SONiX Technology Co., Ltd.

SN8F5702 Series

Datasheet

8051-based Microcontroller

SN8F5702
SN8F570200
SN8F570202
SN8F570210
SN8F570211
SN8F570212
SN8F570213

1 简介

1.1 功能特性

- ◆ **增强型 8051 微控制器：减少指令周期时间（高达 80C51 的 12 倍）**
  - 高达 32MHz 灵活的 CPU 频率
  - 内部 32MHz 时钟发生器（IHRC）
- ◆ **4 KB 非易失性 Flash 存储器（IROM），支持在线编程功能**
- ◆ **256 字节内部 RAM（IRAM）**
- ◆ **13 个中断源：可控制中断的优先等级以及独立的中断向量 12 个内部中断**
  - 1 个外部中断：INT0
  - 1 组 DPTR
  - 2 组 8/16 位定时器，有四种工作模式。
  - 1 组 16 位定时器，带有 4 路比较输出（PWM）和捕获通道。
- ◆ **1 组 8/16 位 PWM 发生器**
  - 每组 PWM 都有 4 个输出通道，带有反相器和死区控制功能
- ◆ **12 位 SAR ADC，包括 10 个外部通道和 2 个内部通道，以及 4 个内部参考电压**
- ◆ **SPI/UART 接口，支持 SMBus 的 I2C 接口**
- ◆ **片上调试**
  - 单线调试接口
  - 2 个硬件断点
  - 无限制软件断点
  - ROM 数据安全保护
  - 看门狗和可编程的外部复位
  - 1.8/2.4/3.3V 低电压检测
  - 工作电压范围大（1.8 V – 5.5 V），温度范围为-40 °C 到 85 °C

1.2 应用领域

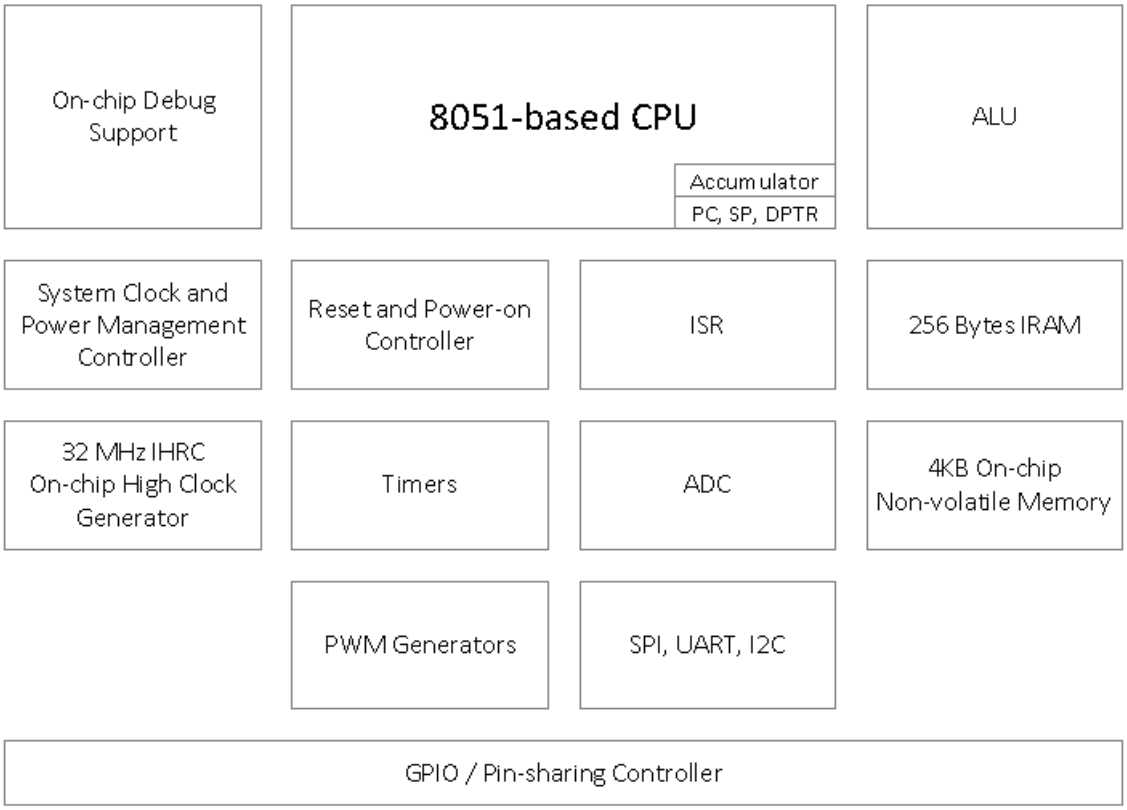
- 无刷直流电机
- 家用自动化产品
- 家电
- 其它

1.3 产品性能表

	I/O	PWM Channels	I2C	SPI	UART	ADC ext. Channels	OPA	CMP	Ext. INT	Package Types
SN8F5702	18	8	V	V	V	10	-	-	1	DIP20, SOP20, TSSOP20, QFN20
SN8F570212	14	6	V	V	TX*(1)	8	-	-	1	SOP16,SSOP16,QFN16
SN8F570210	12	5	-	-	V	6	-	-	1	SOP14
SN8F570211	12	5	V	-	TX*(1)	6	-	-	1	SOP14
SN8F570213	12	5	-	-	V	6	-	-	1	SOP14
SN8F570200	8	3	-	-	TX*(1)	5	-	-	1	MSOP10
SN8F570202	6	3	-	-	TX*(1)	4	-	-	1	SOP8

*(1)只支持 UART TX 模式。

1.4结构框图



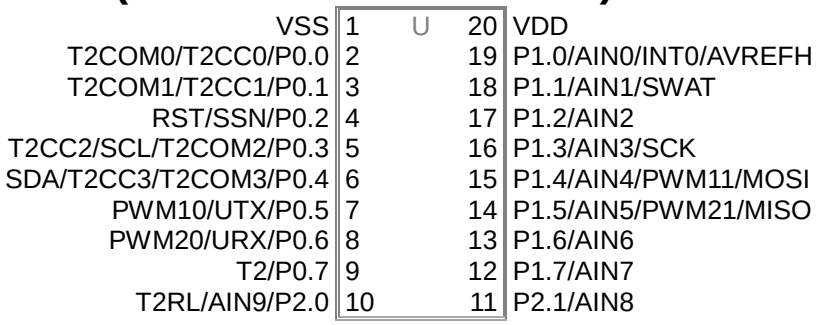
2 目录

3 修订记录

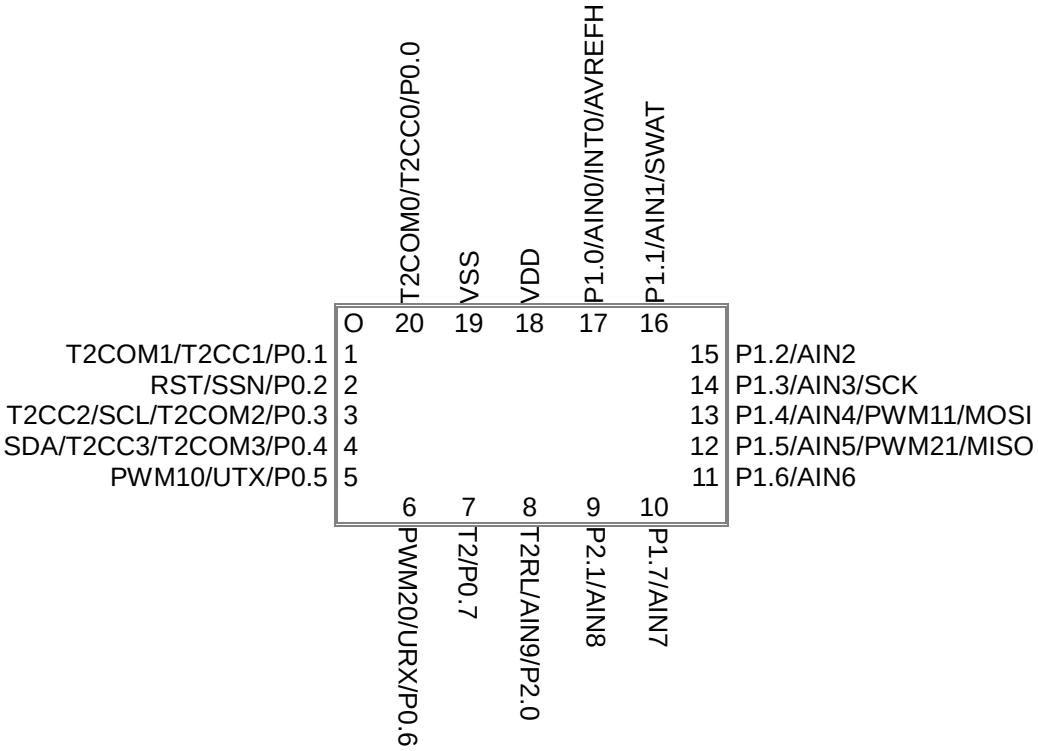
版本	时间	修订说明
1.0	Nov. 2015	初版。
1.4	Dec.2015	1、调整定时器章节和电气特性章节的内容。 2、增加程序存储器安全章节、特殊寄存器章节和 noise filter 章节。 3、调整调试界面的最低要求。 4、更改 SN8F57021 的名称为 SN8F570210。 5、更改 SN8F57022 的名称为 SN8F570200 并调整其引脚配置。 6、更改 SN8F57023 的名称为 SN8F570211 并调整其引脚配置。 7、更改 SN8F57024 的名称为 SN8F270212 并调整其引脚配置。
1.5	Dec. 2015	1、调整电气特性 IHRC 的相关内容。 2、在 UART、SPI 和 I2C 章节中添加省电内容。
1.6	Jun.2016	1、增加 T2 捕捉功能波形图。 2、在特殊功能寄存器章节中添加寄存器宣告章节。 3、增加附录：参考文档章节。 4、增加 ROM 烧录引脚章节。
1.7	Aug.2016	增加 SN8F570202 的相关信息。
1.9	Mar.2017	1、调整产品性能表的部分内容。 2、增加 QFN16 的封装。 3、在看门狗复位章节中增加 WDT 的说明。 4、增加 UART 波特率列表。 5、调整电气特性的部分内容。

4 引脚配置

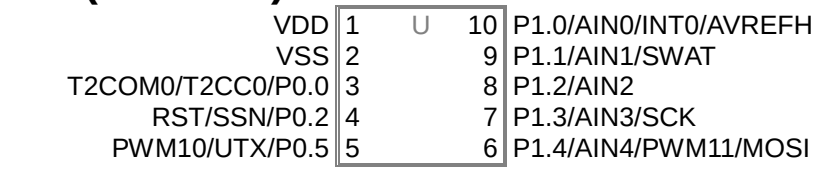
4.1 SN8F5702P/S/T (DIP20/SOP20/TSSOP20)



4.2 SN8F5702J (QFN20)



4.3 SN8F570200A (MSOP10)



4.4 SN8F570210S (SOP14)

VSS	1	U	14	VDD
T2COM0/T2CC0/P0.0	2		13	P1.0/AIN0/INT0/AVREFH
T2COM1/T2CC1/P0.1	3		12	P1.1/AIN1/SWAT
RST/SSN/P0.2	4		11	P1.2/AIN2
PWM10/UTX/P0.5	5		10	P1.3/AIN3/SCK
PWM20/URX/P0.6	6		9	P1.4/AIN4/PWM11/MOSI
T2/P0.7	7		8	P2.0/T2RL/AIN9

4.5 SN8F570211S (SOP14)

VSS	1	U	14	VDD
T2COM0/T2CC0/P0.0	2		13	P1.0/AIN0/INT0/AVREFH
RST/SSN/P0.2	3		12	P1.1/AIN1/SWAT
T2CC2/SCL/T2COM2/ P0.3	4		11	P1.2/AIN2
SDA/T2CC3/T2COM3/P0.4	5		10	P1.3/AIN3/SCK
PWM10/UTX/P0.5	6		9	P1.4/AIN4/PWM11/MOSI
T2/P0.7	7		8	P2.0/T2RL/AIN9

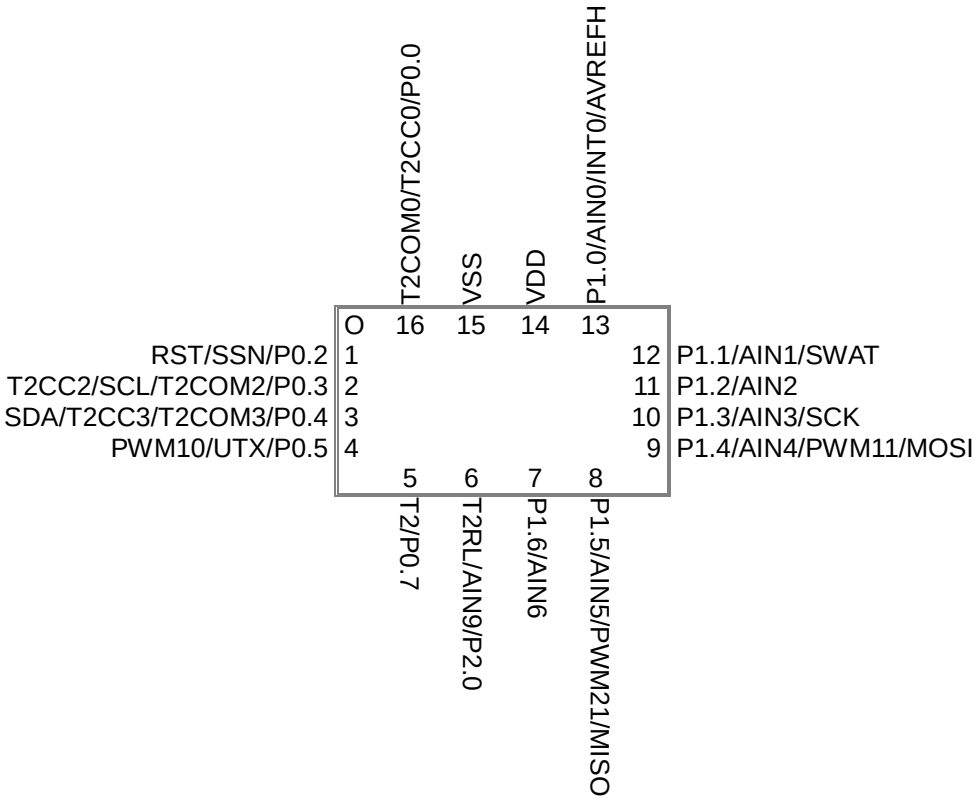
4.6 SN8F570213S (SOP14)

VDD	1	U	14	VSS
T2COM0/T2CC0/P0.0	2		13	P1.0/AIN0/INT0/AVREFH
RST/SSN/P0.2	3		12	P1.1/AIN1/SWAT
RST/SSN/P0.2	4		11	P1.2/AIN2
PWM10/UTX/P0.5	5		10	P1.3/AIN3/SCK
PWM20/URX/P0.6	6		9	P1.4/AIN4/PWM11/MOSI
T2/P0.7	7		8	P2.0/T2RL/AIN9

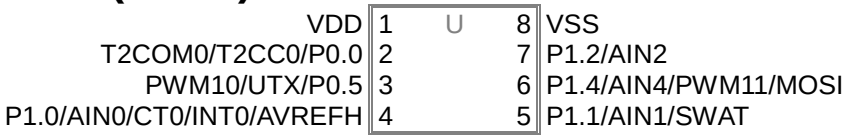
4.7 SN8F570212S/T (SOP16/TSSOP16)

VSS	1	U	16	VDD
T2COM0/T2CC0/P0.0	2		15	P1.0/AIN0/INT0/AVREFH
RST/SSN/P0.2	3		14	P1.1/AIN1/SWAT
T2CC2/SCL/T2COM2/ P0.3	4		13	P1.2/AIN2
SDA/T2CC3/T2COM3/P0.4	5		12	P1.3/AIN3/SCK
PWM10/UTX/P0.5	6		11	P1.4/AIN4/PWM11/MOSI
T2/P0.7	7		10	P1.5/AIN5/PWM21/MISO
T2RL/AIN9/P2.0	8		9	P1.6/AIN6

4.8 SN8F570212J (QFN16)



4.9 SN8F570202S (SOP8)



4.10 引脚说明

电源引脚

引脚名称	类型	功能说明
VDD	Power	电源输入引脚
VSS	Power	电源地（0V）

P0 端口

引脚名称	类型	功能说明
P0.0 T2COM0 T2CC0	Digital I/O Digital Output Digital Input	GPIO Timer2：比较器输出 Timer2：捕获输入
P0.1 T2COM1 T2CC1	Digital I/O Digital Output Digital Input	GPIO Timer2：比较器输出 Timer2：捕获输入
P0.2 Reset SSN	Digital I/O Digital Input Digital Input	GPIO 系统复位（低电平有效） SPI：从动选择引脚（从动模式）
P0.3 T2COM2 T2CC2 SCL	Digital I/O Digital Output Digital Input Digital I/O	GPIO Timer2：比较器输出 Timer2：捕获输入 I2C：时钟输出（主机模式）时钟输入（从机模式）
P0.4 T2COM3 T2CC3 SDA	Digital I/O Digital Output Digital Input Digital I/O	GPIO Timer2：比较器输出 Timer2：捕获输入 I2C：数据引脚
P0.5 UTX PWM10	Digital I/O Digital Output Digital Output	GPIO UART：发送引脚 PWM：可编程的 PWM 输出
P0.6 URX PWM20	Digital I/O Digital Input Digital Output	GPIO UART：接收引脚 PWM：可编程的 PWM 输出
P0.7 T2	Digital I/O Digital Input	GPIO Timer2：事件计数器输入引脚

P1 端口

引脚名称	类型	功能说明
P1.0 AIN0 INT0 AVREFH	Digital I/O Analog Input Digital Input Analog Input	GPIO ADC：输入通道 INT0：外部中断 ADC：外部高参考电压
P1.1 AIN1 SWAT	Digital I/O Analog Input Digital I/O	GPIO ADC：输入通道 调试界面
P1.2 AIN2	Digital I/O Analog Input	GPIO ADC：输入通道
P1.3 AIN3 SCK	Digital I/O Analog Input Digital Output	GPIO ADC：输入通道 SPI：时钟输出（主机模式）时钟输入（从机模式）
P1.4 AIN4 MOSI PWM11	Digital I/O Analog Input Digital I/O Digital Output	GPIO ADC：输入通道 SPI：发送引脚（主机模式）接收引脚（从机模式） PWM：可编程的 PWM 输出
P1.5 AIN5 MISO PWM21	Digital I/O Analog Input Digital I/O Digital Output	GPIO ADC：输入通道 SPI：接收引脚（主机模式）发送引脚（从机模式） PWM：可编程的 PWM 输出
P1.6 AIN6	Digital I/O Analog Input	GPIO ADC：输入通道
P1.7 AIN7	Digital I/O Analog Input	GPIO ADC：输入通道

P2 端口

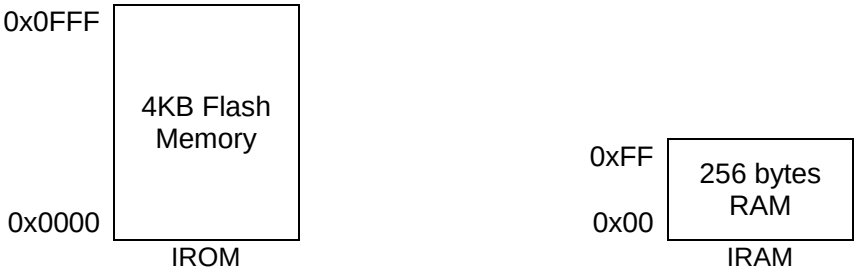
引脚名称	类型	功能说明
P2.0 AIN9 T2RL	Digital I/O Analog Input Digital Input	GPIO ADC：输入通道 Timer2：重装触发输入引脚
P2.1 AIN8	Digital I/O Analog Input	GPIO ADC：输入通道

5 CPU

SN8F5000 系列是一颗增强型的 8051 微控制器，且完全兼容 MCS-51 指令集，因此可以使用目前流行的编译环境仿真（例如 Keil C51）。总的来说，在相同的频率下，SN8F5000 的 CPU 要比原始的 8051 快 9.4 到 12.1 倍。

5.1 存储器结构

SN8F5702 内建了两个存储器：内部 RAM（IRAM）和程序存储器（IROM）。内部 RAM 由 256 个字节组成，具有较高的存取性能（支持直接寻址和间接寻址）。程序存储器是一个 4KB 的非易失性存储器，最快的存取速度可达 8MHz。



5.2 直接寻址：IRAM和SFR

直接寻址指令（如 MOV A, direct）可以访问低 128 字节的内部 RAM（地址范围：00-7FH）和所有系统寄存器（SFR，地址范围：80-0FFH）。

除此之外，内部 RAM 的最低 32 字节（00-1FH）可以看作是 4 组 R0-R7 工作寄存器，这 4 组工作寄存器可通过汇编指令（如 MOV A, R0）进行寻址。内部 RAM 的 20-2FH 以及以 0 或 8 结尾的系统寄存器地址支持位寻址。

5.3 间接寻址：IRAM

虽然通过直接寻址指令去访问内部 RAM 所花的周期比间接寻址要少，但是间接寻址可以访问内部 RAM 的所有区域，且是访问内部 RAM 的高 128 字节(80H~0FFH)的唯一方式。

5.4 程序存储器（IROM）

程序存储器是非易失存储器，可以存储偶尔需要修改的程序代码，ROM 查表数据以及其他数据。可通过调试工具进行更新，例如 SN-Link Adapter II，也可通过在线烧录程序进行自动更新（参考在线烧录章节）。

5.5 程序存储器安全

SN8F5702 内置 ROM 加密机制，防止 Flash ROM 资料被破解。当使能加密功能时，就无法读出 ROM 里面的内容，所有的 ROM 地址都只能读到 00H 的数据。

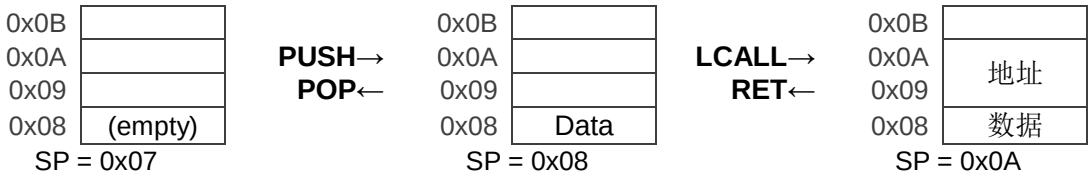
5.6 数据指针

在执行 MOVX 和 MOVC 指令时，数据指针可帮助指定 XRAM 和 IROM 地址。该单片机有 1 组数据指针（DPH/DPL）。DPC 寄存器控制 2 个功能：选择下一个数据指针和自动加减数据指针功能。

自动加减数据指针的功能是在执行 MOVX @DPTR 指令后，可使得指针指向的地址自动加 1 或减 1。因此，它能够连续的访问外部存储器，而不需要重复的去指定数据指针指向的地址。

5.7 堆栈

可从内部 RAM（IRAM）中分出任意一部分作为堆栈使用，但要求手动分配以保证堆栈区域不会与 RAM 中的其他变量重叠。堆栈溢出或者下溢也可能会导致误写 RAM 中的其他变量，故在分配堆栈的区域时必须考虑到这些问题以避免这种情形的发生。



默认情况下，堆栈指针（SP 寄存器）指向 07H，就是指堆栈的区域从 IRAM 地址中的 08H 开始。换句话说，如果计划想将堆栈区域设置为从 IRAM 的 0C0H 开始，就要在系统复位后将 SP 寄存器设置为 0BFH。

一条汇编 PUSH 指令占用堆栈中的一个字节，LCALL，ACALL 指令以及中断分别占用堆栈中的两个字节。POP 指令释放一个字节，RET/RETI 指令释放两个字节。

5.8 堆栈和数据指针寄存器

寄存器名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SP	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
DPL	DPL7	DPL6	DPL5	DPL4	DPL3	DPL2	DPL1	DPL0
DPH	DPH7	DPH6	DPH5	DPH4	DPH3	DPH2	DPH1	DPH0
DPC	-	-	-	-	-	ATMS	ATMD	ATME

SP 寄存器（0x81）

Bit	Field	Type	Initial	说明
7..0	SP	R/W	0x07	堆栈指针

DPL 寄存器（0x82）

Bit	Field	Type	Initial	说明
7..0	DPL[7:0]	R/W	0x00	DPTR 的低字节

DPH 寄存器（0x83）

Bit	Field	Type	Initial	说明
7..0	DPH[7:0]	R/W	0x00	DPTR 的高字节

DPC 寄存器（0x93）

Bit	Field	Type	Initial	说明
7..3	Reserved	R	0x0	
2..1	ATMS/ATMD	R/W	00	自动加减数据指针（使能 ATME 位时有效） 00：执行 MOVX @DPTR 指令后+1 01：执行 MOVX @DPTR 指令后-1 10：执行 MOVX @DPTR 指令后+2 11：执行 MOVX @DPTR 指令后-2
0	ATME	R/W	0	自动加减数据指针功能 0: 关闭 1: 使能

6 特殊功能寄存器

6.1特殊功能寄存器存储器

BIN HEX	000	001	010	011	100	101	110	111
F8	-	P0M	P1M	P2M	-	-	-	PFLAG
F0	B	P0UR	P1UR	P2UR	-	-	-	SRST
E8	-	-	-	-	-	-	-	-
E0	ACC	SPSTA	SPCOM	SPDAT	P1OC	CLKSEL	CLKCMD	TCON0
D8	S0CON2	-	I2CDAT	I2CADR	I2CCON	I2CSTA	SMBSEL	SMBDST
D0	PSW	IEN4	ADM	ADB	ADR	VERFH	P1CON	-
C8	T2CON	-	CRCL	CRCH	TL2	TH2	-	-
C0	IRCON	CCEN	CCL1	CCH1	CCL2	CCH2	CCL3	CCH3
B8	IEN1	IP1	S0RELH	PW1DH	PW1DL	PW1A	PW1CH	IRCON2
B0	-	-	-	-	-	-	-	-
A8	IEN0	IP0	S0RELL	PW1M	PW1YL	PW1YH	PW1BL	PW1BH
A0	P2	-	-	-	-	-	-	-
98	S0CON	S0BUR	IEN2	-	-	P0CON	P2CON	-
90	P1	P1W	-	-	PECMD	PEROML	PERONH	PERAM
88	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON	PEDGE
80	P0	SP	DPL	DPH	-	-	WDTR	PCON

6.2 特殊功能寄存器说明

80H-9FH 寄存器说明

Register	Address	Description
P0	080H	P0 数据缓存器。
SP	081H	堆栈指针寄存器。
DPL	082H	数据指针 0 低字节寄存器。
DPH	083H	数据指针 0 高字节寄存器。
-	084H	-
-	085H	-
WDTR	086H	看门狗定时器清零寄存器。
PCON	087H	系统模式寄存器。
TCON	088H	T0/1 控制寄存器。
TMOD	089H	T0/1 模式寄存器。
TL0	08AH	T0 计数低字节寄存器。
TL1	08BH	T1 计数低字节寄存器。
TH0	08CH	T0 计数高字节寄存器。
TH1	08DH	T1 计数高字节寄存器。
CKCON	08EH	扩展周期寄存器。
PEDGE	08FH	外部中断边沿控制寄存器。
P1	090H	P1 数据缓存器。
P1W	091H	P1 唤醒控制寄存器。
-	092H	-
-	093H	-
PECMD	P94H	在线编程命令寄存器。
PEROML	095H	在线编程 ROM 地址低字节。
PEROMH	096H	在线编程 ROM 地址高字节。
PERAM	097H	在线编程 RAM 分配地址。
S0CON	098H	UART 控制寄存器。
S0BUF	099H	UART 数据缓存器。
IEN2	09AH	中断使能寄存器。
-	09BH	-
-	09CH	-
P0CON	09DH	P0 配置控制寄存器。
P2CON	09EH	P2 配置控制寄存器。
-	09FH	-

0A0-0BF 寄存器说明

Register	Address	Description
P2	0A0H	P2 数据缓存器。
-	0A1H	-
-	0A2H	-
-	0A3H	-
-	0A4H	-
-	0A5H	-
-	0A6H	-
-	0A7H	-
IEN0	0A8H	中断使能寄存器。
IP0	0A9H	中断优先权寄存器。
S0RELL	0AAH	UART 重装低字节寄存器。
PW1M	0ABH	PW1 控制寄存器。
PW1YL	0ACH	PW1 周期控制缓存器低字节。
PW1YH	0ADH	PW1 周期控制缓存器高字节。
PW1BL	0AEH	PW1 B point 死区控制缓存器低字节。
PW1BH	0AFH	PW1 B point 死区控制缓存器高字节。
-	0B0H	-
-	0B1H	-
-	0B2H	-
-	0B3H	-
-	0B4H	-
-	0B5H	-
-	0B6H	-
-	0B7H	-
IEN1	0B8H	中断使能寄存器。
IP1	0B9H	中断优先权寄存器。
S0RELH	0BAH	UART 重装高字节寄存器。
PW1DL	0BBH	PW1 占空比控制寄存器低字节。
PW1DH	0BCH	PW1 占空比控制寄存器高字节。
PW1A	0BDH	PW1 A point 死区控制缓存器。
PW2A	0BEH	PW2 A point 死区控制缓存器。
IRCON2	0BFH	中断请求寄存器。

0C0-0CFH 寄存器说明

Register	Address	Description
IRCON	0C0H	中断请求寄存器。
CCEN	0C1H	T2 比较器/捕捉功能使能寄存器。
CCL1	0C2H	T2 比较器/捕捉功能模块 1 低字节寄存器。
CCH1	0C3H	T2 比较器/捕捉功能模块 1 高字节寄存器。
CCL2	0C4H	T2 比较器/捕捉功能模块 2 低字节寄存器。
CCH2	0C5H	T2 比较器/捕捉功能模块 2 高字节寄存器。
CCL3	0C6H	T2 比较器/捕捉功能模块 3 低字节寄存器。
CCH3	0C7H	T2 比较器/捕捉功能模块 3 高字节寄存器。
T2CON	0C8H	T2 控制寄存器。
-	0C9H	-
CRCL	0CAH	T2 比较器/捕捉功能模块 0& 重装功能低字节寄存器。
CRCH	0CBH	T2 比较器/捕捉功能模块 0& 重装功能高字节寄存器。
TL2	0CCH	T2 计数低字节寄存器。
TH2	0CDH	T2 计数高字节寄存器。
-	0CEH	-
-	0CFH	-
PSW	0D0H	系统标志寄存器。
IEN4	0D1H	中断数据寄存器。
ADM	0D2H	ADC 控制寄存器。
ADB	0D3H	ADC 数据缓存器。
ADR	0D4H	ADC 分辨率选择寄存器。
VREFH	0D5H	ADC 参考电压控制寄存器。
P1CON	0D6H	P1 配置控制寄存器。
-	0D7H	-
S0CON2	0D8H	UART 波特率控制寄存器。
-	0D9H	-
I2CDAT	0DAH	I2C 数据缓存器。
I2CADR	0DBH	I2C 从动地址。
I2CCON	0DCH	I2C 接口操作控制寄存器。
I2CSTA	0DDH	I2C 状态代码。
SMBSEL	0DEH	SMBUS 模式控制寄存器。
SMBDST	0DFH	SMBUS 内部超时寄存器。

0E0-0FFH 寄存器说明

Register	Address	Description
ACC	0E0H	ACC 寄存器。
SPSTA	0E1H	SPI 状态寄存器。
SPCON	0E2H	SPI 控制寄存器。
SPDAT	0E3H	SPI 数据缓存器。
P0OC	0E4H	开漏功能控制寄存器。
CLKSEL	0E5H	时钟切换选择寄存器。
CLKCMD	0E6H	时钟切换控制寄存器。
TCON0	0E7H	T0/1 时钟控制寄存器。
-	0E8H	-
-	0E9H	-
-	0EAH	-
-	0EBH	-
-	0ECH	-
-	0EDH	-
-	0EEH	-
-	0EFH	-
B	0F0H	乘法/除法指令数据缓存器。
P0UR	0F1H	P0 上拉电阻控制寄存器。
P1UR	0F2H	P1 上拉电阻控制寄存器。
P2UR	0F3H	P2 上拉电阻控制寄存器。
-	0F4H	-
-	0F5H	-
-	0F6H	-
SRST	0F7H	软件复位控制寄存器。
-	0F8H	-
P0M	0F9H	P0 输入/输出模式寄存器。
P1M	0FAH	P1 输入/输出模式寄存器。
P2M	0FBH	P2 输入/输出模式寄存器。
-	0FCH	-
-	0FDH	-
-	0FEH	-
PFLAG	0FFH	复位标志寄存器。

6.3 系统寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ACC	ACC7	ACC6	ACC5	ACC4	ACC3	ACC2	ACC1	ACC0
B	B7	B6	B5	B4	B3	B2	B1	B0
PSW	CY	AC	F0	RS1	RS0	OV	F1	P

ACC 寄存器（0xE0）

Bit	Field	Type	Initial	Description
7..0	ACC[7:0]	R/W	0x00	8 位数据寄存器用于转移或操控 ALU 和数据存储器之间的数据，若操作结果溢出（OV）或者由借位（C 或 AC），以及相等情况（P）发生时，该标志位会在 PSW 寄存器中进行设置。

B 寄存器（0xF0）

Bit	Field	Type	Initial	Description
7..0	B[7:0]	R/W	0x00	B 寄存器在使用乘法和除法指令时使用，而且还能作为 scratch-pad 寄存器来保留临时数据。

PSW 寄存器（0xD0）

Bit	Field	Type	Initial	Description
7	CY	R/W	0	进位标志。 0：加法运算后没有进位、减法运算有借位发生或移位后移出逻辑“0”或比较运算的结果<0； 1：加法运算后有进位、减法运算没有借位发生或移位后移出逻辑“1”或比较运算的结果≥0。
6	AC	R/W	0	辅助进位标志。 0：BCD 操作时没有从 ACC 的第三位开始执行； 1：BCD 操作时从 ACC 的第三位开始执行。
5	F0	R/W	0	通用标志位，可任意设定。
4..3	RS[1:0]	R/W	00	寄存器 bank 选择控制位，用于选择工作寄存器 bank。 00：00H-07H（Bank0）；01：08H-0FH（Bank1）； 10：10H-17H（Bank2）；11：18H-1FH（Bank3）。
2	OV	R/W	0	溢出标志。 0：算术操作时，ACC 没有溢出； 1：算术操作时，ACC 溢出。
1	F1	R/W	0	通用标志位，可任意设定。
0	P	R	0	奇偶标志位。 0：A 中 1 的个数为偶数； 1：A 中 1 的个数为奇数

6.4 寄存器宣告

SN8F5702 通过不同的寄存器控制不同的功能，但在 C51/A51 编译器中没有预先定义 SFR 的名称。为编程更容易，就需要增加一个 header 文件来宣告 SFR 名称。

使用汇编语言进行编程时，增加下面的句子：

```
1 $NOMOD51 ; Do not recongnize the 8051-specific predifined special registers.
2 #include <SN8F5702.H>
```

使用 C 运用进行编程是，增加下面的句子：

```
1 #include <SN8F5702.H>
```

增加 header 文件后，用户可使用寄存器的名称进行编程。编译过程中，编译器通过 header 文件将寄存器的名称转换出寄存器的位置。

不同的设备需要使用不同的 header 文件进行宣告，但 option 文件是一样的。

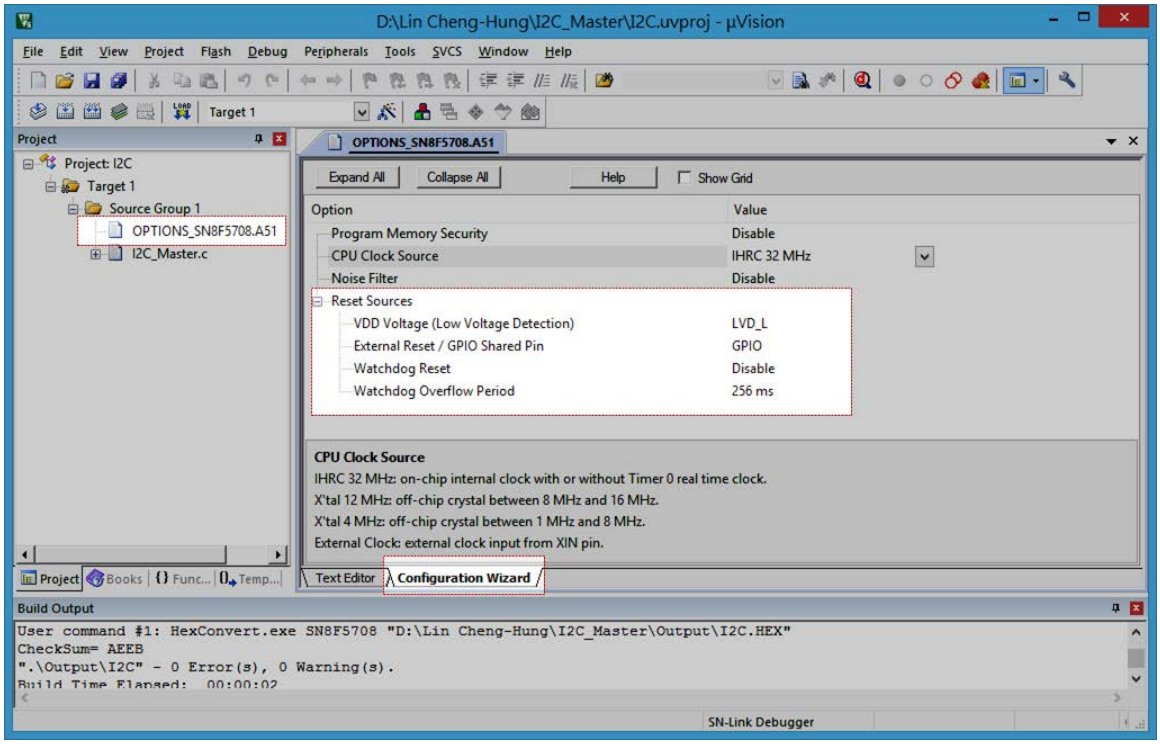
Divice	Header file	Options file
SN8F5702	SN8F5702.H	OPTIONS_SN8F5702.A51
SN8F570200	SN8F570200.H	
SN8F570210	SN8F570210.H	
SN8F570211	SN8F570211.H	
SN8F570212	SN8F570212.H	
SN8F570213	SN8F570213.H	

7 复位和上电控制

复位和上电控制有下列方式：低电压检测（LVD），看门狗，可编程的外部复位引脚和软件复位。前面三种方式可触发额外的上电流程，随后单片机初始化所有的寄存器，并使程序从复位向量（ROM 地址 00H）处重新开始执行。

7.1 复位配置和上电控制

SONiX 发布了一个 SN8F5702_OPTIONS.A51 文件，该文件包含在 SN8F5702 软件包中（从松翰官网 www.sonix.com.tw 下载）。该文件包含了复位源和CPU时钟源选择的合适参数，强烈建议将这个文件加到Keil项目中。SN8F5000 的调试工具手册提供了更详细的内容。



7.2 上电流程

LVD，看门狗和外部复位引脚可触发上电流程，在复位信号结束之后开始上电流程，上电完成之后开始运行程序。总的来说，上电流程包括 2 个阶段：电源稳定期和时钟稳定期。

一般情况下，电源稳定期花费 4.5ms，之后单片机会自动获取 CPU 时钟源的配置。所选择的时钟源驱动起来之后，系统会计数 4096 个时钟周期以确保时钟已稳定。

7.3 LVD复位

低电压检测监控 VDD 引脚的电压，共 3 个电压点：1.8V，2.4V 和 3.3V。根据 LVD 的配置，比较结果可作为系统的复位信号，或者仅作为 LVD24/LVD33 寄存器标志位。下表显示了 4 个不同的 LVD 配置，从 LVD_Max 到 LVD_L，在不同的 VDD 引脚状态下，相对应的复位状态或 LVD24/LVD33 标志位。

Condition	LVD_Max	LVD_H	LVD_M	LVD_L
VDD ≤ 3.3 V	Reset	LVD33 = 1	-	-
VDD ≤ 2.4 V	Reset	Reset	LVD24 = 1	-
VDD ≤ 1.8 V	Reset	Reset	Reset	Reset

7.4 看门狗复位

看门狗是周期性的复位信号发生器，用于监控程序的执行流程。其内部定时器可在程序流程的检测点被清零，因此，只有在发生软件问题后才会产生实际的复位信号。通过写入 5AH 到 WDTR 就可以在程序中设置检测点。

```
1    WDTR = 0x5A;
```

看门狗定时器间隔时间=256*1/(内部低速振荡器频率/WDT前置频率)=256/(F_{ILRC}/WDT前置频率)…SEC

内部低速振荡器	WDT 前置频率	看门狗间隔时间
F_{ILRC}=16KHz	F_{ILRC} /4	256/(16000/4)=64ms
	F_{ILRC} /8	256/(16000/8)=128ms
	F_{ILRC} /16	256/(16000/16)=256ms
	F_{ILRC} /32	256/(16000/32)=512ms

看门狗的工作模式可在选项文件中配置：
Always Mode：其内部定时器在 CPU 的所有工作模式（Normal，IDLE，SLEEP）下都在计数。
Enable Mode：其内部定时器只在 CPU 的 Normal 模式下计数，在 IDLE 和 SLEEP 模式下不会触发看门狗复位。
Disable Mode：其内部定时器在 CPU 的所有工作模式下都不计数，在这种模式下不会触发看门狗复位。

看门狗的工作模式为 Always on 时，系统的功耗较大。

7.5 外部复位引脚

可在选项文件中配置可编程的外部复位引脚，使能外部复位引脚后，系统会监控该复用引脚的逻辑电平，检测到低电平（小于 30%VDD）后立即触发系统复位直到该引脚恢复到高电平（大于 70%VDD）。

可选择的消抖周期可以增强复位信号的稳定性，区别于立即复位，系统复位需要 8ms 长的逻辑低电平时间来避免跳键的发生。小于这个消抖周期的信号都不会影响 CPU 的执行。

7.6 软件复位

连续设置 SRSTREQ 寄存器后会产生软件复位，因此，该过程可使固件有权限去复位单片机（如更新固件之后的复位）。下面这段 C 程序代码通过重复设置 SRST 寄存器的最低位来实现软件复位。

```
1    SRST = 0x01;
2    SRST = 0x01;
```

7.7 复位和上电控制寄存器

寄存器名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	POR	WDT	RST	-	-	LVD24	LVD33	-
SRST	-	-	-	-	-	-	-	SRSTREQ
WDTR	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0

PFLAG 寄存器

Bit	Field	Type	Initial	说明
7	POR	R	0	若单片机已经由 LVD 触发复位，则该位自动置 1
6	WDT	R	0	若单片机已经由看门狗触发复位，则该位自动置 1
5	RST	R	0	若单片机已经由外部复位引脚触发复位，则该位自动置 1
3..2	Reserved	R	0	
2	LVD24	R	0	若 VDD 引脚的电压低于 2.4V，则该位自动置 1
1	LVD33	R	0	若 VDD 引脚的电压低于 3.3V，则该位自动置 1
0	Reserved	R	0	

SRST 寄存器

Bit	Field	Type	Initial	说明
7..1	Reserved	R	0	
0	SRSTREQ	R/W	0	连续设置该位 2 次触发软件复位

8 系统时钟和电源管理

单片机内置 3 个不同的操作模式：Normal 模式，IDLE 模式和 STOP 模式以省电。

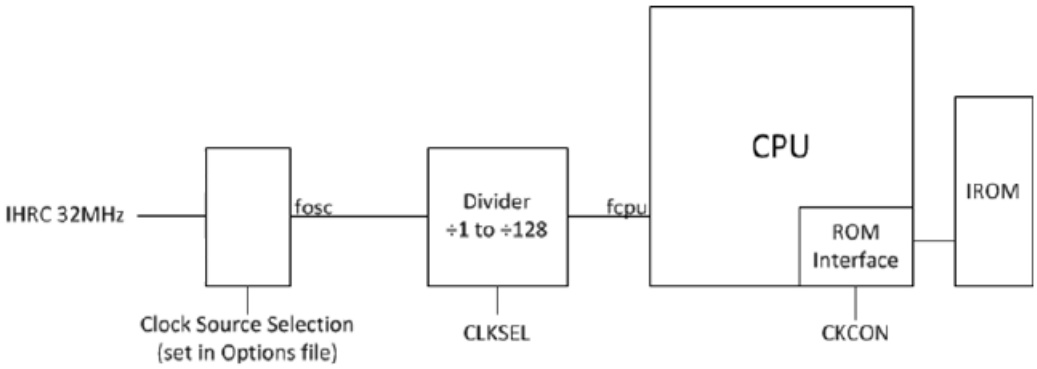
Normal 模式是指 CPU 和外设功能都正常工作，系统时钟由所选择的时钟源，时钟分频数以及程序 ROM 等待周期来共同决定的。IDLE 模式是指 CPU 时钟以及程序运行都暂停了的状态，但保留了外设功能（如定时器，PWM，SPI，UART 和 I2C）。与之相反的是，STOP 模式则禁止所有功能和时钟发生器，直至唤醒信号将系统唤醒进入 Normal 模式。

8.1 系统时钟

该单片机包括内置时钟发生器（IHRC 32MHz），晶体/陶瓷驱动器和外部时钟输入。在复位和上电过程中，系统自动加载所选择时钟源。该时钟源可以看作是 Fosc，Fosc 一旦选定就不能再改变。

之后， Fosc 可以分频为 Fosc/1~Fosc/128，由 CLKSEL 寄存器控制。CPU 使用分频之后的时钟作为它的时钟频率（称作 Fcpu）。写入 69H 到 CLKCMD 寄存器时设置 CLKSEL。

```
CLKSEL = 0x04;       //set fcpu = fosc/8
CLKCMD = 0x69;       //Apply CLKSEL's setting
```



ROM 接口位于 CPU 和 IROM（程序存储器）之间，可设置 ROM 读取周期以支持低速程序存储器。例如：CPU 计划运行在 32MHz，而 IROM 只能运行在 8MHz 以下，则在 CKCON 寄存器中必须设置 ROM 读取周期为增加 3 个以上的 Fcpu。

$$\text{IROM fetching cycle} = \frac{\text{fcpu}}{\text{PWSC}[2:0]+1} \leq 8\text{MHz}, \text{PWSC}[2:0] = 0\sim7$$

8.2 电源管理

复位信号和上电结束后，CPU 以 Fcpu 的速率开始执行程序。此时，CPU 以及所有的外设都正常工作（称之为 Normal 模式）。

PCON 寄存器的最低 2 位（bit0-IDLE 和 bit1-STOP）控制单片机的电源管理部分。

若 IDLE 位设置为 1，只有 CPU 时钟源仍使能。因此，在这种状态下，外设功能（如定时器，PWM 和 I2C）和时钟发生器（IHRC 32MHz）仍然正常工作。P0/P1 输入的任何改变和中断事件都会导致单片机返回到 Normal 模式，IDLE 位自动清零。

若 STOP 位设置为 1，CPU，外设和时钟发生器都处于停止状态，在这个模式下，寄存器中存储的数据和 RAM 都保持不变。P0/P1 输入的任何改变都可将单片机唤醒并使系统继续执行，STOP 位自动清零。

8.3 系统时钟和电源管理寄存器

寄存器名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CLKSEL	-	-	-	-	-	CLKSEL2	CLKSEL1	CLKSEL0
CLKCMD	CMD7	CMD6	CMD5	CMD4	CMD3	CMD2	CMD1	CMD0
CKCON	-	PWSC2	PWSC1	PWSC0	ESYN	EWSC2	EWSC1	EWSC0
PCON	SMOD	-	-	-	P2SEL	GF0	STOP	IDLE
P1W	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W

CLKSEL 寄存器（0xE5）

Bit	Field	Type	Initial	说明
7..3	Reserved	R	0x00	
2..0	CLKSEL[2:0]	R/W	111	CLKSEL 中的设置将在写 CLKCMD 之后生效。 000：Fcpu = Fosc / 128； 001：Fcpu = Fosc / 64； 010：Fcpu = Fosc / 32； 011：Fcpu = Fosc/ 16； 100：Fcpu = Fosc / 8； 101：Fcpu = Fosc / 4； 110：Fcpu = Fosc / 2； 111：Fcpu = Fosc/ 1；

CLKCMD 寄存器（0xE6）

Bit	Field	Type	Initial	说明
7..0	CMD[7:0]	W	0x00	写入 69H 来应用 CLKSEL 的设置。

CKCON 寄存器（0x8E）

Bit	Field	Type	Initial	说明
7	Reserved	R	0	
6..4	PWSC[2:0]	R/W	111	增加读取程序存储器的周期。 000：无； 001：1 个周期； 010：2 个周期； 011：3 个周期； 100：4 个周期； 101：5 个周期； 110：6 个周期； 111：7 个周期；
3	ESYN	R/W	0	增加额外的写数据到 XRAM 的周期。
2..0	EWSC[2:0]	R/W	001	增加读取 XRAM 的周期。 000：无； 001：1 个周期； 010：2 个周期； 011：3 个周期； 100：4 个周期； 101：5 个周期； 110：6 个周期； 111：7 个周期。

PCON 寄存器（0x87）

Bit	Field	Type	Initial	说明
7..3	Reserved	R	0x00	
1	STOP	W	0	1：单片机切换到 STOP 模式。
0	IDLE	W	0	1：单片机切换到 IDLE 模式。

P1W 寄存器（0x91）

Bit	Field	Type	Initial	说明
7..0	P1nW	R/W	0	0：禁止 P1.n 的唤醒功能； 1：使能 P1.n 的唤醒功能。

9 中断

SN8F5702 包含 13 个中断源（1 个外部中断和 12 个外部中断），分为 4 个优先级别。每个中断源包含 1 个或多个中断请求标志。有中断发生时，相对应的中断请求位置为逻辑 1。若同时使能中断使能位和全局中断使能位（EAL=1）时，有中断请求时则执行该中断服务程序（ISR）。多数中断请求标志位必须由软件清零，而有些中断请求标志位由硬件自动清零。最后，执行 RETI 指令后，整个 ISR 结束。所有的中断源，中断向量，优先等级以及控制位，如下表所示：

中断源	使能中断标志位	请求标志位 (IRQ)	IRQ 清除	优先级 / 向量
系统复位	-	-	-	0 / 0x0000
INT0	EX0	IE0	自动清 0	1 / 0x0003
PWM1	EPWM1	PWM1F	软件清 0	2 / 0x0083
I2C	EI2C	SI	软件清 0	3 / 0x0043
Timer 0	ET0	TF0	自动清 0	4 / 0x000B
ADC	EADC	ADCF	软件清 0	5 / 0x008B
SPI	ESPI	SPIF / WCOL SSERR / MODF	软件清 0	6 / 0x004B
T2COM0	ET2C0	TF2C0	自动清 0	7 / 0x0053
Timer 1	ET1	TF1	自动清 0	8 / 0x001B
T2COM1	ET2C1	TF2C1	自动清 0	9 / 0x005B
UART	ES0	TI0 / RI0	软件清 0	10 / 0x0023
T2COM2	ET2C1	TF2C2	自动清 0	11 / 0x0063
Timer 2	ET2 / ET2RL	TF2 / TF2RL	软件清 0	12 / 0x002B
T2COM3	ET2C3	TF2C3	自动清 0	13 / 0x006B

9.1 中断操作

中断操作由中断请求标志位和中断使能位控制。中断请求标志位显示中断源的状态，与中断功能的状态（使能或禁止）无关。同时使能中断使能位和全局中断使能位（EAL=1）且中断请求标志位有效时，程序计数器指向中断向量（03H-08BH），系统执行相对应的中断服务程序 ISR。

9.2 中断优先级别

每个中断源都有其默认的优先级。若同时发生 2 个中断，系统会先执行优先级别高的 ISR，然后再执行优先级别低的 ISR。下一次的 ISR 必须要等待前面的 ISR 执行完成之后才可以执行，不用理会中断的优先级别。

对应特定的优先权需求，需要用到 4 级的优先级别（级别 0-级别 3）。所有的中断源分为 6 个等级组（Group0-Group5），可分别为每组设置一个优先级别，由寄存器 IP0/IP1 设置，级别 3 最高，级别 0 最低。同组的中断源共用同样的优先级别，对于同样的优先级别，按照默认的优先级来排序。

优先级别	IP1.x	IP0.x
Level 0	0	0
Level 1	0	1
Level 2	1	0
Level 3	1	1

首先执行优先级别较高的 ISR，甚至可以打断执行中的优先级别较低的 ISR，直到优先级别较高的 ISR 执行完成之后再执行优先级别较低的 ISR。

Group	中断源			
Group 0	INT0	PWM1	I2C	
Group 1	T0	ADC	SPI	
Group 2			T2 COM0	
Group 3	T1		T2 COM1	
Group 4	UART		T2 COM2	
Group 5	T2		T2 COM3	

IP0, IP1 寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IP0	-	-	IP05	IP04	IP03	IP02	IP01	IP00
IP1	-	-	IP15	IP14	IP13	IP12	IP11	IP10

IP0 寄存器（0XA9）

Bit	Field	Type	Initial	说明
5..0	IP0[5:0]	R/W	0	中断优先权。和 IP1 寄存器的相对应的位结合在一起可以指定相对应的中断的优先级别。
Else	Reserved	R	0	

IP1 寄存器（0XB9）

Bit	Field	Type	Initial	说明
5..0	IP1[5:0]	R/W	0	中断优先权。和 IP0 寄存器的相对应的位结合在一起可以指定相对应的中断的优先级别。
Else	Reserved	R	0	

9.3 中断寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IEN0	EAL	-	ET2	ES0	ET1	-	ET0	EX0
IEN1	ET2RL	-	ET2C3	ET2C2	ET2C1	ET2C0	ESPI	EI2C
IEN2	-	-	-	-	-	-	EADC	-
IEN4	EPWM1	-	-	-	PWM1F	-	-	-
IRCON	TF2RL	TF2	TF2C3	TF2C2	TF2C1	TF2C0	-	-
IRCON2	-	-	-	-	-	-	-	ADCF
TCON	TF1	TR1	TF0	TR0	-	-	IE0	-
S0CON	SM0	SM1	SM20	REN0	TB80	RB80	TI0	RI0
SPSTA	SPIF	WCOL	SSERR	MODF	-	-	-	-
I2CCON	CR2	ENS1	STA	STO	SI	AA	CR1	CR0

IEN0 寄存器（0XA8）

Bit	Field	Type	Initial	说明
7	EAL	R/W	0	所有中断使能控制位。 0：禁止； 1：使能。
5	ET2	R/W	0	T2 定时器中断控制位。 0：禁止； 1：使能。
4	ES0	R/W	0	UART 中断控制位。 0：禁止； 1：使能。
3	ET1	R/W	0	T1 定时器中断控制位。 0：禁止； 1：使能。
1	ET0	R/W	0	T0 定时器中断控制位。 0：禁止； 1：使能。
0	EX0	R/W	0	外部中断 P1.0（INT0）中断控制位。 0：禁止； 1：使能。
Else	Reserved	R	0	

IEN1 寄存器（0XB8）

Bit	Field	Type	Initial	说明
7	ET2RL	R/W	0	T2 定时器外部重装中断控制位。 0：禁止； 1：使能。
5	ET2C3	R/W	0	T2 定时器 COM3 中断控制位。 0：禁止； 1：使能。
4	ET2C2	R/W	0	T2 定时器 COM2 中断控制位。 0：禁止； 1：使能。
3	ET2C1	R/W	0	T2 定时器 COM1 中断控制位。 0：禁止； 1：使能。
2	ET2C0	R/W	0	T2 定时器 COM0 中断控制位。 0：禁止； 1：使能。
1	ESPI	R/W	0	SPI 中断控制位。 0：禁止； 1：使能。
0	EI2C	R/W	0	I2C 中断控制位。 0：禁止； 1：使能。
Else	Reserved	R	0	

IEN2 寄存器（0X9A）

Bit	Field	Type	Initial	说明
1	EADC	R/W	0	ADC 中断控制位。 0：禁止； 1：使能。
Else	Reserved	R	0	

IEN4 寄存器（0XD1）

Bit	Field	Type	Initial	说明
7	EPWM1	R/W	0	PWM1 中断控制位。 0：禁止； 1：使能。
3	PWM1F	R/W	0	PWM1 中断请求标志位。 0：无 PWM1 中断请求； 1：PWM1 请求中断。
Else	Reserved	R	0	

IRCON 寄存器（0xC0）

Bit	Field	Type	Initial	说明
7	TF2RL	R/W	0	T2 定时器外部重装中断请求标志位。 0：无 TF2RL 中断请求； 1：TF2RL 请求中断。
6	TF2	R/W	0	T2 定时器中断请求标志位。 0：无 T2 中断请求； 1：T2 请求中断。
5	TF2C3	R/W	0	T2 定时器 COM3 中断请求标志位。 0：无 T2COM3 中断请求； 1：T2COM3 请求中断。
4	TF2C2	R/W	0	T2 定时器 COM2 中断请求标志位。 0：无 T2COM2 中断请求； 1：T2COM2 请求中断。
3	TF2C1	R/W	0	T2 定时器 COM1 中断请求标志位。 0：无 T2COM1 中断请求； 1：T2COM1 请求中断。
2	TF2C0	R/W	0	T2 定时器 COM0 中断请求标志位。 0：无 T2COM0 中断请求； 1：T2COM0 请求中断。
Else	Reserved	R	0	

IRCON2 寄存器（0XBF）

Bit	Field	Type	Initial	说明
0	ADCF	R/W	0	ADC 中断请求标志位。 0：无 ADC 中断请求； 1：ADC 请求中断。
Else	Reserved	R	0	

TCON 寄存器（0X88）

Bit	Field	Type	Initial	说明
7	TF1	R/W	0	T1 定时器外部重装中断请求标志位。 0：无 T1 中断请求； 1：T1 请求中断。
5	TF0	R/W	0	T0 定时器外部重装中断请求标志位。 0：无 T0 中断请求； 1：T0 请求中断。
1	IE0	R	0	外部中断 P1.0（INT0）中断请求标志位。 0：无 INT0 中断请求； 1：INT0 请求中断。
Else				参考其它章节。

S0CON 寄存器（0X98）

Bit	Field	Type	Initial	说明
1	TI0	R/W	0	UART 发送中断请求标志位，显示 UART 串行传输的完成状态。在在模式 0 的第 8 位结束时，或者其他模式的停止位开始时，由硬件设置为 1；必须由软件清零。 0：无 UART 发送中断请求； 1：UART 发送请求中断。
0	RI0	R/W	0	UART 接收中断请求标志位，当 UART 串行接收完成时，由硬件置 1。在模式 0 的第 8 位结束时，或者其他模式的停止位的中间时，由硬件设置为 1；必须由软件清零。 0：无 UART 接收中断请求； 1：UART 接收请求中断。
Else				参考其它章节。

SPSTA 寄存器（0XE1）

Bit	Field	Type	Initial	说明
7	SPIF	R	0	SPI 完成通讯标志位。 通讯结束时自动设置为 1； 读取 SPSTA、SPDAT 寄存器时自动清零。
4	MODF	R	0	模式错误标志位。
Else				参考其它章节。

I2CCON 寄存器（0XDC）

Bit	Field	Type	Initial	说明
7	SI	R/W	0	串行中断标志位。 当进入了 I2C 的 26 个状态当中的 25 个状态时，SI 标志位会被硬件置 1，只有当 I2C 状态寄存器为 F8h 时，SI 标志位才没有被置 1，表示没有可用的相关状态信息。SI 标志位必须由软件清零，必须通过写 0 到 SI 标志才可以清 SI 标志位，写入 1 到该位并不能更改 SI 的值。
Else				参考其它章节。

10 GPIO

SN8F5702 共有 18 个 GPIO 引脚，不同于 8051 只有开漏输出，SN8F5702 还内置推挽式输出结构，以增强其驱动能力。

10.1 输入输出控制

由 P0M-P2M 寄存器控制输入输出模式。

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M
P1M	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M
P2M	-	-	-	-	-	-	P21M	P20M
P0OC	-	-	-	P15OC	P14OC	P13OC	P06OC	P05OC

P0M: 0xF9, P1M: 0xFA, P2M: 0xFB

Bit	Field	Type	Initial	说明
7	P07M	R/W	0	P0.7 的模式选择控制位 0：输入模式； 1：输出模式。
6	P06M	R/W	0	P0.6 的模式选择控制位 0：输入模式； 1：输出模式。
5	P05M	R/W	0	P0.5 的模式选择控制位 0：输入模式； 1：输出模式。
4..0				其它

P0OC 寄存器（0xE4）

Bit	Field	Type	Initial	说明
7..5		R/W	000	参考 PWM 章节
4	P15OC	R/W	0	P1.5 开漏输出模式控制位。 0：禁止； 1：使能，输出高电平时变为输入模式。
3	P14OC	R/W	0	P1.4 开漏输出模式控制位。 0：禁止； 1：使能，输出高电平时变为输入模式。
2	P13OC	R/W	0	P1.3 开漏输出模式控制位。 0：禁止； 1：使能，输出高电平时变为输入模式。
1	P06OC	R/W	0	P0.6 开漏输出模式控制位。 0：禁止； 1：使能，输出高电平时变为输入模式。
0	P05OC	R/W	0	P0.5 开漏输出模式控制位。 0：禁止； 1：使能，输出高电平时变为输入模式。

10.2 输入数据和输出数据

当从 P0~P2 寄存器进行读操作时，当前引脚的逻辑电平将取决于它的外部状态。在某些情况下，当 IO 口在和其他功能复用时，例如 UART 或 I2C，读操作依然可行。

写给 P0~P2 寄存器的值将被马上锁存，然而，只有在 P0M ~P2M 被设置为输出模式之后才会被输出。如果这个引脚已经是输出模式了，任何写到 P0~P2 寄存器的值将会马上输出到这个引脚上。

寄存器	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	P07	P06	P05	P04	P03	P02	P01	P00
P1	P17	P16	P15	P14	P13	P12	P11	P10
P2	-	-	-	-	-	-	P21	P20

P0: 0x80, P1: 0x90, P2: 0xA0

Bit	Field	Type	Initial	说明
7	P07	R/W	1	读：P0.7 为逻辑低电平 写入 1/0：输出高电平/低电平（P07M=1 时使能）
6	P06	R/W	1	读：P0.6 为逻辑低电平 写入 1/0：输出高电平/低电平（P06M=1 时使能）
5	P05	R/W	1	读：P0.5 为逻辑低电平 写入 1/0：输出高电平/低电平（P05M=1 时使能）
4..0				其它

10.3 内置上拉寄存器

P0UR-P2UR 寄存器管理每个引脚的内部 100KΩ（典型值）的上拉电阻。

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	P07UR	P06UR	P05UR	P04UR	P03UR	P02UR	P01UR	P00UR
P1UR	P17UR	P16UR	P15UR	P14UR	P13UR	P12UR	P11UR	P10UR
P2UR	-	-	-	-	-	-	P21UR	P20UR

P0UR: 0xF1, P1UR: 0xF2, P2UR: 0xF3

Bit	Field	Type	Initial	说明
7	P07UR	R/W	0	P0.7 的内置上拉电阻控制位。 0：禁止*； 1：使能。
6	P06UR	R/W	0	P0.6 的内置上拉电阻控制位。 0：禁止*； 1：使能。
5	P05UR	R/W	0	P0.5 的内置上拉电阻控制位。 0：禁止*； 1：使能。
4..0				其它

* 如果引脚为输出模式或者模拟功能，建议禁止上拉电阻。

10.4 与模拟功能共用的引脚

SN8F5702 内置模拟功能，如 ADC。若使能引脚的模拟功能，强烈建议关闭输入通道的施密特触发。

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1CON	P1CON7	P1CON6	P1CON5	P1CON4	P1CON3	P1CON2	P1CON1	P1CON0
P2CON	-	-	-	-	-	-	P2CON1	P2CON0

P2CON: 0x9E, P1CON: 0xD6

Bit	Field	Type	Initial	说明
7	P1CON7	R/W	0	P1.7 施密特触发控制位 0：使能； 1：禁止。
6	P1CON6	R/W	0	P1.6 施密特触发控制位 0：使能； 1：禁止。
5	P1CON5	R/W	0	P1.5 施密特触发控制位 0：使能； 1：禁止。
4..0				其它

11 外部中断

外部中断源 INT0 内置边沿触发功能，由 PEDGE 寄存器控制。使能外部中断（EX0）和全局中断（EAL）后，发生边沿触发事件时，外部中断请求标志位（IE0）置 1，程序计数器跳转到中断向量（ORG 0003H）并执行中断服务程序。在执行 ISR 之前由硬件清中断请求标志。

11.1 外部中断寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEDGE	-	-	-	-	-	-	EX0G1	EX0G0
IEN0	EAL	-	ET2	ES0	ET1	-	ET0	EX0

PEDGE 寄存器（0X8F）

Bit	Field	Type	Initial	说明
1..0	EX0G[1:0]	R/W	10	外部中断 INT0 触发沿控制位。 00：保留； 01：上升沿； 10：下降沿（默认）； 11：上升/下降沿。
Else	Reserved	R	0	

11.2 示例代码

下面的示例代码程序演示了如何执行 INT0 中断。

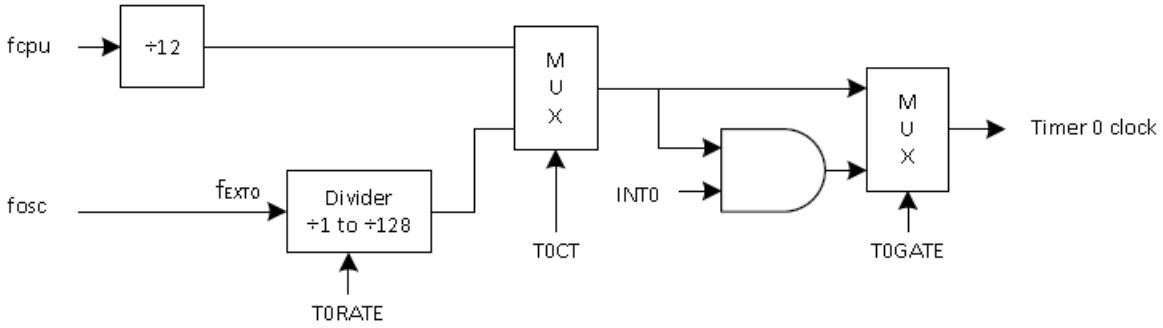
```
#define INT0Rsing        (1 << 0)   //INT0 trigger edge is rising edge
#define INT0Falling      (2 << 0)   //INT0 trigger edge is falling edge
#define INT0LeChge       (3 << 0)   //INT0 trigger edge is level change
#define EINT0            (1 << 0)   //INT0 interrupt enable

void EnableINT(void)
{
  //INT0 rising edge
  PEDGE = INT0Rising;

  //Enable INT0 interrupt
  IEN0 | = EINT0;
  //Enable total interrupt
  IEN0 | = 0x80;

  P0 = 0x00;
  P0M = 0X07;
}

void INT0Interrupt(void) interrupt ISRInt0  //0x03
{  //IE0 clear by hardware
   P00 = ~P00;
}
```

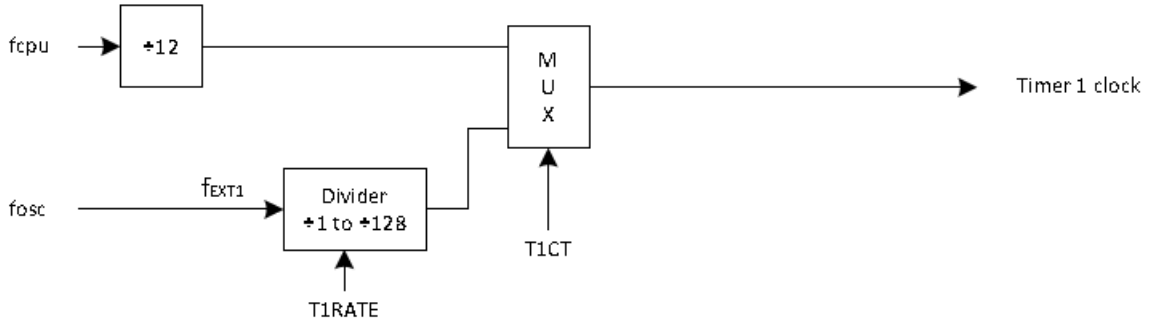
12 定时器 T0 和 T1

T0 和 T1 是 2 个独立的二进制定时器。T0 共有 4 种不同的的操作模式：模式 1：13 位向上计数定时器；模式 2：16 位向上计数定时器；模式 3：8 位向上计数寄存器，支持指定的重装值；模式 4：独立的 2 个 8 位向上计数定时器。而 T1 只有与 T0 相同的模式 0 到模式 2 三种操作模式。T0 和 T1 分别支持 ET0 和 ET1 中断。

12.1 T0 和T1 时钟选择

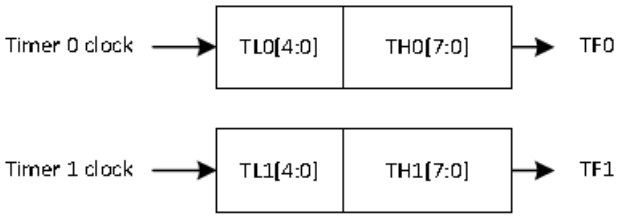
下图阐明了 T0 和 T1 的时钟选择电路，总的来说，时钟源主要为 Fcpu 和 Fosc（IHRC 32MHz）





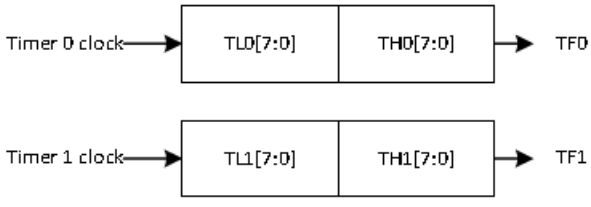
12.2 模式 0：13 位向上计数定时器

模式 0 是 13 位向上计数定时器（TL0/TL1 的高 3bit 无效），具有 2 个时钟源选择：Fcpu 和 Fosc。当 T0GATE/T1GATE 设置为 1 时，定时器的使能和停止可以由 INT0/INT1 引脚控制。一旦定时器的计数器溢出（从 0FF1FH 到 0000H），会立即置位 TF0/TF1 标志。没有使能 ET0/ET1 时，由软件读取该标志；使能 ET0/ET1 时，则由中断控制器进行控制。



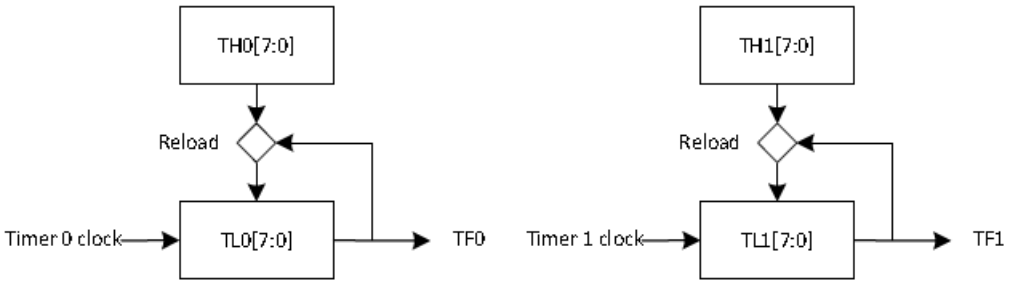
12.3 模式 1：16 位向上计数定时器

模式 1 是 16 位向上计数定时器，具有 2 个时钟源选择：Fcpu 和 Fosc。当 T0GATE/T1GATE 设置为 1 时，定时器的使能和停止可以由 INT0/INT1 引脚控制。一旦定时器的计数器溢出（从 0FFFFH 到 0000H），会立即置位 TF0/TF1。没有使能 ET0/ET1 时，由软件读取标志；使能 ET0/ET1 时，则由中断控制器进行控制。



12.4 模式 2：8 位向上计数定时器（指定重装值）

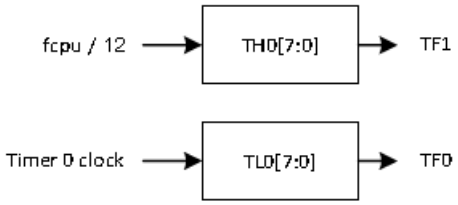
模式 2 是 8 位向上计数定时器（TL0/TL1），指定重装值。计数器溢出时（从 0FFH 到 00H）释放 TF0/TF1 标志给固件或中断控制器；同时定时器复制 TH0/TH1 的值给 TL0/TL1 寄存器。因此，定时器实际上是从 0FFHT 计数到 H0/TH1 的值。



12.5 模式 3（只是T0）：独立的 2 个 8 位向上计数定时器

模式 3 是把 TH0 和 TL0 看作独立的 2 个 8 位定时器。8 位向上计数定时器 TL0 支持 RTC，有 2 种时钟源选择（Fcpu 和 Fosc）；而 TH0 的时钟源固定为 Fcpu/12。当 T0GATE 设置为 1 时，定时器的使能和停止可以由 INT0 引脚控制。

在模式 3 下，由 TR0 使能 TL0 计数器，TL0 溢出后 TF0 置 1。TH0 计数器则由 TR1 控制，TH0 溢出后 TF1 置 1。在该模式下，T1 不会发生溢出事件，可以看作是没有溢出标志产生计数器。



12.6 T0 和T1 寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TCON	TF1	TR1	TF0	TR0	-	-	IE0	-
TCON0	-	T0RATE2	T0RATE1	T0RATE0	-	T1RATE2	T1RATE1	T1RATE0
TMOD	-	T1CT	T1M1	T1M0	T0GATE	T0CT	T0M1	T0M0
TH0	TH07	TH06	TH05	TH04	TH03	TH02	TH01	TH00
TL0	TL07	TL06	TL05	TL04	TL03	TL02	TL01	TL00
TH1	TH17	TH16	TH15	TH14	TH13	TH12	TH11	TH10
TL1	TL17	TL16	TL15	TL14	TL13	TL12	TL11	TL10
IEN0	EAL	-	ET2	ES0	ET1	-	ET0	EX0

TCON 寄存器（0x88）

Bit	Field	Type	Initial	说明
7	TF1	R/W	0	T1 溢出事件显示位。 0：T1 没有溢出； 1：T1 溢出。 该位由中断助力器自动清零，或者由固件手动清零。
6	TR1	R/W	0	T1 功能控制位。 0：禁止； 1：使能。
5	TF0	R/W	0	T0 溢出事件显示位。 0：T0 没有溢出； 1：T0 溢出。 该位由中断助力器自动清零，或者由固件手动清零。
4	TR0	R/W	0	T0 功能控制位。 0：禁止； 1：使能。
3..2	Reserved	R	0	
1	IE0	R/W	0	参考 INT0。
0	Reserved	R	0	

IEN0 寄存器（0xA8）

Bit	Field	Type	Initial	说明
7	EAL	R/W	0	中断使能位，参考中断章节。
3	ET1	R/W	0	T1 中断控制位。 0：禁止； 1：使能。
1	ET0	R/W	0	T0 中断控制位。 0：禁止； 1：使能。

TCON0 寄存器（0xE7）

Bit	Field	Type	Initial	说明
7	Reserved	R	0	
6..4	T0RATE[2:0]	R/W	000	T0 外部时钟源的时钟分频控制位。 000：F_{EXT0} / 128； 001：F_{EXT0} / 64； 010：F_{EXT0} / 32； 011：F_{EXT0} / 16； 100：F_{EXT0} / 8； 101：F_{EXT0} / 4； 110：F_{EXT0} / 2； 111：F_{EXT0} / 1；
3	Reserved	R	0	
2..0	T1RATE[2:0]	R/W	000	T1 外部时钟源的时钟分频控制位。 000：F_{EXT1} / 128； 001：F_{EXT1} / 64； 010：F_{EXT1} / 32； 011：F_{EXT1} / 16； 100：F_{EXT1} / 8； 101：F_{EXT1} / 4； 110：F_{EXT1} / 2； 111：F_{EXT1} / 1。

TH0 / TH1 寄存器（TH0: 0x8C, TH1: 0x8D）

Bit	Field	Type	Initial	说明
7..0	TH0/TH1	R/W	0x00	T0 和 T1 的高字节。

TL0 / TL1 寄存器（TL0: 0x8A, TL1: 0x8B）

Bit	Field	Type	Initial	说明
7..0	TL0/TL1	R/W	0x00	T0 和 T1 的低字节。

TMOD 寄存器（0x89）

Bit	Field	Type	Initial	说明
7	T1GATE	R/W	0	T1 gate 模式控制位。 0：禁止； 1：使能，T1 的停止由 INT1 脚控制。
6	T1CT	R/W	0	T1 时钟源选择。 0：F_{Timer1} = Fcpu / 12； 1：F_{Timer1} = Fosc / T1RATE（参考T1RATE）。
5..4	T1M[1:0]	R/W	00	T1 操作模式控制位。 00：13 位向上计数定时器； 01：16 位向上计数定时器； 10：8 位向上计数定时器，支持重装； 11：保留。
3	T0GATE	R/W	0	T0 gate 模式控制位。 0：禁止； 1：使能，T0 时钟源由 INT0 gated。
2	T0CT	R/W	0	T0 时钟源选择。 0：F_{Timer0} = Fcpu / 12； 1：F_{Timer0} = Fosc / T0RATE（参考T0RATE）。
1..0	T0M[1:0]	R/W	00	T0 操作模式控制位。 00：13 位向上计数定时器； 01：16 位向上计数定时器； 10：8 位向上计数定时器，支持重装； 11：独立的 2 个 8 位向上计数定时器。

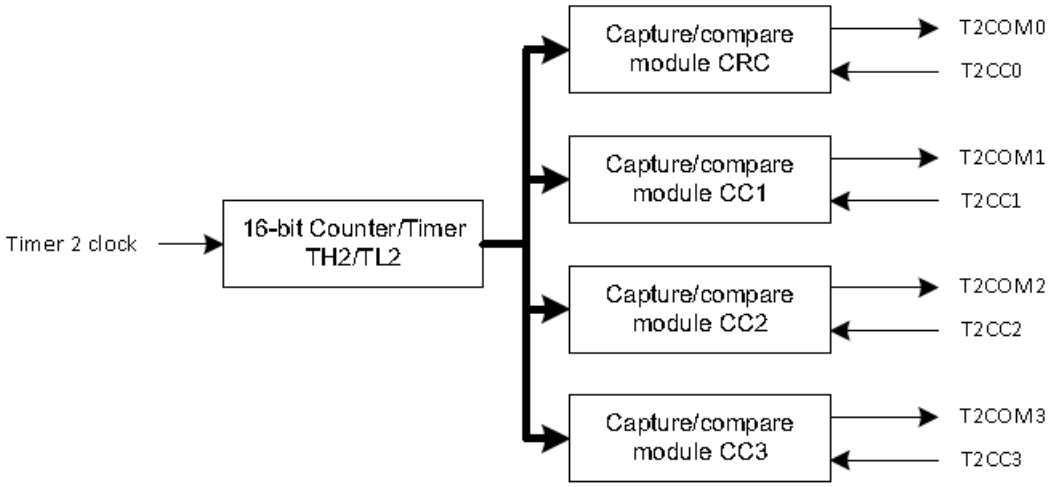
12.7 示例程序代码

下面的示例程序代码显示了在中断下如何执行 T0/T1。

```
#define T0Mode0      (0 << 0)  // T0 mode0, 13-bit counter
#define T0Mode1      (1 << 0) // T0 mode1, 16-bit counter
#define T0Mode2      (2 << 0) // T0 mode2, 8-bit auto-reload counter
#define T0Mode3      (3 << 0) // T0 mode3, T0 two 8-bit counter / T1 no flag
#define T0Gate      (8 << 0) // T0 gating clock by INT0
#define T0ClkFcpu   (0 << 0) // T0 clock source from Fcpu/12
#define T0ClkExt     (4 << 0) // T0 clock source from Fosc or FRTC
#define T0ExtFosc   (0 << 4) // T0 clock source from Fosc
#define T0ExtFRTC   (8 << 4) // T0 clock source from FRTC

#define T1Mode0      (0 << 4)  // T1 mode0, 13-bit counter
#define T1Mode1      (1 << 4) // T1 mode1, 16-bit counter
#define T1Mode2      (2 << 4) // T1 mode2, 8-bit auto-reload counter
#define T1Mode3      (3 << 4) // T1 mode3, T1 stop
#define T1Gate       (8 << 4) // T1 gating clock by INT1
#define T1ClkFcpu    (0 << 4) // T1 clock source from Fcpu/12
#define T1ExtFosc    (4 << 4) // T1 clock source from Fosc

void InitT0T1(void)
{
// T0/T1_Initial
  TH0 = 0x00;
  TL0 = 0x00;
  TH1 = 0x00;
  TL1 = 0x00;
 // T0 mode0 with gating clock by INT0, clock source from Fosc or FRTC
  TMOD | = T0Mode0 | T0GATE | T0ClkExT;
// T0 clock source = FRTC/1;
  TCON0 | = T0ExtFRTC | 0x70;
// T1 mode1, clock source from Fcpu/12
  TMOD | = T1Mode1 | T1ClkFcpu
 // Timer 0/1 enable. Clear TF0/TF1
  TCON | = 0x50
 // Enable T0/T1 interrupt
IEN0 | = 0x5A;
 //Enable total interrupt
  IEN0 | = 0x80;

  P0 = 0x00;
  P0M = 0x03;
}
void T0Interrupt(void) interrupt ISRTimer0   // 0x0B
 {// TF0 clear by hardware
  P00 = ~P00;
 }
void T1Interrupt(void) interrupt ISRTimer1   // 0x1B
{// TF1 clear by hardware
P01 = ~P01;
}
```

13 定时器 T2

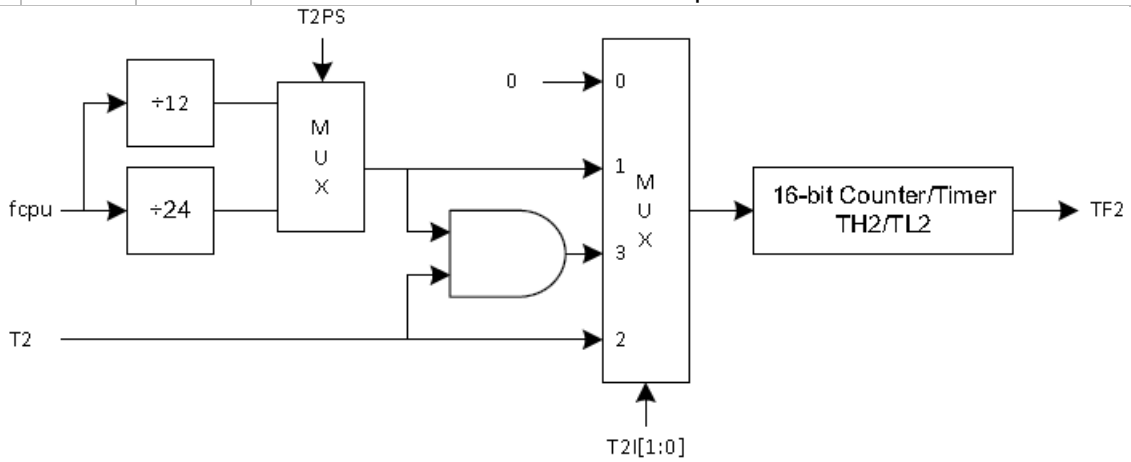
T2 是 16 位向上计数定时器，具有好几个可选的扩展功能：指定重装值，比较器输出（PWM）和捕捉功能。T2 由 1 个 16 位计数/定时器和 4 个 16 位捕捉/比较器模块组成：每个捕捉/比较器模块在使能状态时都有自己相关的 IO 引脚；且每个捕捉/比较器模块都可以作为下列模式独立工作：比较器，上升沿时捕捉，以及写入寄存器时捕捉。



13.1 T2 向上计数控制

T2 共有 3 种操作模式及相应的时钟源：特定的 Fcpu 时钟（Fcpu/12 和 Fcpu/24）；特定的 Fcpu 时钟，带停止控制；和外部时钟输入。下表对 3 种操作模式和相关的寄存器（T2I1，T2I0 和 T2PS）进行了分类，定时器一旦溢出（从 0FFFFH 到 0000H），立即置位 TF2，并由软件进行读/写。T2 的中断功能由 ET2 控制。

T2I1	T2I0	T2PS	T2 时钟源
0	0	X	禁止 T2
0	1	0	fcpu/12
0	1	1	fcpu/24
1	1	0	fcpu/12（T2 引脚为低电平时停止计数，变为高电平后再开始计数）
1	1	1	fcpu/24（T2 引脚为低电平时停止计数，变为高电平后再开始计数）
1	0	X	T2 引脚上升沿（时钟 rate≤ 0.5 * fcpu）

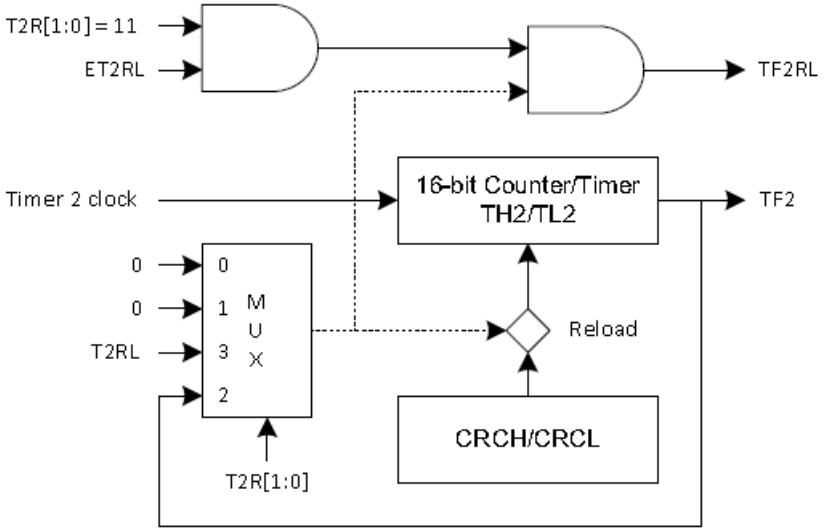


13.2 指定T2 重装值

指定重装值是一项可选择的功能，通过溢出或者外部控制引脚对 T2 计数器进行重装。

若选择溢出重装初值功能后，T2 溢出后，自动复制 CRCH/CRCL 的值到计数器 TH2/TL2。因此，T2 就从 CRCH/CRCL 重复地计数到 0FFFFH。

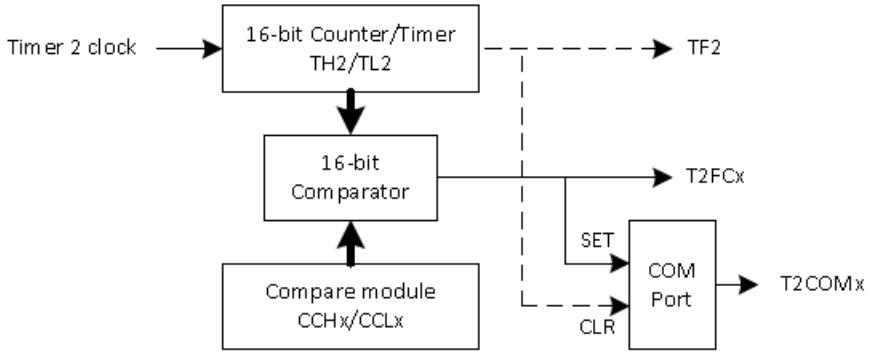
从另一方面来说，外部引脚 T2RL 的下降沿可以选择为重装信号。在这种情况下，若 T2RL 引脚保持稳定，T2 从 0000H 开始计数直到 0FFFFH，但只要 T2RL 引脚有下降信号，CRCH/CRCL 的值随时可能替换计数器的值。然后 T2 继续从 CRCH/CRCL 的值开始计数，若使能外部重装中断（ET2RL 和 ET2 都置 1），则置位外部重装中断标志（TF2RL）。外部中断向量与 T2 中断向量共用，由软件去判断。

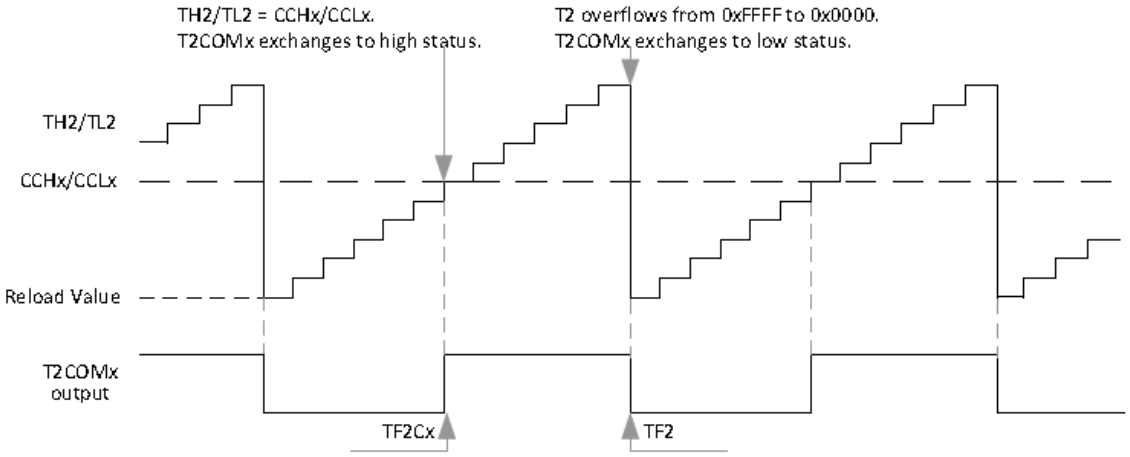


13.3 比较输出（PWM）

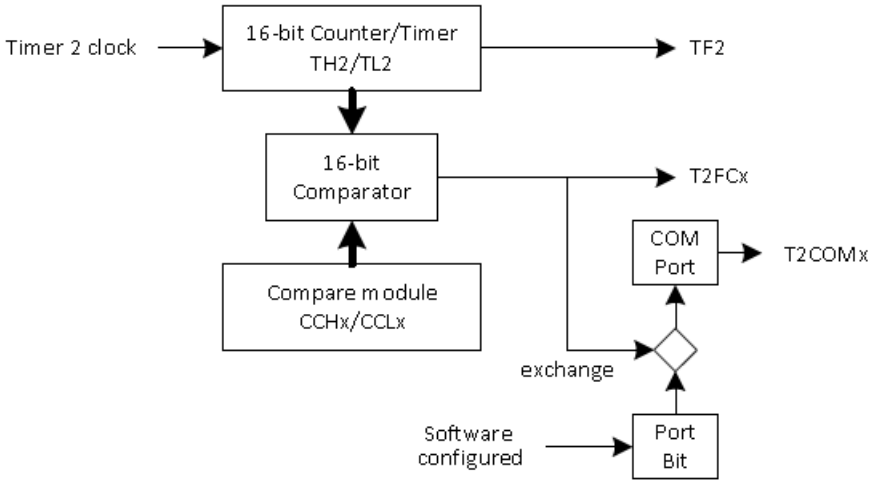
T2 共有 4 组比较输出，每组（CRC/CC1/CC2/CC3）分别与 T2 计数器（TH2/TL2）进行比较，并通过 T2COM0-T2COM3 引脚输出比较结果，比较结果有 2 种输出方式：直接输出和间接输出。

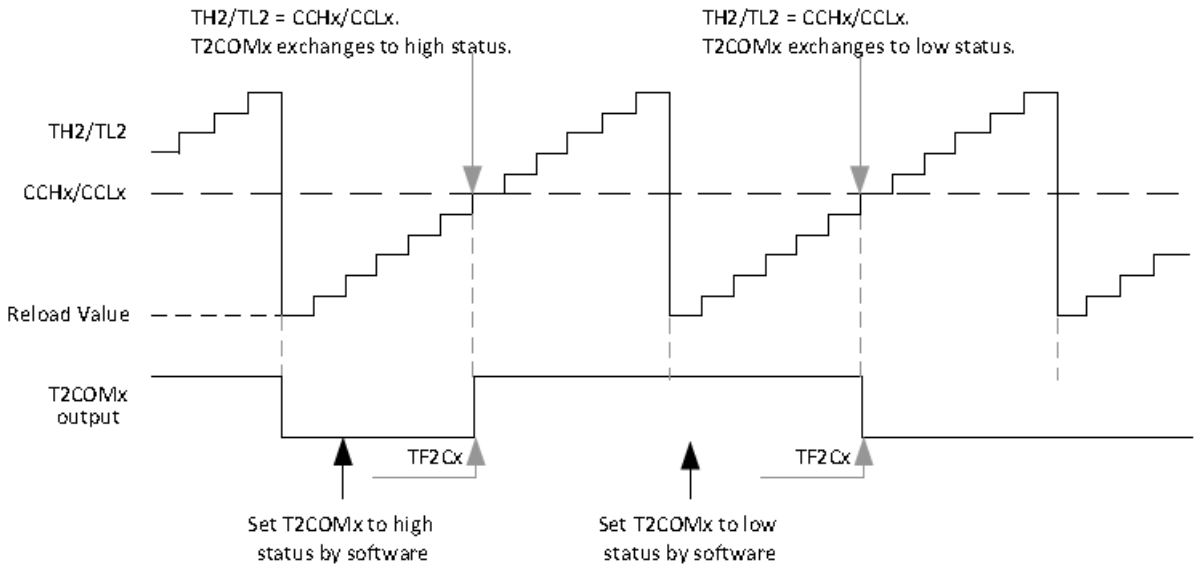
直接输出是指若 CRC/CC1/CC2/CC3 寄存器小于 T2 计数器，其对应的引脚输出低电平；反之若 CRC/CC1/CC2/CC3 大于或等于 T2 计数器则输出高电平。因此其输出状态可在交叉点更改 2 次。CRC/CC1/CC2/CC3 等于 T2 计数器时，给出一个 TF2C0/FT2C1/TF2C2/TF2C3 标志并通过软件进行读/写。比较中断功能由 ET2C0/ET2C1/ET2C2/ET2C3 控制。





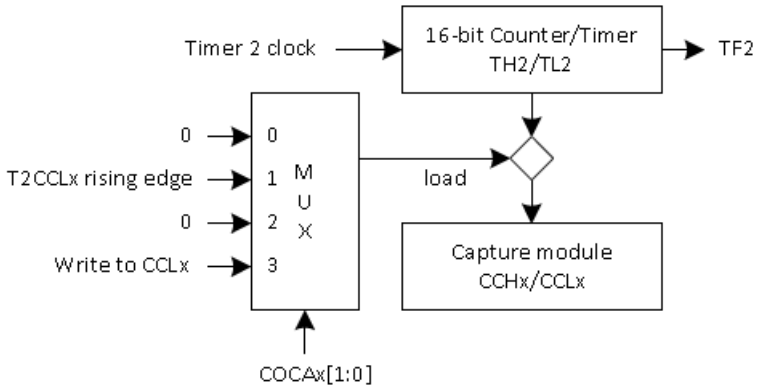
相比之下，间接输出是指保持相对应引脚先前的输出设置状态直到 T2 计数器的值超过 CRC/CC1/CC2/CC3 寄存器的值。在该模式下，可以由软件来控制输出信号的转换，换句话说，就是 TH2/TL2 和 CRC 寄存器的值相等时，P0.0 寄存器位会影响 T2COM0/P0.0 引脚。而 T2 的溢出则不会导致输出变化。



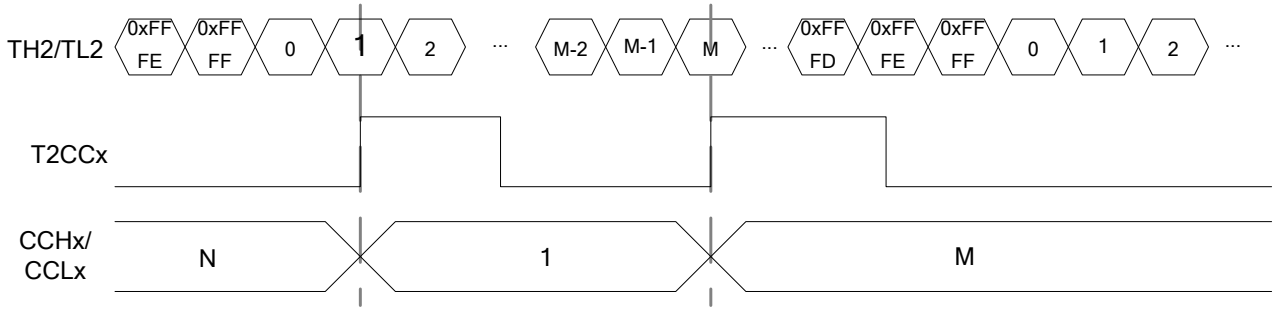


13.4 捕捉功能

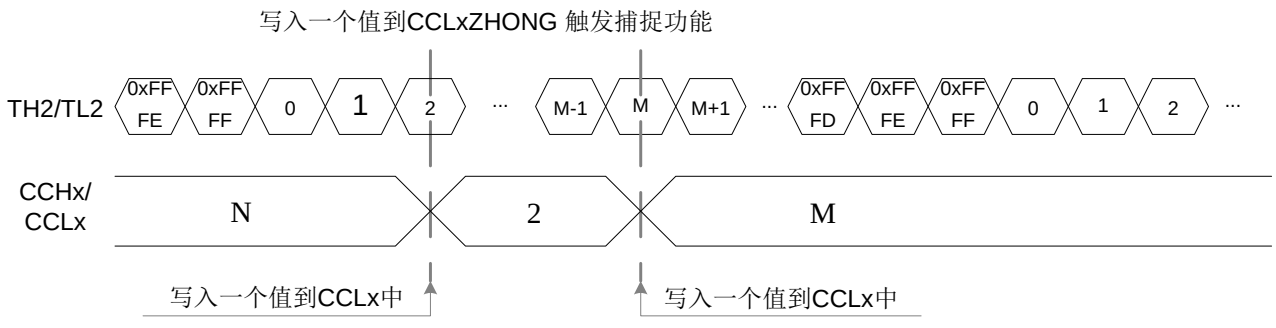
捕捉功能类似于秒表的 split/lap 按键，当 T2 计数器（TH2/TL2）开始向上计数时，一个 split 事件在 CRC/CC1/CC2/CC3 寄存器中记录计数器的值。



split 事件可由硬件或软件提供，T2CC0 引脚可触发硬件 split 事件，复制 TH2/TL2 的值到 CRCH/CRCL 寄存器，而 T2CC1、T2CC2 和 T2CC3 分别控制 CC1-CC3 寄存器。



软件 split 事件由写入 CRCL/CCL1/CCL2/CCL3 寄存器的任何值触发。执行一条写指令到这些寄存器时，TH2/TL2 的当前值会记录在对应的寄存器中。



13.5 T2 寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T2CON	T2PS	I3FR	-	T2R1	T2R0	T2CM	T2I1	T2I0
CCEN	COCA31	COCA30	COCA21	COCA20	COCA11	COCA10	COCA01	COCA00
TH2	TH27	TH26	TH25	TH24	TH23	TH22	TH21	TH20
TL2	TL27	TL26	TL25	TL24	TL23	TL22	TL21	TL20
CRCH	CRCH7	CRCH6	CRCH5	CRCH4	CRCH3	CRCH2	CRCH1	CRCH0
CRCL	CRCL7	CRCL6	CRCL5	CRCL4	CRCL3	CRCL2	CRCL1	CRCL0
CCH3	CCH37	CCH36	CCH35	CCH34	CCH33	CCH32	CCH31	CCH30
CCL3	CCL37	CCL36	CCL35	CCL34	CCL33	CCL32	CCL31	CCL30
CCH2	CCH27	CCH26	CCH25	CCH24	CCH23	CCH22	CCH21	CCH20
CCL2	CCL27	CCL26	CCL25	CCL24	CCL23	CCL22	CCL21	CCL20
CCH1	CCH17	CCH16	CCH15	CCH14	CCH13	CCH12	CCH11	CCH10
CCL1	CCL17	CCL16	CCL15	CCL14	CCL13	CCL12	CCL11	CCL10
IEN0	EAL	-	ET2	ES0	ET1	-	ET0	EX0
IEN1	ET2RL	-	ET2C3	ET2C2	ET2C1	ET2C0	ESPI	EI2C
IRCON	TF2RL	TF2	TF2C3	TF2C2	TF2C1	TF2C0	-	-

T2CON 寄存器（0xC8）

Bit	Field	Type	Initial	说明
7	T2PS	R/W	0	T 前置分频器。 0：Fcpu/12； 1：Fcpu/24。
6	I3FR	R/W	0	比较模式下： 0：T2 的内容与 CRC 寄存器的不相等时发生 COM0 中断； 1：T2 的内容与 CRC 寄存器的相等时发生 COM0 中断。 捕捉模式 0 下： 0：T2CC0 下降沿时，T2 的内容锁存在 CRC 寄存器中； 1：T2CC0 上升沿时，T2 的内容锁存在 CRC 寄存器中。
5	Reserved	R/W	0	
4..3	T2R[1:0]	R/W	00	指定 T2 的重装值。 00：禁止； 01：禁止； 10：通过计数器溢出加载 CRCH/CRCL 的值到 TH2/TL2； 11：通过 T2RL 引脚加载 CRCH/CRCL 的值到 TH2/TL2。
2	T2CM	R/W	0	T2 比较输出： 0：直接输出； 1：间接输出，指定下一个输出状态。
1..0	T2I[1:0]	R/W	00	T2 向上计数控制。 00：禁止； 01：由 T2PS 定义时钟频率； 10：时钟源来自 T2 引脚； 11：由 T2PS 定义时钟频率，带 T2 引脚控制定时停止。

CCEN 寄存器（0xC1）

Bit	Field	Type	Initial	说明
7..6	COCA3[1:0]	R/W	00	CC3 的比较和捕捉功能。 00：禁止； 01：T2CC3 引脚的上升沿捕捉； 10：比较功能； 11：写入 CCL3 寄存器捕捉。
5..4	COCA2[1:0]	R/W	00	CC2 的比较和捕捉功能。 00：禁止； 01：T2CC2 引脚的上升沿捕捉； 10：比较功能； 11：写入 CCL2 寄存器捕捉。
3..2	COCA1[1:0]	R/W	00	CC1 的比较和捕捉功能。 00：禁止； 01：T2CC1 引脚的上升沿捕捉； 10：比较功能； 11：写入 CCL1 寄存器捕捉。
1..0	COCA0[1:0]	R/W	00	CC0 的比较和捕捉功能。 00：禁止； 01：T2CC0 引脚的上升沿捕捉； 10：比较功能； 11：写入 CCL0 寄存器捕捉。

TH2/TL2 寄存器（TH2: 0xCD, TL2: 0xCC）

Bit	Field	Type	Initial	说明
7..6	TH2 /TL2	R/W	0x00	T2 16 位计数寄存器。

CRC 寄存器（CRCH: 0xCB, CRCL: 0xCA）

Bit	Field	Type	Initial	说明
7..6	CRCH[15:0]	R/W	0x00	16 位比较/捕捉寄存器。

CCH3/CCL3 寄存器（CCH3: 0xC7, CCL3: 0xC6）

Bit	Field	Type	Initial	说明
7..6	CCH3/CCL3	R/W	0x00	16 位比较/捕捉寄存器。

CCH2/CCL2 寄存器（CCH2: 0xC5, CCL2: 0xC4）

Bit	Field	Type	Initial	说明
7..6	CCH2 /CCL2	R/W	0x00	16 位比较/捕捉寄存器。

CCH1/CCL1 寄存器（CCH1: 0xC3, CCL1: 0xC2）

Bit	Field	Type	Initial	说明
7..6	CCH1/CCL1	R/W	0x00	16 位比较/捕捉寄存器。

IEN0 寄存器（0xA8）

Bit	Field	Type	Initial	说明
7	EAL	R/W	0	使能中断，请参考中断章节。
5	ET2	R/W	0	使能 T2 中断。
Else				请参考其它章节。

IEN1 寄存器（0xB8）

Bit	Field	Type	Initial	说明
7	ET2RL	R/W	0	T2 定时器外部重装中断控制位。 0：禁止； 1：使能。
6	Reserved	R	0	
5	ET2C3	R/W	0	T2 定时器 COM3 中断控制位。 0：禁止； 1：使能。
4	ET2C2	R/W	0	T2 定时器 COM2 中断控制位。 0：禁止； 1：使能。
3	ET2C1	R/W	0	T2 定时器 COM1 中断控制位。 0：禁止； 1：使能。
2	ET2C0	R/W	0	T2 定时器 COM0 中断控制位。 0：禁止； 1：使能。
Else				请参考其它章节。

13.6 示例程序代码

下面的示例程序代码显示了在中断下如何执行 T2 的比较功能。

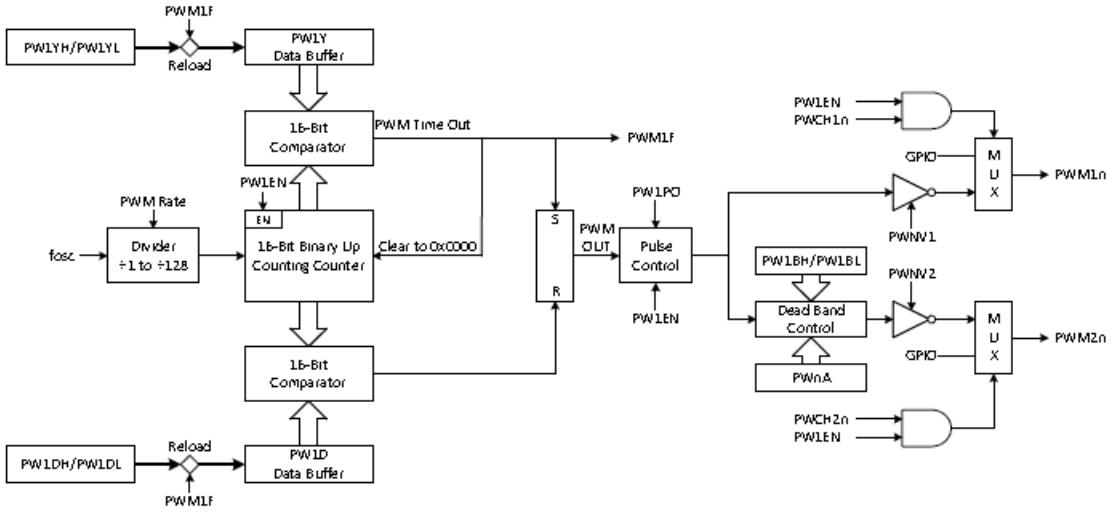
```
#define T2ClkFcpu   (1 << 0) // T2 clock from Fcpu
#define T2ClkPin    (2 << 0) // T2 clock from T2 pin
#define T2ClkGate   (3 << 0) // T2 clock from Fcpu with T2 pin gating
#define T2Fcpu12    (0<<7) // T2 clock = Fcpu/12
#define T2Fcpu24    (1 << 7) // T2 clock = Fcpu/24
#define T2RLMMode0   (2<<3)  // T2 reload mode0 = auto-reload
#define T2RLMMode1   (3 << 3)  // T2 reload mode0 = T2RL falling edge trigger
#define ComMode0     (0 << 2)  // Compare mode = directly method
#define ComMode1     (1 << 2)  // Compare mode = indirectly output method
#define T2COM0EdNe    (0 << 6) // T2COM0 interrupt edge = no equle CRC
#define T2COM0Ede    (1 << 6) // T2COM0 interrupt edge = equle CRC
#define T2COM0En     (2 << 0) // T2COM0 compare function enable
#define T2COM1En     (2 << 2) // T2COM1compare function enable
#define T2COM2En     (2 << 4) // T2COM2compare function enable
#define T2COM3En     (2 << 6) // T2COM3compare function enable

void InitT2(void)
{
// T2_Initial
TH2 = 0x00;
  TL2 = 0x00;
  CRCH = 0x80;
  CRCL = 0x00;
  CCH1 = 0xC0;
  CCL1 = 0x00;
  CCH2 = 0xE0;
  CCL2 = 0x00;
  CCH3 = 0xF0;
  CCL3 = 0x00;

 // T2 clock from Fcpu/24 with T2 pin gating
// Reload mode1 = T2RL falling edge trigger
// Compare mode = directly method
// T2COM0 interrupt trigger = equle CRC
  T2CON | = T2ClkGate | T2Fcpu24 | T2RLMode1 | ComMode0 | T2COM0EdE;

 // Compare function T2COM0/1/2/3 enable
CCEN| = T2COM0En | T2COM1En | T2COM2En | T2COM3En;

 //P07(T2)/P20(T2RL) is input mode with pull-high resister
  P0M &= 0x7F;
  P2M &= 0xFE;
  P0UR&= 0x80;
  P2UR&= 0x01;

// Enable T2RL/T2COM0/1/2/3 interrupt
  IEN1| = 0xBC;
// Enable total/Timer2 interrupt
  IEN0 | = 0xA0;

  P0 = 0x00;
 }
void T2Interrupt(void) interrupt ISRTimer2// 0x2B
 {// TF2/TF2RL clear by software
If ((IRCON & 0x40) ==0x40){
   IRCON & = 0xBF;  //Clear TF2
   P00 = ~P00;
 }
```

```
If ((IRCON & 0x80 == 0x80) {
  IRCON & = 0x7F;  //Clear TF2RL
   P01 = ~P01;
  }
}
Void T2COM0Interrupt(void) interrupt ISRCom1   // 0x53
{// TF2C0 clear by hardware
   P02 = ~P02;
}
 Void T2COM1Interrupt(void) interrupt ISRCom2   // 0x5B
{// TF2C1 clear by hardware
   P03 = ~P03;
}
 Void T2COM2Interrupt(void) interrupt ISRCom3   // 0x63
{// TF2C2 clear by hardware
   P04 = ~P04;
}
 Void T2COM3Interrupt(void) interrupt ISRCom4   // 0x6B
{// TF2C3 clear by hardware
   P05 = ~P05;
}
```

14 PWM

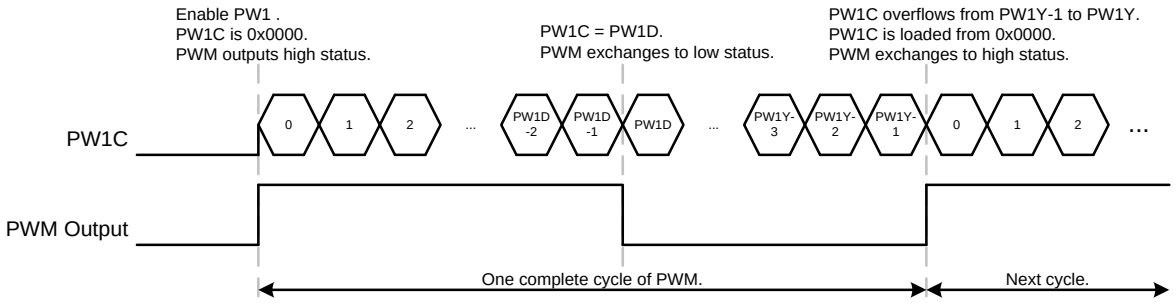
PW1 定时器包含一个 16 位二进制 4 通道 PWM，一个单脉冲 PWM。计数器计数到上限值（PW1Y）后，计数器清零并触发一个中断信号。PWM 的占空比周期由 PW1D 控制。

PWM 还支持单脉冲输出信号，在首个 PWM 周期结束时自行禁止，因此，在这种情况下只产生单个脉冲。PWM 共有 4 个可编程控制的 PWM 通道，与 GPIO 引脚共用，由 PW1CH[5:4, 1:0]位控制。通过使能 PW1CH[5:4, 1:0]的对应位/通道执行输出操作，使能的 PWM 通道从 GPIO 切换到 PWM 输出。禁止 PW1CH[5:4, 1:0]位时，PWM 通道返回到上一个 GPIO 模式。若使能中断，PW1 内置 IDEL 模式唤醒功能。PWM 定时器溢出时（PW1Y-1 到 PW1Y），立即释放 PWM1F，提供软件进行读/写。由 EPWM1 控制 PW1 的中断功能。



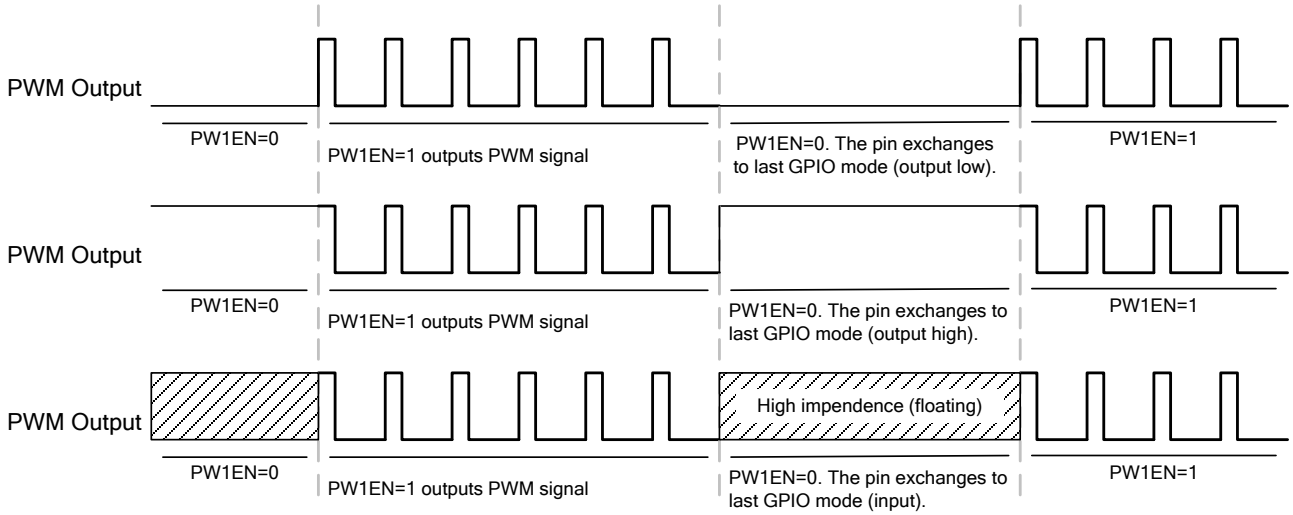
14.1 普通PWM

PW1 定时器内置 PWM 功能，由 PWnEN 和 PW1CH[5:4, 1:0]位控制，PWM10，PWM11，PWM20，PWM21 为输出引脚，并与 GPIO 引脚共用，由 PW1CH[5:4, 1:0]控制。输出 PWM 时，必须设置 PW1EN=1。PWM 输出信号同步完成后，PWM 通道从 GPIO 模式切换到 PWM 输出。PW1EN=0 时，PWM 通道返回上一个 GPIO 模式。PW1Y 和 PW1D 进行比较，其比较结果产生 PWM 信号。PW1C 从 0000H 开始计数时，PWM 输出高电平，即 PWM 的初始状态。PW1C 加载 PW1Y 寄存器的值决定 PWM 的周期和分辨率。PW1C 继续计数，系统比较 PW1C 和 PW1D 的值，PW1C=PW1D 时，PWM 输出低电平，PW1C 继续计数，PW1 定时器溢出（PWnY-1 到 0000H）后，PWM 完成一个信号周期。PW1C 自动重新加载 0000H，并输出高电平以等待下一个周期。PW1D 决定高电平占空比时间，PW1Y 决定 PWM 的分辨率和周期。PW1D 的值不能大于 PW1Y 的值，否则 PWM 信号会出错。PWM 的时钟源为 Fosc，由 PW1RATE[2:0]控制：000=Fosc/128，000=Fosc/64，000=Fosc/32，000=Fosc/16，000=Fosc/8，000=Fosc/4，000=Fosc/2，000=Fosc/1。



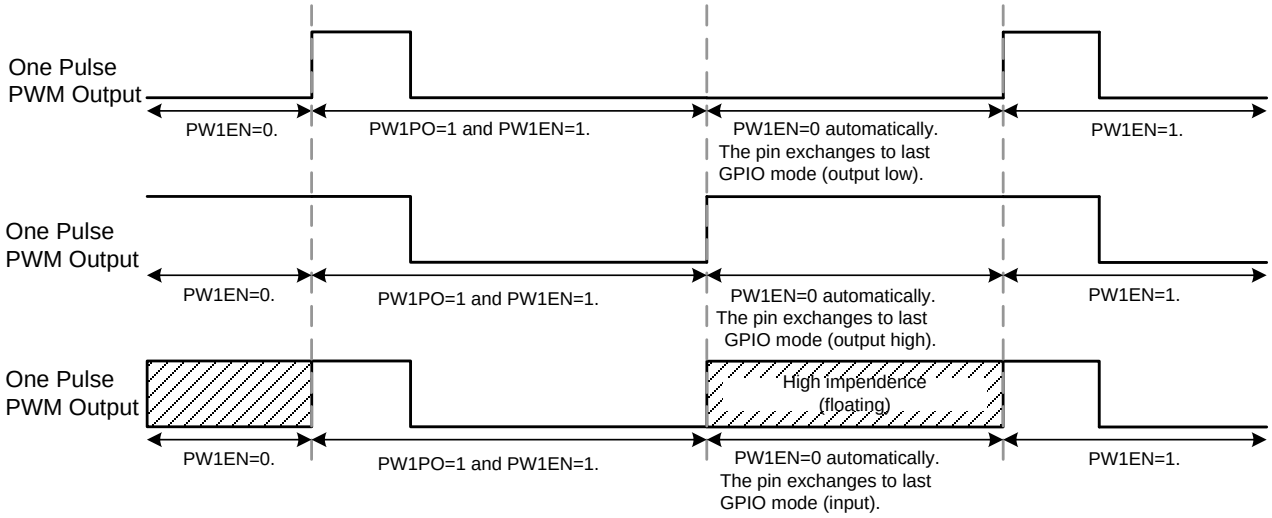
PWM 周期 = PW1Y

PWM 占空比 = (PW1D):(PW1Y-PW1D)



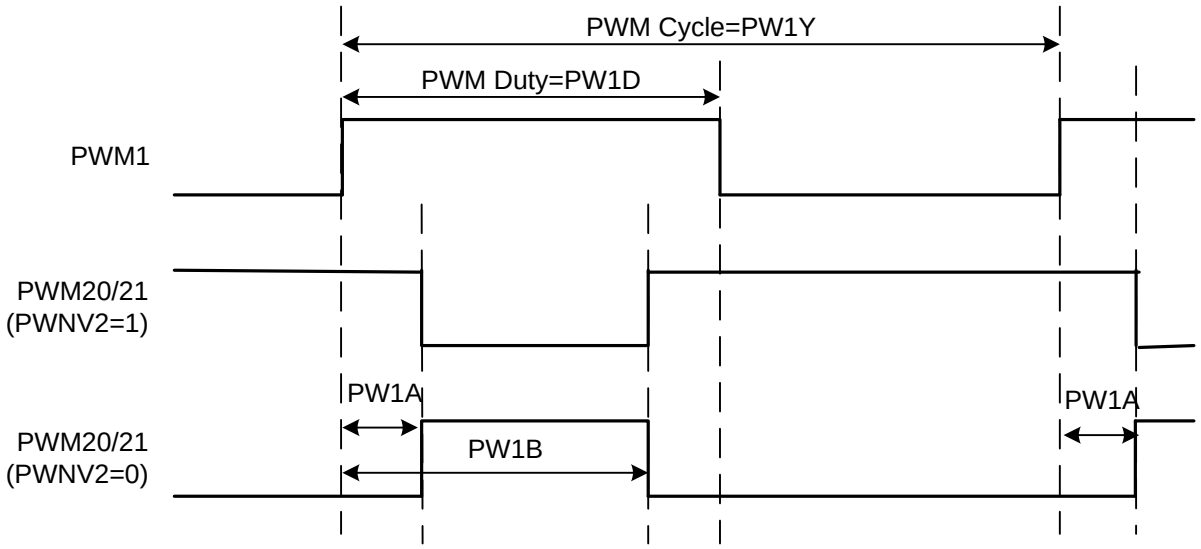
14.2 单脉冲PWM

PW1PO = 0 时，PW1 为 PWM 功能模式。PW1PO = 1 且 PW1EN=1 时，PW1 将输出单脉冲 PWM，同时随着 PW1 计数器溢出而置位 PWM1F。PW1EN 自动清零，脉冲输出引脚返回到上一个 GPIO 状态。若要输出下一个脉冲，则要通过程序重新设置 PW1EN 位为 1。单脉冲 PWM 通道由 PW1CH[5:4, 1:0]位进行选择，PWM 输出引脚 PWM10、PWM11，PWM20，PWM21 与 GPIO 引脚共用，由 PW1CH[5:4, 1:0]选择控制。输出单脉冲 PWM 时，必须设置 PW1PO=PW1EN=1；PWM 输出信号同步完成后，PWM 通道从 GPIO 模式切换到 PWM 输出。单脉冲 PWM 输出完成后，PW1EN=0 时，PWM 通道返回上一个 GPIO 模式。



14.3 死区

PWM 内置反向输出功能，PWNV=1 时，输出反向 PWM 信号；PWNV=0 时，则输出正常 PWM 信号。反向 PWM 输出波形如下图所示：



PWM 的死区是在 PWM 高脉冲宽度区设置的，其死区周期由 PW1A 和 PW1D-PW1B 寄存器编程控制。PWM 两边的死区时间可以设置为对称或不对称。如死区周期的长度大于 PWM 的占空比，就不输出 PWM。

14.4 PWM寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1M	PW1EN	PW1Rate2	PW1Rate1	PW1Rate0	PWNV2	PWCH1	-	PW1PO
PW2CH	-	-	PWCH21	PWCH20	-	-	PWCH11	PWCH10
PW1YH	PW1Y15	PW1Y14	PW1Y13	PW1Y12	PW1Y11	PW1Y10	PW1Y9	PW1Y8
PW1YL	PW1Y7	PW1Y6	PW1Y5	PW1Y4	PW1Y3	PW1Y2	PW1Y1	PW10
PW1BH	PW1B15	PW1B14	PW1B13	PW1B12	PW1B11	PW1B10	PW1B9	PW1B8
PW1BL	PW1B7	PW1B6	PW1B5	PW1B4	PW1B3	PW1B2	PW1B1	PW1B0
PW1DH	PW1D15	PW1D14	PW1D13	PW1D12	PW1D11	PW1D10	PW1D9	PW1D8
PW1DL	PW1D7	PW1D6	PW1D5	PW1D4	PW1D3	PW1D2	PW1D1	PW1D0
PW1A	PW1A7	PW1A6	PW1A5	PW1A4	PW1A3	PW1A2	PW1A1	PW1A0

PWM 寄存器（PW1M: 0xAB）

Bit	Field	Type	Initial	说明
7	PW1EN	R/W	0	PW1 功能控制位。 0：禁止； 1：使能。
6..4	PW1RATE	R/W	000	PWM 定时器时钟源。 000：Fosc / 128； 001：Fosc / 64； 010：Fosc / 32； 011：Fosc / 16； 100：Fosc / 8； 101：Fosc / 4； 110：Fosc / 2； 111：Fosc / 1。
3	PWNV2	R/W	0	PWM20/21 引脚输出控制位。 0：正常输出； 1：反向输出。
2	PWNV1	R/W	0	PWM10/11 引脚输出控制位。 0：正常输出； 1：反向输出。
0	PW1PO	R/W	0	单脉冲功能。 0：禁止； 1：使能。

PW1CH 寄存器（0xBE）

Bit	Field	Type	Initial	说明
7..6	Reserved	R/W	0	
5 4	PWCH21 PWCH20	R/W	0	PWM1 共用引脚控制位。 0：GPDIO； 1：PWM 输出。
3..2	Reserved	R/W	0	
1 0	PWCH11 PWCH10	R/W	0	PWM1 共用引脚控制位。 0：GPDIO； 1：PWM 输出。

PW1YH/PW1YL 寄存器（PW1YH: 0xAD, PW1YL: 0xAC）

Bit	Field	Type	Initial	说明
7..0	PW1YH/L	R/W	0x00	16 位 PWM1 周期控制位。

PW1DH/PW1DL 寄存器（PW1DH: 0xBC, PW1DL: 0xBB）

Bit	Field	Type	Initial	说明
7..0	PW1DH/L	R/W	0x00	16 位 PWM1 占空比控制位。

PW1BH/PW1BL 寄存器（PW1BH: 0xAF, PW1BL: 0xAE）

Bit	Field	Type	Initial	说明
7..0	PW1BH/L	R/W	0x00	16 位 PWM1 死区控制位。

PW1A 寄存器（PW1A: 0xBD）

Bit	Field	Type	Initial	说明
7..0	PW1AH	R/W	0x00	8 位 PWM1 死区控制位。

IEN0 寄存器（0XA8）

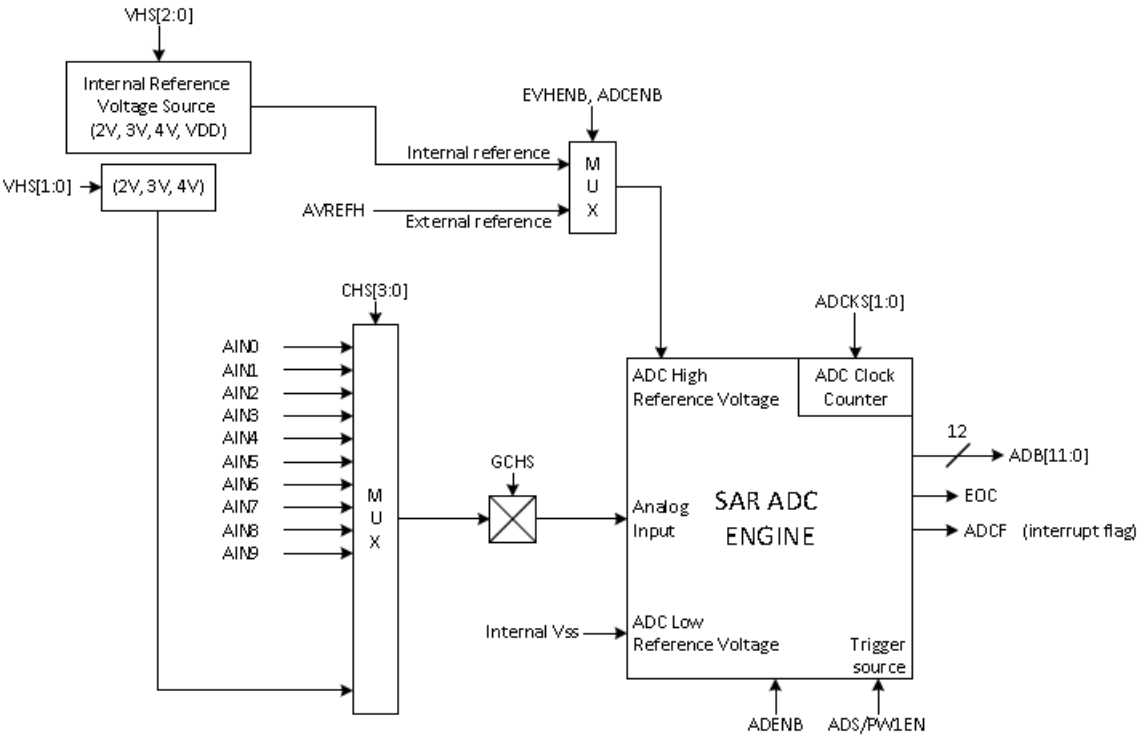
Bit	Field	Type	Initial	说明
7	EAL	R/W	0	所有中断使能控制位。 0：禁止； 1：使能。
Else				参考其它章节。

IEN4 寄存器（0XD1）

Bit	Field	Type	Initial	说明
7	EPWM1	R/W	0	PWM1 中断控制位。 0：禁止； 1：使能。
3	PWM1F	R/W	0	PWM1 中断请求标志位。 0：无 PWM1 中断请求； 1：PWM1 请求中断。
Else	Reserved	R	0	

15 ADC

模拟数字转换（ADC）是一个 SAR 结构，内置 10 个输入通道（AIN0~AIN11），高达 4096 阶的分辨率，能将一个模拟信号转换成相应的 12 位数字信号。ADC 内置的 10 个模拟通道可以测量 10 种不同的模拟信号源，ADC 的分辨率为 12 位。ADC 共有 4 种时钟 rate 来决定 ADC 的转换速率。ADC 参考高电压包括 5 种：4 种内部参考源（Vdd、4V、3V 和 2V）和外部参考源（由 AVREFH 引脚提供）。ADC 内置 P2CON/P1CON 寄存器来设置模拟输入引脚。设置好 ADENB 和 ADS 位后，ADC 开始转换。除 ADS 位可以开始转换模拟信号外，PW1EN/PW2EN/PW3EN 也可以转换模拟信号。ADC 可在 IDLE 模式下工作，ADC 结束后，若使能中断，则将系统从绿色模式下唤醒进入普通模式。



15.1 操作配置

ADC 开始转换前必须先设置好下列配置，具体如下：

－选择和使能 ADC 输入通道（由 CHS[3:0]位和 GCHS 位设置）；

－将 ADC 输入通道设置为输入模式（由 PnM 寄存器设置）；

－必须禁止 ADC 输入通道的内部上拉电阻（由 PnUR 寄存器设置）；

－必须将 ADC 模拟输入通道的控制位设置为 1（由 PnCON 寄存器设置）；

－选择 ADC 高参考电压和低参考电压（由 VREFH 寄存器设置）；

－选择 ADC 时钟频率（由 ADCKS[1:0]位设置）；

－设置 ADENB 位后，ADC 准备开始转换。

15.1.1 开始转换

由 ADENB 位使能 ADC 功能时，要保证由程序启动 ADC。除 ADS 位可以开始转换模拟信号外，PW1EN 也可以转换模拟信号。下列情形可使转换初始化：

－写入 1 到 ADM 寄存器的 ADS 位；

－ADPWS 位为 1 时使能 PWM1；

设置 ADENB 和 ADS 位后，ADC 开始转换。转换结束后，ADS 清零，ADC 电路将 EOC 和 ADCIRQ 置 1，并将转换结果存入 ADB 和 ADR 寄存器中。若使能 ADC 中断（EADC=1），ADC 请求中断，ADC 完成后，ADCF=1 时执行中断服务程序。中断时必须由程序将 ADCF 清零。

注意，ADPWS=1 时，若使能 PWM 作为转换源，则 ADC 继续转换直至禁止 PWM。

15.2 ADC输入通道

SN8F5702 的 ADC 内置 10 个输入通道（AIN0~AIN9），用于测量 10 种不同的模拟信号源，由 CHS[3:0]和 GCHS 位控制。AIN10 则是 2V/3V/4V 的输入通道。外部没有输入引脚，ADC 参考电压必须为内部 VDD 或外部电压，不能是内部 2V/3V/4V。在电池应用中，AIN10 可作为很好的电池检测器。选择一个合适的 AVREFH 值和比较值，系统可内置一个高性能、廉价的电池检测器。TCHEN=1 时，CT0 – CT8 通道用于电容式传感功能。

ADC 输入通道

CHS[3:0]	Channel	Pin name	备注
0000	AIN0	P1.0	
0001	AIN1	P1.1	
0010	AIN2	P1.2	
0011	AIN3	P1.3	
0100	AIN4	P1.4	
0101	AIN5	P1.5	
0110	AIN6	P1.6	
0111	AIN7	P1.7	
1000	AIN8	P2.1	
1001	AIN9	P2.0	
1010	AIN10	Internal 2V or 3V or 4V	电池检测器通道
1011 – 1111	-	-	保留

15.2.1 引脚配置

ADC 输入通道引脚与 P1 和 P2 GPIO 引脚共用，由 CHS[3:0]选择控制。在同一时间，只能设置 P1 和 P2 其中的一个引脚为 ADC 输入通道引脚，P1 和 P2 的其它引脚必须设置为输入引脚，禁止上拉电阻，并先要由程序使能 P1CON/P2CON。通过 CHS[3:0]选择 ADC 输入通道后，GCHS 位设置为 1，使能 ADC。

ADC 输入引脚与 GPIO 引脚共用，当输入一个模拟信号到 CMOS 结构端口时，尤其当模拟信号为 1/2 VDD 时，可能产生额外的漏电流。当 P1 和 P2 输入多个模拟信号时，也会产生额外的漏电流。睡眠模式下，上述漏电流会严重影响到系统的整体功耗。将 PnCON 置 1，其对应的引脚将被设为纯模拟信号输入引脚，从而避免上述漏电流的产生。

注意：ADC 引脚为 GPIO 引脚时，P1CON/P2CON 必须置 0，否则 GPIO 引脚的信号会被隔离。

15.3 参考电压

ADC 内置 5 种高参考电压，由 VREFH 寄存器控制：包括 1 个外部参考电压和 4 个内部参考源（VDD、4V、3V、2V）。EVHENB = 1 时，ADC 参考电压由外部参考源提供（AVREFH/P1.0），此时必须设置 P1.0 为输入模式并禁止上拉电阻。

EVHENB = 0 时，ADC 参考电压由内部参考源提供，并由 VHS[1:0]选择控制。VHS[1:0] = 11 时，ADC 参考源选择 VDD；VHS[1:0] = 10 时，ADC 参考源选择 4V； VHS[1:0] = 01 时，ADC 参考源选择 3V；VHS[1:0] = 00 时，ADC 参考源选择 2V。外部参考源的限制条件为，最高为 VDD，最低为内部最低电平，否则默认为 VDD。若选择 AIN14 为 2V/3V/4V 的输入通道，而不从外部输入，这样 ADC 的高参考电压必须是内部 VDD 或者外部参考电源，不是内部 2V/3V/4V。

15.3.1 信号格式

ADC 采样电压范围为参考电压高/低电平之间。ADC 参考低电压为 VSS（不可更改）；高电压包括 VDD/4V/3V/2V 和外部参考电压（由 P1.0/AVREFH 引脚提供），由 EVHENB 控制。ADC 参考电压的范围为：**（ADC 参考高电压-ADC 参考低电压）≧ 2V**。ADC 参考低电压范围为 VSS=0V，故 ADC 参考高低压范围为 2V~VDD，外部参考高电压也要在此范围内。

－ADC 内部参考低电压=0V。
－ADC 内部参考高低压=VDD/4V/3V/2V。（EVHENB = 0）
－ADC 外部参考高电压= 2V~VDD。（EVHENB = 1）

ADC 采样输入信号电压必须在 ADC 参考低电压和 ADC 参考高电压之间，若 ADC 输入信号的电压不在此范围内，则 ADC 的转换结果会出错（满量程或者为 0）。

－ADC 参考低电压≦ ADC 采样输入信号电压≦ ADC 参考高电压

15.4 转换时间

ADC 转换时间是指从 ADS=1（开始 ADC）到 EOC=1（ADC 结束）所用的时间，由 ADC 时钟频率控制，12 位 ADC 的转换时间为 1/（ADC 时钟/4）*16 S。ADC 的时钟源为 Fosc，包括 Fosc/1，Fosc/2，Fosc8，Fosc/16，由 ADCKS[1:0]位控制。

ADC 的转换时间会影响 ADC 的性能，如果输入高频率的模拟信号，必须要选择一个高频率的 ADC 转换时钟。如果 ADC 的转换时间比模拟信号的转换频率慢，则 ADC 的结果出错。故选择合适的 ADC 时钟频率和 ADC 分辨率才能得到合适的 ADC 转换频率。

$$12\text{ 位 ADC 转换时间} = \frac{16}{\text{ADC 时钟 rate}/4}$$

ADC 转换时间

ADCKS[1:0]	ADC 时钟 rate	fosc = 16MHz		fosc = 32MHz	
		转换时间	转换 rate	转换时间	转换 rate
00	fosc/16	1/(16MHz/16/4)*16 = 64us	15.625kHz	1/(32MHz/16/4)*16 = 32us	31.25kHz
01	fosc/8	1/(16MHz/8/4)*16 = 32us	31.25kHz	1/(32MHz/8/4)*16 = 16us	62.5kHz
10	fosc	1/(16MHz/4)*16 = 4us	250kHz	1/(32MHz/4)*16 = 2us	500kHz
11	fosc/2	1/(16MHz/2/4)*16 = 8us	125kHz	1/(32MHz/2/4)*16 = 4us	250kHz

15.5 数据缓存器

ADC 数据缓存器共 12 位，用来存储 AD 转换结果，8 位只读寄存器 ADB 存放结果的高字节（bit4~bit11），ADR（ADR[3:0]）存放低字节（bit0~bit3）。ADC 数据缓存器是只读寄存器，系统复位后处于未知状态。

AIN 输入电压 vs. ADB 输出数据

AIN n	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
0/4096*VREFH	0	0	0	0	0	0	0	0	0	0	0	0
1/4096*VREFH	0	0	0	0	0	0	0	0	0	0	0	1
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
4094/4096*VREFH	1	1	1	1	1	1	1	1	1	1	1	0
4095/4096*VREFH	1	1	1	1	1	1	1	1	1	1	1	1

15.6 ADC寄存器

ADC 寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	-	CHS3	CHS2	CHS1	CHS0
ADB	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4
ADR	-	GCHS	ADCKS1	ADCKS0	ADB3	ADB2	ADB1	ADB0
VREFH	EVHENB	-	-	ADPWS	-	VHS2	VHS1	VHS0
P1CON	P1CON7	P1CON6	P1CON5	P1CON4	P1CON3	P1CON2	P1CON1	P1CON0
P2CON	-	-	-	-	-	-	P2CON1	P2CON0
IEN2	-	-	-	-	-	-	EADC	-
IRCON2	-	-	-	-	-	-	-	ADCF

ADM 寄存器（0xD2）

Bit	Field	Type	Initial	说明
7	ADENB	R/W	0	ADC 控制位，STOP 模式下，禁止 ADC 以省电。 0：禁止；1：使能。
6	ADS	R/W	0	ADC 转换控制位。 写 1：开始 AD 转换（转换结束后自动清零）。
5	EOC	R/W	0	ADC 状态位。 0：AD 转换过程中； 1：AD 转换结束（硬件自动设置为 1，固件手动清零）。
3..0	CHS[3:0]	R/W	0x00	ADC 输入通道选择位。 0000：CT0/AIN0；0001：CT1/AIN1； 0010：CT2/AIN2；0011：CT3/AIN3； 0100：CT4/AIN4；0101：CT5/AIN5； 0110：CT6/AIN6；0111：CT7/AIN7； 1000：CT8/AIN8；1001：CT9/AIN9； 1010：CT10/AIN10*(1)；others：保留；

*(1) AIN10 为内部 2V/3V/4V 输入通道，没有外部信号输入，ADC 参考电压必须为内部 VDD 或外部电压，不能是内部 2V/3V/4V。

ADB 寄存器（0xD3）

Bit	Field	Type	Initial	说明
7..0	ADB[11:4]	R	-	12 位 AD 转换结果的高字节*。

* ADC 数据缓存器的长度为 12 位，用于存储 AD 转换结果，其高字节存入 ADB 寄存器，低字节存入 ADR[3:0]位。

ADR 寄存器（0xD4）

Bit	Field	Type	Initial	说明
6	GCHS	R/W	0	ADC 全局通道控制位。 0：禁止； 1：使能。
5..4	ADCKS[1:0]	R/W	00	ADC 时钟源选择位。 00 = Fosc/16；01 = Fosc/8；10 = Fosc/1；11 = Fosc/2。
3..0	ADB[3:0]	R	-	12 位 AD 转换结果的低字节*。

* ADC 数据缓存器的长度为 12 位，用于存储 AD 转换结果，其高字节存入 ADB 寄存器，低字节存入 ADR[3:0]位。

VREFH 寄存器（0xD5）

Bit	Field	Type	Initial	说明
7	EVHENB	R/W	0	ADC 内部参考高电压控制位。 0：使能 ADC 内部 VREFH 功能，AVREFH/P1.0 为 GPIO 引脚。 1：禁止 ADC 内部 VREFH 功能，AVREFH/P1.0 为外部 AVREFH*(1)输入引脚。
4	ADPWS	R/W	0	PWM 触发 ADC 开始控制位。 0：禁止 PWM 触发 ADC 开始； 1：使能 PWM 触发 ADC 开始。
2	VHS[2]	R/W	0	ADC 内部参考高低压选择位（AIN10 为内部 2V/3V/4V 输入通道）。 0：由VHS[1:0]*(2)选择内部VREFH功能； 1：ADC 内部 VREFH 功能为内部 VDD。
1..0	VHS[1:0]	R/W	00	ADC 内部参考高低压选择位。 00：2.0V； 01：3.0V； 10：4.0V； 11：VDD。

*(1) AVREFH 的电压值必须在 VDD 到 2.0V 范围内。

*(2) AIN10 为内部 2V/3V/4V 输入通道，没有外部信号输入，ADC 参考电压必须为内部 VDD 或外部电压，不能是内部 2V/3V/4V。

P1CON 寄存器（0xD6）

Bit	Field	Type	Initial	说明
7..0	P1CON[7:0]	R/W	0x00	P1 配置控制位*。 0：P1 可作为模拟输入引脚（ADC 输入引脚）或者数字 GPIO 引脚； 1：P1 只能作为模拟输入引脚。

* P1CON [7:0]配置相关的 P1 引脚为纯模拟输入引脚以避免漏电流。

P2CON 寄存器（0x9E）

Bit	Field	Type	Initial	说明
1..0	P2CON[1:0]	R/W	0x0	P2 配置控制位*。 0：P2 可作为模拟输入引脚（ADC 输入引脚）或者数字 GPIO 引脚； 1：P2 只能作为模拟输入引脚。

* P2CON [7:0]配置相关的 P2 引脚为纯模拟输入引脚以避免漏电流。

IEN2 寄存器（0x9A）

Bit	Field	Type	Initial	说明
1	EADC	R/W	0	ADC 中断控制位。 0：禁止； 1：使能。
Else				请参考其它章节。

IRCON2 寄存器（0xBF）

Bit	Field	Type	Initial	说明
0	ADCF	R/W	0	ADC 中断请求标志位。 0 = 无 ADC 中断请求； 1 = ADC 请求中断。
Else				请参考其它章节。

15.7 示例程序代码

下面的示例程序代码显示了如何设置 AD 采集 AIN5 通道信号，同时有打开中断功能。

```
#define ADCAIN10_VDD   (3 << 0)  //AIN10 = VDD
#define ADCAIN10_4V    (2 << 0)  //AIN10 = 4.0V
#define ADCAIN10_3V    (1 << 0)  //AIN10 = 3.0V
#define ADCAIN10_2V    (0 << 0)  //AIN10 = 2.0V
#define ADCInRefVDD    (1 << 2)  //internal reference from VDD
#define ADCExHighRef   (1 << 7)  //high reference from AVREFH/P2.0
#define ADCSpeedDiv16  (0 << 4)  //ADC clock = fosc/16
#define ADCSpeedDiv8   (1 << 4)  //ADC clock = fosc/8
#define ADCSpeedDiv1   (2 << 4)  //ADC clock = fosc/1
#define ADCSpeedDiv2(3 << 4)  //ADC clock = fosc/2
#define ADCChannelEn(1 << 6)  //enable ADC channel
#define SelAIN5 (5 << 0)  //select ADC channel 5
#define ADCStart(1 << 6)  //start ADC conversion
#define ADCEn          (1 << 7)  //enable ADC
#define EADC           (1 << 3)  //enable ADC interrupt
#define ClearEOC      0xDF;

unsigned int  ADCBuffer;    // data buffer

void ADCInit(void)
{
  P1  = 0x00;
  P1M = 0x80;

  // set AIN5 pin's mode at pure analog pin
  P4CON |= 0x20;     //AIN5/P15
  P4M   &= 0xDF;    //input mode
  P4UR  &= 0xDF;    //disable pull-high

  // configure ADC channel and enable ADC.
  ADM= ADCEn | SelAIN5;
  // enable channel and select conversion speed
  ADR= ADCChannelEn | ADCSpeedDiv1;
  // configure reference voltage
  VREFH = ADCInRefVDD;

  // enable ADC interrupt
  IEN0  |= 0x80;    //enable global interrupt
  IEN2  |= EADC;

  // start ADC conversion
  ADM   |= ADCStart;
}

void ADCInterrupt(void) interrupt ISRAdc
{
  if ((IRCON2 &0x01) == 0x01) {
    P17= ~P17;
    IRCON2  &= 0xFE;   //Clear ADCF
    ADCBuffer = (ADB << 4) + (ADR &0x0F);
    ADM&= ClearEOC;
    ADM|= ADCStart;
  }
}
```

16 UART

UART（通用同步异步收发器）提供 1MHz 灵活的全双工传输模式。UART 共有 4 种操作模式：一种同步，3 种异步的。同步模式发送 8 位数据，没有起始/停止位；其它模式则分别支持 8 位和 9 位数据的发送，包含起始/停止位。

16.1 UART模式 0：同步 8 位收发器

模式 0 下，UART 发送/接收 8 位长度的数据，没有起始/停止位。总线的首位为最低有效位 LSB，波特率固定为 Fcpu/12。通过设置 REN0 位为 1 和将 RI0 位清零开始接收；而通过写入数据到 S0BUF 开始发送。

16.2 UART模式 1：异步 8 位收发器

模式 1 下，UART 操作为典型的格式协议：包括起始位，8 位数据位和停止位。波特率由 S0RELH/SORELL 寄存器，或取决于 BD 寄存器的 T1 的溢出周期控制。

通过写入数据到 S0BUF 寄存器开始发送，发送的首位是起始位（始终为 0），然后是 S0BUF 的数据（首先 LSB），发送完停止位（始终为 1）后，有通知/中断时自动将 TI0 位设置为 1。

若使能接收功能（REN0=1），把数据位的第一位当初最低有效位（LSB），接收完成后，S0BUR 会进行更新。

16.3 UART模式 2/3：异步 9 位收发器

模式 2 和模式 3 都包含起始位、9 位数据位和停止位。模式 2 的波特率只有 2 个选项：Fcpu/32 和 Fcpu/64；而模式 3 的波特率可通过 S0RELH/SOTRLL 寄存器或 T1 的溢出周期来控制。

第 9 位数据位（在 S0BUF 寄存器的最高位传输完毕之后传输）可通过写入 TB80 寄存器来传送，接收后在 RB80 中进行读取。也可以一直设置第 9bit 数据为 1，表示每次发送 2 个 STOP 信号，也可以将第 9bit 数据作为奇偶校验位。

第 9bit 数据也经常被使用来自定义为一对多的通信协议，当 SM20 寄存器设置为 1 时，仅仅收到的第 9bit 数据为 1 时，才会产生接收中断。利用这个功能就可以实现一主多从的通信协议。所有的从机都设置 SM20 为 1，主机端先发送需要通信的从机地址，同时设置第 9bit 为 1，这样所有的从机都会产生接收中断。从机判断接收到的地址是否为发给自己的。如果地址匹配，则对应的从机就将 SM20 清 0，地址不匹配的从机，保持 SM20 为 1。同时主机再发送后续的数据时，同时将第 9bit 数据设置为 0 再发出，那么其他从机就不会再产生接收中断。

16.4 波特率控制

UART 模式 0 的波特率是固定的，为 Fcpu/12；模式 2 有 2 个波特率选项，由 SMOCD 寄存器选择控制，其选项分别为 FCPU/32（SMOD=0）和 Fcpu/64（SMOD=1）。

UART 模式 1 和模式 3 的波特率由 S0RELH/SORELL 寄存器（BD=1）或者 T1 溢出周期（BD=0）产生。通过设置 SMOD 位可以将波特率增加 1 倍。

在模式 1 和模式 3 中若选择 S0RELH/S0RELL（BD=1），则波特率由下面公式产生：

$$\text{Baud Rate} = 2^{\text{SMOD}} \times \frac{\text{fcpu}}{64 \times (1024 - \text{S0REL})}$$

普通 UART 波特率的设置建议如下表设置（Fcpu=32MHz）：

波特率	SMOD	S0RELH	S0RELL	Accuracy
4800 Hz	0	0x03	0x98	-0.16 %
9600 Hz	0	0x03	0xCC	-0.16 %
19200 Hz	0	0x03	0xE6	-0.16 %
38400 Hz	0	0x03	0xF3	-0.16 %
56000 Hz	1	0x03	0xEE	0.79 %
57600 Hz	1	0x03	0xEF	-2.12 %
115200 Hz	1	0x03	0xF7	3.55 %
128 kHz	1	0x03	0xF8	2.34 %
250 kHz	1	0x03	0xFC	0 %
500 kHz	1	0x03	0xFE	0 %
1 MHz	1	0x03	0xFF	0 %

在模式 1 和模式 3 中若选择 T1 溢出周期（BD=0），则波特率由下面公式产生。T1 必须为 8 位自动重装模式，这样才能产生周期性的溢出信号。

$$\text{Baud Rate} = 2^{\text{SMOD}} \times \frac{\text{fcpu}}{32 \times \text{Timer 1 period}}$$

建议设置 T1 溢出间隔时间（T1 时钟=32M），用于普通 UART 波特率（Fcpu=32MHz）

波特率	SMOD	定时器间隔时间	TH1/TL1	Accuracy
4800 Hz	0	6.510us	0x30	0.16 %
9600 Hz	1	6.510us	0x30	0.16 %
19200 Hz	1	3.255us	0x98	0.16 %
38400 Hz	1	1.628us	0xCC	0.16 %
56000 Hz	1	1.116us	0xDC	-0.80 %
57600 Hz	1	1.085us	0xDD	-0.80 %
115200 Hz	1	0.543us	0xEF	2.08 %
128 kHz	1	0.488us	0xF0	-2.40 %
250 kHz	1	0.250us	0xF8	0 %
500 kHz	1	0.125us	0xFC	0 %
1 MHz	1	0.063us	0xFE	0 %

时钟源为 T1 溢出源时，系统时钟 Fcpu 最小设置值小于或等于定时器间隔时时间值。

16.5 省电

UART 模块带有时钟 gating 功能以省电。REN0=0 时，停止 UART 模块内部时钟以减少功耗，此时不能访问 UART 相关寄存器（S0CON、S0BUF、S0CON2、S0RELL、S0RELH 和 SNOD）；反之，REN0=1 时，UART 模块内部时钟正常运行，也能正常访问相关寄存器。初始化 UART 之前，REN0 位必须置为 1。

16.6 UART寄存器

UART 寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
S0CON	SM0	SM1	SM20	REN0	TB80	RB80	TI0	RI0
S0CON2	BD	-	-	-	-	-	-	-
S0BUF	S0BUF7	S0BUF6	S0BUF5	S0BUF4	S0BUF3	S0BUF2	S0BUF1	S0BUF0
PCON	SMOD	-	-	-	P2SEL	GF0	STOP	IDLE
S0RELH	-	-	-	-	-	-	S0REL9	S0REL8
S0RELL	S0REL7	S0REL6	S0REL5	S0REL4	S0REL3	S0REL2	S0REL1	R0REL0
IEN0	EAL	-	ET2	ES0	ET1	EX1	ET0	EX0
P1OC				P06OC	P05OC	P04OC	P01OC	P00OC
P0M	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M
P0	P07	P06	P05	P04	P03	P02	P01	P00

S0CON 寄存器（0x98）

Bit	Field	Type	Initial	说明
7..6	SM[0:1]	R/W	00	UART 模式选择位。 00：模式 0； 01：模式 1； 10：模式 2； 11：模式 3。
5	SM20	R/W	0	多重处理器通讯控制位（模式 2、3）。 0：禁止； 1：使能。
4	REN0	R/W	0	UART 接收功能控制位。 0：禁止； 1：使能。
3	TB0	R/W	0	发送数据的第 9 位（模式 2、3）。
2	RB0	R/W	0	接收数据的第 9 位。
1	TI0	R/W	0	发送 UART 的中断标志。
0	RI0	R/W	0	接收 UART 的中断标志。

S0CON2 寄存器（0xD8）

Bit	Field	Type	Initial	说明
7	BD	R/W	0	波特率产生器选择位（模式 1、3）。 0：T1 溢出周期； 1：由寄存器 S0RELH/S0RELL 控制。
6..0	Reserved	R	0x00	

S0BUF 寄存器（0x99）

Bit	Field	Type	Initial	说明
7..0	S0BUF	R/W	0x00	写入数据的操作触发 UART 通讯（首先 LSB），可在 package 结束时读取接收到的数据。

PCON 寄存器（0x87）

Bit	Field	Type	Initial	说明
7	SMOD	R/W	0	UART 波特率控制位（UART 模式 0、2）。 0：Fcpu/64 1：Fcpu/32
6..0				请参考其它章节。

IEN0 寄存器（0xA8）

Bit	Field	Type	Initial	说明
7	EAL	R/W	0	使能中断，请参考中断章节。
4	ES0	R/W	0	使能 UART 中断。
Else				请参考其它章节。

P1OC 寄存器（0xE4）

Bit	Field	Type	Initial	说明
1	P06OC	R/W	0	0：切换 P0.6（URX）为输入模式（要求）； ~~1：切换P0.6（URX）为开漏模式*。~~
0	P05OC	R/W	0	0：切换 P0.5（UTX）为推拉模式； 1：切换 P0.5（UTX）为开漏模式。
Else				请参考其它章节。

*设置 P06OC 为高电平会导致 URX 不能接收数据。

P0M 寄存器（0xF9）

Bit	Field	Type	Initial	说明
6	P06M	R/W	0	0：设置 P0.6（URX）为输入模式（要求）； ~~1：设置P0.6（URX）为输出模式*。~~
5	P05M	R/W	0	~~0：设置P0.5（UTX）为输入模式*；~~ 1：设置 P0.5（UTX）为输出模式（要求）。
Else				请参考其它章节。

*URX 和 UTX 分别要求为输入模式和输出模式，以对应的接收和发送数据。

P0 寄存器（0x80）

Bit	Field	Type	Initial	说明
6	P06	R/W	0	随时读取该位以监控总线状态。
5	P05	R/W	0	~~0：设置P0.5（UTX）始终为低电平*；~~ 1：P0.5（UTX）输出 UART 数据（要求）。
Else				请参考其它章节。

* 初始化 P05 时一定需要设置为高电平，因为 P05 输出低电平会认为 UART 通信开始，造成会发出一包错误的数据。

16.7 示例程序代码

下面的示例程序代码显示了在中断中如何执行 UART 模式 1。

```
#define SYSUartSM0   (0<< 6)
#define SYSUartSM1   (1 << 6)
#define SYSUartSM2   (2 << 6)
#define SYSUartSM3   (3 << 6)
#define SYSUartREN   (1 << 4)
#define SYSUartSMOD  (1 << 7)
#define SYSUartES0   (1 << 4)

void SYSUartInit(void)
{
  // set UTX, URX pins' mode at here or at GPIO initialization
  P05 = 1;
  P0M = P0M | 0x20& ~0x40;

  // configure UART mode between SM0 and SM3, enable URX
  S0CON = SYSUartSM1 | SYSUartREN;

  // configure UART baud rate
  PCON = SYSUartSMODE1;
  S0CON2 = SYSUartBD1;
  S0RELH = 0x03;
  S0RELL = 0xFE;

  // enable UART interrupt
  IEN0 |= SYSUartES0;

  // send first UTX data
  S0BUF = uartTxBuf;
}

void SYSUartInterrupt(void) interrupt ISRUart
{
  if (TI0 == 1) {
    S0BUF = uartTxBuf;
    TI0 = 0;
  } else if (RI0 == 1) {
    uartRxBuf = S0BUF;
    RI0 = 0;
  }
}
```

17 SPI

串行外设接口 SPI 有 2 种操作模式：主控模式和从动模式。主机通过 SCK 引脚发送总线时钟信号；反之从机设备基于 SCK 引脚的时钟输入处理通讯。从动模式下由寄存器使能片选引脚（SSN）。

17.1 SPI主控模式

SPI 控制模式共有 7 种类型的时钟发生器，从 Fcpu/2~Fcpu/128，产生的时钟由 SCK 引脚（与 P1.3 共用）输出，其 IDLE 状态由 CPOL 控制。

输入输出数据的相位由 CPHZ 寄存器自动指定。主控模式下，MOSI 引脚（与 P1.4 共用）负责输出数据，MISO 引脚（与 P1.5 共用）负责获取来自从机设备的数据。通过写入数据到 SPDAT 寄存器开始 SPI 通讯；数据发送完成后，从 MISO 引脚读取接收到的数据。

主控模式下，有 2 种包含中断功能的状态标志：

－SPIF 寄存器显示一字节数据通讯的结束，若此时使能 ESPI 位则释放中断。

－发送时由 SSN（与 P0.2 共用）低电平状态释放 MODF，通过设置 SSDIS 位屏蔽中断源。

17.2 SPI从动模式

SPI 从动模式监控 SCK 引脚控制 MISO 和 MOSI 通讯，但其最大时钟 rate 限制在 Fcpu/8。在连接到 SPI 总线的配置相同时，从动设备可用于指定 CPOL 和 CPHA 设置。

输入数据时可将从动模式看作是 MOSI 引脚，发送数据时可看作是 MISO 引脚。由于疏忽，SSDIS 寄存器为低电平状态，即从动选择引脚（SSN）是有功能的。若 SSN 引脚为低电平状态，则处理 SPI 通讯；而 SSN 为高电平状态时，从动设备处于悬浮状态。

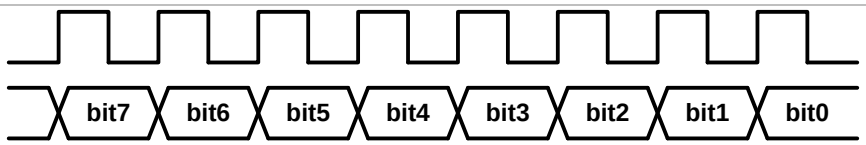
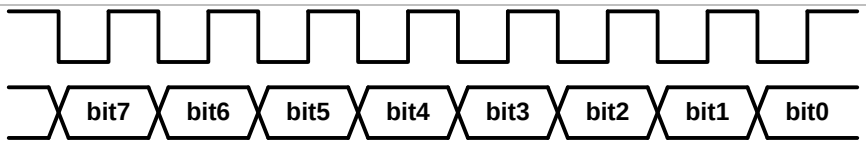
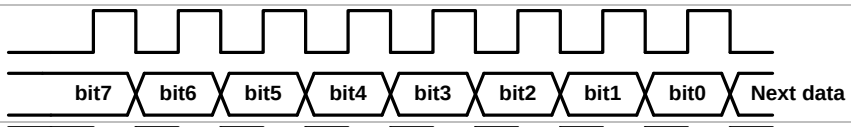
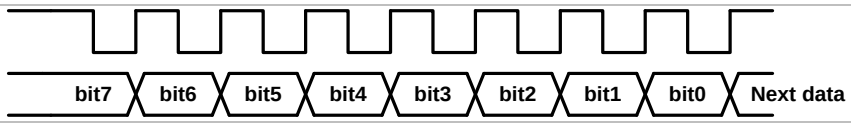
从动模式下，有 2 种包含中断功能的状态标志：

－SPIF 寄存器显示一字节数据通讯的结束，发送 SPDAT 的初始值后，再通过 SPDAT 读取从 MOSI 接收的到的数据。

－MODF 显示的是：在完成一字节数据通讯之前改变从动选择引脚 SSN 为高电平，换句话说就是打断最后一次 SPI 通讯。

17.3 SPI操作

下表说明了 CPOL 和 CPHA 的 4 种不同的设置，和每种组合下的总线操作。

CPOL	CPHA	Diagrams	Description
0	1		SCK 低电平空闲 下降沿锁存数据
1	1		SCK 高电平空闲 上升沿锁存数据
0	0		SCK 低电平空闲 上升沿锁存数据
1	0		SCK 高电平空闲 下降沿锁存数据

17.4 省电

SPI 模块带有时钟 gating 功能以省电。SPEN=0 时，停止 SPI 模块内部时钟以减少功耗，此时不能访问 SPI 相关寄存器（SPCON、SPSTA 和 SPDAT）；反之，SPEN=1 时，SPI 模块内部时钟正常运行，也能正常访问相关寄存器。初始化 SPI 之前，SPEN 位必须置为 1。

17.5 SPI寄存器

SPI 寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SPCON	SPR2	SPEN	SSDIS	MATR	CPOL	CPHA	SPR1	SPR0
SPSTA	SPIF	WCOL	SSERR	MODF	-	-	-	-
SPDAT	SPDAT7	SPDAT6	SPDAT5	SPDAT4	SPDAT3	SPDAT2	SPDAT1	SPDAT0
IEN0	EAL	-	ET2	ES0	ET1	EX1	ET0	EX0
IEN1	ET2RL	-	ET2C3	ET2C2	ET2C1	ET2C0	ESPI	EI2C
P0OC	-	-	-	P15OC	P14OC	P13OC	P06OC	P05OC
P1M	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M

SPCON 寄存器（0xE2）

Bit	Field	Type	Initial	说明
7,1,0	SM[2:0]	R/W	000	SPI 波特率发生器（仅在主控模式下有效）。 000：Fcpu/2 001：Fcpu/4 010：Fcpu/8 011：Fcpu/16 100：Fcpu/32 101：Fcpu/64 110：Fcpu/128 ~~111：reserved~~
6	SPEN	R/W	0	SPI 通讯功能。 0：禁止； 1：使能。
5	SSDIS	R/W	0	从动选择引脚功能（仅在 MSTR = 0，CPHA = 0 时有效）。 0：使能从动选择引脚 SSN 功能； 1：禁止从动选择引脚 SSN 功能。
4	MSTR	R/W	1	SPI 模式选择位。 0：从动模式； 1：主控模式。
3	CPOL	R/W	0	SCK 引脚空闲状态位。 0：SCK 低电平空闲； 1：SCK 高电平空闲
2	CPHA	R/W	1	数据的时钟相位锁存控制位。 0：由第一个时钟沿锁存数据； 1：由第二个时钟沿锁存数据。

SPSTA 寄存器（0xE1）

Bit	Field	Type	Initial	说明
7	SPIF	R	0	SPI 通讯完成标志位。 在通讯结束时自动设置为 1； 通过读取 SPSTA，SPDAT 寄存器自动清零。
6	WCOL	R	0	写操作冲突标志位。 在通讯时对 SPDAT 执行写操作自动设置为 1； 通过读取 SPSTA，SPDAT 寄存器自动清零。
5	SSERR	R	0	同步从动选择引脚错误位。 若 SSN 错误控制时自动设置为 1； 通过清 SPEN 自动清零。
4	MODF	R	0	模式错误标志位。
3..0	Reserved	R	0x00	

SPDAT 寄存器（0xE3）

Bit	Field	Type	Initial	说明
7..0	SPDAT	R/W	0x00	主控模式：写操作触发 SPI 通讯；接收的数据可在一字节通讯结束时读取（SPIF 自动设置为 1）。 从动模式：通过 SCK 输入发送写入的数据；接收的数据可在一字节通讯结束时读取（SPIF 自动设置为 1）。

IEN0 寄存器（0xA8）

Bit	Field	Type	Initial	说明
7	EAL	R/W	0	使能中断。请参考中断章节。
Else				请参考其它章节。

IEN1 寄存器（0xB8）

Bit	Field	Type	Initial	说明
1	ESPI	R/W	0	使能 SPI 中断。
Else				请参考其它章节。

P0OC 寄存器（0xE4）

Bit	Field	Type	Initial	说明
4	P15OC	R/W	0	0：切换 P1.5（MISO）为输入或输出模式； 1：切换 P1.5（MISO）为开漏模式。
3	P14OC	R/W	0	0：切换 P1.4（MOSI）为输入或输出模式； 1：切换 P1.4（MOSI）为开漏模式。
2	P13OC	R/W	0	0：切换 P1.3（SCK）为输入或输出模式； 1：切换 P1.3（SCK）为开漏模式。
Else				请参考其它章节。

P1M 寄存器

Bit	Field	Type	Initial	说明
5	P15M	R/W	0	0：设置P1.5（MISO）为输入模式[主控模式] 1：设置P1.5（MISO）为输出模式[从动模式]
4	P14M	R/W	0	0：设置P1.4（MOSI）为输入模式[从动模式] 1：设置P1.4（MOSI）为输出模式[主控模式]
3	P13M	R/W	0	0：设置P1.3（SCK）为输入模式[从动模式] 1：设置P1.3（SCK）为输出模式[主控模式]
Else				请参考其它章节。

[1]在从动模式下本就要设置SCK为输入模式；在主控模式下建议设置为输出模式。
[2]在主控模式下本就要设置MISO为输入模式；在从动模式下建议设置为输出模式。
[3]在从动模式下本就要设置MOSI为输入模式；在主控模式下建议设置为输出模式。

P0M 寄存器（0xF9）

Bit	Field	Type	Initial	说明
2	P02M	R/W	0	0：设置 P0.2（SSN）为输入模式*；。 1：设置 P0.2（SSN）为输出模式*。

*若从动模式下使能SSN功能：本就是要设置SSN为输入模式。

17.6 示例代码

下面的示例代码程序演示了如何执行 SPI。

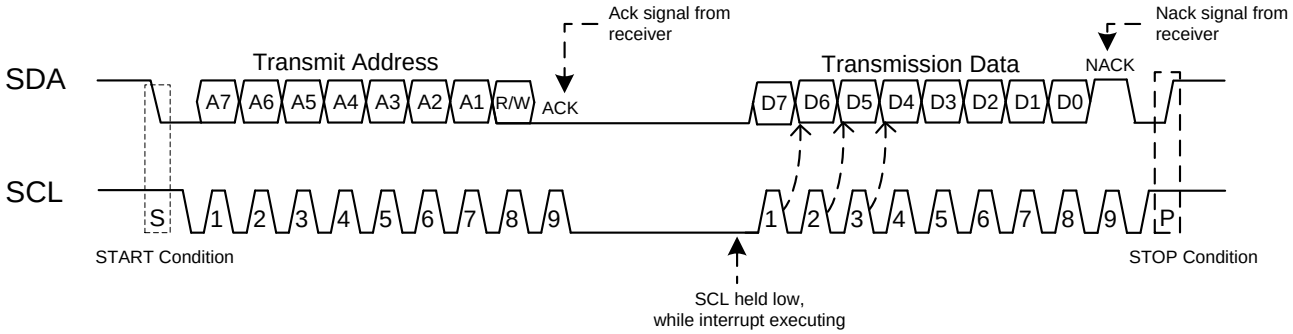
```
#dedine SpiMaster      (1 << 4)   //SPI=Master mode
#define SpiSlave       (1 << 4)   //SPI=Slave mode
#define SpiMode0       (0 << 2)   //SCK idle low, data latch at rising edge
#define SpiMode1       (1 << 2)   //SCK idle low, data latch at falling edge
#define SpiMode2       (2 << 2)   //SCK idle high, data latch at falling edge
#define SpiMode3       (3 << 2)   //SCK idle high, data latch at rising edge
#define SpiEn          (1 << 6)  //Enable SPI
#define SpiSSNEn       (0 << 5)  //SSN pin function enable
#define SpiSSNDis      (1 << 5)  //SSN pin function disable

Unsigned char u8SpiData = 0;     //data buffer
Unsigned char u8TxCompleted = 0;

void SpiMaster(void)
{
 Unsigned char u8RcvData = 0;

 //SCK & MOST = output, MISO = input
 P1M | = 0x18;
 //Enable Spi, Master mode, SSN pin disable, Fclk/128
 //SCK idle low, data latch at falling edge
 SPCON = SpiEn | SpiMaster | SpiMode1 | SpiSSNDis | 0x82;
//Enable Global/SPI interrupt
 IEN1 | = 0x02;
 IEN0 | = 0x80;      //enable global interrupt

 While (1) {
   SPDAT = 0x55;
   while(!u8TxCompleted);    //wait end of transmition
   u8TxCompleted = 0;        //clear sw flag
   u8RcvData = u8SpiData;    //receive 0x66

   SPDAT = 0x99;
   while(!u8TxCompleted);    //wait end of transmition
   u8TxCompleted = 0;        //clear sw flag
   u8RcvData = u8SpiData;    //receive 0xAA
  }
}

void SpiTinterrupt(void) interrupt ISPSpi  //0x4B
{
 Switch (SPSTA)         //Clear SPI flag (SPIF) by reading
 {
   case 0x80:
     u8SpiData = SPDAT;
     u8TxCompleted = 1;
     break;
    case 0x10:
      //Mode Fault
      Break;
  }
}
```

18 I2C

I2C 是串行通讯接口，在单片机与单片机，或单片机与其它外设之间进行数据传输。I2C 可作为主机或从机进行双向 IO 传输数据，SDA（串行数据输出）和 SCL（串行时钟输出）。数据转换时，执行“WRITE”和“READ”操作。主机发送数据给从机时，叫做“WRITE”操作；从机发送数据给主机时，叫做“READ”操作。I2C 也支持多主机通讯，通过一种仲裁方式来决定哪个主机拥有控制总线以及传输数据的权限，从而保证数据传送的正确性。

18.1 I2C协议

I2C 发送结构包括 START（S）开始信号，8 位地址字节，一个或多个数据字节，以及 STOP（P）结束信号。开始信号由主机产生，以便对传输进行初始化。发送的数据为最高有效位（MSB）优先。在地址字节中，高 7 位为地址位，低位为数据方向位（R/W）。R/W=0 时，表示该传输为“WRITE”操作；R/W=1 时，表示该传输为“READ”操作。接收到每个字节后，接收器（主机或从机）必须发送一个 ACK 信号。若发送器没有接收到 ACK 信号，则会识别为 NACK 信号。在 WRITE 操作中，主机发送数据给从机，然后等待从机返回 ACK 信号。在 READ 操作中，从机发送数据给主机，然后等待主机返回 ACK 信号。最后，主机产生一个 STOP 信号来结束数据传输。



18.2 I2C传输模式

I2C 可作为主机/从机，执行 8 位串行数据的发送/接收操作，因此，该模块共有 4 种操作模式：主机发送，主机接收，从机发送和从机接收。

18.2.1 主机发送模式

主机发送模式为主机发送数据给从机，串行时钟由 SCL 输出时，串行数据由 SDA 输出。主机通过发送 START 信号来开始数据发送。发送 START 信号后，开始发送从机设备指定的地址字节。地址字节包含 7 位地址位，第 8 位为数据方向位（R/W），该位设为 0 时使能主机发送。接下来，主机发送一个或多个数据字节给从机，每发送一个字节之后，主机要等待从机返回的 ACK 信号。最后，主机产生 STOP 信号来结束此次数据传输。

18.2.2 主机接收模式

主机接收模式为主机接收来自从机的数据，串行时钟由 SCL 输出时，串行数据由 SDA 输入。主机通过发送 START 信号来开始数据接收。发送 START 信号后，开始发送从机设备指定的地址字节。地址字节包含 7 位地址位，第 8 位为数据方向位（R/W），该位设为 1 时使能主机接收。接下来，主机接收来自从机的一个或多个数据字节，每接收一个字节之后，通过设置 I2CCON 寄存器的 AA 标志位来将 ACK 或 NACK 信号发送给从机。最后，主机产生 STOP 信号来结束此次数据传输。

18.2.3 从机发送模式

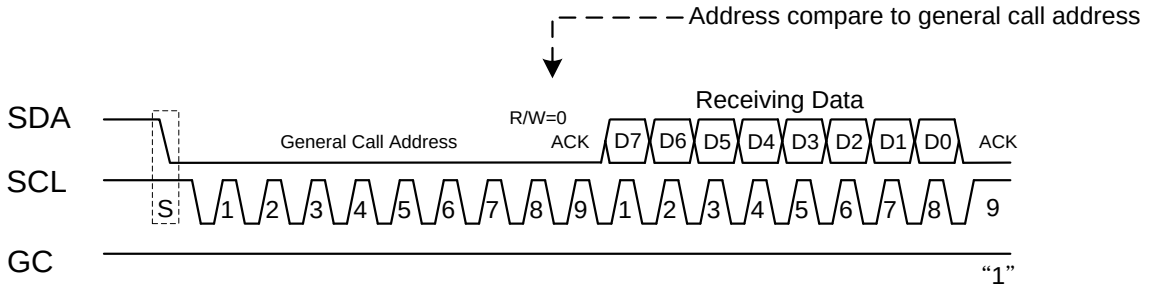
从机发送模式为从机发送数据给主机，串行时钟由 SCL 输入时，串行数据由 SDA 输出。接收到来自主机的 START 信号后开始数据发送。接收到 START 信号后，开始接收从机设备指定的地址字节。地址字节包含 7 位地址位，第 8 位为数据方向位（R/W），该位设为 1 时使能从机发送。若接收到的地址字节与 I2CADR 寄存器中的地址相匹配，从机将发送 ACK 信号；另外，若广播呼叫地址标志位设为 1（GC=1），接收到广播呼叫地址（00H）后，从机设备也会发送一个 ACK 信号。接下来，从机发送一个或多个数据字节给主机，每发送一个字节之后，从机要等待主机返回的 ACK 信号。最后，主机产生 STOP 信号来结束此次数据传输。

18.2.4 从机接收模式

从机接收模式为从机接收来自主机的数据，串行时钟和串行数据都由 SCL 和 SDA 输入。接收到来自主机的 START 信号后开始数据接收。接收到 START 信号后，开始接收从机设备指定的地址字节。地址字节包含 7 位地址位，第 8 位为数据方向位（R/W），该位设为 0 时使能从机接收。若接收到的地址字节与 I2CADR 寄存器中地址相匹配，从机将产生 ACK 信号；另外，若广播呼叫地址标志位设为 1（GC=1），接收到广播呼叫地址（00H）后，从机设备也会产生一个 ACK 信号。接下来，从机接收一个或多个来自主机的数据字节，每接收一个字节之后，从机产生 ACK 或 NACK 信号，通过设置 I2CCON 寄存器的 AA 标志位来将 ACK 或 NACK 信号发送给主机。最后，主机产生 STOP 信号来结束此次数据传输。

18.3 广播呼叫地址

在 I2C 总线中，开始信号之后的第一个字节中的头 7 位为从机地址，只有该地址与从机地址相匹配时，从机才会响应 ACK 信号。只有广播呼叫地址是个例外，广播呼叫地址可以寻址所有的从机设备。当总线上出现广播呼叫地址时，所有设备应当响应一个 ACK 信号。广播呼叫地址是一个由全 0 组成的 7 位特殊地址。广播呼叫地址功能通过 GC 标志位来控制，设置 GC 标志位为 1 将使能广播呼叫地址，清 0 将关闭广播呼叫地址。当 GC=1 时，将会识别是否是广播呼叫地址，否则当 GC=0 时，将会忽略广播呼叫地址。



18.4 串行时钟发生器

在主机模式下，SCL 时钟速率发生器由 I2CCON 寄存器中的 CR[2:0]标志位来控制。

当 CR[2:0]=000~110 时，SCL 时钟速率来自内部时钟源。

$$\text{SCL Clock Rate} = \frac{\text{Fcpu}}{\text{Prescaler}} (\text{Prescaler} = 256\sim60)$$

当 CR[2:0]=111 时，SCL 时钟速率来自 T1 定时器的溢出频率。

$$\text{SCL Clock Rate} = \frac{\text{Timer 1 Overflow}}{8}$$

下表列出了不同设置下的 SCL 时钟速率。

CR2	CR1	CR0	I2C	比特率 (kHz)			
			Prescaler	6MHz	12MHz	16MHz	24MHz
0	0	0	256	23	47	63	92
0	0	1	224	27	54	71	108
0	1	0	192	31	63	83	124
0	1	1	160	37	75	100	148
1	0	0	960	6.25	12.5	17	25
1	0	1	120	50	100	133	200
1	1	0	60	100	200	266	400
1	1	1	(Timer 1 overflow rate)/8				

当 SCL 时钟源是 T1 定时器的溢出频率时，T1 定时器的最大计时值为 0XFB（只支持 0x00~0xFB），且当 T1 定时器的时钟源是 IHRC_32MHz 时，最大的 SCL 时钟速率是 800KHz。

18.5 同步与仲裁

在多主机条件下，同一时间只有一个主机可在总线上传输数据。需要决定由哪个主机可控制总线以及实现传输。时钟同步与仲裁应用于配置多主机传输。时钟同步会在和其他设备同步 SCL 信号时运行。当同一时间有两个主机要传输时，时钟同步将会在 SCL 由高电平转变成低电平的时候开始。如果主机 1 先将 SCL 切换为低电平，那么主机 1 会将 SCL 锁定在低电平状态直到将其切换成高电平状态。不过，如果有其他主机的 SCL 时序依然保持在低电平状态的话，那么主机 1 将 SCL 时序由低电平切换成高电平状态的过程将不会改变 SCL 的状态，SCL 时序将仍保持在低电平状态。也就是说，SCL 线被有最长低电平周期的主机保持在低电平状态。当所有设备的时钟周期都转换到高电平时，SCL 线将由低转成到高电平状态。在这其间，主机 1 将保持在由低电平到高电平的等待状态，之后再继续它的传输。经过时钟同步之后，所有设备的时钟和 SCL 时钟是一样的了。仲裁是用来决定哪个主机能够通过 SDA 信号来完成它的传输。两个主机可能会在同一时间发出开始信号以及传输数据，导致两者会互相影响。仲裁会强制使得其中一个主机失去总线的控制权。数据传输依然会继续，直到两个主机输出了不同的数据信号。如果其中一个主机传输了高电平状态，而另外一个主机传输了低电平状态，SDA 线会被拉低。输出高电平状态的主机将会检测到 SDA 线上的异常，并失去总线的控制权。输出低电平状态的主机成功获得总线的控制权，并继续它的传输，仲裁的过程中不会丢失数据。

18.6 系统管理总线(SMBus)扩展

可选的系统管理总线(SMBus) 协议的硬件支持三种类型的超时检测： (1) Tmext 超时检测：一个字节的累积持续时钟周期；(2)Tsext 超时检测：开始信号束信号之间的累积持续时钟周期；(3)超时检测：低电平时钟周期的测量。

通过 SMBSEL 和 SMBDST 寄存器来控制超时检测。SMBSEL 寄存器中的 SMBEXE 标志位是 SMBus 拓展功能的使能位。当 SMBEXE=1 时，使能 SMBus 拓展功能。否则，关闭 SMBus 拓展功能。超时类型和周期设置由 SMBTOP[2:0]和 SMBDST 寄存器来控制。SMBus 超时的周期由 Tmex，Tsext 以及 Tout 这个 16 位的暂存器来控制。计算公式如下：

$$\text{Tmext/Tsext/Tout} = \frac{\text{Timeout Period(sec)xFcpu(Hz)}}{1024}$$

Tmext 由 Tmext_L 和 Tmext_H 这两个 8 位寄存器组成，Tmext_L 为低字节，Tmext_H 为高字节。Tsext 由 Tsext_L 和 Tsext_H 这两个 8 位寄存器组成，Tsext_L 为低字节，Tsext_H 为高字节。Tout 由 Tout_L 和 Tout_H 这两个 8 位寄存器组成，Tout_L 为低字节，Tout_H 为高字节。

类型	超时周期	Fcpu=24MHz		Fcpu=32MHz	
		十进制	十六进制	十进制	十六进制
Tmext	5ms	118	76	157	9D
Tsext	25ms	586	24A	782	30E
Tout	35ms	821	335	1094	446

通过设置 SMBTOP[2:0]来选择要设置的寄存器类型（如下表所示），将要配置的数据写到 SMBDST 寄存器中即可。

SMBTOP[2:0]	SMBDST	说明
000	Tmext_L	选择 Tmext 寄存器的低字节
001	Tmext_H	选择 Tmext 寄存器的高字节
010	Tsext_L	选择 Tsext 寄存器的低字节
011	Tsext_H	选择 Tsext 寄存器的高字节
100	Tout_L	选择 Tout 寄存器的低字节
101	Tout_H	选择 Tout 寄存器的高字节

当 SMBus 拓展功能使能之后，I2CSTA 寄存器的低三位指示超时的相关信息，如下表所示：

I2CSTA	说明
XXXX X000	没有超时错误
XXXX XXX1	Tout 超时错误
XXXX XX1X	Tsext 超时错误
XXXX X1XX	Tmext 超时错误

18.7 省电

I2C 模块带有时钟 gating 功能以省电。ENS1=0 时，停止 I2C 模块内部时钟以减少功耗，此时不能访问 I2C 相关寄存器（I2CDAT、I2CADR、I2CCON、I2CSTA、SMBSEL 和 SMBDST）；反之，ENS1=1 时，SPI 模块内部时钟正常运行，也能正常访问相关寄存器。初始化 I2C 之前，ENS1 位必须置为 1。

18.8 I2C寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
I2CDAT	I2CDAT7	I2CDAT6	I2CDAT5	I2CDAT4	I2CDAT3	I2CDAT2	I2CDAT1	I2CDAT0
I2CADR	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0	GC
I2CCON	CR2	ENS1	STA	STO	SI	AA	CR1	CR0
I2CSTA	I2CSTA7	I2CSTA6	I2CSTA5	I2CSTA4	I2CSTA3	I2CSTA2	I2CSTA1	I2CSTA0
SMBSEL	SMBEXE	-	-	-	-	SMBSTP2	SMBSTP1	SMBSTP0
SMBDST	SMBD7	SMBD6	SMBD5	SMBD4	SMBD3	SMBD2	SMBD1	SMBD0
IEN0	EAL	-	ET2	ES0	ET1	EX1	ET0	EX0
IEN1	ET2RL	-	ET2C3	ET2C2	ET2C1	ET2C0	ESPI	EI2C
P1M	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M
P0M	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M
P1	P17	P16	P15	P14	P13	P12	P11	P10
P0	P07	P06	P05	P04	P03	P02	P01	P00

I2CDAT 数据寄存器（0xDA）

Bit	Field	Type	Initial	说明
7:0	I2CDAT[7:0]	R/W	0x00	I2CDAT 寄存器存储的是要通过 I2C 总线发送出去的一字节数据，或者是刚从 I2C 总线上接收到的一字节数据。当它不在字节移位的过程中时，控制器可以读写这个 8 位的直接寻址特殊功能寄存器。由于 I2CDAT 寄存器没有缓存和双缓冲，因此当发生 I2C 中断时，用户只能读 I2CDAT 寄存器。

I2CADR 从机地址寄存器（0xDB）

Bit	Field	Type	Initial	说明
7:1	I2CADR[6:0]	R/W	0x00	I2C 从机地址。
0	GC	R/W	0	广播呼叫地址（00H）。 0：忽略； 1：识别。

I2CCON 控制寄存器（0xDC）

Bit	Field	Type	Initial	说明
7,1,0	CR[2:0]	R/W	0	I2C 时钟 rate。 000：Fcpu/256； 001：Fcpu/224； 010：Fcpu/192； 011：Fcpu/160； 100：Fcpu/960； 101：Fcpu/120； 110：Fcpu/60； 111：T1 溢出周期/8。
6	ENS1	R/W	0	I2C 使能位。 0：关闭； 1：使能。
5	STA	R/W	0	START 标志位。 0：没有发送 START 开始信号； 1：若总线空闲则发送 START 开始信号。
4	STO	R/W	0	STOP 标志位。 0：没有发送 STOP 结束信号； 1：若 I2C 总线为主机模式，则发送 STOP 结束信号。
3	SI	R/W	0	串行中断标志位。 当进入了 I2C 的 26 个状态当中的 25 个状态时，SI 标志位会被硬件置 1，只有当 I2C 状态寄存器为 F8h 时，SI 标志位才没有被置 1，表示没有可用的相关状态信息。SI 标志位必须由软件清零，必须通过写 0 到 SI 标志才可以清 SI 标志位，写入 1 到该位并不能更改 SI 的值。
2	AA	R/W	0	有效 ACK 标志位。 0：接收到 1 个字节后返回 NACK； 1：接收到 1 个字节后返回 ACK。

I2CSTA 状态寄存器（0xDD）

Bit	Field	Type	Initial	说明
7:3	I2CSTA[7:3]	R	11111	I2C 状态代码。
2..0	I2CSTA[2:0]	R	000	SMBus 状态代码。

I2C 状态代码和状态

模式	状态代码	I2C 的状态	应用软件响应					I2C 硬件引起的下步操作
			To/from I2CDAT	TO I2CCON				
				STA	STO	SI	AA	
主机发送器/接收器	08H	已发送 START 开始信号	载入 SLA+R	X	0	0	X	发送 SLA+R；接收 ACK。
	10H	已发送重复 START 开始信号	载入 SLA+R	X	0	0	X	发送 SLA+R；接收 ACK。
			载入 SLA+W					发送 SLA+W；I2C 切换到主机发送模式。
主机发送器	18H	已发送 SLA+W；并已接收到 ACK	载入数据字节	0	0	0	X	发送数据字节；接收 ACK。
			无动作	1	0	0	X	发送重复 START 开始信号。
			无动作	0	1	0	X	发送 STOP 结束信号；复位 STO 标志位。
			无动作	1	1	0	X	发送 STOP 结束信号后接着再发送 START 开始信号；复位 STO 标志位。
	20H	已发送 SLA+W；并已接收 NACK	载入数据字节*	0	0	0	X	发送数据字节；接收 ACK。
			无动作	1	0	0	X	发送重复 START 开始信号。
			无动作	0	1	0	X	发送 STOP 结束信号；复位 STO 标志位。
			无动作	1	1	0	X	发送 STOP 结束信号后接着再发送 START 开始信号；复位 STO 标志位。
	28H	已发送 I2CDAT 中的数据字节；并已接收 ACK	加载数据字节	0	0	0	X	发送数据字节；接收 ACK。
			无动作	1	0	0	X	发送重复 START 开始信号。
			无动作	0	1	0	X	发送 STOP 结束信号；复位 STO 标志位。
			无动作	1	1	0	X	发送 STOP 结束信号后接着再发送 START 开始信号；复位 STO 标志位。
	30H	已发送 I2CDAT 中的数据字节；并已接收 NACK	加载数据字节*	0	0	0	X	发送数据字节；接收 ACK。
			无动作	1	0	0	X	发送重复 START 开始信号。
			无动作	0	1	0	X	发送 STOP 结束信号；复位 STO 标志位。
			无动作	1	1	0	X	发送 STOP 结束信号后接着再发送 START 开始信号；复位 STO 标志位。
主机接收器	40H	已发送 SLA+R；并已接收 ACK	无动作	0	0	0	0	接收数据字节；返回 NACK。
			无动作	0	0	0	1	接收数据字节；返回 ACK。
	48H	已发送 SLA+R；并已接收 NACK	无动作	1	0	0	X	发送重复 START 开始信号。
			无动作	0	1	0	X	发送 STOP 结束信号；复位 STO 标志位。
			无动作	1	1	0	X	发送 STOP 结束信号后接着再发送 START 开始信号；复位 STO 标志位。
	50H	已接收数据字节；并已返回 ACK	读数据字节	0	0	0	0	接收数据字节；返回 NACK。
			读数据字节	0	0	0	1	接收数据字节；返回 ACK。
	54H	接收数据字节；并已返回 NACK	读数据字节	1	0	0	X	发送重复 START 开始信号。
			读数据字节	0	1	0	X	发送 STOP 结束信号；复位 STO 标志位。
			读数据字节	1	1	0	X	发送 STOP 结束信号后接着再发送 START 开始信号；复位 STO 标志位。

模式	状态代码	I2C 的状态	应用软件响应					I2C 硬件引起的下步操作
			To/from I2CDAT	TO I2CCON				
				STA	STO	SI	AA	
从机接收器	60H	已接收到对应的 SLA+W ；已返回 ACK	无动作	X	0	0	0/1	接收数据字节；返回 NACK/ACK
	68H	主机在发送 SLA+R/W 时仲裁丢失；并接收到了对应的 SLA+W，已返回 ACK	无动作	X	0	0	0/1	接收数据字节；返回 NACK/ACK
	70H	接收到了广播呼叫地址（00H）；已返回 ACK	无动作	X	0	0	0/1	接收数据字节；返回 NACK/ACK
	78H	主机在发送 SLA+R/W	无动作	X	0	0	0/1	接收数据字节；返回 NACK/ACK

		时仲裁丢失；并接收到广播呼叫地址（00H）；已返回 ACK						
	80H	从机地址已寻址成功；已接收 DATA；并返回 ACK	读数据字节	X	0	0	0/1	接收数据字节；返回 NACK/ACK
	88H	从机地址已寻址成功；已接收 DATA；并返回 NACK	读数据字节	0	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址。
			读数据字节	0	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到。
			读数据字节	1	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址；总线空闲之后将发送开始信号。
			读数据字节	1	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到；总线空闲之后将发送开始信号。
	90H	广播呼叫地址已寻址成功；已接收 DATA；并返回 ACK	读数据字节	X	0	0	0/1	接收数据字节；返回 NACK/ACK
	98H	广播呼叫地址已寻址成功；已接收 DATA；并返回 NACK	读数据字节	0	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址。
			读数据字节	0	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到。
			读数据字节	1	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址；总线空闲之后将发送开始信号。
			读数据字节	1	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到；总线空闲之后将发送开始信号。
	A0H	当从机接收器或者从机发生器仍然被寻址时，接收到了 STOP 结束信号或者重复 START 开始信号	无动作	0	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址。
			无动作	0	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到。
			无动作	1	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址；总线空闲之后将发送开始信号。
			无动作	1	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到；总线空闲之后将发送开始信号。
从机发送器	A8H	已接收对应的 SLA+R；已返回 ACK	载入数据字节	X	0	0	0	发送最后一个字节的数据，并将接收到 ACK 信号
			载入数据字节	X	0	0	1	发送一个字节的数据，并将接收到 ACK 信号。
	B0H	主机在发送 SLA+R/W 时仲裁丢失了；已接收到了对应的 SLA+R ，且已返回 ACK 信号。	载入数据字节	X	0	0	0	发送最后一个字节的数据，并将接收到 ACK 信号
			载入数据字节	X	0	0	1	发送一个字节的数据，并将接收到 ACK 信号。
	B8H	数据字节已经发送，将接收到 ACK 信号。	载入数据字节	X	0	0	0	发送最后一个字节的数据，并将接收到 ACK 信号
			载入数据字节	X	0	0	1	发送一个字节的数据，并将接收到 ACK 信号。
	C0H	数据字节已经发送，并已接收到 NACK 信号	无动作	0	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址。

			无动作	0	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到。
			无动作	1	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址；总线空闲之后将发送开始信号。
			无动作	1	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到；总线空闲之后将发送开始信号。
	C8H	最后一个字节的数据已经发送，并已接收到 ACK 信号。	无动作	0	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址。
			无动作	0	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到。
			无动作	1	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址；总线空闲之后将发送开始信号。
			无动作	1	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到；总线空闲之后将发送开始信号。
其他	F8H	没有可用的相关状态信息；SI=0	无动作	No action				等待或者进行当前的传输
	38H	仲裁丢失	无动作	0	0	0	X	I2C 总线将被释放；将发送开始信号。
			无动作	1	0	0	X	当总线空闲时（进入了主机模式）
	00H	在主机模式下或已选址的从机模式时，总线出错。	无动作	0	1	0	X	在主机模式或已寻址的从机模式时，只有内部硬件才会被影响。在所有情况下，总线已被释放，I2C 接口已切换至未寻址的从机模式，STO 标志位已复位。

“SLA”表示从机地址，“R”表示 R/W=1，“W”表示 R/W=0。
*表示接收到 NACK 之后不意味着通信的结束。

SMBSEL（0xDE）

Bit	Field	Type	Initial	说明
7	SMBEXE	R/W	0	SMBus 扩展功能。 0：禁止； 1：使能。
2..0	SMBSTP[2:0]	R/W	000	SMBus 超时寄存器。

SMBDST（0xDF）

Bit	Field	Type	Initial	说明
7..0	SMBD[7:0]	R/W	0x00	SMBDST 寄存器是用来提供一个读写 SMBus 超时寄存器的窗口。从 SMBDST 寄存器读写的数据实际上是对由 SMBSEL 指定的超时寄存器进行读写的数据。

IEN0（0xA8）

Bit	Field	Type	Initial	说明
7	EAL	R/W	0	中断使能位。请参考中断章节。
Else				请参考其它章节。

IEN1（0xB8）

Bit	Field	Type	Initial	说明
0	EI2C	R/W	0	中断使能位。请参考中断章节。
Else				请参考其它章节。

P1M（0xFA）

Bit	Field	Type	Initial	说明
0	P10M	R/W	0	0：时钟 P1.0（SDA）为输入模式（要求）。 ~~1：时钟P1.0（SDA）为输出入模式。*~~
Else				请参考其它章节。

* 要求 P10M 分别设为输入模式。

P1（0x90）

Bit	Field	Type	Initial	说明
0	P10	R/W	0	0：设置 P1.0（SDA）数据为 0（要求）。 ~~1：设置P1.0（SDA）数据为 1。~~
Else				请参考其它章节。

*要求 P10 分别设为 0。

P0M（0xF9）

Bit	Field	Type	Initial	说明
7	P07M	R/W	0	0：时钟 P0.7（SCL）为输入模式（要求）。 ~~1：时钟P0.7（SCL）为输出入模式。*~~
Else				请参考其它章节。

* 要求 P07M 分别设为输入模式。

P0（0x80）

Bit	Field	Type	Initial	说明
7	P07	R/W	0	0：设置 P0.7（SCL）数据为 0（要求）。 ~~1：设置P0.7（SCL）数据为 1。~~
Else				请参考其它章节。

*要求 P07 分别设为 0。

18.9 示例代码

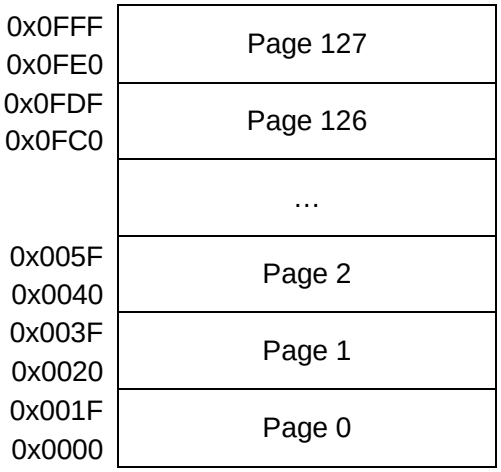
下面的示例代码程序演示了如何执行 I2C

```
Unsigned int I2CAddr;
Unsigned int I2C_TXData0;
Unsigned int I2C_TXDatan;
Unsigned int I2C_RXData0;
Unsigned int I2C_RXDatan;

void I2CInit(void)
{
  P0M & = 0Xe7;     //P03 & P04 as input

 //configure I2C clock(T1)and enable I2C.
 I2CCON | = 0xC3;
 TMOD = 0x60;    //auto reload
 TCOM0 = 0x07;   //Fosc/1
 TH1 = 0xF6;      //400KHz
 TL1 = 0xf6;      //400KHz
 TH1 = 0xD8;      //100KHz
 TH1 = 0xD8;      //100KHz
 TR1 = 1;

 //enable I2C interrupt
 EI2C = 1;
 //enable global interrupt
 EAL = 1;

 I2CCON | = 0x20;         //START (STA) = 1
}

 void I2CTinterrupt(void) interrupt ISRI2C   //0x43
 {
   Switch (I2CSTA)
    {
      //TX mode
      case 0x08:
         I2CCON & = 0xDF;       //START (STA) = 0
         I2CDAT = I2CAddr;      //TX/RX addr
         break;
      case 0x18:                //write first byte
         I2CDAT = I2C_TXDATA0;
         break;
      case 0x28:                //write n byte
         I2CDAT = I2C_TXDATAn;
         break;
      case 0x30:                //STOP (STO)
         I2CCON | = 0x10;
         break;
   // RX mode
      case 0x40:                //get slave addr
         I2CCON | = 0x04;       //AA = 1
         break;
      case 0x50:                //read n byte
         I2C_RXData0 = I2CDAT;
         I2CCON & = 0xFB;       //AA=0
         break
      case 0x58:                //read last byte & stop
         I2C_RXData0 = I2CDAT;
         I2CCON | = 0x10;       //STOP (STO)
        break;
```

```
59      default:
60         I2CCON | = 0x10;       //STOP (STO)
61     }
62
63     I2CCON & = 0xF7;          //Clear I2C flag (SI)
64 }
```

19 在线编程

SN8F5702 内置 4KB 程序存储（IROM），均分为 128 页（每页 32 字节）。在线编程就是使能固件随意的更改每页的数据，换句话说，就是存储数据到非易失存储器或者现场升级固件的一个通道。



程序存储器 IROM

19.1 页编程

由于程序存储器的每页都是 32 字节的长度，页编程就要求 32 字节 IRAM 作为其数据缓存器。

举例来说，假设程序存储器的第 216 页（IROM，0FC0H~0FDFH）为计划升级区域，已经存入 IRAM 地址 60H~7FH 的内容，执行在线编程，只要写入开始 IROM 地址 0FC0H 到 EPROMH/EPROML 寄存器，然后指定缓存器开始地址 60H 到 EPRAM 寄存器，随后写入 0A5A 到 PECMD[11:0]寄存器，复制缓存器的数据到 IROM 的第 126 页。

通常而言，每页的内容都可通过在线编程进行修改，但是第 1 页和最后一页（Page0 和 Page127）分别存储复位向量和上电控制管理的信息，不正确地执行页编程（如编程时切断电源）会导致上电错误或复位错误。

19.2 在线编程寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PERAM	RAM7	RAM6	RAM5	RAM4	RAM3	RAM2	RAM1	RAM0
PEROMH	ROM15	ROM14	ROM13	ROM12	ROM11	ROM10	ROM9	ROM8
PEROML	ROM7	ROM6	ROM5	-	CMD11	CMD10	CMD9	CMD8
PECMD	CMD7	CMD6	CMD5	CMD4	CMD3	CMD2	CMD1	CMD0

PERAM 寄存器（0x97）

Bit	Field	Type	Initial	说明
7..0	EPRAM[7:0]	R/W	0x00	数据缓存器（IRAM）的第一个地址。

PEROMH 寄存器（0x96）

Bit	Field	Type	Initial	说明
7..0	EPRAM[15:8]	R/W	0x00	编程页（IROM）的第一个地址（15^{th}– 8^{th} bit）。

PEROML 寄存器（0x95）

Bit	Field	Type	Initial	说明
7..5	EPRAM[7:5]	R/W	000	编程页（IROM）的第一个地址（7^{th}– 5^{th}）。
4	Reserved	R	0	
3..0	EPCMD[11:8]	R/W	0x0	0xA：使能在线编程。 其它值：禁止在线编程*。

*禁止在线编程可避免误触发 ISP 功能。

PECMD 寄存器（0x94）

Bit	Field	Type	Initial	说明
7..0	EPCMD[7:0]	W	0x0	0x5A：开始整页编程。 ~~其它值：保留*。~~

*不允许写入其它值到 PECMD 寄存器。

19.3 示例程序代码

```
#define SYSUartSM0   (0 << 6)
#define SYSUartSM1   (1 << 6)
#define SYSUartSM2   (2 << 6)
#define SYSUartSM3   (3 << 6)
#define SYSUartREN   (1 << 4)
#define SYSUartSMOD  (1 << 7)
#define SYSUartES0   (1 << 4)

unsigned cahridata dataBuffer[32] _at_0xE0;  // IRAM 0xE0 to 0xFF

void SYSIspSetDataBuffer(unsigned char address, unsigned char data)
{
  dataBuffer[address &0x1F] = data;
}

void SYSIspStart(unsigned int pageAddress)
{
  PERAM = 0x60;
  PEROMH = pageAddress >>8;
  PEROML = (pageAddress &0xE0) | 0x0A;
  PECMD = 0x5A;
}

void SYSIspDisable(void)
{
  PEROML &=0xE0;  // disable ISP function
}
```

20 电气特性

20.1 极限参数

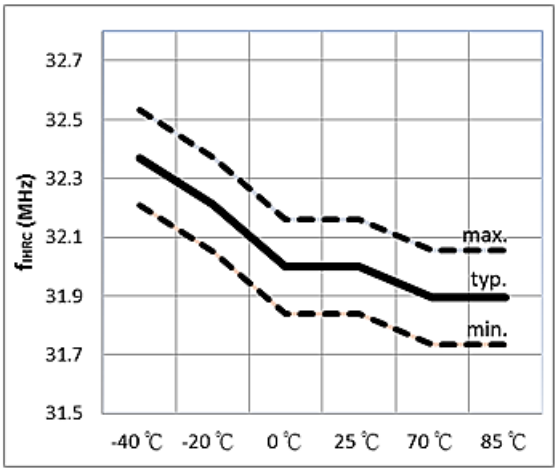
Voltage applied at VDD to VSS.....- 0.3V to 6.0V
Voltage applied at any pin to VSS.....- 0.3V to VDD+0.3V
Operating ambient temperature .....-40°C to 85°C
Storage ambient temperature .....-40°C to 125°C
Junction Temperature..... -40°C to 125°C

20.2 系统操作特性

Parameter		Test Condition	Min	TYP	MAX	UNIT
VDD	Operating voltage	fcpu = 1MHz	1.8		5.5	V
V_{DR}	RAM data retention Voltage		1.5			V
V_{POR}	VDD rising rate*		0.05			V/ms
I_{DD1}	Normal mode supply current	VDD = 3V, fcpu = 0.25MHz		2.20		mA
		VDD = 5V, fcpu = 0.25MHz		2.25		mA
		VDD = 3V, fcpu = 1MHz		2.25		mA
		VDD = 5V, fcpu = 1MHz		2.30		mA
		VDD = 3V, fcpu = 4MHz		2.70		mA
		VDD = 5V, fcpu = 4MHz		2.75		mA
		VDD = 3V, fcpu = 8MHz		3.20		mA
		VDD = 5V, fcpu = 8MHz		3.25		mA
		VDD = 3V, fcpu = 32MHz		4.50		mA
		VDD = 5V, fcpu = 32MHz		4.55		mA
I_{DD2}	STOP mode supply current	VDD = 3V		3.5		μA
		VDD = 5V		4.0		μA
I_{DD3}	IDLE mode supply current	VDD = 3V, 32MHz IHRC		0.63		mA
		VDD = 5V, 32MHz IHRC		0.65		mA
f_{IHRC}	Internal high clock generator	VDD = 1.8V to 5.5V, 25°C	31.84	32	32.16	MHz
		VDD = 1.8V to 5.5V, 25°C to 85°C	31.68		31.99	MHz
		VDD = 1.8V to 5.5V, -40°C to 25°C	32.31		32.64	MHz
F_{ILRC}	Internal low clock generator	VDD = 5V, 25°C	12	16	24	KHz
V_{LVD18}	LVD18 detect voltage	25°C	1.7	1.8	1.9	V
		-40°C to 85°C	1.6	1.8	2.0	V
V_{LVD24}	LVD24 detect voltage	25°C	2.3	2.4	2.4	V
		-40°C to 85°C	2.2	2.4	2.6	V
V_{LVD33}	LVD33 detect voltage	25°C	3.2	3.3	3.4	V
		-40°C to 85°C	3.0	3.3	3.6	V

* Parameter(s) with star mark are non-verified design reference. Ambient temperature is 25°C.

- **IHRC Frequency – Temperature Graph**



20.3 GPIO特性

Parameter		Test Condition	Min	TYP	MAX	UNIT
V_{IL}	Low-level input voltage		VSS		0.3VDD	V
V_{IH}	High-level input voltage		0.7VDD		VDD	V
I_{LEKG}	I/O port input leakage current	V_{IN} = VDD			2	μA
R_{UP}	Pull-up resister	VDD = 3V	100	200	300	kΩ
		VDD = 5V	50	100	150	kΩ
I_{OH}	I/O output source current	VDD = 5V, V_O = VDD-0.5V	12	16		mA
I_{OL1}	I/O sink current (P1, P07, P2)	VDD = 5V, V_O = VSS+0.5V	15	20		mA
I_{OL2}	I/O sink current (P00 – P06)	VDD = 5V, V_O = VSS+1.5V	80	100		mA

* Ambient temperature is 25°C.

20.4 ADC特性

Parameter		Test Condition	Min	TYP	MAX	UNIT
V_{ADC}	Operating voltage		2.0		5.5	V
V_{AIN}	AIN channels input voltage	VDD = 5V	0		V_{REFH}	V
V_{REFH}	AVREFH pin input voltage	VDD = 5V	2		VDD	V
V_{EREF}	External reference voltage	VDD = 5V	2		VDD	V
V_{IREF}	Internal VDD reference voltage	VDD = 5V		VDD		V
	Internal 4V reference voltage	VDD = 5V	3.92	4	4.08	V
	Internal 3V reference voltage	VDD = 5V	2.94	3	3.06	V
	Internal 2V reference voltage	VDD = 5V	1.96	2	2.04	V
I_{AD}	ADC current consumption	VDD = 3V		0.65		mA
		VDD = 5V		0.90		mA
f_{ADCLK}	ADC clock	VDD = 3V			16	MHz
		VDD = 5V			32	MHz
f_{ADSMP}	ADC sampling rate	VDD = 3V			250	kHz
		VDD = 5V			500	kHz
t_{ADEN}	ADC function enable period	VDD = 5V	100			μs
DNL	Differential nonlinearity*	f_{ADSMP} = 62.5kHz		±2		LSB
		f_{ADSMP} = 250kHz		±2		LSB
		f_{ADSMP} = 500kHz		±5		LSB
INL	Integral Nonlinearity*	f_{ADSMP} = 62.5kHz		±2		LSB
		f_{ADSMP} = 250kHz		±2		LSB
		f_{ADSMP} = 500kHz		±5		LSB
NMC	No missing code*	f_{ADSMP} = 62.5kHz	10	11	12	Bit
		f_{ADSMP} = 250kHz		11		Bit
		f_{ADSMP} = 500kHz		9		Bit
V_{OFFSET}	Input offset voltage	Non-trimmed	-10	0	10	mV
		Trimmed	-2	0	2	mV

* Parameters with star mark: VDD = 5V, V_{REFH} = 2.4V, 25°C.

20.5 Flash存储器特性

Parameter		Test Condition	Min	TYP	MAX	UNIT
V_{dd}	Supply voltage		1.8		5.5	V
T_{en}	Endurance time	25℃		*100K		cycle
Iwrt	Write current	25℃		3	4	mA
T_{wrt}	Write time	Write 1 page = 32 bytes, 25℃		6	8	ms

*Parameters with star mark are non-verified design reference.

21 指令集

本章对 SN8F5702 的综合汇编指令进行分类，总共包括 5 类：算术运算、逻辑运算、数据传送运算、布尔操作和程序分支指令，这些指令全部都与标准 8051 兼容。

符号说明

Symbol	说明
Rn	Working register R0 R7.
direct	One of 128 internal RAM locations or any Special Function register.
@Ri	Indirect internal or external RAM location addressed by register R0 0r R1.
#data	8-bit constant (immediate operand).
#data16	16-bit constant (immediate operand).
Bit	One of 128 software flags located in internal RAM, or any flag of bit-addressedable Special Function Registers.
addr16	Destination address for LCALL or LJMP, can be anywhere within the 64K-byte page of program memory address space.
addr11	Destination address for ACALL or AJMP, within the same 2K-byte page of program memory as the first byte of the following instruction.
rel	SJMP and conditional jumps include an 8-bit offset byte. Its range is +127/-128 bytes relative to the first byte of the following instruction.
A	Accumulator.

运算指令

助记符	描述
ADD A, Rn	A dd register to accumulator.
ADD A, direct	Add directly addressed data to accumulator.
ADD A, @Ri	Add indirectly addressed data to accumulator.
ADD A, #data	Add immediate data to accumulator.
ADDC A, Rn	Add register to accumulator with carry.
ADDC A, direct	Add directly addressed data to accumulator with carry.
ADDC A, @Ri	Add indirectly addressed data to accumulator with carry.
ADDC A, #data	Add immediate data to accumulator with carry.
SUBB A, Rn	Subtract register from accumulator with borrow.
SUBB A, direct	Subtract directly addressed data from accumulator with borrow.
SUBB A, @Ri	Subtract indirectly addressed data from accumulator with borrow.
SUBB A, #data	Subtract immediate data from accumulator with borrow.
INC A	Increment accumulator.
INC Rn	Increment register.
INC direct	Increment directly addressed location.
INC @Ri	Increment indirectly addressed location.
INC DPTR	Increment data pointer.
DEC A	Decrement accumulator.
DEC Rn	Decrement register.
DEC direct	Decrement directly addressed location.
DEC @Ri	Decrement indirectly addressed location.
MUL AB	Multiply A and B.
DIV	Divide A by B.
DA A	Decimally adjust accumulator.

逻辑运算指令

助记符	描述
ANL A, Rn	AND register to accumulator.
ANL A, direct	AND directly addressed data to accumulator.
ANL A, @Ri	AND indirectly addressed data to accumulator.
ANL A, #data	AND immediate data to accumulator.
ANL direct, A	AND accumulator to directly addressed location.
ANL direct, #data	AND immediate data to directly addressed location.
ORL A, Rn	OR register to accumulator.
ORL A, direct	OR directly addressed data to accumulator.
ORL A, @Ri	OR indirectly addressed data to accumulator.
ORL A, #data	OR immediate data to accumulator.
ORL direct, A	OR accumulator to directly addressed location.
ORL direct, #data	OR immediate tdata to directly addressed location.
XRL A, Rn	Exclusive OR (XOR) register to accumulator.
XRL A, direct	XOR directly addressed data to accumulator.
XRL A, @Ri	XOR indirectly addressed data to accumulator.
XRL A, #data	XOR immediate data to accumulator.
XRL direct, A	XOR accumulator to directly addressed location.
XRL direct, #data	XOR immediatedata directly addressed location.
CLR A	Clear accumulator.
CPL A	Complement accumulator.
RL A	Rotate accumulator left.
RLC A	Rotate accumulator left through carry.
RR A	Rotate accumulator right.
RRC A	Rotate accumulator right through carry.
SWAP A	Swap nibbles within the accumulator.

数据传输指令

助记符	描述
MOV A, Rn	Move register to accumulator.
MOV A, direct	Move directly addressed data to accumulator.
MOV A, @Ri	Move indirectly addressed data to accumulator.
MOV A, #data	Move immediate data to accumulator.
MOV Rn, A	Move accumulator to register.
MOV Rn, direct	Move directly addressed data to register.
MOV Rn, #data	Move immediate data to register.
MOV direct, A	Move accumulator to direct.
MOV direct, Rn	Move register to direct.
MOV direct1, direct2	Move directly addressed data to directly addressed location.
MOV direct, @Ri	Move indirectly addressed data to directly addressed location.
MOV direct, #data	Move immediate data to directly addressed location.
MOV @Ri, A	Move accumulator to indirectly addressed location.
MOV @Ri, direct	Move directly addressed data to indirectly addressed location.
MOV @Ri, #data	Move immediate dato to indirecly addressed location.
MOV DPTR, #data16	Load data pointer with a 16-bit immediate.
MOVC A, @A+DPTR	Load accumulator with a code byte relative to DPTR.
MOVC A, @A+PC	Load accumulator with a code byte relative to PC.
MOVX A, @Ri	Move external RAM (8-bit address) to accumulator.
MOVX A, @DPTR	Move external RAM (16-bit address) to accumulator.
MOVX @Ri, A	Move accumulator to external RAM (8-bit address).
MOVX @DPTR, A	Move accumulator to external RAM (16-bit address).
PUSH direct	Push directly addressed data onto stack.
POP direct	Pop directly addressed data location from stack.
XCH A, Rn	Exchange register with accumulator.
XCH A, direct	Exchange directly addressed location with accumulator.
XCH A, @Ri	Exchange indirect RAM with accumulator.
XCHD A, @Ri	Exchange low-order nibbles of indirect and accumulator.

布尔运算指令

助记符	描述
CLR C	Clear carry flag.
CLR bit	Clear directly addressed bit.
SETB C	Set carry flag.
SETB bit	Set directly addressed bit.
CPL C	Complement carry flag.
CPL bit	Complement directly addressed bit.
ANL C, bit	AND directly addressed bit to carry flag.
ANL C, /bit	AND complement of directly addressed bit to carry.
ORL C, bit	OR directly addressed bit to carry flag.
ORL C, /bit	OR complement of directly addressed bit to carry.
MOV C, bit	Move directly addressed bit to carry flag.
MOV bit, C	Move carry flag to directly addressed bit.

程序跳转指令

助记符	描述
ACALL addr11	Absolute subroutine call
LCALL addr16	Long subroutine call
RET	Return from subroutine
RETI	Return from interrupt
AJMP addr11	Absolute jump
LJMP addr16	Long jump
SJMP rel	Short jump (relative address)
JMP @A+DPTR	Jump indirect relative to the DPTR
JZ rel	Jump if accumulator is zero
JNZ rel	Jump if accumulator is not zero
JC rel	Jump if carry flag is set
JNCrel	Jump if carry flag is not set
JB bit, rel	Jump if directly addressed bit is set
JNB bit, rel	Jump if directly addressed bit is not set
JBC bit, rel	Jump if directly addressed bit is set and clear bit
CJNE A, direct, rel	Compare directly addressed data to accumulator and jump if not equal
CJNE A, #data, rel	Compare immediate data to accumulator and jump if not equal
CJNE Rn, #data, rel	Compare immediate data to register and jump if not equal
CJNE @Ri, #data, rel	Compare immediate to indirect and jump if not equal
DJNZ Rn, rel	Decrement register and jump if not zero
DJNZ direct, rel	Decrement directly addressed lacation and jump if not zero
NOP	No operation for one cycle

22 调试界面

调试界面（SWAT），与 GPIO P1.1 共用一个引脚，可更新内置程序存储器 IROM，并可与开发环境配合工作。若单片机在上电前连接到 SN-Link，P1.1 共用引脚会自动设置为调试界面功能；相反，若单片机在上电过程中没有检测到任何握手信号，该引脚会设置为其它功能。

22.1 最低要求

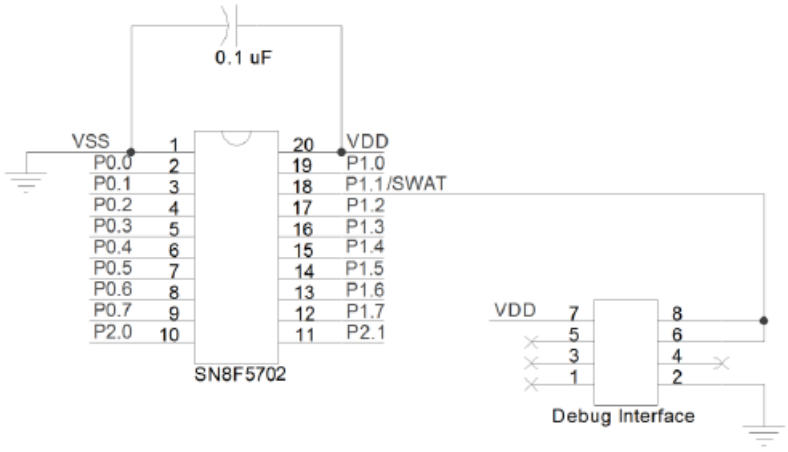
下面几项内容是建立合适的开发环境的基本要求，其兼容性已经经过验证，在后面的版本中可以很好的执行。SN-Link的相关信息可以SONiX网址 www.sonix.com.tw下载，Keil C51 可从 www.keil.com/c51下载。

- **SN-Link Adapter II**：更新到 V3.1。
- **SN-Link Driver for Keil C51**：V1.00.32.
- **Keil C51**：V 9.50a 和 9.54a。

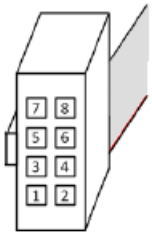
22.2 调试界面硬件

下面的电路图显示了如何正确地连接单片机到 SWAT 引脚和 SN-Link Adapter II。

开始调试之前，必须切断单片机的电源（VDD）。把 SWAT 引脚连接到 SN-Link 的第 6 脚和第 8 脚，SN-Link 的第 2 脚和第 7 脚分别连接 VSS 和 VDD。打开单片机，就会自动开始握手操作，SN-Link 的红色指示灯（Run）显示连接成功。（详细内容请参考 SN8F5000 调试工具使用手册。）



example circuit



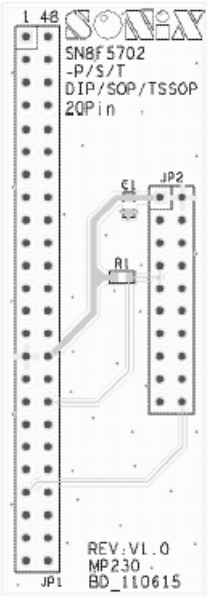
SN-Link header

23 ROM 烧录引脚

SN-LINK 和 MP5 Writer 支持 SN8F5702 系列 Flash ROM 的擦除/烧录/校正。

- **SN-LINK：调试界面。**
- **MP5 Writer：SN8F5702 系列。**

23.1 MP5 Writer转接板引脚配置



JP7 (Mapping to 48-pin text tool)

DIP 1	1	48	DIP48
DIP 2	2	47	DIP47
DIP 3	3	46	DIP46
DIP 4	4	45	DIP45
DIP 5	5	44	DIP44
DIP 6	6	43	DIP43
DIP 7	7	42	DIP42
DIP 8	8	41	DIP41
DIP 9	9	40	DIP40
DIP10	10	39	DIP39
DIP11	11	38	DIP38
DIP12	12	37	DIP37
DIP13	13	36	DIP36
DIP14	14	35	DIP35
DIP15	15	34	DIP34
DIP16	16	33	DIP33
DIP17	17	32	DIP32
DIP18	18	31	DIP31
DIP19	19	30	DIP30
DIP20	20	29	DIP29
DIP21	21	28	DIP28
DIP22	22	27	DIP27
DIP23	23	26	DIP26
DIP24	24	25	DIP25

Writer JP7/JP6

VDD	1	2	GND
-	3	4	-
-	5	6	-
SWAT	7	8	-
SWAT	9	10	-
-	11	12	-
-	13	14	-
-	15	16	-
-	17	18	-
-	19	20	PDB

JP5 for Writer transition board
JP6 for dice and >48 pin package

23.2 MP5 Writer烧录引脚配置

SN8F5702P/S/T （DIP20/SOP20/TSSOP20）

Writer 接口		MCU and JP7 48-pin Text tool 引脚配置		
JP5/JP6 引脚编号	JP5/JP6 引脚名称	MCU 引脚编号	MCU 引脚名称	JP7 引脚编号
1	VDD	20	VDD	34
2	GND	1	VSS	15
7	SWAT	18	P1.1	32
9	SWAT	18	P1.1	32
20	PDB	7	P0.5	21

SN8F5702J（QFN20）

Writer 接口		MCU and JP7 48-pin Text tool 引脚配置		
JP5/JP6 引脚编号	JP5/JP6 引脚名称	MCU 引脚编号	MCU 引脚名称	JP7 引脚编号
1	VDD	18	VDD	32
2	GND	19	VSS	33
7	SWAT	16	P1.1	30
9	SWAT	16	P1.1	30
20	PDB	5	P0.5	19

SN8F570200A (MSOP10)

Writer 接口		MCU and JP7 48-pin Text tool 引脚配置		
JP5/JP6 引脚编号	JP5/JP6 引脚名称	MCU 引脚编号	MCU 引脚名称	JP7 引脚编号
1	VDD	1	VDD	20
2	GND	2	VSS	21
7	SWAT	9	P1.1	28
9	SWAT	9	P1.1	28
20	PDB	5	P0.5	24

SN8F570202S (SOP8)

Writer 接口		MCU and JP7 48-pin Text tool 引脚配置		
JP5/JP6 引脚编号	JP5/JP6 引脚名称	MCU 引脚编号	MCU 引脚名称	JP7 引脚编号
1	VDD	1	VDD	21
2	GND	8	VSS	28
7	SWAT	5	P1.1	25
9	SWAT	5	P1.1	25
20	PDB	3	P0.5	23

SN8F570210S (SOP14)

Writer 接口		MCU and JP7 48-pin Text tool 引脚配置		
JP5/JP6 引脚编号	JP5/JP6 引脚名称	MCU 引脚编号	MCU 引脚名称	JP7 引脚编号
1	VDD	14	VDD	31
2	GND	1	VSS	18
7	SWAT	12	P1.1	29
9	SWAT	12	P1.1	29
20	PDB	5	P0.5	22

SN8F570211S (SOP14)

Writer 接口		MCU and JP7 48-pin Text tool 引脚配置		
JP5/JP6 引脚编号	JP5/JP6 引脚名称	MCU 引脚编号	MCU 引脚名称	JP7 引脚编号
1	VDD	14	VDD	31
2	GND	1	VSS	18
7	SWAT	12	P1.1	29
9	SWAT	12	P1.1	29
20	PDB	6	P0.5	23

SN8F570212S/T (SOP16/TSSOP16)

Writer 接口		MCU and JP7 48-pin Text tool 引脚配置		
JP5/JP6 引脚编号	JP5/JP6 引脚名称	MCU 引脚编号	MCU 引脚名称	JP7 引脚编号
1	VDD	16	VDD	32
2	GND	1	VSS	17
7	SWAT	14	P1.1	30
9	SWAT	14	P1.1	30
20	PDB	6	P0.5	22

SN8F570213S (SOP14)

Writer 接口		MCU and JP7 48-pin Text tool 引脚配置		
JP5/JP6 引脚编号	JP5/JP6 引脚名称	MCU 引脚编号	MCU 引脚名称	JP7 引脚编号
1	VDD	1	VDD	18
2	GND	14	VSS	31
7	SWAT	12	P1.1	29
9	SWAT	12	P1.1	29
20	PDB	5	P0.5	22

24 订购信息

SONiX 的单片机的表面印有三栏信息：商标，单片机全称和日期码。

SONiX Logo

Full Name

SN8F 5702 JG
8-bit MCU | Device Series | Package

Date Code

15 5 J AEB11
Year | Mo. | Date | Internal Usage

○

24.1 命名规则

单片机全称	封装类型
SN8F5702W	Wafer
SN8F5702H	Dice
SN8F5702PG	PDIP，20 脚，绿色封装
SN8F7502SG	SOP，20 脚，绿色封装
SN8F5702TG	TSSOP，20 脚，绿色封装
SN8F5702JG	QFN，20 脚，绿色封装
SN8F570200AG	MSOP，10 脚，绿色封装
SN8F570210SG	SOP，14 脚，绿色封装
SN8F570211SG	SOP，14 脚，绿色封装
SN8F570212SG	SOP，16 脚，绿色封装
SN8F570212TG	TSSOP，16 脚，绿色封装
SN8F570213SG	SOP，14 脚，绿色封装
SN8F570202SG	SOP，8 脚，绿色封装

24.2 日期码

日期码包括 2 个信息：生产日期和产品序列号。生产日期是公共信息，详见下表。

Year	15: 2015 16: 2016 17: 2017 et cetera
Month	1: January 2: February 3: March A: October B: November C: December et cetera
Date	1: 01 2: 02 3: 03 A: 10 B: 11 et cetera

25 附录：参考文档

SONiX为用户提供了参考文档，可以更快地熟悉SN8F5000 的性能。（从SONIX官网：www.sonix.com.tw下载）

文档名称	文档说明
SN8F5000 Starter-Kit User Manual	该文档介绍 SN8F5000 系列的 Starter-Kit，帮用户先选择合适的 Starter-Kit 进行产品开发。
SN8F5000 Family Instruction Set	该文档详细介绍 8051 的指令集，并进行简单示例说明。
SN8F5000 Family Instruction Mapping Table	该文档介绍 8-bit Flash/OTP Type 与 8051 Flash Type 相对应的指令。
SN8F5000 Package Information	该文档介绍 SN8F5000 系列单片机的机械数据，如高度、宽度和倾斜度等信息。
SN8F5000 Debug Tool Manual	该文档教用户按照 Keil C51 软件，并生产一个新的工程用于产品开发。

SN8F5702 Series

Datasheet

8051-based Microcontroller

公司总部
台湾新竹县竹北市台元街 36 号 10 楼之一（台元科技园区）
TEL：+886-3-5600888
FAX：+886-3-5600889

台北办事处
台北市松德路 171 号 15 楼之 2
TEL：+886-2-27591980
FAX：+886-2-27598180
mkt@sonix.com.tw
sales@sonix.com.tw

香港办事处
香港新界沙田火炭禾盛街 11 号，中建电讯大厦 26 楼 03 室
TEL：+852-2723-8086
FAX：+852-2723-9179
hk@sonix.com.tw

松翰深圳
中国广东省深圳市高新科技园区
TEL：+86-755-2671-9666
FAX：+86-755-2671-9786
mkt@sonix.com.tw
sales@sonix.com.tw

USA Office
TEL: +1-714-3309877
TEL: +1-949-4686539
tlightbody@earthlink.net

Japan Office
2F, 4 Chome-8-27Kudanminami
Chiyoda-ku,Tokyo, Japan
TEL: +81-3-6272-6070
FAX: +81-3-6272-6165
jpsales@sonix.com.tw

FAE Support via email
8-bit Microcontroller Products:
sa1fae@sonix.com.tw
All Products: fae@sonix.com.tw

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for Microprocessors - MPU ***category:***

Click to view products by SONIX ***manufacturer:***

Other Similar products are found below :

MCIMX6D5EYM12AD A2C00010998 A ALXD800EEXJCVD C3 LS1020ASE7KQB LS1020AXE7KQB A2C00010729 A T1022NSE7MQB T1024NXE7PQA T1042NSN7WQB MPC8313EVRADDC BOXSTCK1A8LFCL LS1021ASE7KQB LS1021ASN7KQB MPC855TZQ80D4 MPC8569VJAUNLB P5020NSN7QMB P5020NXE7TNB T1024NXN7MQA T2080NXE8MQB T2080NXN8PTB MCIMX6L3EVN10AB T2080NXE8PTB T1024NXE7MQA CM8063501521600S R19L LS1043AXE7MQB T1024NXN7PQA LS1043ASE7QQB LS1012AXE7HKA T4240NSN7PQB MVF30NN152CKU26 FH8067303534005S R3ZM R9A07G044L24GBG#AC0 SVF311R3K2CKU2 HW8076502640002S R38F R7S721030VLFP#AA0 M0516LBN MCF5208CVM166 MCIMX6S6AVM08AC MCIMX6U5DVM10AC TEN54LSDV23GME MC68302AG33C MC68302EH16C MCF5233CVM150 MCIMX6D6AVT10AD MCIMX6G1CVM05AB MPC8245LZU350D MPC8280VVQLDA MPC8314ECVRAGDA MPC8314VRAGDA MPC8315VRAGDA