8051-Based MCU

MG82F6B08/
MG82F6B001/
MG82F6B104
数据手册

版本: 1.05

2023/07 version 1.05

特性

- 1-T 80C51 中央处理器
  - 带有堆栈溢出警告检测器
- MG82F6B08 / 6B001/ 6B104 内含 8K 字节闪存 ROM 和 512 字节 EEPROM
  - ISP 空间可以选择为 **0.5KB/1.0KB~3.5KB**
  - 512 字节 EEPROM 擦/写次数: **100,000**
  - 灵活的 IAP 大小空间设置
  - 密码保护程序区访问
  - Flash 擦/写次数：20,000
  - Flash 数据保留时间：100 年 25° C
  - **MG82F6B08 / 6B001/ 6B104 默认空间配置**
    - AP 程序空间(6.5KB, 0000h~19FFh)
    - IAP 数据空间(无)
    - ISP 引导码空间 (1.5KB, 1A00h~1FFFh)
- RAM: **1K** 字节
  - 片内 256 字节高速缓存
  - 片内 **768** 字节扩展 RAM(XRAM)
  - 访问 XRAM 支持页面选择
- 双数据指针
- 中断控制
  - **12** 个中断源，4 级优先级
  - **2** 个带滤波器的外部中断 nINT0、nINT1
  - 所有的外部中断支持高/低或上升/下降沿触发
- **MG82F6B08 / 6B001/ 6B104** 共有 **7/8** 个定时器
  - RTC 定时器和 WDT 定时器
  - 定时器 0、定时器 1、定时器 2
  - PCA0，可编程计数器阵列 0
  - S0 BRG
  - 如果定时器 2 用于分立模式，**MG82F6B08 / 6B001/ 6B104** 则共有 **8** 个定时器
- **3** 个 16-位定时/计数，定时器 0、定时器 1、定时器 2
  - 定时器 0~2 的同步运行使能（具有同样的停止和重载功能）
  - 定时器 2 有新的 6 个操作模式，它们有着 8 个时钟源和 8 个捕获源
  - 定时器 2 可以分立位两个 8 位定时器
  - 时钟输出(CCO)在 T2CKO
  - 定时器 0~2 支持 PWM 模式
  - 定时器 2 支持占空比捕获功能
- **1** 个可编程 16 位计数/定时阵列(PCA0)有着 **4** 个比较/PWM 模块
  - PCA0 有 **4** 个 CCP(捕获/比较/PWM)模块
  - 可重载 16 位基准计数器支持可变长度的 PWM
  - 捕获模式，16 位软件定时器模式和高速输出模式
  - 缓冲捕获模式可以监控窄脉冲输入
  - 支持 POEM0
  - 8/10/12/16 位可调 PWM 模式，可被配置成：
    - 最高 **4** 通道无缓冲 10/12/16 位 PWM，或
    - 最高 **4** 通道带缓冲 2~8 位 PWM，或
    - 最高 **2** 通道带缓冲 9~16 位 PWM
  - PCA0 PWM 模式 0~3 具有死区控制，中止控制和中心对齐选择

- 8 个输入的键盘中断(KBI)
- 10 位单端 ADC
  - 可编程转换率高达 **666K** sps
  - **MG82F6B08 / 6B001/ 6B104** 有 **6** 通道外部输入通道和 2 个内部输入通道：(IVR/1.4V)、VSS
  - 支持 ADC 结果的窗口检测功能
- 内置参考电压 (IVR14)
- 模拟比较器 0
  - 在 ACNI0 可选内部参考电压(IVR/1.4V)
  - 可从掉电或闲置状态唤醒
  - 可做滤波和输出到内部定时器捕获
- 增强型 UART (S0)
  - 帧错误侦测
  - 自动地址识别
  - 最大波特率可达 **2MHz**
  - 模式 4 支持 SPI 主机，SPICLK 速率可达 **4MHz**
  - 内置波特率发生器(S0BRG)支持不同波特率的 TX 或 RX
  - 模式 5 支持具有自动波特率的 LIN 总线协议
- 1 个主/从 SPI 串行接口
  - 高达 **2** 个 SPI 主机，包括串口 S0 模式 4
  - 在 SPI 从机模式下支持菊花链功能
- **2** 个主/从两线串行接口：TWI0/ I2C0 和 STWI (SI2C)
  - **1** 个主/从硬件引擎：TWI0/ I2C0
  - I2C0 主机模式可达 1MHz 和 I2C0 从机模式可达 400KHz
  - 1 个软件 TWI/I2C、STWI/SI2C、开始/停止串行信号侦测(SID)
  - I2C0 支持多从机地址检测
- 可编程看门狗定时器(WDT)，时钟来源为 ILRCO 或 SYSCLK/12
  - 通过 CPU 或上电一次性使能
  - WDT 溢出可中断 CPU 或复位 CPU
  - 掉电模式(watch 模式)下支持看门狗(WDT)功能，用于自动唤醒功能
- 实时时钟（RTC）模块，时钟源来自 ECKI、ILRCO、WDTPS、WDTOF、SYSCLK 或 SYSCLK/12
  - 可编程中断周期从微秒到分钟唤醒
  - 21-位系统定时器
- 蜂鸣器功能
- 通用逻辑(GPL/CRC)
  - 位序颠倒功能
  - 16 位 CRC 引擎 (CCITT-16 多项式)
  - 支持 flash 容量的自动 CRC
  - 可编程 CRC 的初始种子功能
- 片上调试接口(OCD)
- 在 **10/8** 脚封装中最大 **8/6** 个通用输入输出口(GPIO)
  - P3 可以设置成准双向口模式，推挽输出模式，开漏集输出模式和仅输入模式
  - P0 和 P4 可以设置为推挽输出模式，开漏集输出模式
  - P4.7 共用 RST
  - 可编程通用输入输出口(GPIO)的驱动力
  - 每一个脚具有片内上拉使能

- 时钟源
  - 内部 **16MHz/22.12MHz** 振荡器(IHRCO)：工厂校准到±2%，典型值
  - 内部低功耗 32KHz RC 振荡器(ILRCO)
  - 外部时钟输入(ECKI)在 **P4.5**，可达到 **24MHz**
  - 内部 RC 振荡输出在 **P4.5**
- 两个低电压监测
  - BOD0：监测 **2.35V**
  - BOD1：选择监测电压为 **4.2V/3.6V/2.4V & 2.7V**
  - 中断 CPU 或复位 CPU
  - (BOD1)在掉电模式下唤醒 CPU
- 多种功耗模式：掉电模式、空闲模式、慢频模式、副频模式、RTC 模式、watch 模式和 monitor 模式
  - 所有的中断能唤醒空闲(IDLE)模式
  - **7/6 (MSOP10/SOP8)**只引脚的 **10** 个中断源能唤醒掉电模式
  - 慢频模式和副频模式支持低速 MCU 运转
  - RTC 模式在掉电模式下支持实时时钟(RTC)恢复 CPU
  - Watch 模式在掉电模式下支持看门狗(WDT)恢复 CPU
  - Monitor 模式在掉电模式下支持 BOD1 恢复 CPU
- 工作电压范围：2.4 V – 5.5V
- 工作频率范围：**16**MHz(最高)
  - 系统时钟最快 **24MHz** @ 2.4V – 5.5V
  - CPU 最快 **12MHz** @ 2.4V – 5.5V，**0 – 16MHz** @ 2.7V – 5.5V
- 16 字节唯一 ID 码
- 工作温度：
  - 工业级(-40° C 到 +85° C)*
- 封装：
  - SOP8: MG82F6B08AS8 (8K)
  - SOP8: MG82F6B001AS8 (8K)
  - SOP8: MG82F6B104AS8 (8K)
  - MSOP10: MG82F6B08AG10 (8K)

*: 抽样检测

目录

图表

表单

1. 概述

MG82F6B08 / 6B001/ 6B104是基于80C51的高效1-T结构的单芯片微处理器，每条指令需要1~7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，**MG82F6B08 / 6B001/ 6B104**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。

MG82F6B08 / 6B001/ 6B104有**8K**字节的内置Flash存储器用于保存代码。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。内部还提供了512字节EEPROM用于用户数据，该EEPROM独立于内嵌的8K FLASH空间，可被CPU软件或ICP修改，这些功能都由内建的电荷泵提供编程用的高压不需要外部提供高压。

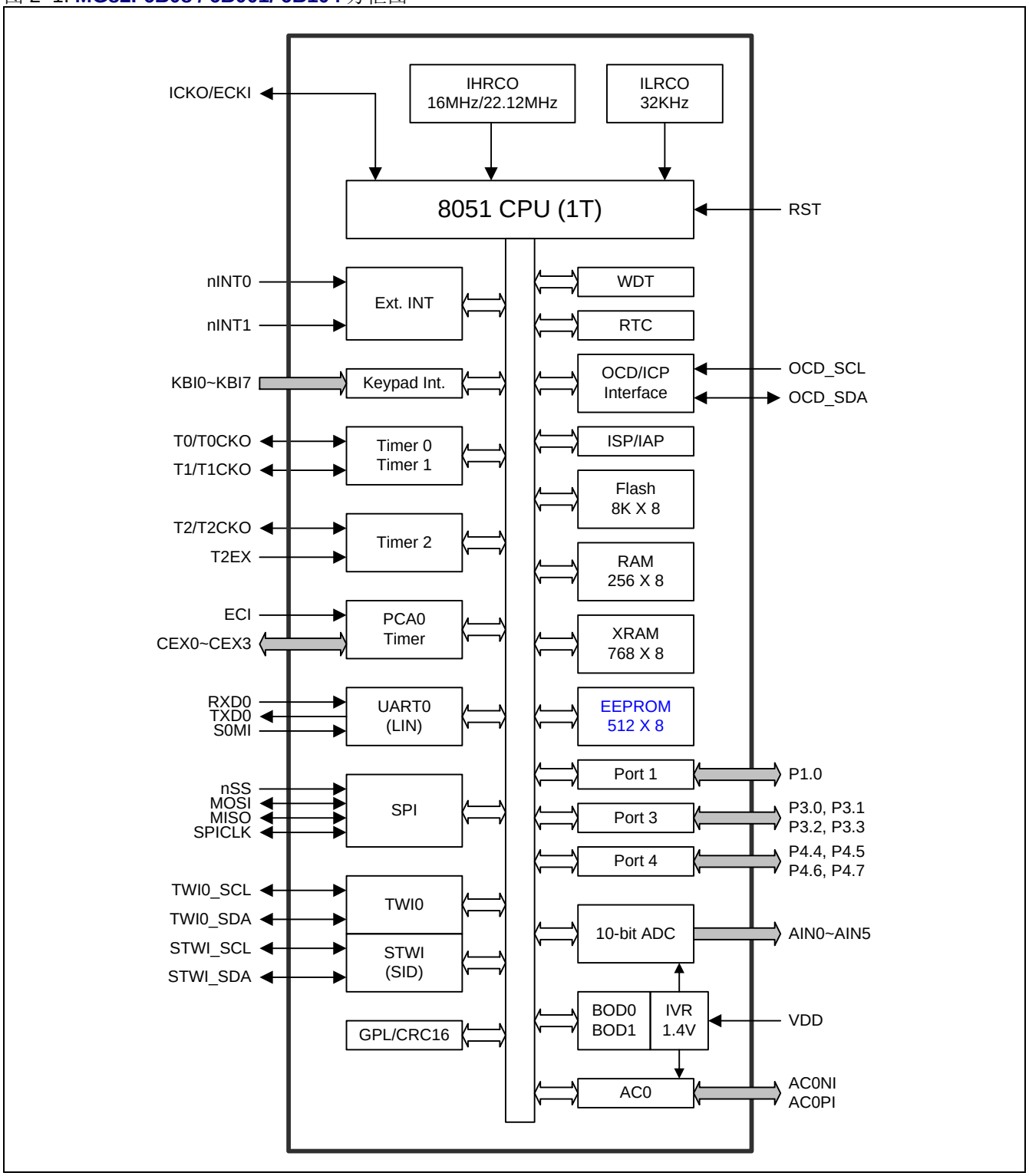
MG82F6B08 / 6B001/ 6B104有**2**个带高低电平触发选项的外部中断，一个多源4级中断控制，三个定时/计数器外，**MG82F6B08 / 6B001/ 6B104**有**6**个额外的I/O端口引脚，**768**字节外部数据存储器(XRAM)，**400Ksps**12位ADC，3个16位定时器，一个4通道带死区控制的PCA(PCA0)，一个8位SPI，二个TWI/ I2C (TWI0/ I2C0和STWI/ SI2C)，键盘中断，模拟比较器，看门狗定时器，实时时钟(RTC)模块，两个低电压监测器，一个高精度的内部振荡(IHRCO)，一个内部低速振荡(ILRCO)和一个增型的串口(UART0)用来促进多处理器的通讯，LIN总线模式及一个速度增强设备 (X2/X4 模式)。

MG82F6B08 / 6B001/ 6B104 有多种工作模式可以减少耗电量：空闲模式，掉电模式，慢频模式，副频模式，RTC 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。实时时钟(RTC)模式支持所有模式下的实时时钟功能，watch 模式，在掉电模式或空闲模式下 WDT 溢出作为一个自动定时器来唤醒 CPU。Monitor 模式，在掉电模式监测电压，当电压特别低的时候会复位。

另外，**MG82F6B08 / 6B001/ 6B104** 装配有笙泉独家的(OCD)接口可以用于在线仿真(ICE)，OCD 接口提供在片内和在系统不干扰调试并且不占用任何资源。支持 ICE 应用中的几个必须的操作例如：复位、全速、停止、单步、全速到光标和断点设置。软件开发期间使用者不需要使用任何的开发板或者传统的 ICE 上应用的插头转接器，使用者只需要连接好 OCD 接口， 这强有力的接口使得开发非常容易。

2. 方框图

图 2–1. **MG82F6B08 / 6B001/ 6B104** 方框图



3. 特殊功能寄存器

3.1. SFR 地址映射(页 0~F)

表 3–1. SFR 地址映射 (页 0~F)

		0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8	0	--	CH	CCAP0H	CCAP1H	CCAP2H	CCAP3H	--	--
	1			--	--				
	~F			--	--				
F0	0	*B*	PAOE	PCAPWM0	PCAPWM1	PCAPWM2	PCAPWM3	--	--
	1			--	--				
	~F			--	--				
E8	0	*P4*	CL	CCAP0L	CCAP1L	CCAP2L	CCAP3L	--	--
	1			--	--				
	~F			--	--				
E0	0~F	*ACC*	WDTCR	IFD	IFADRH	IFADRL	IFMT	SCMD	ISPCR
D8	0	*CCON*	CMOD	CCAPM0	CCAPM1	CCAPM2	CCAPM3	--	--
	1			--	--				
	~F			--	--				
D0	0	*PSW*	SIADR	SIDAT	SISTA	SICON	KBPATN	KBCON	KBMASK
	1		--	--	--	--			
	2		SIA2	SIA2M	--	--			
	~F		--	--	--	--			
C8	0	*T2CON*	T2MOD	RCAP2L	RCAP2H	TL2	TH2	CLRL	CHRL
	1	--	--	--	--	--	--		
	~F	--	--	--	--	--	--		
C0	0	--	XICFG	--	ADCFG0	ADCON0	ADCDL	ADCDH	CKCON0
	1		XICFG1		ADCFG1				
	2		--		ADCFG2				
	3				ADCFG3				
	4				ADCFG4				
	B		AUXIE0		ADCFG11				
	C		--		ADCFG12				
	D				ADCFG13				
	E				ADCFG14				
	F								
B8	0	*IP0L*	SADEN/ S0CR1	--	--	PWMCR	CRC0DA	RTCCR	CKCON1
	1					PDTCRA			
	~F		--			--			
B0	0	*P3*	P3M0	P3M1	P4M0	--	--	RTCTM	IP0H
	1				--	--			
	2				--	PDRVC0			
	3				--	PDRVC1			
	~F				--				
A8	0~F	*IE*	SADDR	--	--	SFRPI	EIE1	EIP1L	EIP1H
A0	0	--	AUXR0	AUXR1	AUXR2	AUXR3	--	--	--
	1					AUXR4			
	2					AUXR5			
	3					AUXR6			
	4					AUXR7			
	5					--			
	6					AUXR9			
	7					AUXR10			
	8					AUXR11			
	~F					--			
98	0	*S0CON*	S0BUF	S0BRT	S0BRC	S0CFG	S0CFG1	AC0CON	AC0MOD
	1~F	--	--	--	--	--	--	--	--
90	0	*P1*	P1M0	P1M1	--	--	--	BOREV	PCON1
	1			--	T2MOD1		TREN0		
	2			P4M1	--		TRLC0		
	3			--	--		TSPC0		
	8			P1FDC	--		--		
	9			--	--		--		
	A			P4FDC	--		--		
	~F			--	--		--		
88	0~F	*TCON*	TMOD	TL0	TL1	TH0	TH1	SFIE	XRPS
80	0~F	--	SP	DPL	DPH	SPSTAT	SPCON	SPDAT	PCON0
		0/8	**1/9**	**2/A**	**3/B**	**4/C**	**5/D**	**6/E**	**7/F**

注: 用户需要设置 SFRPI=0x00 ~ 0x0F 作为 SFR 的页访问。
(MCU 在中断时不会保留 SFRPI 的值。用户需要使用软件来保留 SFRPI 的值。)

SFRPI: SFR 页索引寄存器

SFR 页 = 0~F

SFR 地址 = 0xAC 复位值 = xxxx-0000

7	6	5	4	3	2	1	0
--	--	--	--	IDX3	IDX2	IDX1	IDX0
W	**W**	**W**	**W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~4：保留位。当 SFRPI 写入时，这些位软件必须写“0”

Bit 3~0: SFR 页索引

IDX[3:0]	可选页
0000	页 0
0001	页 1
0010	页 2
0011	页 3
……	……
……	……
……	……
1111	页 F

3.2. SFR 位分配(页 0~F)

表 3–2. SFR 位分配(页 0~F)

符号	描述	地址 (HEX)	页 (HEX)	位地址及符号								复位值
				位-7	位-6	位-5	位-4	位-3	位-2	位-1	位-0	
SP	堆栈指针	81	*0~F*	.7	.6	.5	.4	.3	.2	.1	.0	00000111
DPL	数据指针低 8 位	82	*0~F*	.7	.6	.5	.4	.3	.2	.1	.0	00000000
DPH	数据指针高 8 位	83	*0~F*	.7	.6	.5	.4	.3	.2	.1	.0	00000000
SPSTAT	SPI 状态寄存器	84	*0~F*	SPIF	WCOL	THRF	SPIBSY	MODF	--	--	SPR2	00000000
SPCON	SPI 控制寄存器	85	*0~F*	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	00000100
SPDAT	SPI 数据寄存器	86	*0~F*	.7	.6	.5	.4	.3	.2	.1	.0	00000000
PCON0	电源控制寄存器 0	87	*0~F*	SMOD1	SMOD0	GF	POF0	GF1	GF0	PD	IDL	00010000
TCON	*定时器控制寄存器*	*88*	*0~F*	*TF1*	*TR1*	*TF0*	*TR0*	*IE1*	*IT1*	*IE0*	*IT0*	*00000000*
TMOD	定时器模式寄存器	89	*0~F*	T1GATE	T1C/T	T1M1	T1M0	T0GATE	T0C/T	T0M1	T0M0	00000000
TL0	定时器 0 低 8 位	8A	*0~F*	.7	.6	.5	.4	.3	.2	.1	.0	00000000
TL1	定时器 1 低 8 位	8B	*0~F*	.7	.6	.5	.4	.3	.2	.1	.0	00000000
TH0	定时器 0 高 8 位	8C	*0~F*	.7	.6	.5	.4	.3	.2	.1	.0	00000000
TH1	定时器 1 高 8 位	8D	*0~F*	.7	.6	.5	.4	.3	.2	.1	.0	00000000
SFIE	系统标志中断使能	8E	*0~F*	SIDFIE	1	1	RTCFIE	SPWIE	BOF1IE	BOF0IE	WDTFIE	01100000
XRPS	XRAM 页选择	8F	*0~F*	0	0	0	0	0	0	.1	.0	xxxxxx00
P1	*端口 1*	*90*	*0~F*	--	--	--	--	--	--	--	*P1.0*	*11111111*
P1M0	P1 模式寄存器 0	91	*0~F*	--	--	--	--	--	--	--	P1M0.0	00000000
P1M1	P1 模式寄存器 1	92	0	--	--	--	--	--	--	--	P1M1.0	11111111
P4M1	P4 模式寄存器 1	92	2	P4M1.7	P4M1.6	P4M1.5	P4M1.4	--	--	--	--	11111111
P3FDC	P3 快速驱动控制	92	7	--	--	--	--	P3FDC.3	P3FDC.2	P3FDC.1	P3FDC.0	00000000
P1FDC	P1 快速驱动控制	92	8	--	--	--	--	--	--	--	P1FDC.0	00000000
P4FDC	P4 快速驱动控制	92	A	--	P4FDC.6	P4FDC.5	P4FDC.4	--	--	--	--	00000000
T2MOD1	T2 模式 1 寄存器	93	1	TL2CS	TF2IG	TL2IS	T2CKS	T2MS1	CP2S2	CP2S1	CP2S0	00000000
TREN0	定时器使能寄存器 0	95	1	--	--	TR2LE	--	--	TR2E	TR1E	TR0E	00000000
TRLC0	定时器重载控制寄存器 0	95	2	--	--	TL2RLC	--	--	T2RLC	T1RLC	T0RLC	00000000
TSPC0	定时器停止控制寄存器 0	95	3	--	--	TL2SC	--	--	T2SC	T1SC	T0SC	00000000
BOREV	位序颠倒	96	0~F	.7	.6	.5	.4	.3	.2	.1	.0	00000000
PCON1	电源控制寄存器 1	97	0~F	SWRF	EXRF	0	RTCF	SPWF	BOF1	BOF0	WDTF	00000000
S0CON	*串口 0 控制寄存器*	*98*	*0*	*SM00 /FE*	*SM10*	*SM20*	*REN0*	*TB80*	*RB80*	*TI0*	*RI0*	*00000000*
S0BUF	串口 0 缓存	99	0	.7	.6	.5	.4	.3	.2	.1	.0	xxxxxxxx
S0BRT	S0 波特率定时器	9A	0	.7	.6	.5	.4	.3	.2	.1	.0	00000000
S0BRC	S0 波特率计数器	9B	0	.7	.6	.5	.4	.3	.2	.1	.0	00000000
S0CFG	S0 配置寄存器	9C	0	URTS	SMOD2	URM0X3	SM30	S0DOR	BTI	UTIE	SMOD3	00001000
S0CFG1	S0 配置寄存器 1 (LINCFG)	9D	0	SBF0	TXER0	S0SB16	ATBR0	TXRX0	SYNC0	--	--	000000xx
AC0CON	AC0 控制寄存器	9E	0	AC0LP	AC0PDX	AC0OUT	AC0F	AC0EN	AC0INV	AC0M1	AC0M0	00x00000
AC0MOD	AC0 模式寄存器	9F	0	0	0	0	0	NVRL	AC0FLT	0	0	00000000
AUXR0	辅助寄存器 0	A1	*0~F*	P45OC1	P45OC0	0	PBKF	0	0	INT1H	INT0H	00000000
AUXR1	辅助寄存器 1	A2	*0~F*	0	0	CRCDS1	CRCDS0	0	0	0	DPS	00000000
AUXR2	辅助寄存器 2	A3	*0~F*	STAF	STOF	0	C0PLK	T1X12	T0X12	T1CKOE	T0CKOE	00000000
AUXR3	辅助寄存器 3	A4	0	0	T0PS0	BPOC1	BPOC0	S0PS0	TWIPS1	TWIPS0	T0XL	00000000
AUXR4	辅助寄存器 4	A4	1	T2PS1	T2PS0	0	T1PS0	0	0	AC0OE	AC0FLT1	00000000
AUXR5	辅助寄存器 5	A4	2	0	C0IC2S0	0	0	C0PS1	C0PS0	ECIPS0	C0COPS	00000000
AUXR6	辅助寄存器 6	A4	3	0	0	0	0	0	T2FCS	SnMIPS	S0COPS	00000000
AUXR7	辅助寄存器 7	A4	4	1	1	C0CKOE	SPI0M0	0	0	0	0	11000000
AUXR9	辅助寄存器 9	A4	6	0	SIDPS0	T1G1	T0G1	C0FDC1	C0FDC0	0	0	00000000
AUXR10	辅助寄存器 10	A4	7	1	AC0HC0	0	SPIPS0	S0PS1	SPFACE	TWICF	0	11000000
AUXR11	辅助寄存器 11	A4	8	P30AM	P33AM	0	0	0	POEM0	C0M0	C0OFS	00000000
IE	*中断使能*	*A8*	*0~F*	*EA*	*0*	*ET2*	*ES0*	*ET1*	*EX1*	*ET0*	*EX0*	*00000000*
SADDR	从机地址	A9	*0~F*	.7	.6	.5	.4	.3	.2	.1	.0	00000000
SFRPI	SFR 页索引	AC	*0~F*	--	--	--	--	IDX3	IDX2	IDX1	IDX0	xxxx0000
EIE1	扩展中断使能 1	AD	*0~F*	EAC0	ETWI0	EKB	--	ESF	EPCA	EADC	ESPI	00000000
EIP1L	扩展中断优先级 1 低	AE	*0~F*	PAC0L	PTWI0L	PKBL	--	PSFL	PPCAL	PADCL	PSPIL	00000000
EIP1H	扩展中断优先级 1 高	AF	*0~F*	PAC0H	PTWI0H	PKBH	--	PSFH	PPCAH	PADCH	PSPIH	00000000
P3	*端口 3*	*B0*	*0~F*	--	--	--	--	*P3.3*	*P3.2*	*P3.1*	*P3.0*	*11111111*
P3M0	P3 模式寄存器 0	B1	*0~F*	--	--	--	--	P3M0.3	P3M0.2	P3M0.1	P3M0.0	00000000
P3M1	P3 模式寄存器 1	B2	*0~F*	--	--	--	--	P3M1.3	P3M1.2	P3M1.1	P3M1.0	00000000
P4M0	P4 模式寄存器 0	B3	0	P4M0.7	P4M0.6	P4M0.5	P4M0.4	--	--	--	--	10110000
PDRVC0	端口驱动控制 0	B4	2	--	P3DC0	--	--	--	P1DC0	--	--	00000000
PDRVC1	端口驱动控制 1	B4	3	0	--	--	--	--	--	P4DC1	--	00000000
RTCTM	RTC 定时器寄存器	B6	*0~F*	RTCCS1	RTCCS0	RTCCT5	RTCCT4	RTCCT3	RTCCT2	RTCCT1	RTCCT0	01111111
IP0H	中断优先级 0 高	B7	*0~F*	--	--	PT2H	PSH	PT1H	PX1H	PT0H	PX0H	00000000
IP0L	*中断优先级 0 低*	*B8*	*0~F*	--	--	*PT2L*	*PSL*	*PT1L*	*PX1L*	*PT0L*	*PX0L*	*00000000*

符号	描述	地址 (HEX)	页 (HEX)	位地址及符号								复位值
				位-7	位-6	位-5	位-4	位-3	位-2	位-1	位-0	
SADEN	从机地址屏蔽	B9	*0~F*	.7	.6	.5	.4	.3	.2	.1	.0	00000000
S0CR1	S0 控制 1	B9	*0~F*	S0TR	S0TX12	S0TCK	S0RCK	S0CKOE	ARTE	--	S0BC0	00000000
PWMCR	PWM 控制寄存器	BC	0	PCAE	EXDT	PBKM	PBKE1.1	PBKE1.0	PBKE0.2	PBKE0.1	PBKE0.0	00000000
PDTCRA	PWM 死区控制寄存器 A	BC	1	DTPS1	DTPS0	DT.5	DT.4	DT.3	DT.2	DT.1	DT.0	00000000
CRC0DA	CRC0 数据端口	BD	0~F	.7	.6	.5	.4	.3	.2	.1	.0	00000000
RTCCR	RTC 控制寄存器	BE	0~F	RTCE	RTCO	RTCRL5	RTCRL4	RTCRL3	RTCRL2	RTCRL1	RTCRL0	00111111
CKCON1	时钟控制 1	BF	0~F	--	--	--	--	OSCSTA3	OSCSTA2	OSCSTA1	OSCSTA0	00000000
XICFG	外部中断配置	C1	0	INT1IS1	INT1IS0	INT0IS1	INT0IS0	--	--	X1FLT	X0FLT	00000000
XICFG1	外部中断配置 1	C1	1	--	--	--	--	--	--	X1FLT1	X0FLT1	00000000
AUXIE0	外部中断使能 0	C1	B	0	0	EPIE	CFIE	0	0	0	0	00000000
ADCFG0	ADC 配置 0	C3	0	ADCKS2	ADCKS1	ADCKS0	ADRJ	ACHS	SMPF	ADTM1	ADTM0	00000000
ADCFG1	ADC 配置 1	C3	1	IGADCI	EADCWI	SMPFIE	SIGN	AOS.3	AOS.2	AOS.1	AOS.0	00000000
ADCFG2	ADC 配置 2	C3	2	SHT.7	SHT.6	SHT.5	SHT.4	SHT.3	SHT.2	SHT.1	SHT.0	00000000
ADCFG3	ADC 配置 3	C3	3	ADPS1	ADPS0	--	--	ARES1	ARES0	--	--	01000000
ADCFG4	ADC 配置 4	C3	4	--	ADWM0	ADTM3	ADTM2	--	--	--	--	00000000
ADCFG11	ADC 配置 11	C3	B	WHB.3	WHB.2	WHB.1	WHB.0	1	1	1	1	11111111
ADCFG12	ADC 配置 12	C3	C	WHB.11	WHB.10	WHB.9	WHB.8	WHB.7	WHB.6	WHB.5	WHB.4	11111111
ADCFG13	ADC 配置 13	C3	D	WLB.3	WLB.2	WLB.1	WLB.0	0	0	0	0	00000000
ADCFG14	ADC 配置 14	C3	E	WLB.11	WLB.10	WLB.9	WLB.8	WLB.7	WLB.6	WLB.5	WLB.4	00000000
ADCON0	ADC 控制 0	C4	0~F	ADCEN	ADCWI	CHS3	ADCI	ADCS	CHS2	CHS1	CHS0	00000000
ADCDL	ADC 数据低 8 位	C5	0~F	ADCV.3	ADCV.2	ADCV.1	ADCV.0	--	--	--	--	0000xxxx
ADCDH	ADC 数据高 8 位	C6	0~F	ADCV.11	ADCV.10	ADCV.9	ADCV.8	ADCV.7	ADCV.6	ADCV.5	ADCV.4	00000000
CKCON0	时钟控制 0	C7	0~F	AFS	--	--	--	CCKS	SCKS2	SCKS1	SCKS0	00010000
T2CON	*定时器 2 控制寄存器*	*C8*	*0*	*TF2*	*EXF2*	*RCLK/ TF2L*	*TCLK/ TL2IE*	*EXEN2*	*TR2*	*C/T2*	*CP/RL2*	*00000000*
T2MOD	定时器 2 模式寄存器	C9	0	T2SPL	TL2X12/ T2EIP	T2EXH	T2X12	TR2L	TR2LC	T2OE	T2MS0	00000000
RCAP2L	定时器 2 捕获低 8 位	CA	0	.7	.6	.5	.4	.3	.2	.1	.0	00000000
RCAP2H	定时器 2 捕获高 8 位	CB	0	.7	.6	.5	.4	.3	.2	.1	.0	00000000
TL2	定时器 2 低 8 位	CC	0	.7	.6	.5	.4	.3	.2	.1	.0	00000000
TH2	定时器 2 高 8 位	CD	0	.7	.6	.5	.4	.3	.2	.1	.0	00000000
CLRL	CL 重载寄存器	CE	0~F	.7	.6	.5	.4	.3	.2	.1	.0	00000000
CHRL	CH 重载寄存器	CF	0~F	.7	.6	.5	.4	.3	.2	.1	.0	00000000
PSW	*程序状态字*	*D0*	0~F	*CY*	*AC*	*F0*	*RS1*	*RS0*	*OV*	*F1*	*P*	*00000000*
SIADR	TWI0 地址寄存器	D1	0	.7	.6	.5	.4	.3	.2	.1	GC	00000000
SIA2	TWI0 2nd 地址寄存器	D1	2	.7	.6	.5	.4	.3	.2	.1	A2E	00000000
SIDAT	TWI0 数据寄存器	D2	0	.7	.6	.5	.4	.3	.2	.1	.0	00000000
SIA2M	SIA2 掩码寄存器	D2	2	SIA2M.7	SIA2M.6	SIA2M.5	SIA2M.4	SIA2M.3	SIA2M.2	SIA2M.1	1	11111111
SISTA	TWI0 状态寄存器	D3	0	.7	.6	.5	.4	.3	.2	.1	.0	11111000
SICON	TWI0 控制寄存器	D4	0	CR2	ENSI	STA	STO	SI	AA	CR1	CR0	00000000
KBPATN	键盘模式	D5	0~F	.7	.6	.5	.4	.3	.2	.1	.0	11111111
KBCON	键盘控制	D6	0~F	KBCS1	KBCS0	KBES	--	0	0	PATN_ SEL	KBIF	00000000
KBMASK	键盘中断掩码	D7	0~F	.7	.6	.5	.4	.3	.2	.1	.0	00000000
CCON	*PCA 控制寄存器*	*D8*	0~F	*CF*	*CR*	*0*	*0*	*CCF3*	*CCF2*	*CCF1*	*CCF0*	*00000000*
CMOD	PCA 模式寄存器	D9	0~F	CIDL	0	BME2	BME0	CPS2	CPS1	CPS0	ECF	00000000
CCAPM0	PCA 模块 0 模式	DA	0	DTE0	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	00000000
CCAPM1	PCA 模块 1 模式	DB	0	0	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	00000000
CCAPM2	PCA 模块 2 模式	DC	0~F	DTE2	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2	00000000
CCAPM3	PCA 模块 3 模式	DD	0~F	0	ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3	00000000
ACC	*累加器*	*E0*	0~F	*ACC.7*	*ACC.6*	*ACC.5*	*ACC.4*	*ACC.3*	*ACC.2*	*ACC.1*	*ACC.0*	*00000000*
WDTCR	看门狗控制寄存器	E1	0~F	WREN	NSW	ENW	CLRW	WIDL	PS2	PS1	PS0	00000000
IFD	ISP Flash 数据	E2	0~F	.7	.6	.5	.4	.3	.2	.1	.0	11111111
IFADRH	ISP Flash 地址高 8 位	E3	0~F	.7	.6	.5	.4	.3	.2	.1	.0	00000000
IFADRL	ISP Flash 地址低 8 位	E4	0~F	.7	.6	.5	.4	.3	.2	.1	.0	00000000
IFMT	ISP 模式表	E5	0~F	MS.7	MS.6	MS.5	MS.4	MS.3	MS.2	MS.1	MS.0	00000000
SCMD	ISP 系列命令	E6	0~F	.7	.6	.5	.4	.3	.2	.1	.0	xxxxxxxx
ISPCR	ISP 控制寄存器	E7	0~F	ISPEN	SWBS	SRST	CFAIL	0	0	PBSY	EEPF	00000000
P4	*端口 4*	*E8*	0~F	*P4.7*	*P4.6*	*P4.5*	*P4.4*	*1*	*1*	*1*	*1*	*11111111*
CL	PCA 基准定时器低	E9	0~F	.7	.6	.5	.4	.3	.2	.1	.0	00000000
CCAP0L	PCA 模块 0 捕获低	EA	0	.7	.6	.5	.4	.3	.2	.1	.0	00000000
CCAP1L	PCA 模块 1 捕获低	EB	0	.7	.6	.5	.4	.3	.2	.1	.0	00000000
CCAP2L	PCA 模块 2 捕获低	EC	0~F	.7	.6	.5	.4	.3	.2	.1	.0	00000000
CCAP3L	PCA 模块 3 捕获低	ED	0~F	.7	.6	.5	.4	.3	.2	.1	.0	00000000
B	*B 寄存器*	*F0*	0~F	*B.7*	*B.6*	*B.5*	*B.4*	*B.3*	*B.2*	*B.1*	*B.0*	*00000000*
PAOE	PWM 额外输出使能	F1	0~F	POE3	0	0	POE2	POE1	0	0	POE0	10011001
PCAPWM0	PCA PWM0 模式	F2	0	P0RS1	P0RS0	0	0	0	P0INV	ECAP0H	ECAP0L	00000000

符号	描述	地址 (HEX)	页 (HEX)	位地址及符号								复位值
				位-7	位-6	位-5	位-4	位-3	位-2	位-1	位-0	
PCAPWM1	PCA PWM1 模式	F3	0	P1RS1	P1RS0	0	0	0	P1INV	ECAP1H	ECAP1L	00000000
PCAPWM2	PCA PWM2 模式	F4	0~F	P2RS1	P2RS0	0	0	0	P2INV	ECAP2H	ECAP2L	00000000
PCAPWM3	PCA PWM3 模式	F5	0~F	P3RS1	P3RS0	0	0	0	P3INV	ECAP3H	ECAP3L	00000000
CH	PCA 基准定时器高	F9	0~F	.7	.6	.5	.4	.3	.2	.1	.0	00000000
CCAP0H	PCA 模块 0 捕获高	FA	0	.7	.6	.5	.4	.3	.2	.1	.0	00000000
CCAP1H	PCA 模块 1 捕获高	FB	0	.7	.6	.5	.4	.3	.2	.1	.0	00000000
CCAP2H	PCA 模块 2 捕获高	FC	0~F	.7	.6	.5	.4	.3	.2	.1	.0	00000000
CCAP3H	PCA 模块 3 捕获高	FD	0~F	.7	.6	.5	.4	.3	.2	.1	.0	00000000

3.3. 辅助 SFR 映射(P 页)

MG82F6B08 / 6B001/ 6B104 特殊功能寄存器(SFR)有一个辅助索引 P 页，它读写的方法跟标准的 8051 特殊功能寄存器的不一样。 像访问 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 **11 个物理字节地址**和 **5 个逻辑字节地址**。 11 个物理字节地址包括 IAPLB、CKCON2、CKCON3、CKCON4、PCON2、PCON3、PCON4、SPCON0、DCON0、RTCTM 和 RTCCR。6 个逻辑字节地址包括 PCON0、PCON1、CKCON0、WDTCR 和 P4。 在 0~F 页访问这 5 个逻辑地址会得到相同的 SFR 值。 更多详细的信息请参考章节 “28 P 页 SFR 访问”。

表 3–3. 辅助 SFR 映射（P 页）

		0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8		--	--	--	--	--	--	--	--
F0		--	--	--	--	--	--	--	--
E8		P4	--	--	--	--	--	--	--
E0		--	WDTCR	--	--	--	--	--	--
D8		--	--	--	--	--	--	--	--
D0		--	--	--	--	--	--	--	--
C8		--	--	--	--	--	--	--	--
C0		--	--	--	--	--	--	--	CKCON0
B8		--	--	--	--	--	--	--	--
B0		--	--	--	--	--	--	--	--
A8		--	--	--	--	--	--	--	--
A0		--	--	--	--	--	--	--	--
98		--	--	--	--	--	--	--	--
90		--	--	--	--	--	--	--	PCON1
88		--	--	--	--	--	--	--	--
80		--	--	--	--	--	--	--	PCON0
78		--	--	--	--	--	--	--	--
70		--	--	--	--	--	--	--	--
68		--	--	--	--	--	--	--	--
60		--	--	--	--	--	--	--	--
58		--	--	--	--	--	--	--	--
50		--	--	--	SPHB	RTCCR	RTCTM	--	--
48		SPCON0	--	--	--	DCON0	--	--	--
40		CKCON2	CKCON3	CKCON4	--	PCON2	PCON3	PCON4	--
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. 辅助特殊功能寄存器位分配(P 页)

表 3–4. 辅助 SFR 位分配(P 页)

符号	描述	地址	位地址及符号								复位值
			位-7	位-6	位-5	位-4	位-3	位-2	位-1	位-0	
物理字节											
IAPLB	IAP 低边界	03H	IAPLB6	IAPLB5	IAPLB4	IAPLB3	IAPLB2	IAPLB1	IAPLB0	0	
CKCON2	时钟控制 2	40H	0	1	0	IHRCOE	0	0	OSCS1	OSCS0	01010000
CKCON3	时钟控制 3	41H	WDTCS1	WDTCS0	FWKP	WDTFS	MCKD1	MCKD0	1	0	00000110
CKCON4	时钟控制 4	42H	RCSS2	RCSS1	RCSS0	RPSC2	RPSC1	RPSC0	RTCCS3	RTCCS2	00000000
PCON2	电源控制 2	44H	AWBOD1	0	BO1S1	BO1S0	BO1RE	EBOD1	BO0RE	1	0000x1x1
PCON3	电源控制 3	45H	IVREN	0	0	SPWRE	0	0	0	0	00000000
PCON4	电源控制 4	46H	0	0	0	BO1S2	0	0	0	0	00000000
SPCON0	SFR 页面控制 0	48H	0	0	P4CTL	WRCTL	0	CKCTL0	PWCTL1	PWCTL0	00000000
DCON0	设备控制 0	4CH	HSE	IAPO	0	0	0	IORCTL	RSTIO	OCDE	10000011
SPHB	SP 高位边界	53H	1	1	1	1	SPHB.3	SPHB.2	SPHB.1	SPHB.0	11111111
RTCCR	RTC 控制寄存器	54H	RTCE	RTCO	RTCRL5	RTCRL4	RTCRL3	RTCRL2	RTCRL1	RTCRL0	00111111
RTCTM	RTC 定时器寄存器	55H	RTCCS1	RTCCS0	RTCCT5	RTCCT4	RTCCT3	RTCCT2	RTCCT1	RTCCT0	01111111
逻辑字节											
PCON0	电源控制 0	87H	SMOD1	SMOD0	GF	POF0	GF1	GF0	PD	IDL	00010000
PCON1	电源控制 1	97H	SWRF	EXRF	0	RTCF	SPWF	BOF1	BOF0	WDTF	00000000
CKCON0	时钟控制 0	C7H	AFS	0	0	1	CCKS	SCKS2	SCKS1	SCKS0	00010000
WDTCR	看门口控制寄存器	E1H	WREN	NSW	ENW	CLRW	WIDL	PS2	PS1	PS0	00000000
P4	端口 4	E8H	P4.7	P4.6	P4.5	P4.4	1	1	1	1	11111111

P 页 SFR 写的示例代码：

```
Check PBSY =0;
IFADRH = 0x00;
ISPCR = ISPEN;                  //使能 IAP/ISP
IFMT = MS2;                    // P 页写，IFMT =0x04
IFADRL = SPCON0;                  //设置 P 页 SFR 地址
IFD |= CKCTL0;                  //设置 CKCTL0
SCMD = 0x46;                   //
SCMD = 0xB9;                   //
IFMT = Flash_Standby;              // IAP/ISP 等待，IFMT =0x00
ISPCR &= ~ISPEN;
```

4. 引脚结构

4.1. 封装指南

图 4–1. MG82F6B08AG10 MSOP10顶视图

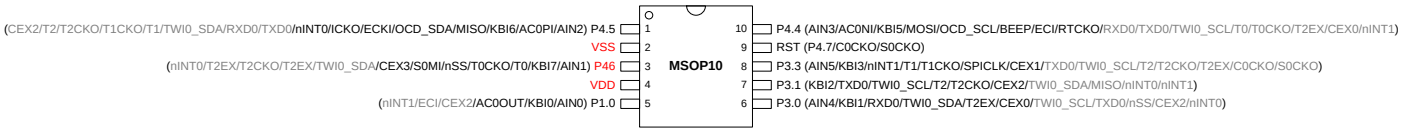


图 4–2. MG82F6B08AS8 SOP8顶视图

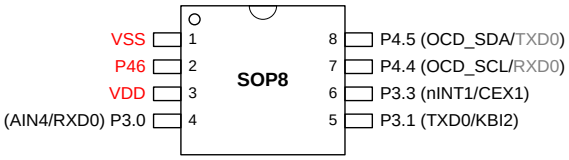


图 4–3. MG82F6B001AS8 SOP8顶视图

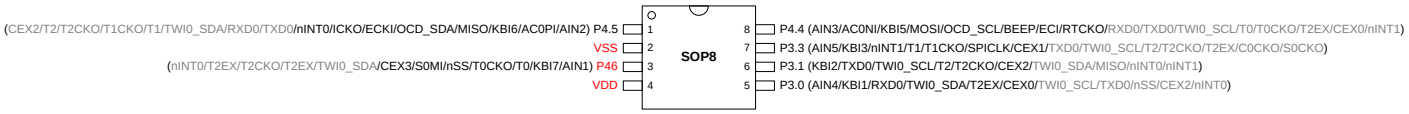
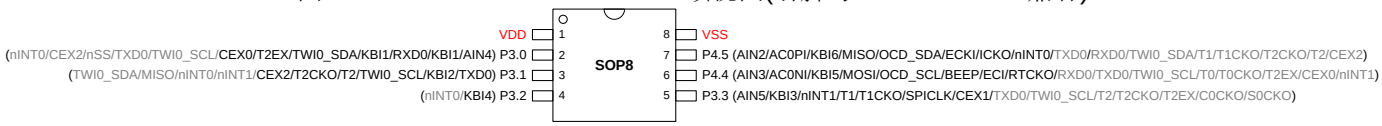


图 4–4. MG82F6B104AS8 SOP8顶视图(引脚与MG86FE/L104兼容)



4.2. 引脚描述

表 4–1. **MG82F6B08** 引脚描述

助记符	引脚号		I/O 类型	描述
	10-Pin MSOP	8-Pin SOP		
P1.0 (AIN0) (KBI0) (AC0OUT)	5	--	I/O	*Port 1.0。 * AIN0：ADC 模拟输入通道 0。 * KBI0：键盘输入 0。 * AC0OUT：模拟比较器 0 输出
P3.0 (AIN4) (KBI1) (RXD0) (TWI0_SDA) (T2EX) (CEX0)	6	4	I/O	* Port 3.0. * AIN4：ADC 模拟输入通道 4。 * KBI1：键盘输入 1。 * RXD0 ：UART0 串行输入口。 * TWI0_SCL：TWI0/I2C0 串行时钟。 * T2EX：定时器/计数器 2 外部控制输入。 * CEX0：PCA0 模块-0 外部 I/O。
P3.1 (KBI2) (TXD0) (TWI0_SCL) (T2) (T2CKO) (CEX2)	7	5	I/O	* Port 3.1. * KBI2：键盘输入 2。 * TXD0：UART0 串行输出口。 * TWI0_SCL：TWI0/I2C0 串行时钟。 * T2：定时器/计数器 2 外部时钟输入。 * T2CKO：定时器 2 可编程时钟输出。 * CEX2：PCA0 模块-2 外部 I/O。
P3.3 (AIN5) (KBI3) (nINT1) (T1) (T1CKO) (SPICLK) (CEX1)	8	6	I/O	* Port 3.3. * AIN5：ADC 模拟输入通道 5。 * KBI3：键盘输入 3。 * nINT1：外部中断 1 输入。 * T1：定时器/计数器 1 外部时钟输入。 * T1CKO：定时器 1 可编程时钟输出。 * SPICLK：SPI 时钟，用于主机输出，从机输入。 * CEX1：PCA0 模块-1 外部 I/O。
P4.4 (AIN3) (KBI5) (AC0NI) (MOSI) (OCD_SCL) (BEEP) (ECI) (RTCKO)	10	7	I/O	* Port 4.4. * AIN3：ADC 模拟输入通道 3。 * KBI5：键盘输入 5。 * AC0NI：模拟比较器 0 负极输入。 * MOSI：SPI 主出从入。 * OCD_SCL：OCD 接口串行时钟。 * BEEP：蜂鸣器输出。 * ECI：PCA 外部时钟输入。 * RTCKO：RTC 可编程时钟输出。
P4.5 (AIN2) (KBI6) (AC0PI) (MISO) (OCD_SDA) (ECKI) (ICKO) (nINT0)	1	8	I/O	* Port 4.5. * AIN2：ADC 模拟输入通道 2。 * KBI6：键盘输入 6。 * AC0PI：模拟比较器 0 正极输入。 * MISO：SPI 主入从出。 * OCD_SDA：OCD 接口串行数据。 * ECKI：外部时钟输入模式的时钟输入引脚。 * ICKO：内部时钟（MCK）输出。 * nINT0：外部中断 0 输入。
P4.6 (AIN1) (KBI7) (T0) (T0CKO) (nSS) (S0MI) (CEX3)	3	2	I/O	* Port 4.6. * AIN1：ADC 模拟输入通道 1。 * KBI7：键盘输入 7。 * T0：定时器/计数器 0 外部时钟输入。 * T0CKO：定时器 0 可编程时钟输出。 * nSS：SPI 从机选择。 * S0MI：串口 0 SPI 主机模式数据输入。 * CEX3：PCA0 模块-3 外部 I/O。
RST (P4.7) (C0CKO) (S0CKO)	9	--	I O	* RST：外部 RESET 输入，高电平有效。 * Port 4.7 仅输出。 *注意：当 P4.7/RST 用作 IO 模式。强烈建议避免将其设置为输入模式，以避免上电过程中因外部输入高电平而导致芯片锁死在复位阶段。* * C0CKO：PCA 计数器可编程时钟输出。 * S0CKO：S0BRT 可编程时钟输出。
VDD	4	3	P	供电输入。
VSS	2	1	G	地，0V 参考点。

表 4–2. **MG82F6B001** 引脚描述

助记符	引脚号 8-Pin SOP	I/O 类型	描述
P3.0 (AIN4) (KBI1) (RXD0) (TWI0_SDA) (T2EX) (CEX0)	5	I/O	* Port 3.0. * AIN4：ADC 模拟输入通道 4。 * KBI1：键盘输入 1。 * RXD0 ：UART0 串行输入口。 * TWI0_SCL：TWI0/I2C0 串行时钟。 * T2EX：定时器/计数器 2 外部控制输入。 * CEX0：PCA0 模块-0 外部 I/O。
P3.1 (KBI2) (TXD0) (TWI0_SCL) (T2) (T2CKO) (CEX2)	6	I/O	* Port 3.1. * KBI2：键盘输入 2。 * TXD0：UART0 串行输出口。 * TWI0_SCL：TWI0/I2C0 串行时钟。 * T2：定时器/计数器 2 外部时钟输入。 * T2CKO：定时器 2 可编程时钟输出。 * CEX2：PCA0 模块-2 外部 I/O。
P3.3 (AIN5) (KBI3) (nINT1) (T1) (T1CKO) (SPICLK) (CEX1)	7	I/O	* Port 3.3. * AIN5：ADC 模拟输入通道 5。 * KBI3：键盘输入 3。 * nINT1：外部中断 1 输入。 * T1：定时器/计数器 1 外部时钟输入。 * T1CKO：定时器 1 可编程时钟输出。 * SPICLK：SPI 时钟，用于主机输出，从机输入。 * CEX1：PCA0 模块-1 外部 I/O。
P4.4 (AIN3) (KBI5) (AC0NI) (MOSI) (OCD_SCL) (BEEP) (ECI) (RTCKO)	8	I/O	* Port 4.4. * AIN3：ADC 模拟输入通道 3。 * KBI5：键盘输入 5。 * AC0NI：模拟比较器 0 负极输入。 * MOSI：SPI 主出从入。 * OCD_SCL：OCD 接口串行时钟。 * BEEP：蜂鸣器输出。 * ECI：PCA 外部时钟输入。 * RTCKO：RTC 可编程时钟输出。
P4.5 (AIN2) (KBI6) (AC0PI) (MISO) (OCD_SDA) (ECKI) (ICKO) (nINT0)	1	I/O	* Port 4.5. * AIN2：ADC 模拟输入通道 2。 * KBI6：键盘输入 6。 * AC0PI：模拟比较器 0 正极输入。 * MISO：SPI 主入从出。 * OCD_SDA：OCD 接口串行数据。 * ECKI：外部时钟输入模式的时钟输入引脚。 * ICKO：内部时钟（MCK）输出。 * nINT0：外部中断 0 输入。
P4.6 (AIN1) (KBI7) (T0) (T0CKO) (nSS) (S0MI) (CEX3)	3	I/O	* Port 4.6. * AIN1：ADC 模拟输入通道 1。 * KBI7：键盘输入 7。 * T0：定时器/计数器 0 外部时钟输入。 * T0CKO：定时器 0 可编程时钟输出。 * nSS：SPI 从机选择。 * S0MI：串口 0 SPI 主机模式数据输入。 * CEX3：PCA0 模块-3 外部 I/O。
VDD	4	P	供电输入。
VSS	2	G	地，0V 参考点。

表 4–3. **MG82F6B104** 引脚描述

助记符	引脚号 8-Pin SOP	I/O 类型	描述
P3.0 (AIN4) (KBI1) (RXD0) (TWI0_SDA) (T2EX) (CEX0)	2	I/O	* Port 3.0. * AIN4：ADC 模拟输入通道 4。 * KBI1：键盘输入 1。 * RXD0 ：UART0 串行输入口。 * TWI0_SCL：TWI0/I2C0 串行时钟。 * T2EX：定时器/计数器 2 外部控制输入。 * CEX0：PCA0 模块-0 外部 I/O。
P3.1 (KBI2) (TXD0) (TWI0_SCL) (T2) (T2CKO) (CEX2)	3	I/O	* Port 3.1. * KBI2：键盘输入 2。 * TXD0：UART0 串行输出口。 * TWI0_SCL：TWI0/I2C0 串行时钟。 * T2：定时器/计数器 2 外部时钟输入。 * T2CKO：定时器 2 可编程时钟输出。 * CEX2：PCA0 模块-2 外部 I/O。
P3.2 (KBI4)	4	I/O	* Port 3.2. * KBI4：键盘输入 4。
P3.3 (AIN5) (KBI3) (nINT1) (T1) (T1CKO) (SPICLK) (CEX1)	5	I/O	* Port 3.3. * AIN5：ADC 模拟输入通道 5。 * KBI3：键盘输入 3。 * nINT1：外部中断 1 输入。 * T1：定时器/计数器 1 外部时钟输入。 * T1CKO：定时器 1 可编程时钟输出。 * SPICLK：SPI 时钟，用于主机输出，从机输入。 * CEX1：PCA0 模块-1 外部 I/O。
P4.4 (AIN3) (KBI5) (AC0NI) (MOSI) (OCD_SCL) (BEEP) (ECI) (RTCKO)	6	I/O	* Port 4.4. * AIN3：ADC 模拟输入通道 3。 * KBI5：键盘输入 5。 * AC0NI：模拟比较器 0 负极输入。 * MOSI：SPI 主出从入。 * OCD_SCL：OCD 接口串行时钟。 * BEEP：蜂鸣器输出。 * ECI：PCA 外部时钟输入。 * RTCKO：RTC 可编程时钟输出。
P4.5 (AIN2) (KBI6) (AC0PI) (MISO) (OCD_SDA) (ECKI) (ICKO) (nINT0)	7	I/O	* Port 4.5. * AIN2：ADC 模拟输入通道 2。 * KBI6：键盘输入 6。 * AC0PI：模拟比较器 0 正极输入。 * MISO：SPI 主入从出。 * OCD_SDA：OCD 接口串行数据。 * ECKI：外部时钟输入模式的时钟输入引脚。 * ICKO：内部时钟（MCK）输出。 * nINT0：外部中断 0 输入。
VDD	1	P	供电输入。
VSS	8	G	地，0V 参考点。

4.3. 功能复用

许多 I/O 引脚，除了普通的 I/O 口功能之外，还能复用其他内部功能。对于这些数码外设，默认状态是 GPIOs。然而，用户可以通过 AXRU0~AXUR3 相对应的控制位重新定义端口的功能。

AUXR0:辅助寄存器 0

SFR 页 = 0~F

SFR 地址 = 0xA1 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P45OC1	P45OC0	0	PBKF	0	0	INT1H	INT0H
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7~6：P4.5 功能配置控制位 1 和位 0，这两位仅仅当内部 RC 振荡(IHRCO 或 ILRCO)被选择为系统时钟源时有效。当外部时钟输入模式，P4.5 专用于时钟输入。在内部振荡模式，P4.5 为普通 I/O 或时钟源发生器提供下列选项，当 P60OC[1:0] 索引为非 P4.5 GPIO 功能时，P4.5 将驱动内部 RC 振荡器输出为其它设备提供时钟源。

P45OC[1:0]	P4.5 function	I/O mode
0 0	P4.5	By P4M1.5 & P4M0.5
0 1	MCK	By P4M1.5 & P4M0.5
1 0	MCK/2	By P4M1.5 & P4M0.5
1 1	MCK/4	By P4M1.5 & P4M0.5

了解详情，请参考“8 系统时钟” P4.5 作为时钟输出功能时，建议设置{P4M1.5, P4M0.5}为“1”来选择 P4.5 为推挽输出模式。

AUXR1:辅助寄存器 1

SFR 页 = 0~F

SFR 地址 = 0xA2 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	0	CRCDS1	CRCDS0	0	0	0	DPS
W	W	R/W	R/W	W	W	W	R/W

AUXR2:辅助寄存器 2

SFR 页 = 0~F

SFR 地址 = 0xA3 复位值 = 0000-0000

7	6	5	4	3	2	1	0
STAF	STOF	0	C0PLK	T1X12	T0X12	T1CKOE	T0CKOE
R/W	R/W	W	W	R/W	R/W	R/W	R/W

AUXR3:辅助寄存器 3

SFR 页 = 仅 **0** 页

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	T0PS0	BPOC1	BPOC0	S0PS0	TWIPS1	TWIPS0	T0XL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7：保留，当 AUXR3 被写入时，软件必须给该位写“**0**”。

Bit 6：T0PS0，定时器 0 端口引脚选择。

T0PS0	T0/T0CKO
0	P4.6
1	P4.4

Bit 5~4：BPOC1~0，蜂鸣器输出控制位

BPOC[1:0]	P4.4 function	I/O mode
0 0	P4.4	By P4M0.4 & P4M1.4
0 1	ILRCO/32	By P4M0.4 & P4M1.4
1 0	ILRCO/16	By P4M0.4 & P4M1.4
1 1	ILRCO/8	By P4M0.4 & P4M1.4

P4.4 建议设置为推挽输出模式以使用蜂鸣器功能。

Bit 3：S0PS0，串口 0 (UART0) 端口引脚选择位。(S0PS1 在 AUXR10.3)。

S0PS1~0	RXD0	TXD0
0 0	P3.0	P3.1
0 1	P4.4	P4.5
1 0	P4.5	P3.0
1 1	P4.5	P4.4

Bit 2~1：TWIPS1~0，TWI0 端口引脚选择位[1:0]。

TWIPS1~0	TWI0_SCL	TWI0_SDA
0 0	P3.1	P3.0
0 1	P4.4	P4.5
1 0	P3.0	P3.1
1 1	P3.3	P4.6

AUXR4：辅助寄存器 4

SFR 页 = **仅 1 页**

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
T2PS1	T2PS0	0	T1PS0	0	0	AC0OE	AC0FLT1
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6：T2PS1~0，定时器 2 端口引脚选择位[1:0]。

T2PS1~0	T2/T2CKO	T2EX
0 0	P3.1	P3.0
0 1	P3.3	P4.6
1 0	P4.6	P3.3
1 1	P4.5	P4.4

Bit 5：保留，当 AUXR4 被写入时，软件必须给该位写“**0**”。

Bit 4：T1PS1~0，定时器 1 端口引脚选择位[1:0]。

T1PS0	T1/T1CKO
0	P3.3
1	P4.5

Bit 1：AC0OE，AC0OUT 输出使能到引脚。
0：禁用 AC0OUT 输出到引脚。
1：使能 AC0OUT 输出到 **P1.0**。

AUXR5:辅助寄存器 5

SFR 页 = **仅 2 页**

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	C0IC2S0	0	0	C0PS1	C0PS0	ECIPS0	C0COPS
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 6：C0IC2S0 PCA0 输入通道 2 输入选择。

C1IC2S0	CEX2 input
0	CEX2 Port Pin
1	T2EXI

Bit 3~2：C0PS[1:0]，PCA0 引脚选择 1 & 0。

C0PS1~0	CEX0	CEX1	CEX2	CEX3
0 0	P3.0	P3.3	P3.1	P4.6
0 1	P4.4	P3.3	P4.5	P4.6
1 0	P4.4	P3.3	P3.0	P4.6
1 1	P4.4	P3.3	P1.0	P4.6

Bit 1：ECIPS0，PCA0 ECI 端口引脚选择位 0。

ECIPS0	ECI
0	P4.4
1	P1.0

Bit 0：C0COPS，PCA0 时钟输出(C0CKO)端口引脚选择位。

C0COPS	C0CKO
0	P4.7
1	P3.3

AUXR6:辅助寄存器 6

SFR 页 = 仅 **3** 页

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	0	0	0	0	T2FCS	SnMIPS	S0COPS
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 2：T2FCS，保留给芯片测试。

Bit 0：S0COPS，S0BRG 时钟输出(S0CKO)端口引脚选择位。

S0COPS	S0CKO
0	P4.7
1	P3.3

AUXR7:辅助寄存器 7

SFR 页 = 仅 **4** 页

SFR 地址 = 0xA4 复位值 = 1100-0000

7	6	5	4	3	2	1	0
1	1	C0CKOE	SPI0M0	0	0	0	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 5：C0CKOE，PCA0 时钟输出使能。

0：禁止 PCA0 时钟输出。

1：PCA0 基准定时器溢出率的二分之一时钟输出使能。

AUXR9:辅助寄存器 9

SFR 页 = 仅 **6** 页

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	SIDPS0	T1G1	T0G1	C0FDC1	C0FDC0	0	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 6：SID/STWI 引脚选择。

SIDPS0	STWI_SCL	STWI_SDA
0	nINT1	S0MI
1	TWI0_SCL	TWI0_SDA

Bit 5：T1G1，定时器 1 门控源选择。

T1G1, T1GATE	T1 门控源
0 0	禁用
0 1	INT1 激活
1 0	TF2 激活
1 1	KBI 激活

Bit 4：T0G1，定时器 1 门控源选择。

T0G1, T0GATE	T0 门控源
0 0	禁用
0 1	INT0 激活
1 0	TF2 激活
1 1	KBI 激活

AUXR10:辅助寄存器 10

SFR 页 = **仅 7 页**

SFR 地址 = 0xA4 复位值 = 1100-0000

7	6	5	4	3	2	1	0
1	AC0HC0	0	SPIPS0	S0PS1	SPFACE	TWICF	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 4：SPIPS0，SPI 端口引脚选择位。

SPIPS0	nSS	MOSI	MISO	SPICLK
0 0	P4.6	P4.4	P4.5	P3.3
0 1	P3.0	P4.4	P3.1	P3.3

Bit 3：S0PS1，串口 0 选择 1。（该功能与 AUXR3.3, S0PS0 组合）

Bit 1：TWICF，TWI0/I2C0 串行时钟输入滤波。
0：禁用 TWICF 功能。
1：使能 TWICF 功能。

AUXR11:辅助寄存器 11

SFR 页 = **仅 8 页**

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P30AM	P33AM	0	0	0	POEM0	C0M0	C0OFS
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

XICFG:外部中断配置寄存器

SFR 页 = **仅 0 页**

SFR 地址 = **0xC1** 复位值 = 0000-0000

7	6	5	4	3	2	1	0
INT1IS.1	INT1IS.0	INT0IS.1	INT0IS.0	0	0	X1FLT	X0FLT
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6：INT1IS.1~0，由 INT1IS.2 决定功能的 nINT1 输入引脚选择位如下表定义。

INT1IS.1~0	选择 nINT1 的端口引脚
0 0	P3.3
0 1	P3.1
1 0	P4.4
1 1	P1.0

Bit 5~4：INT0IS.1~0，由 INT0IS.2 决定功能的 nINT0 输入引脚选择位如下表定义。

INT0IS.1~0	选择 nINT0 的端口引脚
0 0	P4.5
0 1	P3.0
1 0	P3.2
1 1	P4.6

XICFG1:外部中断配置寄存器 1

SFR 页 = **仅 1 页**

SFR 地址 = **0xC1** 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	0	0	0	0	0	X1FLT1	X0FLT1
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

5. 8051 CPU 功能描述

5.1. CPU 寄存器

PSW:程序状态字

SFR 页 = 0~F
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，祥见章节“6.2 片内数据存储器 RAM”。一些指令参考这些内存(RAM)的位置比如从 R0 到 R7。

奇偶位反映累加器内 1 的个数的状况，累加器中 1 的个数是奇数则 P=1，否则 P=0 。

SP:堆栈指针

SFR 页 = 0~F
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，复位后默认值为为 0x07。

DPL:数据指针低字节

SFR 页 = 0~F
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 16 位 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 页 = 0~F

SFR 地址 = 0x83 复位值 = 0000-0000

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

DPH 是 16 位 DPTR 的高字节，DPTR 用来间接访问 XRAM 和程序空间。

ACC:累加器

SFR 页 = 0~F

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 页 = 0~F

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

MG82F6B08 / 6B001/ 6B104 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 1~7 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，指令的时序比标准的 8051 更快。

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“33 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。

寄存器指令寻址(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。

特殊寄存器指令寻址
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。

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，**MG82F6B08 / 6B001/ 6B104** 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 **8K** 字节。在 **MG82F6B08 / 6B001/ 6B104** 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。**MG82F6B08 / 6B001/ 6B104** 只有 256 字节的内部和 **768** 字节的片上扩展存储器(XRAM)。

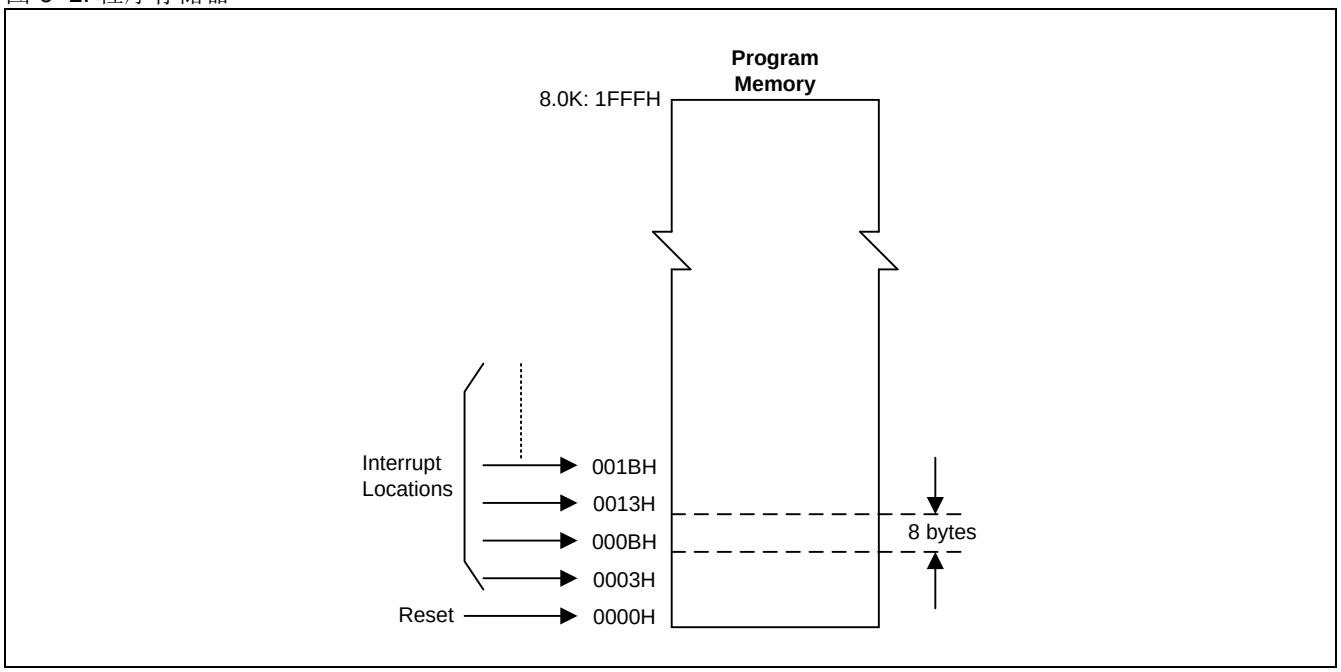
MG82F6B08 / 6B001/ 6B104 提供 512 字节 EEPROM 用以存储用户数据。

6.1. 片内程序存储器（FLASH）

程序存储器用来保存让 CPU 进行处理的程序代码，如图 6–1 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 6–1. 程序存储器



6.2. 片内数据存储器 RAM

图 6–2 向 **MG82F6B08 / 6B001/ 6B104** 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 6–3 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 6–4 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 6–2. 数据存储器

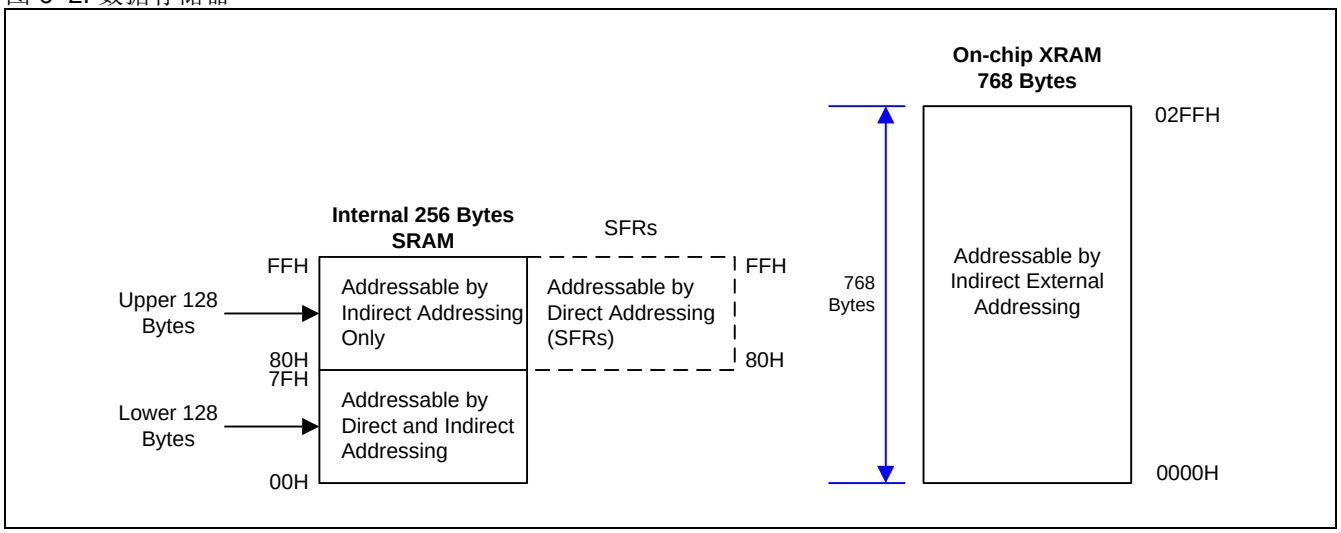


图 6–3. 内部 RAM 的低 128 字节

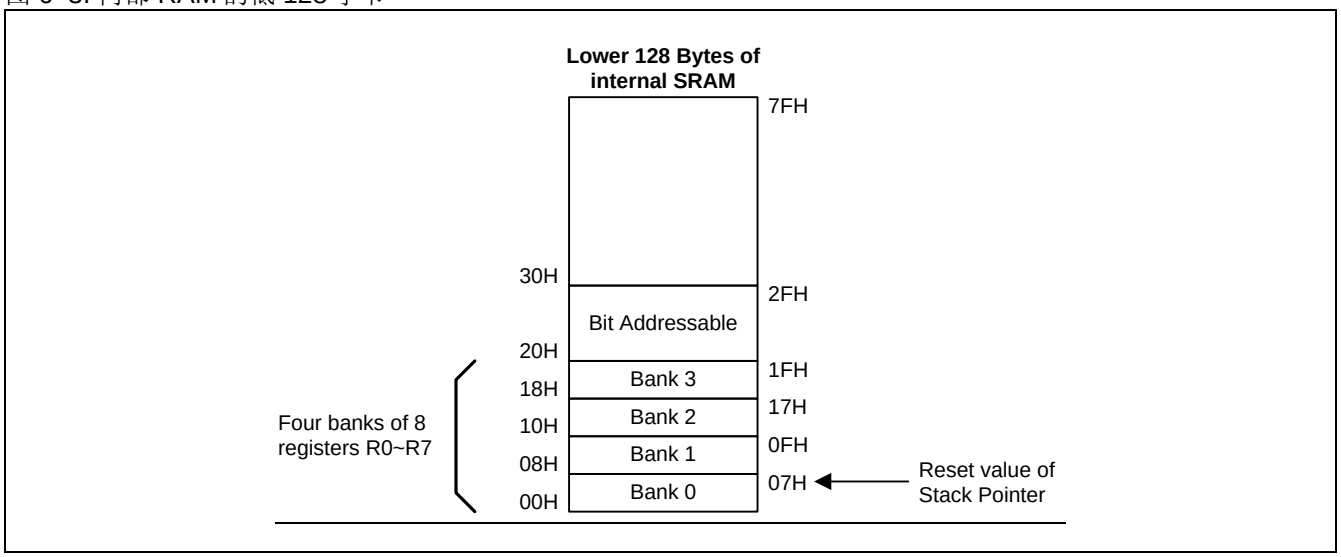
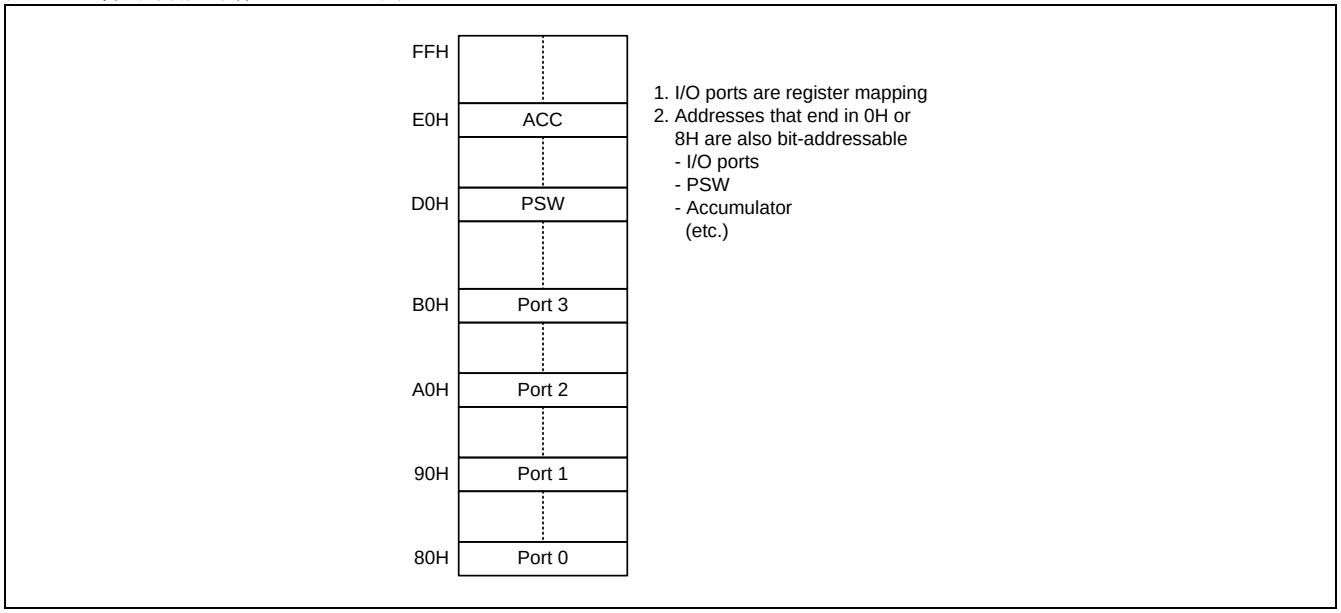


图 6–4. 特殊功能寄存器 SFR 空间



6.3. 片内扩展 RAM (XRAM)

访问片上扩展 RAM(XRAM)，参考图 6–2，这 **768** 字节的 XRAM(0000H to **02FFH**)可以被外部移动指令“MOVX @Ri” 和“MOVX @DPTR”间接访问，在 C51 编译器中，使用“pdata”或“xdata”声明变量分配到 XRAM 中，编译后，被“pdata”或“xdata”声明过的变量将分别通过“MOVX @Ri”或“MOVX @DPTR”指令进行存取，这样 **MG82F6B08 / 6B001/ 6B104** 硬体才能正确访问 XRAM。

6.4. 关于 C51 编译器的声明标识符

C51 编译器的声明识别符与 **MG82F6B08 / 6B001/ 6B104** 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

外部数据或片上的扩展 RAM(XRAM)；通过“MOVX @DPTR”指令访问标准 80C51 的 64K 存储空间。**MG82F6B08 / 6B001/ 6B104** 有 **768** 字节的片内 xdata 存储空间。

pdata

分页的外部数据(256 字节)或片上的扩展 RAM(XRAM)：重叠的 256 字节的存储器地址通过“MOVX @Ri”指令访问。**MG82F6B08 / 6B001/ 6B104** 有 256 字节片上 pdata 存储器它与片上 xdata 存储器共享。

code

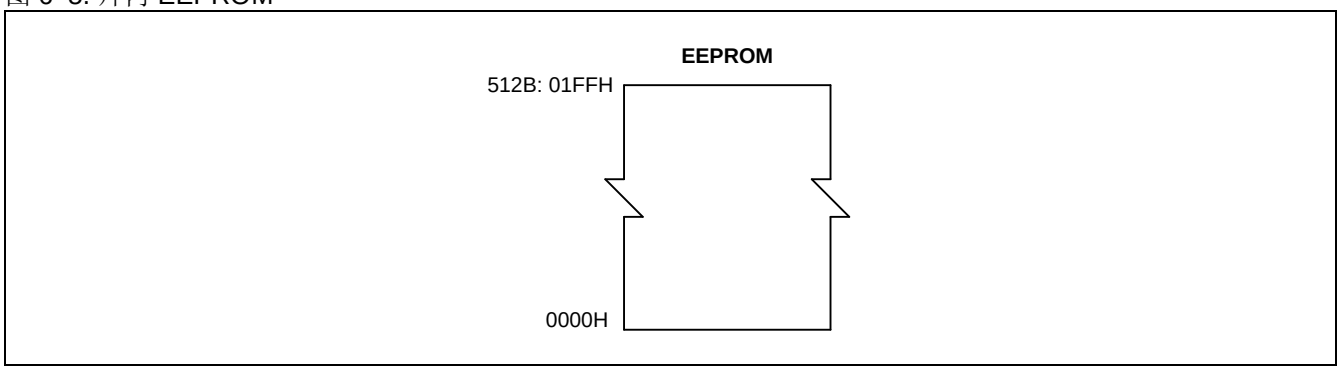
8K 字节程序存储空间。通过“MOVC @A+DTPR”访问，作为程序部分被读取。**MG82F6B08 / 6B001/ 6B104** 有 **8K** 字节的片上程序存储器。

6.5. 片内 EEPROM

MG82F6B08 / 6B001/ 6B104 包含 **512** 字节数据 EEPROM 空间，其寿命可达至少 100,000 次擦/写周期。该 EEPROM 的详细性能请参照章节“32.7 EEPROM 特性”。

The in **MG82F6B08 / 6B001/ 6B104** 的 EEPROM 被设置为一个单独的区域，可进行单字节读写。通过其他 IFMT 设定的 ISP/IAP 流可进行 EEPROM 和 CPU 间的访问。在 EEPROM 编程中，CPU 不会被停止。有个状态位 PBSY (ISCR.1)，代表 EEPROM 处于编程中的繁忙状态。EEPROM 编程标志会被存在 EEPF (ISPCR.0)以表示编程流已完成。该标志也支持在系统标志中断中作为中断源进行中断。请参照“27.2 MG82F6B08 / 6B001/ 6B104 EEPROM 访问流程”章节，以获取 EEPROM 的访问流程细节。

图 6–5. 片内 EEPROM



7. 外部数据存储器(XRAM)访问

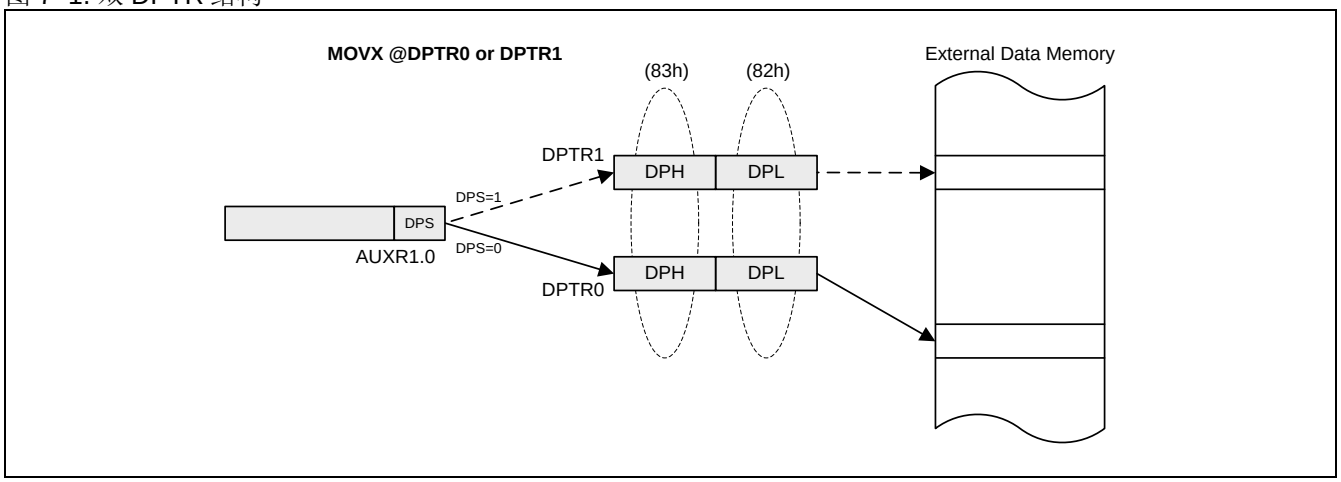
MG82F6B08 / 6B001/ 6B104 系列 MCU 内含有 **768** 字节被映射到外部数据存储空间的数据存储器(XRAM)。外部数据存储空间可以使用外部移动指令(MOVX)和数据指针(DPTR)访问，或使用(R0 或 R1)的 MOVX 间接访问模式。如果 MOVX 指令使用 8 位寻址操作(比如@R1)，16 位地址的高字节则由 XRAM 的页选择寄存器(XRPS)决定。

使用 MOVX 指令访问**内部的** XRAM 存储空间。MOVX 指令有使用两种间接寻址方法。第一种方法是使用数据指针(DPTR)，一个包含外部数据存储器(XRAM)读写有效地址的 16 位寄存器。第二种方法是使用 R0 或 R1 结合 XRPS 寄存器来获取有效的外部数据存储器(XRAM)地址。

7.1. MOVX 在 16 位地址的双数据指针寄存器(DPTR)应用

如图 7–1所示的双DPTR结构是能让芯片指定外部数据存储器的定位地址的一种方法。有两个16位DPTR寄存器，和一个称作为DPS(AUXR1.0)的控制位，允许在程序代码和外部存储器之间的切换。

图 7–1. 双 DPTR 结构



DPTR 指令

使用 DPS 位的六条指令参考 DPTR 的当前选择，如下：

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

AUXR1:辅助寄存器 1

SFR 页 = 0~F

SFR 地址 = 0xA2 复位值 = 0000-0000

7	6	5	4	3	2	1	0
--	--	CRCDS1	CRCDS0	--	--	--	DPS
W	**W**	**R/W**	**R/W**	**W**	**W**	**W**	**R/W**

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

DPS	选择 DPTR
0	DPTR0
1	DPTR1

DPL:数据指针低 8 位

SFR 页 = 0~F

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 寄存器是 16 位 DPTR 的低字节。DPTR 用于间接访问 XRAM 和闪存(Flash)存储器的编址。

DPH:数据指针高 8 位

SFR 页 = 0~F

SFR 地址 = 0x83 复位值 = 0000-0000

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

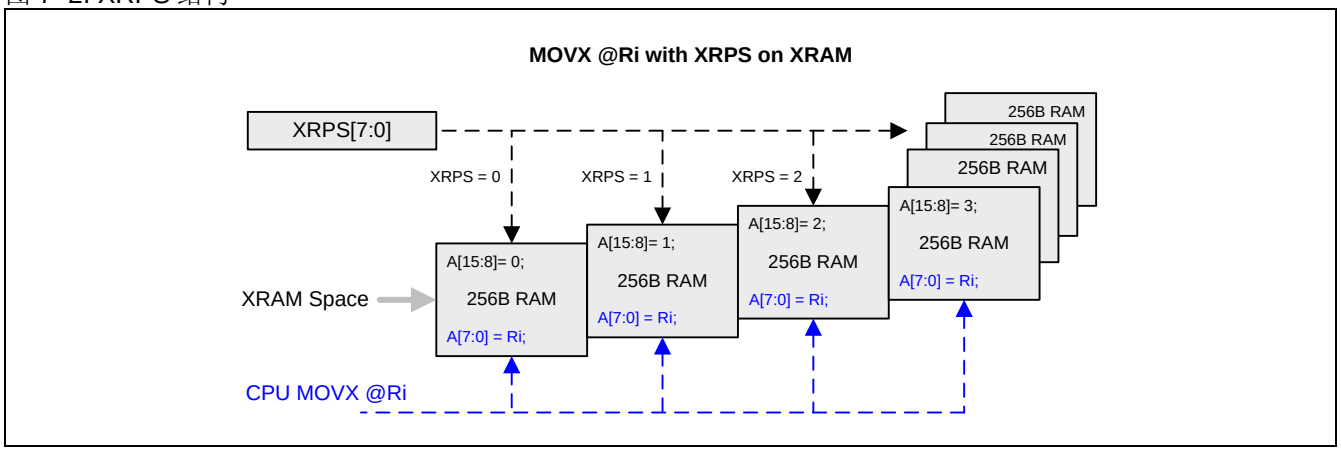
DPH寄存器是16位DPTR的高字节。DPTR用于间接访问XRAM和闪存(Flash)存储器的编址。

7.2. MOVX 在有 XRPS 的 8 位地址应用

MOVX 指令的 8 位地址是由 XRPS SFR 的值作为有效地址的高 8 位和 R0 或 R1 的值作为有效地址的低 8 位构成。

该功能可提供给设计者更高效率的访问 XRAM，访问整个 XRAM 空间需要 2 字节地址，软件编译器将通过使用 DPTR 访问特殊内存空间，来编译这 2 字节地址。其结果就是增加了更多指令，并降低效率。但是如果对全局“pdata”变量使用 XRPS 指令，编译器将编译成 MOVX@Ri 以减少额外指令以增强内存访问性能。

图 7–2. XRPS 结构



XRPS: XRAM 页选择寄存器

SFR 页 = 0~F

SFR 地址 = 0x8F 复位值 = 0000-0000

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

Bit 7~2：保留位。当 XRPS 被写入时，这些位必须软件写“0”。

Bit 1~0：XRPS，XRAM 的页选择。XRPS 寄存器是 16 位外部数据存储器地址的高字节，当用于 8 位 MOVX 指令时有效地选择 RAM 的 256 字节页。因此 XRPS 寄存器的高位(保留位)总应是零，XRPS 决定 XRAM 访问的是哪一页。在 **MG82F6B08 / 6B001/ 6B104** 中，XRPS 索引 **3** 页的 256 字节页 RAM (XRPS = 00 ~ 10)。

例如：如果 XRPS = 0x01，则访问 XRAM 的 0x0100 到 0x01FF 地址。

8. 系统时钟

系统时钟有 **3** 个时钟来源：内部高速 RC 震荡器 (IHRCO)，内部低速 RC 震荡器(ILRCO) 和外部频率输入。如图 8–1 所示 **MG82F6B08 / 6B001/ 6B104** 系统时钟结构。

MG82F6B08 / 6B001/ 6B104 总是由 IHRCO **16MHz** 启动，软件可以根据应用要求自由切换 **3** 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。在外部时钟输入模式 (ECKI)，时钟源来自 **P4.5**。

内建 IHRCO 提供 2 种频率 **16MHz 和 22.12MHz** 供软件选择，可以给系统时钟提供高精度的频率。详细的 IHRCO 性能，请参考章节“32.4 IHRCO 特性”。在 IHRCO 或 ILRCO 模式，**P4.5** 可以作为内部 *MCK* 或 2 分频时钟*(MCK/2)*输出或 4 分频时钟*(MCK/4)*输出给其他系统时钟源应用。

内置 ILRCO 提供低功耗的 32KHz 低速频率给 WDT 和系统时钟源使用。对于需低功耗运行的软件，MCU 可以选择 ILRCO 作为系统时钟源。若要查找详细的 ILRCO 性能，请参考章节“32.5 ILRCO 特性”。在 ILRCO 模式下，可以将 **P4.5** 配置为内部 *MCK* 或 2 分频时钟*(MCK/2)*输出或 4 分频时钟*(MCK/4)*输出为系统中的应用。

通过时钟分配器分配 **3** 种时钟源的一种作为系统时钟(SYSCLK)，如图 8–1 所示。用户能通过设置 CKCON3 寄存器的 MCKD1~MCKD0 和 CKCON0 寄存器的 SCKS2~SCKS0 位获得适当的系统时钟。

在芯片复位后，系统时钟默认为 IHRCO= 16MHz/2 的 8MHz。其 MCKD1~MCKD0 位为“01”。

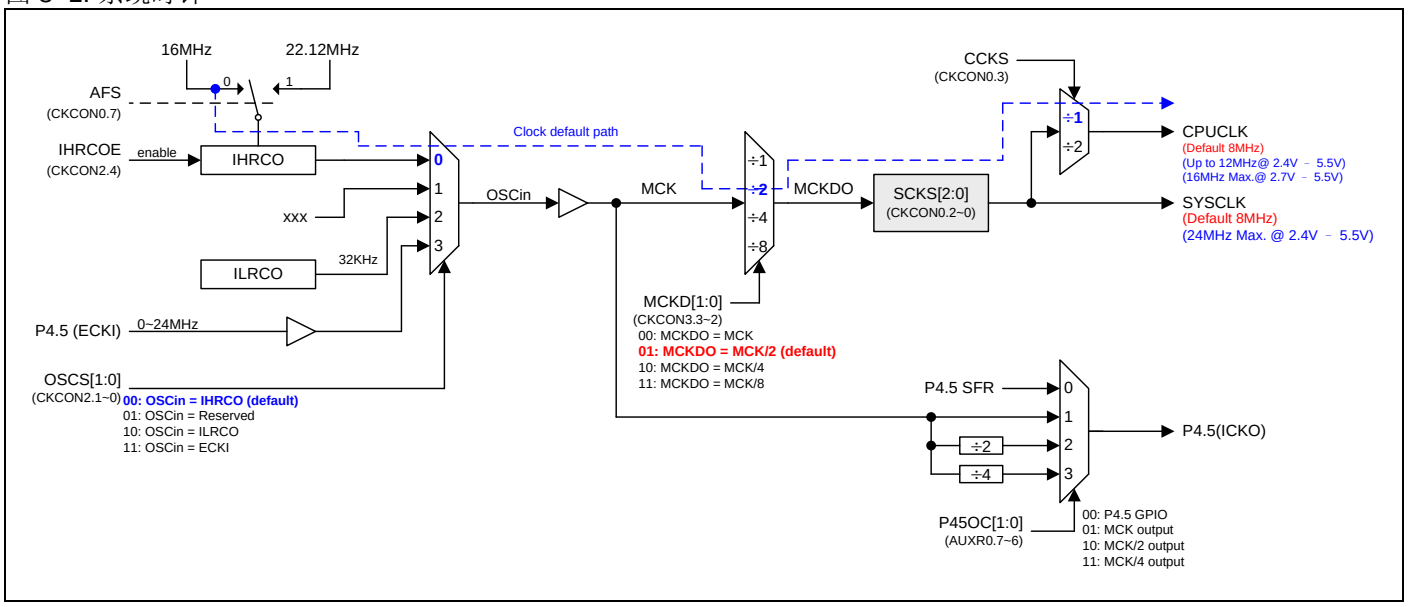
8.1. 时钟结构

MG82F6B08 / 6B001/ 6B104 的主要时钟系统如图 8–1 所示。CPUCLK 初始震荡源是通过 MCKD 分频的内部高速振荡器 IHRCO/2 的 8MHz。结合时钟倍频器和分频器产生不同的频率。最大 CPUCLK 如下：

- SYSCLK 最大到 24MHz @ 2.4V – 5.5V
- CPUCLK 最大到 12MHz @ 2.4V – 5.5V; 最大到 16MHz @ 2.7V – 5.5V

如果需要更高频的应用，当 CPUCLK > 48KHz 时 HSE(DCON0.7)需要置位。

图 8–1. 系统时钟



8.2. 时钟源切换

系统时钟有 **3** 个时钟来源：内部高速 RC 震荡器 (IHRCO)，内部低速 RC 震荡器(ILRCO) 和外部频率输入。如图 8–1 所示 **MG82F6B08 / 6B001/ 6B104** 系统时钟结构。**MG82F6B08 / 6B001/ 6B104** 总是以 IHRCO/2 的 **8MHz** 启动，软件可以通过设定 OSCS[1:0]来选择时钟源，但必须等时钟稳定后才能切换。

8.3. 时钟寄存器

CKCON0:时钟控制寄存器 0

SFR 页 = **0~F & P**

SFR 地址 = 0xC7 复位值 = 0001-0000

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

Bit 7：AFS，交替频率选择。
0: 选择 IHRCO 为 16MHz。
1: 选择 IHRCO 为 22.12MHz。

Bit 3：CCKS，CPU 时钟选择。
0：选择 SYSCLK 为 CPU 时钟。
1：选择 SYSCLK/2 为 CPU 时钟。

Bit 2~0：SCKS2 ~ SCKS0，可编程系统时钟选项。

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

CKCON1:时钟控制寄存器 1

SFR 页 = **0~F**
SFR 地址 = 0xBF 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	0	0	0	OSCSTA3	OSCSTA2	OSCSTA1	OSCSTA0
R	**R**	**R**	**R**	**R**	**R**	**R**	**R**

Bit 3~0: OSCSTA[3:0]
0001：OSCin MUX 使用 IHRCO 时钟源。
0010：保留。
0100：OSCin MUX 使用 ILRCO 时钟源。
1000：OSCin MUX 使用 ECKI 时钟源。
其它：OSCin MUX 正在切换时钟。

CKCON2:时钟控制寄存器 2

SFR 页 = **仅 P 页**
SFR 地址 = 0x40 复位值 = 0101-0000

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

Bit 4：IHRCOE，内部高频 RC 振荡使能。
0：禁止内部高频 RC 振荡器。
1：使能内部高频 RC 振荡器。如果软件设置这个位，在 IHRCOE 位使能后，必须等待 **32 us** IHRCOE 才能稳定输出。

Bit 1~0：OSCS[1:0]，OSCin 时钟源选择。

OSCS[1:0]	OSCin 源选择
0 0	IHRCO
0 1	保留
1 0	ILRCO
1 1	ECKI，外部时钟输入(P4.5)作为 OSCin。

CKCON3:时钟控制寄存器 3

SFR 页 = **仅 P 页**
SFR 地址 = 0x41 复位值 = 0000-0110

7	6	5	4	3	2	1	0
WDTCS1	WDTCS0	FWKP	WDTFS	MCKD1	MCKD0	1	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6：WDTCS1~0。WDT 时钟源选择。

Bit 5：FWKP，MCU 快速唤醒控制。
0：选择 MCU 从掉电模式正常唤醒时间大约 **90~120us**。
1：选择 MCU 从掉电模式快速唤醒时间大约 **15~30us**。

Bit 4：WDTFS，WDT 溢出源选择。
0：选择 WDT 位-7 溢出作为 WDT 事件源。
1：选择 WDT 位-0 溢出作为 WDT 事件源。

Bit 3~2：MCKD[1:0]，MCK 驱动器输出选择。默认是 MCK/2。

MCKD[1:0]	MCKDO 频率	例如 MCK = 12MHz	例如 MCK = 48MHz
0 0	MCKDO = MCK	MCKDO = 16MHz	MCKDO = 22MHz
0 1	MCKDO = MCK/2	MCKDO = 8MHz	MCKDO = 11MHz
1 0	MCKDO = MCK/4	MCKDO = 4MHz	MCKDO = 5.5MHz
1 1	MCKDO = MCK/8	MCKDO = 2MHz	MCKDO = 2.76MHz

AUXR0: 辅助寄存器 0

SFR 页 = 0~F
SFR 地址 = 0xA1 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P45OC1	P45OC0	0	PBKF	0	0	INT1H	INT0H
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7~6：P4.5 功能配置控制位 1 和位 0。这两位仅仅当内部 RC 振荡（IHRCO 或 ILRCO）被选择为系统时钟源时有效。当外部时钟输入模式，P4.5 专用于时钟输入。在内部振荡模式，P4.5 为普通 I/O 或时钟源发生器提供下列选项。当 P45OC[1:0] 索引为非 P4.5 GPIO 功能时，P4.5 将驱动内部 RC 振荡器输出为其它设备提供时钟源。

P45OC[1:0]	P4.5 功能	I/O 模式
0 0	P4.5	By P4M1.5 & P4M0.5
0 1	MCK	By P4M1.5 & P4M0.5
1 0	MCK/2	By P4M1.5 & P4M0.5
1 1	MCK/4	By P4M1.5 & P4M0.5

P4.5 作为时钟输出功能时, 建议设置{P4M1.5, P4M0.5}为“01”来选择 P4.5 为推挽输出模式。

DCON0: 设备控制 0

SFR 页 = **仅 P 页**
SFR 地址 = 0x4C POR = 1000-0011

7	6	5	4	3	2	1	0
HSE	IAPO	0	0	0	IORCTL	RSTIO	OCDE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7：HSE，高速运行使能。
0：选择 CPU 运行在低速模式(F_{CPUCLK} ≤ 48Hz)这样减慢内部电路从而降低功耗。

软件流禁用 HSE
步骤1：切换F_{CPUCLK} ≤ 48KHz。
步骤2：设置HSE = 0。

1：使能 MCU 全速运行(F_{CPUCLK} > 48KHz)。在 SYSCLK 选择高频时钟(>48KHz)之前，软件必须置位 HSE 切换到用于高速运行的内部电路。

软件流使能 HSE
步骤1：设置HSE = 1。
步骤2：切换F_{CPUCLK} > 48KHz。

9. 看门狗定时器(WDT)

9.1. WDT 结构

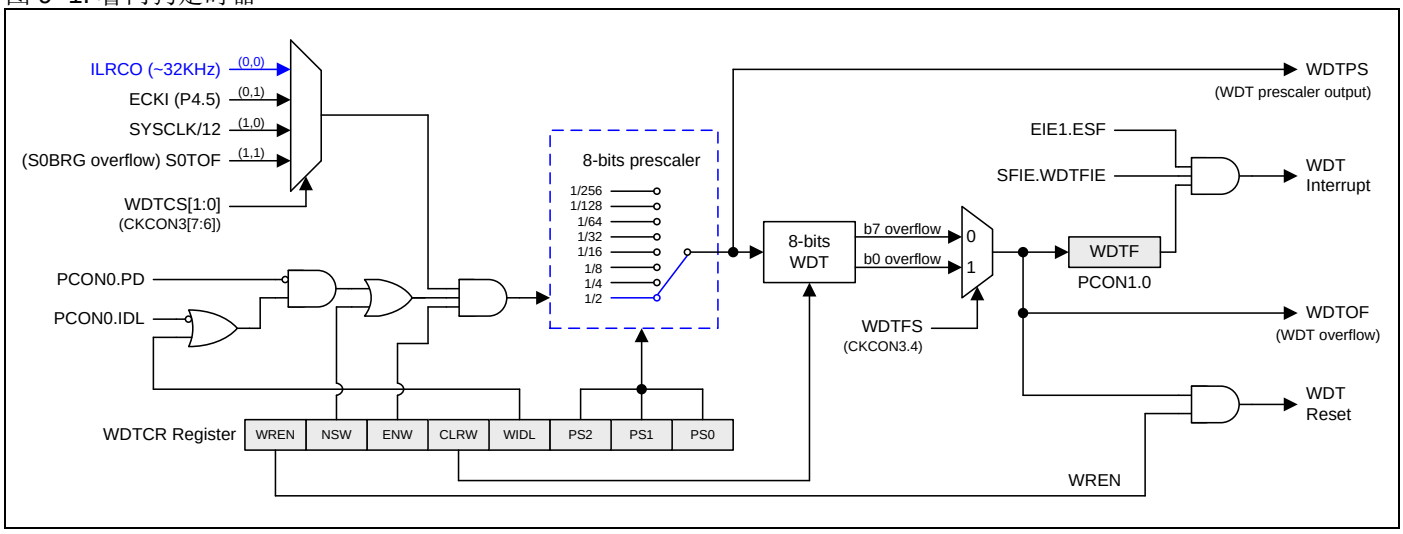
看门狗定时器 (WDT) 用来使程序从跑飞或死机状态恢复的一个手段。WDT 由一个 8 位独立定时器、一个 8 分频器和一个控制寄存器(WDTCR)组成。**MG82F6B08 / 6B001/ 6B104** 的 WDT 结构框图如图 9–1 所示。

WDT 时钟源有 **4** 种选择。在 WDT 使能之前必须配置好时钟源。默认的时钟源来自 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 具有一次使能生效功能，只要在通用 SFR 页写“1”则生效。P 页下的 SFR 访问 WDTCR 可以通过写“0”到 WDTCR.7~5 禁止 WREN，NSW 和 ENW。详情请参考章节“9.4 WDT 寄存器”和“28 P 页 SFR 访问”。

图 9–1. 看门狗定时器



9.2. WDT 在空闲期间

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 NSWDT 使能，WDT 会一直保持计数不管位 WIDL 设置情况 。

9.3. WDT 在掉电模式期间(自动唤醒)

掉电模式，如果 NSW(WDTCR.6)使能，ILRCO 不会停。MCU 进入 Watch 模式，WDT 将作为一个自动唤醒功能。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。此功能当 WDT 时钟源来自 **ILRCO** 或 **ECKI** 外部时钟输入有效。

9.4. WDT 寄存器

WDTCR:看门狗定时器控制寄存器

SFR 页 = 0~F & P

SFR 地址 = 0xE1 POR = XXX0-XXXX (0000-0111)

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，不能用软件在页 0~F 中清除，**但在 P 页中，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 在掉电模式停止计数 MCU。
1：WDT 在掉电模式(Watch Mode)或空闲模式下永远不会停止计数 MCU。一旦 NSW 已经设置，不能用软件在页 0~F 中清除，**但在 P 页中, 软件能修改其值“0”或“1”。**

Bit 5：ENW，使能 WDT 标志。
0：禁止 WDT 运行。此位仅能被 POR 清除。
1：使能 WDT。一旦 ENW 位被设置，不能用软件在页 0~F 中清除，**但在 P 页中, 软件能修改其值“0”或 “1”。**

Bit 4：CLRW，WDT 清零位。
0：写“ 0”到此位 WDT 没有任何操作。
1：写“1”到此位会清除 8 位 WDT 计数器到 000H。注意此位没有必须写“0”清除.当此位设置“1”时清除 WDT 重新计数。

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

Bit 2~0：PS2 ~ PS0，选择分频器输出作 WDT 基准时钟输入。

当 WDTFS (CKCON3.4) = 0，WDT 时钟源= ILRCO 或 SYSCLK/12

PS[2:0]	分频值	WDT 时间 (WDT 时钟= ILRCO)	WDT 时间 (WDT 时钟= SYSCLK/12) (SYSCLK = IHRCO, 16MHz)
0 0 0	2	16 ms	0.384 ms
0 0 1	4	32 ms	0.768 ms
0 1 0	8	64 ms	1.536 ms
0 1 1	16	128 ms	3.072 ms
1 0 0	32	256 ms	6.144 ms
1 0 1	64	512 ms	12.288 ms
1 1 0	128	1024 ms	24.576 ms
1 1 1	256	2048 ms	49.512 ms

当 WDTFS (CKCON3.4) = 1，WDT 时钟源= ILRCO

PS[2:0]	分频值	WDT 时间 (时钟源= ILRCO)
0 0 0	2	245 us= 125 us +120us
0 0 1	4	370 us= 250 us +120us
0 1 0	8	620 us= 500 us +120 us
0 1 1	16	1.12 ms= 1 ms+120 us
1 0 0	32	2.12 ms= 2 ms +120us
1 0 1	64	4.12 ms= 4 ms +120us
1 1 0	128	8.12 ms= 8 ms +120us
1 1 1	256	16.12 ms= 16 ms

注意：当 WDT 时钟源为 ILRCO，WDT 内部逻辑延时大约为 120us，因此，我们建议增加 120us 的 WDT 周期。

CKCON3:时钟控制寄存器 3

SFR 页 = **仅 P 页**

SFR 地址 = 0x41 复位值 = 0000-0110

7	6	5	4	3	2	1	0
WDTCS1	WDTCS0	FWKP	WDTFS	MCKD1	MCKD0	1	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6：WDTCS1~0，WDT 时钟源选择位[1:0]。

WDTCS1~0	WDT 时钟源
00	ILRCO
01	ECKI (P4.5)
10	SYSCLK/12
11	S0TOF

Bit 4：WDTFS，WDT 溢出源选择位。
0：选择 WDT 位 7 溢出作为 WDT 事件源。
1：选择 WDT 位 0 溢出作为 WDT 事件源。

PCON1:电源控制寄存器 1

SFR 页 = 0~F & P

SFR 地址 = 0x97 POR = 0000-0000

7	6	5	4	3	2	1	0
SWRF	EXRF	0	RTCF	SPWF	BOF1	BOF0	WDTF
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 1：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 WDTF。

SFIE:系统标志中断使能寄存器

SFR 页 = 0~F

SFR 地址 = 0x8E POR = 0110-0000

7	6	5	4	3	2	1	0
SIDFIE	1	1	RTCFIE	SPWIE	BOF1IE	BOF0IE	WDTFIE
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 0：WDTFIE，使能 WDTF (PCON1.0)中断。
0：禁用 WDTF 中断。
1：使能 WDTF 中断。

9.5. 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，如下所述。

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)位 ENW 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 P 页 SFR 的操作机制也不行。

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：WDT 不停止。
- ☑：使能。置位 WDTCR.NSW 使能 WDT 在掉电模式(watch 模式)也保持运行。
- ☐：禁止。清除 WDTCR.NSW 禁止 WDT 在掉电模式下(watch 模式)运行。

HWENW：硬件载入 WDTCR 的“ENW”。
- ☑：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。

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。

WDSFWP:
- ☑：使能。特殊功能寄存器 WDTCR 中的 WREN、NSW、WIDL、PS2、PS1 和 PS0 软件写保护。
- ☐：禁止。特殊功能寄存器 WDTCR 中的 WREN、NSW、WIDL、PS2、PS1 和 PS0 可被软件改写。

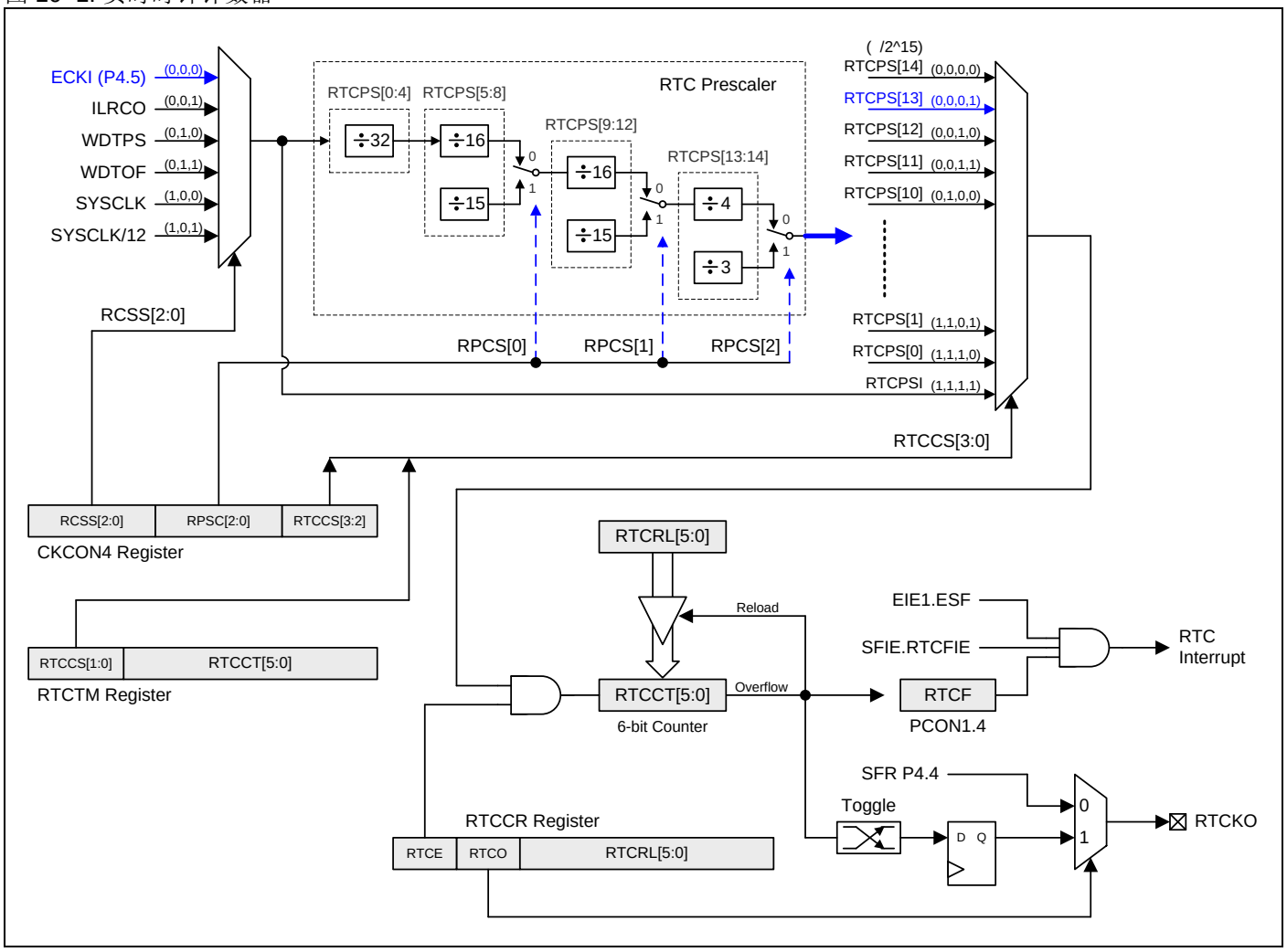
10. 实时时钟(RTC)/系统时钟

MG82F6B08 / 6B001/ 6B104 有一个简单的实时时钟允许使用者不停的记一个准确的时间在其它设备在掉电模式下。实时时钟能用于唤醒或中断源。实时时钟是一个 21 位的计数器包含 0~15 位的一个预分频器和一个 6 位的重载计数器。当其溢出，这个 6 位计数器会被重新加载并且 RTCF 旗标被设置。预分频器的时钟源有 **6** 种选择，在使能 WDT 之前需要设置 RCSS[2:0]选择一个时钟源。**MG82F6B08 / 6B001/ 6B104** 的 RTC 结构如图 10–1 所示。

RTC 模组输入是 32.768KHz ECKI 可以程控提供时间段为 0.5S 到 64S。这个计数器也可以提供一个定时功能为 SYSCLK 一个系统定时功能。最大的系统溢出时间是 SYSCLK/2^21。ILRCO 提供内部时钟源给 RTC 模块。WDT 分频器的 WDTPS 和 WDTOF 及 WDT 的溢出提供更长的分频源满足更长的唤醒时间需要。在 RTCE 使能之前 RTC 时钟源必须配置好。

RTCO 使能 RTC 溢出输出到端口引脚。只有上电复位会重置 RTC 和它相应的特殊功能寄存器为默认值。

图 10–1. 实时时钟计数器



10.1. RTC 寄存器

RTCCR:实时时钟控制寄存器

SFR 页 = **0~F & P**

SFR 地址 = **0xBE/0x54** POR = 0011-1111

7	6	5	4	3	2	1	0
RTCE	RTCO	RTCRL.5	RTCRL.4	RTCRL.3	RTCRL.2	RTCRL.1	RTCRL.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：RTCE，RTC 使能。
0：停止 RTC 计数器，RTCCT。
1：使能 RTC 计数器并且当 RTCCT 溢出时置位 RTCF，当 RTCE 被设置，CPU 不能访问 RTCTM，只有当 RTCE 被清除后才能访问。

Bit 6：RTCO，RTC 输出使能。 RTCKO 输出频率是(RTC 溢出率)/2。
0：禁止 RTCKO 输出。
1：使能 RTCKO 输出在 **P4.4**。

Bit 5~0：RTCRL[5:0]，RTC 计数器重载值寄存器。当寄存器被 CPU 访问，且 RTCCT 溢出时寄存器值会被重载到 RTCCT。

RTCTM:实时时钟定时器寄存器

SFR 页 = **0~F & P**

SFR 地址 = **0xB6/0x55** POR = 0111-1111

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

Bit 7~6：RTCCS.1~0，RTC 时钟选择。默认是“01”。

RTCCS.3~0	时钟源	RTC 中断周期	最小周期
0 0 0 0	RTCPS[14] (/2^15)	1S ~ 64S 当 **P4.5** = 32768Hz	1S
0 0 0 1	RTCPS[13] (/2^14)	0.5S ~ 32S 当 **P4.5** = 32768Hz	0.5S (默认)
0 0 1 0	RTCPS[13] (/2^13)	0.25S ~ 16S 当 **P4.5** = 32768Hz	0.25S
……	……	……	……
1 0 1 0	RTCPS[4] (/2^5)	976us ~ 62.46ms 当 **P4.5** = 32768Hz	976 us
1 0 1 1	RTCPS[3] (/2^4)		488 us
1 1 0 0	RTCPS[2] (/2^3)		244 us
1 1 0 1	RTCPS[1] (/2^2)	122us ~ 3.9ms 当 **P4.5** = 32768Hz	122 us
1 1 1 0	RTCPS[0] (/2^1)	61us ~ 1.952ms 当 **P4.5** = 32768Hz	61 us
1 1 1 1	RTCPSI (/2^0)	30.5us ~ 976us 当 **P4.5** = 32768Hz	30.5 us

Bit 5~0：RTCCT[5:0]，RTC 计数器寄存器。通过选择不同的时钟源 RTCCS[1:0]来选择 RTC 功能或系统定时功能。 当计数器溢出，置位 RTCF 旗标并且 RTCFIE 使能会产生系统旗标中断。最大的 RTC 溢出时间为 64 秒。

CKCON4:时钟控制寄存器4

SFR 页 = **仅 P 页**

SFR 地址 = 0x42 复位值 = 0000-0000

7	6	5	4	3	2	1	0
RCSS2	RCSS1	RCSS0	RPSC2	RPSC1	RPSC0	RTCCS3	RTCCS2
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~5：RTC 时钟源选择[2:0]。

RCSS2, RCSS1, RCSS0	RTC Clock Selection
0 0 0	ECKI (**P4.5**)
0 0 1	ILRCO
0 1 0	WDTPS
0 1 1	WDTOF
1 0 0	SYSCLK
1 0 1	SYSCLK / 12
1 1 0	保留
1 1 1	保留

PCON1:电源控制寄存器1

SFR 页 = 0~F & P

SFR 地址 = 0x97 POR = 0000-0000

7	6	5	4	3	2	1	0
SWRF	EXRF	0	RTCF	SPWF	BOF1	BOF0	WDTF
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 4：RTCF，RTC 溢出标志。

0：这位必须通过软件写**“1”**清除，软件写“0”不操作。

1：当 RTCCT 溢出此位仅仅被硬件置位，写**“1”**清除 RTCF。

SFIE:系统标志中断使能寄存器

SFR 页 = 0~F

SFR 地址 = 0x8E POR = 0110-0000

7	6	5	4	3	2	1	0
SIDFIE	1	1	RTCFIE	SPWIE	BOF1IE	BOF0IE	WDTFIE
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 4：RTCFIE，使能 RTCF(PCON1.4)中断。

0：禁止 RTCF 中断。

1：使能 RTCF 中断。如果使能。RTCF 能唤醒 CPU 在空闲模式或掉电模式。

11. 系统复位

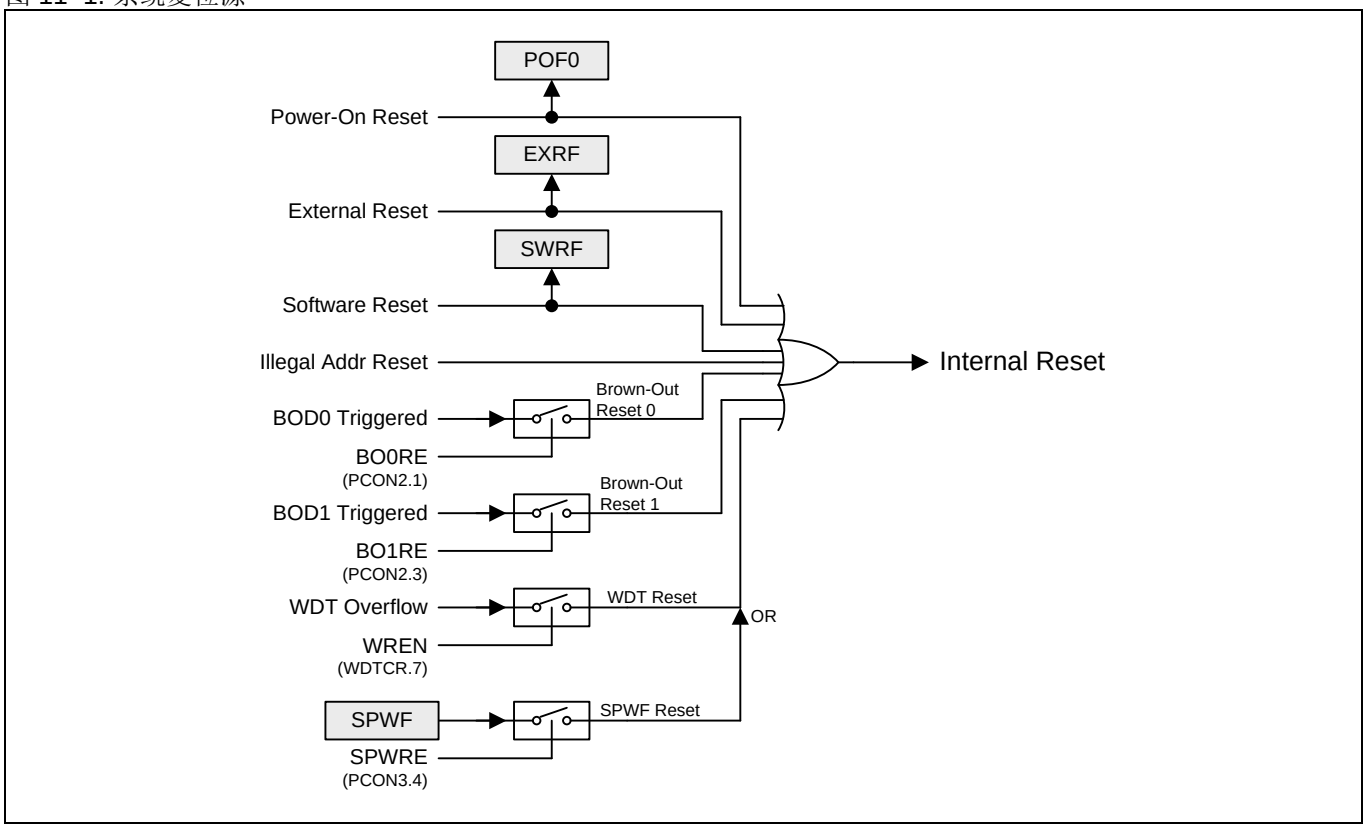
复位期间， 所有的 I/O 寄存器都设置为初始值，程序会根据 OR 设置选择从复位向量的 0000H 开始运行，或者从 ISP 地址开始运行。**MG82F6B08 / 6B001/ 6B104** 有 **8** 种复位源：上电复位，外部复位，软件复位，非法地址复位，低电压监测 0 复位，低电压监测 1 复位和 WDT 复位。系统复位源(**MG82F6B08 / 6B001/ 6B104**) 如
图 11–1 所示。

下面的选项描述复位产生源及其相应的控制寄存器和指示标志。

11.1. 复位源

MG82F6B08 / 6B001/ 6B104 复位系统和所有的复位源如
图 11–1 所示。

图 11–1. 系统复位源



11.2. 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 页 = 0~F & P　　POR = 0001-0000
SFR 地址 = 0x87　　复位值 = 000X-0000

7	6	5	4	3	2	1	0
SMOD1	SMOD0	GF	POF0	GF1	GF0	PD	IDL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 4：POF0，上电复位标志 0。
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置位。POF0 也能有软件置位。

上电标志 POF0 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、低电压监测复位、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：POF0 必须由软件清除。

11.3. 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 RST 引脚上连接可靠的硬件复位电路。

PCON1: 电源控制寄存器 1

SFR 页 = 0~F & P
SFR 地址 = 0x97　　POR = 0000-0000

7	6	5	4	3	2	1	0
SWRF	EXRF	0	RTCF	SPWF	BOF1	BOF0	WDTF
R/W	R/W	R/W	R/W	R/W	R/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 EXRF。

11.4. 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 页 = 0~F
SFR 地址 = 0xE7　　POR = 0000-0000

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

Bit 6：SWBS，软件执行起始选择控制。
0：复位软件从 AP 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：写“0”无操作。
1：写“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 页 = 0~F & P

SFR 地址 = 0x97 POR = 0000-0000

7	6	5	4	3	2	1	0
SWRF	EXRF	0	RTCF	SPWF	BOF1	BOF0	WDTF
R/W	R/W	R/W	R/W	R/W	R/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 SWRF。

11.5. 低电压检测复位

MG82F6B08 / 6B001/ 6B104 中，有两个低电压监测器(BOD0 和 BOD1)监测电源电压(VDD)，低电压监测器(BOD0) 的监测固定点为 VDD=2.35V，低电压监测器(BOD1)的监测固定点可以被软件选择为 VDD= **4.2V, 3.6V, 2.4V & 2.7V**，如果 VDD 电压低于 BOD0 或 BOD1 监测点，则置位相关联的 BOF0 和 BOF1 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 指示一个低电压监测器(BOD0)复位发生；如果 BO1RE (PCON2.3)被使能，BOD1 事件将触发一个 CPU 复位并置位 BOF1 指示一个低电压监测器(BOD1)复位发生。

BOD1 复位注意事项

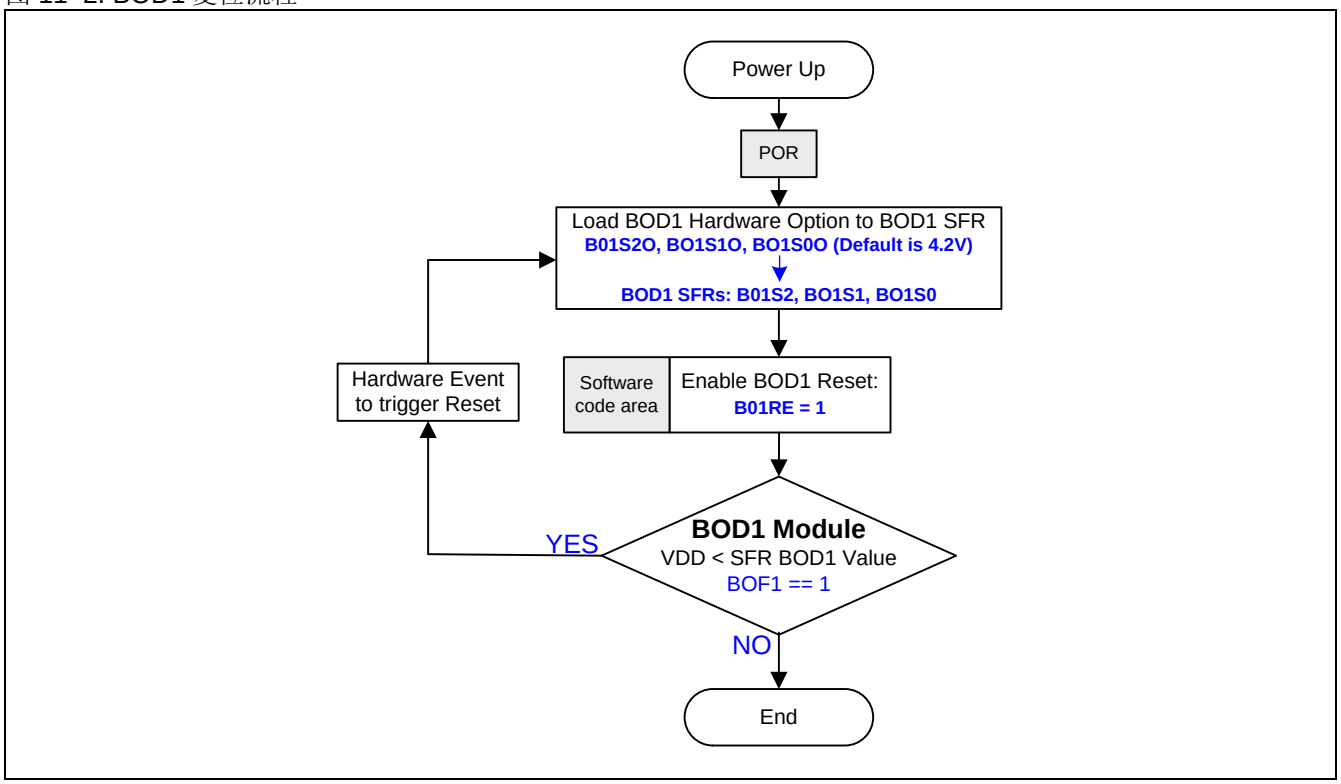
当 BOD1 复位，会重载 BOA1 硬件选项位(B01S2O, BO1S1O, BO1S0O，请参照“30 硬件选项”章节以获取细节)进入 BOD1 SFR (B01S2, BO1S1, BO1S0)。

有两种方式控制 BOD1 复位：

1. 硬件选项：
   a. 使用 H/W 选项设置 BOD1 电压检测电平。
   b. 设置 HW 选项 BO1REO 使能 BOD1 复位，用户也可以在软件中按需使能或禁用 BOD1 复位。

图 11–2 展示了 BOD1 复位工作流程。

图 11–2. BOD1 复位流程

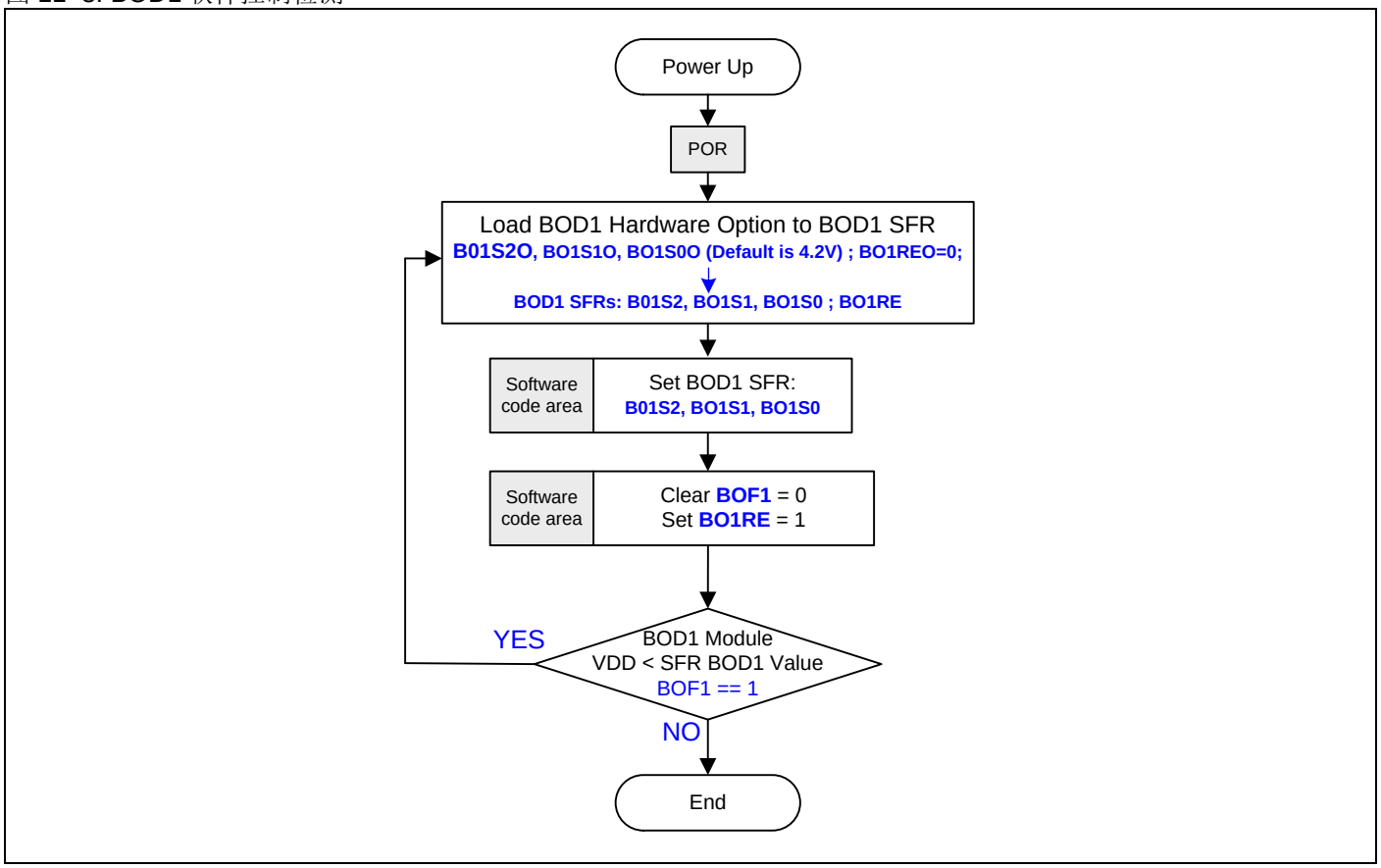


软件控制 BOD1：

a. 不通过硬件选项位 BO1REO 使能 BOD1（默认禁用）
b. 通过 BO1S[2:0]设置 BOD1 检测电压
c. 清除 BOD1 标志，BOF1 = 0
d. 使能 BOD1 复位

图 11–3 是 BOD1 软件控制流。

图 11–3. BOD1 软件控制检测



PCON1: 电源控制寄存器 1

SFR 页 = 0~F & P

SFR 地址 = 0x97 POR = 0000-0000

7	6	5	4	3	2	1	0
SWRF	EXRF	0	RTCF	SPWF	BOF1	BOF0	WDTF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 2：BOF1，BOF1(复位)标志。
0：这位必须通过软件写“**1**”清零，写“0”无操作。
1：当 VDD 电压碰到 BOD1 监测点时，硬件置位此位，写“**1**”清零。如果 BO1RE(PCON2.3)被使能，BOD1 事件将触发一个 CPU 复位并置位 BOF1 指示一个低电压监测器(BOD1)复位发生。

Bit 1：BOF0，BOF0(复位)标志。
0：这位必须通过软件写“**1**”清零，写“0”无操作。
1：当 VDD 电压碰到 BOD0 监测点时，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 指示一个低电压监测器(BOD0)复位发生。

11.6. 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 页 = 0~F & P
SFR 地址 = 0x97 POR = 0000-0000

7	6	5	4	3	2	1	0
SWRF	EXRF	0	RTCF	SPWF	BOF1	BOF0	WDTF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 0：WDTF，WDT 溢出/复位标志。
0：这位必须通过软件写“**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。

11.7. 非法地址复位

MG82F6B08 / 6B001/ 6B104 如果软件程序运行到非法的地址比如超出程序空间(ROM)范围，将触发 CPU 复位并且置标志位。

11.8. 堆栈检测复位

SPHB: 堆栈高位边界

SFR 页 = P Only
SFR 地址 = 0x53 复位值 = 1111-1111

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

SPHB，用于堆栈边界检测报警。
若 SPHB == 1111-1111, SPWF 会被置起，当 SP ≥ 1111-1111.
若 SPHB == 1111-0000, SPWF 会被置起，当 SP ≥ 1111-0000.

PCON1: 电源控制寄存器 1

SFR 页 = 0~F & P
SFR 地址 = 0x97 POR = 0000-0000

7	6	5	4	3	2	1	0
SWRF	EXRF	0	RTCF	SPWF	BOF1	BOF0	WDTF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 3：SPWF，SP 警告标志。
0：这位必须通过软件写“**1**”清零，写“0”无操作。
1：当 SP ≥ SPHB 时硬件置位此位，当 SP < SPHB 时写“**1**”清零。

PCON3: 电源控制寄存器 3

SFR 页 = **仅 P 页**
SFR 地址 = 0x45 POR = 0000-0000

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

Bit 4：SPWRE，SPWF 触发 MCU 复位。
0：禁用 SPWF 触发 MCU 复位。
1：使能 SPWF 触发 MCU 复位。

12. 电源管理

MG82F6B08 / 6B001/ 6B104 支持两个电源监测模块(低电压监测器(BOD0 和 BOD1)模块)，和 7 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、RTC 模式、Watch 模式和 Monitor 模式。

BOD0 和 BOD1 通过 BOF0 和 BOF1 标志位报告电源状态，软件可以通过这个状态产生中断或复位。7 种电源节能模式提供不同的节能应用，通过对 CKCON0、CKCON2、CKCON3、CKCON4、PCON0、PCON1、PCON2、PCON3、PCON4、RTCCR 和 WDTCR 寄存器的访问来操作这些电源事件。

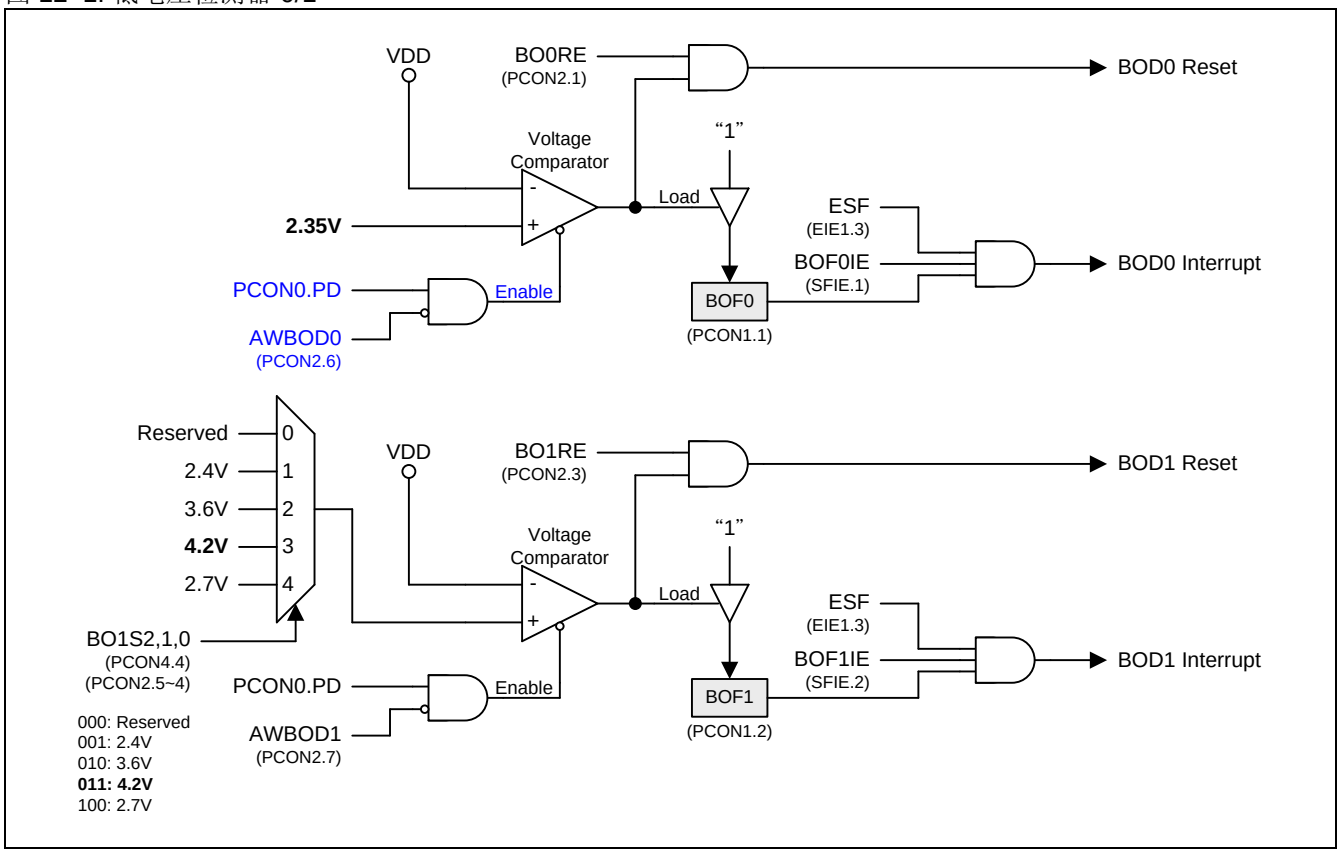
12.1. 低电压检测器

MG82F6B08 / 6B001/ 6B104 有两个低电压监测器 (BOD0& BOD1)通过比较固定的触发电压来监测芯片电压，图 12–1 是 BOD0 和 BOD1 功能逻辑图，BOD0 监测固定触发电压为 VDD=**2.35V**，BOD1 监测固定触发电压为 VDD=(4.2V/3.6V/2.4V/2.7V)。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，BOD1 有同样的标志 BOF1，也有同样的中断功能。如果 AWBOD1 (PCON2.7)使能，这个中断(BOD1)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。BOD1 也有同样的复位功能设置相关的控制位 BO1RE(PCON2.3)，如果 AWBOD1 (PCON2.7)位被使能，BOD1 也能重新启动掉电模式。

如果 BOD1 在应用中没有使用，为了节省功耗可以通过软件清除 EBOD1 (PCON2.2)来禁止 BOD1。

图 12–1. 低电压检测器 0/1



12.2. 省电模式

12.2.1. 慢频模式

程序设置位 SCKS2~SCKS0(CKCON0 寄存器，参考章节“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。

12.2.2. 副频模式

设置 OSCS1~0 选择 ILRCO 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，参考章节“8 系统时钟”) 使用者可以使 MCU 的速度最低到 250HZ。

12.2.3. RTC 模式

MG82F6B08 / 6B001/ 6B104 有一个简单的 RTC 模块允许用户在设备部分掉电时继续运行准确的定时器。在 RTC 模式，RTC 模块作为一个“时钟”功能并且能在 RTC 溢出时唤醒芯片的掉电模式。详细描述请参考章节“10 实时时钟(RTC)/系统时钟”。

12.2.4. Watch 模式

如果看门狗被使能并且位 NSW 被设置，看门狗在掉电模式保持运行支持自动唤醒功能，这个在 **MG82F6B08 / 6B001/ 6B104** 应用中叫 Watch 模式。当 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，更详细信息请参考章节“9 看门狗定时器(WDT)”和章节“14 中断”。

12.2.5. Monitor 模式

如果 AWBOD1(PCON2.3)被设置，BOD1 即使在掉电模式下，低电压监测功能 BOD1 会有效，这就是 **MG82F6B08 / 6B001/ 6B104** 应用中的 Monitor 模式。当 BOD1 触发到监测电压，软件选择中断或系统复位来唤醒 CPU 并硬件置位 BOF1，更详细信息请参考章节“12.1 低电压检测器”和章节“14 中断”。

12.2.6. 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、定时器 2、KBI、ADC、AC0、S0、TWI0/I2C0、RTC、BOD0 和 BOD1 仍然处于工作状态。在空闲模式下 PCA 和 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。

当 MCU 在空闲模式和掉电模式时为了降低功耗 ADC 输入通道必须设置为“***仅模拟输入***”。

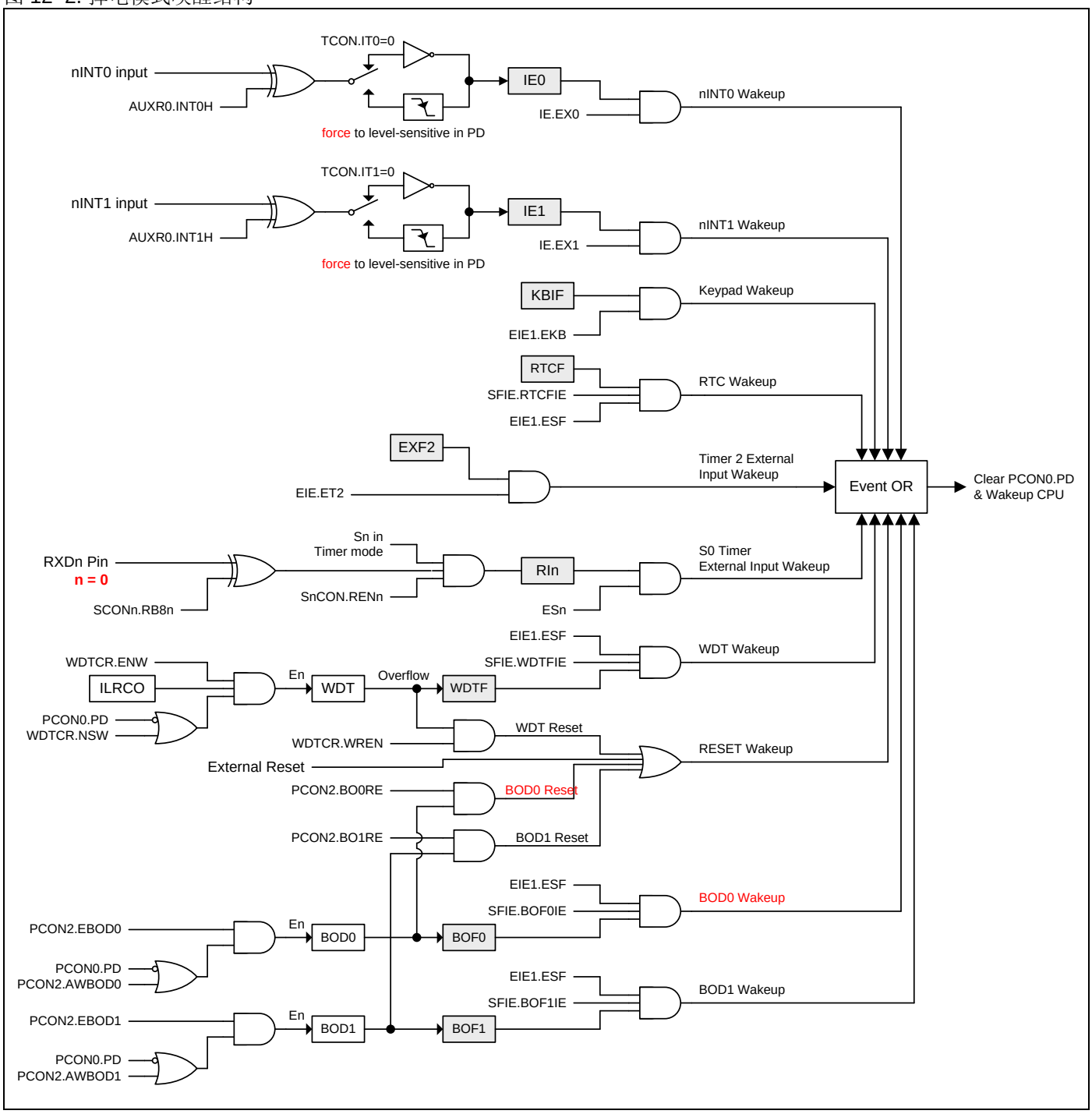
12.2.7. 掉电模式

可以通过软件的方法置位 PCON0.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、使能的外部中断、使能的 KBI、使能的 RTC (RTC 模式) 、使能的 BOD1(monitor 模式)或使能的不停止的 WDT 都能使系统退出掉电模式。

如果有下列情况发生，使用者至少要等 4us 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 为非悬浮状态。

MG82F6B08 / 6B001/ 6B104 掉电模式唤醒结构如图 12–2 所示。

图 12–2. 掉电模式唤醒结构



12.2.8. 中断唤醒掉电模式

2 个外部中断都能终止掉电模式， 外部中断 nINT0 和 nINT1 能退出掉电模式。为了能唤醒掉电模式， 中断 nINT0 或 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。

12.2.9. 复位唤醒掉电模式

外部复位唤醒掉电模式有点类似于中断。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。

12.2.10. KBI 键盘唤醒掉电模式

MG82F6B08 / 6B001/ 6B104 中 KBI.7~0 具有键盘中断唤醒功能，通过使能 KBI 模块的控制寄存器。软件可以设置不同的端口引脚作为 KBI 输入。

通过使能 KBI 唤醒掉电模式有点类似中断唤醒。在 KBI 模式下且已经使能 KBI 中断(EIE1.5，EKB)，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 会响应 KBI 中断并执行中断服务程序。

12.3. 电源控制寄存器

PCON0: 电源控制寄存器 0

SFR 页 = 0~F & P　　POR = 0001-0000
SFR 地址 = 0x87　　复位值 = 000X-0000

7	6	5	4	3	2	1	0
SMOD1	SMOD0	GF	POF0	GF1	GF0	PD	IDL
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 4：POF0，上电标志 0。
0：这位必须由软件写“**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：CPU 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。

Bit 0：IDL，空闲模式控制位。
0：CPU 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。

PCON1: 电源控制寄存器 1

SFR 页 = 0~F & P
SFR 地址 = 0x97　　POR = 0000-0000

7	6	5	4	3	2	1	0
SWRF	EXRF	0	RTCF	SPWF	BOF1	BOF0	WDTF
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：SWRF，软件复位标志。
0：这位必须由软件写“**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：这位必须由软件写“**1**”清零。
1：当外部复位产生时硬件置位此位。

Bit 4：RTCF，RTC 溢出标志。
0：这位必须由软件写“**1**”清零。写“0”无操作。
1：当 RTCCT 溢出时此位仅由硬件置位。写“**1**”则清除 RTCF。

Bit 3：SPWF，SP 警告标志。
0：这位必须通过软件写“**1**”清零，写“0”无操作。
1：当 SP ≥ SPHB 时硬件置位此位，当 SP < SPHB 时写“**1**”清零。

Bit 2：BOF1，低电压监测标志 1。
0：这位必须由软件写“**1**”清零。
1：当电源电压触及到低电压监测器 1 电压(4.2V/3.6V/2.4V/2.7V)时，硬件置位此位

Bit 1：BOF0，低电压监测标志 0。
0：这位必须由软件写“**1**”清零。
1：当电源电压触及到低电压监测器 0 电压(**2.35V**)时，硬件置位此位。

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

PCON2: 电源控制寄存器 2

SFR 页 =仅 **P** 页

SFR 地址 = 0x44 POR = 0000-0101

7	6	5	4	3	2	1	0
AWBOD1	0	BO1S1	BO1S0	BO1RE	EBOD1	BO0RE	1
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：AWBOD1，掉电模式(PD)下 BOD1 的唤醒。
0：掉电模式(PD)下禁止 BOD1。
1：掉电模式(PD)下保持 BOD1。

Bit 6：保留位。当写 PCON2 寄存器时，这位软件必须写“0”。

Bit 5~4：BO1S[1:0]，低电压监测器 1 监测电压选择。(BO1S2 在 PCON4.4)

BO1S[2:0]	BOD1 监测电压
0 0 0	保留
0 0 1	2.4V
0 1 0	3.6V
0 1 1	4.2V
1 0 0	2.7V
其它	保留

Bit 3：BO1RE，BOD1 复位使能。
0：当 BOF1 已经设置，禁止低电压监测 1(BOD1)系统复位。
1：当 BOF1 已经设置，使能低电压监测 1(BOD1)系统复位。

Bit 2：EBOD1，使能 BOD1 监测 VDD 下降到 BO1S1~0 设置的固定值。
0：禁止 BOD1 监测电源电压降低芯片功耗。
1：使能 BOD1 监测电源电压 VDD。

Bit 1：BO0RE，BOD0 复位使能。
0：当 BOF0 已经设置，禁止低电压监测 0(BOD0)系统复位。
1：当 BOF0 已经设置，使能低电压监测 0(BOD0)系统复位(VDD 触到 2.35V)。

Bit 0：保留位。当写 PCON2 寄存器时，这位软件必须写“**1**”。

PCON3: 电源控制寄存器 3

SFR 页 = 仅 **P** 页

SFR 地址 = 0x45 POR = 0000-0000

7	6	5	4	3	2	1	0
IVREN	0	0	SPWRE	0	0	0	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：IVREN，内部参考电压使能。
0：禁止片内 IVR (1.4V)。
1：使能片内 IVR (1.4V)。

Bit 6~5：保留位。当写 PCON3 寄存器时，这些位软件必须写“0”。

Bit 4：SPWRE，SPWF 触发 MCU 复位。
0：禁用 SPWF 触发 MCU 复位。
1：使能 SPWF 触发 MCU 复位。

Bit 3~0：保留位。当写 PCON3 寄存器时，这些位软件必须写“0”。

PCON4: 电源控制寄存器 4

SFR 页 = **仅 P 页**

SFR 地址 = 0x46 POR = 0000-0000

7	6	5	4	3	2	1	0
0	0	0	BO1S2	0	0	0	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 4：BOD1 监测电压选择位 2。

13. I/O 口配置

MG82F6B08 / 6B001/ 6B104 有下列 I/O 端口：P1.0, P3.0, P3.1, P3.2, P3.3, P4.4, P4.5, P4.6 和 P4.7。如果禁止外部复位脚功能，P4.7 是有效的。准确的可用 I/O 引脚数量由封装类型决定。见表 13–1。

表 13–1. 可用 I/O 引脚数量

封装类型	I/O 封装类型	引脚数量
10-pin	P1.0, P3.0, P3.1, P3.3, P4.4, P4.5, P4.6, P4.7	8 or 7 (RST 选择)
8-pin	P3.0, P3.1, P3.3, P4.4, P4.5, P4.6	6
8-pin (MG82F6B104)	P3.0, P3.1, P3.2, P3.3, P4.4, P4.5	6

13.1. IO 结构

MG82F6B08 / 6B001/ 6B104 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)。 端口 3 缺省值是弱上拉的准双向口模式。

其它口属于第二类，这些口有四种模式分别是仅模拟输入、上拉电阻的集电极开漏输出、集电极开漏输出和推挽输出。默认设置是仅模拟输入，也就意味着带有高阻状态的输入模式。

下面章节描述所有类型的 I/O 模式的配置。

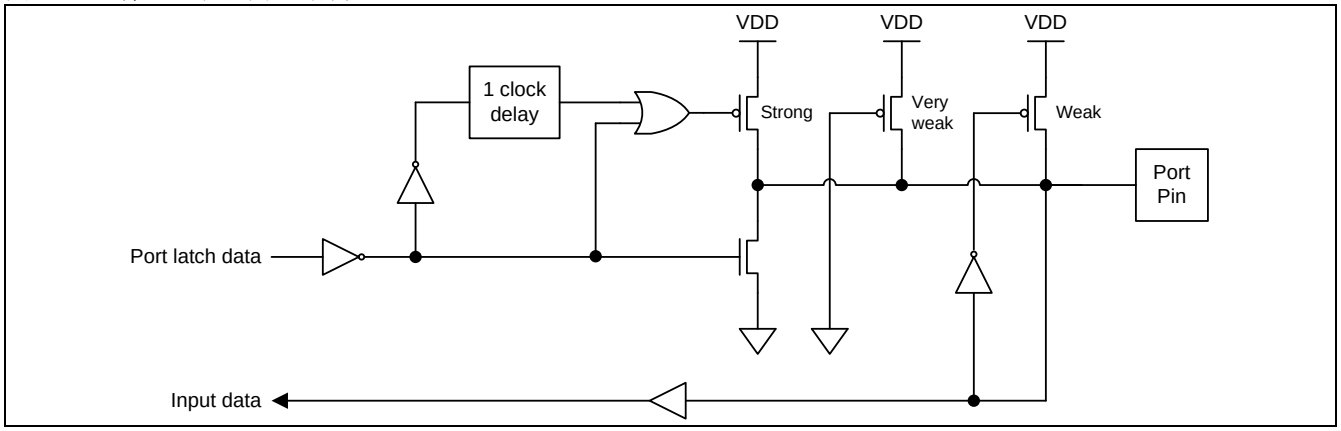
13.1.1. 端口 3 准双向口结构

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。

端口 3 准双向口结构如图 13–1 所示。

图 13–1. 端口 3 准双向口结构

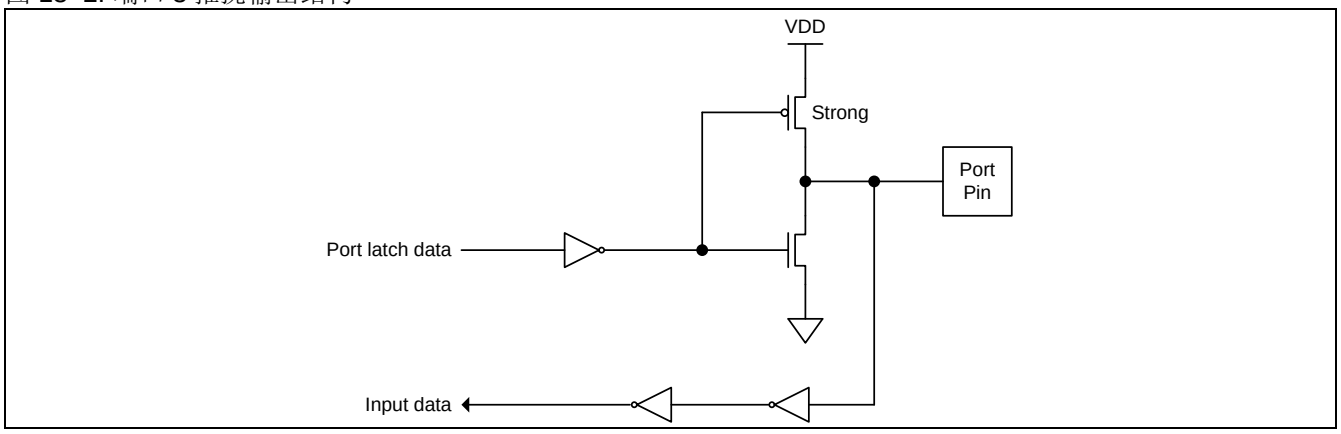


13.1.2. 端口 3 推挽输出结构

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。

推挽输出结构见图 13–2。

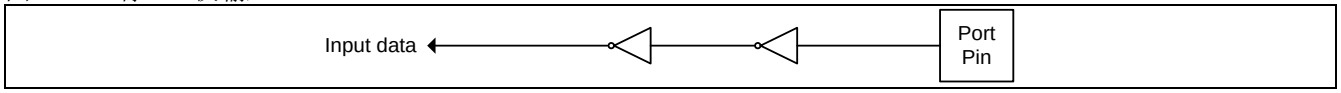
图 13–2. 端口 3 推挽输出结构



13.1.3. 端口 3 仅输入(高阻抗输入)结构

仅输入配置在端口 3 引脚上没有任何上拉电阻，如下图 13–3 所示。

图 13–3. 端口 3 仅输入

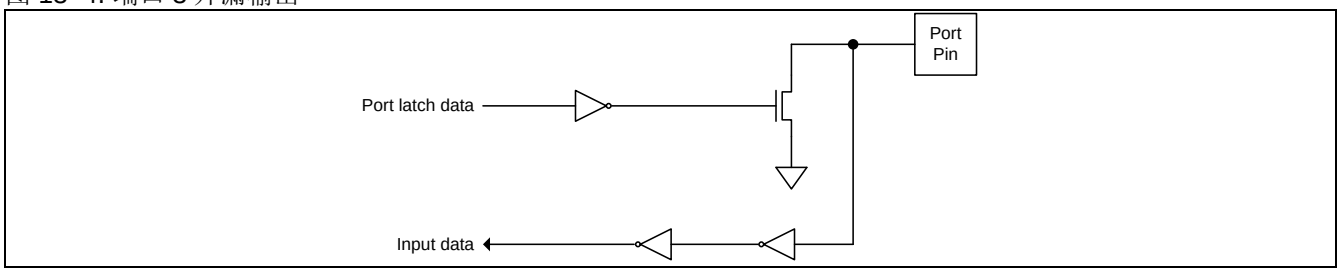


13.1.4. 端口 3 开漏输出结构

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。

端口 3 开漏输出结构如图 13–4 所示。

图 13–4. 端口 3 开漏输出

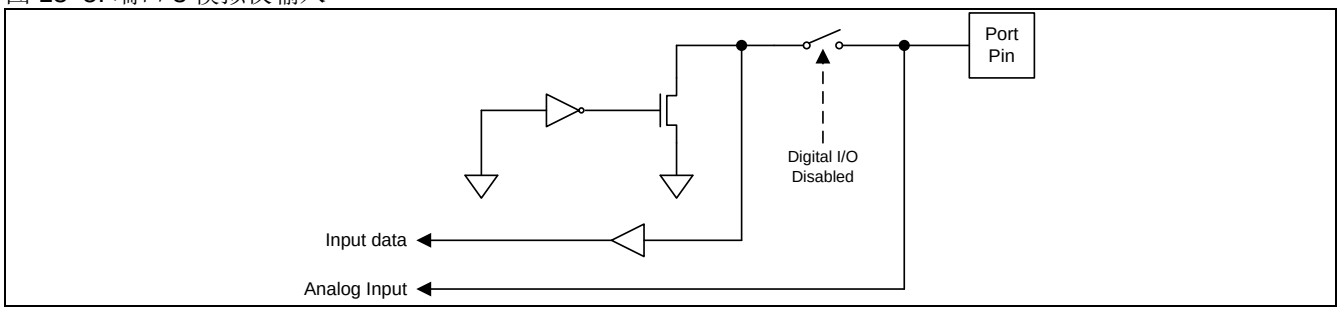


13.1.5. 端口 3 模拟输入结构

端口 3 的模拟仅输入结构没有任何数字功能，作为 ADC 或模拟比较器输入应用，用户可以保持这种结构的端口设置。如果应用在数字功能的端口引脚，用户必须把端口引脚配置成相关联的结构。

仅模拟输入端口结构如图 13–5 所示。

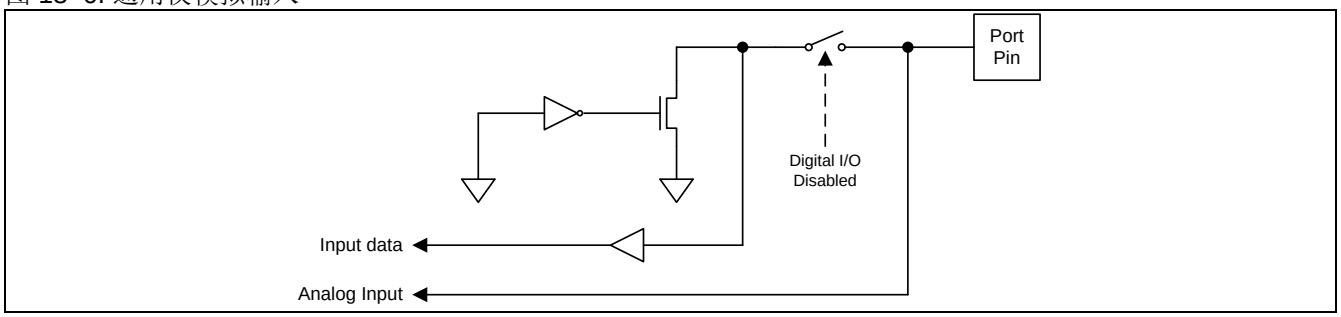
图 13–5. 端口 3 模拟仅输入



13.1.6. 通用仅模拟输入结构

在通用端口引脚上仅模拟输入结构是默认设置。作为 ADC 或模拟比较器输入应用，用户可以保持这种结构的端口设置。如果应用在数字功能的端口引脚，用户必须把端口引脚配置成相关联的结构。仅模拟输入端口结构如图 13–6 所示。

图 13–6. 通用仅模拟输入

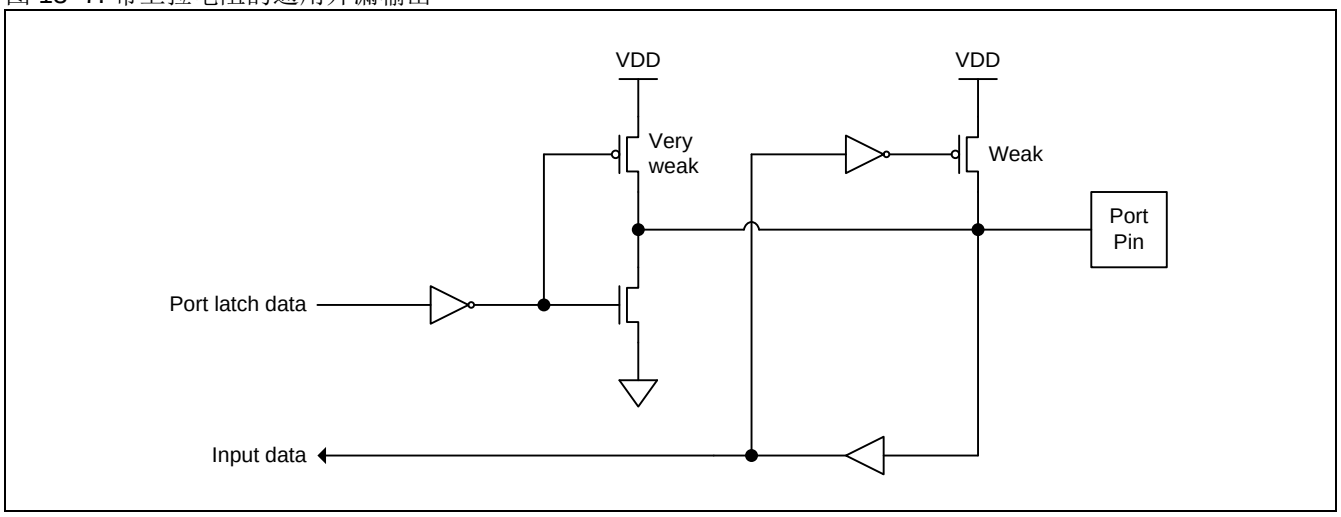


13.1.7. 带上拉电阻的通用开漏输出

带上拉电阻的开漏输出结构是在通用端口引脚上使能开漏输出模式的片内上拉电阻。

带上拉电阻的开漏输出端口结构如图 13–7 所示。

图 13–7. 带上拉电阻的通用开漏输出

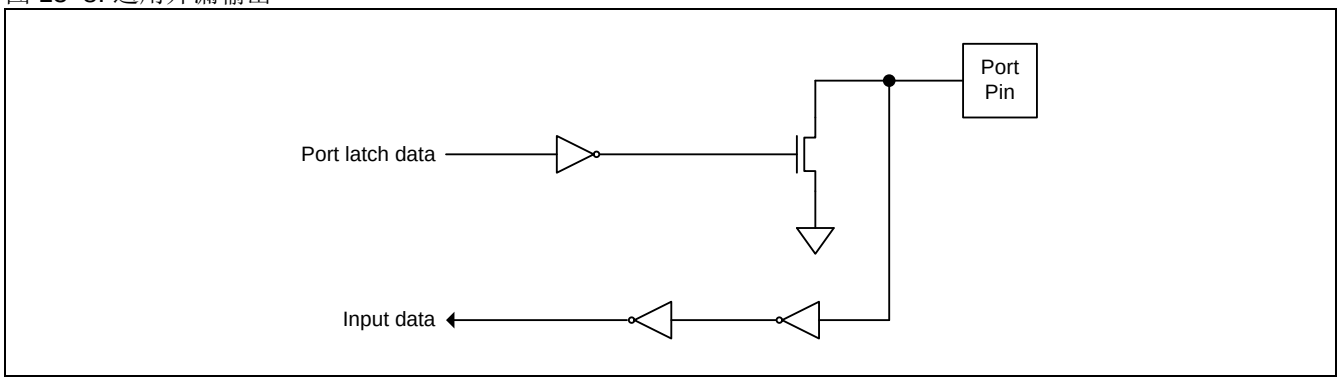


13.1.8. 通用开漏输出结构

通用端口引脚上的开漏输出结构跟端口 3 的开漏输出模式一样的功能。

通用端口集电极开漏结构如图 13–8 所示。

图 13–8. 通用开漏输出



13.1.9. 通用端口的数字输入结构

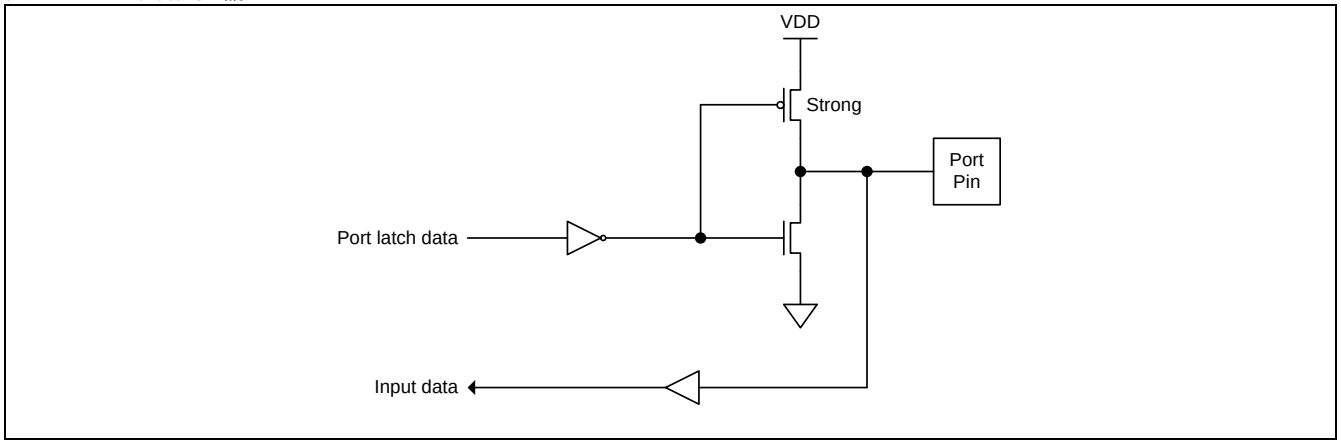
通过设置输出模式为“开漏”并且写逻辑“1”到端口数字寄存器的相应位来配置端口引脚为数字输入。例如，通过设置 P1M0.0=0、P1M1.0=0 并且 P1.0=1，这样 P1.0 配置为数字输入。

13.1.10. 通用推挽输出结构

通用端口引脚上的推挽输出结构跟端口 3 的推挽输出模式一样的功能。

通用端口推挽输出结构如图 13–9 所示。

图 13–9. 通用推挽输出



13.1.11. 端口引脚输出驱动力选择

MG82F6B08 / 6B001/ 6B104 输入/输出有两种驱动力可选适合不同应用的输出阻抗。详情请参考章节“13.2.4 端口输出驱动力控制寄存器”。

13.1.12. 端口引脚输出快速驱动选择

MG82F6B08 / 6B001/ 6B104 输入/输出有两种驱动速度可选适合不同应用的输出频率。详情请参考章节“13.2.5 端口输出快速驱动控制寄存器”。

13.2. I/O 口寄存器

MG82F6B08 / 6B001/ 6B104 所有的端口可通过软件个别的、独立的配置其工作模式。端口 3 有 4 种工作模式，如表 13–2。两个模式寄存器用于选择每个端口 3 引脚的输出类型。仅端口 3 支持准双向模式且系统复位之后它们为准双向模式。

表 13–2. 端口 3 配置设定

P3xAM	P3M0.y	P3M1.y	端口模式
0	**0**	**0**	**准双向（默认）**
0	0	1	推挽输出
0	1	0	仅输入（高阻抗输入）
0	1	1	集电极开漏输出
1	0	0	仅模拟输入

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。

这里 x=0、3 定义了控制位、P30AM 和 P33AM，用他们来选择 P3.0 和 P3.3 位仅模拟输入。

其它的通用口引脚有四种模式见表 13–3。二个模式寄存器位选择每个引脚的输出类型且系统复位之后这些端口引脚为仅输入。

表 13–3. 通用端口配置设定

PxM0.y	PxM1.y	端口模式
0	**1**	**仅输入（默认）**
1	1	带上拉电阻的集电极开漏
0	0	集电极开漏输出/通用数字输入(端口引脚设置为“1”)
1	0	推挽输出

这里 x= **1, 4** (端口)，y=0~7(端口引脚号)。寄存器 PxM0 和 PxM1 列举了每个引脚的描述。

13.2.1. 端口 1 寄存器

P1:端口 1 寄存器

SFR 页 = 0~F

SFR 地址 = 0x90 复位值 = 1111-1111

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

Bit 7~0：端口 1 输出数据通过 CPU 置位/清零。

P1M0:端口 1 模式寄存器 0

SFR 页 = 0~F

SFR 地址 = 0x91 复位值 = 0000-0000

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

P1M1:端口 1 模式寄存器 1

SFR 页 = **仅 0 页**

SFR 地址 = **0x92** 复位值 = 1111-1111

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

13.2.2. 端口 3 寄存器

P3:端口 3 寄存器

SFR 页 = 0~F
SFR 地址 = 0xB0 复位值 = 1111-1111

7	6	5	4	3	2	1	0
1	1	1	1	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：端口 3 输出数据通过 CPU 置位/清零。

P3M0:端口 3 模式寄存器 0

SFR 页 = 0~F
SFR 地址 = 0xB1 复位值 = 0000-0000

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

P3M1:端口 3 模式寄存器 1

SFR 页 = 0~F
SFR 地址 = 0xB2 复位值 = 0000-0000

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

AUXR11:辅助寄存器 11

SFR 页 = 仅 8 页
SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P30AM	P33AM	0	0	0	POEM0	C0M0	C0OFS
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7：P30AM，P3.0 模拟输入模式使能。
0：P3.0 通用输入输出模式由 P3M0 和 P3M1 控制。
1：设置 P3.0 为模拟输入模式作为 ADC12 的 AIN4 输入。

Bit 6：P33AM，P3.3 模拟输入模式使能。
0：P3.3 通用输入输出模式由 P3M0 和 P3M1 控制。
1：设置 P3.3 为模拟输入模式作为 ADC12 的 AIN5 输入。

13.2.3. 端口 4 寄存器

P4:端口 4 寄存器

SFR 页 = 0~F
SFR 地址 = 0xE8 复位值 = 1111-1111

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

Bit 7~0：端口 4 输出数据通过 CPU 置位/清零。
P4.5 和 P4.4 复用 OCD_SDA 和 OCD_SCL。
P4.7 复用 RST 输入。

P4M0:端口 4 模式寄存器 0
SFR 页 = **仅 0 页**
SFR 地址 = **0xB3** 复位值 = 1011-0000

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

注意：当 P4.7/RST 用作普通 IO 功能时，强烈建议不要将其设置为输入，以避免 MCU 上电过程中外部输入高电平导致 MCU 锁死在复位状态。

P4M1:端口 4 模式寄存器 1
SFR 页 = **仅 2 页**
SFR 地址 = **0x92** 复位值 = 1111-1111

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

注意：当 P4.7/RST 用作普通 IO 功能时，强烈建议不要将其设置为输入，以避免 MCU 上电过程中外部输入高电平导致 MCU 锁死在复位状态。

13.2.4. 端口输出驱动力控制寄存器

MG82F6B08 / 6B001/ 6B104 除了 P4.7 之外所有端口引脚都有二种软件可选的驱动力。请参考端口引脚的驱动力信息。

PDRVC0:端口驱动力控制寄存器 0
SFR 页 = **仅 2 页**
SFR 地址 = **0xB4** 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	P3DC0	0	0	0	P1DC0	0	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 6：P3DC0，端口 3 低 4 位输出驱动力控制。
0：P3.3 ~ P3.0 输出选择为高驱动力。
1：P3.3 ~ P3.0 输出选择为低驱动力。

Bit 2：P1DC0，端口 1 低 4 位输出驱动力控制。
0：P1.3 ~ P1.0 输出选择为高驱动力。
1：P1.3 ~ P1.0 输出选择为低驱动力。

PDRVC1:端口驱动力控制寄存器 1
SFR 页 = **仅 3 页**
SFR 地址 = **0xB4** 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	0	0	0	0	0	P4DC1	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~2：保留位。当写 PDRVC1 寄存器时软件，这些位必须写 “0”。

Bit 1：P4DC1，端口 4 高 4 位输出驱动力控制。
0：P4.6 ~ P4.4 输出选择为高驱动力。
1：P4.6 ~ P4.4 输出选择为低驱动力。

Bit 0：保留位。当写 PDRVC1 寄存器时软件，这位必须写 “0”。

13.2.5. 端口输出快速驱动控制寄存器

MG82F6B08 / 6B001/ 6B104 所有端口引脚(除了 P4.7 之外)都有二种软件可选的快速驱动。请参考端口引脚的快速驱动信息。

P3FDC:端口 3 快速驱动控制寄存器
SFR 页 = **仅 7 页**
SFR 地址 = **0x92** 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	0	0	0	P3FDC.3	P3FDC.2	P3FDC.1	P3FDC.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**RW**	**RW**

Bit 3~0：端口 3 输出快速驱动控制通过 CPU 置位/清零。
0：禁止端口引脚输出快速驱动。
1：使能端口引脚输出快速驱动。

P1FDC:端口 1 快速驱动控制寄存器
SFR 页 = **仅 8 页**
SFR 地址 = **0x92** 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	P1FDC.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**RW**	**RW**

Bit 0：端口 1 输出快速驱动控制通过 CPU 置位/清零。
0：禁止端口引脚输出快速驱动。
1：使能端口引脚输出快速驱动。

P4FDC:端口 4 快速驱动控制寄存器
SFR 页 = **仅 A 页**
SFR 地址 = **0x92** 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	P4FDC.6	P4FDC.5	P4FDC.4	0	0	0	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：保留位。当写 P4FDC 寄存器时软件，这些位必须写 “0”。

Bit 6~4：端口 4 输出快速驱动控制通过 CPU 置位/清零。
0：禁止端口引脚输出快速驱动。
1：使能端口引脚输出快速驱动。

Bit 3~0：保留位。当写 P4FDC 寄存器时软件，这些位必须写 “0”。

14. 中断

MG82F6B08 / 6B001/ 6B104 有 **12** 个带 4 级优先级的中断源。这些中断源有几个特殊功能寄存器 SFR 与设定四个级别的中断优先级相关。这些特殊功能寄存器分别是 IE, IP0L, IP0H, EIE1, EIP1L 和 EIP1H。IP0H(中断优先级 0 高字节)、EIP1H(扩展中断优先级 1 高字节)寄存器使四个级别的中断结构合理分配。四个级别的中断优先级在处理这些中断源时更加灵活。

14.1. 中断结构

表 14–1 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位或清零，这和硬件置位或清零结果相同。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。

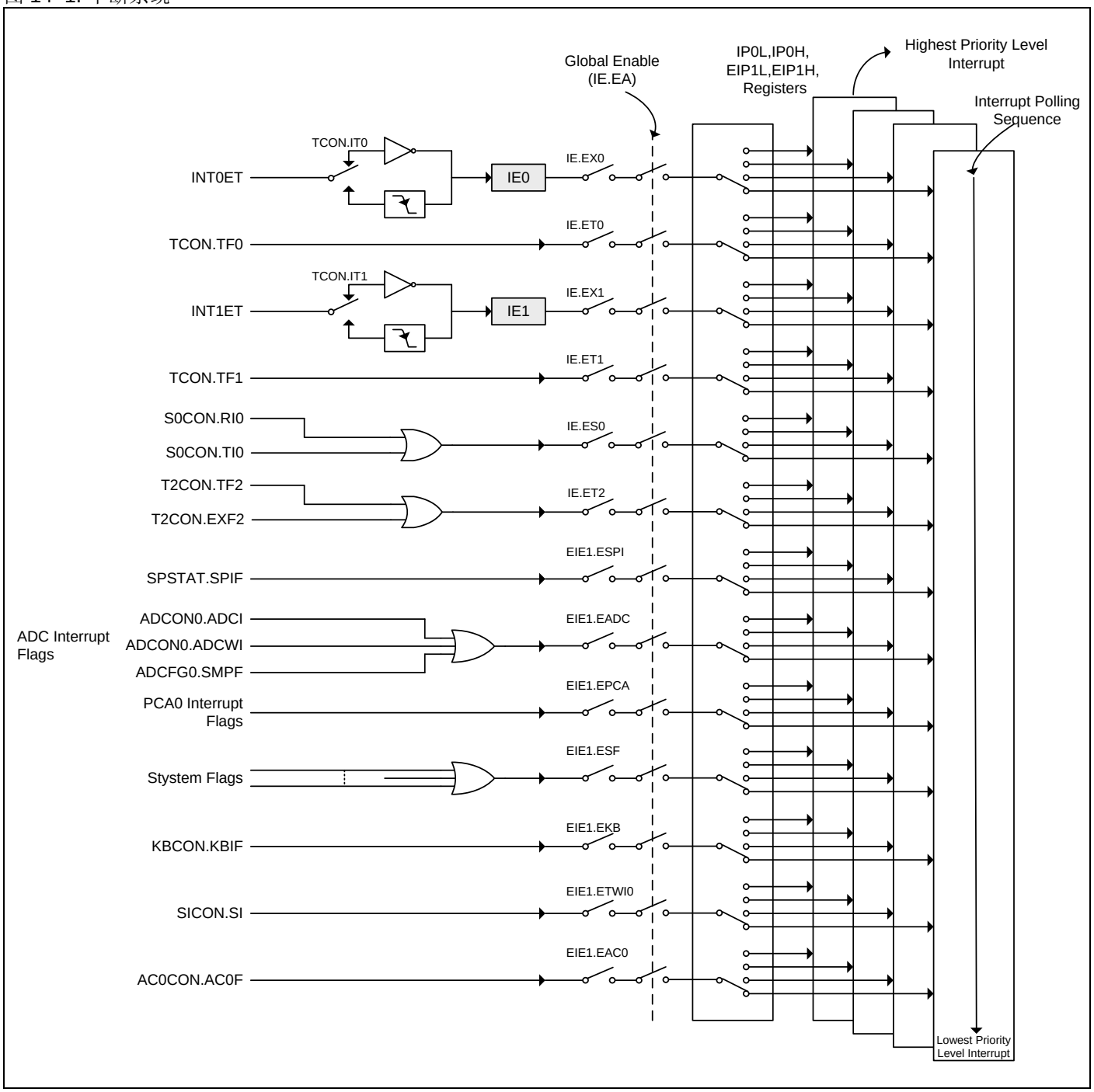
整个中断系统如图 14–1所示。每一个中断将在下面部分做简单的描述。

表 14–1. 中断源

序号	中断名称	使能位	请求位	优先级位	优先级	向量地址
#0	外部中断 0，nINT0	EX0	IE0	[PX0H, PX0L]	(最高)	0003H
#1	定时器 0	ET0	TF0	[PT0H, PT0L]	…	000Bh
#2	外部中断 1，nINT1	EX1	IE1	[PX1H, PX1L]	...	0013H
#3	定时器 1	ET1	TF1	[PT1H, PT1L]	**…**	001BH
#4	串口 0	ES0	RI0, TI0	[PS0H, PS0L]	**…**	0023H
#5	定时器 2	ET2	TF2, EXF2 (TF2L)	[PT2H, PT2L]	**…**	002Bh
#6	保留	--	--	--	--	0033H
#7	SPI	ESPI	SPIF	[PSPIH, PSPIL]	**…**	003BH
#8	ADC	EADC	ADCI, ADCWI, SMPF	[PADCH, PADCL]	**…**	0043H
#9	PCA0	EPCA	CF, CCFn (n=0~7)	[PPCAH, PPCAL]	**…**	004Bh
#10	系统标志	ESF	(Note 1)	[PSFH, PSFL]	**…**	0053H
#11	键盘中断	EKB	KBIF	[PKBH, PKBL]	**…**	005BH
#12	TWI0/I2C0	ETWI0	SI	[PTWI0H, PTWI0L]	**…**	0063H
#13	模拟比较器	EAC0	AC0F	[PAC0H, PAC0L]	**…**	006BH

注 1：系统标志中断标志位包括：PCON1 寄存器的 WDTF, BOF0, BOF1, RTCF, SPWF, CFAIL 和 EEPF；S0CON 寄存器的 TI0；AUXR2 寄存器的 STAF 和 STOF。

图 14–1. 中断系统


Global Enable
(IE.EA)
IP0L,IP0H,
EIP1L,EIP1H,
Registers
Highest Priority Level
Interrupt
Interrupt Polling
Sequence
TCON.IT0
INT0ET
IE0
IE.EX0
TCON.TF0
IE.ET0
TCON.IT1
INT1ET
IE1
IE.EX1
TCON.TF1
IE.ET1
S0CON.RI0
S0CON.TI0
IE.ES0
T2CON.TF2
T2CON.EXF2
IE.ET2
SPSTAT.SPIF
EIE1.ESPI
ADC Interrupt
Flags
ADCON0.ADCI
ADCON0.ADCWI
ADCFG0.SMPF
EIE1.EADC
PCA0 Interrupt
Flags
EIE1.EPCA
Stystem Flags
EIE1.ESF
KBCON.KBIF
EIE1.EKB
SICON.SI
EIE1.ETWI0
AC0CON.AC0F
EIE1.EAC0
Lowest Priority
Level Interrupt

14.2. 中断源

表 14–2. 中断源标志

序号	中断名称	请求位	位的位置
#0	外部中断 0， nINT0	IE0	TCON.1
#1	定时器 0	TF0	TCON.5
#2	外部中断 1， nINT1	IE1	TCON.3
#3	定时器 1	TF1	TCON.7
#4	串口 0	RI0, TI0	S0CON.0 S0CON.1
#5	定时器 2	TF2, EXF2, (TF2L)	T2CON.7 T2CON.6 T2CON.5
#6	保留	--	--
#7	SPI	SPIF	SPSTAT.7
#8	ADC	ADCI, ADCWI, SMPF	ADCON0.4 ADCON0.6 ADCFG0.2
#9	PCA0	CF, CCFn (n=0~3),	CCON.7 CCON.3~0
#10	系统标志	WDTF, BOF0, BOF1, SPWF, RTCF, STAF, STOF, CFAIL EEPF (TI0)	PCON1.0 PCON1.1 PCON1.2 PCON1.3 PCON1.4 AUXR2.7 AUXR2.6 ISPCR.4 ISPCR.0 S0CON.1
#11	键盘中断	KBIF	KBCON.0
#12	TWI0/I2C0	SI	SICON.3
#13	模拟比较器	AC0F	AC0CON.4

通过 TCON 寄存器的位 IT0 和 IT1 及 XICON 寄存器的位 IT2 可以设定外部中断 0(INT0)、外部中断 1(INT1)和外部中断 2(nINT2)为电平触发或边沿触发。实际产生这些中断的标志位是 TCON 的 IE0 和 IE1，XICON 的 IE2。当中断设置成边缘触发模式时，进入中断服务程序硬件将清除外部中断所产生的标志位，否则由外部请求源控制这个标志位。

TF0 和 TF1 产生定时器 0 和定时器 1 中断，多数情况下这两个标志位由它们对应的定时/计数寄存器翻转事件置位。定时器中断发生后，进入中断服务程序，硬件将清除这个标志位。

串口 0 中断由位 RI0 和位 TI0 的逻辑或产生。执行中断服务程序后不会被硬件清除须由软件清零，可以在中断服务程序中查询 RI0 和 TI0 判断是接收中断还是发送中断。

定时器/计数器 2 中断由两个标志位 TF2 或 EXF2 产生。如果定时器 2 在分立模式，TL2 的溢出将置位另一个中断标志 TF2L。跟串口一样，执行中断服务程序后这些标志位不会被硬件清除。

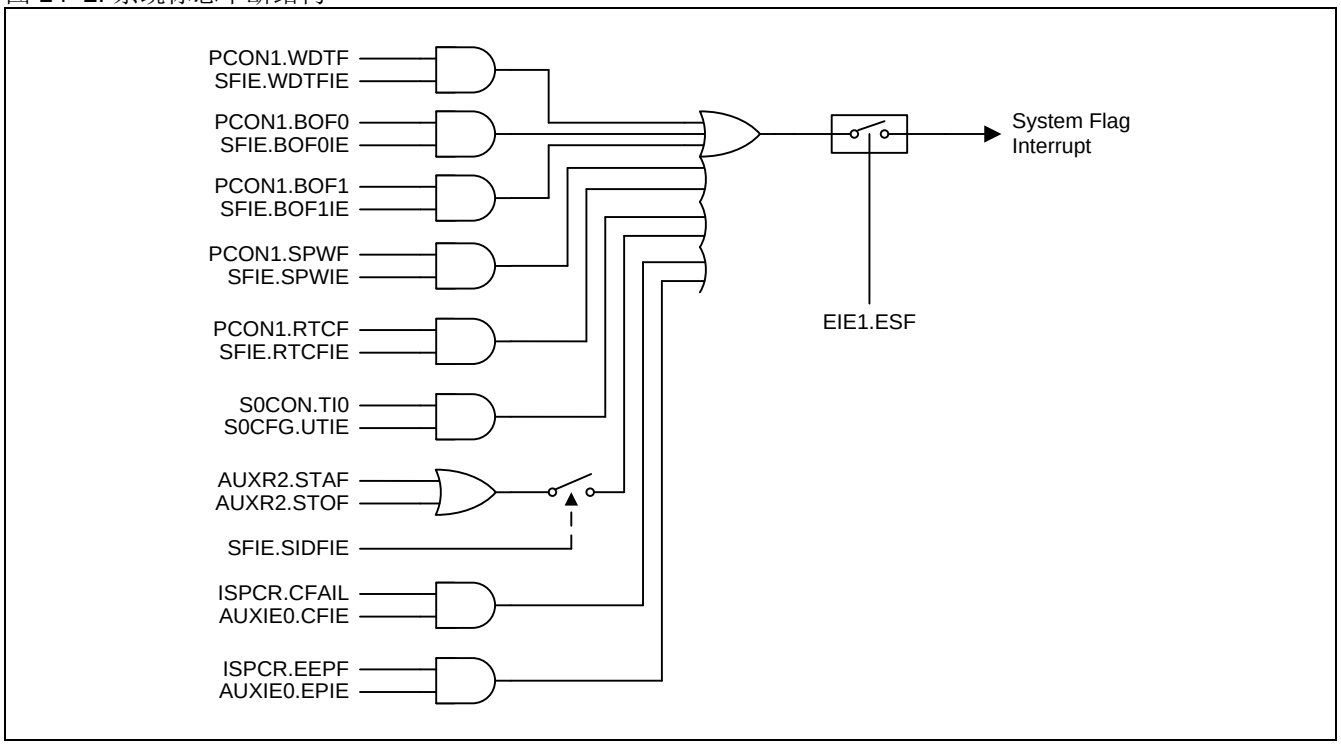
SPI 中断由寄存器 SPSTAT 里的 SPIF 位产生，SPI 引擎完成一个 SPI 传送后置该标志位。该标志位在执行中断服务程序后不会被硬件清除。

ADC 中断由寄存器 ADCON0 里的 ADCI 和 ADCWI 位，和 ADCDG0 里的 SMPF 产生。该标志位在执行中断服务程序后不会被硬件清除。

PCA0 中断由寄存器 CCON 里的 CF、CCF3、CCF2、CCF1 和 CCF0 位产生。这些标志位在执行中断服务程序后不会被硬件清除。中断服务程序应当轮询这些标志位去判断是哪一个请求服务，并且在软件里清除这些中断标志位。

系统标志中断由 RTCF、BOF1、BOF0、WDTF、SPWF、CFAIL、EEPF、TI0、STAF 和 STOF 位产生。STAF 和 STOF 存在寄存器 AUXR2 里，由串行接口监测置位这两个标志位。串行口 TI 标志可以通过置位 UTIE 选择与系统标志中断共享中断向量。复位标志存在于寄存器 PCON1 里。RTC 计数器溢出置位 RTCF。片内低电压监测器(BOD1 和 BOD0)监测到低电压时置位 BOF1 和 BOF0。看门狗溢出置位 WDTF。SPWF 会在 SP 监视器检测到堆栈检测警告中置起，***CFAIL & EEPF***。这些标志位在执行中断服务程序后不会被硬件清除。系统标志中断结构如图 14–2 所示。

图 14–2. 系统标志中断结构



键盘中断由 KBCON 寄存器的位 KBIF 来产生，KBIF 由键盘模块遇到键输入来置位。执行中断服务程序后不会被硬件清除。

TWI0/I2C0 中断由 SICON 寄存器的位 SI 来产生，SI 由 TWI0/I2C0 引擎检测到一个新的总线状态来置位。执行中断服务程序后不会被硬件清除。

AC0 中断由 AC0CON 寄存器的位 AC0F 来产生，AC0F 由 AC0OUT 改变的上升沿、下降沿或双沿来置位。执行中断服务程序后不会被硬件清除。

所有这些中断标志都能被软件置位或清零，跟硬件置位或清零的结果是一样的。也就是说，中断能通过软件来产生也可以软件来取消。

14.3. 中断使能

表 14–3. 中断使能

序号	中断名称	使能位	位的位置
#0	外部中断 0，nINT0	EX0	IE.0
#1	定时器 0	ET0	IE.1
#2	外部中断 1，nINT1	EX1	IE.2
#3	定时器 1	ET1	IE.3
#4	串口 0	ES0	IE.4
#5	定时器 2	ET2	IE.5
#6	保留	--	--
#7	SPI	ESPI	EIE1.0
#8	ADC	EADC	EIE1.1
#9	PCA0	EPCA	EIE1.2
#10	系统标志	ESF	EIE1.3
#11	键盘中断	EKB	EIE1.5
#12	TWI0/I2C0	ETWI0	EIE1.6
#13	模拟比较器	EAC0	EIE1.7

MG82F6B08 / 6B001/ 6B104 有 **12** 个中断源可用。每个中断源可以通过 IE、EIE1 寄存器的中断使能位置位或清零各自中断使能或禁止。IE 也提供一个全局中断使能位(EA)，此位清零可以立刻禁止所有中断。如果此位置位中断由相应的中断使能位各自使能或禁止。如果此位清零则所有中断被禁止。

14.4. 中断优先级

服务中断的优先级除了有 4 个级别比 80C51 多 2 个之外跟 80C51 一样。优先级位决定每个中断的优先级(见表 14–1)。IP0L、IP0H、EIP1L、EIP1H、EIP2L 和 EIP2H 跟 4 个级别优先级中断相关。位的值和优先级的关系如表 14–4 所示。

表 14–4. 中断优先级

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

每个中断源都有两个中断优先级相关位。一个位在 IPnH 寄存器另一个在 IPnL 寄存器。高优先级中断不会被低优先级中断打断。如果两个不同优先级的中断请求同时出现，较高优先级将被执行。如果相同优先级的中断请求同时出现，则按照内部优先级排序执行。同一优先级的内部优先级排序和中断向量地址如表 14–2 所示。

14.5. 中断处理

每一个系统时钟周期将采样每一个中断标志。在下一个系统时钟采样成功。如果其中一个标志在第一个周期置位，第二个周期(轮询周期)找到并且只要没有被下列条件阻止则中断系统产生一个硬件调用(LCALL)相应的中断服务程序。

阻止条件：

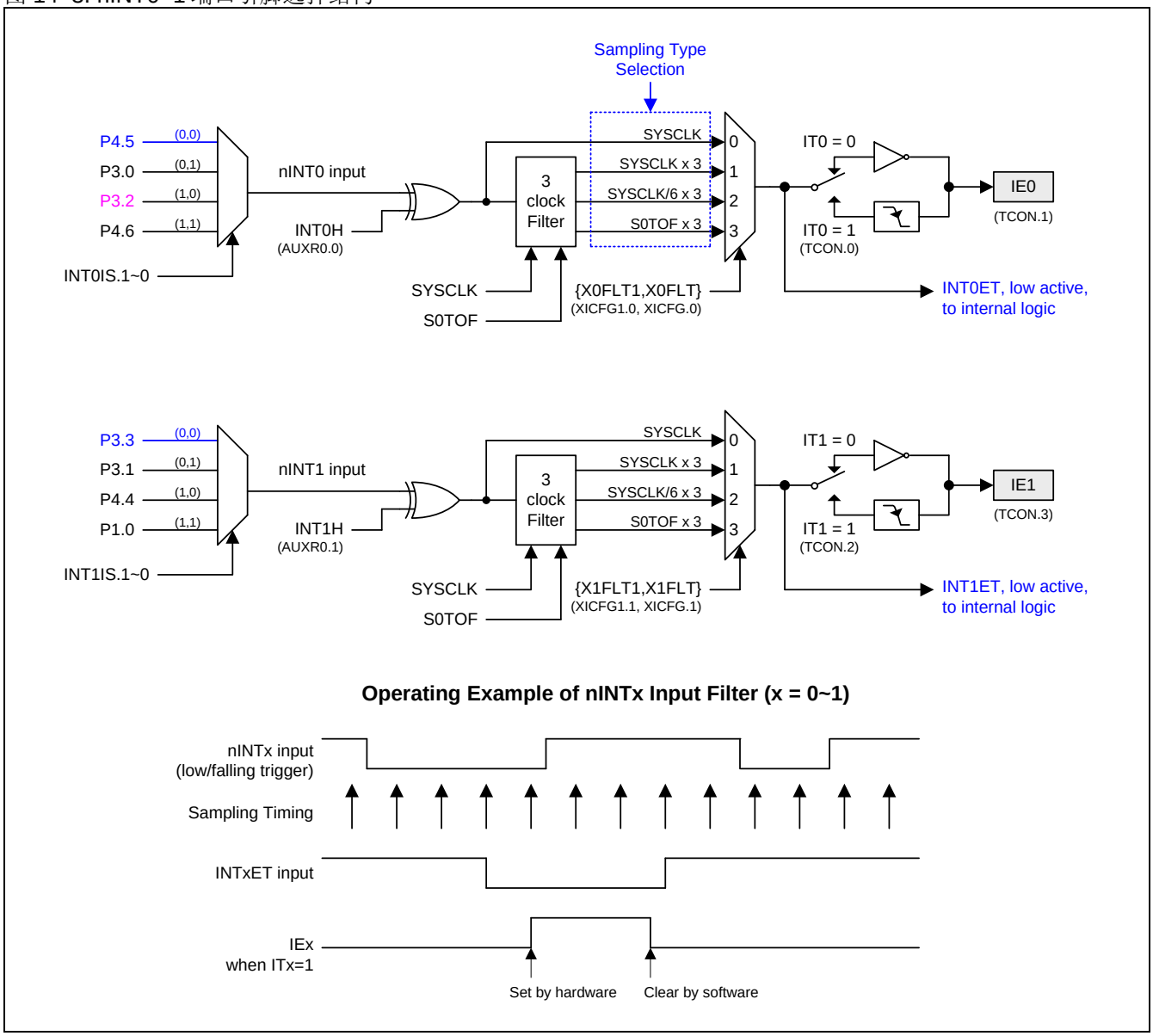
- 进行中已经有一个同级或更高级优先级的中断。
- 进行中当前周期(轮询周期)不是指令执行结束周期。
- 指令进行是 RETI 或 IE、IP0L、IP0H、EIE1、EIP1L、和 EIP1H 寄存器的写操作。

上述三个条件中的任意一个将阻止硬件中断调用(LCALL)去中断服务程序。条件 2 确保中断进入任意一个服务程序之前指令执行完毕。条件 3 确保如果在 RETI 执行或 IE 或 IP 的任何访问之后，进入中断服务程序之前至少一个或更多指令被执行。

14.6. nINTx 输入源选择和输入滤波器(x=0~1)

MG82F6B08 / 6B001/ 6B104 提供灵活的 nINT0 和 nINT1 输入源选择去共享端口引脚输入。这些引脚支持使用滤波器滤波，也可作为其他模块的输入触发器，比如定时器、ADC、PCA、I2C 等。

图 14–3. nINT0~1 端口引脚选择结构



14.7. 中断寄存器

TCON:定时器/计数器控制寄存器

SFR 页 = 0~F
SFR 地址 = 0x88 复位值 = 0000-0000

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

Bit 3：IE1，外部中断 1(nINT1)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(nINT1) 由边沿或电平触发(由 IT1 设置)硬件置位。

Bit 2：IT1，外部中断 1(nINT1)类型控制位。
0：软件清零选择低电平触发外部中断 1(nINT1)。如果 INT1H(AUXR0.1)置位，则高电平触发外部中断 1(nINT1)。
1：软件置位选择下降沿触发外部中断 1(nINT1)。如果 INT1H(AUXR0.1)置位，则上升沿触发外部中断 1(nINT1)。

Bit 1：IE0，外部中断 0(nINT0)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 (nINT0)由边沿或电平触发(由 IT0 设置)硬件置位。

Bit 0：IT0，外部中断 0(nINT0)类型控制位。
0：软件清零选择低电平触发外部中断 0(nINT0)。如果 INT0H(AUXR0.0)置位，则高电平触发外部中断 0(nINT0)。
1：软件置位选择下降沿触发外部中断 0(nINT0)。如果 INT0H(AUXR0.0)置位，则上升沿触发外部中断 0(nINT0)。

IE:中断使能寄存器

SFR 页 = 0~F
SFR 地址 = 0xA8 复位值 = 0000-0000

7	6	5	4	3	2	1	0
EA	0	ET2	ES0	ET1	EX1	ET0	EX0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7：EA，总中断使能位。
0：禁止所有中断。
1：使能所有中断。

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

Bit 4：ES，串口 0 中断(UART0)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(nINT1)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(nINT0)使能。
0：禁止外部中断 0。
1：使能外部中断 0。

AUXR0:辅助寄存器 0

SFR 页 = 0~F
SFR 地址 = 0xA1 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P45OC1	P45OC0	0	PBKF	0	0	INT1H	INT0H
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 4：PBKF，PWM 终止标志。此位由 PWM 终止源使能置位。如果此位置位，则使能的 PWM 通道 0~3 将被锁住并且输出引脚保持最初的 GPIO 状态。
0：没有 PWM 终止事件出现。仅由软件清零。
1：PWM 终止事件出现或软件触发一个 PWM 终止。

Bit 1：INT1H，INT1 高电平/上升沿触发使能。
0：保留 INT1 在选择的端口引脚上低电平或下降沿触发。
1：设置 INT1 在选择的端口引脚上高电平或上升沿触发。

Bit 0：INT0H，INT0 高电平/上升沿触发使能。
0：保留 INT0 在选择的端口引脚上低电平或下降沿触发。
1：设置 INT0 在选择的端口引脚上高电平或上升沿触发。

IP0L:中断优先级 0 低字节寄存器

SFR 页 = 0~F
SFR 地址 = 0xB8 复位值 = 0000-0000

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

Bit 5：PT2L，定时器 2 中断优先级低位。
Bit 4：PSL，串口 0(UART0)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 中断优先级低位。
Bit 1：PT0L，定时器 0 中断优先级低位。
Bit 0：PX0L，外部中断 0 中断优先级低位。

IP0H:中断优先级 0 高字节寄存器

SFR 页 = 0~F
SFR 地址 = 0xB7 复位值 = 0000-0000

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

Bit 5：PT2H，定时器 2 中断优先级高位。
Bit 4：PSH，串口 0(UART0)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 中断优先级高位。
Bit 1：PT0H，定时器 0 中断优先级高位。
Bit 0：PX0H，外部中断 0 中断优先级高位。

EIE1:扩展中断使能 1 寄存器

SFR 页 = 0~F

SFR 地址 = 0xAD 复位值 = 0000-0000

7	6	5	4	3	2	1	0
EAC0	ETWI0	EKB	0	ESF	EPCA	EADC	ESPI
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：EAC0，模拟比较器 0（AC0）中断使能。
0：禁止 AC0 中断。
1：使能 AC0 中断。

Bit 6：ETWI0，TWI0/ I2C0 中断使能。
0：禁止 TWI0/ I2C0 中断。
1：使能 TWI0/ I2C0 中断。

Bit 5：EKBI，键盘中断使能。
0：当键盘控制模块的 KBCON.KBIF 置位时禁止中断。
1：当键盘控制模块的 KBCON.KBIF 置位时使能中断。

Bit 4：保留位。当写 EIE1 寄存器时软件，这些位必须写 “0”。

Bit 3：ESF，系统标志中断使能。
0：当 PCON1 的位{ RTCF、BOF1、BOF0、WDTF }置位，AUXR2 的位{STAF、STOF}置位，或 TI0 置位与 UTIE 一起置位时禁止中断。
1：当 PCON1 的位{RTCF、BOF1、BOF0、WDTF }置位，AUXR2 的位{STAF、STOF}置位，或 TI0 置位与 UTIE 一起相关联的系统标志置位时使能中断。

Bit 2：EPCA，PCA0 中断使能。
0：禁止 PCA0 中断。
1：使能 PCA0 中断。

Bit 1：EADC，ADC 中断使能。
0：当 ADC 模块的 ADCON0.ADCI 置位禁止中断。
1：当 ADC 模块的 ADCON0.ADCI 置位使能中断。

Bit 0：ESPI，SPI 中断使能。
0：当 SPI 模块的 SPSTAT.SPIF 置位禁止中断。
1：当 SPI 模块的 SPSTAT.SPIF 置位使能中断。

EIP1L:扩展中断优先级 1 低字节寄存器

SFR 页 = 0~F

SFR 地址 = 0xAE 复位值 = 0000-0000

7	6	5	4	3	2	1	0
PAC0L	PTWI0L	PKBL	0	PSFL	PPCAL	PADCL	PSPIL
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：PAC0L，AC0 中断优先级低位。
Bit 6：PTWI0L，TWI0 中断优先级低位。
Bit 5：PKBL，键盘中断优先级低位。
Bit 4：保留位。当写 EIP1L 寄存器时软件，这些位必须写 “0”。
Bit 3：PSFL，系统标志中断优先级低位。
Bit 2：PPCAL，PCA0 中断优先级低位。
Bit 1：PADCL，ADC 中断优先级低位。
Bit 0：PSPIL，SPI 中断优先级低位。

EIP1H:扩展中断优先级 1 高字节寄存器

SFR 页 = 0~F

SFR 地址 = 0xAF 复位值 = 0000-0000

7	6	5	4	3	2	1	0
PAC0H	PTWI0H	PKBH	0	PSFH	PPCAH	PADCH	PSPIH
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：PAC0H，AC0 中断优先级高位。
Bit 6：PTWI0H，TWI0 中断优先级高位。
Bit 5：PKBH，键盘中断优先级高位。
Bit 4：保留位。当写 EIP1H 寄存器时软件，这些位必须写 “0”。
Bit 3：PSFH，系统标志中断优先级高位。
Bit 2：PPCAH，PCA0 中断优先级高位。
Bit 1：PADCH，ADC 中断优先级高位。
Bit 0：PSPIH，SPI 中断优先级高位。

XICFG:外部中断配置寄存器

SFR 页 = **仅 0 页**

SFR 地址 = **0xC1** 复位值 = 0000-0000

7	6	5	4	3	2	1	0
INT1IS.1	INT1IS.0	INT0IS.1	INT0IS.0	0	0	X1FLT	X0FLT
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6：INT1IS.1~0，nINT1 输入引脚选择位如下表。

INT1IS.1~0	nINT1 输入引脚选择
0 0	P3.3
0 1	P3.1
1 0	P4.4
1 1	P1.0

Bit 5~4：INT0IS.1~0，nINT0 输入引脚选择位如下表。

INT0IS.1~0	Selected Port Pin of nINT0
0 0	P4.5
0 1	P3.0
1 0	P3.2
1 1	P4.6

Bit 3~2：保留位。当写 XICFG 寄存器时软件，这些位必须写 “0”。

Bit 1：X1FLT，nINT1 滤波模式控制。和 X1FLT1 (XICFG1.1)一起选择 nINT1 的输入滤波模式。

X1FLT1, X1FLT	nINT1 输入滤波模式
0 0	禁止
0 1	SYSCLK x 3
1 0	SYSCLK/6 x 3
1 1	S0TOF x 3

Bit 0：X0FLT，nINT0 滤波模式控制。和 X0FLT1 (XICFG1.0)一起选择 nINT0 的输入滤波模式。

X0FLT1, X0FLT	nINT0 输入滤波模式
0 0	禁止
0 1	SYSCLK x 3
1 0	SYSCLK/6 x 3
1 1	S0TOF x 3

XICFG1:外部中断配置寄存器

SFR 页 = **仅 1 页**

SFR 地址 = **0xC1** 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	0	0	0	0	0	X1FLT1	X0FLT1
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~2：保留位。当写 XICFG1 寄存器时软件，这些位必须写 “0”。

Bit 1：X1FLT1，nINT1 滤波模式控制。和 X1FLT (XICFG.1)一起选择 nINT1 的输入滤波模式。参考寄存器 XICFG 有关 nINT1 输入滤波模式定义的描述。

Bit 0：X0FLT1，nINT0 滤波模式控制。和 X0FLT (XICFG.0)一起选择 nINT0 的输入滤波模式。参考寄存器 XICFG 有关 nINT0 输入滤波模式定义的描述。

SFIE:系统标志中断使能寄存器

SFR 页 = 0~F

SFR 地址 = 0x8E POR = 0110-0000

7	6	5	4	3	2	1	0
SIDFIE	1	1	RTCFIE	SPWIE	BOF1IE	BOF0IE	WDTFIE
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：SIDFIE，串行接口(STWI/SI2C)侦测标志中断使能。
0：禁止 SIDF(STAF 或 STOF) 中断。
1：使能 SIDF(STAF 或 STOF) 中断共享系统标志中断。

Bit 4：RTCFIE，使能 RTCF (PCON1.4) 中断。
0：禁止 RTCF 中断。
1：使能 RTCF 中断。

Bit 3：SPWIE，使能 SPWF (PCON1.3) 中断。
0：禁止 SPWF 中断。
1：使能 SPWF 中断。

Bit 2：BOF1IE，使能 BOF1 (PCON1.2)中断。
0：禁止 BOF1 中断。
1：使能 BOF1 中断。

Bit 1：BOF0IE，使能 BOF0 (PCON1.1)中断。
0：禁止 BOF0 中断。
1：使能 BOF0 中断。

Bit 0：WDTFIE，使能 WDTF (PCON1.0)中断。
0：禁止 WDTF 中断。
1：使能 WDTF 中断。

PCON1:电源控制寄存器 1

SFR 页 = 0~F & P

SFR 地址 = 0x97 POR = 0000-0000

7	6	5	4	3	2	1	0
SWRF	EXRF	0	RTCF	SPWF	BOF1	BOF0	WDTF
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：SWRF，软件复位标志。
0：必须由软件写**“1”**才能清零。
1：如果一个软件复位发生，则此位硬件置位。

Bit 6：EXRF，外部复位标志。
0：必须由软件写**“1”**才能清零。
1：如果一个外部复位发生，则此位硬件置位。

Bit 4：RTCF，RTC 溢出标志。
0：必须由软件写**“1”** 才能清零。软件写“0”无操作。
1：当 RTCCT 溢出时，硬件置位。写**“1”**将清零。

Bit 3：SPWF，SP 警告标志。
0：必须由软件写**“1”** 才能清零。软件写“0”无操作。
1：该位只会在 SP ≥ SPHB 时被硬件置起，当 SP < SPHB 时写**“1”**将清零。

Bit 2：BOF1，低电压监测标志 1。
0：必须由软件写**“1”**才能清零。
1：如果低电压监测器 1 监测到工作电压匹配监测电平(4.2V/3.6V/2.4V/2.7V)，则此位硬件置位。

Bit 1：BOF0，低电压监测标志 0。
0：必须由软件写“**1**”才能清零。
1：如果低电压监测器 0 监测到工作电压匹配监测电平(1.7V)，则此位硬件置位。

Bit 0：WDTF，WDT 溢出标志。
0：必须由软件写“**1**”才能清零。
1：如果一个 WDT 溢出发生，则此位硬件置位。

AUXR2:辅助寄存器 2

SFR 页 = **仅 0 页**
SFR 地址 = 0xA3 复位值 = 0000-0000

7	6	5	4	3	2	1	0
STAF	STOF	0	C0PLK	T1X12	T0X12	T1CKOE	T0CKOE
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：STAF，STWI(SID)的起始标志侦测。
0：软件写“0”清零。在 MCU 复位过程中，STAF 会被置起，因此需要在软件初始化时清除 STAF。
1：硬件置位，表示在 STWI 总线上发生了一个**起始**动作。

Bit 6：STOF，STWI(SID)的停止标志侦测。
0：软件写“0”清零。
1：硬件置位，表示在 STWI 总线上发生了一个**停止**动作。

15. 定时器/计数器

MG82F6B08 / 6B001/ 6B104 有 **3** 个 16 位定时器/计数器：定时器 0，定时器 1，定时器 2。所有这些操作既可配置为定时器或事件记数器。

定时器功能，定时器预分频是每 12 个时钟周期加 1。换句话说，定时器是标准 C51 机器周期计数一次。AUXR2.T0X12、AUXR2.T1X12 和 T2MOD.T2X12 可以设置定时器 0/1/2 每个时钟周期计数一次。这样就是标准 C51 定时器 12 倍的速度。结合 T0C/T、T0XL 和 T0X12 定时器 0 时钟输入可选择其它的预分频。

计数器功能，负跳变时寄存器加 1，根据相应的外部输入引脚 T0，T1 或 T2。在这些功能中，每个定时器时钟周期对外部输入信号进行采样。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，新计数值在这一时钟周期后的下一周期结束时出现在寄存器中。

15.1. 定时器 0 和定时器 1

15.1.1. 定时器 0/1 模式 0

在模式 0，定时器寄存器配置为一个 PWM 产生器。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。定时器 0 这些控制位{T0XL, T0X12, T0C/T}设置时钟源来计数。并且也用 TR0 和{T0G1, T0GATE}选择门控源来阻止触发信号停止计数。定时器 1 这些控制位{T1X12, T1C/T}设置时钟源来计数。并且也用 TR1 和{T1G1, T1GATE}选择门控源来阻止触发信号停止计数。定时器 0 和 1 的模式 0 操作是一样的。定时器 0/1 的 PWM 功能结构图见图 15–1 和图 15–2。

图 15–1. 定时器 0 模式 0 结构

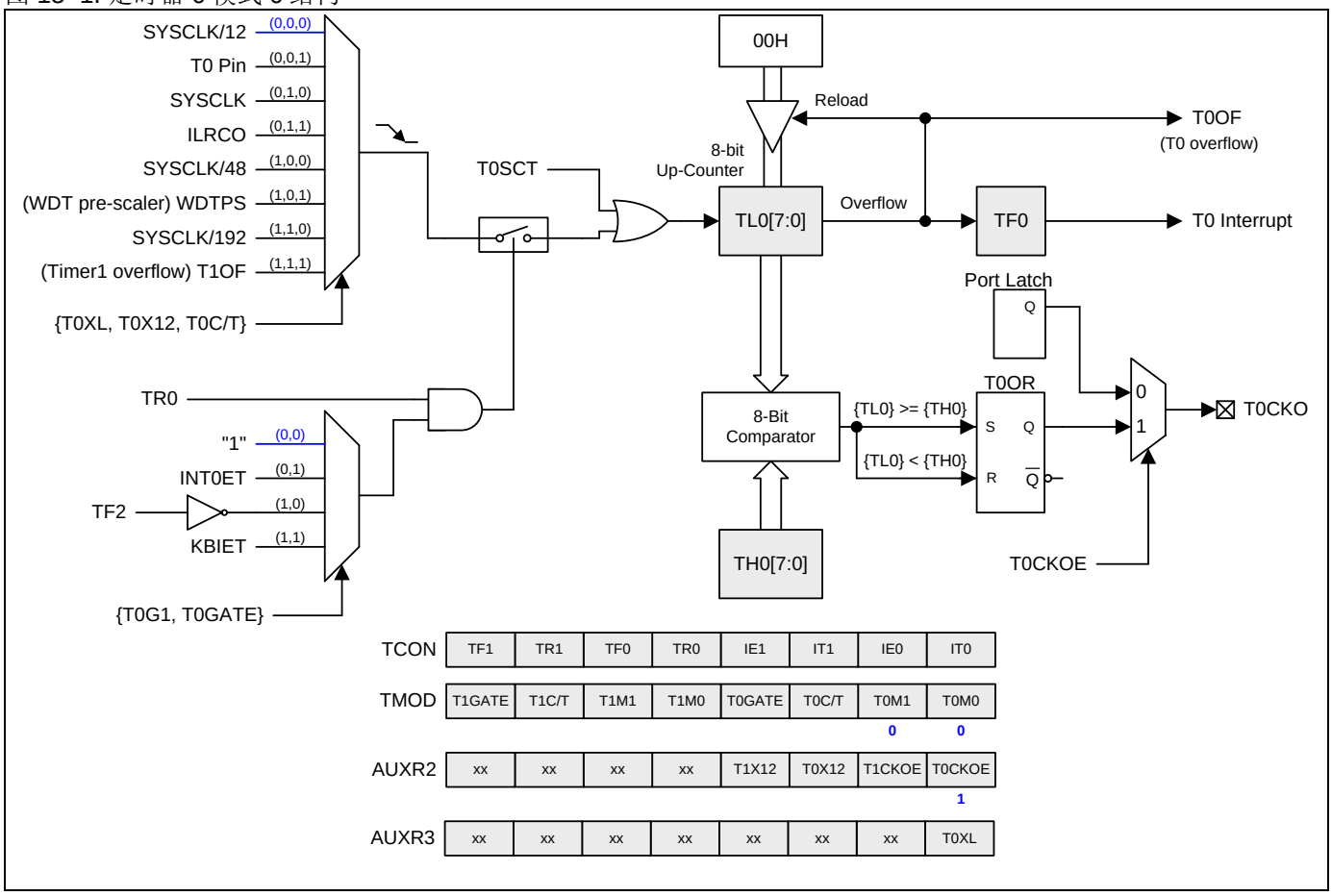
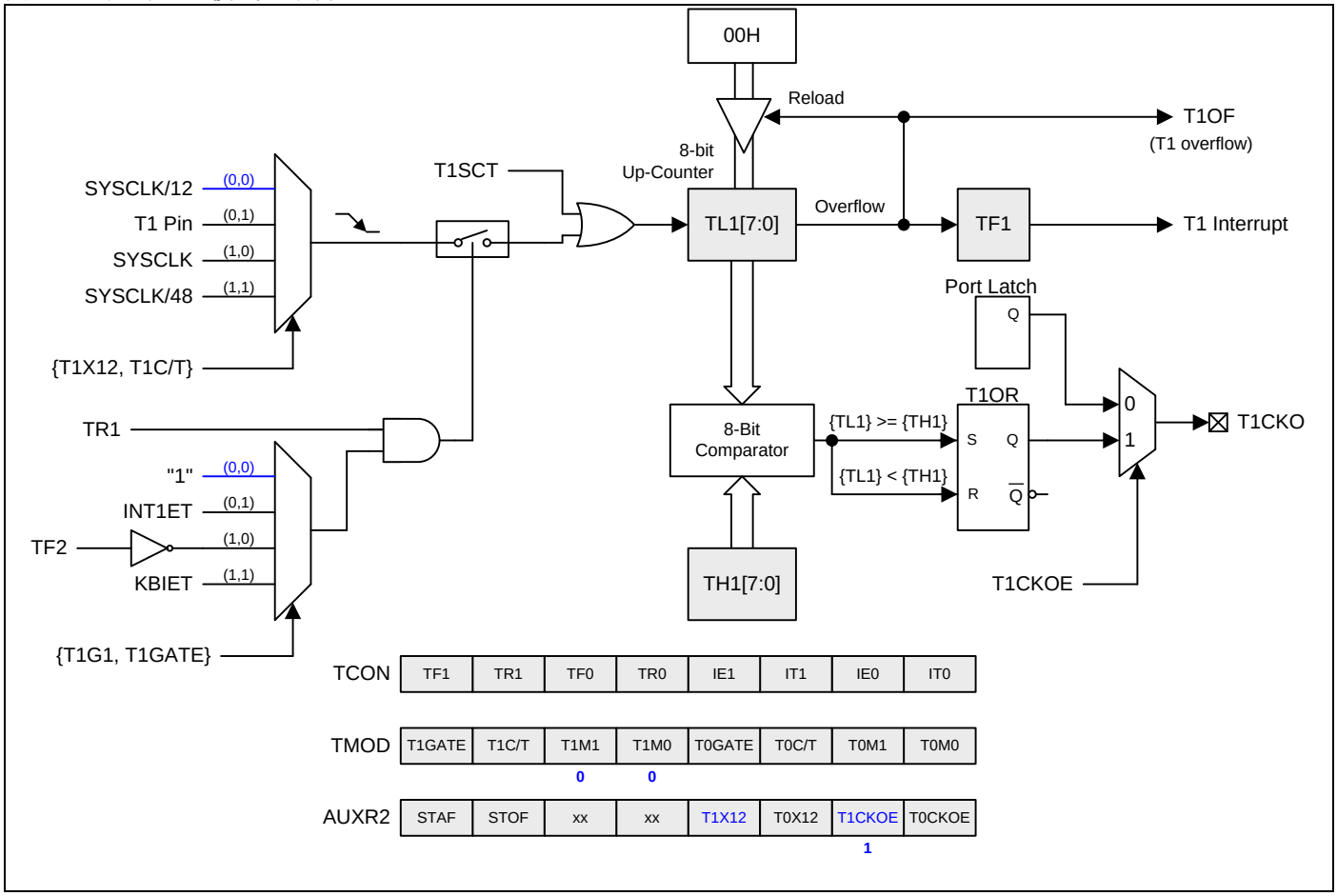


图 15–2. 定时器 1 模式 0 结构



15.1.2. 定时器 0/1 模式 1

在模式 1 定时器 0/1 配置成一个 16 位定时器或计数器。TxGATE、INTxET 和 TRx 的功能和模式 0 一样。定时器 0/1 模式 1 的结构图见图 15–3 和图 15–4。

图 15–3. 定时器 0 模式 1 结构

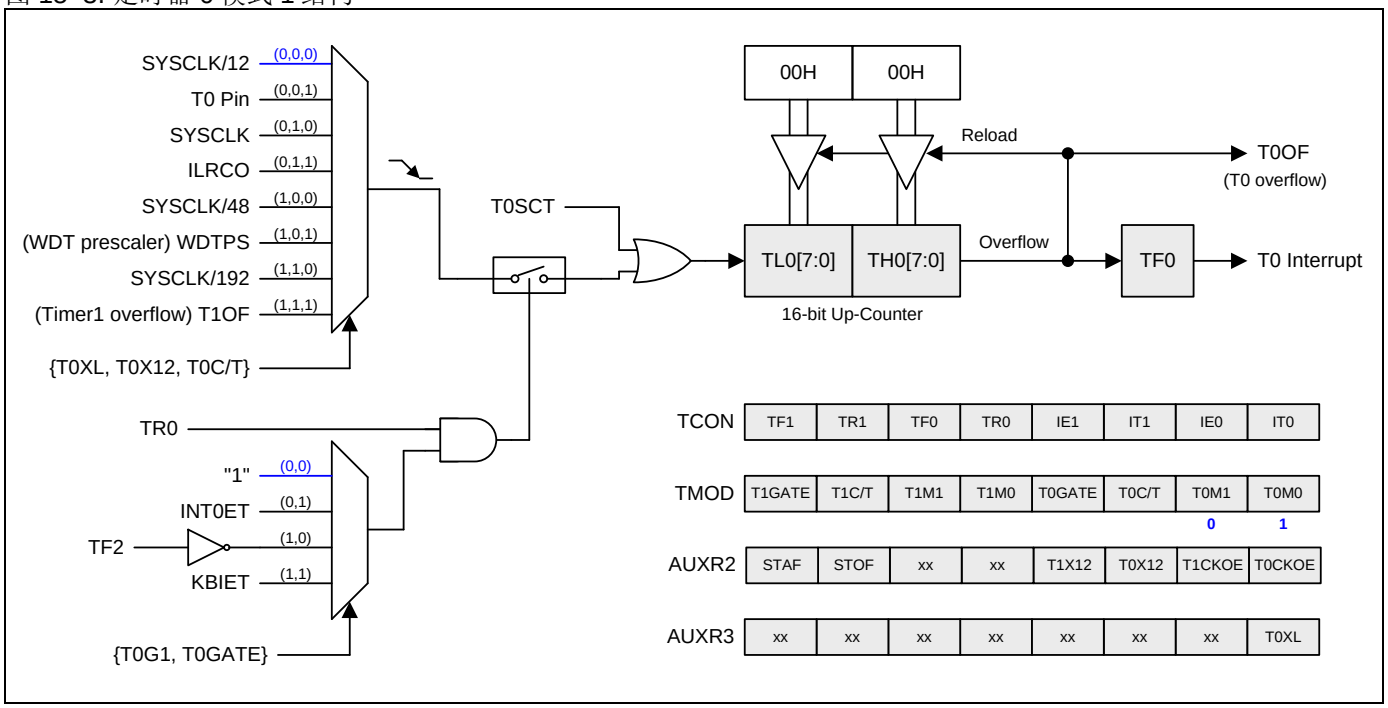
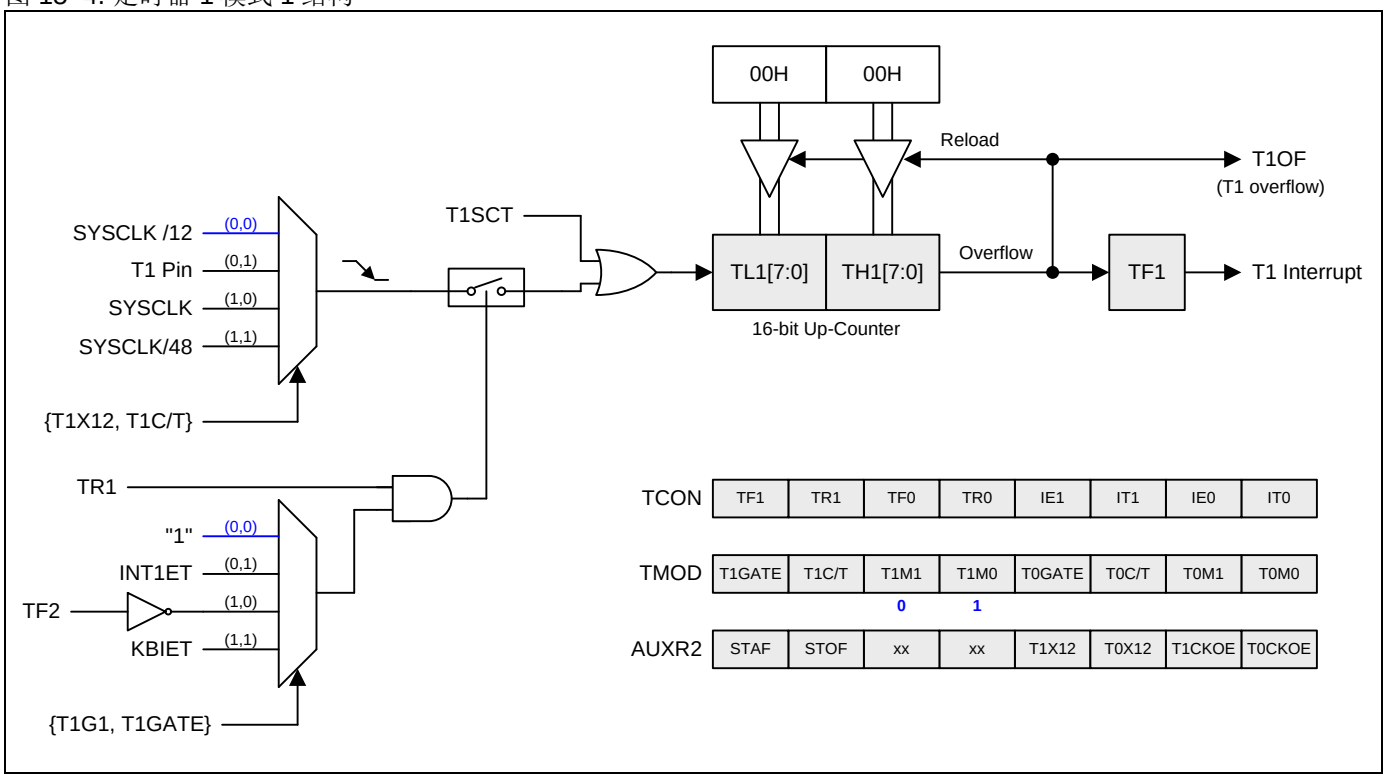


图 15–4. 定时器 1 模式 1 结构



15.1.3. 定时器 0/1 模式 2

模式 2 配置定时器寄存器为一个自动加载的 8 位计数器(TLx)。TLx 溢出不仅置位 TFx，而且也将 THx 的内容加载到 TLx，THx 内容由软件预置，加载不会改变 THx 的值。定时器 0 和 1 的模式 2 操作是一样的。定时器 0/1 模式 2 的结构图见图 15–5 和图 15–6。

图 15–5. 定时器 0 模式 2 结构

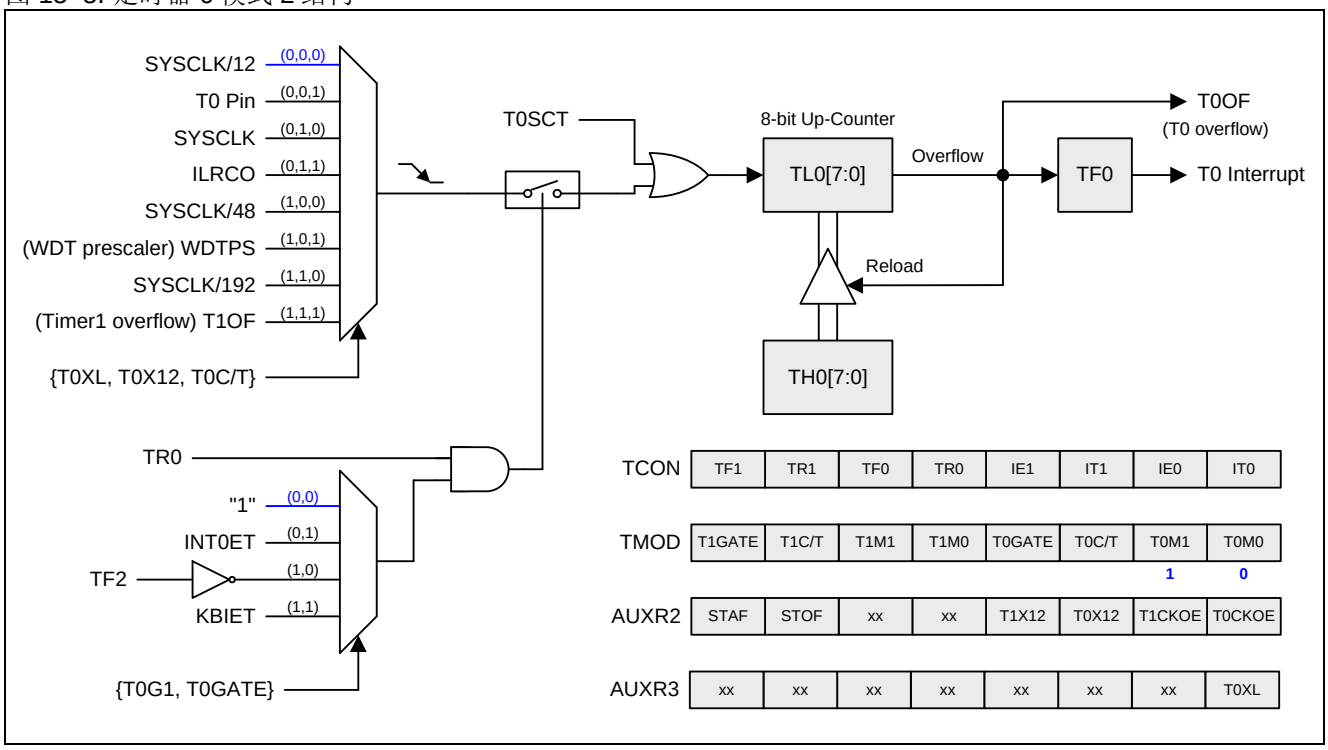
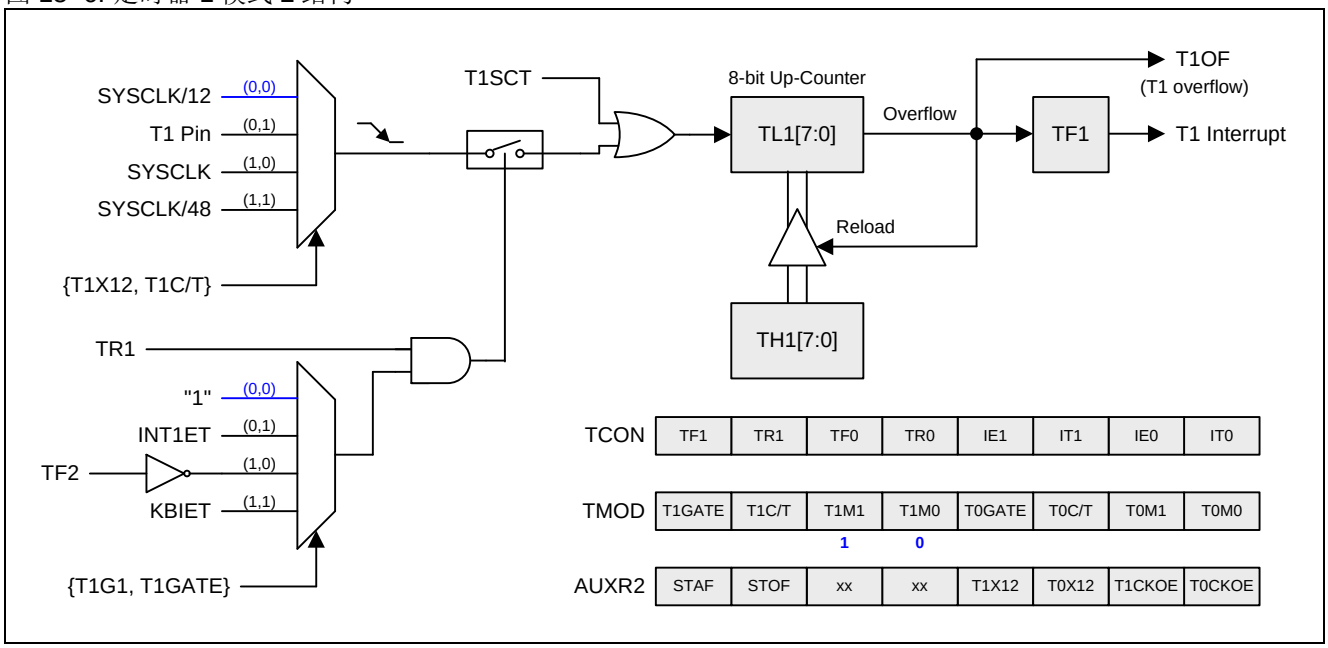


图 15–6. 定时器 1 模式 2 结构



15.1.4. 定时器 0/1 模式 3

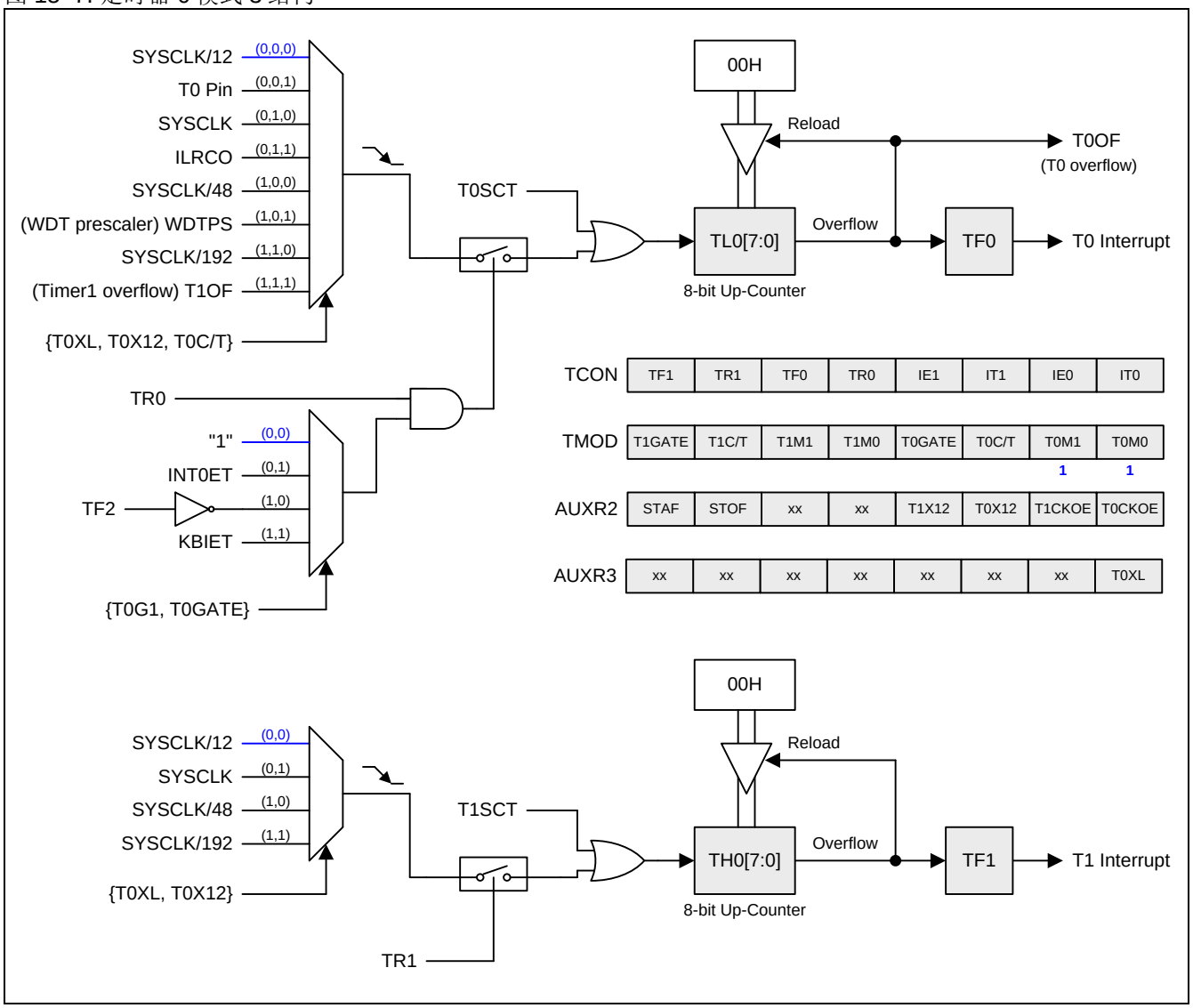
当定时器 0 和定时器 1 处于模式 3 时，定时器 1 的一些寄存器将与定时器 0 一起使用。同時定时器 1 在模式 3 下被挂起，即使设置 TR1=1。

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：T0XL、T0X12、 T0C /T、T0GATE、TR0 和 TF0。TH0 锁定为定时器功能(不能作为外部事件计数器)，定时器 1 的 TR1 将被用于控制 TH0。TH0 溢出将置位 TF1 并触发定时器 1 中断。

为了保证 TL1 不发生故障，请在定时器 0 设置为模式 3 之前将定时器 1 设置为模式 3。

定时器 0 模式 3 的结构图见图 15–7。

图 15–7. 定时器 0 模式 3 结构



15.1.5. 定时器 0/1 可编程时钟输出

定时器 0 和 1 有一个时钟输出模式(当 TxCKOE=1)。此模式下，定时器 0 或 1 操作在 8 位自动重载占空比为 1:1 的可编程时钟发生器。产生的时钟在 P3.4 (T0CKO)和 P3.5 (T1CKO)独立输出。8 位定时器(TL0)每个输入时钟加一，在定时器 0 模块。8 位定时器(TL1)每个输入时钟加一，在定时器 1 模块。定时器从载入值到溢出重复计数。一旦溢出，(TH0,TH1)的值被载入到(TL0, TL1)同时计数。图 15–8 和图 15–9 给出了定时器 0/1 时钟输出频率公式。图 15–10 和图 15–11 给出了定时器 0/1 时钟输出结构。

图 15–11 展示了定时器 1 的时钟输出结构。

图 15–8. 定时器 0 时钟输出公式

$$\text{T0 Clock-out Frequency} = \frac{\text{T0 Clock Frequency}}{2 \times (256 - \text{TH0})}$$

图 15–9. 定时器 1 时钟输出公式

$$\text{T1 Clock-out Frequency} = \frac{\text{T1 Clock Frequency}}{2 \times (256 - \text{TH1})}$$

注意：
(1) 定时器 0/1 溢出标志 TF0/1，在定时器 0/1 溢出时置位。
(2) 当 SYSCLK=12MHz 及选择 SYSCLK/12 为定时器 0/1 的时钟源，定时器 0/1 可编程输出频率范围从 1.95KHz 到 500KHz。
(3) 当 SYSCLK=12MHz 及选择 SYSCLK 为定时器 0/1 的时钟源，定时器 0/1 可编程输出频率范围从 23.44KHz 到 6MHz。

图 15–10. 定时器 0 时钟输出模式

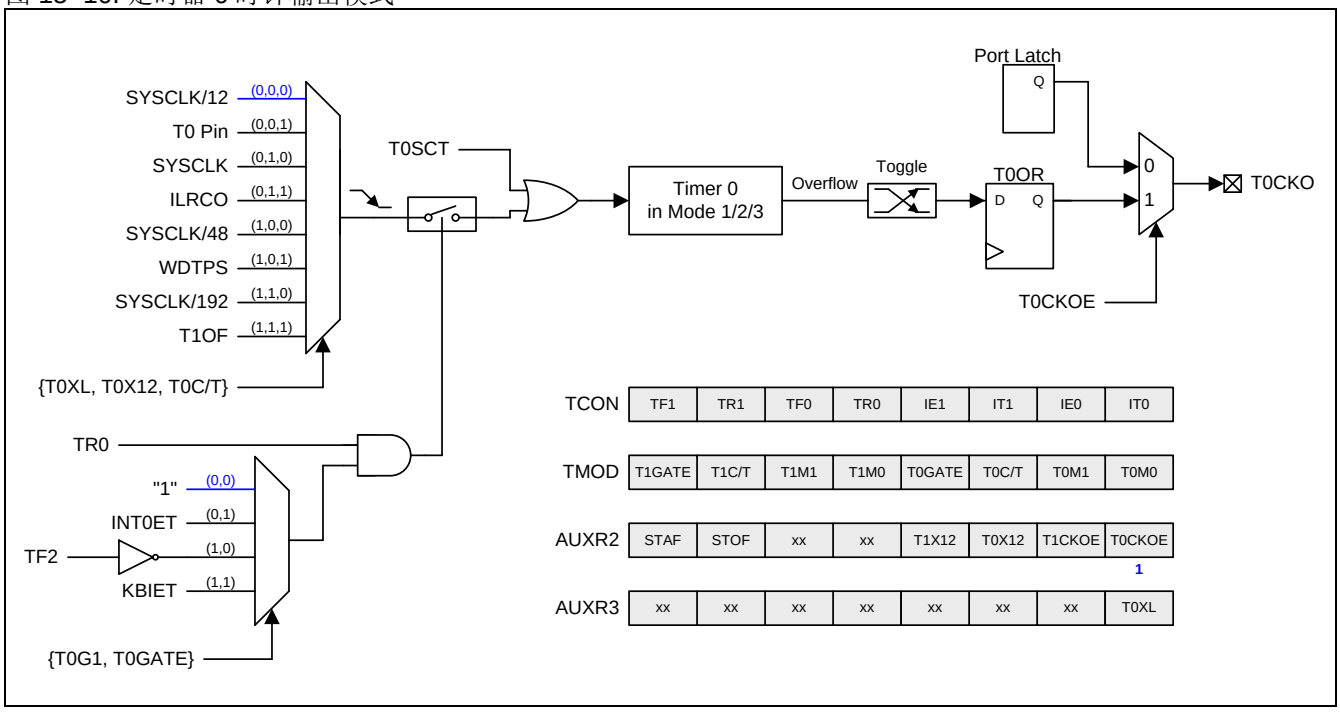
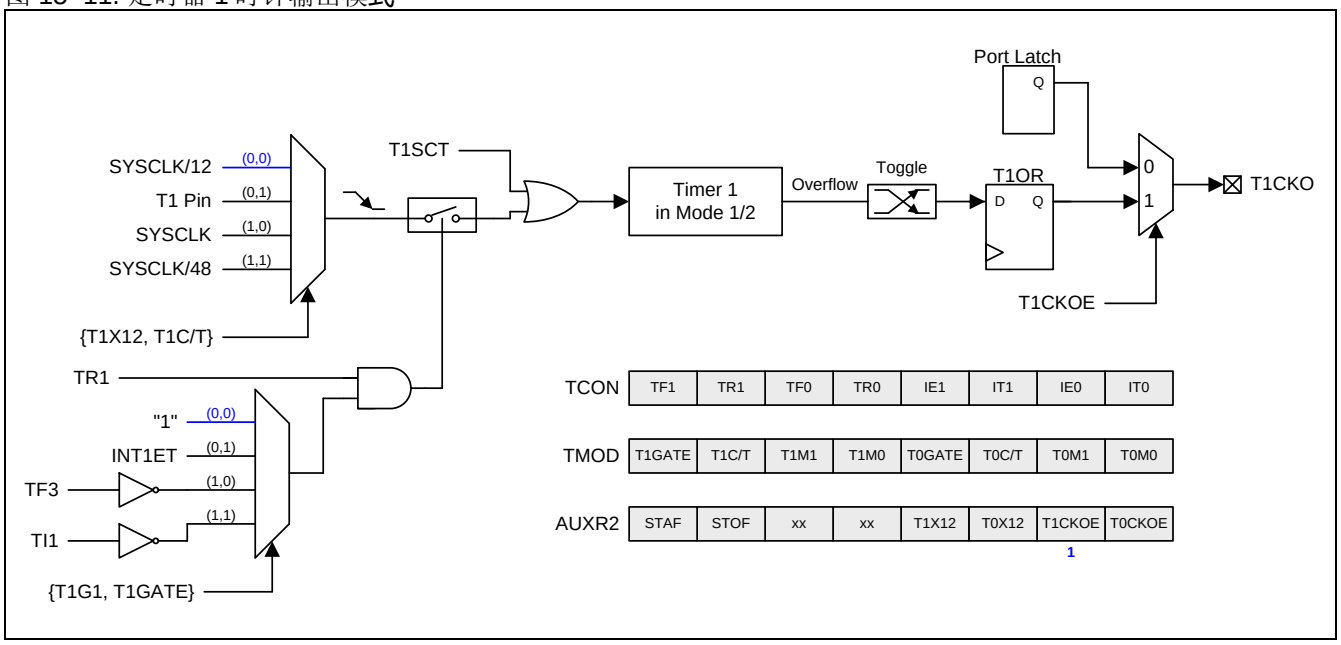


图 15–11. 定时器 1 时钟输出模式



定时器 0/1 时钟输出模式如何编程

- 选择定时器 0/1 的时钟源。
- 从公式计算出 8 位自动加载值并输入到 TH0/TH1 寄存器。
- 在 TL0/TL1 寄存器输入一个跟自动加载值相同 8 位初始值。
- AUXR2 寄存器的 T0CKOE/T1CKOE 置位。
- 通过设置 TCON 寄存器的 TR0/TR1 位启动定时器 0/1。

时钟输出模式，定时器 0/1 溢出不会中断。这跟定时器 1 被用作波特率发生器相似。定时器 1 即可用作波特率发生器也可同时用作时钟发生器。注意，波特率和时钟输出频率都是相同的定时器 1 溢出率。因此在这类应用中软件通常禁止定时器 0/1 中断。

15.1.6. 定时器 0/1 寄存器

TCON:定时器/计数器控制寄存器

SFR 页 = 0~F
SFR 地址 = 0x88 复位值 = 0000-0000

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

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

Bit 6：TR1，定时器 1 运行控制位。
0：关闭定时器/计数器 1。
1：开启定时器/计数器 1。

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

Bit 4：TR0，定时器 0 运行控制位。
0：关闭定时器/计数器 0。
1：开启定时器/计数器 0。

TMOD:定时器/计数器模式控制寄存器

SFR 页 = 0~F
SFR 地址 = 0x89 复位值 = 0000-0000

7	6	5	4	3	2	1	0
T1GATE	T1C/T	T1M1	T1M0	T0GATE	T0C/T	T0M1	T0M0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
|←---------- Timer1 ----------→				|←---------- Timer0 ----------→			

Bit 7：T1Gate，定时器 1 门控制位。

T1G1, T1GATE	T1 门控源
0 0	禁止
0 1	INT1 激活
1 0	TF2 激活
1 1	KBI 激活

Bit 6：T1C/T，定时器 1 时钟源选择位。控制着 4 种时钟源的定时器 1 作为定时器或计数器。详情参见 AUXR2.T1X12 的描述。

Bit 5~4：操作模式选择。

T1M1	T1M0	定时器 1 操作模式
0	0	定时器 1 的 8 位 PWM 产生器
0	1	定时器 1 工作在 16 位定时器/计数器模式
1	0	定时器 1 工作在 8 位自动装载定时器/计数器模式
1	1	(定时器 1) 定时器/计数器停止

Bit 3：T0Gate，定时器 0 门控制位。

T0G1, T0GATE	T0 门控源
0 0	禁止
0 1	INT0 激活
1 0	TF2 激活
1 1	KBI 激活

.
Bit 2：T0C/T，定时器 0 时钟源选择位。控制着 8 种时钟源的定时器 0 作为定时器或计数器。详情参见 AUXR2.T0X12 的描述。

Bit 1~0：操作模式选择。

T0M1	T0M0	定时器 0 操作模式
0	0	定时器 0 的 8 位 PWM 产生器
0	1	定时器 0 工作在 16 位定时器/计数器模式
1	0	定时器 0 工作在 8 位自动装载定时器/计数器模式
1	1	TL0 是 8 位定时器/计数器，TH0 锁定 8 位定时器

TL0:定时器 0 低字节寄存器

SFR 页 = 0~F

SFR 地址 = 0x8A 复位值 = 0000-0000

7	6	5	4	3	2	1	0
TL0.7	TL0.6	TL0.5	TL0.4	TL0.3	TL0.2	TL0.1	TL0.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TH0:定时器 0 高字节寄存器

SFR 页 = 0~F

SFR 地址 = 0x8C 复位值 = 0000-0000

7	6	5	4	3	2	1	0
TH0.7	TH0.6	TH0.5	TH0.4	TH0.3	TH0.2	TH0.1	TH0.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TL1:定时器 1 低字节寄存器

SFR 页 = 0~F

SFR 地址 = 0x8B 复位值 = 0000-0000

7	6	5	4	3	2	1	0
TL1.7	TL1.6	TL1.5	TL1.4	TL1.3	TL1.2	TL1.1	TL1.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TH1:定时器 1 高字节寄存器

SFR 页 = 0~F

SFR 地址 = 0x8D 复位值 = 0000-0000

7	6	5	4	3	2	1	0
TH1.7	TH1.6	TH1.5	TH1.4	TH1.3	TH1.2	TH1.1	TH1.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

AUXR2:辅助寄存器 2

SFR 页 = 0~F

SFR 地址 = 0xA3 复位值 = 0000-0000

7	6	5	4	3	2	1	0
STAF	STOF	0	C0PLK	T1X12	T0X12	T1CKOE	T0CKOE
R/W	R/W	W	W	R/W	R/W	R/W	R/W

Bit 3：T1X12，和 T1C/T 一起控制定时器 1 时钟源。

T1X12, T1C/T	定时器 1 时钟选择
0 0	SYSCLK/12
0 1	T1 引脚输入
1 0	SYSCLK
1 1	SYSCLK/48

Bit 2：T0X12，T0XL 和 T0C/T 一起控制定时器 0 时钟源选择。

T0XL, T0X12, T0C/T	定时器 0 时钟选择
0 0 0	SYSCLK/12
0 0 1	T0 引脚输入
0 1 0	SYSCLK
0 1 1	ILRCO
1 0 0	SYSCLK/48
1 0 1	WDTPS
1 1 0	SYSCLK/192
1 1 1	T1OF

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 时钟输出在 T1CKO 端口引脚。

Bit 0：T0CKOE，定时器 0 时钟输出使能。
0：禁止定时器 0 时钟输出。
1：使能定时器 0 时钟输出在 T0CKO 端口引脚。

AUXR3:辅助寄存器 3

SFR 页 = **仅 0 页**
SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	T0PS0	BPOC1	BPOC0	S0PS0	TWIPS1	TWIPS0	T0XL
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：保留位。当写 AUXR3 寄存器时软件，这些位必须写 "**0**"。

Bit 6：T0PS1~0，定时器 0 端口引脚选择位[1:0]。

T0PS0	T0/T0CKO
0	P4.6
1	P4.4

Bit 0：T0XL 是定时器 0 预分频控制位。T0XL 功能定义请参考 T0X12 (AUXR2.2)。

AUXR4:辅助寄存器 4

SFR 页 = **仅 1 页**
SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
T2PS1	T2PS0	0	T1PS0	0	0	AC0OE	AC0FLT1
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 5：保留位。当写 AUXR4 寄存器时软件，这些位必须写 "**0**"。

Bit 4：T1PS1~0，定时器 1 端口引脚选择位[1:0]。

T1PS0	T1/T1CKO
0	P3.3
1	P4.5

AUXR9:辅助寄存器 9

SFR 页 = **仅 6 页**

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	SIDPS0	T1G1	T0G1	C0FDC1	C0FDC0	0	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 5：T1G1，定时器 1 门控源选择。

T1G1, T1GATE	T1 门控源
00	禁止
01	INT1 激活
10	TF2 激活
11	KBI 激活

Bit 4: T0G1, Gating source selection of Timer 0.

T0G1, T0GATE	T0 门控源
00	禁止
01	INT0 激活
10	TF1 激活
11	KBI 激活

15.2. 定时器 2

定时器 2 是一个 16 位定时器/计数器，既可作为一个定时器也可以作为一个事件计数器，由 T2CKS, T2X12 和 C/T2 选择。定时器 2 有 5 种通过 T2CON, T2MOD 和 T2MOD1 相关位定义的操作模式：捕获、自动加载定时器(向上或向下计数)、8 位 PWM、波特率发生器和可编程时钟输出。

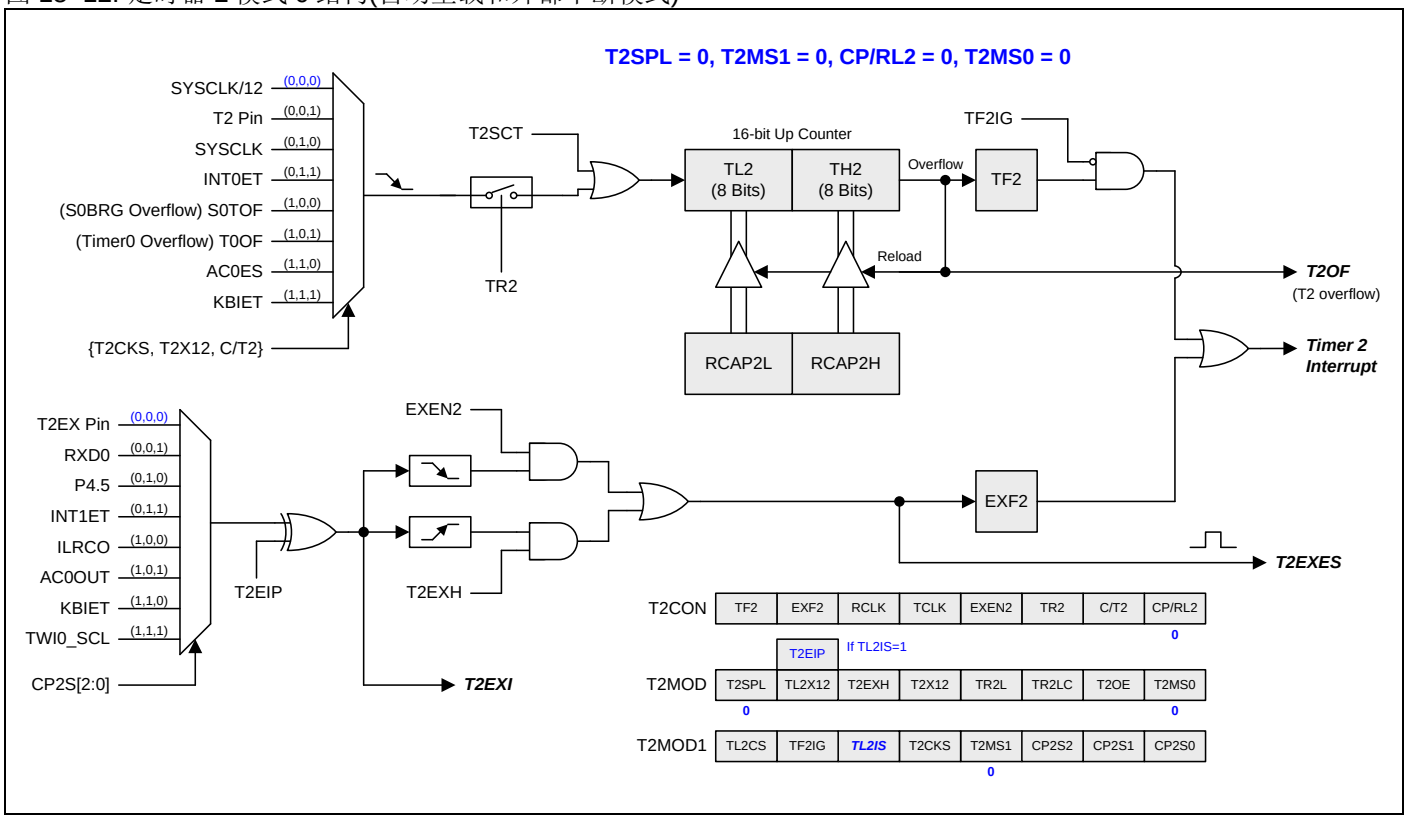
15.2.1. 定时器 2 模式 0(自动重载和外部中断)

在这个模式中，定时器 2 提供一个 16 位的自动重载定时器/计数器。TF2 是定时器 2 的溢出标志，是一个中断会被 TF2IG 阻断的定时器 2 中断源。EXEN2 使能 T2EXI 引脚的下跳沿置位 EXF2，EXF2 作为一个外部中断与 TF2 共享定时器 2 中断。T2EXI 有 8 种定时器 2 的外部输入选择。T2EXH 的功能与 EXEN2 一样，只是 T2EXH 使能 T2EXI 引脚的上升沿置位 EXF2。

本模块中的定时器 2 溢出事件(T2OF)可以作为时钟输入或事件源输出到其它外设。

定时器 2 模式 0 如图 15–12 所示。

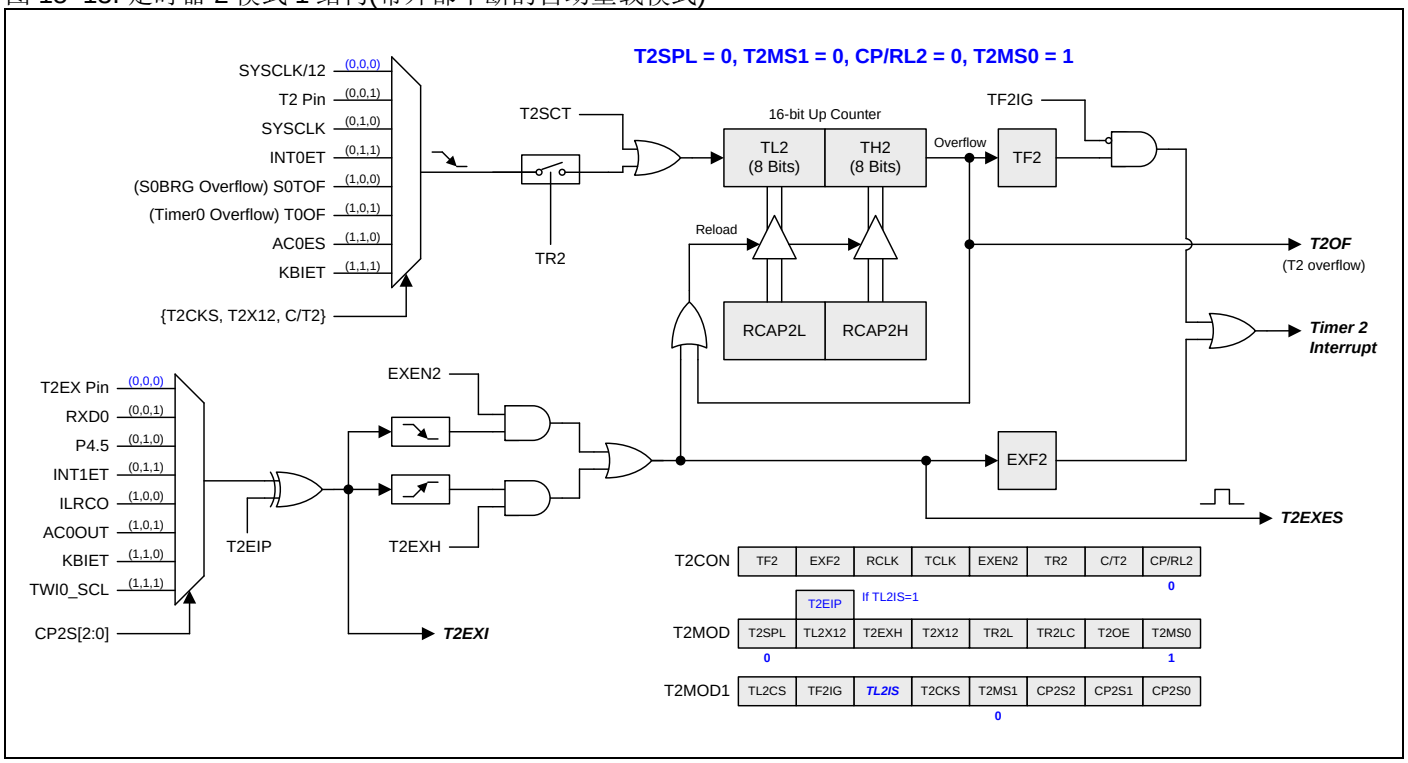
图 15–12. 定时器 2 模式 0 结构(自动重载和外部中断模式)



15.2.2. 定时器 2 模式 1(带外部中断的自动重载)

如图 15–13 所示，定时器 2 模式 1，使能定时器 2 自动向上计数。本模式有 T2CON 寄存器的 EXEN2 决定的两种选择。如果 EXEN2=0，定时器 2 向上计数到 0xFFFFH 且直到溢出而置位 TF2(溢出标志)。这时定时器 2 寄存器被重载入 RCAP2L 和 RCAP2H 的 16 位数据。RCAP2L 和 RCAP2H 的值由软件预设。如果 EXEN2=1，16 位重载会被一个溢出或一个 T2EXI(8 个定时器 2 外部输入的 1 个)引脚的下跳沿触发。跳变同样置位 EXF2。如果定时器 2 中断使能，无论 TF2 或 EXF2 置位则产生中断。T2EXH 的功能与 EXEN2 一样，只是 T2EXH 使能 T2EXI 引脚的上升沿置位 EXF2。

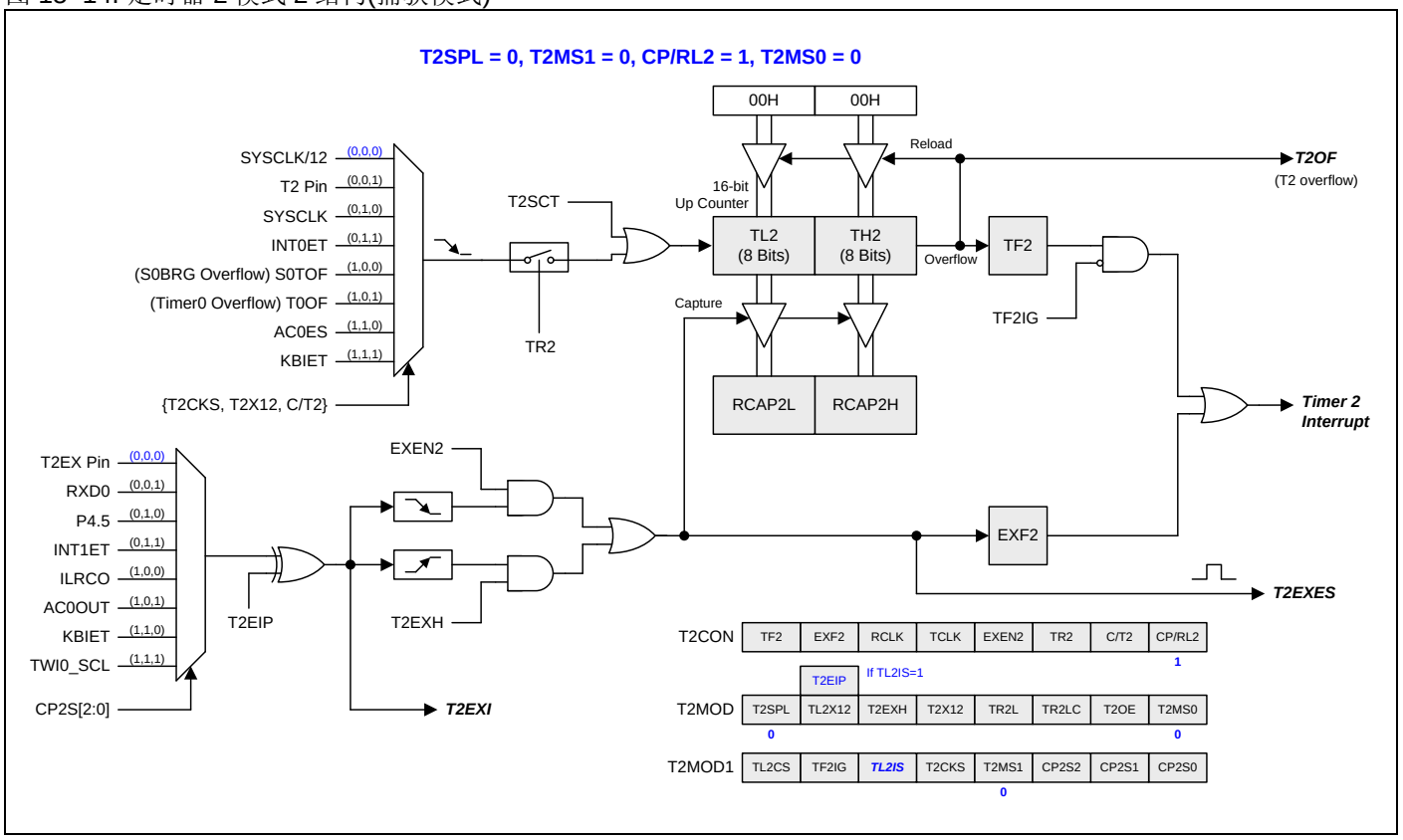
图 15–13. 定时器 2 模式 1 结构(带外部中断的自动重载模式)



15.2.3. 定时器 2 模式 2(捕获)

由 T2CON 寄存器的位 EXEN2 做二种选择的捕获模式如图 15–14 所示。如果 EXEN2=0，定时器 2 是一个 16 位定时器或计数器，向上溢出而置位 TF2(定时器 2 溢出标志)。TF2 常用来产生中断(IE 寄存器使能定时器 2 中断相关位)。如果 EXEN2=1，定时器 2 仍然具有前面的功能，增加了 T2EXI 引脚下跳沿(8 个定时器 2 外部输入的 1 个)把定时器 2 寄存器(TH2 和 TL2)各自捕获到寄存器(RCAP2H 和 RCAP2L)。另外，T2EXI 引脚跳变使 T2CON 寄存器的 EXF2 置位，且 EXF2(像 TF2)会产生一个跟定时器 2 溢出一样位置的中断。T2EXH 的功能与 EXEN2 一样，只是 T2EXH 使能 T2EXI 引脚的上升沿置位 EXF2。

图 15–14. 定时器 2 模式 2 结构(捕获模式)

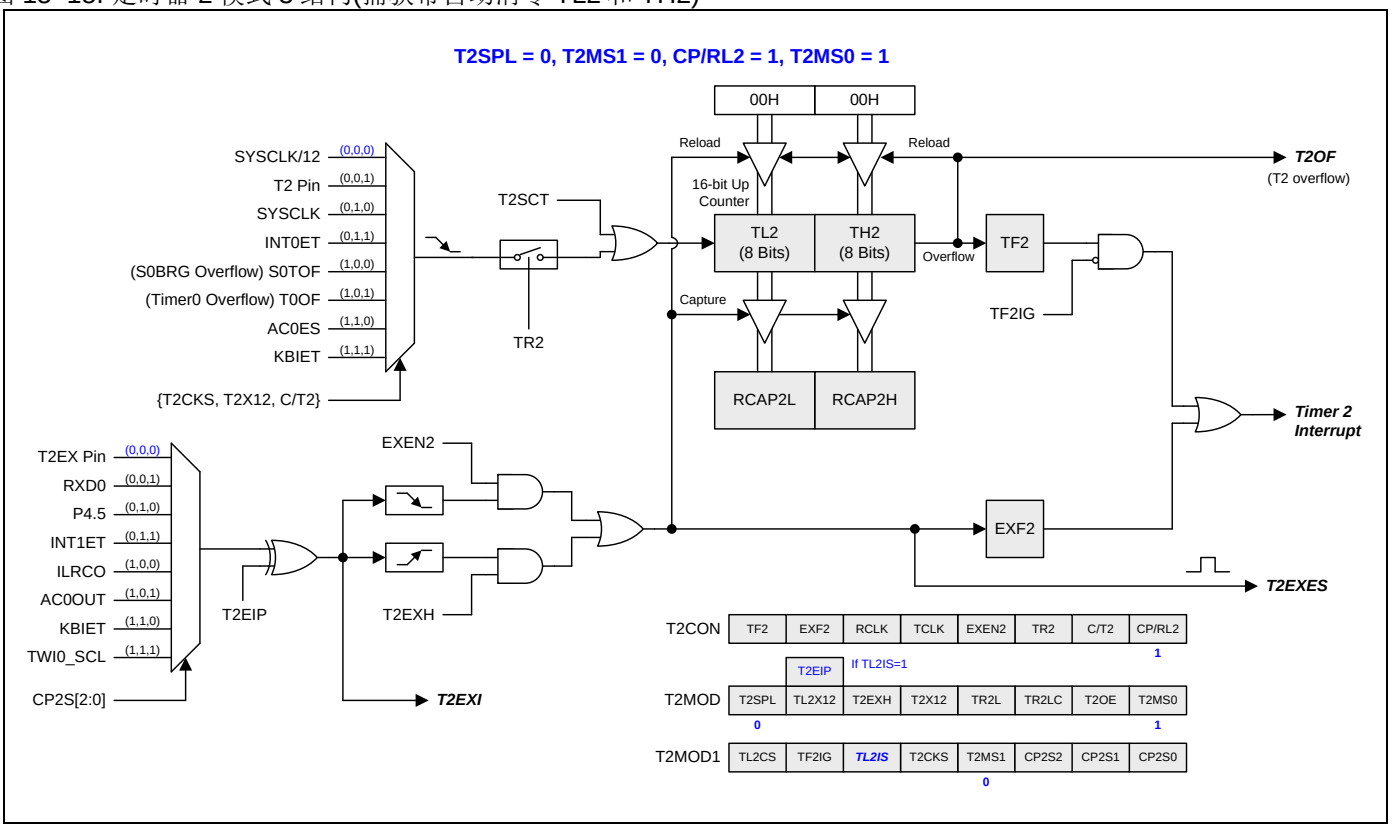


15.2.4. 定时器 2 模式 3(捕获带自动清零)

定时器 2 模式 3 与定时器 2 模式 2 的功能相似。有一点不同的就是 T2EXES、EXF2 置位，不但定时器 2 会被捕获而且 TL2 和 TH2 也会被清零。

定时器 2 模式 3 如图 15–15 所示。

图 15–15. 定时器 2 模式 3 结构(捕获带自动清零 TL2 和 TH2)



15.2.5. 定时器 2 模式 6 (占空比捕获)

定时器 2 模式 6 支持捕获输入波形的周期时间或占空比。信号的三个边沿可以计算出周期和占空比。在占空比捕获模式，需要将 TH2，TL2 清零。然后通过置位 TR2 开启捕获模式，但是计数器还没有启动，它会等到第一个边缘进入到外部触发通道，例如 T2EX 引脚。这表示第一个边值是 00H。在第一次触发边缘后，计数器开始计数。请注意，T2EXI 的第一个触发边缘必须是上升沿。即使你设置了 T2EXH，第一个边沿也会被忽略以触发 EXF2。

其次，如果只想计算脉冲宽度，那么可以设置 EXEN2 通过第二个边来触发 EXF2。在本例中，当第二个边缘触发计时器将其值捕获到 RCAP2H: RCAP2L 中时，它还触发用于中断的 EXF2，以标识它已经完成。

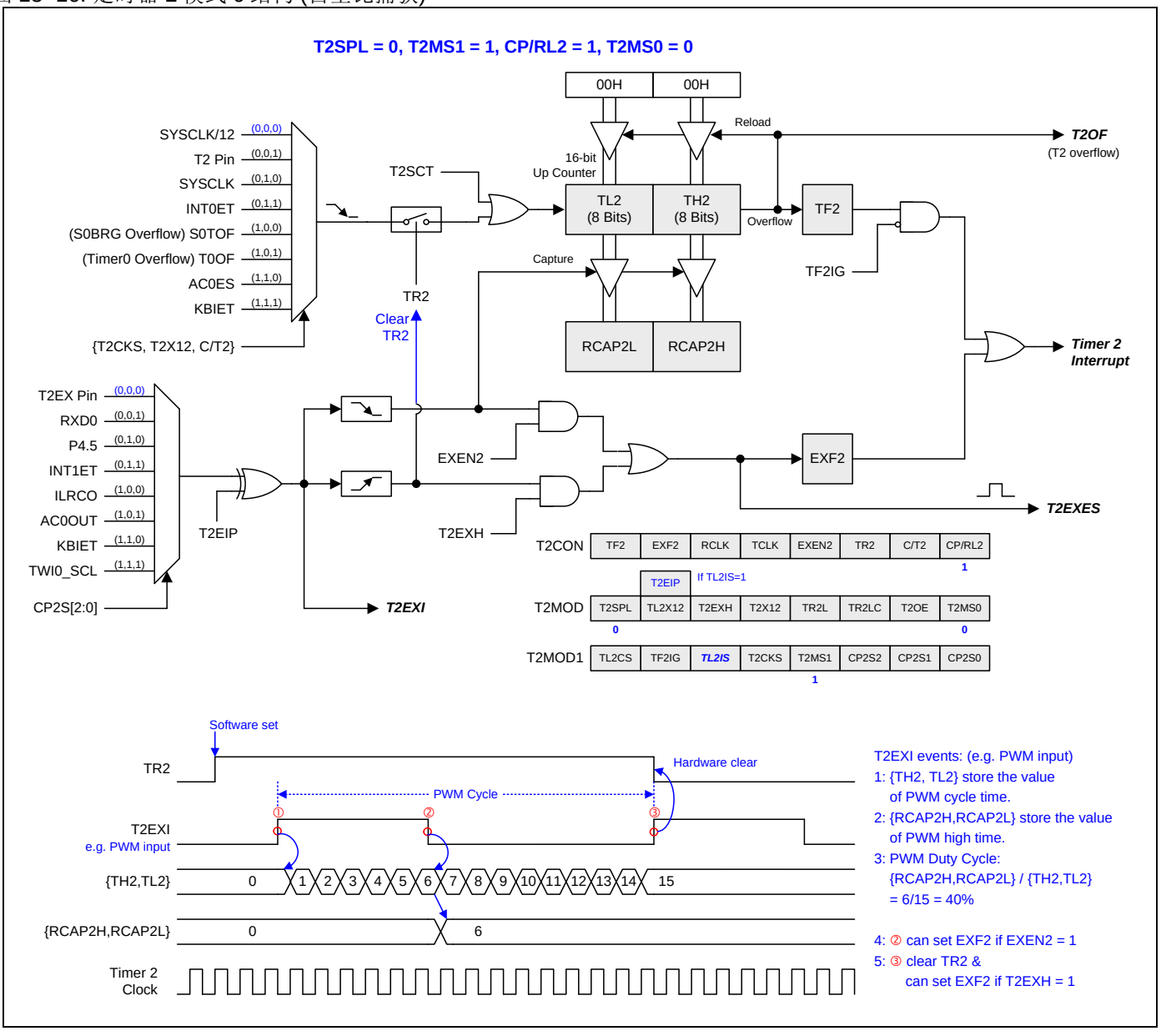
但是如果你想获得周期时间，那么就要通过清除 EXEN2 去阻止第二边沿。

当第三个边沿到来，它将自动清除 TR2 来停止计数器。

使用 TH2: TL2(第三条边)、RCAP2H: RCAP2L(第二条边)和 0 (第一条边)来计算周期时间和占空比。

定时器 2 模式 6 如图 15–16 所示

图 15–16. 定时器 2 模式 6 结构 (占空比捕获)



15.2.6. 分立定时器 2 模式 0 (自动重载和外部中断)

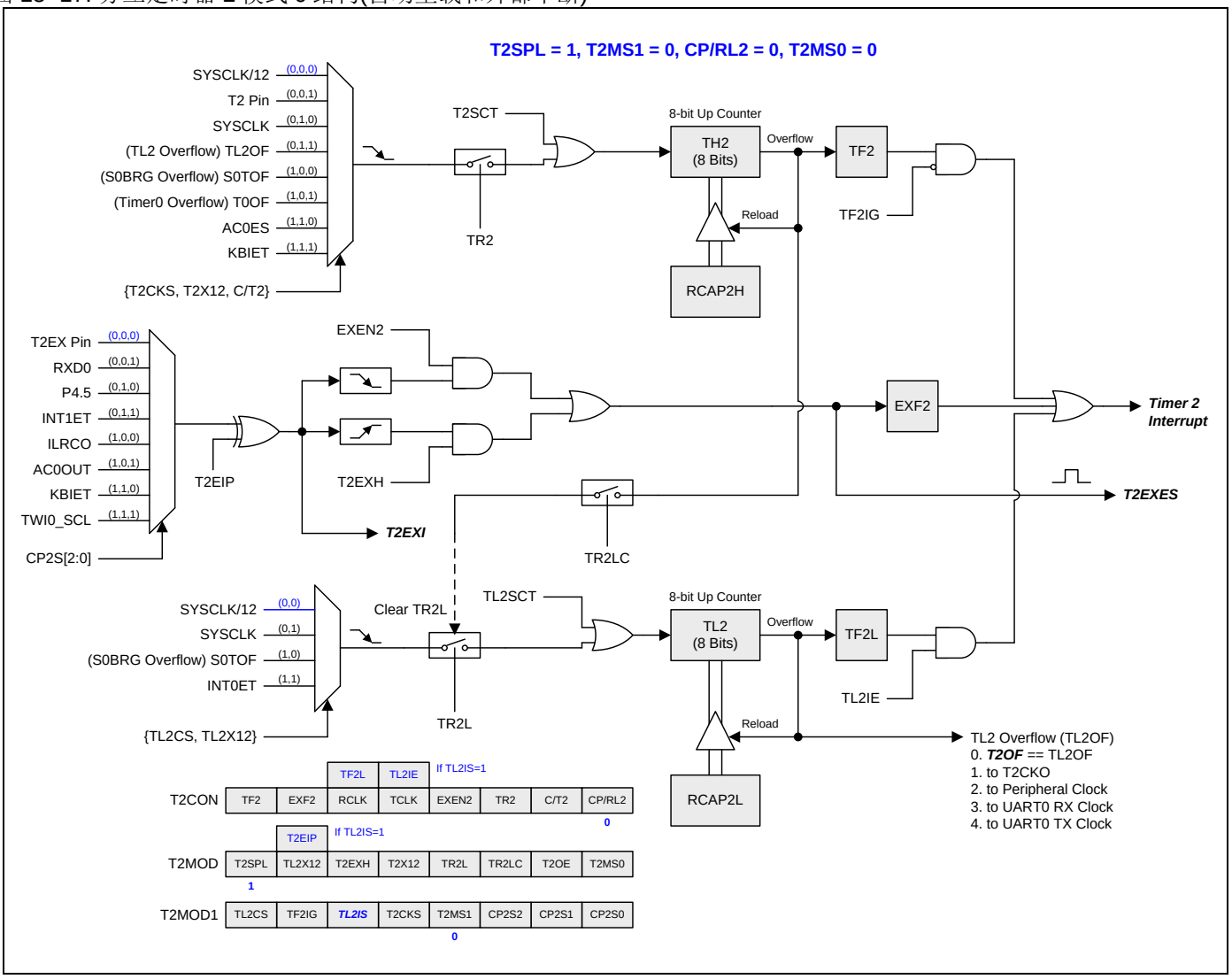
本模式中 T2SPL 置位，定时器 2 变为两个 8 位定时器(TH2 和 TL2)。两个 8 位定时器都是向上计数如图 15–17 所示。TH2 保存 RCAP2H 的重载值和保持 16 位模式一样的 8 个时钟源输入选择。8 位定时器功能跟 16 位模式的定时器 2 模式 0 相似。TL2 保存 4 个时钟输入选择的 RCAP2L 重载值。T2CON 的位 TR2 控制着 TH2 的运行。T2MOD 的位 TR2L 控制着 TL2 的运行。当 TR2LC 置位时 TH2 溢出会停止 TR2L 的运行。

分立模式有 3 个中断标志 EXF2、TF2 和 TF2L。EXF2 与 16 位模式一样的功能用来侦测 T2EXI 引脚的跳变。TF2IG 控制 TF2 在 TH2 从 0xFF 到 0x00 溢出时是否置位。TL2 从 0xFF 到 0x00 溢出时 TF2L 置位，TL2IE 使能中断。EXF2、TF2 和 TF2L 中断标志硬件不会清零且必须软件清零。

顺便说，16 位模式中的定时器 2 溢出事件(T2OF)将被分立模式中的 TL2 溢出事件(TL2OF)取代。

如果 T2MOD1 的位 TL2IS=0，位 T2CON.5~4 是 RCLK 和 TCLK 的功能。如果 TL2IS=1，位 T2CON.5~4 是 TF2L、TL2IE 和 T2EIP 的功能。

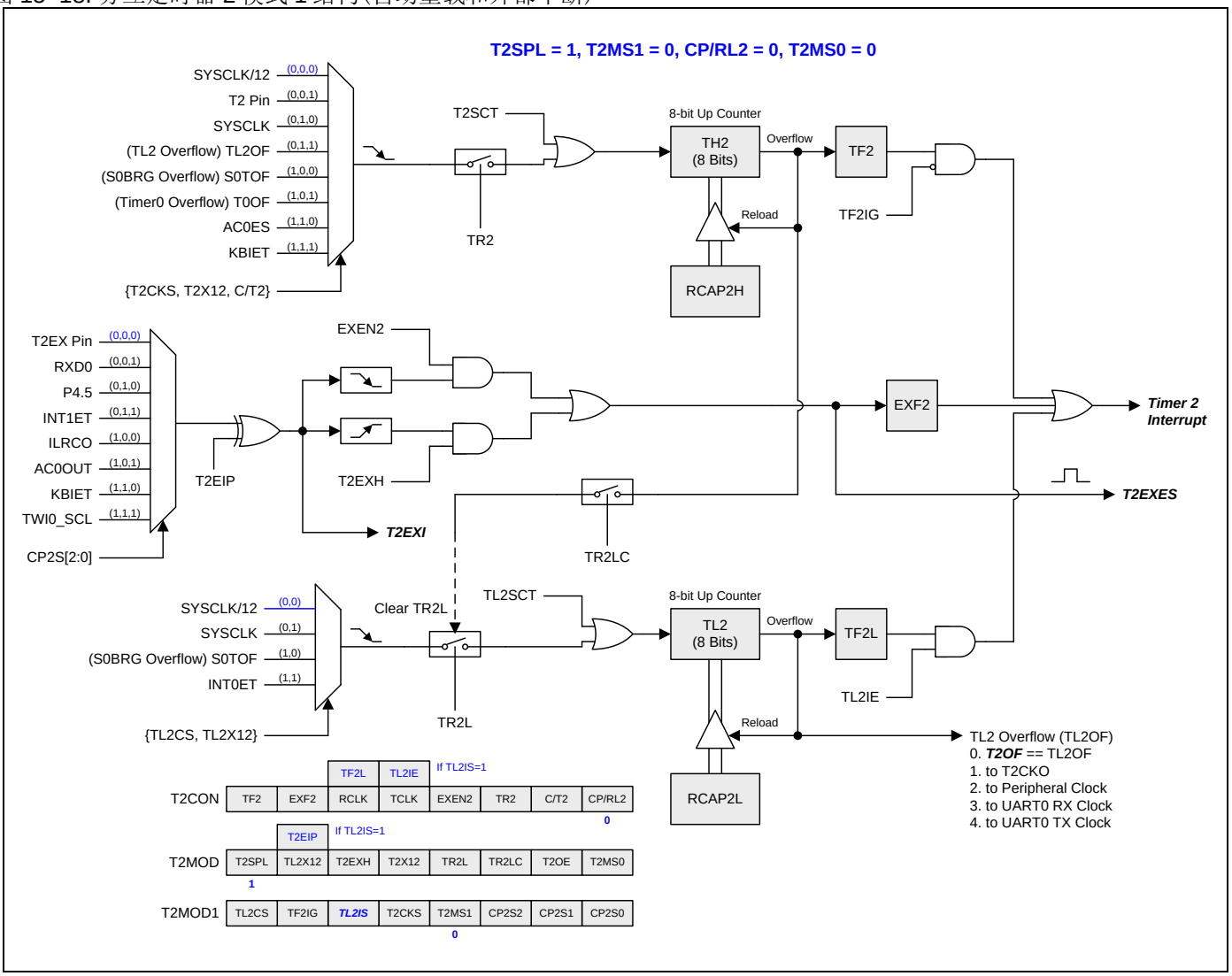
图 15–17. 分立定时器 2 模式 0 结构(自动重载和外部中断)



15.2.7. 分立定时器 2 模式 1 (自动重载和外部中断)

本模式中 T2SPL 置位，定时器 2 变为两个 8 位定时器如图 15–18 所示。跟定时器 2 模式 1 相似的功能且保持与分立定时器 2 模式 0 一样的中断方式。

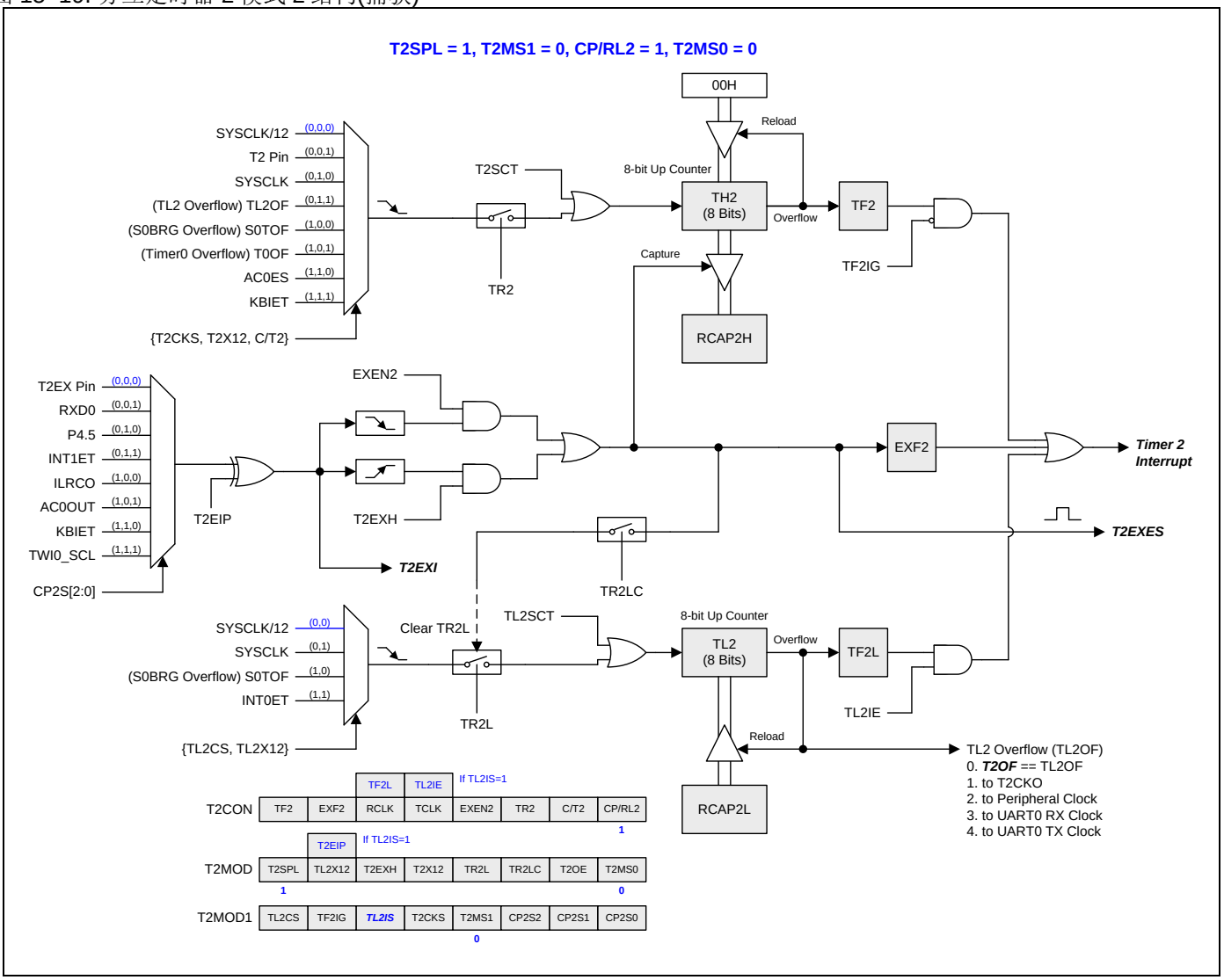
图 15–18. 分立定时器 2 模式 1 结构(自动重载和外部中断)



15.2.8. 分立定时器 2 模式 2 (捕获)

本模式中 T2SPL 置位，定时器 2 变为两个 8 位定时器如图 15–19 所示。跟定时器 2 模式 2 相似的功能且保持与分立定时器 2 模式 0 一样的中断方式。

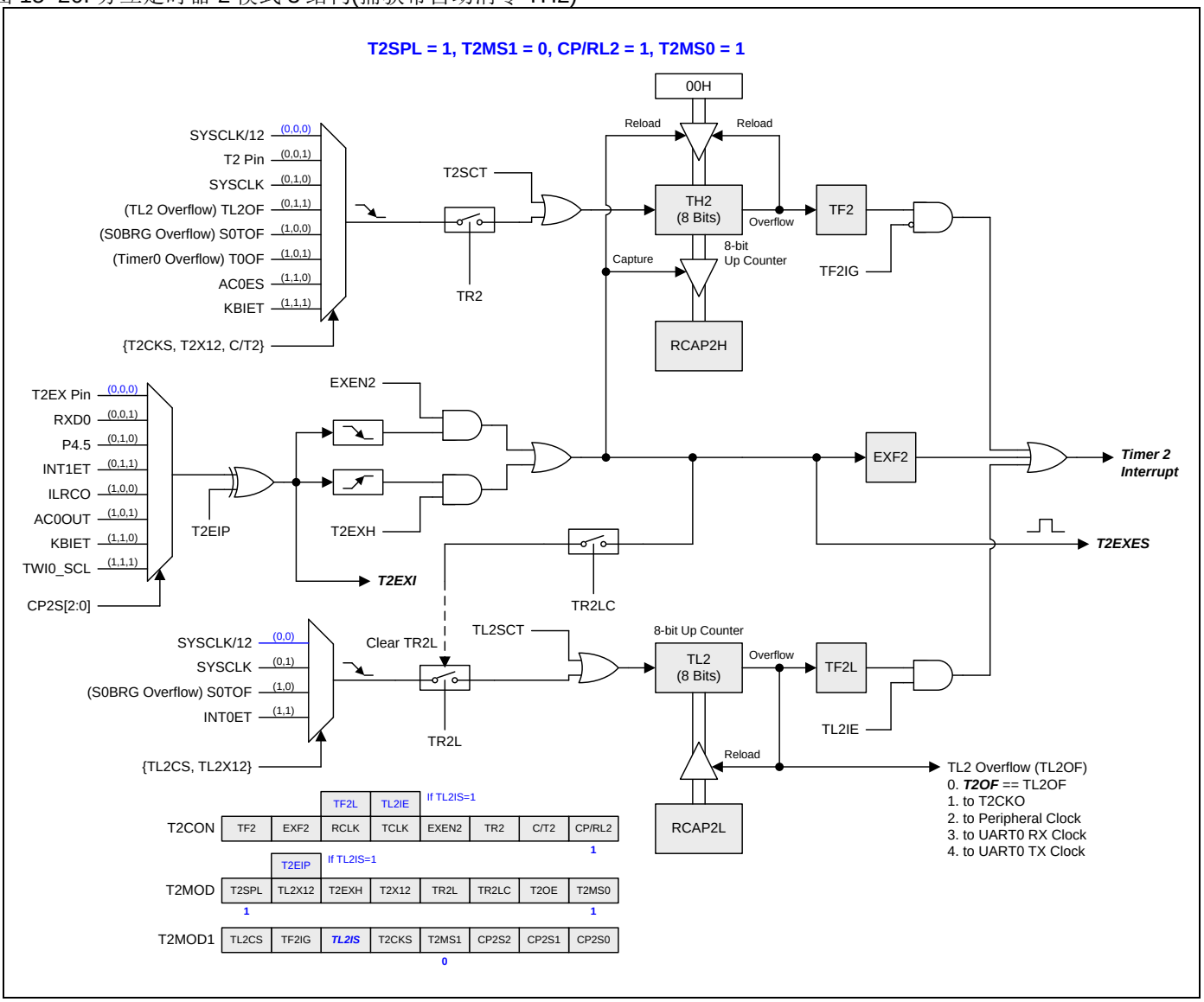
图 15–19. 分立定时器 2 模式 2 结构(捕获)



15.2.9. 分立定时器 2 模式 3 (捕获带自动清零)

本模式中 T2SPL 置位，定时器 2 变为两个 8 位定时器如图 15–20 所示。跟定时器 2 模式 3 相似的功能且保持与分立定时器 2 模式 0 一样的中断方式。

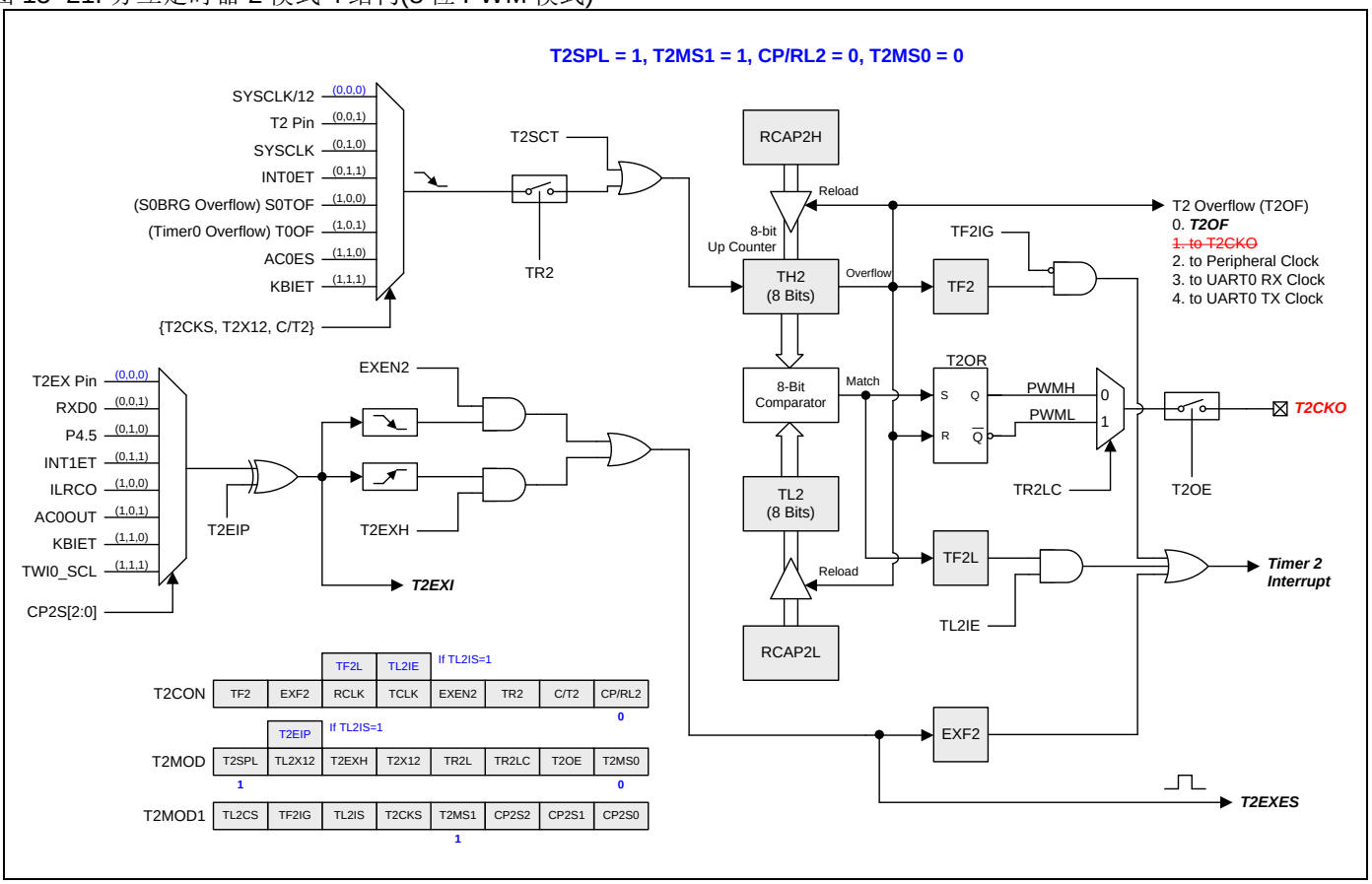
图 15–20. 分立定时器 2 模式 3 结构(捕获带自动清零 TH2)



15.2.10. 分立定时器 2 模式 4 (8 位 PWM 模式)

本模式，定时器 2 是一个 8 位的 PWM 模式如图 15–21 所示。TH2 和 RCAP2H 相结合为一个 8 位的自动重载计数器。这二个寄存器的软件配置决定 PWM 周期。TL2 是 PWM 比较寄存器用来生成 PWM 波形。RCAP2L 是 PWM 缓冲寄存器且在此寄存器中软件更新 PWM 数据。每次 TH2 溢出事件置位 TF2 且 RCAP2L 值载入到 TL2。PWM 信号输出到 T2CKO 功能引脚且输出的开关由 T2MOD 寄存器的位 T2OE 决定。

图 15–21. 分立定时器 2 模式 4 结构(8 位 PWM 模式)



15.2.11. 波特率发生器模式(BRG)

T2CON 寄存器的 RCLK 和/或 TCLK 位允许串口发送和接收波特率源可选择定时器 1 或定时器 2。当 TCLK=0 时，定时器 1 作为串口传送波特率发生器。当 TCLK=1，定时器 2 作为串口传送波特率发生器。RCLK 对串口接收波特率有相同的功能。有了这两位，串口可以有不同的接收和发送波特率，一个通过定时器 1 来产生，另一个通过定时器 2 来产生。

图 15–22 所示定时器 2 在波特率发生器模式 UART 引擎产生 RX 和 TX 时钟(见图 17–6.)。波特率发生器模式像自动加载模式，翻转时将把寄存器 RCAP2H 和 RCAP2L 的值加载到定时器 2 的寄存器，RCAP2H 和 RCAP2L 的值由软件预置。

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

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

注意：
当定时器 2 用作波特率发生器时，参考章节 “17.8.4 模式 1 和 3 波特率” 波特率模式 1 和 3 获取波特率设定值。

如果定时器 2 在分立模式 0，TL2 和 RCAP2L 相结合为一个 8 位的波特率发生器如 图 15–22. 定时器 2 波特率发生器模式 所示。TL2 溢出置位 TF2L，TL2IE 使能中断。TH2 和 RCAP2H 充当一个具有定时器 2 中断能力的自动重载定时器/计数器。

图 15–22. 定时器 2 波特率发生器模式

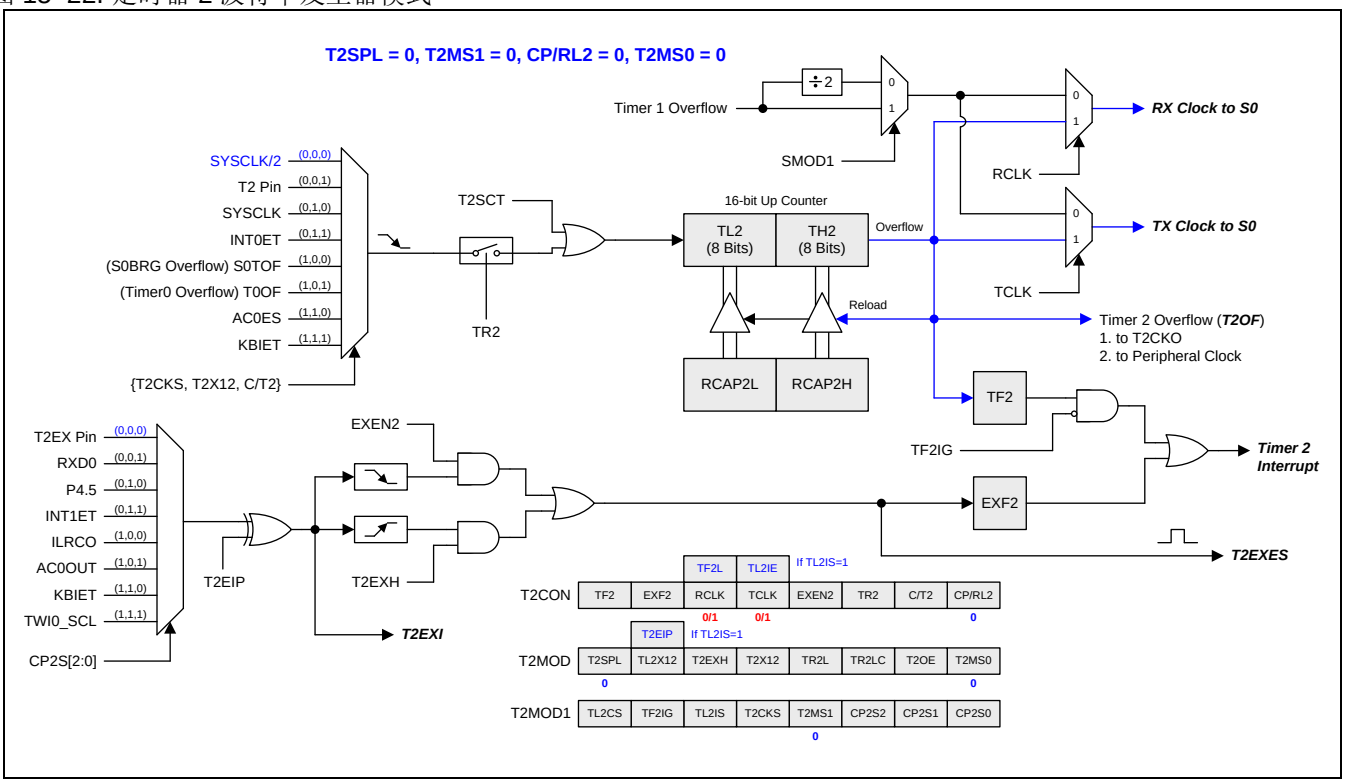
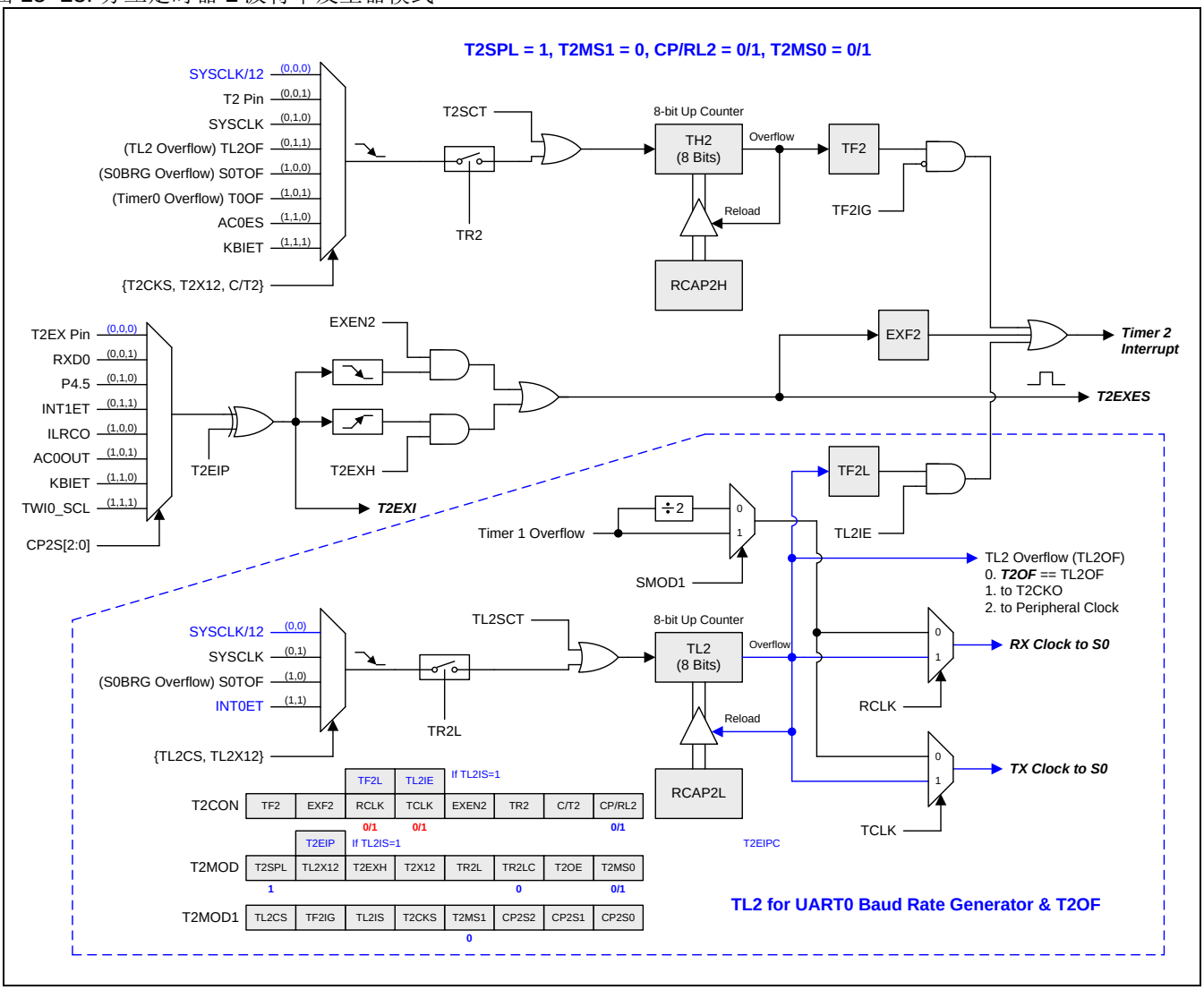


图 15–23. 分立定时器 2 波特率发生器模式



15.2.12. 定时器 2 可编程时钟输出

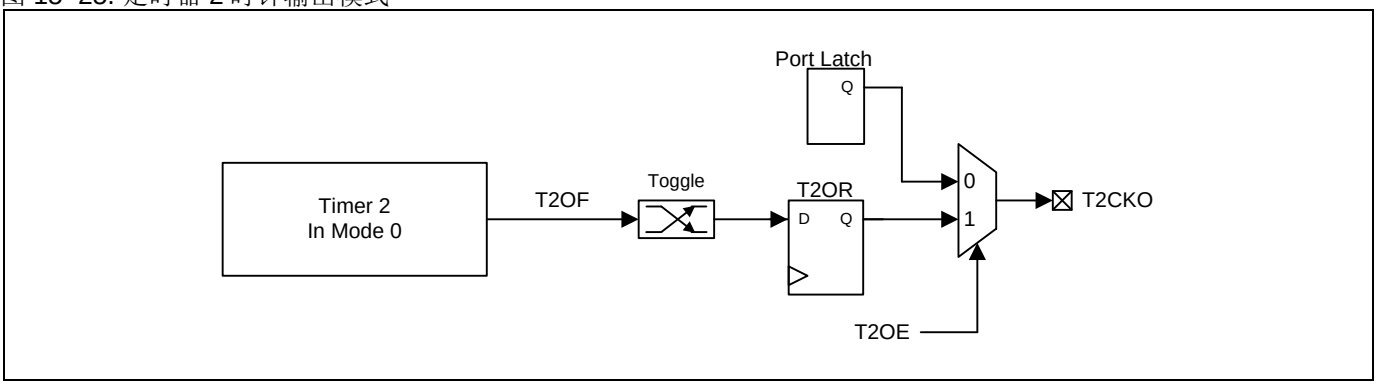
定时器 2 有一个时钟输出模式(当 CP/RL2=0 并且 T2OE=1)。在这个模式，定时器 2 运行为一个占空比为 50%的可编程时钟输出。产生的时钟从 T2CKO 输出。输入时钟(SYSCLK/2 或 SYSCLK)使 16 位定时器(TH2,TL2)加一。定时器从载入值到溢出重复计数。一旦溢出，(RCAP2H, RCAP2L))的值被载入到(TH2,TL2)同时计数。图 15–24 给出了定时器 2 时钟输出频率计算公式。定时器 2 的时钟输出结构如图 15–25 所示。

图 15–24. 定时器 2 时钟输出频率计算公式

$$\text{T2 Clock-out Frequency} = \frac{\text{T2 Clock Frequency}}{2 \times (65536 - (\text{RCAP2H, RCAP2L}))}$$

注意：
(1)定时器 2 溢出标志 TF2,在定时器 2 溢出时置位可产生中断。但是 TF2 中断会被 T2MOD1 寄存器的位 TF2IG 锁住。
(2)当 SYSCLK=12MHz 及 SYSCLK/2 作为定时器 2 时钟源, 定时器 2 可编程输出频率范围从 45.7Hz 到 3MHz。
(3)当 SYSCLK=12MHz 及 SYSCLK 作为定时器 2 时钟源, 定时器 2 可编程输出频率范围从 91.5Hz 到 6MHz。

图 15–25. 定时器 2 时钟输出模式



定时器 2 时钟输出模式如何编程

- 选择定时器 2 时钟源。
- 从公式计算出 16 位自动加载值并输入到 RCAP2H 和 RCAP2L 寄存器。
- 在 TH2 和 TL2 寄存器输入一个跟自动加载值相同的初始值。
- T2MOD 寄存器的 T2OE 置位。
- T2CON 寄存器的 TR2 置位启动定时器 2。

在时钟输出模式，定时器 2 翻转产生中断。这和用作波特率发生器时相似。可同时使用定时器 2 作为一个波特率发生器和时钟发生器。注意，波特率和时钟输出都由定时器 2 的溢出速率来决定且中断会被 TF2IG 锁住。

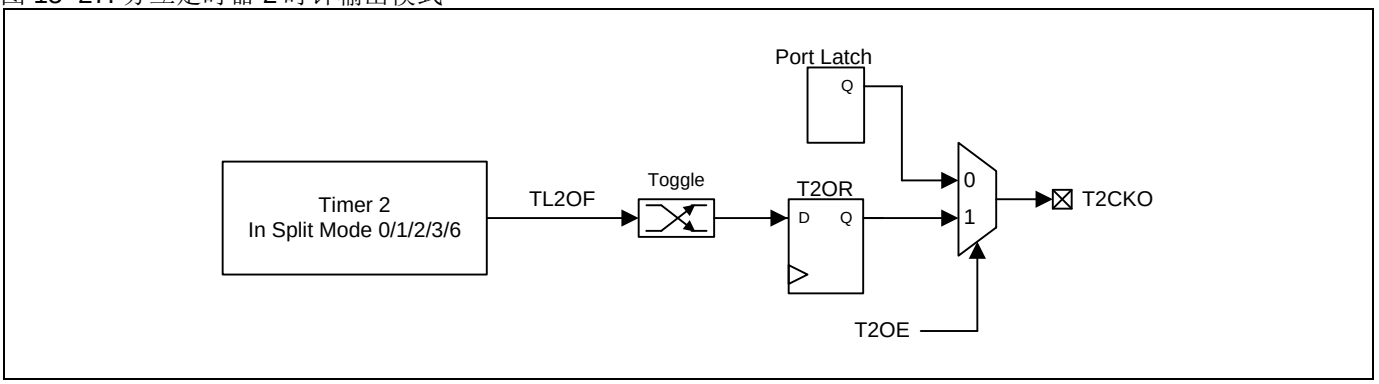
如果定时器 2 在分立模式，时钟输出功能由 TL2 溢出产生且输出时钟频率为 TL2 溢出率的二分之一。当 TL2 溢出时 RCAP2L 是 TL2 重载值。TL2 有 4 种时钟源选择。在使能分立定时器 2 时钟输出功能之前，软件必须结束 TL2 时钟源配置。图 15–26 给出了 TL2 时钟输出频率公式。分立定时器 2 的时钟结构如图 15–27 所示。

图 15–26. 分立定时器 2 时钟输出公式

$$\text{Split T2 Clock-out Frequency} = \frac{\text{TL2 Clock Frequency}}{2 \times (256 - \text{RCAP2L})}$$

注意：
(1) TL2 溢出标志 TF2L,在 TL2 溢出时置位产生中断。但是 TF2L 中断由 T2CON 寄存器的位 TL2IE 使能。
(2)当 SYSCLK=12MHz 及 SYSCLK/12 作为 TL2 时钟源, TL2 可编程输出频率范围从 1.95KHz 到 500KHz。
(3)当 SYSCLK=12MHz 及 SYSCLK 作为 TL2 时钟源, TL2 可编程输出频率范围从 23.44Hz 到 6MHz。

图 15–27. 分立定时器 2 时钟输出模式



分立定时器 2 时钟输出模式如何编程

- 选择 TL2 时钟源。
- 从公式计算出 8 位自动加载值并输入到 RCAP2L 寄存器。
- 在 TL2 寄存器输入一个跟自动加载值相同的初始值。
- T2MOD 寄存器的 T2OE 置位。
- T2MOD 寄存器的 TR2L 置位启动定时器 2。

在时钟输出模式，TL2 翻转产生中断。这和 TL2 用作波特率发生器时相似。可同时使用 TL2 作为一个波特率发生器和时钟发生器。注意，在分立定时器 2 模式下波特率和时钟输出都由 TL2 的溢出速率来决定。TF2L 中断由 T2CON 寄存器的 TL2IE 位使能。

15.2.13. 定时器 2 寄存器

T2CON:定时器 2 控制寄存器

SFR 页 =仅 **0** 页

SFR 地址 = 0xC8 复位值 = 0000-0000

7	6	5	4	3	2	1	0
TF2	EXF2	RCLK/ TF2L	TCLK/ TL2IE	EXEN2	TR2	C/T2	CP/RL2
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：TF2，定时器 2 溢出标志。
0：TF2 必须软件清零。
1：定时器 2 溢出 TF2 置位。当 RCLK=1 或 TCLK=1 时，TF2 不会被置位。

Bit 6：EXF2，定时器 2 外部标志。
0：EXF2 必须软件清零。
1：当 EXEN2=1 且在 T2EX 上有负跳变导致重载或捕获，或者 T2EXH=1 并且在 T2EX 上有一个正跳变，将置位定时器外部标志。当定时器 2 中断使能时，EXF2=1 时将引起 CPU 进入定时器 2 中断向量程序。

TL2IS (T2MOD1.5)必须清零而使能位 RCLK 的访问。
Bit 5：RCLK，接收时钟控制位
0：定时器 1 溢出用于接收时钟。
1：定时器 2 溢出用于串口模式 1 和 3 的接收时钟。

TL2IS (T2MOD1.5)必须置位而使能位 TF2L 的访问。
Bit 5：TF2L，在定时器 2 分立模式中 TL2 溢出标志。
0：TF2L 必须软件清零。
1：在定时器2分立模式中TL2溢出TF2L置位。

TL2IS (T2MOD1.5)必须清零而使能位 TCLK 的访问。
Bit 4：TCLK，发送时钟控制位。
0：定时器 1 溢出用于发送时钟。
1：定时器 2 溢出用于串口模式 1 和 3 的发送时钟。

TL2IS (T2MOD1.5) 必须置位而使能位 TL2IE 的访问。
Bit 4：TL2IE，TF2L 中断使能。
0：禁止 TF2L 中断。
1：使能共享定时器 2 中断入口的 TF2L 中断。

Bit 3：EXEN2，定时器 2 外部使能位在 T2EX 引脚的负跳变。
0：定时器 2 忽略 T2EX 引脚的负跳变事件。
1：如果定时器 2 没有用作串口时钟，在 T2EX 的负跳变时捕获或加载并作为结果。如果定时器 2 配置为串口 0 的时钟，T2EX 保持外部信号侦测并产生 EXF2 标志响应中断。

Bit 2：TR2，定时器 2 运行控制位。如果在定时器 2 分立模式中，仅控制 TH2。
0：定时器 2 停止运行。
1：定时器 2 开启运行。

Bit 1：C/T2，定时器或计数器输入源选择位。该位和T2X12与T2CKS一起决定定时器2的输入来源。如下定义：

T2CKS, T2X12, C/T2	定时器 2 时钟源	分立模式下 TH2 时钟选择
0 0 0	SYSCLK/12	SYSCLK/12
0 0 1	T2 引脚输入	T2 引脚输入
0 1 0	SYSCLK	SYSCLK
0 1 1	INT0ET	TL2OF
1 0 0	S0TOF	S0TOF
1 0 1	T0OF	T0OF
1 1 0	AC0ES	AC0ES
1 1 1	KBIET	KBIET

Bit 0：CP/RL2，定时器 2 模式控制位。参考 T2MOD.T2MS0 的功能定义描述。

T2MOD:定时器 2 模式寄存器

SFR 页 =**仅 0 页**

SFR 地址 = 0xC9 复位值 = 0000-0000

7	6	5	4	3	2	1	0
T2SPL	TL2X12/ T2EIP	T2EXH	T2X12	TR2L	TR2LC	T2OE	T2MS0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：T2SPL，定时器 2 分立模式控制。
0：禁止定时器 2 的分立模式。
1：使能定时器 2 的分立模式。模式 6 分立模式被保留

TL2IS (T2MOD1.5)必须清零而使能位 TL2X12 的访问。
Bit 6：TL2X12，定时器 2 分立模式下 TL2 时钟控制位。

TL2CS, TL2X12	TL2 时钟选择
0 0	SYSCLK/12
0 1	SYSCLK
1 0	S0TOF
1 1	INT0ET

TL2IS (T2MOD1.5)必须清零而使能位 T2EIP 的访问。
Bit 6：T2EIP，T2EXI 输入信号反相控制位。
0：T2EXI 输入信号不反相。
1：T2EXI 输入信号反相。

Bit 5：T2EXH，定时器 2 外部 T2EX 引脚的正跳变使能标志。
0：定时器 2 忽略 T2EX 引脚的正跳变事件。
1：如果定时器 2 没有用作串口 0 时钟，在 T2EX 的正跳变时捕获或加载并作为结果。如果定时器 2 配置为串口 0 的时钟，T2EX 保持外部信号侦测并产生 EXF2 旗标响应中断。

Bit 4：T2X12，定时器 2 时钟源选择。参考 C/T2 的功能定义描述。

Bit 3：TR2L，在定时器 2 分立模式中，TL2 运行控制位。
0：停止 TL2。
1：使能 TL2。

Bit 2：TR2LC，TR2L 清除控制位。
0：禁止硬件事件清零 TR2L。
1：使能 TH2 溢出(定时器 2 在模式 0/1)或者捕获输入(定时器 2 在模式 2/3)时自动清零 TR2L。

Bit 1：T2OE，定时器 2 时钟输出使能位。
0：禁止定时器 2 时钟输出。
1：使能定时器 2 时钟输出。

Bit 0：T2MS0，定时器 2 模式选择位 0。

T2MS1, CP/RL2, T2MS0	定时器 2 模式选择
0 0 0	模式 0：自动重载和外部中断
0 0 1	模式 1：自动重载带外部中断
0 1 0	模式 2：捕获模式
0 1 1	模式 3：捕获模式自动清零
1 0 0	模式 4：8-Bit PWM (如果 T2SPL = 1)
1 1 0	模式 6：占空比捕获
其它	保留

T2MOD1:定时器 2 模式寄存器 1

SFR 页 = **仅 1 页**

SFR 地址 = 0x93 复位值 = 0000-0000

7	6	5	4	3	2	1	0
TL2CS	TF2IG	TL2IS	T2CKS	T2MS1	CP2S2	CP2S1	CP2S0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：TL2CS，在定时器 2 分立模式下的 TL2 时钟选择选择。参考 T2MOD.TL2X12 的功能描述。

Bit 6：TF2IG，TF2 中断忽略。
0：使能 TF2 中断。默认是使能的。
1：禁止 TF2 中断。

Bit 5：TL2IS，TF2L 和 TL2IE 访问控制。
0：使能 RCLK 和 TCLK 的访问功能在位 T2CON.5~4，TL2X12 访问在 T2MOD.6。
1：使能 TF2L 和 TL2IE 的访问功能在位 T2CON.5~4，TL2EIP 访问在 T2MOD.6。

Bit 4：T2CKS，定时器 2 时钟输入选择。参考 C/T2 的功能描述。

Bit 3：T2MS1，定时器 2 模式选择位 1。参考 T2MOD.T2MS0 的功能描述。

Bit 2~0：CP2S.2~0，此 3 位定义定时器 2 的捕获源选择。

CP2S.2~0	定时器 2 捕获源选择
0 0 0	T2EX 引脚
0 0 1	RXD0
0 1 0	P4.5
0 1 1	INT1ET
1 0 0	ILRCO
1 0 1	AC0OUT
1 1 0	KBIET
1 1 1	TWI0_SCL

TL2:定时器 2 低字节寄存器

SFR 页 = **仅 0 页**

SFR 地址 = 0xCC 复位值 = 0000-0000

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

TH2:定时器 2 高字节寄存器

SFR 页 = **仅 0 页**

SFR 地址 = 0xCD 复位值 = 0000-0000

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

RCAP2L:定时器 2 捕获低字节寄存器

SFR 页 = 仅 **0** 页

SFR 地址 = 0xCA 复位值 = 0000-0000

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

RCAP2H:定时器 2 捕获高字节寄存器

SFR 页 = 仅 **0** 页

SFR 地址 = 0xCB 复位值 = 0000-0000

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

AUXR4:辅助寄存器 4

SFR 页 = 仅 **1** 页

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
T2PS1	T2PS0	0	T1PS0	0	0	AC0OE	AC0FLT1
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6：T2PS1~0，定时器 2 端口引脚选择[1:0]。

T2PS1~0	T2/T2CKO	T2EX
0 0	P3.1	P3.0
0 1	P3.3	P4.6
1 0	P4.6	P3.3
1 1	P4.5	P4.4

15.3. 定时器全局控制

当应用要求所有定时器在同步模式下工作时，可是设置寄存器来启动、重载和停止所有定时器。

15.3.1. 所有定时器运行的全局使能

当应用要求所有定时器在同步模式下工作时，仅需置位 TREN0 的 TRxE 或 TRxLE，同时启动定时器。在写入“1”之后，这些寄存器将被硬件自动清零。

TREN0:定时器运行使能寄存器 0

SFR 页 = **仅 1 页**

SFR 地址 = 0x95 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	0	TR2LE	0	0	TR2E	TR1E	TR0E
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6：保留。在 TREN0 写入时，这位软件必须写“0”。

Bit 5：TR2LE，当定时器 2 在分立模式下时，这个位上写“1”设置 TR2L 使能(TR2L=1)来控制 TL2。在写入“1”之后，这个位被硬件自动清零。这个位上写“0”无作用。

Bit 4~3：保留。在 TREN0 写入时，这位软件必须写“0”。

Bit 2：TR2E，这个位上写“1”设置 TR2 使能(TR2=1)。在写入“1”之后，这个位被硬件自动清零。这个位上写“0”无作用。

Bit 1：TR1E，这个位上写“1”设置 TR1 使能(TR1=1)。在写入“1”之后，这个位被硬件自动清零。这个位上写“0”无作用。

Bit 0：TR0E，这个位上写“1”设置 TR0 使能(TR0=1)。在写入“1”之后，这个位被硬件自动清零。这个位上写“0”无作用。

15.3.2. 所有定时器重载的全局使能

TRLC0:定时器重载控制寄存器 0

SFR 页 = **仅 2 页**

SFR 地址 = 0x95 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	0	TL2RLC	0	0	T2RLC	T1RLC	T0RLC
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6：保留。在 TRLC0 写入时，这位软件必须写“0”。

Bit 5：TL2RLC，当定时器 2 在分立模式下时，这个位上写“1”强制 TL2 重载条件激活. 在写入“1”之后，这个位被硬件自动清零。这个位上写“0”无作用。

Bit4~3：保留。在 TRLC0 写入时，这位软件必须写“0”。

Bit 2：T2RLC，当定时器 2 在非分立模式下时，这个位上写“1”强制 TH2 和 TL2 重载。或者在分立模式下强制 TH2 重载。如果定时器在占空比捕获模式下强制重载无效。在写入“1”之后，这个位被硬件自动清零。这个位上写“0”无作用。

Bit 1：T1RLC，定时器 1 模式 2 下，这个位上写“1”强制 TL1 重载。在写入“1”之后，这个位被硬件自动清零。这个位上写“0”无作用。

Bit 0：T0RLC，定时器 0 模式 2 下，这个位上写“1”强制 TL0 重载。在写入“1”之后，这个位被硬件自动清零。这个位上写“0”无作用。

15.3.3. 所有定时器停止的全局使能

TSPC0:定时器停止控制寄存器 ***0***

SFR 页 = **仅 3 页**
SFR 地址 = 0x95 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	0	TL2SC	0	0	T2SC	T1SC	T0SC
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6：保留。在 TSPC0 写入时，这位软件必须写“0”。

Bit 5：TL2SC，当定时器 2 在分立模式下时，这个位上写“1”设置 TR2L 禁止(TR2L=0)。在写入“1”之后，这个位被硬件自动清零。这个位上写“0”无作用。

Bit 4~3：保留。在 TSPC0 写入时，这位软件必须写“0”。

Bit 2：T2SC，这个位上写“1”设置 TR2 禁止(TR2=0)。在写入“1”之后，这个位被硬件自动清零。这个位上写“0”无作用。

Bit 1：T1SC，这个位上写“1”设置 TR1 禁止(TR1=0)。在写入“1”之后，这个位被硬件自动清零。这个位上写“0”无作用。

Bit 0：T0SC，这个位上写“1”设置 TR0 禁止(TR0=0)。在写入“1”之后，这个位被硬件自动清零。这个位上写“0”无作用。

16. 可编程计数器阵列(PCA0)

MG82F6B08 / 6B001/ 6B104 带有一个可编程计数器阵列(PCA0)，该功能与标准定时/计数器相比以更少的 CPU 占用提供了更多的定时能力。它的优点包括减少了软件复杂度并提高了精度。

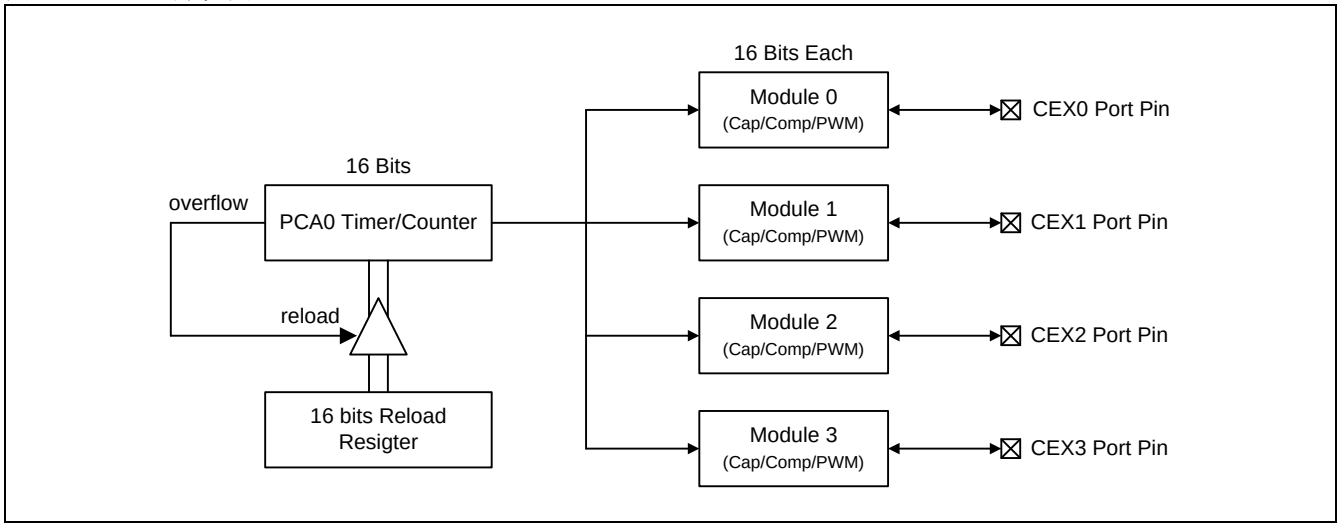
16.1. PCA 概述

PCA0 由一个专用定时/计数器作为一个 4 组比较/捕获/PWM 模块。PCA0 的功能方框图如图 16–1 所示。需要注意的是 PCA0 定时器和模块都是 16 位的。如果一个外部事件同一个模块关联，那么该功能就和相应的端口引脚共享。若某模块没有使用端口引脚，这个引脚还可以用作标准 I/O。

模块 0~3 中的任一组都可以编程为如下任意模式：
-上升和/或下降沿捕获
-软件定时器(比较)
-高速输出(比较输出)
-脉宽调制(PWM)输出
-脉宽调制匹配的比较输出(COPM)

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

图 16–1. PCA 方框图



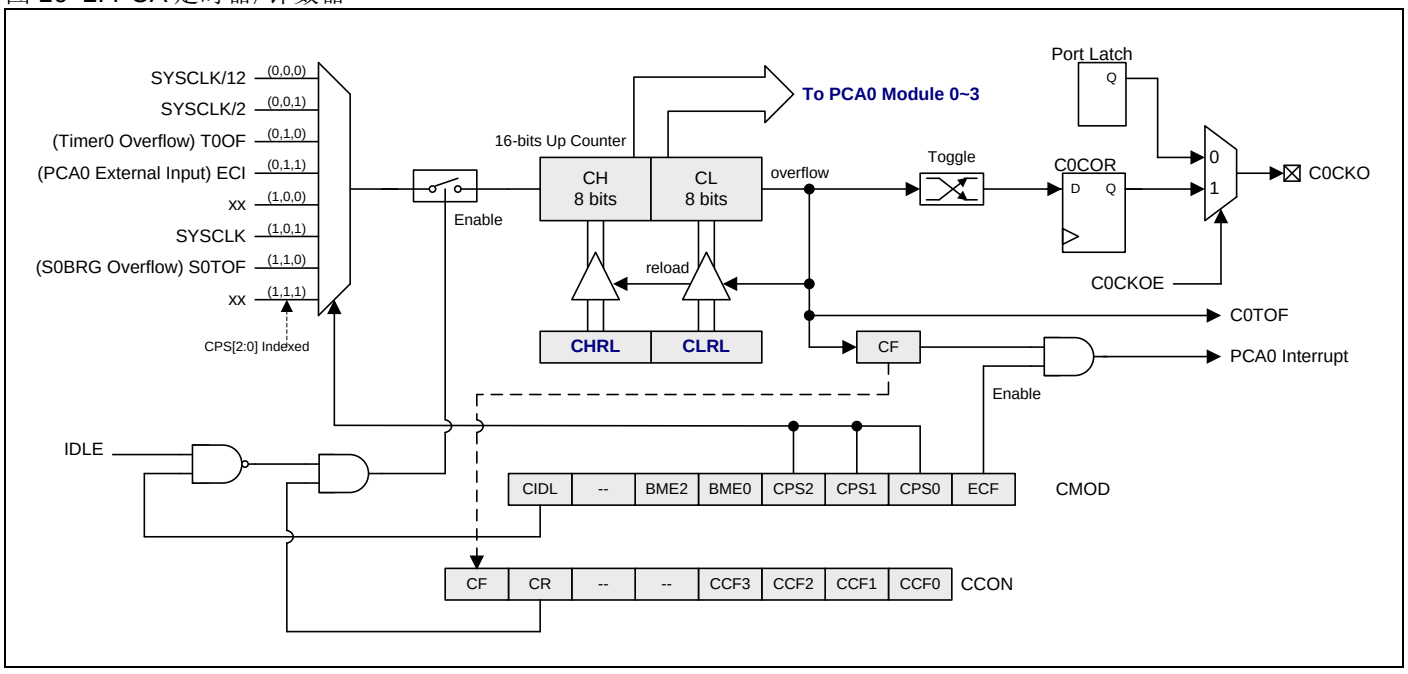
16.2. PCA 定时器/计数器

PCA0 的定时器/计数器由一个可以自动重载的 16 位定时器由寄存器 CH 和 CL(计数值的高低字节)，CHRL 和 CLRL(重载寄存器的高低字节)组成，如图 16–2 所示。{CH + CL}计数器每一次溢出 CHRL 和 CLRL 被重载入 CH 和 CL，这样可以改变 PCA 周期时间提供可变的 PWM 周期，比如 7 位或 9 位的 PWM。

{CH + CL}是所有模块的共有时间基准，它的时钟输入可以从以下来源选择：
- 1/12 系统时钟频率，
- 1/2 系统时钟频率，
-定时器 0 溢出，可以让低频时钟源输入到 PCA 定时器，
-外部时钟输入，ECI 引脚的负跳变，
-直接从系统时钟，
- S0BRG 溢出，S0TOF，

特殊功能寄存器 CMOD 包含了计数脉冲选择位(CPS2、CPS1 和 CPS0)来指定 PCA0 定时器时钟源。这个寄存器也包括了 ECF 位来使能计数器{CH+CL}溢出中断。并且计数器溢出切换 C0COR，当 C0CKOE 使能将输出到端口引脚。此外，用户可以在空闲模式下设置计数器待机位(CIDL)，来关闭 PCA0 定时器。这样可以进一步降低空闲模式下的功耗。

图 16–2. PCA 定时器/计数器



CMOD: PCA 计数器模式寄存器

SFR 页 = **仅 0 页**

SFR 地址 = 0xD9 复位值 = 0000-0000

7	6	5	4	3	2	1	0
CIDL	0	BME2	BME0	CPS2	CPS1	CPS0	ECF
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：CIDL，PCA0 计数器空闲模式控制。
0：PCA0 计数器在空闲模式下继续运行。
1：空闲模式下关闭 PCA0 计数器。

Bit 5：BME2，PCA0 模块 2/3 缓冲模式使能。仅在捕获模式，PWM 模式或 COPM 模式下的 PCA0 模块 2 和 3 有效。
0：PCA0 模块 2/3 禁止缓冲模式。
1：PCA 0 模块 2/3 使能缓冲模式。

Bit 4：BME0，PCA0 模块 0/1 缓冲模式使能。仅在捕获模式，PWM 模式或 COPM 模式下的 PCA0 模块 0 和 1 有效。
0：PCA0 模块 0/1 禁止缓冲模式。
1：PCA0 模块 0/1 使能缓冲模式。

Bit 3~1：CPS2~0，PCA 计数器时钟源选择位。

CPS2	CPS1	CPS0	PCA 时钟源
0	0	0	内部时钟，(system clock)/12
0	0	1	内部时钟，(system clock)/2
0	1	0	定时器 0 溢出
0	1	1	来自 ECI 引脚的外部时钟
1	0	0	保留
1	0	1	内部时钟，(system clock)/1
1	1	0	S0BRT 溢出
1	1	1	保留

Bit 0：ECF，使能 PCA 计数器溢出中断。
0：当 CF 位(CCON 寄存器中)置位时禁止中断。
1：当 CF 位(CCON 寄存器中)置位时使能中断。

如下所示的 CCON 寄存器包含 PCA 运行控制位和 PCA 定时器与每个模块的标志。要运行 PCA，CR 位(CCON.6)必须软件置位。要关闭 PCA，可以清除该位。PCA 计数器溢出时，CF(CCON.7)置位，并且若 CMOD 寄存器的 ECF 置位，还会产生一个中断。CF 位只能软件清零。CCF0 到 CCF3 是模块 0 到模块 3 的各自中断标志位，当发生一个匹配或捕获事件时，硬件置位，这些位必须软件清零。PCA 中断系统如 图 16–3. PCA 中断系统 所示。

CCON: PCA 计数器控制寄存器

SFR 页　　= 仅 0 页
SFR 地址　　= 0xD8　　复位值 = 0000-0000

7	6	5	4	3	2	1	0
CF	CR	0	0	CCF3	CCF2	CCF1	CCF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7：CF，PCA 计数器溢出标志
0：只能软件清零。
1：计数器溢出时硬件置位。CF 标志在 CMOD 寄存器的 ECF 置位时会产生一个中断。CF 可以硬件或软件置位。

Bit 6：CR，PCA 计数器运行控制位。
0：软件清零关闭 PCA 计数器。
1：软件置位开启 PCA 计数器。

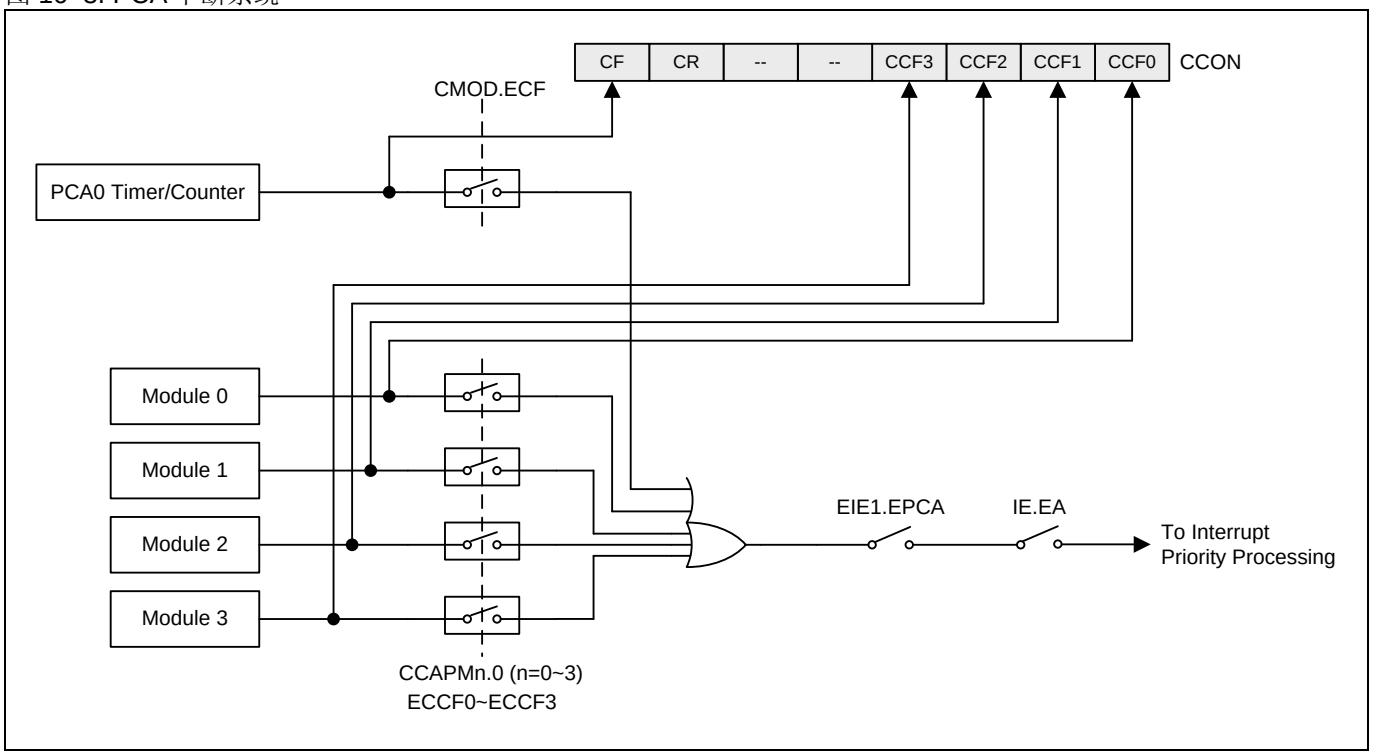
Bit 3：CCF3，PCA 模块 3 中断标志。
0：只能软件清零。
1：当发生一个匹配或捕获事件时，硬件置位。

Bit 2：CCF2，PCA 模块 2 中断标志。
0：只能软件清零。
1：当发生一个匹配或捕获事件时，硬件置位。

Bit 1：CCF1，PCA 模块 1 中断标志。
0：只能软件清零。
1：当发生一个匹配或捕获事件时，硬件置位。

Bit 0：CCF0，PCA 模块 0 中断标志。
0：只能软件清零。
1：当发生一个匹配或捕获事件时，硬件置位。

图 16–3. PCA 中断系统



CH: PCA 基准时钟高字节

SFR 页 = 0 ~ F

SFR 地址 = 0xF9 复位值 = 0000-0000

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

CL: PCA 基准时钟低字节

SFR 页 = 0 ~ F

SFR 地址 = 0xE9 复位值 = 0000-0000

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

CHRL: PCA CH 重载寄存器

SFR 页 = 0 ~ F

SFR 地址 = 0xCF 复位值 = 0000-0000

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

Bit 7~0：CHRL，CH 的重载值。

CLRL: PCA CL 重载寄存器

SFR 页 = 0 ~ F

SFR 地址 = 0xCE 复位值 = 0000-0000

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

Bit 7~0：CLRL，CL 的重载值。

16.3. 比较/捕获模块

比较/捕获模块 0~3 中的每一组都有一个模式寄存器，叫做 CCAPMn(n = 0、1、2、3)，用来选择其工作模式。ECCFn 位控制当模块中断标志置位时每个模块的中断使能。

CCAPMn: PCA 模块比较/捕获寄存器，n=0~3
SFR 页 = **n=0~1 时仅 0 页 (n=2~3 则是所有页)**
SFR 地址 = 0xDA~0xDD 复位值 = 0000-0000

7	6	5	4	3	2	1	0
DTEn	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：DTEn，使能 PWMHn/PWMLn 输出对的死区时间控制。此位仅在 n= 0 和 2 有效且当 PWM 通道操作在缓冲模式死区时间功能激活。通道的缓冲模式由 CMOD 的 BME0=或 BME2 使能
0：PWMn 输出禁止死区时间控制。
1：PWMn 输出使能死区时间控制。

Bit 6：ECOMn，比较器使能。
0：禁止数字比较器功能。
1：使能数字比较器功能。

Bit 5：CAPPn，正跳变捕获使能。
0：禁止 PCA 捕获功能在 CEXn 引脚上正跳变侦测。
1：使能 PCA 捕获功能在 CEXn 引脚上正跳变侦测。

Bit 4：CAPNn，负跳变捕获使能。
0：禁止 PCA 捕获功能在 CEXn 引脚上负跳变侦测。
1：使能 PCA 捕获功能在 CEXn 引脚上负跳变侦测。

Bit 3：MATn，匹配控制。
0：禁止数字比较器匹配事件去置位 CCFn。
1：PCA 计数器同相应模块的比较/捕获寄存器匹配时 CCON 寄存器的 CCFn 置位。

Bit 2：TOGn，切换控制。
0：禁止数字比较器匹配事件去切换 CEXn。
1：PCA 计数器同相应模块的比较/捕获寄存器匹配时使 CEXn 引脚切换。

Bit 1：PWMn，PWM 控制。
0：禁止 PCA 模块中的 PWM 模式。
1：使能 PWM 功能并使 CEXn 引脚用作脉宽调制输出。

Bit 0：ECCFn，使能 CCFn 中断。
0：禁止 CCON 寄存器中的比较/捕获标志位 CCFn 产生中断。
1：使能 CCON 寄存器中的比较/捕获标志位 CCFn 产生中断。

注意：CAPNn (CCAPMn.4)位和 CAPPn (CCAPMn.5)位决定了捕获输入的信号脉冲沿，若这两位同时设置，则正负跳变都会发生捕获。

每个模块都有一对 8 位比较/捕获寄存器(CCAPnH, CCAPnL)。这些寄存器用来存储一个捕获事件发生的时间或者一个比较时间产生的时间。当模块用于 PWM 模式时，除这两个寄存器之外，一个扩展寄存器 PCAPWMn 用来扩展输出占空比的范围，扩展的范围从 0%到 100%，步距是 1/256。关于 10/12/16 位 PWM 详情请参考章节 16.4.6 和 16.4.7。

CCAPnH: PCA 模块 n 捕获高寄存器，n=0~3

SFR 页 = **n=0~1 时仅 0 页 (n=2~3 则是所有页)**

SFR 地址 = 0xFA~0xFD 复位值 = 0000-0000

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

CCAPnL: PCA 模块 n 捕获低寄存器，n=0~3

SFR 页 = **n=0~1 时仅 0 页 (n=2~3 则是所有页)**

SFR 地址 = 0xEA~0xED 复位值 = 0000-0000

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

PCAPWMn: PWM 模式辅助寄存器，n=0~3

SFR 页 = **n=0~1 时仅 0 页 (n=2~3 则是所有页)**

SFR 地址 = 0xF2~0xF5 复位值 = 0000-0000

7	6	5	4	3	2	1	0
PnRS1	PnRS0	0	0	0	PnINV	ECAPnH	ECAPnL
R/W	**R/W**	**W**	**W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6：PnRS1~0，PWMn 分别率设置位 1~0。
00：8 位 PWMn，当[CH, CL]计数从 XXXX-XXXX-1111-1111 到 XXXX-XXXX-0000-0000 时溢出激活。
01：10 位 PWMn，当[CH, CL]计数从 XXXX-XX11-1111-1111 到 XXXX-XX00-0000-0000 时溢出激活。
10：12 位 PWMn，当[CH, CL]计数从 XXXX-1111-1111-111 到 XXXX-0000-0000-0000 时溢出激活。
11：16 位 PWMn，当[CH, CL]计数从 1111-1111-1111-1111 到 0000-0000-0000-0000 时溢出激活。

Bit 5~4：保留位。当 PCAPWMn 被写入时，这些位必须软件写“0”。

Bit 2：PnINV，比较/PWM 输出(C0PnOR)在 CEXn 引脚上反转。
0：比较/PWM 输出(C0PnOR)不反转。
1：比较/PWM 输出(C0PnOR)反转。

Bit 1：ECAPnH，扩展第 9 位(MSB)，联合 CCAPnH 形成 9 位寄存器用于 PWM 模式。

Bit 0：ECAPnL，扩展第 9 位(MSB)，联合 CCAPnL 形成 9 位寄存器用于 PWM 模式。

16.4. PCA 操作模式

不同 PCA 功能对应的 CCAPMn 寄存器设置如表 16–1 所示。

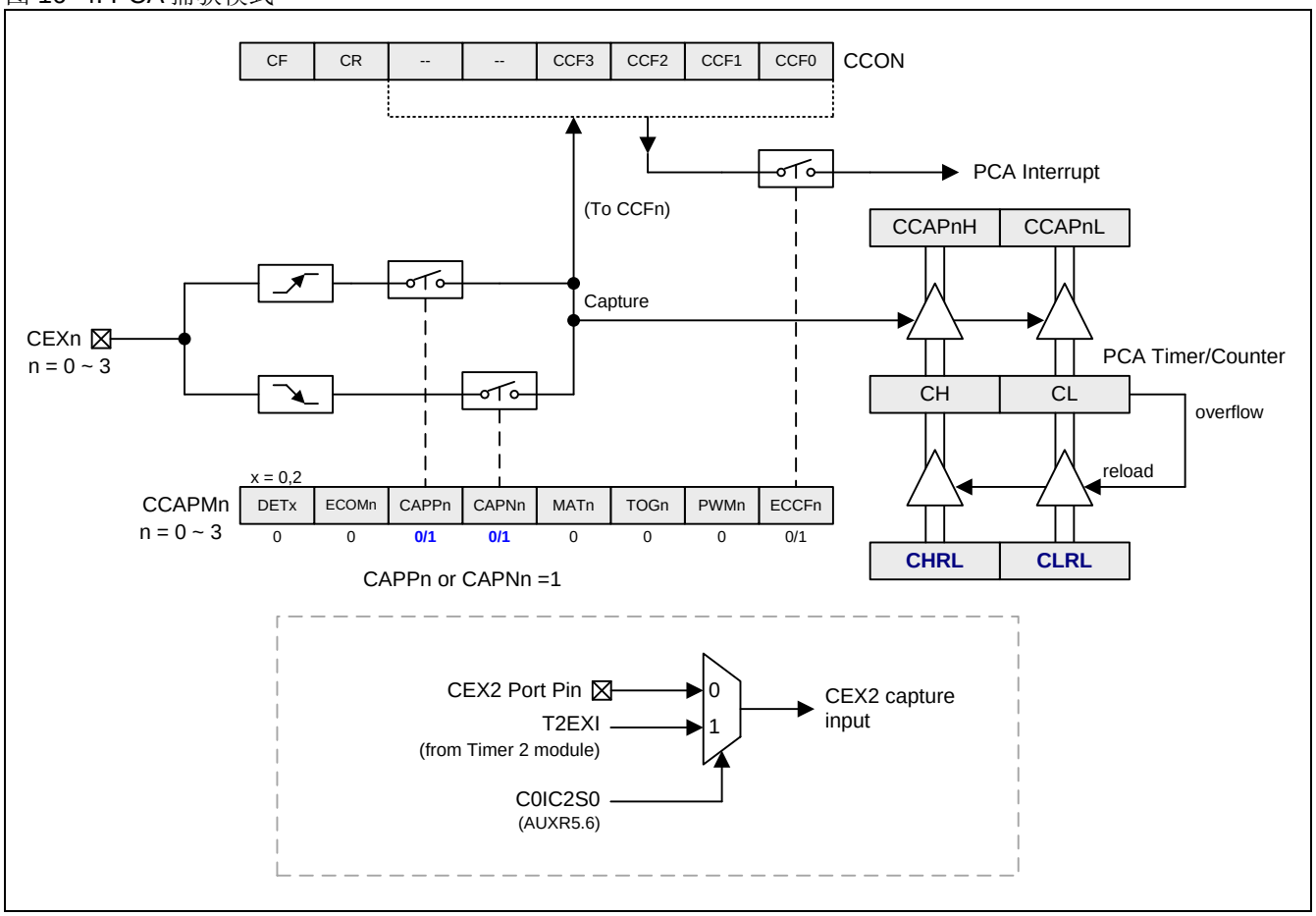
表 16–1. PCA 模块模式

ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn	模块功能
0	0	0	0	0	0	0	无操作
X	1	0	0	0	0	X	CEXn 引脚正跳变触发 16 位捕获
X	0	1	0	0	0	X	CEXn 引脚负跳变触发 16 位捕获
X	1	1	0	0	0	X	CEXn 引脚正负跳变触发 16 位捕获
1	0	0	1	0	0	X	16 位软件定时器（比较）
1	0	0	1	1	0	X	16 位高速输出(HSO)
1	0	0	0/1	0	1	X	脉宽调制器 (PWM)
1	0	0	0	1	1	X	PWM 匹配事件的比较输出(COPM)
1	0	1	0	0	1	X	FIFO 数据模式

16.4.1. 捕获模式

要让某一PCA模块工作在捕获模式，CAPN和CAPP任何一位或两位必须置位。外部CEX输入会在每次跳变时采样。当有效跳变发生时，PCA硬件会将PCA计数器寄存器值(CH和CL)载入到模块的捕获寄存器(CCAPnH和CCAPnL)。若模块的CCFn和ECCFn标志同时置位，会产生一个中断。

图 16–4. PCA 捕获模式



16.4.2. 缓冲捕获模式

为了捕获窄的输入信号，缓冲捕获模式是必要的。如果使能，将把奇数模块捕获数据寄存器(CCAPnH，CCAPnL，n= 1、3)送到偶数模块捕获数据寄存器(通道0、2)。这不影响模块0/2的捕获操作。BME0使能通道0/1的缓冲操作。BME2控制模块2/3。

图 16–5. PCA 缓冲捕获模式 (BMEn=1, n= 0, 2)

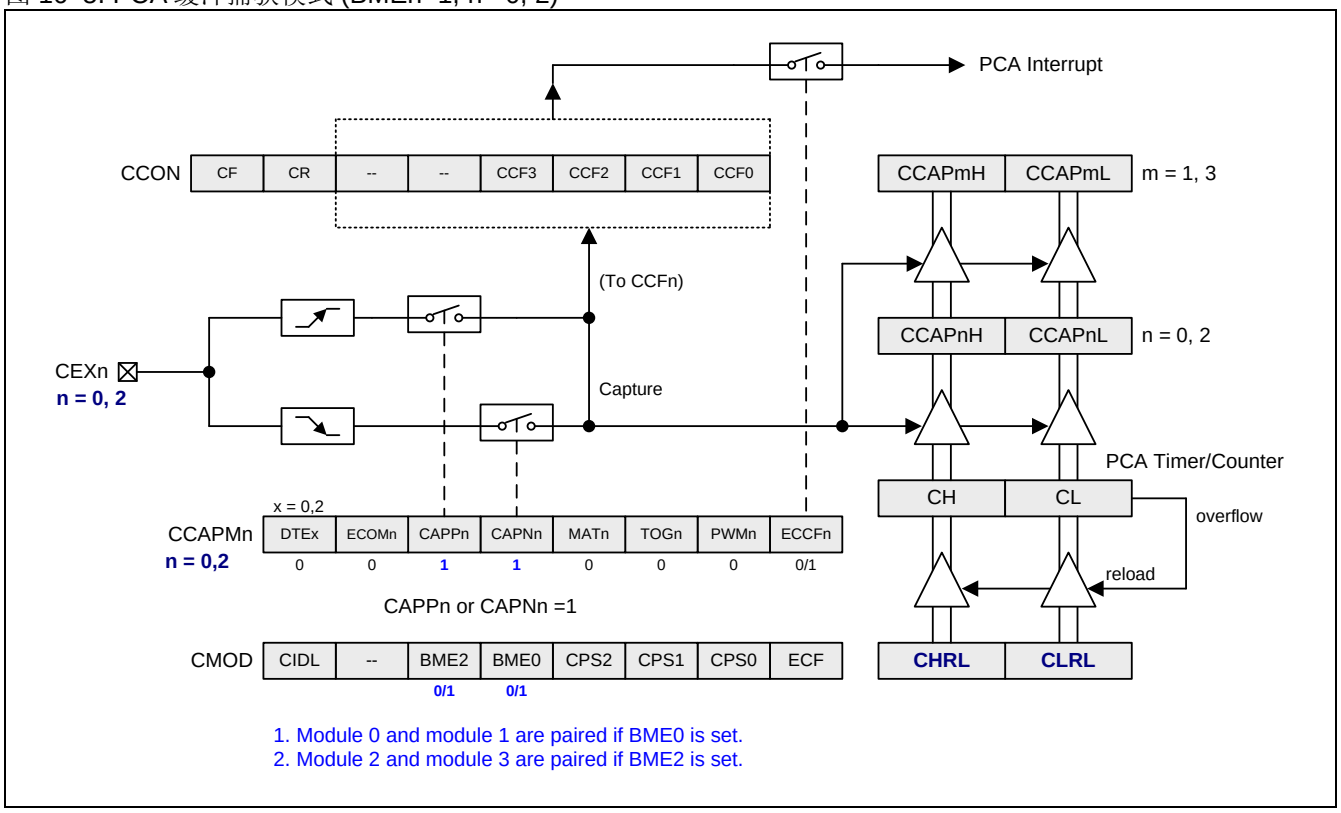
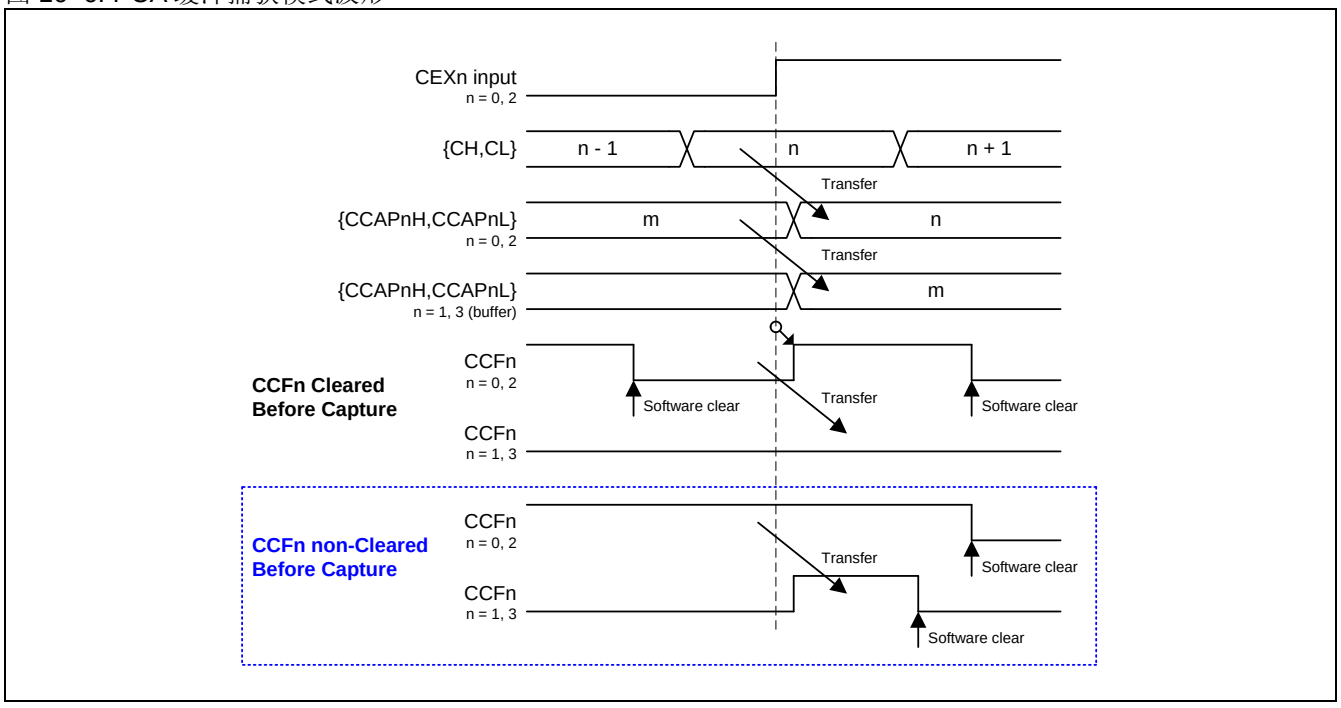


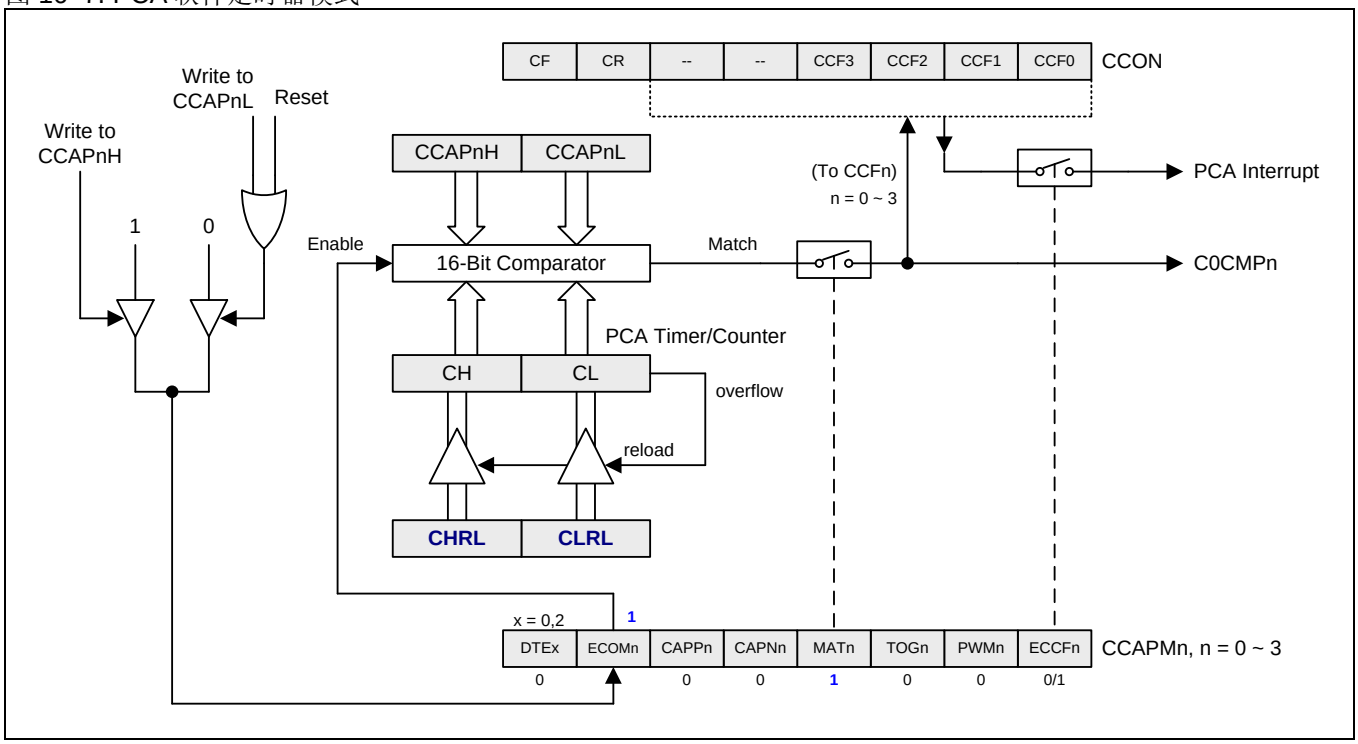
图 16–6. PCA 缓冲捕获模式波形



16.4.3. 16 位软件定时器模式(比较模式)

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

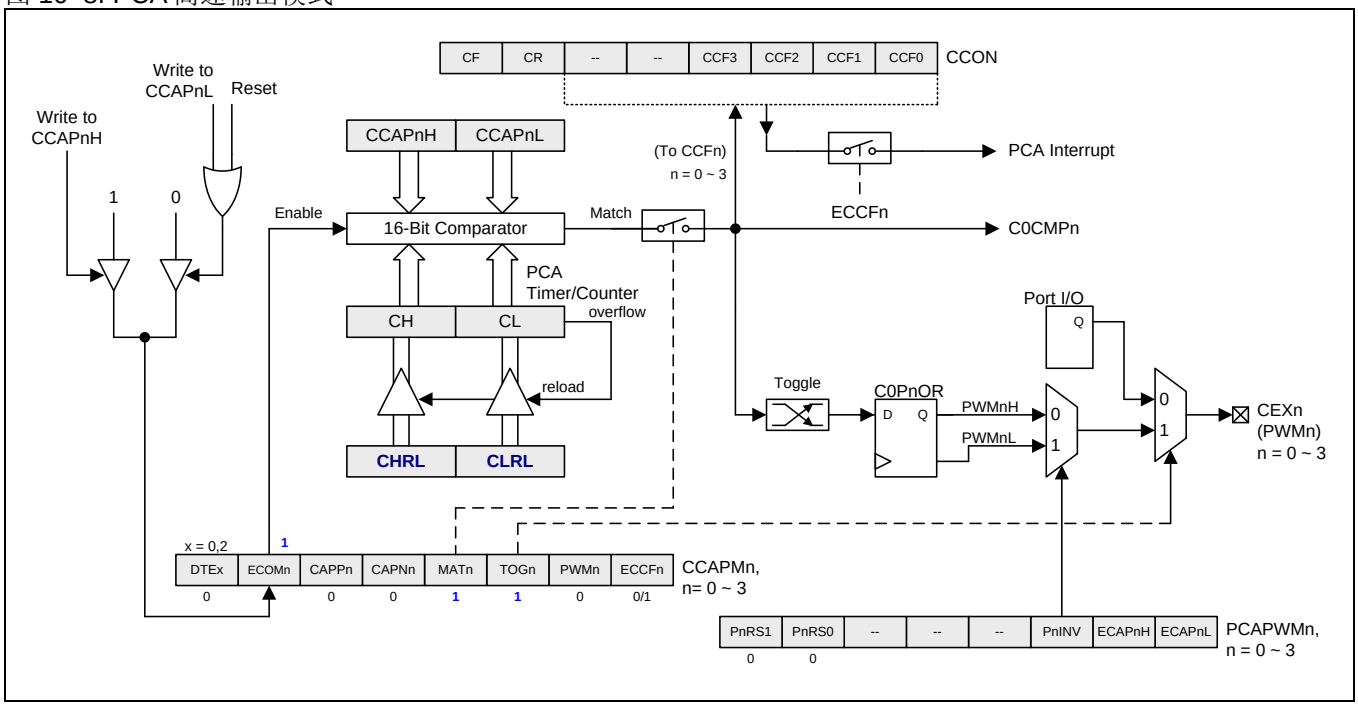
图 16–7. PCA 软件定时器模式



16.4.4. 高速输出模式(比较输出模式)

这种模式下，每当PCA计数器与模块捕获寄存器(CCAPnH和CCAPnL)值相等时，与PCA模块相关联的CEX输出就会触发。为激活这种模式，CCAPMn寄存器的TOG、MAT和ECOM位必须都置为1。

图 16–8. PCA 高速输出模式



16.4.5. 缓冲 8 位 PWM 模式

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

占空比取决于模块捕获寄存器 CCAPnL 与扩展的第 9 位 ECAPnL 的值。当 9 位数据{0,[CL]}值小于{ ECAPnL, [CCAPnL] }组成的 9 位数据时，输出低电平，相等或大于时输出高电平。

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

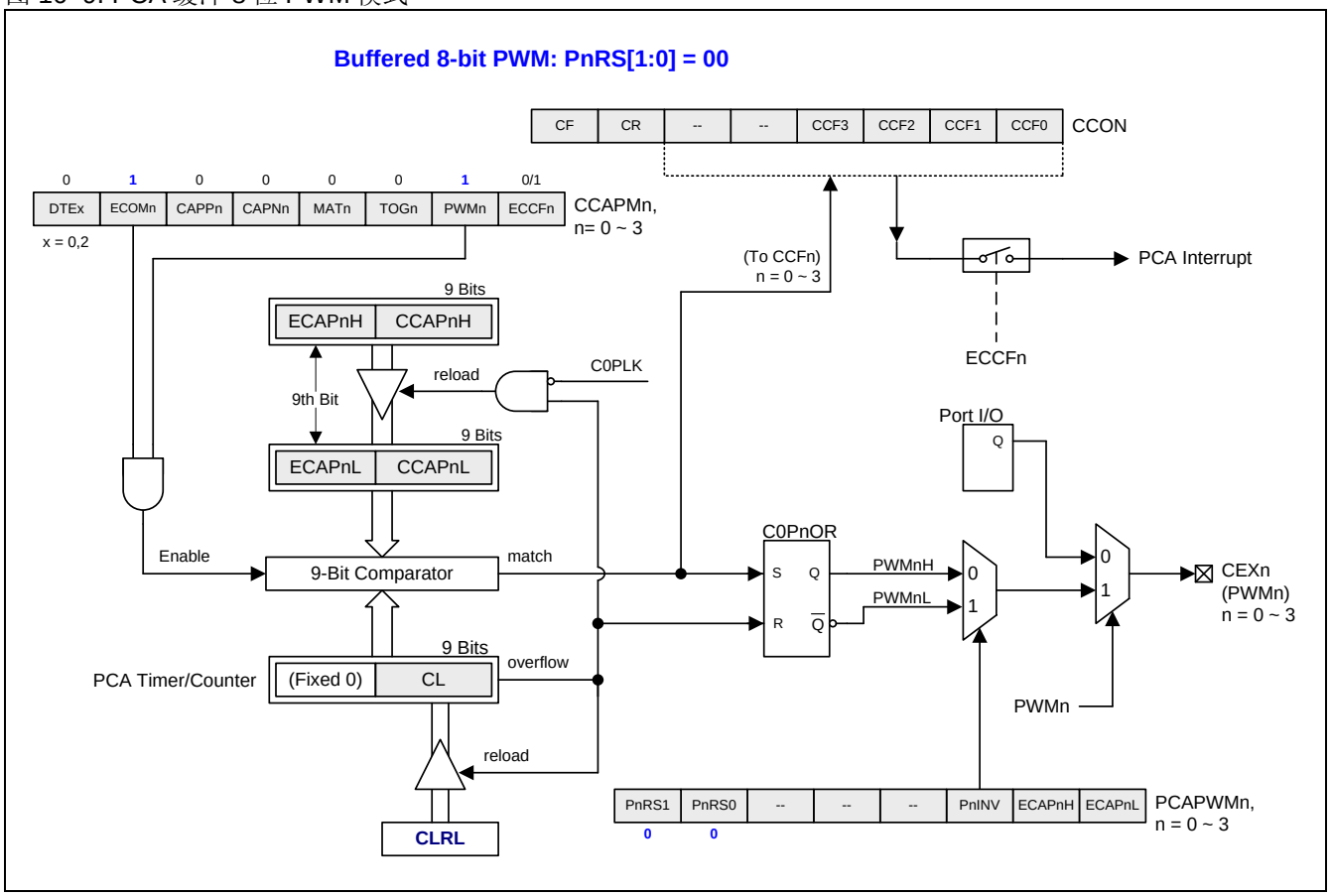
使用 9 位比较，输出的占空比可以真正实现从 0%到 100%可调。占空比计算公式如下：

占空比 = 1 – { ECAPnH, [CCAPnH] } / 256

这里， [CCAPnH]是 CCAPnH 寄存器的 8 位值，ECAPnH(PCAPWMn 寄存器的第 1 位)是 1 位值。所以，{ ECAPnH, [CCAPnH] } 组成了 9 位比较器用的 9 位值。

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

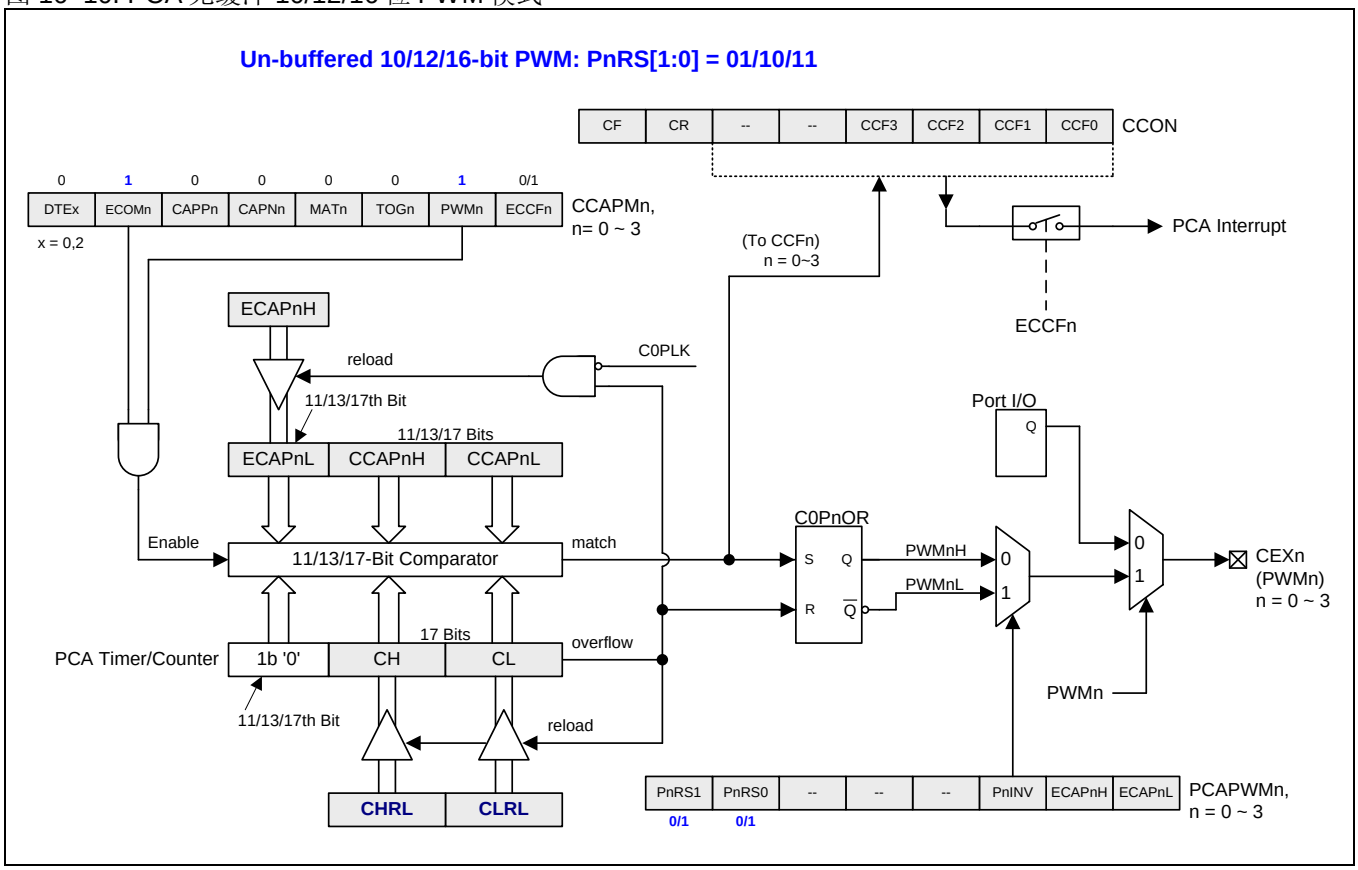
图 16–9. PCA 缓冲 8 位 PWM 模式



16.4.6. 无缓冲 10/12/16 位 PWM 模式

PCA 提供可变的 PWM 模式以增强控制能力。有额外无缓冲的 10/12/16 位 PWM 被分配给每一路及每一路 PWM 有不同的分辨率操作能力。

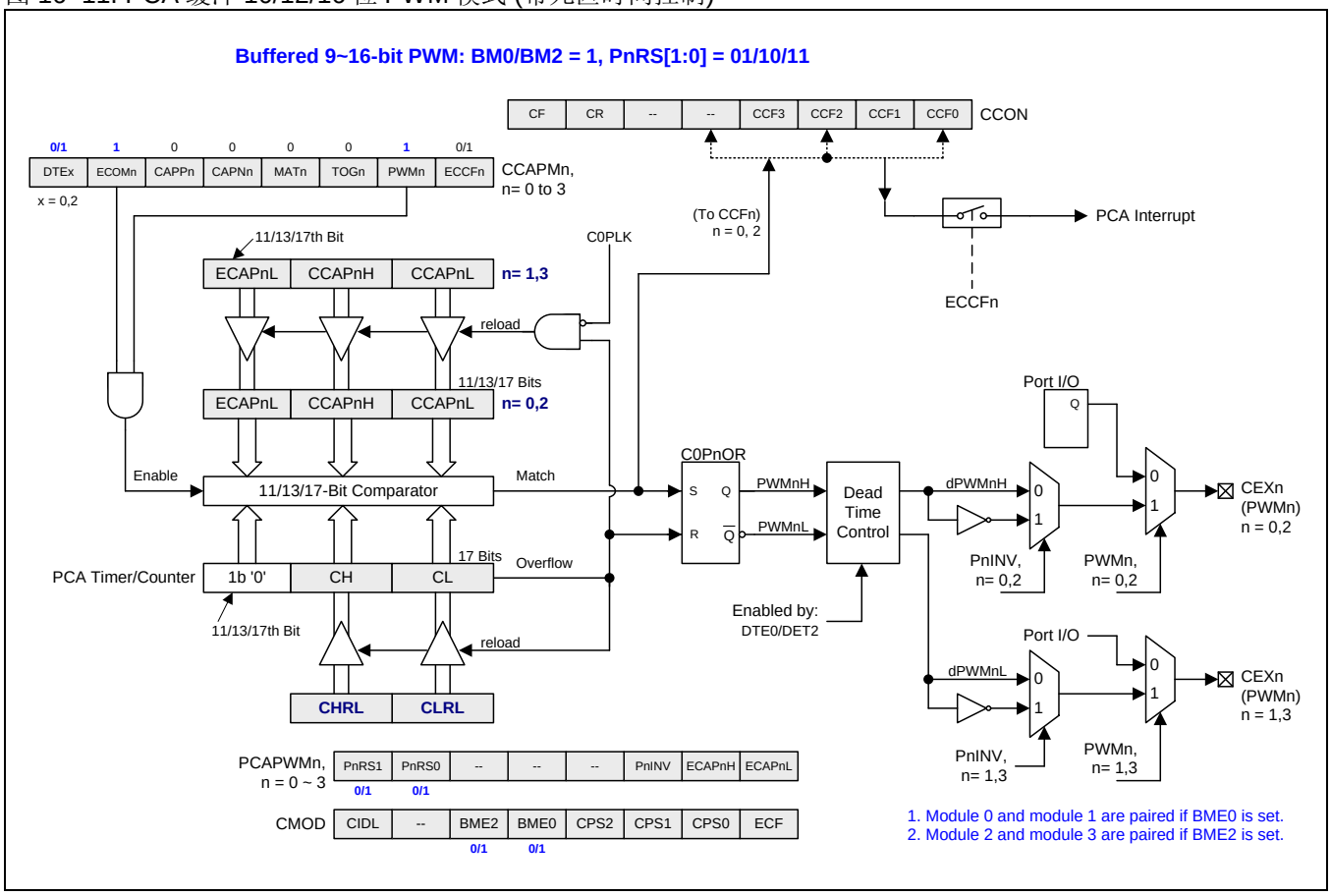
图 16–10. PCA 无缓冲 10/12/16 位 PWM 模式



16.4.7. 缓冲 10/12/16 位 PWM 模式

如果使用 10/12/16 位 PWM 模式，在将数据写入 CCAPnH 和 CCAPnL 时，将会导致意外的占空比，因为 8 位 CPU 每次只能写一个字节。要完成完全设置，需要两个写周期，当第一个字节被写入时，比较器将输出意料之外的占空比。如果应用程序在更改占空比时需要精确控制，则需要使用缓冲 PWM 模式。

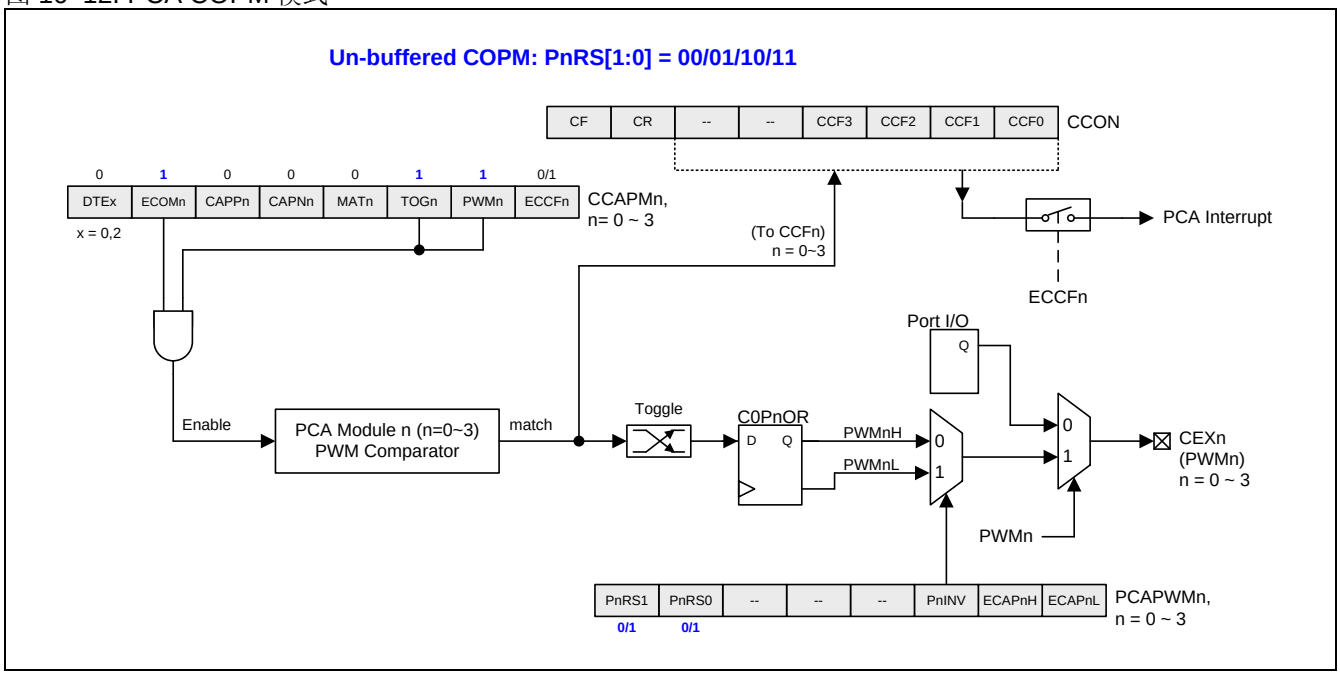
图 16–11. PCA 缓冲 10/12/16 位 PWM 模式 (带死区时间控制)



16.4.8. COPM 模式

PWM 匹配模式下的比较输出与高速输出模式相似，但它使用 PCA0 PWM 比较器而不是固定的 16 位比较器。它为应用程序提供了更多的灵活性。例如，如果 PCA0 比较器使用 8 位 PWM，可以比高速输出模式输出更高的频率。

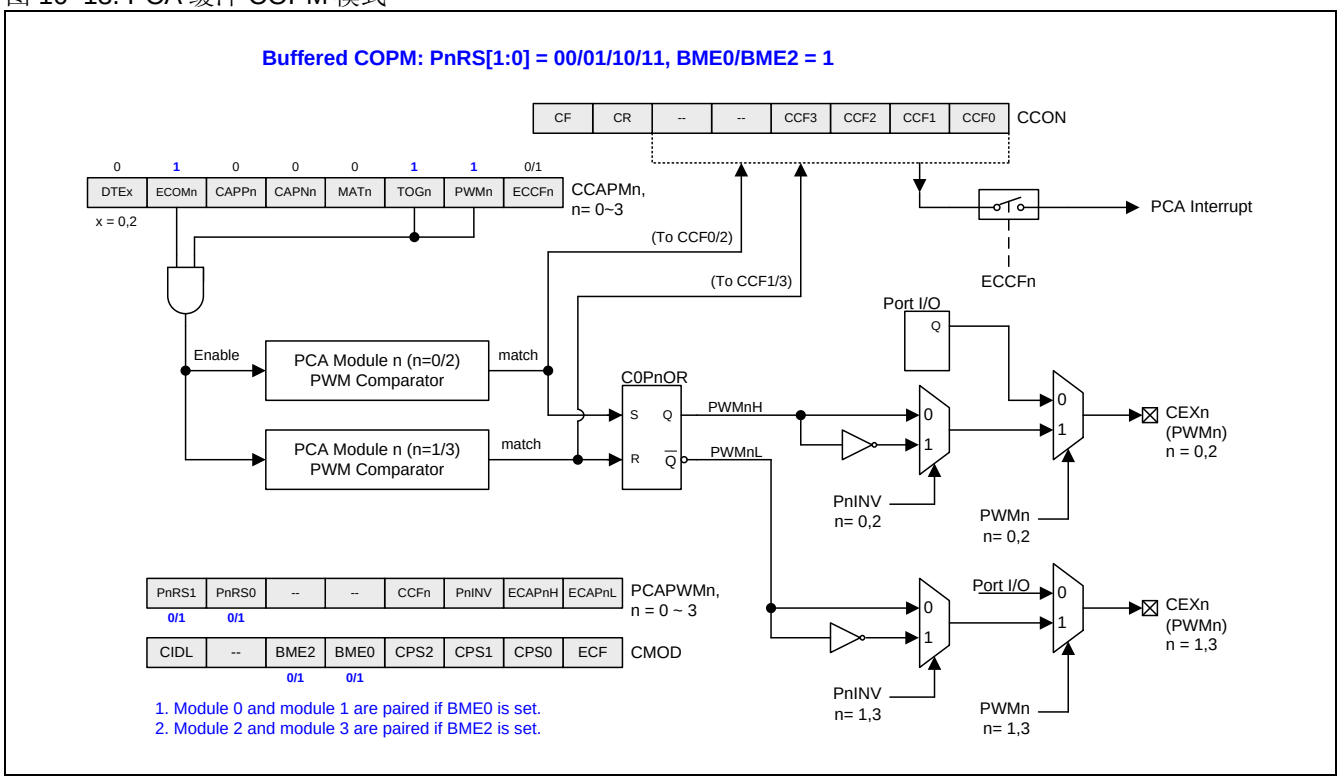
图 16–12. PCA COPM 模式



16.4.9. 缓冲 COPM 模式

如果应用程序需要对 PWM 信号进行任何相位控制，则需要设置 PCA 模块为缓冲 COPM 模式。一组 PCA 模块(n=0&1 / 2&3)可以对 PWM 信号的一个周期的两个边缘的时间延迟进行编程。这意味着你可以设置波形的开始和结束点。当 2 个或 3 个相关 PWM 信号可以设置彼此之间的相移时，这是很有用的。

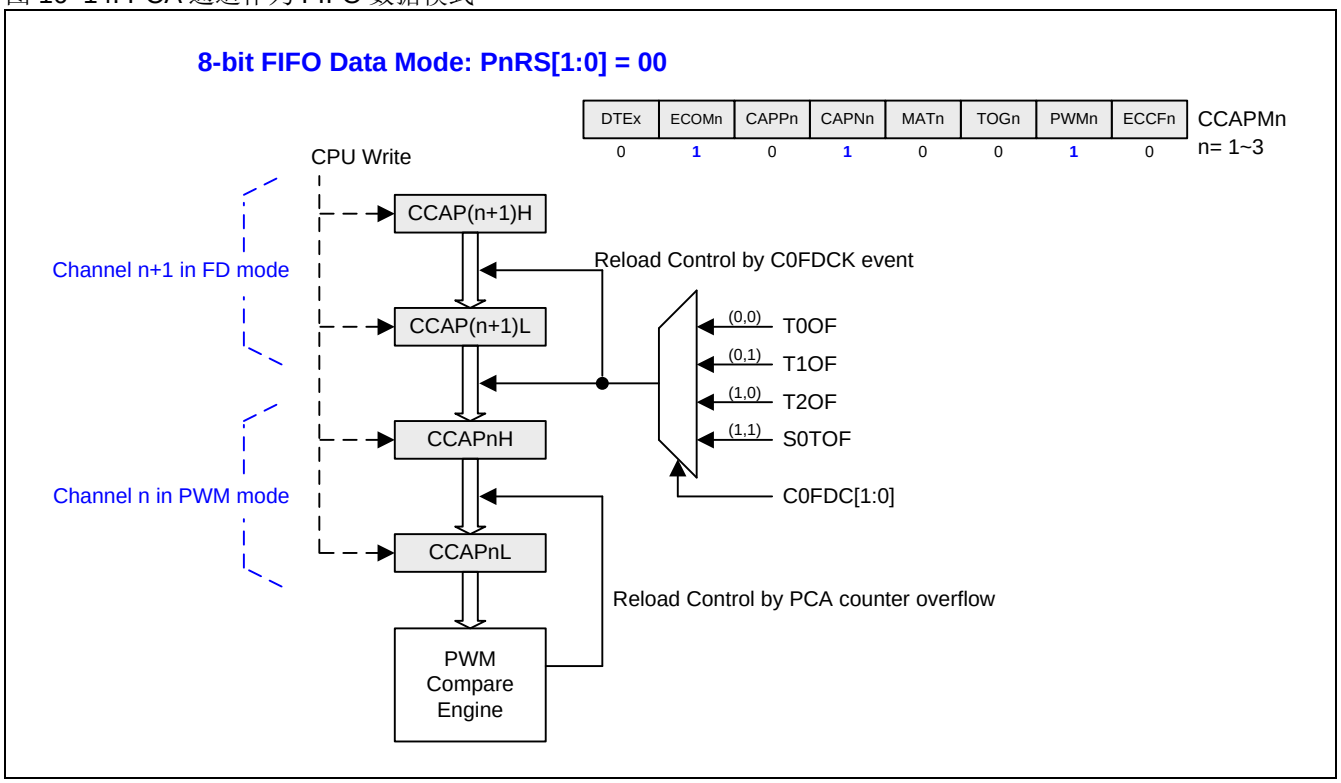
图 16–13. PCA 缓冲 COPM 模式



16.4.10. FIFO 数据模式

在这个模式用户能设置 CCAPnL、CCAPnH、CCAP(n+1)L 和 CCAP(n+1)H 作为一个缓冲链。在所有这些缓冲设置之后，通过 T0OF、T1OF、T3OF 或 S0TOF 顺序触发来改变占空比。这个功能使能，CPU 可以离开让它自己运行而获得更多时间做其它的操作。比如，功率变换器从轻载到重载开始升高电压，将更好用于设置比目标开启周期更大的占空比，然后降低占空比一步一步靠近目标占空比。在缓冲中设置所有占空比并且让它自己完成。

图 16–14. PCA 通道作为 FIFO 数据模式



通道 FIFO 数据模式由 C0FDCK 移动。
C0FDCK 源选择，更新 PCA FIFO 数据模式的时钟选择。

C0FDC1~0	C0FDCK
00	T0OF
01	T1OF
10	T2OF
11	S0TOF

AUXR9:辅助寄存器 9
SFR 页 = **仅 6 页**
SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	SIDPS0	T1G0	T0G1	C0FDC1	C0FDC0	0	0
W	**W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

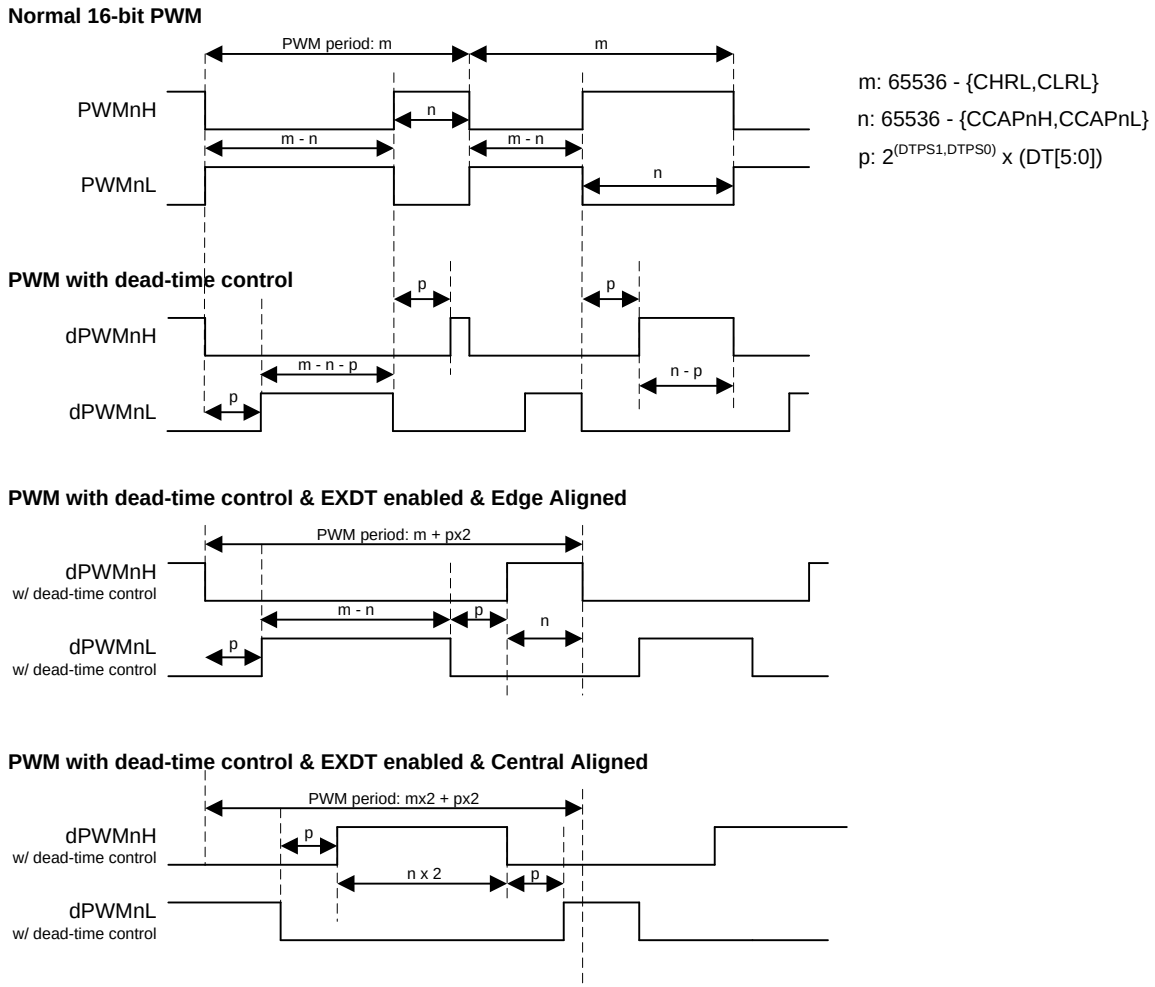
Bit 3~2：C0FDC1~0，C0FDCK 选择[1:0].

C0FDC1~0	C0FDCK
00	T0OF
01	T1OF
10	T2OF
11	S0TOF

16.4.11. 增强型 PWM 控制

PCA 提供可变的 PWM 模式增强 PWM 应用的控制能力。这些额外的 10/12/16 位 PWM 可以分配到每个通道且每个 PWM 通道可以同时运行在不同的分辨率和不同的相位延时。

图 16–15. PWM 带死区时间控制的波形



CCAPMn: PCA 模块比较/捕获寄存器，n=0~3

SFR 页 = **n=0~1 时仅 0 页 (n=2~3 则是所有页)**

SFR 地址 = 0xDA~0xDD 复位值 = 0000-0000

7	6	5	4	3	2	1	0
DTEn	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：DTEn，使能 PWMHn/PWMLn 输出对的死区时间控制。此位仅在 n= 0 和 2 有效且当 PWM 通道操作在缓冲模式死区时间功能激活。通道的缓冲模式由 CMOD 的 BME0 或 BME2 使能

0：PWMn 输出禁止死区时间控制。

1：PWMn 输出使能死区时间控制。

PDTCRA: PWM 死区控制寄存器-A

SFR 页 = **仅 1 页**

SFR 地址 = **0xBC** 复位值 = 0000-0000

7	6	5	4	3	2	1	0
DTPS1	**DTPS0**	**DT5**	**DT4**	**DT3**	**DT2**	**DT1**	**DT0**
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6：DTPS1~0，死区计数器的时钟预分频。

DTPS[1:0]	预分频选择
00	SYSCLK
01	SYSCLK/2
10	SYSCLK/4
11	SYSCLK/8

Bit 5~0：DT5~0，死区时间控制位。

DT[5:0]	死区时间
000000	禁止死区时间
000001	预分频器时钟 X 1
000010	预分频器时钟 X 2
000011	预分频器时钟 X 3
…..	……
111110	预分频器时钟 X 62
111111	预分频器时钟 X 63

PWMCR: PWM 控制寄存器

SFR 页 = **仅 0 页**

SFR 地址 = **0xBC** 复位值 = 0000-0000

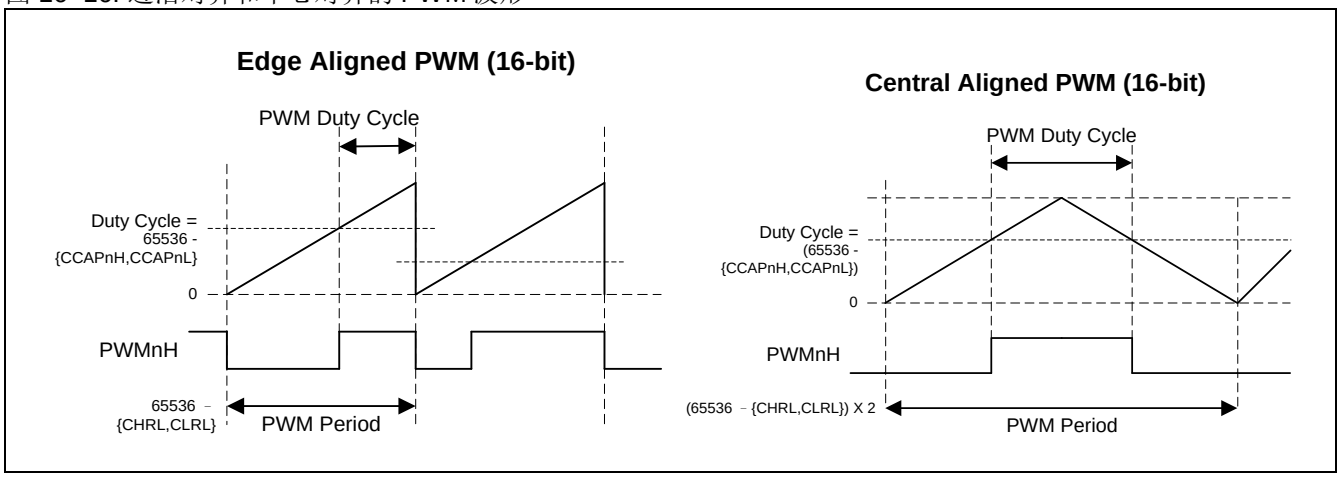
7	6	5	4	3	2	1	0
PCAE	**EXDT**	**PBKM**	**PBKE1.1**	**PBKE1.0**	**PBKE0.2**	**PBKE0.1**	**PBKE0.0**
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：PCAE，PWM 中心对齐使能。PCAE 控制使能的 PWM 通道为中心对齐调制包括 PWM 的缓冲模式或非缓冲模式。在这个 PWM 模式下，PWM 频率是半边沿对齐模式。此功能仅在 PWMO0~3 激活。

0：设置 PWM 功能为边沿对齐调制。

1：使能 PWM 功能为中心对齐调制。仅支持 CHRL 和 CLRL 设定的 8/10/12/16 位分辨率。

图 16–16. 边沿对齐和中心对齐的 PWM 波形



Bit 6：EXDT，PWM 时间的扩展死区时间。此功能使能将使非 PWM 通道功能改变为 PWM 通道。比如捕获模式，软件定时器模式和高速输出模式。

0：禁止 M + **2P**。

1：在使能的 PWM 通道使能 M + **2P**。

Bit 5：PBKM，PWM 中止模式选择。

0：锁存模式。

1：逐周期模式。

图 16–17. 中止控制的锁存模式波形

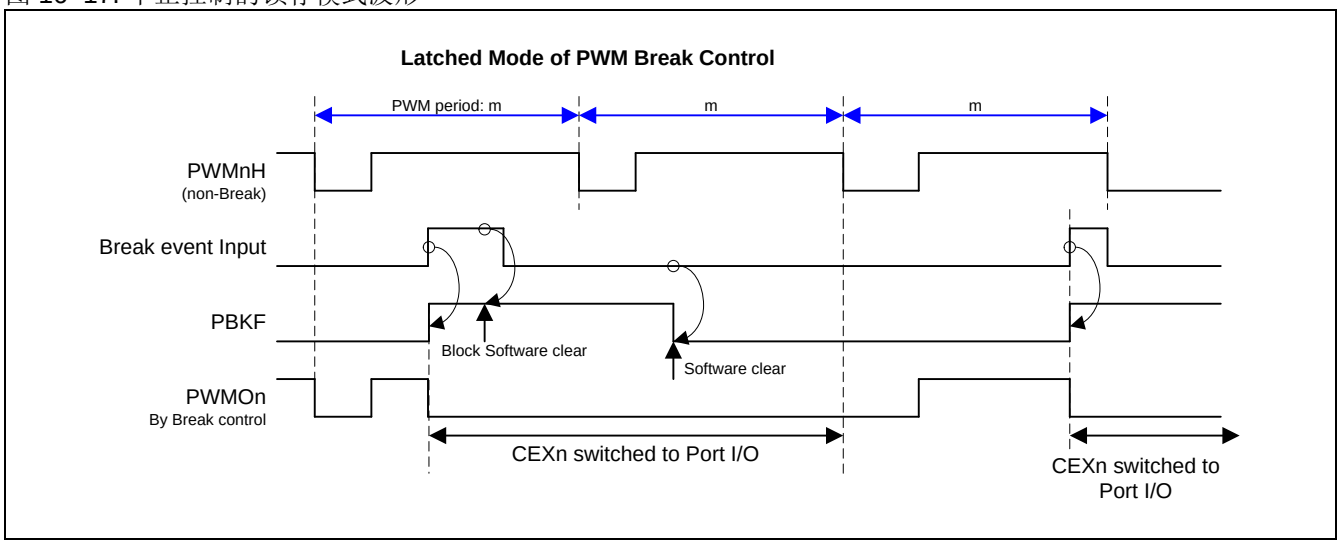
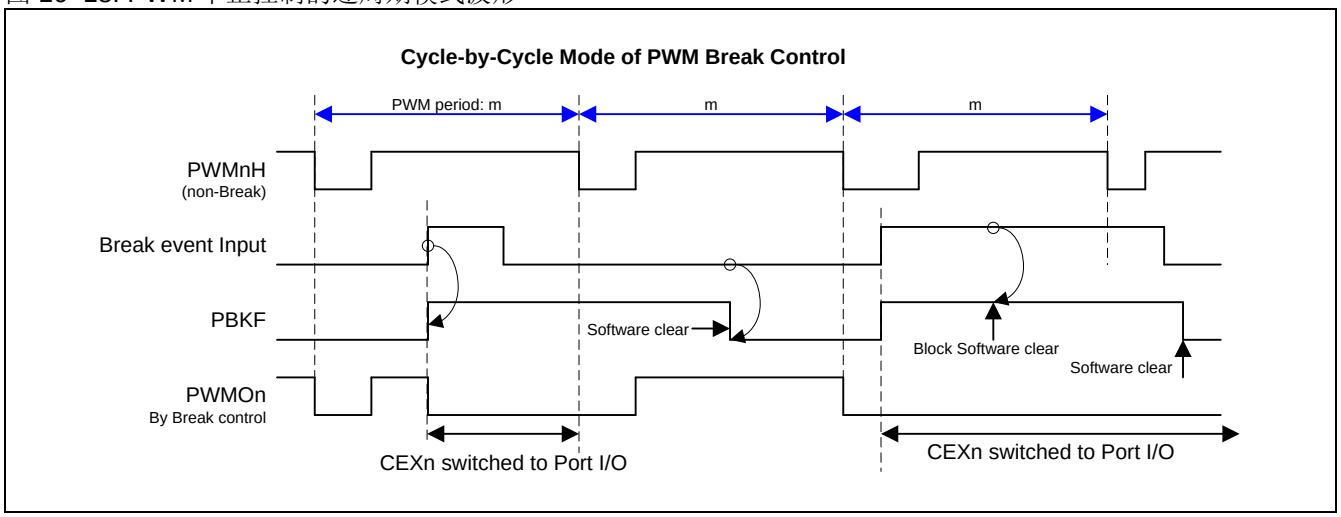


图 16–18. PWM 中止控制的逐周期模式波形



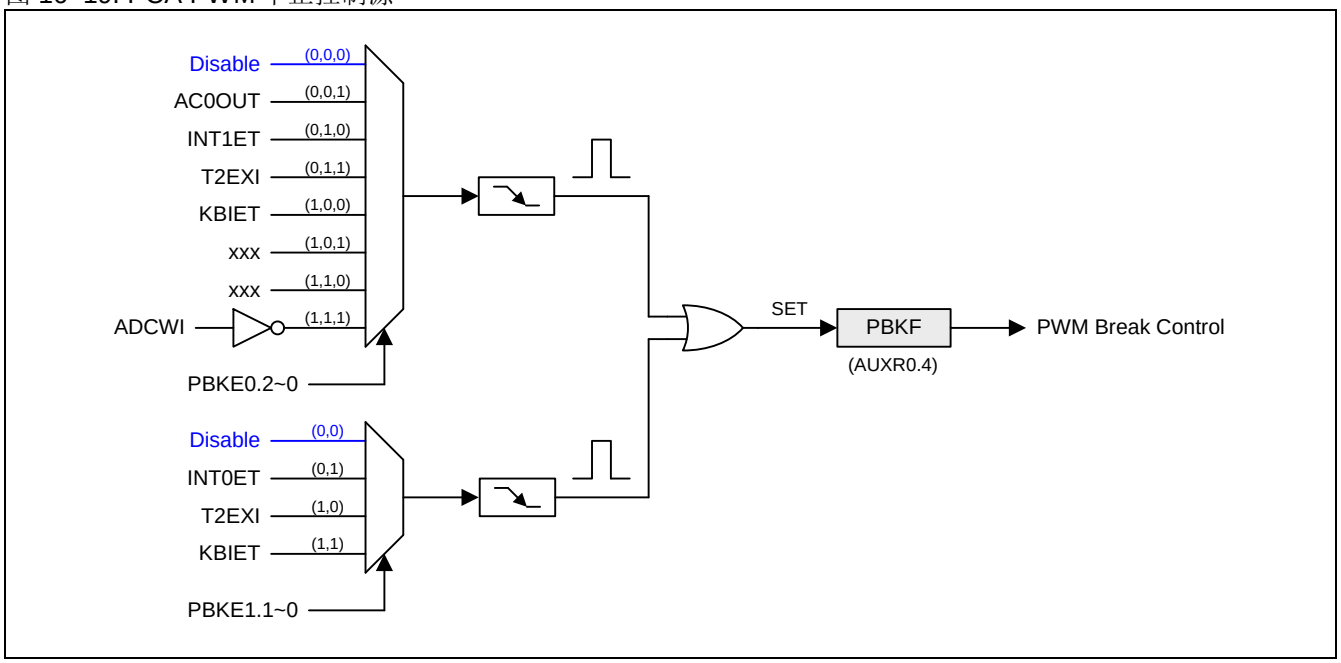
Bit 4~3：PBKE1.1~0，PWM 中止使能 1 选择。此功能仅在 CEXn 输出模式有效(n=0~3)。

PBKE1[1:0]	PWM 中止源
0 0	禁止 PWM 中止源 1
0 1	INT0ET
1 0	T2EXI
1 1	KBIET，KBI 匹配激活

Bit 2~0：PBKE0.2~0，PWM 中止使能 0 选择。此功能仅在 CEXn 输出模式有效(n=0~3)。

PBKE0[2:0]	PWM 中止源
0 0 0	禁止 PWM 中止源 0
0 0 1	AC0OUT
0 1 0	INT1ET，nINT1 激活
0 1 1	T2EXI
1 0 0	KBIET，KBI 匹配激活
1 0 1	保留
1 1 0	保留
1 1 1	ADCWI 激活

图 16–19. PCA PWM 中止控制源



AUXR0:辅助寄存器 0

SFR 页 = 0~F

SFR 地址 = 0xA1 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P45OC1	P45OC0	0	PBKF	0	0	INT1H	INT0H
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 4：PBKF，PWM 中止事件标志。此位由 PWM 中止源使能设置。如果此位设置，则使能的 PWM 通道 0~3 将被锁住并且输出引脚保持最初的 GPIO 状态。

0：没有 PWM 中止事件出现。仅由软件清零。

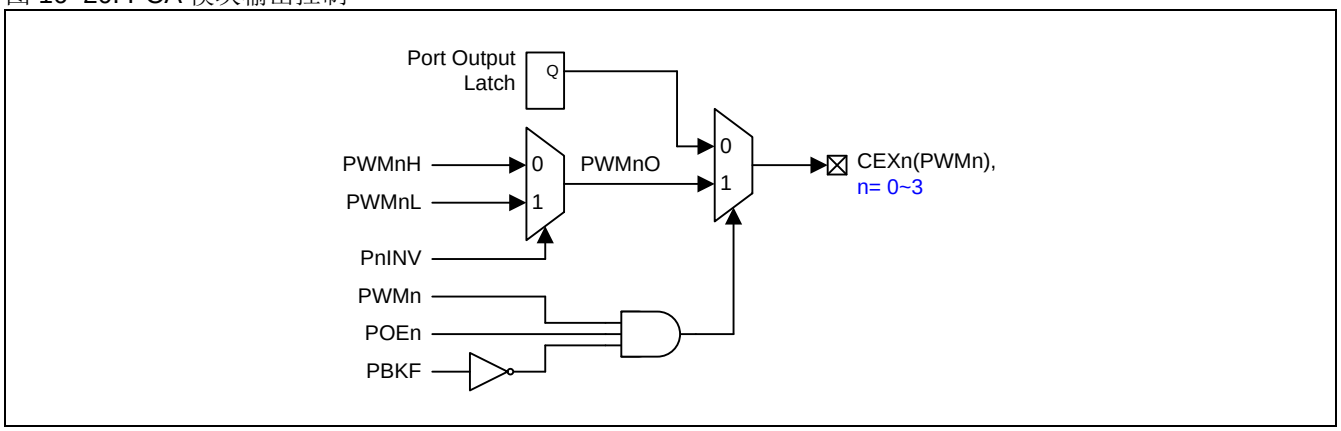
1：PWM 中止事件出现或软件触发一个 PWM 中止。

16.4.12. PCA 模块输出控制

PCA 模块具有多种输出控制模式，可以选择用于不同的应用。CEXn (n=0~3)可编程为一般 I/O 端口或 PCA 模块(PWM) 0~3 的输出。当 PWM 被分配到 CEXn 时，PnINV 可以在正常或反向之间切换 PWM 信号。POEn 可以用来启用或禁用端口引脚输出 PWM 信号。

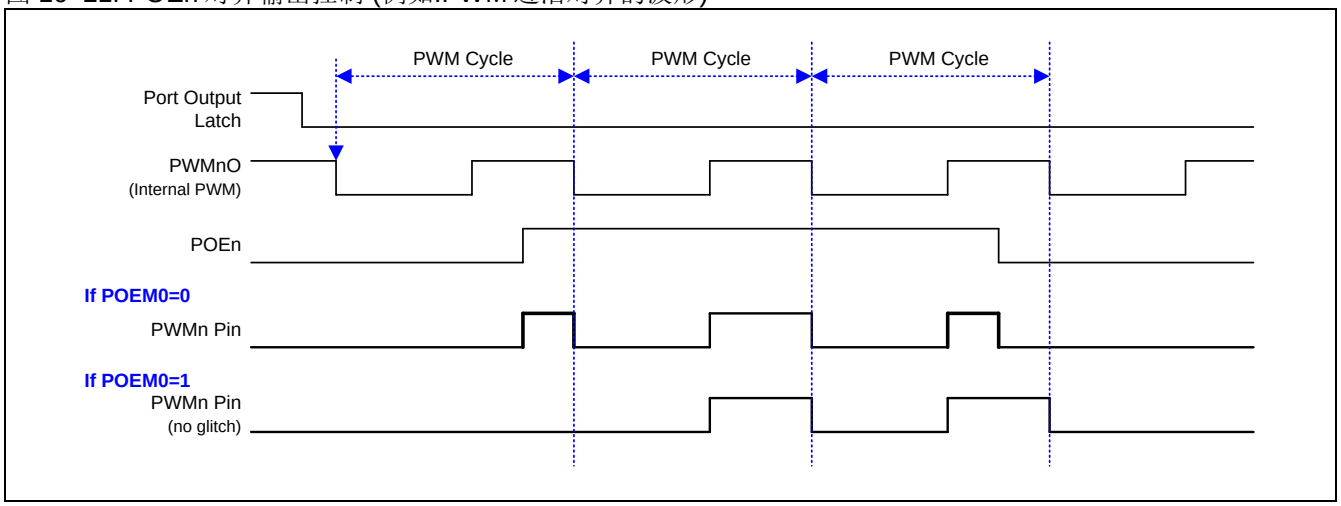
CEXn (n=0~3)可以使用 PBKF，PWM 中止标志，中止 PWM 输出。如果此位设置，则使能的 PWM 通道 0~3 将被锁住并且输出引脚保持最初的 GPIO 状态。

图 16–20. PCA 模块输出控制



MG82F6B08 / 6B001/ 6B104 的 POEM0 控制 POEn 输出时序与 PWM 周期对齐。这个对齐功能的配置和波形如图 16–21 所示。

图 16–21. POEn 对齐输出控制 (例如.PWM 边沿对齐的波形)



PAOE: PWM 额外输出使能寄存器

SFR 页 = 0~F

SFR 地址 = 0xF1 复位值 = 1001-1001

7	6	5	4	3	2	1	0
POE3	0	0	POE2	POE1	0	0	POE0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7：POE3，PCA PWM3 主通道(PWM3O)输出控制。
0：禁止 PWM3O 在端口引脚输出。
1：使能 PWM3O 在端口引脚输出。**默认是使能**。

Bit 4：POE2，PCA PWM2 主通道(PWM2O)输出控制。
0：禁止 PWM2O 在端口引脚输出。
1：使能 PWM2O 在端口引脚输出。**默认是使能**。

Bit 3：POE1，PCA PWM1 主通道(PWM1O)输出控制。
0：禁止 PWM1O 在端口引脚输出。
1：使能 PWM1O 在端口引脚输出。**默认是使能**。

Bit 0：POE0，PCA PWM0 主通道(PWM0O)输出控制。
0：禁止 PWM0O 在端口引脚输出。
1：使能 PWM0O 在端口引脚输出。**默认是使能**。

AUXR2: Auxiliary Register 2

SFR Page = 0~F

SFR Address = 0xA3 RESET = 0000-0000

7	6	5	4	3	2	1	0
STAF	STOF	0	C0PLK	T1X12	T0X12	T1CKOE	T0CKOE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 4: C0PLK, PCA0 缓冲 PWM/COPM 更新控制
0: 缓冲 PWM/COPM 在 PCA 基础定时器溢出时自动更新。
1: 禁止缓冲 PWM/COPM 自动更新。

AUXR7:辅助寄存器 7

SFR 页 = **仅 4 页**

SFR 地址 = 0xA4 复位值 = 1100-0000

7	6	5	4	3	2	1	0
1	1	C0CKOE	SPI0M0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 5：C0CKOE，PCA 时钟输出使能。
0：禁止 PCA 时钟输出。
1：使能 PCA 基准定时器溢出率二分频的时钟输出。

AUXR11:辅助寄存器 11

SFR 页 = **仅 8 页**

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P30AM	P33AM	0	0	0	POEM0	C0M0	C0OFS
R/W	R/W	R/W	R/W	W	R/W	R/W	R/W

Bit 2：POEM0, PCA0 POEn 控制 0。
0：POEn 功能在 CPU 写入后立即生效
1：POEn 功能与 PWM 周期对齐

AUXR5:辅助寄存器 5

SFR 页 = **仅 2 页**

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	C0IC2S0	0	0	C0PS1	C0PS0	ECIPS0	C0COPS
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 6：C0IC2S0, PCA0 输入通道 2 输入端口引脚选择。

C1IC2S0	CEX2 input
0	CEX2 Port Pin
1	T2EXI

Bit 3~2：C0PS[1:0], PCA0 端口引脚选择 1 & 0。

C0PS1~0	CEX0	CEX1	CEX2	CEX3
0 0	P3.0	P3.3	P3.1	P4.6
0 1	P4.4	P3.3	P4.5	P4.6
1 0	P4.4	P3.3	P3.0	P4.6
1 1	P4.4	P3.3	P1.0	P4.6

Bit 1：ECIPS0, PCA0 ECI 端口引脚选择 0。

ECIPS0	ECI
0	P4.4
1	P1.0

Bit 0：C0COPS, PCA0 时钟 (C0CKO) 端口引脚选择。

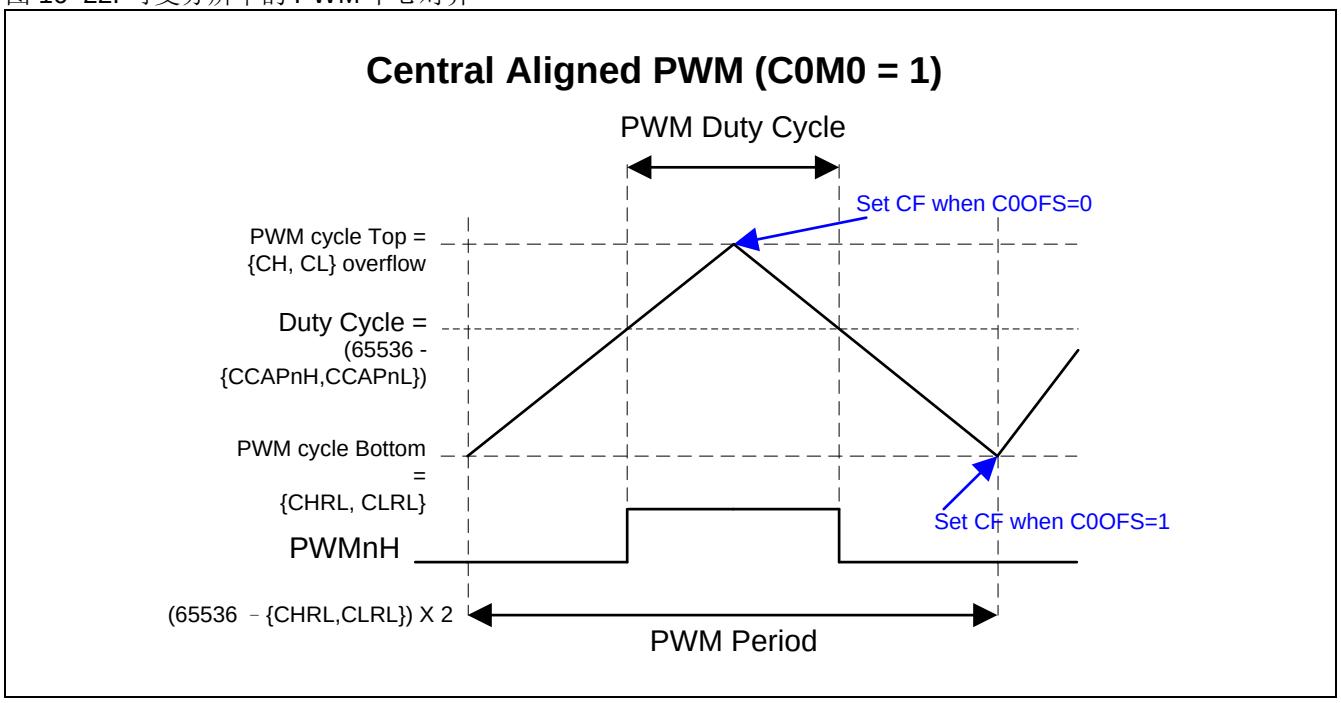
C0COPS	C0CKO
0	P4.7
1	P3.3

16.4.13. 可变分辨率的 PWM 中心对齐

在章节“增强型 PWM 控制”中，定义 PWM 中心对齐仅支持 8/10/12/16 位分辨率。并且在那个模式，所有 PCA 功能，如捕获或比较在非 PWM 模块中是有效的。

如果需要应用可变分辨率的 PWM 中心对齐，要同时置位 PCAE 及 C0M0 使能此功能。当可变分辨率的中心对齐模式使能，PCA 在其它非 PWM 模块中能支持所有比较模式。请注意当 C0M0 = 1 使用 PWM 中心对齐，基准定时器需要使用 16 位 0xFFFF 减去{CCAPnH, CCAPnL}的值避免预期外的错误。

图 16–22. 可变分辨率的 PWM 中心对齐



AUXR11:辅助寄存器 11

SFR 页 = **仅 8 页**

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P30AM	P33AM	0	0	0	POEM0	C0M0	C0OFS
R/W	**R/W**	**R/W**	**R/W**	**W**	**R/W**	**R/W**	**R/W**

Bit 1：C0M0，PCA 模式控制 0。
0：PWM 中心对齐不支持可变分辨率。
1：使能 PCA 支持可变分辨率的 PWM 中心对齐。使能此功能，PCAE (PWMCR.7)也需要置位。

Bit 0：C0OFS，当 C0M0 使能 PCA 溢出标志选择。
0：PWM 中心对齐周期的顶部 CF 置位。
1：PWM 中心对齐周期的低部 CF 置位。

17. 串口 0 (UART0)

MG82F6B08 / 6B001/ 6B104 支持一个全双工的串口，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。串口的接收和发送寄存器都通过特殊寄存器 S0BUF 来访问。写到 S0BUF 加载到传送寄存器，当从 S0BUF 读时是一个物理上独立分离的接收寄存器。

17.1. 串口 0 模式选择

串口可以工作在 **5** 种标准模式和 **8** 个增强模式：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。UART0 的模式 4 支持 SPI 主机工作，速率设置跟模式 0 一样。
增强模式请参考章节 17.11 串口增强型功能。

表 17–1. 串口 0 模式选择

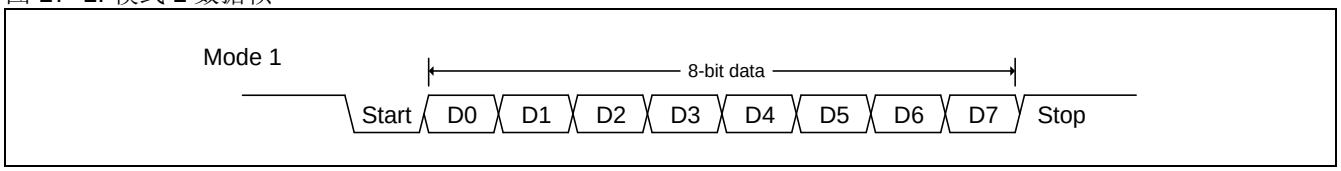
SM30,SM00, SM10	S0RCK	S0TCK	MODE	功能	波特率时基	注意
000	0	0	0	移位寄存器	SYSCLK/12 or SYSCLK/4 (URM0X3=1)	
001	0	0	1	8 位 UART	定时器 1 或定时器 2 溢出	当 SMOD1 & SMOD2 =1，计数器不能为 满额-1 或 满额-2 (如. 254, 255, 65534, 65535)
010	0	0	2	9 位 UART	SYSCLK/64, /32, /16, or /8	
011	0	0	3	9 位 UART	定时器 1 或定时器 2 溢出	当 SMOD1 & SMOD2 =1，计数器不能为 满额-1 或 满额-2 (如. 254, 255, 65534, 65535)
100	0	0	4	SPI 主机	SYSCLK/12 or SYSCLK/4 (URM0X3=1)	
000	0	1	增强	shift register	S0BRG 溢出	S0BRT 不能为 255
001	0/1	0/1	增强	8 位 UART	TX 或 RX 可选 S0BRG 溢出	SMOD1 & SMOD2 不能同时为 1
010	0	1	增强	9 位 UART	TX: S0BRG 溢出 RX: SYSCLK/64, /32 or /16	SMOD1 & SMOD2 不能同时为 1
010	1	0	增强	9 位 UART	TX: SYSCLK/64, /32 or /16 RX: S0BRG 溢出	SMOD1 & SMOD2 不能同时为 1
010	1	1	增强	纯定时器	仅定时器功能	
011	0/1	0/1	增强	9 位 UART	TX 或 RX 可选 S0BRG 溢出	SMOD1 & SMOD2 不能同时为 1
100	0	1	增强	SPI 主机	S0BRG 溢出	S0BRT 不能为 255
101	1	1	增强	LIN 总线	S0BRG 溢出和自动波特率	SMOD1 & SMOD2 不能同时为 1
其它				保留		

注: Mode0～4 使用 S0RCK 和 S0TCK 的默认值，要重置为增强模式，请参考 17.11 串口增强型功能的详细描述。

模式 0：8 位数据(低位先出)通过 RXD0 传送和接收。TXD0 总是作为输出移位时钟。波特率可通过 S0CFG 寄存器的 URM0X3 位选择为系统时钟频率的 1/12 或 1/4。**MG82F6B08 / 6B001/ 6B104** 串口模式 0 的时钟极性也可以软件选择。在串行数据移入或移出之前它由 P3.1 的状态决定。图 17–4 和图 17–5 所示模式 0 的时钟极性波形。

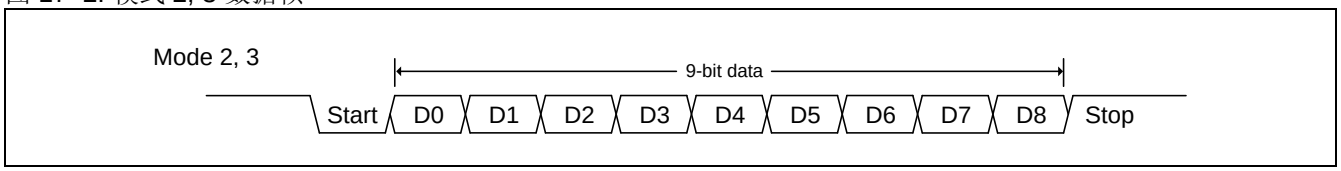
模式 1：10 位通过 TXD0 传送或通过 RXD0 接收，数据帧包括一个起始位(0)，8 个数据位(低位优先)，和一个停止位(1) (如图 17–1 所示)。在接收时，停止位进入到专用寄存器(S0CON)的 RB80。波特率是可变的。

图 17–1. 模式 1 数据帧



模式 2：11 位通过 TXD0 传送或通过 RXD0 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) (如图 17–2 所示)。在传送时，第 9 个数据位(TB80 在 S0CON 寄存器)可以分配为 0 或者 1。例如，奇偶检验位(P，在 PSW 寄存器)可以移到 TB80 中。在接收时，第九个数据位到 S0CON 寄存器中的 RB80，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。

图 17–2. 模式 2, 3 数据帧



模式 3：除了波特率是可变之外，模式 3 与模式 2 一样。

在四种模式中，使用 S0BUF 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 RI0=0 且 REN0=1 时启动接收。在其它模式，在 REN0=1 时，收到起始位时启动接收。

除了标准操作外，UART0 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。

17.2. 串口 0 模式 0

串行数据通过 RXD0 读入和输出。TXD0 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 S0CFG 寄存器中的 URM0X3 选择为系统时钟的 1/12 或 1/4。

串口模式 0 的简化功能框图如图 17–3 所示。

使用 S0BUF 作为一个目的寄存器可通过任何指令来启动传输。“写到 S0BUF”信号触发 UART0 引擎开始发送。S0BUF 里面的数据在 TXD0(P3.1)脚的每一个上升沿移出到 RXD0(P3.0)脚。八个上升沿移位时钟过后，硬件置 TI0 为 1 标志发送完成且中断向量可以由 BTI 和 UTIE 切换到系统标志中断。模式 0 发送时序见图 17–4。

当 REN0=1 和 RI0=0 时接收启动。在下一个指令周期，RX0 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。

接收是由直接 RX 时钟到 TXD0 引脚的额外输出功能的移位时钟来实现的。当接收激活时，在移位时钟的下降沿采样 RXD0(P3.0)脚并移到寄存中。八个下降沿移位时钟过后，硬件置 RI0 为 1 标志接收完成。模式 0 接收时序见图 17–5。

图 17–3. 串口 0 模式 0

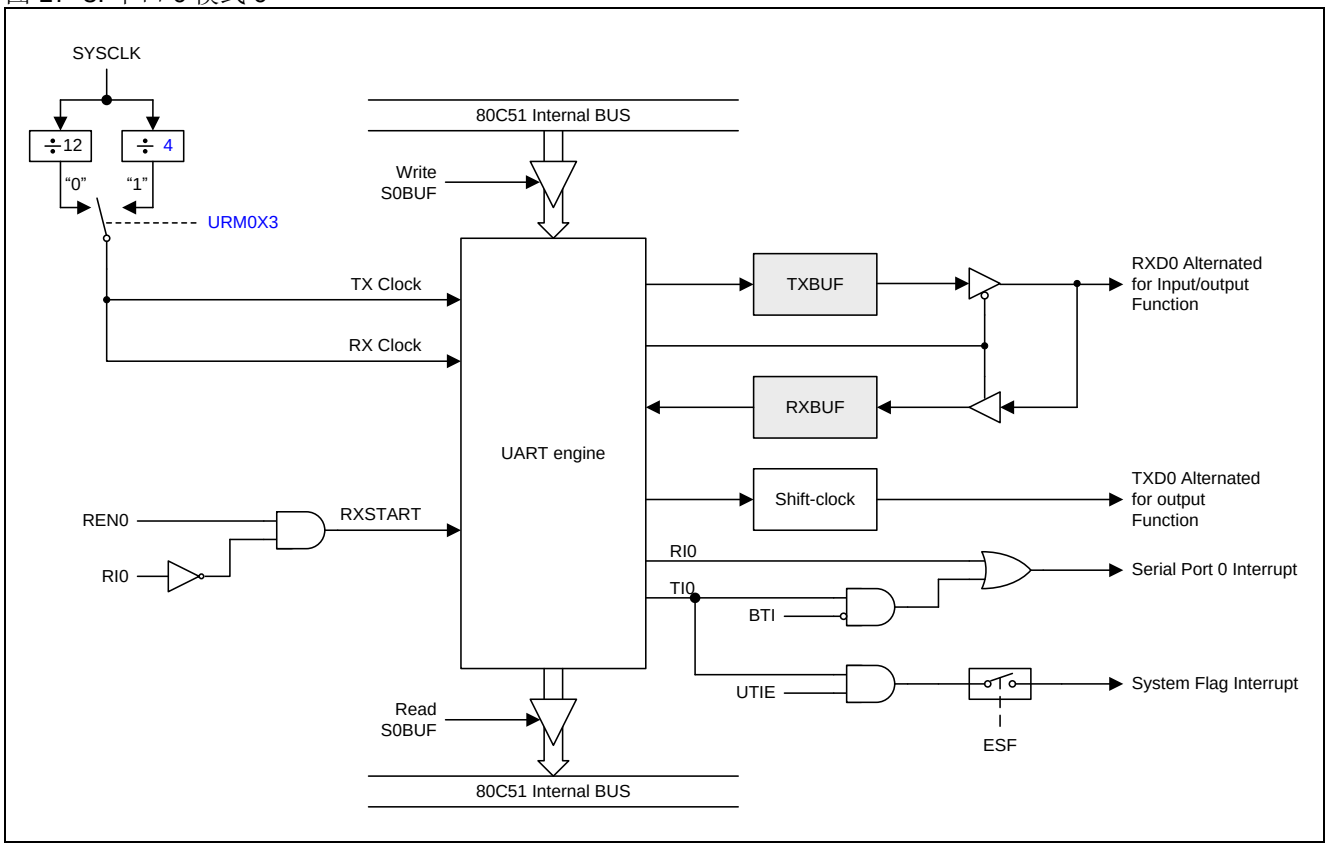


图 17–4. 模式 0 发送波形

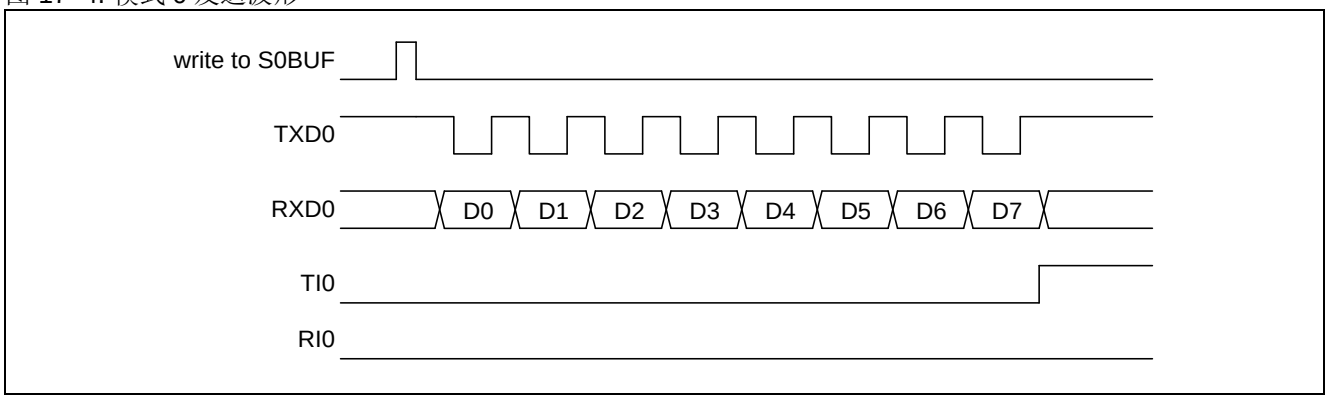
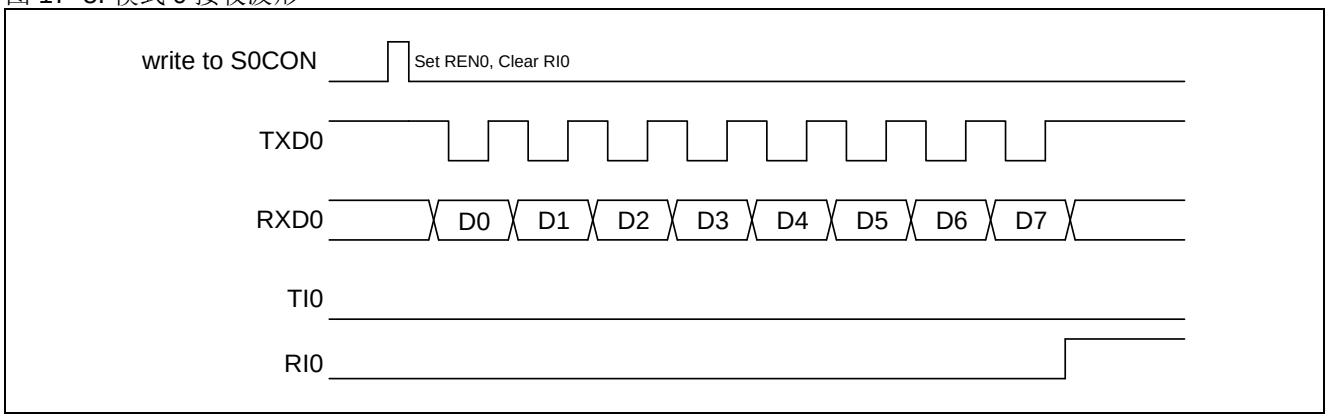


图 17–5. 模式 0 接收波形



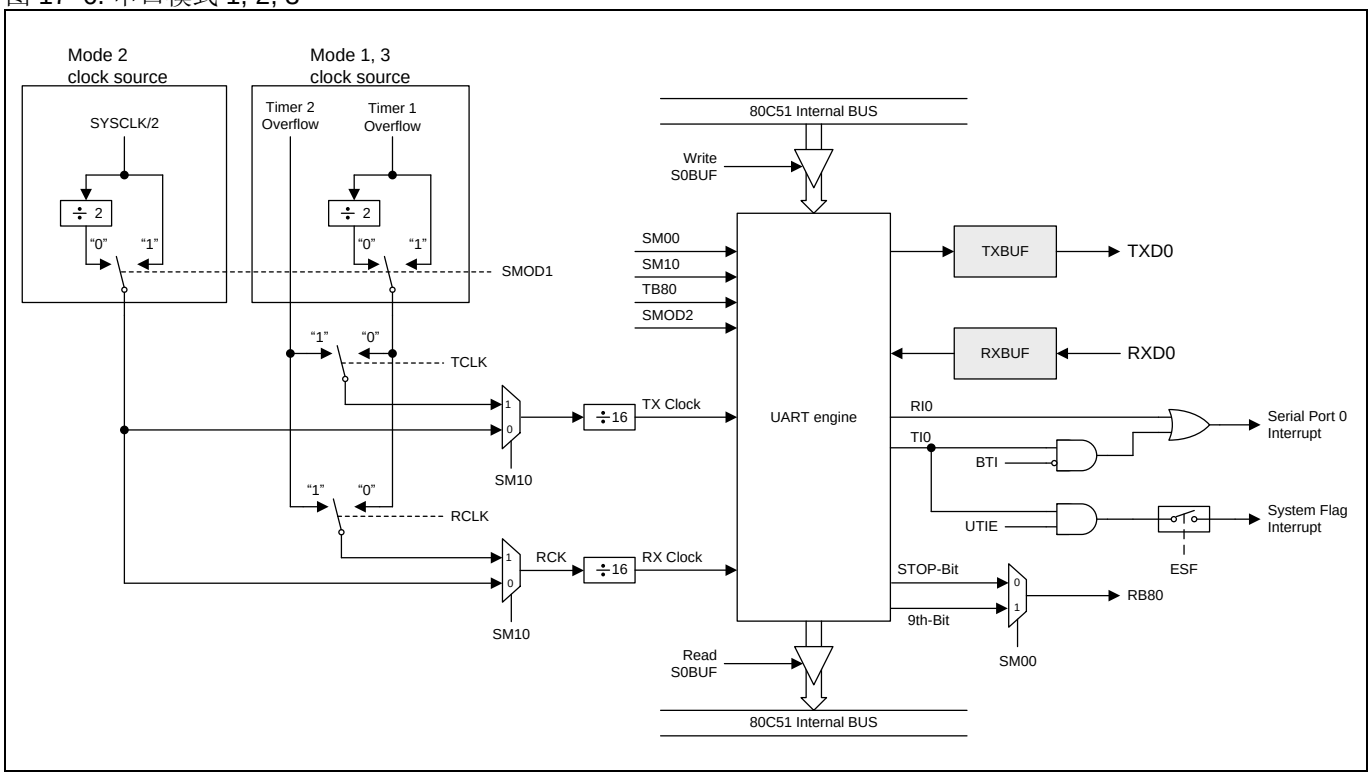
17.3. 串口 0 模式 1

通过 TXD0 发送 10 位数据或通过 RXD0 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 S0CON 的 RB80，波特率由定时器 1 或定时器 2 的溢出速率来决定。模式 1 数据帧时序如图 17–1 所示并且模式 1 的简化功能框图如图 17–6 所示。

使用 S0BUF 作为目的寄存器的任何指令来启动传输。“写到 S0BUF”的信号请求 UART0 引擎开始发送，当收到一个发送请求后，UART0 将在 TX 时钟的上升沿开始发送。S0BUF 中的数据从 TXD0 引脚串行输出，数据帧如图 17–1 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 TI0 表示发送结束，并且它的中断向量可以由 BTI 和 UTIE 切换到系统标志中断。

当串口 0 控制器在 RCK 采样时钟下检测到在 RXD0 有负跳变的起始位时接收开始。在 RXD0 引脚上的数据将被串行口 0 的位侦测器采样。当收到停止位后，硬件置位 RI0 表示接收结束并把停止位加载到 S0CON 寄存器的 RB80。

图 17–6. 串口模式 1, 2, 3



17.4. 串口 0 模式 2 和模式 3

通过 TXD0 传送 11 位或通过 RXD0 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(TB80)可分配为 0 或 1。在接收时，数据的第 9 位将进入到 S0CON 的 RB80。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。

模式 2 和 3 数据帧如图 17–2 所示，模式 2 和模式 3 的串行口功能框图如图 17–5 所示。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。

“写到 S0BUF”的信号请求 UART0 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，UART0 将在 TX 时钟的上升沿开始发送。S0BUF 中的数据从 TXD0 引脚串行输出，数据帧如图 17–2 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 TI0 表示发送结束,并且它的中断向量可以由 BTI 和 UTIE 切换到系统标志中断。

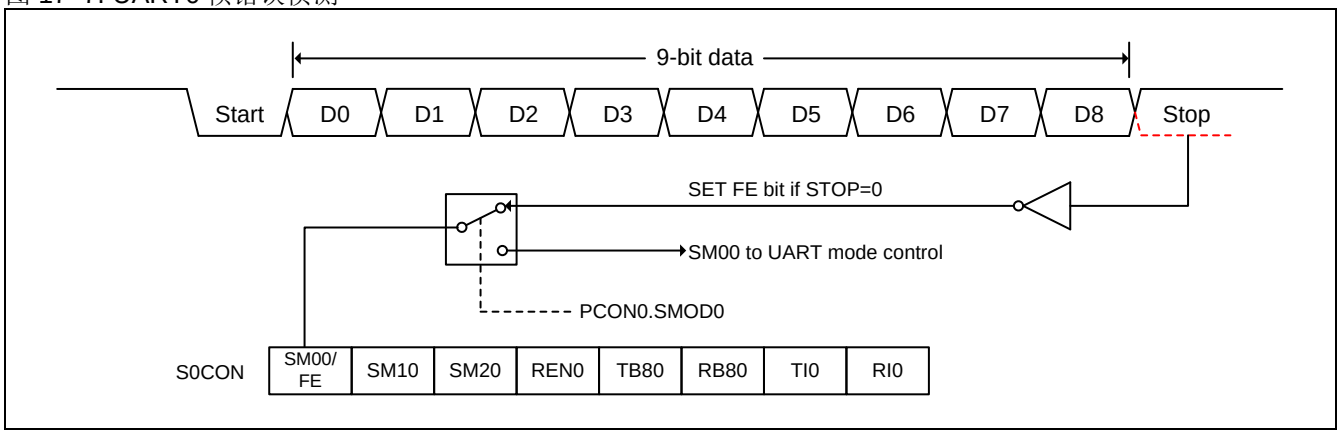
当串口 0 控制器在 RCK 采样时钟下检测到在 RXD0 有负跳变的起始位时接收开始。在 RXD0 引脚上的数据将被串口 0 的位侦测器采样。当收数据接收完后，硬件置位 RI0 表示接收结束并把第 9 位加载到 S0CON 寄存器的 RB80。

在四种模式中，使用 S0BUF 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 RI0=0 且 REN0=1 时启动接收。在其它模式，在 REN0=1 时，收到有负跳变的起始位时启动接收。

17.5. 帧错误侦测

开启帧错误侦测功能后，UART0会在通讯中侦测是否丢失停止位，如果丢失一个停止位，就设置S0CON寄存器的FE标志位。FE标志位和SM00标志位共享SCON0.7，SMOD0标志位(PCON.6)决定S0CON.7究竟代表哪个标志，如果SMOD0位(PCON0.6)置位则S0CON.7就是FE标志，SMOD0位清零则S0CON.7就是SM00标志。当S0CON.7代表FE时，只能软件清零。参考图 17–7。

图 17–7. UART0 帧错误侦测



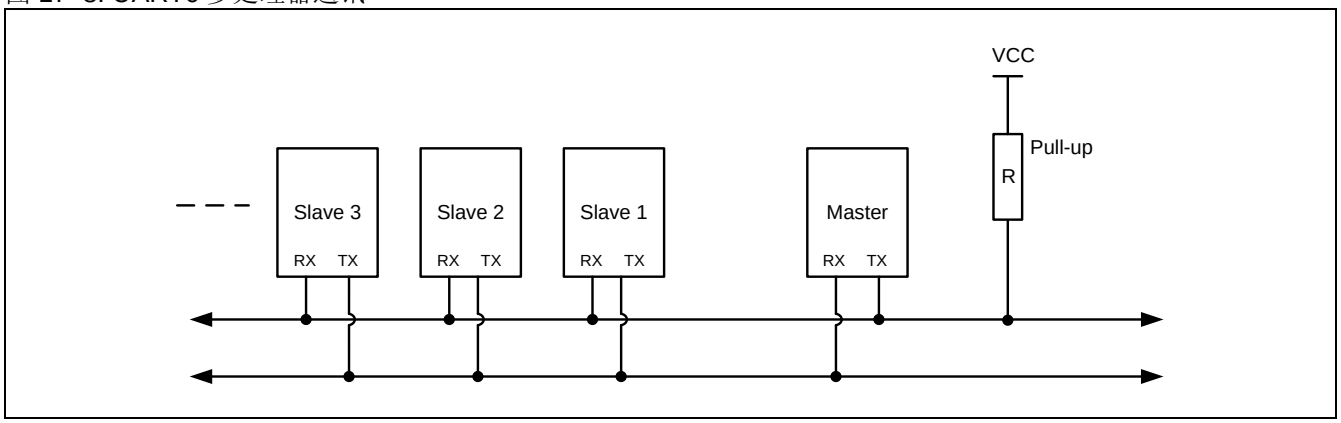
17.6. 多处理器通讯

模式 2 和 3 在用作多处理器通讯时有特殊的规定如图 17–8 所示。在这两种模式，接收 9 个数据位。第 9 个数据位存入 RB80 之后，接着进来一个停止位。端口可以编程为：在 RB80=1 时，当收到停止位后，串口中断将激活。这种特征通过设置 SM20 位(在 S0CON 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 SM20=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侦测收到的字节是否是自己的地址。从机地址将清除 SM20 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 SM20 置位，并继续他们的工作，忽略进来的数据字节。

SM20 在模式 0 和模式 1 没有影响，但是可以用来侦测停止位的有效性。在接收模式 1 中，如果 SM20=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 17–8. UART0 多处理器通讯

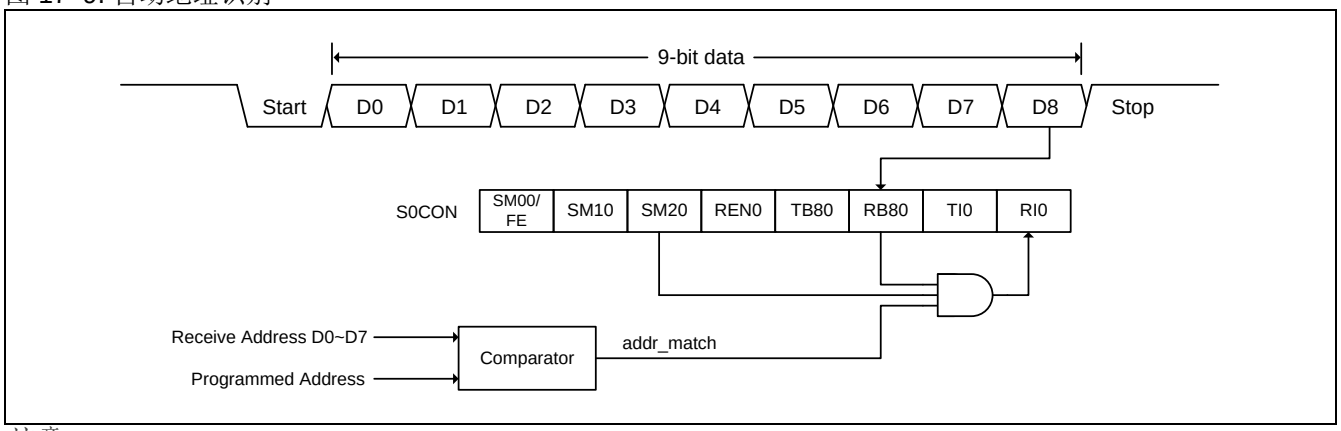


17.7. 自动地址识别

自动地址识别通过硬件比较可以让 UART0 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 S0CON 的 SM20 位来开启。

在 9 位数据 UART0 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(RI0)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 17–9。

图 17–9. 自动地址识别



注意:

(1) 收到匹配地址后(addr_match=1)，清 SM20 以接收数据字节。

(2) 收完全部数据字节后,置 SM20 为 1 以等待下一个地址。

在 8 位模式，即模式 1 下，如果 SM20 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 RI0 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。两个特殊功能寄存器(SADDR 和 SADEN 地址掩码寄存器)用来定义从机地址。

SADEN 用来定义 SADDR 中的位，哪些是可用的，哪些位是“忽略”的。使用 SADEN 掩码和 SADDR 执行一个逻辑与操作创建一个“给定”地址，该地址将用作从机的地址，并且主机可以将该“给定”地址在总线上发送，以从多个从机上识别出该从机。

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的创建都是通过 SADDR 和 SADEN 的逻辑或，0 在结果中按忽略处理。大部分情况下，体现忽略处理是所有为 1，使用十六进制的 FF 作为广播地址。

当复位时，SADDR (SFR 地址 0xA9)和 SADEN (SFR 地址 0xB9)被加载为 0。这将生成一个“给定”地址为“忽略”(“XXXX XXXXb”)和一个“广播地址”地址为“忽略”(“XXXX XXXXb”)。这有效地禁用了自动寻址模式，并允许微控制器使用标准的 80C51 类型 UART 驱动，而这些驱动不使用该特性。

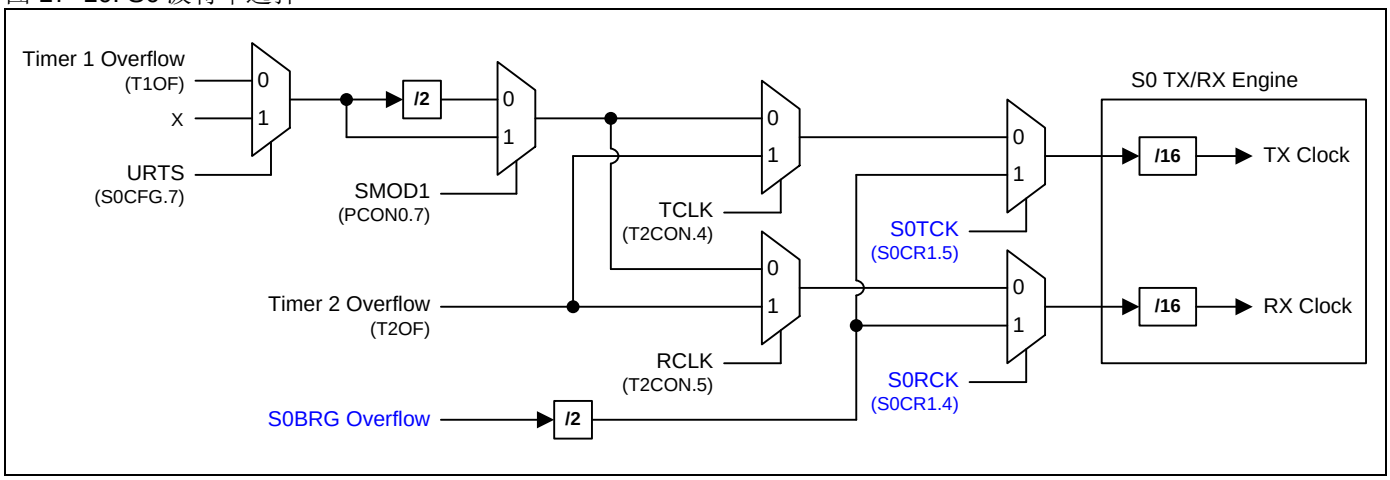
17.8. 波特率设置

位T2X12 (T2MOD.4)、T1X12 (AUXR2.3)、URM0X3 (S0CFG.5)和SMOD2 (S0CFG.6)提供一个新的波特率选项设置，如下所列。关于增强模式，请参照“17.11 串口增强型功能”。

17.8.1. S0 的波特率选择

UART0 运行在模式 1 和模式 3 时，清零 T2CON 寄存器的位 TCLK 和 RCLK 软件可选择定时器 1 作为波特率发生器。此刻，如果 URTS(S0CFG.7)置位，定时器 1 的溢出信号将被 UART1 的波特率发生器(S1BRG)取代。换句话说，一旦 RCLK=0、TCLK=0 和 URTS=1 用户可采用 S1BRG 作为 UART0 模式 1 或模式 3 的波特率发生器。这样，定时器 1 可以自由地做其它操作。当然 UART1(模式 1 或模式 3)也是这样运行，这两个串行口(UART)将有同样的波特率。

图 17–10. S0 波特率选择



17.8.2. 移位寄存器模式波特率（模式 0 和增强模式）

$$\text{Mode 0 Baud Rate} = \frac{F_{SYSCLK}}{n}$$; n=12, if URM0X3=0 ; n=4, if URM0X3=1

注意：

如果 URM0X3=0，波特率公式跟标准 8051 一样。

Enhance Mode:

$$\frac{2^{(SMOD2+2)}}{32} \times \frac{F_{SYSCLK}}{12 \times (256 - S0BRT)}$$;S0TX12 = 0

$$\frac{2^{(SMOD2+2)}}{32} \times \frac{F_{SYSCLK}}{1 \times (256 - S0BRT)}$$;S0TX12 = 1

17.8.3. 模式 2 波特率

当 S0BC0 = 0,

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

当 S0BC0 = 1,

$$\text{Mode 2 Baud Rate} = \frac{2^{SMOD1} \times 2^{(SMOD2 \times 2)}}{48} \times F_{SYSCLK}$$

注意：
如果 SMOD2=0，波特率公式跟标准 8051 一样。如果 SMOD2=1，波特率设置有增强功能。表 17–2 定义了模式 2 波特率发生器由 SMOD2 因数决定的波特率设置。

表 17–2. SMOD2 在模式 2 的应用标准

SMOD2	SMOD1	波特率	备注	推荐的最大接收误差 (%)
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%

注意：当定时器 1 在双倍波特率 x4(SMOD1=1 和 SMOD2=1)模式时，TH1 不能等于 254 和 255。

表 17–3. @ F_{SYSCLK}=16MHz & S0BC0 = 0 时串口 0（S0）模式 2 波特率

Baud Rate	SMOD2	SMOD1	Error
250,000	0	0	0.0%
500,000	0	1	0.0%
1,000,000	1	0	0.0%
2,000,000	1	1	0.0%

表 17–4. @ F_{SYSCLK}=16MHz & S0BC0 = 1 时串口 0（S0）模式 2 波特率

Baud Rate	SMOD2	SMOD1	Error
333,333	0	0	0.0%
666,667	0	1	0.0%
1,333,333	1	0	0.0%
2,666,667	1	1	0.0%

17.8.4. 模式 1 和 3 波特率&增强模式

17.8.4.1. 使用定时器 1 作为波特率发生器

当 S0BC0 = 0,

$$\text{Mode 1, 3 Baud Rate} = \frac{2^{\text{SMOD2 X (SMOD1 + 1)}} \times F_{SYSCLK}}{32 \times (65536 - (\text{RCAP2H, RCAP2L}))} \; ; \text{T2X12=0}$$

$$\text{or} = \frac{2^{\text{SMOD2 X (SMOD1 +1)}} \times F_{SYSCLK}}{16 \times (65536 - (\text{RCAP2H, RCAP2L}))} \; ; \text{T2X12=1}$$

当 S0BC0 = 1,

$$\text{Mode 1, 3 Baud Rate} = \frac{2^{\text{SMOD1}} \text{ X } 2^{(\text{SMOD2 X 2})}}{24} \text{ X } \frac{F_{SYSCLK}}{12 \times (256 - \text{TH1})} \; ; \text{T1X12=0}$$

$$\text{or} = \frac{2^{\text{SMOD1}} \text{ X } 2^{(\text{SMOD2 X 2})}}{24} \text{ X } \frac{F_{SYSCLK}}{1 \times (256 - \text{TH1})} \; ; \text{T1X12=1}$$

注意：

如果 SMOD2=0，T1X12=0，波特率公式跟标准 8051 一样。如果 SMOD2=1，波特率设置有增强功能。表 17–5 定义了定时器 1 波特率发生器由 SMOD2 因数决定的波特率设置。

表 17–5. SMOD2 在模式 1 和 3 使用定时器 1 的应用标准

SMOD2	SMOD1	波特率	备注	推荐的最大接收误差 (%)
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%

注意：当定时器 1 在双倍波特率 x4(SMOD1=1 和 SMOD2=1) 模式时，TH1 不能等于 254 和 255。

图 17–6 ~ 表 17–9 列出了 8 位自动加载模式的定时器 1 中各种常用的波特率和怎样获得。对于非标准波特率，当 F_{SYSCLK} = 24MHz 时，频率最大是 3MHz。

表 17–6. 定时器 1 产生的常用波特率@ F_{SYSCLK}=16.0MHz & *S0BC0 = 0*

Baud Rate	**TH1**, 重载值					
	T1X12=0 & SMOD2=0			**T1X12=1 & SMOD2=0**		
	SMOD=0	SMOD=1	误差	SMOD1=0	SMOD1=1	误差
1200	221(0.79%)	187(0.64%)	0.79%	--	--	--
2400	239(2.12%)	221(0.79%)	2.12%	48	--	0.16%
4800	--	239	2.12%	152	48	0.16%
9600	--	--	--	204	152	0.16%
14400	--	--	--	221	186	0.79%
19200	--	--	--	230	204	0.16%
28800	--	--	--	239(2.21%)	221(0.79%)	2.12%
38400	--	--	--	243	230	0.16%
57600	--	--	--	--	239	2.12%
115200	--	--	--	--	--	--

表 17–7. 定时器 1 产生的高波特率@ F_{SYSCLK}=16.0MHz & *S0BC0 = 0*

Baud Rate	**TH1**, 重载值					
	T1X12=0 & SMOD2=0			**T1X12=0 & SMOD2=0**		
	SMOD=0	SMOD=1	误差	SMOD1=0	SMOD1=1	误差
115.2K	--	--	--	239(2.21%)	221(0.79%)	2.12%
230.4K	--	--	--	--	239	2.12%
460.8K	--	--	--	--	--	--

表 17–8. 定时器 1 产生的常用波特率@ F_{SYSCLK}=16.0MHz & *S0BC0 = 1*

Baud Rate	**TH1**, 重载值					
	T1X12=0 & SMOD2=0			**T1X12=0 & SMOD2=0**		
	SMOD=0	SMOD=1	误差	SMOD1=0	SMOD1=1	误差
1200	210(0.64%)	163(0.44%)	0.64%	--	--	--
2400	233	210	0.64%	--	--	--
4800	--	233	0.64%	117	--	0.08%
9600	--	--	--	187(0.64%)	117(0.08%)	0.64%
14400	--	--	--	210(0.64%)	163(0.44%)	0.64%
19200	--	--	--	221(0.79%)	187(0.64%)	0.79%
28800	--	--	--	233	210	0.64%
38400	--	--	--	239(2.21%)	221(0.79%)	2.12%
57600	--	--	--	--	233	0.64%
115200	--	--	--	--	--	--

表 17–9. 定时器 1 产生的高波特率@ F_{SYSCLK}=16.0MHz & *S0BC0 = 1*

Baud Rate	**TH1**, 重载值					
	T1X12=0 & SMOD2=0			**T1X12=0 & SMOD2=0**		
	SMOD=0	SMOD=1	误差	SMOD1=0	SMOD1=1	误差
115.2K	--	--	--	233	--	0.64%
230.4K	--	--	--	--	--	--
460.8K	--	--	--	--	--	--

17.8.4.2. 使用定时器 2 作为波特率发生器

当定时器 2 作波特率发生器时(T2CON 寄存器中的 TCLK 或 RCLK 任一位为‘1’)，波特率如下：

当 S0BC0 = 0,

$$\text{Mode 1, 3 Baud Rate} = \frac{2^{\text{SMOD2 X (SMOD1 + 1)}} \times F_{SYSCLK}}{32 \times (65536 - (\text{RCAP2H, RCAP2L}))} \quad ; \text{T2X12=0}$$

$$\text{or} = \frac{2^{\text{SMOD2 X SMOD1 +1)}} \times F_{SYSCLK}}{16 \times (65536 - (\text{RCAP2H, RCAP2L}))} \quad ; \text{T2X12=1}$$

当 S0BC0 = 1,

$$\text{Mode 1, 3 Baud Rate} = \frac{2^{\text{SMOD2 X (SMOD1 + 1)}} \times F_{SYSCLK}}{24 \times (65536 - (\text{RCAP2H, RCAP2L}))} \quad ; \text{T2X12=0}$$

$$\text{or} = \frac{2^{\text{SMOD2 X (SMOD1 + 1)}} \times F_{SYSCLK}}{12 \times (65536 - (\text{RCAP2H, RCAP2L}))} \quad ; \text{T2X12=1}$$

注意：

如果 SMOD2=0，波特率公式跟标准 8051 一样。如果 SMOD2=1，波特率设置有增强功能。表 17–10 定义了定时器 2 波特率发生器由 SMOD2 因数决定的波特率设置。

表 17–10. SMOD2 在模式 1 和 3 使用定时器 2 的应用标准

SMOD2	SMOD1	波特率	备注	推荐的最大接收误差 (%)
0	X	缺省波特率	标准功能	± 3%
1	0	双倍波特率	标准功能	± 3%
1	1	双倍波特率 **X2**	增强型功能	± 2%

注意：

当定时器 2 在双倍波特率 x2(SMOD1=1 和 SMOD2=1)模式时，RCAP2H 和 RPAC2L 不能等于 65534 和 65535。

图 17–11 ~ **表 17–14** 列出了定时器 2 中各种常用的波特率和怎样获得

表 17–11. 定时器 2 产生的常用波特率@ F_{SYSCLK}=16.0MHz *& S0BC0 = 0*

Baud Rate	**[RCAP2H, RCAP2L]**, 重载值					
	T2X12=0 & SMOD2=0			**T2X12=1 & SMOD2=0**		
	SMOD=0	SMOD=1	误差	SMOD=0	SMOD=1	误差
1200	65120	65120	0.16%	64704	64704	0.16%
2400	65328	65328	0.16%	65120	65120	0.16%
4800	65432	65432	0.16%	65328	65328	0.16%
9600	65484	65484	0.16%	65432	65432	0.16%
14400	65501	65501	0.79%	65466	65466	0.79%
19200	65510	65510	0.16%	65484	65484	0.16%
28800	65519	65519	2.12%	65501	65501	0.79%
38400	65523	65523	0.16%	65510	65510	0.16%
57600	--	--	--	65519	65519	2.12%
115200	--	--	--	--	--	--

表 17–12. 定时器 2 产生的高波特率@ F_{SYSCLK}=16.0MHz *& S0BC0 = 0*

Baud Rate	**[RCAP2H, RCAP2L]**, 重载值					
	T2X12=0 & SMOD2=0			**T2X12=0 & SMOD2=0**		
	SMOD=0	SMOD=1	误差	SMOD=0	SMOD=1	误差
115.2K	--	65519	2.12%	65519(2.21%)	65515(0.79%)	2.12%
230.4K	--	--	--	--	65519	2.12%

表 17–13. 定时器 2 产生的常用波特率@ F_{SYSCLK}=16.0MHz & *S0BC0 = 1*

Baud Rate	**[RCAP2H, RCAP2L]**, 重载值					
	T2X12=0 & SMOD2=0			**T2X12=1 & SMOD2=0**		
	SMOD=0	SMOD=1	误差	SMOD=0	SMOD=1	误差
1200	64980	64980	0.08%	64424	64424	0.08%
2400	65258	65258	0.08%	64980	64980	0.08%
4800	65397	65397	0.08%	65258	65258	0.08%
9600	65467	65467	0.64%	65397	65397	0.08%
14400	65490	65490	0.64%	65443	65443	0.44%
19200	65501	65501	0.79%	65467	65467	0.64%
28800	65513	65513	0.64%	65490	65490	0.64%
38400	65519	65519	2.21%	65501	65501	0.79%
57600	--	--	--	65513	65513	0.64%
115200	--	--	--	--	--	--

表 17–14. 定时器 2 产生的高波特率@ F_{SYSCLK}=24.0MHz

Baud Rate	**[RCAP2H, RCAP2L]**, 重载值					
	T2X12=0 & SMOD2=1			**T2X12=1 & SMOD2=1**		
	SMOD=0	SMOD=1	误差	SMOD=0	SMOD=1	误差
115.2K	--	65513	0.64%	--	--	--
230.4K	--	--	--	--	--	--

17.8.4.3. 使用分割定时器 2 作为波特率发生器

当定时器 2 为分割模式，并且被用作波特率发生器时(T2CON 的 TCLK 或 RCLK 为'1'), 波特率计算公式如下。

当 S0BC0 = 0,

$$\text{Mode 1, 3 Baud Rate} = \frac{2^{\text{SMOD2 X (SMOD1 + 1)}} \times F_{SYSCLK}}{16 \times 12 \times (256 - \text{RCAP2L})} \quad ; \text{TL2X12=0}$$

$$\text{or} = \frac{2^{\text{SMOD2 X SMOD1 +1)}} \times F_{SYSCLK}}{16 \times 1 \times (256 - \text{RCAP2L})} \quad ; \text{TL2X12=1}$$

当 S0BC0 = 1,

$$\text{Mode 1, 3 Baud Rate} = \frac{2^{\text{SMOD2 X (SMOD1 + 1)}} \times F_{SYSCLK}}{12 \times 12 \times (256 - \text{RCAP2L})} \quad ; \text{TL2X12=0}$$

$$\text{or} = \frac{2^{\text{SMOD2 X SMOD1 +1)}} \times F_{SYSCLK}}{12 \times 1 \times (256 - \text{RCAP2L})} \quad ; \text{TL2X12=1}$$

注:

表 17–15 在分割定时器 2 波特率发生器中使用 SMOD2 和 SMOD1 因子定义波特率设置。

表 17–15. SMOD2 在分割定时器 2 模式 1&3 下的应用条件

SMOD2	SMOD1	波特率	备注	推荐的最大接收误差 (%)
0	X	缺省波特率	标准功能	± 3%
1	0	双倍波特率	标准功能	± 3%
1	1	双倍波特率 **X2**	增强型功能	± 2%

注：当定时器 2 在双倍波特率 X2 模式下(SMOD1=1 & SMOD2=1)，RPAC2L 不能等于 254&255.

17.8.4.4. 使用串口 0 波特率定时器作为波特率发生器(S0BRG)

MG82F6B08 / 6B001/ 6B104 的 S0 中嵌入了一个专用的波特率发生器(S0BRG)，其详细功能在“17.11 串口增强型功能”一节中进行了描述。当使用 S0BRG 作为 S0 的波特率发生器时，波特率计算公式如下：

当 S0BC0 = 0,

$$\text{Mode 1, 3 Baud Rate} = \frac{2^{(\text{SMOD2})}}{32} \times \frac{F_{SYSCLK}}{12 \times (256 - \text{S0BRT})} \quad ; \text{S0TX12=0, SMOD1=0}$$

$$\text{or} = \frac{2^{(\text{SMOD2})}}{32} \times \frac{F_{SYSCLK}}{1 \times (256 - \text{S0BRT})} \quad ; \text{S0TX12=1, SMOD1=0}$$

当 S0BC0 = 1,

$$\text{Mode 1, 3 Baud Rate} = \frac{2^{(\text{SMOD2})}}{24} \times \frac{F_{SYSCLK}}{12 \times (256 - \text{S0BRT})} \quad ; \text{S0TX12=0, SMOD1=0}$$

$$\text{or} = \frac{2^{(\text{SMOD2})}}{24} \times \frac{F_{SYSCLK}}{1 \times (256 - \text{S0BRT})} \quad ; \text{S0TX12=1, SMOD1=0}$$

注：

表 17–16 在 S0BRG 波特率发生器中使用 SMOD2 和 SMOD1 因子定义波特率设置。

表 17–16. SMOD22 在 S0BRG 模式 1&3 下的应用条件

SMOD2	SMOD1	波特率	备注	推荐最大接收误差(%)
0	0	默认波特率	标准功能	± 3%
1	0	双倍波特率	增强功能	± 3%
X	1	不支持		

17.9. 串口 0 模式 4 (SPI 主机)

MG82F6B08 / 6B001/ 6B104 的串口 0 嵌入了一个额外的模式 4 来支持 SPI 主机引擎。模式 4 由 SM30、SM00 和 SM10 选择。请参考“表 17–1. 串口 0 模式选择”

URM0X3 也可控制 SPI 的传输速度。如果 URM0X3 = 0，则 SPI 的时钟频率是 SYSCLK/12。如果 URM0X3 = 1，则 SPI 的时钟频率是 SYSCLK/4

MG82F6B08 / 6B001/ 6B104 的 SPI 主机使用 TXD0 作为 SPICLK，RXD0 作为 MOSI，以及 S0MI 作为 MISO。而 nSS 由 MCU 软件选择在其它端口引脚。SPI 连接如图 17–11 所示。他支持多从机通讯架构见图 17–12。

图 17–11. 串口 0 模式 4，单主机和单从机架构 (n = 0)

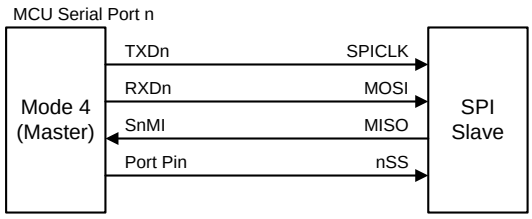
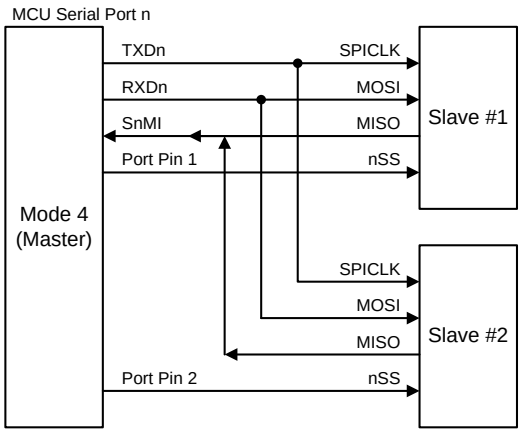


图 17–12. 串口 0 模式 4，单主机和多从机架构(n = 0)



SPI 主机能满足笙泉 MG82/84 系列 MCU(由 CPOL、CPHA 和 DORD 选择)的全功能 SPI 模块的传输。在 CPOL 和 CPHA 条件下，**MG82F6B08 / 6B001/ 6B104** 很容易初始化 SPI 的时钟极性去适合他们使用。串行口模式 4 的 4 个 SPI 工作模式如表 17–17 所示。

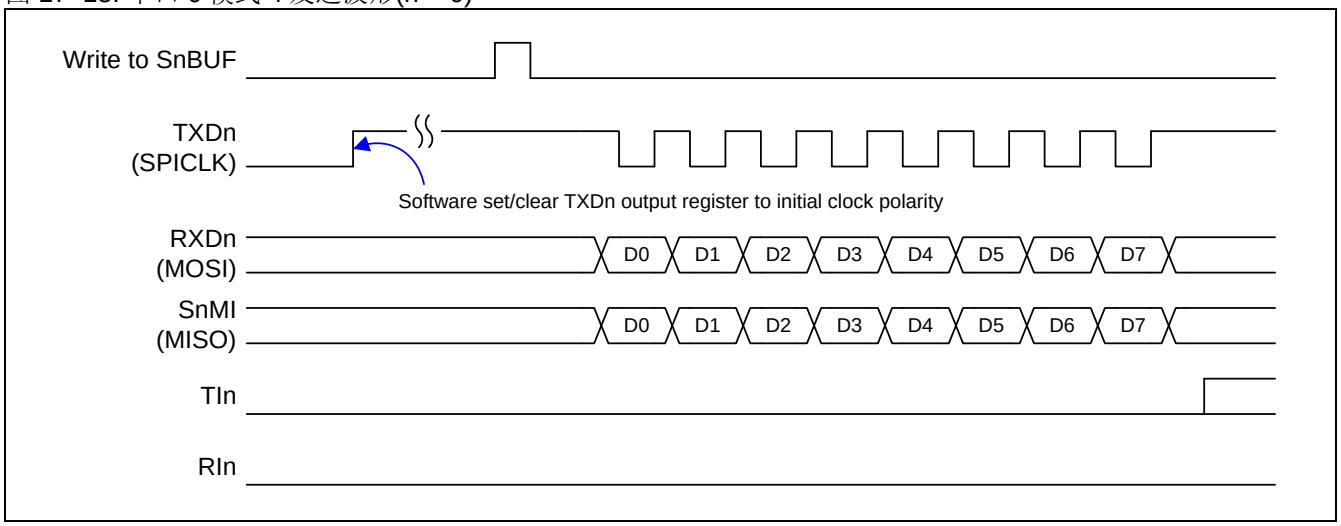
表 17–17. 串口 0 模式 4 的 SPI 模式配置

SPI Mode	CPOL	CPHA	在 TXD0 的配置
0	0	0	清除 TXD0 输出寄存器 为 “0”
1	0	1	清除 TXD0 输出寄存器 为 “0”
2	1	0	设置 TXD0 输出寄存器 为 “1”
3	1	1	设置 TXD0 输出寄存器 为 “1”

SPI 系列传输的位序控制(DORD)，**MG82F6B08 / 6B001/ 6B104** 提供了位序控制(S0DOR)。S0DOR 的默认值是“1”且位序控制为低位在先(LSB)。

由任何指令把 S0BUF 作为目的寄存器的使用初始化发送。“写到 S0BUF”信号触发 UART 引擎开始发送。S0BUF 的数据作为 MOSI 串行数据被移位到 RXD0 引脚。SPI 移位时钟在 TXD0 引脚上作为 SPICLK 输出。8 个移位时钟上升沿之后，硬件生效 TI0 表示发送结束。同时 S0MI 引脚也被采样且移位到移位寄存器。然后，“读取 S0BUF”会获得 SPI 的移入数据。模式 0 的发送波形如图 17–13 所示。模式 4 中 RI0 不会生效。

图 17–13. 串口 0 模式 4 发送波形(n = 0)



17.10. 串口 0 寄存器

串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。3 个寄存器 PCON0、AUXR2 和 S0CFG 是与波特率的设定有关。

S0CON:串口 0 控制寄存器
SFR 页 = **仅 0 页**
SFR 地址 = 0x98 复位值 = 0000-0000

7	6	5	4	3	2	1	0
SM00/FE	SM10	SM20	REN0	TB80	RB80	TI0	RI0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：FE，帧错误位。SMOD0 必须置位才能访问 FE 位。
0：FE 位不会被有效的帧清零，它应当被软件清零。
1：当检测到一个无效的停止位时，该位被接收器置位。

Bit 7：串口 0 模式位 0，(SMOD0 必须 = 0 才能访问位 SM00)。
Bit 6：串口 0 模式位 1。

SM30	SM00	SM10	Mode	描述	波特率
0	0	0	0	移位寄存器	SYSCLK/12 or SYSCLK/4
0	0	1	1	8-位 UART	可变
0	1	0	2	9-位 UART	SYSCLK/64, /32, /16, /8
0	1	1	3	9-位 UART	可变
1	0	0	4	**SPI 主机**	SYSCLK/12 or SYSCLK/4
1	0	1	5	保留	保留
1	1	0	6	保留	保留
1	1	1	7	保留	保留

Bit 5：串口 0 模式位 2。
0：禁止 SM20 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 将不能被置位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：REN0，使能串口接收。
0：软件清零将禁止接收。
1：软件置位使能接收。

Bit 3：TB80，在模式 2 和 3 时第 9 位数据被传送，根据需要通过软件置位或清零。

Bit 2：RB80，在模式 2 和 3 时收到的第 9 位数据。在模式 1，如果 SM20=0，RB80 是收到数据的停止位。在模式 0，RB80 没有使用。

Bit 1：TI0，发送中断标志。
0：必须由软件清零。
1：在模式 0 时，在第 8 位数据位时序后由硬件置位。其它模式中，在发送停止位之初由硬件置位。

Bit 0：RI0，接收中断标志。
0：必须由软件清零。
1：在模式 0 时，在第 8 位数据位时序后由硬件置位。其它模式中(除留意 SM20 外)，在接收停止位的中间时刻由硬件置位。

S0BUF:串口 0 缓冲寄存器

SFR 页 = **仅 0 页**

SFR 地址 = 0x99 复位值 = XXXX-XXXX

7	6	5	4	3	2	1	0
S0BUF.7	S0BUF.6	S0BUF.5	S0BUF.4	S0BUF.3	S0BUF.2	S0BUF.1	S0BUF.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~0：在发送和接收时作缓冲寄存器。

SADDR:从机地址寄存器

SFR 页 = 0~F

SFR 地址 = 0xA9 复位值 = 0000-0000

7	6	5	4	3	2	1	0
SADDR.7	SADDR.6	SADDR.5	SADDR.4	SADDR.3	SADDR.2	SADDR.1	SADDR.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 (SMOD3 = 0)

SFR 页 = 0~F

SFR 地址 = 0xB9 复位值 = 0000-0000

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

SADDR 和 SADEN 组合来形成自动地址识别的要求/广播地址。事实上，SADEN 是 SADDR 的“屏蔽”寄存器。如下所示。

SADDR = 1100 0000
SADEN = 1111 1101
Given = 1100 00x0 给定的从机地址将被核对，除了位 1 被视为“不关心”

每个从机的广播地址为 SADDR 和 SADEN 进行逻辑“或”的结果。结果中为“0”被认为“不关心”。在系统复位后，SADDR 和 SADEN 都被初始化为 0。这样生成了所有的“不关心”要求地址和所有的“不关心”广播地址。这将禁止自动地址侦测功能。

PCON0:电源控制寄存器 0

SFR 页 = 0~F POR = 0001-0000

SFR 地址 = 0x87 复位值 = 0000-0000

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，帧错误选择。
0：S0CON.7 作 SM0 功能。
1：S0CON.7 作 FE 功能。注：当帧错误后不管 SMOD0 什么状态 FE 都将置位。

S0CFG:串口 0 配置寄存器

SFR 页 =**仅 0 页**

SFR 地址 = **0x9C** 复位值 = 0000-1000

7	6	5	4	3	2	1	0
URTS	SMOD2	URM0X3	SM30	S0DOR	BTI	UTIE	SMOD3
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：URTS，UART0 定时器选择。
0：定时器 1 或定时器 2 作为模式 1 或模式 3 的波特率发生器。
1：当定时器 1 被选作串口 0 模式 1 或模式 3 的波特率发生器时定时器 1 溢出信号被串口 1 波特率定时器溢出信号取代。
(请参考章节“17.8.4 模式 1 和 3 波特率”)。

Bit 6：SMOD2，UART0 额外双倍波特率选择。
0：禁止 UART0 额外双倍波特率。
1：使能 UART0 额外双倍波特率。

Bit 5：URM0X3，串口 0 模式 0 和模式 4 波特率选择。
串口 1 模式 0 和模式 4 波特率选择：
0：清零选择 SYSCLK/12 作串口 0 模式 0 和模式 4 波特率。
1：置位选择 SYSCLK/4 作串口 0 模式 0 和模式 4 波特率。
串口 0 模式 2：
0：清零选择 SYSCLK/32 或/64 作串口 0 波特率。
1：置位选择 SYSCLK/96 或/192 作串口 0 波特率。

Bit 4：SM30，串口模式控制位 3。

Bit 3：S0DOR，串口 0 所有操作模式的数据位序控制。
如果串口 0 不是定时器模式：
0：数据字节高位在先(MSB)传送。
1：数据字节低位在先(LSB)传送。默认是 S0DOR 为“1”。
如果串口 0 是定时器模式：
0：设置 S0BRG 为 8 位重载定时器/计数器模式。
1：设置 S0BRG 为 16 位定时器/计数器模式。

Bit 2：BTI，在串口 0 中断阻止 TI0。
0：保留 TI0 作为一个串口 0 中断源。
1：阻止 TI0 作为一个串口 0 中断源。

Bit 1：UTIE，在系统标志中断里使能 S0 TI0
0：禁止在系统标志中断里中断向量共享给 TI0。
1：设置 TI0 标志将与系统标志中断共享中断向量。

Bit 0：SMOD3，S0CR1 访问控制。
0：禁止 S0CR1 访问。CPU 访问特殊功能寄存器地址 0xB9 为读/写 SADEN。
1：使能 S0CR1 访问。CPU 访问特殊功能寄存器地址 0xB9 为读/写 S0CR1。

AUXR2:辅助寄存器 2

SFR 页 = 0~F

SFR 地址 = 0xA3 复位值 = 0000-0000

7	6	5	4	3	2	1	0
STAF	STOF	0	C0PLK	T1X12	T0X12	T1CKOE	T0CKOE
R/W	**R/W**	**W**	**W**	**R/W**	**R/W**	**R/W**	**R/W**

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：置位选择 SYSCLK 作时钟源。若置位，在模式 1 和模式 3 中 UART0 选择定时器 1 作波特率则速率是标准 8051 的 12 倍。

AUXR3:辅助寄存器 3

SFR 页 = 仅 **0** 页

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	T0PS0	BPOC1	BPOC0	S0PS0	TWIPS1	TWIPS0	T0XL
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 3：S0PS0, 串口 0 引脚选择 0。(S0PS1 在 AUXR10.3)

S0PS1~0	RXD0	TXD0
00	P3.0	P3.1
01	P4.4	P4.5
10	P4.5	P3.0
11	P4.5	P4.4

AUXR6:辅助寄存器 6

SFR 页 = 仅 **3** 页

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	0	0	0	0	T2FCS	SnMIPS	S0COPS
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 0：S0COPS, S0BRG 时钟输出(S0CKO)端口引脚选择。

S0COPS	S0CKO
0	P4.7
1	P4.4

17.11. 串口增强型功能

如果 SMOD3 (S0CFG.0)置位，特殊功能寄存器地址 0xB9 被当作 S0CR1 访问。S0CR1 控制串口 0 的增强型功能包括：

- 使能 S0 嵌入的波特率发生器，S0BRG。
- 使能 S0TX 或 RX 由 S0BRG 选择波特率时间基准。
- 使能 S0BRG 成为一个通用定时器。
- 使能 S0 进入 LIN 总线模式。

S0CR1:串口 0 控制寄存器 1 (SMOD3 = 1)

SFR 页 = 0~F

SFR 地址 = 0xB9 复位值 = 0000-0000

7	6	5	4	3	2	1	0
S0TR	S0TX12	S0TCK	S0RCK	S0CKOE	ARTE	0	S0BC0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：S0TR，UART0 波特率发生器控制位。
0：清零停止 S0BRG 操作。
1：设置使能 S0BRG 操作。

Bit 6：S0TX12，S0BRG 时钟源选择。
0：清零选择 SYSCLK/12 作为 S0BRG 时钟源。
1：设置选择 SYSCLK 作为 S0BRG 时钟源。

Bit 5：S0TCK，S0 控制位选择 S0BRG 溢出作为 UART0 发送时钟。
0：促使定时器 1 或定时器 2 溢出用作发送时钟。
1：促使 S0 用 S0BRG 溢出作发送时钟且操作模式控制。

Bit 4：S0RCK，S0 控制位选择 S0BRG 溢出作为 UART0 接收时钟。
0：促使定时器 1 或定时器 2 溢出用作接收时钟。
1：促使 S0 用 S0BRG 溢出作接收时钟且操作模式控制。

Bit 3：S0CKOE，S0BRG 时钟输出控制。
0：禁止 S0BRG 时钟输出到 S0CKO。
1：使能 S0BRG 时钟输出到 S0CKO。

Bit 2：ARTE，自动重发使能。
0：禁止自动重发。
1：使能自动重发。

Bit 0：S0BC0，S0 波特率控制 0。
0：在 S0 模式 1/2/3/5 选择 UART 过采样为 16。
1：在 S0 模式 1/2/3/5 选择 UART 过采样为 12。

S0BRT:串口 0 波特率定时器重载寄存器

SFR 页 = **仅 0 页**

SFR 地址 = 0x9A 复位值 = 0000-0000

7	6	5	4	3	2	1	0
S0BRT.7	S0BRT.6	S0BRT.5	S0BRT.4	S0BRT.3	S0BRT.2	S0BRT.1	S0BRT.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~0：波特率定时器发生器的重载值寄存器与定时器 1 的工作方式相似。

S0BRC:串口 0 波特率计数寄存器

SFR 页 = **仅 0 页**

SFR 地址 = 0x9B 复位值 = 0000-0000

7	6	5	4	3	2	1	0
S0BRC.7	S0BRC.6	S0BRC.5	S0BRC.4	S0BRC.3	S0BRC.2	S0BRC.1	S0BRC.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~0：波特率计数器寄存器与定时器 1 的工作方式相似。这个寄存器可由软件读/写。如果 S0TR (S0CR1.7) = 0，软件写 S0BRT 数据将同时保存到 S0BRT 和 S0BRC。如果 S0TR= 1，软件写 S0BRT 数据将不保存到 S0BRC。

17.11.1. S0 波特率发生器(S0BRG)

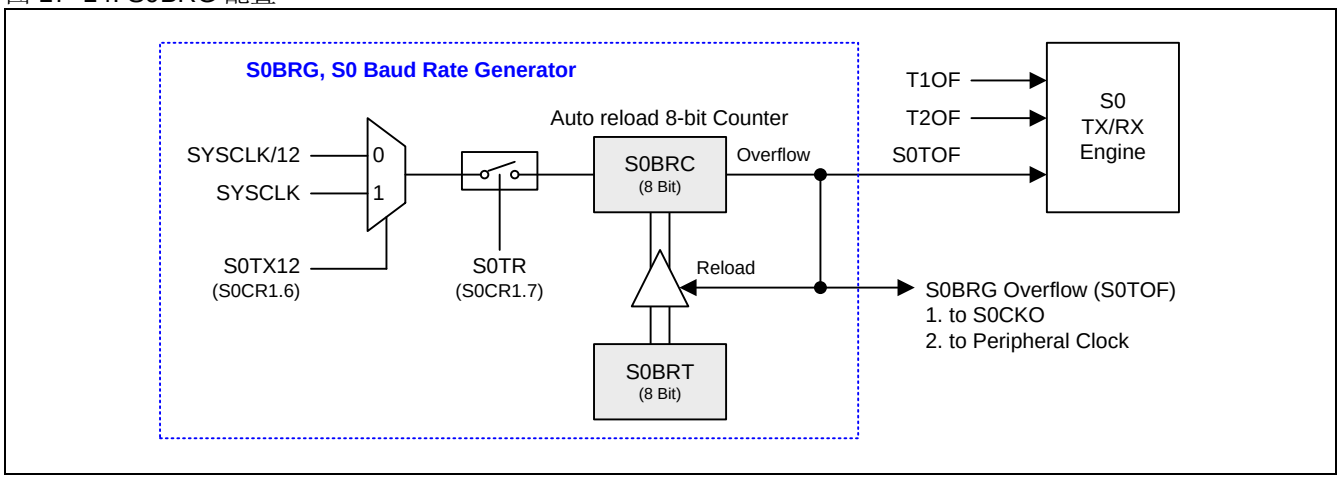
MG82F6B08 / 6B001/ 6B104 有一个嵌入的波特率发生器来产生串口 0 操作的时钟。它由一个 8 位的向上计数器(S0BRC)和一个 8 位的重载寄存器(S0BRT)构成。S0BRC 溢出(S0TOF)是 UART0 串行引擎在所有操作模式下的时间基准且触发 S0BRT 值重载到 S0BRC 保证连续计数。

如果 S0TR= 0，软件写 S0BRT 数据将同时修改 S0BRC。在 S0TR 使能开启 S0BRC 计数之后，当 S0BRT 写入时不影响 S0BRC。修改 S0BRC 不会影响 S0BRT 的值。

波特率发生器也提供时钟输出的时间基准，S0CKO，为 S0BRC 溢出率的二分频(S0TOF/2)。S0TOF 也供给其它外设的时钟输入切换源。无论 S0 运行还是待运行，S0BRG 总是给这些外设提供时间基准服务。

串口 0 波特率发生器结构如图 17–14 所示。

图 17–14. S0BRG 配置



17.11.2. 独立波特率发生器(S0BRG)应用于串口 0(S0)

为了给 S0 更大的灵活性，可以选择波特率发生器 S0BRG 作为 S0 波特率源。.
串口 0 波特率选择配置如请参考“图 17–10. S0 波特率选择”

17.11.3. S0 LIN 总线寄存器

S0CFG1:串口 0 配置寄存器 1
SFR 页 = **仅 0 页**
SFR 地址 = **0x9D** 复位值 = 0000-00xx

7	6	5	4	3	2	1	0
SBF0	TXER0	S0SB16	ATBR0	TXRX0	SYNC0	0	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**W**	**W**

Bit 7：SBF0，S0 同步中止标志。
0：必须软件清零。
1：LIN 总线侦测到中止事件的结束硬件置位。在主机模式，与 TI0 标志相连置位。在从机模式，与 RI0 标志相连置位。

Bit 6：TXER0，S0 的 LIN 发送错误。
0：必须软件清零。
1：在 TX 模式，LIN 总线侦测到发送错误硬件置位。

Bit 5：S0SB16，S0 同步中止 16 位使能。
0：在主机模式选择 13 位同步中止发送。
1：在主机模式选择 16 位同步中止发送。

Bit 4：ATBR0，S0 自动波特率。
0：在同步域结束硬件自动清零。
1：在同步域之前，软件置位在从机 RX 模式执行 LIN 总线同步域自动波特率调整。

Bit 3：TXRX0，S0 LIN 总线的 TX/RX 选择。
0：选择 LIN 总线接口引擎为 RX 功能。
1：选择 LIN 总线接口引擎为 TX 功能。

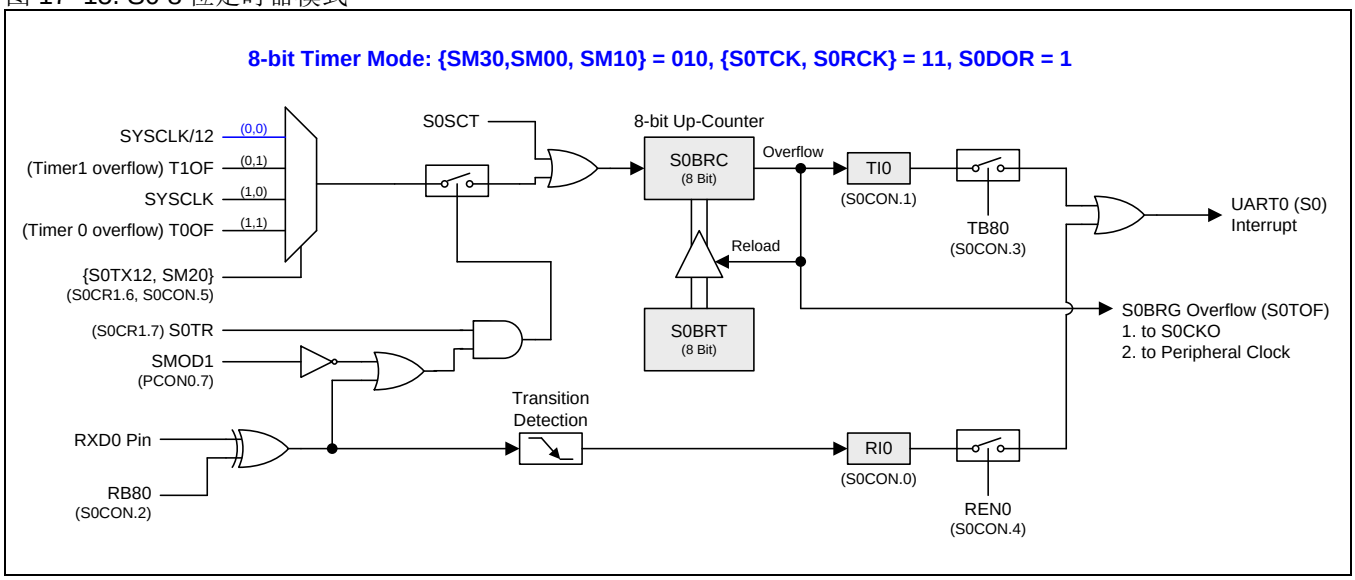
Bit 2：SYNC，S0 同步中止控制位。
0：当主机模式同步中止发送或从机模式接收到自动清零。
1：软件置位。如果主机模式置位，下一个写 S0BUF 将发送一个同步中止到 LIN 总线。如果从机模式置位，LIN 接口引擎将等待接收一个同步中止。

Bit 1~0：保留位，写寄存器时，此位必须写“0”。

17.11.4. S0 当做 8 位定时器模式

S0 的 8 位定时器模式如图 17–15 所示。

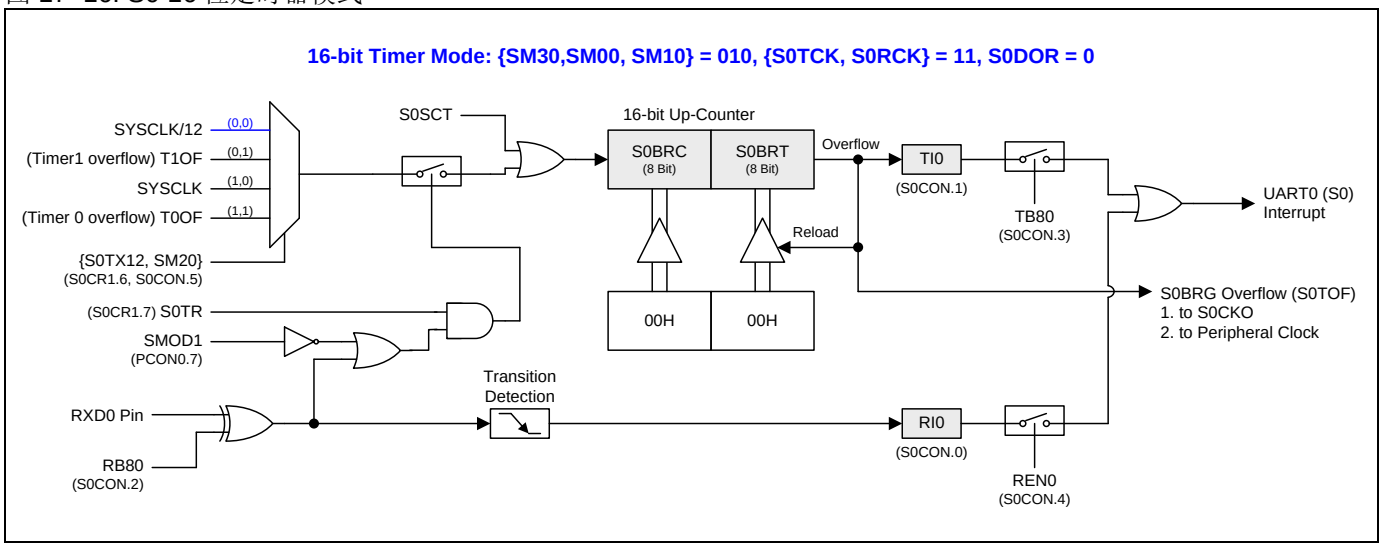
图 17–15. S0 8 位定时器模式



17.11.5. S0 当做 16 位定时器模式

S0 的 16 位定时器模式如图 17–16 所示。

图 17–16. S0 16 位定时器模式



17.11.6. S0BRG 可编程时钟输出

S0BRG 有一个时钟输出模式如图 17–17 和图 17–18 所示。

图 17–17. S0BRG 时钟输出(S0BRG 为 8 位定时器模式)

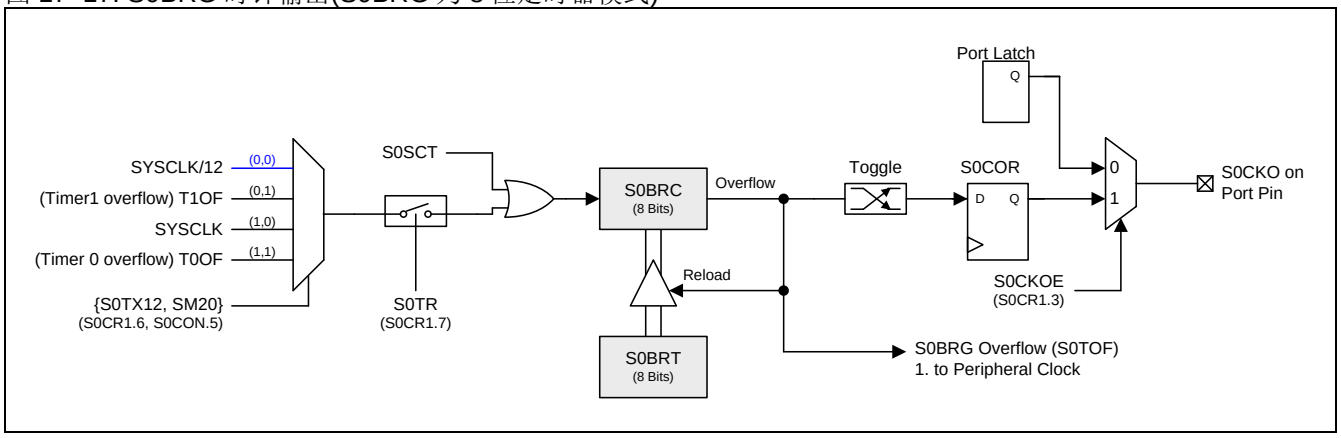
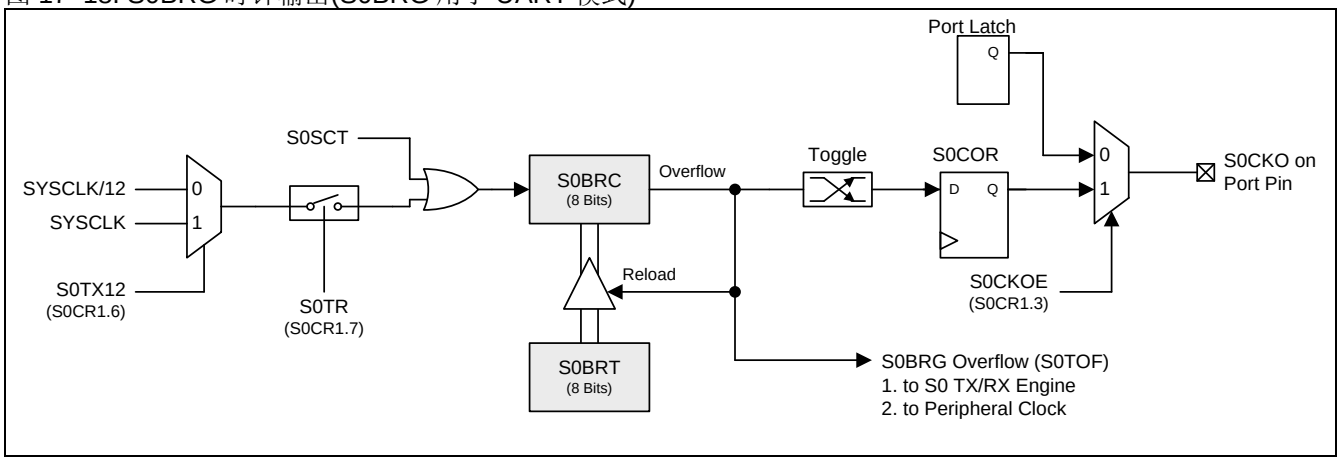


图 17–18. S0BRG 时钟输出(S0BRG 用于 UART 模式)



AUXR6:辅助寄存器 6

SFR 页 = **仅 3 页**

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	0	0	0	0	T2FCS	SnMIPS	S0COPS
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

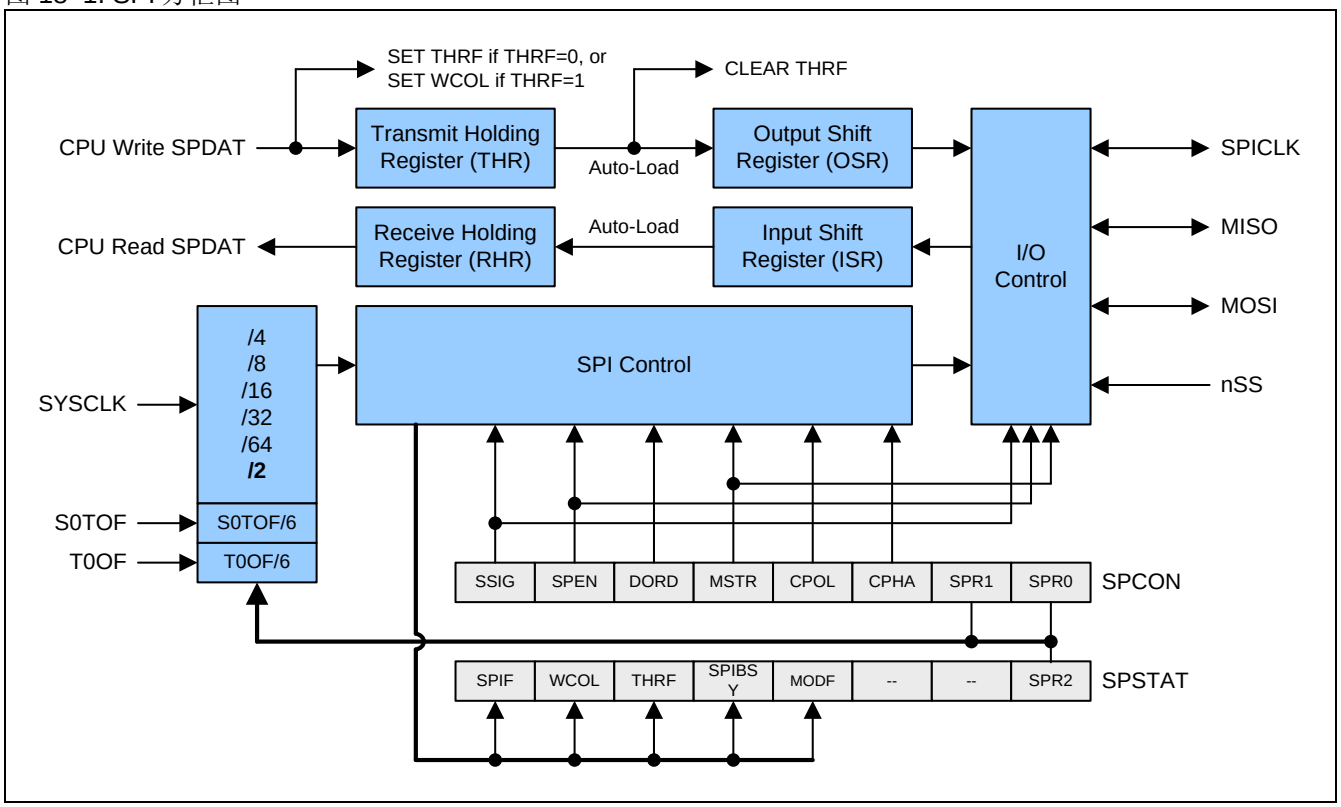
Bit 0：S0COPS, S0BRG 时钟输出(S0CKO)端口引脚选择。

S0COPS	S0CKO
0	P4.7
1	P4.4

18. 串行外围接口(SPI)

MG82F6B08 / 6B001/ 6B104 提供了一个高速串行外设接口(SPI)。SPI 接口是一种全双工、高速同步通讯总线，有两种操作模式：主机模式和从机模式。在 24MHz 的系统时钟下主机模式支持高达 **12Mbps** 速率或从机模式高达 **6Mbps**。在 SPI 状态寄存器(SPSTAT)里有三个标志传送完成标志(SPIF)，写冲突标志(WCOL)和模式错误标志(MODF)。与传统的 SPI 相比较，一个经过特别设计的发送保持寄存器(THR)显著改善了传输效率且 THRF 标志表明 THR 是满或空。SPI 工作下忙状态由只读标志 SPIBSY 指示。

图 18–1. SPI 方框图



SPI 接口有 4 个引脚：MISO、MOSI、SPICLK 和 nSS。

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

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

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

注意：引脚输出选项见 AUXR10 参考章节“4.3 功能复用”。

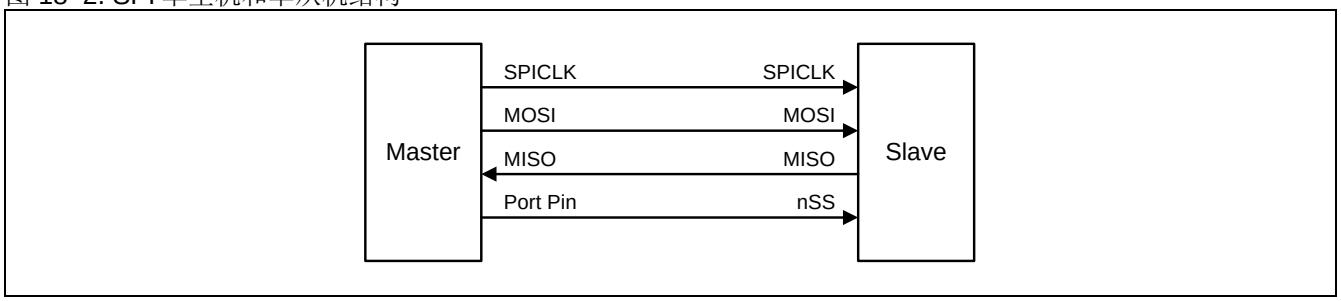
注意，即使 SPI 被配置成主机运行(MSTR=1)，它仍然可以被 nSS 引脚的低电平拉成从机(若 SSIG=0)，一旦发生这种情况，SPIF 位(SPSTAT.7)置位并且 SPEN 会被清零。 (参考章节“18.2.3 nSS 引脚模式改变”) 。

18.1. 典型 SPI 配置

18.1.1. 单主机和单从机

对于主机：任何端口，包括 SSGPIO，都可以用来控制从机的 nSS 片选引脚。
对于从机：SSIG 为‘0’，且 nSS 引脚决定该设备是否被选中。

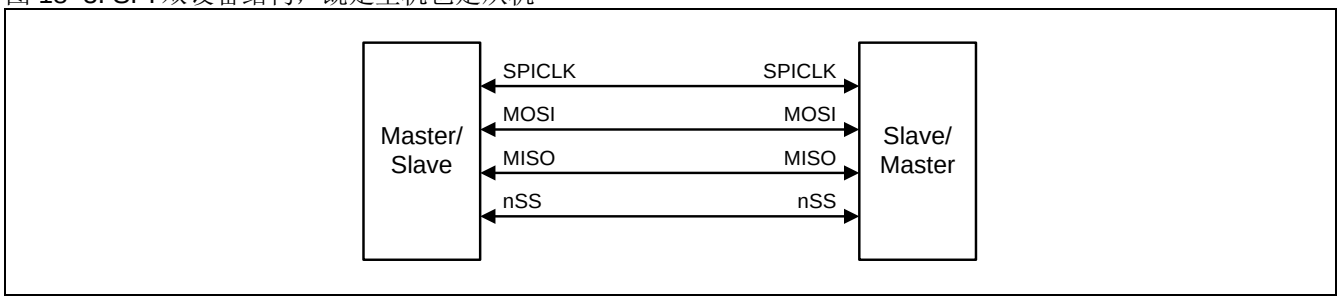
图 18–2. SPI 单主机和单从机结构



18.1.2. 双设备，既是主机也是从机

两个彼此连接的设备，均可成为主机或从机。没有 SPI 操作时，都可以被通过设置 MSTR=1、SSIG=0 和 nSS 双向口配置或开漏带上拉配置成主机。任何一方要发起传输，它可以配置 nSS 输出并强行拉低，使另一个设备发生“被改成从机模式”事件。(参考章节“**错误!未找到引用源。 错误!未找到引用源。**”)。

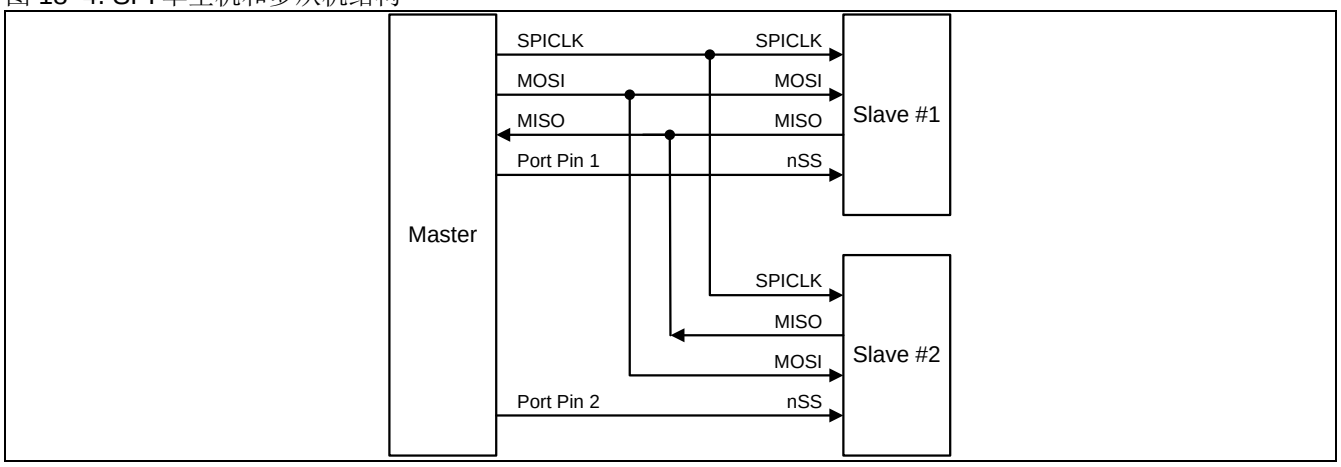
图 18–3. SPI 双设备结构，既是主机也是从机



18.1.3. 单主机和多从机

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

图 18–4. SPI 单主机和多从机结构



18.2. SPI 配置

表 18–1 不但列出了主机/从机模式的配置，而且列出了这些模式的用法和引脚状态。

表 18–1. SPI 主机和从机选择

SPEN (SPCON.6)	SSIG (SPCON.7)	nSS -pin	MSTR (SPCON.4)	模式	MISO -引脚	MOSI -引脚	SPICLK -引脚	注释
0	X	X	X	SPI 禁止	输入	输入	输入	SPI 相关端口引脚是通用端口引脚。
1	0	0	0	从机 (被选中)	输出	输入	输入	被选择为从机。
1	0	1	0	从机 (未被选中)	高阻	输入	输入	未被选中。
1	0	0	1 ➔ 0	从机 (通过模式改变)	输出	输入	输入	模式被改为从机。 若 nSS 被拉低，MSTR 被硬件自动清‘0’，且 SPEN 清零，MODF 置位。
1	0	1	1	主机 (空闲)	输入	Hi-Z	Hi-Z	MOSI 和 SPICLK 在主机待机时被置为高阻，以防止总线冲突。
				主机 (激活)		输出	输出	MOSI 和 SPICLK 在主机活动时推挽输出。
1	1	X	0	从机	输出	输入	输入	
1	1	X	1	主机	输入	输出	输出	

“X” 意味着 “忽略”。

18.2.1. 一个从机的补充注意事项

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

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

18.2.2. 一个主机的补充注意事项

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

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

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

18.2.3. nSS 引脚模式改变

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

18.2.4. 发送保持寄存器满标志

为了提高 SPI 发送速度一个特殊设计保持寄存器(THR)可以减短 CPU 数据移动字节与字节传送的延迟时间。THRF(SPSTAT.5)置位表明 THR 的数据是有效的并且等待发送。如果 THR 是空的(THRF=0)，软件写一个字节数据到 SPDAT 数据将存储在 THR 并且 THRF 置位。如果输出移位寄存器(OSR)是空的，硬件立刻将 THR 数据移到 OSR 并且 THRF 清零。在 SPI 主机模式，OSR 有效数据将触发 SPI 发送。在 SPI 从机模式，OSR 有效数据等待另一个 SPI 主机移出数据。如果 THR 是非空(THRF=1)，软件写一个字节数据到写冲突标志 WCOL (SPSTAT.6)将置位。

18.2.5. 写冲突

MG82F6B08 / 6B001/ 6B104 的 SPI 在发送方向和接收方向是双缓冲数据器。发送新数据在 THR 空时才能写入到缓冲器 THR。只读标志 THRF 表示 THR 是空或非空。在 THRF 为“1”时数据寄存器被写入数据冲突标志 WCOL (SPSTAT.6)将置位。这种情况下，SPDAT 写入操作将被忽略。

主机或从机检测到写冲突时，主机异常是主机传输过程中有非空控制；从机是在主机初始化传输没有控制结束时出现冲突。

WCOL 软件写“1”清零。

18.2.6. SPI 时钟速率选择

SPI 时钟率选择(主机模式)使用 SPCON 寄存器的 SPR1 和 SPR0 位及 SPSTAT 寄存器的 SPR2 来设置，如表 18–2 所示。

表 18–2. SPI 串行时钟速率

SPR2	SPR1	SPR0	SPI 时钟选择	SPI 时钟率@ SYSCLK=12MHz	SPI 时钟率@ SYSCLK=24MHz
0	0	0	SYSCLK/4	4 MHz	6 MHz
0	0	1	SYSCLK/8	2 MHz	3 MHz
0	1	0	SYSCLK/16	1 KHz	1.5 MHz
0	1	1	SYSCLK/32	500 KHz	750 KHz
1	0	0	SYSCLK/64	250 KHz	375 KHz
1	0	1	SYSCLK/2	8 MHz	12 MHz
1	1	0	**S0TOF/6**	可变	可变
1	1	1	**T0OF/6**	可变	可变

注意：

1. *SYSCLK 是系统时钟。*
2. *S0TOF 是 UART0 波特率发生器溢出。*
3. *T0OF 是定时器 0 溢出。*

18.3. 数据模式

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

表 18–3. SPI 模式定义

SPI Mode	CPOL	CPHA	前沿	后沿
0	0	0	采样(上升沿)	设置(下降沿)
1	0	1	设置(上升沿)	采样(下降沿)
2	1	0	采样(下降沿)	设置(上升沿)
3	1	1	设置(下降沿)	采样(上升沿)

图 18–5. SPI 在 CPHA=0 时从机传送格式

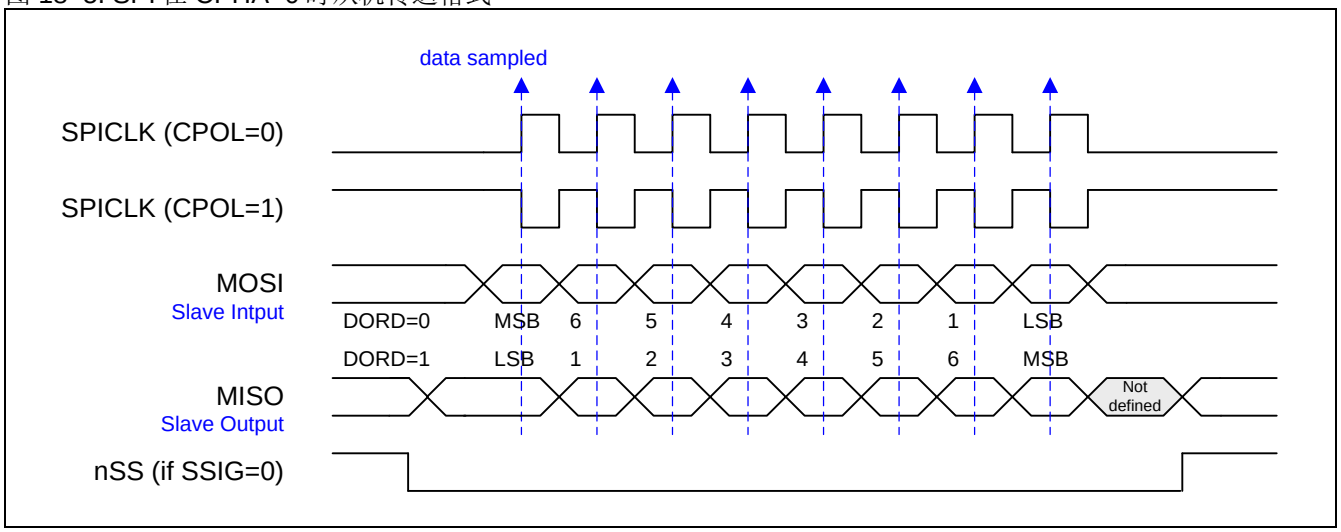


图 18–6. SPI 在 CPHA=1 时从机传送格式

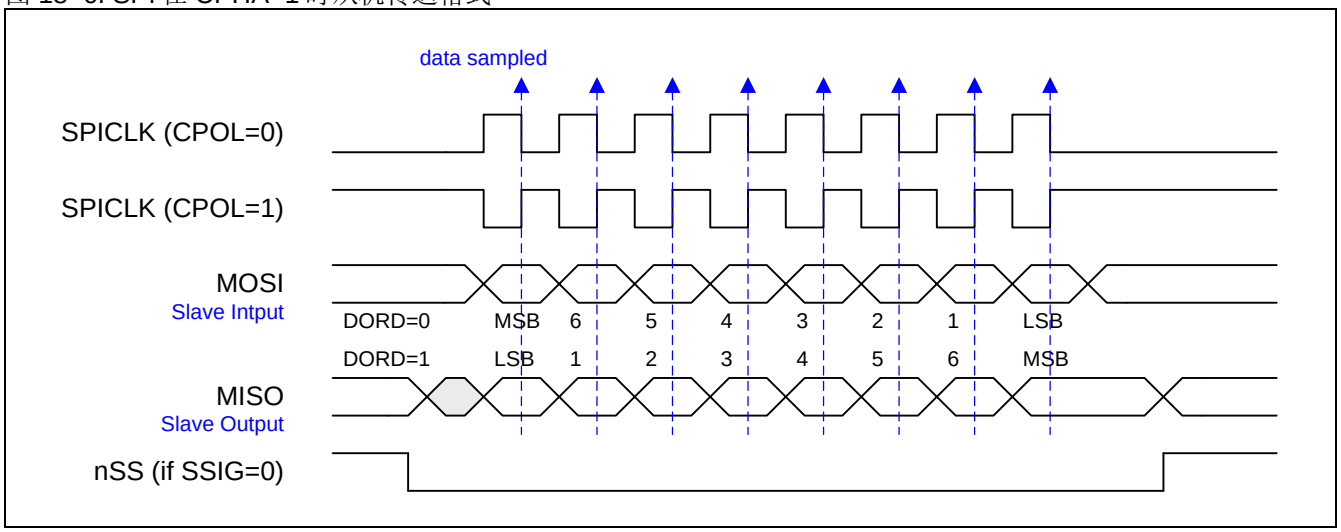


图 18–7. SPI 在 CPHA=0 时主机传送格式

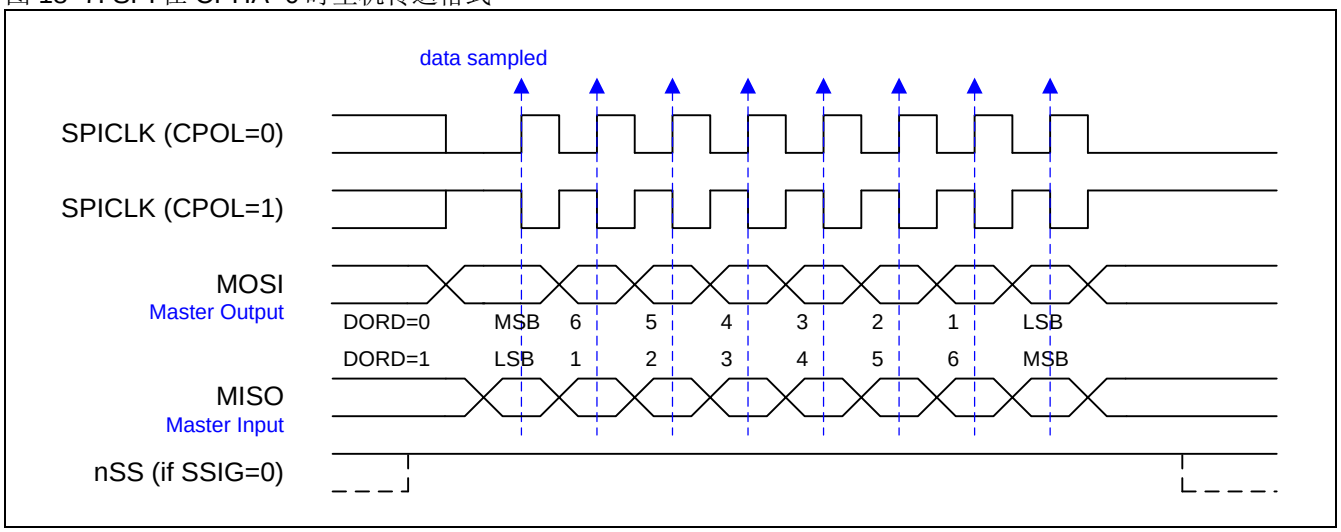
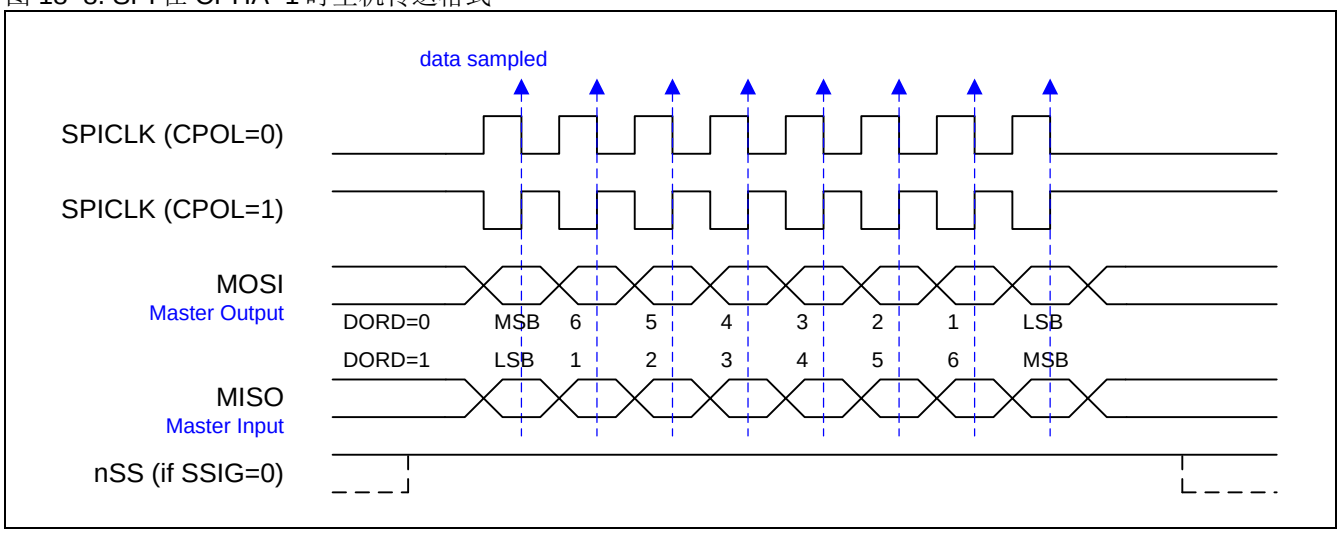


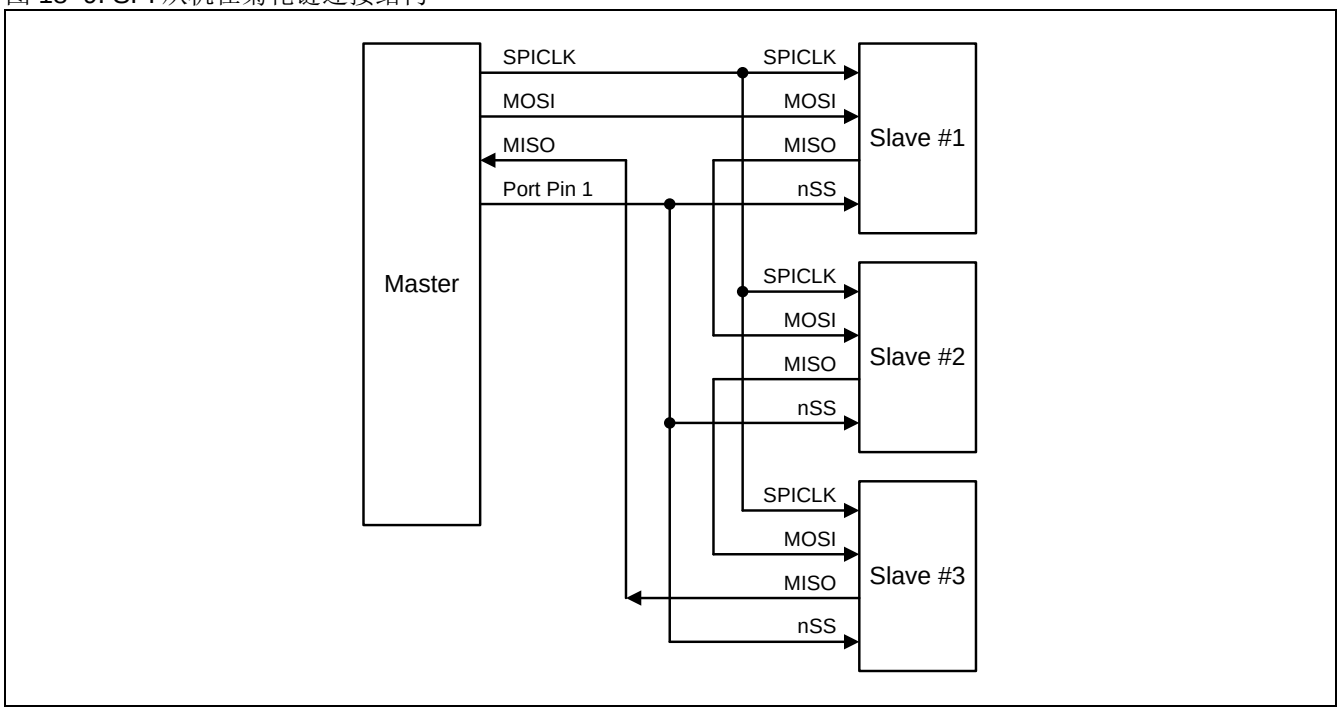
图 18–8. SPI 在 CPHA=1 时主机传送格式



18.4. 菊花链连接

如果 SPI0 被定义为从机模式，它可以连接成一个菊花链结构。第一个的从机的输出连接到第二个从机的输入，第二个从机的输出连接到第三个从机的输入，以此类推。最后一个从机的输出连接到主机的输入。在第二组时钟脉冲期间，每个从机发送出与第一组时钟脉冲接收到的信号完全相同的信号。整个链作为一个大的通信移位寄存器。菊花链功能仅需从主机接一条从机片选线 (nSS)。

图 18–9. SPI 从机在菊花链连接结构



18.4.1. 菊花链配置

如何 SPI 从机为菊花链

- 配置 SPCON 去定义数据模式和选择 SPI0 为从机模式
- 置位 SPI0M0 (AUXR7.4) 使能 SPI0 为菊花链模式
- 查询 SPIF 获得菊花链通信

18.5. SPI 寄存器

下面是 SPI 操作的相关特殊功能寄存器：

SPCON: SPI 控制寄存器

SFR 页 = 0~F

SFR 地址 = 0x85 复位值 = 0000-0100

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

Bit 7：SSIG，忽略 nSS。
0：nSS 引脚决定该设备是主机还是从机。
1：MSTR 位决定该设备是主机还是从机。

Bit 6：SPEN，SPI 使能。
0：SPI 接口禁用，所有 SPI 引脚可作为通用 I/O 口使用。
1：SPI 使能。

Bit 5：DORD，SPI 数据位序。
0：传送数据时高位在先(MSB)。
1：传送数据时低位在先(LSB)。

Bit 4：MSTR，主机/从机模式选择。
0：SPI 从机模式。
1：SPI 主机模式。

Bit 3：CPOL，SPI 时钟极性选择。
0：SPICLK 空闲是为低电平。SPICLK 时钟脉冲前沿是上升沿，而后沿是下降沿。
1：SPICLK 空闲是为高电平。SPICLK 时钟脉冲前沿是下降沿，而后沿是上升沿。

Bit 2：CPHA，SPI 时钟相位选择。
0：/SS 引脚低电平 (SSIG=0)开始放数据并在 SPICLK 后沿改变数据。数据在 SPICLK 的前沿采样。
1：SPICLK 脉冲前沿放数据，后沿采样。
(注：如果 SSIG=1，CPHA 必须不为 1，否则这个功能是没有定义的。)

Bit 1~0：SPR1-SPR0, SPI 时钟率选择位 0 和 1 (主机模式，与 SPR2 配合使用)

SPR2	SPR1	SPR0	SPI 时钟选择	SPI 时钟率@ SYSCLK=12MHz	SPI 时钟率@ SYSCLK=24MHz
0	0	0	SYSCLK/4	4 MHz	6 MHz
0	0	1	SYSCLK/8	2 MHz	3 MHz
0	1	0	SYSCLK/16	1 KHz	1.5 MHz
0	1	1	SYSCLK/32	500 KHz	750 KHz
1	0	0	SYSCLK/64	250 KHz	375 KHz
1	**0**	**1**	**SYSCLK/2**	8 MHz	12 MHz
1	1	0	**S0TOF/6**	可变	可变
1	1	1	**T0OF/6**	可变	可变

注意：

1.SYSCLK 是系统时钟。

2.S0TOF 是 UART0 波特率发生器溢出。

3. T0OF 是定时器 0 溢出。

SPSTAT: SPI 状态寄存器

SFR 页 = 0~F

SFR 地址 = 0x84 复位值 = 0000-XXX0

7	6	5	4	3	2	1	0
SPIF	WCOL	THRF	SPIBSY	MODF	0	0	SPR2
R/W	**R/W**	**R**	**R**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：SPIF，SPI 传输完成标志。
0：**软件写“1”SPIF 清零。**
1：当一次串行传输完成时，SPIF 位置位，同时若 SPI 中断允许，会产生一个中断。若 nSS 引脚在主机模式下被拉低且 SSIG=0，SPIF 位也会置位以表明“模式改变”。

Bit 6：WCOL，SPI 写冲突标志。
0：**软件写“1” WCOL 清零。**
1：SPI 数据寄存器(SPDAT)在数据传输过程中被写入此位置位(见章节“18.2.5 写冲突”)。

Bit 5：THRF，发送保持寄存器(THR)非空标志。只读。
0：表明 THR 是“空的”。当 THR 为空时此位被硬件清零，这意味着 THR 中的数据已经被装入移位输出寄存器进行发送，而现在用户可以向 SPDAT 写下一个要发送的数据。
1：表明 THR 是“非空”。当软件向 SPDAT 写数据时由硬件置位。

Bit 4：SPIBSY，SPI 忙标志。只读。.
0：表示 SPI 是空闲状态并且所有的移位寄存器是空的。
1：置位表示 SPI 传输进行中(主机或从机)。

Bit 3：MODF，模式错误标志。当检测到主机模式冲突时(nSS 为低电平，MSTEN=1 并且 SSIG=0)，硬件置该位为 1。如果中断使能，就会产生一个中断。该位不会由硬件自动清零，必须由软件写“1”清零。

Bit 2~1：保留位，写寄存器时，此位必须写“0”。

Bit 0：SPR2, SPI 时钟率选择位 2(与 SPR1 和 SPR0 相配合)

SPDAT: SPI 数据寄存器

SFR 页 = 0~F

SFR 地址 = 0x86 复位值 = 0000-0000

7	6	5	4	3	2	1	0
(MSB)							(LSB)
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

SPDAT 有两个物理缓冲器供发送和接收过程中各自独立写入和读取。

AUXR7:辅助寄存器 7

SFR 页 = **仅 4 页**

SFR 地址 = 0xA4 复位值 = 1100-0000

7	6	5	4	3	2	1	0
1	1	C0CKOE	SPI0M0	0	0	0	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 4：SPI0M0，SPI0 模式控制位 0。控制 SPI 的菊花链连接。
0：禁止这个模式控制。
1：使能这个模式控制。

AUXR10: 辅助寄存器 10

SFR 页 = **仅 7 页**

SFR 地址 = 0xA4 复位值 = 1100-0000

7	6	5	4	3	2	1	0
1	AC0HC0	0	SPIPS0	S0PS1	SPFACE	TWICF	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 4：SPIPS1~0, SPI 引脚选择位 [1:0]。

SPIPS0	nSS	MOSI	MISO	SPICLK
0	P4.6	P4.4	P4.5	P3.3
1	P3.0	P4.4	P3.1	P3.3

Bit 2：SPFACE, SPIF 自动清零使能位。

0：禁止，SPIF 只能软件清零。

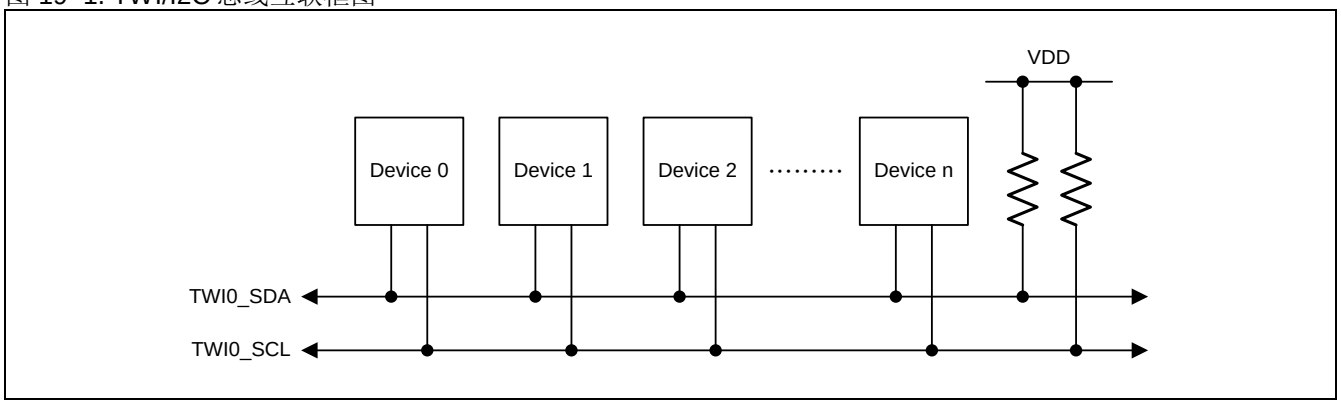
1：使能，SPIF 会被 CPU ***读/写*** SPDAT 操作清零。

19. 双线串行接口(TWI0/I2C0)

双线串行接口是一个双线双向总线。双线串行接口(TWI)很适合于典型的处理器应用。**MG82F6B08 / 6B001/ 6B104** 内嵌 1 个 TWI/I2C。支持多从机地址识别。

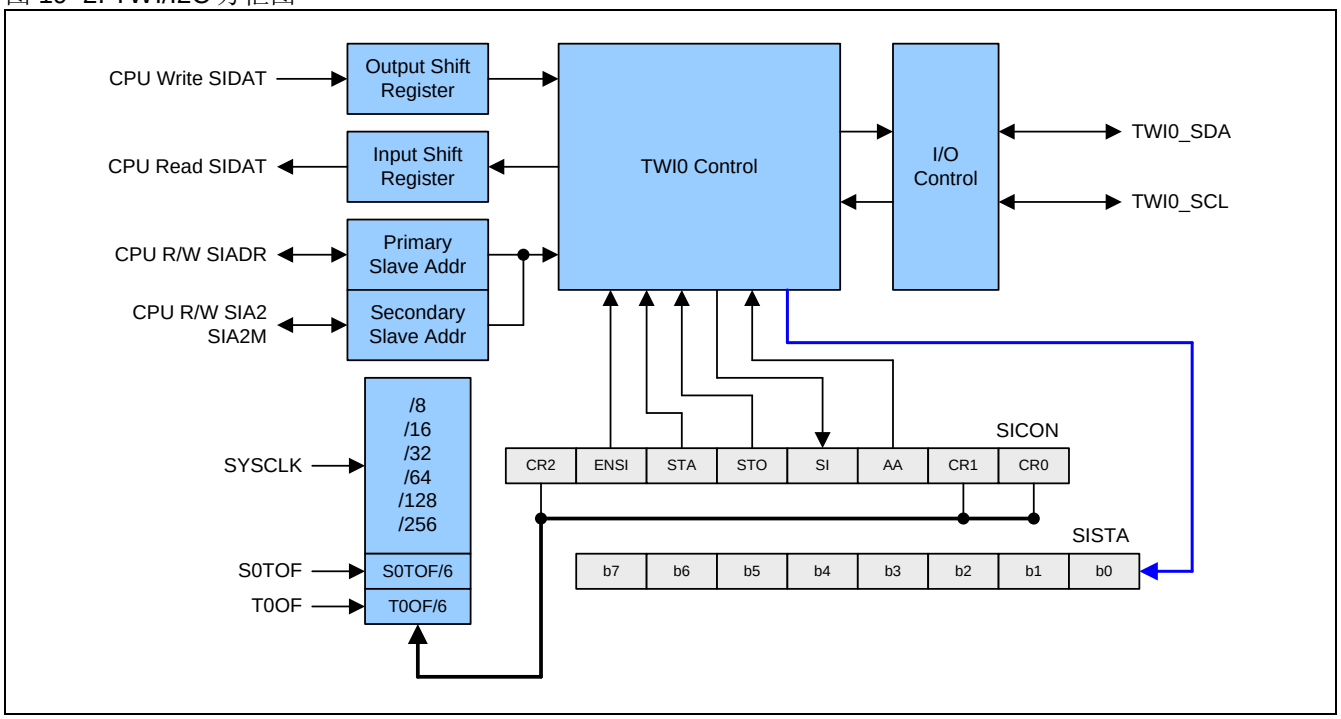
TWI/ I2C 协议允许系统设计人员只用两根双向传输线来连接多达 128 个不同的设备，一根用于时钟(TWI0_SCL)，一根用于数据(TWI0_SDA)。双线串行接口(TWI)由 TWI0_SDA(串行数据)、TWI0_SCL(串行时钟)控制产生和同步，仲裁逻辑以及起始/停止(START/STOP)。唯一需要的外部硬件就是在 TWI/ I2C 的每根传输线上添加一个上拉电阻。所有连接到总线的设备都有各自的地址，而且 TWI/ I2C 协议解决了总线仲裁的问题。

图 19–1. TWI/I2C 总线互联框图



TWI/ I2C 总线可以操作在主机或从机也可以是多主机。CPU 通过 SICON(串行接口控制寄存器)、SISTA(串行接口状态寄存器)、SIDAT(串行接口数据寄存器，用于发送和接收 TWI/ I2C 数据)、SIADR(串行接口从机地址寄存器)这四个特殊功能寄存器与 TWI 相连。TWI 硬件通过两根数据线与串行总线相连：SDA(串行数据线)和 SCL(串行时钟线)。

图 19–2. TWI/I2C 方框图



19.1. 操作模式

TWI/ I2C 有 4 种操作模式：1) 主机/发送模式，2)主机/接收模式，3)从机/发送模式，4)从机/接收模式。SI 软件清零之后 SICON 寄存器的位 STA、STO 和 AA 决定 TWI 硬件下一个执行的是哪一个操作。当下一个操作完成，SISTA 寄存器将更新一个新状态同时 SI 也会硬件置位。现在，中断服务程序会被调用(如果 TWI/ I2C 中断使能)，软件可以通过新的状态区分需要调用哪一个子程序。

19.1.1. 主机发送模式

在主机发送模式，一些数据字节可以发送到一个从机接收器。在进入主机发送模式前，SICON 必须作如下设置：

SICON

7	6	5	4	3	2	1	0
CR2	ENSI	STA	STO	SI	AA	CR1	CR0
位速率	1	0	0	0	x	位速率	

CR0、CR1 和 CR2 定义了串行位速率。ENSI 必须设置为逻辑 1 来使能 TWI/ I2C。如果 AA 位复位，在其它设备成为总线的主机时，TWI/ I2C 将不会应答它自身的从机地址或广播地址。也就是说，如果 AA 复位，TWI/ I2C 不能进入从机模式。STA、STO 与 SI 必须复位。

置位 STA 也许可以立即进入主机发送模式。TWI/ I2C 逻辑将检测串行总线并且在总线空闲时产生一个 START 信号。发送完 START 信号后，串行中断标志(SI)将被置位，并且状态寄存器(SISTA)中的状态编码将为 08H。这个状态编码必须用于指示一个中断服务程序加载从机地址和数据方向位(SLA+W)到 SIDAT。SICON 的 SI 位必须清零，串行传输才能继续进行。

当从机地址与方向位发送完，并且接收到一个应答位后，串行中断标志(SI)会再次被置位。SISTA 可能为以下的编码：在主机模式为 18H，20H 或 38H，如果从机模式使能(AA=1)，也可以为 68H，78H 或 B0H。在这些状态编码下对应的操作将在随后的工作流程图中详细叙述。在一个重复的 START 信号后(状态编码 10H)，TWI 可以通过向 SIDAT 写入 SLA+R 进入主机接收模式。

19.1.2. 主机接收模式

在主机接收模式，可以从从机发送器接收一定数量的字节数据。SICON 也必须如主机发送模式一样初始化。开始信号发送后，中断服务程序必须向 SIDAT 写入 7 位从机地址与数据方向位(SLA+R)。SICON 的 SI 位必须清零，串行传输才能继续进行。

在从机地址与数据方向位发送完并且接收到应答位后，串行中断标志(SI)重新置位。SISTA 可能为以下的编码：在主机模式为 40H，48H 或 38H，如果从机模式使能(AA=1)，也可以为 68H，78H 或 B0H。这些状态编码下对应的操作将在随后的工作流程图中详细叙述。在一个重复的 START 信号后(状态编码 10H)，TWI 可以通过向 SIDAT 写入 SLA+W 进入主机接收模式。

19.1.3. 从机发送模式

在从机发送模式下，一些数据字节发送给主机接收。SIADR 和 SICON 必须如下初始化从机发送模式：

SIADR

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	GC

|<---------- Own Slave Address ---------->|

高 7 位是响应被主机寻址的 TWI/ I2C 地址。如果 LSB (GC)置位，TWI 将应答广播地址(00H)；否则将忽略广播地址。

SICON

7	6	5	4	3	2	1	0
CR2	ENSI	STA	STO	SI	AA	CR1	CR0
x	1	0	0	0	1	x	x

在从机模式下 CR0、CR1 和 CR2 不影响 TWI/ I2C。ENSI 必须置位去使能 TWI/ I2C。AA 必须置位去使能 TWI 应答自己的从机地址或广播地址。STA、STO 和 SI 必须清零。

当 SIADR 和 SICON 初始化之后，TWI/ I2C 会等待直到其从机地址被寻址并且数据方向为“1”(R)，TWI/ I2C 将工作于从机发送模式。在接收到自身的从机地址以及“R”位后，串行中断标志(SI)置位，并且可以从 SISTA 读出一个可用的状态编码。这些状态编码可以用作指示一个中断服务程序，在这些状态编码下对应的操作将在随后的工作流程图详细叙述。当 TWI/ I2C 处于主机模式时，如果仲裁失败也可能进入从机发送模式(参考 B0H 状态)。

如果在一次传输的过程中 AA 位复位，TWI/ I2C 将发送完当前字节的数据后进入 C0H 或 C8H 状态。TWI 会转换到未被寻址从机模式，如果主机继续传输，TWI/ I2C 将会忽略主机接收器，因此主机总是接收到“1”。当 AA 复位时，TWI/ I2C 不会回应其从机地址或广播地址，但是会继续监测串行总线。在任何时候可以通过置位 AA 恢复，这意味着 AA 位可用于暂时从总线中隔离 TWI/ I2C。

19.1.4. 从机接收模式

在从机接收模式，会从主机发送器接收一些数据字节。数据传送的初始化与从机发送模式一样。

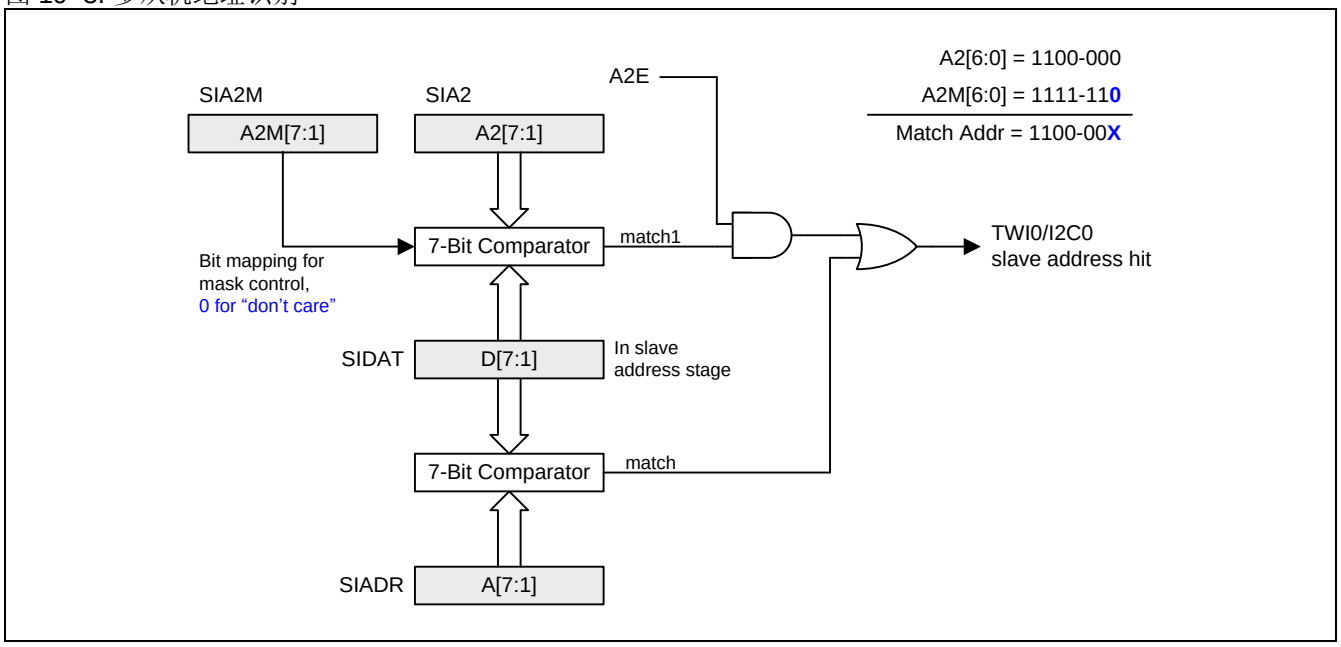
SIADR 与 SICON 初始化后，TWI/ I2C 会等待直到其从机地址被寻址并且数据方向为“0”(W)，TWI/ I2C 将工作于从机接收模式。在接收到其从机地址与“W”位后，串行中断标志(SI)置位，并且可以从 SISTA 读出一个可用的状态编码。这些状态编码可以用作指示一个中断服务程序，在这些状态编码下对应的操作将在随后的工作流程图详细叙述。当 TWI/ I2C 处于主机模式时，如果仲裁失败可能进入从机接收模式(参考状态 68H 和 78H)。

如果在一次传输的过程中 AA 位被复位，TWI/ I2C 会在接收到下一个字节后回复 NACK(逻辑 1)。当 AA 复位时，TWI/ I2C 不会响应自身的从机地址或广播地址，但是会继续监测串行总线。在任何时候可以通过置位 AA 恢复，这意味着 AA 位可用于暂时从总线中隔离 TWI/ I2C。

19.1.5. 多从机地址识别

在 **MG82F6B08 / 6B001/ 6B104** 中 SIADR 定义 TWI/I2C 的首从地址。**MG82F6B08 / 6B001/ 6B104** 还提供带有掩码功能的从地址，该掩码功能在 SIA2 和 SIA2M 上实现。在从地址掩码 SIA2M[7:1]的位元位置上的 1，使能所接收的从地址与这些位元的从地址 SIA2[7:1]之间进行比较。从地址掩码中的一个 0 表示该位将在比较中被“忽略”。在这种情况下，收到的从机地址在该位上可以是 1 或 0。

图 19–3. 多从机地址识别



19.2. 混合状态

有两个 SISTA 编码没有与已经定义的 TWI/ I2C 硬件状态对应，描述如下：

S1STA = F8H:
这个状态编码表明还没有相应的信息可用，因为串行中断标志(SI)还没有置位。这种情况发生在状态转换之间和 TWI/ I2C 未涉及串行传输时。

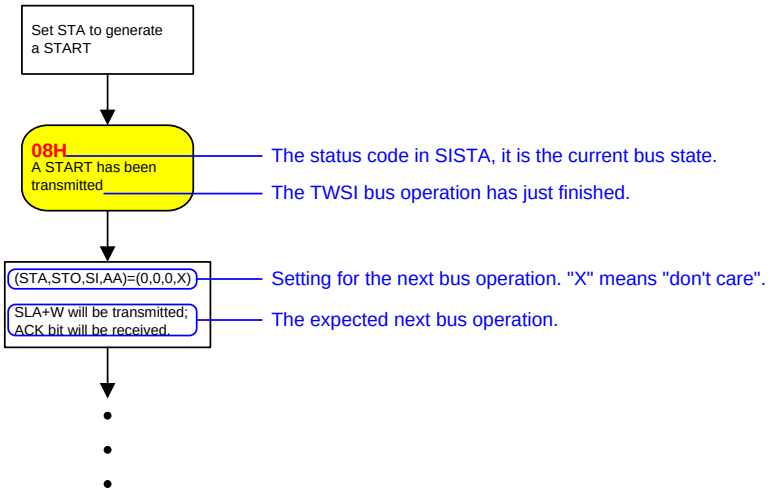
S1STA = 00H:
这个状态编码表明在一个 TWI/ I2C 串行传输过程中发生总线错误。当一个 START 或 STOP 信号在一帧的非法位置发送时，总线错误就会发生。例如：在传输一个字节地址、数据时，或者在应答位。总线错误也会在外界干扰扰乱内部 TWI/ I2C 信号时发生。当总线错误发生时，SI 被置位，STO 标志必须置位并且 SI 必须软件清零用来从总线错误中恢复。这会使 TWI/ I2C 进入“未被寻址”的从机状态(已定义的状态)并且清除 STO 标志(SICON 的其它位不受影响)。TWI/ I2C0_SDA 与 TWI/ I2C0_SCL 线将被释放(不会发送 STOP 信号)。

19.3. 使用 TWI/I2C

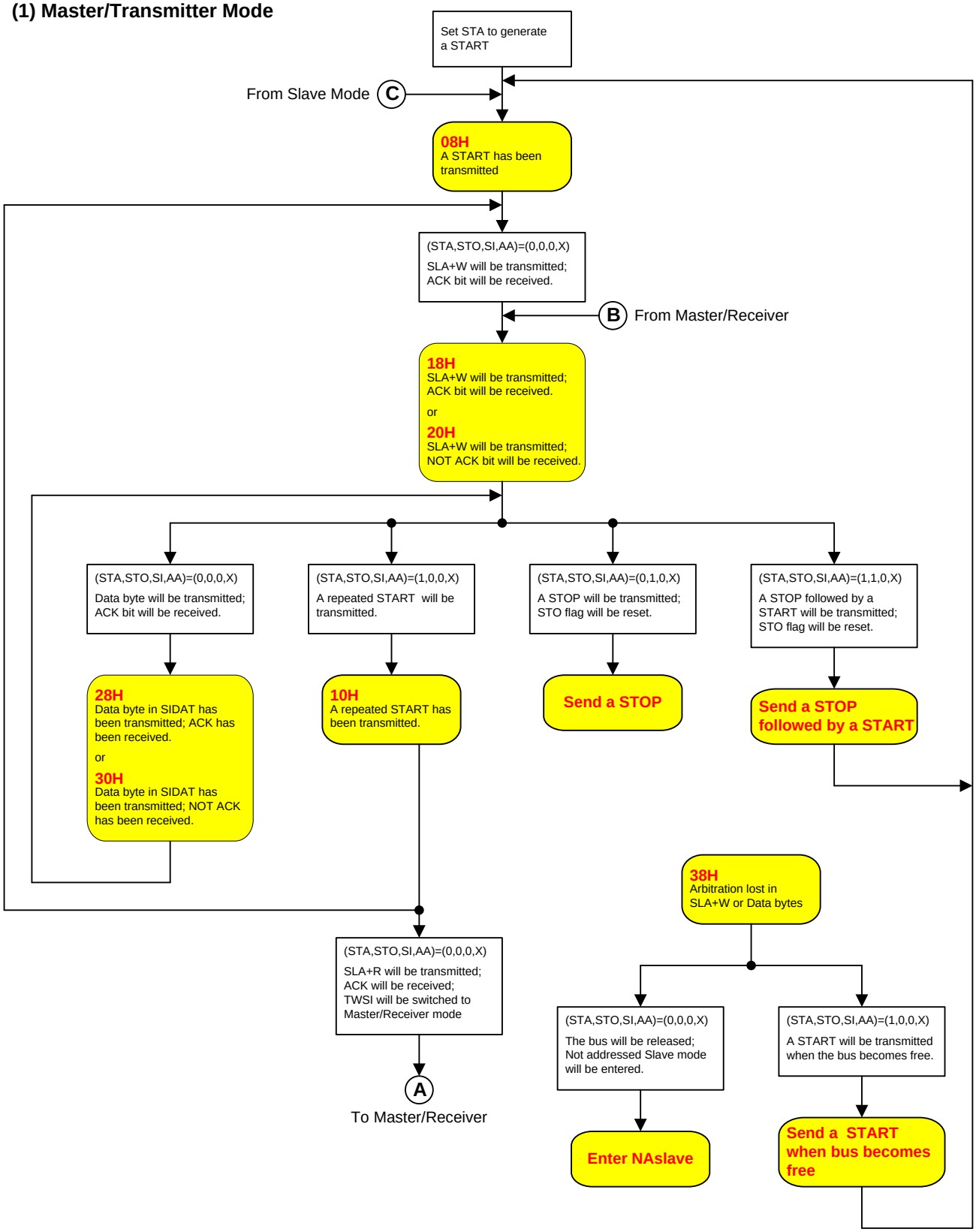
TWI/ I2C 是面向字节并且基于中断的。中断会在所有总线事件后发生，例如接收到一字节数据或发送 START 信号。因为 TWI/ I2C 是基于中断的，应用软件可以自由的在一个 TWI/ I2C 字节发送的过程中处理其它工作。注意，TWI/ I2C 中断标志位(EIE1.6)与 EA 位允许应用程序选择在 SI 标志出现时是否产生中断请求。当 SI 标志出现时，表明 TWI/ I2C 已经完成一个操作并且等待程序响应。此时状态寄存器 SISTA 保存的状态编码表明 TWI/ I2C 总线的当前状态。用户程序可以通过对 STA，STO 和 AA 位(在 SICON 中)进行适当的编程来决定接下来 TWI/ I2C 总线将如何运行。

下面的操作流程图将指导用户通过“状态到状态”(state-by-state)的操作来使用 TWI/ I2C。首先，用户应该向 SIADR 写入自身的从机地址(参考前面对 SIADR 的描述)。作为主机时，在初始 SICON 后，第一步为置位“STA”来向总线产生一个 START 信号。作为从机时，在初始化 SICON 后，TWI/ I2C 等待直到被寻址。然后参考操作流程图对 SICON 的 STA，STO，SI，AA 位进行适当的编程来进行后续动作。当 SI 清零后 TWI 硬件就会进行下一步动作，因此推荐使用如下两个步骤：先对 STA，STO 与 AA 编程，然后清零 SI 位(可以使用“CLR SI”指令)来进行可靠的操作。

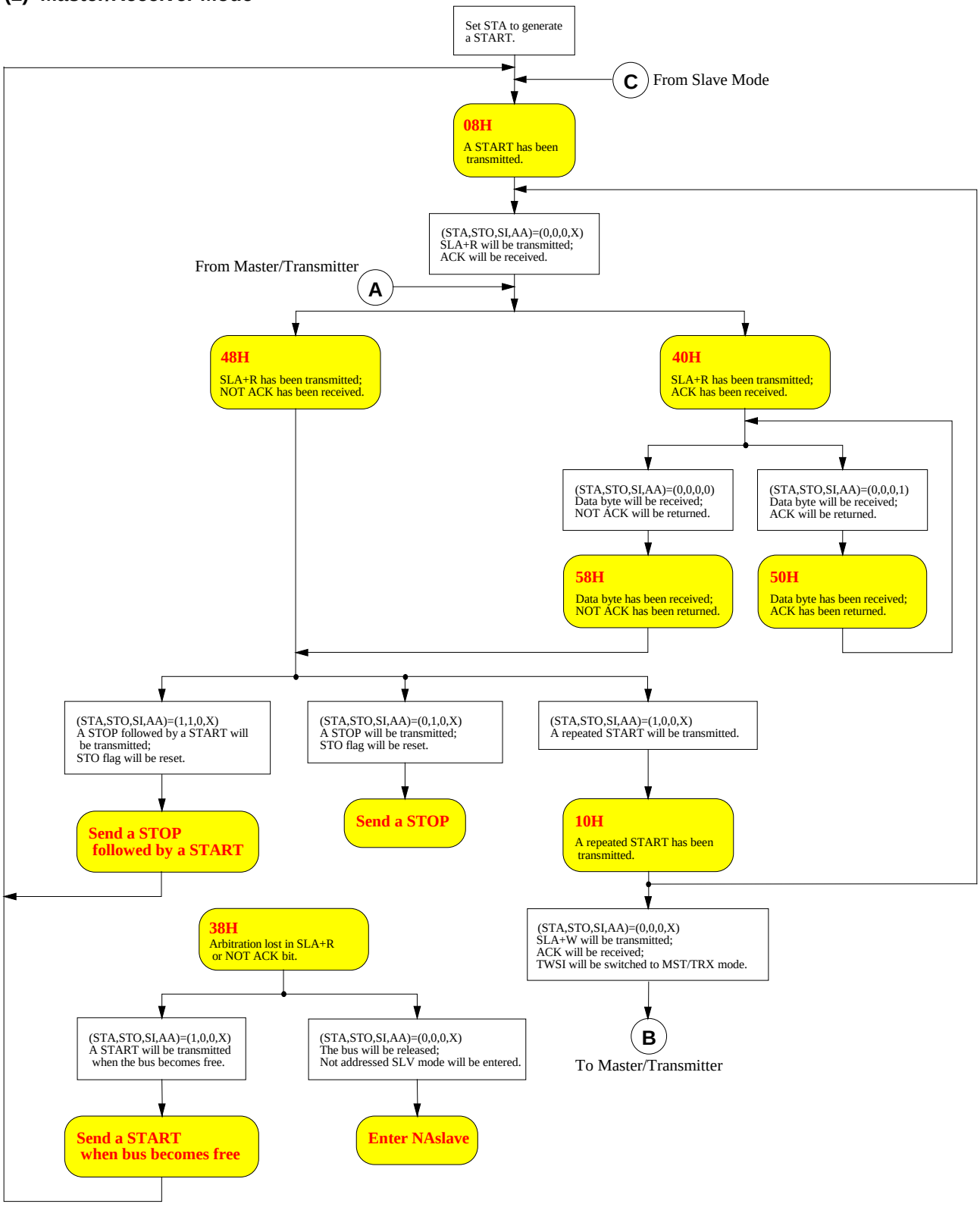
下面的图指出如何读流程图



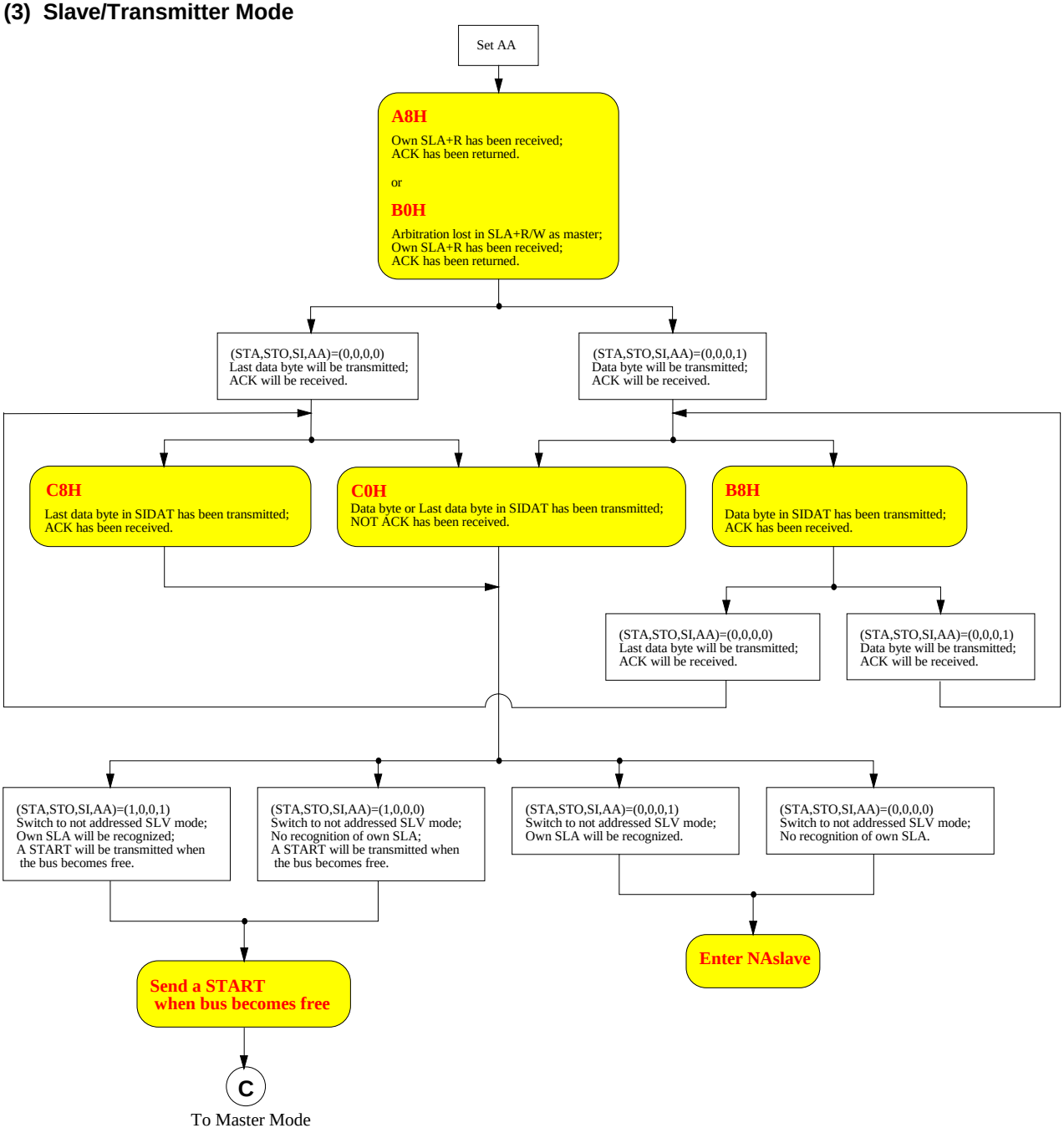
(1) Master/Transmitter Mode



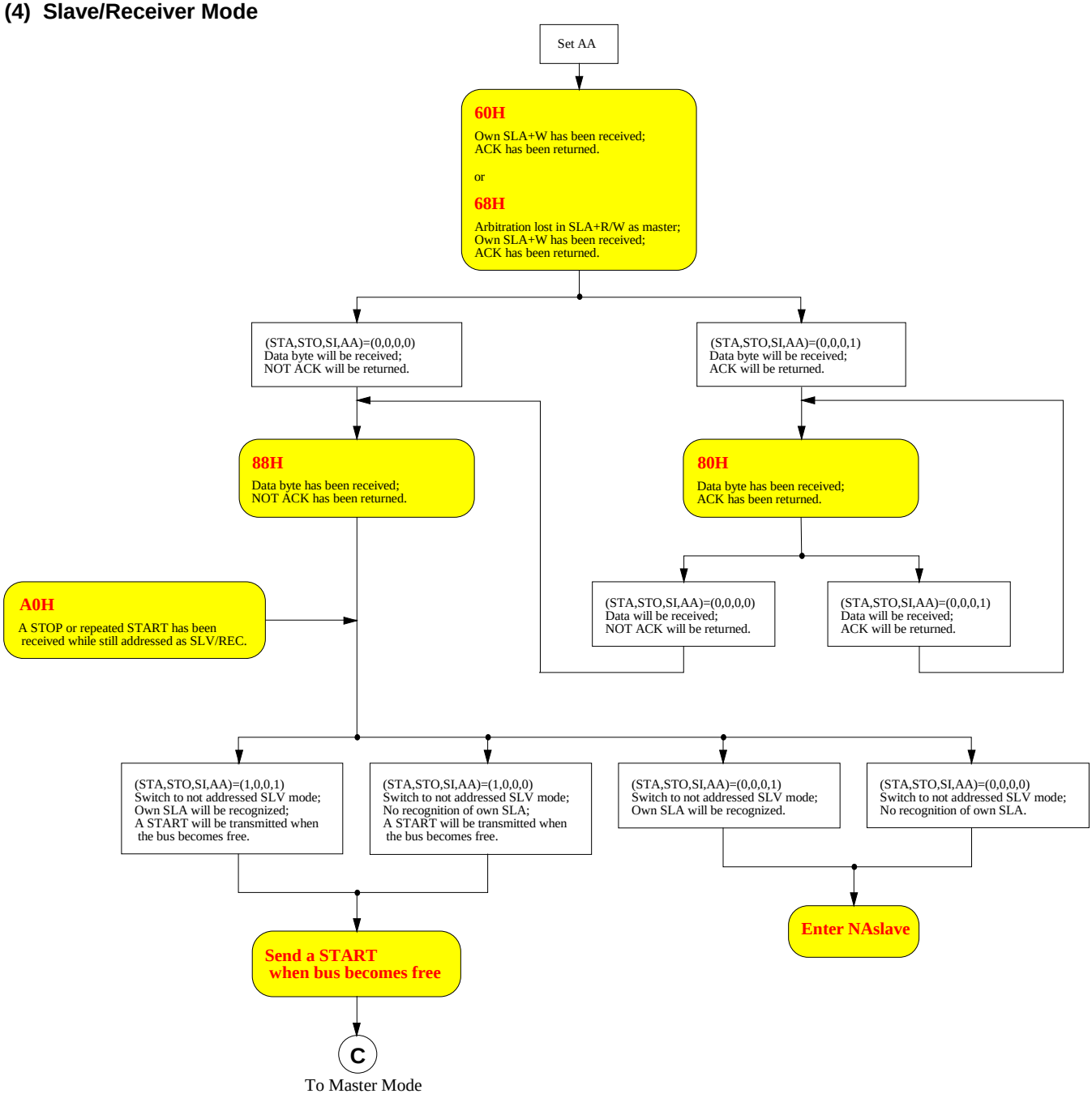
(2) Master/Receiver Mode



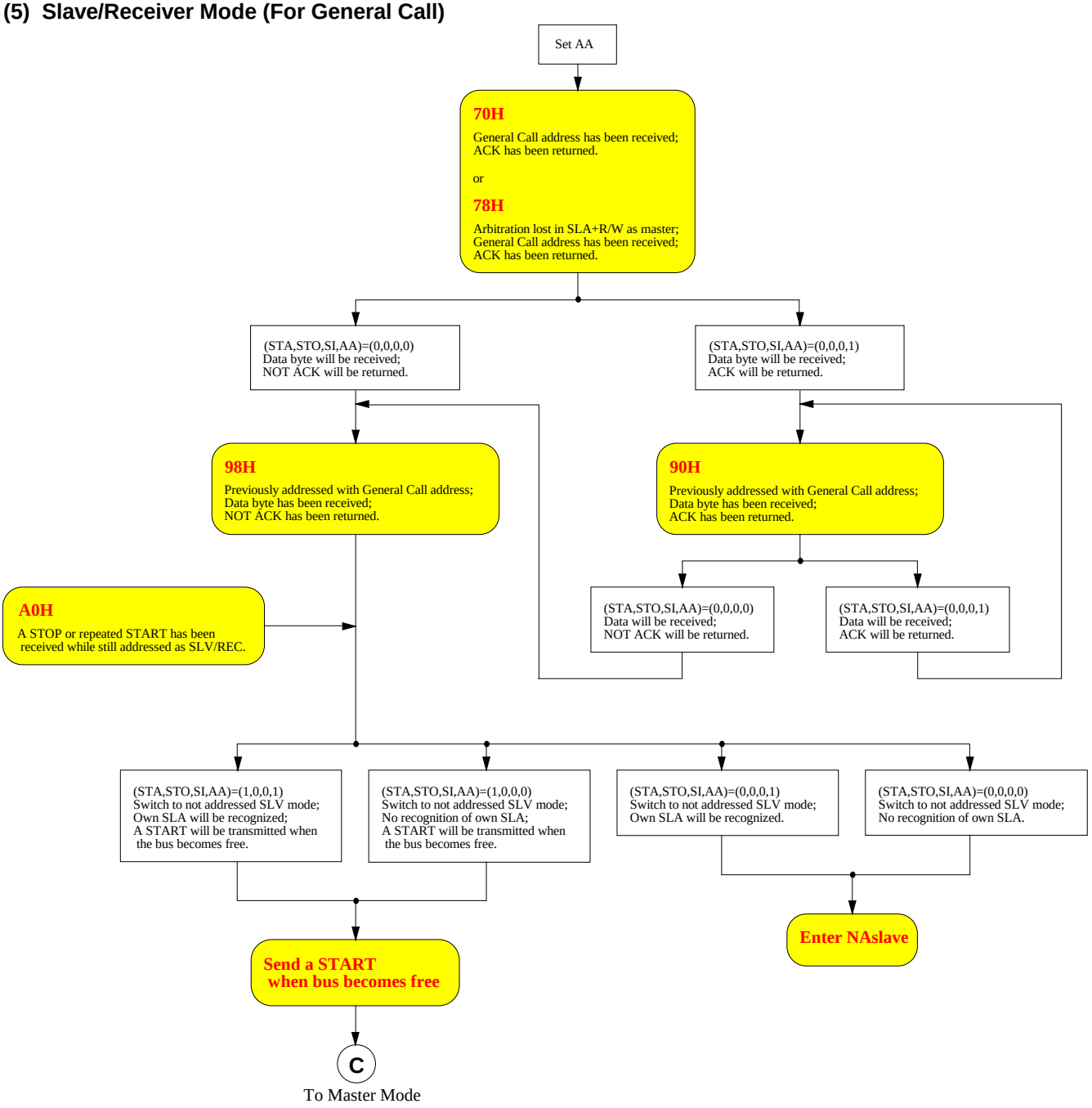
(3) Slave/Transmitter Mode



(4) Slave/Receiver Mode



(5) Slave/Receiver Mode (For General Call)



19.4. TWI0/I2C0 寄存器

SIADR: TWI0/I2C0 地址寄存器

SFR 页 = **仅 0 页**

SFR 地址 = 0xD1 复位值 = 0000-0000

7	6	5	4	3	2	1	0
A6	A5	A4	A3	A2	A1	A0	GC
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

CPU 可以直接对此寄存器进行读写。SIADR 不受 TWI0/ I2C0 硬件的影响。当 TWI0/ I2C0 处于主机模式时此寄存器的值会被忽略。当处于从机模式时，寄存的高七位必须被用于本机的从机地址，并且当最低位(GC)置位时，广播地址(00H)会被识别，否则忽略。在 START 状态后，最高位与从 TWI0/ I2C0 总线上收到的首位相对应。

SIDAT: TWI0/I2C0 数据寄存器

SFR 页 = **仅 0 页**

SFR 地址 = 0xD2 复位值 = 0000-0000

7	6	5	4	3	2	1	0
SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

此寄存器保存着一字节将要发送或者刚接收到的数据。在没有进行移位工作时，CPU 可以直接对此寄存器进行读写。这种情况发生在 TWI0/ I2C0 正处于一个被定义的状态并且串行中断标志位(SI)置位。只要 SI 被置位，SIDAT 中的数据总是保持稳定的。在数据被移出时，总线上的数据同时移入，SIDAT 总保存着总线上出现的最后一个字节数据。因此在仲裁失败时，主机发送切换为从机接收的过程会在 SIDAT 中产生一个正确的数据。

SIDAT 与 ACK 标志位组成一个 9 位的移位寄存器，可以在移入或移出一个 8 位的数据后，跟随一个应答位。ACK 标志由 TWI0 硬件控制，CPU 访问不到。串行数据在 TWI0/ I2C0_SCL 的上升沿移入 SIDAT 寄存器。当一字节的数据完全移入 SIDAT 后，SIDAT 中的数据将是可以用的，并且控制逻辑会在第 9 个时钟周期返回一个应答位。串行数据在 TWI0/ I2C0_SCL 的下降沿从 SIDAT 寄存器移出。

CPU 向 SIDAT 写入数据后，SD7 位将首先出现在 SDA 线上。9 个时钟周期后，SIDAT 中的 8 位数据将被发送完成，并且通过应答位返回 ACK 标志。注意发送出去的 8 位数据会移回 SIDAT。

SICON: TWI0/I2C0 控制寄存器

SFR 页 = **仅 0 页**

SFR 地址 = 0xD4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
CR2	ENSI	STA	STO	SI	AA	CR1	CR0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

CPU 可以直接读写此寄存器。其中两个位会受 TWI0/I2C0 硬件的影响：SI 位会在串行中断请求时置位，STO 位会在总线出现 STOP 状态时清零。STO 位也会在 ENS1=0 时清零。

Bit 7：CR2，TWI0/ I2C0 时钟率选择位 2 (与 CR1 和 CR0 一起使用)。

Bit 6：ENSI，TWI0/I2C0 硬件使能位。
ENSI 为“0”时，TWI0/I2C0_SDA 与 TWI0_SCL 输出为高阻态，TWI0/I2C0_SDA 与 TWI0/I2C0_SCL 输入信号被忽略，TWI/I2C0 处于未寻址从机状态，SICON 的 STO 位被强制置为“0”，但不影响其它位。TWI0/I2C0_SDA 与 TWI0/I2C0_SCL 可用作通用 I/O 引脚。ENSI 为“1”时，TWI0/I2C0 使能，TWI0/I2C0_SDA 和 TWI0/I2C0_SCL 端口锁存器(比如 P4.1 和 P4.0)必须设置为逻辑 1 并且 I/O 模式必须配置成开漏模式以用于接下来的串行通讯。

Bit 5：STA，开始(START)标志。
当 STA 位被置位进入主机模式时，TWI0/I2C0 硬件将检查串行总线的状态，若总线空闲将产生一个 START 信号。若总线忙，TWI0/I2C0 将等待 STOP 信号出现并且在一个延迟后产生 START 信号。如果 STA 在 TWI0/I2C0 已经是处于主机模式并且一个或多个字节已被发送或接收的情况下置位，TWI0/I2C0 会发送一个重复的 START 信号。STA 可以在任何时候置位，也可以在 TWI0/I2C0 是一个被寻址的从机时置位。当 STA 位复位时，无 START 或重复的 START 信号产生。

Bit 4：STO，停止(STOP)标志。
当 TWI0/I2C0 处于主机模式时，置位 STO 会向串行总线发送一个 STOP 信号。当在总线上侦测到 STOP 信号时，TWI0/I2C0 硬件清除 STO 标志。在从机模式时，置位 STO 标志可从总线错误状态恢复。在这种情况下不会向总线发送 STOP 信号，但是 TWI0/I2C0 硬件表现就像已经接收到一个 STOP 信号，并且转换到未被寻址的从机接收模式。STOP 标志自动被硬件清零。如果 STA 与 STO 位同时置位，若 TWI0/I2C0 处于主机模式将产生一个 STOP 信号(当处于从机模式时将产生一个内部的 STOP 信号，但不发送)，接着发送一个 START 信号。

Bit 3：SI，串行中断标志。
当一个新的 TWI0/I2C0 状态出现在 SISTA 寄存器时，SI 标志会被硬件置位。如果 TWI0/I2C0 中断允许，中断服务程序将会运行。唯一不会使 SI 置位的状态是指出没有相关状态信息可以获得的 F8H。当 SI 置位时，TWI0/I2C0_SCL 线上的低电平会延长，并且串行传输暂停。TWI0/I2C0_SCL 线上的高电平不受串行中断 SI 标志影响。SI 必须由软件写“0”清零。SI 标志复位时不会产生中断请求，TWI0/I2C0_SCL 线上的时钟也不会延长。

Bit 2：AA，确认应答标志。
如果 AA 标志设为“1”，一个应答 ACK(TWI0/I2C0_SDA 低电平)将在 TWI0/I2C0_SCL 的应答时钟周期内回复，当：
1)接收到本机的从机地址。
2) TWI0/I2C0 处于主机/接收模式时，接收到一字节的数据。
3)TWI0/I2C0 处于已被寻址的从机/接收模式时，接收到一字节的数据。

如果 AA 标志设为“0”，一个无应答 NACK(TWI0/I2C0_SDA 高电平)将在 TWI0/I2C0_SCL 的应答时钟周期内回复，当：
1)TWI0/I2C0 处于主机/接收模式时，接收到一字节的数据
2) TWI0/I2C0 处于已被寻址的从机/接收模式时，接收到一字节的数据

Bit 7, 1~0：CR2、CR1 和 CR0，时钟率选择位
TWI0/I2C0 处于主机模式时，这三个位决定串行时钟频率。最高主机模式时钟频率可达 1MHz。当 TWI0/I2C0 处于从机模式时，时钟频率并不重要，因为 TWI0/I2C0 会自动同步任何主机的时钟频率，高达 400KHz。表 19–1 给出不同的时钟速率设置：

表 19–1. TWI0/I2C0 串行时钟速率

CR2	CR1	CR0	TWI0/ I2C0 时钟选择	TWI0/ I2C0 时钟率@ SYSCLK=16MHz
0	0	0	SYSCLK/8	2 MHz Note1
0	0	1	SYSCLK/16	1 MHz
0	1	0	SYSCLK/32	500 KHz
0	1	1	SYSCLK/64	250 KHz
1	0	0	SYSCLK/128	125 KHz
1	0	1	SYSCLK/256	75 KHz
1	1	0	S0TOF/6	可变
1	1	1	T0OF/6	可变

注意：
1. *最大 TWI0/I2C0 时钟速率在 1MHz 以下，设置 SYSCLK=8MHz 将产生 1MHz 时钟速率。*
2. *SYSCLK 是系统时钟。*
3. *S0TOF 是 UART0 波特率发生器溢出。*
4. *T0OF 是定时器 0 溢出。*

SISTA: TWI0/I2C0 状态寄存器
SFR 页 = **仅 0 页**
SFR 地址 = 0xD3 复位值 = 1111-1000

7	6	5	4	3	2	1	0
SIS7	SIS6	SIS5	SIS4	SIS3	SIS2	SIS1	SIS0
R	**R**	**R**	**R**	**R**	**R**	**R**	**R**

SISTA 是一个 8 位的只读寄存器。低三位总是为 0，高五位保存状态编码，可以组成多个可能的状态编码。当 SISTA 为 F8H 时，没有串行中断请求。SISTA 的其它值用于定义相应的 TWI0/ I2C0 状态。当进入这些状态的一种时，会请求串行中断(SI=1)。在 SI 硬件置位时，一个有效的状态编码会存于 SISTA 中。

另外，状态 00H 表示总线错误。当一个 START 或 STOP 信号在不符合规定的位置发送时会产生总线错误，如一个地址/数据的内部或者刚好在应答位上。

SIA2: TWI0/I2C0 2nd 地址寄存器

SFR 页 = 仅 **2** 页
SFR 地址 = 0xD1 复位值 = 0000-0000

7	6	5	4	3	2	1	0
A2.6	A2.5	A2.4	A2.3	A2.2	A2.1	A2.0	A2E
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~1：TWI0/I2C0 的 2nd 从机地址。

Bit 0：A2E, 2nd 从机地址识别使能控制。
0：禁止 2nd 从机地址识别。
1：使能 2nd 从机地址识别。

SIA2M: TWI0/I2C0 2nd 地址掩码寄存器

SFR 页 = 仅 **2** 页
SFR 地址 = 0xD2 复位值 = 1111-1111

7	6	5	4	3	2	1	0
A2M.6	A2M.5	A2M.4	A2M.3	A2M.2	A2M.1	A2M.0	1
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

SIA2 寄存器与 SIA2M 寄存器组合用于第二地址识别。实际上，SIA2M 是作为 SIA2 寄存器的掩码寄存器。示例如下：

SIA2[7:1] = 1100 000
SIA2M[7:1] = 1111 110
2nd ADR[7:1] = 1100 00x 第二从机地址将被检测，除了位 1 被“忽略”外。

Bit 0：保留位，写寄存器时，此位必须写“1”。

AUXR3:辅助寄存器 3

SFR 页 = 仅 **0** 页
SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	T0PS0	BPOC1	BPOC0	S0PS0	TWIPS1	TWIPS0	T0XL
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 2~1：TWIPS1~0, TWI0/I2C0 端口选择[1:0]。

TWIPS1~0	TWI0_SCL	TWI0_SDA
0 0	P3.1	P3.0
0 1	P4.4	P4.5
1 0	P3.0	P3.1
1 1	P3.3	P4.6

AUXR10: 辅助寄存器 10

SFR 页 = **仅 7 页**

SFR 地址 = 0xA4 复位值 = 1100-0000

7	6	5	4	3	2	1	0
0	AC0HC0	0	SPIPS0	S0PS1	SPFACE	TWICF	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 1：TWICF, TWI0/I2C0 串行时钟输入滤波。
0：禁止 TWICF 功能。
1：使能 TWICF 功能。

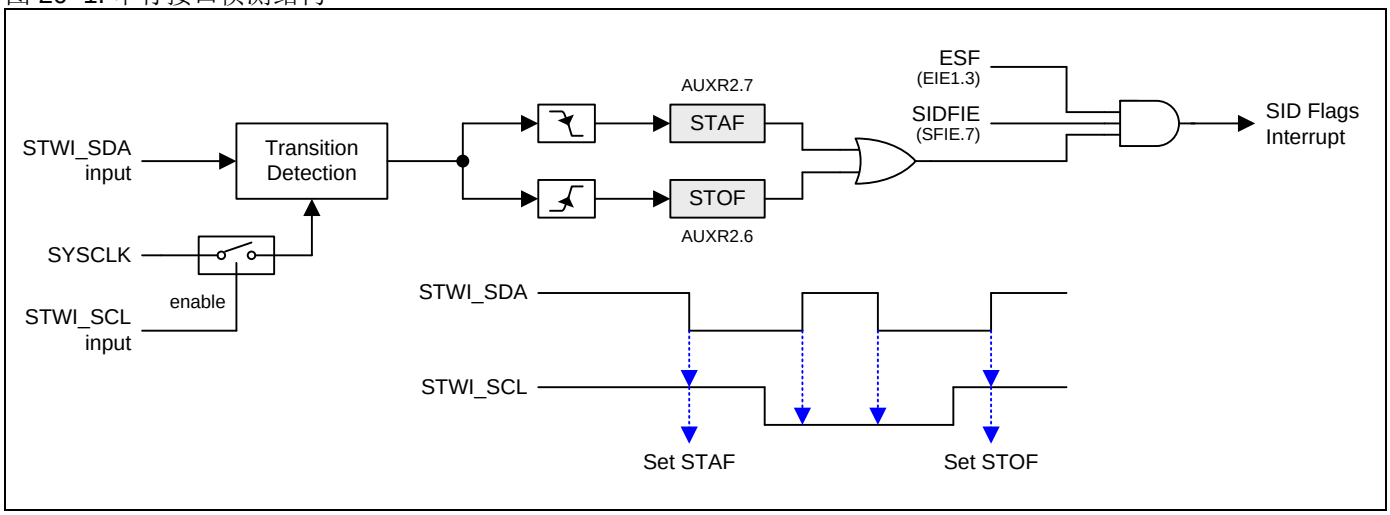
20. 串行接口侦测(STWI/SI2C)

串行接口侦测模块(SID)总是监控软件双线串行接口(STWI/SI2C)的“Start”和“Stop”状态。STWI _SCL 是串行时钟信号和 STWI _SDA 是串行数据信号。如果任何匹配条件被侦测到，硬件设置 STAF 和 STOF 标志位。软件可以决定这两个标志或设置 SIDFIE (SFIE.7) 与系统标志共享中断向量。并且 STWI _SCL 位于 nINT1 将帮助 MCU 通过 nINT1 中断来判断串行数据。软件可以说使用这些资源来实施一个可变的 TWI 从机设备。

20.1. SID 结构

STAF 和 STOF 侦测的结构，中断结构和事件侦测波形如图 20–1 所示。

图 20–1. 串行接口侦测结构



20.2. SID 寄存器

AUXR2:辅助寄存器 2

SFR 页 = 0~F

SFR 地址 = 0xA3 复位值 = 0000-0000

7	6	5	4	3	2	1	0
STAF	STOF	0	C0PLK	T1X12	T0X12	T1CKOE	T0CKOE
R/W	R/W	W	W	R/W	R/W	R/W	R/W

Bit 7：STAF，STWI(SID)的起始标志侦测。
0：软件写“0”清零。STAF 有可能在上电过程中被置位，所以需要在软件初始化时将 STAF 清除。
1：硬件置位，表示在 STWI 总线上发生了一个起始动作。

Bit 6：STOF，STWI(SID)的停止标志侦测。
0：软件写“0”清零。STOF 有可能在上电过程中被置位，所以需要在软件初始化时将 STOF 清除。
1：硬件置位，表示在 STWI 总线上发生了一个停止动作。

SFIE:系统标志中断使能寄存器

SFR 页 = 0~F

SFR 地址 = 0x8E POR = 0110-0000

7	6	5	4	3	2	1	0
SIDFIE	1	1	RTCFIE	SPWIE	BOF1IE	BOF0IE	WDTFIE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7：SIDFIE，串行接口(STWI/SI2C)侦测标志中断使能。
0：禁止 SID 标志(STAF 或 STOF)中断。
1：使能 SID 标志(STAF 或 STOF)中断。

AUXR9:辅助寄存器 9

SFR 页 = **仅 6 页**

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	SIDPS0	T1G1	T0G1	C0FDC1	C0FDC0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7：保留位，写寄存器时，此位必须写“0”。

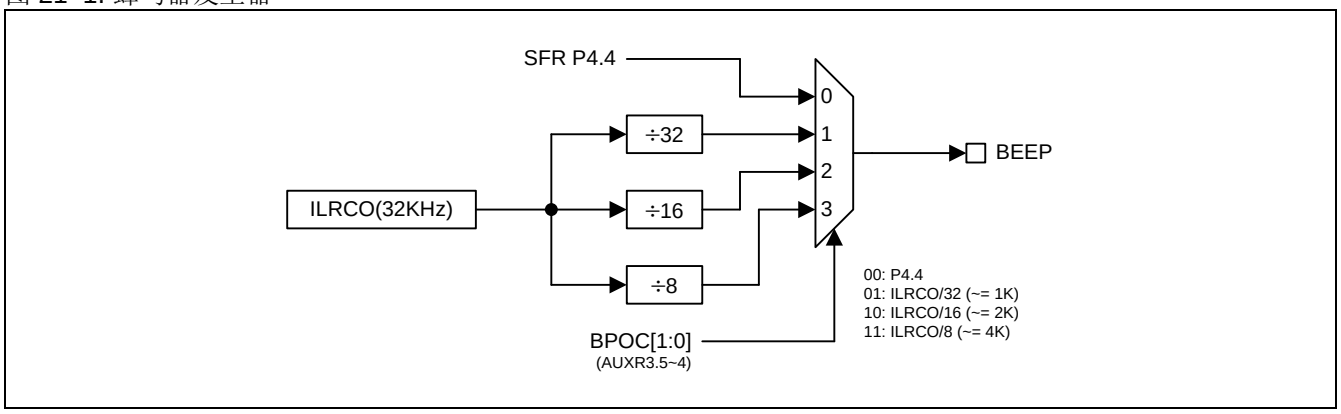
Bit 6：SID/STWI 端口引脚选择。

SIDPS0	STWI_SCL	STWI_SDA
0	nINT1	S0MI
1	TWI0_SCL	TWI0_SDA

21. 蜂鸣器

蜂鸣器功能输出信号产生声音在 BEEP 引脚。信号来自 ILRCO 的分频，频率范围大约在 1、2 或 4 kHz。图 21–1 所示蜂鸣器发生器电路。但是 ILRCO 不是精确的时钟源。更详细的 ILRCO 频率偏差范围请参考章节“32.5 ILRCO 特性”。

图 21–1. 蜂鸣器发生器



21.1. 蜂鸣器寄存器

AUXR3: 辅助寄存器 3

SFR 页 = 仅 **0** 页

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	T0PS0	BPOC1	BPOC0	S0PS0	TWIPS1	TWIPS0	T0XL
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 5~4：BPOC1~0，蜂鸣器输出控制位。

BPOC[1:0]	P4.4 功能	I/O 模式
00	P4.4	By P4M0.4 & P4M1.4
01	ILRCO/32	By P4M0.4 & P4M1.4
10	ILRCO/16	By P4M0.4 & P4M1.4
11	ILRCO/8	By P4M0.4 & P4M1.4

P4.4 用于蜂鸣器功能，推荐设置 P4.4 工作在推挽输出模式。.

蜂鸣器功能使用 P4.4，在使能蜂鸣器功能之前请禁止 OCD 功能。

DCON0: 设备控制寄存器 0

SFR 页 = 仅 **P** 页

SFR 地址 = 0x4C 复位值 = 1000-0011

7	6	5	4	3	2	1	0
HSE	IAPO	0	0	0	IORCTL	RSTIO	OCDE
R/W	**R/W**	**R/W**	**W**	**W**	**W**	**R/W**	**R/W**

Bit 0：OCDE，OCD 使能。
0：在 P4.4 和 P4.5 禁止 OCD 接口。
1：在 P4.4 和 P4.5 使能 OCD 接口。

22. 键盘中断(KBI)

键盘中断功能主要用于当 KBI.7~0 等于或不等于某个值时产生一个中断，这个功能可以用作总线地址识别或键盘键码识别。

有 3 个特殊功能寄存器与此功能相关。键盘中断掩码寄存器(KBMASK)用来定义哪些 KBI 输入引脚可以产生中断。键盘样式寄存器(KBPATN)用来定义与键盘输入值进行比较的值，比较匹配时硬件置键盘中断控制寄存器(KBCON)中的键盘中断标志(KBIF)，若 EIE1 中的 EKBI 中断允许且 EA=1，则还会产生一个中断。键盘中断控制寄存器(KBCON)中的 PATN_SEL 位用来定义比较是“相等”还是“不等”匹配。键盘输入可以分配给不同的端口引脚，详情请参考章节“4.3 功能复用”。

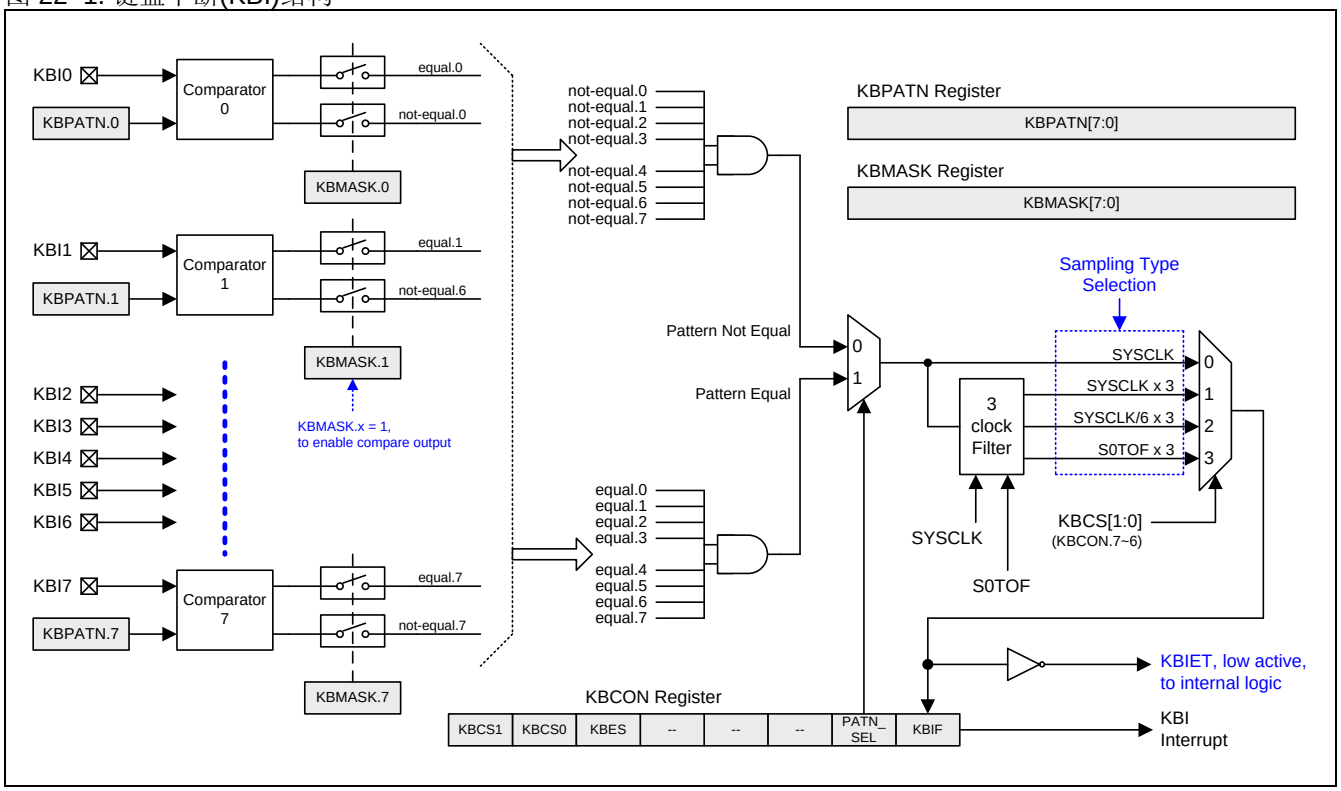
为了使用键盘中断作为“键盘”中断，用户需要设置 KBPATN=0xFF 和 PATN_SEL=0(不相等)，然后将任意按键连接到 KBMASK 寄存器定义的相应端口，按下时硬件就会置位中断标志 KBIF，并当中断使能时产生中断。重写 KBPATN 以清除内部逻辑以允许下一个按键事件。

这个中断可以将 CPU 从空闲模式或掉电模式下唤醒。这个功能在手持设备，电池供电系统等要求低功耗而且易用的设备上特别有用。

当在边沿触发上使用“不相等”（PATN_SEL = 0），则需重写 KBPATN 为 0xFF 使能下一边沿触发事件中断。

22.1. KBI 结构

图 22–1. 键盘中断(KBI)结构



22.2. KBI 寄存器

下面是键盘中断(KBI)操作相关的特殊功能寄存器：

KBPATN:键盘样式寄存器

SFR 页 = 0~F
SFR 地址 = 0xD5 复位值 = 1111-1111

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

Bit 7~0：KBPATN.7~0：键盘样式，复位值是 0xFF。

KBCON:键盘控制寄存器

SFR 页 = 0~F
SFR 地址 = 0xD6 复位值 = 0000-0000

7	6	5	4	3	2	1	0
KBCS1	KBCS0	KBES	--	0	0	PATN_SEL	KBIF
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W

Bit 7~6：KBCS1~0，KBI 滤波模式控制。

KBCS1~0	KBI 输入滤波模式
00	Disabled
01	SYSCLK x 3
10	SYSCLK/6 x 3
11	S0TOF x 3

Bit 5：KBES，KBI 触发模式选择。
0：设置 KBI 模块为电平侦测模式。
1：设置 KBI 模块为边沿侦测模式。

Bit 3 ~ 2：保留位，写寄存器时，此位必须写“0”。

Bit 1：PATN_SEL，样式匹配极性选择。
0：键盘输入不等于 KBPATN 中用户定义的样式时产生中断。
1：键盘输入等于 KBPATN 中用户定义样式时产生中断。

Bit 0：KBIF，键盘中断标志。KBIF 默认值是“1”。
0：必须由软件写入‘0’来清零。
1：键盘输入匹配用户定义的 KBPATN、KBMASK 和 PATN_SEL 设置条件时置位。

KBMASK:键盘中断掩码寄存器

SFR 页 = 0~F
SFR 地址 = 0xD7 复位值 = 0000-0000

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

KBMASK.7：置位时，使能 KBI7 输入作为键盘中断源之一。
KBMASK.6：置位时，使能 KBI6 输入作为键盘中断源之一。
KBMASK.5：置位时，使能 KBI5 输入作为键盘中断源之一。
KBMASK.4：置位时，使能 KBI4 输入作为键盘中断源之一。
KBMASK.3：置位时，使能 KBI3 输入作为键盘中断源之一。
KBMASK.2：置位时，使能 KBI2 输入作为键盘中断源之一。
KBMASK.1：置位时，使能 KBI1 输入作为键盘中断源之一。
KBMASK.0：置位时，使能 KBI0 输入作为键盘中断源之一。

23. 通用逻辑(GPL-CRC)

MG82F6B08 / 6B001/ 6B104 内置一个具有 CCITT16 (CRC16 0x1021)标准的通用逻辑循环冗余检验功能。具有多种应用可编程初始值(种子值)的 8 位数据 CRC 接受流写入 CRC0DI。高字节 CRC0SH (CRCDS0~1=01)和低字节 CRC0SL (CRCDS0~1=00)设置的 16 位初始值(种子值)。结果保存在 CRC0RH (CRCDS0~1=01)和 CRC0RL (CRCDS0~1=00)。

通用逻辑循环冗余检验(GPL-CRC)还有一个通过动态检测 Flash 数据正确性的自动重载引擎直接来自 Flash 存储器的数据路径。

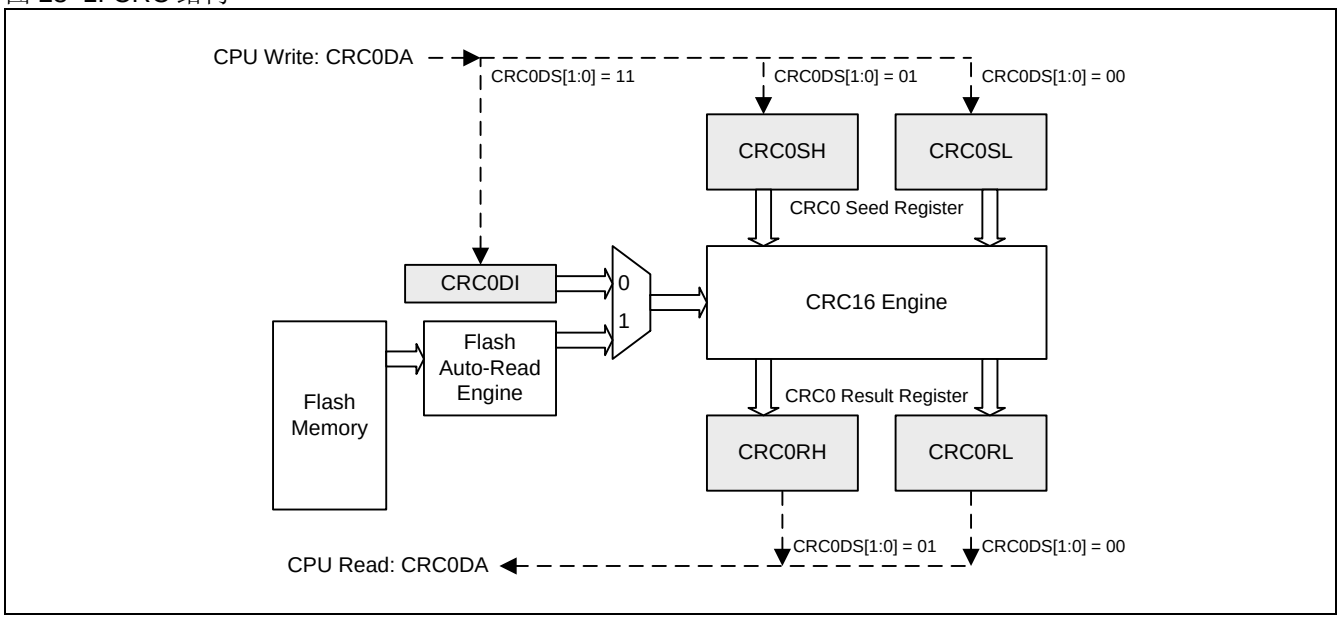
通用逻辑循环冗余检验(GPL-CRC)也结合数据颠倒功能。写数据到 BROVE 寄存器当从 BROVE 读取回来位序就自动颠倒了。高位在先(MSB)变成了低位在先(LSB)。

23.1. GPL-CRC 结构

在正常模式下，需要在 CRC0SH 和 CRC0SL 设置种子然后写数据到 CRC0DI 启动转换。

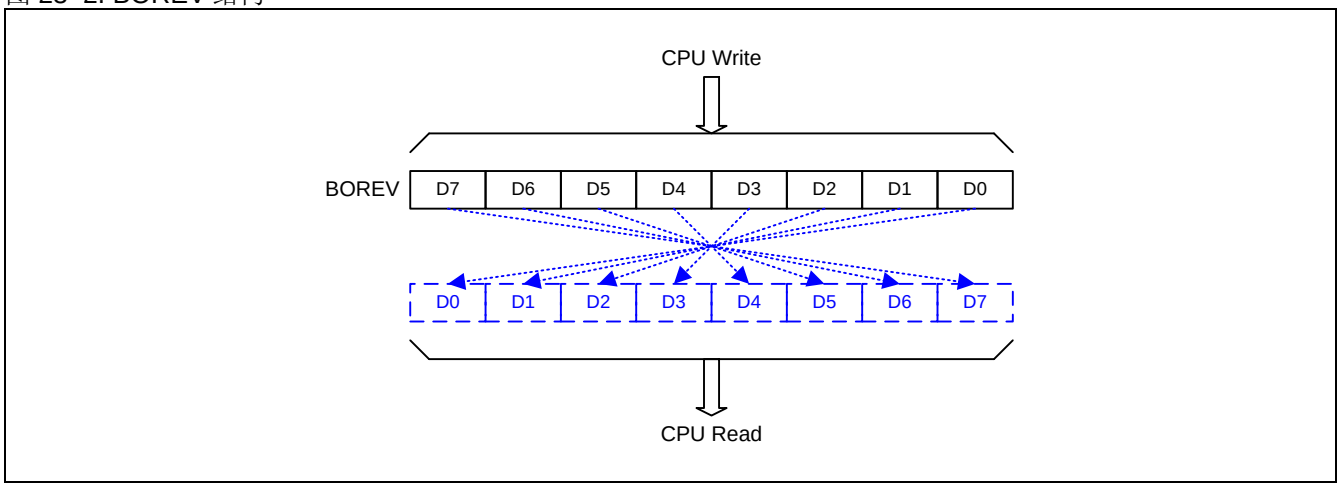
在 Flash 自动重载模式下，需要保持 CRCDS1~0 在“0x00”。并且如下步骤所示：
1. 设置重载区的起始地址 IFADRH 和 IFADRL。
2. 结合 IAPLB(7 位)和 9 位 1-1111-1111 设置结束地址。
3. 设置 IFMT 寄存器(ISP/IAP Flash 模式)为 0x80 作为 Flash 自动重载模式。
4. 顺序将 0x46h 和 0xB9h 写入 SCMD 寄存器触发 CRC 计算。

图 23–1. CRC 结构



23.2. GPL-BOREV 结构

图 23–2. BOREV 结构



23.3. GPL 寄存器

CRC 操作相关的特殊功能寄存器如下：

CRC0DA: CRC0 数据端口

SFR 页 = 0~F

SFR 地址 = 0xB6 复位值 = 0000-0000

7	6	5	4	3	2	1	0
CRC0DA.7	CRC0DA.6	CRC0DA.5	CRC0DA.4	CRC0DA.3	CRC0DA.2	CRC0DA.1	CRC0DA.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~0：CRC0 数据端口。CRC0 数据访问定义如下表：

CRCDS1~0	CPU R/W	CRC0 数据选择	描述
00	写	CRC0SL	CRC0 数据种子低位寄存器。
01	写	CRC0SH	CRC0 数据种子高位寄存器。
10	写	--	保留。
11	写	CRC0DI	CRC0 数据输入寄存器。
00	读	CRC0RL	CRC0 结果低位寄存器。
01	读	CRC0RH	CRC0 结果高位寄存器。
10	读	--	保留。
11	读	--	保留。

AUXR1:辅助控制寄存器 1

SFR 页 = 0~F

SFR 地址 = 0xA2 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	0	CRCDS1	CRCDS0	0	0	0	DPS
W	**W**	**R/W**	**R/W**	**W**	**W**	**W**	**R/W**

Bit 5~4：CRCDS1~0。CRC0 数据端口选择 1~0。

BOREV:位序颠倒数据寄存器

SFR 页 = 0~F

SFR 地址 = 0x96 复位值 = 0000-0000

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

Bit 7~0：BOREV7~0，数据读/写的位序颠倒功能。

如何字节写到 BOREV 读出来位序颠倒了，也就是写入低位在先(LSB)将变成高位在先(MSB)。例如：

如果 0xA0 写入到 BOREV，读回来将是 0x05。

如果 0x01 写入到 BOREV，读回来将是 0x80。

IFMT: ISP/IAP/EEPROM 模式表

SFR 页 = 0~F

SFR 地址 = 0xE5 复位值 = xxxx-x000

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

Bit 7~4：保留位，写 IFMT 寄存器时，此位必须写“0000_0”。

Bit 3~0：ISP/IAP/EEPROM/Page-P 操作模式选择

MS[7:0]	模式
0 0 0 0-0 0 0 0	待机。
0 0 0 0-0 0 0 1	AP/IAP-存储器字节读。
0 0 0 0-0 0 1 0	保留。
0 0 0 0-0 0 1 1	保留。
0 0 0 0-0 1 0 0	P 页 SFR 写。
0 0 0 0-0 1 0 1	P 页 SFR 读。
1 0 0 0-0 0 0 0	CRC 的自动 flash 读。
1 0 0 0-0 0 0 1	Flash 字节读地址加一功能。
1 0 0 0-0 0 1 0	Flash 字节编程地址加一功能。一直留在目标页
1 1 0 0-0 0 0 0	EEPROM 字节读
1 1 0 0-0 0 0 1	保留。
1 1 0 0-0 0 1 0	保留。
1 1 0 0-0 0 1 1	保留。
1 1 0 0-1 0 0 0	EEPROM 字节编程
Others	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。

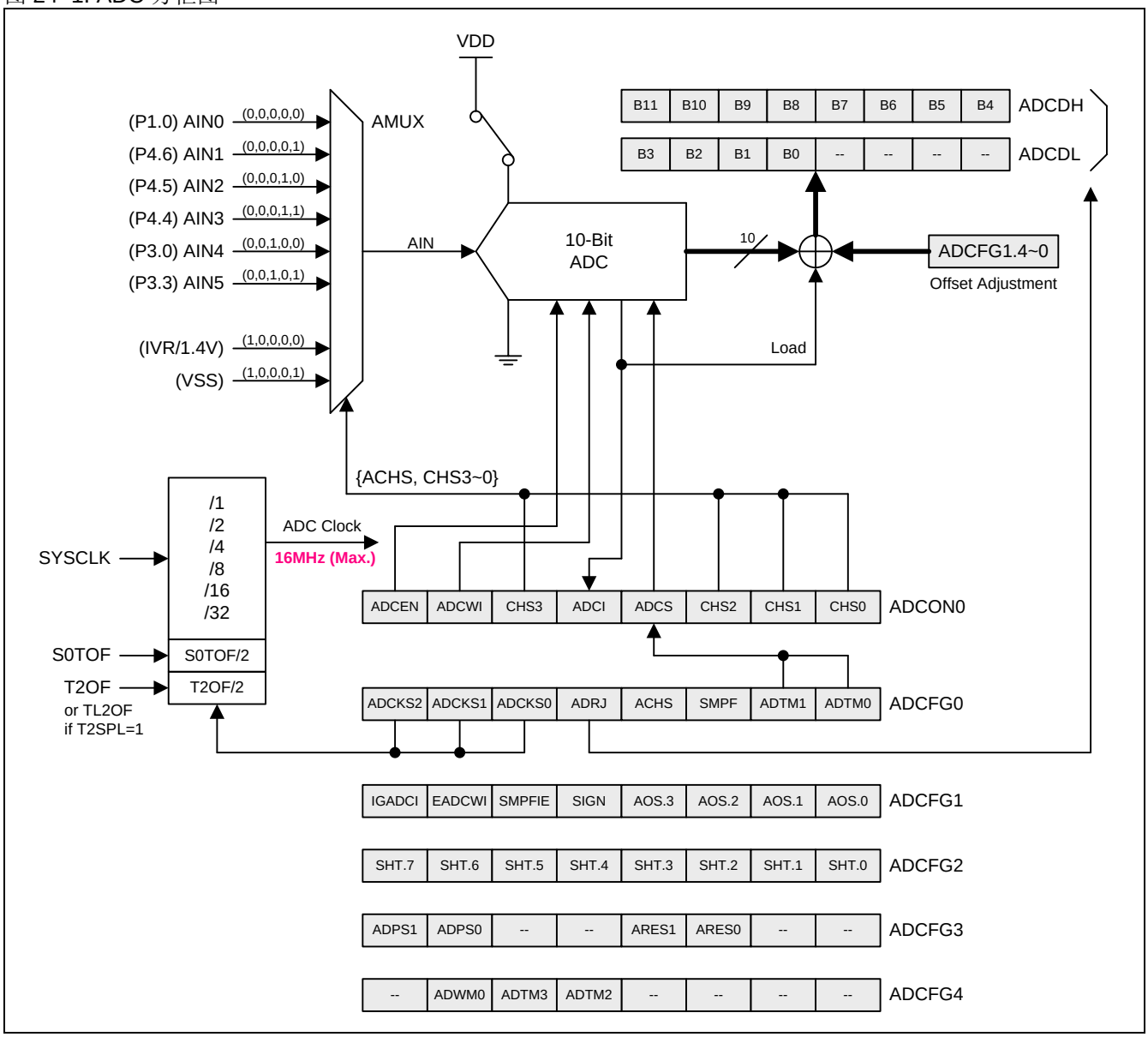
如果软件选择 CRC 的自动 flash 读模式，flash 起始地址由 IFADRH 和 IFADRL 定义。flash 结束地址在{IAPLB + 9 位 1-1111-1111}。

24.10 位 ADC

MG82F6B08 / 6B001/ 6B104 的 ADC 子系统由一个模拟多路器(AMUX)和一个 **666K**sps、**10** 位逐次逼近型模数转换器组成。多路器(AMUX)可以通过特殊功能寄存器进行配置通道，如图 24–1.。ADC 运行在单一节点模式，并且可以配置测量 AIN0 ~ AIN5 输入引脚的任何一个或内部参照。仅当 ADC 控制寄存器(ADCON0)的 ADCEN 位被置逻辑 1 的时候 ADC 子系统被使能，ADCEN 设置为逻辑 0 的话 ADC 子系统低功耗关闭

24.1. ADC 结构

图 24–1. ADC 方框图



24.2. ADC 操作

ADC 最大转换速度可以达到 **666K** sps。 ADC 转换时钟由 ADCFG0 寄存器的 ADCKS2~0 位决定是系统时钟分频、S0BRG 溢出或者定时器 2 的溢出。ADC 转换时钟不能超过 **16MHz**。

转换完成后(ADCI 为 1)，转换结果从 ADC 结果寄存器(ADCDH，ADCDL)中得到。作为单节点 ADC，转换结果是：

$$\text{ADC Result} = \frac{V_{IN} \times 1024}{\text{VDD Voltage}}$$

24.2.1. ADC 输入通道

模拟多路器(AMUX)选择输入口给 ADC，允许任何一个 AIN5~0 引脚成为被测量的单节点模式和一个内部电压参照(IVR，1.4V)。通过 ADCON0 寄存器的 **CHS3~0** 位和 ADCFG0 寄存器的 ACHS 位选择进入 ADC 测量的通道(见图 24–1.)。对被选择的引脚测量的是对地(GND)电压。

24.2.2. ADC 内部电压参考

默认的 ADC 参考电压是 VDD。如果 VDD 不是固定在某个电压，那么使用以下步骤读取电压:

1) 将模拟多路复用器(AMUX)设置为 IVR。
2) 通过 ADC 转换和存储 IVR 值。(提示：不同的 VDD 电压会得到不同的 IVR 回读值，但是 IVR 固定在 1.4V。因此，这个读回值可以作为参考值。)
3) 使用 IVR 读取返回参考值来计算 VDD 值。现在 VDD 得到了一定的值，可以作为参考电压。
4) 使用参考电压转换输入电压。

24.2.3. 开启一个转换

在使用 ADC 功能之前，用户应：

1) 设置 ADC 数据分辨率为 10 位或 8 位模式。
2) 置位 ADCEN 启动 ADC 硬件。
3) 通过 ADCKS2、ADCKS1 和 ADCKS0 位配置 ADC 输入时钟。
4) 通过位 ACHS、CHS3、CHS2、CHS1 和 CHS0 选择模拟输入通道。
5) 将所选引脚配置成仅模拟输入模式。
6) 通过 ADRJ 位配置 ADC 转换结果输出形式。

现在，用户就可以置位 ADCS 来启动 AD 转换了。转换时间取决于 ADCKS2、ADCKS1 和 ADCKS0 位。一旦转换结束，硬件自动清除 ADCS 位，设置中断标志 ADCI，并将 **10** 位的转换结果按照 ADRJ 的设置存入 ADCDH 和 ADCDL。如果用户设置 ADCS 并且选择 ADC 的触发模式是 S0BRG/定时器 2 溢出或全速运行，这样 ADC 保持不断转换直到 ADCEN 清零或 ADC 配置成手动模式。

如上所述，中断标志 ADCI，由硬件设置以表明一次转换完成。因此，有两种方法检测 AD 转换是否完成：(1)软件检测 ADCI 中断标志；(2)设置 EIE1 寄存器 EADC 位和 IE 寄存器 EA 位使能 ADC 中断。这样，转换结束就会跳入中断服务进程。无论(1) 或 (2), ADCI 标志都必须在下次转换前用软件清零。

24.2.4. ADC 转换率

用户可以根据输入的模拟信号频率选择合适的转换速度。ADC 的最大输入时钟是 **16MHz** 并且操作在最少 **24** 个 ADC 转换时钟的转换时间。用户可以通过 ADCFG0 寄存器的 ADCKS2~0 (ADCFG0.7~5)、SHT (ADCFG2.7~0)和 HA (ADCFG3.5)来配置转换速率。下面公式是一个 ADC 转换的时钟个数：

$$\text{ADC Conversion Rate} = \frac{\text{ADC Clock Freq.}}{(24 + X)} \quad ; X = SHT, 0\sim255$$

注意输入信号是交流信号(AC)，f_N，假设采样率是 f_S，基于奈奎斯特定理，f_S 要大于 2 倍的 f_N 确保测量的精度。

例如，

1. 为了得到 666K Hz 的采样率：
   若 SYSCLK =16MHz 并且 ADCKS = SYSCLK，SHT = 0，
   这样转换速率= 16MHz /(24+0) =666K Hz)。
   (此案例，交流 (AC)输入信号频率 f_N 要低于 333KHz 确保测量的精度。)
2. 为了得到 100K Hz 的采样率：
   若 SYSCLK =16MHz 并且 ADCKS = SYSCLK/4，SHT = 16，
   这样转换速率= 16MHz/4/(24+16) =100K Hz) 。
   此案例，交流 (AC)输入信号频率 f_N 要低于 50KHz 确保测量的精度。

24.2.5. ADC 中断

MG82F6B08 / 6B001/ 6B104 包括 3 种 ADC 中断源:

1. ADCI，当一个 A/D 转换完成，ADCI 置位引发一个中断。此标志中断可以被 IGADCI (ADCFG1.7)阻止。
2. SMPF，当一个 ADC 通道采样和保持完成引发一个中断。此标志中断可以由 SMPFIE (ADCFG1.5)使能。
3. ADCWI，在 ADC 窗口比较模式下，当窗口比较数据匹配出现此中断标志被保持。如果此中断使能引发一个中断。此中断标志由 EADCWI(ADCFG1.6)使能。

图 24–2. ADC 中断

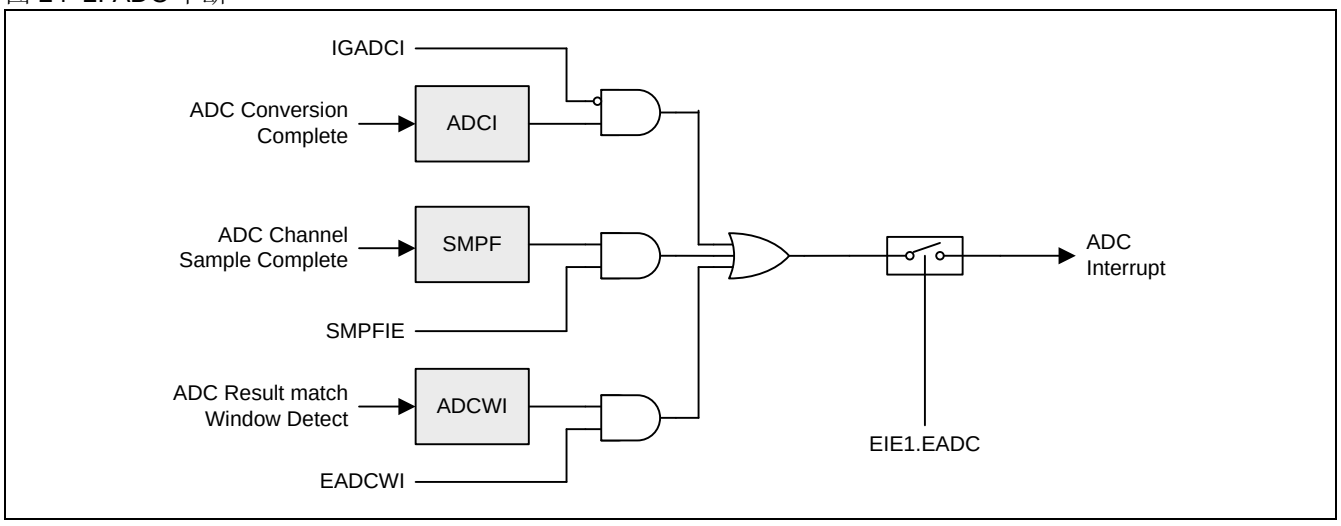
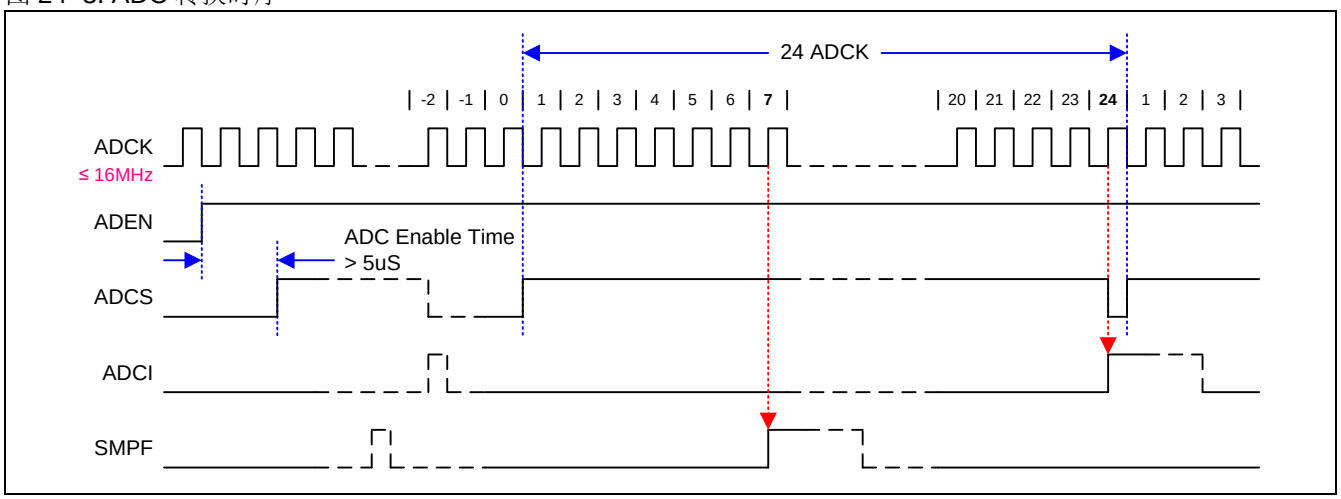


图 24–3. ADC 转换时序

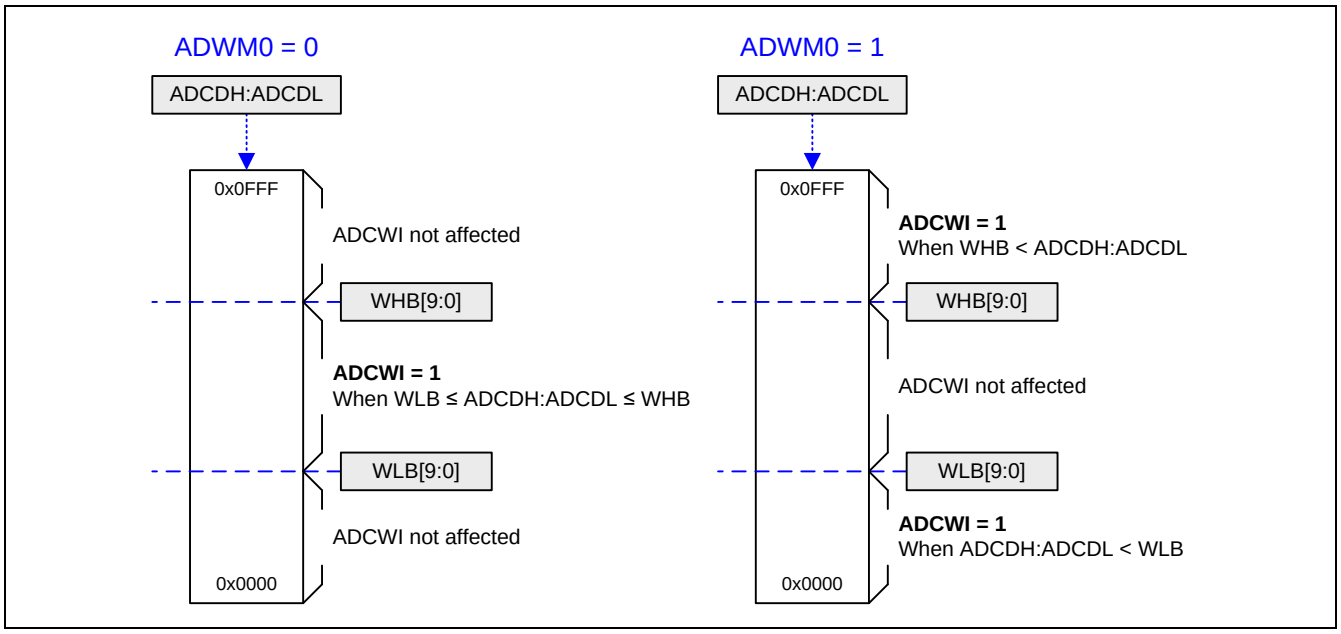


24.2.6. ADC 窗口侦测

MG82F6B08 / 6B001/ 6B104 ADC 的可编程窗口侦测器根据用户设置好的闸值持续比较 ADC 输出寄存器，侦测到期望的值将通知系统。特别是当使用中断驱动系统时可以提供更快的响应时间同时节省代码空间和减少 CPU 使用率以提高效能。窗口侦测器中断标志(ADCWI)也可以在查询模式使用。窗口高边界(WHB[9:0] , {ADCFG12, ADCFG11})和低边界(WLB[9:0], {ADCFG14, ADCFG13})存放边界值。使用者可以定义想要抓取的 ADC 转换值是落在边界内或边界外。两种窗口侦测模式如下图所示：

1. ADWM0 = 0：当 ADC 转换值是在边界内 ADCWI 中断标志被举起。这意味着条件 WLB[9:0] ≤ ADCDH:ADCDL ≤ WHB[9:0]是真，ADCWI 被举起。
2. ADWM0 = 1：当 ADC 转换值是在边界外 ADCWI 中断标志被举起。这意味着条件 WLB[9:0]>ADCDH:ADCDL 或 ADCDH:ADCDL> WHB[9:0]是真，ADCWI 被举起。
3.

图 24–4. ADC 窗口侦测



ADC 窗口侦测的另一个应用为区别电压是大于或小于一个特定电压。
例如：

1. 目标电压≥条件电压：ADWM0 = 0，设置条件值在 WLB 和设置 WHB = 0x3FF。
2. 目标电压≤条件电压：ADWM0 = 0，设置条件值在 WHB 和设置 WLB = 0。
3. 目标电压>条件电压：ADWM0 = 1，设置条件值在 WHB 和设置 WLB = 0。
4. 目标电压<条件电压：ADWM0 =1，设置条件值在 WLB 和设置 WHB = 0x3FF。

24.2.7. I/O 引脚用于 ADC 功能

用作 A/D 转换的模拟输入引脚也可以保持其数字 I/O 输入输出功能。为了获得恰当的模拟性能，用作 ADC 的引脚应当禁止其数字输出，将引脚设为模拟仅输入模式。当 ADCI5~0 引脚应用于模拟信号且此引脚不需要数字输入，软件设置相关引脚为仅模拟输入模式减小数字输入缓冲的功耗。模拟输入功能的端口配置描述在“表 13–3. 通用端口配置设定”和参考章节“13.2 I/O 口寄存器”。

24.2.8. 空闲和掉电模式

在空闲和掉电模式下，若 ADC 功能打开，它将消耗一部分的电流。因此，为了降低待机和掉电模式下的功耗，可以在进入掉电和空闲模式前关闭 ADC 硬件(ADCEN =0)。

在掉电模式下，ADC 不工作。如果在空闲模式下软件触发 ADC 操作，ADC 将完成转换并置位 ADC 中断标志 ADCI。当 ADC 中断使能(EADC，EIE1.1)置位时，ADC 中断将把 CPU 从空闲模式唤醒。

24.2.9. 如何提高 ADC 的精度

使用 ADC 测量电压，其精度可能受到很多因素的影响，例如单片机 VDD 的电源噪声或基准电压的容限等。**MG82F6B08 / 6B001/ 6B104** 于出厂前在 VDD 等于 3.3v 的条件下校准内部参考电压- IVR，并使用 ADC 读取，将其 ADC 值并存储在 Flash 中作为预设值。用以下公式计算 AIN 电压，而不是用测量 VDD 来计算得到 1LSB 精度的电压。

- 推算 IVR 电压 (已经在 VDD=3.3V 下测量了)

$$IVR\ Voltage = \frac{IVR_{ADC\_PreStored\_Value} * 3300}{1024} (mV) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

- 使用比例关系计算 I/O 引脚的电压

$$AIN\ Voltage = \frac{IVR\ Voltage * AIN_{ADC\_Value}}{IVR_{ADC\_Value}} (mV) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

注: 请参考章节” 26.3 如何读取 IVR (2.4V) ADC 预存值”来读取 IVR ADC 预设值。

24.3. ADC 寄存器

ADCON0: ADC 控制寄存器

SFR 页 = 0~F

SFR 地址 =**0xC4** 复位值=0000-0000

7	6	5	4	3	2	1	0
ADCEN	ADCWI	CHS3	ADCI	ADCS	CHS2	CHS1	CHS0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：ADCEN，ADC 使能。
0：清零而关闭 ADC 模块。
1：开启 ADC 模块。在 ADCS 置位之前至少需要 5us 的 ADC 使能时间。

Bit 6：ADCWI，ADC 窗口比较中断标志。
0：ADC 窗口比较自最后一个标志清零之后没有匹配的数据。此标志必须软件清零。
1：ADC 窗口比较有匹配的数据此标志置位。EADCWI (ADCFG1.6)置位引发一个中断。

Bit 5：CHS3，结合 CH2~0 选择 ADC 输入通道。

Bit 4：ADCI，ADC 中断标志。
0：此标志必须软件清零。
1：一次 A/D 转换完成时此标志置位，若中断允许则还会产生一个中断。此标志中断可以被 IGADCI (ADCFG1.7)阻止。

Bit 3：ADCS，ADC 转换启动。
0：ADCS 不会被软件清零。
1：软件置此位启动一次 A/D 转换。转换完成，ADC 硬件会自动清除 ADCS 并且 ADCI 置位。无论 ADCS 或 ADCI 为“1”时将不会开始新的 A/D 转换。

Bit 2~0：CHS2 ~ CHS0，ADC 模拟多路器输入通道选择位。

单节点模式:

ACHS	CHS3~0	通道选择
0	0 0 0 0	AIN0 (P1.0)
0	0 0 0 1	AIN1 (P4.6)
0	0 0 1 0	AIN2 (P4.5)
0	0 0 1 1	AIN3 (P4.4)
0	0 1 0 0	AIN4 (P3.0)
0	0 1 0 1	AIN5 (P3.3)
1	0 0 0 0	Int. VREF (IVR/1.4V)
1	0 0 0 1	AVSS
	其它	保留

ADCFG0: ADC 配置寄存器 0

SFR 页 = **仅 0 页**

SFR 地址 = **0xC3** 复位值= 0000-0000

7	6	5	4	3	2	1	0
ADCKS2	ADCKS1	ADCKS0	ADRJ	ACHS	SMPF	ADTM1	ADTM0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~5: ADC 转换时钟选择位

ADCKS[2:0]	ADC 时钟选择
0 0 0	SYSCLK
0 0 1	SYSCLK/2
0 1 0	SYSCLK/4
0 1 1	SYSCLK/8
1 0 0	SYSCLK/16
1 0 1	SYSCLK/32
1 1 0	S0TOF/2
1 1 1	T2OF/2

注意:

1. *SYSCLK 是系统时钟。*
2. *S0TOF 是 UART0 波特率发生器溢出。*
3. *T2OF 是 Timer2 溢出。*

Bit 4：ADRJ，ADC 结果向右对齐选择。
0：高 8 位转换结果存在 ADCDH[7:0]，低 2 位转换结果存在 ADCDL[7:4]。
1：高 2 位转换结果存在 ADCDH[3:0]，低 8 位转换结果存在 ADCDL[7:0]。

如果 ADRJ = 0

ADCDH: ADC 数据高字节寄存器

SFR 页 = 0~F
SFR 地址 = 0xC6 复位值= xxxx-xxxx

7	6	5	4	3	2	1	0
(B11)	(B10)	(B9)	(B8)	(B7)	(B6)	(B5)	(B4)
R	R	R	R	R	R	R	R

ADCDL: ADC 数据低字节寄存器

SFR Page = 0~F
SFR 地址 =**0xC5** 复位值= xxxx-xxxx

7	6	5	4	3	2	1	0
(B3)	(B2)	--	--	--	--	--	--
R	R	R	R	R	R	R	R

如果 ADRJ = 1

ADCDH

7	6	5	4	3	2	1	0
--	--	--	--	--	--	(B11)	(B10)
R	R	R	R	R	R	R	R

ADCDL

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

在单节点模式下，转换结果是 10 位的无符号整数。输入的测量值从“0”到 VDD(VREF)x 1023/1024。下表列举了向右对齐和向左对齐数据。ADCDH 和 ADCDL 寄存器没有用到的位都是“0”。

输入电压 (单节点模式)	ADCDH:ADCDL (ADRJ = 0)	ADCDH:ADCDL (ADRJ = 1)
VREF+ x 1023/1024	0xFFC0	0x03FF
VREF+ x 512/1024	0x8000	0x0200
VREF+ x 256/1024	0x4000	0x0100
VREF+ x 128/1024	0x2000	0x0080
0	0x0000	0x0000

Bit 3：ACHS，ADC 辅助通道选择。结合 ACHS 和 CHS3~0 选择 ADC 输入通道。

Bit 2：SMPF，ADC 通道采样和保持标志。
0：此标志必须软件清零。
1：一个 ADC 通道采样和保持完成此标志置位。若中断允许则还会产生一个中断。此标志中断可以由 SMPFIE (ADCFG1.5)使能。

Bit 1~0: ADC 触发模式选择

ADTM[1:0]	ADC 转换开启选择
0 0	ADCS 置位
0 1	Timer 0 溢出
1 0	全速模式
1 1	S0 BRG 溢出

ADCFG1: ADC 配置寄存器 1

SFR 页 = **仅 1 页**
SFR 地址 = **0xC3** 复位值= 0000-0000

7	6	5	4	3	2	1	0
IGADCI	EADCWI	SMPFIE	SIGN	AOS.3	AOS.2	AOS.1	AOS.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：IGADCI，忽略 ADCI 中断。
0：使能 ADCI 中断。缺省是使能的。
1：禁止 ADCI 中断。

Bit 6：EADCWI，ADCWI 中断使能。
0：禁止 ADCWI 中断。
1：使能 ADCWI 中断共享 ADC 中断向量。

Bit 5：SMPFIE，SMPF 中断使能。
0：禁止 SMPF 中断。
1：使能 SMPF 中断共享 ADC 中断向量。

Bit 4~0：SIGN 和 AOS.3~0。这个寄存器的值将校正保存在{ADCDH, ADCDL}上的 ADC 转换结果，用来消除偏移量。软件可以动态收集 ADC 的偏移值。软件也可以将这个值存入到 **MG82F6B08 / 6B001/ 6B104** 的 IAP 区域，用它作为一个 ADC 偏移量校正的常规参数。下表列举了 ADC 转换结果的 AD0ROC 校正值。
.

{Sign, AOS.[3:0]}	{ADCDH, ADCDL}值
0_1111	ADC 转换结果 + 15
0_1110	ADC 转换结果 + 14
……	……
0_0010	ADC 转换结果 + 2
0_0001	ADC 转换结果 + 1
0_0000	ADC 转换结果 + 0
1_1111	ADC 转换结果 – 1
1_1110	ADC 转换结果 – 2
……	……
1_0001	ADC 转换结果 – 15
1_0000	ADC 转换结果 – 16

ADCFG2: ADC 配置寄存器 2

SFR 页 = **仅 2 页**
SFR 地址 = **0xC3** 复位值= 0000-0000

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

Bit 7~0: SHT[7:0], 扩展 ADC 采样时间。SHT 的值是 0~255 ADC 时钟.

ADCFG3: ADC 配置寄存器 3

SFR 页 = **仅 3 页**

SFR 地址 = **0xC3** 复位值= 0100-0000

7	6	5	4	3	2	1	0
ADPS1	ADPS0	0	0	ARES1	ARES0	0	0
R/W	**R/W**	**W**	**W**	**R/W**	**R/W**	**R/W**	**W**

Bit 7~6: ADPS1~0, ADC 节能模式选择位 3~2.

ADPS[1:0]	ADC 节能控制
0 0	高功耗，高速度
0 1	中高功耗，中高速度（默认）
1 0	中低功耗，中低速度
1 1	低功耗，低速度

Bit 5~4: 保留位。当 ADCFG3 被写入时，这些位必须软件写“0”

Bit 3~2: ARES1~0, ADC 数据分辨率选择位 1~0.

ARES[1:0]	ADC 数据分辨率选择
0 0	保留
0 1	10 位数据
1 0	8 位数据
1 1	保留

注意: 硬件默认是 “00b”,请在开启ADC功能前设为 “01b”。

Bit 1 ~ 0: 保留位。当 ADCFG3 被写入时，这些位必须软件写“0”

ADCFG4: ADC 配置寄存器 4

SFR 页 = **仅 4 页**

SFR 地址 = **0xC3** 复位值= 0000-0000

7	6	5	4	3	2	1	0
0	ADWM0	ADTM3	ADTM2	0	0	0	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: 保留位。当 ADCFG4 被写入时，这些位必须软件写“0”

Bit 6：ADWM0，ADC 窗口侦测器的模式选择。
0：当 ADCDH:ADCDL 值在 WHB 和 WLB 定义的范围之内置位 ADCWI。
1：当 ADCDH:ADCDL 值在 WHB 和 WLB 定义的范围之外置位 ADCWI。

Bit 5~4：ADC 触发模式选择 3~2。

ADTM[3:0]	ADC 开启转换选择	源
0 0 0 0	Set ADCS	Software
0 0 0 1	Timer 0 溢出(T0OF)	Timer 0
0 0 1 0	全速模式	ADC
0 0 1 1	S0 BRG 溢出(S0TOF)	S0 BRG
0 1 0 0	KBIET	KBI
0 1 0 1	INT1ET	nINT1
0 1 1 0	保留	保留
0 1 1 1	INT0ET	nINT0
1 0 0 0	T2EXES	Timer 2
1 0 0 1	AC0ES	AC0
1 0 1 0	保留	保留
1 0 1 1	保留	保留
1 1 0 0	PCA0 溢出(C0TOF)	PCA0 计数器
1 1 0 1	保留	保留
1 1 1 0	保留	保留
1 1 1 1	保留	保留

Bit 3~0: 保留位。当 ADCFG4 被写入时，这些位必须软件写“0”。

ADCFG11: ADC 配置寄存器 11

SFR 页 = 仅 **B** 页

SFR 地址 = **0xC3** 复位值= 1111-1111

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

ADCFG12: ADC 配置寄存器 12

SFR 页 = 仅 **C** 页

SFR 地址 = **0xC3** 复位值= 1111-1111

7	6	5	4	3	2	1	0
WHB.11	WHB.10	WHB.9	WHB.8	WHB.7	WHB.6	WHB.5	WHB.4
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

WHB.9~0: ADC 窗口高边界值。

ADCFG13: ADC 配置寄存器 13

SFR 页 = 仅 **D** 页

SFR 地址 = **0xC3** 复位值= 1111-1111

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

ADCFG14: ADC 配置寄存器 14

SFR 页 = 仅 **E** 页

SFR 地址 = **0xC3** 复位值= 0000-0000

7	6	5	4	3	2	1	0
WLB.11	WLB.10	WLB.9	WLB.8	WLB.7	WLB.6	WLB.5	WLB.4
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

WLB.9~0: ADC 窗口低边界值。

PCON3: 电源控制寄存器 3

SFR 页 = 仅 **P** 页

SFR 地址 = 0x45 POR = 0000-0000

7	6	5	4	3	2	1	0
IVREN	0	0	SPWRE	0	0	0	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: IVREN, 内部电压参照使能。
0: 禁止片内 IVR (1.4V)。
1: 使能片内 IVR (1.4V)。

Bit 6~5: 保留位。当 PCON3 被写入时，这些位必须软件写“0”。

Bit 3~0: 保留位。当 PCON3 被写入时，这些位必须软件写“0”。

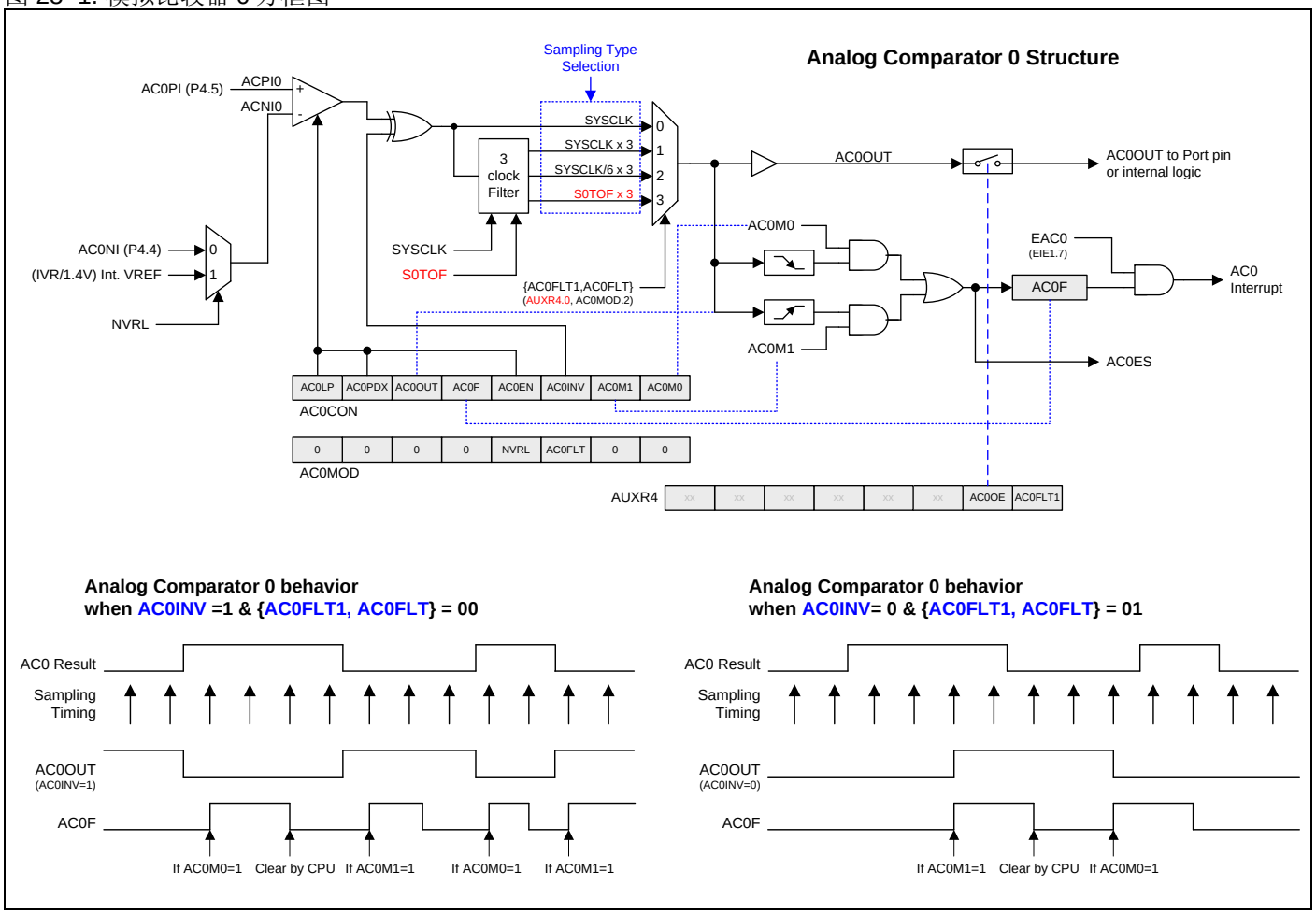
25. 模拟比较器 0 (AC0)

MG82F6B08 / 6B001/ 6B104 内含 **1** 个模拟比较器模块。通过比较 VIN+和 VIN-之间的电压电平，可以将模拟信号转换成数字信息。这些模块的比较结果可以发送到内部逻辑的端口引脚。

1. V_{IN}+上的输入信号:
2. V_{IN}-上有 2 种参考电压可以使用：
   a. 从引脚 AC0NI 输入：如果应用程序需要一个非常精确的电压，那么它可以在该 I/O 引脚上使用一个精确的电压源。
   b. IVR：内部电压参考, 1.4V。
3. 结合时钟滤波器，可设定 3 种不同采样率，滤除不同噪声，减少软件去抖，提高系统整体效率。
4. 中断方式选择：AC0 的中断可通过上升沿、下降沿、双边沿触发。

25.1. AC0 结构

图 25–1. 模拟比较器 0 方框图



25.2. AC0 寄存器

AC0CON:模拟比较器 0 控制&状态寄存器

SFR 页 = **仅 0 页**

SFR 地址 = 0x9E 复位值 = 00X0-0000

7	6	5	4	3	2	1	0
AC0LP	AC0PDX	AC0OUT	AC0F	AC0EN	AC0INV	AC0M1	AC0M0
R/W	**R/W**	**R**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：AC0LP，模拟比较器 0 低功耗使能。
0：禁止 AC0 低功耗模式。
1：使能 AC0 低功耗模式。

Bit 6：AC0PDX，PD 模式下模拟比较器 0 控制。
0：PD 模式下模拟比较器 0 关闭。
1：PD 模式下模拟比较器 0 继续工作。
如果 AC0EN，AC0PDX，EAC0 置位，PD 模式中的比较器只能在低电平或高电平模式下唤醒 CPU。

Bit 5：AC0OUT，这是比较器输出的只读位。

AC0 输入	AC0INV = 0	AC0INV = 1
ACPI0(+) > ACNI0(-)	AC0OUT = 1	AC0OUT = 0
ACPI0(+) < ACNI0(-)	AC0OUT = 0	AC0OUT = 1

Bit 4：AC0F，模拟比较器 0 中断标志位。
0：这个标志位必须由软件清零。
1：当比较器输出满足 AC0M[1:0]位所指定的条件并且 AC0EN 置位时置位。可以通过设置/清除 EIE1 的第 7 位来启用/禁用中断。

Bit 3：AC0EN，模拟比较器 0 使能。
0：清除这个位将强制比较器输出低，并防止进一步的事件置位 AC0F。
1：此位置位使能比较器。

Bit 2：AC0INV，模拟比较器 0 输出反相使能位。
0：AC0 输出不反相。
1：AC0 输出反相。

Bit 1~0：AC0M[1:0]，模拟比较器 0 中断模式。

AC0M[1:0]	AC0 中断模式
0 0	保留
0 1	比较器 0 侦测输出下降沿
1 0	比较器 0 侦测输出上升沿
1 1	比较器 0 侦测输出突变

AC0MOD:模拟比较器 0 模式寄存器

SFR 页 = **仅 0 页**

SFR 地址 = 0x9F 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	0	0	0	NVRL	AC0FLT	0	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7 ~ 5：保留位，写 AC0MOD 寄存器时，此位必须写“0”。

Bit 3：NVRL，负端电压参考范围选择。

Bit 2：AC0FLT，模拟比较器 0 输出滤波控制。和 AC0FLT1 (AUXR4.0)一起选择 AC0OUT 滤波模式。

AC0FLT1, AC0FLT	AC0OUT 滤波模式
0 0	禁止
0 1	SYSCLK x 3
1 0	SYSCLK/6 x 3
1 1	S0TOF x 3

Bit 1 ~ 0：保留位，写 AC0MOD 寄存器时，此位必须写“0”。

AUXR10:辅助寄存器 10

SFR 页 = **仅 7 页**

SFR 地址 = 0xA4 复位值 = 1100-0000

7	6	5	4	3	2	1	0
1	AC0HC0	0	SPIPS0	S0PS1	SPFACE	TWICF	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：AC1HC0，AC1 滞环控制 0。
0：禁止在 AC1 上滞环输入。
1：使能在 AC1 上滞环输入。默认是使能。

AUXR4:辅助寄存器 4

SFR 页 = **仅 1 页**

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
T2PS1	T2PS0	0	T1PS0	0	0	AC0OE	AC0FLT1
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 1：AC0OE，使能 AC0OUT 输出在端口引脚上。
0：禁止 AC0OUT 输出在端口引脚上。
1：使能 AC0OUT 输出在 P1.0 上。

Bit 0：AC0FLT1，AC0 输出滤波控制 1。

PCON3:电源控制寄存器 3

SFR 页 = **仅 P 页**

SFR 地址 = 0x45 POR = 0000-0000

7	6	5	4	3	2	1	0
IVREN	0	0	SPWRE	0	0	0	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

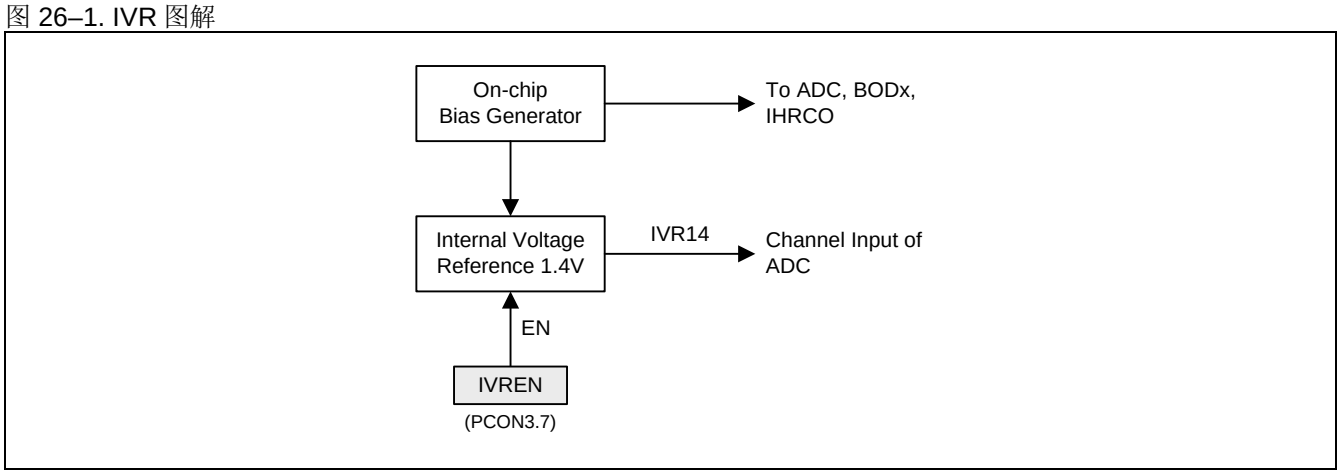
Bit 7：IVREN, 内部电压参考使能。
0：禁止内部 IVR (1.4V)。
1：使能内部 IVR (1.4V)。

26. 内部参考电压(IVR, 1.4V)

可以将 IVR 作为 AC0 和 ADC 的参考电压。它典型的输出是 1.4V，可以由 IVREN 禁用。

26.1. IVR (1.4V) 结构

图 26–1. IVR 图解



26.2. IVR 寄存器

PCON3: 电源控制寄存器 3

SFR 页 = 仅 **P** 页

SFR 地址 = 0x45 POR = 0000-0000

7	6	5	4	3	2	1	0
IVREN	0	0	SPWRE	0	0	0	0
R/W	**R/W**	**R/W**	**R/W**	**W**	**W**	**W**	**W**

Bit 7：IVREN, 内部电压参考使能。
0：禁止内部 IVR (1.4V)。
1：使能内部 IVR (1.4V)。

26.3. 如何读取 IVR (1.4V) ADC 预存值

IVR 在工厂已经被校准@VDD=3.3V. 并将其 ADC 值存储在 Flash 的预留区域内，供用户根据 ADC 值计算电压值。这意味着用户不需要在产线上对 ADC 进行校准。它可以节省测试时间和成本。请参考下列代码读取预存储的 IVR ADC 值，并参考章节"24.2.9 如何提高 ADC 的精度"来了解如何提高 ADC 的测量精度。

请注意此 IVR ADC 预存储值是基于工厂校准的 ADC 偏移量。使用 AOS[3:0]更改偏移量设置时，应将其添加到 IVR ADC 预存储值中，以保持精度。

```
void Get_Prestored_IVR(void)
{
	while(PBSY != 0);

	ISPCR = ISP_ENABLE;
	IFMT = 0x06;
	IFADRH = 0x00;
	IFADRL = 0x4C;

	SCMD = 0x46;
	SCMD = 0xB9;
	Trim_IVR_ADC_Value.B[0] = IFD;
	IFADRL ++;

	SCMD = 0x46;
	SCMD = 0xB9;
	Trim_IVR_ADC_Value.B[1] = IFD;

	ISPCR = ISP_DISABLE;
}
```

27. ISP 和 IAP/EEPROM

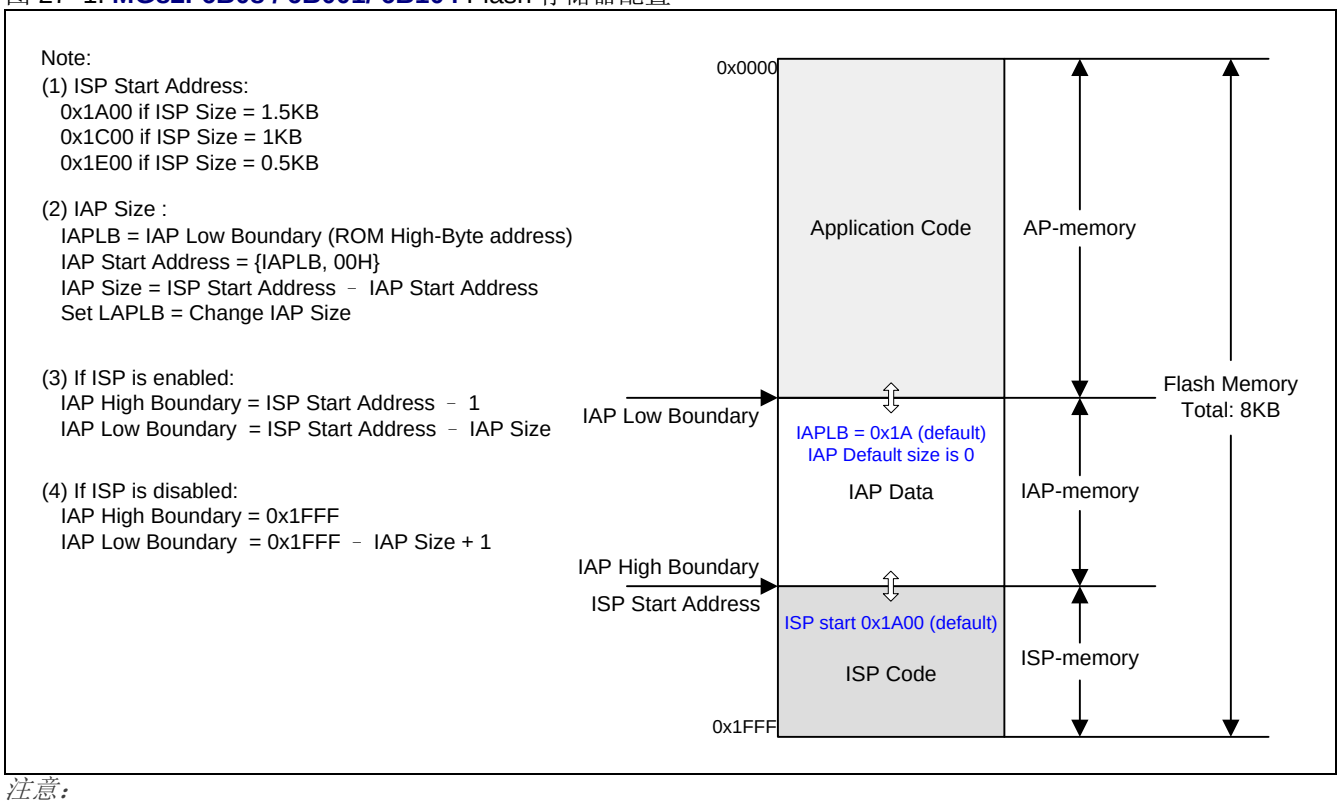
MG82F6B08 / 6B001/ 6B104 的 Flash 存储器分区为 AP-存储器，IAP-存储器和 ISP-存储器。AP-存储器用于存放用户的应用程序。IAP 用于存放非易失性应用数据，ISP-存储器用于储存在系统编程的引导程序。当 MCU 运行在 ISP 区域时，MCU 可以修改 AP 和 IAP 存储器用于程序更新。如果 MCU 运行在 AP 区域，软件仅能修改 IAP 存储器用于更新应用数据。

MG82F6B08 / 6B001/ 6B104 还提供了 EEPROM 作为另一个存储应用数据的空间。它可以不提前进行擦除，直接进行字节编程。EEPROM 提供超过 100,000 次擦写寿命。

27.1. MG82F6B08 / 6B001/ 6B104 Flash 存储器配置

MG82F6B08 / 6B001/ 6B104 总共有 **64K/32K** 字节的 Flash，**MG82F6B08 / 6B001/ 6B104** 的 Flash 配置如图 27–1 所示。ISP 存储空间可以被禁止或由硬件选项 0.5KB 步距配置最大 **3.5K** 字节。IAP 存储空间大小由 IAP 低边界和高边界决定。IAP 低边界由 IAPLB 寄存器的值决定。IAP 高边界与 ISP 的起始地址相关，ISP 存储空间由硬件选项决定。IAPLB 寄存器值由硬件选项配置或 AP 软件编程设定。所有 AP、IAP 和 ISP 存储空间共享总 **8K** 字节的存储空间。

图 27–1. **MG82F6B08 / 6B001/ 6B104** Flash 存储器配置



注意：

*笙泉公司 **MG82F6B08 / 6B001/ 6B104** 的默认 flash 存储器配置是：1.5K ISP 和加密。1.5K ISP 区域是镶入有笙泉专利的 COMBO ISP 代码通过一条线就能在线下载的 1-线 ISP 协议及串口(COM)ISP 协议。*

27.2. MG82F6B08 / 6B001/ 6B104 EEPROM 访问流程

MG82F6B08 / 6B001/ 6B104 有 **2** 种访问模式访问 EEPROM：字节编写和读取模式。MCU 软件可使用这些模式向 EEPROM 读写数据。

在访问 EEPROM 的流程中，MCU 可不需要被停止，做其他事情。当访问 EEPROM 时候，PBSY 会被置起，访问 EEPROM 前请检查 PBSY 标志以避免冲突。在字节读写模式完成时，EEPF 会被置起，需被软件清除。若想连续写多个字节进 EEPROM，每个字节直接都需要检查 EEPF。

本章展示了不同 EEPROM 模式的流程图和范例代码。

EEPROM 字节编程

步骤1：检查PBSY = 0，以确认EEPROM进入待机状态。
步骤2：在ISPCR中设置ISPEN，以使能ISP/IAP/EEPROM流程。
步骤3：在IFMT寄存器上设置MS=**0xC8**选择“EE字节编程模式”。
步骤4：填入字节地址到IFADRH和IFADRL寄存器。
步骤5：填入被编程数据到IFD寄存器。
步骤6：顺序地在SCMD 寄存器写入0x46h 然后0xB9h触发一个EEPROM处理。
步骤7：等待EEPF = 1，该流程完成，然后清除EEPF = 0。
步骤8：清零ISPEN和MS=0x00关闭ISP/IAP流程。

EEPROM 字节读

步骤1：检查PBSY = 0，以确认EEPROM进入待机状态。
步骤2：在ISPCR中设置ISPEN，以使能ISP/IAP/EEPROM流程。
步骤3：在IFMT寄存器上设置MS=**0xC0**选择“EE字节读模式”。
步骤4：填入字节地址到IFADRH和IFADRL寄存器。
步骤5：顺序地在SCMD 寄存器写入0x46h 然后0xB9h触发一个EEPROM处理。
步骤6：现在，FLASH数据已在IFD寄存器中。
步骤7：清零ISPEN和MS=0x00关闭ISP/IAP流程。

MG82F6B08 / 6B001/ 6B104的页擦除，字节编程和读取的详细描述见下面章节：

27.2.1. EEPROM 注意事项

EEPROM 处理中的中断

与 ISP/IAP Flash 访问不同，当访问 EEPROM 时，MCU 会继续运行，意味着中断仍可正常工作。

写入数据到 EEPROM 后的二次确认

为保证程序涉及的数据正确性，我们需要在写入数据到 EEPROM 后读回数据，以确认刚才的编程是正确的。

访问 EEPROM 的目的地址

MG82F6B08 有 512 字节 EEPROM，其地址由 IFADRH:IFADRL 定义，一旦访问目的地超过 0x1FF，硬件将自动忽略超过 0x200 的地址。
举个例子：
1. 若目标地址是 0x2FF，则真实的 EEPROM 地址将来到 0x0FF。
2. 若有有 260 字节数据需要存入 EEPROM，不建议使用 IFADRL 作为计数器，因为当 IFADRL 等于 0xFF，它不会向 IFADRH 产生进位。

EEPROM 使用寿命

内嵌 EEPROM 使用寿命为 100,000 个擦/写周期。意味着擦除后进行写的操作流程不应进行超过 100,000 次。用户需要注意更新 EEPROM 数据的频率。

27.2.2. EEPROM 字节编程模式

MG82F6B08 / 6B001/ 6B104 的 EEPROM 中的任何位数据均可被读写。目标 EEPROM 地址被 IFADRH 和 IFADRL 定义。图 27–2 展示了 ISPIAP 操作中的 EEPROM 字节编程模式。

图 27–2. EEPROM 字节编程流程

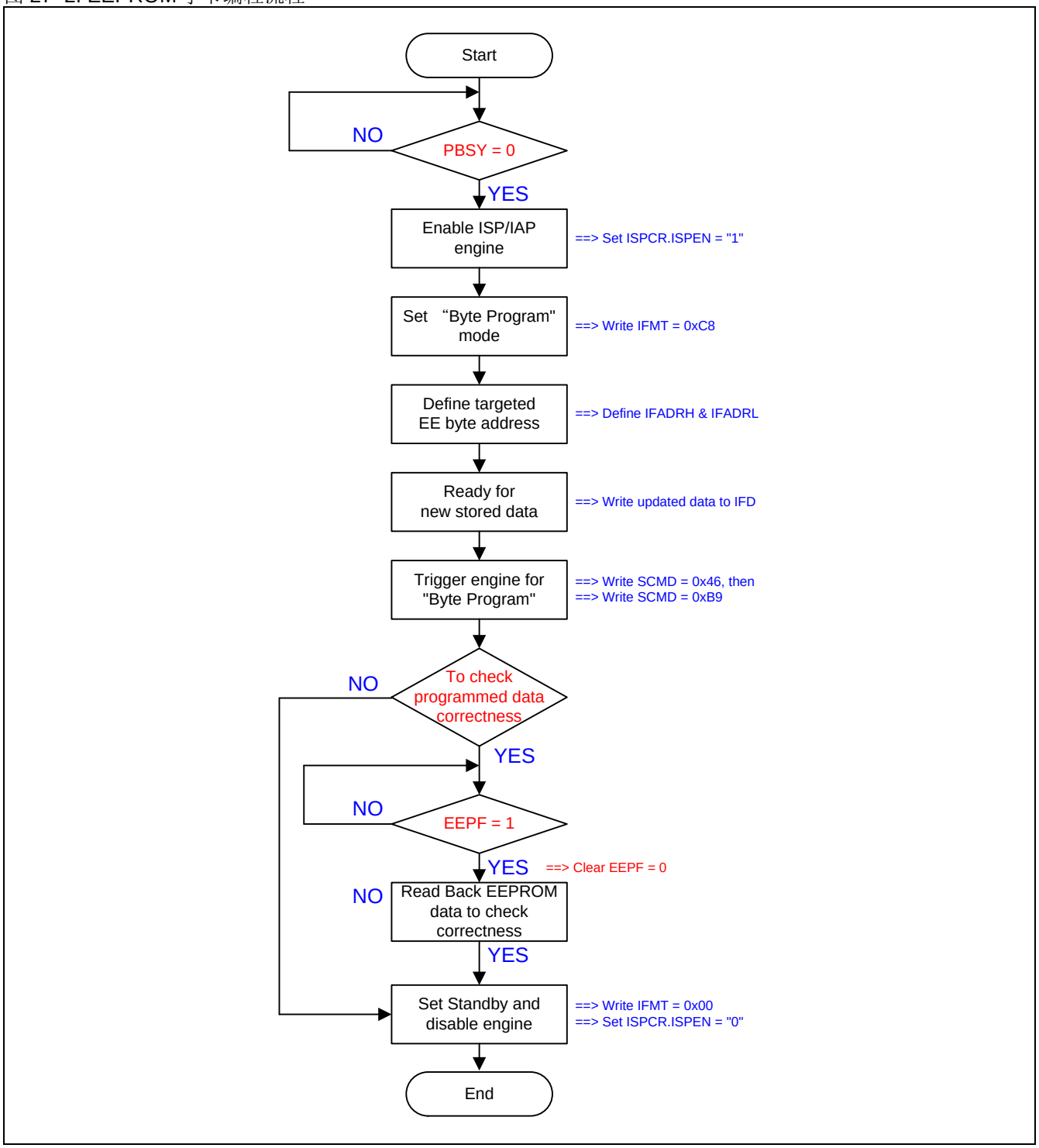


图 27–3 展示了 EEPROM 字节编程模式的范例程序。

图 27–3. EEPROM 字节编程模式范例程序

```
  CHK   PBSY=0          ; 确认 EEPROM 进入待机状态

MOV   ISPCR,#10000000b ; ISPCR.7 = 1,使能 ISP
MOV   IFMT,#0C8h       ; 选择 EEPROM 字节编程模式
MOV   IFADRH,??        ;字节地址填入[IFADRH,IFADRL]
MOV   IFADRL,??        ;
MOV   IFD,??           ;要写入的数据填入 IFD
MOV   SCMD,#46h        ; 触发 ISP/IAP 过程
MOV   SCMD,#0B9h       ;

  CHK   EEPF=1         ; 确认 EEPROM 进入待机状态
  ORL   ISPCR,#01h     ; 清除 EEPF 标志，字节编程完成
;

MOV   IFMT,#00h        ; 选择待机模式
MOV   ISPCR,#00000000b ; ISPCR.7 = 0, 禁用 ISP
```

27.2.3. EEPROM 字节读模式

MG82F6B08 / 6B001/ 6B104 的“字节读取”模式提供了从 EEPROM 读取数据的功能。IFADRH 和 IFADRL 指向 EEPROM 物理字节地址。从 EEPROM 读取到的数据会被存入 IFD。强烈建议在进行数据编程后通过读取模式校对 EEPROM 数据。
图 27–4 展示了 EEPROM 字节读取流程。

图 27–4. EEPROM 字节读取流程

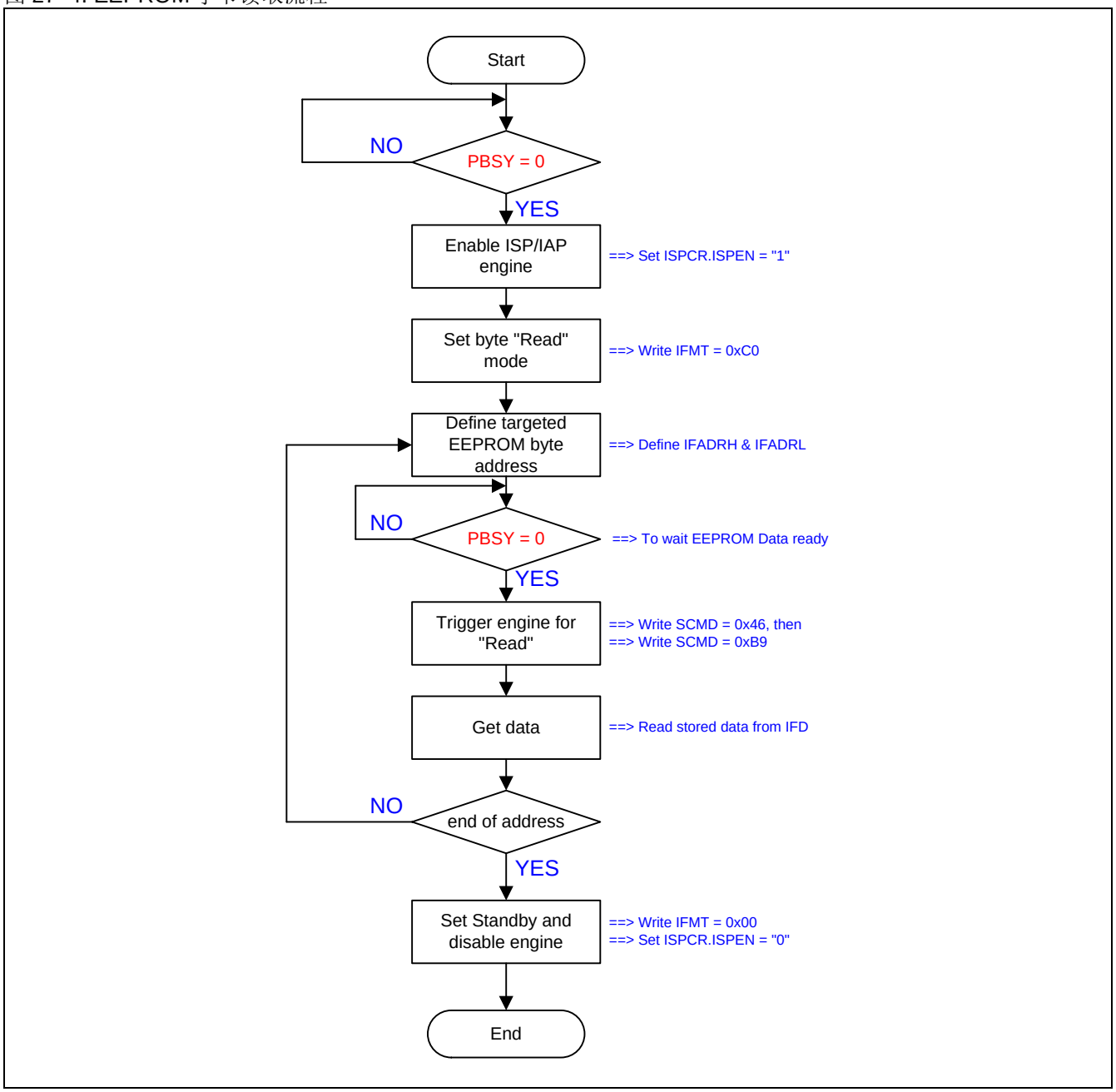


图 27–5 展示了 EEPROM 字节读取的范例程序。

图 27–5. EEPROM 字节读取范例程序

```
  CHK   PBSY=0          ; 确认 EEPROM 进入待机状态

MOV   ISPCR,#10000011b  ; ISPCR.7=1, 使能 ISP
MOV   IFMT,#0C0h        ; 选择读取模式
MOV   IFADRH,??         ; 字节地址填入[IFADRH,IFADRL]
MOV   IFADRL,??         ;
MOV   SCMD,#46h         ; 触发 ISP/IAP 过程
MOV   SCMD,#0B9h        ;

  CHK   PBSY=0          ; 确认数据已在 IFD 中就绪
  ;
MOV   A,IFD             ; 要读取的数据已在 IFD 中
MOV   IFMT,#00h         ; 选择待机模式
MOV   ISPCR,#00000000b  ; ISPCR.7 = 0, 禁用 ISP
```

27.3. MG82F6B08 / 6B001/ 6B104 Flash 在 SP/IAP 上的访问

MG82F6B08 / 6B001/ 6B104 给 ISP 和 IAP 应用提供三种 flash 访问模式：页擦除模式，页面编程模式及读取模式。MCU 软件使用这三种模式去更新 Flash 的数据和获取 Flash 的数据。

当操作 EEPROM 时，CPU 不会停止。一旦用户在 EEPROM 操作过程中读写 ISP/IAP Flash, CPU 将被 ISP/IAP 相关命令暂停，直到 EEPROM 完成当前操作，才会继续执行 ISP/IAP Flash 相关操作。例如，如果 EEPROM 在字节编程中，通常需要几毫秒来完成操作。如果此时执行 ISP/IAP Flash 访问，CPU 将会暂停，直到 ISP/IAP Flash 访问完成。因此，建议在进行 flash 相关访问前检查 PBSY，以避免不必要的 CPU 等待。

本章展示了不同 Flash 模式的流程图和范例代码。

页擦除(每页 64 字节)

步骤 1：检查 PBSY = 0，以确认 EEPROM 进入待机状态。
步骤 2：在 ISPCR 寄存器上置位 ISPEN 使能 ISP/IAP 流程。
步骤 3：在 IFMT 寄存器上设置 MS=**0xC3** 选择页擦除模式。
步骤 4：填入页地址到 IFADRH 和 IFADRL 寄存器。
步骤 5：顺序的在 SCMD 寄存器写入 0x46h 然后 0xB9h 触发一个 ISP 处理。
步骤 6：清零 ISPEN 和 MS=0x00 关闭 ISP/IAP 流程

页编程(每页 64 字节)

步骤 1：检查 PBSY = 0，以确认 EEPROM 进入待机状态。
步骤 2：在 ISPCR 寄存器上置位 ISPEN 使能 ISP/IAP 流程。
步骤 3：在 IFMT 寄存器上设置 MS=**0xC2** 选择**字节编程**模式。
步骤 4：填入字节地址到 IFADRH 和 IFADRL 寄存器。
步骤 5：填入被编程数据到 IFD 寄存器。
步骤 6：顺序地在 SCMD 寄存器写入 0x46h 然后 0xB9h 触发一个 ISP 处理。
步骤 7：返回步骤 3，填入下一位数据字节，重复 64 次，以完成 1 页数据写入。
步骤 8：在 IFMT 寄存器上设置 MS=**0xC1** 选择页**编程**模式。
步骤 9：顺序地在 SCMD 寄存器写入 0x46h 然后 0xB9h 触发一个 ISP 处理。
步骤 10：清零 ISPEN 和 MS=0x00 关闭 ISP/IAP 流程

字节读取

步骤 1：检查 PBSY = 0，以确认 EEPROM 进入待机状态。
步骤 2：在 ISPCR 寄存器上置位 ISPEN 使能 ISP/IAP 流程。
步骤 3：在 IFMT 寄存器上设置 MS=**0x01** 选择读取模式。
步骤 4：填入字节地址到 IFADRH 和 IFADRL 寄存器。
步骤 5：顺序地在 SCMD 寄存器写入 0x46h 然后 0xB9h 触发一个 ISP 处理。
步骤 6：现在,Flash 数据在 IFD 寄存器上。
步骤 7：清零 ISPEN 和 MS=0x00 关闭 ISP/IAP 流程

MG82F6B08 / 6B001/ 6B104的页擦除，页面编程和读取的详细描述见下面章节：

27.3.1. ISP/IAP Flash 页擦除模式

MG82F6B08 / 6B001/ 6B104 的 flash 数据任何一位只能编程为“0”。如果用户需要写“1”到 flash 数据，flash 需要擦除。但是在 **MG82F6B08 / 6B001/ 6B104** 的 ISP/IAP 操作中的 flash 擦除只支持“页擦除”模式，一页擦除将写“1”到一页的所有数据位。**MG82F6B08 / 6B001/ 6B104** 的一页有 64 个字节并且页的起始地址排列到 A8~A0=0x000。目标 flash 地址由 IFADRH 和 IFADRL 决定。这样，在 flash 页擦除模式，IFADRH.0(A8)和 IFADRL.7~0(A7~A0)必须写“0”选择正确的页地址。在 ISP/IAP 操作的 flash 页擦除流程如图 27–6 所示。

图 27–6. ISP/IAP 页擦除流程

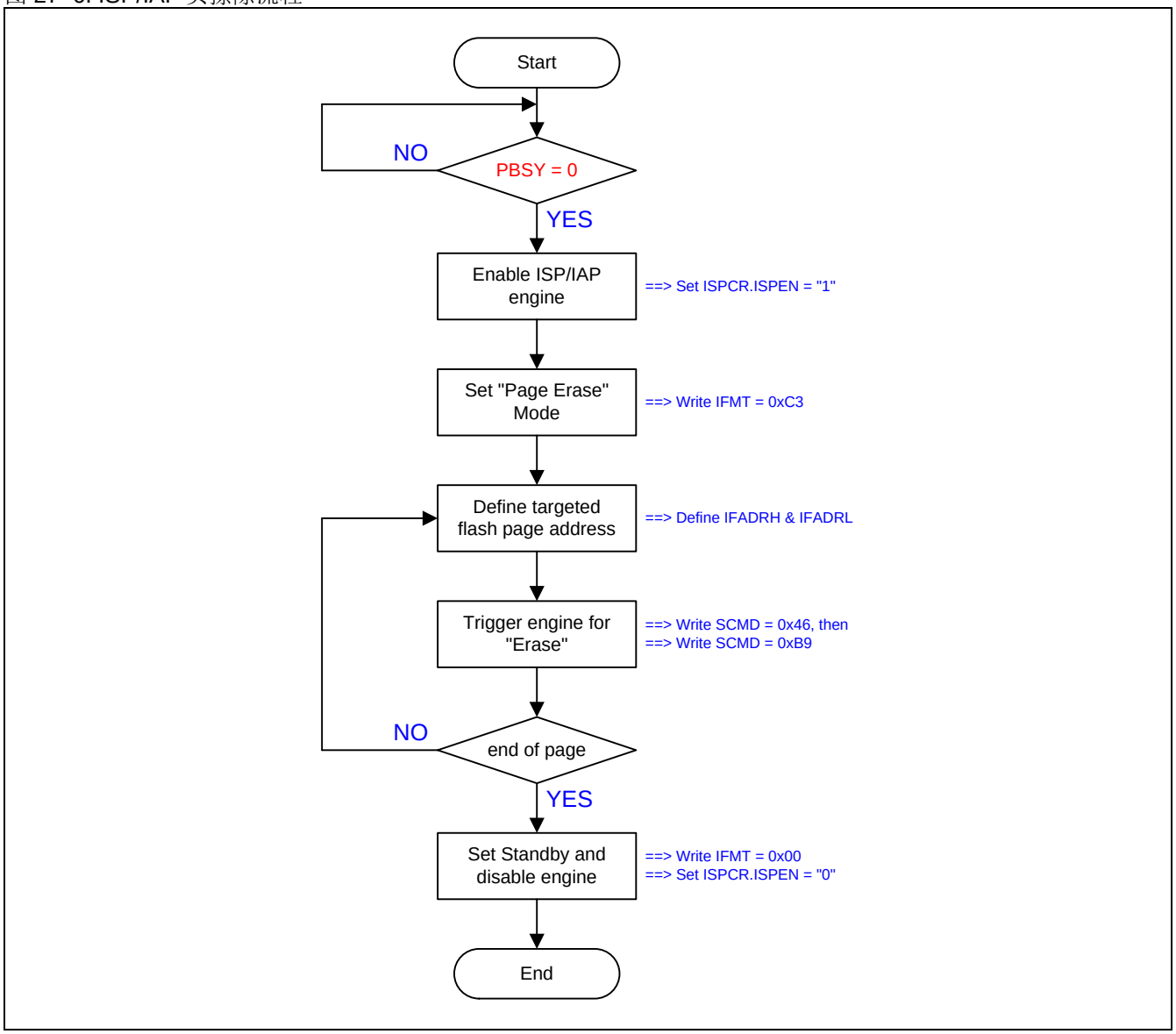


图 27–7 展示了 ISP/IAP 页擦除操作范例程序。

图 27–7. ISP/IAP 页擦除操作范例程序

```
CHK    PBSY=0          ; 确认引擎进入待机状态

MOV    ISPCR,#10000000b ; ISPCR.7 = 1, 使能 ISP
MOV    IFMT,#0C3h       ; 选择页擦除模式
MOV    IFADRH,??        ; 页地址填到[IFADRH,IFADRL]
MOV    IFADRL,??        ;

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

; 现在，MCU 将停在这直到 ISP/IAP 处理完成

MOV    IFMT,#00h        ; 选择待命模式
MOV    ISPCR,#00000000b ; ISPCR.7 = 0, 禁用 ISP
```

27.3.2. ISP/IAP Flash 页编程模式

MG82F6B08 / 6B001/ 6B104 编程模式提供 Flash 存储空间的暂存字节写操作来更新数据。IFADRH 和 IFADRL 指向 Flash 的物理字节地址。IFD 存储编程到 Flash 的内容。在将数据写入 Flash 之前，应该将其存储在 64 字节的数据锁存器中(设置 IFMT = 0xC2 来访问数据锁存器)。ISP/IAP 操作的 Flash 页编程流程如图 27–8 所示。

图 27–8. ISP/IAP 页编程流程

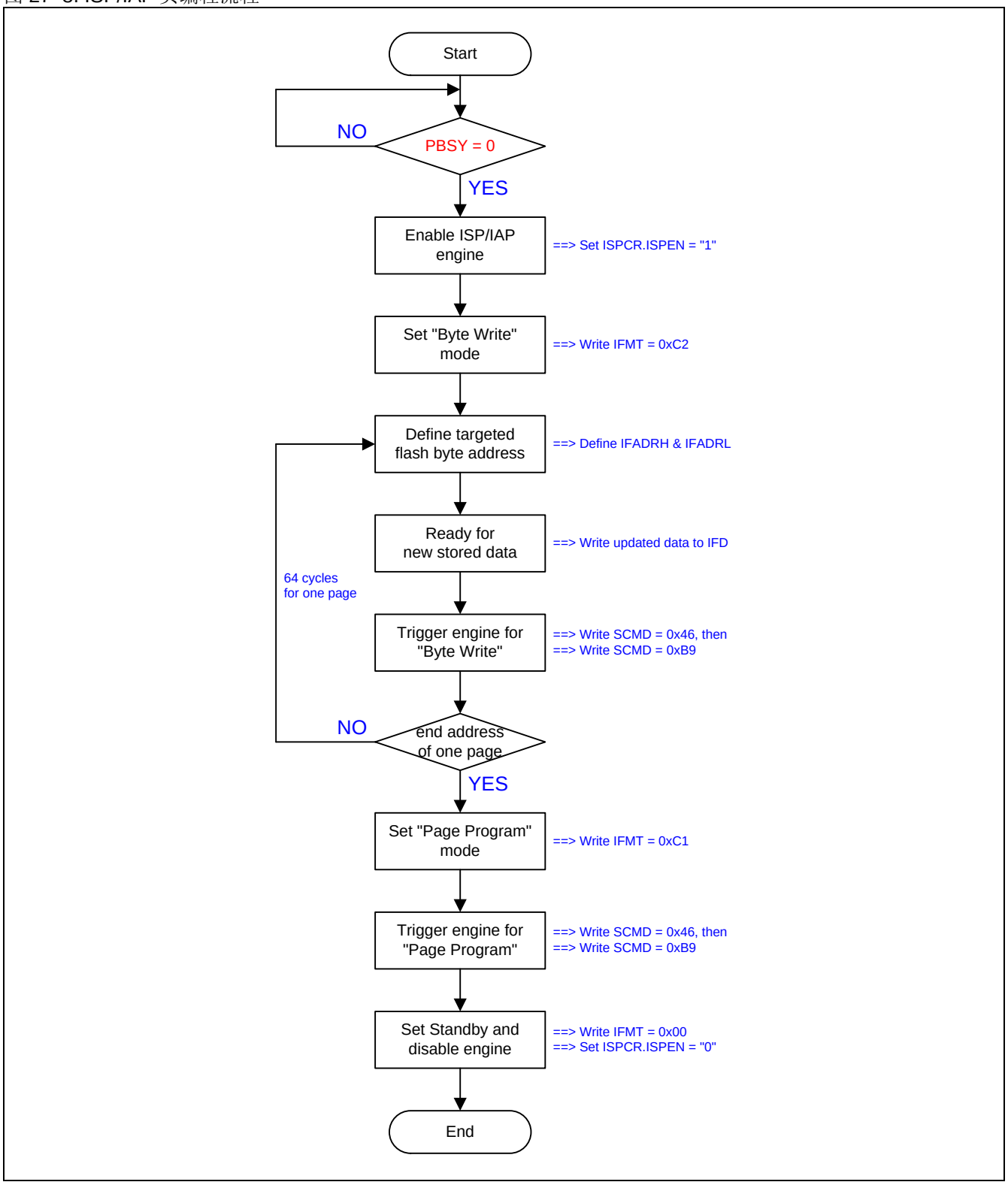


图 27–9 展示了 ISP/IAP 页编程操作范例程序。

图 27–9. ISP/IAP 页编程操作范例程序

```
CHK    PBSY=0                    ; 确认引擎进入待机状态

MOV    ISPCR,#10000011b          ; ISPCR.7=1, 使能 ISP
MOV    IFMT,#0C2h                ; 选择编程模式
MOV    IFADRH,??                 ; 字节地址填到[IFADRH,IFADRL]
MOV    IFADRL,??                 ;
MOV    IFD,??                    ; 编程数据填到 IFD
MOV    SCMD,#46h                 ; 触发 ISP/IAP 处理
MOV    SCMD,#0B9h                ;

; 现在，MCU 将停在这直到 ISP/IAP 处理完成


INC    A
CJNE   A,#40h, Byte_Write_Loop   ;1 个页面 64 个周期

MOV    IFMT,#0C1h                ;MS[7: 0]=[1,1,0,0,0,0,0,1],选择页面编程模式
MOV    SCMD,#46h                 ; 触发 ISP/IAP 处理
MOV    SCMD,#0B9h      ;
; 现在，MCU 将停在这直到 ISP/IAP 处理完成

MOV    IFMT,#00h                 ; 选择待命模式
MOV    ISPCR,#00000000b          ; ISPCR.7 = 0, 禁用 ISP
```

27.3.3. 如何进行 ISP/IAP Flash 字节编程

MG82F6B08/6B001/6B104 不支持 Flash 字节编程模式。用户可以在 RAM 中分配 64 字节的数据数组来存储数据。然后用页编程方式从 RAM 写到 ISP/IAP Flash。或使用 EEPROM 存储字节数据。

27.3.4. ISP/IAP Flash 读模式

MG82F6B08 / 6B001/ 6B104 读取模式提供从 Flash 存储空间获取已存储数据的字节读取操作。IFADRH 和 IFADRL 指向 Flash 的物理字节地址。IFD 存储从 Flash 读取到的内容。建议在数据编程或页擦除之后通过读取模式核对 Flash 数据。
ISP/IAP 操作下的 Flash 字节读取流程如图 27–10 所示。

图 27–10. ISP/IAP 字节读流程

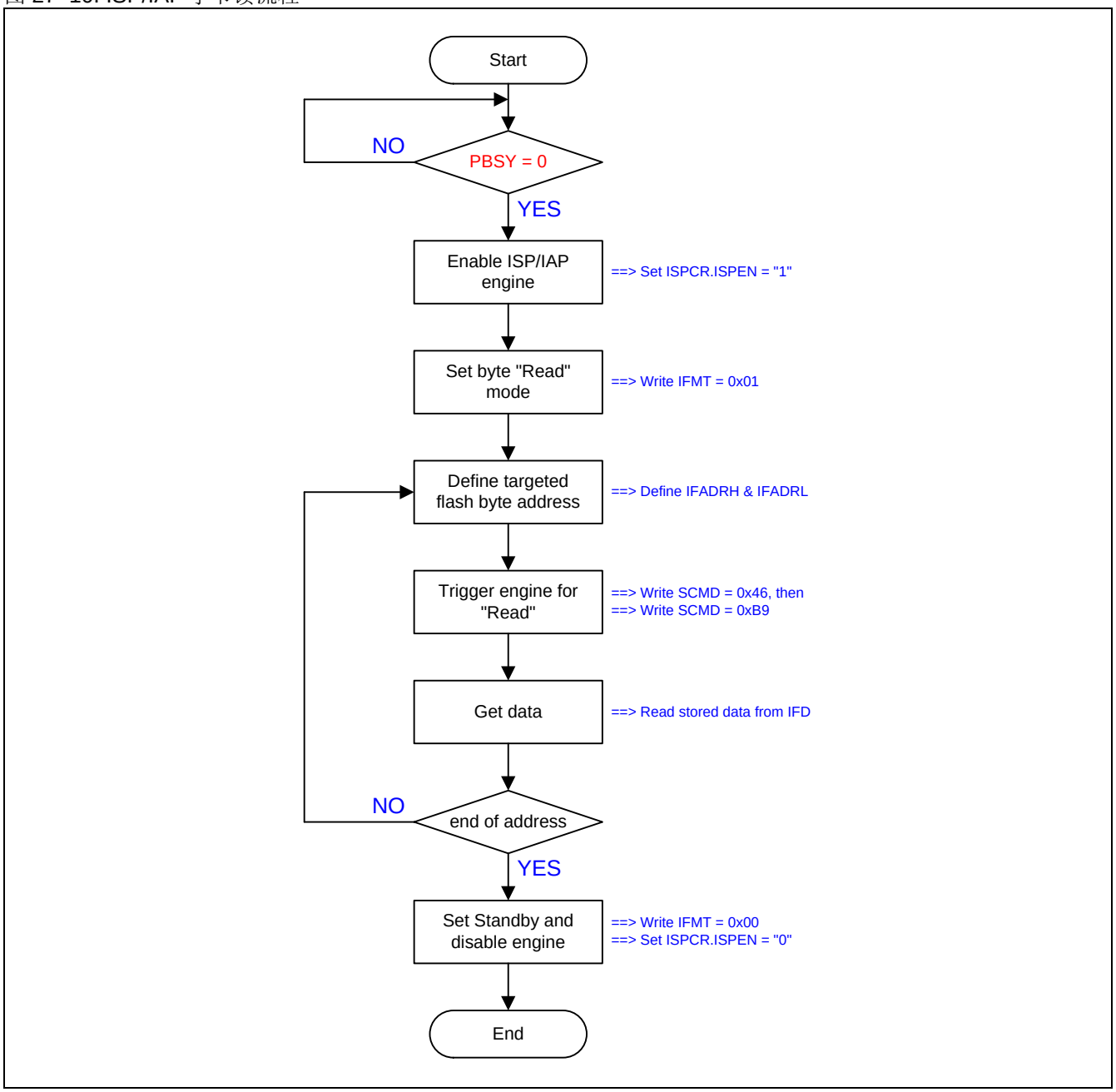


图 27–11 展示了 ISP/IAP 字节读操作范例程序。

图 27–11. ISP/IAP 字节读操作范例程序

```
CHK    PBSY=0          ; 确认引擎进入待机状态

MOV    ISPCR,#10000011b ; ISPCR.7=1, 使能 ISP
MOV    IFMT,#01h        ; 选择读模式
MOV    IFADRH,??        ; 字节地址填写到[IFADRH,IFADRL]
MOV    IFADRL,??        ;
MOV    SCMD,#46h        ; 触发 ISP/IAP 处理
MOV    SCMD,#0B9h       ;
; 现在，MCU 将停在这直到 ISP/IAP 处理完成
MOV    A,IFD            ; 现在，数据已经存在 IFD 里
MOV    IFMT,#00h        ; 选择待命模式
MOV    ISPCR,#00000000b ; ISPCR.7 = 0, 用 ISP
```

27.4. ISP 操作

ISP 意指在系统可编程，不需要在实际的终端产品上移除 MCU 芯片就可以更新用户的应用程序(AP 存储空间)和非易失性应用数据(IAP 存储空间)。这个可使用性就有一个宽的现场应用范围。ISP 模式使用引导程序来编程 AP 存储空间和 IAP 存储空间。

注意：

(1) 在用 ISP 功能之前，使用者必须先配置 ISP-存储器空间并用通用烧写器或笙泉的烧写器插入 ISP 代码(引导程序)到 ISP-存储器中。

(2) ISP-存储器中的 ISP 代码只能编程 AP-存储器和 IAP-存储器。

在 ISP 操作完成之后,软件写“001”到 ISPCR.7 ~ ISPCR.5 这样会触发一个软件复位(RESET)并且使 CPU 再启动到应用程序存储空间(AP)的 0x0000 地址。

如我们所知，ISP 代码的作用就是编程 AP 存储空间和 IAP 存储空间。因此，***MCU 为了执行 ISP 代码必须从 ISP 存储空间启动***。根据 MCU 如何从 ISP 存储空间启动，有两种方法执行在系统可编程。

27.4.1. 硬件启动 ISP 方法

在上电复位时为了使 MCU 直接从 ISP 存储空间启动，MCU 的硬件选项 HWBS 和 ISP 存储空间必须使能。硬件选项的 ISP 进入方法叫做硬件访问。一旦 HWBS 和 ISP 存储空间使能,当上电复位时 MCU 总是从 ISP 存储空间启动去执行 ISP 代码(引导程序)。ISP 代码做的第一件事是核对是否有 ISP 请求。如果没有 ISP 请求， ISP 代码触发软件复位(设置 ISPCR.7~5 为“101”)使 MCU 在启动到 AP 存储空间去运行用户应用程序。

如果额外的硬件选项 HWBS2 与 HWBS 及 ISP 存储空间一起使能，MCU 在上电复位或**外部复位**结束之后总从 ISP 存储空间启动。通过外部复位信号提供另外一个硬件访问进入 ISP 模式。第一上电复位之后，**MG82F6B08 / 6B001/ 6B104** 通过外部复位触发而执行 ISP 操作并且不用等待下一次的上电复位，这适合不断电系统去应用硬件方法启动 ISP 功能。

27.4.2. 软件启动 ISP 方法

当 MCU 运行在 AP 存储空间时，软件访问 ISP 通过触发软件复位使 MCU 从 ISP 存储空间启动。这种情况，HWBS 或 HWBS2 不用使能。仅有的方法是当 MCU 运行在 AP 存储空间时同时设置 ISPCR.7~5 为“111”触发软件复位 MCU 从 ISP 存储空间启动。注意： ISP 存储空间必须通过硬件选项配置一个有效空间来保留 ISP 模式给软件方法启动 ISP 应用。

27.4.3. ISP 注意事项

ISP 代码开发

尽管 ISP 存储空间的 ISP 代码是可编程的，ISP 存储空间在 MCU 的 Flash 中有一个 ***ISP 起始地址***(**MG82F6B08 / 6B001/ 6B104** 见图 27–1)，但是并不意味着你需要在你的源代码中加入这个偏移量(***ISP 起始地址***)。代码偏移量硬件自动处理。用户只需像在 AP 存储空间开发应用程序一样开发。

ISP 期间的中断

在触发 ISP/IAP flash 处理之后，内部 ISP 处理时 MCU 将停止一会儿直到处理完成。此时，如果中断已使能则中断事件将排队等待服务。一旦 ISP/IAP flash 处理完成，MCU 继续运行并且如果中断标志仍然有效则排队中的中断将立即服务。不过用户需要意识到下列事项：

(1) 当 MCU 停止在 ISP 处理时，任何中断都不能实时服务。
(2) 低/高电平触发外部中断 nINTx，必须保持到 ISP 处理完成，否则将被忽略。

ISP 和空闲模式

MG82F6B08 / 6B001/ 6B104 不使用空闲模式执行 ISP 功能。反而 ISP/IAP 引擎操作 Flash 存储空间将冻结 CPU 的运行。一旦 ISP/IAP 运行结束，CPU 将继续并且推进紧跟着 ISP/AP 激活的指令。

ISP 的访问目标

如前所述, ISP 用来编程 AP 存储空间和 IAP 存储空间。一旦访问目标地址超出 IAP 存储空间的最后一个字节之外, 硬件将自动忽略 ISP 处理的触发。这样 ISP 触发是无效的并且硬件不做任何事情。

ISP 的 Flash 持久期

内置 Flash 的持久期是 20,000 写周期，换句话说写周期不能超过 20,000 次。这样用户必须注意应用中需要频繁更新 AP 存储空间和 IAP 存储空间这一点。

27.5. 在应用编程(IAP)

MG82F6B08 / 6B001/ 6B104 内建一个 EEPROM，但当空间仍不足或考虑兼容其他系列 MCU 时，应用程序运行时在 Flash 存储空间里允许一些区域被应用成非易失性数据存储区。这个有用特点能使用在断电后还需要保存数据的应用中。

事实上，IAP 的操作除了 Flash 存储空间被划分在不同的区域之外与 ISP 一样。ISP 操作的可编程 Flash 范围在 AP 存储空间和 IAP 存储空间，而 IAP 操作的范围只在 IAP 存储空间。

注意：

*(1) **MG82F6B08 / 6B001/ 6B104** 的 IAP 特点，软件通过写 IFMT 定义的 IAPLB 寄存器声明 IAP 存储空间。IAP 存储空间也可以通过通用的烧入器/编程器或笙泉专利的烧入器/编程器来配置 IAPLB 的初始值。*

*(2)执行 IAP 的程序代码是在 AP 存储空间并且**仅能编程 IAP** 存储空间而**不能编程 ISP** 存储空间*

27.5.1. MG82F6B08 / 6B001/ 6B104 IAP 存储边界/范围

如果ISP存储空间被声明，IAP存储空间范围由IAP和ISP起始地址决定如下：

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

如果ISP 存储空间没有被声明，IAP存储空间范围由下列公式决定：

*IAP 高边界 = **0x1FFF**.*
*IAP 低边界 = **0x1FFF** – IAP + 1.*

例如，如果ISP存储空间是1K字节，这样ISP的起始地址是**0x1C00**，并且IAP存储空间是1K字节，此时IAP存储空间的范围就在**0x1800** ~ **0x1BFF**。**MG82F6B08 / 6B001/ 6B104**的IAP低边界由IAPLB寄存器决定，IAPLB寄存器可以在用户AP程序里用软件修改来调整IAP大小。

该芯片中，IAP功能默认为禁用（IAPLB=0x1A）。若需要访问IAP，只需将IAPLB设置到空白的FLASH空间即可。

27.5.2. 更新 IAP-存储中的数据

ISP/IAP 相关的特殊功能寄存器见章节“27.6 ISP/IAP/EEPROM 寄存器 ”。

由于 IAP 存储空间是 Flash 存储空间的一部分，Flash 擦除仅提供**页擦除，没有字节擦除**。为了在 IAP 存储空间更新“一个字节”，用户不能直接编程一个新数据到那个字节。正确的步骤如下：

步骤 1：保存整页 flash 数据(64 字节)到包含被更新数据的 XRAM 缓冲区。
步骤 2：擦除此页(**使用 ISP/IAP Flash 页擦除模式**)。
步骤 3：在 XRAM 缓冲区修改新数据字节。
步骤 4：编程 XRAM 缓冲区的被更新数据到此页(**使用 ISP/IAP Flash 编程模式**)。

为了读取IAP 存储空间数据，用户可以使用**ISP/IAP Flash读取模式**获取目标数据。

27.5.3. IAP 注意事项

IAP 期间的中断

在触发 ISP/IAP flash 处理之后，内部 IAP 处理时 MCU 将停止一会儿直到处理完成。此时，如果中断已使能则中断事件将排队等待服务。一旦 ISP/IAP flash 处理完成，MCU 继续运行并且如果中断标志仍然有效则排队中的中断将立即服务。不过用户需要意识到下列事项：

(1) 当 MCU 停止在 IAP 处理时，中断不能实时服务。
(2) 低/高电平触发外部中断 nINTx，必须保持到 IAP 处理完成，否则将被忽略。

IAP 和空闲模式

MG82F6B08 / 6B001/ 6B104 不使用空闲模式执行 IAP 功能。反而 ISP/IAP 引擎操作 Flash 存储空间将冻结 CPU 的运行。一旦 ISP/IAP 运行结束，CPU 将继续并且推进紧跟着 ISP/AP 激活的指令。

IAP 的访问目标

如前所述，IAP 用来编程 IAP 存储空间。一旦访问目标地址不在 IAP 存储空间之内，硬件将自动忽略 IAP 处理的触发。这样 IAP 触发是无效的并且硬件不做任何事情。

读取 IAP 数据的另一种方法

IAP 存储空间读取 Flash 数据，除了使用 Flash 的读取模式之外，另一个方法是使用“MOVC A,@A+DPTR”指令。这里，DPTR 和 ACC 各自填入想要的地址和偏移量。并且访问目标必须在 IAP 存储空间内，否则读取的数据将不确定。注意使用‘MOVC’指令比使用 Flash 的读取模式更快。

IAP 的 Flash 持久期

内置 Flash 的持久期是 20,000 擦除/写周期，换句话说擦除再写周期不能超过 20,000 次。这样用户必须注意应用中需要频繁更新 IAP 存储空间这一点。

27.6. ISP/IAP/EEPROM 寄存器

下面专门描述ISP，IAP和P页相关的特殊功能寄存器：

IFD: ISP/IAP 数据寄存器

SFR 页 = 0~F
SFR 地址 = 0xE2 复位值 = 1111-1111

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

IFD 是 ISP/IAP/P 页操作的数据端口寄存器。在 ISP/IAP/P 页写操作时 IFD 的数据将被写入到期望的地址，在 ISP/IAP/P 页读操作时 IFD 的值是读到期望地址的数据。

IFADRH: ISP/IAP 地址高八位

SFR 页 = 0~F
SFR 地址 = 0xE3 复位值 = 0000-0000

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

IFADRH 是所有 ISP/IAP 模式下的高 8 位地址。在 P 页模式下没有定义。

IFADRL: ISP/IAP 地址低八位

SFR 页 = 0~F
SFR 地址 = 0xE4 复位值 = 0000-0000

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

IFADRL 是所有 ISP/IAP/P 页模式下的低 8 位地址。在 flash 页擦除时，IFADRL 可以不用理会。

IFMT: ISP/IAP/EEPROM Mode Table

SFR 页 = 0~F
SFR 地址 = 0xE5 复位值 = xxxx-x000

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

Bit 7~0：ISP/IAP/EEPROM/ P 页操作模式选择

MS[7:0]	模式
0 0 0 0-0 0 0 0	待机
0 0 0 0-0 0 0 1	AP/IAP-存储器字节读
0 0 0 0-0 0 1 0	保留
0 0 0 0-0 0 1 1	保留
0 0 0 0-0 1 0 0	P 页 SFR 写
0 0 0 0-0 1 0 1	P 页 SFR 读
1 0 0 0-0 0 0 0	CRC 的自动 flash 读。
1 0 0 0-0 0 0 1	Flash 字节读地址加一功能。
1 0 0 0-0 0 1 0	Flash 字节编程地址加一功能。一直留在目标页
1 1 0 0-0 0 0 0	EEPROM 字节读
1 1 0 0-0 0 0 1	Flash 页编程
1 1 0 0-0 0 1 0	Flash 字节写入 64 字节数据锁存
1 1 0 0-0 0 1 1	Flash 页擦除
1 1 0 0-1 0 0 0	EEPROM 字节写
其它	保留

IFMT 是用来选择闪存是用来执行众多的 ISP/IAP 功能还是选择 P 页寄存器的访问。
如果软件选择 CRC 的自动 flash 读模式，flash 起始地址由 IFADRH 和 IFADRL 定义。flash 结束地址在{IAPLB + 9 位 1-1111-1111}。

SCMD:连续命令字寄存器

SFR 页　　= 0~F
SFR 地址　　= 0xE6　　复位值 = xxxx-xxxx

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

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

ISPCR: ISP 控制寄存器

SFR 页　　= 0~F
SFR 地址　　= 0xE7　　POR = 0000-0000

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

Bit 7：ISPEN，ISP/IAP/P 页操作使能。
0：所有的 ISP/IAP/P 页编程/擦除/读都是被禁止的。
1：使能 ISP/IAP/P 页编程/擦除/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：无操作。
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被阻止。

Bit 3~2：保留位，写寄存器时，此位必须写“0”。

Bit 1：PBSY，EE 访问流程中繁忙，只读。
0：EEPROM 处于待机状态。
1：EEPROM 处于繁忙状态。

Bit 0：EEPF，EEPROM 数据编程完成标志。
0：EEPF 必须被软件清除。
1：EEPROM 编程完成时，该位会被硬件置起。

IAPLB: IAP 低边界

SFR 页 = **仅 P 页**

SFR 地址 = 0x03 复位值 = 0111-000x

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。因为一个闪存页是 64 字节，所以 IAPLB 必须是偶数。
为了读取 IAPLB，MCU 需要在 P 页里定义 IFADRL 地址，IMFT 模式选择 P 页读及 ISPCR.ISPEN 置位。并且在 SCMD 依次写入 0x46h 和 0xB9h，这样 IAPLB 的值就会出现在 IFD。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 存储区如下。

IAP 低边界= IAPLB[7:0] x256

IAP 高边界= ISP 起始地址 – 1。

如果 IAPLB=0x16 且 ISP 起始地址是 0x1A00，这样 IAP 存储区位于 0x1600 ~ 0x19FF。

另外要注意一点，IAP 的低边界地址不能大于 ISP 的起始地址。

27.6.1. ISP/IAP 范例程序

ISP 操作的范例程序如图 27–12 所示。

图 27–12. ISP/IAP 范例程序

```
;*****************************************************************************
; Demo Program for the ISP
;*****************************************************************************
IFD     DATA   0E2h
IFADRH   DATA   0E3h
IFADRL   DATA   0E4h
IFMT     DATA   0E5h
SCMD    DATA   0E6h
ISPCR    DATA   0E7h
;
        MOV    ISPCR,#10000000b  ;ISPCR.7=1, enable ISP

;=============================================================================
; 1. Page Erase Mode (64 bytes per page)
;=============================================================================
        CALL   WAIT_PBSY_FREE
MOV    IFMT,#0C3h ;MS[7: 0]=[1,1,0,0,0,0,1,1], select Page Erase Mode
        MOV    IFADRH,??  ;fill page address in IFADRH & IFADRL
        MOV    IFADRL,??  ;
        MOV    SCMD,#46h  ;trigger ISP processing
        MOV    SCMD,#0B9h ;
        ;Now in processing...(CPU will halt here until complete)

;=============================================================================
; 2. Page Program Mode (64 bytes per page)
;=============================================================================
        CALL   WAIT_PBSY_FREE
MOV    IFMT,#0C2h ;MS[7: 0]=[1,1,0,0,0,0,1,0], select Byte Write Mode
        CLR    A
Byte_Write_Loop:
        MOV    IFADRH,??  ;fill byte address in IFADRH & IFADRL
        MOV    IFADRL,??  ;
        MOV    IFD,??     ;fill the data to be programmed in IFD
        MOV    SCMD,#46h  ;trigger ISP processing
        MOV    SCMD,#0B9h ;
        ;Now in processing...(CPU will halt here until complete)
        INC     A
        CJNE   A,#40h, Byte_Write_Loop

        MOV    IFMT,#0C1h ;MS[7: 0]=[1,1,0,0,0,0,0,1], select Page Program Mode
        MOV    SCMD,#46h  ;trigger ISP processing
        MOV    SCMD,#0B9h ;
        ;Now in processing...(CPU will halt here until complete)

;=============================================================================
; 3. Verify using Read Mode
;=============================================================================
        CALL   WAIT_PBSY_FREE
        MOV    IFMT,#01h ;MS[7: 0]=[0,0,0,0,0,0,0,1], select Byte Read Mode
        MOV    IFADRH,??  ;fill byte address in IFADRH & IFADRL
        MOV    IFADRL,?? ;
        MOV    SCMD,#46h  ;trigger ISP processing
        MOV    SCMD,#0B9h ;
        ;Now in processing...(CPU will halt here until complete)
        MOV    A,IFD      ;data will be in IFD
CJNE   A,wanted,ISP_error ;compare with the wanted value
...

WAIT_PBSY_FREE:
        MOV    A,ISPCR ;
        JB     ACC.1,WAIT_PBSY_FREE ;
        RET

ISP_error:
...
```

```
;
```

27.6.2. EEPROM 范例程序

EEPROM 操作的范例程序如图 27–13 所示。

图 27–13. EEPROM 范例程序

```
;*****************************************************************************
; Demo Program for the EEPROM
;*****************************************************************************
;
IFD     DATA   0E2h
IFADRH   DATA   0E3h
IFADRL   DATA   0E4h
IFMT     DATA   0E5h
SCMD     DATA   0E6h
ISPCR    DATA   0E7h
;
        MOV    ISPCR,#10000000b  ;ISPCR.7=1, enable ISP


;==============================================================================
; 1. Byte Program Mode
;==============================================================================
        CALL   WAIT_PBSY_FREE
MOV     IFMT,#0C8h ;MS[7: 0]=[1,1,0,0,1,0,0,0], select EEPROM Byte Program Mode
        MOV    IFADRH,??  ;fill byte address in IFADRH & IFADRL
        MOV    IFADRL,??  ;
        MOV    IFD,??     ;fill the data to be programmed in IFD
        MOV    SCMD,#46h  ;trigger ISP processing
        MOV    SCMD,#0B9h ;
        CALL   WAIT_EEPF_SET
        ORL    ISPCR,#01h ;clear EEPF flag and byte program procedure is finish

;==============================================================================
; 2. Verify using Read Mode
;==============================================================================
        CALL   WAIT_PBSY_FREE
        MOV    IFMT,#0C0h ;MS[7: 0]=[1,1,0,0,0,0,0,0], select EEPROM Byte Read Mode
        MOV    IFADRH,??  ;fill byte address in IFADRH & IFADRL
        MOV    IFADRL,?? ;
        MOV    SCMD,#46h  ;trigger ISP processing
        MOV    SCMD,#0B9h ;
        CALL   WAIT_PBSY_FREE
        MOV    A,IFD      ;data will be in IFD
CJNE    A,wanted,EE_error ;compare with the wanted value
...

WAIT_PBSY_FREE:
        MOV    A,ISPCR ;
        JB     ACC.1,WAIT_PBSY_FREE ;
        RET

WAIT_EEPF_SET:
        MOV    A,ISPCR ;
        JNB    ACC.0, WAIT_EEPF_SET ;
        RET


EE_error:
...
;
```

28. P 页 SFR 访问

MG82F6B08 / 6B001/ 6B104 内建一个特别的 SFR 页 (P 页)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。如果 IFMT=05H 则 P 页读操作，在 SCMD 激活之后 IFD 的数据将是 IFADRL 索引的特殊功能寄存器(SFR)的值。

以下是 P 页的 SFR 功能定义：

IAPLB: IAP 低边界

SFR 页 = **仅 P 页**

SFR 地址 = 0x03 复位值 = 1111-111x

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

Bit 7~0：IAPLB 决定 IAP 存储区的最低边界。因为一个闪存页是 512 字节，所以 IAPLB 必须是偶数。
为了读取 IAPLB，MCU 需要在 P 页里定义 IFADRL 地址，IMFT 模式选择 P 页读及 ISPCR.ISPEN 置位。并且在 SCMD 依次写入 0x46h 和 0xB9h，这样 IAPLB 的值就会出现在 IFD。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 存储区如下。
IAP 低边界= IAPLB[7:0] x256
IAP 高边界= ISP 起始地址 – 1。
如果 IAPLB=0xE0 且 ISP 起始地址是 0xF000，这样 IAP 存储区位于 0xE000 ~ 0xEFFF。

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 = **仅 P 页**

SFR 地址 = 0x40 复位值 = 0101-0000

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

Bit 7~5：保留位，写 CKCON2 寄存器时，此位必须写“0”。

Bit 4：IHRCOE，内部高频 RC 震荡使能位。
0: 禁止内部高频 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，在 IHRCOE 位使能后，必须等待 **32 us** IHRCOE 才能稳定输出。

Bit 3~2：保留位，写 CKCON2 寄存器时，此位必须写“0”。

Bit 1~0：OSCS[1:0]，OSCin 时钟源选择。

OSCS[1:0]	OSCin 时钟源选择
0 0	IHRCO
0 1	保留
1 0	ILRCO
1 1	ECKI，外部时钟输入(P4.5)作为 OSCin。

CKCON3:时钟控制寄存器 3

SFR 页 = 仅 **P** 页

SFR 地址 = 0x41 复位值 = 0000-0110

7	6	5	4	3	2	1	0
WDTCS1	WDTCS0	FWKP	WDTFS	MCKD1	MCKD0	1	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6：WDTCS1~0。WDT 时钟源选择。

Bit 5：FWKP, MCU 快速唤醒控制。
0：选择 MCU 从掉电模式正常唤醒时间大约 **90~120us**。
1：选择 MCU 从掉电模式快速唤醒时间大约 **15~30us**。

Bit 4：WDTFS。WDT 溢出源选择。
0：选择 WDT 位-7 溢出作为 WDT 事件源。
1：选择 WDT 位-0 溢出作为 WDT 事件源。

Bit 3~2：MCKD[1:0]，MCK 分频输出选择。(MCKDO 默认 8MHz)

MCKD[1:0]	MCKDO 频率	若 MCK = 16MHz	若 MCK = 22MHz
0 0	MCKDO = MCK	MCKDO = 16MHz	MCKDO = 22MHz
0 1	MCKDO = MCK/2	MCKDO = 8MHz	MCKDO = 11MHz
1 0	MCKDO = MCK/4	MCKDO = 4MHz	MCKDO = 5.5MHz
1 1	MCKDO = MCK/8	MCKDO = 2MHz	MCKDO = 2.75MHz

Bit 1~0：MCDS[1:0]，保留用于测试。当 CKCON3 被写入时，软件必须写“10”。

CKCON4:时钟控制寄存器 4

SFR 页 = 仅 **P** 页

SFR 地址 = 0x42 复位值 = 0000-0000

7	6	5	4	3	2	1	0
RCSS2	RCSS1	RCSS0	RPSC2	RPSC1	RPSC0	RTCCS3	RTCCS2
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~5：RTC 时钟源选择[2:0]。

RCSS2, RCSS1, RCSS0	RTC 时钟源选择
0 0 0	ECKI (P4.5)
0 0 1	ILRCO
0 1 0	WDTPS
0 1 1	WDTOF
1 0 0	SYSCLK
1 0 1	SYSCLK / 12
1 1 0	保留
1 1 1	保留

PCON2:电源控制寄存器 2

SFR 页 = 仅 **P** 页

SFR 地址 = 0x44 POR = 0011-0101

7	6	5	4	3	2	1	0
AWBOD1	0	BO1S1	BO1S0	BO1RE	EBOD1	BO0RE	1
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：AWBOD1，掉电模式(PD)下 BOD1 的唤醒。
0：掉电模式(PD)下禁止 BOD1。
1：掉电模式(PD)下保持 BOD1。

Bit 6：保留位，写 PCON2 寄存器时，此位必须写“0”。

Bit 5~4：BO1S[1:0]。低电压监测器 1 监测电压选择。这两个位初始值是从硬件选项中的 BO1S1O 和 BO1S0O 载入。（BO1S2 在 PCON4.4）

BO1S[2:0]	BOD1 监测电压
0 0 0	保留
0 0 1	2.4V
0 1 0	3.6V
0 1 1	4.2V
1 0 0	2.7V
其它	保留

Bit 3：BO1RE，BOD1 复位使能。
0：当 BOF1 已经设置，禁止低电压监测 1(BOD1)系统复位。
1：当 BOF1 已经设置，使能低电压监测 1(BOD1)系统复位。

Bit 2：EBOD1，使能 BOD1 监测 VDD 下降到 BO1S1~0 设置的固定值。
0：禁止 BOD1 监测电源电压降低芯片功耗。
1：使能 BOD1 监测电源电压 VDD。

Bit 1：BO0RE，BOD0 复位使能。
0：当 BOF0 已经设置，禁止低电压监测 0(BOD0)系统复位。
1：当 BOF0 已经设置，使能低电压监测 0(BOD0)系统复位(VDD 触到 2.35V)。

Bit 0：保留位。写 PCON2 寄存器时，此位必须写“**1**”。

PCON3:电源控制寄存器 3
SFR 页 = **仅 P 页**
SFR 地址 = 0x45 POR = 0000-0000

7	6	5	4	3	2	1	0
IVREN	0	0	SPWRE	0	0	0	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：IVREN，内部参考电压使能。
0：禁止片内 IVR (1.4V)。
1：使能片内 IVR (1.4V)。

Bit 6~5：保留位，写 PCON3 寄存器时，此位必须写“0”。

Bit 4：SPWRE, SPWF 触发一个 MCU 复位(软件).
0: 禁止 SPWF 触发 MCU 复位。
1: 使能 SPWF 触发 MCU 复位。

Bit 3~0：保留位，写 PCON3 寄存器时，此位必须写“0”。

PCON4:电源控制寄存器 4
SFR 页 = **仅 P 页**
SFR 地址 = 0x46 POR = 0000-0000

7	6	5	4	3	2	1	0
0	0	0	BO1S2	0	0	0	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 4：BOD1 检测电压选择位 2。

SPCON0: SFR 页控制 0

SFR 页 = **仅 P 页**

SFR 地址 = 0x48 POR = 0000-0000

7	6	5	4	3	2	1	0
0	0	P4CTL	WRCTL	0	CKCTL0	PWCTL1	PWCTL0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6：保留位，写 SPCON 寄存器时，此位必须写“0”。

Bit 5：P4CTL，P4 SFR 访问控制。
如果 P4CTL 置位，则 P4 禁止在 0~F 页改写。P4 在 0~F 页保持读取。但是在 SFR P 页软件拥有改写权利。

Bit 4：WRCTL，WDTCR SFR 访问控制。
如果 WRCTL 置位，则 WRCTL 禁止在 0~F 页改写。WRCTL 在 0~F 页保持读取。但是在 SFR P 页软件拥有改写权利。

Bit 2：CKCTL0，CKCON0 SFR 访问控制。
如果 CKCTL0 置位，则 CKCON0 禁止在 0~F 页改写。CKCON0 在 0~F 页保持读取。但是在 SFR P 页软件拥有改写权利。

Bit 1：PWCTL1，PCON1 SFR 访问控制。
如果 PWCTL1 置位，则 PCON1 禁止在 0~F 页改写。PCON1 在 0~F 页保持读取。但是在 SFR P 页软件拥有改写权利。

Bit 0：PWCTL0，PCON0 SFR 访问控制。
如果 PWCTL0 置位，则 PCON0 禁止在 0~F 页改写。PCON0 在 0~F 页保持读取。但是在 SFR P 页软件拥有改写权利。

DCON0:设备控制寄存器 0

SFR 页 = **仅 P 页**

SFR 地址 = 0x4C 复位值 = 1000-0011

7	6	5	4	3	2	1	0
HSE	IAPO	0	0	0	IORCTL	RSTIO	OCDE
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**W**	**R/W**	**W**

Bit 7：HSE，高速运行使能。
0：选择 CPU 运行在低速模式(FCPUCLK≤48KHz)这样减慢内部电路从而降低功耗。

软件禁用 HSE 流程
步骤1：切换F_{CPUCLK} ≤ 48KHz。
步骤 2：设置 HSE = 0。

1：使能 MCU 全速运行(FCPUCLK>48KHz)。在 SYSCLK 选择高频时钟(>6MHz)之前，软件必须置位 HSE 切换到用于高速运行的内部电路。

软件使能 HSE 流程
步骤1：设置HSE = 1。
步骤 2：切换 F_{CPUCLK} >48KHz。

Bit 6：IAPO，仅 IAP 功能。
0：保留 IAP 区服务于 IAP 功能和程序代码执行。
1：IAP 区禁止程序代码执行并且仅服务于 IAP 功能

Bit 5~3：保留位，写 DNCON0 寄存器时，此位必须写“**0**”。

Bit 2：IORCTL，GPIO 复位控制。
0：端口 6(Port 6)所有复位事件下保持复位。
1：如果此位置位，端口 6(Port 6)仅通过 POR/LVR/Ext_Reset/BOR0/BOR1 (如果 BOR0/1 是使能的)复位。

Bit 1：RSTIO，RST 功能为 I/O。
0：选择 I/O 引脚功能为 P47。
1：选择 I/O 引脚功能为外部复位输入(RST)。

Bit 0：OCDE，OCD 使能。
0：在 P4.4 和 P4.5 禁止 OCD 接口。
1：在 P4.4 和 P4.5 使能 OCD 接口。

SPHB:堆栈指针高边界

SFR 页 = **仅 P 页**
SFR 地址 = 0x53 复位值 = 1111-1111

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

Bit 7~4：保留位，写 SPHB 寄存器时，此位必须写“**1**”。

如果 SPHB == 1111-1111, 当 SP ≥ 1111-1111，将置位 SPWF。
如果 SPHB == 1111-0000, 当 SP ≥ 1111-0000，将置位 SPWF。

29. 辅助特殊功能寄存器

AUXR0:辅助寄存器 0

SFR 页 = 0~F
SFR 地址 = 0xA1 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P45OC1	P45OC0	0	PBKF	0	0	INT1H	INT0H
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7~6：P4.5 功能配置控制位 1 和位 0。这两位仅仅当内部 RC 振荡（IHRCO 或 ILRCO）被选择为系统时钟源时有效。外部时钟输入模式下，P4.5 专用于时钟输入。在内部振荡模式， P4.5 为普通 I/O 或时钟源发生器提供下列选项。当 P45OC[1:0] 索引为非 P4.5 GPIO 功能时，P4.5 将驱动内部 RC 振荡器输出为其它设备提供时钟源。

P45OC[1:0]	P4.5 function	I/O mode
0 0	P4.5	By P4M1.5 & P4M0.5
0 1	MCK	By P4M1.5 & P4M0.5
1 0	MCK/2	By P4M1.5 & P4M0.5
1 1	MCK/4	By P4M1.5 & P4M0.5

了解详情，请参考章节“8 系统时钟”。P4.5 作为时钟输出功能时，建议设置{P4M1.5, P4M0.5}为“01”来选择 P4.5 为推挽输出模式。

Bit 5：保留位，写 AUXR0 寄存器时，此位必须写“**0**”。

Bit 4：PBKF，PWM 终止标志。此位由 PWM 终止源使能置位。如果此位置位，则使能的 PWM 通道 0~3 将被锁住并且输出引脚保持最初的 GPIO 状态。
0：没有 PWM 终止事件出现。仅由软件清零。
1：PWM 终止事件出现或软件触发一个 PWM 终止。

Bit 3~2：保留位，写 AUXR0 寄存器时，此位必须写“**0**”。

Bit 1：INT1H，INT1 高电平/上升沿触发使能。
0：保留 INT1 在选择的端口引脚上低电平或下降沿触发。
1：设置 INT1 在选择的端口引脚上高电平或上升沿触发。

Bit 0：INT0H，INT0 高电平/上升沿触发使能。
0：保留 INT0 在选择的端口引脚上低电平或下降沿触发。
1：设置 INT0 在选择的端口引脚上高电平或上升沿触发。

AUXR1:辅助控制寄存器 ***1***

SFR 页 = 0~F
SFR 地址 = 0xA2 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	0	CRCDS1	CRCDS0	0	0	0	DPS
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7~6：保留位，写 AUXR1 寄存器时，此位必须写“**0**”。

Bit 5~4：CRCDS1~0。CRC0 数据端口选择 1~0。

Bit 3~1：保留位，写 AUXR1 寄存器时，此位必须写“**0**”。

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

DPS	选择 DPTR
0	DPTR0
1	DPTR1

AUXR2:辅助控制寄存器 2

SFR 页 = 0~F

SFR 地址 = 0xA3 复位值 = 0000-0000

7	6	5	4	3	2	1	0
STAF	STOF	0	C0PLK	T1X12	T0X12	T1CKOE	T0CKOE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7：STAF，STWI/SI2C 的起始标志侦测。
0：软件写“0”清零。
1：硬件置位，表示在 STWI 总线上发生了一个起始动作。

Bit 6：STOF，STWI /SI2C 的停止标志侦测。
0：软件写“0”清零。
1：硬件置位，表示在 STWI 总线上发生了一个停止动作。

Bit 5：保留位，写寄存器时，此位必须写“0”。

Bit 4: C0PLK, PCA 缓冲 PWM/COPM 更新控制
0: 缓冲 PWM/COPM 在 PCA 基础定时器溢出时自动更新.
1:禁止缓冲 PWM/COPM 自动更新

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：置位选择 SYSCLK 作时钟源。

Bit 3：T0X12，当 C/T=0 时，定时器 0 时钟源选择。
0：清零选择 SYSCLK/12。
1：置位选择 SYSCLK 作时钟源。

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 时钟输出在 T1CKO 端口引脚。

Bit 0：T0CKOE，定时器 0 时钟输出使能。
0：禁止定时器 0 时钟输出。
1：使能定时器 0 时钟输出在 T0CKO 端口引脚。

AUXR3:辅助寄存器 3

SFR 页 = **仅 0 页**

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	T0PS0	BPOC1	BPOC0	S0PS0	TWIPS1	TWIPS0	T0XL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7：保留位，写寄存器时，此位必须写“**0**”。

Bit 6：T0PS0，定时器 0 端口引脚选择。

T0PS0	T0/T0CKO
0	P4.6
1	P4.4

Bit 5~4：BPOC1~0，蜂鸣器输出控制位。

BPOC[1:0]	P4.4 功能	I/O 模式
0 0	P4.4	By P4M0.4 & P4M1.4
0 1	ILRCO/32	By P4M0.4 & P4M1.4
1 0	ILRCO/16	By P4M0.4 & P4M1.4
1 1	ILRCO/8	By P4M0.4 & P4M1.4

P4.4 用于蜂鸣器功能，推荐设置 P4.4 工作在推挽输出模式。

Bit 3：S0PS0，串口 0 引脚选择 0。(S0PS1 在 AUXR10.3，SnMIPS 在 AUXR6.1)

SnMIPS, S0PS1~0	RXD0/ S0MOSI	TXD0/ S0SPCK	S0MISO	S0nSS
0 0 0	P3.0	P3.1	P33	P46
0 0 1	P4.4	P4.5	P33	P46
0 1 0	P4.5	P3.0	P33	P46
0 1 1	P4.5	P4.4	P33	P46
1 0 0 (1XX)	P4.4	P3.3	P31	P30

Bit 2~1：TWIPS1~0，TWI0/I2C0 端口选择[1:0]。

TWIPS1~0	TWI0_SCL	TWI0_SDA
0 0	P3.1	P3.0
0 1	P4.4	P4.5
1 0	P3.0	P3.1
1 1	P3.3	P4.6

Bit 0：是定时器 0 预分频控制位。T0XL 功能定义请参考 T0X12。

AUXR4:辅助寄存器 4

SFR 页 = **仅 1 页**

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
T2PS1	T2PS0	0	T1PS0	0	0	AC0OE	AC0FLT1
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6：T2PS1~0，定时器 2 端口引脚选择[1:0]。

T2PS1~0	T2/T2CKO	T2EX
0 0	P3.1	P3.0
0 1	P3.3	P4.6
1 0	P4.6	P3.3
1 1	P4.5	P4.4

Bit 5：保留位，写寄存器时，此位必须写“**0**”。

Bit 4：T1PS1~0，定时器 1 端口引脚选择。

T1PS0	T1/T1CKO
0	P3.3
1	P4.5

Bit 1：AC0OE，使能 AC0OUT 输出在端口引脚上。
0: 禁止 AC0OUT 输出在端口引脚上。
1: 使能 AC0OUT 输出在 P1.0 上.

Bit 0：AC0FLT1，AC0 输出滤波控制 1。

AUXR5:辅助寄存器 5

SFR 页 = **仅 2 页**

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	C0IC2S0	0	0	C0PS1	C0PS0	ECIPS0	C0COPS
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：保留位，写寄存器时，此位必须写“**0**”。

Bit 6：C0IC2S0，PCA0 输入通道 2 输入选择。

C1IC2S0	CEX2 input
0	CEX2 Port Pin
1	T2EXI

Bit 5 ~ 4：保留位，写寄存器时，此位必须写“**0**”。

Bit 3 ~ 2：C0PS[1:0]，PCA0 引脚选择位 1 & 0。

C0PS1~0	CEX0	CEX1	CEX2	CEX3
0 0	P3.0	P3.3	P3.1	P4.6
0 1	P4.4	P3.3	P4.5	P4.6
1 0	P4.4	P3.3	P3.0	P4.6
1 1	P4.4	P3.3	P1.0	P4.6

Bit 1：ECIPS0，PCA0 ECI 引脚选择位 0。

ECIPS0	ECI
0	P4.4
1	P1.0

Bit 0：C0COPS，PCA0 时钟输出 (C0CKO)引脚选择。

C0COPS	C0CKO
0	P4.7
1	P3.3

AUXR6:辅助寄存器 6

SFR 页 = **仅 3 页**

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	0	0	0	0	T2FCS	SnMIPS	S0COPS
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7 ~ 3：保留位，写寄存器时，此位必须写“**0**”。

Bit 3：T3FCS，保留用于芯片测试。

Bit 2：T2FCS，保留用于芯片测试。

Bit 1：SnMIP，串口 0 引脚选择位。

Bit 0：S0COPS，S0BRG 时钟输出(S0CKO) 引脚选择位。

S0COPS	S0CKO
0	P4.7
1	P3.3

AUXR7:辅助寄存器 7

SFR 页 = **仅 4 页**

SFR 地址 = 0xA4 复位值 = 1100-0000

7	6	5	4	3	2	1	0
1	1	C0CKOE	SPI0M0	0	0	0	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 5：C0CKOE，PCA0 时钟输出使能。
0：禁止 PCA0 时钟输出。
1：PCA0 基准定时器溢出率的二分之一时钟输出使能。

Bit 4：SPI0M0，SPI0 模式控制位 0.它控制 SPI 菊花链应用。
0：禁用模式控制。
1：使能模式控制。

AUXR9: 辅助寄存器 9

SFR 页 = **仅 6 页**

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
0	SIDPS0	T1G1	T0G1	C0FDC1	C0FDC0	0	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：保留位，写寄存器时，此位必须写“**0**”。

Bit 6：SID/STWI 引脚选择位。

SIDPS0	STWI_SCL	STWI_SDA
0	nINT1	S0MI
1	TWI0_SCL	TWI0_SDA

Bit 5：T1G1，定时器 1 门控源选择。

T1G1, T1GATE	T1 门控源
0 0	禁止
0 1	INT1
1 0	TF2
1 1	KBI

Bit 4：T0G1，定时器 0 门控源选择。

T0G1, T0GATE	T0 门控源
0 0	禁止
0 1	INT0
1 0	TF2
1 1	KBI

Bit 3~2：C0FDC1~0，C0FDCK 选择[1:0]。

C0FDC1~0	C0FDCK
0 0	T0OF
0 1	T1OF
1 0	T2OF
1 1	S0TOF

Bit 1~0：保留位，写寄存器时，此位必须写“**0**”。

AUXR10:辅助寄存器 10

SFR 页 = **仅 7 页**

SFR 地址 = 0xA4 复位值 = 1100-0000

7	6	5	4	3	2	1	0
1	AC0HC0	0	SPIPS0	S0PS1	SPFACE	TWICF	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 6：AC0HC0，AC0 滞环控制 0。
0：禁止在 AC0 上滞环输入。
1：使能在 AC0 上滞环输入。默认是使能。

Bit 3：S0PS1, 串口 0 引脚选择位 1。(此功能在 AUXR3.3，S0PS0 中已描述)

Bit 2：SPFACE，SPIF 自动清零使能位。
0：禁止，SPIF 只能软件清零。
1：使能，SPIF 会被 CPU 读/写 SPDAT 操作清零。

Bit 1：TWICF，TWI0/I2C0 串行时钟输入滤波。
0：禁止 TWICF 功能。
1：使能 TWICF 功能。

AUXR11:辅助寄存器 11

SFR 页 = **仅 8 页**

SFR 地址 = 0xA4 复位值 = 0000-0000

7	6	5	4	3	2	1	0
P30AM	P33AM	0	0	0	POEM0	C0M0	C0OFS
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7：P30AM，P3.0 模拟输入模式使能。
0：P3.0 GPIO 模式被 P3M0 和 P3M1 控制。
1：强制 P3.0 作为模拟输入模式用于 ADC12 的 AIN4 通道输入。

Bit 6：P33AM，P3.3 模拟输入模式使能。
0：P3.3 GPIO 模式被 P3M0 和 P3M1 控制。
1：强制 P3.3 作为模拟输入模式用于 ADC12 的 AIN5 通道输入。

Bit 2：POEM0，PCA0 POEn 控制 0。
0：POEn 功能在 CPU 写入后立即生效。
1：POEn 功能与 PWM 周期对齐。

Bit 1：C0M0，PCA 模式控制 0。
0：PWM 中心对齐不支持可变分辨率。
1：使能 PCA 支持可变分辨率的 PWM 中心对齐。使能此功能，PCAE (PWMCR.7)也需要置位。

Bit 0：C0OFS，当 C0M0 使能 PCA 溢出标志选择。
0：PWM 中心对齐周期的顶部 CF 置位。
1：PWM 中心对齐周期的低部 CF 置位。

SFRPI: SFR 页索引寄存器

SFR 页 = 0~F

SFR 地址 = 0xAC 复位值 = xxxx-0000

7	6	5	4	3	2	1	0
--	--	--	--	IDX3	IDX2	IDX1	IDX0
W	**W**	**W**	**W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~4：保留位。写寄存器时，此位必须写“0”。

Bit 3~0：SFR 页索引。

IDX[3:0]	可选页
0000	页 0
0001	页 1
0010	页 2
0011	页 3
……	……
……	……
……	……
1111	页 F

30. 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 ICE Adapter”(这个 ICE 也支持 ICP 编程功能。参考章节“31.4 在芯片编程功能”)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。**MG82F6B08 / 6B001/ 6B104** 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：**0x1FFF**。下表列举了 ISP 空间选项。默认设定，**MG82F6B08 / 6B001/ 6B104** 空间被配置为 **1.5K**，并嵌入了 Megawin COMBO ISP 引导码,通过 Megawin 1-线 ISP 协议和串口 ISP 协议，进行在线设备 FW 更新。

ISP 空间大小	**MG82F6B08 / 6B001/ 6B104** ISP 起始地址
3.5K bytes	1200
3.0K bytes	1400
2.5K bytes	1600
2.0K bytes	1800
1.5K bytes	1A00
1.0K bytes	1C00
0.5K bytes	1E00
No ISP Space	--

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 0 字节。

BO1S1O, BO1S0O:
☑,☑：选择 BOD1 监测电压 2.4V。
☑,☐：选择 BOD1 监测电压 3.6V。
☐,☑：选择 BOD1 监测电压 4.2V。
☐,☐：选择 BOD1 监测电压 2.7V。

BO0REO:
☑：使能。BOD0 将触发复位事件使得 CPU 从 AP 程序起始地址运行(1.7V)。
☐：禁止。BOD0 不能触发 CPU 复位。

BO1REO:
☑：使能。BOD1 (4.2V、3.7V、2.4V 或 2.0V)将触发复位事件使得 CPU 从 AP 程序起始地址运行。
☐：禁止。BOD1 不能触发 CPU 复位。若想使用软件选择 BOD1 检测电压，不要使能 BO1REO，请参照章节“11.5 低电压检测复位”以获取详情。

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 运行(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。

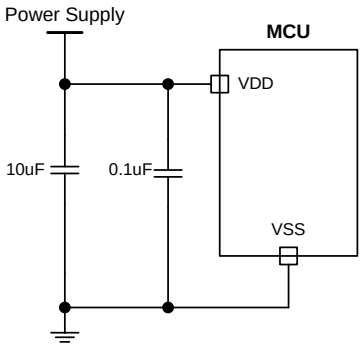
☐：禁止。WDT 特殊寄存器，WDTCR 的 WREN、NSW、WIDL、PS2、PS1 和 PS0 位，由软件自由写。

31. 应用说明

31.1. 电源电路

MG82F6B08 / 6B001/ 6B104 的工作电源变化可以从 2.4V 到 5.5V 但是增加一些外部去耦和滤波电容是必须的，如图 31–1 所示。

图 31–1. 电源电路



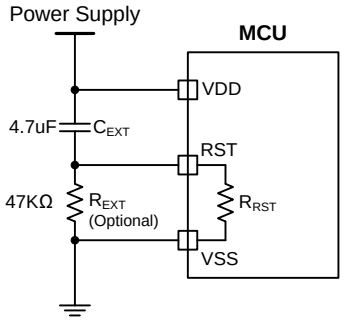
31.2. 复位电路

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 31–2 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。

RRST 的值见章节“32.2 直流特性”。

图 31–2. 复位电路



31.3. ICP 和 OCD 接口电路

MG82F6B08 / 6B001/ 6B104 包含一个笙泉专有的在芯片调试接口，它允许在元器件已经安装在产品上在芯片编程(ICP)和在线调试(OCD)。ICP 和 OCD 共享同样的接口使用一个时钟线(ICP_SCL/OCD_SCL)和一个双向数据线(ICP_SDA/OCD_SDA)完成主机与设备之间的数据传送。

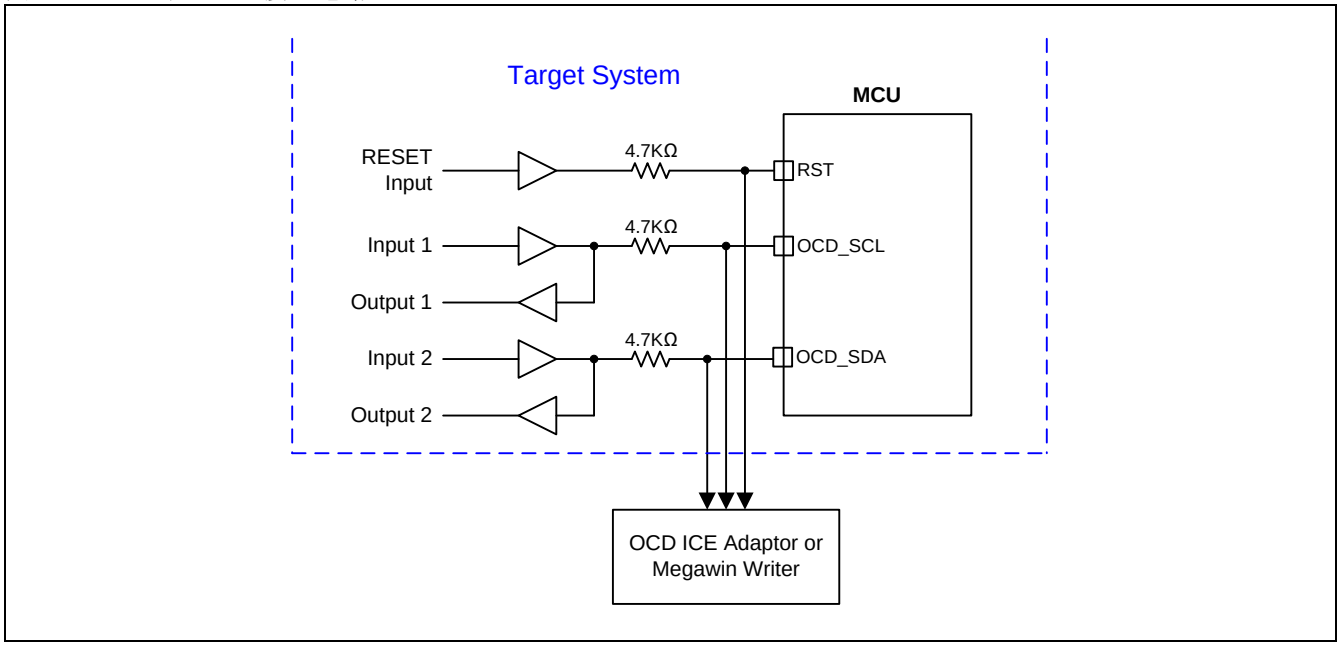
ICP 接口允许的 ICP_SCL/ICP_SDA 引脚与用户应用共享，使得可以实现在芯片 FLASH 编程。这是可行的，因为当芯片在 Halt 状态时执行 ICP 通信，此时芯片上的外围设备和用户软件都是失效的。在 halt 状态，ICP 接口能够安全的“借用”ICP_SCL (P4.4)和 ICP_SDA (P4.5)引脚。在大多应用中，必须用外部电阻来隔开 ICP 电路和用户应用电路。一种典型的隔离方法如图 31–3 所示。

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。

上电后，**MG82F6B08 / 6B001/ 6B104** 的 P4.4 和 P4.5 被配置成 OCD_SCL/OCD_SDA 用于在线调试功能。这是可行的，因为 OCD 通信是在 CPU Halt 状态下执行，此时用户软件是无效的。在 halt 状态，OCD 接口可以安全的使用 OCD_SCL(P4.4)和 OSC_SDA(P4.5)引脚。就像上面提到的隔离 ICP 接口，如图 31–3，用外部电阻来隔开 ICP 电路和用户应用电路。

如果用户放弃 OCD 功能，软件可以通过清零 PCON3 的位 0(OCDE)来配置 OCD_SCL 和 OCD_SDA 引脚作为 P4.4 和 P4.5。当用户想重新使用 OCD 功能，用户可以置 OCDE 为 1 来切换 P4.4 和 P4.5 到 OCD_SCL 和 OCD_SDA。或者用 ICP“擦除”在芯片 FLASH 清除用户软件来停止端口的却换。

图 31–3. ICP 和 OCD 接口电路



31.4. 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(经由在线调试通道)，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。

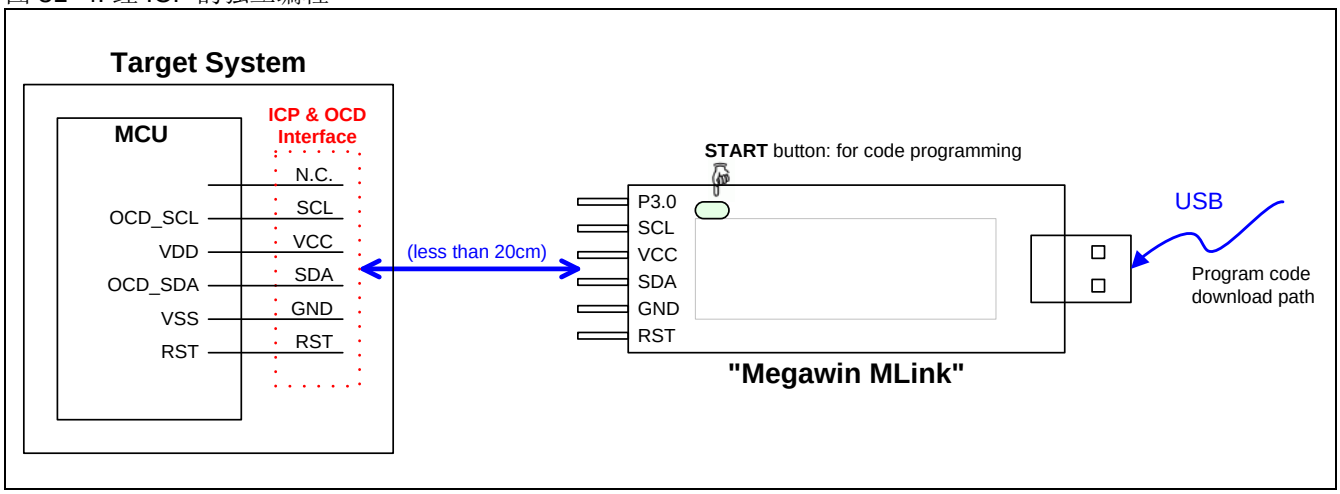
专用的 6 脚“Megawin 8051 ICE Adapter”可以支持 **MG82F6B08 / 6B001/ 6B104** 在线路编程。“Megawin 8051 ICE Adapter”有在系统的存储器来存储用户的程序和器件选项。因此，该工具可以完成一个便携的，独立的编程，而不用连线主机，如连接该工具到 PC。下面列举了 ICP 功能的特点：

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。ICP 独立编程的系统框图如图 31–4 所示。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 脚“Megawin 8051 ICE Adapter”传送编程数据到目标 MCU； RST 线用来暂停 MCU；VCC 和 GND 是 6 脚“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 脚“Megawin 8051 ICE Adapter”。

图 31–4. 经 ICP 的独立编程



31.5. 在线调试功能

MG82F6B08 / 6B001/ 6B104 预备了一个用于在线仿真(ICE)的 Megawin 专用的在线调试(OCD)接口。这个 OCD 接口提供在芯片和系统不干扰的调试，且不占用任何的目标系统资源。支持 ICE 的几种必要操作，如复位，运行，停止，单步运行，运行到光标和断点设置。

使用 OCD 技术，Megawin 提供“Megawin 8051 OCD ICE”给用户，如图 31–5 所示。用户在开发过程中不必准备任何的开发板，或者用在传统 ICE 探头的转换座。所有这些，用户仅需在系统上保留一个 6-脚的连接器用于专用的 OCD 接口：P3.0、RST、VCC、OCD_SDA、OCD_SCL 和 GND ，如图 31–5 所示

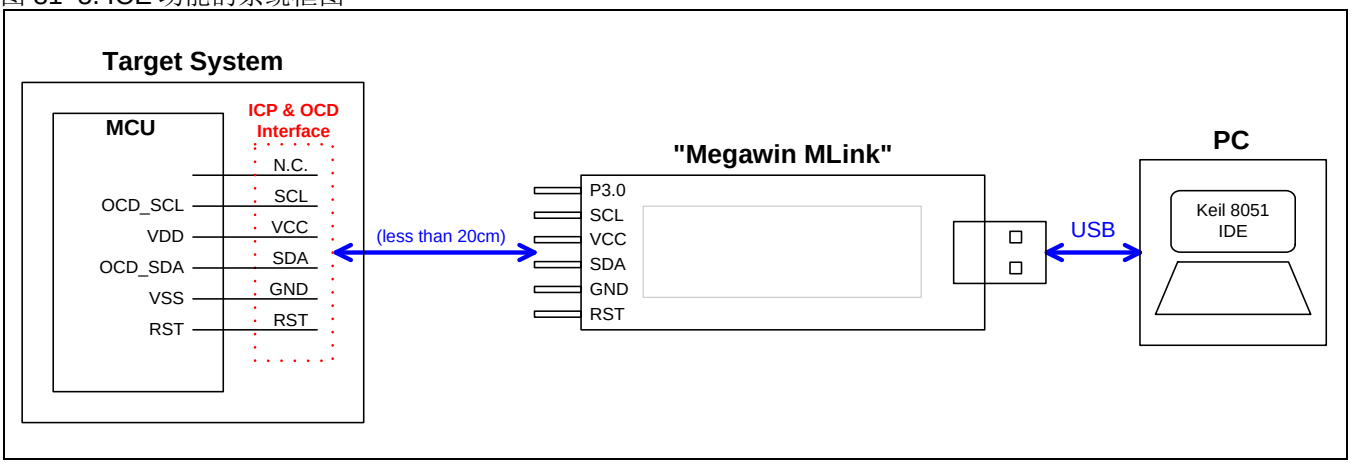
另外，最有力的功能是，它可以直接让用户的系统连接到 Keil 8051 IDE 软件进行仿真，它直接利用 Keil IDE’s dScope-Debugger 功能。当然，所有的特点都基于你使用的 Keil 8051 IDE 软件。

注意： “Keil ”是“Keil Elektronik GmbH and Keil Software, Inc. ”的注册商标。

特点

- 笙泉科技专用的 OCD(在芯片调试)技术
- 在芯片和在系统实时调试
- 5 针 OCD 专用串行接口，不占用目标资源
- 直接连接 Keil IDE 软件的调试功能
- USB 连接目标板与主机(PC)
- 有用的调试操作：复位，运行，停止，单步运行和运行到光标
- 可编程断点，可在仿真中插入 4 个断点
- 数个帮助调试串口：寄存器/反汇编/监视/存储区窗口
- 源代码级(汇编或 C 语言)调试能力

图 31–5. ICE 功能的系统框图



注意：有关 OCD ICE 的更多详细信息，请联系笙泉。

32. 电气特性

32.1. 最大绝对额定值

参数	范围	单位
环境温度	-40 ~ +85	°C
存储温度	-65 ~ + 150	°C
任意 GPIO 口或 RST 对地电压	-0.5 ~ VDD + 0.5	V
VDD 对地电压	-0.5 ~ +6.0	V
VDD 到地的最大电流	200	mA
任意引脚最大灌电流	40	mA

**注意：实际参数超过上述各项“绝对最大额定值”可能会对设备造成永久性损坏。这些参数是一个设备进行正常功能操作的应力额定值，任何超过上述各项的条件都不被建议，否则可能会影响设备运行的稳定性。长时间暴露在最高值状态也可能影响设备稳定性。*

32.2. 直流特性

VDD = 5.0V±10%, VSS = 0V, T_A = 25 °C，每个机器周期执行 NOP，除非另有说明

标号	参数	测试环境	极限			单位
			最小	典型	最大	
输入/输出特性						
V_{IH1}	输入高电平(所有 I/O 口)	Except RST	0.6			VDD
V_{IH2}	输入高电平(RST)		0.75			VDD
V_{IL1}	输入低电平(所有 I/O 口)	Except RST			0.15	VDD
V_{IL2}	输入低电平(RST)				0.2	VDD
I_{IH}	输入高漏电流 (所有 I/O 口)	V_{PIN} = VDD		0	±1	uA
I_{IL1}	逻辑 0 输入电流(P3 在准双向口模式或片内上拉电阻的输入端口)	V_{PIN} = 0.4V		-20	-30	uA
I_{IL2}	逻辑 0 输入电流(所有仅输入或开漏输出口)	V_{PIN} = 0.4V		0	-1	uA
I_{H2L}	逻辑 1 到 0 输入转变电流 (P3 在准双向口模式) (2)	V_{PIN} =1.8V		-300	-450	uA
I_{OH1}	输出高电流(P3 在准双向口模式)	VDD=5V; V_{PIN} =2.4V	-180	-285		uA
		VDD=3.3V; V_{PIN} =2.4V	-50	-80		uA
		VDD=2.4V; V_{PIN} =2.0V	-20	-28		uA
I_{OH2}	输出高电流(所有推挽输出口)	VDD=5V; V_{PIN} =2.4V	-32	-37		mA
		VDD=3.3V; V_{PIN} =2.4V	-9	-12		mA
		VDD=2.4V; V_{PIN} =2.0V	-3	-4.5		mA
I_{OH3}	输出高电流(在低驱动力时所有推挽输出口，除 RST 引脚外)	VDD=5V; V_{PIN} =2.4V	-7.5	-12.5		mA
		VDD=3.3V; V_{PIN} =2.4V	-3	-4.5		mA
		VDD=2.4V; V_{PIN} =2.0V	-0.8	-1.5		mA
I_{OL1}	输出低电流(所有 I/O 口)	VDD=5V; V_{PIN} =0.4V	15	21		mA
		VDD=3.3V; V_{PIN} =0.4V	13	16.5		mA
		VDD=2.4V; VPIN =0.4V	9	12		mA
I_{OL2}	输出低电流(在低驱动力时所有推挽输出口，除 RST 引脚外)	VDD=5V; V_{PIN} =0.4V	1.8	2.5		mA
		VDD=3.3V; V_{PIN} =0.4V	1.2	1.99		mA
		VDD=2.4V; VPIN =0.4V	0.8	1.4		mA
R_{RST}	内部复位下拉电阻	VDD=5V		130		Kohm
		VDD=3.3V		215		Kohm
		VDD=2.4V		350		Kohm
功耗						
I_{OP1}	正常模式工作电流	SYSCLK = 16MHz @ IHRCO		5.3		mA
I_{OP2}		SYSCLK = 16MHz @ IHRCO, VDD = 5V with ADC 400K sps		6.7		mA
I_{OPS1}	低速模式工作电流	SYSCLK = 16MHz /128 @ IHRCO		0.8		mA
I_{IDLE1}	空闲模式工作电流	SYSCLK = 16MHz @ IHRCO		2.5		mA
I_{IDLE2}		SYSCLK = 16MHz /128 @ IHRCO		0.75		mA
I_{IDLE3}		SYSCLK = 32KHz @ ILRCO		33		uA
I_{SUB1}	副频模式工作电流	SYSCLK = 32KHz @ ILRCO, BOD1 禁止		40		uA
I_{SUB2}		SYSCLK = 32KHz/128 @ ILRCO, BOD1 禁止		30		uA
I_{WAT}	Watch 模式工作电流	WDT = 32KHz @ ILRCO 在掉电模式		8.6		uA
I_{MON1}	Monitor 模式工作电流	BOD1 使能在掉电模式		7.9		uA
I_{RTC1}	RTC 模式操作电流	RTC 运行在掉电模式，VDD = 5.0V		8.6		uA
I_{PD1}	掉电模式电流			1.3		uA
BOD0/BOD1 特性						
V_{BOD0}	BOD0 监测电压	T_A = -40° C to +85°C		2.35		V
V_{BOD10}	BOD1 监测电压为 2.7V	T_A = -40° C to +85° C		2.7		V
V_{BOD10}	BOD1 监测电压为 2.4V	T_A = -40° C to +85° C		2.4		V
V_{BOD11}	BOD1 监测电压为 3.6V	T_A = -40° C to +85° C		3.6		V

V_{BOD11}	BOD1 监测电压为 4.2V	T_A = -40° C to +85° C		4.2		V
I_{BOD1}	BOD1 功耗	T_A = +25°C, VDD=5.0V				uA
		T_A = +25°C, VDD=3.3V				
工作环境						
V_{PSR}	上电边沿速率	T_A = -40°C to +85°C	0.05			V/ms
V_{POR1}	上电复位有效电压	T_A = -40°C to +85°C			0.1	V
V_{OP1}	CPU 工作速度 0-16MHz	T_A = -40° C to +85° C	2.7		5.5	V
V_{OP2}	CPU 工作速度 0-12MHz	T_A = -40° C to +85° C	2.4		5.5	V

(1) 数据基于特性所得，非产品测试。

(2) I/O 在准双向模式下，当输入电压从高转向低并且越过闸值电压，内部“弱上拉”将关闭。I_{H2L} 表示的是电流接近闸值电压时的电流，详见参考“图 13–1. 端口 3 准双向口结构”。

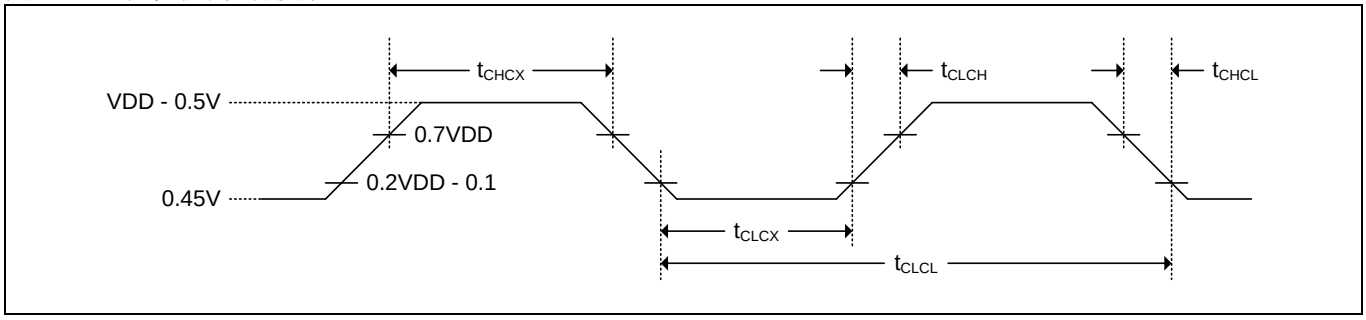
(3) 所有电流流入芯片是正值，电流流出芯片为负值。

32.3. 外部时钟特性

VDD = 2.4V ~ 5.5V, VSS = 0V, T_A = -40°C to +85°C，除非另有说明

标号	参数	晶振 ECKI 模式		单位
		最小	最大	
$1/t_{CLCL}$	晶振频率	0	24	MHz
$1/t_{CLCL}$	晶振频率 (VDD = 2.4V ~ 5.5V)	0	12	MHz
t_{CLCL}	时钟周期	27.7		ns
t_{CHCX}	高时间	0.4T	0.6T	t_{CLCL}
t_{CLCX}	低时间	0.4T	0.6T	t_{CLCL}
t_{CLCH}	上升时间		5	ns
t_{CHCL}	下降时间		5	ns

图 32–1. 外部时钟驱动波形



32.4. IHRCO 特性

参数	测试环境	极限			单位
		最小	典型	最大	
电源电压	IHRCO = 16MHz	2.4		5.5	V
	IHRCO = 22.12MHz	4.5		5.5	V
IHRCO 频率	T_A = +25°C, AFS = 0		16		MHz
	T_A = +25°C, AFS = 1		22.12		MHz
IHRCO= 16MHz 频率误差 (工厂校对)	TA = +25°C	-2.0		+2.0	%
	TA = -40°C to +85°C	-3.8(1)		+3.8(1)	%
	TA = -40°C to +85°C, VDD = 5V	-3.8(1)		+3.8(1)	%
	TA = -40°C to +85°C, VDD = 3.3V	-3.4(1)		+3.4(1)	%
	TA = -40°C to +85°C, VDD = 2.4V	-3.6(1)		+3.6(1)	%
IHRCO= 22.12MHz 频率误差(工厂校对)	TA = +25°C	-1.2(1)		+1.5(1)	%
	TA = -40°C to +85°C	-3.0(1)		+3.0(1)	%
IHRCO= 16MHz @ MCK/2, MCK/4, MCK/8 與 MCK/1 之频率误差	TA = -40°C to +85°C, VDD = 5V	+1(1)		+1.9(1)	%
	TA = -40°C to +85°C, VDD = 3.3V	-0.8(1)		+0(1)	%
	TA = -40°C to +85°C, VDD = 2.4V	-1.45(1)		+0.5(1)	%
IHRCO= 22.12MHz MCK/2, MCK/4, MCK/8 與 MCK/1 之频率误差	TA = -40°C to +85°C, VDD = 5V	-2.4(1)		-0.5(1)	%
IHRCO 启动时间	TA = -40°C to +85°C			32(1)	us
IHRCO 功耗	TA = +25°C, VDD=5.0V		TBD (1)		uA

(1) 数据基于特性所得，非产品测试。

32.5. ILRCO 特性

参数	测试环境	极限			单位
		最小	典型	最大	
电源电压		2.4		5.5	V
ILRCO 频率	TA = +25°C		32		KHz
ILRCO 频率误差	TA = +25°C	-8(1)		+8(1)	%
	TA = -40°C to +85°C	-20(1)		+20(1)	%

(1) 数据基于特性所得，非产品测试。

32.6. Flash 特性

参数	测试环境	极限			单位
		最小	典型	最大	
电源电压	TA = -40°C to +85°C	2.4		5.5	V
Flash 写 (擦除/编程)电压	TA = -40°C to +85°C	2.4		5.5	V
Flash 擦除/编程周期	TA = -40°C to +85°C	20,000			次
Flash 数据保留期	TA = +25°C	100			年
Flash 擦除时间	CPUCLK = 8MHz		3．3		ms
Flash 页编程时间	CPUCLK = 8MHz		1.75		ms
Flash 字节锁存时间	CPUCLK = 8MHz		1.75		us
	CPUCLK = 16MHz		880		ns
Flash 字节读时间	CPUCLK = 8MHz		1.375		us
	CPUCLK = 16MHz		690		ns

32.7. EEPROM 特性

参数	测试环境	极限			单位
		最小	典型	最大	
电源电压	TA = -40°C to +85°C	2.4		5.5	V
EEPROM 编程电压	TA = -40°C to +85°C	2.4		5.5	V
EEPROM 编程周期	TA = -40°C to +85°C	100,000			次
EEPROM 数据保留期	TA = +25°C	100			年
EEPROM 字节读时间	CPUCLK=8MHz		4.9		us
	CPUCLK=16MHz		2.5		us
EEPROM 字节编程时间	CPUCLK=8MHz		4.25		ms

32.8. ADC 特性

VDD=5.0V, T_A= -40°C ~ +85°C，除非另有说明

参数	测试环境	极限			单位
		最小	典型	最大	
电源电压		2.4		5.5	V
分辨率			10		位
DC 精度 T_A= -40°C ~ +85°C					
整体非线性	VDD = VREF+ ≥ 3.3V, 666K sps, ADC 功耗等级= 中低功耗 (ADPS[1:0] = 10b)	-1.3		4.2	
	VDD = VREF+ = 2.4V, 666K sps, ADC 功耗等级= 中低功耗 (ADPS[1:0] = 10b)	-1.4		1.5	
差分非线性	VDD = VREF+ ≥ 3.3V, 666K sps, ADC 功耗等级= 中低功耗 (ADPS[1:0] = 10b)	-1.0		1.2	LSB
	VDD = VREF+ = 2.4V, 666K sps, ADC 功耗等级= 中低功耗 (ADPS[1:0] = 10b)	-1.0		1.4	

偏移错误	VDD= 2.4V~5.5V	-3		+3	LSB
转换率					
SAR 转换时钟			16		MHz
转换时间(SAR 时钟为单位)			24		时钟
建议转换率	VDD = VREF+ ≥ 2.4V, ADC power level = Middle low (ADPS[1:0] = 10b)			666	K sps
模拟输入					
输入电压范围	单端(AIN+ – GND)	0		VDD	V
C_{ADC} 输入电容 note1			4.3	6	pF
采样输入开关电阻 note1	VDD = 5V		470 note2	990	Ω
	VDD = 4.2V		530 note2	1.18K	Ω
	VDD = 3.3V		680 note2	1.78K	Ω
	VDD = 2.7V		1K note2	3.43K	Ω
	VDD = 2.4V		1.47K note2	7.16K	Ω
通道切换稳定时间					
原引脚和目标引脚是在 VDD 或 GND 之间切换电压	CH0(VDD)→CH1(GND)		0		us
	CH0(GND)→CH1(VDD)		0		
原引脚电压=VDD，切换到带下拉电阻的目标引脚 r	CH0(VDD)→CH1(51K 下拉)		6.69		
	CH0(VDD)→CH1(10K 下拉)		1.06		
原引脚电压=GND，切换到带上拉电阻的目标引脚	CH0(GND)→CH1(51K 上拉)		7.81		
	CH0(GND)→CH1(10K 上拉)		1.56		
原引脚电压=VDD，切换到电阻分压（VDD/2）	CH0(VDD)→CH1(VDD/2，51K 电阻分压)		2.88		
	CH0(VDD)→CH1(VDD/2，10K 电阻分压)		3.13		
原引脚电压=VDD，切换到电阻分压（VDD/2）	CH0(GND)→CH1(VDD/2，51K 电阻分压)		0.38		
	CH0(GND)→CH1(VDD/2，10K 电阻分压)		0.44		
功耗					
电源电流	ADPS<1:0>=00		1.160		mA
	ADPS<1:0>=01		1.150		
	ADPS<1:0>=10		1.130		
	ADPS<1:0>=11		1.105		

(1) 数据由设计保证，非产品测试

(2) 输入电压 < ½VDD

32.9. IVR 特性

VDD=5.0V±10%, VSS=0V, T_A =-40°C to +85°C, C_{LOAD}=4.7upF/0.1ohm-ESR 除非另有说明

参数	测试环境	极限			单位
		最小	典型	最大	
电源范围					
电源电压		2.4	5.0	5.5	V
功耗	普通功耗状态	43		67	uA
	低功耗状态		0.1		uA
DC 精度					
输出电压	-40°C ~ +85°C	1.37	1.4	1.43	V
电压变化到整个温度范围	VDD = 3.3V±10mV			13	mV

32.10. 模拟比较器 AC0 特性

VDD=5.0V, T_A= -40° C ~ +85°C 除非另有说明

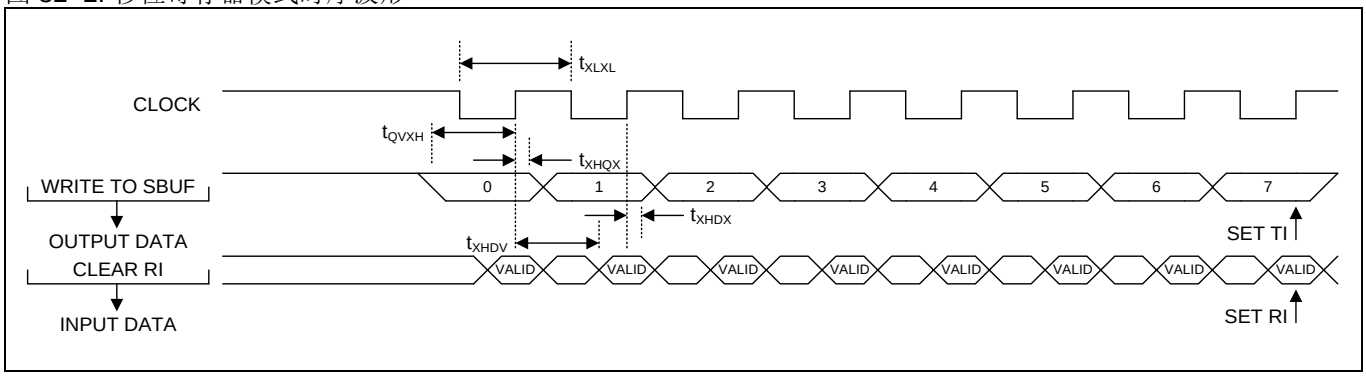
参数	测试环境	极限			单位
		最小	典型	最大	
电源范围					
电源电压		2.4	5.0	5.5	V
工作电流	Normal Power State				uA
	Low Power State				uA
R-Ladder (32 阶)	VDD= 5.0V				uA
R-Ladder (24 阶)	VDD= 5.0V				uA
DC 精度					
输入电压范围	Rail to Rail	50		VDD-900	mV
AC0 输入偏移电压 (正常模式)	VDD= 5.0V		4	10	mV
	VDD= 3.0V		4	10	mV
AC0 输入偏移电压 (低功耗模式)	VDD= 5.0V		5	11	mV
	VDD= 3.0V		5.5	11	mV
输入共模电压		50		VDD-900	mV
AC0 比较器滞环电压 (正常模式)	VDD= 5.0V		2	4.0	mV
	VDD= 3.0V		1.5	2.5	
AC0 比较器滞环电压 (低功耗模式)	VDD= 5.0V		2	4	mV
	VDD= 3.0V		1.5	2.5	
响应时间(正常模式)	上升(V_{DD} = 5V, Ta = 25° C, V_{OD}=50mV, V_{CM} < 50mV)		760		ns
	下降(V_{DD} = 5V, Ta = 25° C, V_{OD}=50mV, V_{CM} < 50mV)		450		ns
	上升(V_{DD} ≥ 3V, V_{OD}=500mV, 0.5V ≤ V_{CM} ≤ VDD-0.9V)		780		ns
	下降(V_{DD} ≥ 3V, V_{OD}=500mV, 0.5V ≤ V_{CM} ≤ VDD-0.9V)		500		ns
响应时间(低功耗模式)	上升(V_{DD} = 5V, Ta = 25° C, V_{OD}=50mV, V_{CM} < 50mV)		1.2		us
	下降(V_{DD} = 5V, Ta = 25° C, V_{OD}=50mV, V_{CM} < 50mV)		0.6		us
	上升(V_{DD} ≥ 3V, V_{OD}=500mV, 0.5V ≤ V_{CM} ≤ VDD-0.9V)		1.27		us
	下降(V_{DD} ≥ 3V, V_{OD}=500mV, 0.5V ≤ V_{CM} ≤ VDD-0.9V)		0.7		us

32.11. 串口时序特性

VDD = 5.0V±10%, VSS = 0V, T_A = -40°C to +85°C，除非另有说明

标号	参数	URM0X3 = 0		URM0X3 = 1		单位
		最小	最大	最小	最大	
t_{XLXL}	串口时钟周期	12T		4T		T_{SYSCLK}
t_{QVXH}	设置输出数据到时钟上升沿	10T-20		2T-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	5		5		ns
t_{XHDV}	时钟上升沿到输入数据有效		2T-10		2T-10	ns

图 32–2. 移位寄存器模式时序波形



32.12. SPI 时序特性

VDD = 5.0V±10%, VSS = 0V, T_A = -40°C to +85°C，除非另有说明

标号	参数	最小	最大	单位
主机模式时序				
1/(t_{MCKH} + t_{MCKL})	SPI 时钟频率@VDD = 2.4V ~ 5.5V		22.12MHz	MHz
t_{MCKH}	SPICLK 高时间	1T		T_{SYSCLK}
t_{MCKL}	SPICLK 低时间	1T		T_{SYSCLK}
t_{MIS}	MISO 有效到 SPICLK 采样边沿	10		ns
t_{MIH}	SPICLK 转变边沿到 MISO 变化	0		ns
t_{MOH}	SPICLK 转变边沿到 MOSI 变化		10	ns
从机模式时序				
1/(t_{CKH} + t_{CKL})	SPI 时钟频率@VDD = 2.4V ~ 5.5V		11.06MHz	MHz
t_{SE}	nSS 下降沿到第一个 SPICLK 边沿	2T		T_{SYSCLK}
t_{SD}	最后一个 SPICLK 边沿到 nSS 上升沿	2T		T_{SYSCLK}
t_{SEZ}	nSS 下降沿到 MISO 有效		4T	T_{SYSCLK}
t_{SDZ}	nSS 上升沿到 MISO 高阻		4T	T_{SYSCLK}
t_{CKH}	SPICLK 高时间	2T		T_{SYSCLK}
t_{CKL}	SPICLK 低时间	2T		T_{SYSCLK}
t_{SIS}	MOSI 有效到 SPICLK 采样边沿	1T		T_{SYSCLK}
t_{SIH}	SPICLK 采样边沿到 MOSI 变化	1T		T_{SYSCLK}
t_{SOH}	SPICLK 移位边沿到 MISO 变化		2T	T_{SYSCLK}
t_{SLH}	最后的 SPICLK 边沿到 MISO 变化 (仅 CPHA = 1)	1T	2T	T_{SYSCLK}

图 32–3. CPHA=0 时 SPI 主机传送波形

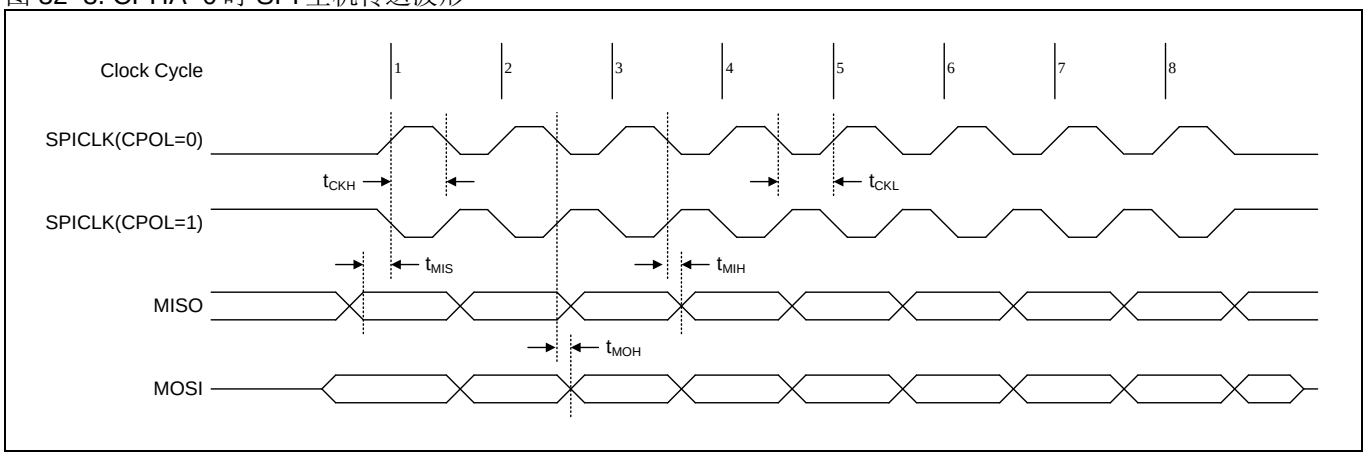


图 32–4. CPHA=1 时 SPI 主机传送波形

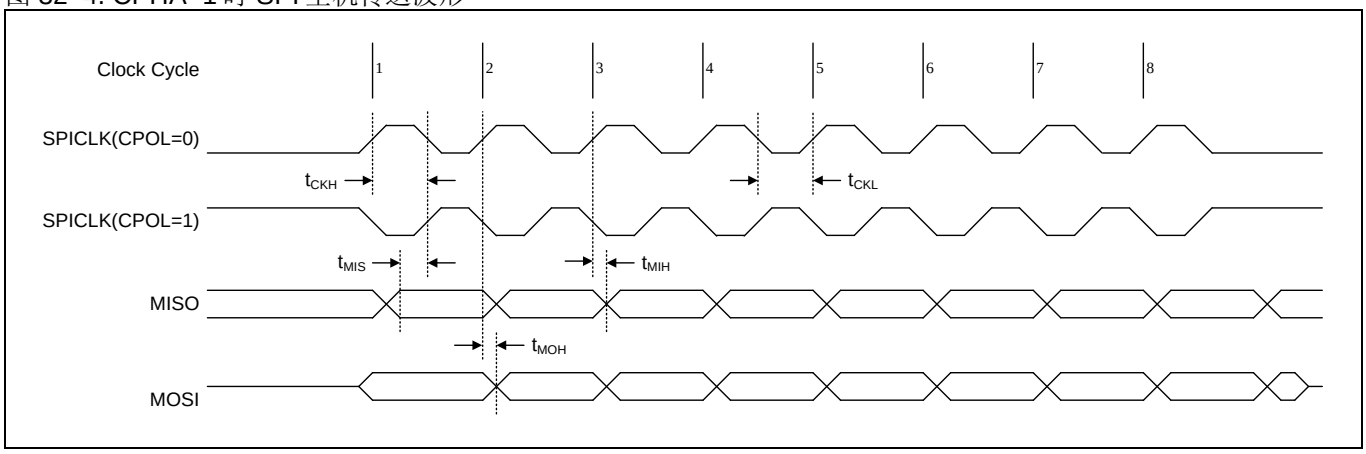


图 32–5. SPI CPHA=0 时 SPI 从机传送波形

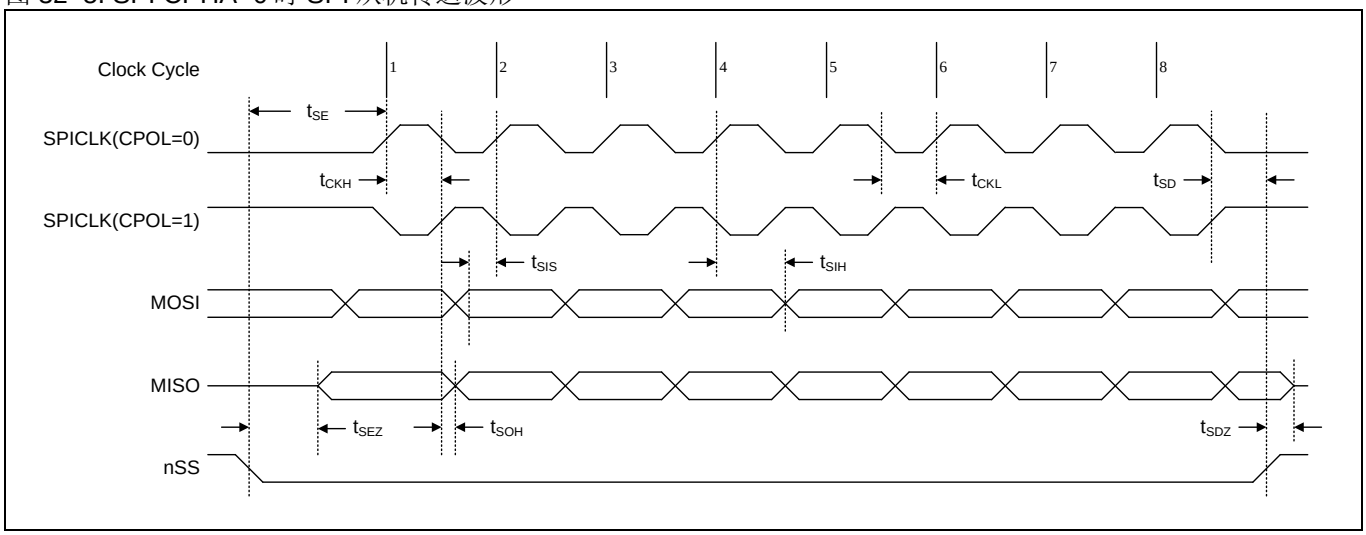
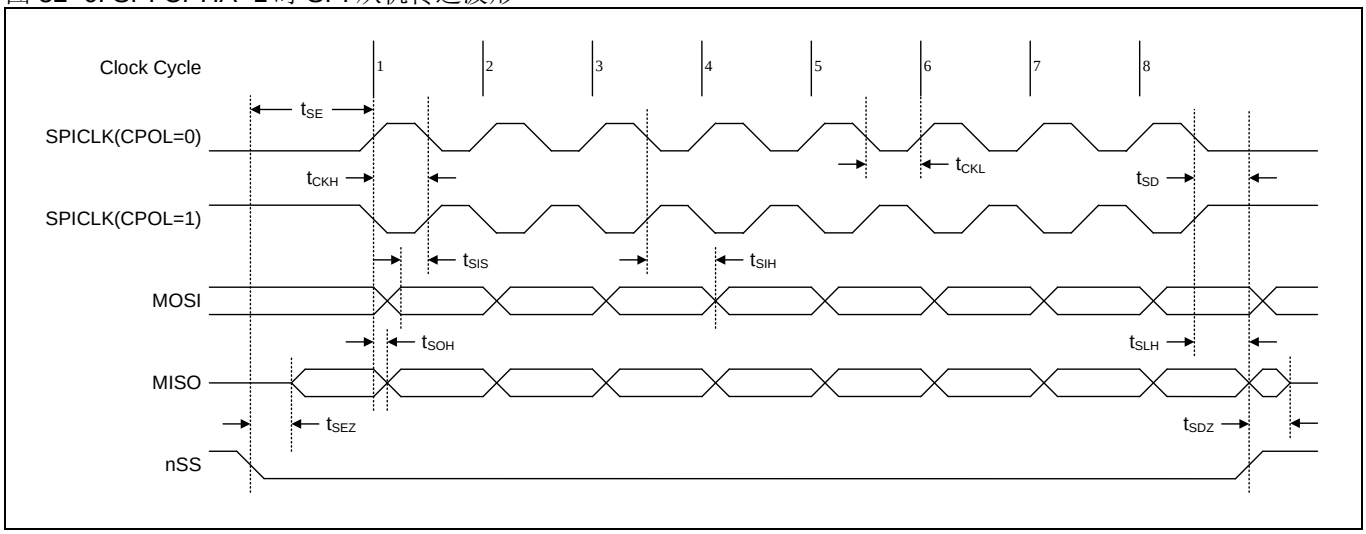


图 32–6. SPI CPHA=1 时 SPI 从机传送波形



33. 指令集

表 33–1. 指令集

助记符	描述	字节	执行周期
数据传送			
MOV A,Rn	寄存器 Rn 中的内容送到累加器中	1	1
MOV A,direct	直接地址单元中的内容送到累加器中	2	2
MOV A,@Ri	工作寄存器 Ri 指向的地址单元中的内容送到累加器中	1	2
MOV A,#data	立即数送到累加器中	2	2
MOV Rn,A	累加器中内容送到寄存器 Rn 中	1	2
MOV Rn,direct	直接寻址单元中的内容送到寄存器 Rn 中	2	4
MOV Rn,#data	立即数直接送到寄存器 Rn 中	2	2
MOV direct,A	累加器送到直接地址单元	2	3
MOV direct,Rn	寄存器 Rn 中的内容送到直接地址单元	2	3
MOV direct,direct	直接地址单元中的内容送到另一个直接地址单元	3	4
MOV direct,@Ri	工作寄存器 Ri 指向的地址单元中的内容送到直接地址单元	2	4
MOV direct,#data	立即数送到直接地址单元	3	3
MOV @Ri,A	累加器送到以工作寄存器 Ri 指向的地址单元中	1	3
MOV @Ri,direct	直接地址单元中内容送到以工作寄存器 Ri 指向的地址单元中	2	3
MOV @Ri,#data	立即数送到以工作寄存器 Ri 指向的地址单元中	2	3
MOV DPTR,#data16	16 位常数的高 8 位送到 DPH，低 8 位送到 DPL	3	3
MOVC A,@A+DPTR	以 DPTR 为基地址变址寻址单元中的内容送到累加器中	1	4
MOVC A,@A+PC	以 PC 为基地址变址寻址单元中的内容送到累加器中	1	4
MOVX A,@Ri	内置 RAM(8 位地址)的数据送入累加器中	1	3
MOVX A,@DPTR	寄存器 RI 指向扩展 RAM 地址(8 位地址)中的内容送到 ACC 中	1	3
MOVX @Ri,A	数据指针指向扩展 RAM 地址(16 位地址)中的内容送到 ACC 中	1	3
MOVX @DPTR,A	累加器中的内容送到寄存器 RI 指向的扩展 RAM 地址(8 位地址)	1	3
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			
ADD A,Rn	将寄存器 RN 中的内容加到累加器中	1	2
ADD A,direct	直接地址单元中的内容加到累加器中	2	3
ADD A,@Ri	寄存器工作寄存器 RI 指向的地址单元中的内容加到累加器中	1	3
ADD A,#data	立即数加到累加器中	2	2
ADDC A,Rn	累加器与工作寄存器 RN 中的内容、连同进位位相加，结果存在	1	2
ADDC A,direct	累加器与直接地址单元的内容、连同进位位相加，结果存在累加	2	3
ADDC A,@Ri	累加器与工作寄存器 RI 指向的地址单元中的内容、连同进位位	1	3
ADDC A,#data	累加器与立即数、连同进位位相加，结果存在累加器中	2	2
SUBB A,Rn	累加器与工作寄存器中的内容、连同借位位相减，结果存在累加	1	2
SUBB A,direct	累加器与直接地址单元中的内容、连同借位位相减，结果存在累	2	3
SUBB A,@Ri	累加器与工作寄存器 RI 指向的地址单元中内容、连同借位位相	1	3
SUBB A,#data	累加器与立即数、连同借位位相减，结果存在累加器中	2	2
INC A	累加器中的内容加 1	1	2
INC Rn	寄存器 RN 的内容加 1	1	3
INC direct	直接地址单元中的内容加 1	2	4
INC @Ri	工作寄存器 RI 指向的地址单元中的内容加 1	1	4

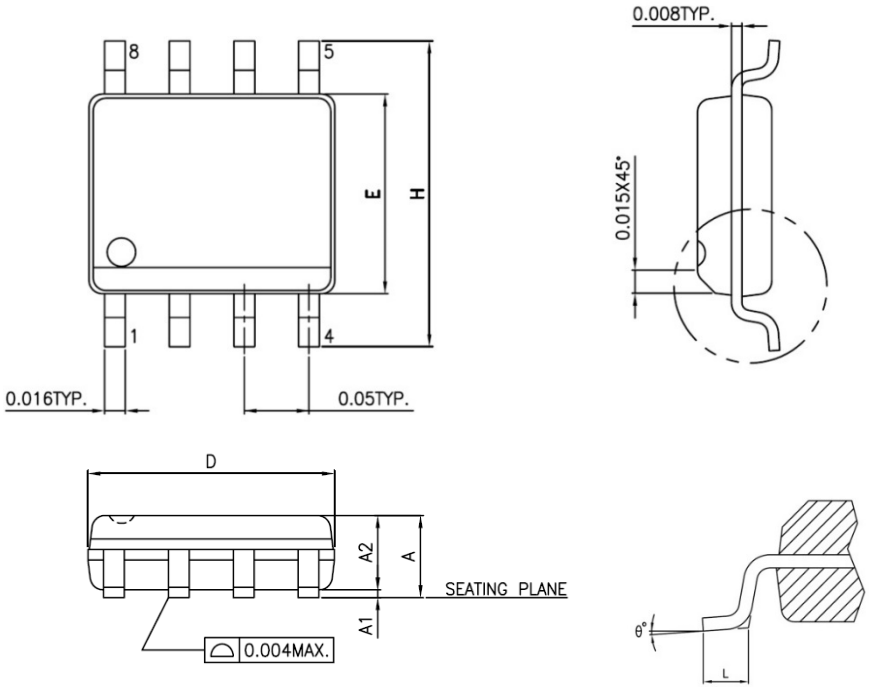
助记符	描述	字节	执行周期
DEC A	数据指针 DPTR 的内容加 1	1	2
DEC Rn	累加器中的内容减 1	1	3
DEC direct	寄存器 RN 中的内容减 1	2	4
DEC @Ri	直接地址单元中的内容减 1	1	4
INC DPTR	工作寄存器 RI 指向的地址单元中的内容减 1	1	1
MUL AB	ACC 中内容与寄存器 B 中内容相乘，其结果低位存在 ACC 中、	1	4
DIV AB	ACC 中内容除以寄存器 B 中内容，商存在 ACC，而余数存在寄	1	5
DA A	ACC 十进制调整	1	4
逻辑运算			
ANL A,Rn	累加器和寄存器 RN 中的内容相“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相 “与”	2	3
ORL 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
MOV C,bit	进位位和直接地址位的反码相 “或”	2	3
MOV bit,C	直接地址位数据送入进位位	2	4

助记符	描述	字节	执行周期
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	累加器不等于直接地址单元的内容，则转移到偏移量所指向的地	3	5
CJNE A,#data,rel	累加器不等于立即数，则转移到偏移量所指向的地址，否则程序	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	3	4
CJNE @Ri,#data,rel	RI 指向的地址单元不等于立即数，则转移到偏移量所指向的地	3	5
DJNZ Rn,rel	寄存器 RN 减 1，如不等于 0，则转移到偏移量所指向的地址，	2	4
DJNZ direct,rel	直接地址单元减 1，如不等于 0，则转移到偏移量所指向的地址	3	5
NOP	空操作指令	1	1

34. 封装尺寸

34.1. SOP-8 (150mil) 尺寸

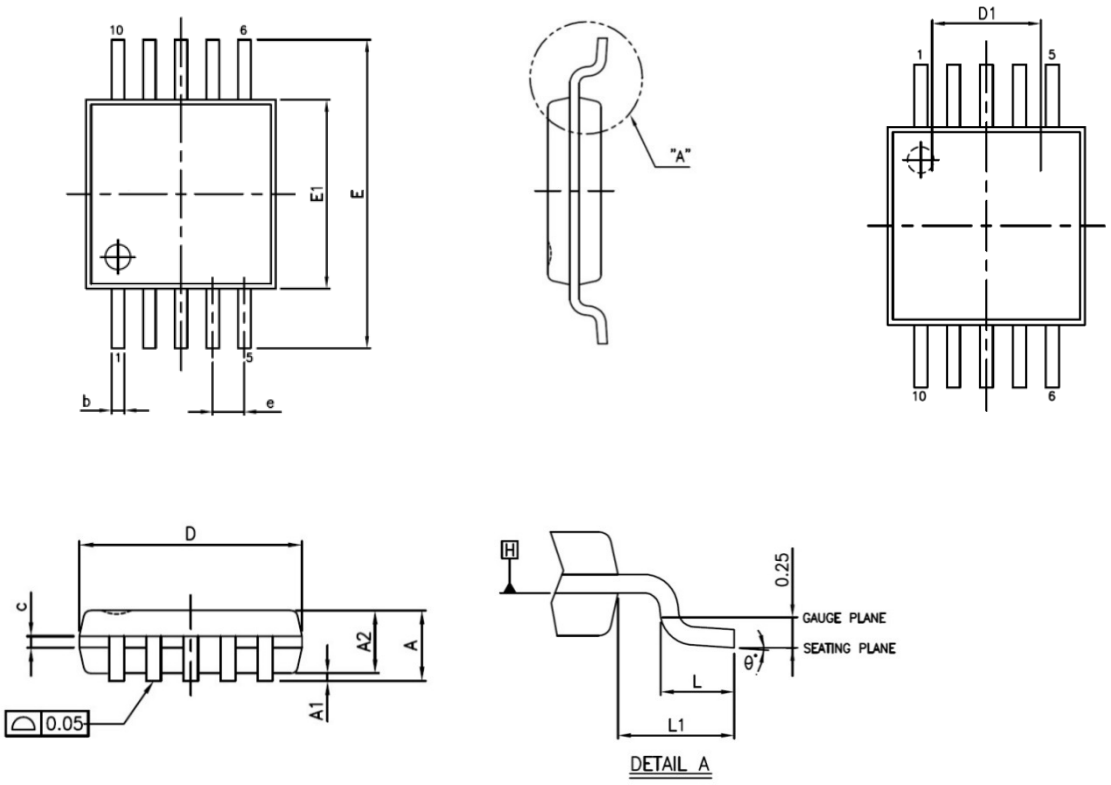
图 34-1. SOP-8 (150 mil)尺寸



单位	毫米		英寸	
标号	最小	最大	最小	最大
A	1.346	1.752	0.052	0.068
A1	0.101	0.254	0.003	0.010
A2	1.346	1.498	0.052	0.058
D	4.800	4.978	0.188	0.195
E	3.810	3.987	0.150	0.156
H	5.791	6.197	0.227	0.243
L	0.406	1.270	0.015	0.050
θ	0°	8°	0	8

34.2. MSOP-10 (150mil)尺寸

图 34-2. MSOP-10 (3.0x3.0x0.85)



单位	毫米			英寸		
标号	最小	典型	最大	最小	典型	最大
A	----	----	1.10	----	----	0.043
A1	0.00	----	0.15	0.000	----	0.005
A2	0.75	0.85	0.95	0.029	0.033	0.037
b	0.17	----	0.27	0.006	----	0.010
c	0.08	----	0.23	0.003	----	0.009
D	3.00 BSC			0.118 BSC		
E	4.90 BSC			0.192 BSC		
E1	3.00 BSC			0.118 BSC		
e	0.50 BSC			0.019 BSC		
L	0.40	0.60	0.80	0.015	0.023	0.031
L1	0.95 REF			0.037 REF		
ɵ	0	----	8	0	----	8

35. 版本历史

表 35–1. 版本历史

版本	描述	日期
0.18	1. 初始版本。	2021/04/20
0.19	1. 在PCON1中修正BOF0和BOF1的电压。 2. 在表13-2中修正P33M为P33AM。 3. 在PDRVC1中修正Bit 7~0为 Bit 0。 4. 在图 15-9中修正Timer0为Timer1。 5. 在章节16.2中移除PCA时钟描述。 6. 修改CCON描述，CCF5改为CCF3。 7. 移除表16-1的注意 8. 修改PCA通道的相关描述，6通道改为4通道 9. 修正章节17.10中RXD0和TXD0的引脚分配。 10. 移除章节18.5中的QPIEN。 11. 修正Flash页面大小，512改为64字节。 12. 在章节27.3，修正PBSY=0的描述，ISP/IAP引擎改为EEPROM引擎。 13. 在章节27.3，增加描述解释为什么在Flash操作前需要检测PBSY。. 14. 在章节31.4和31.5中，用MLink替代8051 OCD ICE。 15. 在IFMT中增加3种Flash访问模式。 16. 在章节27.3.3中增加ISP/IAP字节编程流程，并且在章节27.3.2中修改ISP/IAP页编程流程。	2021/05/30
1.00	1. 增加10位ADC的相关资料。 2. 增加C0PLK的描述。 3. 修改Timer1模式3的描述。 4. 修改KBI的描述。 5. 增加提高ADC精度的方法。5 6. 移除Timer2 分割模式 6。 7. 在章节32.3中移除晶振模式。 8. 在章节32.4和32.5中，IHRCO和ILRCO的电源电压1.8V修改为2.4V。 9. 在章节32.12中修改SPI时钟。	2022/05/06
1.01	1. 修正17.8.4.2节S0波特率公式的错字 2. 修正第 32.2 节中斜率的错字 3. 修正第10章RTC时钟的错字 4. 更正10.1节RTCTM的页面信息 5. 修正15.3.2节T2RLC功能说明。 强制重载模式仅在占空比模式下不可用。 6. 修正 WDTFS 功能说明，Bit 0：选择 WDT bit-7。 7. 将 IHRCO 公差从 4.5% 修改为 3%	2022/05/29
1.02	1. 将26.3小节的IVR(2.4V)修正为IVR(1.4V)	2022/06/15
1.03	1. 修正IHRCO 规格 2. 增加ADC特性数据 3. 修正表4-1及表4-2 PIN脚号错误	2023/05/05
1.04	1. 修改IHRCO规范描述	2023/05/10
1.05	1. 将图4-4中的MG82F6B104添加ADC备用PIN 2. 修复了特性章节中的IHRCO误差	2023/07/18

36. 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.