Megawin

MG84FL54B 全速 USB 八位微处理器

用户手册

（第Ⅰ版）

原著：By Vincent Y. C. Yu

策划/整理： 许 意 义

翻译：www.ourDEV.cn 网友

ckvlhf
yplin27
mutoudonggua 木头东瓜
litchiate 草真多

校对：www.ourDEV.cn 网友

lukeunderwood

1. 概述

MG84FL54B是基于高级嵌入Flash工艺的增强型单片式8位微处理器。指令集与8051完全兼容。藉由增强型内核，使其完成一条指令只需要1-7个时钟周期。相比与需要12-48个时钟周期才能完成一条指令的传统8051单片机，它有着更高的性能。因此，在与8051具有相同的处理能力的情况下，系统频率可以更低，从而能大大降低系统功耗。

MG84FL54B拥有16KB可并行编程（通过通用编程器）和ISP编程（通过USB DFU）的Flash存储器。 ISP功能可以让产品在不需要取下微控制器的情况下更新程序。固件更新功能大大扩展了它的应用领域。另外一个很有用的功能是在应用编程 (IAP),，它使设备可以利用Flash存储器保存非易失性数据。

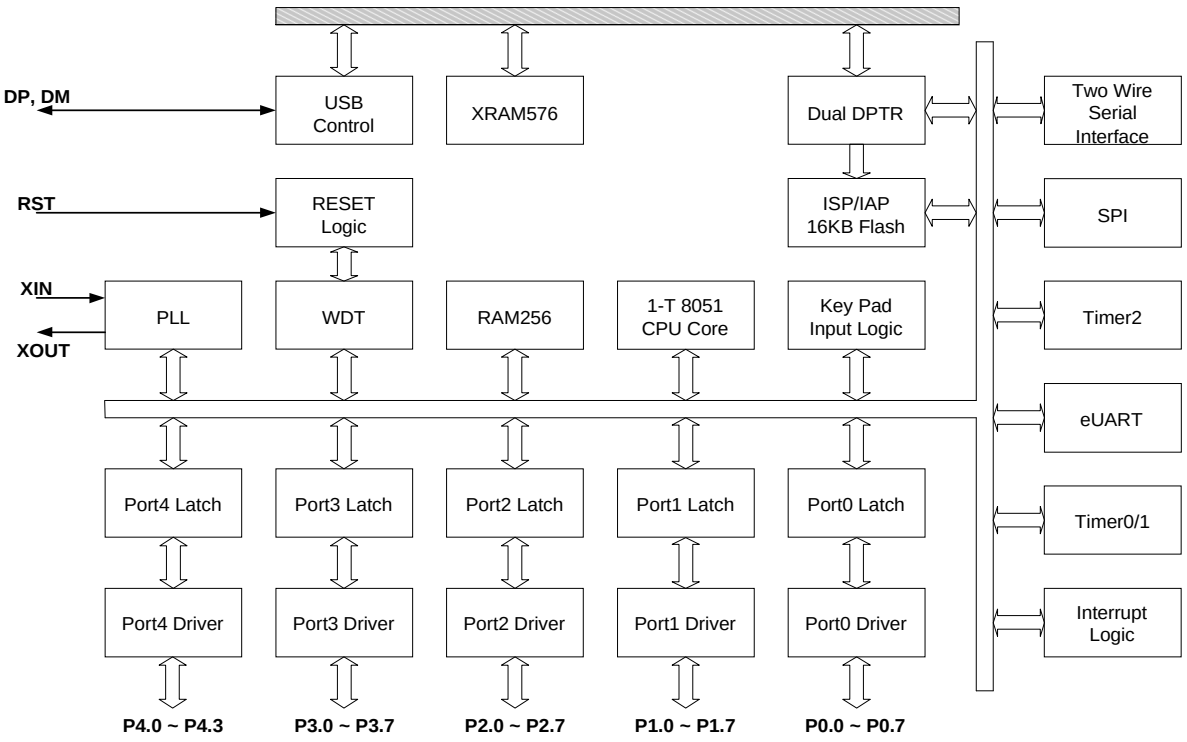
MG84FL54B除了256字节的RAM外，还在片内扩展了576字节的外部存储器（XRAM）。具有4个8位I/O端口和1个4位I/O端口，3个16位定时/计数器，多个中断源两级优先级的中断结构，一个增强型UART接口。更多新增的功能例如KBI，SPI，TWSI总线和USB1.1使它成为适合广泛应用领域的强大微控制器。

2. 特性

- ⌘ 1-T 8051内核
- ⌘ 片上16Kb具有ISP/IAP功能的Flash程序存储器
- ⌘ 256字节RAM与片上576字节外部RAM（XRAM）
- ⌘ 双数据指针
- ⌘ 四个半 I/O 端口
- ⌘ 三个16位定时器
- ⌘ 增强型串口
- ⌘ 两级中断优先级
- ⌘ 附加外部中断，INT2与INT3
- ⌘ 键盘中断 (P0)
- ⌘ 掉电唤醒模式
- ⌘ 串行外围接口(SPI)
- ⌘ 两线串行接口(TWSI)
- ⌘ 单次使能看门狗定时器(WDT)
- ⌘ 可编程系统时钟
- ⌘ USB 2.0且兼容1.1
  - 内置全速(12Mbps)USB收发器
  - 控 制 指 令 兼 容 Intel 8X931
  - 一个256字节FIFO为USB共享缓冲
    - ◴最大64字节EP0 控制输入/输出(control-in/out)缓冲
    - ◴最大64字节EP1 批量/中断输入（bulk/interrupt-in）缓冲
    - ◴最大64字节EP2 批量/中断/同步输入（bulk/interrupt/isochronous-in）缓冲，可以在批量（bulk）与同步（isochronous）操作下配置为双32字节双数据缓冲模式
    - ◴最大64字节EP3 批量/中断/同步输出（bulk/interrupt/isochronous-out）模式，可以在批量（bulk）与同步（isochronous）操作下配置为双32字节双数据缓冲模式。此外，还可以把EP3设置成中断输入（interrupt-in）缓冲。
  - 支持USB暂停/恢复与远程唤醒
  - 软件控制的USB连接/断开机制
  - 支持USB DFU
- ⌘ 省电模式
  - 空闲模式
  - 掉电模式
- ⌘ 工作电压
  - VDD_IO ： 2.4 ~ 5.5V， VDD_CORE与VDD_PLL ： 2.7V ~ 3.6V，VDDA ： 3.0V~3.6V
  - 内 置 低电压复位电路
- ⌘ 工作温度
  - 工业级 (-40°C to +85°C)*
- ⌘ 最高系统频率
  - 高达24MHz的工业范围
- ⌘ Flash 质量标准:
  - Flash 数据寿命:20K次擦/写循环
  - Flash 数据保存: 室温下100年
- ⌘ 双重代码保护
- ⌘ 封装: LQFP-48

*: 样品测试

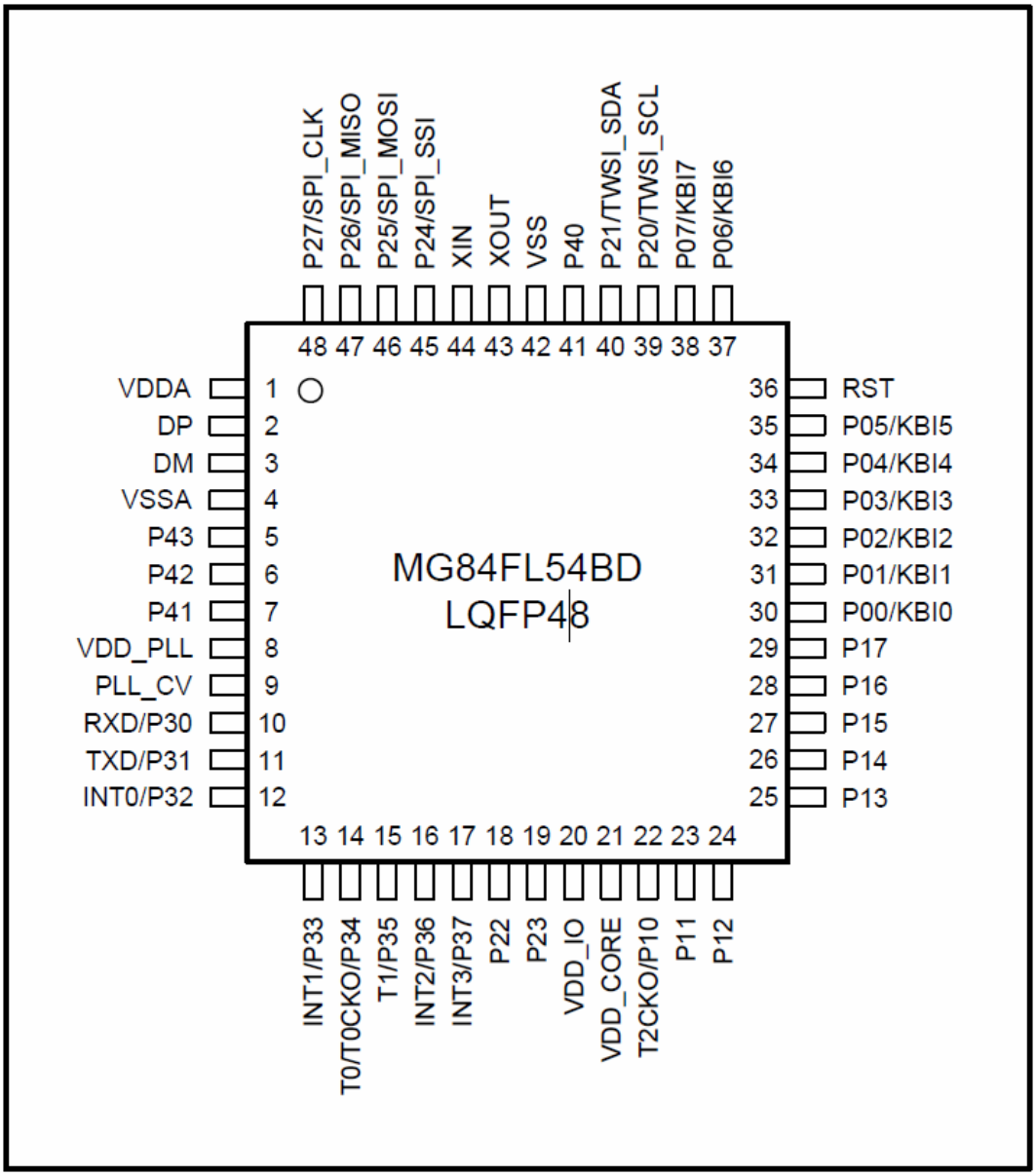
3. 方框图



MG84FL54B Block Diagram

4. 引脚结构

4.1. 48引脚 LQFP 封装



4.2. 引脚定义

引脚序号	名称	类型	描述
1	VDDA	P	3.3V 模拟电源
2	DP	I/O	USB DP
3	DM	I/O	USB DM
4	VSSA	P	模拟地
5	P4.3	I/O	P4.3
6	P4.2	I/O	P4.2
7	P4.1	I/O	P4.1
8	VDD_PLL	P	PLL电源输入
9	PLL_CV	I/O	内部PLL基准
10	P3.0 /RXD	I/O	P3.0，串口接收端
11	P3.1 /TXD	I/O	P3.1，串口发送端
12	P3.2 /INT0	I/O	P3.2，外中断0
13	P3.3 /INT1	I/O	P3.3，外中断1
14	P3.4 /T0 /T0CKO	I/O	P3.4，定时器0外部输入，定时器0时钟输出
15	P3.5 /T1	I/O	P3.5，定时器1外部输入
16	P3.6 /INT2	I/O	P3.6，外中断2
17	P3.7 /INT3	I/O	P3.7，外中断3
18	P2.2	I/O	P2.2.
19	P2.3	I/O	P2.3.
20	VDD_IO	P	I/O引脚数字电源
21	VDD_CORE	P	I/O内部逻辑数字电源
22	P1.0 /T2CKO	I/O	P1.0，定时器2时钟输出
23	P1.1	I/O	P1.1
24	P1.2	I/O	P1.2.
25	P1.3	I/O	P1.3.
26	P1.4	I/O	P1.4.
27	P1.5	I/O	P1.5.
28	P1.6	I/O	P1.6.
29	P1.7	I/O	P1.7.
30	P0.0	I/O	P0.0，键盘输入0
31	P0.1	I/O	P0.1，键盘输入1
32	P0.2	I/O	P0.2，键盘输入2.
33	P0.3	I/O	P0.3，键盘输入3.

34	P0.4	I/O	P0.4，键盘输入4.
35	P0.5	I/O	P0.5，键盘输入5.
36	RST	I	系统复位，高电平有效
37	P0.6	I/O	P0.6，键盘输入6.
38	P0.7	I/O	P0.7，键盘输入7.
39	P2.0 /TWSI_SCL	I/O	P2.0，TWSI时钟端
40	P2.1 /TWSI_SDA	I/O	P2.1，TWSI数据端
41	P4.0	I/O	P4.0
42	VSS	P	数字地
43	XOUT	O	晶振输出引脚
44	XIN	I	晶振输入引脚
45	P2.4 /SPI_SSI	I/O	P2.4，SPI从机选择端.
46	P2.5 /SPI_MOSI	I/O	P2.5，SPI主机输出从机输入端
47	P2.6 /SPI_MISO	I/O	P2.6，SPI主机输入从机输出端
48	P2.7 /SPI_CLK	I/O	P2.7，SPI时钟端

5. 特殊功能寄存器(SFRs)

5.1. SFR映射地址

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
F8H	**SICON**								FFH
F0H	**B**								F7H
E8H	**P4**								EFH
E0H	**ACC**	**WDTCR**	**IFD**	**IFADRH**	**IFADRL**		**SCMD**	**ISPCR**	E7H
D8H									DFH
D0H	**PSW**	**SIADR**	**SIDAT**	**SISTA**		**KBPATN**	**KBCON**	**KBMASK**	D7H
C8H	**T2CON**	**T2MOD**	**RCAP2L**	**RCAP2H**	**TL2**	**TH2**			CFH
C0H	**XICON**							**CKCON**	C7H
B8H	**IP**	**SADEN**						**CKCON2**	BFH
B0H	**P3**	**P3M0**	**P3M1**	**P4M0**	**P4M1**				B7H
A8H	**IE**	**SADDR**				**AUXIE**	**AUXIP**		AFH
A0H	**P2**						**AUXR2**	**TSTWD**	A7H
98H	**SCON**	**SBUF**							9FH
90H	**P1**	**P1M0**	**P1M1**	**P0M0**	**P0M1**	**P2M0**	**P2M1**		97H
88H	**TCON**	**TMOD**	**TL0**	**TL1**	**TH0**	**TH1**	**AUXR**		8FH
80H	**P0**	**SP**	**DPL**	**DPH**	**SPSTAT**	**SPCTL**	**SPDAT**	**PCON**	87H
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	

5.2. 标准 8051 SFRs

标志	功能描述	地址	位地址与标志								复位值
			Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	
ACC*	累加器	E0H									00H
B*	B寄存器	F0H									00H
PSW*	状态标志寄存器	D0H	D7H CY	D6H AC	D5H F0	D4H RS1	D3H RS0	D2H OV	D1H -	D0H P	00H
SP	堆栈指针	81H									07H
DPH	数据指针高8位	83H									00H
DPL	数据指针低8位	82H									00H
P0*	端口0	80H	87H P0.7 KBI7	86H P0.6 KBI6	85H P0.5 KBI5	84H P0.4 KBI4	83H P0.3 KBI3	82H P0.2 KBI2	81H P0.1 KBI1	80H P0.0 KBI0	FFH
P1*	端口1	90H	97H P1.7	96H P1.6	95H P1.5	94H P1.4	93H P1.3	92H P1.2	91H P1.1	90H P1.0	FFH
P2*	端口2	A0H	A7H P2.7 SPICLK	A6H P2.6 MISO	A5H P2.5 MOSI	A4H P2.4 /SS	A3H P2.3	A2H P2.2	A1H P2.1 SDA	A0H P2.0 SCL	FFH
P3*	端口3	B0H	B7H P3.7 INT3	B6H P3.6 INT2	B5H P3.5 T1	B4H P3.4 T0	B3H P3.3 /INT1	B2H P3.2 /INT0	B1H P3.1 TXD	B0H P3.0 RXD	FFH
IP*	中断优先级	B8H	BFH PX3	BEH PX2	BDH PT2	BCH PS	BBH PT1	BAH PX1	B9H PT0	B8H PX0	00H
IE*	中断使能	A8H	AFH EA	AEH -	ADH ET2	ACH ES	ABH ET1	AAH EX1	A9H ET0	A8H EX0	00H
TMOD	定时器模式	89H	GATE	C/-T	M1	M0	GATE	C/-T	M1	M0	00H
TCON*	定时器控制	88H	8FH TF1	8EH TR1	8DH TF0	8CH TR0	8BH IE1	8AH IT1	89H IE0	88H IT0	00H
T2CON*	定时器2控制	C8H	CFH TF2	CEH EXF2	CDH RCLK	CCH TCLK	CBH EXEN2	CAH TR2	C9H C/-T2	C8H CP/-RL2	00H
TH0	定时器0高字节	8CH									00H
TL0	定时器0低字节	8AH									00H
TH1	定时器1高字节	8DH									00H
TL1	定时器1低字节	8BH									00H
TH2	定时器2高字节	CDH									00H
TL2	定时器2低字节	CCH									00H
RCAP2H	定时器2捕获，高位	CBH									00H
RCAP2L	定时器2捕获，低位	CAH									00H
SCON*	串口控制	98H	9FH SM0/FE	9EH SM1	9DH SM2	9CH REN	9BH TB8	9AH RB8	99H TI	98H RI	00H
SBUF	串口控制缓冲区	99H									xxH
PCON	电源控制	87H	SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL	00H

注意:
*: 可位寻址
-:保留位

5.3. 增强型 SFRs

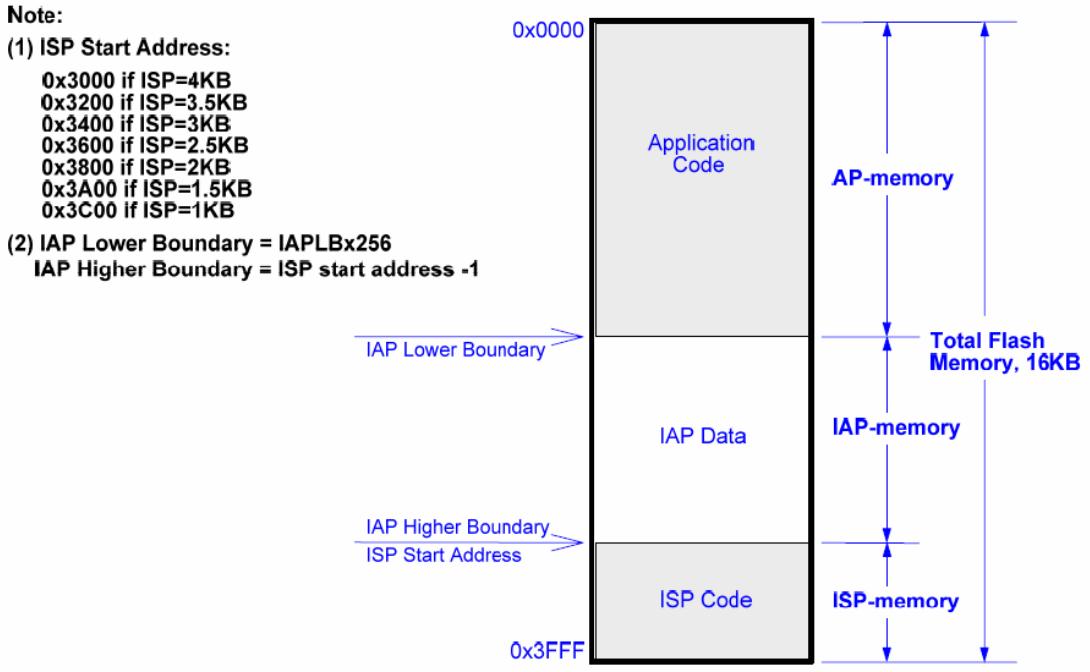
标志	描述	地址	位地址与标志								复位值
			Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	
中断											
XICON*	外部中断控制	C0H	C7H IL3	C6H EX3	C5H IE3	C4H IT3	C3H IL2	C2H EX2	C1H IE2	C0H IT2	00H
AUXIE	辅助中断使能	ADH	EUSB	ETWSI	EKB	-	-	-	-	ESPI	00H
AUXIP	辅助中断优先级	AEH	PUSB	PTWSI	PKB	-	-	-	-	PSPI	00H
I/O口											
P4*	P4	E8H	EFH -	EEH -	EDH -	ECH -	EBH P4.3	EAH P4.2	E9H P4.1	E8H P4.0	FFH
P0M0	P0模式寄存器0	93H	P0M0.7	P0M0.6	P0M0.5	P0M0.4	P0M0.3	P0M0.2	P0M0.1	P0M0.0	00H
P0M1	P0模式寄存器1	94H	P0M1.7	P0M1.6	P0M1.5	P0M1.4	P0M1.3	P0M1.2	P0M1.1	P0M1.0	00H
P1M0	P1模式寄存器0	91H	P1M0.7	P1M0.6	P1M0.5	P1M0.4	P1M0.3	P1M0.2	P1M0.1	P1M0.0	00H
P1M1	P1模式寄存器1	92H	P1M1.7	P1M1.6	P1M1.5	P1M1.4	P1M1.3	P1M1.2	P1M1.1	P1M1.0	00H
P2M0	P2模式寄存器0	95H	P2M0.7	P2M0.6	P2M0.5	P2M0.4	P2M0.3	P2M0.2	P2M0.1	P2M0.0	00H
P2M1	P2模式寄存器1	96H	P2M1.7	P2M1.6	P2M1.5	P2M1.4	P2M1.3	P2M1.2	P2M1.1	P2M1.0	00H
P3M0	P3模式寄存器0	B1H	P3M0.7	P3M0.6	P3M0.5	P3M0.4	P3M0.3	P3M0.2	P3M0.1	P3M0.0	00H
P3M1	P3模式寄存器1	B2H	P3M1.7	P3M1.6	P3M1.5	P3M1.4	P3M1.3	P3M1.2	P3M1.1	P3M1.0	00H
P4M0	P4模式寄存器0	B3H	-	-	-	-	P4M0.3	P4M0.2	P4M0.1	P4M0.0	00H
P4M1	P4模式寄存器1	B4H	-	-	-	-	P4M1.3	P4M1.2	P4M1.1	P4M1.0	00H
键盘中断											
KBCON	键盘控制	D6H	-	-	-	-	-	-	PTNS	KPI	00H
KBPATN	键盘模式	D5H									FFH
KBMASK	键盘中断掩码	D7H									00H
串口											
SADEN	从机地址掩码	B9H									00H
SADDR	从机地址	A9H									00H
2线串口（TWSI）											
SICON*	TWSI控制寄存器	F8H	CR2	ENSI	STA	STO	SI	AA	CR1	CR0	00H
SIADR	TWSI地址寄存器	D1H	(Own Slave Address)							GC	00H
SIDAT	TWSI数据寄存器	D2H									00H
SISTA	TWSI状态寄存器	D3H									F8H
SPI											
SPCTL	SPI控制寄存器	85H	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	04H
SPSTAT	SPI状态寄存器	84H	SPIF	THRE	-	-	-	-	-	SPR2	00H
SPDAT	SPI 数据寄存器	86H									00H
其他											
AUXR	辅助寄存器	8EH	-	-	BRADJ0	-	T2X12	-	-	DPS	00H
AUXR2	辅助寄存器2	A6H	T0X12	T1X12	URM0x6	-	-	-	-	T0CKOE	00H
T2MOD	定时器2模式控制	C9H	T2CPCF	-	DUTY1	DUTY0	FIXV	FIXEN	T2OE	DCEN	00H
CKCON	时钟控制	C7H	XCKS4	XCKS3	XCKS2	XCKS1	XCKS0	CKS2	CKS1	CKS0	28H
CKCON2	时钟控制2	BFH	-	-	OSCDR0	-	EN_USB	EN_PLL	PLL_RDY	CK_SEL	00H
WDTCR	看门狗定时器	E1H	WRF	-	ENW	CLRW	WIDL	PS2	PS1	PS0	00H
ISP											
ISPCR	ISP控制寄存器	E7H	ISPEN	SWBS	SWRST	-	-	-	MS1	MS0	00H

IFADRH	ISP Flash地址高位	E3H		00H
IFADRL	ISP Flash地址低位	E4H		00H
IFD	ISP Flash数据	E2H		FFH
SCMD	ISP命令序列	E6H		xxH

注意:
*: 可位寻址
-:保留位

6. Flash 存储器结构

总共16Kb Flash Memory。



注意:
实际上，笙泉样品的Flash存储器已经配置为2kB ISP，1kB IAP并且已经锁定。2kB的ISP区已经加载了笙泉专用的通过USB DFU 进行ISP的代码。更多关于USB DFU的信息，请参考MG84FL54B的开发板。

7.片内扩展RAM (XRAM)

除了256字节的片内RAM，还在片内扩展了576字节XRAM。它们能够用 “MOVX @Ri”和“MOVX @DPTR”指令访问。

在软件中使用XRAM

For KEIL-C51 编译器中，将变量分配到XRAM中，需要使用“pdata”或者“xdata”声明。编译后，被声明的“pdata”与“xdata”变量通过“ MOVX @Ri” 与 “MOVX @DPTR”指令进行访问。使用者可以通过“*Keil Software — Cx51 Compiler User's Guide*”获得更多信息。

Explicitly Declared Memory Types

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

The following table summarizes the available memory type specifiers.

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

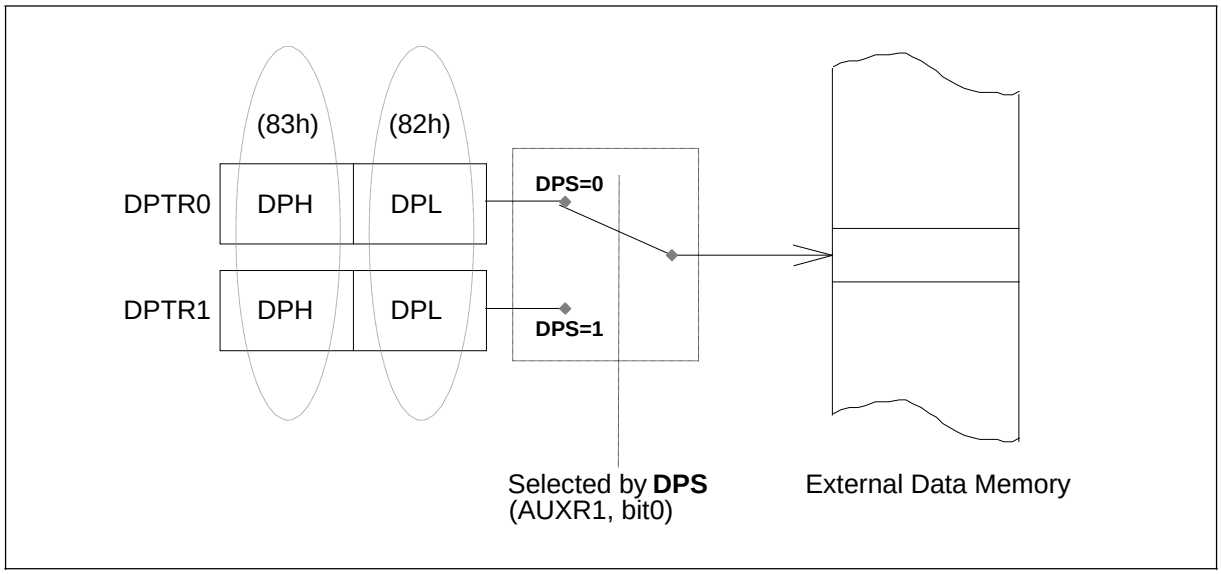
As with the **signed** and **unsigned** attributes, you may include memory type specifiers in the variable declaration.

Example:

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

8.双数据指针(DPTR)

双数据指针结构（见下图）是一种用于寻址外部数据储存空间的方法。有两个16位的DPTR寻址外部存储空间，DPS (AUXR.0)位能够被软件操作来选择这两个DPTR中的一个



DPTR指令

以下六条指令涉及到当前的DPS选择DPTR：

```
INC   DPTR ;数据指针加1
MOV   DPTR,#data16 ; 将16位常数送入DPTR
MOV   A,@A+DPTR ;移动与DPTR有关的代码到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 ;与DPTR有关的间接跳转
```

AUXR (地址8EH, 辅助寄存器)

7	6	5	4	3	2	1	0
-	-	BRADJ0	-	T2X12	-	-	DPS

DPS: DPTR选择位，用于选择DPTR0还是DPTR1.

选择DPTR0还是DPTR1必须由软件对DPS位进行设置

DPS		DPTR selected
0		DPTR0
1		DPTR1

9. I/O端口

9.1. 端口结构

此芯片具有 P0 —P4共5个I/O端口。每个端口的各个引脚都可被独立地配置成以下四种模式之一：*准双向（标准8051I/O端口）*、*推挽输出*、*漏极开路输出*及*输入口（高阻态）*。每个端口引脚的输入端都有一个施密特触发器以消除噪声。 每个端口都有两个寄存器PxM0与PxM1，它们用于配置I/O脚的工作模式。X=0~4。

Table :端口设置

PxM0.y	PxM1.y	端口模式
0	0	准双向
0	1	推挽输出
1	0	输入口（高阻态）
1	1	漏极开路输出

当x=0~4 (端口序号)，且y=0~7 (端口引脚)。寄存器PxM0和PxM1如下。

P0M0 (地址93H, P0模式寄存器0)

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

P0M1 (地址94H, P0模式寄存器1)

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

P1M0 (地址91H, P1模式寄存器0)

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

P1M1 (地址92H, P1模式寄存器1)

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

P2M0 (地址95H, P2模式寄存器0)

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

P2M1 (地址96H, P2模式寄存器1)

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

P3M0 (地址B1H, P3模式寄存器0)

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

P3M1 (地址B2H, P3模式寄存器1)

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

P4M0 (地址B3H, P 4模式寄存器0)

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

P4M1 (地址B4H, P4模式寄存器1)

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

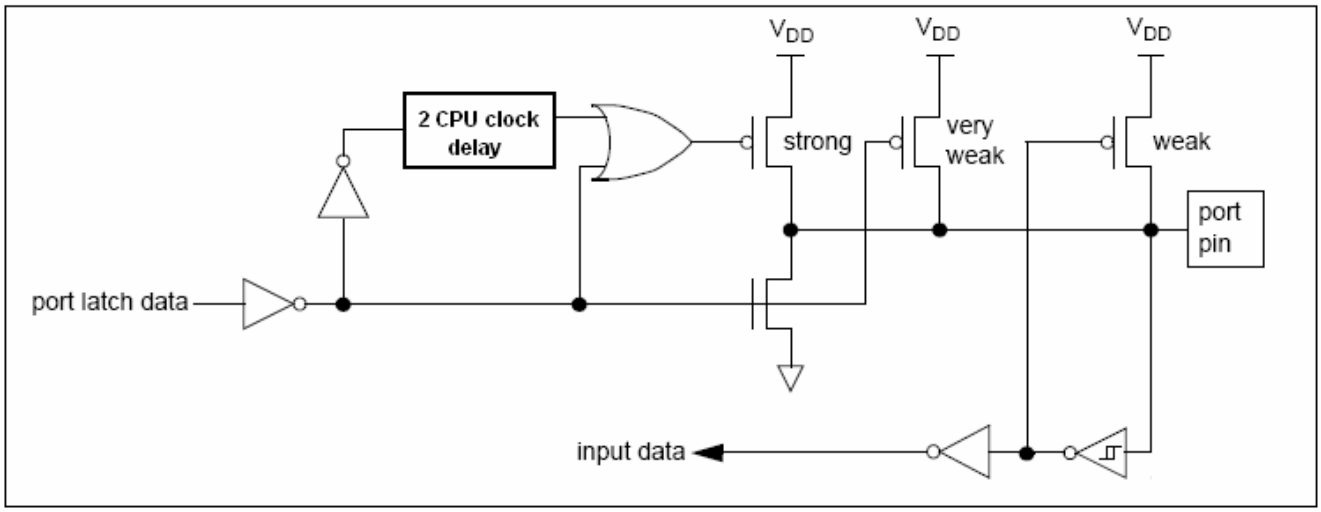
9.1.1. 准双向

端口引脚在准双向模式下与标准8051类似。准双向端口作为输入和输出时不需要对端口重新配置。之所以能够这样，是因为当端口输出逻辑“1”时驱动能力很弱，允许外部器件拉低引脚。当输出为低时有较强的驱动能力，能够吸收大电流。在准双向模式下有3个上拉晶体管用于不同的应用。

其中的一种上拉，称为“微上拉”，只要端口寄存器包含逻辑“1”它就会导通。在引脚悬空时它提供一个非常小的电流将引脚拉高。

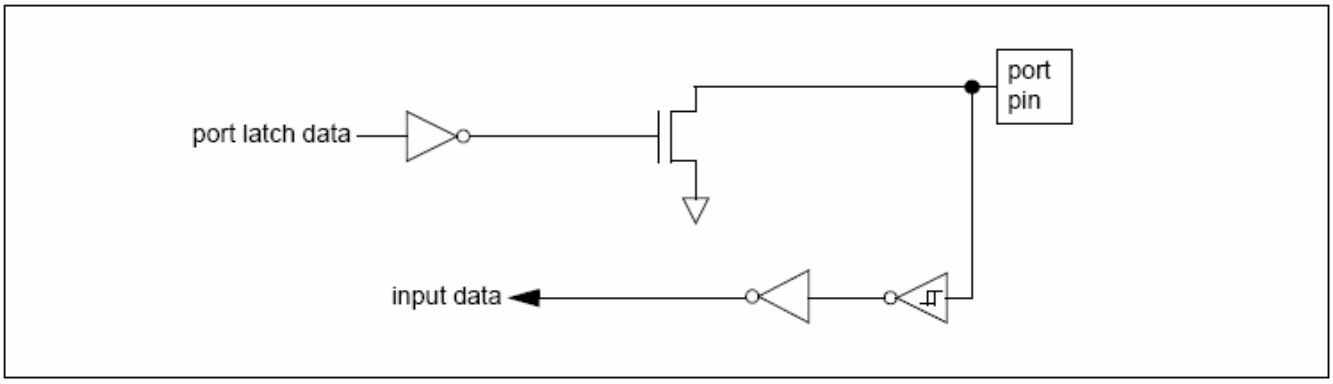
第二种上拉称为“弱上拉”，当端口寄存器对应引脚位为逻辑“1”，且此引脚自身也是逻辑“1”时将打开。当输出为“1”时这种上拉结构提供主要的输出电流。如果引脚被外部器件拉低，弱上拉将关闭，只剩下微上拉仍然打开。为了在这种状态下将引脚拉低，外部器件必须吸收超过弱上拉所能提供的电流，并且拉低引脚电压至输入门限电压以下。

第三种上拉称为“强上拉”。这种上拉用于加速当端口寄存器从逻辑“0”到逻辑“1”时准双向端口的上升沿跳变。经过两个CPU时钟延时后，强上拉打开，快速将端口引脚拉高。



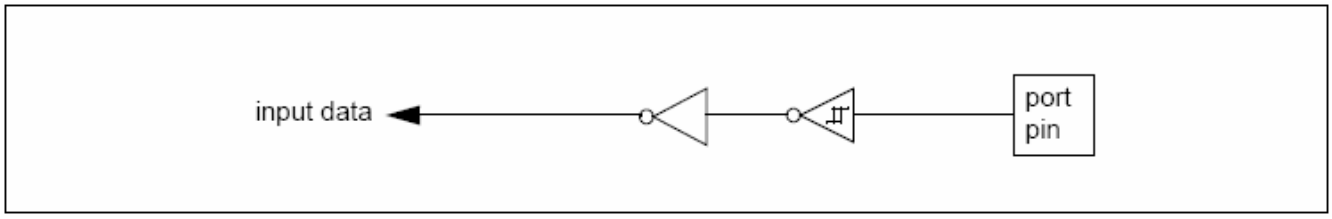
9.1.2. 漏极开路输出

在开漏状态下，当端口寄存器包含逻辑“0”时，所有的上拉都会关闭，只有端口引脚的晶闸管下拉打开。应用此模式时引脚必须进行外部上拉，通常是由一个电阻接至VDD。此模式下的下拉结构与准双向时相同。此外，输入路径和准双向模式相同。



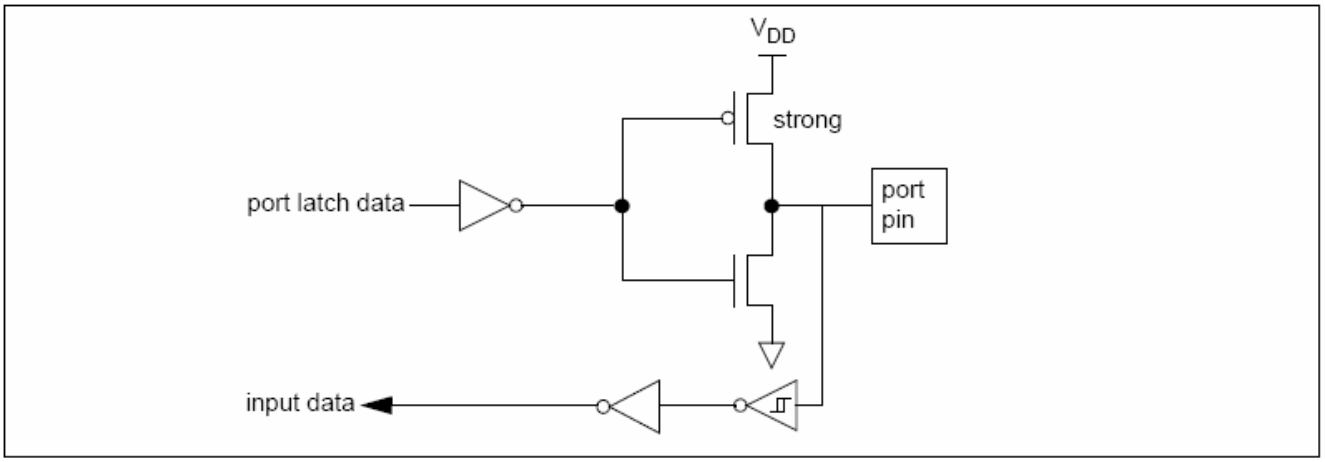
9.1.3. 输入口(高阻态)

输入状态下配置了一个施密特触发器但是没有上拉电阻。



9.1.4. 推挽输出

推挽输出有着和开漏输出和准双向输出一样的下拉结构，但是当端口寄存器为“1”时提供一个持续的强上拉。当需要较大的电流输出时可以配置成推挽模式输出。另外，此状态下的输入路径与准双向模式的相同。



9.2. 最大输出能力

当端口作为输出功能时（作为源或吸收电流），为了避免造成永久性的损坏，使用者必须注意不管是在3.3V还是5V的电压下其漏电流或拉电流都*不能超过40mA*。这就意味着灌电流和拉电流在同一时刻不超过40mA芯片才是安全的。

10. 三个16位定时器

10.1. 定时器0和定时器1

上电或者复位后，定时器0及定时器1默认的功能与操作与传统的8051MCU完全兼容。唯一不同的是除了Fosc/12外用户还可以选择另一个定时频率Fosc。AUXR2寄存器的第7位与第6位提供此项功能。

AUXR2 (地址A6H，辅助寄存器2)

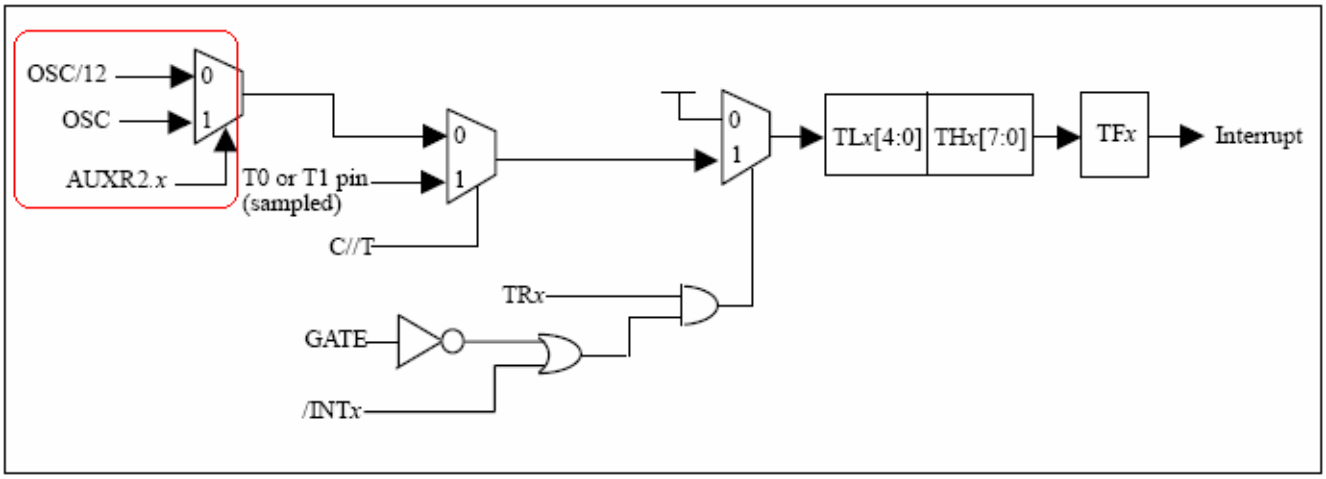
7	6	5	4	3	2	1	0
T0X12	T1X12	URM0x6	-	-	-	-	T0CKOE

T0X12: C/T=0时定时器0时钟源选择。
置位时选择Fosc作为定时器时钟源，清零时选择Fosc/12。

T1X12: C/T=0时定时器1时钟源选择。
置位时选择Fosc作为定时器时钟源，清零时选择Fosc/12。
T0CKOE: 置位允许Timer 0的时钟在P3.4输出。

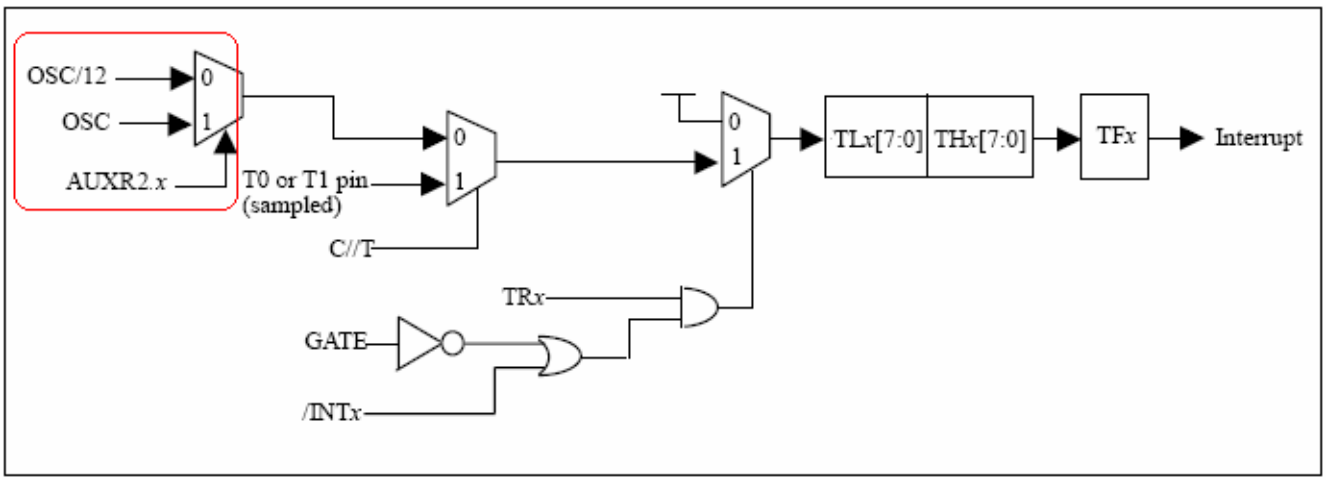
下图描述了Timer 0与Timer1选择不同的时钟源的原理

10.1.1. 模式0：13位计数器



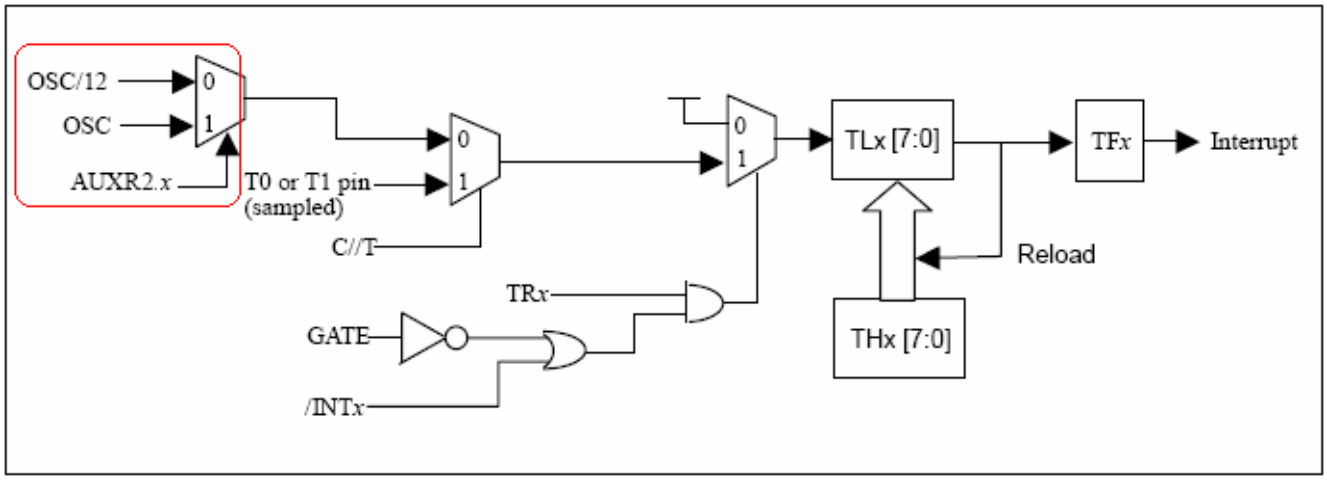
OSC即Fosc，系统时钟

10.1.2. 模式1：16位计数器



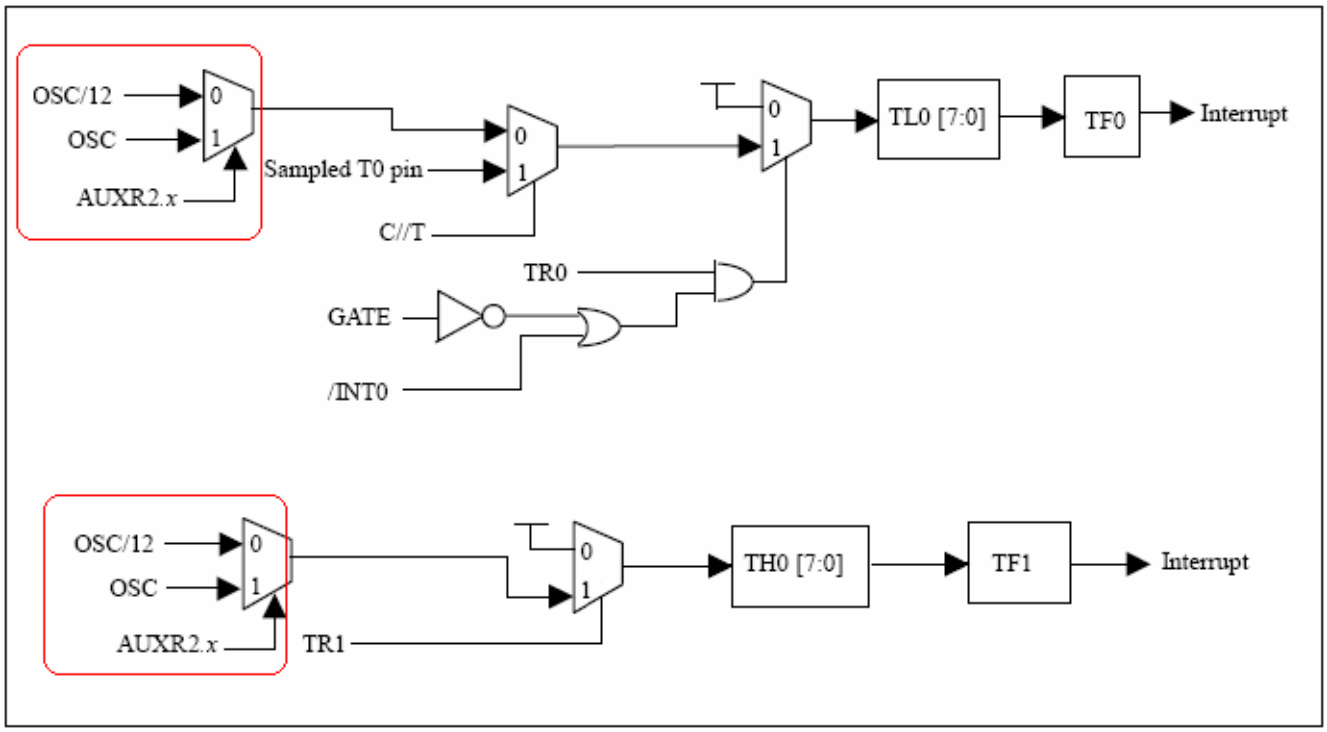
OSC即Fosc，系统时钟

10.1.3.模式2：8位自动加载方式



OSC即Fosc，系统时钟.

10.1.4. 模式3：定时器0作双8位计数器



OSC 即 Fosc, 系统时钟.

10.1.5. 定时器0可编程时钟输出模式

用户可通过设置定时器0为8位自动重载模式并且置位 **T0CKOE** 而在P3.4输出占空比为50%的方波。当然，必须置位TR0 (TCON.4)才能启动此功能。在12MHz的系统频率下，输出的方波频率范围是1903Hz——6MHz。

频率计算公式如下(定时器0溢位率的一半)

$$\text{输出时钟频率} = \frac{\text{系统频率}}{n \times (256 - TH0)}$$

如果

T0X12=0 n=24

T0X12=1 n=2

(Fosc 为系统时钟.)

10.2 定时器 2

三个特殊功能寄存器，AUXR, T2MOD 和 T2CON，与定时器 2 的操作有关系，列出如下。

AUXR (地址=8EH, 辅助寄存器)

7	6	5	4	3	2	1	0
-	-	BRADJ0	-	T2X12	-	-	DPS

T2X12： 当 C/T2 (T2CON.1)=0 在捕获模式和自动加载模式的时候，选择定时器 2 的时钟源。
置位选择 Fosc 作为时钟源；清零选择 Fosc/12 为时钟源。

T2MOD (地址=C9H,定时器2模式控制寄存器)

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

T2OE： 定时器 2 的时钟输出使能：0，禁止；1，使能。
DCEN： 定时器 2 的向下计数使能：0，禁止；1，使能。

T2CON (地址=C8H, 定时器2控制寄存器)

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

TF2：定时器 2 的溢出标志。定时器 2 溢出时置位，必须由软件清零。当 RCLK=1 或 TCLK=1 时 TF2 不会置位。

EXF2： 定时器 2 的外部标志。当 T2XEN=1 且 T2EX 脚上有一个负跳变而引起捕获或者加载时，该标志置位。

RCLK： 接收时钟标志。置位时，在模式 1 和模式 3 下定时器 2 的溢出脉冲作为串口的接收时钟。清零时，定时器 1 的溢出作为接收时钟。

TCLK： 发送时钟标志。置位时，在模式 1 和模式 3 下定时器 2 的溢出脉冲作为串口的发送时钟。清零时，定时器 1 的溢出作为发送时钟。

EXEN2： 定时器 2 外部使能标志。置位时，如果定时器 2 没有被用于串口时钟，则当 T2EX 脚有一个负跳变时，允许捕获或加载发生。清零时，定时器 2 忽略 T2EX 脚上的变化。

TR2： 定时器 2 的启动/停止控制。逻辑 1 启动定时器。

C/T2： 定时器或计数器选择器。清零时，选择为内部定时器；置位时，选择为外部事件计数器（下降沿触发）。

CP/-RL2：捕获/加载标志位。置位时，如果 EXEN2=1，则当 T2EX 脚有一个负跳变时产生捕获。清零时，如果 EXEN2 = 1，则当定时器2溢出或者 T2EX 脚有一个负跳变时产生自动加载。当 RCLK=1 或 TCLK=1，这个位被忽略并且在定时器2溢出时强制自动加载。

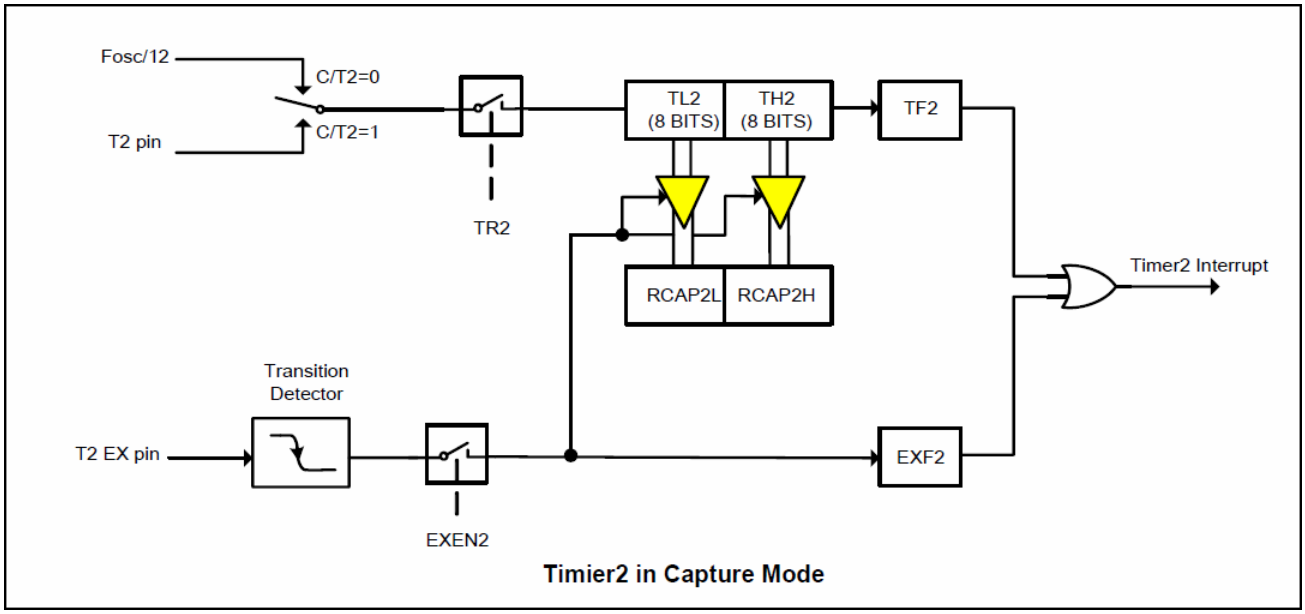
复位以后，DCEN=0（T2MOD.0），使得定时器2的功能与标准8502的功能一模一样（总是向上计数）。当 DCEN=1时，定时器2可以根据 T2EX 脚（P1.1）的逻辑电平而向上或者向下计数。下面的表格展示了定时器2的工作模式。

Table： 定时器2运行模式

RCLK + TCLK	CP/-RL2	TR2	DCEN	T2OE	模式
X	X	0	X	0	关闭
1	X	1	0	0	波特率发生器
0	1	1	0	0	16位捕获
0	0	1	0	0	16位自动加载 (仅向上计数)
0	0	1	1	0	16位自动加载(向上计数或向下计数)
0	0	1	0	1	时钟输出

10.2.1 捕获模式（CP）

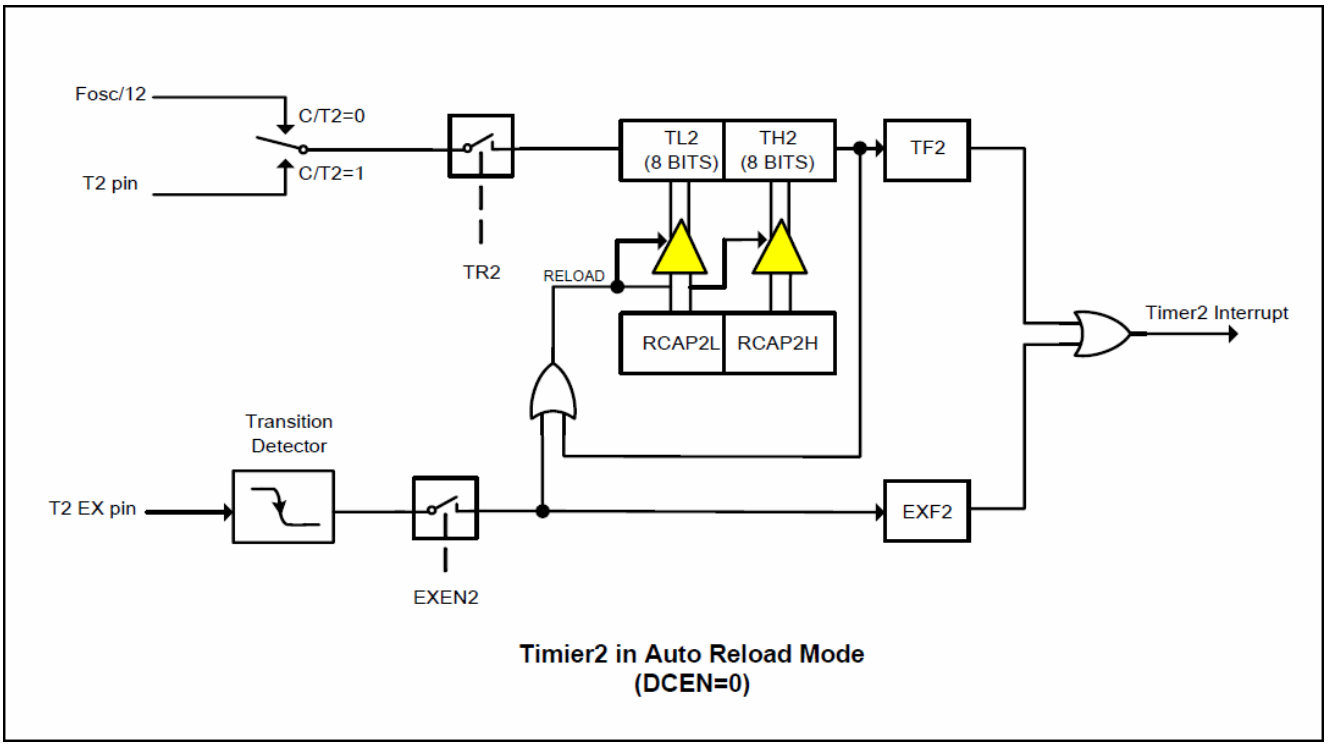
在捕获模式，通过 Fosc/12 或者外部脚(T2)从1到0的跳变使定时器2计数增加。如果 EXEN2=1，TR2 控制定时器2， 当外部输入信号 T2EX 有下降沿跳变时，引起定时器2的寄存器 TH2 和 TL2 分别对应的捕获到 RCAP2H 和RCAP2L。定时器2溢出时 TF2 置位。如果 EXEN2=1，T2EX 脚上从1到0跳变将使 EXF2 置位。TF2 和 EXF2 是为中断服务请求做准备的。



定时器2捕获模式

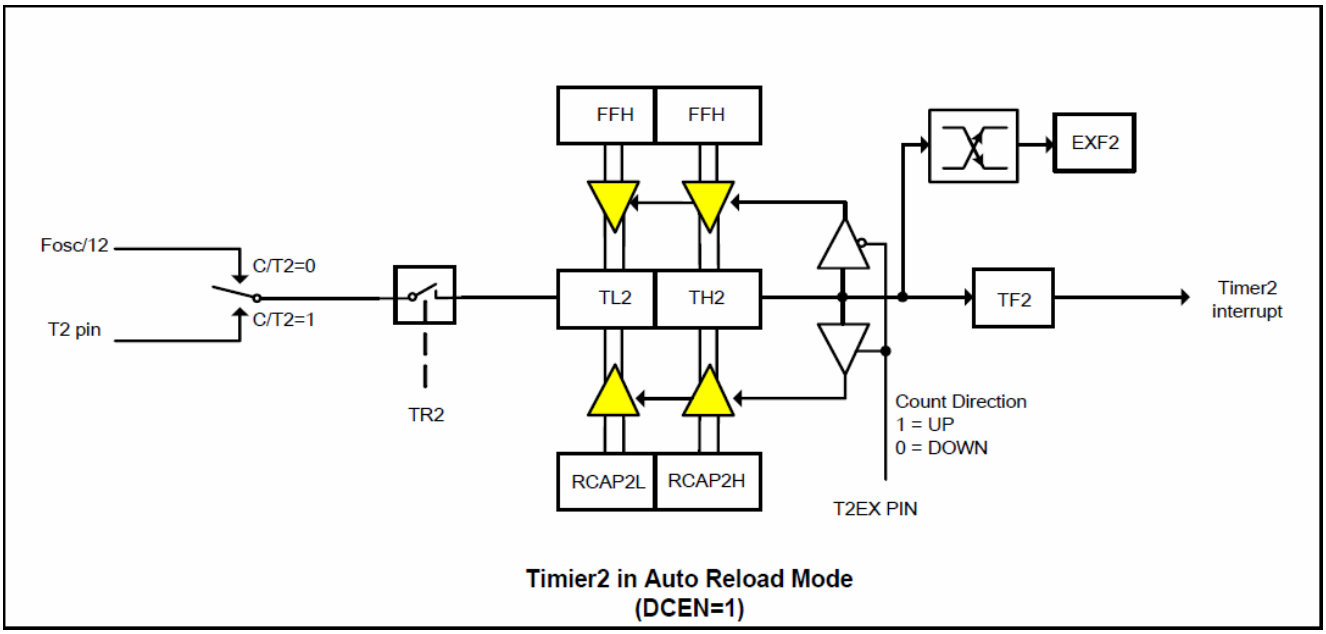
10.2.2 自动加载模式（AR）

在自动加载模式，定时器2既可以配置成向上计数也可以配置成向下计数，这取决于 T2MOD 寄存器中的 DCEN 位。当使用复位后的状态(DCEN = 0 , CP/RL2 = 0)时，定时器2处于自动加载模式并且只能向上计数。当定时器2发生溢出或者 T2EX 脚上有一个从1到0的跳变时，将装载 RCAP2H 和 RCAP2L 的内容到定时器2，同时分别置位 TF2和 EXF2。



定时器2自动加载模式（DCEN = 0）

当 DCEN=1 且在自动加载模式时，定时器2可以配置成向上或向下计数。计数的方向取决于T2EX脚。若 T2EX=1, 则向上计数，否则向下计数。定时器2溢出会置位TF2并且翻转EXF2。当 DCEN=1 且在自动加载模式下时，EXF2 不会产生中断请求。如果计数的方向是向下的，当定时器2中的计数值与保存在 RCAP2H 和 RCA2PL 中的值相等时，定时器2会装载0XFFFF。但是当向上计数的时候，定时器2的溢出会把 RCAP2H 和 RCA2PL 中的值装载到定时器2。



定时器 2 自动加载模式(DCEN = 1)

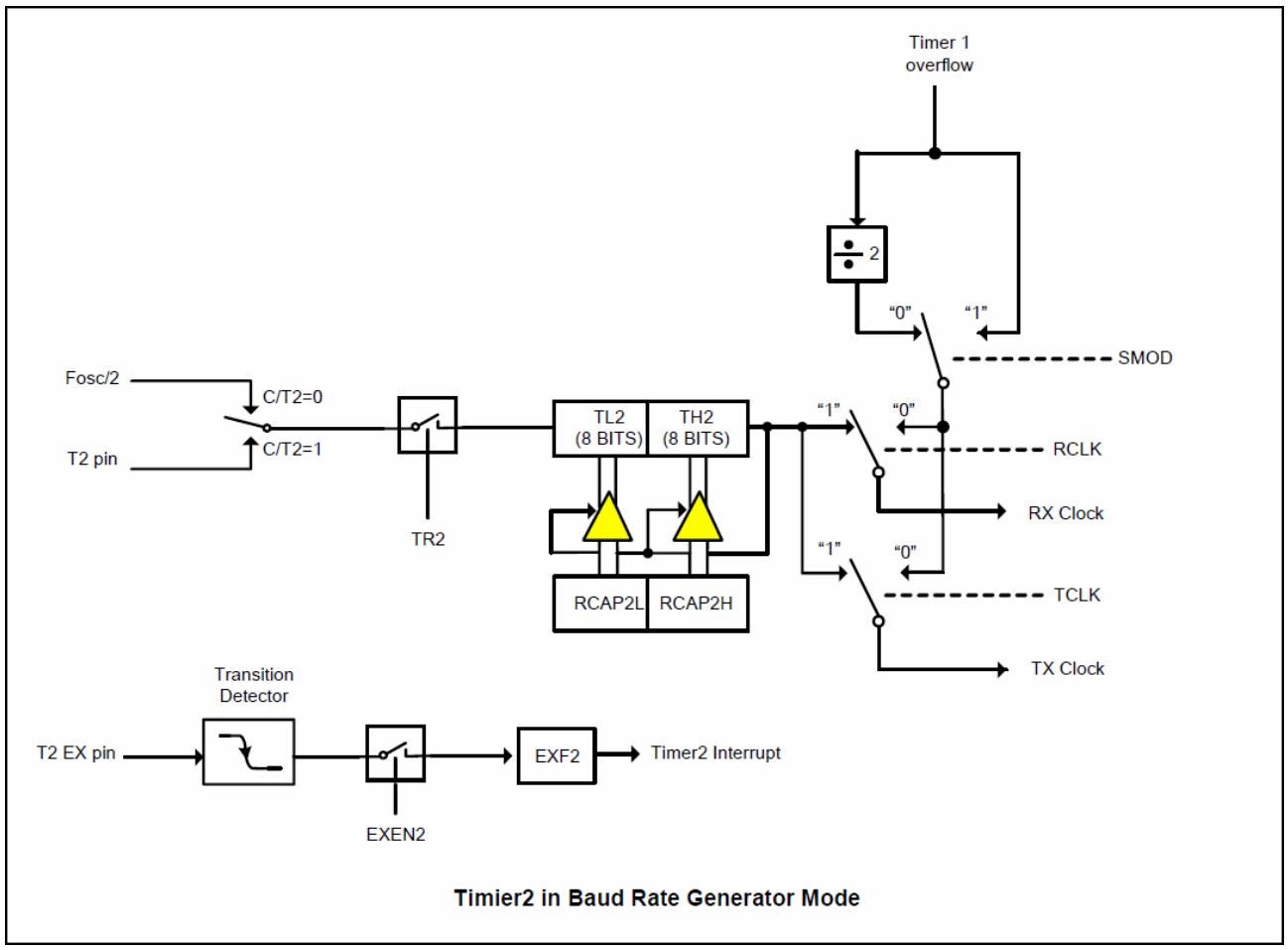
10.2.3 波特率发生器模式(BRG)

定时器 2 可以被配置产生不同的波特率。T2CON 寄存器的 RCLK 和或 RCLK 位允许串行口发送和接收波特率既可来自定时器 1 也可来自定时器 2。当 TCLK=0 时，定时器 1 或者 S2BRT 作为串行口发送波特率发生器。当 TCLK=1，定时器 2 作为串行口发送波特率发生器。RCLK 对串行口波特率有相同的功能。有了这两位，串行口可以有不同的接收和发送波特率，一个通过定时器 1 或 S2BRT 来产生，另一个通过定时器 2 来产生。

在波特率发生器模式，除了 T2EX 脚不能控制重载外，定时器的操作非常像自动重载时的向上计数。定时器 2 的溢出把 RCAP2H 和 RCAP2L 中的值装载到定时器 2 中，但是 TF2 不会置位。如果 EXEN2=1 ,则 P2EX 引脚上的从 1 到 0 的跳变将会置位 EXF2 以请求中断服务。

串行口在模式 1 和模式 3 下的波特率取决于定时器 2 的溢出速率，公式如下：

$$模式1和模式3波特率 = 晶振频率/(2\times(65536-[RCAP2H,RCAP2L]))$$



定时器 2 波特率发生器模式

10.2.4 定时器 2 可编程时钟输出模式

定时器 2 有一个时钟输出模式(当 CP/-RL2 = 0 且 T2OE = 1)。在这个模式下，定时器 2 操作起来就像一个占空比为 50％的可编程时钟发生器。产生的时钟从 P1.0 引脚输出。输入时钟为 Fosc/2, 16 位定时器[TH2, TL2]自动增加，其中 Fosc 是系统时钟。定时器重复计数直到溢出，然后重新加载。一旦溢出，[RCAP2H, RCAP2L]中的内容加载到[TH2, TL2]中，以便连续计数。*注意定时器 2 的溢出标志位 TF2， 在这个模式下，通常不会置位。下面这个公式给出了时钟的输出频率。*

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

（对于一个 12MHz 的系统时钟，定时器 2 的可编程时钟输出范围可以达到 45.7Hz 到 3 MHz.）

如何编程配置定时器2为可编程时钟输出模式：

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

在时钟输出模式，定时器2翻转不会产生中断，这和用作波特率发生器时相似。因此，可以使定时器2同时作为一个波特率发生器和时钟发生器。但是注意，在这种配置下，波特率的速率和时钟频率不是独立的，因为这两个功能都使用了相同的加载值，寄存器[RCAP2H, RCAP2L]中的值。

11 .增强型UART

11.1 帧错误检测

在 SMOD0 位(PCON的位6)置位时，当检测到一个无效的停止位时，硬件会把FE位(SCON.7)置位。收到有效帧时FE不会自动清除，而是需要用软件清除。

SCON (地址=98H, 串行口控制寄存器)

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

SM0/FE：

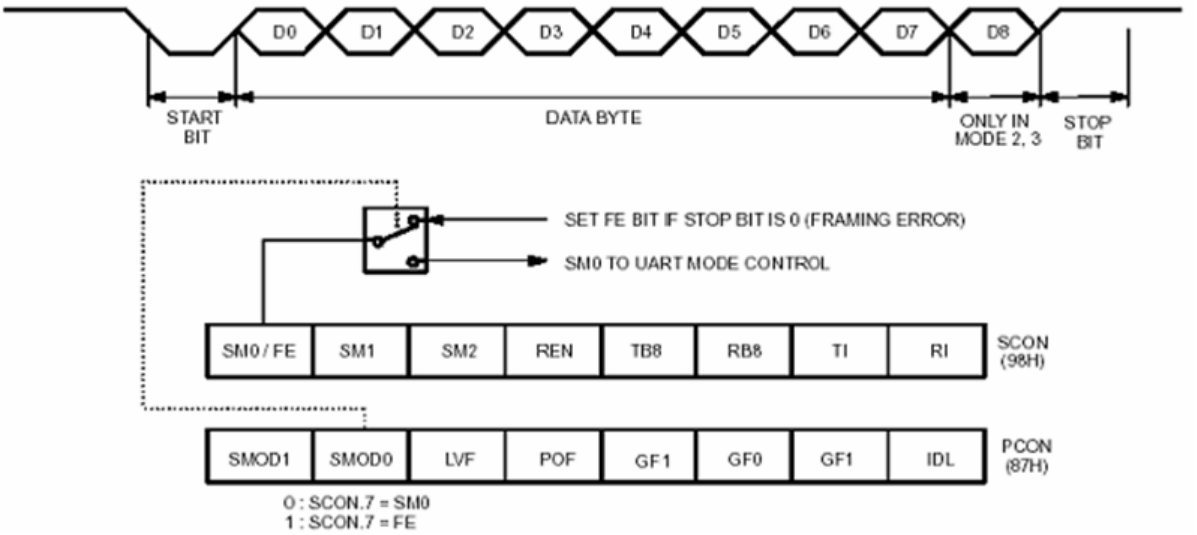
SM0： 串行口模式位 0(SMOD0 必须为 0 来访问 SM0 位)。

FE： 帧错误位(SMOD0 必须为 1 来访问 FE 位)。

PCON (地址=87H, 电源控制寄存器)

7	6	5	4	3	2	1	0
SMOD	SMOD0	LVF1	POF	GF1	GF0	PD	IDL

SMOD0：清零时 SCON.7 的功能为“SM0”，置位时 SCON.7 的功能为“FE”。



11.2 自动地址识别

自动地址识别通过硬件比较可以让UART识别串行码流中的特定地址。该功能免去了使用软件检测串口数据中的串行口地址，节约了大量的软件开销。该功能通过设定SCON的SM2位开启。在9位数据UART模式下，即模式2和模式3，收到特定地址或广播地址时自动置位接收中断(RI)标志。9位模式的第9位信息需要用1来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下图。

8位模式也叫模式1。在该模式下，如果SM2置位并且在8位地址与给定地址或广播地址核对一致后收到有效停止位，则RI置位。

模式0是移位寄存器模式，SM2位被忽略。

使用自动地址识别功能，可以让一个主机通过给定一个或多个地址，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用来定义从机地址和地址掩码，即 SADDR 寄存器和 SADEN 寄存器。

SADEN 用来定义SADDR中的哪些位是需要用到的，哪些位是“无关紧要”的。SADEN 掩码和 SADDR 寄存器通过逻辑与来定义供主机寻址从机的“给定” 地址。该地址让多个从机进行排他性的识别。

下面的实例帮助理解这个方案的通用性：

从机 0

SADDR = 1100 0000

SADEN = 1111 1101

地址 = 1100 00X0

从机 1

SADDR = 1100 0000

SADEN = 1111 1110

地址 = 1100 000X

上面的例子中 SADDR 是相同的值，而使用 SADEN 数据被用来区分两个从机。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。因此，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，因为这个地址的 第0位=0(给从机0)，第1位=0(给从机1)。

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

SADDR = 1100 0000

SADEN = 1111 1001

地址 = 1100 0XX0

从机 1

SADDR = 1110 0000

SADEN = 1111 1010

地址 = 1110 0X0X

从机 2

SADDR = 1110 0000

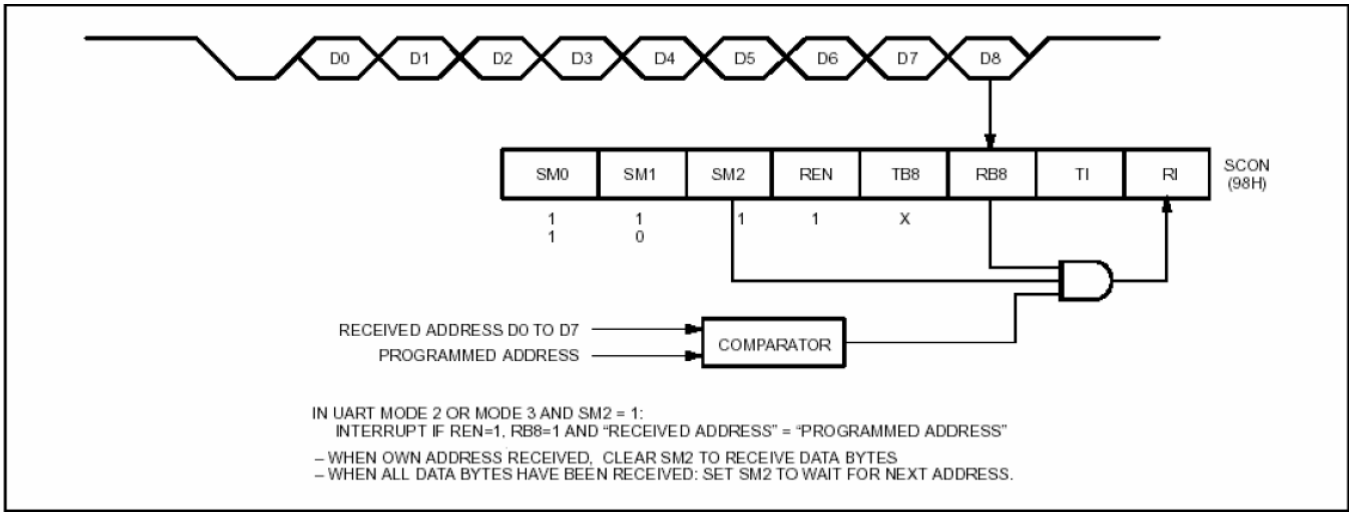
SADEN = 1111 1100

地址 = 1110 00XX

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

每个从机的广播地址为 SADDR 和 SADEN 的逻辑或，0按“不需关心”处理。大部分情况下，不关心位用1处理，从而使用十六进制的FF作为广播地址。

复位后，SADDR（SFR地址0A9H）和 SADEN（SFR地址0B9H）值均为0。这样造成给定地址无需关心，同时广播地址也无需关心，从而有效的禁用了自动地址识别模式，并且使该处理器不使用这个特性，而运行于标准80C51的UART模式下。



11.3 波特率设置

除了波特率的设置，串口的四种模式的操作跟标准的8051的操作是一样的。三个寄存器PCON,AUXR,AUXR2与波特率的设置有关。

PCON(地址 = 87H，电源控制寄存器)

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

SMOD：双倍速波特率控制

AUXR (地址 =8EH, 辅助寄存器)

7	6	5	4	3	2	1	0
-	-	BRADJ	-	T2X12	-	-	DPS

BRADJ： 波特率调整指示。该位用来提高串口的波特率，其选择如下表所示。

BRADJ0	提高波特率设置
0	默认波特率
1	双倍波特率

AUXR2 (地址 = A6H, 辅助寄存器2)

7	6	5	4	3	2	1	0
T0X12	T1X12	URM0X6	-	-	-	-	T0CKOE

T1X12：定时器1的时钟源选择位。置位选择 Fosc 作为时钟源，清零选择 Fosc/12作为时钟源。

URM0X6：串行口模式0的波特率选择位。置位选择 Fosc/2.

T1X12 和 URM0X6 这两位为波特率的设置提供了一种新的选择，列出如下。

模式0的波特率

URM0X6 = 0	URM0X6 = 1
$B.R. = \frac{Fosc}{12}$ (跟标准的8051一样)	$B.R. = \frac{Fosc}{2}$

模式2的波特率

BRADJ = 0	BRADJ = 1
$B.R. = \frac{2^{SMOD}}{64} \times Fosc$ (跟标准的8051一样)	$B.R. = \frac{2^{SMOD}}{32} \times Fosc$

模式1与模式3的波特率

(1)使用定时器1作为波特率发生器

	BRDJ = 0	BRDJ = 1
T1X12 = 0	$B.R. = \frac{2^{SMOD}}{32} \times \frac{Fosc}{12 \times (256 - TH1)}$ （跟标准的8051是一样的）	$B.R. = \frac{2^{SMOD}}{16} \times \frac{Fosc}{12 \times (256 - TH1)}$
T1X12 = 1	$B.R. = \frac{2^{SMOD}}{32} \times \frac{Fosc}{(256 - TH1)}$	$B.R. = \frac{2^{SMOD}}{16} \times \frac{Fosc}{(256 - TH1)}$ **注1**

注1：对 Fosc = 12 MHz, 如果{BRDJ1,BRDJ0}=01,T1X12 = 1,SMOD = 1,我们就可以得出：

波特率 = (2/16) x 12MHz / (256-243) = 115385 位/每秒。

其中，在标准的115200 位/每秒偏移 +0.16%，UART(串口)通信是可以接受的。

(2) 使用定时器2作为波特率发生器

当定时器2作为波特率发生器(T2CON 中的 TCLK 位或 RCLK 位为1)，其波特率如下所示。

BRDJ = 0	BRDJ = 1
$B.R. = \frac{Fosc}{32} \times \frac{1}{65536 - [RCAP2H, RCAP2L]}$ （与标准8051是一样的）	$B.R. = \frac{Fosc}{8} \times \frac{1}{65536 - [RCAP2H, RCAP2L]}$ **注2**

注2：对 Fosc = 12 MHz, 如果 BRDJL = 1 并且 [RCAP2H,L] = 65523, 我们可以得到

波特率=(12MHz/8) x 1/(65536-65523) = 115385位/每秒

其中，在标准的115200 位/每秒偏移 +0.16%，UART(串口)通信是可以接受的。

12 中断

MG84FL54B总共有12个中断源。每一个中断源都能够通过对中断使能寄存器（IE, AUXIE 和 XICON）中相应的位置位或清零来独立的使能或者禁止。并且每一个中断源都能通过对中断优先级寄存器（ IP 和 AUXIP）中相应的位编程，即置位或清零来设置两级中断优先级。在控制和操作这些中断源时，两级中断优先级能够使之灵活操作。以下就是所有中断源的列表：

Table：中断源

中断源序号	中断名称	中断使能位	中断标志位	优先级位	中断优先级	向量地址
#1	外部中断, INT0	EX0	IE0	PX	最高	0003H
#2	定时器0	ET0	TF0	PT	.	000BH
#3	外部中断, INT1	EX1	IE1	PX	.	0013H
#4	定时器1	ET1	TF1	PT	.	001BH
#5	串口中断	ES	RI+TI	P	.	0023H
#6	定时器2	ET2	TF2+ EXF2	PT	.	002BH
#7	外部中断, INT2*	EX2	IE2	PX	.	0033H
#8	外部中断, INT3*	EX3	IE3	PX	.	003BH
#9	SPI	ESPI	SPIF	PSPI	.	0043H
#10	-	-	-	-	.	004BH
#11	-	-	-	-	.	0053H
#12	-	-	-	-	.	005BH
#13	-	-	-	-	.	0063H
#14	键盘中断	EKBI	KBIF	PKBI	.	006BH
#15	两线串行端口中断	ETWSI	SI	PTWSI	.	0073H
#16	USB中断	EUSB	（见注1）	PUSB	最低	007BH

提示1：USB中断标志包括：

（1） URST, URSM 和USUS：都包含在USB寄存器 UPCON 中。

（2） UTXD0, URXD0, UTXD1, UTXD2, ASOFIF 和 SOFIF：都包含在USB寄存器 UIFLG 中。

（3） UTXD3, URXD3, URXD4 和 UTXD5： 都包含在USB寄存器 UIFLG1 中。

IE (地址=A8H, 中断使能寄存器)

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

EA： 总中断禁止位。EA = 0，禁止所有中断。EA = 1，每个中断可以通过置位或清零其使能位来单独配置其中断使能。

ET2： 定时器2 中断使能位。

ES：串行口中断使能位。

ET1： 定时器1中断使能位。

EX1： 外部中断1使能位。

ET0： 定时器0中断使能位。

EX0： 外部中断0使能位。

AUXIE (地址=ADH, 中断使能寄存器 2)

7	6	5	4	3	2	1	0
EUSB	ETWSI	EKBI	-	-	-	-	ESPI

EUSB：USB中断使能位。

ETWSI：二线串行端口中断使能位。

EKBI：键盘中断使能位。

ESPI：SPI中断使能位。

XICON (地址=C0H, 外部中断控制寄存器)

7	6	5	4	3	2	1	0
IL3	EX3	IE3	IT3	IL2	EX2	IE2	IT2

IL3：外部中断3电平控制位。1：上升沿/高电平触发；0：下降沿/低电平触发。
EX3：外部中断3使能位。
IE3：外部中断3中断标志位。
IT3：外部中断3类型控制位。1：边沿触发；0：电平触发。
IL2：外部中断2电平控制位。1：上升沿/高电平触发；0：下降沿/低电平触发。
EX2：外部中断2使能位。
IE2：外部中断2中断标志位。
IT2：外部中断2类型控制位。1：边沿触发；0：电平触发。

IP (地址=B8H, 中断优先级寄存器)

7	6	5	4	3	2	1	0
PX3	PX2	PT2	PS	PT1	PX1	PT0	PX0

PX3：外部中断3优先级位。
PX2：外部中断2优先级位。
PT2：定时器2优先级位。
PS：串行口中断优先级位。
PT1：定时器1优先级位。
PX1：外部中断1优先级位。
PT0：定时器0优先级位。
PX0：外部中断0优先级位。

AUXIP (地址=AEH, 辅助中断优先级寄存器)

7	6	5	4	3	2	1	0
PUSB	PTWSI	PKBI	-	-	-	-	PSPI

PUSB：USB中断优先级位。

PTWSI：二线串行接口中断优先级位

PKBI：键盘中断优先级位

PSPI：SPI中断优先级位

12.1. 两级中断优先级

寄存器IP和AUXIP的值决定了每一个中断的优先级。下表表示了每一个中断优先级和寄存器值之间的对应关系。

Table：由IP寄存器决定的中断优先等级：

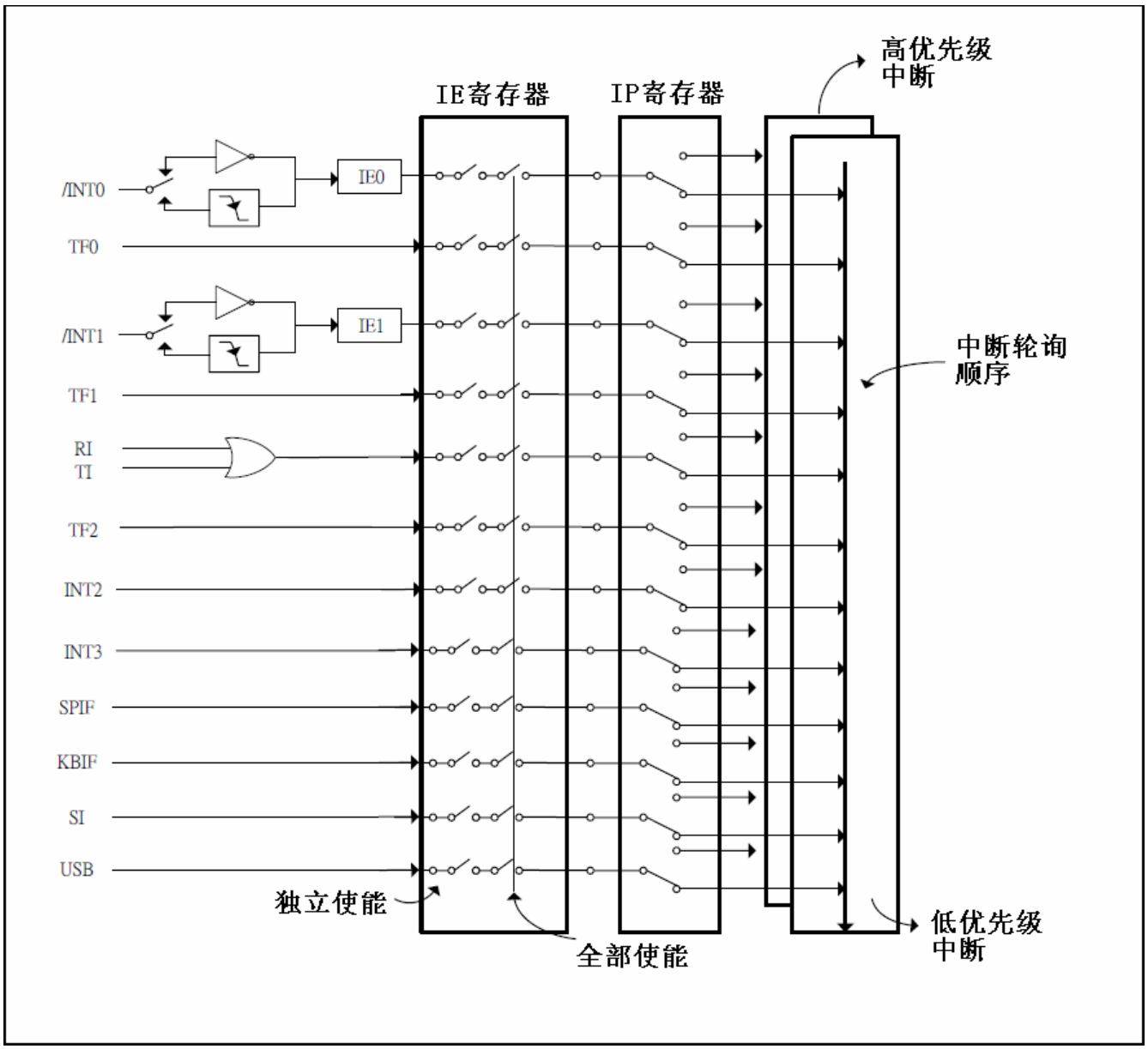
IP.x	中断优先级水平
1	Level 1 (高优先级)
0	Level 0 (低优先级)

表：由AUXIP寄存器决定的中断优先等级：

AUXIP.x	Interrupt Priority Level
1	Level 1 (高优先级)
0	Level 0 (低优先级)

举例：如果IP.3=1,则定时器1的优先级为1，其高于优先级为0的情况（IP.3=0）

12.2. 中断系统



12.3. ISP/IAP中的中断注意事项

在ISP/IAP时，CPU将停止一会儿来做内部ISP/IAP处理。在此期间，已被使能的中断将被挂起不执行。一旦ISP/IAP处理结束，CPU 将继续运行，如果此时中断标志位还存在，将立刻依次执行已产生的中断。但是，用户必须了解以下情况：

(1) MCU 正在执行ISP/IAP过程时，任何中断服务程序都不能执行。

*(2) **低电平触发**外部中断，/INT0, /INT1, INT2 和 INT3必须一直保持低电平，直到 ISP/IAP过程结束，否则这些中断将被忽略.*

13. 附加的外部中断 (INT2 和 INT3)

MCU有两个增加的外部中断输入：INT2 (P3.6) 和 INT3 (P3.7)。除了在边沿触发模式(ITx = 1)下，可以被编程为上升沿或下降沿触发，在电平触发模式(ITx = 0)下，可以被编程为高电平或低电平触发外，他们和 /INT0(P3.2) or /INT1 (P3.3)一样。以下为涉及到他们的操作的特殊功能寄存器

XICON (地址=C0H, 外部中断控制寄存器)

7	6	5	4	3	2	1	0
IL3	EX3	IE3	IT3	IL2	EX2	IE2	IT2

IL3：外部中断3电平控制位：1：上升沿/高电平触发；0：下降沿/低电平触发。

EX3：外部中断3使能位。

IE3：外部中断3中断标志位。

IT3：外部中断3类型控制位。1：边沿触发；0：电平触发。

IL2：外部中断2电平控制位：1：上升沿/高电平触发；0：下降沿/低电平触发。

EX2：外部中断2使能位。

IE2：外部中断2中断标志位。

IT2：外部中断2类型控制位。1：边沿触发；0：电平触发。

IP (地址=B8H, 中断优先级寄存器)

7	6	5	4	3	2	1	0
PX3	PX2	PT2	PS	PT1	PX1	PT0	PX0

PX3：外部中断3优先级位

PX2：外部中断2优先级位

PT2：定时器2中断优先级位

PS：串行口中断优先级位

PT1：定时器1优先级位

PX1：外部中断1优先级位

PT0：定时器0中断优先级位

PX0：外部中断0优先级位

14. 键盘中断

键盘中断功能主要用于当P0口等于或不等于某个值时产生一个中断，这个功能可以用作总线地址识别或键盘键码识别. 用户可以通过配置特殊功能寄存器来配置端口，从而区分不同的任务。

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)，用来定义连接到P0口哪些引脚可以触发中断。键盘模式寄存器(KBPATN)，用来定义与P0口值进行比较的值。条件匹配时硬件置位键盘中断控制寄存器(KBCON)中的键盘中断标志(KBIF) 。若 AUXIE 寄存器中的 EKBI位为1且EA=1，则会产生一个中断。键盘中断控制寄存器(KBCON)中的PATN_SEL位用于定义比较匹配是“相等”还是“不等”。

为了使键盘中断跟“键盘”中断一样，用户需要设置 KBPATN=0xFF 和 PATN_SEL=0 (不相等)。这样的话，将任意连接到 KBMASK 寄存器定义的相应P0口的按键，按下时硬件就会置位中断标志KBIF，并当中断使能时产生一个中断。这个中断可以将CPU从空闲模式或掉电模式下唤醒。

下面是键盘中断功能相关特殊功能寄存器：

KBPATN (地址=D5H，键盘模式寄存器)

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

KBPATN.7-0： 键盘模式，复位值为 0xFF.

KBCON (地址=D6H，键盘控制寄存器，复位值=xxxx,xx00B)

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

PATN_SEL： 模式匹配极性选择.

置位：P0端口值**等于** KBPATN 中用户定义模式时产生中断。

清零：P0端口值**不等于** KBPATN 中用户定义模式时产生中断。

KBIF：键盘中断标志。P0端口值与KBPATN，KBMASK，PATN_SEL设置条件匹配时置位。需要软件写‘0’清零。

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

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

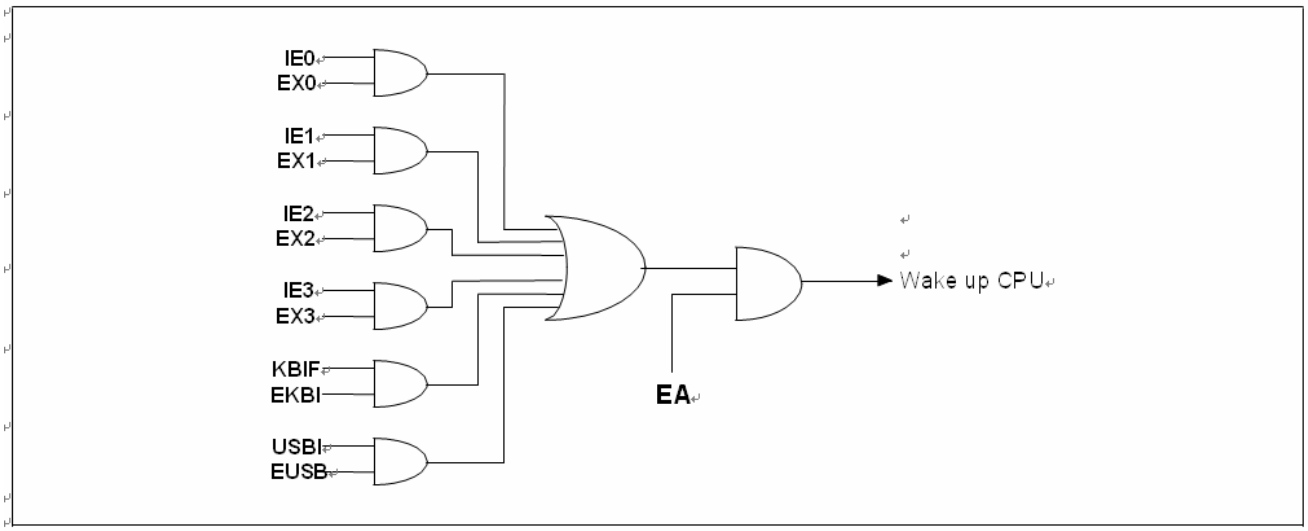
KBMASK.7： 置位时，使能P0.7作为键盘中断源 (KBI7).
KBMASK.6： 置位时，使能P0.6作为键盘中断源 (KBI6).
KBMASK.5： 置位时，使能P0.5作为键盘中断源 (KBI5).
KBMASK.4： 置位时，使能P0.4作为键盘中断源 (KBI4).
KBMASK.3： 置位时，使能P0.3作为键盘中断源 (KBI3).
KBMASK.2： 置位时，使能P0.2作为键盘中断源 (KBI2).
KBMASK.1： 置位时，使能P0.1作为键盘中断源 (KBI1).
KBMASK.0： 置位时，使能P0.0作为键盘中断源 (KBI0).

15. 从掉电模式唤醒

当CPU进入掉电模式，外部中断如 (/INT0、 /INT1、INT2、 INT3)，键盘中断，USB中断等可以将CPU唤醒，前提是这些中断被使能。

15.1. 掉电唤醒源

以下配图说明了掉电唤醒源的组成



如果CPU被唤醒，中断服务程序会执行，直到遇到RETI语句，接下来执行的语句是将其置为掉电模式的下一个语句。

15.2. 掉电唤醒示例代码

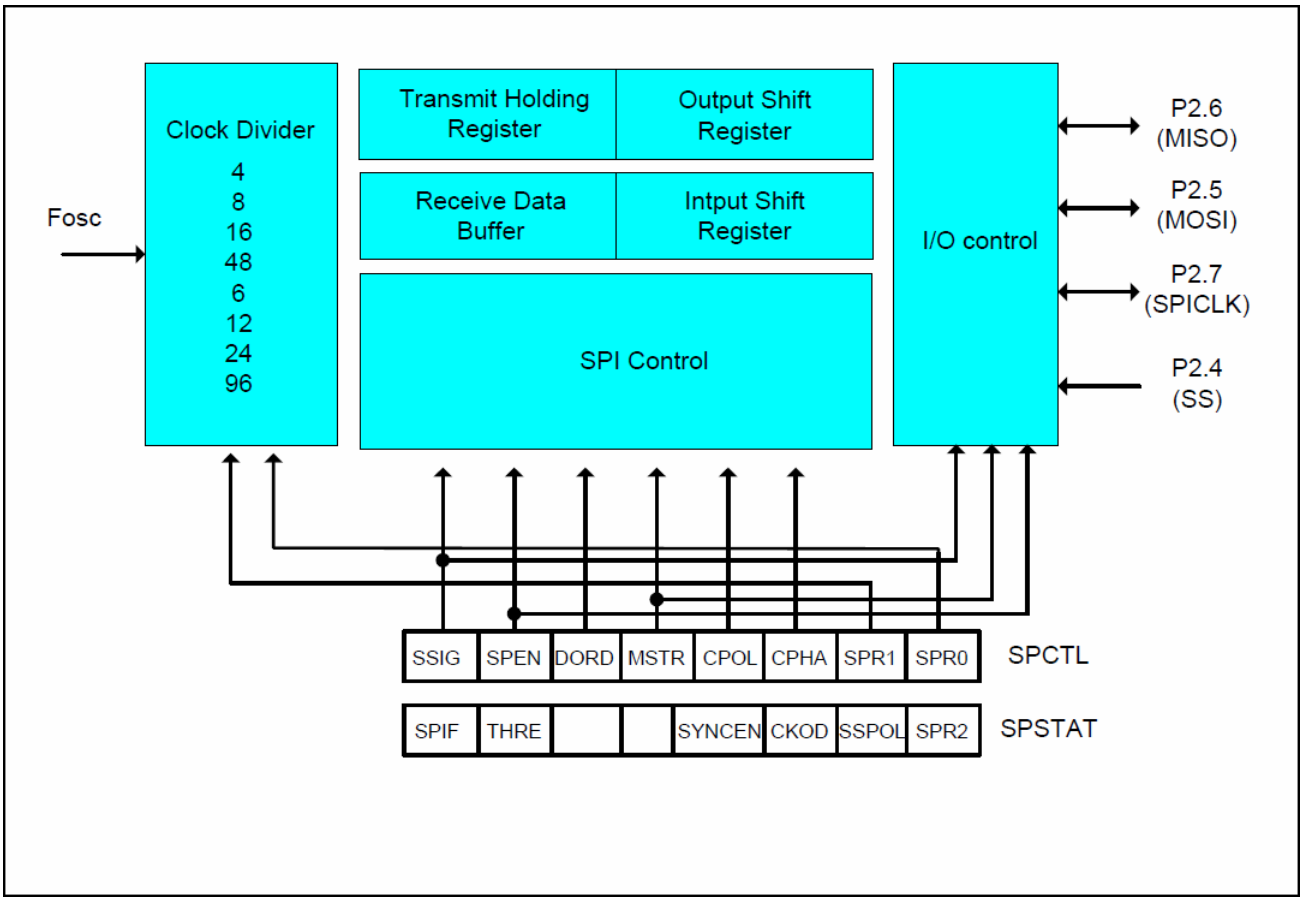
示例： 使用/INT0 唤醒的例子.

```
;*************************************************************************************************
; 用/INT0 中断来唤醒休眠
;*************************************************************************************************
INT0            BIT     0B2H        ; P3.2
EA              BIT     0AFH        ; IE.7
EX0             BIT     0A8H        ; IE.0
                CSEG    AT   0000h
                JMP     start
;
                CSEG    AT   0003h  ; /INT0 中断向量, 地址0003h
                JMP     IE0_isr
IE0_isr:
                CLR     EX0
                ;... 嵌入应用程序
                ;...
                RETI
;
start:
                ;...
                ;...
                SETB    INT0        ; 拉高 P3.2
                CLR     IE0         ; 清除 /INT0 中断标志
                SETB    IT0         ; may select falling-edge/low-level triggered
                SETB    EA          ; 充许中断
                SETB    EX0         ; 使能 /INT0 中断
                ORL     PCON,#02h   ; 使MCU进入休眠模式
                NOP                 ; ！ 注意:这必须有一个 NOP
摘要：
               ;如果/INT0是下降沿触发, MCU被唤醒, 进入"IE0_isr",
               ;然后回到这里继续运行 ！
               ;...
               ;...
;
```

16. 串行外围接口(SPI)

MG84FL54B提供了一个高速串行通讯接口（SPI）。SPI接口是一种全双工、高速同步通讯总线，有两种操作模式：主机模式和从机模式。无论哪种模式，在12MHz系统时钟下支持高达4Mbps的通讯速度。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。

SPI结构图



SPI接口有4个引脚： MISO (P2.6), MOSI (P2.5), SPICLK (P2.7) 和/SS (P2.4)：

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

• /SS是从机选择端。典型配置中，SPI主机可以使用其某个端口选择某一个SPI设备作为当前从机，一个SPI 从机设备使用它的/SS引脚确定自己是否被选中。但是当 SPEN (SPCTL.6) = 0 或 SSIG (SPCTL.7) = 1时，/SS引脚被忽略。

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

下面是SPI操作相关寄存器：

SPCTL (地址=85H, SPI控制寄存器)

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

SSIG： 忽略/SS

若SSIG=1, MSTR位决定该设备是主机还是从机。

若SSIG=0, /SS引脚决定该设备是主机还是从机。

SPEN： SPI使能

若SPEN=1, SPI使能。

若SPEN=0, SPI接口禁用，所有SPI引脚可作为通用I/O口使用。

DORD： SPI 数据顺序

1 ： 传送数据时先传数据字节最低位。

0 ： 传送数据时先传数据字节最高位。

MSTR： 主机/从机模式选择

CPOL： SPI时钟极性选择

1： SPICLK待机是为高电平，SPICLK时钟脉冲前沿是下降沿，而后沿是上升沿。

0： SPICLK待机是为低电平，SPICLK时钟脉冲前沿是上升沿，而后沿是下降沿。

CPHA： SPI时钟相位选择

1： SPICLK脉冲前沿放数据，后沿采样。

0： /SS引脚低电平 (SSIG=0)开始放数据并在SPICLK后沿改变数据. 数据在SPICLK的前沿采样。

(注 ： 若 SSIG=1, CPHA必须不为1，否则就是为定义操作.)

SPR1-SPR0： SPI 时钟速率选择 (主机模式，与SPR2配合)

{SPR2,SPR1,SPR0}= 000:Fosc/3 100: Fosc/16

001:Fosc/6 101: Fosc/24

010:Fosc/8 110: Fosc/48

011:Fosc/12 111: Fosc/96 (*Fosc是系统时钟.)*

SPSTAT (地址=84H, SPI状态寄存器)

7	6	5	4	3	2	1	0
SPIF	THRE	-	-	-	-	-	SPR2

SPIF： SPI 传输完成标志

当一次串行传输完成时，SPIF位置位，若SPI中断允许，则会产生一个中断。若 /SS 引脚在主机模式下被拉低且SSIG=0，SPIF位也会置位以表明“模式改变”。SPIF通过软件写“1”清零。

THRE（只读）：发送保持寄存器（YHR）空标志

0：表明THR是“空的”。

此位当THR为空时由硬件清零。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。

SPR2：SPI时钟速率选择（与SPR1和SPR0配合）

SPDAT (地址=86H, SPI数据寄存器)

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

SPDAT有两个物理缓冲器用于传输过程中的写入和读取

（1） 对于写入：SPDAT 用 THR暂存将要装入移位输出寄存器输出的数据，

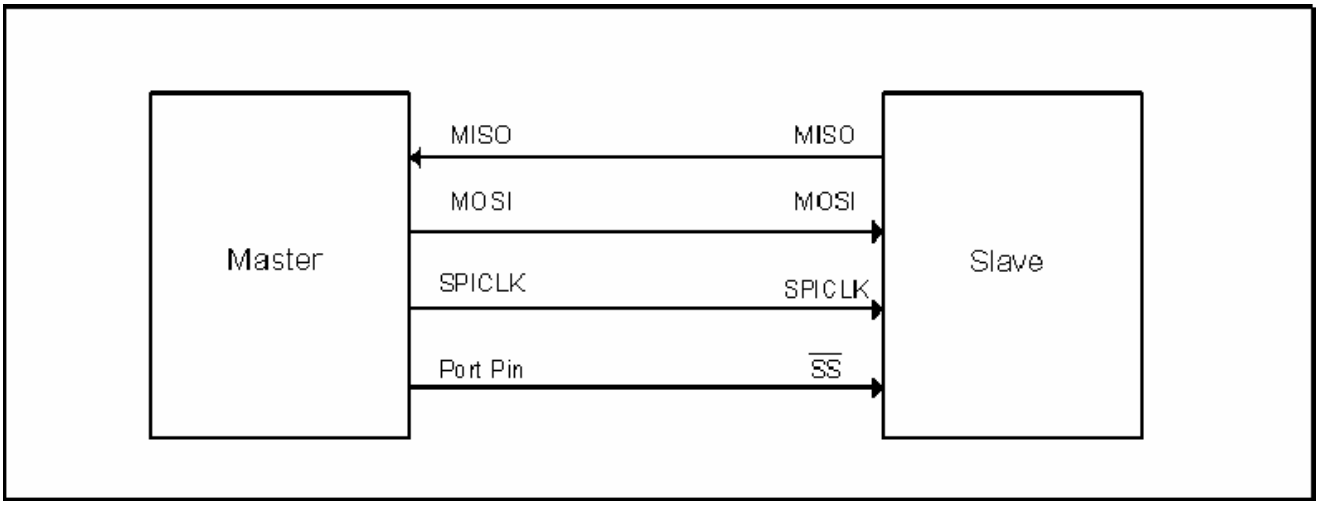
（2） 对于读取：SPDAT 用 输入移位寄存器，包含接收到的数据。

16.1 典型 SPI 配置

16.1.1 单主机和单从机

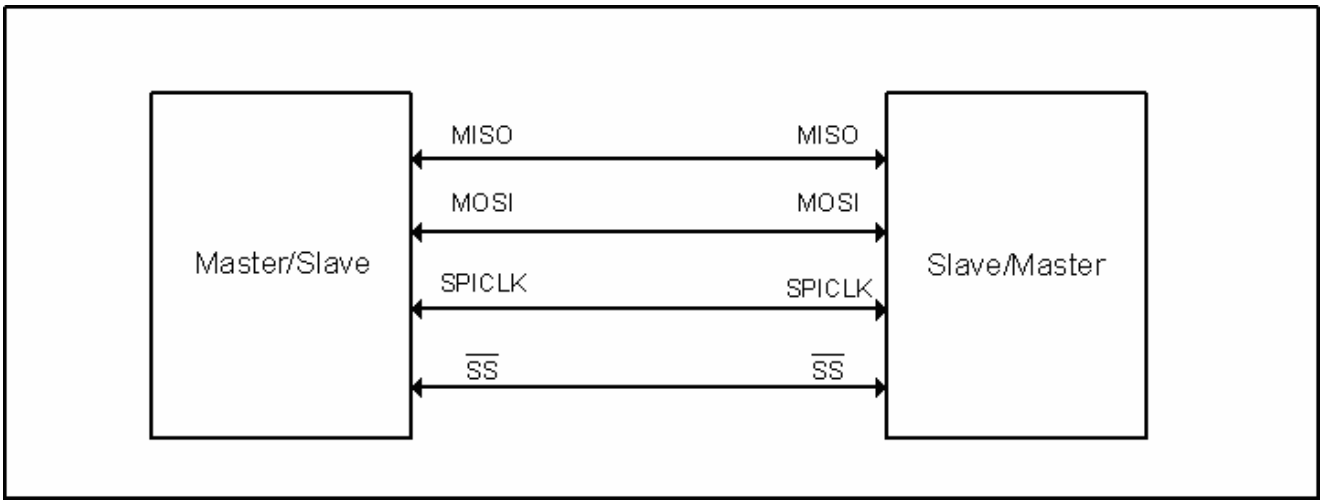
对于主机： 任何端口，包括P2.4 (/SS)，都可以用来控制从机的/SS片选引脚。

对于从机： SSIG 为 ‘0’， /SS引脚决定该设备是否被选中。



16.1.2 双驱动器,可以是主机或从机

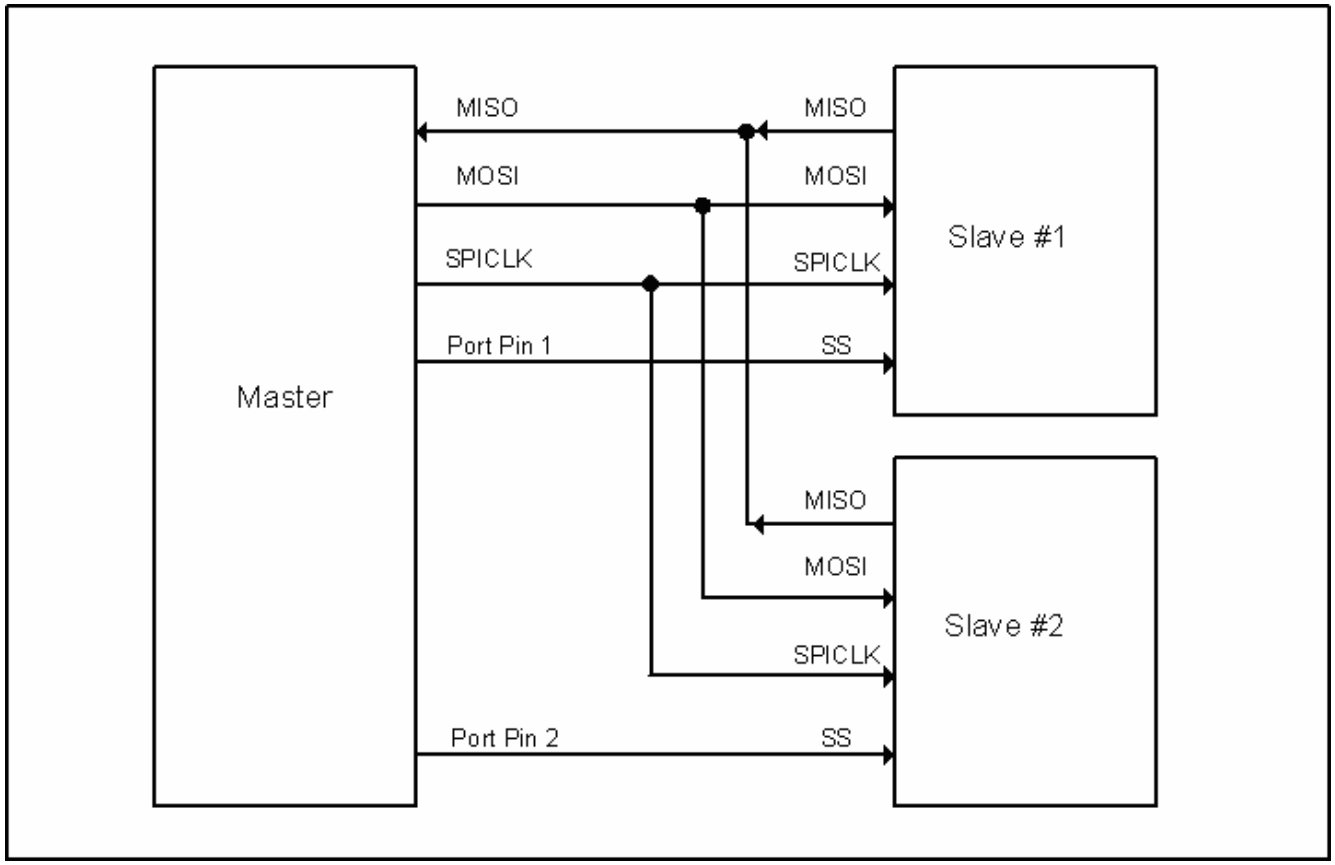
两个彼此连接的设备，均可成为主机或从机，没有SPI操作时，都可以通过设置MSTR=1， SSIG=0, P2.4 (/SS)双向口配置成主机。任何一方要发起传输，它可以配置P2.4位输出并强行拉低，使另一个设备发生“被改成从机模式”事件。



16.1.3.单主机和多从机

对于主机：任何端口，包括 P2.4 (/SS)，都可以用来控制从机的/SS 片选引脚。

对于从机：SSIG 为 ‘0’， 通过各自的/SS 引脚决定是否被选中。



16.2. SPI配置

Table: SPI 主从机选择

SPEN (SPCTL.6)	SSIG (SPCTL.7)	/SS -引脚	MSTR (SPCTL.4)	模式	MISO -引脚	MOSI -引脚	SPICLK -引脚	备注
0	X	X	X	SPI 禁用	输入	输入	输入	P2.4~P2.7 用作通用I/O
1	0	0	0	从机 (被选中)	输出	输入	输入	被选择为从机
1	0	1	0	从机 (未被选中)	高阻	输入	输入	未被选中
1	0	0	1 → 0	从机 (通过模式改变)	输出	输入	输入	若/SS被拉低，MSTR被硬件自动清‘0’，模式被改为从机
1	0	1	1	主机 (待机)	输入	高阻	高阻	MOSI和SPICLK在主机待机时被置为高阻，以防止总线冲突。
				主机 (活动)		输出	输出	MOSI和SPICLK在主机活动时是推挽的。
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” 表示”无需关心”。

16.2.1. 从机注意事项

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

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

16.2.2. 主机注意事项

SPI通讯中，传输总是由主机发起。如果SPI已经使能（SPEN=1）并且作为主机运行，则向SPI数据寄存器（SPDAT）写入数据即可启动SPI时钟生成器和数据传输。大约在半个到一个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，主机可以通过拉低相应的/SS引脚来选择一个从机作为当前从机。写入主机SPDAT寄存器的数据将从主机的MOSI引脚移出，输出到从机的MOSI引脚。同时，被选中的从机的SPDAT寄存器中的数据，从MISO引脚输出到主机的MISO引脚。

移出一字节后，SPI时钟发生器停止，传输完成标志（SPIF）置位，若SPI中断使能则产生一个中断。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。主机的数据移向从机的同时，从机的数据也在移向主机。这意味着在一次传输过程中，主从机数据进行了交换。

16.2.3. /SS引脚的模式改变

若 SPEN=1, SSIG=0, MSTR=1 且 /SS引脚=1，SPI使能在主机模式。在这种情况下，另一个主机可以通过将此/SS引脚拉低来选择该设备作为从机并且开始向其发送数据。为避免总线冲突，该设备将成为一个从机，从而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SPI中断使能，则会产生一个SPI中断。用户程序应当经常检查MSTR位。若该位因被选为从机而清零，而用户又想继续保持该SPI在主机模式下，则必须重新置位MSTR，否则该SPI将保持从机模式。

16.2.4.无写数据冲突指示

SPI在发送与接收方向均为双缓冲。在前一个数据载入到输出移位寄存器用于发送前，新的数据不能写入发送保持寄存器（THR）。当THRE位（SPSTAT.6）置位表明用户可以向THR寄存器写入新的数据用于接下来的传输。相对于有写数据冲突指示的体系而言，这种体系结构有更高的数据吞吐量。

16.2.5. SPI 时钟频率选择

SPI时钟频率选择（主机模式），使用SPCTL寄存器的SPR1和SPR0位来选择，如下表所示。

Table: 串行时钟速率

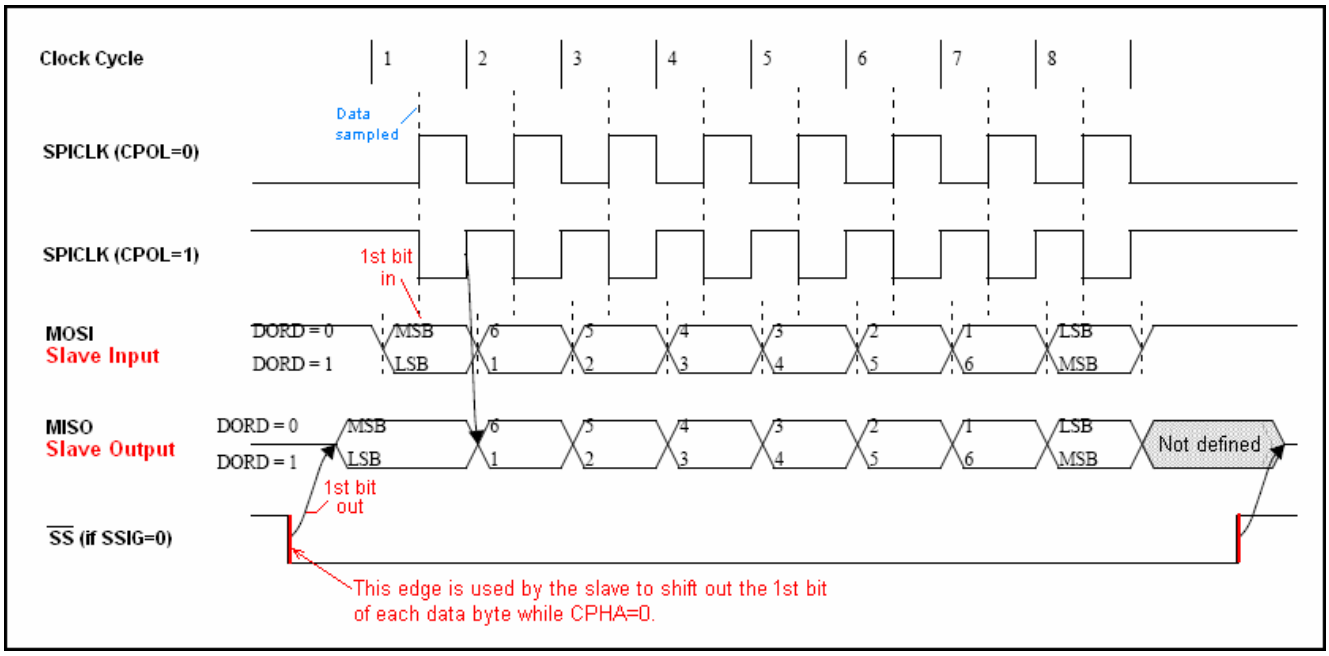
SPR2	SPR1	SPR0	SPI 时钟速率 @ Fosc=12MHz	Fosc 分频
0	0	0	3 MHz	4
0	0	1	2 MHz	6
0	1	0	1.5 MHz	8
0	1	1	1 MHz	12
1	0	0	750 KHz	16
1	0	1	500 KHz	24
1	1	0	250 KHz	48
1	1	1	125 KHz	96

这里的Fosc为系统时钟。

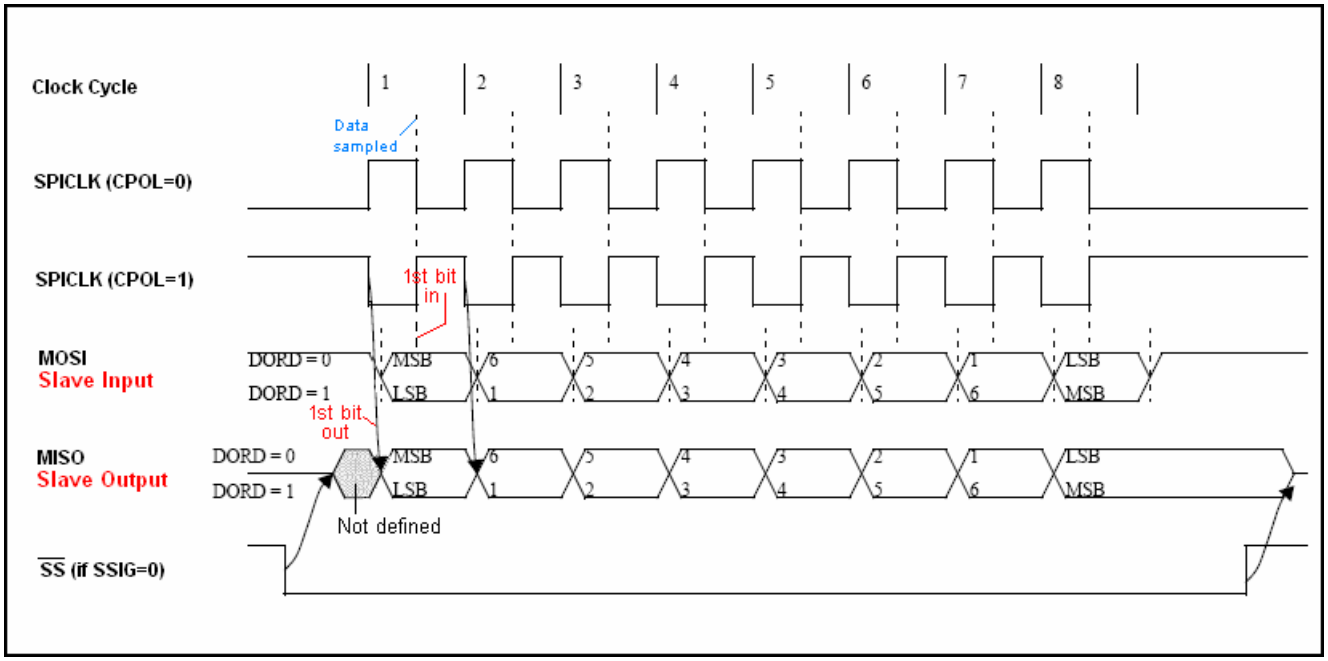
16.3. 数据模式

时钟相位 (CPHA) 位允许用户设定数据采样和改变时的时钟沿。时钟极性位，CPOL，允许用户设定时钟极性。下面的图例显示了不同CPHA设置下的SPI时序。

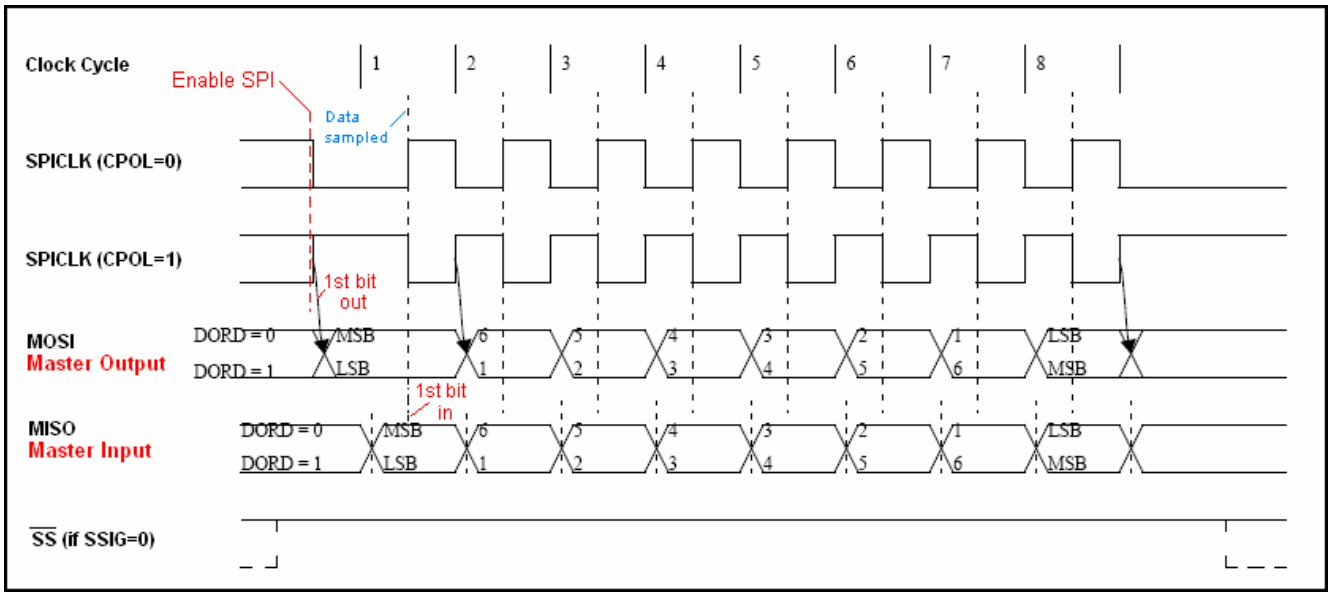
16.3.1. SPI 从机传送， CPHA=0



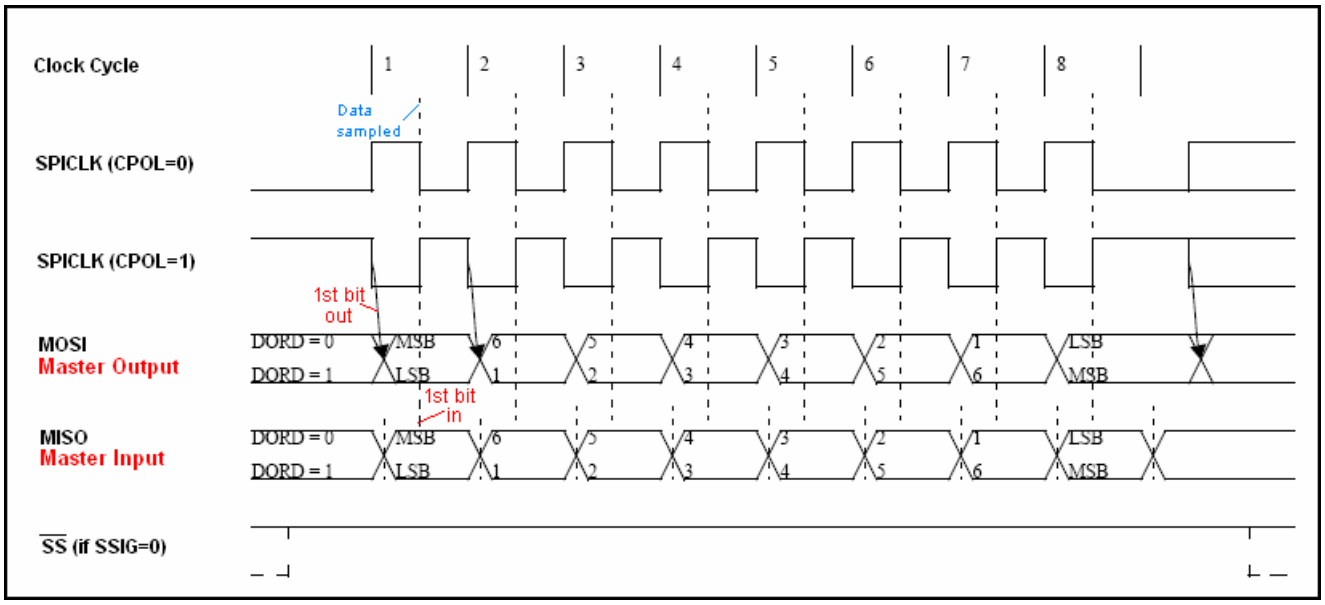
16.3.2. SPI从机传送，CPHA=1



16.3.3. SPI 主机传送，CPHA=0



16.3.4. SPI主机传送，CPHA=1



17. 双线串行接口 (TWSI)

特点

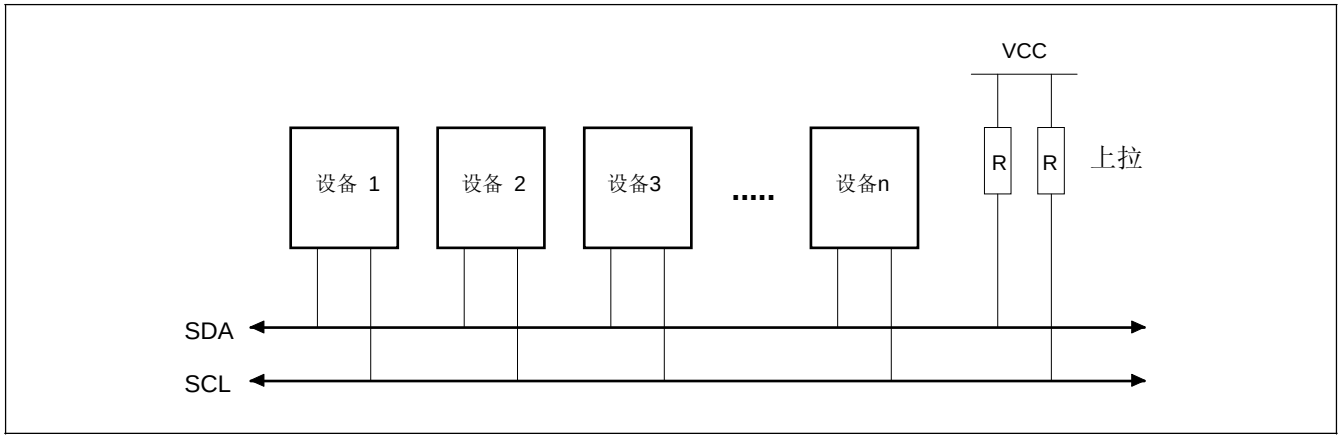
- 简单，却强大而灵活的通讯接口，只需要两根线。
- 支持主机与从机操作，并且器件可以工作于发送器模式或接收器模式。
- 7位的地址空间允许多达128个不同的从机地址。
- 支持多主机仲裁。
- 高达400kHz的数据传输速率。
- 可编程的从机地址，支持公共地址。

描述

双线串行接口（TWSI）很适合于典型的处理器应用。TWSI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（SCL），一根用于数据（SDA）。唯一需要的外部硬件就是在TWSI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自己的地址，而且TWSI协议解决了总线仲裁的问题。

CPU通过下面四个特殊功能寄存器与TWSI相连：SIADR（串行接口地址寄存器），SIDAT（串行接口数据寄存器），SICON（串行接口控制寄存器），SISTA（串行接口状态寄存器）。TWSI硬件通过两根数据线与串行总线相连：SDA（串行数据线，P2.1），SCL（串行时钟线，P2.0）。

TWSI 总线的连接



17.1. TWSI特殊功能寄存器

串行接口地址寄存器，SIADR，地址=D1H

CPU可以直接对此寄存器进行读写。SIADR不受TWSI硬件的影响。当TWSI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，寄存器的高七位必须被用于储存本机的从机地址，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TWSI总线收到的首位相对应。

SIADR (地址=D1H, TWSI 地址寄存器)

7	6	5	4	3	2	1	0
(A6)	(A5)	(A4)	(A3)	(A2)	(A1)	(A0)	GC

|<---------- 自身从机地址 ----------->|

串行接口数据寄存器，SIDAT，地址=D2H

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TWSI正处于一个被定义的状态并且串行中断标记位（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，主机切换为从机的过程会在SIDAT中产生一个正确的数据。

SIDAT (地址=D2H, TWSI 数据寄存器)

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

<---------- 移位方向 ---------->

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TWSI硬件控制，CPU访问不能。串行数据在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标记。注意发送出去的8位数据会移回SIDAT。

串行接口控制寄存器，SICON，地址=F8H

CPU可以直接读写此寄存器。其中两个位会受TWSI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。

SICON (地址=F8H, TWSI 控制寄存器)

7	6	5	4	3	2	1	0
CR2	ENSI	STA	STO	SI	AA	CR1	CR0

ENSI, TWSI 硬件使能位
ENSI为“0”时，SDA与SCL输出为高阻态，SDA与SCL输入信号被忽略，TWSL处于未被寻址（not-addressed）从机状态，SICON的STO位被强制为“0”，但不影响其他位。P2.1（SDA）与P2.0(SCL)可用作漏极开路I/O引脚。ENSI为“1”时，TWSI使能，P2.1和P2.0端口寄存器必须设置为逻辑1以用于接下来的串行通讯。

STA，开始标志

当STA位被置位而进入主机模式时，TWSI硬件将检查串行总线的状态，若总线空闲将产生一个开始信号。若总线为非空闲，TWSI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TWSI已经是处于主机模式并且一个或多个字节已被发送或接受的情况下置位，TWSI会发送一个REPEATED START信号。STA可以在任何时候置位，也可以在TWSI是一个被寻址的从机时置位。当STA位复位时，无START或REPEATED START信号产生。

STO，停止标志

当TWSI处于主机模式时，置位STO会向串行总线发送一个STOP信号。当在总线上检测到STOP信号，TWSI硬件清除STO标记。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TWSI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。STO标志自动被硬件清零。如果STA与STO位同时置位，若TWSI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。

SI，串行中断标志

当一个新的TWSI状态出现在SISTA寄存器时，SI标志会被硬件置位。如果TWSI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状态可以获得的 F8H。当SI置位时，SCL线上的低电平会延长，并且串行传输暂停。SCL线上的高电平不受SI标志影响。SI必须由软件清零。SI标记复位时不会产生中断请求，SCL线上的时钟也不会延长。

AA,断言应答位

如果AA标志设为“1”，一个ACK（SDA低电平）将在SCL的应答时钟周期内回复，当：

1）接收到本机的从机地址。
2）TWSI处于主机/接收模式时，接收到一字节的数据。
3）TWSI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NACK（SDA高电平）将在SCL的应答时钟周期内回复，当：

1）TWSI处于主机/接收模式时，接收到一字节的数据。
2）TWSI处于已被寻址的从机/接收模式时，接收到一字节的数据。

CR0, CR1 与 CR2, 时钟速率位

TWSI处于主机模式时，这三个位决定串行时钟频率。当TWSI处于从机模式时，时钟频率并不重要，因为TWSI会自动同步任何主机的时钟频率，高达100KHz。下表给出不同的时钟速率设置：

Table: 串行时钟速率

CR2	CR1	CR0	串行时钟速率 @ Fosc=12MHz	Fosc 分频
0	0	0	1.5 MHz	8
0	0	1	1M KHz	12
0	1	0	400 KHz	30
0	1	1	200 KHz	60
1	0	0	100 KHz	120
1	0	1	50 KHz	240
1	1	0	25 KHz	480
1	1	1	12.5 KHz	960

这里的 Fosc 是系统时钟.

状态寄存器，SISTA，地址=D3H

SISTA是一个8位的只读寄存器。低三位总是为0，高5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的其他取值用于定义相应的TWSI状态。当进入这些状态的一种时，会请求串行中断（SI=1）。在SI被硬件置位时，一个有效的状态编码会存在于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，如在一个地址/数据的内部或者刚好在应答位上。

SISTA (地址=D3H, TWSI 状态寄存器)

7	6	5	4	3	2	1	0
(b7)	(b6)	(b5)	(b4)	(b3)	(b2)	(b1)	(b0)

17.2. 工作模式

TWSI有4种工作模式：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。SCION的STA，STO与AA决定SI被软件清零后TWSI硬件的下一步操作。当下一步操作完成，SISTA会更新为新的状态编码并且SI会在同一时间被硬件置位。这时，中断服务程序开始运行（如果TWSI中断使能），并且新的状态编码可以用于决定运行哪个代码分支。

17.2.1.主机发送模式

在主机发送模式，一定数量的字节数据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
比特率	1	0	0	0	x	比特率	

CR0，CR1和CR2定义了串行位速率。ENSI必须设置为逻辑1来使能TWSI。如果AA位复位，在其他设备成为总线的主机时，TWSI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，TWSI不能进入从机模式。STA，STO与SI必须复位。

现在，可以通过使用SETB指令置位STA，从而进入主机发送模式。TWSI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以及数据方向位（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，串行中断标记（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REPEATED START信号后（状态10H），TWSI可以通过向SIDAT写入SLA+R进入主机接收模式。

17.2.2.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为（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受到应答位后，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。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REPEATED START信号后（状态10H），TWSI可以通过向SIDAT写入SLA+W进入主机发送模式。

17.2.3. 从机发送模式

在从机发送模式，一定量的字节数据会发送到一个主机接收器。SIADR与SICON必须写入如下数据，用来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

|<---------- 自身从机地址---------->|

高七位为当TWSI被主机寻址时的响应地址。如果LSB（GC）置位，TWSI将响应广播地址（00H）；否则广播地址将被忽略。

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

当处于从机模式时，CR0，CR1与CR2不会对TWSI产生影响。ENSI必须设置为“1”来使能TWSI。AA位必须置位，使TWSI应答自身从机地址或者广播地址。STA，STO和SI必须清除为“0”。

SIADR与SICON初始化后，TWSI会等待直到其从机地址被寻址并且数据方向位为“1”（R），TWSI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TWSI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位被复位，TWSI将发送完当前字节的数据然后进入C0H或C8H状态。TWSI会转换到未被寻址从机模式，如果主机继续传输，TWSI将会忽略主机接收器，因此主机总是接收到“1”。当AA复位时，TWSI不会回应其从机地址或广播地址，但是会继续监听串行总线。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TWSI。

17.2.4. 从机接收模式

在从机接收模式，会从主机发送器接收一定数量的数据。数据传送的初始化与从机发送模式一致。

SIADR与SICON初始化后，TWSI会等待直到其从机地址被寻址并且数据方向位为“0”（W），TWSI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中读出一个可用的状态编码。这些状态编码可以用作指向一个中断服务程序，在这些状态编码下对应的操作将在随后的工作流程图中详细叙述。当TWSI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）。

如果在传输的过程中AA位被复位，TWSI会在接收到下一个字节后回复NACK（逻辑1）。当AA复位时，TWSI不会响应自身的从机地址或者广播地址，但是会继续监听串行总线。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TWSI。

17.3. 混合状态

有两个SISTA编码没有与已定义的TWSI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TWSI未涉及串行传输时。

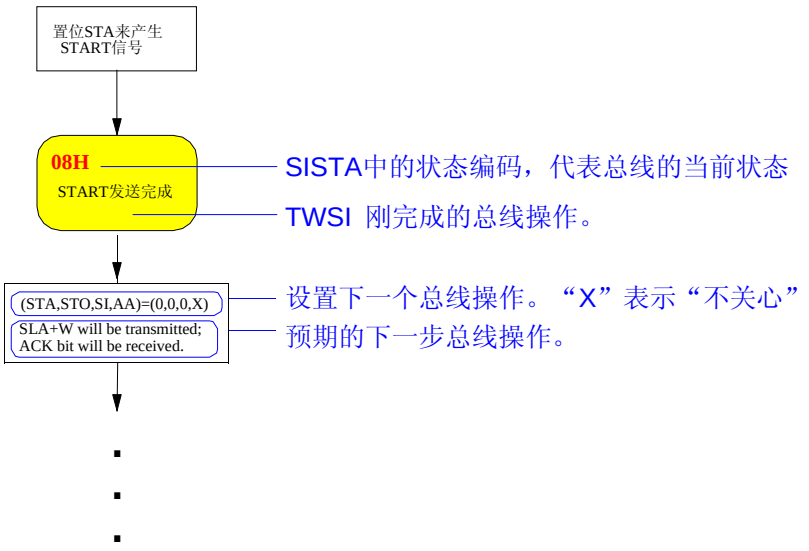
S1STA = 00H:
这个状态编码表明在一个TWSI串行传输过程中发生总线错误。当一个START或STOP信号在一帧的不合法位置发送时，总线错误就会发生。例如：在传输一个字节地址、数据时，或者在应答位。总线错误也会在外界干扰扰乱内部TWSI信号时发生。当总线错误发生时，SI被置位，STO标记必须置位并且SI必须软件清零以用来从总线错误恢复。这会使TWSI进入“未被寻址”（not-addressed）从机状态（已定义的状态）并且清除STO标记（SICON的其它位不受影响）。SDA与SCL线将被释放（不会发送STOP信号）。

17.4. 使用TWSI

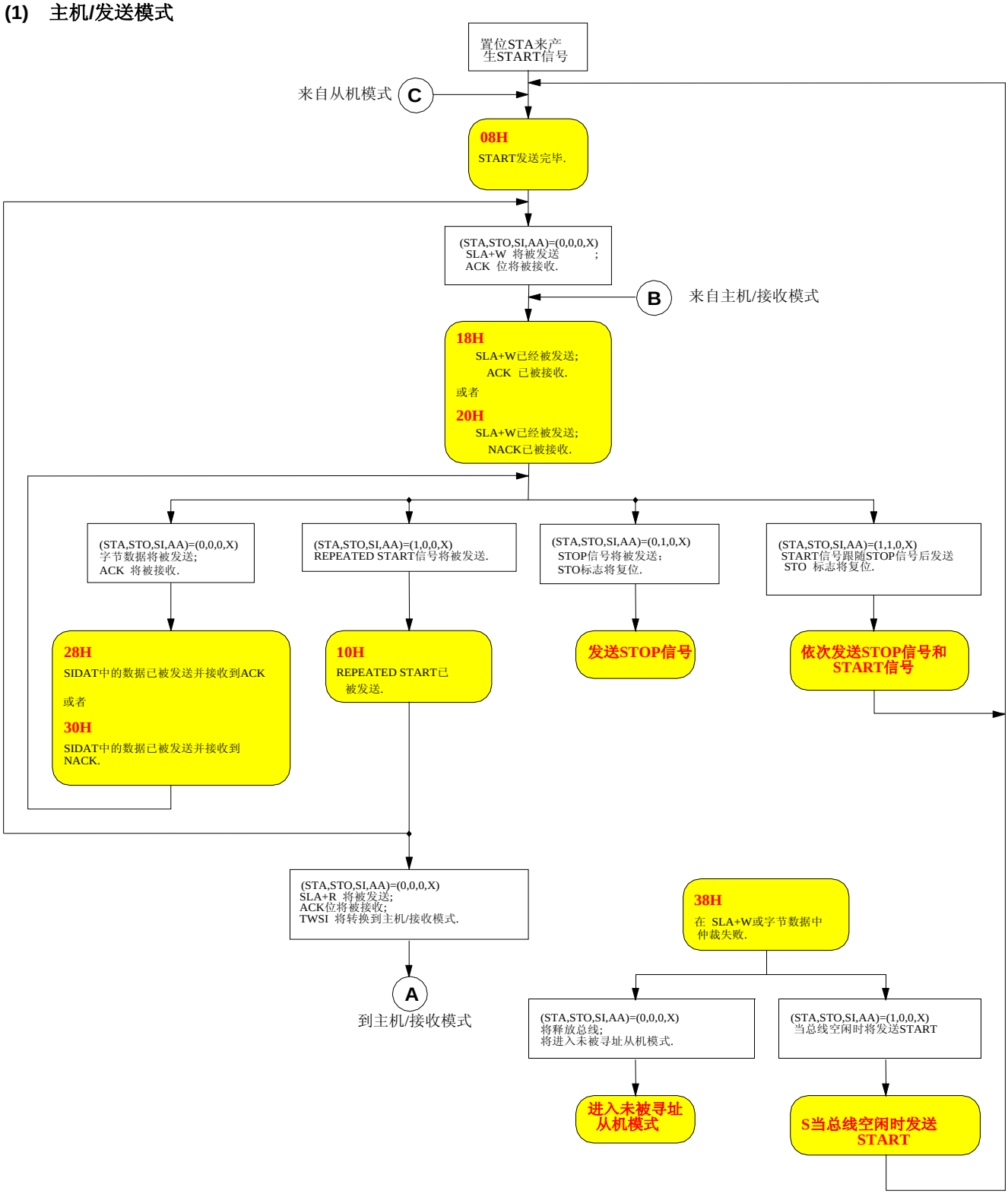
TWSI是面向字节并且基于中断的。中断会在所有总线事件后发生，例如接收到一字节数据或发送START信号。因为TWSI是基于中断的，应用软件可以自由地在一个 TWSI 字节发送的过程中处理其他操作。注意，TWSI 中断标志位（AUXIE.6）与 EA 位允许应用程序选择在 SI 标志出现时是否产生中断请求。当SI标志出现时，表明 TWSI 已经完成一个操作并且等待程序响应。此时状态寄存器 SISTA 保存的状态编码表明TWSI总线的当前状态。用户程序可以通过对STA，STO和AA位（在SICON中）进行适当的编程来决定接下来TWSI总线将如何运行。

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TWSI。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，TWSI 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的动作。当SI清零后TWSI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。

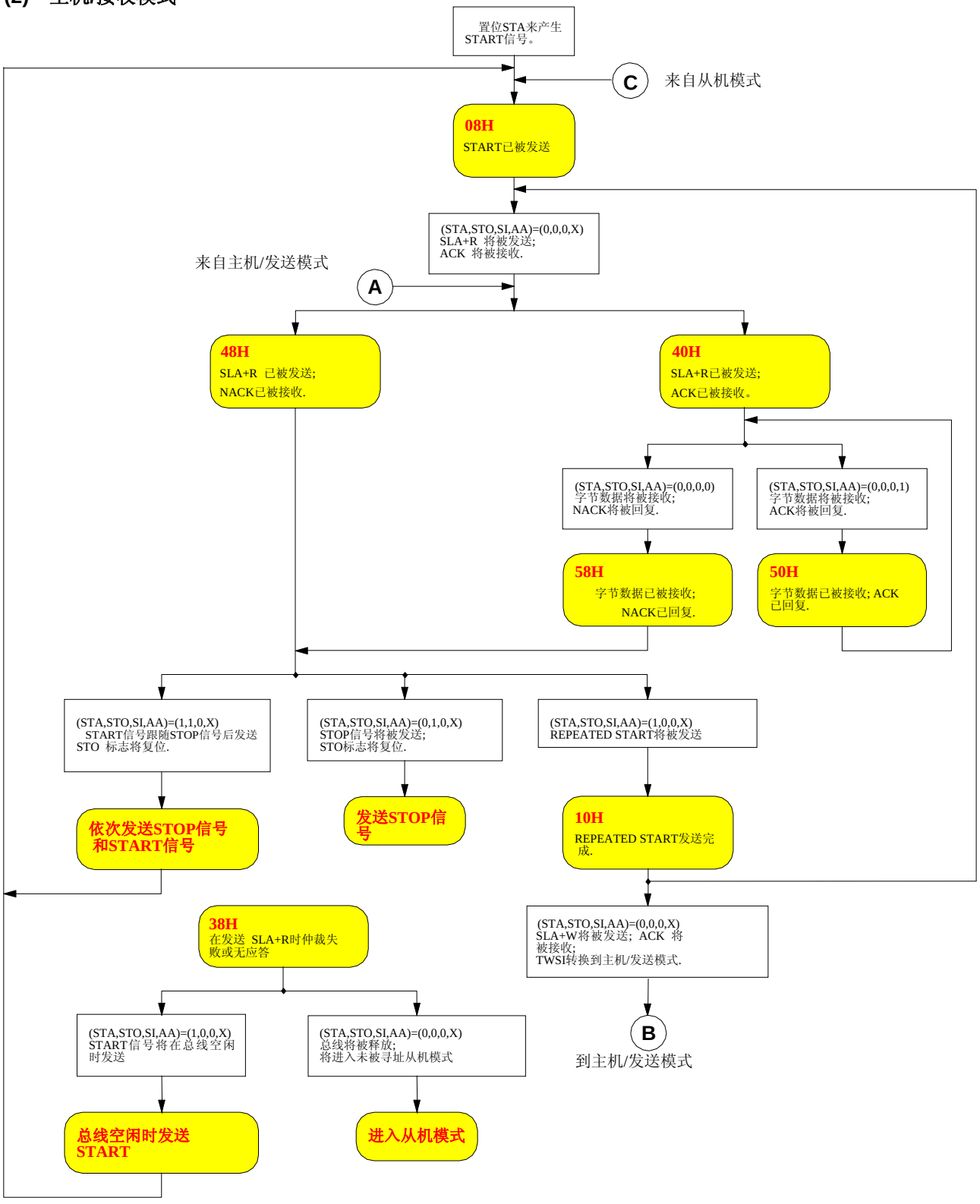
下面的图指出如何阅读流程图。



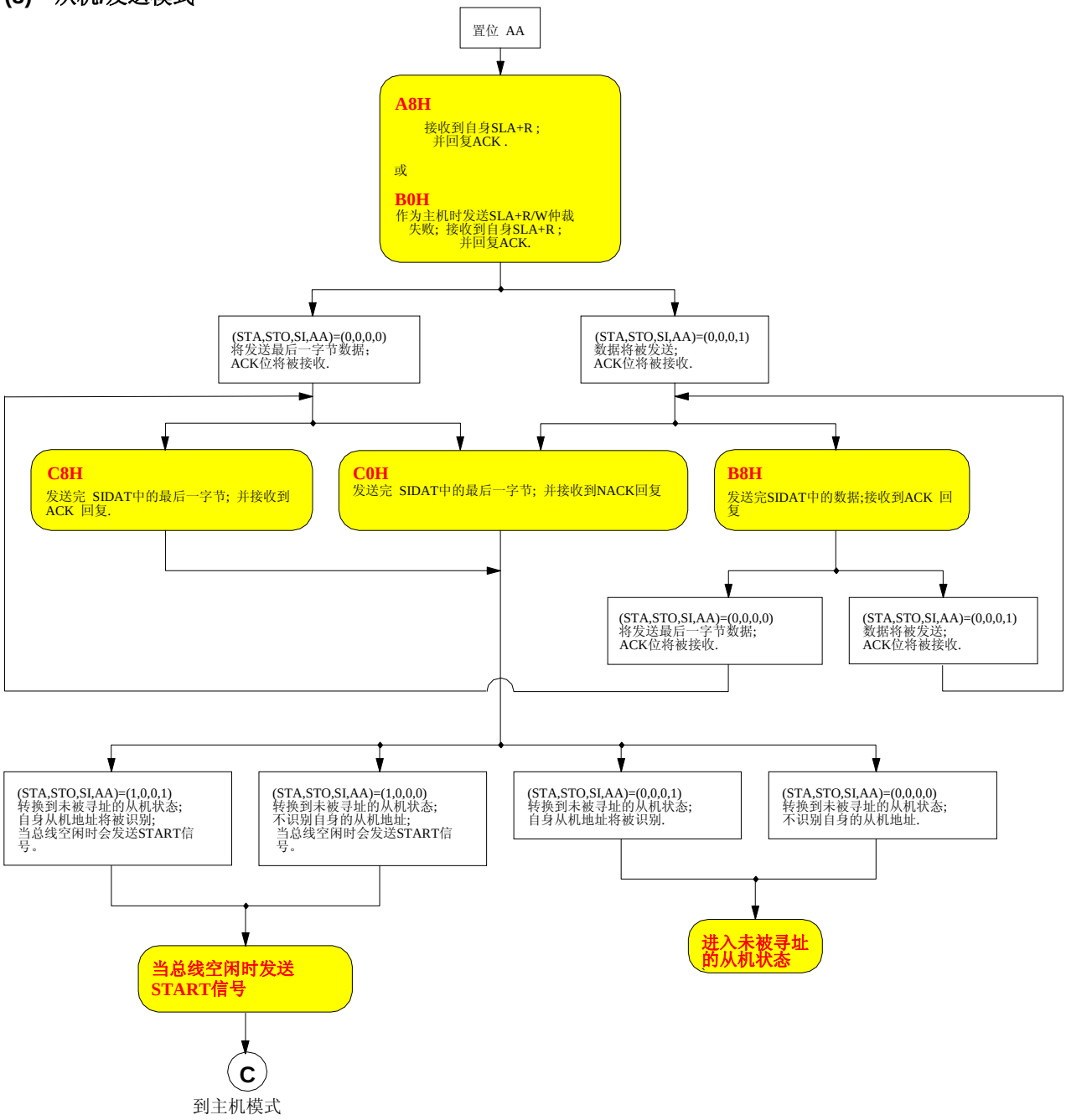
(1) 主机/发送模式


置位STA来产生START信号
来自从机模式
C
08H
START发送完毕.
(STA,STO,SI,AA)=(0,0,0,X)
SLA+W 将被发送 ;
ACK 位将被接收.
B
来自主机/接收模式
18H
SLA+W已经被发送;
ACK 已被接收.
或者
20H
SLA+W已经被发送;
NACK已被接收.
(STA,STO,SI,AA)=(0,0,0,X)
字节数据将被发送;
ACK 将被接收.
(STA,STO,SI,AA)=(1,0,0,X)
REPEATED START信号将被发送.
(STA,STO,SI,AA)=(0,1,0,X)
STOP信号将被发送;
STO标志将复位.
(STA,STO,SI,AA)=(1,1,0,X)
START信号跟随STOP信号后发送
STO 标志将复位.
28H
SIDAT中的数据已被发送并接收到ACK
或者
30H
SIDAT中的数据已被发送并接收到NACK.
10H
REPEATED START已被发送.
发送STOP信号
依次发送STOP信号和START信号
(STA,STO,SI,AA)=(0,0,0,X)
SLA+R 将被发送;
ACK位将被接收;
TWSI 将转换到主机/接收模式.
38H
在 SLA+W或字节数据中仲裁失败.
A
到主机/接收模式
(STA,STO,SI,AA)=(0,0,0,X)
将释放总线;
将进入未被寻址从机模式.
(STA,STO,SI,AA)=(1,0,0,X)
当总线空闲时将发送START
进入未被寻址从机模式
S当总线空闲时发送START

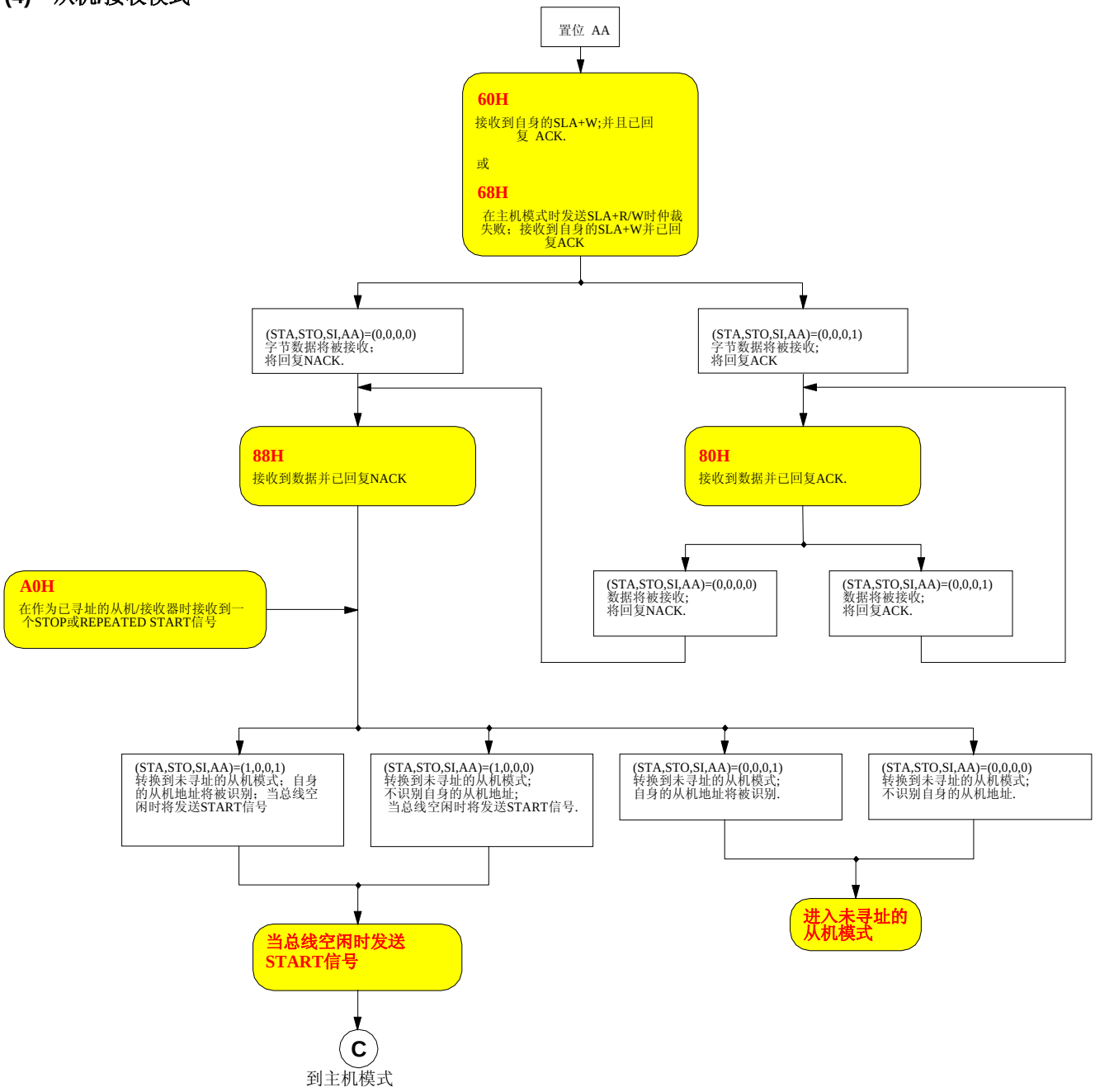
(2) 主机/接收模式


置位STA来产生START信号。
C
来自从机模式
08H
START已被发送
(STA,STO,SI,AA)=(0,0,0,X)
SLA+R 将被发送;
ACK 将被接收.
来自主机/发送模式
A
48H
SLA+R 已被发送;
NACK已被接收.
40H
SLA+R已被发送;
ACK已被接收。
(STA,STO,SI,AA)=(0,0,0,0)
字节数据将被接收;
NACK将被回复.
(STA,STO,SI,AA)=(0,0,0,1)
字节数据将被接收;
ACK将被回复.
58H
字节数据已被接收;
NACK已回复.
50H
字节数据已被接收; ACK已回复.
(STA,STO,SI,AA)=(1,1,0,X)
START信号跟随STOP信号后发送
STO 标志将复位.
(STA,STO,SI,AA)=(0,1,0,X)
STOP信号将被发送;
STO标志将复位.
(STA,STO,SI,AA)=(1,0,0,X)
REPEATED START将被发送
依次发送STOP信号和START信号
发送STOP信号
10H
REPEATED START发送完成.
38H
在发送 SLA+R时仲裁失败或无应答
(STA,STO,SI,AA)=(0,0,0,X)
SLA+W将被发送; ACK 将被接收;
TWSI转换到主机/发送模式.
(STA,STO,SI,AA)=(1,0,0,X)
START信号将在总线空闲时发送
(STA,STO,SI,AA)=(0,0,0,X)
总线将被释放;
将进入未被寻址从机模式
B
到主机/发送模式
总线空闲时发送START
进入从机模式

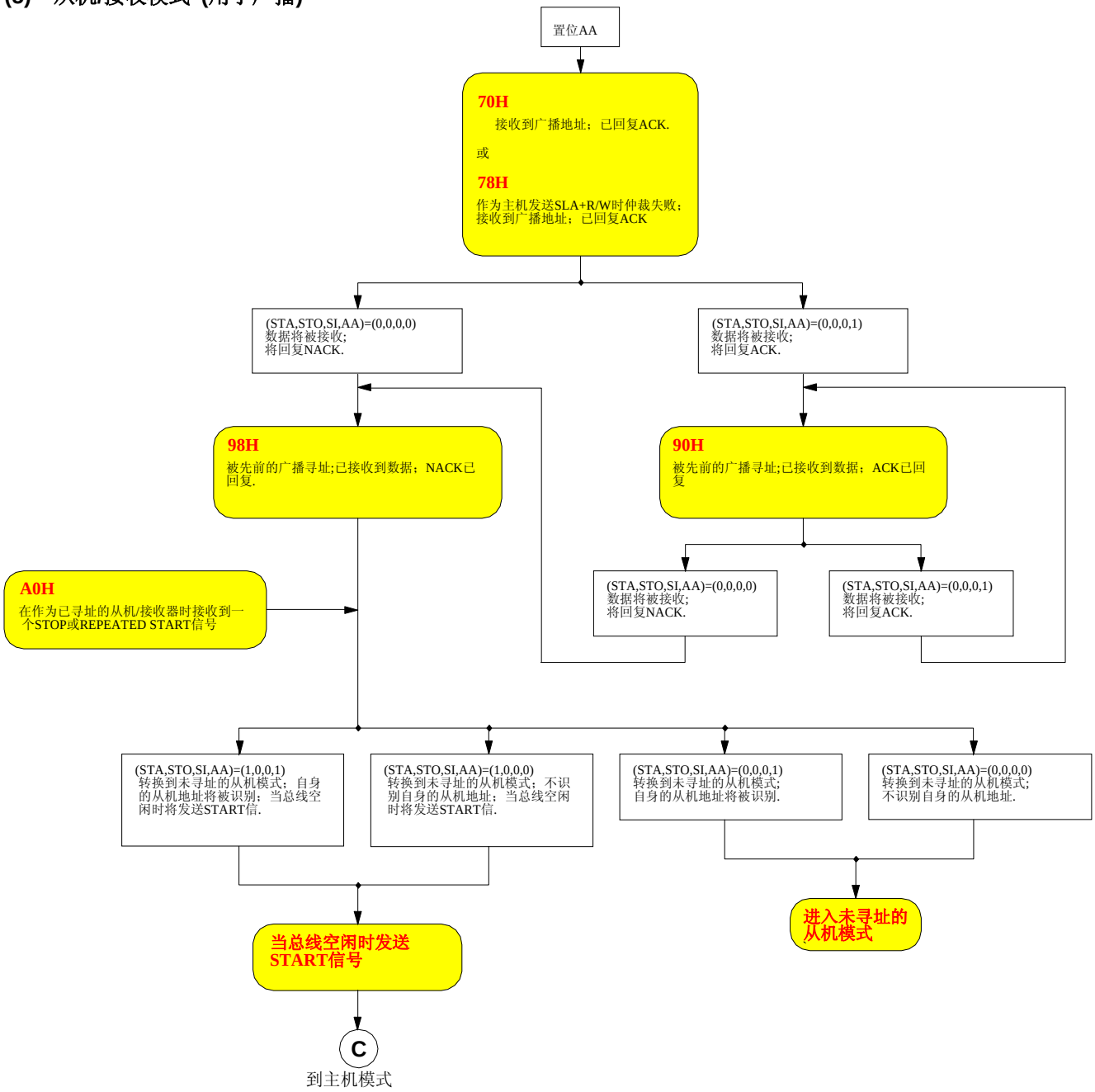
(3) 从机/发送模式


置位 AA
A8H
接收到自身SLA+R；
并回复ACK .
或
B0H
作为主机时发送SLA+R/W仲裁
失败；接收到自身SLA+R；
并回复ACK.
(STA,STO,SI,AA)=(0,0,0,0)
将发送最后一字节数据；
ACK位将被接收.
(STA,STO,SI,AA)=(0,0,0,1)
数据将被发送；
ACK位将被接收.
C8H
发送完 SIDAT中的最后一字节；并接收到ACK 回复.
C0H
发送完 SIDAT中的最后一字节；并接收到NACK回复
B8H
发送完SIDAT中的数据;接收到ACK 回复
(STA,STO,SI,AA)=(0,0,0,0)
将发送最后一字节数据；
ACK位将被接收.
(STA,STO,SI,AA)=(0,0,0,1)
数据将被发送；
ACK位将被接收.
(STA,STO,SI,AA)=(1,0,0,1)
转换到未被寻址的从机状态;
自身从机地址将被识别;
当总线空闲时会发送START信号。
(STA,STO,SI,AA)=(1,0,0,0)
转换到未被寻址的从机状态;
不识别自身的从机地址;
当总线空闲时会发送START信号。
(STA,STO,SI,AA)=(0,0,0,1)
转换到未被寻址的从机状态;
自身从机地址将被识别.
(STA,STO,SI,AA)=(0,0,0,0)
转换到未被寻址的从机状态;
不识别自身的从机地址.
进入未被寻址的从机状态
当总线空闲时发送START信号
C
到主机模式

(4) 从机/接收模式


置位 AA
60H
接收到自身的SLA+W;并且已回复 ACK.
或
68H
在主机模式时发送SLA+R/W时仲裁失败；接收到自身的SLA+W并已回复ACK
(STA,STO,SI,AA)=(0,0,0,0)
字节数据将被接收;
将回复NACK.
(STA,STO,SI,AA)=(0,0,0,1)
字节数据将被接收;
将回复ACK
88H
接收到数据并已回复NACK
80H
接收到数据并已回复ACK.
A0H
在作为已寻址的从机/接收器时接收到一个STOP或REPEATED START信号
(STA,STO,SI,AA)=(0,0,0,0)
数据将被接收;
将回复NACK.
(STA,STO,SI,AA)=(0,0,0,1)
数据将被接收;
将回复ACK.
(STA,STO,SI,AA)=(1,0,0,1)
转换到未寻址的从机模式；自身的从机地址将被识别；当总线空闲时将发送START信号
(STA,STO,SI,AA)=(1,0,0,0)
转换到未寻址的从机模式;
不识别自身的从机地址;
当总线空闲时将发送START信号.
(STA,STO,SI,AA)=(0,0,0,1)
转换到未寻址的从机模式;
自身的从机地址将被识别.
(STA,STO,SI,AA)=(0,0,0,0)
转换到未寻址的从机模式;
不识别自身的从机地址.
进入未寻址的从机模式
当总线空闲时发送START信号
C
到主机模式

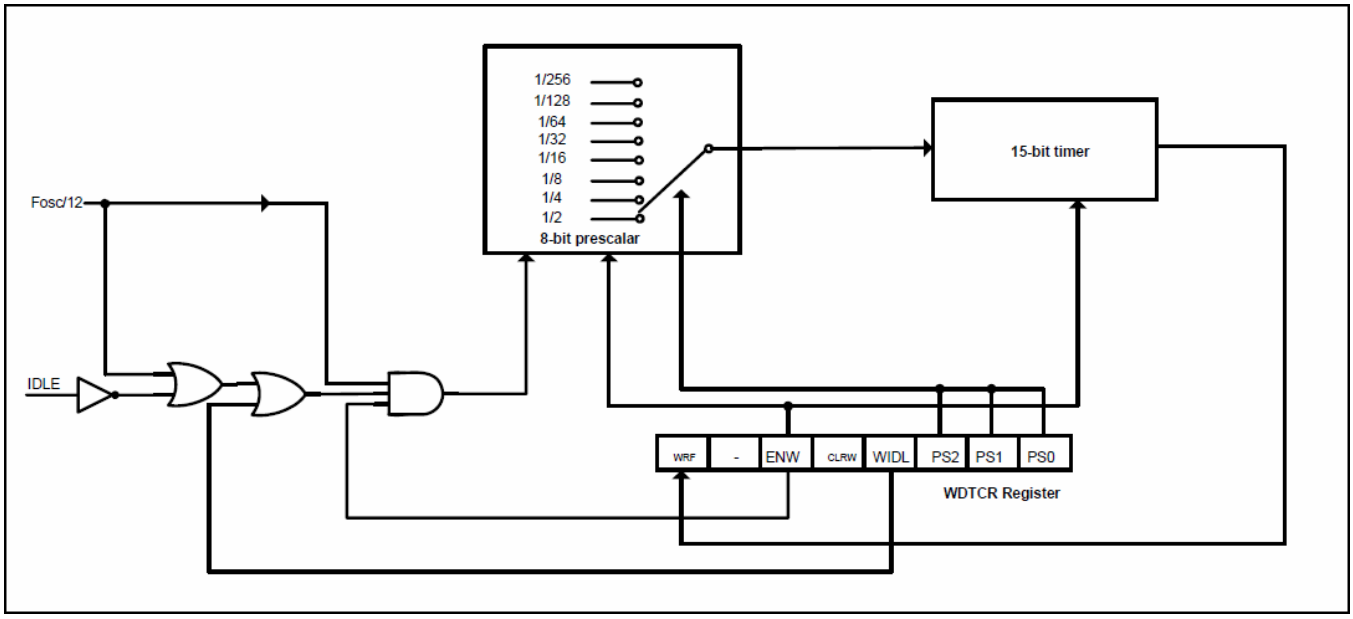
(5) 从机/接收模式 (用于广播)



18. 单次使能看门狗定时器（One-Time-Enabled WDT)

WDT被看作是CPU程序跑飞时的一种恢复方法。WDT由一个15位的自由运行计数器，一个8位预分频器和一个控制寄存器（WDTCR）组成。WDT可以使用系统时钟（Fosc）。结构图如下所示：

18.1. WDT 结构图



用户必须置位ENW（WDTCR.5）来使能WDT。当WDT使能后，计数器会每隔（***12 x 预分频 / Fosc***）增加一。用户必须在WDT溢出前向CLRW位（WDTCR.4）写“1”来将其清零。当WDT溢出，MCU会自动复位并重新运行。

为什么WDT被称为“单次使能”“One-time Enabled”？这是因为：*一旦通过置位ENW来使能WDT，除非通过上电复位来清除ENW位，否则没有其他方法来禁用WDT*。在硬件（RST引脚）复位、软件复位和WDT复位后，WDTCR寄存器会保存先前的值而不改变。譬如：如果WDTCR现时的值为0x2D，当前面所述的三种复位发生后，其值仍然保持为0x2D而不是0x00。只有上电复位才可以将其初始化为0x00.

WDTCR (地址=E1H, 看门狗定时器控制寄存器)

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

注意：这是一个只写的寄存器，并且只能通过上电复位来恢复为初始值

WRF：WDT复位标记。当WDT溢出时，此位由硬件置位，并且需要由软件清零。

ENW：使能WDT，置位使能WDT。*（注意：一旦置位，只能通过上电复位来清零）*

CLRW：清零WDT。向此位“**写1**”来清零WDT。*（注意：不需要通过“写0”来清除）*

WIDL：WDT空闲状态。置位此位可以使WDT在MCU空闲状态时继续计数

PS2～PS0：预分频选择。参考下面的表：

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

18.2. 空闲状态和掉电状态下的WDT

WIDL位（WDTCR.3）决定WDT是否在空闲状态继续计数。置位此位可以使WDT在空闲状态继续计数。

18.3. WDT硬件自动使能

WDTCR除了可以被软件初始化外，也可以通过硬件选项HWENW，HWWIDL和HWPS[2:0]来在上电时自动初始化。这需要使用一个通用编程器或烧录器来编程，如下所述：

如果HWENW被编程为“使能”（enabled），则硬件会在上电时自动对WDTCR寄存器进行如下初始化：

(1) 置位ENW

(2) 加载HWWIDL到WIDL位

(3) 加载HWPS[2:0]到PS[2:0].

例如：：

I如果HWWIDL 和 HWPS[2:0] 分别被编程为1和5, 则**WDTCR在MCU上电时**将被初始化为 0x2D 。如下所示：

WDTCR (Watch-Dog-Timer Control Register)

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

18.4. WDT 溢出周期

WDT溢出周期由下面公式确定：

2^{15} x (12 x 预分频 / Fosc), or ***2^{15} x (12 x 预分频 / RCosc)***.

下面的图标给出MCU运行于6MHz与12MHz时WDT的溢出周期。这个周期是用户清除WDT以防止芯片复位的最大时间间隔。

Table: WDT 在Fosc = 6MHz & 12MHz时的溢出周期

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

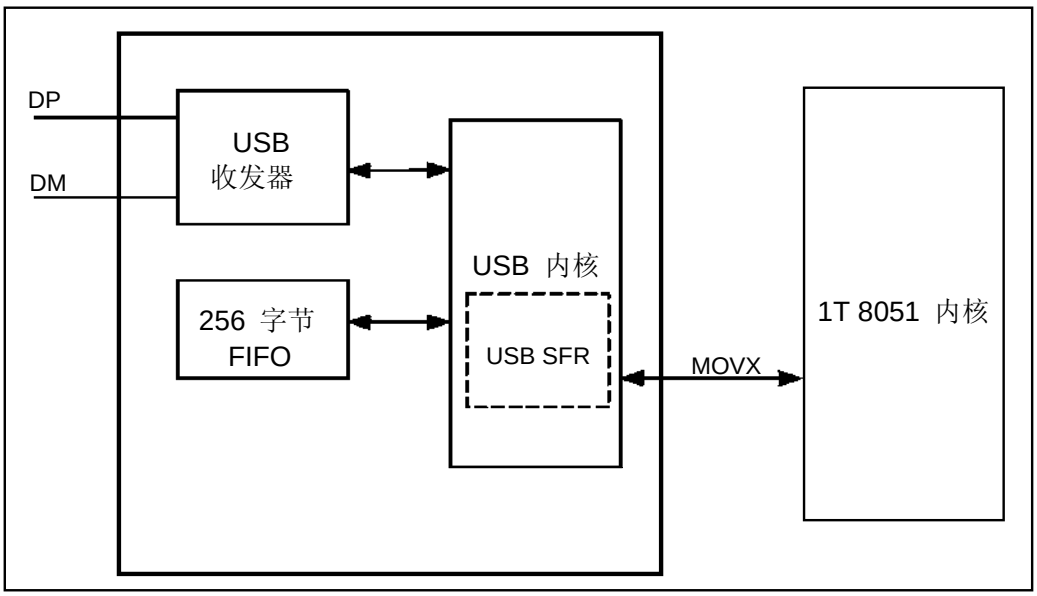
18.5. WDT示例代码

条件: Fosc=6MHz
目标: WDT 溢出周期 = 1.048 秒

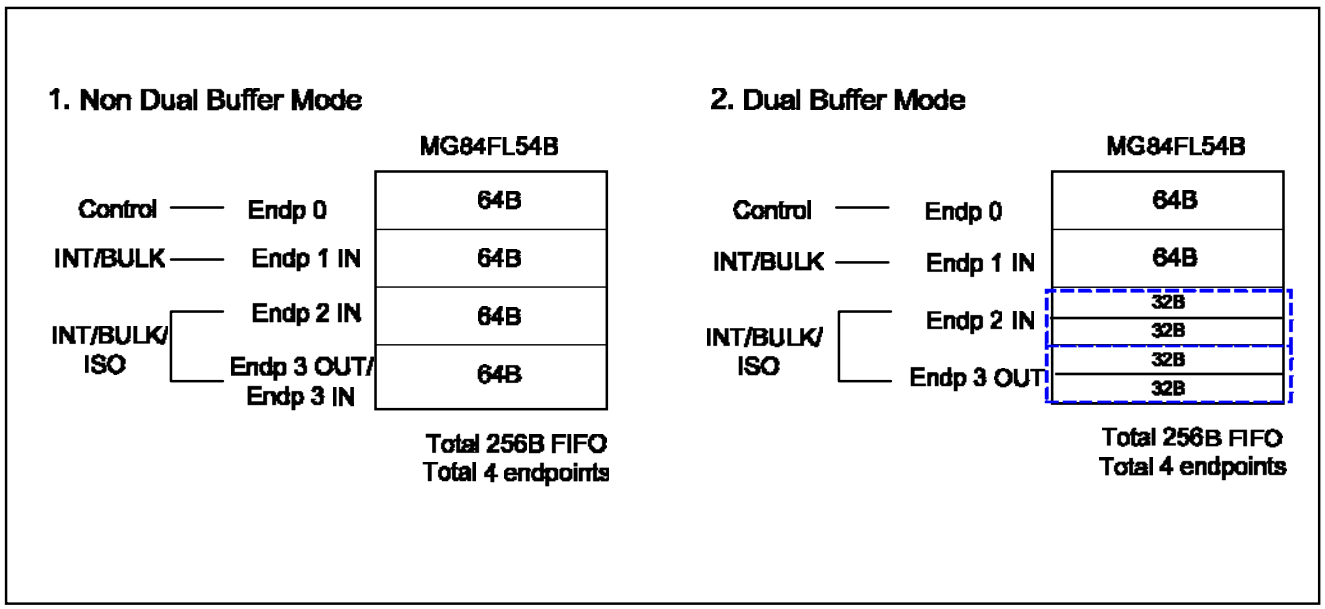
```
WDTCR_buf DATA  30h                ;为WDTCR寄存器声明一个缓冲空间
                                   ;(因为WDTCR为只写的寄存器)
start:
          ;...
          ;...

          MOV    WDTCR_buf,#00h   ;清零缓冲空间

          ANL    WDTCR_buf,#0F8h ;(PS2,PS1,PS0)=(0,1,1), 预分频=16
          ORL    WDTCR_buf,#03h   ;@Fosc=6MHz, 看门狗溢出周期=1.048s
          MOV    WDTCR,WDTCR_buf ;

          ORL    WDTCR_buf,#20h   ;使能WDT
          MOV    WDTCR,WDTCR_buf ;写入WDTCR寄存器

main_loop:
          ORL    WDTCR_buf,#10h   ;清除WDT
          MOV    WDTCR,WDTCR_buf ;
          ;...
          ;...
          JMP    main_loop

          ANL    WDTCR buf,#0DFh ;禁止WDT

          MOV    WDTCR,WDTCR_buf ;
```

19. 通用串行总线 (USB)

19.1. USB 结构图



19.2. USB FIFO 管理



19.3. USB特殊功能寄存器

用户需要通过使能PLL（置位 EN_PLL）和使能USB功能（置位 EN_USB）来激活USB操作。清零 EN_USB 位会让停止USB操作并使USB功能模块进入掉电模式。这两个控制位位于CKCON2寄存器中，如下：

CKCON2 (地址=BFH, 时钟控制寄存器 2)

7	6	5	4	3	2	1	0
-	-	OSCDR0	-	EN_USB	EN_PLL	PLL_RDY	CK_SEL

EN_USB: USB功能使能控制位。置位/清零来使能/禁用USB功能

EN_PLL: PLL使能位。1：使能；0：禁用。

PLL_RDY:此位为只读的。如果此位置位则表明PLL已锁定，如果清零则PLL未锁定。

用于USB操作的特殊功能寄存器位于外部数据地址空间，并且与外部物理数据储存器共用0xFF00-0xFFFF地址。也就是说，用户应该使用“***MOVX @DPTR***”指令来访问这些USB特殊功能寄存器。

19.3.1. USB SFR 映射地址

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
FFF8H									FFFFH
FFF0H		**EPINDEX**	**TXSTAT**	**TXDAT**	**TXCON**		**TXCNT**		FFF7H
FFE8H									FFEFH
FFE0H		**EPCON**	**RXSTAT**	**RXDAT**	**RXCON**		**RXCNT**		FFE7H
FFD8H	**UADDR**	**IEN**	**UIE**	**UIFLG**	**UIE1**	**UIFLG1**			FFDFH
FFD0H									FFD7H
FFC8H		**UPCON**							FFCFH
FFC0H	**DCON**		**SIOCTL**						FFC7H
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	

19.3.2. USB SFR 描述

符号	描述	"MOVX" 地址	位符号 Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	复位值
DCON	设备控制寄存器	C0H	-	-	-	-	EP3DIR	-	-	-	xxxx0xxxB
UADDR	USB地址寄存器	D8H	-	UADD6	UADD5	UADD4	UADD3	UADD2	UADD1	UADD0	x0000000B
UPCON	USB电源控制寄存器	C9H	CONEN	-	URWU	-	-	URST	URSM	USUS	0x0xx000B
IEN	中断允许寄存器	D9H	-	-	-	-	-	EFSR	EF	-	xxxxx00xB
UIE	USB中断允许寄存器	DAH	SOFIE	ASOFIE	-	UTXIE2	-	UTXIE1	URXIE0	UTXIE0	00x0x000B
UIFLG	USB中断标志寄存器	DBH	SOFIF	ASOFIF	-	UTXD2	-	UTXD1	URXD0	UTXD0	00x0x000B
UIE1	USB中断允许寄存器1	DCH	-	-	-	-	-	-	URXIE3	UTXIE3	xxxxxx00B
UIFLG1	USB中断标志寄存器1	DDH	-	-	-	-	-	-	URXD3	UTXD3	xxxxxx00B
EPINDEX	端点索引寄存器	F1H	-	-	-	-	-	-	EPINX1	EPINX0	xxxxxx00B
EPCON	端点控制寄存器	E1H	RXSTL TXISO	TXSTL TXEPEN	RXDBM	TXDBM	RXISO			RXEPEN	00000000B
RXSTAT	端点接收状态寄存器	E2H	RXSEQ	RXSETUP	STOVW	EDOVW	RXSOVW	ISOOVW	- -		000000xxB
RXDAT	FIFO接收数据寄存器	E3H	RXD7	RXD6	RXD5	RXD4	RXD3	RXD2	RXD1	RXD0	xxxxxxxxB
RXCON	FIFO接收控制寄存器	E4H	RXCLR	-	-	RXFFRC	-	-	-	-	0xx0xxxxB
RXCNT	FIFO接收字节计数寄存器	E6H	-	RXBC6	RXBC5	RXBC4	RXBC3	RXBC2	RXBC1	RXBC0	00000000B
TXSTAT	端点发送状态寄存器	F2H	TXSEQ	-	-	-	TXSOVW	-	-	-	0xxx0xxxB
TXDAT	FIFO发送数据寄存器	F3H	TXD7	TXD6	TXD5	TXD4	TXD3	TXD2	TXD1	TXD0	xxxxxxxxB
TXCON	FIFO发送控制寄存器	F4H	TXCLR	-	-	TXFFRC	-	-	-	-	0xx0xxxxB
TXCNT	FIFO发送字节计数寄存器	F6H	-	TXBC6	TXBC5	TXBC4	TXBC3	TXBC2	TXBC1	TXBC0	00000000B
SIOCTL	串行I/O控制寄存器	C2H	DPI	DMI	-	-	-	-	-	-	xxxxxxxxB

DCON (设备控制寄存器, 地址=C0H, 系统复位值=xxxx-0xxx, 读写)

7	6	5	4	3	2	1	0
-	-	-	-	EP3DIR	-	-	-

Bit7~4： 保留，总是写0。

Bit3： EP3DIR-- USB 端点3 方向选择
当此位置“1”, EP3 将作为一个输入端点。
当此位清“0”, EP3 将作为一个输出端点。默认为输出端点。

Bit2~0： 保留，总是写0。

UADDR (USB 地址寄存器, 地址=D8H, 系统/USB复位值=x000-0000, 读写)

7	6	5	4	3	2	1	0
-	UADD6	UADD5	UADD4	UADD3	UADD2	UADD1	UADD0

Bit7： 保留

Bit6~0： UADD[6：0]-- USB 设备地址
保存USB设备地址的寄存器。在总线枚举时，被写入由主机（HOST）分配的唯一值。

UPCON (USB电源控制寄存器, 地址=C9H, 系统/USB复位值=0x0x-x000, 读写)

7	6	5	4	3	2	1	0
CONEN	-	URWU	-	-	URST	FRSM	FSUS

Bit7： CONEN-- USB连接使能
复位后默认清'0' 。FW 应该置'1' 以使能与上级主机（HOST）或集线器（HUB）的连接。

Bit6： 保留

Bit5： URWU-- USB远程唤醒触发
当微控制器被外部触发唤醒后，此位将被微控制器置位，以发起一个在USB总线上的远程唤醒事件。当远程唤醒结束后将由硬件清零。除非设备被暂停，不要置这一位。

Bit4~3： 保留

Bit2： URST-- USB重置标志
当USB功能模块检测到USB总线复位时由硬件置位。如果置位，会对微控制器产生一个USRT中断。在USB复位中断的服务程序中，通过软件清除此位。软件对此位写"1"清除。

Bit1： URSM-- USB恢复标志
当USB模块检测到USB总线上的恢复状态时由硬件置位。如果置位，会对微控制器产生一个中断。在处理USB功能模块的恢复中断的服务程序中，通过软件清除此位。软件对此位写"1"清除。

Bit0： USUS—USB挂起标志
当USB功能模块检测到USB总线上的挂起状态时由硬件置位。如果置位，会对微控制器产生一个中断。在USB功能模块的挂起中断服务程序中，软件应该在进入挂起模式之前清除此位。软件对此位写"1"清除。

IEN 中断使能寄存器, 地址=D9H, 系统复位值=xxxx-x00x, 读写)

7	6	5	4	3	2	1	0
-	-	-	-	-	EFSR	EF	-

Bit7~3： 保留.

Bit2： EFSR-- 使能USB功能模块的挂起/恢复中断
如果置位，将使能USB功能模块的 UPCON 事件中断。USB功能模块的挂起/恢复/远程唤醒/USB复位中断的使能位。此位不受USB复位影响。默认为"0"。

Bit1： EF—使能USB功能模块的中断标志
如果置位，将使能USB功能模块的 UIFLG 中断。USB功能模块的端点发送/接收完成中断的使能位。此位不受USB复位影响。默认为"0"。

Bit0： 保留

UIE (USB中断使能寄存器，地址=DAH，系统/USB复位值=00x0-x000，读/写)

7	6	5	4	3	2	1	0
SOFIE	ASOFIE	-	UTXIE2	-	UTXIE1	URXIE0	UTXIE0

Bit7： SOFIE—主机（HOST）帧起始（SOF）接收中断使能
如果置位，允许主机（HOST）帧起始（SOF）接收中断。默认为"0"。

Bit6： ASOFIE-- ART SOF 接收中断使能
如果置位，允许 ART 帧起始（SOF）接收中断。默认为"0"。

Bit5： 保留

Bit4： UTXIE2—USB功能模块发送中断使能2
如果置位，允许USB端点2的发送完成中断(UTXD2)。默认为"0"。

Bit3： 保留

Bit2： UTXIE1-- USB 功能模块发送中断使能1
如果置位，允许USB端点1的发送完成中断(UTXD1)。默认为"0"。

Bit1： URXIE0-- USB 功能模块接收中断使能 0
如果置位，允许USB端点0的接收完成中断(URXD0)。默认为"0"。

Bit0： UTXIE0—USB 功能模块发送中断使能 0
如果置位，允许USB端点0的发送完成中断(UTXD0)。默认为"0"。

UIFLG (USB中断标志寄存器，地址=DBH，系统/USB复位值=00x0-x000，读/写)

7	6	5	4	3	2	1	0
SOFIF	ASOFIF	-	UTXD2	-	UTXD1	URXD0	UTXD0

Bit7： SOFIF—主机（HOST） 帧起始（SOF）接收中断标志
检测到主机（HOST）帧起始（SOF）后由硬件置位。微控制器可以读/写清除这一位。软件写"1"时清除。

Bit6： ASOFIF-- ART 帧起始（SOF）接收中断标志
检测到ART SOF后由硬件置位。微控制器可以读/写清除这一位。软件写"1"时清除。

Bit5： 保留

Bit4： UTXD2-- 端点2的USB发送完毕标志
检测到端点2发送完毕后由硬件置位。微控制器可以读/写清除这一位。当软件写"1"时清除。

Bit3： 保留

Bit2： UTXD1-- 端点1的USB发送完毕标志
检测到端点1发送完毕后由硬件置位。微控制器可以读/写清除这一位。当软件写"1"时清除。

Bit1： URXD0-- 端点0的USB接收完毕标志
检测到端点0接收完毕后由硬件置位。微控制器可以读/写清除这一位。当软件写"1"时清除。

Bit0： UTXD0-- 端点0的USB发送完毕标志
检测到端点0发送完毕后由硬件置位。微控制器可以读/写清除这一位。当软件写"1"时清除。

UIE1 (USB中断使能寄存器 1，地址=DCH，系统/USB复位值=xxxx-xx00,读/写)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	URXIE3	UTXIE3

Bit7~2： 保留，总是写 "0"。

Bit1： URXIE3-- USB 功能模块接收中断使能 3
置位将使能USB端点3的接收完毕中断(URXD3)。默认为"0"。

Bit0： UTXIE3-- USB 功能模块发射中断使能 3
置位将使能USB端点3的发送完毕中断(UTXD3)。默认为"0"。

UIFLG1 (USB中断标志寄存器 1，地址=DDH，系统/USB复位值=xxxx-xx00，读/写)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	URXD3	UTXD3

Bit7~2： 保留.

Bit1： URXD3-- 端点3的 USB接收完毕标志
检测到端点3接收完成后由硬件置位。微控制器可以读/写清除这一位。
软件写"1"时清除。只有 DCON.EP3DIR 置位时，这一位无效。

Bit0： UTXD3-- 端点3的 USB发送完毕标志
检测到端点3发送完成后由硬件置位。微控制器可以读/写清除这一位。
软件写"1"时清除。只有 DCON.EP3DIR 置"0"时，这一位无效。

EPINDEX (端点索引寄存器，地址=F1H，系统/USB复位值=xxxx-x000，读/写)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	EPINX1	EPINX0

Bit7~2： 保留，总是写 "0"

Bit1~0： EPINX[1：0]-- 端点索引位 [1：0]
2'b00：启用端点 0
2'b01：启用端点 1
2'b10：启用端点 2
2'b11：启用端点 3

EPCON (端点控制寄存器，端点索引，地址=E1H，系统/USB复位值=0000-0000,读/写)

7	6	5	4	3	2	1	0
RXSTL	TXSTL	RXDBM	TXDBM	RXISO	RXEPEN	TXISO	TXEPEN

Bit7：RXSTL-- 接收端点禁止.
置位以禁止接收端点

Bit6：TXSTL-- 发送端点禁止.
置位以禁止发送端点

Bit5：RXDBM-- 接收端点双缓冲模式.
置位将使能输出（OUT）事件的双缓冲传输。默认为"0"。只对端点3接收模式有效。

Bit4： TXDBM-- 发送端点双缓冲模式.
置位将使能输入（IN）事件的双缓冲传输。默认为"0"。只对端点2有效。

Bit3： RXISO-- 接收同步模式使能.
置位将配置端点为同步输出传输模式。禁止则端点为批量/中断输出传输模式。默认为"0"。只对端点3接收模式有效。

Bit2： RXEPEN-- 接收端点使能.
置位将使能接收端点。禁止将不会对有效的输出（OUT）或设置（SETUP）令牌做出反应。复位后，此位指向端点0并被使能。

Bit1： TXISO-- 发送同步方式使能.
置位将配置端点为同步输入传输模式。禁止则端点为批量/中断输入传输模式。默认为"0"。只对端点2有效。

Bit0： TXEPEN-- 发送端点使能.
置位将使能发送端点。禁止将不会对有效的输入（IN）令牌做出反应。复位后，此位指向端点0并被使能。

RXSTAT (端点接收状态寄存器， 端点索引， 地址=E2H， 系统/USB复位值=0000-0xxx，读/写)

7	6	5	4	3	2	1	0
RXSEQ	RXSETUP	STOVW	EDOVW	RXSOVW	ISOOVW	-	-

Bit7： RXSEQ-- 接收端点序列位(读，有条件的写).
当一个输出（OUT）令牌的确认（ACK）握手应答完成时，这一位将被转换。如果新的 RXSEQ 值的写入时， RXSOVW 位是置位的，那么 RXSEQ 位是可以写入的。

Bit6： RXSETUP-- 收到设置（SETUP）事件.
当收到有效的设置（SETUP）事件后由硬件置位。当检测到设置（SETUP）事件或者软件准备好处理控制传输的数据/状态步骤时清除此位。

Bit5： STOVW-- 开始重写标志 (只读).
当收到一个控制端点的设置（SETUP）令牌后由硬件置位。用来表示接收 FIFO 正在被新的 SETUP 数据重写。只对控制端点有用。

Bit4： EDOVW-- 结束重写标志.
这位在设置（SETUP）事件的握手阶段由硬件置位。在软件读 FIFO 数据被清除。只对控制端点有用。

Bit3： RXSOVW-- 接收数据序列重写位.
置位以允许 RXSEQ 的值被改写。写"0"对 RXSEQ 没影响。读取的结果一直为"0"。

Bit2： ISOOVW-- 同步接收数据重写位.
当微控制器读取剩余数据，同时USB主机（host）传送下一个数据而产生 FIFO 访问冲突时，由硬件置位。软件可以使用这位来确定数据是否被重写。置位时，软件应该写"0"来清除这位。

Bit1~0： 保留.

RXDAT (接收FIFO数据寄存器，端点索引，地址=E3H，系统/USB复位值=xxxx-xxxx，只读)

7	6	5	4	3	2	1	0
RXD7	RXD6	RXD5	RXD4	RXD3	RXD2	RXD1	RXD0

Bit7~0： RXD[7：0]-- 接收FIFO数据.

由 EPINDEX 指定的接收 FIFO 的数据，在这个寄存器中储存和读取。

RXCON (接收FIFO控制寄存器，端点索引，地址=E4H，系统/USB复位值=0xxx-0xxx，只写)

7	6	5	4	3	2	1	0
RXCLR	-	-	RXFFRC	-	-	-	-

Bit7： RXCLR--接收 FIFO 清空.
置位将清除整个 FIFO。所有的 FIFO 回到复位状态。当清除操作结束后硬件将清除此位。

Bit6~5： 保留.

Bit4： RXFFRC-- 接收 FIFO 读完成.
当读完数据串后置位以释放接收 FIFO。FIFO 释放操作完毕后硬件将清除此位。

Bit3~0： 保留.

RXCNT (接收 FIFO 字节数寄存器，端点索引，地址=E6H，系统/USB复位=0000-0000，只读)

7	6	5	4	3	2	1	0
-	RXBC6	RXBC5	RXBC4	RXBC3	RXBC2	RXBC1	RXBC0

Bit6~0： RXBC[6：0]--接收字节计数.
储存由 EPINDEX 指定的接收 FIFO 里收到的数据的字节数.

TXSTAT (端点传输状态寄存器，端点索引，地址=F2H，系统/USB复位值=0xxx-0xxx，读/写)

7	6	5	4	3	2	1	0
TXSEQ	-	-	-	TXSOVW	-	-	-

Bit7： TXSEQ-- 传输端点序列位 (读，有条件的写).

这位将在下个标识域（PID）被传送，并且通过输入（IN）事件的有效确认（ACK）应答握手翻转。如果新的 TXSEQ 值的写入时，TXOVW 位是置位的，那么 TXSEQ 位是可以写入的。

Bit6~4： 保留.

Bit3： TXSOVW-- 发送数据序列重写位.
置位以允许 TXSEQ 位的值被改写，写"0"对 TXSEQ 没有效果。读这位结果一直为"0".

Bit2~0： 保留.

TXDAT (发送 FIFO 数据寄存器，端点索引，地址=F3H，系统/USB复位值=xxxx-xxxx，只写)

7	6	5	4	3	2	1	0
TXD7	TXD6	TXD5	TXD4	TXD3	TXD2	TXD1	TXD0

Bit7~0： TXD[7：0]-- 发送FIFO数据.
要被传送到由 EPINDEX 指定的 FIFO 的数据写入这个寄存器。

TXCON (发送 FIFO 控制寄存器，端点索引，地址=F4H，系统/USB复位值=0xxx-0xxx，只写)

7	6	5	4	3	2	1	0
TXCLR	-	-	TXFFRC	-	-	-	-

Bit7： TXCLR-- 发送 FIFO 清空.

置位以清除整个发送 FIFO。所有的 FIFO 回到复位状态。当清除操作结束后硬件将清除此位。

Bit6~5： 保留.

Bit4： TXFFRC—发送 FIFO 写入完成

数据写入完成后置位释放发送FIFO。当 FIFO 释放完毕后硬件将清除此位。只有当软件在写完 TXCNT 寄存器后才能写这一位。

Bit3~0： 保留.

TXCNT (发送 FIFO 字节数寄存器， 端点索引， 地址=F6H， 系统/USB复位值=xxxx-xxxx，只写)

7	6	5	4	3	2	1	0
-	TXBC6	TXBC5	TXBC4	TXBC3	TXBC2	TXBC1	TXBC0

Bit6~0： TXBC[6：0]-- 发送字节计数.

储存由 EPINDEX 指定的发送 FIFO 里的数据的字节数.

SIOCTL (串行I/O控制寄存器，地址=C2H，系统复位值=xxxx-xxxx，只读)

7	6	5	4	3	2	1	0
DPI	DMI	-	-	-	-	-	-

Bit7： DPI-- USB DP 口状态，只读.

读USB DP口状态.

Bit6： DMI-- USB DM 口状态，只读.

读USB DM口状态.

Bit5~0： 保留.

20. 在系统编程(ISP)

Flash程序储存器支持并行编程和串行在系统编程(ISP)。并行编程模式提供高速编程。ISP允许芯片在最终产品上通过软件控制再编程。这个功能非常实用使在应用现场改写应用程序成为可能。

在使用ISP功能之前，用户应该通过通用编程器配置一个ISP空间。关于在系统编程（ISP）的配置，请参照“ISP空配置”部分。

以下是有关在系统编程（ISP）操作的特殊寄存器

ISPCR： 地址=E7H, ISP控制寄存器

IFADRH： 地址=E3H, ISP Flash地址高字节寄存器

IFADRL： 地址=E4H, ISP Flash地址低字节寄存器

IFD： 地址=E2H, ISP Flash数据寄存器

SCMD： 地址=E6H, ISP命令序列寄存器.

ISPCR (地址=E7H, ISP控制寄存器)

7	6	5	4	3	2	1	0
ISPEN	SWBS	SWRST	Reserved	MISPF	Reserved	MS1	MS0

注意： 复位值为 #00000000B.

Bit7： ISPEN： 置位将使能ISP功能.

Bit6： SWBS：
软件引导选择。软件复位后，若置位则选择从ISP空间启动，清零则选择从应用程序空间启动.

Bit5： SWRST： 置位触发软件复位.

Bit4： 保留

Bit3： MISPF, 笙泉（Megawin）专有ISP标志。如果在应用程序区域通过USB DFU使用Megawin专用的ISP代码, cpu必须同时置位 SWRST 和 MISPF 来触发 ISP 程序。如果用户只进行软件复位或者执行 IAP 程序，必须写这位"0".

Bit2： 保留.

Bit1~0： MS1~MS0, ISP模式选择, 如下所示

MS1	MS0	ISP 模式
0	0	等待
0	1	读
1	0	字节编程
1	1	页擦除

20.1. ISP操作描述

在ISP操作之前，用户应该先设置 CKCON 寄存器的 XCKS4~XCKS0 位适当的值。
(参照"系统时钟"部分.)

页擦除 (每页64字节)

步骤1： 置 ISPCR 寄存器中的[MS1,MS0]=[1,1] 选择页擦除模式.
步骤2： 将页地址写入 IFADRH 和 IFADRL 寄存器.
步骤3： 顺序写0x46然后写0xB9到 SCMD 寄存器以触发ISP程序.

字节编程

步骤1： 置 ISPCR 寄存器中的[MS1,MS0]=[1,0] 选择字节编程模式.
步骤2： 将字节地址写入 IFADRH 和 IFADRL 寄存器.
步骤3： 在 IFD 寄存器写入要编程的数据.
步骤4： 顺序写0x46然后写0xB9到 SCMD 寄存器以触发ISP程序.

读

步骤1： 置 ISPCR 寄存器中的[MS1,MS0]=[0,1] 选择读模式.
步骤2： 将字节地址写入 IFADRH 和 IFADRL 寄存器.
步骤3： 顺序写0x46然后写0xB9到 SCMD 寄存器以触发ISP程序.
步骤4： 现在，Flash数据存于 IFD 寄存器.

20.2. ISP示例代码

```
;*************************************************************************************************
; ISP 演示程序
;*************************************************************************************************
IFD             DATA    0E2h
IFADRH          DATA    0E3h
IFADRL          DATA    0E4h
ISPTME          DATA    0E5h
SCMD            DATA    0E6h
ISPCR           DATA    0E7h
;
                MOV     ISPCR,#10000000b        ;ISPCR.7=1, 使能 ISP
;=========================================================================
; 1. 页擦除模式 (64 字节每页)
;=========================================================================
                ORL     ISPCR,#03h              ;[MS1,MS0]=[1,1], 选择页擦除模式
                MOV     IFADRH,??               ;在 IFADRH 和 IFADRL 写入页地址
                MOV     IFADRL,??               ;
                MOV     SCMD,#46h               ;触发 ISP 过程
                MOV     SCMD,#0b9h              ;
                ; 处理中...                     (CPU 将暂停于此 直到处理完毕)

;=========================================================================
; 2. 字节编程模式
;=========================================================================
                ORL     ISPCR,#02h              ;[MS1,MS0]=[1,0], 选择字节编程模式
                ANL     ISPCR,#0FEh             ;
                MOV     IFADRH,??               ;在 IFADRH 和 IFADRL 写入字节地址
                MOV     IFADRL,??               ;
                MOV     IFD,??                  ;将要编程的数据写入 IFD
                MOV     SCMD,#46h               ;触发 ISP 过程
                MOV     SCMD,#0b9h              ;
                ;处理中...                      (CPU 将暂停于此 直到处理完毕)

;=========================================================================
; 3. 使用读模式进行比较
;=========================================================================
                ANL     ISPCR,#0FDh ;[MS1,MS0]=[0,1], 选择字节读模式
                ORL     ISPCR,#01h              ;
                MOV     IFADRH,??               ;在 IFADRH 和 IFADRL 写入字节地址
                MOV     IFADRL,??               ;
                MOV     SCMD,#46h               ;触发 ISP 过程
                MOV     SCMD,#0b9h              ;
                ;处理中...                      (CPU 将暂停于此 直到处理完毕)
                MOV     A,IFD                   ;数据在 IFD 中
                CJNE    A,wanted,ISP_error      ;和期望的数值进行比较
...
ISP_error:
...
;
```

21. 在应用编程 (IAP)

这枚芯片可以在应用编程（IAP），可以实现在程序运行过程中将一些 FLASH 空间中用于储存非易失性数据。这个有用的特性可以被用于那些数据需要在断电后保存的应用中。因此不必使用外部串行 EEPROM(如93c46,24c01,等)来储存这些非易失性数据。

21.1. IAP储存区范围

实际上，除了编程的储存器的范围不一样外，IAP的操作和ISP一样，ISP操作的储存器范围为应用程序储存器，而IAP操作的范围是配置好的IAP储存区。

在使用IAP功能之前，用户应该先使用通用编程器通过设置**OR1** (IAPLB) 来配置一个IAP储存区。IAP存储区的范围通过以下的公式，由IAPLB和ISP的开始地址确定。

IAP 下边界 = IAPLBx256,

IAP 上边界 = ISP 开始地址 –1.

例如，如果IAPLB=0x12，ISP开始地址=0x3800，IAP区的范围为0x1200~0x37FF。

请参照OR1(IAPLB)的“熔丝配置”。

21.2. 更新IAP储存区的数据

因为IAP储存区是FLASH储存区的一部分，而FLASH擦除中只能页擦除而不是字节擦除。所以更新IAP储存区中的“一个字节”，用户不能直接写入数据到那个字节。
以下为正确步骤：

1) 保存包含要更新的数据的整个页(512字节)到缓冲区。
2) 擦除这个页。***(使用 ISP页擦除模式)***
3) 更新缓冲区中所需更新的字节。
4) 将缓冲区中更新后的数据写入到原来的页中。***(使用 ISP字节编程模式)***

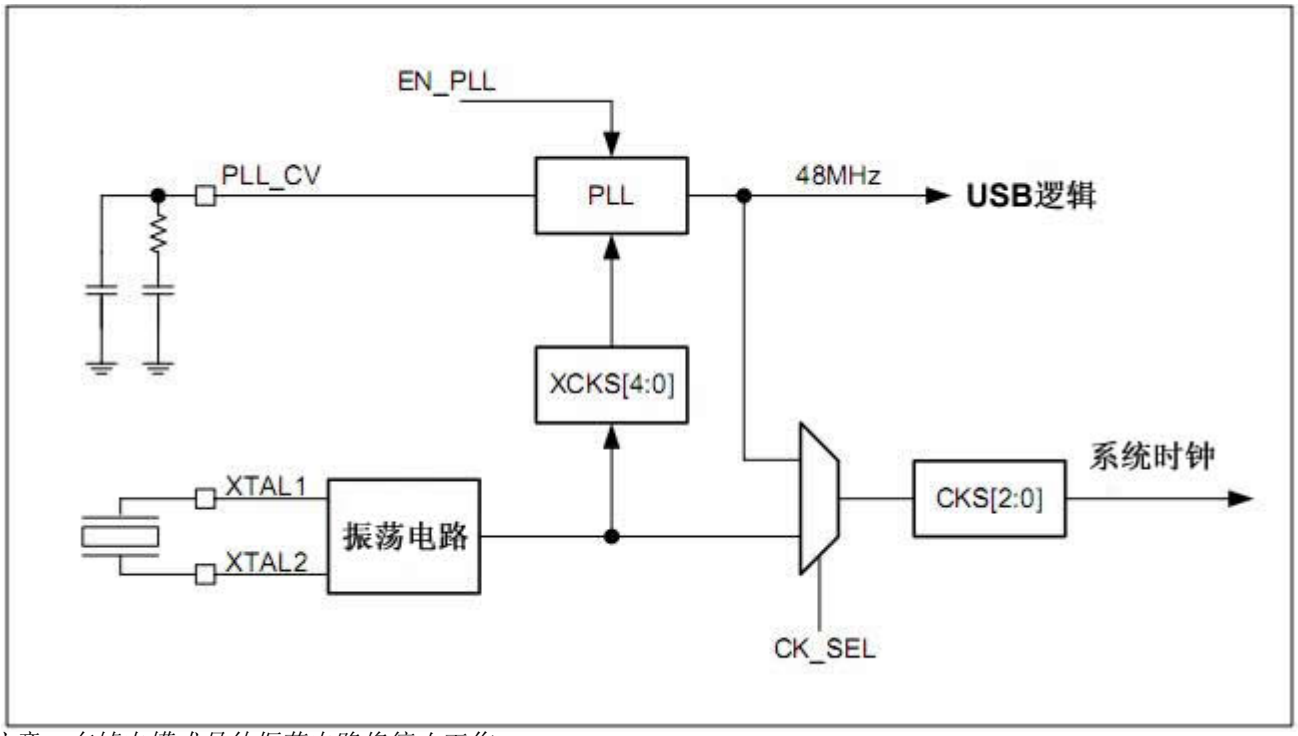
要读IAP储存区中的数据，用户可以使用“MOVC A,@A+DPTR”指令或者***ISP读模式。***

22. 系统时钟

22.1. 可编程系统时钟

系统时钟(或CPU时钟)可编程且来源可选。用户可以通过设置 CKS2~CKS0(CKCON.2 ~ CKCON.0) 来编程系统时钟频率，设置 CK_SEL (CKCON2.0) 来选择时钟来源。系统时钟方框图如下所示。

系统时钟方框图



注意：在掉电模式晶体振荡电路将停止工作。

CKCON (地址=C7H, 时钟控制寄存器)

7	6	5	4	3	2	1	0
XCKS4	XCKS3	XCKS2	XCKS1	XCKS0	CKS2	CKS1	CKS0

XCKS4~XCKS0 (或 XCKS[4：0])： 根据晶体频率写入适当的值，如下表所列：

[XCKS4~XCKS0] = OSCin – 1， 其中*OSCin=1~32 (MHz)*。

举例来说,
(1) 如果晶体频率为12MHz, 将11即01011B写入[XCKS4~XCKS0] 。
(2) 如果晶体频率为6MHz,将5即00101B写入[XCKS4~XCKS0]。

CKS2~CKS0： 系统时钟分频器选择位

CKS2	CKS1	CKS0	Fosc (系统时钟)
0	0	0	*晶体频率 默认*
0	0	1	晶体频率 /2
0	1	0	晶体频率 /4
0	1	1	晶体频率 /8
1	0	0	晶体频率 /16
1	0	1	晶体频率 /32
1	1	0	晶体频率 /64
1	1	1	晶体频率 /128

CKCON2 (地址=BFH, 时钟控制寄存器2)

7	6	5	4	3	2	1	0
-	-	OSCDR0	-	EN_USB	EN_PLL	PLL_RDY	CK_SEL

OSCDR0： 片上晶体振荡器驱动控制位.
0选择最大驱动，1选择最小驱动。

EN_PLL： PLL使能位。置位使能；清零禁止。

PLL_RDY： 用于指示PLL状态是否准备好，只读。

CK_SEL： 系统时钟驱动输入选择位。置位，时钟=48MHz；清零，时钟=晶体频率。

在默认状态下，CKCON的复位值是0x00 (CK_SEL=0, CKS2/1/0=000B)，所以系统时钟就是XTAL1引脚的信号。用户可以随时修改 CKCON 的值以获得新的系统时钟，并且修改结束后立即生效。

在不关心系统时钟频率的应用中，用户可以向 CKS2~CKS0 位写入一个非零值以减慢进入空闲模式时的系统时钟，达到减少功耗的目的。

22.2. 片内晶体振荡器驱动控制

为了减小晶体振荡电路产生的电源消耗，设计了一种智能的驱动控制机构。CKCON2的 OSCDR0 控制位用于控制此机构。上电时，这位被设为0选择最大驱动以启动晶体振荡，当MCU顺利运行后，用户可以编辑此位以保持晶体振荡稳定，参照下表.

OSCDR0	XTAL范围
0	高达32MHz(TBD)
1	低至1MHz(TBD)

23. 上电复位

直到VCC电源达到上电复位 (POR)电压之前，CPU不会工作。这意味着当VCC电压低于上电复位电压时就会进入上电复位。

有效的上电复位信号将上电标志位***(POF)***置“1”，有两种情况使得上电复位信号有效：

(1) 电压上升中,如冷启动
(2) VCC电压低于上电复位电压

它帮助用户区分CPU是冷启动(上电)还是热启动，如RST引脚复位，软件复位(ISPCR.5)，看门狗定时器复位。上电标志位***(POF)***应该被软件清除。

PCON (地址=87H, 电源控制寄存器)

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

POF： 上电标志.
在上电期间(冷启动)或者VCC电压低于上电复位(POR)电压时， POF 由硬件置"1"。它可以软件清除或者置位，并且不受任何热启动影响，如RST引脚复位，软件复位(ISPCR.5)，看门狗定时器复位。注意这一标志位应该软件清除。

注意：如果使用Megawin专用ISP代码，上电标志将在MCU上电过程中被ISP代码清"0"。并且应用程序必须在ISP过程中一直写这位"0"。Megawin发送的MG84FL54B样品已经默认加入ISP代码，如果用户无需ISP代码或者要加入自己的ISP代码，可以使用烧写工具来擦除和重新编程为用户的设置。

24. 熔丝位选项

熔丝位具有多种硬件选项，只能通过通用编程器修改。

OR0 (选项寄存器0)

7	6	5	4	3	2	1	0
ISP_S2	ISP_S1	ISP_S0	-	HWBS	HWBS2	SB	LOCK

HWBS：
0 (使能)：上电时，如果配置了ISP空间，则MCU从ISP空间启动。
1 (禁止)： MCU总是从应用程序空间启动。

HWBS2：
0 (使能)：除了上电过程外，如果配置了ISP空间，则RST引脚的复位也将强迫MCU从ISP空间启动。
1 (禁止)：启动位置由 **HWBS** 决定。

SB：
0 (使能)：安全设定，加密代码，通用编程器或烧录器禁止读出。
1 (禁止)：代码不加密。

LOCK：
0 (使能)：安全设定，代码锁定。
1 (禁止)：不锁定。

{ISP_S2, ISP_S1, ISP_S0}： 见下表

ISP_S2, ISP_S1, ISP_S0	ISP-空间大小	ISP 开始地址
0, 0, 0	4K bytes	0x3000
0, 0, 1	3.5K bytes	0x3200
0, 1, 0	3K bytes	0x3400
0, 1, 1	2.5K bytes	0x3600
1, 0, 0	2K bytes	0x3800
1, 0, 1	1.5K bytes	0x3A00
1, 1, 0	1K bytes	0x3C00
1, 1, 1	(没有配置ISP空间)	-

OR1 (选项寄存器1)

7	6	5	4	3	2	1	0
IAPLB							

IAPLB用于确定IAP空间的下边界，因为一个FLASH页有**512**字节，所以IAPLB必须是一个偶数。

IAP存储区的范围通过以下的公式，由IAPLB和ISP的开始地址确定。

IAP 下边界 = IAPLBx256

IAP 上边界 = ISP 开始地址 –1

例如，如果IAPLB=0x12，ISP开始地址=0x3800，IAP区的范围为0x1200~0x37FF。

OR2 (选项寄存器2)

7	6	5	4	3	2	1	0
-	-	-	PSMEN	-	-	-	-

PSMEN：
0 (使能)： 使能节电模式。
1 (禁止)： 禁止节电模式。

OR3 (选项寄存器3)

7	6	5	4	3	2	1	0
WDTCR_WP	-	HWENW	-	HWWIDL	HWPS2	HWPS1	HWPS0

WDTCR_WP：
0 (使能)：
如果CPU运行在应用程序空间， WDTCR 寄存器将被写保护，除 CLRW 位之外。
如果CPU运行在ISP空间， WDTCR 寄存器将被写保护，除 CLRW，PS2，PS1，PS0 位之外。
1 (禁止)： WDTCR 寄存器可以自由写入。

HWENW： (带有变量 HWWIDL 和 HWPS[2：0])：
0 (使能)： 上电时硬件自动使能看门狗定时器
也就是说：
在 WDTCR 寄存器, 硬件自动
(1) 置位 ENW,
(2) 载入 HWWIDL 到 WIDL 位,并且
(3) 载入 HWPS[2：0] 到 PS[2：0] 位.

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

WDTCR (看门狗定时器控制寄存器)

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

载入 HWENW　载入 HWWIDL　载入 HWPS[2:0]

1 (禁止)： 上电时看门狗定时器无活动。

25. 指令集

除了执行时间，也就是说执行一条指令的时钟周期以外，MG84FL54B指令集与标准8051指令集完全兼容。最短的执行时间为1个系统时钟周期(1/Fosc)，最长执行时间为6个系统时钟周期(6/Fosc)

介绍指令集之前，用户需注意：

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

25.1. 算术运算指令

助记符	描述	字节	执行时钟周期
算术运算指令			
ADD A,Rn	将寄存器Rn中的内容加到累加器中	1	2
ADD A,direct	直接地址单元中的内容加到累加器中	2	3
ADD A,@Ri	寄存器工作寄存器Ri指向的地址单元中的内容加到累加器中	1	3
ADD A,#data	立即数加到累加器中	2	2
ADDC A,Rn	累加器与工作寄存器Rn中的内容、连同进位位相加，结果存在累加器中	1	2
ADDC A,direct	累加器与直接地址单元的内容、连同进位位相加，结果存在累加器中	2	3
ADDC A,@Ri	累加器与工作寄存器Ri指向的地址单元中的内容、连同进位位相加，结果存在累加器中	1	3
ADDC A,#data	累加器与立即数、连同进位位相加，结果存在累加器中	2	2
SUBB A,Rn	累加器与工作寄存器中的内容、连同借位位相减，结果存在累加器中	1	2
SUBB A,direct	累加器与直接地址单元中的内容、连同借位位相减，结果存在累加器中	2	3
SUBB A,@Ri	累加器与工作寄存器Ri指向的地址单元中内容、连同借位位相减，结果存在累加器中	1	3
SUBB A,#data	累加器与立即数、连同借位位相减，结果存在累加器中	2	2
INC A	累加器中的内容加1	1	2
INC Rn	寄存器Rn的内容加1	1	3
INC direct	直接地址单元中的内容加1	2	4
INC @Ri	工作寄存器Ri指向的地址单元中的内容加1	1	4
INC DPTR	数据指针DPTR的内容加1	1	1
DEC A	累加器中的内容减1	1	2
DEC Rn	寄存器Rn中的内容减1	1	3
DEC direct	直接地址单元中的内容减1	2	4
DEC @Ri	工作寄存器Ri指向的地址单元中的内容减1	1	4
MUL AB	ACC中内容与寄存器B中内容相乘，其结果低位存在ACC中、高位存在寄存器B中	1	4
DIV AB	ACC中内容除以寄存器B中内容，商存在ACC，而余数存在寄存器B中	1	5
DA A	ACC十进制调整	1	4

25.2. 逻辑操作指令

助记符	描述	字节	执行时钟周期
逻辑操作指令			
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

25.3. 数据传送指令

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

注1： 当控制位EXTRAM=1, 所有 “MOVX” 指令均指向外部数据存储区。

当控制位EXTRAM=0,所有“MOVX”指令均指向片内扩展XRAM区(如果地址= 0x0000~0x0FFF)和USB特殊功能寄存器(如果地址= 0xFFC0~0xFFFF).

注2： 访问外部数据存储区的机器周期时间计算方法是：

7 + 2 x (ALE延长时钟) + (RW延长时钟)

25.4. 布尔操作指令

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

25.5. 控制和转移指令

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

26. 极限参数

参数	范围	单位
工作温度	-55 ~ + 125	℃
存放温度	-65 ~ + 150	℃
任何GPIO引脚或者RST引脚对地电压	-0.3 ~ VDD_IO + 0.3	V
DP，DM 和 PLL_CV引脚对地电压	-0.3 ~ VDDA，VDD_PLL + 0.3	V
VDD_IO引脚对地电压	-0.5 ~ + 6.2	V
VDDA，VDD_PLL，VDD_CORE对地电压	-0.3 ~ +4.2	V
最大VDD到地总电流	400	mA
任何引脚能够吸收的最大电流	40	mA

*注意：器件超过“极限参数范围”可能导致永久性损坏。工作或者存储超出这个范围是不推荐的，且可能影响器件的可靠性。

27. 电气特性

27.1. 整体直流电气特性

除非另外说明VSS = 0V, TA = 25 ℃, VDD_IO= 5.0V, VDD_CORE= VDDA= VDD_PLL= 3.3V,

符号I	参数	测试条件	极限			单位
			min	典型	max	
V_{IH1}	输入高电压,P0,P1,P2 和P3	VDD_IO= 5.0V	2.0			V
V_{IH2}	输入高电压,RESET引脚	VDD_IO= 5.0V	3.5			V
V_{IL}	输入低电压	VDD_IO= 5.0V			0.8	V
I_{OL}	输出低电流	V_{PIN} = 0.45V	12	20		mA
I_{OH1}	输出高电流(推挽)	V_{PIN} = 2.4V	12	20		mA
I_{OH2}	输出高电流(准双向)	V_{PIN} = 2.4V		220		uA
I_{IL1}	逻辑0输入电流(准双向)	V_{PIN} = 0.45V		17	50	uA
I_{IL2}	逻辑0输入电流(仅输入)	V_{PIN} = 0.45V		0	10	uA
I_{LK}	输入漏电流(开漏输出)	V_{PIN} = VDD_IO		0	10	uA
I_{H2L}	逻辑1到0转变电流	V_{PIN} = 1.8V		230	500	uA
I_{OP}	工作电流	F_{OSC} = 12MHz		12	18	mA
I_{IDLE}	空闲模式电流	F_{OSC} = 12MHz		6	9	mA
I_{PD}	掉电电流	VDD_IO= 5.0V		0.1	10	uA
R_{RST}	外部复位下拉电阻	VDD_IO= 5.0V	100		150	Kohm

除非另外说明VSS = 0V, TA = 25 ℃, VDD_IO= VDD_CORE= VDDA= VDD_PLL= 3.3V,

符号	参数	测试条件	极限			单位
			min	典型	max	
V_{IH1}	输入高电压 for P1 and P3	VDD_IO= 3.3V	2.0			V
V_{IH2}	输入高电压,RESET引脚	VDD_IO= 3.3V	2.8			V
V_{IL}	输入低电压	VDD_IO= 3.3V			0.8	V
I_{OL}	输出低电流	V_{PIN} = 0.45V	8	14		mA
I_{OH1}	输出高电流(推挽)	V_{PIN} = 2.4V	4	8		mA
I_{OH2}	输出高电流(准双向)	V_{PIN} = 2.4V		64		uA
I_{IL1}	逻辑0输入电流(准双向)	V_{PIN} = 0.45V		7	50	uA
I_{IL2}	逻辑0输入电流(仅输入)	V_{PIN} = 0.45V		0	10	uA
I_{LK}	输入漏电流(开漏输出)	V_{PIN} = V_{CC}		0	10	uA
I_{H2L}	逻辑1到0转变电流(P1,3)	V_{PIN} = 1.4V		100	600	uA
I_{OP}	工作电流	F_{OSC} = 12MHz		9	14.5	mA
I_{IDLE}	空闲模式电流	F_{OSC} = 12MHz		3.5	5.3	mA
I_{PD}	掉电电流	VDD_IO= 3.3V		0.1	10	uA
R_{RST}	外部复位下拉电阻	VDD_IO= 3.3V	160		240	Kohm

27.2. USB收发器电气特性

除非另外说明VSS = 0V, TA = 25 ℃, VDD_IO= 2.4V~5.5V, VDD_CORE= VDDA= VDD_PLL= 3.3V

符号	参数	测试条件	极限			单位
			min	典型	max	
发送器						
V_{OH}	输出高电压		2.8			V
V_{OL}	输出低电压				0.8	V
V_{CRS}	输出交叉点		1.3		2.0	V
Z_{DRVH}	高电平输出阻抗		28		44	ohm
Z_{DRVL}	低电平输出阻抗		28		44	ohm
R_{PU}	上拉电阻	On DP	1.425	1.5	1.575	Kohm
T_R	输出上升时间		4		20	ns
T_F	输出下降时间		4		20	ns
接收器						
V_{DI}	差分输入灵敏度	| DP – DM |	0.2			V
V_{CM}	差分输入共模范围		0.8		2.5	V
I_L	输入漏电流	上拉禁止		<1.0		uA

28. 应用领域

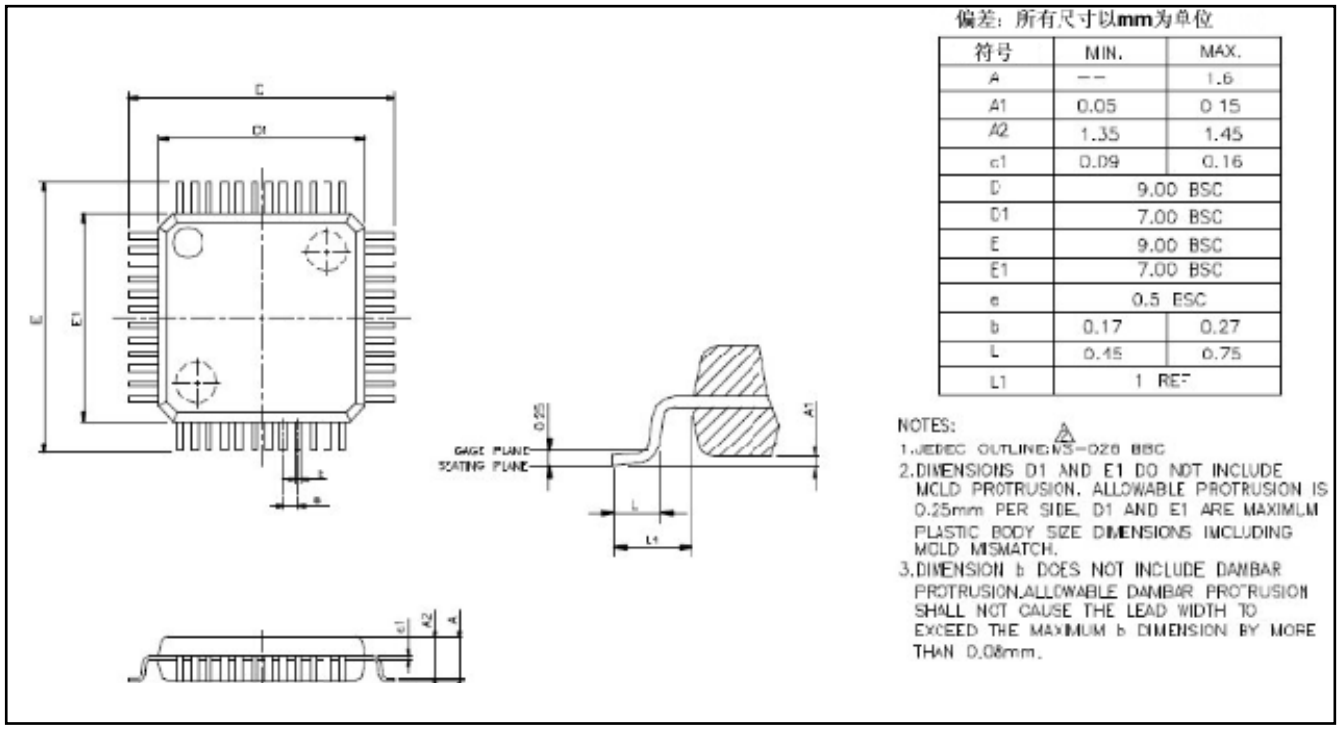
- ⌘ 家电
- ⌘ 保健
- ⌘ POS 控制
- ⌘ 无线适配器
- ⌘ 游戏操纵杆
- ⌘ 无线键盘鼠标

29. 订货信息

零件编号	温度范围	封装	包装	工作电压
MG84FL54BD	**-40℃~85℃**	**LQFP-48**	**托盘**	**3.3V**

30. 封装尺寸

MG84FL54BD (LQFP-48)



31. 版本历史

版本	日期	页数	描述
V0.96	2007/11		- 初次发布数据手册.
V0.97	2007/12	P95,96	- 增加极限参数和电气特性.
V0.98	2008/01	P7 P9,10 P91 P95,96	- 修改闪存数据保存时间 7年到100年. - 增加P10的T2CKO. - 修改直流电气特性表R_{RST} max/min 值. - 定稿电气特性.
A1	2008/06		- 初始版本
A2	2008/12		- 规格化文挡

译后感：

由于英文版没有对整个USB模块进行概览性的介绍，翻译时对寄存器功能进行理解就变得比较困难，很难在头脑中建立起一个相对完整、容易理解的模型。原文只是给出了“USB结构图”和“USB FIFO管理图”两个框图。这两个框图除了图注文字外，没有给出相关的说明性文字。而接下来USB部分所有的篇幅都在说明各种寄存器。由于对USB部分不熟悉，USB部分寄存器说明的翻译，可能用词不够确切。

建议：

厂家在下次修订手册的时候，增加对整个USB模块进行概览性的介绍，对一些USB协议中出现的缩写词，如SOF,ACK,STALL等加注解释，让我们更容易学习和理解。

lukeunderwood 潜水的熊猫

2009年3月