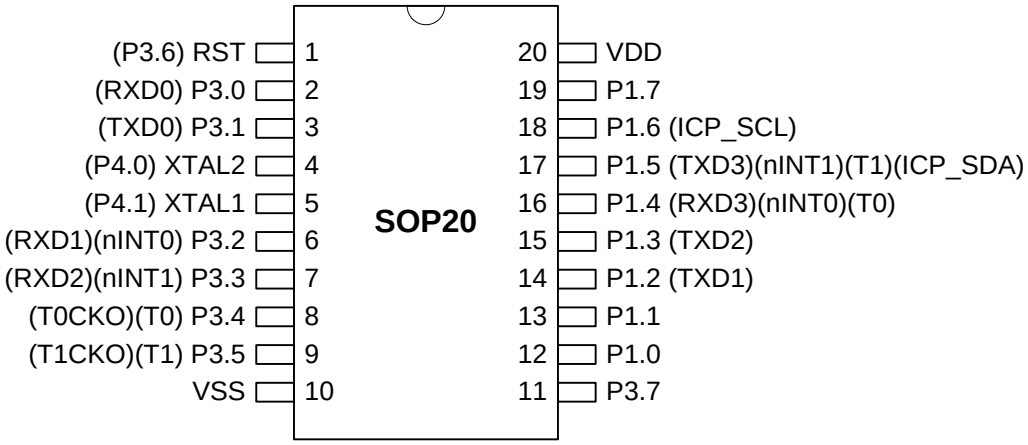


图 4－2. MA86E/L104AS16 顶视图

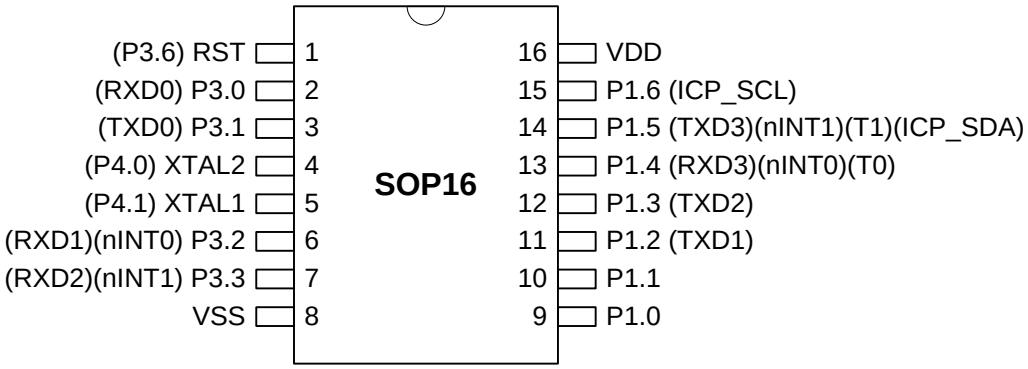
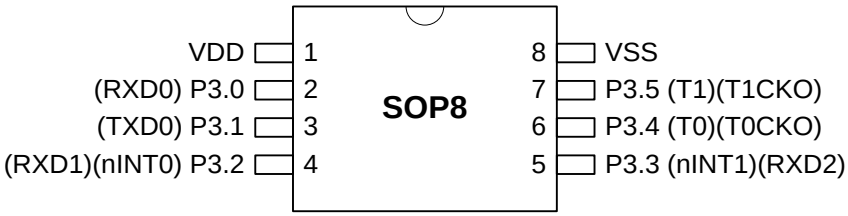


图 4–3. MA86E/L104AS8 顶视图



4.2. 引脚定义

表 4–1. 引脚定义

助记符	引脚号			I/O 类型	描叙
	20-Pin SOP	16-Pin SOP	8-Pin SOP		
P1.0	12	9	--	I/O	* Port 1.0.
P1.1	13	10	--	I/O	* Port 1.1.
P1.2 (TXD1)	14	11	--	I/O	* Port 1.2. * TXD1: TXD channel 1 output of UART repeater.
P1.3 (TXD2)	15	12	--	I/O	* Port 1.3. * TXD2: TXD channel 2 output of UART repeater.
P1.4 (RXD3) (nINT0) (T0)	16	13	--	I/O	* Port 1.4. * RXD3: RXD channel 3 input of UART repeater. * nINT0: Alternate function for nINT0 input. * T0: Alternate function for T0 input.
P1.5 (TXD3) (nINT1) (T1) **(ICP_SDA)**	17	14	--	I/O	* Port 1.5. * TXD3: TXD channel 3 output of UART repeater. * nINT1: Alternate function for nINT1 input. * T1: Alternate function for T1 input. * ICP_SDA: Serial data of ICP interface.
P1.6 **(ICP_SCL)**	18	15	--	I/O	* Port 1.6. * ICP_SCL: Serial clock of ICP interface.
P1.7	19	--	--	I/O	* Port 1.7.
P3.0 (RXD0)	2	2	2	I/O	* Port 3.0. * RXD0: UART0 serial input port.
P3.1 (TXD0)	3	3	3	I/O	* Port 3.1. * TXD0: UART0 serial output port.
P3.2 (nINT0) (RXD1)	6	6	4	I/O	* Port 3.2. * nINT0: external interrupt 0 input. * RXD3: RXD channel 3 input of UART repeater.
P3.3 (nINT1) (RXD2)	7	7	5	I/O	* Port 3.3. * nINT1: external interrupt 1 input. * RXD3: RXD channel 3 input of UART repeater.
P3.4 (T0) (T0CKO)	8	--	6	I/O	* Port 3.4. * T0: Timer/Counter 0 external input. * T0CKO: programmable clock-out from Timer 0.
P3.5 (T1) (T1CKO)	9	--	7	I/O	* Port 3.5. * T1: Timer/Counter 1 external input. * T1CKO: programmable clock-out from Timer 1.
P3.7	11	--	--	I/O	* Port 3 bit-7.
P4.0 (XTAL2) (ECKI) (ICKO)	4	4	--	I/O O I O	* Port 4.0. *XTAL2: Output of on-chip crystal oscillating circuit. * ECKI: In external clock input mode, this is clock input pin. * ICKO: IHRCO clock output.
P4.1 (XTAL1)	5	5	--	I/O I	* Port 4.1 * XTAL1: Input of on-chip crystal oscillating circuit.
RST (P3.6)	1	1	--	I I/O	** RST: External RESET input, high active. * Port 3.6. RST has the alternate function for P3.6.
VDD	20	16	1	P	Power supply input.
VSS	10	8	8	G	Ground, 0 V reference.

4.3. 备选功能转换

许多 I/O 口，除了有普通的 I/O 功能外，还得为内置的外设提供备选功能。默认状态下，端口 1 和端口 0 为外设 UART,Timer0,Timer1,nINT0 和 nINT1 提供备选功能。然而，用户可以通过设置相对应的控制位选择 P1.4 和 P1.5 提供备选功能。这是特别有用的软件编程。

AUXR0: 辅助寄存器 0

SFR 页 = Normal

SFR 地址 = 0xA1 复位值 = 000x-0000

7	6	5	4	3	2	1	0
P40OC1	P40OC0	P40FD	--	P1FS1	P1FS0	INT1H	INT0H
R/W	**R/W**	**R/W**	**W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6: P4.0 功能配置控制位 1 和 0。这两位仅在系统时钟来源选择为 IHRCO 时有效。在 crystal 模式，P4.0 和 P4.1 的功能是 XTAL2 和 XTAL1。在外部时钟输入模式，P4.0 被定义为时钟输入引脚。在 IHRCO 工作时，P4.0 提供作为 GPIO 或时钟源发生器的选项。当 P40OC[1:0]索引到非 P4.0GPIO 功能时，P4.0 将驱动 IHRCO 输出为其它设备提供时钟源。

P40OC[1:0]	P4.0 功能	I/O 模式
00	P40	By P4M0.0
01	IHRCO	By P4M0.0
10	IHRCO/2	By P4M0.0
11	IHRCO/4	By P4M0.0

当 P4.0 作为时钟输出功能时，相对应的设置 P4M0.0 为 “1” 它能选择 P4.0 作为推挽输出模式

Bit 3~2: P1.4 和 P1.5 备选功能选项.

P1FS[1:0]	P1.4	P1.5
00	P1.4	P1.5
01	**--**	Output for TXD1
10	Input for nINT0	Input for nINT1
11	Input for T0	Input for T1

5. 8051 CPU 功能描叙

5.1. CPU 寄存器

PSW: **程序状态字**

SFR 地址 = 0xD0　　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
AC: 辅助进位标志
F0: 用户可设定的标志位 0
RS1: 寄存器组选择位 1
RS0: 寄存器组选择位 0
OV: 溢出标志
F1: 用户可设定的标志位 1
P: 奇偶标志

程序状态字 (PSW) 包含反映 CPU 当前状态的几个状态位。PSW 属于特殊功能寄存器 SFR 区，包含进位标志，辅助进位标志 (应用于 BCD 操作)， 两个寄存器组选择位，溢出标志， 奇偶标志和两个用户可设定的标志位。

进位标志，不仅有算术运算的进位功能，也充当许多布尔运算的“累加器”。

RS0 和 RS1 被用来选择 4 组中的任意一组寄存器组，详见内部 RAM 章节 7.2。

奇偶位反映 1S 内累加器数字和的状况， 1S 内累加器中数字和是奇数则 P=1 否则 P=0 。

SP: **堆栈指针**

SFR 地址 = 0x81　　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

堆栈指针保持栈顶位置，每执行一个 PUSH 指令，会自动增加，缺损值为 0X07H。

DPL: **数据指针低字节**

SFR 地址 = 0x82　　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

DPL 是 DPTR 的低字节，DPTR 用来间接访问 XRAM 和程序空间。

DPH: **数据指针高字节**

SFR 地址 = 0x83　　RESET = 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

DPH 是 DPTR 的高字节，DPTR 用来间接访问 XRAM 和程序空间。

ACC: 累加器

SFR 地址 = 0xE0　　复位值 = 0000-0000

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

算术运算的累加器.

***B: B* 寄存器**

SFR 地址 = 0xF0　　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**

另一个算术运算的累加器

5.2. CPU 时序

MA86E/L104是基于80C51的高效1-T结构的单芯片微处理器，与8051指令集兼容，每条指令需要1~6个时钟信号(比标准8051快6~7倍)。使用流线型结构同标准的8051结构比较大大增加了指令完成的速度 ，指令的时序也和标准的8051不同。

多数8051执行指令，一个区别是建立在机器周期和时钟周期之间，机器周期来自2到12个时钟周期长度。然而，1-T结构的80C51执行指令是基于单独的时钟周期时序。所有指令时序被指定在时钟周期期间。关于1T-80C51指令更详细的说明，请参考“指令集”，这里有每一条指令的助记符、字节数、时钟周期数。

5.3. CPU 寻址 模式

直接寻址*(DIR)*

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。

间接寻址*(IND)*

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 或堆栈指针，16 位地址的地址寄存器只能是 16 位的“数据指针”寄存器，DPTR。

寄存器操作（寻址）*(REG)*

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。

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。

立即寻址*(IMM)*

常量的数值可以在程序存储器中跟随操作码。

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。

6. 存储器组织

像所有的 80C51 一样，**MA86E/L104** 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 4K 字节。在 **MA86E/L104** 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。

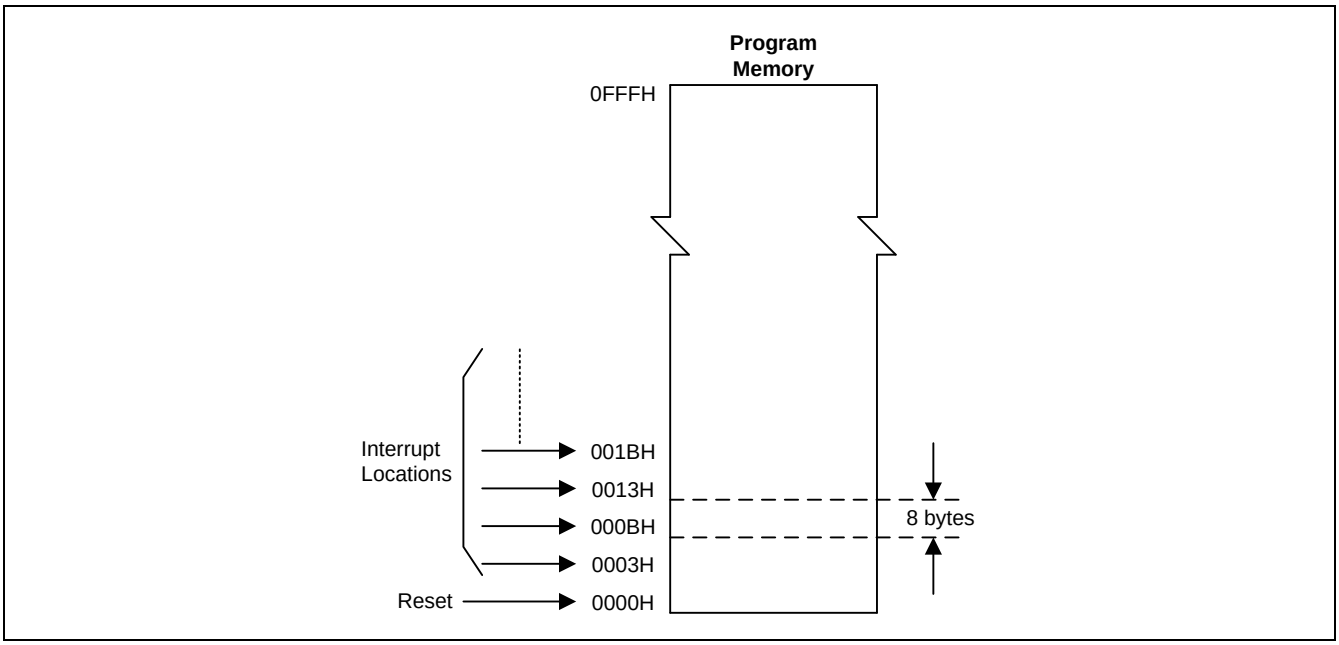
数据存储器使用与程序存储器不同的地址空间。**MA86E/L104** 只有 256 字节的内部没有任何外部的数据存储器。

6.1. 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，如图 7-1 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 7-1 程序存储器



6.2. 数据存储器

图 7-2 向MA86E/L104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（80H—FFH），但他们实际上是分开的。

如图 7-3 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。

图 7-4 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。

.

图 7-2 数据存储器

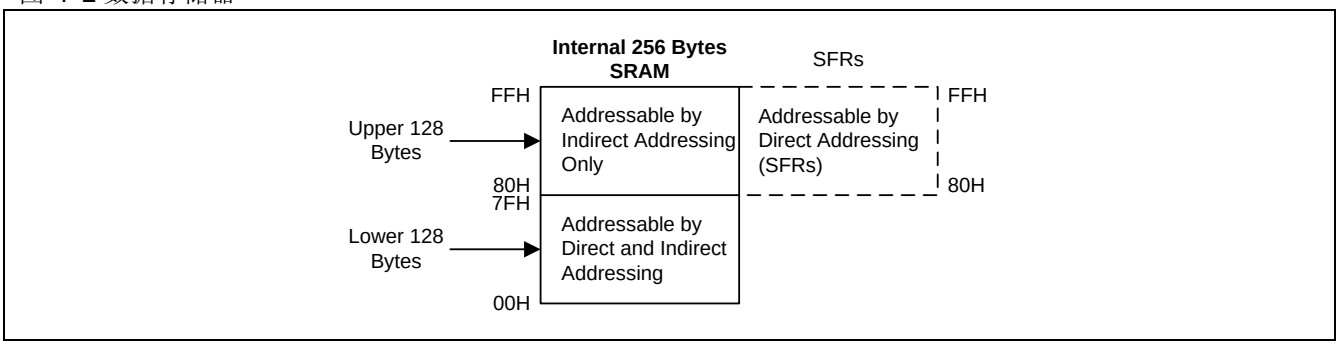


图 7-3 内部 RAM 的低 128 字节

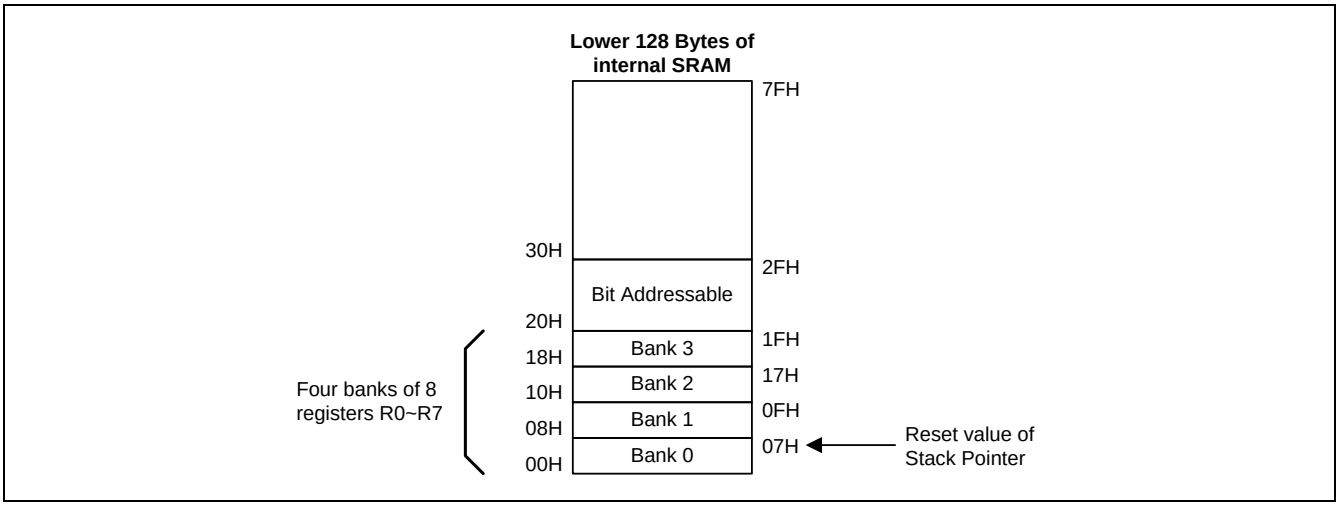
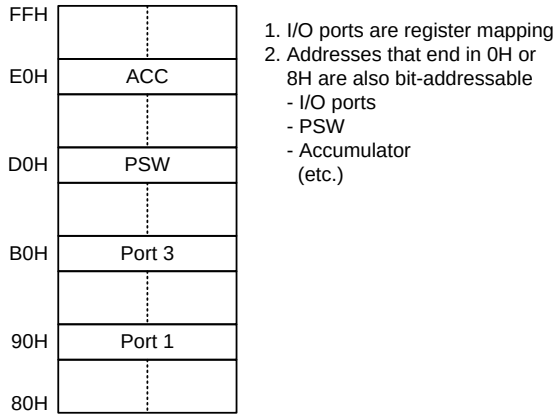


图 7-4 SFR 空间


FFH
E0H
ACC
D0H
PSW
B0H
Port 3
90H
Port 1
80H
1. I/O ports are register mapping
2. Addresses that end in 0H or 8H are also bit-addressable
- I/O ports
- PSW
- Accumulator
(etc.)

6.3. 关于 C51 编译器的声明标识符

C51 编译器的声明识别符与 MA86E/L104 存储空间的对应关系如下：

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
没有外部数据或片上的扩展 RAM (XRAM)。

pdata
没有分页的外部数据(256 字节) 或片上的扩展 RAM（XRAM)。

code
4K 程序存储空间。通过“MOVC @A+DTPR”访问，作为程序部分被读取。

7. 数据指针寄存器 (DPTR)

MA86E/L104 DPTR 只有一种设置。 MA86E/L104 不支持访问外部存储器和 MOVX 指令。

8. 系统时钟

系统时钟有 4 个时钟源：内部快频 RC 震荡器 (IHRCO)，外部晶振 ，内部慢频 RC 震荡器(ILRCO) 和外部频率输入。如图 9-1 所示 **MA86E/L104 系统时钟结构**。

MA86E/L104 默认值是 IHRCO **24.0MHz** MHz 2 分频作为系统时钟，并保留晶振脚 P4.0/P4.1 普通 I/O 口的特性。软件可以根据应用要求自由切换 4 种时钟的任意一种作为系统时钟， 但必须等时钟稳定后才能切换。如果软件选择外部时钟模式， 脚 P4.0 和 P4.1 分配给 XTAL2 和 XTAL1. 并且 P4.0/P4.1 普通 I/O 功能失效。 在外部时钟输入模式 (ECKI)， 时钟源来自 P4.0 ， P4.1 仍然是普通 I/O 口。

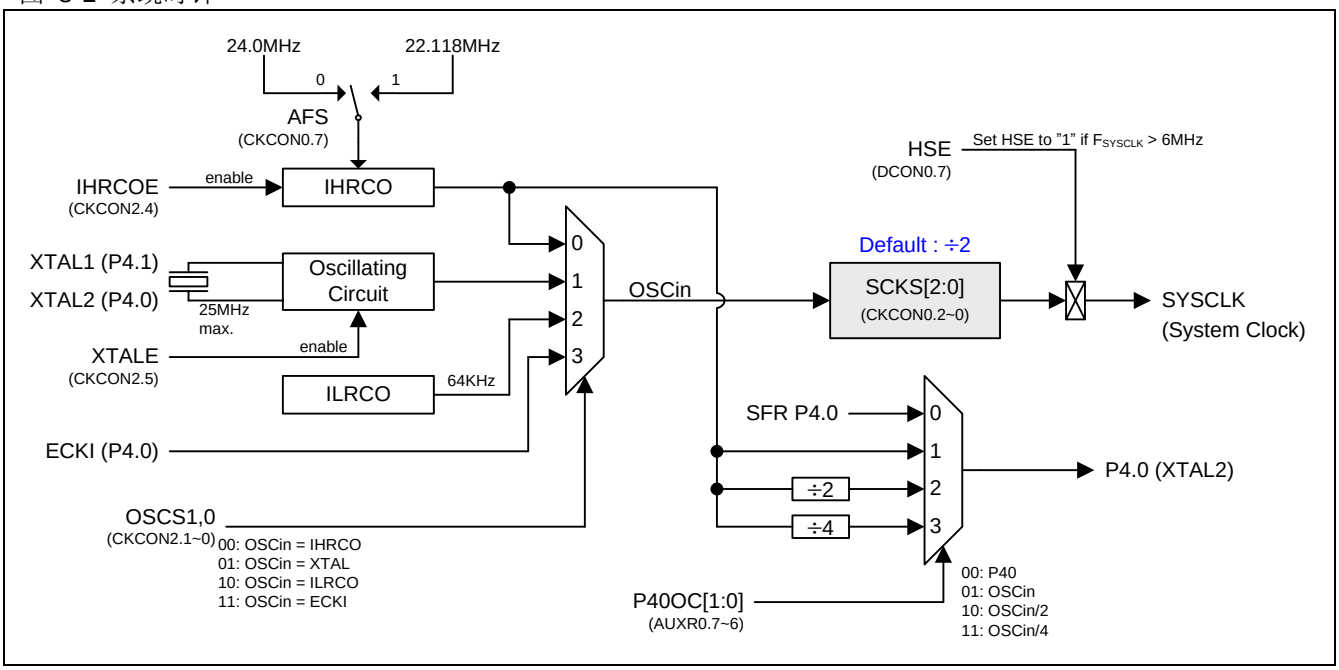
软件可以选择 IHRCO 两种频率，另一种是 **22.118MHz**。 通过软件设置位 AFS （CKCON0.7）来选择， 两种 IHRCO 频率 22.118MHz 和 24.0 MHz 都是高精度，在全工作电压和温度范围内频漂在 **±4%** 以内。 在 IHRCO 或 ILRCO 模式， P4.0 可以作为全时钟（*OSCin*）输出或 2 分频时钟（*OSCin/2*）输出或 4 分频时钟（*OSCin/4*）输出给其他系统时钟源应用。

时钟分配器分配 4 种时钟源的一种为系统时钟 *SYSCLK*， 如下图所示：图 8–1。 用户能通过设置 SCKS2~SCKS0 位（ CKCON0 寄存器）来获得理性的时钟。在上电复位后系统时钟的默认是 2 分频。

8.1. 时钟结构

图 8–1 显示了 **MA86E/L104** 的时钟系统. 系统时钟可以于外部晶振或内部振荡作为来源。

图 8-1 系统时钟



8.2. 时钟寄存器

CKCON0:时钟控制寄存器 0

SFR 页 = 普通 & P 页

SFR 地址 = 0xC7 复位值 = 0xxx-x001

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

Bit 7: AFS, 频率选择
0: 选择 IHRCO 输出 **24.0MHz**。
1: 选择 IHRCO 输出 **22.118MHz**。

Bit 6~3: 保留位。 写 CKCON0 时，这 4 个位必须写“0” 。

Bit 2~0: SCKS2 ~ SCKS0, 系统时钟分频器选择位， SCKS[2:0] 缺损值“001” 选择的系统时钟是 OSCin/2.

SCKS[2:0]	System Clock
0 0 0	OSCin
0 0 1	OSCin /2
0 1 0	OSCin /4
0 1 1	OSCin /8
1 0 0	OSCin /16
1 0 1	OSCin /32
1 1 0	OSCin /64
1 1 1	OSCin /128

CKCON2:时钟控制寄存器 2

SFR Page = P

SFR 地址 = 0x40 复位值 = **0001-0000**

7	6	5	4	3	2	1	0
0	0	XTALE	IHRCOE	0	0	OSCS1	OSCS0
W	**W**	**R/W**	**R/W**	**W**	**W**	**R/W**	**R/W**

Bit 7~6: 保留位。写 CKCON0 时，这 2 个位必须写“0”

Bit 5：XTALE， 外部晶振(XTAL) 使能。
0：禁止 XTAL 震荡电路。 在这种情况下， XTAL2 XTAL1 表现为 Port 4.0 Port 4.1.
1：使能 XTAL 震荡电路。如果软件设置这个位，必须等待 **3 ms** XTAL 才能稳定输出。

Bit 4： IHRCOE，内部快频 RC 震荡使能位。 默认设置为 MCU 时钟源
0： 禁止内部快频 RC 震荡电路。
1： 使能内部快频 RC 震荡电路。如果软件设置这个位，必须等待 **32 us** IHRCO 才能稳定输出。

Bit 3~2： 保留位。写 CKCON0 时，这 2 个位必须写“0”。

Bit 1~0：OSCS[1:0]，OSCin 时钟选择。OSCin 缺损选项是 IHRCO.

OSCS[1:0]	OSCin 时钟选择
0 0	IHRCO
0 1	XTAL
1 0	ILRCO
1 1	ECKI, 外部时钟输入在 (P4.0) OSCin.

AUXR0: 辅助寄存器 0

SFR 地址 = 0xA1　　复位值 = 000x-0000

7	6	5	4	3	2	1	0
P40OC1	P40OC0	P40FD	--	P1FS1	P1FS0	INT1H	INT0H
R/W	**R/W**	**R/W**	**W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6: P40 输出配置控制位 1 和位 0，这两位仅仅当内部 RC 震荡（IHRCO 或 ILRCO）被选择为系统时钟源时有效。这种情况， XTAL2 和 XTAL1 改变功能作 P4.0 和 P4.1。在外部时钟输入模式，P4.0 专用于时钟输入脚，在内部震荡模式下，P4.0 为普通 I/O 或时钟源发生器提供下列选项， 当 P40OC[1:0] 索引为非 P4.0 GPIO 功能时，P4.0 将驱动内部振荡器（IHRCO 或 ILRCO） 输出为其它设备提供时钟源。

P40OC[1:0]	P4.0 功能	P4.0 I/O 模式
00	P40	By P4M0.0
01	OSCin	By P4M0.0
10	OSCin/2	By P4M0.0
11	OSCin/4	By P4M0.0

当 P4.0 作为时钟输出功能，设置 P4M0.0 为 “1” 选者 P4.0 为推挽输出

Bit 5: P40FD, P4.0 快速驱动标志。
0: P4.0 默认驱动输出
1: P4.0 快速驱动输出使能。若 P4.0 被配置为时钟输出， 当 P4.0 输出频率大于 12MHz （5V） 或者大于 6MHz （3V） 时使能此位。

DCON0: 设备控制 0

SFR 页　　= 仅 P 页

SFR 地址 = 0x4C　　RESET = **1000-0010**

7	6	5	4	3	2	1	0
HSE	IAPO	0	0	0	0	RSTIO	0
W	**W**	**W**	**W**	**W**	**W**	**W**	**W**

Bit 7: HSE, 使能高速工作
0: 选择 MCU 在低速模式下运行，降低内部电路的速度来减小功耗。
1: 如果 F_{SYSCLK} > 6MHz.，置位使能 MCU 全速运行

8.3. 系统时钟示例代码

(1)规定功能: IHRCO 从 24MHz 更改到 22.118MHz

汇编语言代码范例:
ORL CKCON0,#(AFS) ; 选择 IHRCO 输出 22.118MHz，AFS=0x80
C 语言代码范例:
CKCON0 |= AFS; //选择 IHRCO 输出 22.118MHz，AFS=0x80

(2). 规定功能: 系统时钟(SYSCLK)更改为OSCin/1 (默认为OSCin/2)

汇编语言代码范例:
ANL CKCON0,#(AFS) ; 设置 SCKS[2:0] = 0 来选择系统时钟(SYSCLK)为 OSCin/1
C 语言代码范例:

CKCON0 &= ~(SCKS2 | SCKS1 | SCKS0); //系统时钟(SYSCLK)为 OSCin/1
// SCKS[2:0], 系统时钟(SYSCLK)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

(3). 规定功能: 当 MCU 使用 IHRCO 或 ILRCO 作为时钟源时，选择外部晶振(XTAL)作为时钟源(OSCin) (默认为 IHRCO)

汇编语言代码范例:

```
    MOV     IFADRL,#(CKCON2)          ; 索引 P 页地址为 CKCON2
    CALL    _page_p_sfr_read          ; 读取 CKCON2 的数据

    ORL     IFD,#(XTALE)              ; 使能外部晶振(XTALE)
                                      ;
    CALL    _page_p_sfr_write         ; 写数据到 CKCON2,系统时钟(SYSCLK )必须小于 25MHz

check_XTOR:                           ; 检测外部晶振(XTAL)振荡准备好
    MOV     A,AUXR1
    JNB     ACC.4,check_XTOR          ; 等待 XTOR(AUXR1.4)为 1

    ANL     IFD,#~(OSCS1 | OSCS0)     ; OSCin 时钟源更改为外部晶振(XTAL)
    ORL     IFD,#(OSCS0)
    CALL    _page_p_sfr_write         ; 写数据到 CKCON2

    ANL     IFD,#~(IHRCOE)            ; 如果 MCU 从 IHRCO 更改之后禁止 IHRCO
    CALL    _page_p_sfr_write         ; 写数据到 CKCON2
```

C 语言代码范例:

```
    IFADRL = CKCON2;                  //索引 P 页地址为 CKCON2
    page_p_sfr_read();                //读取 CKCON2 的数据

    IFD |=XTALE;                      //使能外部晶振(XTALE)
    page_p_sfr_write ();              //写数据到 CKCON2,系统时钟(SYSCLK )必须小于 25MHz

    while(AUXR1 & XTOR == 0x00);      //检测外部晶振(XTAL)振荡准备好
                                      //等待 XTOR(AUXR1.4)为 1

    IFD &= ~(OSCS1 | OSCS0);          // OSCin 时钟源更改为外部晶振(XTAL)
    IFD |= OSCS0;
    page_p_sfr_write ();              //写数据到 CKCON2

    IFD &= ~IHRCOE;                   //如果 MCU 从 IHRCO 更改之后禁止 IHRCO
    page_p_sfr_write();               //写数据到 CKCON2
```

(4). 规定功能: 当 MCU 使用 IHRCO, ECKI 或 XTAL 作为时钟源时，选择 ILRCO 作为时钟源(OSCin) (默认为 IHRCO)

汇编语言代码范例:

```
MOV     IFADRL,#(CKCON2)          ; 索引 P 页地址为 CKCON2
CALL    _page_p_sfr_read          ; 读取 CKCON2 的数据

ANL     IFD,#~(OSCS1 | OSCS0)     ; OSCin 时钟源更改为 ILRCO
ORL     IFD,#(OSCS1)
CALL    _page_p_sfr_write         ; 写数据到 CKCON2

ANL     IFD,#~(XTALE | IHRCOE)    ; 禁止 XTAL 和 IHRCO
CALL    _page_p_sfr_write         ; 写数据到 CKCON2

MOV     IFADRL,#(DCON0)           ; 索引 P 页地址为 DCON0
CALL    _page_p_sfr_read          ; 读取 DCON0 的数据

ANL     IFD,#~(HSE)               ; 当系统时钟(SYSCLK ≤ 6MHz)时为了省电禁止 HSE
CALL    _page_p_sfr_write         ; 写数据到 DCON0
```

C 语言代码范例:

```
IFADRL = CKCON2;                  //索引 P 页地址为 CKCON2
page_p_sfr_read();                //读取 CKCON2 的数据

IFD = ~(OSCS1 | OSCS0);           // OSCin 时钟源更改为 ILRCO
IFD |= OSCS1;
page_p_sfr_write();               //写数据到 CKCON2

IFD &= ~(XTALE | IHRCOE);         //禁止 XTAL 和 IHRCO
page_p_sfr_write();               //写数据到 CKCON2

IFADRL = DCON0;                   //索引 P 页地址为 DCON0
page_p_sfr_read();                //读取 DCON0 的数据

IFD &= ~HSE;                      //当系统时钟(SYSCLK ≤ 6MHz)时为了省电禁止 HSE
```

```
page_p_sfr_write();                    //写数据到 DCON0
```

(5). 规定功能: 当 MCU 使用 IHRCO 或 ILRCO 作为时钟源时，选择 ECKI 作为时钟源(OSCin) (默认为 IHRCO)

汇编语言代码范例:

```
MOV     IFADRL,#(CKCON2)          ; 索引 P 页地址为 CKCON2
CALL    _page_p_sfr_read          ; 读取 CKCON2 的数据

ORL     IFD,#(OSCS1 | OSCS0)      ; OSCin 时钟源更改为 ECKI
CALL    _page_p_sfr_write         ; 写数据到 CKCON2,系统时钟(SYSCLK )必须小于 25MHz

ANL     IFD,#~(XTALE | IHRCOE)    ; 禁止 IHRCO 和 XTAL
CALL    _page_p_sfr_write         ; 写数据到 CKCON2
```

C 语言代码范例:

```
IFADRL = CKCON2;                  // 索引 P 页地址为 CKCON2
page_p_sfr_read();                //读取 CKCON2 的数据

IFD |= OSCS1 | OSCS0;             // OSCin 时钟源更改为 ECKI
page_p_sfr_write ();              //写数据到 CKCON2,系统时钟(SYSCLK )必须小于 25MHz

IFD &= ~(XTALE | IHRCOE);         //禁止 IHRCO 和 XTAL
page_p_sfr_write ();              //写数据到 CKCON2
```

(6). 规定功能: 当 MCU 使用 ILRCO, ECKI 或 XTAL 作为时钟源时，选择 IHRCO 作为时钟源(OSCin)

汇编语言代码范例:

```
MOV     IFADRL,#(CKCON2)          ; 索引 P 页地址为 CKCON2
CALL    _page_p_sfr_read          ; 读取 CKCON2 的数据

ORL     IFD,#(IHRCOE)             ; 使能 IHRCO
CALL    _page_p_sfr_write         ; 写数据到 CKCON2

Delay_32us
```

```
ANL     IFD,#~(OSCS1 | OSCS0)        ; OSCin 时钟源更改为 IHRCO
CALL    _page_p_sfr_write            ; 写数据到 CKCON2

ANL     IFD,#~(XTALE)                ; 禁止 XTAL
CALL    _page_p_sfr_write            ; 写数据到 CKCON2
```

C 语言代码范例:

```
IFADRL = CKCON2;                     //索引 P 页地址为 CKCON2
page_p_sfr_read();                   //读取 CKCON2 的数据

IFD |= IHRCOE;                       // 使能 IHRCO
page_p_sfr_write();                  //写数据到 CKCON2

Delay 32us

IFD &= ~(OSCS1 | OSCS0);             // OSCin 时钟源更改为 IHRCO
page_p_sfr_write();                  //写数据到 CKCON2

IFD &= ~ XTALE;                      // 禁止 XTAL
page_p_sfr_write();                  //写数据到 CKCON2
```

(7). 规定功能: IHRCO 频率输出在 P4.0

汇编语言代码范例:

```
MOV     P4M0,#P4M00                  ; 设置 P4.0 为推挽输出模式
ANL     AUXR0,#~(P40OC1|P40OC0)      ; P4.0 更改为通用输入输出口(GPIO)功能
ORL     AUXR0,#(P40OC0|P4FD)         ; P4.0 = IHRCO 频率 + 引脚快速驱动
                                     ; P40OC[1:0]  |  P4.0
                                     ; 00          |  GPIO
                                     ; 01          |  IHRCO/1
                                     ; 10          |  IHRCO/2
                                     ; 11          |  IHRCO/4
```

C 语言代码范例:

```
    P4M0 |= P4M00;                          //设置 P4.0 为推挽输出模式
    AUXR0 &= ~(P40OC0 | P40OC1);             // P4.0 更改为通用输入输出口(GPIO)功能
    AUXR0 |= (P40OC0 | P4FD);                // P4.0 输出 IHRCO/1
//    AUXR0 = P40OC1|P4FD;                   // P4.0 输出 IHROC/2
//    AUXR0 = P40OC1|P40OC0|P4FD;            // P4.0 输出 IHRCO/4
```

9. 看门狗定时器 (WDT)

9.1. WDT 结构

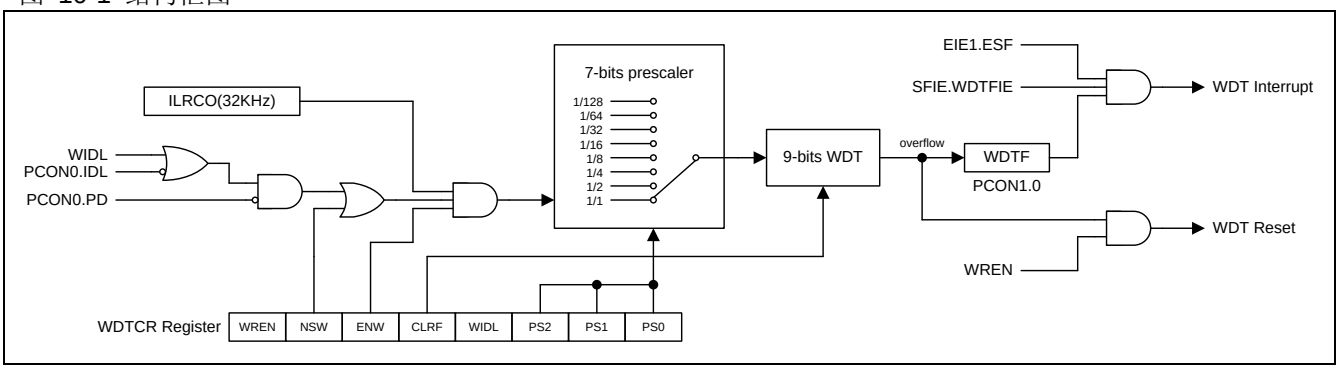
看门狗定时器 (WDT) 用来使程序从跑飞或死机状态恢复的一个手段。WDT 由一个 9 位独立定时器、一个 7 分频器和一个控制寄存器(WDTCR)组成。 图 10-1 显示 **MA86E/L104** WDT 结构框图。

当 WDT 使能，时钟源来自 32KHz ILRCO。WDT 溢出会设置位 WDTF PCON1.0， 也能产生中断通过使能位 WDTFIE (SFIE.0) 和 ESF (EIE1.3). 溢出也能触发系统复位通过设置位 WREN (WDTCR.7) 。软件可以在溢出之前在 CLRW 位 (WDTCR.4)上写“1” 来清除它，可以阻止 WDT 溢出。

一旦 WDT 使能通过设置位 ENW， 将没有办法使之失效除非上电复位或在 page-p SFR 覆盖 ENW，能清除位 ENW。 WDTCR 会保持以前的值不会改变在硬件(RST-pin)复位、软件复位和 WDT 复位后 。

WREN, NSW 和 ENW 都是一次性使能生效，写“1”使能。在 Page-P 中写“0”到位 WDTCR.7~5 禁止 WREN, NSW 和 ENW 使用。祥见 WDT 章节和 P 页访问章节。

图 10-1 结构框图



9.2. WDT 在掉电模式和空闲模式期间

空闲模式， 位标志 WIDL (WDTCR.3) 决定 WDT 是否计数。 设置这个位能让 WDT 在空闲模式一直计数。如果硬件选项 WDTRCO 使能， WDT 会一直保持计数不管位 WIDL 设置情况 。

掉电模式， ILRCO 不会停如果 NSW (WDTCR.6) 使能。 这会让 WDT 保持计数即使掉电模式下(Watch 模式)。WDT 溢出后，软件能设置进入中断或复位 唤醒 CPU。

9.3. WDT 寄存器

WDTCR: **控制寄存器**

SFR 地址 = 0xE1　　POR = 0000-0111 (xxx0_xxxx 硬件选项)

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

Bit 7: WREN, WDT 复位使能标志， 初始值随硬件选项 WRENO。
0: WDT 溢出不产生复位 。 WDT 溢出标志 WDTF 可以供软件检测或触发中断。
1: WDT 溢出产生系统复位。 一旦 WREN 已经设置， 不能用软件在常规页中清除，**但在 page P 中，软件能修改其值 “0” 或 “1”。**

Bit 6: NSW. 不停止的 WDT 标志。 初始值随硬件选项 NSWDT.
0: WDT 停止计数 MCU 在掉电模式。
1: WDT 永远不会停止计数 MCU 在掉电模式或空闲模式。 一旦 NSW 已经设置， 不能用软件在常规页中清除，**但在 page P 中，软件能修改其值 “0” 或 “1”。**

Bit 5: ENW. 使能 WDT 标志。
0: 禁止 WDT 运行， 这个位仅仅能被 POR 清除。
1: 使能 WDT，一旦 ENW 已经被设置，不能用软件在常规页中清除，**但在 page P 中，软件能修改其值 “0” 或 “1”。**

Bit 4: CLRW. WDT 清零位。
0: 在 WDT 里写 “0” 到这位无效
1: 写 “1” 到到这位将清零 9 位 WDT 计数器到 000H。注意这位没有必要写 “0” 去清除它。置位它将清除 WDT 重新计数。

Bit 3: WIDL. WDT 空闲模式控制。
0: WDT 停止计数 MCU 在空闲模式。
1: WDT 保持计数 MCU 在空闲模式。

Bit 2~0: PS2 ~ PS0，选择分频器输出作 WDT 基础时钟输入（分频系数设置）

PS[2:0]	分频值	WDT 时间
0 0 0	1	15 ms
0 0 1	2	31 ms
0 1 0	4	62 ms
0 1 1	8	124 ms
1 0 0	16	248 ms
1 0 1	32	496 ms
1 1 0	64	992 ms
1 1 1	128	1.984 S

PCON1:电源控制寄存器 1

SFR 地址 = 0x97　　上电复位值 = **00xx-0x00**

7	6	5	4	3	2	1	0
SWRF	EXRF	--	--	KBIF	--	BOF0	WDTF
R/W	**R/W**	**W**	**W**	**R/W**	**W**	**R/W**	**R/W**

Bit 0: WDTF, WDT 溢出标志。
0: 必须由软件写**“1”** 清除， 软件写“:0” 不操作。

1: 当 WDT 溢出时硬件置位此位，写 **“1”** 清除。

9.4. WDT 硬件选项

除了由软件初始化外，WDTCR 寄存器还能在上电的时候由硬件选项 WRENO,NSWDT,HWENW,HWWIDL 和 HWPS[2:0]来自动初始化，这些选项通过通用编程器来编程，如下所叙。

如果 HWENW 编程为“使能”，则硬件在上电时为 WDTCR 寄存器作如下的初始化工作：（1）ENWI 位置 1。（2）载入 WRENO 的值到 WREN 位。(3)载入 NSWDT 的值到 NSW 位。(4)载入 HWWIDL 的值到 WIDL 位。（5）载入 HWPS【2：0】的值到 PS【2：0】位。

如果 HWENW 和 WDSFWP 都被编程为“使能”，则硬件仍然会在上电时由 WDT 硬件选项初始化 WDTCR 寄存器的内容。之后，任何对 WDTCR 的位的写动作都会被忽略，除了写“1”到 WDTCR.4(CLRW)位来清 WDT 之外，即使通过对 Page-P SFR 的操作机制也不行。I

WRENO:
使能: 置位 WDTCR.WREN 以使能 WDTF 系统复位功能。
禁止: 清除 WDTCR.WREN 以禁止 WDTF 系统复位功能。

NSWDT:
使能：使能 WDT 在掉电模式也保持运行，设置位 WDTCR.NSW (watch 模式)。
禁止: 禁止 WDT 在掉电模式下运行，清除位 WDTCR.NSW 。

HWWIDL, HWPS2, HWPS1, HWPS0:
当 HWENW 被使能，上电复位时，这四个保险丝位将被载入到特殊功能寄存器 WDTCR 中。

HWENW:
使能： 上电时自动硬件使能看门狗定时器，并且自动加载 WRENO, NSWDT, HWWIDL 和 HWPS2~0 的值到 WDTCR 中。
禁止： 上电时看门狗定时器（WDT）不自动使能。

WDSFWP: 禁止软件写 WDTCR 寄存器
使能： 特殊功能寄存器 WDTCR 软件写保护。
禁止： 特殊功能寄存器 WDTCR 可被软件改写。

9.5. WDT 示例代码

(1) 规定功能: 使能 WDT 并且选择 WDT 周期为 248 毫秒(ms)

汇编语言代码范例:

```
ANL     PCON1,#(WDTF)                   ; 清除 WDTF 标志(写“1”)
MOV     WDTCR,#(ENW | CLRW | PS2)       ; 使能 WDT 计数器并且设置 WDT 周期为 248 毫秒(ms)
```

C 语言代码范例:

```
PCON1 &= WDTF;                          //清除 WDTF 标志(写“1”)
WDTCR = (ENW | CLRW | PS2);             //使能 WDT 计数器并且设置 WDT 周期为 248 毫秒(ms)
                                        // PS[2:0]   | WDT  周期选择
                                        //    0      | 15ms
                                        //    1      | 31ms
                                        //    2      | 62ms
                                        //    3      | 124ms
                                        //    4      | 248ms
                                        //    5      | 496ms
                                        //    6      | 992ms
                                        //    7      | 1.984s
```

(2) 规定功能: 如何禁止 WDT

汇编语言代码范例:

```
MOV     IFD,WDTCR                       ; 读取 WDTCR 数据
ANL     IFD,#~(ENW)                     ; 清除 ENW 而禁止 WDT

MOV     IFADRL,#(WDTCR_P)               ; 索引 P 页地址为 WDTCR_P
CALL    _page_p_sfr_write               ; 写数据到 WDTCR
```

C 语言代码范例:

```
IFD = WDTCR;                            //读取 WDTCR 数据
```

```
IFD &= ~ENW;                        //清除 ENW 而禁止 WDT

IFADRL = WDTCR_P;                   //索引 P 页地址为 WDTCR_P
page_p_sfr_write();                 //写数据到 WDTCR
```

(3). 规定功能: 使能 WDT 复位功能并且选择 WDT 周期为 62 毫秒(ms)

汇编语言代码范例:

```
ANL     PCON1,#(WDTF)               ; 清除 WDTF 标志(写“1”)
MOV     WDTCR,#(WREN | CLRW | PS1)  ; 使能 WDT 复位功能并且设置 WDT 周期为 62 毫秒(ms)

ORL     WDTCR,#(ENW)                ; 使能 WDT 计数器, WDT 运行
```

C 语言代码范例:

```
PCON1 &= WDTF;                      //清除 WDTF 标志(写“1”)
WDTCR = WREN | CLRW | PS1;          //使能 WDT 复位功能并且设置 WDT 周期为 62 毫秒(ms)

WDTCR |= ENW;                       //使能 WDT 计数器, WDT 运行
```

(4). 规定功能:使能 WDTCR 的写保护

汇编语言代码范例:

```
ANL     PCON1,#(WDTF)               ; 清除 WDTF 标志(写“1”)
MOV     WDTCR,#(ENW | CLRW | PS2)   ; 使能 WDT 计数器并且设置 WDT 周期为 248 毫秒(ms)

MOV     IFADRL,#(SPCON0)            ; 索引 P 页地址为 SPCON0
CALL    _page_p_sfr_read            ; 读取 SPCON0 数据

ORL     IFD,#(WRCTL)                ; 使能 WDTCR 的写保护
CALL    _page_p_sfr_write           ; 写数据到 SPCON0

MOV     IFD,WDTCR                   ; 读取 WDTCR 数据
ORL     IFD,#(CLRW)                 ; 使能 CLRW
```

```
MOV       IFADRL,#(WDTCR_P)          ; 索引 P 页地址为 WDTCR_P
CALL      _page_p_sfr_write          ; 写数据到 WDTCR  而清零 WDT 计数器
```

C 语言代码范例:

```
PCON1 &= WDTF;                       //清除 WDTF 标志(写“1”)
WDTCR = ENW | CLRW | PS2;            //使能 WDT 计数器并且设置 WDT 周期为 248 毫秒(ms)

IFADRL = SPCON0;                     //索引 P 页地址为 SPCON0
page_p_sfr_read();                   //读取 SPCON0 数据

IFD |= WRCTL;                        //使能 WDTCR 的写保护
page_p_sfr_write();                  //  写数据到 SPCON0

IFD = WDTCR;                         //读取 WDTCR 数据
IFD |= CLRW;                         //  使能 CLRW

IFADRL = WDTCR_P;                    //索引 P 页地址为 WDTCR_P
page_p_sfr_write();                  //写数据到 WDTCR  而清零 WDT 计数器
```

10. 系统复位

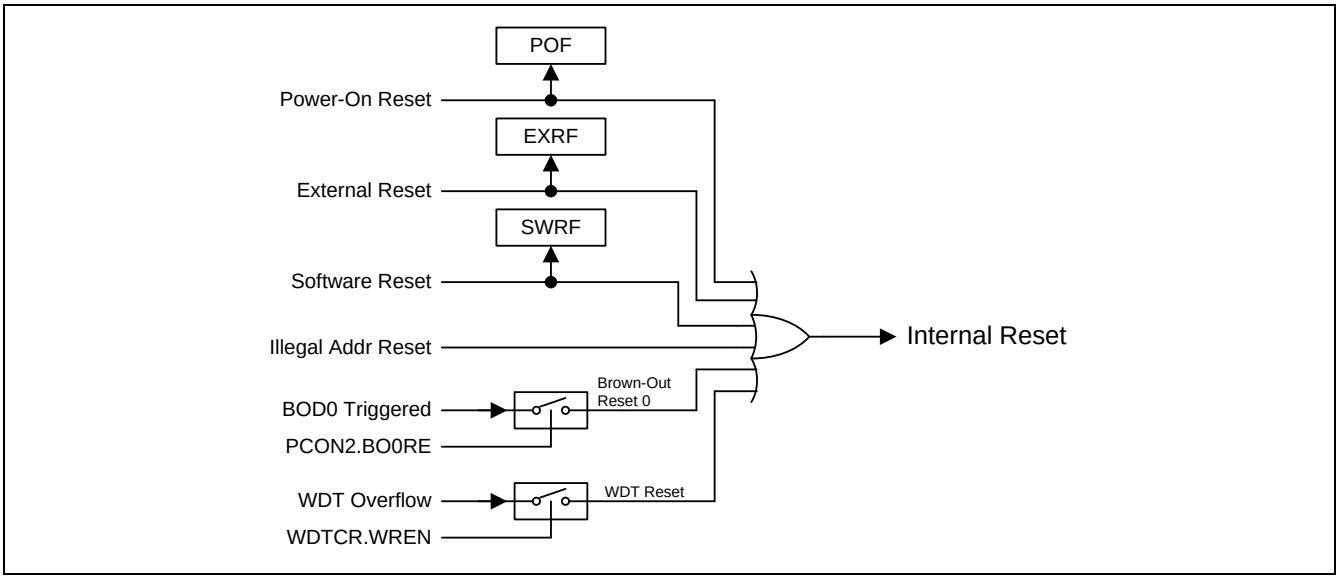
复位期间， 所有的 I/O 寄存器都设置为初始值，程序会根据 OR 设置选择从复位向量的 0000H 开始运行，或者根据 OR 设置从 ISP 地址开始运行。The **MA86E/L104** 有 6 种复位源 ： 上电复位， 外部复位， 软件复位， 非法地址复位， WDT 复位和 低电压复位。 如图所示：图 11-1 系统复位源（ **MA86E/L104**）。

下面的选项描叙复位产生源及其相应的控制寄存器和指示标志。

10.1. 复位源

图 11-1 展示了 **MA86E/L104** 的复位系统，和所有复位源

图 11-1 系统复位源



10.2. 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。

PCON0: 电源控制寄存器 0

SFR 地址 = 0x87　　POR = 0001-0000, 复位值 = 000X-0000

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

Bit 4: POF. Power-On 旗标.
0: 这标志必须通过软件清零以便认出下一个复位类型。
1: 当VDD从0 伏上升到正常电压时硬件复位， POF 也能有软件置位。

上电标志 POF 在上电过程中由硬件置“1” 或当 VDD 电压降到 V_{POR} 电压之下时由硬件置“1”。它能通过软件来清除但不受任何热复位（譬如：外部 RST 引脚复位、掉电检测器 Brown-Out 复位、软件(ISPCR.5)复位和 WDT 复位）的影响。它帮助用户检测 CPU 是否从上电开始运行。注意：POF 必须由软件清除。

PCON1: 电源控制寄存器 1

SFR 地址 = 0x97 POR = **00xx-0x00**

7	6	5	4	3	2	1	0
SWRF	EXRF	**POF1**	IARF	KBIF	--	BOF0	WDTF
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**W**	**R/W**	**R/W**

Bit 4: POF1，上电标志 1。是 POF 标志复制品， 所以，这个标志 POF1 与 POF0 一样反应相同的电源状况。
0: 这位必须通过软件清零，写**“1”** 清零，写 “:0” 无效。
1: 当VDD 由0V上升到普通电压时，硬件置位此位，写**“1”** 清零。

10.3. 外部复位

保持复位引脚 RST 至少 24 个振荡周期的高电平，将产生一个复位信号，为确保 MCU 正常工作，必须在 RET 引脚上连接可靠的硬件复位电路。

PCON1: 电源控制寄存器 1

SFR Address = 0x97 POR = 0010-0x00

7	6	5	4	3	2	1	0
SWRF	**EXRF**	POF1	IARF	KBIF	--	BOF0	WDTF
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**W**	**R/W**	**R/W**

Bit 6: EXRF，外部复位标志。
0: 这位必须通过软件清零，写**“1”** 清零，写 “:0” 无效。
1: 若外部复位产生则被硬件置位，写**“1”** 清零。

10.4. 软件复位

软件通过对 SWRST(ISPCR.5) 位写 “1” 触发一个系统热复位， 软件复位后，硬件置位 SWRF 标志(PCON1.7)。SWBS 标志决定 CPU 是从 ISP 还是 AP 区域开始运行程序。

ISPCR: ISP 控制寄存器

SFR Address = 0xE5 RESET = 0000-xxxx

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

Bit 6: SWBS, 软件引导控制
0: 用来选择软件复位后从 AP-空间。
1: 用来选择软件复位后从 ISP-空间。

Bit 5: SWRST, 软件复位触发控制
0: 无操作
1: 产生软件系统复位，它将被硬件自动清除。

PCON1: 电源控制寄存器 1

SFR 地址 = 0x97 POR = 0010-0x00

7	6	5	4	3	2	1	0

SWRF	EXRF	POF1	IARF	KBIF	--	BOF0	WDTF
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**W**	**R/W**	**R/W**

Bit 7: SWRF, 软件复位标志。
0: 这位必须通过软件清零 ，写“1” 清零，写 “0”无操作。
1: 软件复位产生时硬件置位此位，写“1” 清零。

10.5. 掉电检测器（Brown-Out）复位

MA86E/L104 中，掉电检测器(BOD0)检测电源电压（VDD），掉电检测器(BOD0) 的检测固定点为 VDD=4.0V(E 系列) 2.6V(L 系列)， 如果 VDD 电压低于 BOD0 检测点， 则置位 BOF0 标志，如果 BO0RE (PCON2.1) 被使能，BOD0 事件将触发一个 CPU 复位并置位 BOF0 指示一个掉点检测器（BOD0） 复位发生。

PCON1: 电源控制寄存器 1

SFR 地址 = 0x97 POR =**00xx-0x00**

7	6	5	4	3	2	1	0
SWRF	EXRF	--	--	KBIF	--	**BOF0**	WDTF
R/W	**R/W**	**W**	**W**	**R/W**	**W**	**R/W**	**R/W**

Bit 1: BOF0, BOF0 复位标志
0: 这位必须通过软件清零 ，写“1” 清零，写 “0”无操作。
1: 当 VDD 电压碰到 BOD0 检测点时，硬件置位此位，写“1” 清零。如果 BO0RE (PCON2.1) 被使能，BOD0 事件将触发一个 CPU 复位并置位 BOF0 指示一个掉点检测器（BOD0）复位发生。

10.6. WDT 复位

当 WDT 使能开始计数， WDT 溢出时置位 WDTF 标志 。 如果 WREN (WDTCR.7) 使能， WDT 溢出将引起一个系统热复位， 软件可以读 WDTF 标志来确认 WDT 复位发生。

PCON1: 电源控制寄存器 1

SFR Address = 0x97 POR = **00xx-0x00**

7	6	5	4	3	2	1	0
SWRF	EXRF	--	--	KBIF	--	BOF0	**WDTF**
R/W	**R/W**	**W**	**W**	**R/W**	**W**	**R/W**	**R/W**

Bit 0: WDTF, WDT 溢出/复位 标志。
0: 这位必须通过软件清零 ，写“1” 清零，写 “0”无操作。
1: 当 WDT 溢出产生时硬件置位此位，写“1” 清零。如果 位 WREN (WDTCR.7) 被设置， WDTF 标志指示一个 WDT 复位产生。

10.7. 非法地址复位

MA86E/L104 中，如果程序运行到非法地址诸如超过 ROM 限制程序地址时触发一个 CPU 热复位并置位 IARF（PCON1.4）标志，以指示一个非法地址复位发生。

10.8. 复位示例代码

(1) 规定功能: 触发一个软件复位

汇编语言代码范例:

```
ORL     ISPCR,#SWRST                ; 触发一个软件复位
```

C 语言代码范例:

```
ISPCR |= SWRST;                     //触发一个软件复位
```

(2). 规定功能: 使能 BOD0 复位

汇编语言代码范例:

```
MOV     IFADRL,#PCON2               ; 索引 P 页地址为 PCON2
CALL    _page_p_sfr_read            ; 读取 PCON2 数据

ORL     IFD,#BO0RE                  ; 使能 BOD0 复位功能
CALL    _page_p_sfr_write           ; 写数据到 PCON2
```

C 语言代码范例:

```
IFADRL = PCON2;                     // 索引 P 页地址为 PCON2
page_p_sfr_read();                  // 读取 PCON2 数据

IFD |= BO0RE;                       //使能 BOD0 复位功能
page_p_sfr_write();                 // 写数据到 PCON2
```

11. 电源管理

MA86E/L104 支持一个电源监测模块（掉电侦察器(BOD0)模块），和 6 种电源节能模式：空闲模式（IDLE）、掉电模式（Power-Down）、慢频模式、 副频模式、Watch 模式 、 Monitor 模式。

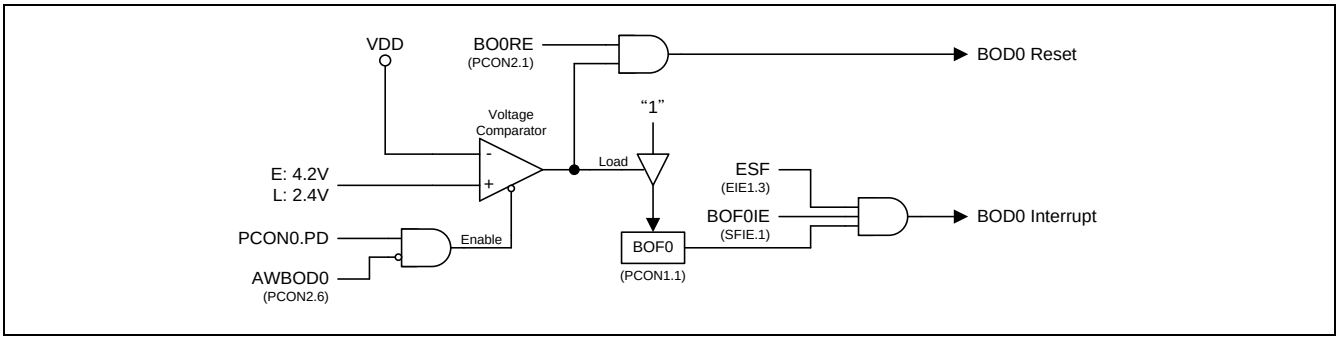
通过 BOF0 标志位 BOD0 报告电源状态 ， 软件可以通过这个状态产生中断或复位。6 种电源节能模式提供不同的节能应用， 通过对 CKCON0, CKCON2, PCON0, PCON1, PCON2 和 WDTCR 寄存器的访问来操作这些电源事件。.

11.1. 电源监控模块

I**MA86E/L104**，有一个片上检测器 (BOD0)通过比较固定的触发电压来检测芯片电压， 图 12-1 是 BOD0 功能逻辑图， BOD0 检测固定触发电压为 VDD=4.0V （ **5V** 应用） 和 2.6V 在（**3.3V** 应用）。 当 VDD 降到触发电压以下时， BOF0 (PCON1.1)标志被置位， 如果 ESF (EIE1.3) 和 BOF0IE (SFIE.1) 被使能， 不管是普通模式或空闲模式都能产生一个中断请求以响应(BOD0)事件， 如果 AWBOD0 (PCON2.6)使能，这个中断也能唤醒掉电模式。

当 BO0RE (PCON2.1) 被使能， BOD0 事件产生一个系统复位并硬件置位 BOF0 指示一个 BOD0 复位事件已经产生。在普通模式和空闲模式下 BOD0 事件能重新启动 CPU，如果 AWBOD0 (PCON2.6)位被使能，也能重新启动掉电模式。

图 11-1 电源监控检测器



11.2. 电源节省模式

11.2.1. 慢频模式

程序设置位 SCKS2~SCKS0(CKCON0 寄存器，参考系统时钟选项"8 系统时钟"为非 0/0/0 值，可以减慢 MCU 的工作速度达到节能的目的， 使用者考量在特殊的程序段使用合适的慢速度，原则上不应该影响系统的其他功能。而且，应该在普通的程序段恢复到正常的速度。

11.2.2. 副频模式

设置 OSCS1~0 选择 OSCS1~0 作为系统时钟，MCU 的工作频率会慢下来， 32KHz ILRCO 系统频率使 MCU 工作在特别慢的速度和功耗下， 另外设置 SCKS2~SCKS0 位 (CKCON0 寄存器，参考系统时钟选项""8 系统时钟") 使用者可以使 MCU 的速度最低到 250HZ。

11.2.3. Watch 模式

如果看门狗被使能并且位被设置 ， 看门狗在掉电模式保持运行，这个在 **MA86E/L104 应用**中叫 Watch 模式。 当 When WDT 溢出， 软件选择中断或系统复位来唤醒 CPU 并硬件置位 WDTF 。 通过定义 WDT 预分频最大唤醒时间能到 2 秒， 更详细信息请参考 “9 看门狗定时器 (WDT)”章节和“13 中断”章节。

11.2.4. Monitor 模式 (仅仅使用于 L-系列)

如果 AWBOD1 (PCON3.3) 被设置， 即使在掉电模式下，掉电检测功能 BOD1 会有效，这就是 **MA86E/L104 应**用中的 Monitor 模式。 当 BOD1 触发到检测电压， 软件选择中断或系统复位来唤醒 CPU 并硬件置位 BOF1 ，更详细信息请参考“11.1 电源监控模块”和“13 中断” ， 这功能仅仅实用于 L-系列。

11.2.5. 空闲模式

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 端口也保持当前的逻辑状态。空闲模式保持外部设置当有中断来时能唤醒 CPU， 空闲模式下定时器 0、定时器 1、UART、KBI、BOD0 仍然处于工作状态。在空闲模式下 WDT 唤醒 CPU 有条件制约。 任何使能的中断源或复位都能终止空闲模式， 一个中断会退出空闲模式， 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。

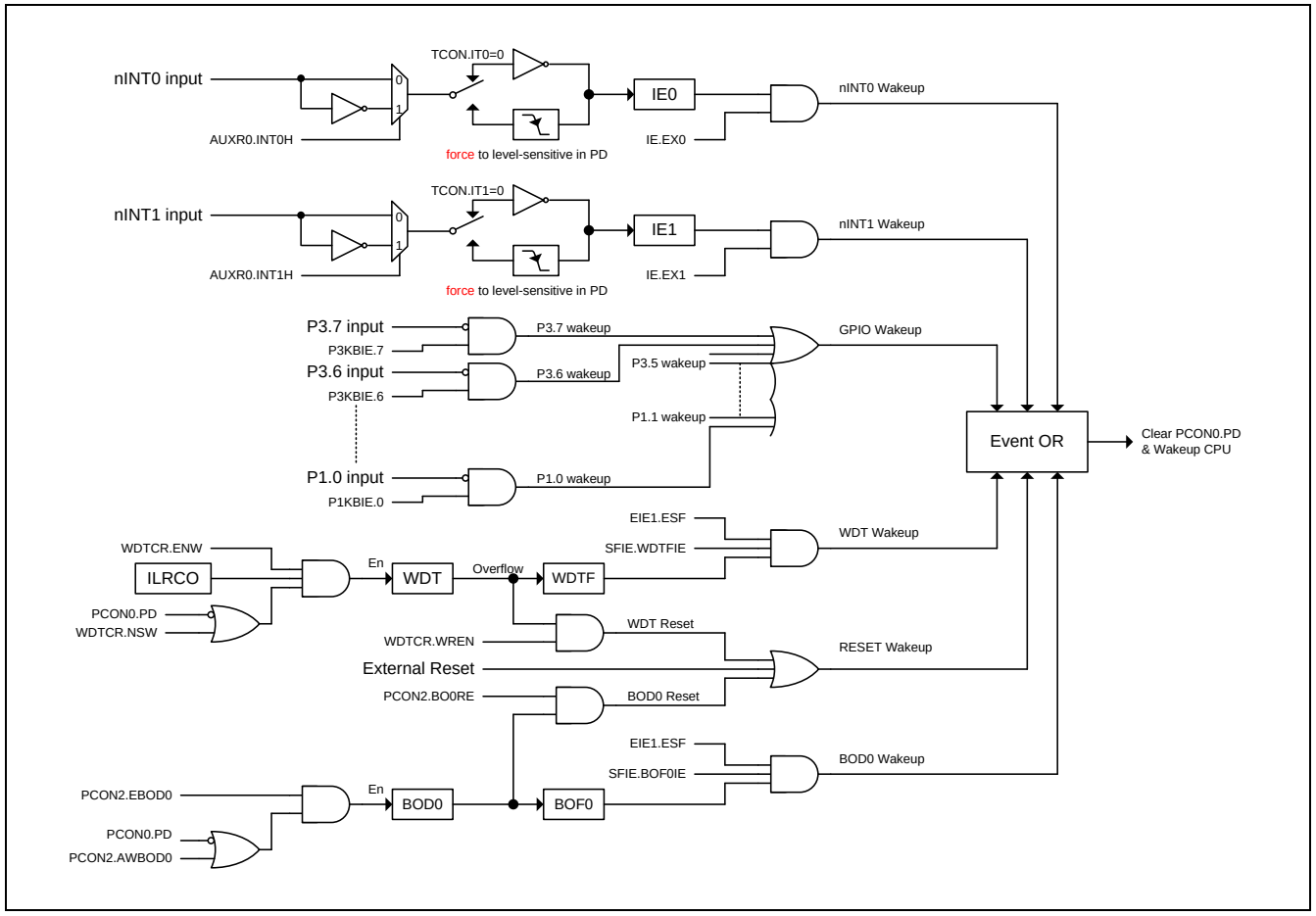
11.2.6. 掉电模式

可以通过软件的方法置位 PCON. PD 使设备进入掉电模式， 掉电模式下，震荡器停止震荡， 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；但如果电源电压低于芯片工作电压，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、使能的 KBI 使能的 BOD0 或 使能的没有停止的 WDT 能是系统推出掉电模式。

如果有下列情况发生，使用者至少要等 4 微秒后才能进入或再次进入掉电模式： 刚开始运行代码(任何形式的复位后面)，或者刚刚退出掉电模式。 为了在掉电模式达到最小功耗， 软件必须设置所有的 I/O 为悬浮状态， 包含封装中没有漏出来的 I/O。例如：**P1.7~P1.0 和 P4.1~P4.0 在 MA86E/L104AE8 (SOP8) 的封装中都没有秀出来，软件必须设置 P1/P4 SFR 控制位为 “0” (输出低) 来避免脚位在掉电模式中处于悬浮状态。**

图 11-2 掉电唤醒结构标示了 **MA86E/L104** 在掉电模式中唤醒的进程。

图 11-2 掉电唤醒结构



11.2.7. 中断唤醒掉电模式

两个外部中断都能终止掉电模式， 外部中断 nINT0 (P3.2), nINT1 (P3.3) 能退出掉电模式， 为了能唤醒掉电模式，中断 nINT0, nINT1 必须使能并且设置为电平触发操作，如果外部中断使能且设置是边沿触发（上升或下降），他们会被硬件强置为电平触发（低电平或高电平）。

一个中断终止掉电模式，唤醒时间取决内部定时。 当中断口产生下降沿时 ， 掉电模式被终止，震荡重新启动， 并且一个内部计数器开始计数， 在内部计数器没有计满之前内部时钟不允许被应用 CPU 也不能运行指令。 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待系统问题。

11.2.8. 复位唤醒掉电模式

如果 P3.6 设置为 RST 脚 ，RST 脚唤醒有点类似于中断， 复位脚有上升沿电平时系统退出掉电模式， 震荡重新启动， 且一个内部计数器开始计数， 在内部计数器没有计满之前内部时钟不允许被应用 CPU 也不能运行指令。复位脚必须保持长时间的高电平以保证系统完全复位，复位脚变低电平时开始执行程序。

值得注意的是当空闲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）。

11.2.9. KBI 键盘唤醒掉电模式

MA86E/L104 的键盘中断，P1.7 ~ P1.0, P3.7~P3.4, **P41, P40,** P31 和 P30 具有唤醒能力，可以通过 KBI 模块的控制寄存器 P1KBIE 、 P3KBIE ， P3.2/nINT0 、 P3.3/nINT1 进行使能。

通过使能 KBI 唤醒掉电模式有点类似中断唤醒， 当使能 KBI 的 IO 口有低电平时系统退出掉电模式， 震荡重新启动， 且一个内部计数器开始计数， 在内部计数器没有计满之前内部时钟不允许被应用 CPU 也不能运行指令。计数溢出后，CPU 响应 KBI 中断并入中断服务程序， 细节请参考“16 键盘中断(KBI)”。

11.3. 电源控制寄存器

PCON0: 电源控制寄存器 0

SFR 地址 = 0x87　　POR = **00xx-0x00**

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 4: POF，上电标志
0: 这位必须由软件清零，写“1”清零。
1: 当上电复位产生时硬件置位此位。

Bit 1: PD，掉电控制位 。
0: 软件清零或任何一个退出掉电模式的事件发生时硬件清零。
1: 置位则激活掉电操作（即进入掉电模式）。

Bit 0: IDL，空闲模式控制位。
0: 软件清零或任何一个退出空闲模式的事件发生时硬件清零。
1: 置位则激活空闲操作（即进入空闲模式）。

PCON1: 电源控制寄存器 1

SFR 地址 = 0x97　　POR = 0010-0x00

7	6	5	4	3	2	1	0
SWRF	EXRF	--	--	**KBIF**	--	BOF0	WDTF
R/W	**R/W**	**W**	**W**	**R/W**	**W**	**R/W**	**R/W**

Bit 7: SWRF，软件复位标志。
0: 这位必须由软件清零，写“1”清零。
1: 当软件复位产生时硬件置位此位。

Bit 6: EXRF，外部复位标志。
0: 这位必须由软件清零，写“1”清零。
1: 当外部复位产生时硬件置位此位。

Bit 5~4: 保留. 当写 PCON1 时，软件必须在这些位上写“0”

Bit 3: KBIF 键盘中断标志。
0: 这位必须由软件清零，写“1”清零。
1: 当键盘中断复位产生时硬件置位此位。

Bit 2:保留当写 PCON1 寄存器时此位必须填“0”。

Bit 1: BOF0, Brown-Out 侦察标志 0.
0: 这位必须由软件清零，写“1”清零。
1: 当电源监控复位产生时硬件置位此位(E: 4.2V, L: 2.4V)。

Bit 0: WDTF, WDT 溢出标志
0: 这位必须由软件清零，写“1”清零。
1: 当 WDT 复位产生时硬件置位此位。

PCON2: 电源控制寄存器 2

SFR Page = **P Only**

SFR 地址 = 0x44 POR = x0xx-xx01

7	6	5	4	3	2	1	0
--	**AWBOD0**	--	--	--	--	BO0RE	**1**
W	**W**	**W**	**W**	**W**	**W**	**W**	**W**

Bit 7: 保留,写 PCON2 寄存器时此位必须填“0”。

Bit 6: AWBOD0，在掉电模式下（PD）使能电源监控模式（BOD0）控制位。
0: 电源监控模式（BOD0）在掉电模式下失效。
1: 电源监控模式（BOD0）在掉电模式下有效。

Bit 5~2: 保留,写 PCON2 寄存器时此几位必须填“0”。

Bit 1: BO0RE, BOD0 复位使能标志，初始为 OR1.BO0REO 取反值。
0: 当 BOF0 已经设置，禁止电源监控（BOD0）系统复位。
1: 当 BOF0 已经设置，使能电源监控（BOD0）系统复位（VDD 触到 4.0V(E) 或 2.6V(L)）。

Bit 0: 保留给测试用，写 PCON2 寄存器时此位必须填“1”。

P1KBIE: Port 1 KBI 使能控制寄存器

SFR 地址 = 0xD7 RESET = 0000-0000

7	6	5	4	3	2	1	0
P17KBIE	P16KBIE	P15KBIE	P14KBIE	P13KBIE	P12KBIE	P11KBIE	P10KBIE
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~0: 每个 P1 引脚键盘输入功能使能控制位。
0: 相对应的引脚键盘输入功能失效
1: 相对应的引脚键盘输入功能使能，低电压触发。

P3KBIE: Port 3 KBI 使能控制寄存器

SFR 地址 = 0xD6 RESET = 0000-0000

7	6	5	4	3	2	1	0
P37KBIE	P36KBIE	P35KBIE	P34KBIE	**P41KBIE**	**P40KBIE**	P31KBIE	P30KBIE
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~0: P3.7 ~ P3.4, P4.1, P4.0, P3.1 和 P3.0 引脚键盘输入功能使能控制位。
0: 相对应的引脚键盘输入功能禁止
1: 相对应的引脚键盘输入功能使能，低电压触发。

11.4. 电源控制示例代码

(1) 规定功能: 选择系统时钟分频为OSCin/128 的低速模式 (默认为OSCin/2)

汇编语言代码范例:

```
ORL     CKCON0,#(SCKS0 | SCKS1 | SCKS2)    ; 选择系统时钟分频为 OSCin/128

MOV     IFADRL,#DCON0                      ; 索引 P 页地址为 DCON0
CALL    _page_p_sfr_read                   ; 读取 DCON0  数据

ANL     IFD,#~(HSE)                        ; 当系统时钟 SYSCLK ≤ 6MHz  为了省电禁止 HSE
CALL    _page_p_sfr_write                  ; 写数据到 DCON0
```

C 语言代码范例:

```
CKCON0 |= (SCKS2 | SCKS1 | SCKS0);       //选择系统时钟分频为 OSCin/128.

IFADRL = DCON0;                          // 索引 P 页地址为 DCON0
page_p_sfr_read();                       // 读取  DCON0  数据.

IFD &= ~HSE;                             //当系统时钟 SYSCLK ≤ 6MHz  为了省电禁止 HSE
page_p_sfr_write();                      // 写数据到 DCON0
```

(2) 规定功能: 选择系统时钟分频为OSCin/2 的副频模式 (默认为OSCin/2=32KHz/2=16KHz)

汇编语言代码范例:

```
MOV     IFADRL,#CKCON2                   ; 索引 P 页地址为 CKCON2
CALL    _page_p_sfr_read                 ; 读取  CKCON2  数据

ANL     IFD,#~(OSCS1|OSCS0)              ; OSCin 时钟源更改为 ILRCO
ORL     IFD,#OSCS1
CALL    _page_p_sfr_write                ; 写数据到 CKCON2
```

```
ANL     IFD,#~(IHRCOE|XTALE)        ; 禁止 IHRCO 和 XTAL
CALL    _page_p_sfr_write           ; 写数据到 CKCON2

MOV     IFADRL,#DCON0)              ; 索引 P 页地址为 DCON0
CALL    _page_p_sfr_read            ; 读取 DCON0 数据

ANL     IFD,#~(HSE)                 ; 当系统时钟 SYSCLK ≤ 6MHz 为了省电禁止 HSE
CALL    _page_p_sfr_write           ; 写数据到 DCON0

MOV     A,CKCON0                    ; 选择系统时钟为 OSCin/2
ANL     A,#~(SCKS2|SCKS1|SCKS0)
ORL     A,#SCKS0
MOV     CKCON0,A
```

C 语言代码范例:

```
IFADRL = CKCON2;                    // 索引 P 页地址为 CKCON2
page_p_sfr_read();                  // 读取 CKCON2 数据

IFD &= ~(OSCS1 | OSCS0);            // OSCin 时钟源更改为 ILRCO
IFD |= OSCS1;
page_p_sfr_write();                 // 写数据到 CKCON2

IFD = IFD & ~(IHRCOE|XTALE);        //禁止 IHRCO 和 XTAL
page_p_sfr_write();                 // 写数据到 CKCON2

IFADRL = DCON0;                     // 索引 P 页地址为 DCON0
page_p_sfr_read();                  // 读取 DCON0 数据

IFD = IFD & ~(HSE);                 //当系统时钟 SYSCLK ≤ 6MHz 为了省电禁止 HSE
page_p_sfr_write();                 // 写数据到 DCON0

ACC = CKCON0;                       // 选择系统时钟为 OSCin/2
ACC &= ~(SCKS2 | SCKS1 | SCKS0);
ACC |= SCKS0;
CKCON0 = ACC;
```

(3). 规定功能: 现在 MCU 运行在外部振荡(XTAL) 模式

汇编语言代码范例:

```
    MOV     IFADRL,#CKCON2              ; 索引 P 页地址为 CKCON2
    CALL    _page_p_sfr_read            ; 读取 CKCON2 数据

    ORL     IFD,#(XTALE)                ; 使能外部振荡(XTAL)振荡
    CALL    _page_p_sfr_write           ; 写数据到 CKCON2

check_XTOR_0:                           ; 检测外部振荡(XTAL)振荡准备好
    MOV     A,AUXR1
    JNB     ACC.4,check_XTOR_0          ; 等待 XTOR(AUXR1.4)为 1

    ANL     IFD,#~(OSCS1|OSCS0)         ; OSCin 时钟源更改为外部振荡(XTAL)
    ORL     IFD,#OSCS0
    CALL    _page_p_sfr_write           ; 写数据到 CKCON2

    ANL     IFD,#~(IHRCOE)              ; 如果 MCU 从 IHRCO 切换过来则禁止 IHRCO
    CALL    _page_p_sfr_write           ; 写数据到 CKCON2

    MOV     IFADRL,#DCON0               ; 索引 P 页地址为 DCON0
    CALL    _page_p_sfr_read            ; 读取 DCON0 数据

    ANL     IFD,#~(HSE)                 ; 当系统时钟 SYSCLK ≤ 6MHz 为了省电禁止 HSE
    CALL    _page_p_sfr_write           ; 写数据到 DCON0

    ANL     CKCON0,#~(SCKS2|SCKS1|SCKS0) ; 系统时钟 SYSCLK = OSCin/1
```

C 语言代码范例:

```
    IFADRL = CKCON2;                    // 索引 P 页地址为 CKCON2
    page_p_sfr_read();                  // 读取 CKCON2 数据

    IFD |= XTALE;                       //使能外部振荡(XTAL)振荡
    page_p_sfr_write();                 // 写数据到 CKCON2
```

```
    while( AUXR1&XTOR == 0x00 );              //检测外部振荡(XTAL)振荡准备好
                                              //等待 XTOR(AUXR1.4)为 1

    IFD &= ~(OSCS1 | OSCS0);                  // OSCin 时钟源更改为外部振荡(XTAL)
    IFD |= OSCS0;
    page_p_sfr_write ();                      // 写数据到 CKCON2

    IFD &= ~IHRCOE;                           // 如果 MCU 从 IHRCO 切换过来则禁止 IHRCO
    page_p_sfr_write();                       // 写数据到 CKCON2.

    IFADRL = DCON0;                           // 索引 P 页地址为 DCON0
    page_p_sfr_read();                        // 读取 DCON0 数据.

    IFD &= ~HSE;                              //当系统时钟 SYSCLK ≤ 6MHz 为了省电禁止 HSE
    page_p_sfr_write();                       // 写数据到 DCON0

    CKCON0 &= ~(SCKS2 | SCKS1 | SCKS0);   // 系统时钟 SYSCLK = OSCin/1
```

(4). 规定功能: 使能 2 秒(s)周期的 Watch 模式

汇编语言代码范例:

```
    ORG       0003Bh
SystemFlag_ISR:
    ANL       PCON1,#(WDTF)                   ; 清除 WDT 标志(写“1”)
    RETI

main:
    ANL       PCON1,#WDTF                     ; 清除 WDT 标志 (写“1”)
    ORL       WDTCR,#(NSW|ENW|PS2|PS1|PS0)
                                              ;使能 WDT 和 NSW (对 watch 模式)
                                              ;设置 PS[2:0] = 7 来选择 WDT 周期为 1.984 秒(s)

    ORL       SFIE,#WDTFIE                    ; 使能 WDT 中断
    ORL       EIE1,#ESF                       ; 使能系统标志中断
    SETB      EA                              ; 使能全局中断
```

```
    ORL     PCON0,#PD                       ; 设置 MCU 为掉电模式

;   MCU 等待唤醒
```

C 语言代码范例:

```
void SystemFlag_ISR (void) interrupt 7
{
    PCON1 &= WDTF;                          //清除 WDT 标志(写“1”)
}

viod main (void)
{
    PCON1 &= WDTF;                          //清除 WDT 标志 (写“1”)
    WDTCR |= (NSW | ENW | PS2 | PS1 | PS0);  //使能 WDT 和 NSW (对 watch 模式)
                                            //设置 PS[2:0] = 7 来选择 WDT 周期为 1.984 秒(s)

    SFIE |= WDTFIE;                         //使能 WDT 中断
    EIE1 |= ESF;                            //使能系统标志中断
    EA = 1;                                 //使能全局中断

    PCON0 |= PD;                            //设置 MCU 为掉电模式

//  MCU 等待唤醒
}
```

(5). 规定功能: Monitor 模式 (仅 L-系列)

汇编语言代码范例:

```
    ORG     0003Bh
SystemFlag_ISR:
    ANL     PCON1,#(BOF0)                   ; 清除 BOD0 标志(写“1”)
    RETI

main:
    MOV     IFADRL,#PCON2                   ; 索引 P 页地址为 PCON2
```

```
    CALL    _page_p_sfr_read            ; 读取 PCON2 数据

    ORL     IFD,#AWBOD0                 ; 在掉电模式使能 BOD0 工作
    CALL    _page_p_sfr_write           ; 写数据到 PCON2

    ORL     SFIE,#BOF0IE                ; 使能 BOF0 中断
    ORL     EIE1,#ESF                   ; 使能系统标志中断
    SETB    EA                          ; 使能全局中断

    ORL     PCON0,#PD                   ; 设置 MCU 为掉电模式

;   MCU 等待唤醒
```

C 语言代码范例:

```
void SystemFlag_ISR() interrupt 7
{
    PCON1 &= BOF0;                      // 清除 BOD0 标志(写“1”)
}

void main()
{
    IFADRL = PCON2;                     // 索引 P 页地址为 PCON2
    page_p_sfr_read();                  // 读取 PCON2 数据

    IFD |= AWBOD0;                      //在掉电模式使能 BOD0 工作
    page_p_sfr_write();                 // 写数据到 PCON2


    SFIE |= BOF0IE;                     //使能 BOF0 中断
    EIE1 |= ESF;                        //使能系统标志中断
    EA = 1;                             //使能全局中断

    PCON0 |= PD;                        //设置 MCU 为掉电模式

//  MCU 等待唤醒
```

```
}
```

(6). 规定功能:外部晶振 (XTAL)在掉电模式下安全并且快速唤醒

汇编语言代码范例:

```
    MOV     IFADRL,#CKCON2          ; 索引 P 页地址为 CKCON2
    CALL    _page_p_sfr_read        ; 读取 CKCON2 数据

    ORL     IFD,#IHRCOE             ; 使能 IHRCO
    CALL    _page_p_sfr_write       ; 写数据到 CKCON2

    Delay_32us

    ANL     IFD,#~(OSCS1|OSCS0)     ; OSCin 时钟源更改到 IHRCO
    CALL    _page_p_sfr_write       ; 写数据到 CKCON2

    ORL     PCON0,#PD               ; 设置 MCU 为掉电

;   MCU 等待唤醒

check_XTOR:                         ; 检测外部振荡(XTAL)振荡准备好
    MOV     A,AUXR1
    JNB     ACC.4,check_XTOR        ; 等待 XTOR(AUXR1.4)为 1

    ANL     IFD,#~(OSCS1 | OSCS0)   ; OSCin 时钟源更改为外部振荡(XTAL)
    ORL     IFD,#(OSCS0)
    CALL    _page_p_sfr_write       ; 写数据到 CKCON2

    ANL     IFD,#~(IHRCOE)          ; 如果 MCU 从 IHRCO 切换过来则禁止 IHRCO
    CALL    _page_p_sfr_write       ; 写数据到 CKCON2
```

C 语言代码范例:

```
    IFADRL = CKCON2;                // 索引 P 页地址为 CKCON2
    page_p_sfr_read();              // 读取 CKCON2 数据
```

```
    IFD |= IHRCOE;                          // 使能 IHRCO
    page_p_sfr_write();                     // 写数据到 CKCON2

    Delay_32us();

    IFD &= ~(OSCS1 | OSCS0);                // OSCin 时钟源更改到 IHRCO
    page_p_sfr_write();                     // 写数据到 CKCON2

    PCON0 |= PD;                            //设置 MCU 为掉电

//  MCU 等待唤醒

    while(AUXR1 & XTOR == 0x00);            //检测外部振荡(XTAL)振荡准备好
                                            //等待 XTOR(AUXR1.4)为 1

    IFD &= ~(OSCS1 | OSCS0);                // OSCin 时钟源更改为外部振荡(XTAL)
    IFD |= OSCS0;
    page_p_sfr_write ();                    // 写数据到 CKCON2

    IFD &= ~IHRCOE;                         //如果 MCU 从 IHRCO 切换过来则禁止 IHRCO
    page_p_sfr_write();                     // 写数据到 CKCON2.
```

12. 输入输出口配置

MA86E/L104 有下列 I/O 端口：P1.0~P1.7, P3.0~P3.5 及 P3.7。RST 引脚与 P3.6 有互换功能。如果选择外部振荡器做系统时钟输入则 P4.0 和 P4.1 被配置为 XTAL2 和 XTAL1。准确的可用 I/O 引脚数量由封装类型决定。见表 12-1。

表 12-1 可用引脚数量

封装类型	I/O 引脚	引脚数量
20-pin PDIP	P1.0~P1.7, P3.0~P3.5, P3.7, RST(P3.6), XTAL2(P4.0), XTAL1(P4.1)	15 or 16 (RST/P3.6) or 18 (INTOSC 使能)
16-pin PDIP	P1.0~P1.6, P3.0~P3.3, RST(P3.6), XTAL2(P4.0), XTAL1(P4.1)	11 or 12 (RST/P3.6) or 14 (INTOSC 使能)
8-pin SOP	P3.0~P3.5	6

12.1. 输入输出口结构

MA86E/L104 输入输出口分成两个配置类型。第一类仅仅是端口 3 有四种模式，这四种模式有：准双向口(标准 8051 的 I/O 端口)、推挽输出、集电极开漏输出和输入(高阻抗输入)。

其它口属于第二类，这些口有两种模式分别是推挽输出和上拉电阻的集电极开漏输出。

下面描述这四种类型的 I/O 模式的配置。

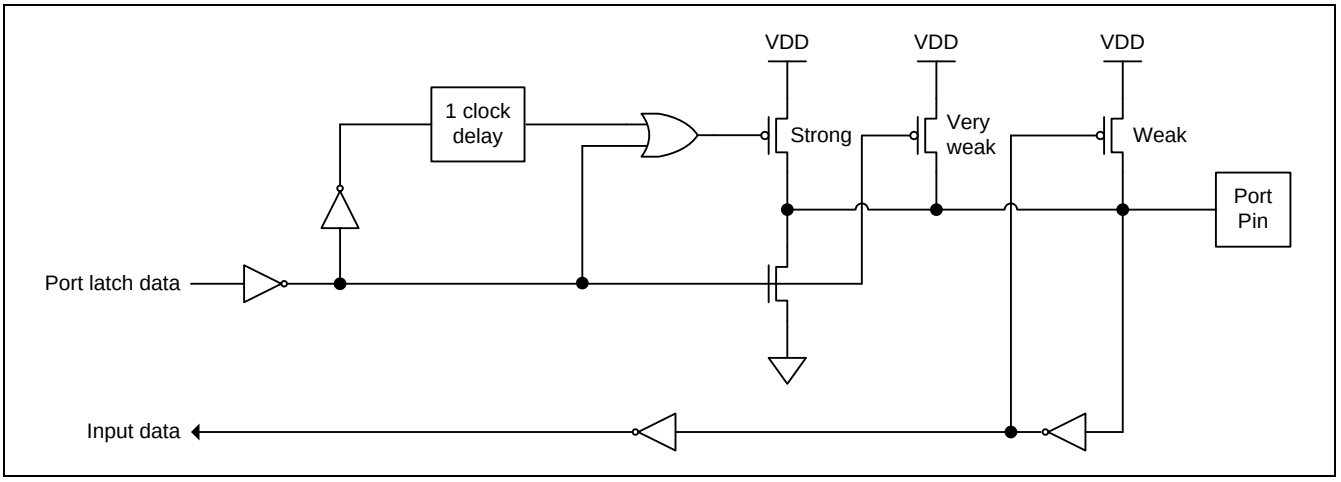
12.1.1. 端口 3 准双向口

端口 3 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，当端口寄存器发生从逻辑 0 到逻辑 1 跳变时，强上拉打开一个 CPU 时钟，快速将端口引脚拉高。

准双向端口 3 配置如图 13-1 所示。

图 12-1 准双向端口 3 配置

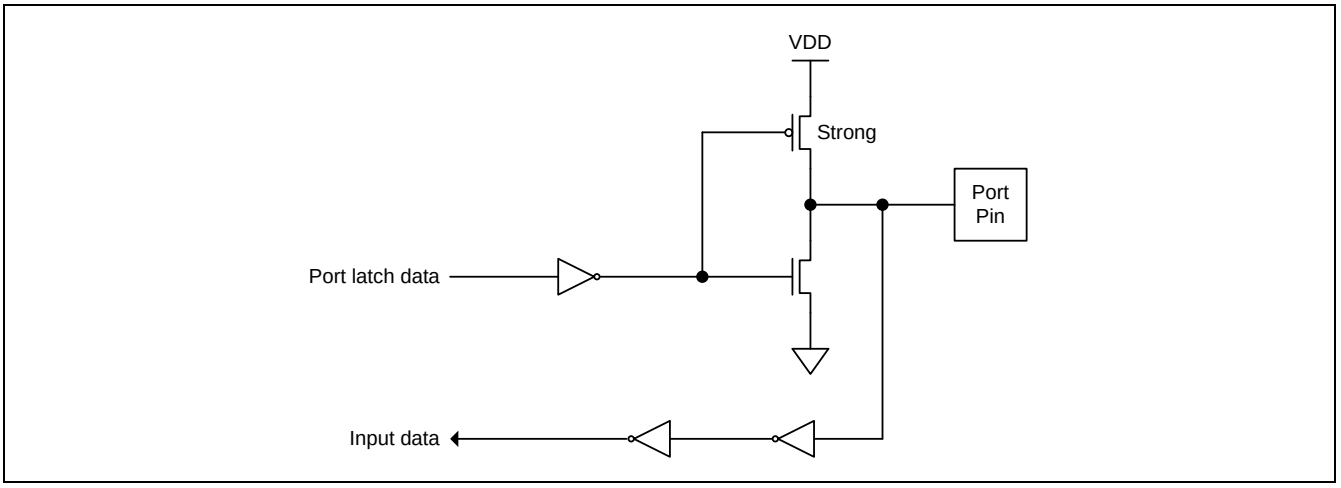


12.1.2. 端口 3 推挽输出

端口 3 推挽输出配置与开漏输出、准双向输出模式有着相同的下拉结构，但是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，在这种配置下端口的输入路径与准双向模式相同。

端口 3 推挽输出配置见 13-2。

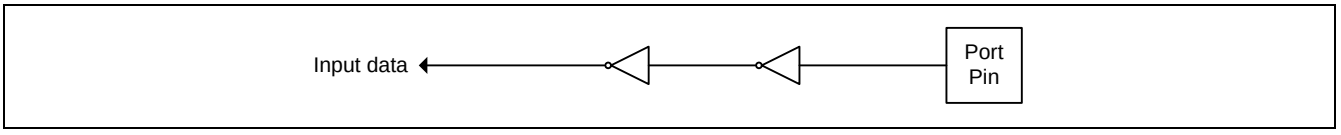
图 12-2 端口 3 推挽输出



12.1.3. 端口 3 仅是输入（高阻抗输入）模式

仅输入配置在引脚上没有任何上拉电阻，如下图 13-3 所示。

图 12-3 端口 3 仅是输入

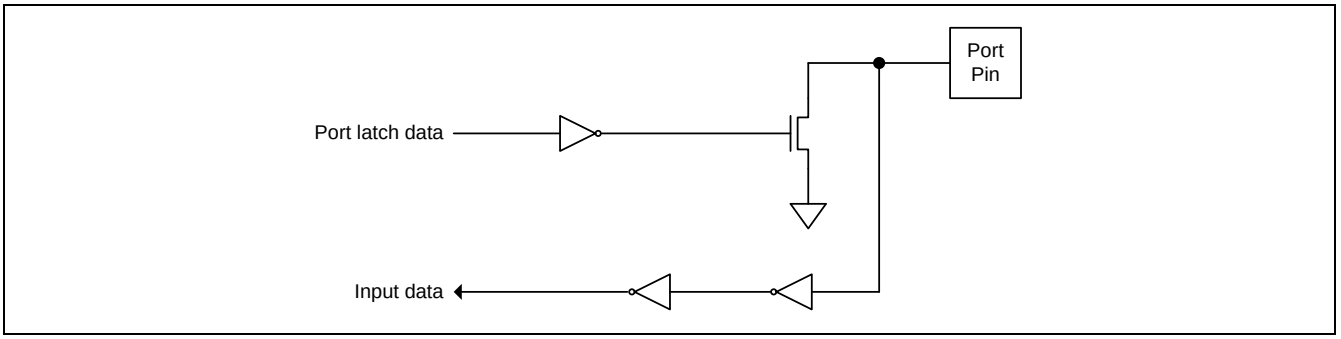


12.1.4. 端口 3 开漏输出

端口 3 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。在应用中使用这个配置，端口引脚必须有外部上拉，典型的是将电阻接到 VDD。这个模式的下拉和准双向端口的模式相同。另外，在这种配置下端口的输入路径与准双向模式相同。

开漏输出端口 3 配置如图 13-4 所示。

图 12-4 端口 3 开漏输出

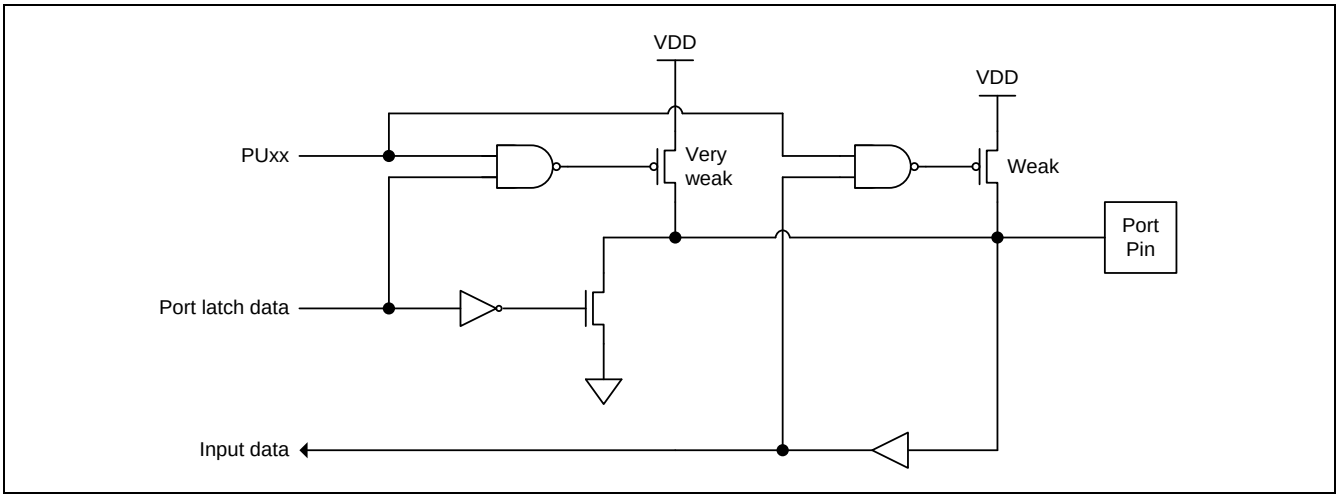


12.1.5. 通用端口集电极开漏输出结构

当端口数据寄存器写”0”时，通用端口的集电极开漏输出仅是驱动端口的下拉晶体管。在应用中使用这个配置，端口引脚可以选择外部上拉或 PUCON0 置位使能片内的内部上拉。

通用端口集电极开漏结构如图 13-5 所示。

图 12-5 通用端口集电极开漏输出

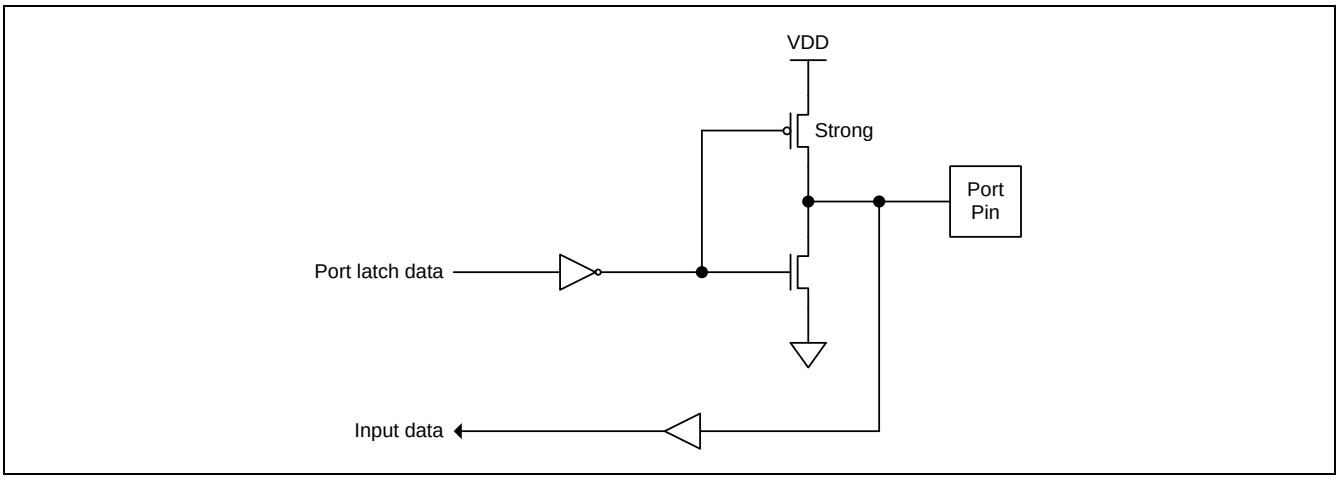


12.1.6. 通用端口推挽输出结构

通用端口推挽输出结构与集电极开漏输出有相同的下拉结构，但是端口数据寄存器写”1”提供一个强上拉输出。通用端口推挽输出模式应用在需要强的源电流输出。另外，此结构下的端口输入与集电极开漏输出模式一样。

通用端口推挽输出结构如图 13-6 所示。

图 12-6 通用端口推挽输出



12.1.7. 通用端口输入结构

端口引脚在设置为集电极开漏模式并且端口数据寄存器写“1” 的情况下可以作为输入口。例如,P1M0.7=0 及 P1.7 写入”1”，这样 P1.7 就定义为输入口。

12.2. 输入输出口寄存器

MA86E/L104 的端口 3 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 13-2 所示。每个端口有两个模式寄存器来选择各端口引脚的输出类型。

表 12-2 端口 3 配置设定

P3M0.y	P3M1.y	端口模式
0	0	准双向
0	1	推挽输出
1	0	输入口 (高阻抗输入)
1	1	集电极开漏输出

这里 y=0~7(端口引脚号)。寄存器 P3M0 和 P3M1 列举了每个引脚的描述。

其它的通用口引脚有两种模式见表 13-3，一个模式寄存器位选择每个引脚的输出类型。

表 12-3 端口配置设定

PxM0.y	端口模式
0	集电极开漏输出
1	推挽输出

这里 x=1, 4 (端口)，y=0~7(端口引脚号)。寄存器 P1M0 和 P4M0 列举了每个引脚的描述。

12.2.1. 端口 1 寄存器

P1: 端口 1 寄存器

SFR 地址 = 0x90　　复位初始值 = 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 通过软件置位/清零 。

P1M0: 端口 1 模式寄存器 0

SFR 地址 = 0x91　　复位初始值= 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**

0: 端口定义为集电极开漏输出。
1: 端口定义为推挽输出。

12.2.2. 端口 3 寄存器

P3: 端口 3 寄存器

SFR 地址 = 0xB0　　复位初始值= 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 通过软件置位/清零 。

P3M0: 端口 3 模式寄存器 0

SFR 地址 = 0xB1　　复位初始值= 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: Port 3 Mode Register 1

SFR 地址 = 0xB2　　复位初始值= 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**

12.2.3. 端口 4 寄存器

P4: 端口 4 寄存器

SFR 地址 = 0xE8　　复位初始值= xxxx-xx11

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

Bit 7~2: 保留。

Bit 1~0: P4.1~P4.0 通过软件置位/清零。当内部振荡器被使能做系统时钟时这两个 I/O 被激活，这样 XTAL1 和 XTAL2 成为 P4.1 和 P4.0。

P4M0: 端口 4 模式寄存器 0

SFR 地址 = 0xB3　　复位初始值= xxxx-xx00

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

0: 端口定义为集电极开漏输出。
1: 端口定义为推挽输出。

12.2.4. 上拉控制寄存器

PUCON0: 端口上拉控制寄存器 0

SFR 地址 = 0xB4　　复位初始值= x0xx-00xx

7	6	5	4	3	2	1	0
--	PU40	--	--	PU11	PU10	--	--
W	**R/W**	**W**	**W**	**R/W**	**R/W**	**W**	**W**

Bit 7: 保留位。当 PUCON0 改写时，此位必须软件写”0”。

Bit 6: 端口 4 低四位上拉使能控制
0: 在集电极开漏输出模式禁止 P4.0 和 P4.1 上拉。
1: 在集电极开漏输出模式使能 P4.0 和 P4.1 上拉。

Bit 5~4: 保留位。当 PUCON0 改写时，这两位必须软件写”0”。

Bit 3: 端口 1 高四位上拉使能控制
0: 在集电极开漏输出模式禁止 P1.7~P1.4 上拉。
1: 在集电极开漏输出模式使能 P1.7~P1.4 上拉。

Bit 2: 端口 1 低四位上拉使能控制
0: 在集电极开漏输出模式禁止 P1.3~P1.0 上拉。
1: 在集电极开漏输出模式使能 P1.3~P1.0 上拉。

Bit 1~0: 保留位。当 PUCON0 改写时，这两位必须软件写”0”。

12.3. GPIO 示例代码

(1). 规定功能: 设置 P1.0 为片内上拉电阻使能的输入模式

汇编语言代码范例:

```
ANL     P1M0,#~P1M00            ; 配置 P1.0 为漏极开路模式
SETB    P10                     ; 设置 P1.0 数据为“1”而使能输入模式
ORL     PUCON0,#PU10            ; 使能 P1.3~P1.0 片内上拉电阻
```

C 语言代码范例:

```
P1M0 &= P1M00;                  //配置 P1.0 为漏极开路模式
P10 = 1;                        //设置 P1.0 数据为“1”而使能输入模式
PUCON0 |= PU10;                 //使能 P1.3~P1.0 片内上拉电阻
```

(2). 规定功能: 选择 RST 引脚为 P3.6

汇编语言代码范例:

```
MOV     IFADRL,#DCON0           ; 索引 P 页地址为 DCON0
CALL    _page_p_sfr_read        ; 读取 DCON0 数据

ANL     IFD,#~(RSTIO)           ; 选择 I/O 功能为 P36
CALL    _page_p_sfr_write       ; 写数据到 DCON0
```

C 语言代码范例:

```
IFADRL = DCON0;                 // 索引 P 页地址为 DCON0
page_p_sfr_read();              // 读取 DCON0 数据.

IFD &= ~RSTIO;                  //选择 I/O 功能为 P36
page_p_sfr_write();             // 写数据到 DCON0
```

13. 中断

MA86E/L104 有四级中断优先级的 6 个中断源。与这四级中断有关联的特殊功能寄存器有 IE、 IP0L、 IP0H、EIE1、EIP1L 及 EIP1H。IP0H（中断优先级 0 高位)和 EIP1H（外部中断优先级 1 高位） 寄存器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。

13.1. 中断结构

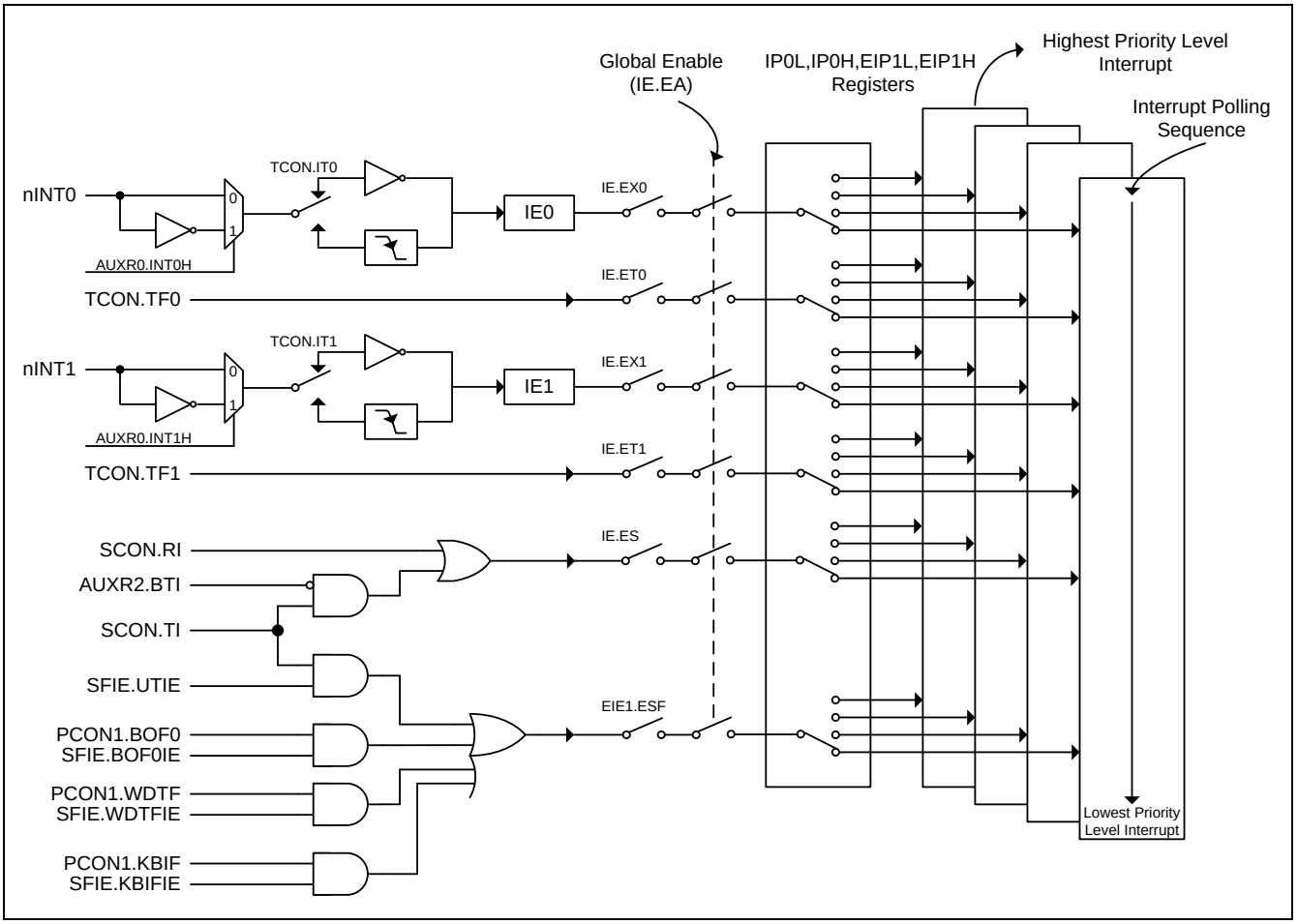
表 13-1列出了所有的中断源。使能位被允许，中断请求时硬件会产生一个中断请求标志，当然，总中断使能位EA（IE寄存器)必须使能。中断请求位能由软件置1或清零，这和硬件置1或清零结果相同。同理，中断可以由软件产生或取消，中断优先级位决定每个中断产生的优先级，多个中断同时产生时依照中断优先级顺序处理。中断向量地址表示中断服务程序的入口地址。

图 13-1展示了整个中断系统。每一个中断将在下面部分做简单的描述。

表 13-1 中断源

	中断源	使能位	请求位	优先级位	优先级顺序	中断向量地址
#1	外部中断 0, nINT0	EX0	IE0	[PX0H, PX0L]	(最高)	0003H
#2	定时器 0	ET0	TF0	[PT0H, PT0L]	...	000Bh
#3	外部中断 1, nINT1	EX1	IE1	[PX1H, PX1L]	...	0013H
#4	定时器 1	ET1	TF1	[PT1H, PT1L]	...	001BH
#5	串行口	ES	RI, TI	[PSH, PSL]	...	0023H
#6	系统标志	ESF	BOF0,WDTF KBIF,(TI)	[PSFH, PSFL]	(最低)	002BH

图 13-1 中断系统



13.2. 中断源

表 13-2 中断源旗标

	中断源	请求位	请求位位置
#1	外部中断 0, nINT0	IE0	TCON.1
#2	定时器 0	TF0	TCON.5
#3	外部中断, nINT1	IE1	TCON.3
#4	定时器 1	TF1	TCON.7
#5	串行口	RI0 TI0	SCON0.0 SCON0.1
#6	系统标志	WDTF BOF0 KBIF (TI)	T2CON.7 T2CON.6

外部中断 nINT0 和 nINT1 分别通过 TCON 的 IT0 和 IT1 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 和 IE1。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 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，定时器 0 工作在模式 3 时除外)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 的逻辑产生。进入中断服务程序后，这些标志均不会被硬件清除。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。.

系统标志中断由 PCON1 的 KBIF, BOF0 和 WDTF 产生。KBIF 由 KBI 事件置位。BOF0 在检测到低电压时置位。WDTF 看门狗溢出置位。这些标志位进入中断服务程序后由片内硬件清除标志位。

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。

13.3. 中断使能

表 13-3. 中断使能

No	中断源	使能位	使能位位置
#1	外部中断 0, nINT0	EX0	IE.0
#2	定时器 0	ET0	IE.1
#3	外部中断 1, nINT1	EX1	IE.2
#4	定时器 1	ET1	IE.3
#5	串行口	ES0	IE.4
#6	系统标志	ESF & (UTIE, KBIFIE, BOF0IE, WDTFIE)	EIE1.3 & SFIE.7,3,1,0

MA86E/L104 有 11 个可用的中断源，通过设置寄存器 IE 和 EIE1 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’则使能全局中断（使能和禁止由各自单独的设置位决定）。EA 清‘0’，所有中断被禁止。

13.4. 中断优先级

中断服务程序的优先级同 80C51 一样，除了 4 个优先级对应 80C51 的 2 个优先级之外。优先级的设置位（参考表 13-1）决定每个中断的优先级别，IP0L, IP0H, EIP1L 和 EIP1H 组合成四级优先级中断，表 13-44 标示设置位的值于优先级的对应关系。

表 13-4 中断优先级

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

每一个中断有两个相应位表现它的优先级。一个是 IPxH 寄存器，另外一个是 IPxL 寄存器。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。

13.5. 中断处理

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：

阻止条件：

- 一个同等或高优先级的中断正在处理。
- 当前机器周期不是正在执行的指令的最后一个机器周期。
- 正在执行指令 RETI 或正在写和中断相关的寄存器(IE, IP0L, IPH, EIE1, EIP1L 及 EIP1H 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（如 IE 或 IP 等）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。C

13.6. TI 的特别中断向量

TI 标志位的串行口中断可以通过 BTI (AUXR2.6)来屏蔽。如果 BTI 置位，则 TI 置位也不会产生串行口中断，这样串行口中断仅仅是 RI 标志位中断。

如果 UTIE (SFIE.7)置位，TI 标志位会产生系统标志中断。在此模式下，TI 中断与 KBIF, BOF0 和 WDTF 一起共用系统标志中断。

13.7. 中断寄存器

TCON:定时器/计数器控制寄存器

SFR 地址 = 0x88　　复位初始值 = 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**

Bit 3: IE1, 外部中断 1 请求标志。
0: 如果是边沿触发的中断则在进入中断向量后硬件清零。
1: 外部中断 1 由边沿或电平触发（由 IT1 设置）硬件置标志。

Bit 2: IT1: 外部中断 1 类型控制位
0: 软件选择低电平触发外部中断 0。如果 INT1H (AUXR0.1)置位，则 nINT1 高电平触发外部中断 0。
1: 软件选择下降沿触发外部中断 0。如果 INT1H (AUXR0.1)置位，则 nINT1 上升沿触发外部中断 0。

Bit 1: IE0, 外部中断 0 请求标志。
0: 如果是边沿触发的中断则在进入中断向量后硬件清零。
1: 外部中断 0 由边沿或电平触发（由 IT0 设置）硬件置标志。

Bit 0: IT0: 外部中断 0 类型控制位
0: 软件选择低电平触发外部中断 0。如果 INT0H (AUXR0.0)置位，则 nINT0 高电平触发外部中断 0。
1: 软件选择下降沿触发外部中断 0。如果 INT0H (AUXR0.0)置位，则 nINT0 上升沿触发外部中断 0。

IE: 中断使能寄存器

SFR 地址 = 0xA8　　复位初始值= 0xx0-0000

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

Bit 7: EA, 总中断使能位
0: 全局禁止所有中断。
1: 全局使能所有中断

Bit 6: 保留。当对 IE 进行写的时候必须对此位写“0”

Bit 5: 保留。当对 IE 进行写的时候必须对此位写“0”

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

Bit 3: ET1, 定时器 1 中断使能
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 中断使能
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
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 中断使能
0: 禁止外部中断 0
1: 使能外部中断 0

EIE1: 扩展中断使能 1 寄存器

SFR 地址 = 0xAD　　复位初始值= xxxx-0xxx

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

Bit 7~4: 保留。当对 IEI1 进行写的时候必须对这些位写“0”

Bit 3: ESF, 使能系统标志中断
0: 当 PCON1 的 KBIF, BOF0, WDTF 位或 SCON 的 TI 位置位时禁止中断。
1: 当 PCON1 的 KBIF, BOF0, WDTF 位或 SCON 的 TI 位在 SFIE 相应使能时使能中断。

Bit 2~0: 保留。当对 IEI1 进行写的时候必须对这些位写“0”

SFIE: 系统标志中断使能寄存器

SFR 地址 = 0x8E　　复位初始值= 0xxx-0x00

7	6	5	4	3	2	1	0
UTIE	--	--	--	KBIFIE	--	BOF0IE	WDTFIE
R/W	**W**	**W**	**W**	**R/W**	**W**	**R/W**	**R/W**

Bit 7: 串行口 TI 中断使能
0: 清零系统标志中断禁止 TI 中断。
1: 置位系统标志中断使能 TI 中断。

Bit 6~4: 保留。当对 SFIE 进行写的时候必须对这些位写“0”。

Bit 3: KBIFIE, KBIF (PCON1.3)中断使能
0: 清零禁止 KBIF 中断。
1: 置位使能 KBIF 中断。

Bit 2: 保留。当对 SFIE 进行写的时候必须对此位写“0”。

Bit 1: BOF0IE, BOF0 (PCON1.1) 中断使能
0: 清零禁止 BOF0 中断。
1: 置位使能 BOF0 中断。

Bit 0: WDTFIE, WDTF (PCON1.0)中断使能
0: 清零禁止 WDTF 中断。
1: 置位使能 WDTF 中断。

IP0L:中断优先级 0 低 8 位寄存器

SFR 地址 = 0xB8　　复位初始值= xxx0-0000

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

Bit 7~5: 保留。当对 IP0L 进行写的时候必须对这些位写“0”。
Bit 4: PSL, 串口中断优先级低位。

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, 外部中断 0 优先级低位。

IP0H: 中断优先级 0 高 8 位寄存器

SFR 地址 = 0xB7　　复位初始值= xxx0-0000

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

Bit 7~5: 保留。当对 IP0H 进行写的时候必须对这些位写“0”。
Bit 4: PSH, 串口中断优先级高位。
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, 外部中断 0 优先级高位。

EIP1L: 扩展中断优先级 1 低 8 位寄存器

SFR 地址 = 0xAE　　复位初始值= xxxx-0xxx

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

Bit 7~4: 保留。当对 EIP1L 进行写的时候必须对这些位写“0”。
Bit 3: PSFL, 系统标志中断优先级低位。
Bit 2~0: 保留。当对 EIP1L 进行写的时候必须对这些位写“0”。

EIP1H: 扩展中断优先级 1 高 8 位寄存器

SFR 地址 = 0xAF　　复位初始值= xxxx-0xxx

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

Bit 7~4: 保留。当对 EIP1H 进行写的时候必须对这些位写“0”。
Bit 3: PSFH, 系统标志中断优先级高位。
Bit 2~0: 保留。当对 EIP1H 进行写的时候必须对这些位写“0”。

AUXR0: 辅助寄存器 0

SFR 地址 = 0xA1　　复位初始值= 000x-0000

7	6	5	4	3	2	1	0
P40OC1	P40OC0	P40FD	--	P1FS1	P1FS0	**INT1H**	**INT0H**
R/W	**R/W**	**R/W**	**W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 1: INT1H, INT1 高电平/上升沿触发使能
0: 保留 nINT1 引脚在低电平或下降沿触发 INT1。
1: 使能 nINT1 引脚在高电平或上升沿触发 INT1。

Bit 0: INT0H, INT0 高电平/上升沿触发使能
0: 保留 nINT0 引脚在低电平或下降沿触发 INT0。
1: 使能 nINT0 引脚在高电平或上升沿触发 INT0。

13.8. 中断示例代码

(1). 规定功能: 在掉电模式下设置 INT0 高电平唤醒 MCU

汇编语言代码范例:

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

main:

    SETB    P32                         ;

    ORL     IP0L,#PX0L                  ; 选择 INT0 中断优先级
    ORL     IP0H,#PX0H                  ;

    ORL     AUXR0,#INT0H                ; 设置 INT0 高电平激活

    JB      P32,$                       ; 确认 P3.2 输入低

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

    ORL     PCON0,#PD                   ; 设置 MCU 进入掉电模式
```

C 语言代码范例:

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

```
void main(void)
{
    P32 = 1;

    IP0L |= PX0L;                       //选择 INT0 中断优先级
    IP0H |= PX0H;

    AUXR0 |= INT0H;                     //设置 INT0 高电平激活

    while(P32);                         //确认 P3.2 输入低

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

    PCON0 |= PD;                        //设置 MCU 进入掉电模式
}
```

14. 定时器/计数器

MA86E/L104 有两个 16 位的定时器/计数器：定时器 0 和定时器 1。每一个包含两个 8 位寄存器 THx 和 TLx(这里，x=0、1 或 2)。它们可配置为定时器或事件计数器。

定时器功能，TLx 寄存器每 12 个系统时钟周期（标准 C51 的机器周期）或每 1 个系统时钟周期（是标准 C51 的 12 倍）加 1，通过软件设置 AUXR2.T0X12 或 AUXR2.T1X12 位来选择 。每 12 个系统时钟周期加一，计数速率是 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 或 T1 寄存器加 1。在这功能中，每个定时器时钟周期对外部输入信号(T0 和 T1 引脚)进行采样，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。

14.1. 定时器 0 和 定时器 1

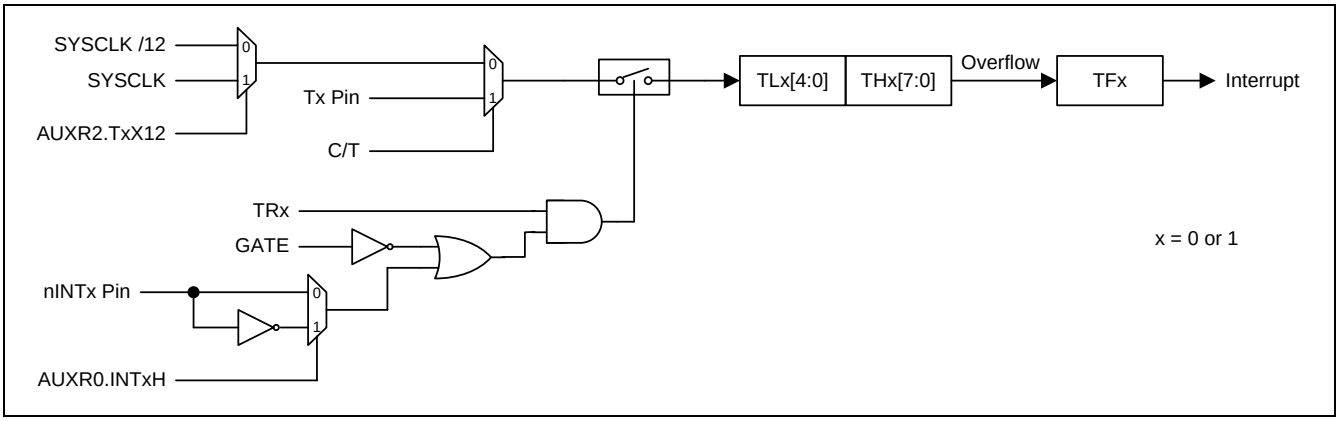
14.1.1. 模式 0 结构

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。定时器0和定时器1的模式0运作时相同的。

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。

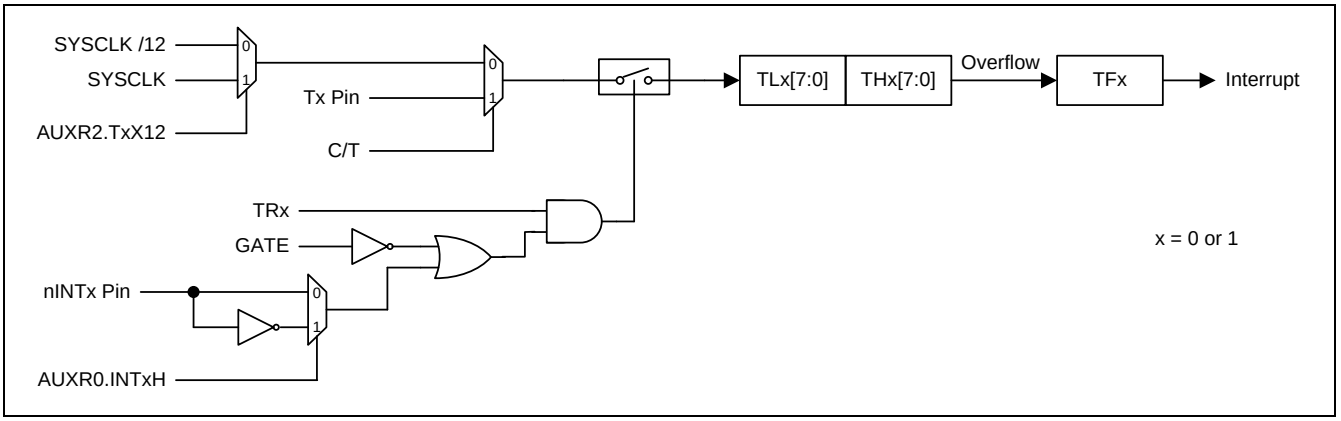
图 14-1 定时器 0/1 模式 0 结构



14.1.2. 模式 1 结构

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。

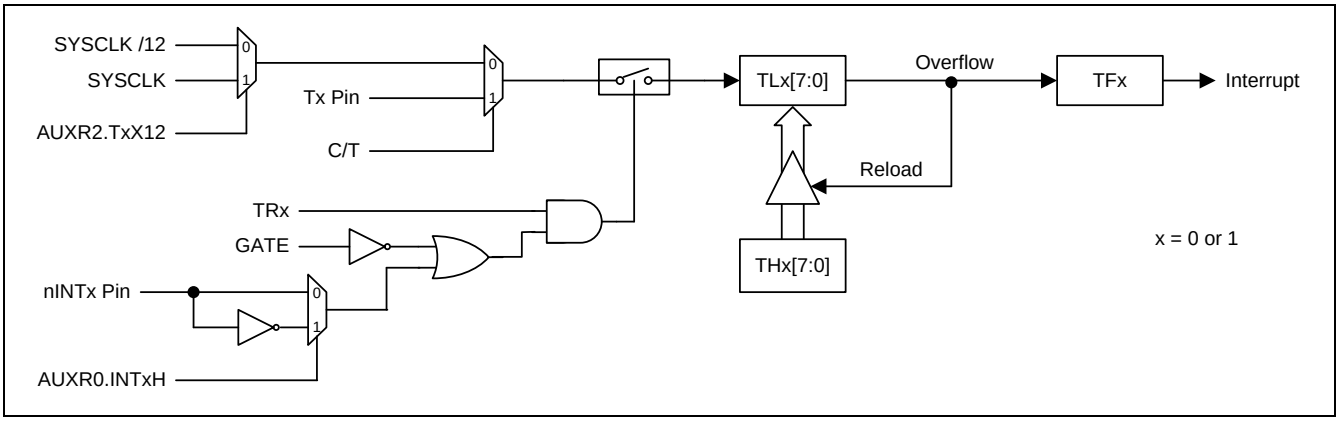
图 14-2. 定时器 0/1 模式 1 结构



14.1.3. 模式 2 结构

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。定时器 0 和定时器 1 的模式 2 运作时相同的。

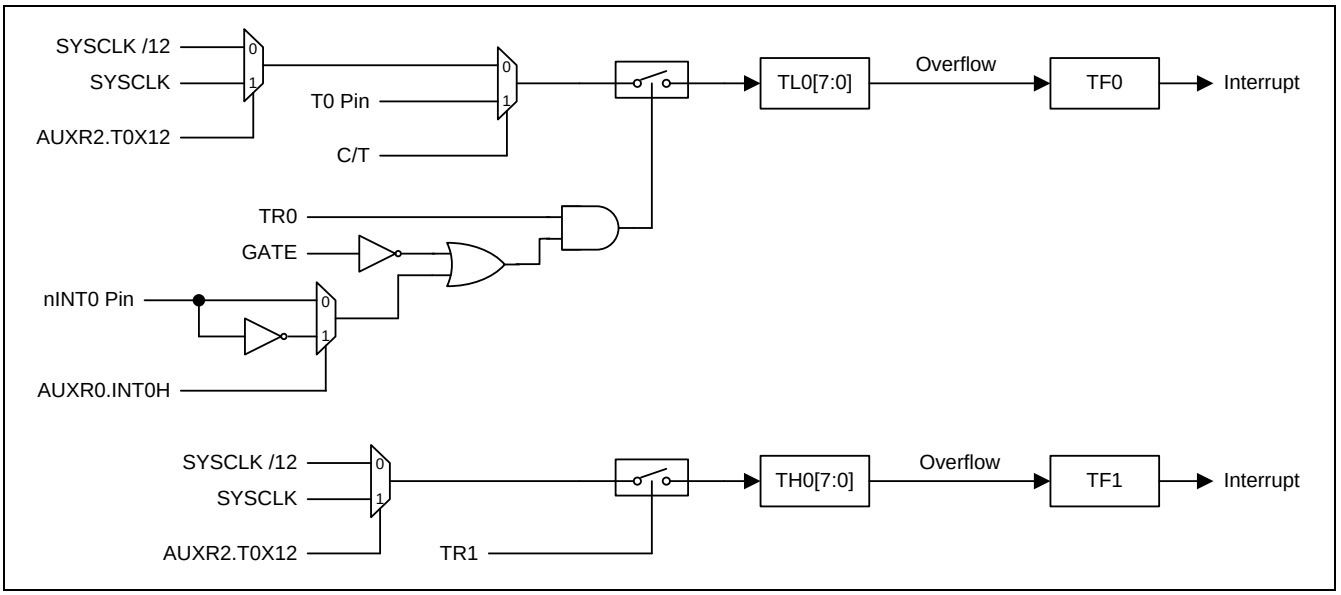
图 14-3. 定时器 0/1 模式 2 结构



14.1.4. 模式 3 结构

定时器1在模式3保持计数值。效果和设置TR1=1一样。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(不能成为外部事件计数器)且接替定时器1来使用TR1和TF1，TH0控制定时器1中断。

图 14-4. 定时器 0 模式 3 结构



14.1.5. 定时器 0/1 可编程时钟输出

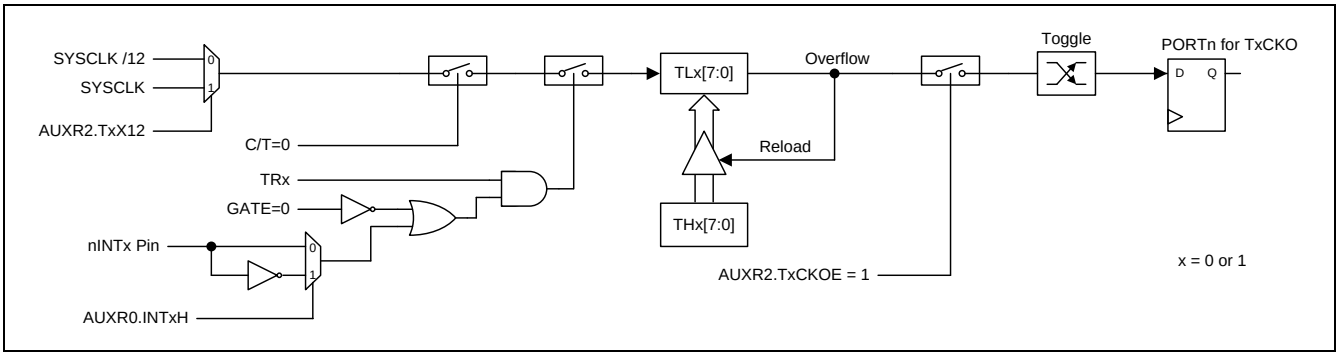
在 C/Tx=0 及 TxCKOE=1 定时器 0 和 1 是个时钟输出模式。在此模式下，定时器 0 和 1 运行在占空比为 1:1 八位自动装载的可编程时钟产生器。产生的时钟在 P3.4 (T0CKO) 和 P3.5 (T1CKO)各自独立输出。八位定时器(TL0 or TL1)由输入时钟 (SYSCLK/12 或 SYSCLK) 加 1。定时器从一个装载的初值一直计数到溢出，一旦溢出(TH0,TH1)的值就会载入到(TL0, TL1)继续计数。下面是输出频率的计算公式：

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

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

注意：
(1)定时器 0/1 的溢出标志在溢出时将被设置 但是不产生中断。
(2)系统时钟 SYSCLK=12MHz 及 TxX12=0,定时器 0/1 可编程频率输出范围在 1.95KHz 到 500KHz。
(3) 系统时钟 SYSCLK=12MHz 及 TxX12=1, 定时器 0/1 可编程频率输出范围在 23.43KHz 到 6MHz。

图 14-5. 定时器 0/1 始终输出模式



定时器 0/1 在输出模式怎样编程

- AUXR2 寄存器的 T0X12/T1X12 位选择定时器 0/1 的时钟源。
- 置位 AUXR2 寄存器的 T0CKOE/T1CKOE。
- TMOD 寄存器的 C/T 位清零。
- 决定输出频率的 8 位装载数据写入 TH0/TH1 寄存器。
- 决定输出频率的 8 位装载数据同时写入 TL0/TL1 寄存器。
- 置位 TCON 寄存器的 TR0/TR1 位启动定时器 0/1。

在时钟输出模式定时器 0/1 不会产生中断，这个跟定时器 1 被用作波特率发生器一样。这样定时器 1 就可以同时作为时钟输出和波特率发生器。注意：时钟输出频率和波特率都是跟定时器 1 的溢出率一样。

14.1.6. 定时器 0/1 寄存器

TCON:定时/寄存器控制寄存器

SFR 地址 = 0x88　　复位初始值 = 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**

Bit 7: TF1, 定时器 1 溢出标志位。
0: 处理器进入中断向量程序由硬件清零，或由软件清零。
1: 定时器/计数器溢出时由硬件置位，或由软件置位。

Bit 6: TR1, 定时器 1 运行控制位。
0: 软件清零关闭定时器/计数器 1。
1: 软件置位开启定时器/计数器 1。

Bit 5: TF0, 定时器 0 溢出标志位。
0: 处理器进入中断向量程序由硬件清零，或由软件清零。
1: 定时器/计数器溢出时由硬件置位，或由软件置位。

Bit 4: TR0, 定时器 0 运行控制位。
0: 软件清零关闭定时器/计数器 0。
1: 软件置位开启定时器/计数器 0。

TMOD: 定时/寄存器模式控制寄存器

SFR 地址 = 0x89　　复位初始值= 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 ----------→|		

Bit 7/3: Gate,定时器 1/0 门控制
0: 禁止定时器 1/0 门控位。
1: 使能定时器 1/0 门控位。当门控位置位时，只有在 nINT0 或 nINT1 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。

Bit 6/2: C/T, 定时器或计数器功能选择位。
0: 清零为定时器功能(从内部系统时钟输入)。
1: 置位为计数器功能(从 T0 或 T1 引脚输入)。

Bit 5~4/1~0: 工作模式选择

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

TL0: 定时器 0 低 8 位寄存器

SFR 地址 = 0x8A　　复位初始值 = 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 8 位寄存器

SFR 地址 = 0x8C　　复位初始值= 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 8 位寄存器

SFR 地址 = 0x8B　　复位初始值= 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 8 位寄存器

SFR 地址 = 0x8D　　复位初始值= 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　　复位初始值= 0000-0000

7	6	5	4	3	2	1	0
URXR	BTI	URM0X6	SMOD2	**T1X12**	**T0X12**	T1CKOE	T0CKOE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 3: T1X12, 当 C/T=0 时，定时器 1 的时钟源选择。
0: 清零选择 SYSCLK/12 作为定时器 1 时钟源。
1: 置位选择 SYSCLK 作为定时器 1 时钟源。

Bit 2: T0X12, 当 C/T=0 时，定时器 0 的时钟源选择。
0: 清零选择 SYSCLK/12 作为定时器 0 时钟源。
1: 置位选择 SYSCLK 作为定时器 0 时钟源。

Bit 1: T1CKOE, 定时器 1 时钟输出使能
0: 禁止定时器 1 时钟输出。
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 1 时钟输出。
1: 使能定时器 1 时钟从 P3.4 输出。

14.1.7. 定时器 0/1 示例代码

(1). 规定功能:系统时钟 SYSCLK = ILRCO 时定时器 T0 以 320Hz 的频率唤醒空闲模式

汇编语言代码范例:

```
    ORG     0000Bh
time0_isr:
    to do…
    RETI

main:                                   ;
;   //选择系统时钟 Sysclk 为 ILRCO
    MOV     IFADRL,#(CKCON2)            ; 索引 P 页地址为 CKCON2
    CALL    _page_p_sfr_read            ; 读取 CKCON2 数据

    ANL     IFD,#~(OSCS1 | OSCS0)       ; OSCin 时钟源更改为 ILRCO
    ORL     IFD,#(OSCS1)
    CALL    _page_p_sfr_write           ; 写数据到 CKCON2

    ANL     IFD,#~(XTALE | IHRCOE)      ; 禁止 XTAL 和 IHRCO
    CALL    _page_p_sfr_write           ; 写数据到 CKCON2

    MOV     IFADRL,#(DCON0)             ; 索引 P 页地址为 DCON0
    CALL    _page_p_sfr_read            ; 读取 DCON0 数据

    ANL     IFD,#~(HSE)                 ; 当系统时钟 SYSCLK ≤ 6MHz 为了省电禁止 HSE
    CALL    _page_p_sfr_write           ; 写数据到 DCON0

    ANL     CKCON0,#(AFS)               ; 选择 SCKS[2:0] = 0 = OSCin/1

    ORL     AUXR2,#T0X12                ; 选择 SYSCLK/1 作为定时器 0 时钟输入
    ANL     AUXR0,#~T0XL                ;

    MOV     TH0,#(256-100)              ; 设置定时器 0 溢出率= SYSCLK x 100
    MOV     TL0,#(256-100)              ;
```

```
    ANL     TMOD,#(0F0h|T0M1)          ; 设置定时器 0 工作在模式 2
    ORL     TMOD,#T0M1                 ;
    CLR     TF0                        ; 清除定时器 0 标志

    ORL     IP0L,#PT0L                 ; 选择定时器 0 中断优先级
    ORL     IP0H,#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     PCON0,#IDL                 ; 设置 MCU 进入空闲模式
```

C 语言代码范例:

```
void time0_isr(void) interrupt 1
{
    To do…
}

void main(void)
{
    IFADRL = CKCON2;                    // 索引 P 页地址为 CKCON2
    page_p_sfr_read();                  // 读取 CKCON2 数据.

    IFD = ~(OSCS1 | OSCS0);             // OSCin 时钟源更改为 ILRCO
    IFD |= OSCS1;
    page_p_sfr_write();                 // 写数据到 CKCON2

    IFD &= ~(XTALE | IHRCOE);           //禁止 XTAL 和 IHRCO
    page_p_sfr_write();                 // 写数据到 CKCON2

    IFADRL = DCON0;                     // 索引 P 页地址为 DCON0
    page_p_sfr_read();                  // 读取 DCON0 数据

    IFD &= ~HSE;                        //当系统时钟 SYSCLK ≤ 6MHz 为了省电禁止 HSE
```

```
    page_p_sfr_write();                 //  写数据到 DCON0

    CKCON0 &= AFS;                      //  选择 SCKS[2:0] = 0 = OSCin/1

    AUXR2 |= T0X12;                     //选择 SYSCLK/1 作为定时器 0 时钟输入
    AUXR0 &= ~T0XL;

    TH0 = TL0 = (256-100);              //设置定时器 0 溢出率= SYSCLK x 100
    TMOD &= 0xF0;                       //设置定时器 0 工作在模式 2
    TMOD |= T0M1;
    TF0 = 0;                            //清除定时器 0 标志

    IP0L |= PT0L;                       //选择定时器 0 中断优先级
    IP0H |= 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

    PCON0=IDL;                          //设置 MCU 进入空闲模式
}
```

(2). 规定功能: SYSCLK/48 时钟源设置定时器 0 的时钟输出

汇编语言代码范例:

```
    CLR     TR0                         ;

    ANL     P3M0,#0EFh                  ; 设置 P3.4(T0CKO)为推挽输出
    ORL     P3M1,#010h                  ;
    ORL     AUXR2,#T0CKOE               ; 使能 T0CKO

    ANL     AUXR2,#~T0X12               ; 选择 SYSCLK/48 作为定时器 0 时钟输入
    ORL     AUXR0,#T0XL                 ;

    MOV     TH0,#0FFh                   ;
```

```
MOV       TL0,#0FFh                    ;

ANL       TMOD,#0F0h                   ; 设置定时器 0 工作在模式 2
ORL       TMOD,#T0M1                   ;

SETB      TR0                          ; 启动定时器 0 运行
```

C 语言代码范例:

```
TR0 = 0;

P3M0 &= 0xEF;                          //设置 P3.4(T0CKO)为推挽输出
P3M1 |= 0x10;
AUXR2 |= T0CKOE;                       //  使能 T0CKO

AUXR2 &= ~T0X12;                       //选择 SYSCLK/48 作为定时器 0 时钟输入
AUXR0 |= T0XL;

TH0 = TL0 = 0xFF;

TMOD &= 0xF0;                          //设置定时器 0 工作在模式 2
TMOD |= T0M1;

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

(3). 规定功能: SYSCLK 时钟源设置定时器 1 的时钟输出

汇编语言代码范例:

```
ORL       P3M1,#020h                   ; 设置 P3.5(T1CKO)为推挽输出
ANL       P3M0,#0DFh                   ;

ORL       AUXR2,#(T1X12|T1CKOE)        ; 选择 SYSCLK 作为定时器 1 时钟输入
                                       ; 使能 T1CKO

MOV       TH1,#0FFh                    ;
MOV       TL1,#0FFh                    ;
```

```
ANL     TMOD,#00Fh              ; 设置定时器 1 工作在模式 2
ORL     TMOD,#T1M1              ;

SETB    TR1                     ; 启动定时器 1 运行
```

C 语言代码范例:

```
P3M1 |= 0x20;                   //设置 P3.5(T1CKO)为推挽输出
P3M0 &= 0xDF;

AUXR2 |= (T1X12|T1CKOE);        //选择 SYSCLK 作为定时器 1 时钟输入
                                //  使能 T1CKO

TH1 = TL1 = 0xFF;

TMOD &= 0x0F;                   //设置定时器 1 工作在模式 2
TMOD |= T1M1;

TR1 = 1;                        //启动定时器 1 运行
```

15. 串行口(UART)

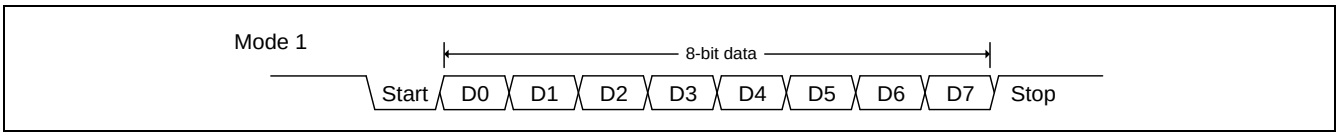
MA86E/L104 支持一个全双工的串行口，意思是可以同时发送和接收数据。它有一个接收缓冲，意味着在前一个接收到的字节没有从寄存器读出前，就可以开始接收第二个字节。但是，如果第一个字节在第二个字节接收完成前仍然没有被读出，则其中的一个字节将会丢失。串行口的接收和发送寄存器都通过特殊寄存器 SBUF 来访问。写到 SBUF0 加载到传送寄存器，当从 SBUF 读时是一个物理上独立分离的接收寄存器。

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 寄存器的 URM0X6 位选择为系统时钟频率的 1/12 或 1/2。

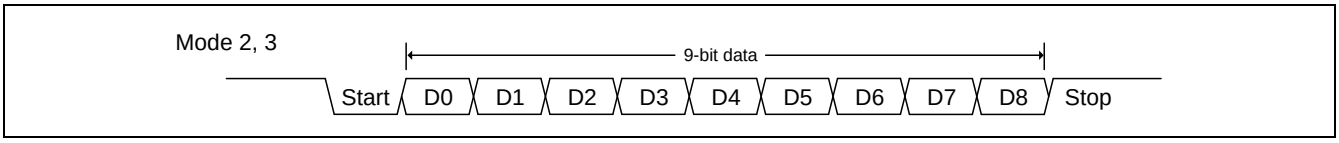
模式 1：10 位通过 TXD 传送或通过 RXD 接收，数据帧包括一个起始位(0)，8 个数据位(低位优先)，和一个停止位(1)见图 15-1。在接收时，停止位进入到专用寄存器(SCON)的 RB8。波特率是可变的。

图 15-1. 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) 见图 15-2。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。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 15-2. 模式 2, 3 数据帧



模式 3：模式 3 除了波特率可变之外其它的都相同。

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。

15.1. 串行口模式 0

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)。波特率可通过AUXR2寄存器中的URM0X6选择为系统时钟的 1/12或1/2。图 15-3 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。图 15-4 显示了模式 0 的传送时序图。

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，且在下一个时钟阶段激活接收。

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。图 15-5 显示了模式 0 的接收时序图。

图 15-3. 串行口模式 0

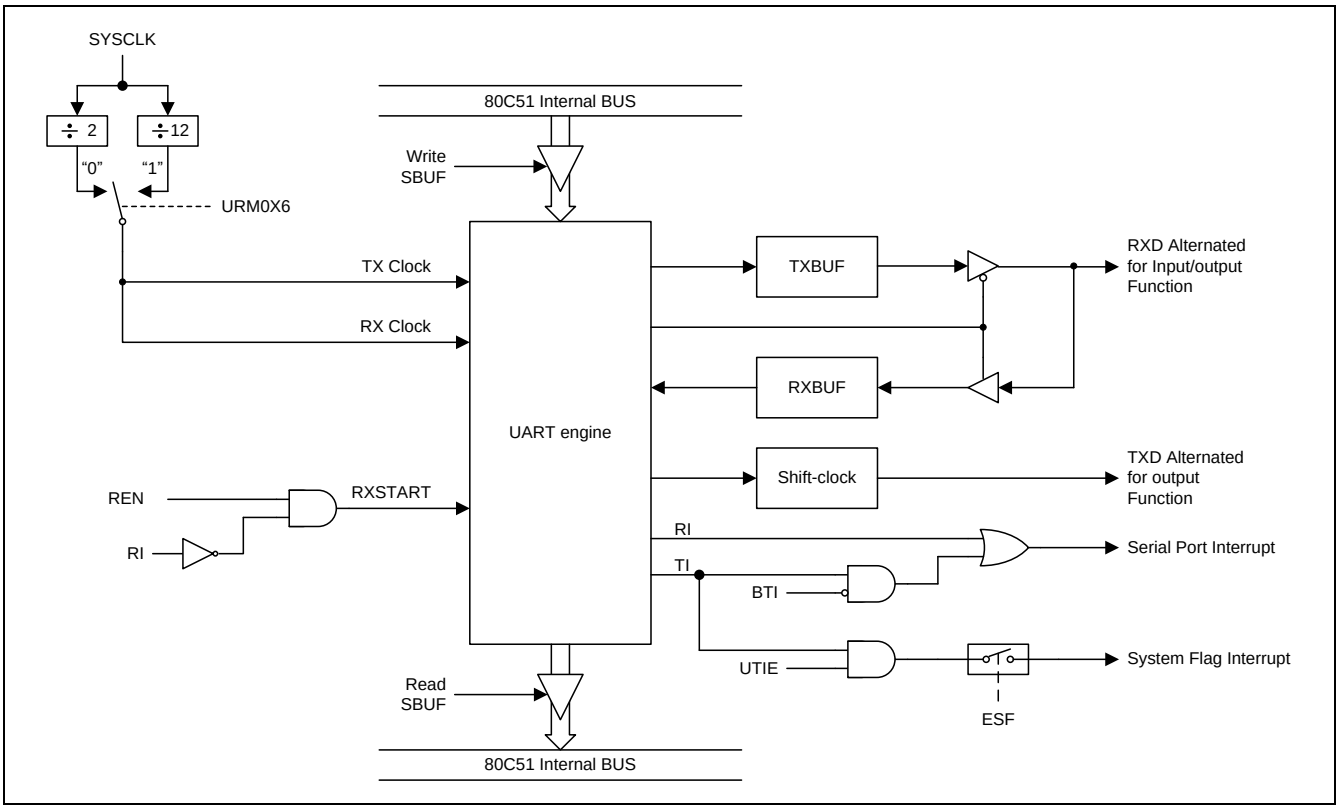


图 15-4. 模式 0 传送时序图

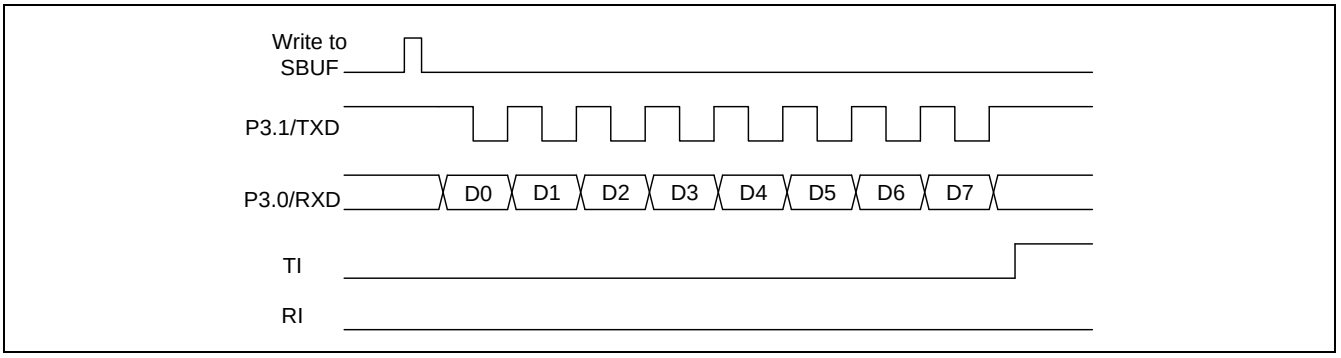
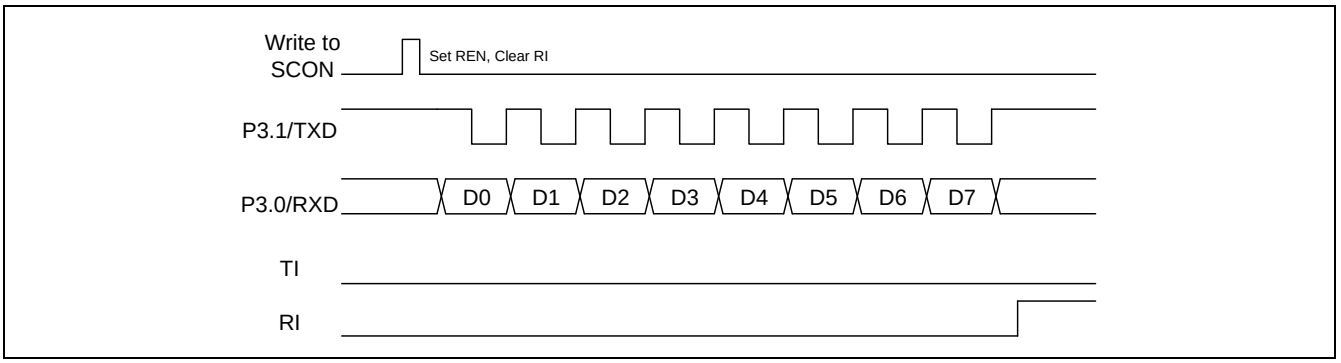


图 15-5. 模式 0 接收时序图



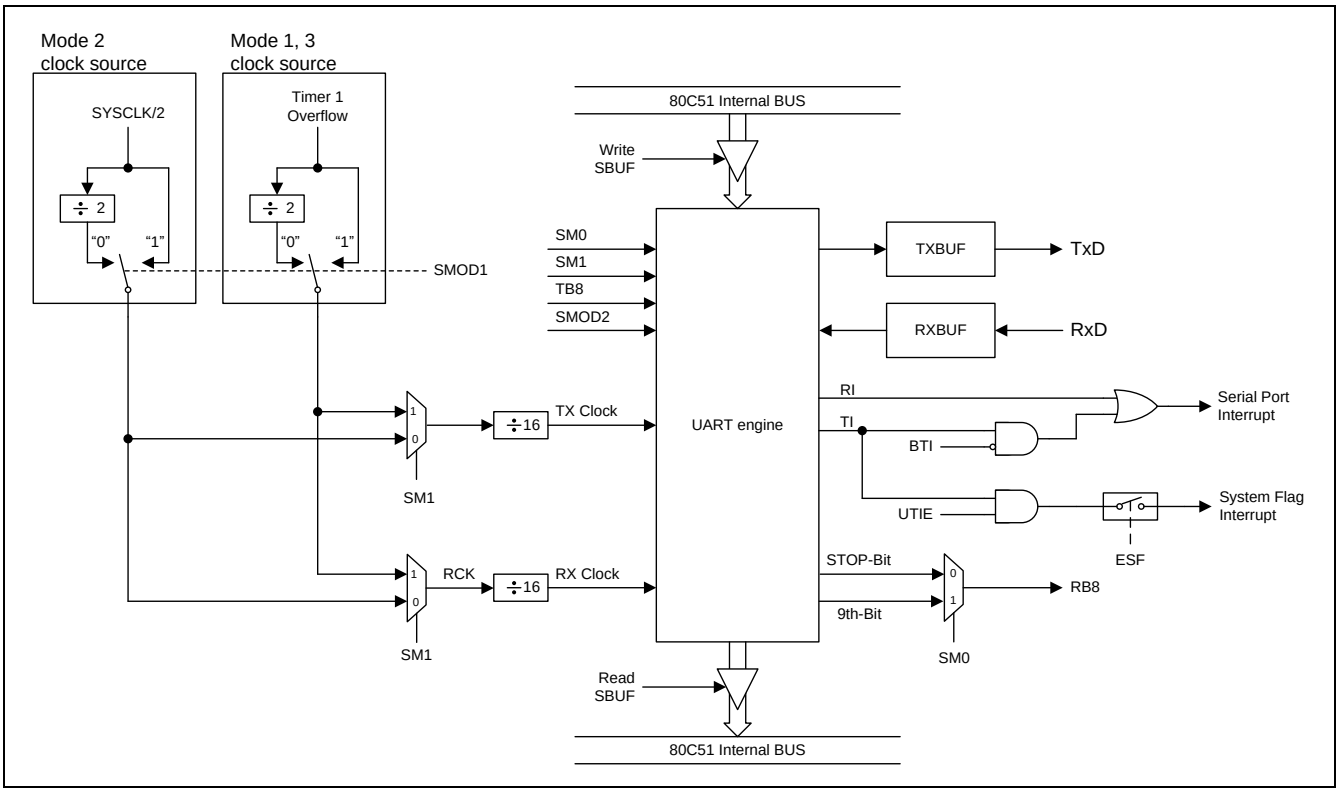
15.2. 串行口模式 1

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 的溢出速率来决定。图 15-1 显示了模式 1 的数据帧和图 15-6 显示了模式 1 的串行口简化功能图。

使用 SBUF 作为目的寄存器的任何指令来启动传输。写到 SBUF 的信号请求 UART 引擎开始发送，当收到一个发送请求后，UART 将在 TX 时钟的上升沿开始发送。SBUF 中的数据从 TXD 引脚串行输出，数据帧如图 15-1 所示及数据宽度根据 TX 时钟不同而不同。当 8 位数据发送完后，硬件将置位 TI 表示发送结束。

当串行口控制器在 RCK 采样时钟下检测到在 RXD 有 1 到 0 跳变的起始位时接收开始。在 RXD 引脚上的数据将被串行口的位侦查器采样。当收到停止位后，硬件置位 RI 表示接收结束并把停止位加载到 SCON 寄存器的 RB8。

图 15-6. 串行口模式 1, 2, 3



15.3. 串行口模式 2 和模式 3

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 传送部分不同的仅仅是传送移位寄存器的第 9 位。见图 15-2 显示模式 2 和 3 的数据帧和图 15-6 显示模式 2 和 3 的串行口简化功能图。

写到 SBUF 的信号请求 UART 引擎加载 TB8 到发送移位寄存器的第 9 位并开始发送，当收到一个发送请求后，UART 将在 TX 时钟的上升沿开始发送。SBUF 中的数据从 TXD 引脚串行输出，数据帧如图 16-2 所示及数据宽度根据 TX 时钟不同而不同。当 9 位数据发送完后，硬件将置位 TI 表示发送结束。

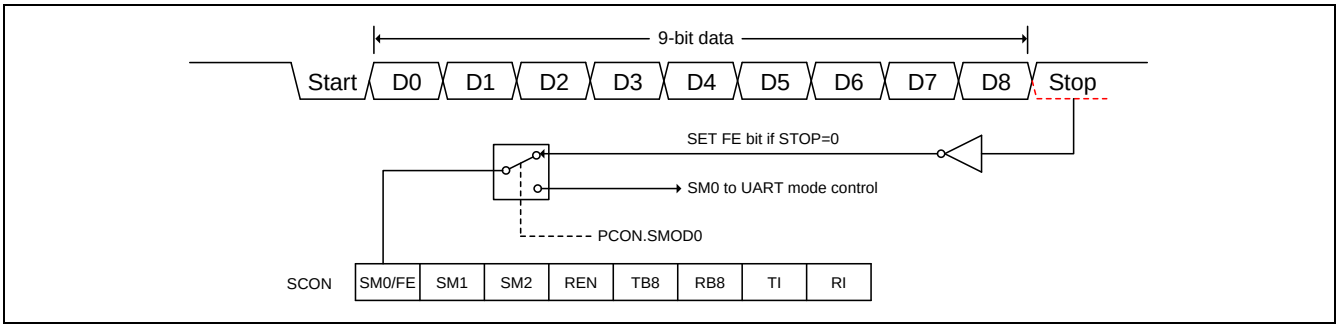
当串行口控制器在 RCK 采样时钟下检测到在 RXD 有 1 到 0 跳变的起始位时接收开始。在 RXD 引脚上的数据将被串行口的位侦查器采样。当收数据接收完后，硬件置位 RI 表示接收结束并把第 9 位加载到 SCON 寄存器的 RB8。

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，收到有 1 到 0 跳变的起始位时启动接收。

15.4. 错误帧检测

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。参考图15-7。

图 15-7. 串行口错误帧检测



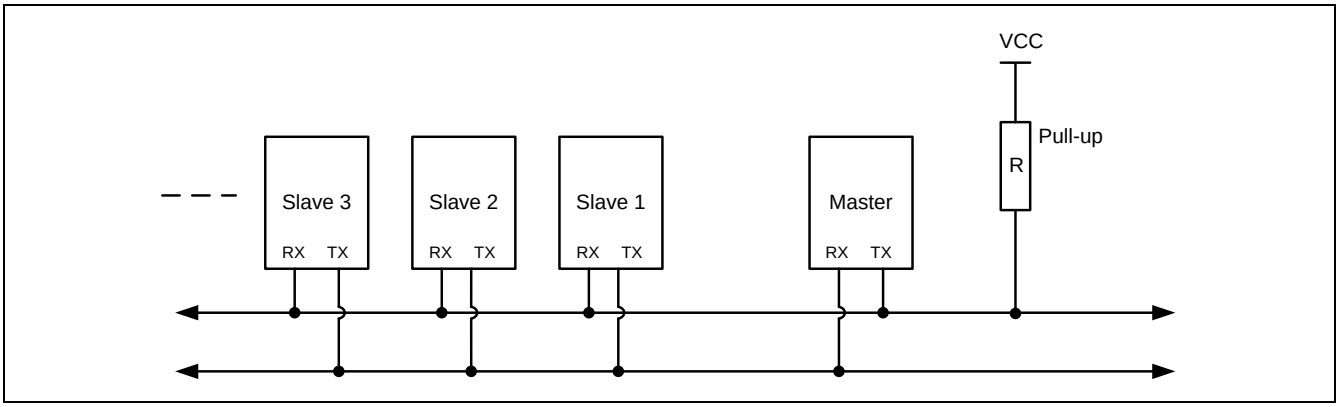
15.5. 多处理器通讯

模式 2 和 3 在用作多处理器通讯时有特殊的规定见图 15-8。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 15-8. 串行口多处理器通讯



15.6. 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 15-9。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，SM20 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
Given = 1100 00X0	Given = 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：

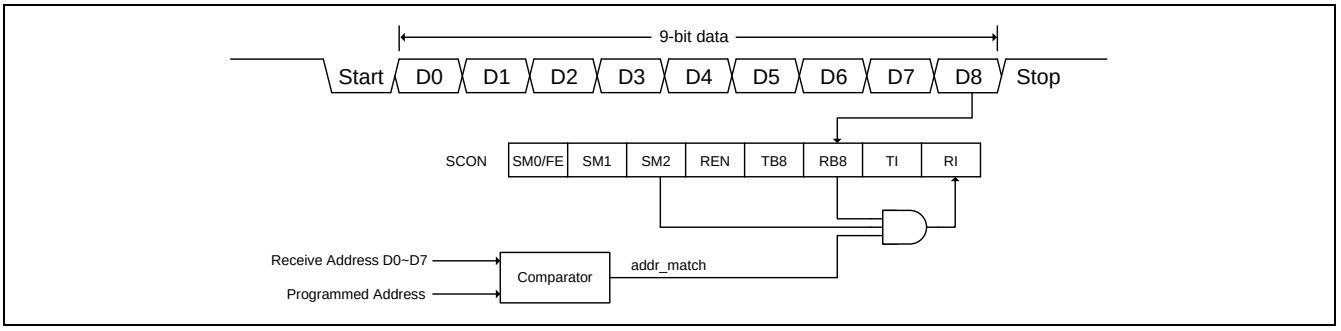
从机 0	从机 1	从机 2
SADDR = 1100 0000	SADDR = 1110 0000	SADDR = 1110 0000
SADEN = 1111 1001	SADEN = 1111 1010	SADEN = 1111 1100
Given = 1100 0XX0	Given = 1110 0X0X	Given = 1110 00XX

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 0xA9）和 SADEN（SFR 地址 0xB9）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 15-9. 自动地址识别



注意： *(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

15.7. 波特率设置

AUXR2 寄存器的位T1X12, URM0X6 和 SMOD2 提供一个新的波特率选项，见如下描述。

15.7.1. 波特率模式 0

$$\text{Mode 0 Baud Rate} = \frac{F_{SYSCLK}}{n}$$; n=12, if URM0X6=0 ; n=2, if URM0X6=1

注意:

如果 URM0X6=0, 则波特率公式与标准 8051 一样。

15.7.2. 波特率模式 2

$$\text{Mode 2 Baud Rate} = \frac{2^{SMOD1} \times 2^{(SMOD2 \times 2)}}{64} \times (F_{SYSCLK})$$

注意:

如果 SMOD2=0, 则波特率公式与标准 8051 一样。如果 SMOD2=1，这是一个增强型的波特率设置功能。在波特率模式 2 产生器里由 SMOD2 决定的波特率见下表。

表 15-1. SMOD2 应用条件在模式 Mode 2

SMOD2	SMOD1	波特率	注意	建议最大接收错误率 (%)
0	0	缺省波特率	标准功能	± 3%
0	1	双倍波特率	标准功能	± 3%
1	0	双倍波特率 **X2**	增强型功能	± 2%
1	1	双倍波特率 **X4**	增强型功能	± 1%

15.7.3. 波特率模式 1 和 3

使用定时器 1 作为波特率产生器

$$\text{Mode 1, 3 Baud Rate} = \frac{2^{SMOD1} \times 2^{(SMOD2 \times 2)}}{32} \times \frac{F_{SYSCLK}}{12 \times (256 - TH1)}$$; T1X12=0

$$\text{or} = \frac{2^{SMOD1} \times 2^{(SMOD2 \times 2)}}{32} \times \frac{F_{SYSCLK}}{1 \times (256 - TH1)}$$; T1X12=1

注意:

如果 SMOD2=0 和 T1X12=0, 则波特率公式与标准 8051 一样。如果 SMOD2=1，这是一个增强型的波特率设置功能。。在时钟 1 产生器里由 SMOD2 决定的波特率见下表。

表 15-2. SMOD2 应用条件在模式 1 & 3 使用定时器 1

SMOD2	SMOD1	波特率	注意	建议最大接收错误率 (%)
0	0	缺省波特率	标准功能	± 3%
0	1	双倍波特率	标准功能	± 3%
1	0	双倍波特率 **X2**	增强型功能	± 2%
1	1	双倍波特率 **X4**	增强型功能	± 1%

表 15-3 ~ 表 15-10 列出从 Timer 1 的 8 位自动加载模式中产生的各种常用的波特率

表 15-3. Timer 1 产生常用的波特率 @ F_{SYSCLK}=11.0592MHz

波特率	**TH1**, 重裁值					
	T1X12=0 & SMOD2=0			**T1X12=1 & SMOD2=0**		
	SMOD1=0	SMOD1=1	Error	SMOD1=0	SMOD1=1	Error
1200	232	208	0.0%	--	--	--
2400	244	232	0.0%	112	--	0.0%
4800	250	244	0.0%	184	112	0.0%
9600	253	250	0.0%	220	184	0.0%
14400	254	252	0.0%	232	208	0.0%
19200	--	253	0.0%	238	220	0.0%
28800	255	254	0.0%	244	232	0.0%
38400	--	--	--	247	238	0.0%
57600	--	255	0.0%	250	244	0.0%
115200	--	--	--	253	250	0.0%
230400	--	--	--	--	253	0.0%

表 15-4. Timer 1 产生高波特率 @ F_{SYSCLK} =11.0592MHz

波特率	**TH1**, 重载值					
	T1X12=0 & SMOD2=1			**T1X12=1 & SMOD2=1**		
	SMOD1=0	SMOD1=1	Error	SMOD1=0	SMOD1=1	Error
230.4K	--	255	0.0%	250	244	0.0%
460.8K	--	--	--	253	250	0.0%
691.2K	--	--	--	254	252	0.0%
921.6K	--	--	--	--	253	0.0%
1.3824M	--	--	--	255	254	0.0%
2.7648M	--	--	--	--	255	0.0%

表 15-5. Timer 1 产生常用的波特率 @ F_{SYSCLK}=22.1184MHz

波特率	**TH1**, 重载值					
	T1X12=0 & SMOD2=0			**T1X12=1 & SMOD2=0**		
	SMOD1=0	SMOD1=1	Error	SMOD1=0	SMOD1=1	Error
1200	208	160	0.0%	--	--	--
2400	232	208	0.0%	--	--	0.0%

4800	244	232	0.0%	112	--	0.0%
9600	250	244	0.0%	184	112	0.0%
14400	252	248	0.0%	208	160	0.0%
19200	253	250	0.0%	220	184	0.0%
28800	254	252	0.0%	232	208	0.0%
38400	--	253	0.0%	238	220	0.0%
57600	255	254	0.0%	244	232	0.0%
115200	--	255	0.0%	250	244	0.0%
230400	--	--	--	253	250	0.0%
460800	--	--	--	--	253	0.0%

表 15-6. Timer 1 产生高波特率 @ F_{SYSCLK}=22.1184MHz

波特率	**TH1**, 重载值					
	T1X12=0 & SMOD2=1			**T1X12=1 & SMOD2=1**		
	SMOD1=0	SMOD1=1	Error	SMOD1=0	SMOD1=1	Error
460.8K	--	255	0.0%	250	244	0.0%
691.2K	--	--	--	252	248	0.0%
921.6K	--	--	--	253	250	0.0%
1.3824M	--	--	--	254	252	0.0%
1.8432M				--	253	0.0%
2.7648M	--	--	--	255	254	0.0%
5.5296M	--	--	--	--	255	0.0%

表 15-7. Timer 1 产生常用波特率 @ F_{SYSCLK}=12.0MHz

波特率	**TH1**, 重载值					
	T1X12=0 & SMOD2=0			**T1X12=1 & SMOD2=0**		
	SMOD1=0	SMOD1=1	Error	SMOD1=0	SMOD1=1	Error
1200	230	204	0.16%	--	--	--
2400	243	230	0.16%	100	--	0.16%
4800	--	243	0.16%	178	100	0.16%
9600	--	--	--	217	178	0.16%
14400	--	--	--	230	204	0.16%
19200	--	--	--	--	217	0.16%
28800	--	--	--	243	230	0.16%
38400	--	--	--	246	236	2.34%
57600	--	--	--	--	243	0.16%
115200	--	--	--	--	--	--

表 15-8. Timer 1 产生高波特率 @ F_{SYSCLK}=12.0MHz

波特率	**TH1**, 重载值					
	T1X12=0 & SMOD2=1			**T1X12=1 & SMOD2=1**		
	SMOD1=0	SMOD1=1	Error	SMOD1=0	SMOD1=1	Error
115.2K	--	--	--	243	230	0.16%

230.4K	--	--	--	--	243	0.16%
460.8K	--	--	--	--	--	--

表 15-9. Timer 1 产生高波特率 @ F_{SYSCLK}=24.0MHz

波特率	**TH1**, 重载值					
	T1X12=0 & SMOD2=0			**T1X12=1 & SMOD2=0**		
	SMOD1=0	SMOD1=1	Error	SMOD1=0	SMOD1=1	Error
1200	204	152	0.16%	--	--	--
2400	230	204	0.16%	--	--	--
4800	243	230	0.16%	100	--	0.16%
9600	--	243	0.16%	178	100	0.16%
14400	--	--	--	204	152	0.16%
19200	--	--	--	217	178	0.16%
28800	--	--	--	230	204	0.16%
38400	--	--	--	--	217	0.16%
57600	--	--	--	243	230	0.16%
115200	--	--	--	--	243	0.16%

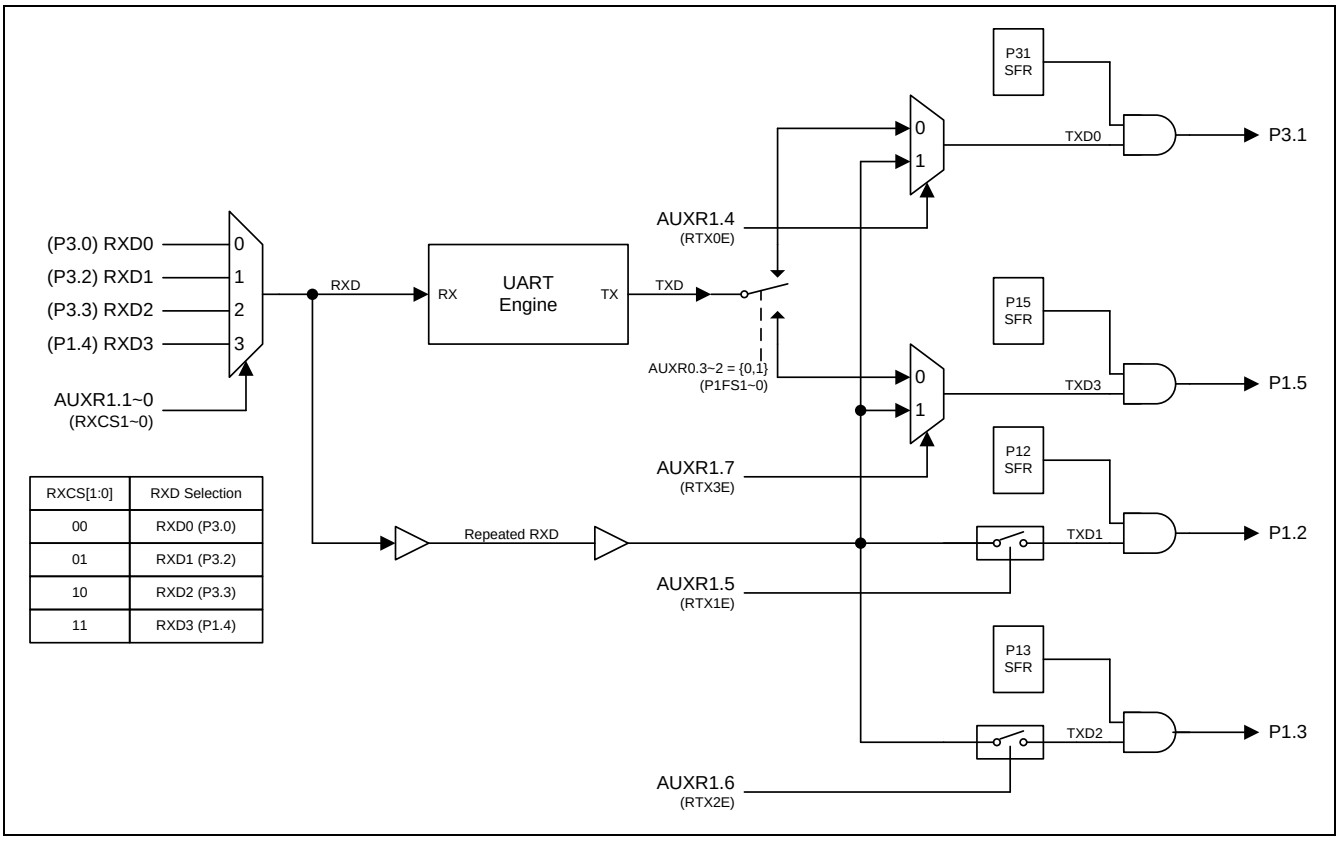
表 15-10. Timer 1 产生高波特率 @ F_{SYSCLK}=24.0MHz

波特率	**TH1**, 重载值					
	T1X12=0 & SMOD2=1			**T1X12=1 & SMOD2=1**		
	SMOD1=0	SMOD1=1	Error	SMOD1=0	SMOD1=1	Error
230.4K	--	--	--	243	230	0.16%
460.8K	--	--	--	--	243	0.16%
691.2K	--	--	--	--	--	--
921.6K	--	--	--	--	--	--

15.8. 串行口重复模式

图 15-10 显示了 MA86E/L104 的串行口重复工作。软件可以配置从 RXD 到 TXD 的数据流通道。它可以 4 选 1RXD 输入到 UART 引擎，同时将数据直接路由到 4 个 TXD 输出。这个重复功能仅支持串口模式 1,2 和 3。

图 15-10. Serial Port Repeater



AUXR1: 辅助寄存器 1

SFR 地址 = 0xA2　　复位初始值 = 0000-0000

7	6	5	4	3	2	1	0
RTX3E	RTX2E	RTX1E	RTX0E	--	--	RXCS1	RXCS0
R/W	**R/W**	**R/W**	**R/W**	**W**	**W**	**R/W**	**R/W**

15.9. 串行口寄存器

串行口的四种操作模式除波特率的设定之外都与标准的 8051 相同。此三个寄存器 PCON, AUXR 和 AUXR2 是与波特率的设定有关。

SCON: 串行口控制寄存器

SFR 地址 = 0x98　　复位初始值 = 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。SMOD0 位必须设置才能访问帧错误位（FE）。
0: 帧错误位（FE）不会被有效帧清零但是可以软件清零。
1: 当非法的停止位出现时接收端会把帧错误位（FE）设置。

Bit 7: 串行口模式位 0， (SMOD0 位必须为零才能作为 SM0 访问)
Bit 6: 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 SYSCLK /2
0	1	1	8 位通用串行口	可变的
1	0	2	9 位通用串行口	SYSCLK/64, /32, /16 或 /8
1	1	3	9 位通用串行口	可变的

Bit 5: 串行口模式位 2。
0: 禁止 SM2 功能。
1: 在模式 2 和 3 时使能地址自动识别，如果 SM20=1 那么 RI0 将不能设置，除非接收到的第 9 位数据(RB80)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 SM20=1 那么 RI0 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，SM20 可以为 0。
Bit 4: REN, 使能串行接收。
0: 软件清零是禁止串行接收。
1: 软件设置是使能串行接收。

Bit 3: TB8, 在模式 2 和 3 下此位是被传送数据的第 9 位。由软件清零或设置。

Bit 2: RB8, 在模式 2 和 3 下此位是接收到数据的第 9 位。在模式 1 下，如果 SM2 = 0, RB8 则是接收到的停止位。在模式 0，RB8 没有用到。

Bit 1: TI. 传送中断标志位。
0: 必须由软件清零。
1: 在模式 0 下第 8 位传送时间结束硬件设置，在其它模式下传送停止位开始硬件设置。

Bit 0: RI. 接收中断标志位。
0: 必须由软件清零。
1: 在模式 0 下第 8 位接收时间结束硬件设置，在其它模式下接收停止位一半时开始硬件设置（除了 SM2 之外）。

SBUF: 串行口缓冲寄存器

SFR 地址 = 0x99　　复位初始值= 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: 作为传送和接收的缓冲寄存器。

SADDR: 从地址寄存器

SFR 地址 = 0xA9　　复位初始值= 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　　复位初始值= 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 组合来预置地址，事实上，SADEN 是 SADDR 的“屏蔽”寄存器，如下图所示

SADDR = 1100 0000
SADEN = 1111 1101
Given = 1100 00x0 ⟶ 这“Given”从机地址将被选中
Bit1 作“不关心”处理

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　　上电复位初始值 = 00X1-0000, 复位初始值= 00X0-0000

7	6	5	4	3	2	1	0
SMOD1	SMOD0	GF	POF	GF1	GF0	PD	IDL
R/W	R/W	R/W	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。注：当帧错误后不管 SMOD0 什么状态 FE 都将置 1。

AUXR1: 辅助寄存器 1

SFR 地址 = 0xA2　　RESET = 0000-0000

7	6	5	4	3	2	1	0
RTX3E	RTX2E	RTX1E	RTX0E	--	--	RXCS1	RXCS0
R/W	**R/W**	**R/W**	**R/W**	**W**	**W**	**R/W**	**R/W**

Bit 7: RTX3E, 使能 RXD 重复到 TXD3
0: 禁止 TXD3(P1.5)的 RTX3E 功能
1: 使能 TXD3(P1.5)的 RXD 重复机制

Bit 6: RTX2E, 使能 RXD 重复到 TXD2
0: 禁止 TXD2(P1.3)的 RTX2E 功能
1: 使能 TXD2(P1.3)的 RXD 重复机制

Bit 5: RTX1E, 使能 RXD 重复到 TXD1
0: 禁止 TXD1(P1.2)的 RTX1E 功能
1: 使能 TXD1(P1.2)的 RXD 重复机制

Bit 4: RTX0E, 使能 RXD 重复到 TXD0
0: 禁止 TXD0(P3.1)的 RTX0E 功能
1: 使能 TXD0(P3.1)的 RXD 重复机制

Bit 3~2: 保留，写 AUXR1 时，在这些位上必须写"0"。
Bit 1~0: RXD 通道选择.

RXCS[1:0]	RXD channel selection
00	P3.0
01	P3.2
10	P3.3
11	P1.4

AUXR2: 辅助寄存器 2

SFR 地址 = 0xA3　　复位初始值= 0000-0000

7	6	5	4	3	2	1	0
URXR	BTI	URM0X6	SMOD2	T1X12	T0X12	T1CKOE	T0CKOE
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: URXR, 串行口 RX 选项.
0: 清零是禁止此选项。
1: 设置是使能此选项。

Bit 6: BTI, 阻止 TI 在串行口中断。
0: 保留 TI 作为串行口中断源。
1: 阻止 TI 作为串行口中断源。

Bit 5: URM0X6, 串行口模式 0 波特率选择。
0: 清零选择 SYSCLK/12 作为串行口模式 0 波特率。
1: 设置选择 SYSCLK/2 作为串行口模式 0 波特率。

Bit 4: SMOD2, 额外 X2/X4 波特率选择。
0: 禁止额外 X2/X4 波特率选择。
1: 使能额外 X2/X4 波特率选择。

Bit 3: T1X12, 当 C/T=0 时，定时器 1 时钟源选择。
0: 清零选择 SYSCLK/12。

1: 设置选择 SYSCLK 作时钟源。

SFIE: 系统标志中断使能寄存器

SFR 地址 = 0x8E　　复位初始值= 0xxx-0x00

7	6	5	4	3	2	1	0
UTIE	--	--	--	KBIFIE	--	BOF0IE	WDTFIE
R/W	**W**	**W**	**W**	**R/W**	**W**	**R/W**	**R/W**

Bit 7: UTIE, UART TI 系统标志中断标志。
0: 禁止中断向量共享 TI 系统标志中断。
1: 使能中断向量共享 TI 系统标志中断。

15.10. 串行口示例代码

(1). 规定功能:串行口输入 RI 唤醒空闲模式

汇编语言代码范例:

```
    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:
;   中断处理
    CLR     RI                          ;清除 RI 标志
    RETI                                ;中断返回

TI_ISR:
;   中断处理
    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

    MOV     IP0L,#PSL                   ; 选择串行口 S0 中断优先级
    MOV     IP0H,#PSH                   ;

    SETB    ES                          ; 使能串行口 S0 中断
    SETB    EA                          ; 使能全局中断

    ORL     PCON0,#IDL;                 ; 设置 MCU 进入空闲模式
```

C 语言代码范例:

```
void uart_ri_idle_isr(void) interrupt 4
{    if(RI)                                  //判断是否串行输入中断
     {
          RI=0;                              //清除 RI 标志
          // to do ...
     }

     if(TI)                                  //判断是否串行发送中断
     {
          TI=0;                              //清除 TI 标志
          // to do ...
     }
}

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

     IP0L = PSL;                             //选择串行口 S0 中断优先级
     IP0H = PSH;                             //

     ES = 1;                                 //使能串行口 S0 中断
     EA = 1;                                 //  使能全局中断

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

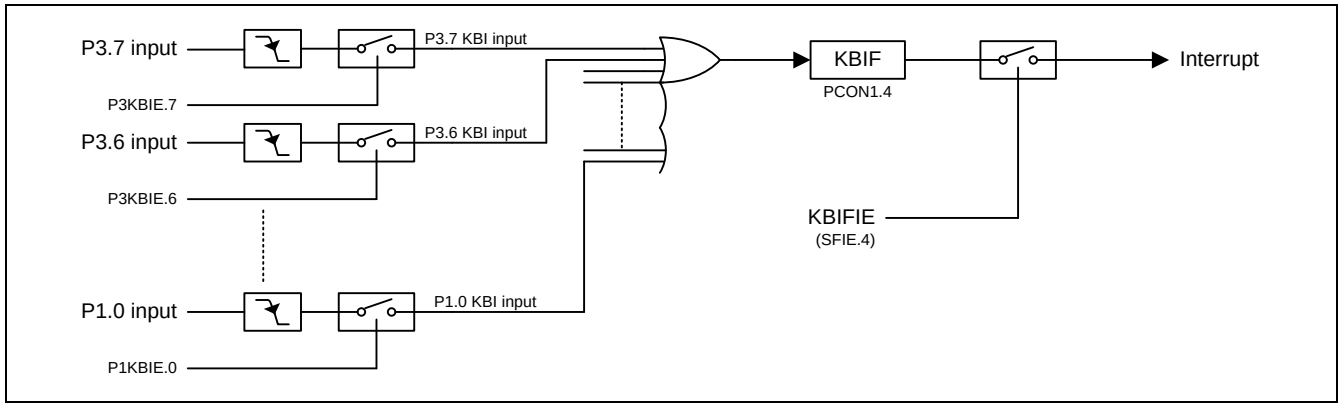
16. 键盘中断(KBI)

键盘中断功能是当键盘使能的脚出现下降沿时产生一个简单中断。此功能用作键盘识别。图 16-1 列出了键盘中断功能的结构。

此功能有两个特殊功能寄存器，P1KBIE 和 P3KBIE 的每一位控制相应的脚位在下降沿出现时使能或禁止键盘中断(KBI)。任意一个被识别的键盘中断（KBI）事件将会促使硬件设置中断标志（KBIF）如果相应的中断是使能的话则产生中断。不需要使能（KBIFIE）总中断，仅使能键盘中断（KBI）脚就可以在下降沿唤醒 CPU 的空闲模式或低电平唤醒 CPU 的掉电模式。

16.1. 键盘中断结构图

图 16-1 键盘中断结构图



16.2. 键盘中断寄存器

P1KBIE: 端口 1 键盘中断(KBI)使能控制寄存器

SFR 地址 = 0xD7　　复位初始值 = 0000-0000

7	6	5	4	3	2	1	0
P17KBIE	P16KBIE	P15KBIE	P14KBIE	P13KBIE	P12KBIE	P11KBIE	P10KBIE
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~0: 端口 1 每个脚键盘中断使能位。
0: 禁止相应的脚产生键盘中断功能。
1: 使能相应的脚产生键盘中断功能。

P3KBIE: 端口 3 键盘中断(KBI)使能控制寄存器

SFR 地址 = 0xD6　　复位初始值= 0000-0000

7	6	5	4	3	2	1	0
P37KBIE	P36KBIE	P35KBIE	P34KBIE	**P41KBIE**	**P40KBIE**	P31KBIE	P30KBIE
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~0: 端口 P3.7 ~ P3.4, P4.1, P4.0, P3.1 和 P3.0 键盘中断使能位。
0: 禁止相应的脚产生键盘中断功能。
1: 使能相应的脚产生键盘中断功能。

PCON1:电源控制寄存器 1

SFR 地址 = 0x97　　上电复位初始值 = 0010-0X00

7	6	5	4	3	2	1	0
SWRF	EXRF	--	--	**KBIF**	--	BOF0	WDTF
R/W	**R/W**	**W**	**W**	**R/W**	**W**	**R/W**	**R/W**

Bit 3: KBIF, 键盘中断标志

0: 此位必须通过软件写”1”来清零，软件写”0”是空操作。

1: 此位仅仅在使能的键盘中断(KBI)脚出现下降沿时设置。软件写”1”来清除 KBIF。在掉电模式下，在使能的键盘中断(KBI)脚出现低电平时设置。

16.3. 键盘中断示例代码

(1). 规定功能: P1.3~P1.0 口执行键盘中断 KBI 功能

汇编语言代码范例:

```
    ORG     0003Bh
SystemFlag_ISR:                         ;
    PUSH    ACC                         ; ACC 压栈

    MOV     A,PCON1                     ;
    JNB     ACC.3,Not_KBIF_ISR          ;是否键盘中断 KBI
    To do……
    ANL     PCON1,#KBIF                 ; 清除键盘中断 KBI 标志(写 “1”)

Not_KBIF_ISR:
    POP     ACC                         ; ACC 退栈
    RETI                                ;中断返回

main:
    ORL     PUCON0,#PU10                ; 使能 P1.3~P1.0 片内上拉电阻
    ANL     P1M0,#0F0h                  ; 设置 P1.3~P1.0 为漏极开路输出
    ORL     P1,#00Fh                    ; 设置 P1.3~P1.0 为输入模式

    ORL     EIP1L,#PSFL                 ; 选择系统标志中断优先级
    ORL     EIP1H,#PSFH                 ;

    MOV     P1KBIE,#00Fh                ; 使能 P1.3~P1.0 键盘中断 KBI 功能
;   MOV     P3KBIE,#0xF3                ; 使能 P3 键盘中断 KBI 功能
;   MOV     P3KBIE,#0x0C                ; 使能 P4.1~0 键盘中断 KBI 功能

    ANL     PCON1,#KBIF                 ; 清除键盘中断 KBI 标志(写 “1”)

    ORL     SFIE,#KBIFIE                ; 使能 KBI 中断
    ORL     EIE1,#ESF                   ; 使能系统标志中断
    SETB    EA                          ; 使能全局中断
```

```
    ORL     PCON0,#PD                   ; 设置 MCU 进入掉电模式
```

C 语言代码范例:

```
void SystemFlag_ISR(void) interrupt 7
{   if(PCON1 & KBIF)
    {
        PCON1 &= KBIF;                  //清除键盘中断 KBI 标志(写 “1”)
    }
}

void main(void)
{
    PUCON0 |= PU10;                     //使能 P1.3~P1.0 片内上拉电阻
    P1M0 &= 0xF0;                       //设置 P1.3~P1.0 为漏极开路输出
    P1 |= 0x0F;                         //设置 P1.3~P1.0 为输入模式

    EIP1L |= PSFL;                      //选择系统标志中断优先级
    EIP1H |= PSFH;

    P1KBIE = 0x0F;                      //使能 P1.3~P1.0 键盘中断 KBI 功能
//  P3KBIE = 0xF3;                      //使能 P3 键盘中断 KBI 功能
//  P3KBIE = 0x0C;                      //使能 P4.1~0 键盘中断 KBI 功能

    PCON1&= KBIF;                       //清除键盘中断 KBI 标志(写 “1”)

    SFIE |= KBIFIE;                     //使能 KBI 中断
    EIE1 |= ESF;                        //使能系统标志中断
    EA = 1;                             //  使能全局中断

    PCON0 |= PD;                        //设置 MCU 进入掉电模式
}
```

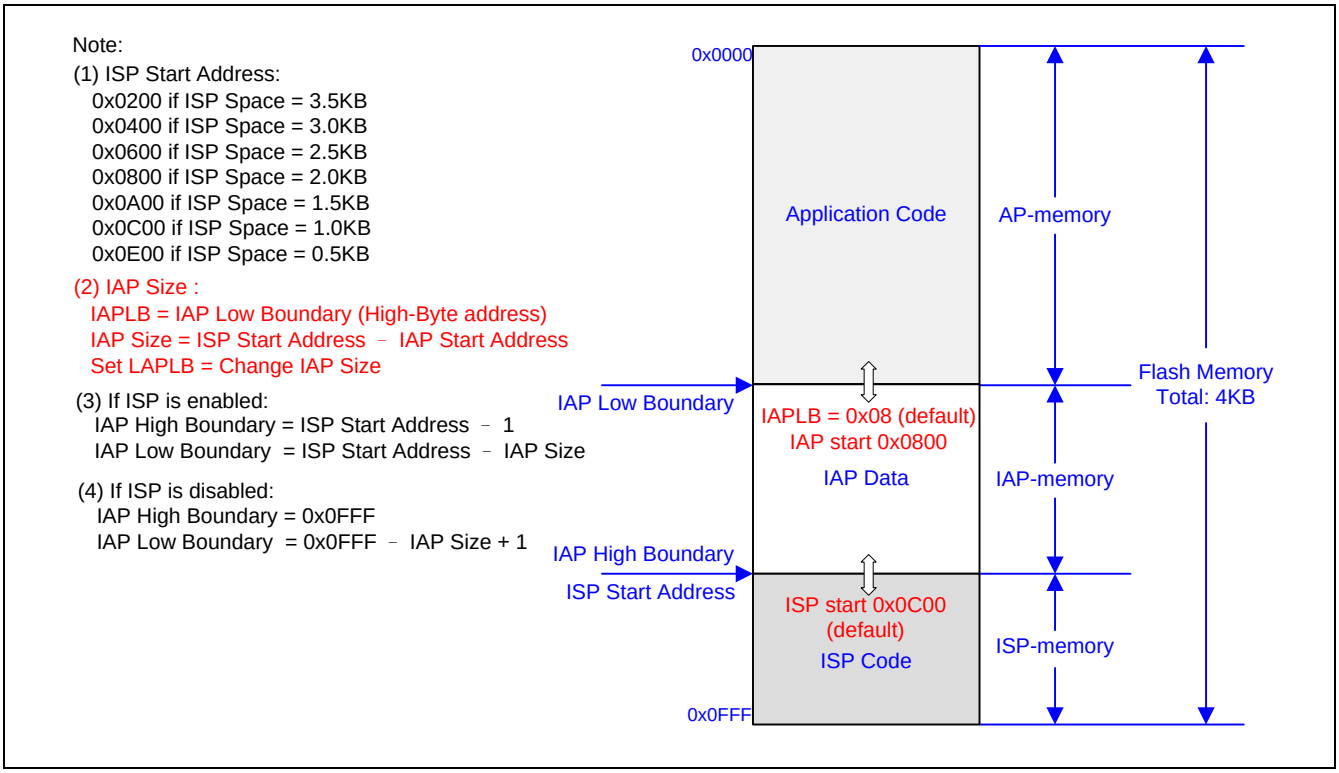
17. ISP and IAP

MA86E/L104 的 FLASH 存储分区为 AP-存储区，IAP-存储区和 ISP-存储区。AP 存储区用来存储用户的应用程序；IAP-存储区用来存储非易失性数据；ISP-存储区存储用于在系统编程的根引导程序。当 MCU 在 ISP 区运行是，MCU 可以修改 AP 和 IAP 存储，用于程序更新。如果 MCU 在 AP 区运行，软件仅能修改 IAP 存储的数据。

17.1. MA86E/L104 闪存存储配置

MA86E/L104 总共有 4K 字节的 Flash， 图 17-1 显示了 Flash 的配置。ISP-存储可以由硬件选项配置为禁止或最大到 3.5K 字节。IAP 空间定位在 IAP 低边界到 IAP 高边界之间。IAP 低边界由 IAPLB 寄存器的值决定。IAP 高边界与 ISP 的起始地址有关，它取决于硬件选项定义的 ISP 空间大小。IAPLB 寄存器可以由硬件选项或软件编程去配置。所有的 AP,IAP,ISP 空间都共享总的 4KFlash 空间。

图 17-1. MA86E/L104 闪存存储配置



注意:

笙泉公司样品的默认设置是：1K ISP, 1K IAP 和加密。1K ISP 文件是有笙泉专利的通过一条线就能在线下载的 1-线 ISP 协议。1KIAP 大小可以有软件根据应用需要重新配置

17.2. MA86E/L104 在 ISP/IAP 访问 Flash

对于 ISP 和 IAP 应用，MA86E/L104 提供了两种 Flash 访问模式。编程模式和读模式。MCU 软件使用这两种模式去更新新的数据到 Flash 和获取 Flash 里的内容。这段列出了用于各种 Flash 模式的流程图和示例代码。

17.2.1. ISP/IAP Flash 编程模式

MA86E/L104 的“编程”模式为向 Flash 存储更新新的数据提供了字节写操作。IFADRH 和 IFADRL 指向物理 Flash 字节地址。IFD 存储了需要更新到 Flash 去的内容。图 17-2 列出了在 ISP/IAP 操作中的 Flash 字节编程流程图

图 17-2. ISP/IAP 字节编程流程

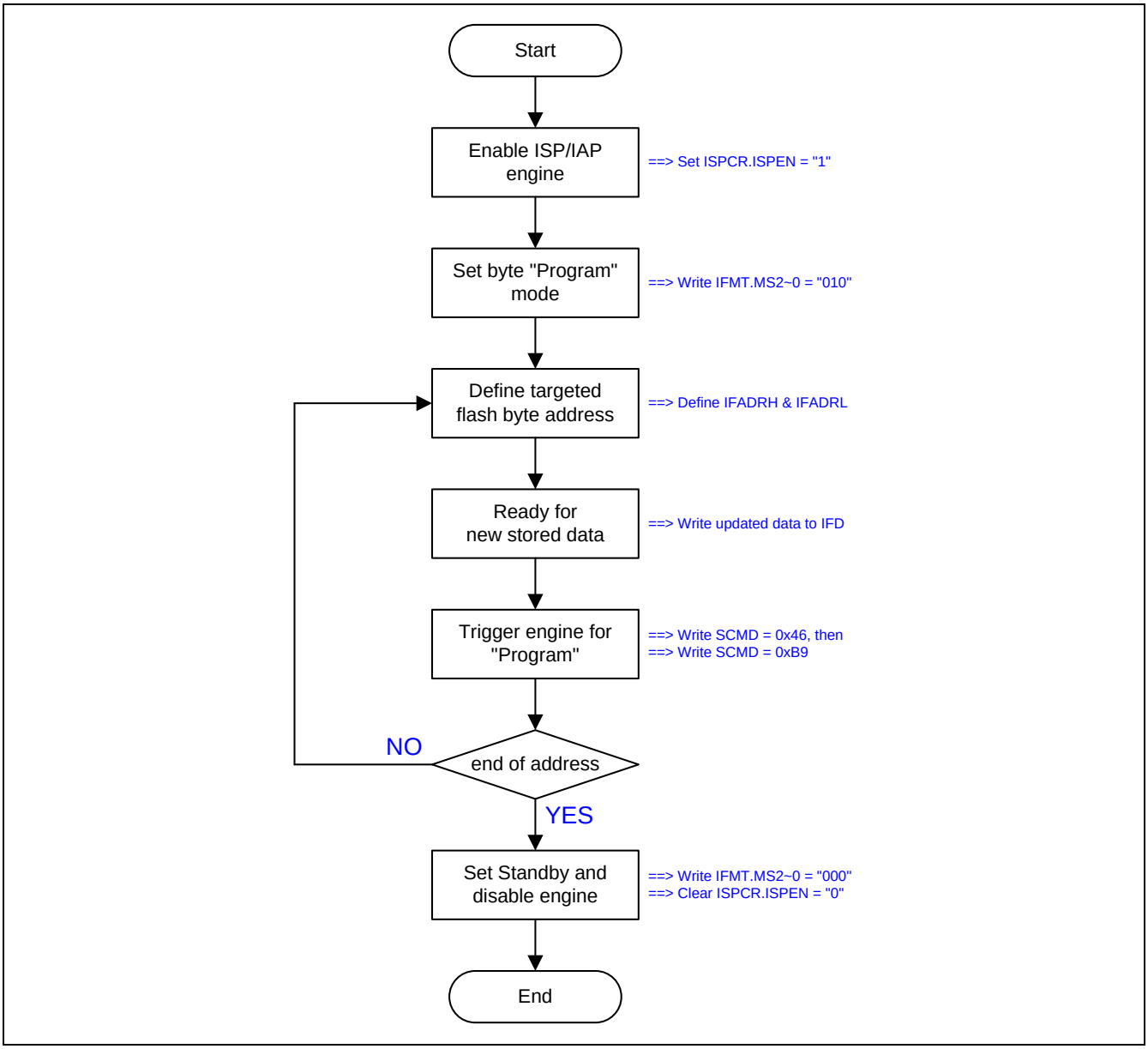


图 17-3 列出了 ISP/IAP 字节编程的示例代码

图 17-3. ISP/IAP 字节编程示例代码

```
MOV     ISPCR,#10000011b ; ISPCR.7=1, enable ISP

MOV     IFMT,#02h         ; select Program Mode

MOV     IFADRH,??         ; fill [IFADRH,IFADRL] with byte address
MOV     IFADRL,??         ;

MOV     IFD,??            ; fill IFD with the data to be programmed

MOV     SCMD,#46h         ;trigger ISP/IAP processing
MOV     SCMD,#0B9h        ;

;Now, MCU will halt here until processing completed

MOV     IFMT,#00h         ; select Standby Mode
MOV     ISPCR,#00000000b ; ISPCR.7 = 0, disable ISP
```

17.2.2. ISP/IAP Flash 读模式

MA86E/L104 的“读”模式提供了字节读操作从 Flash 里获取存储的数据。IFADRH 和 IFADRL 指向物理 Flash 字节地址。IFD 存储从 Flash 读到的数据。 建议在数据编程或页擦除后用读模式去校对 Flash 里的数据。
图 17-4 列出了 ISP/IAP 操作的 Flash 字节读流程

图 17-4 ISP/IAP 字节读流程

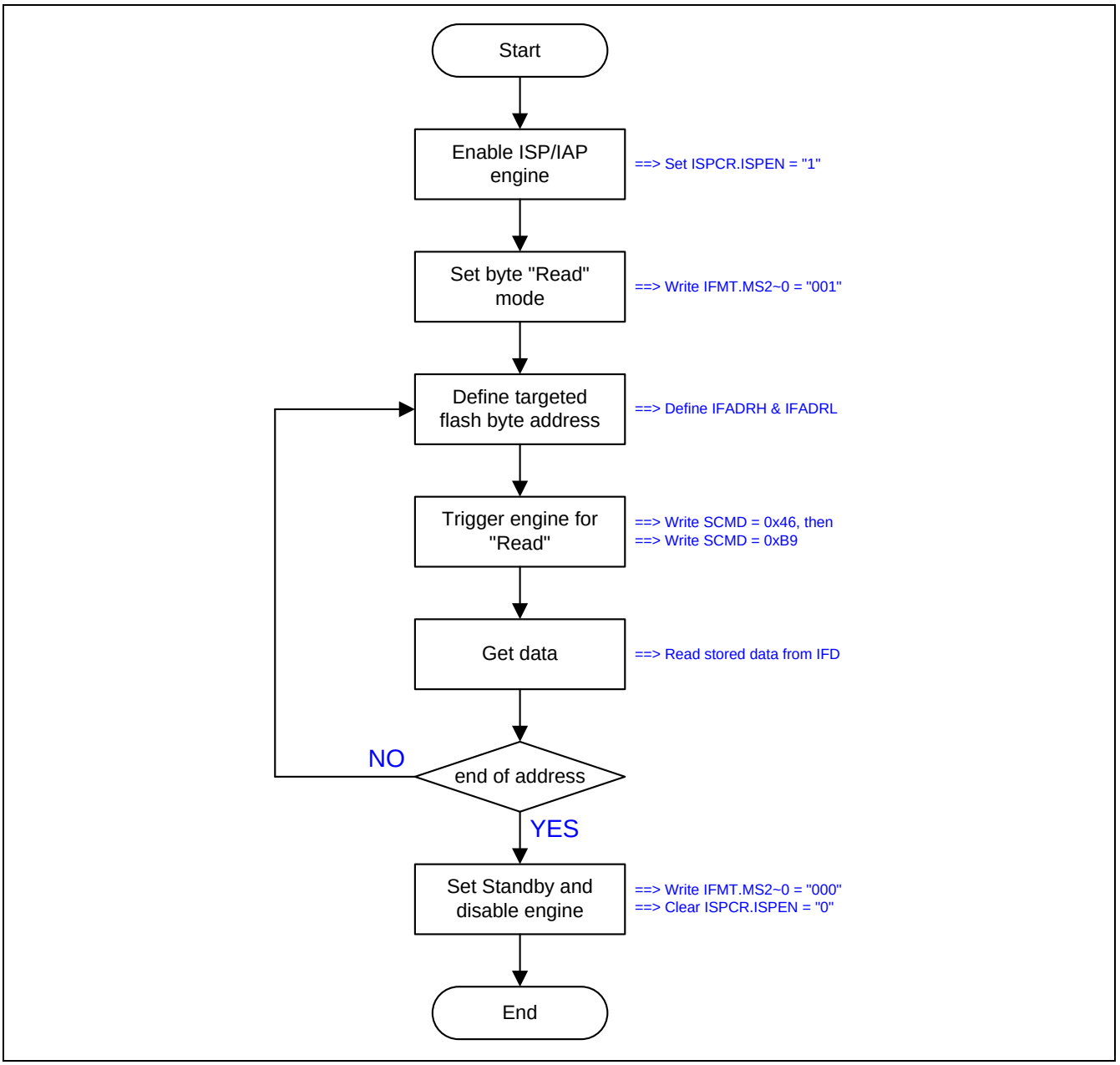


图 17-5 列出了 ISP/IAP 字节读操作的示例代码

图 17-5. ISP/IAP 字节读示例代码

```
MOV     ISPCR,#10000011b ; ISPCR.7=1, enable ISP

MOV     IFMT,#01h        ; select Read Mode

MOV     IFADRH,??        ; fill [IFADRH,IFADRL] with byte address
MOV     IFADRL,??        ;

MOV     SCMD,#46h        ; trigger ISP/IAP processing
MOV     SCMD,#0B9h       ;

;Now, MCU will halt here until processing completed

MOV     A,IFD            ; now, the read data exists in IFD

MOV     IFMT,#00h        ; select Standby Mode
MOV     ISPCR,#00000000b ; ISPCR.7 = 0, disable ISP
```

17.3. ISP 操作

ISP 意思是在系统编程，它可以不用从用户实际目标产品上移动 MCU 芯片，而能更新用户的应用程序（在 AP-存储区）和非易失性数据（在 IAP-存储区）。这个有益的性能让在广阔的野外更新应用程序成为可能。ISP 模式用于引导程序编程 AP-存储和 IAP 存储。

注意:

(1) 在使用 ISP 功能之前，用户应当用一个通用 Writer/Programmer 或者 Megawin 专用的 Writer/Proagmmer 配置一个 ISP-存储空间和将 ISP 代码（引导程序）写入进 ISP-存储区里。

(2) 在 ISP-存储区的 ISP 代码仅能编程 AP-存储和 IAP –存储。

IAP 操作已经完成后，软件写“001”到 ISPCR.7~ISPCR.5，触发软件复位并且让 CPU 重启进入应用程序存储区（AP-存储区）的地址 0x0000。

我们知道，ISP 代码的目的是编程 AP 存储区和 ISP 存储区。因此，**MCU 必须在 ISP-存储区启动去执行 ISP 代码**。有两种方式去实现在线编程如何让 MCU 从 ISP-存储区启动。

17.3.1. ISP 硬件途径

为了让 MCU 在上电的时候直接从 ISP-存储区启动，MCU 的硬件选项和 ISP-存储区必须使能。ISP 由硬件选项进入的方式叫做硬件途径。一但 HSBS 和 ISP-存储区使能，MCU 将在上电的时候总是从 ISP-存储区启动去执行 ISP 代码(引导程序)。ISP 代码的第一见事应当是去检测是否有 ISP 请求。 如果没有 ISP 请求，ISP 代码应当触发一个软件复位（同时间设定 ISPCR.7~5 为“101”），让 MCU 从 AP-存储区重启去运行用户的应用程序。

如果 HWBS 和 ISP 存储区使能外，HWBS2 也使能，MCU 将在上电后或**外部复位完成后总是**从 ISP-存储区启动。这用外部复位信号为进入 ISP 模式提供了另一个硬件途径。在首次上电后，**MA86E/L104** 可以由外部复位触发去执行 ISP 操作，并且不用等待下一次的上电，这适合不掉电系统去应用 ISP 硬件途径功能。

17.3.2. ISP 软件途径

当 MCU 正在 AP-存储区运行时，ISP 软件途径触发一个软件复位让 MCU 从 ISP-存储器启动。在这个方发里，HWBS 和 HSWBS2 都不使能。正在 AP-存储区运行时，MCU 从 ISP-存储区启动的唯一方式就是软件复位，同一时间设置 ISPC.7~5 为“111”。注意，为了 ISP 软件途径应用，ISP-存储区必须由硬件选项配置一个有效的空间去预留 ISP 模式。

17.3.3. ISP 注意事项

ISP 代码开发

尽管 ISP 代码被编程在 ISP-存储区，在 MCU 的 FLASH 里有 *ISP 起始地址* (见图 17-1), 这并不意味着你必须在你的源码中设定这起始地址= *ISP 起始地址* 。代码的起始地址是由硬件自动操作的。用户仅须像开发在 AP-存储区里的一个应用程序一样。

ISP 期间的中断

触发 ISP/IAP Flash 操作之后，MCU 将停止一会用于内部 ISP 操作，直到操作完成。这个时候，如果之前中断已经使能了，中断将排队等候。一旦处理完成，MCU 继续运行，并且如果排队中的中断的中断标志仍旧激活，则它将立即被处理。然而，用户应当知道下列所说的：

(1) 在 ISP 处理，MCU 停止的时候，任何中断都不能立刻被服务。
(2) nINTx 低/高电平触发外部中断,必须保持触发状态，直到 ISP 完成，否则，它们将被忽略。

ISP 和 Idle 模式

MA86E/L104 不能利用 Idle 模式去执行 ISP 功能。 相反的，它会冻结 CPU 运行，为 ISP/IAP 引擎操作释放 Flash 存储。一旦 ISP/IAP 操作完成，CPU 将恢复并前进到激活 ISP/IAP 操作的指令的下条指令。

ISP 访问目标

之前提到，ISP 用来编程 AP 存储区和 IAP 存储区。一旦访问目标地址超过 IAP –存储区最后字节的地址，硬件将自动忽略触发的 ISP 过程。触发的 ISP 无效，并且硬件不做任何事。

ISP Flash 寿命

嵌入的 Flash 寿命是 100 次写周期。也就是说，写周期不应当超过 100 次。因此用户应当在需频繁更新 AP-存储和 IAP-存储的应用里注意它。

17.4. 在应用中编程 (IAP)

MA86E/L104 有一个*在应用中编程（IAP）*的功能，它允许在应用程序运行的时候把 FLASH 里的一些空间用于非易失性数据存储。这是个很有用的功能，它可以应用到那些需要掉电保存数据的应用中。因此，就没有必要用一个外部串行 EEPROM (如 93C46, 24C01, .., 等等) 来保存非易失性数据存储。

实际上，除了编程的Flash范围不同外，IAP和ISP的操作是一样的。ISP操作的可编程Flash范围包括AP存储空间，IAP操作的范围是被定义IAP存储的空间。

注意:

*(1) 对于**MA86E/L104** 功能，软件应当通过写P页SFR中的IAPLB来配置一个IAP存储空间。T也可以通过一个通用烧录/编程器或Megawin专用的烧录/编程器去配置相应的IAPLB的初始值去配置IAP存储空间。.*

(2) 执行IAP的程序代码应位于AP程序空间并且仅能编程IAP存储空间而不是ISP储存空间

17.4.1. IAP-存储边界/范围

如果定义了ISP存储空间，那么IAP存储空间的范围就由IAP和ISP起始地址设定，如下：

IAP 高边界 = ISP起始地址 –1.
IAP 低边界 = ISP 起始地址 - IAP.

如果没有定义ISP存储空间，则IAP存储空间的范围有下列公式得出：

IAP 高边界 = 0x0FFF.
IAP 低边界 = 0x0FFF – IAP + 1.

例如,如果ISP存储空间是1K，那么ISP的起始地址就是0x0C00，ISP存储空间是1K，则IAP存储空间的范围被定位在0x0800 ~ 0x0BFF。**MA86E/L104**的IAP低边界是由寄存器IAPLB定义的，可以软件修改这个寄存器，以便在用户的AP程序里调整IAP的大小。

17.4.2. 更新 ISP 存储的数据

ISP/IAP有关的特殊功能寄存器.

更新 IAP 存储的“一个字节”，用户可以直接编程一个新的数据进入那个字节。读 IAP 存储的数据，用户可通过***ISP/IAP Flash 读模式***去获取目标数据。.

17.4.3. IAP 注意事项

IAP 期间的中断

触发 ISP/IAP Flash 操作用于在应用编程之后，MCU 将停止一会用于内部 ISP 操作，直到操作完成。这个时候，如果之前中断已经使能了，中断将排队等候。一旦处理完成，MCU 继续运行，并且如果排队中的中断的中断标志仍旧激活，则它将立即被处理。然而，用户应当知道下列所说的：

(1) 在 IAP 处理，MCU 停止的时候，任何中断都不能立刻被服务。
(2) nINTx 低/高电平触发外部中断,必须保持触发状态，直到 IAP 完成，否则，它们将被忽略。

IAP 和 Idle 模式

MA86E/L104 不能利用 Idle 模式去执行 IAP 功能。 相反的，它会冻结 CPU 运行，为 ISP/IAP 引擎操作释放 Flash 存储。一旦 ISP/IAP 操作完成，CPU 将恢复并前进到激活 ISP/IAP 操作的指令的下条指令。

IAP 访问目标

之前提到，IAP 仅用来编程 IAP 存储区。一旦访问目标地址不在 IAP 存储区内，硬件将自动忽略触发的 IAP 过程。触发的 IAP 无效，并且硬件不做任何事。

读 IAP 数据的第二种方式

要读 IAP 存储区的 Flash 数据，除了使用 Flash 读模式之外，第二种方式是使用指令“MOVC A,@A+DPTR”。这里，DPTR 和 ACC 用想要的地址和偏移量分别填入。并且，访问目标必须在 IAP 存储区内，否则读到数据将是不确定的。注意，使用“MOVC”指令比用 Flash 读模式要更快。

IAP Flash 寿命

嵌入的 Flash 寿命是 100 次写周期。也就是说，写周期不应当超过 100 次。因此用户应当在需频繁更新 IAP-存储的应用里注意它。

17.5. ISP/IAP 寄存器

下面专门描述访问ISP,IAP和P页SFP 的特殊功能寄存器：

IFD: ISP/IAP 闪存数据寄存器

SFR 页 = 普通

SFR 地址 = 0xE2 复位初始值= 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 是 ISP/IAP 操作的数据端口寄存器。 在 ISP/IAP/Page-P 写操作时 IFD 的数据将被写入到期望的地址，在 ISP/IAP/Page-P 读操作时 IFD 的值是读到期望地址的数据。

IFADRH: ISP/IAP 高 8 位地址

SFR 页 = 普通

SFR 地址 = 0xE3 复位初始值= 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 是所有 ISP/IAP 模式下的高 8 位地址。在 P 页模式没有定义

IFADRL: ISP/IAP 低 8 位地址

SFR 页 = 普通

SFR 地址 = 0xE4 复位初始值= 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 是所有 ISP/IAP/Page-P 模式下的低 8 位地址。

IFMT: ISP/IAP 闪存模式表

SFR 页 = 普通

SFR 地址 = 0xE5 复位初始值= xxxx-x000

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

Bit 7~3: 保留。当 IFMT 被写时这些位必须写入”0”。

Bit 2~0: ISP/IAP 操作模式选择

MS[2:0]	Mode
0 0 0	备用
0 0 1	AP/IAP-存储 Flash 字节读
0 1 0	AP/IAP-存储 Flash 字节写
0 1 1	**保留**
1 0 0	**P 页 SFR 写**
1 0 1	**P 页 SFR 读**
其它	**保留**

IFMT 是用来选择闪存是用执行众多的 ISP/IAP 功能还是选择 P 页寄存器的访问。

IAPLB: IAP 低边界地址

SFR 页 = **P Only**

SFR 地址 = 0x03 复位初始值= 1111-111x

7	6	5	4	3	2	1	0
IAPLB							0
W	**W**	**W**	**W**	**W**	**W**	**W**	**W**

Bit 7~0: IAPLB 决定 IAP 存储区的最低边界。由于闪存有每页有 512 字节，所以 IAPLB 必须是偶数。
读 IAPLB，MCU 必须先将 IAPLB 在 P 页的地址写如 IFADRL，IMFT 设置成 IAPLB 读模式，然后置位 ISPCR.ISPEN，然后顺序将 0x46 和 0xB9 写入 SCMD，最后 IAPLB 的值就会在 IFD 中。写 IAPLB，MCU 首先将 IAPLB 的新值写入 IFD 中，其次索引 IFADRL ，选择 IMFT ，使能 ISPCR.ISPEN ；然后设置 SCMD 。这样 IAPLB 就会更新到最新的顺序。

由 IAPLB 及 ISP 起始地址决定的 IAP 存储区见下列表。

IAP 低边界 = IAPLBx256, 及

IAP 高边界 = ISP 起始地址 – 1.

例如， IAPLB=0x08 及 ISP 起始地址是 0x0C00, 那么 IAP 存储区就是 0x0800 ~ 0x0BFF.

另外要注意一点，IAP 的低边界地址不能大于 ISP 的起始地址。

SCMD: Sequential Command Data register

SFR Page = Normal

SFR 地址 = 0xE6 复位初始值= 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**

SCMD 是激活 ISP/IAP/Page-P 的命令口。如果 SCMD 连续填入 0x46h, 0xB9h 并且 ISPCR.7 = 1，ISP/IAP/Page-P 被激活。

ISPCR: ISP 控制寄存器

SFR 页 = 普通

SFR 地址 = 0xE7 复位初始值= 0000-0xxx

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

Bit 7: ISPEN, ISP/IAP 操作使能。
0: 所有的 ISP/IAP/Page-P 编程/读都是被禁止的。
1: 使能 ISP/IAP/Page-P 编程/读功能。

Bit 6: SWBS, 软件执行起始选择控制。
0: 复位软件从主存储区开始执行。
1: 复位软件从 ISP 存储区开始执行。

Bit 5: SWRST, 软件复位触发控制。
0: 没有操作
1: 产生软件系统复位，硬件自动清零。

Bit 4: CFAIL, ISP/IAP 操作命令失败指示。
0: 最后一次 ISP/IAP 命令成功。

1: 最后一次 ISP/IAP 命令失败。失败的原因是闪存访问错误。

Bit 3~0: 保留。当 ISPCR 被写时这些位软件必须写入”0”。

17.6. ISP/IAP 示例代码

(1). 规定功能: ISP/IAP flash 读取的通用功能子程序

汇编语言代码范例:

```
_ixp_read:
ixp_read:
    MOV     ISPCR,#ISPEN                ; 使能 ISP/IAP 功能
    MOV     IFMT,#MS0                   ; IFMT =0x01，ISP/IAP 为读取模式

    MOV     SCMD,#046h                  ;触发 ISP/IAP 处理
    MOV     SCMD,#0B9h                  ;

    MOV     IFMT,#000h                  ; IFMT =0x00，选择备用功能
    ANL     ISPCR,#~ISPEN               ; 禁止 ISP/IAP 功能

    RET
```

C 语言代码范例:

```
void ixp_read (void)
{
    ISPCR = ISPEN;                      //使能 ISP/IAP 功能
    IFMT = IxP_Flash_Read;              // IFMT =0x01，ISP/IAP 为读取模式

    SCMD = 0x46;                        //触发 ISP/IAP 处理
    SCMD = 0xB9;                        //

    IFMT = Flash_Standby;               //IFMT =0x00，选择备用功能
    ISPCR &= ~ISPEN;                    //禁止 ISP/IAP 功能
}
```

(2). 规定功能: ISP/IAP flash 编程的通用功能子程序

汇编语言代码范例:

```
_ixp_program:
ixp_program:
    PUSH    ACC

    PUSH    IFADRH                      ;
    PUSH    IFADRL                      ;
    PUSH    IFD                         ;

    MOV     IFADRL,#WDTCR               ;
    MOV     IFD,WDTCR                   ;
    ORL     IFD,#CLRW                   ;
    CALL    page_p_sfr_write            ;

    POP     IFD                         ;
    POP     IFADRL                      ;
    POP     IFADRH                      ;

    MOV     ISPCR,#ISPEN                ; 使能 ISP/IAP 功能
    MOV     IFMT,#MS1                   ; ISP/IAP 写模式，IFMT =0x02
    MOV     SCMD,#046h                  ;
    MOV     SCMD,#0B9h                  ;

    PUSH    IFD                         ;
    MOV     A,IFD                       ;
    CPL     A                           ;
    MOV     IFD,A                       ;

    MOV     IFMT,#MS0                   ; IFMT =0x01，ISP/IAP 为读取模式
    MOV     SCMD,#046h                  ;
    MOV     SCMD,#0B9h                  ;

;   if(reg[2] == IFD)
    POP     ACC                         ;
    CJNE    A,IFD,ixp_first_progrma_fail    ;
    JMP     ixp_progrma_Pass            ;
```

```
ixp_first_progrma_fail:
;       page_p_sfr_write (WDTCR_P,(WDTCR | CLRW)); //
        MOV         IFD,A                       ;
        PUSH        IFADRH                      ;
        PUSH        IFADRL                      ;
        PUSH        IFD                         ;

        MOV         IFADRL,#WDTCR               ;
        MOV         IFD,WDTCR                   ;
        ORL         IFD,#CLRW                   ;
        CALL        page_p_sfr_write            ;

        POP         IFD                         ;
        POP         IFADRL                      ;
        POP         IFADRH                      ;

        ANL         ISPCR,#~CFAIL               ;
        MOV         IFMT,#MS1                   ; ISP/IAP 写模式，IFMT =0x02
        MOV         SCMD,#046h                  ;
        MOV         SCMD,#0B9h                  ;

        PUSH        IFD                         ;
        MOV         A,IFD                       ;
        CPL         A                           ;
        MOV         IFD,A                       ;

        MOV         IFMT,#MS0                   ; IFMT =0x01，ISP/IAP 为读取模式
        MOV         SCMD,#046h                  ;
        MOV         SCMD,#0B9h                  ;

;       if(reg[2] == IFD)
        POP         ACC
        CJNE        A,IFD,ixp_second_progrma_Fail
ixp_progrma_Pass:
;       SETB        ixp_program_state           ; 成功为 1
        CLR         ixp_program_state           ; 成功为 0
```

```
end_ixp_program:
    MOV     IFMT,#000h                  ;
    ANL     ISPCR,#~ISPEN               ;

    POP     ACC                         ;
    RET                                 ;

ixp_second_progrma_Fail:
;   CLR     ixp_program_state           ; 失败为 0
    SETB    ixp_program_state           ; 失败为 1
    JMP     end_ixp_program             ;
```

C 语言代码范例:

```
bit ixp_program(void)
{   unsigned char reg[3];

    reg[0] = IFADRH;                    //
    reg[1] = IFADRL;                    //
    reg[2] = IFD;

    IFADRL = WDTCR_P;                   //
    IFD = (WDTCR | CLRW);               //
    page_p_sfr_write ();                //

    IFADRH = reg[0];                    //
    IFADRL = reg[1];                    //
    IFD = reg[2];                       //

    ISPCR = ISPEN;                      //使能 ISP/IAP 功能
    IFMT = IxP_Flash_Program;           // ISP/IAP 写模式，IFMT =0x02

    SCMD = 0x46;
    SCMD = 0xB9;

    IFD = ~reg[2];                      //
```

```
    IFMT = IxP_Flash_Read;                  // IFMT =0x01，ISP/IAP 为读取模式
    SCMD = 0x46;                            //
    SCMD = 0xB9;                            //

    if(reg[2] == IFD)                       //
      {
        IFMT = Flash_Standby;               // IFMT =0x00，选择备用功能
        ISPCR &= ~ISPEN;                    //
        return(Pass);                       //
      }
    else                                    //
    {   IFADRL = WDTCR_P;                   //
        IFD = (WDTCR | CLRW);               //
        page_p_sfr_write ();                //

        IFADRH = reg[0];                    //
        IFADRL = reg[1];                    //
        IFD = reg[2];                       //

        ISPCR &= ~CFAIL;                    //
        IFMT = IxP_Flash_Program;           // ISP/IAP 写模式，IFMT =0x02

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

        IFD = ~reg[2];                      //

        IFMT = IxP_Flash_Read;              // IFMT =0x01，ISP/IAP 为读取模式
        SCMD = 0x46;                        //
        SCMD = 0xB9;                        //

        IFMT = Flash_Standby;               // IFMT =0x00，选择备用功能
        ISPCR &= ~ISPEN;                    //

        if(reg[2] != IFD)                   //
        {
            return(Fail);                   //
```

```
        }
                else                                    //
        {
            return(Pass);                               //
        }
    }
}
```

(3). 规定功能: 失败(CFAIL)检测的 ISP 示例代码,如果是失败(CFAIL)则写两次。

汇编语言代码范例:

```
isp_hw_approached:
    MOV     DPH,#High(00000h)           ;
    MOV     DPL,#LOW(00000h)            ;

isp_hw_write_loop:
    MOV     IFADRL,DPL                  ;
    MOV     IFADRH,DPH                  ;
    MOV     IFD,#055h                   ; ISP  编程数据
    CALL    ixp_program                 ;

;   JNB     ixp_program_state,isp_hw_write_fail
                                        ; 失败为 0

    JB      ixp_program_state,isp_hw_write_fail
                                        ; 失败为 1
isp_hw_write_next:
    INC     DPTR                        ;

    MOV     A,DPH                       ;
    CJNE    A,#01Ch,isp_hw_write_loop   ;

    ANL     ISPCR,#~SWBS                ;
    ORL     ISPCR,#SWRST                ;

isp_hw_write_fail:
```

```
;           to do ...
      JMP       isp_hw_write_next                   ;
```

C 语言代码范例:

```
      unsigned short addr=0x0000;                   //
      do{ IFADRH = (unsigned char)(addr >>8);
            IFADRL = (unsigned char)addr ;          //
            IFD = 0x55;                             //
            if(ixp_program() == Fail);              //
            {
//                  to do ...
            }
      }while(++addr != 0x1C00);                     //
      ISPCR = SWRST;                                //  选择 AP 启动和软件复位
```

(4). 规定功能:软件启动 ISP 的示例代码

汇编语言代码范例:

```
      MOV       ISPCR,#(SWBS|SWRST)             ;
```

C 语言代码范例:

```
      ISPCR = (SWBS | SWRST) ;                  //
```

(5). 规定功能: IAPLB 更改片内 flash 的 4K IAP 边界

汇编语言代码范例:

```
      MOV       IFADRL,#IAPLB                   ;
      MOV       IFD,#(HIGH(4096))&0xFE          ;
      CALL      page_p_sfr_write                ;
```

C 语言代码范例:

```
IFADRL = IAPLB;                                    //
IFD = (((unsigned char)(4096 >> 8)) & 0xFE);    //
page_p_sfr_write();                                 //
```

18. P 页特殊功能寄存器访问

MA86E/L104 内建一个特别的 P 页寄存器 (Page P) 用来存储 MCU 操作的控制寄存器。这些特殊功能寄存器在不同 IFMT 下通过 ISP/IAP 操作来访问。在 P 页访问时，IFADRH 必须设置为"00"及 IFADRL 索引 P 页内特殊功能寄存器地址。如果 IFMT= 04H 则 P 页写操作，在 SCMD 激活之后 IFD 的数据会被载入到 IFADRL 索引的特殊功能寄存器。这些特殊功能寄存器在 P 页模式下不支持读功能。

下面描述的是 P 页里的特殊功能寄存器：

IAPLB: IAP 低边界地址

SFR 页 = **P**

SFR 地址 = 0x03 复位初始值 = 1111-1111

7	6	5	4	3	2	1	0
IAPLB							**0**
W	**W**	**W**	**W**	**W**	**W**	**W**	**W**

Bit 7~0: IAPLB 决定 IAP 存储区的最低边界。
写 IAPLB ，首先 MCU 把新的 IAPLB 设定值写入 IFD ；其次索引 IFADRL ，选择 IMFT ，使能 ISPCR.ISPEN ；然后设置 SCMD 。这样 IAPLB 就会更新到最新的顺序。

由 IAPLB 及 ISP 起始地址决定的 IAP 存储区见下列表。

IAP 低边界 = IAPLBx256, 及
IAP 高边界 = ISP 起始地址 – 1.

例如， IAPLB=0x08 及 ISP 起始地址是 0x0C00, 那么 IAP 存储区就是 0x0800 ~ 0x0BFF.

另外要注意一点，IAP 的低边界地址不能大于 ISP 的起始地址。

CKCON2: 时钟控制寄存器 2

SFR 页 = **P**

SFR 地址 = 0x40 复位初始值 = 0001-xx00

7	6	5	4	3	2	1	0
--	--	XTALE	IHRCOE	--	--	OSCS1	OSCS0
W	**W**	**W**	**W**	**W**	**W**	**W**	**W**

Bit 7~6: 保留。当 CKCON2 写入时，这两个位软件必须写"0"。

Bit 5: XTALE, 外部晶振 (XTAL)使能。
0: 禁止（XTAL）振荡电路。 此时 XTAL2 及 XTAL1 当做 P 4.0 及 P 4.1。
1: 使能（XTAL）振荡电路。如果此位是通过 CPU 软件来设置的话，则在 XTALE 使能之后需要 3 毫秒才能稳定输出。

Bit 4: IHRCOE, 内部高频 RC 振荡使能。默认为置位用于 MCU 时钟来源
0: 禁止内部高频 RC 振荡。
1: 使能内部高频 RC 振荡。如果此位是通过 CPU 软件来设置的话，则在 IHRCOE 使能之后需要 32 微秒才能稳定输出。

Bit 3~2: 保留。当 CKCON2 写入时，这两个位软件必须写"0"。

Bit 1~0: OSC[1:0], OSCin 来源选择。

OSCS[1:0]	OSCin 来源选择
0 0	IHRCO
0 1	XTAL
1 0	ILRCO
1 1	ECKI, 外部时钟输入 (P4.0) 作为 OSCin.

PCON2: 电源控制寄存器 2

SFR 页 = **P**

SFR 地址 = 0x44 复位初始值= x0xx-xx01

7	6	5	4	3	2	1	0
--	**AWBOD0**	--	--	--	--	BO0RE	**1**
W	**W**	**W**	**W**	**W**	**W**	**W**	**W**

Bit 7: 保留。当 PCON2 写入时，这两个位软件必须写”0”。

Bit 2: AWBOD0, 在掉电模式下 BOD0 唤醒使能。
0: 在掉电模式下禁止 BOD0 唤醒。
1: 在掉电模式下 BOD0 保持运行。

Bit 5~2: 保留。当 PCON2 写入时，这些位软件必须写”0”。

Bit 1: BO0RE, BOD0 复位使能。初始值通过硬件选项 BO0REO 来改变。
0: 当 BOF0 设置时，禁止 BOD0 触发系统复位。
1: 当 VDD 为 4.2V(E) 或 2.4V(L)而 BOF0 设置时，使能 BOD0 触发系统复位。

Bit 0: 测试保留。当 PCON2 写入时，此位软件必须写”1”。

SPCON0: 特殊功能寄存器页控制 0

SFR 页 = **P**

SFR 地址 = 0x48 复位初始值= xxx0-x000

7	6	5	4	3	2	1	0
--	--	--	WRCTL	--	CKCTL0	PWCTL1	PWCTL0
W	**W**	**W**	**W**	**W**	**W**	**W**	**W**

Bit 7~5: 保留。当 SPCON0 写入时，这些位软件必须写”0”。

Bit 4: WRCTL. WDTCR 特殊功能寄存器访问控制。
如果 WRCTL 设置，则在普通页里 WDTCR 特殊功能寄存器禁止修改。在普通页里 WDTCR 仅是保持特殊功能寄存器读的功能。但是在特殊功能寄存器 P 页里软件总是拥有修改的能力。

Bit 3: 保留。当 SPCON0 写入时，此位软件必须写”0”。

Bit 2: CKCTL0. CKCON0 特殊功能寄存器访问控制。
如果 CKCTL0 设置，则在普通页里 CKCON0 特殊功能寄存器禁止修改。在普通页里 CKCON0 仅是保持特殊功能寄存器读的功能。但是在特殊功能寄存器 P 页里软件总是拥有修改的能力。

Bit 1: PWCTL1. PCON1 特殊功能寄存器访问控制。
如果 PWCTL1 设置，则在普通页里 PCON1 特殊功能寄存器禁止修改。在普通页里 PCON1 仅是保持特殊功能寄存器读的功能。但是在特殊功能寄存器 P 页里软件总是拥有修改的能力。

Bit 0: PWCTL0. PCON0 特殊功能寄存器访问控制。

如果 PWCTL0 设置，则在普通页里 PCON0 特殊功能寄存器禁止修改。在普通页里 PCON0 仅是保持特殊功能寄存器读的功能。但是在特殊功能寄存器 P 页里软件总是拥有修改的能力。

DCON0: 设备控制 0

SFR 页 = **P**

SFR 地址 = 0x4C 复位初始值= 10xx-xx1x

7	6	5	4	3	2	1	0
HSE	IAPO	--	--	--	--	RSTIO	--
W	**W**	**W**	**W**	**W**	**W**	**W**	**W**

Bit 7: HSE, 高速运行使能。
0: 禁止 MCU 高速运行。
1: 使能 MCU 高速运行。(F_{SYSCLK}> 6MHz)

Bit 6: IAPO, 仅是 IAP 功能。
0: 保持 IAP 区是 IAP 功能和代码执行。
1: 禁止 IAP 区代码执行，IAP 区仅是 IAP 功能。

Bit 5~2: 保留。当 DCON0 写入时，这些位软件必须写”0”。

Bit 1: RSTIO, RST 引脚功能
0: 选择引脚作为 P3.6 脚。
1: 选择引脚作为 RST 脚。

Bit0: 保留。当 DCON0 写入时，这些位软件必须写”0”。

18.1. P 页示例代码

(1). 规定功能: P 页特殊功能寄存器(SFR)读取的通用功能子程序

汇编语言代码范例:

```
_page_p_sfr_read:
page_p_sfr_read:
    MOV     IFADRH,000h
    MOV     IFMT,#(MS2|MS0)             ; P 页读取，IFMT =0x05

    ANL     ISPCR,#CFAIL                ;
    ORL     ISPCR,#ISPEN                ; 使能 IAP/ISP 功能

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

    MOV     IFMT,#000h                  ; IAP/ISP 备用模式，IFMT =0x00
    ANL     ISPCR,#~ISPEN               ; 禁止 IAP/ISP 功能

    RET
```

C 语言代码范例:

```
void page_p_sfr_read (void)
{
    IFADRH = 0x00;                      //

    ISPCR = ISPEN;                      //使能 IAP/ISP 功能
    IFMT = (MS0 | MS2);                 // P 页读取，IFMT =0x05

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

    IFMT = Flash_Standby;               // IAP/ISP 备用模式，IFMT =0x00
    ISPCR &= ~ISPEN;                    //禁止 IAP/ISP 功能
}
```

(2). 规定功能: P 页特殊功能寄存器(SFR)写的通用功能子程序

汇编语言代码范例:

```
_page_p_sfr_write:
page_p_sfr_write:
    MOV     IFADRH,000h                 ;

    MOV     ISPCR,#ISPEN                ; 使能 IAP/ISP 功能
    MOV     IFMT,#MS2                   ; P 页写，IFMT =0x04

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

    MOV     IFMT,#000h                  ; IAP/ISP 备用模式，IFMT =0x00
    ANL     ISPCR,#~ISPEN               ; 禁止 IAP/ISP 功能

    RET
```

C 语言代码范例:

```
void page_p_sfr_write (void)
{
    IFADRH = 0x00;

    ISPCR = ISPEN;                      //使能 IAP/ISP 功能
    IFMT = MS2;                         // P 页写，IFMT =0x04

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

    IFMT = Flash_Standby;               // IAP/ISP 备用模式，IFMT =0x00
    ISPCR &= ~ISPEN;
}
```

(3). 规定功能: 使能 PWCTL0 在 P 页控制 PCON0.PD

汇编语言代码范例:

```
MOV     IFADRL,#SPCON0          ;
CALL    page_p_sfr_read         ;

ORL     IFD,#PWCTL0             ; 设置 PWCTL0
CALL    page_p_sfr_write        ;

MOV     IFD,PCON0               ; 设置 PCON0

ORL     IFD,#PD                 ; 写 PCON0 并且掉电
MOV     IFADRL,#PCON0_P         ;
CALL    page_p_sfr_write        ;
```

C 语言代码范例:

```
IFADRL = SPCON0;                //
page_p_sfr_read();              //

IFD |= PWCTL0;                  // 设置 PWCTL0
page_p_sfr_write();             //

IFD = PCON0;                    // 读取 PCON0

IFD |= PD;                      // 写 PCON0
IFADRL = PCON0_P;               //
page_p_sfr_write();             //
```

(4). 规定功能: 使能 CKCTL0 在 P 页更改系统时钟 SYSCLK 分频器 (CKCON0)

汇编语言代码范例:

```
MOV     IFADRL,#SPCON0          ;
CALL    page_p_sfr_read         ;
```

```
ORL     IFD,#CKCTL0                 ; 设置 CKCTL0
CALL    page_p_sfr_write            ;

MOV     IFD,CKCON0                  ; 读取 CKCON0

ORL     IFD,#(AFS | SCKS0)          ; 写 CKCON0 及设置 AFS
MOV     IFADRL,#CKCON0_P            ; 系统时钟 SYSCLK / 2
CALL    page_p_sfr_write
```

C 语言代码范例:

```
IFADRL = SPCON0;                    //
page_p_sfr_read ();                 //

IFD |= CKCTL0;                      // 设置 CKCTL0
page_p_sfr_write();                 //

IFD = CKCON0;                       // 读取 CKCON0

IFD |= (AFS | SCKS0);               //
IFADRL = CKCON0_P;                  //
page_p_sfr_write();                 // 写 CKCON0
```

19. 辅助特殊功能寄存器

AUXR0: 辅助寄存器 0

辅助寄存器 ***0*** 地址 = 0xA1 复位初始值 = 0000-0000

7	6	5	4	3	2	1	0
P40OC1	P40OC0	P40FD	--	**P1FS1**	**P1FS0**	INT1H	INT0H
R/W	**R/W**	**R/W**	**W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6: P4.0 输出设定控制位 1 位 0。当选择内部 RC 震荡（IHRCO 或 ILRCO）作为系统时钟时这两个位才起作用。在晶振模式下， XTAL2 和 XTAL1 可以作为 P4.0 和 P4.1。在外部时钟输入模式下，P4.0 作为外部时钟输入引脚。在内部振荡环境下，P4.0 提供下表设定选择作为通用输入输出口或时钟源产生器。当 P4.0OC[1:0] 设定为非 P4.0 时，P4.0 将驱动片内 RC 振荡器（IHRCO 或 ILRCO）输出作为其它设备的时钟源。

P40OC[1:0]	P4.0 功能	P4.0 I/O 模式
00	P4.0	By P4M0.0
01	IHRCO	By P4M0.0
10	IHRCO/2	By P4M0.0
11	IHRCO/4	By P4M0.0

对于 P4.0 的时钟输出功能，建议设置 P4M0 为“1”，选择 P4.0 作为推挽输出模式

Bit 5: P40FD, P4.0 快速驱动。
0: P4.0 作为缺省驱动输出。
1: P4.0 快速驱动输出使能。若 P4.0 被配置为时钟输出， 当 P4.0 输出频率大于 12MHz （5V） 或者大于 6MHz （3V） 时使能此位。

Bit 4: 保留。在写 AUXR0 时，该位软件必须写“0”
Bit 3~2: P1.4 和 P1.5 设定选择。

P1FS[1:0]	P1.4	P1.5
00	P1.4	P1.5
01	**--**	TXD1 输出
10	nINT0 输入	nINT1 输入
11	T0 输入	T1 输入

Bit 1: INT1H, INT1 高电平/上升沿触发使能。
0: 保留 P3.3 的低电平或下降沿作为 INT1 触发。
1: 设置 P3.3 的高电平或上升沿作为 INT1 触发。

Bit 0: INT0H, INT0 高电平/上升沿触发使能。
0: 保留 P3.2 的低电平或下降沿作为 INT0 触发。
1: 设置 P3.2 的高电平或上升沿作为 INT0 触发。

AUXR1: 辅助控制寄存器 1

辅助寄存器 1 地址 = 0xA2 复位初始值 = 0000-0000

7	6	5	4	3	2	1	0
RTX3E	RTX2E	RTX1E	RTX0E	--	--	RXCS1	RXCS0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: RTX3E, RXD 重复到 TXD3 使能。
0: 禁止 RXD 重复到 TXD3 (P1.5)。
1: 使能 RXD 重复到 TXD3 (P1.5)。

Bit 6: RTX2E, RXD 重复到 TXD2 使能。
0: 禁止 RXD 重复到 TXD2 (P1.3)。

1: 使能 RXD 重复到 TXD2 (P1.3).
Bit 5: RTX1E, RXD 重复到 TXD1 使能。
0: 禁止 RXD 重复到 TXD1 (P1.2)。
1: 使能 RXD 重复到 TXD1 (P1.2)。

Bit 4: RTX0E, RXD 重复到 TXD0 使能。
0: 禁止 RXD 重复到 TXD0 (P3.1).
1: 使能 RXD 重复到 TXD0 (P3.1).

Bit 3~2: 保留。 当 AUXR1 需要改写时这两位必须写入"0"。
Bit 1~0: RXD 通道选择。

RXCS[1:0]	RXD 通道选择
00	P3.0
01	P3.2
10	P3.3
11	P1.4

AUXR2:辅助寄存器 2

辅助寄存器 2 地址= 0xA3　　复位初始值 = 0000-0000

7	6	5	4	3	2	1	0
URXR	BTI	URM0X6	SMOD2	T1X12	T0X12	T1CKOE	T0CKOE
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: URXR, 串行口 RX 选项。
0: 清零是禁止此选项。
1: 设置是使能此选项。

Bit 6: BTI, 阻止 TI 在串行口中断。
0: 保留 TI 作为串行口中断源。
1: 阻止 TI 作为串行口中断源。

Bit 5: URM0X6, 串行口模式 0 波特率选择。
0: 清零是选择 SYSCLK/12 作为串行口模式 0 波特率。
1: 设置是选择 SYSCLK/2 作为串行口模式 0 波特率。

Bit 4: SMOD2, 额外的波特率 X2/X4 选择。
0: 使能额外的波特率 X2/X4。
1: 禁止额外的波特率 X2/X4。

Bit 3: T1X12, 当 C/T=0 时，定时器 1 时钟源选择。
0: 清零是选择 SYSCLK/12 作为定时器 1 时钟源。
1: 设置是选择 SYSCLK 作为定时器 1 时钟源。

Bit 2: T0X12, 当 C/T=0 时，定时器 0 时钟源选择。
0: 清零是选择 SYSCLK/12 作为定时器 0 时钟源。
1: 设置是选择 SYSCLK 作为定时器 0 时钟源。

Bit 1: T1CKOE, 定时器 1 时钟输出使能。
0: 禁止定时器 1 时钟输出。
1: 使能定时器 1 时钟输出。

Bit 0: T0CKOE, 定时器 0 时钟输出。
0: 禁止定时器 0 时钟输出。

1: 使能定时器 0 时钟 P3.4 输出。

20. 硬件选项

MCU 的硬件选项定义了器件的性能，它不能由软件编程和控制。硬件选项仅能由通用编程器，“Megawin 8051 Writer U1”或“Megawin 8051 OCD_ICE/ICP Programmer”来编程。整片擦除后，所有的硬件选项被设置成“禁止”状态，没有配置 ISP 空间和 IAP 空间 。 **MA86E/L104** 有下列的硬件选项

LOCK:
☑: 使能. 加密上锁，使得用通用编程器读取代码锁定为 0xFF
☐: 禁止. 没有上锁

ISP 存储空间:
由其指定 ISP 空间的起始地址。它的高边界由 Flash 的结束地址限定，例如：0x0FFF。下表列举了 ISP 空间选项。默认设定， **MA86E/L104** ISP 空间被配置为 1K， 并嵌入了 Megawin 专用的 ISP 代码使用 Megawin 1-线 ISP 协议来执行在系统编程。

ISP 空间大小	ISP 起始地址
3.5K bytes	0x0200
3K bytes	0x0400
2.5K bytes	0x0600
2K bytes	0x0800
1.5K bytes	0x0A00
1K bytes	0x0C00
0.5K bytes	0x0E00
无 ISP 空间	--

HWBS:
☑: 使能. 上电时，如果 ISP 空间有配置，则 MCU 从 ISP 空间启动
☐: 禁止. MCU 总是从 AP 空间启动

HWBS2:
☑: 使能. 如果 ISP 空间有配置，不仅上电，而且所有复位都是从 ISP 空间启动
☐: 禁止. 由 HWBS 决定 MCU 从哪里启动

IAP 存储空间:
IAP 存储空间指定用户定义的 IAP 空间。IAP 存储空间可以由硬件选项或者 MCU 软件修改 IAPLB 来配置。默认，它被配置为 1KB

BO0REO:
☑: 使能. BOD0 将触发复位事件使得 CPU 从 AP 程序起始地址允许(2.2V)
☐: 禁止. BOD0 不能触发 CPU 复位

WRENO:
☑: 使能. 置位 WDTCR.WREN 使能 WDTF 产生一个系统复位
☐: 禁止. 清零 WDTCR.WREN 禁止 WDTF 产生一个系统复位

NSWDT: 不停止 WDT
☑: 使能. 置位 WDTCR.NSW 在掉电模式下使能 WDT 运行 (watch 模式).
☐: 禁止. 清零 WDTCR.NSW 在掉电模式下禁止 WDT 允许(禁止 Watch 模式).

HWENW: 硬件加载“ENW”到 WDTCR.

☑: 使能. 上电后使能 WDT 并且加载 WRENO, NSWDT, HWWIDL 和 HWPS2~0 的内容到 WDTCR

☐: 禁止. 上电后 WDT 不会自动使能

HWWIDL, HWPS2, HWPS1, HWPS0:

当 HWENW 使能，上电后这 4 个熔丝位的内容将被加载到 WDTCR

WDSFWP:

☑: 使能. WDT 特殊寄存器，WDTCR 的 WREN, NSW, WIDL, PS2, PS1 和 PS0 位,将被写保护

☐: 禁止. WDT 特殊寄存器，WDTCR 的 WREN, NSW, WIDL, PS2, PS1 和 PS0 位,由软件自由写

P36EN:

☑: 使能. RSTIO (DCON0.1)将被清零，RST 引脚用作 P3.6 口

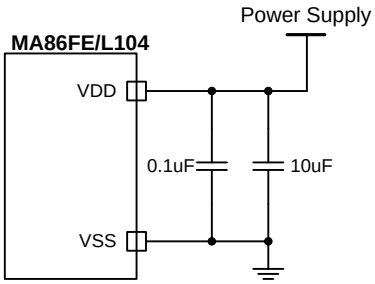
☐: 禁止. RSTIO (DCON0.1)将被置位，保留 RST 引脚功能

21. 应用说明

21.1. 电源电路

MA86E/L104 的工作电源变化对于 E 类型可以从 4.2V 到 5.5V，对于 L 类型可以从 2.4V 到 3.6V，但是增加一些外部解耦和滤波电容是必须的，如图 21–1 所示。

图 21–1. 电源电路



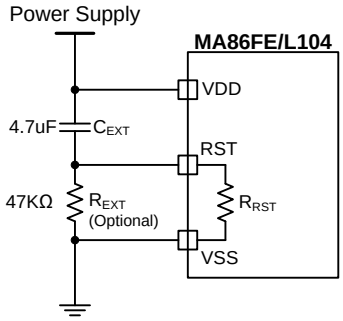
21.2. 复位电路

通常，上电可以成功产生上电复位，然而，为了上电时 MCU 产生一个可靠的复位，有必要加外部复位。外部复位电路如图 21–2 所示，它由一个连接到 VDD（电源）的电容 C_{EXT} 和一个连接到 VSS(地)的电阻组成。

一般的, R_{EXT} 是可选的，因为 RST 引脚有一个内部下拉电阻(R_{RST}). 这个对 VSS 的内部扩散电阻在仅使用一个外部对 VDD 的电容 C_{EXT} 时也可产生一个上电复位

R_{RST} 的值见 “**錯誤! 找不到參照來源。 錯誤! 找不到參照來源。**”。

图 21–2. 复位电路



21.3. XTAL 振荡电路

为了能成功起振 (最大到 24MHz), 电容 C1 和 C2 是必须的，如图 21–3 所示。通常， C1 和 C2 使用相同的值。. 表 21–1 列举了 C1 & C2 在不同晶振下的值。

图 21–3. XTAL 振荡电路

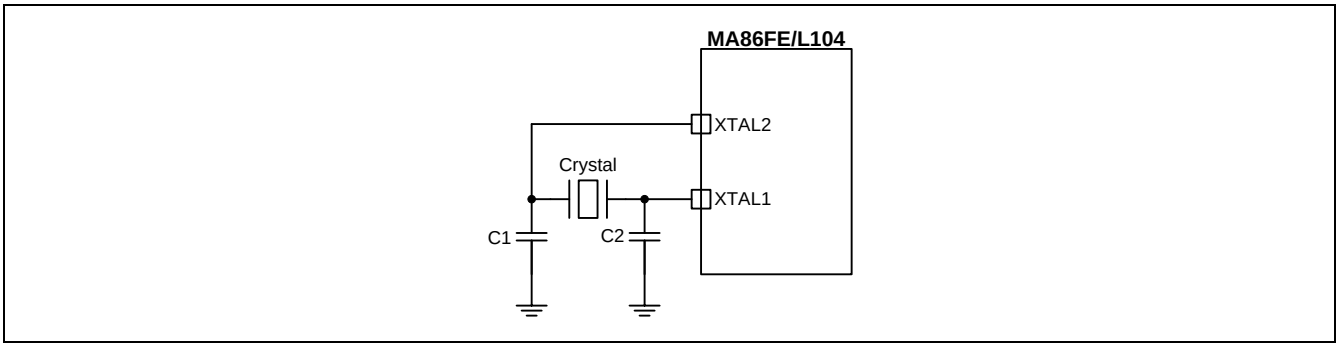


表 21–1. 振荡电路的电容 C1&C2 参照表

晶振	C1, C2 电容
16MHz ~ 25MHz	10pF
6MHz ~ 16MHz	15pF
2MHz ~ 6MHz	33pF

21.4. ICP 接口电路

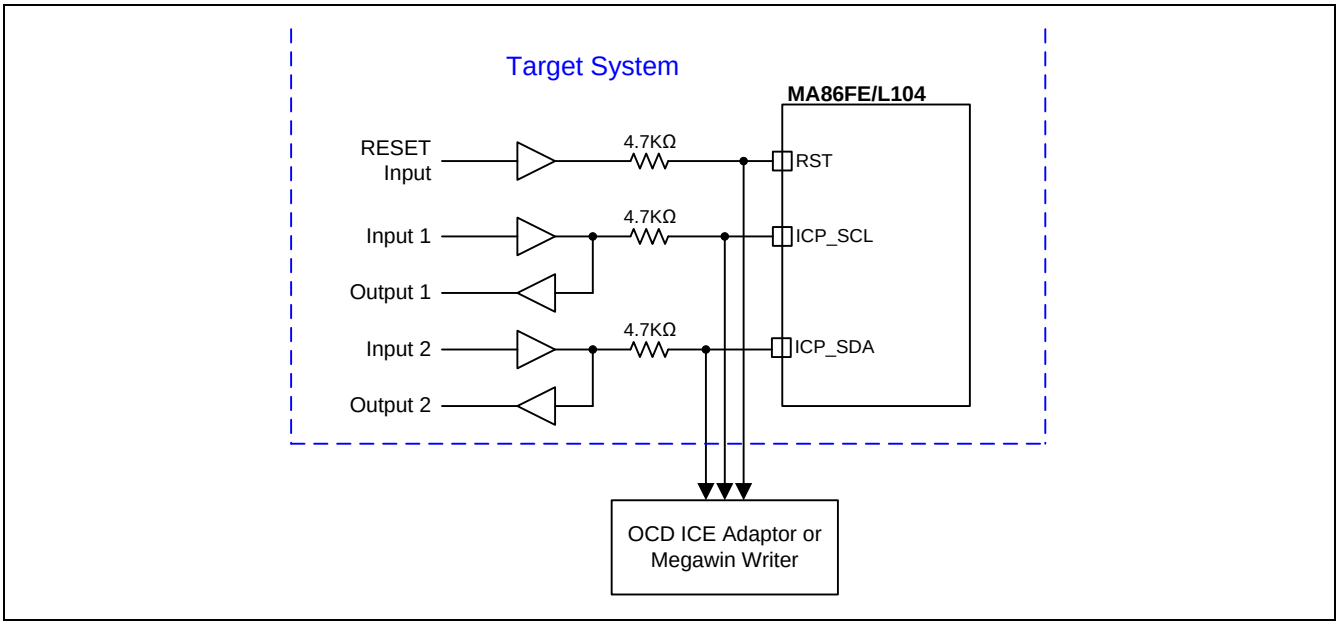
MA86E/L104 包含一个笙泉专有的在芯片编程接口，它允许在元器件已经安装在产品上在芯片编程(ICP)。ICP 接口使用一个时钟线和个双向数据线完成主机向器件编程操作。

ICP 接口允许的 ICP_SCL/ICP_SDA 引脚与用户应用共享，使得可以实现在芯片 FLASH 编程。这是可行的，因为当芯片在 Halt 状态时执行 ICP 通信，此时芯片上的外围设备和用户软件都是失效的。在 halt 状态，ICP 接口能够安全的“借用”ICP_SCL (P1.6)和 ICP_SDA (P1.5) 引脚。 在大多应用中，必须用外部电阻来隔开 ICP 电路和用户应用电路。. 图 21–4. 显示了一种典型的隔离方法。

强烈建议在目标系统建立 ICP 接口电路。它保留了整个软件编程和硬件选项配置的能力。

注意: MA86E/L104AS8 不支持 ICP 接口

图 21–4. ICP 和 OCD 接口电路



21.5. 在芯片编程功能

ICP，就像传统的并行编程方式，可以编程 MCU 的任何区域，包括 FLASH 和 MCU 的硬件选项。并且，得益于它专用的串行接口，使得 ICP 可以更新 MCU 而不用从用户的产品上卸下 MCU，就像 ISP 做的那样。

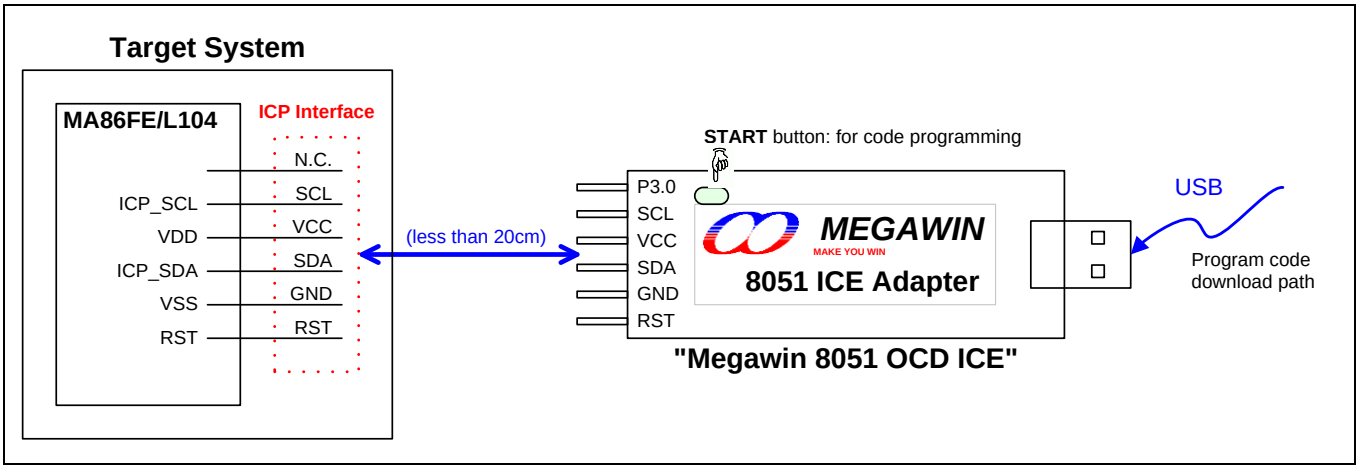
专用的 6 脚 “Megawin 8051 ICE Adapter” 可以支持 **MA86E/L104** 在线路编程。“Megawin 8051 ICE Adapter” 有在系统的存储器来存储用户的程序和器件选项。因此，该工具可以完成一个便携的，独立的编程，而不用连线主机，如连接该工具到 PC。下面列举了 ICP 功能的特点：

特点

- 不必在目标芯片上预编程一个引导程序。
- 专用串行接口;不占用 IO 口
- 目标芯片不必在运行状态；仅需电源。
- 便携，独立的工作，而无需主机的干预。

以上特点使得 ICP 非常有利于用户。特别的，在编程数据下载后的便携独立工作，尤其有利于没有 PC 的地方使用。图 21–5. 显示了 ICP 独立编程的系统框图。ICP 接口仅需 **5** 个引脚： SDA 线和 SCL 线是串行数据和串行时钟，用来从 6-pin “Megawin 8051 ICE Adapter”传送编程数据到目标 MCU； RST 线用来 hal MCU； VCC & GND 是 6-pin “Megawin 8051 ICE Adapter” 用于便携编程应用的电源输入。 USB 连接器可以直接的插入 PC 的 USB 端口，用来从 PC 下载编程数据到 6-pin “Megawin 8051 ICE Adapter”。

图 21–5.经由 ICP 的独立编程



22. 电气特性

22.1. 最大绝对额定值

MA86E104

参数	额定值	单位
工作温度	-40 ~ +85	°C
存储温度	-65 ~ + 150	°C
所有输入输出口及复位脚对 VSS 电压	-0.5 ~ VDD + 0.5	V
VDD 对 VSS 电压	-0.5 ~ +6.0	V
VDD 对 VSS 的最大电流	200	mA
任意端口最大输出灌电流	40	mA

***注:** 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。

MA86L104

参数	额定值	单位
工作温度	-40 ~ +85	°C
存储温度	-65 ~ + 150	°C
所有输入输出口及复位脚对 VSS 电压	-0.3 ~ VDD + 0.3	V
VDD 对 VSS 电压	-0.3 ~ +4.2	V
VDD 对 VSS 的最大电流	200	mA
任意端口最大输出灌电流	40	mA

***注:** 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。

22.2. 直流特性

MA86E104

VDD = 5.0V±10%, VSS = 0V, T_A = 25 ℃并且 CPU 空运行，除非另外说明

符号	参数	测试条件	界限			单位
			最小	典型	最大	
输入/输出特性						
V_{IH1}	输入高电压(所有输入输出口)	除了 RST,P4.0,P4.1	2.0			V
V_{IH2}	输入高电压 (P4.0,P4.1)		3.5			V
V_{IH3}	输入高电压(复位脚/P3.6)		4.0			
V_{IL1}	输入低电压(所有输入输出口)	除了 RST,P4.0,P4.1			0.8	V
V_{IL2}	输入低电压(P4.0,P4.1)				1.0	V
V_{IL3}	输入低电压(复位脚/P3.6)				1.0	V
I_{IH}	输入高漏电流(所有输入输出口)	V_{PIN} = VDD		0	10	uA
I_{IL1}	逻辑电平 0 输入电流(P3 在准双向口，或输入脚内部上拉电阻)	V_{PIN} = 0.4V		28	60	uA
I_{IL2}	逻辑电平 0 输入电流 (所有输入口或漏极开路口)	V_{PIN} = 0.4V		0	10	uA
I_{H2L}	逻辑电平从 1 到 0 输入转换电流 (P3 在准双向口，或输入脚内部上拉电阻)	V_{PIN} =1.8V		330	500	uA
I_{OH1}	输出高时电流(P3 在准双向输入输出口模式)	V_{PIN} =2.4V	150	220		uA
I_{OH2}	输出高时电流(所有推挽式输出口)	V_{PIN} =2.4V	12			mA
I_{OL1}	输出低时电流(所有输入输出口)	V_{PIN} =0.4V	12			mA
R_{RST}	内部复位下拉电阻			77		Kohm
功　耗						
I_{OP1}	正常模式下工作电流	SYSCLK = 24MHz @IHRCO		8.5		mA
I_{OP2}		SYSCLK = 12MHz @IHRCO		5.9		mA
I_{OP3}		SYSCLK = 6MHz @IHRCO&HSE=0		4.1		mA
I_{OP4}		SYSCLK = 3MHz @IHRCO&HSE=0		2.4		mA
I_{OP5}		SYSCLK = 24MHz @XTAL		10		mA
I_{OP6}		SYSCLK = 12MHz @XTAL		6.9		mA
I_{OP7}		SYSCLK = 6MHz @XTAL&HSE=0		5		mA
I_{OP8}		SYSCLK = 2MHz @XTAL&HSE=0		2.5		mA
I_{OPS1}	低速模式电流	SYSCLK = 24MHz/128 @IHRCO		0.86		mA
I_{IDLE1}	闲置模式电流	SYSCLK = 24MHz @IHRCO		2.1		mA
I_{IDLE2}	闲置模式电流	SYSCLK = 24MHz @XTAL		3.6		mA
I_{IDLE3}	闲置模式电流	SYSCLK = 24MHz/128 @IHRCO		0.77		mA
I_{IDLE4}	闲置模式电流	SYSCLK = 24MHz/128 @XTAL		2.2		mA
I_{IDLE5}	闲置模式电流	SYSCLK = 64KHz @ILRCO		20		uA
I_{sub1}	子时钟模式工作电流	SYSCLK = 64KHz @ILRCO, &HSE=0		55		uA

I_{sub2}		SYSCLK = 64KHz/128 @ILRCO, &HSE=0		13		uA
I_{WAT}	Watch 模式电流	WDT = 64KHz @ILRCO 在掉电模式下		2.5		uA
I_{MON1}	Monitor 模式电流	BOD0 使能在掉电模式下		10		uA
I_{PD1}	掉电模式电流			**0.1**	**5**	uA
BOD0 特性						
V_{BOD0}	BOD0 检测电平	T_A = -40℃ to +85℃	3.8(1)	4.0	4.2(1)	V
工作环境						
V_{PSR}	上电边沿速率	T_A = -40℃ to +85℃	0.05			V/ms
V_{OP1}	工作于 0~25MHz	T_A = -40℃ to +85℃	4.5		5.5	V
V_{OP2}	工作于 0~12MHz	T_A = -40℃ to +85℃	4.2		5.5	V

(1)数据基于特性结果，而不是产品测试结果

MA86L104

VDD = 3.3V±10%, VSS = 0V, T_A = 25 ℃ 并且 CPU 空运行，除非另外说明

符号	参数	测试条件	界限			单位
			最小	典型	最大	
输入/输出特性						
V_{IH1}	输入高电压(所有输入输出口)	除了 RST,P4.0,P4.1	2.0			V
V_{IH2}	输入高电压 (P4.0,P4.1)		2.4			V
V_{IH3}	输入高电压(复位脚/P3.6)		2.7			
V_{IL1}	输入低电压(所有输入输出口)	除了 RST,P4.0,P4.1			0.8	V
V_{IL2}	输入低电压(P4.0,P4.1)				0.8	V
V_{IL3}	输入低电压(复位脚/P3.6)				0.8	V
I_{IH}	输入高漏电流(所有输入输出口)	V_{PIN} = VDD		0	10	uA
I_{IL1}	逻辑电平 0 输入电流(P3 在准双向口，或输入脚内部上拉电阻)	V_{PIN} = 0.4V		10	30	uA
I_{IL2}	逻辑电平 0 输入电流 (所有输入口或漏极开路口)	V_{PIN} = 0.4V		0	10	uA
I_{H2L}	逻辑电平从 1 到 0 输入转换电流 (P3 在准双向口，或输入脚内部上拉电阻)	V_{PIN} =1.8V		120	250	uA
I_{OH1}	输出高时电流(P3 在准双向输入输出口模式下)	V_{PIN} =2.4V	40	80		uA
I_{OH2}	输出高时电流(所有推挽式输出口)	V_{PIN} =2.4V	8			mA
I_{OL1}	输出低时电流(所有输入输出口)	V_{PIN} =0.4V	8			mA
R_{RST}	内置复位下拉电阻			93		Kohm
功　耗						
I_{OP1}	正常模式下工作电流	SYSCLK = 24MHz @IHRCO		8.8		mA
I_{OP2}		SYSCLK = 12MHz @IHRCO		5.7		mA
I_{OP3}		SYSCLK = 6MHz @IHRCO&HSE=0		4		mA
I_{OP4}		SYSCLK = 3MHz @IHRCO&HSE=0		2.4		mA
I_{OP5}		SYSCLK = 24MHz @XTAL		8.9		mA
I_{OP6}		SYSCLK = 12MHz @XTAL		5.6		mA
I_{OP7}		SYSCLK = 6MHz @XTAL&HSE=0		3.8		mA
I_{OP8}		SYSCLK = 2MHz @XTAL&HSE=0		1.6		mA
I_{OPS1}	低速模式电流	SYSCLK = 24MHz/128 @IHRCO		0.86		mA
I_{IDLE1}	闲置模式电流	SYSCLK = 24MHz @IHRCO		2.1		mA
I_{IDLE2}	闲置模式电流	SYSCLK = 24MHz @XTAL		2.3		mA
I_{IDLE3}	闲置模式电流	SYSCLK = 24MHz/128 @IHRCO		0.77		mA
I_{IDLE4}	闲置模式电流	SYSCLK = 24MHz/128 @XTAL		0.88		mA
I_{IDLE5}	闲置模式电流	SYSCLK = 64KHz @ILRCO		20		uA
I_{sub1}	子时钟模式工作电流	SYSCLK = 64KHz @ILRCO, &HSE=0		55		uA
I_{sub2}		SYSCLK = 64KHz/128 @ILRCO, &HSE=0		13		uA
I_{WAT}	Watch 模式电流	WDT = 64KHz @ILRCO 在掉电模式下		2.5		uA

I_{MON1}	Monitor 模式电流	BOD0 使能在掉电模式下		10		uA
I_{PD1}	掉电模式电流			**0.1**	**5**	uA
BOD0 特性						
V_{BOD0}	BOD0 检测电平	T_A = -40℃ to +85℃	2.45(1)	2.6	2.75[1)]	V
工作环境						
V_{PSR}	上电边沿速率	T_A = -40℃ to +85℃	0.05			V/ms
V_{OP1}	工作于 0~25MHz	T_A = -40℃ to +85℃	2.7		3.6	V
V_{OP2}	工作于 0~12MHz	T_A = -40℃ to +85℃	2.4		3.6	V

(1)数据基于特性结果，而不是产品测试结果

22.3. 外部时钟特性

MA86E104:

VDD = 4.5V ~ 5.5V, VSS = 0V, T_A = -40℃ to +85℃，除非另外说明

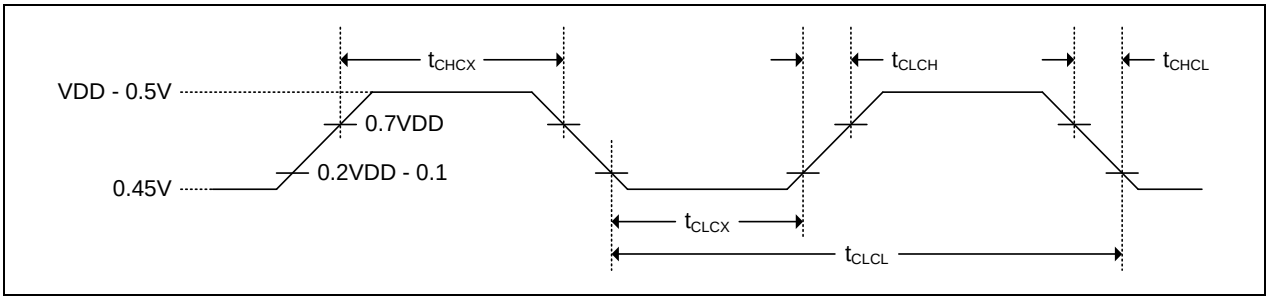
符号	参数	晶振				单位
		晶振模式		ECKI 模式		
		最小.	最大	最小.	最大	
$1/t_{CLCL}$	晶振频率	2	25	0	25	MHz
$1/t_{CLCL}$	晶振频率 (VDD = 4.2V ~ 5.5V)	2	12	0	12	MHz
t_{CLCL}	时钟周期	40		40		ns
t_{CHCX}	高时间	0.4T	0.6T	0.4T	0.6T	t_{CLCL}
t_{CLCX}	低时间	0.4T	0.6T	0.4T	0.6T	t_{CLCL}
t_{CLCH}	上升时间		5		5	ns
t_{CHCL}	下降时间		5		5	ns

MA86L104

VDD = 2.7V ~ 3.6V, VSS = 0V, T_A = -40℃ to +85℃，除非另外说明

符号	参数	晶振				单位
		晶振模式		ECKI 模式		
		最小.	最大	最小.	最大	
$1/t_{CLCL}$	晶振频率	2	25	0	25	MHz
$1/t_{CLCL}$	晶振频率 (VDD = 2.4V ~ 3.6V)	2	12	0	12	MHz
t_{CLCL}	时钟周期	40		40		Ns
t_{CHCX}	高时间	0.4T	0.6T	0.4T	0.6T	t_{CLCL}
t_{CLCX}	低时间	0.4T	0.6T	0.4T	0.6T	t_{CLCL}
t_{CLCH}	上升时间		5		5	Ns
t_{CHCL}	下降时间		5		5	Ns

图 22–1. 外部时钟波形



22.4. IHRCO 特性

MA86E104:

参数	测试环境	界限			单位
		最小	典型	最大	
支持电压		4.5		5.5	V
IHRCO 频率	TA = +25℃, AFS = 0		24		MHz
	TA = +25℃, AFS = 1		22.118		MHz
IHRCO 频率误差 (工厂校对)	TA = +25℃	-1.0		+1.0	%
	TA = -40℃ to +85℃	-4.0(1)		+4.0(1)	%
IHRCO 启动时间	TA = -40℃ to +85℃			32(1)	us
IHRCO 功耗	TA = +25℃, VDD=5.0V		770		uA

(1)数据基于特性结果，而不是产品测试结果

MA86L104:

参数	测试环境	界限			单位
		最小	典型	最大	
支持电压		2.7		3.6	V
IHRCO 频率	TA = +25℃, AFS = 0		24		MHz
	TA = +25℃, AFS = 1		22.118		MHz
IHRCO 频率误差 (工厂校对)	TA = +25℃	-1.0		+1.0	%
	TA = -40℃ to +85℃	-4.0(1)		+4.0(1)	%
IHRCO 启动时间	TA = -40℃ to +85℃			32(1)	us
IHRCO 功耗	TA = +25℃, VDD=5.0V		770		uA

(1)数据基于特性结果，而不是产品测试结果

22.5. ILRCO 特性

MA86E104:

参数	测试环境	界限			单位
		最小	典型	最大	
支持电压		4.2		5.5	V
ILRCO 频率	TA = +25℃		64		KHz
ILRCO 频率误差	TA = +25℃	-30(1)		+30(1)	%
	TA = -40℃ to +85℃	-50(1)		+50(1)	%

(1)数据基于特性结果，而不是产品测试结果

MA86L104:

参数	测试环境	界限			单位
		最小	典型	最大	
支持电压		2.4		3.6	V
ILRCO 频率	TA = +25℃		64		KHz
ILRCO 频率误差	TA = +25℃	-30(1)		+30(1)	%
	TA = -40℃ to +85℃	-50(1)		+50(1)	%

(1)数据基于特性结果，而不是产品测试结果

22.6. Flash 特性

MA86E104:

参数	测试环境	界限			单位
		最小	典型	最大	
支持电压	TA = -40℃ to +85℃	4.2		5.5	V
Flash 写 (擦除/编程) 电压	TA = -40℃ to +85℃	4.5		5.5	V
Flash 擦除/编程 周期	TA = -40℃ to +85℃	100			次
Flash 数据保留	TA = +25℃	100			年

MA86L104:

参数	测试环境	界限			单位
		最小	典型	最大	
支持电压	TA = -40℃ to +85℃	2.4		3.6	V
Flash 写 (擦除/编程) 电压	TA = -40℃ to +85℃	2.7		3.6	V
Flash 擦除/编程 周期	TA = -40℃ to +85℃	100			次
Flash 数据保留	TA = +25℃	100			年

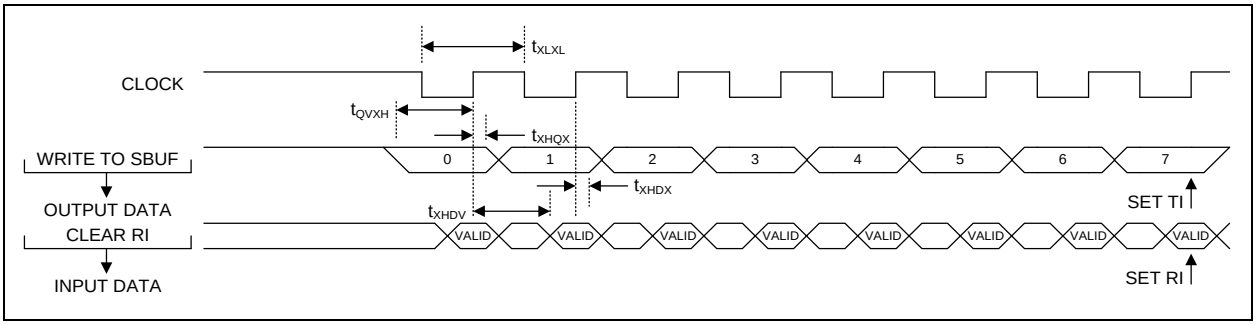
22.7. 串行口时序特性

MA86E104: VDD = 5.0V±10%, VSS = 0V, T_A = -40℃ to +85℃，除非另外说明

MA86L104: VDD = 3.3V±10%, VSS = 0V, T_A = -40℃ to +85℃，除非另外说明

Symbol	参数	URM0X6 = 0		URM0X6 = 1		单位
		最小.	最大	最小.	最大	
t_{XLXL}	串行口时钟周期	12T		2T		T_{SYSCLK}
t_{QVXH}	设置输出数据到时钟上升沿	10T-20		T-20		ns
t_{XHQX}	上升沿后保持输出数据	T-10		T-10		ns
t_{XHDX}	上升沿后保持输入数据	0		0		ns
t_{XHDV}	时钟上升沿到输入数据有效		10T-20		2T-20	ns

图 22–2. 移位寄存器模式时序图



23.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

助记符	描述	长度（字节）	周期（时钟）
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	内置 XRM（8 位地址）的数据送入累加器中	1	**不支持**
MOVX A,@DPTR	内置 XRM（16 位地址）的数据送入累加器中	1	**不支持**
MOVX @Ri,A	累加器的数据送入内置 XRM（8 位地址）中	1	**不支持**
MOVX @DPTR,A	累加器的数据送入内置 XRM（16 位地址）中	1	**不支持**
MOVX A,@Ri	外部 XRM（8 位地址）的数据送入累加器中	1	**不支持**
MOVX A,@DPTR	外部 XRM（16 位地址）的数据送入累加器中	1	**不支持**
MOVX @Ri,A	累加器的数据送入外部 XRM（8 位地址）中	1	**不支持**
MOVX @DPTR,A	累加器的数据送入外部 XRM（16 位地址）中	1	**不支持**
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
算术运算 ***S***			
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
DEC A	DPTR ← DPTR +1	1	2
DEC Rn	Acc ← Acc - 1	1	3
DEC direct	Rn ← Rn -1	2	4
DEC @Ri	direct ← direct -1	1	4
INC DPTR	Ri ← Ri -1	1	1
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	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

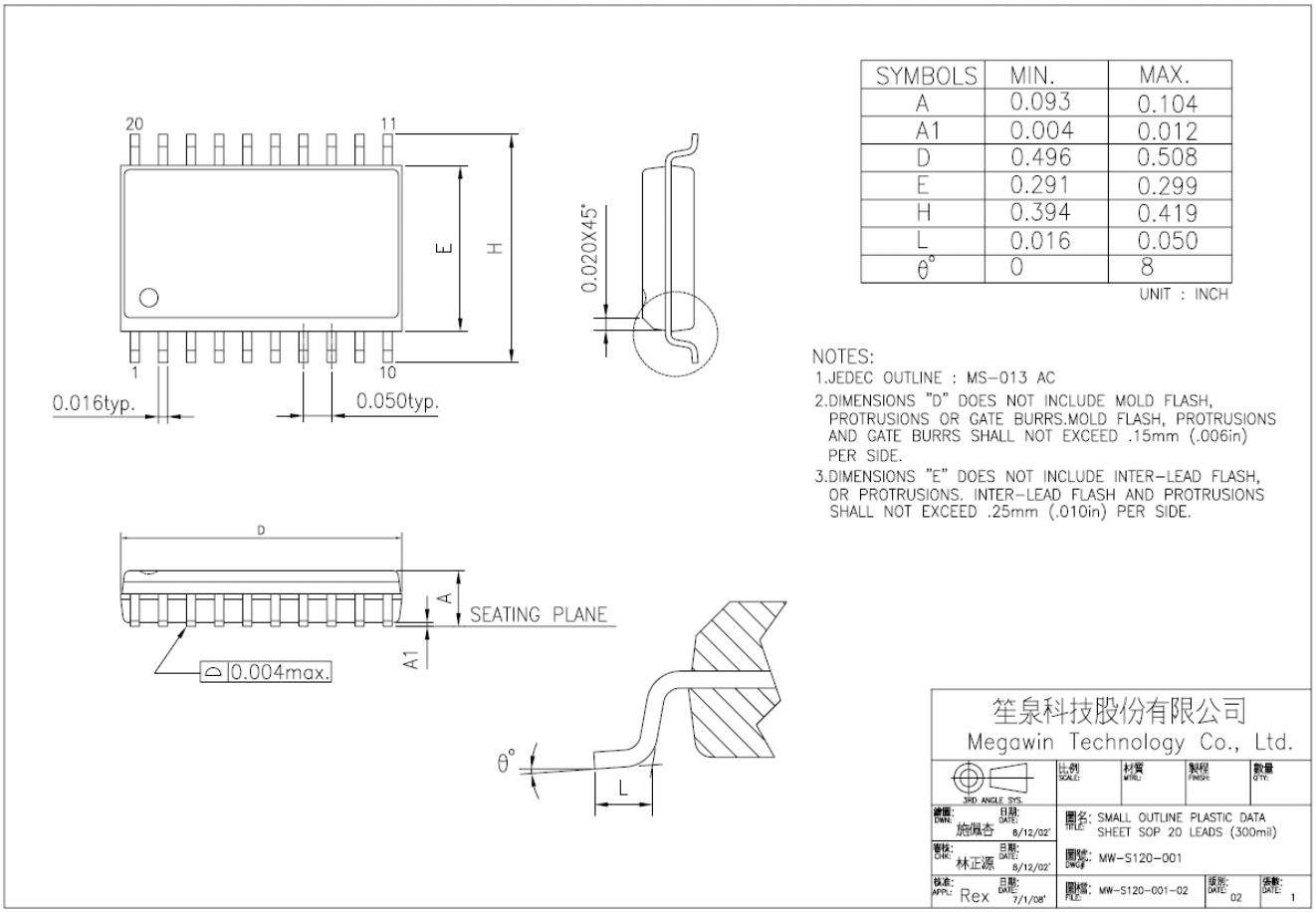
ORL direct,#data	直接地址单元中的内容和立即数相“或”	3	4
XRL A,Rn	累加器和寄存器Rn中的内容相“异或”	1	2
XRL A,direct	累加器和直接地址单元中的内容相“异或”	2	3
XRL A,@Ri	累加器和工作寄存器Ri指向的地址单元中的内容相“异或”	1	3
XRL A,#data	累加器和立即数相“异或”	2	2
XRL direct,A	直接地址单元中的内容和累加器相“异或”	2	4
XRL direct,#data	直接地址单元中的内容和立即数相“异或”	3	4
CLR A	累加器内容清“0”	1	1
CPL A	累加器按位取反	1	2
RL A	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
位逻辑运算			
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
位逻辑跳转			
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
程序跳转			
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
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

24. 封装尺寸

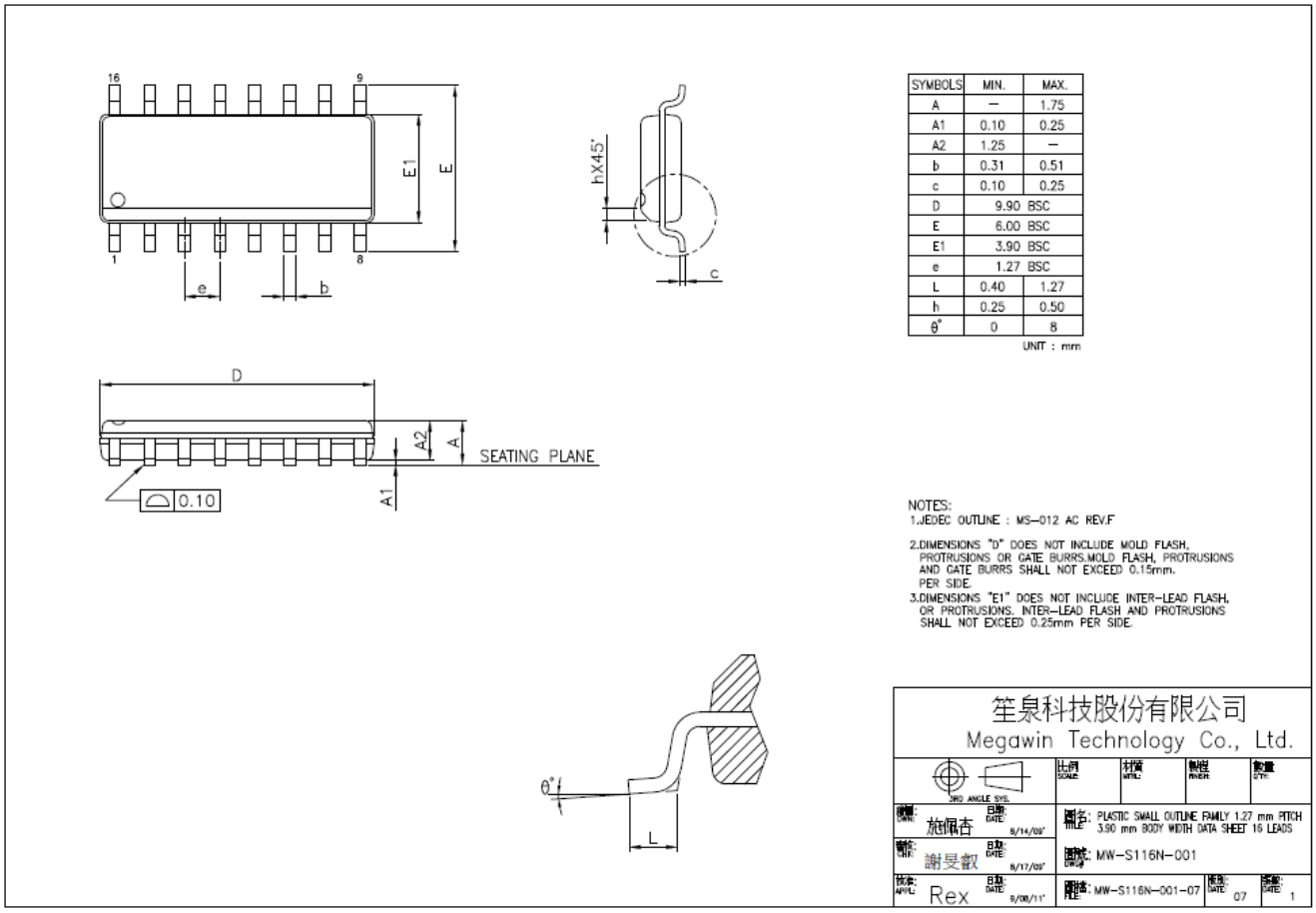
24.1. SOP-20

Figure 24–1. SOP-20



24.2. SOP-16

Figure 24–2. SOP-16



SYMBOLS	MIN.	MAX.
A	—	1.75
A1	0.10	0.25
A2	1.25	—
b	0.31	0.51
c	0.10	0.25
D	9.90 BSC	
E	6.00 BSC	
E1	3.90 BSC	
e	1.27 BSC	
L	0.40	1.27
h	0.25	0.50
θ°	0	8

UNIT : mm

NOTES:
1.JEDEC OUTLINE : MS-012 AC REV.F

2.DIMENSIONS "D" DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BURRS.MOLD FLASH, PROTRUSIONS AND GATE BURRS SHALL NOT EXCEED 0.15mm. PER SIDE.

3.DIMENSIONS "E1" DOES NOT INCLUDE INTER-LEAD FLASH, OR PROTRUSIONS. INTER-LEAD FLASH AND PROTRUSIONS SHALL NOT EXCEED 0.25mm PER SIDE.

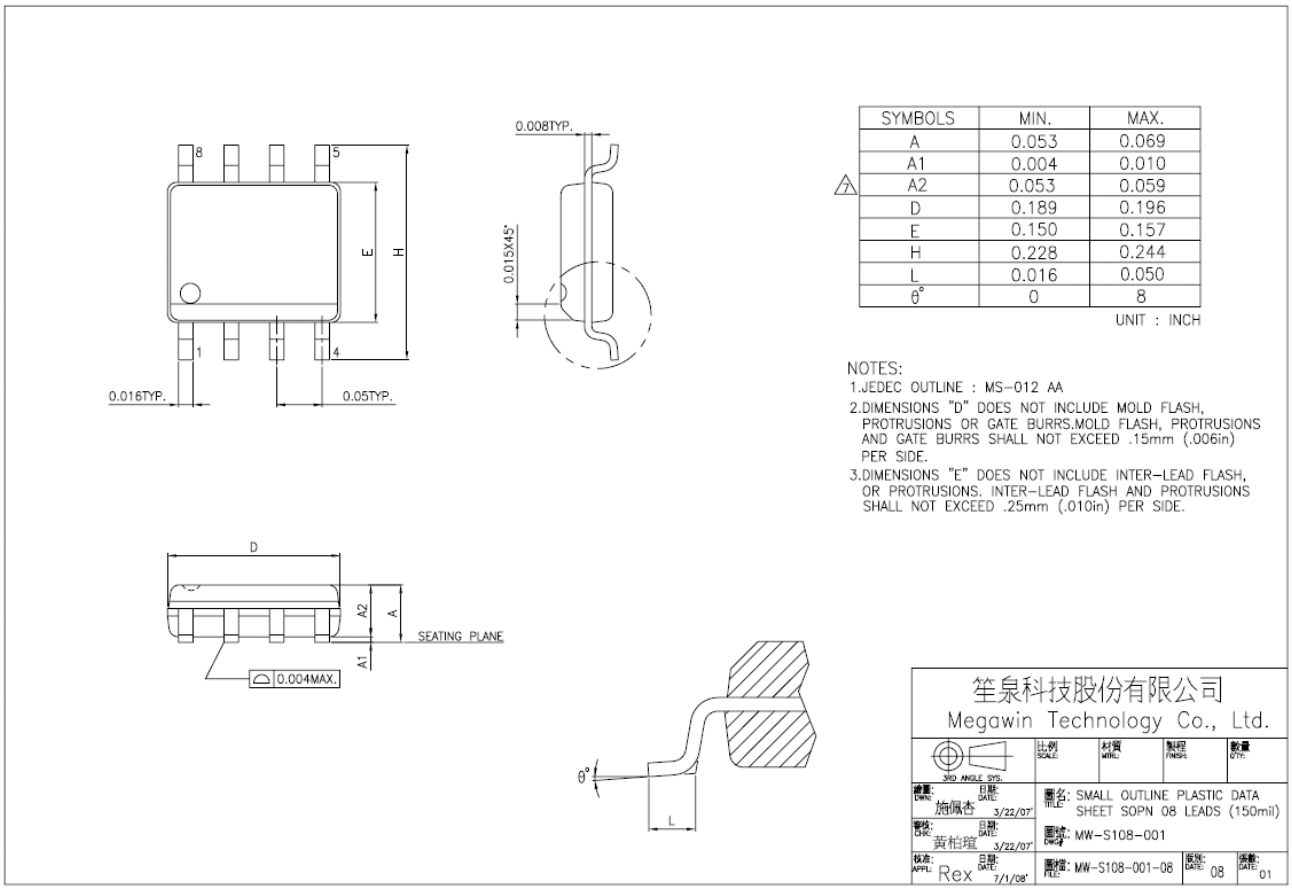
笙泉科技股份有限公司
Megawin Technology Co., Ltd.

3RD ANGLE SYS.

設計: 施佩杏	日期: 8/14/09'	圖名: PLASTIC SMALL OUTLINE FAMILY 1.27 mm PITCH 3.90 mm BODY WIDTH DATA SHEET 16 LEADS	
審核: 謝旻叡	日期: 8/17/09'	圖號: MW-S116N-001	
核准: Rex	日期: 9/08/11'	圖檔: MW-S116N-001-07	版次: 07

24.3. SOP-8

Figure 24–2. SOP-8



25. 修订历史

表 25-1 修订历史

版本	描述	日期
V1.21	1. MA86E/L104 初次版本	2011/05/13
V1.22	新增 8-pin 封装管脚	2011/06/07
V1.24	新增管脚封装型号	2011/06/21
V1.30	新增详细说明	2012/02/07
V1.40	补充详细说明	2012/08/01
V1.41	波特率公式内，SMOD 描述错误，修正为 SMOD1	2013/01/24
V1.42	修改 Flash 写/擦次数说明	2013/03/27
V1.43	修改图档 MG 错误标示说明 新增 Flash 架构图 增加 PageP 例程说明	2013/04/11
A1.0	修正格式建立新版本	2013/04/24
A1.2	增加例程，修正 IE 寄存器位置错误	2013/10/09
A1.3	修正 SOP-16 封装尺寸图	2015/05/27

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。