SONiX Technology Co., Ltd.

SN8F5703 Series

Datasheet

8051-based Microcontroller

SN8F5703
SN8F570320
SN8F570321
SN8F570310
SN8F570311

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 8MHz 灵活的 CPU 频率
  - 内部 32MHz 时钟发生器(IHRC)，1MHz 到 16MHz 晶振，外部同步时钟源
  - 实时时钟，带 32.768KHz 晶振
- ◆ **8 KB Flash 存储器（IROM），**支持在线编程功能
- ◆ **256 字节内部 RAM（IRAM）**
- ◆ **256 字节外部 RAM（XRAM）**
- ◆ **15 个中断源 :可控制中断的优先等级以及独立的中断向量**
  - 13 个内部中断
  - 2 个外部中断：INT0，INT1
  - 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 16 位 PWM 发生器**
  - 每组 PWM 都有 6 个输出通道，带有反相器和死区控制功能
- ◆ **12 位 SAR ADC，**包括 11 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**
  - 单线调试窗口
  - 3 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
  - 1.8V 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

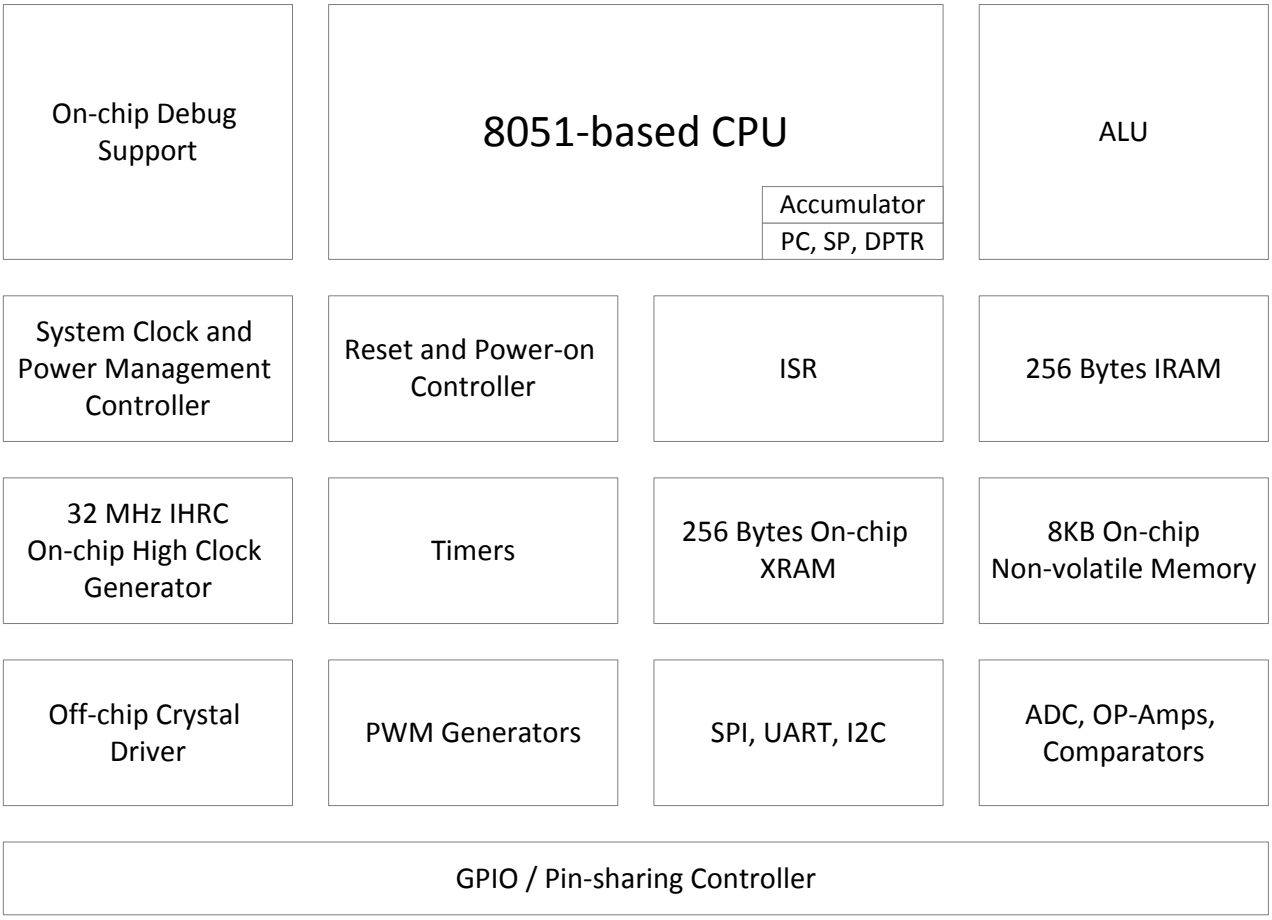
1.2 应用领域

- 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
SN8F5703	22	10	V	V	V	11	1	1	2	SOP24,SSOP24, TSSOP24, QFN24
SN8F570320	18	6	V	-	V	10	1	1	2	DIP20,SOP20,TSSOP20
SN8F570321	18	8	V	V	V	9	1	1	2	QFNP20
SN8F570310	14	3	V	-	V	7	1	1	2	DIP16,SOP16
SN8F570311	14	7	V	V	V	5	-	-	1	QFN16

1.4 结构框图



2 目录

3 修订记录

版本	时间	修订说明
1.0	Sep. 2015	1. 初版。
1.1	Oct. 2015	1. 调整定时器和电气特性章节。 2. 增加程序存储器加密章节和 noise filter 章节。 3. 增加特殊功能寄存器章节。
1.2	Oct. 2015	1. 更新电器特性章节。 2. 增加 TSSOP24 脚位图。
1.3	Nov. 2015	1. SN8F57031 更名为 SN8F570320。 2. SN8F57032 更名为 SN8F570310。
1.4	Nov. 2015	1. 调整 SN8F570310 脚位图。
1.5	Dec. 2015	1. 调整 IHRC 特性。
1.6	Apr. 2016	1. 增加定时器 2 捕获功能波形图。 2. 调整 OPA 特性章节。 3. 在特殊功能寄存器章节添加寄存器宣告章节。 4. 增加附录：参考文档章节。 5. 增加 ROM 烧录引脚章节。 6. 增加 QFN 24 脚位图。 7. 修改错误，遗漏等。
1.7	Aug. 2016	1. 修改错误，遗漏等。 2. 调整电源管理和在线编程章节。 3. 调整 PW1M & PW1YH/L 寄存器说明。 4. ADC & 比较器特性章节增加 VIREF 参数。
1.8	Oct. 2016	1. 增加 UART 波特率表。 2. 看门狗复位章节增加 WDT 说明。
1.9	Dec. 2016	1. 调整性能说明。 2. 调整电气特性章节。 3. 增加 SN8F570321 (QFN20)和 SN8F570311 (QFN 16)脚位图。
2.0	Aug. 2017	1. 修改错误，遗漏等。 2. 调整产品性能表。 3. 调整 UART 波特率控制章节。 4. 调整 PFLAG 寄存器初始值。 5. 更新寄存器宣告章节。 6. 增加引脚电路图。
2.1	Sep. 2017	1. 增加封装信息。

2.2	Nov. 2017	1. 调整 LVD 相关内容。
2.3	Dec.2017	1. 增加设计注释描述。
2.4	Jun.2018	1. 修改错误，遗漏，等。 2. 增加引脚特性部分。 3. 修改内部&外部 RAM 选择描述。 4. 修改程序存储器部分描述。 5. 修改复位配置与上电控制部分描述。 6. 修改系统时钟选择描述。 7. 增加高速时钟和实际时钟部分。 8. 增加系统时钟校对部分。 9. 增加系统操作模式章节。 10. 修改中断优先级部分描述。 11. 增加中断章节示例代码。 12. 修改 UART 章节描述和波特率表。 13. IIC 章节增加协议描述图并修改时钟频率表。 14. 调试界面章节被重新命名为开发环境章节。修改开发环境章节描述。增加开发工具部分。 15. 增加 SN8F5703 Start-kit 章节。 16. 修改 ROM 编程引脚章节描述。增加 MP5 硬件连接，SN-LINK ISP 编程，SN-LINK ISP 编程引脚图部分。 17. 更新设备命名部分。
2.5	Sep.2018	1. 修改错误，遗漏，等。 2. 修改 SPI 章节描述。 3. 修改 SOP24 封装描述。
2.6	Oct.2018	1. 修改错误，遗漏，等。 2. 修改系统时钟部分描述。 3. 修改 normal 模式功耗值。 4. 删除 SN8F5703K 引脚分配和 SKDIP24 的封装信息。 5. 修改引脚电路图部分。
2.7	Feb.2019	1. 修改错误，遗漏，等。 2. 更新封装尺寸信息和说明。 3. 修改 ADC 输入补偿范围。 4. 修改上电顺序和系统时钟。 5. 修改封装信息章节。 6. 修改 SPI 操作章节描述。 7. 修改 T0/T1 章节描述。

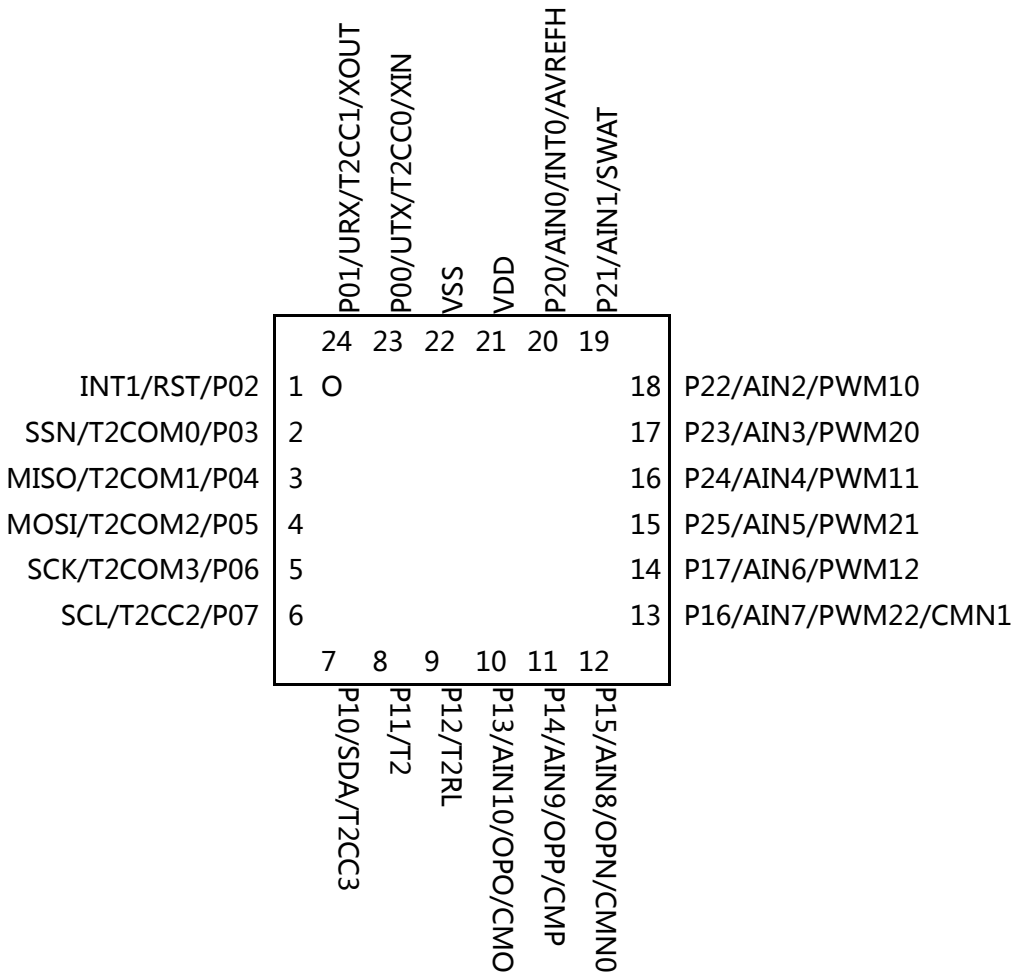
SONiX 公司保留对以下所有产品在可靠性，功能和设计方面的改进作进一步说明的权利。SONiX 不承担由本手册所涉及的产品或电路的运用和使用所引起的任何责任，SONiX 的产品不是专门设计来应用于外科植入、生命维持和任何 SONiX 产品的故障会对个体造成伤害甚至死亡的领域。如果将 SONiX 的产品应用于上述领域，即使这些是由 SONiX 在产品设计和制造上的疏忽引起的，用户应赔偿所有费用、损失、合理的人身伤害或死亡所直接或间接产生的律师费用，并且用户保证 SONiX 及其雇员、子公司、分支机构和销售商与上述事宜无关。

4 引脚配置

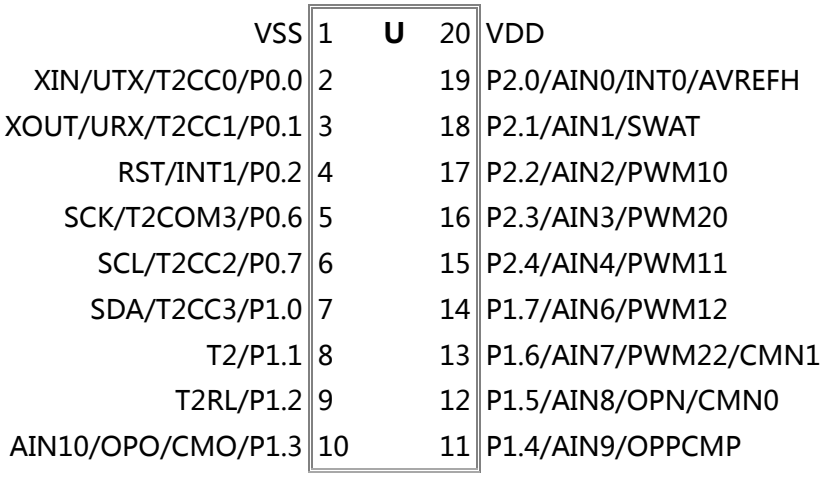
4.1 SN8F5703S/X/T (SOP24/SSOP24/TSSOP24)

VSS	1	**U**	24	VDD
XIN/UTX/T2CC0//P0.0	2		23	P2.0/AIN0/INT0/AVREFH
XOUT/URX/T2CC1/P0.1	3		22	P2.1/AIN1/SWAT
RST/INT1/P0.2	4		21	P2.2/AIN2/PWM10
SSN/T2COM0/P0.3	5		20	P2.3/AIN3/PWM20
MISO/T2COM1/P0.4	6		19	P2.4/AIN4/PWM11
MOSI/T2COM2/P0.5	7		18	P2.5/AIN5/PWM21
SCK/T2COM3/P0.6	8		17	P1.7/AIN6/PWM12
SCL/T2CC2/P0.7	9		16	P1.6/AIN7/PWM22/CMN1
SDA/T2CC3/P1.0	10		15	P1.5/AIN8/OPN/CMN0
T2/ P1.1	11		14	P1.4/AIN9/OPP/CMP
T2RL/P1.2	12		13	P1.3/AIN10/OPO/CMO

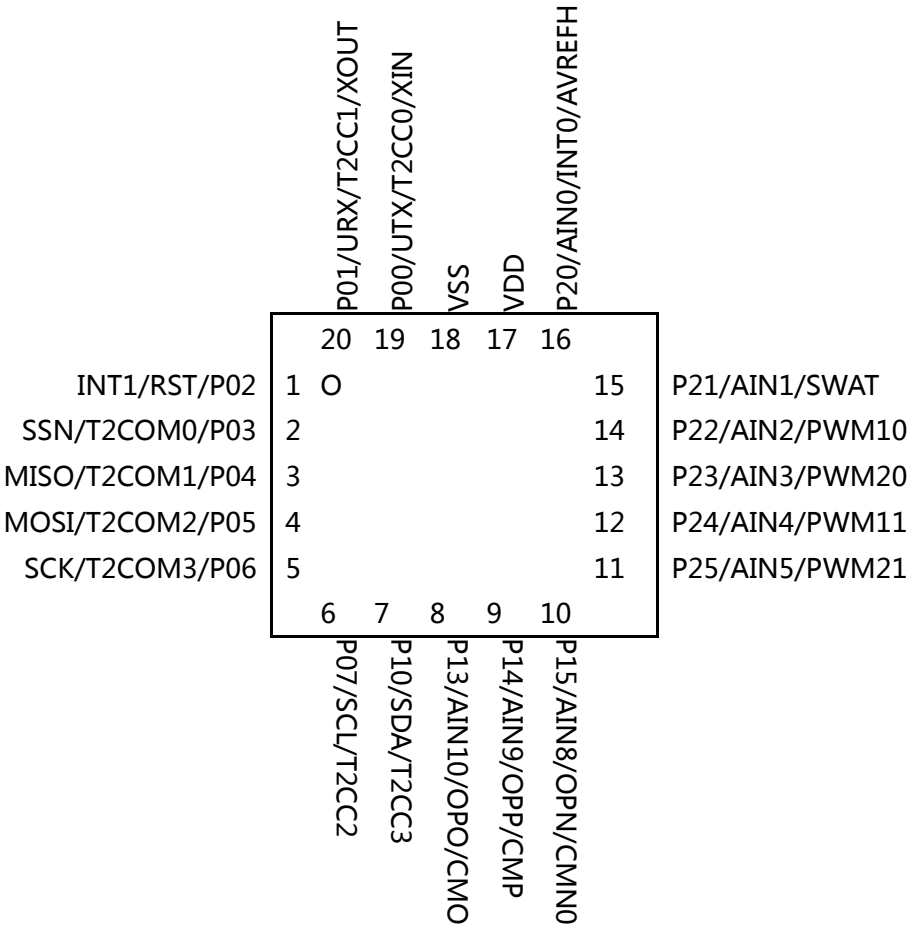
4.2 SN8F5703J (QFN24)



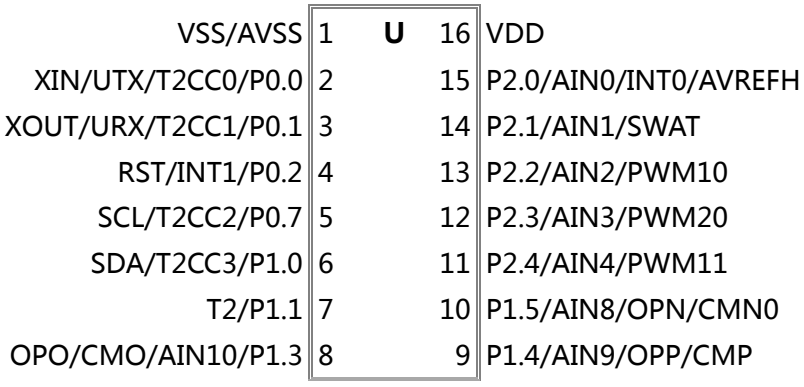
4.3 SN8F570320P/S/T (DIP20/SOP20/TSSOP20)



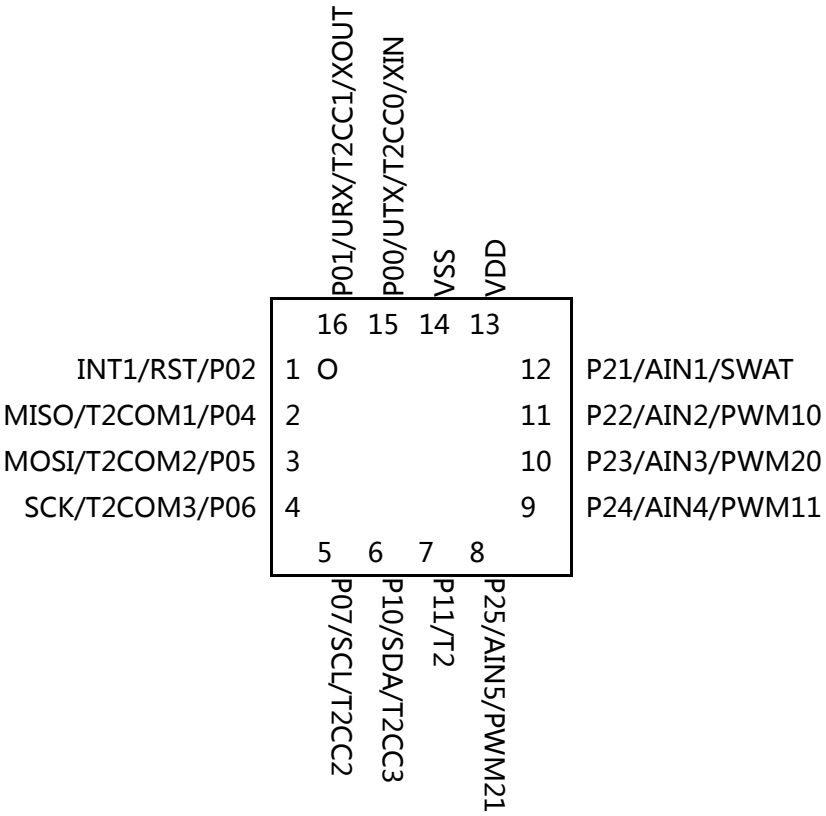
4.4 SN8F570321J (QFN20)



4.5 SN8F570310P/S (DIP16/SOP16)



4.6 SN8F570311J (QFN16)



4.7 引脚说明

电源引脚

引脚名称	类型	功能说明
VDD	Power	电源（1.8~5.5V）
VSS	Power	接地（0V）

P0 端口

引脚名称	类型	功能说明
P0.0	Digital I/O	GPIO
XIN	Analog Input	系统时钟：外部时钟输入
UTX	Digital Output	UART：数据输出引脚
T2CC0	Digital Input	Timer2：比较器输入
P0.1	Digital I/O	GPIO
XOUT	Analog Output	系统时钟：外部晶振输出
URX	Digital Input	UART：数据接收引脚
T2CC1	Digital Input	Timer2：比较器输入
P0.2	Digital I/O	GPIO
Reset	Digital Input	系统复位（低电平有效）
INT1	Digital Input	INT1：外部中断
P0.3	Digital I/O	GPIO
SSN	Digital Input	SPI：从动选择引脚（从动模式）
T2COM0	Digital Output	Timer 2：比较器输出
P0.4	Digital I/O	GPIO
MISO	Digital I/O	SPI：接收引脚（主机模式）发送引脚（从机模式）
T2COM1	Digital Output	Timer 2：比较器输出
P0.5	Digital I/O	GPIO
MOSI	Digital I/O	SPI：发送引脚（主机模式）接收引脚（从机模式）
T2COM2	Digital Output	Timer 2：比较器输出
P0.6	Digital I/O	GPIO
SCK	Digital I/O	SPI：时钟输出（主机模式）时钟输入（从机模式）
T2COM3	Digital Output	Timer 2：比较器输出
P0.7	Digital I/O	GPIO
SCL	Digital I/O	I2C：时钟输出（主机模式）时钟输入（从机模式）
T2CC2	Digital Input	Timer2：比较器输入

P1 端口

引脚名称	类型	功能说明
P1.0	Digital I/O	GPIO
SDA	Digital I/O	I2C：数据引脚
T2CC2	Digital Input	Timer2：比较器输入
P1.1	Digital I/O	GPIO
T2	Digital Input	Timer 2：事件计数器输入引脚
P1.2	Digital I/O	GPIO
T2RL	Digital Input	Timer 2：重装触发输入引脚
P1.3	Digital I/O	GPIO
AIN10	Analog Input	ADC：输入通道
OPO	Analog Output	OPA：输出
CMO	Digital Output	Comparator 0：输出
P1.4	Digital I/O	GPIO
AIN9	Analog Input	ADC：输入通道
OPP	Analog Input	OPA：正极输入
P1.5	Digital I/O	GPIO
AIN8	Analog Input	ADC：输入通道
OPN	Analog Input	OPA：负极输入
CMN0	Analog Input	Comparator：负极输入 0
P1.6	Digital I/O	GPIO
AIN7	Analog Input	ADC：输入通道
PWM22	Digital Output	PWM：可编程的 PWM 输出
CMN1	Analog Input	Comparator：负极输入 1
P1.7	Digital I/O	GPIO
AIN6	Analog Input	ADC：输入通道
PWM12	Digital Output	PWM：可编程的 PWM 输出

P2 端口

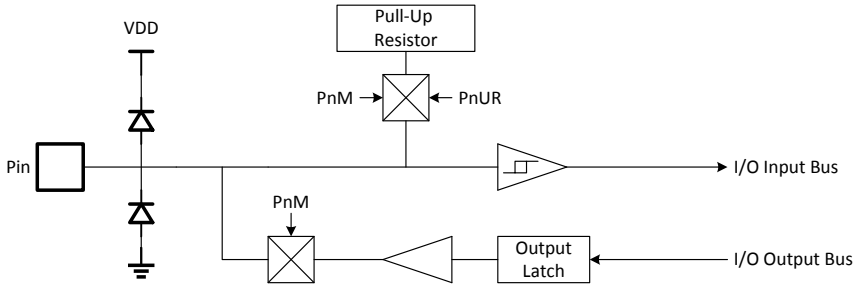
引脚名称	类型	功能说明
P2.0	Digital I/O	GPIO
AIN0	Analog Input	ADC：输入通道
INT0	Digital Input	INT0：外部中断
AVREFH	Analog Input	ADC：外部高参考电压
P2.1	Digital I/O	GPIO
AIN1	Analog Input	ADC：输入通道
SWAT	Digital I/O	单线仿真接口
P2.2	Digital I/O	GPIO
AIN2	Analog Input	ADC：输入通道
PWM10	Digital Output	PWM：可编程的 PWM 输出
P2.3	Digital I/O	GPIO
AIN3	Analog Input	ADC：输入通道
PWM20	Digital Output	PWM：可编程的 PWM 输出
P2.4	Digital I/O	GPIO
AIN4	Analog Input	ADC：输入通道
PWM11	Digital Output	PWM：可编程的 PWM 输出
P2.5	Digital I/O	GPIO
AIN5	Analog Input	ADC：输入通道
PWM21	Digital Output	PWM：可编程的 PWM 输出

4.8 引脚特性

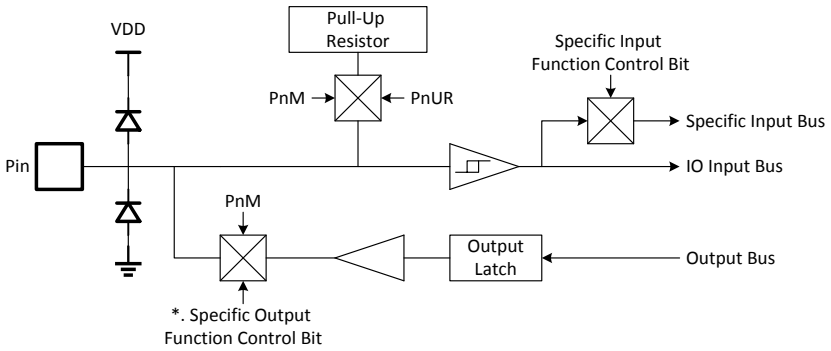
Port	Open-Drain	Sink Current 100mA VSS+1.5V	Sink Current 20mA VSS+0.5V	External Interrupt	Wakeup (Level change)	Shared Pin
P0.0	V	V	-	-	V	UTX/T2CC0/XIN
P0.1	V	V	-	-	V	URX/T2CC1/XOUT
P0.2	-	V	-	V	V	INT1/RST
P0.3	-	V	-	-	V	SSN/T2COM0
P0.4	V	V	-	-	V	MISO/T2COM1
P0.5	V	V	-	-	V	MOSI/T2COM2
P0.6	V	V	-	-	V	SCK/T2COM3
P0.7	-	V	-	-	V	SCL/T2CC2
P1.0	-	V	-	-	V	SDA/T2CC3
P1.1	-	-	V	-	V	T2
P1.2	-	-	V	-	V	T2RL
P1.3	-	-	V	-	V	AIN10/OPO/CMO
P1.4	-	-	V	-	V	AIN9/OPP/CMP
P1.5	-	-	V	-	V	AIN8/OPN/CMN0
P1.6	-	-	V	-	V	AIN7/PWM22/CMN1
P1.7	-	-	V	-	V	AIN6/PWM12
P2.0	-	-	V	V	-	AIN0/INT0/AVREFH
P2.1	-	-	V	-	-	AIN1/SWAT
P2.2	-	-	V	-	-	AIN2/PWM10
P2.3	-	-	V	-	-	AIN3/PWM20
P2.4	-	-	V	-	-	AIN4/PWM11
P2.5	-	-	V	-	-	AIN5/PWM21

4.9 4.9 引脚电路图

GPIO

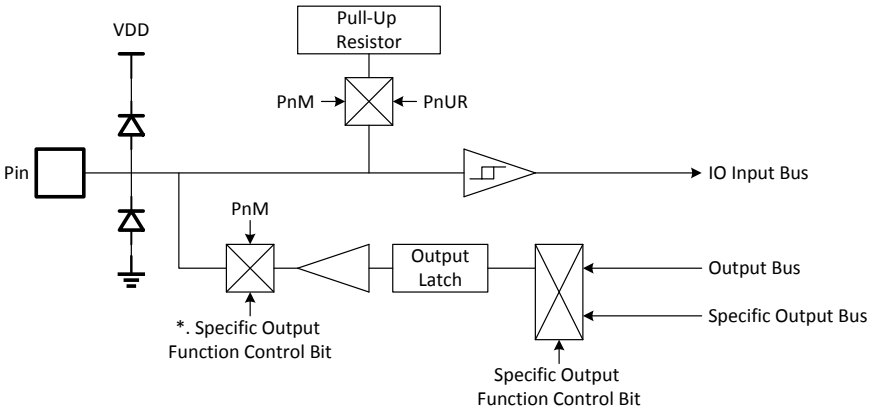


双向 I/O，与特殊数字输入功能引脚共用，如 INT2。



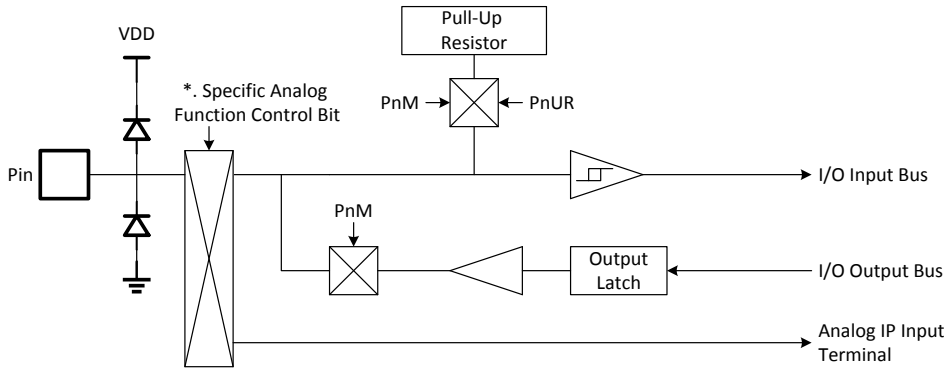
***. Some specific functions switch I/O direction directly, not through PnM register.**

双向 I/O，与特殊数字输出功能引脚共用，如 PWM, SIO,UART。



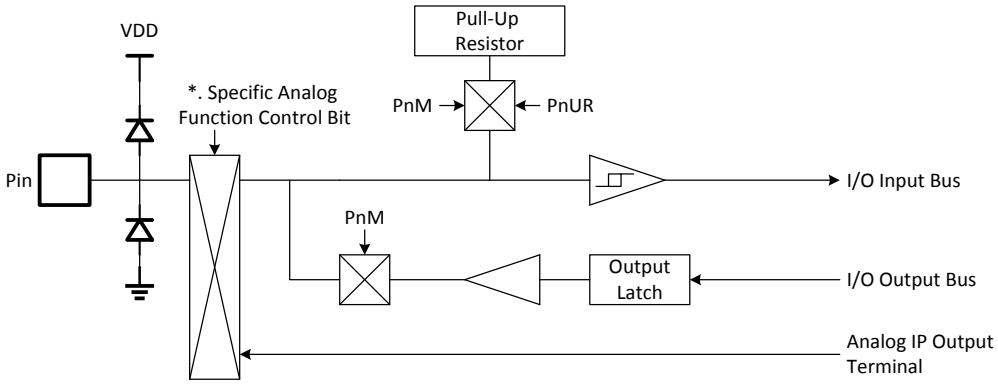
***. Some specific functions switch I/O direction directly, not through PnM register.**

双向 I/O，与特殊模拟输入功能引脚共用，如 XIN, ADC。



***. Some specific functions switch I/O direction directly, not through PnM register.**

双向 I/O，与特殊模拟输出功能引脚共用，如 XOUT...



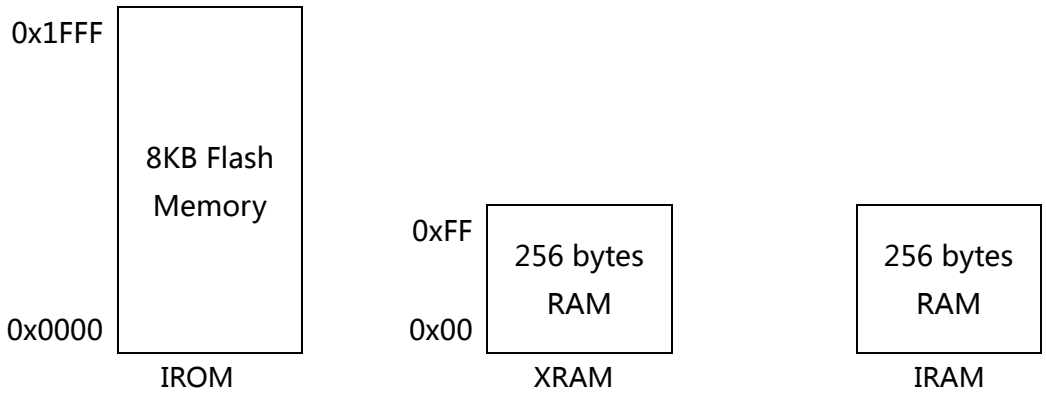
***. Some specific functions switch I/O direction directly, not through PnM register.**

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

SN8F5703 内建了三个存储器：内部 RAM(IRAM)，外部 RAM(XRAM)，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）。外部 RAM 由 256 个字节组成，但需要更长的存取周期。程序存储器是一个 8KB 的 FLASH 存储器，最快的存取速度可达 8MHz。



5.2 内部RAM（IRAM）

256 x 8bit RAM(内部数据存储器)

地址	RAM 位置	
000h 01Fh	工作寄存器区	
020h 02Fh	位可寻址区	
030h … … 07Fh	通用区	
080h … … … 0FFh	通用区 (间接寻址)	特殊功能寄存 (直接寻址)

00h-7Fh 的 RAM 支持直接寻址或间接寻址

Bank0 结束

内部数据存储器的 256 字节 RAM 是一个标准的 8051 RAM 访问配置。以下为存储器空间分配：

- 0x00-0x7F：低 128 字节的内部 RAM 可直接寻址或间接寻址；
- 0x80-0xFF：高 128 字节的内部 RAM 只能间接寻址；

高 128 字节内部 RAM 占用的地址空间与特殊功能寄存器(SFR)相同，但在物理上与 SFR 的空间是分离的。当一个指令访问地址高于 0x7F 时，CPU 可以根据指令的寻址方式访问高 128 字节的数据 RAM 还是访问 SFR。

- 0x80-0xFF：特殊功能寄存器(SFR)只能直接寻址；

低 128 字节包含工作寄存器区域与位可寻址区域：

- 0x00-0x1F：工作寄存器区包含 4 个 bank ,每个 bank 有 8 个工作寄存器(R0-R7),由寄存器 PSW 的 RS0/RS1 位选择；
- 0x20-0x2F：位可寻址区域(地址)；

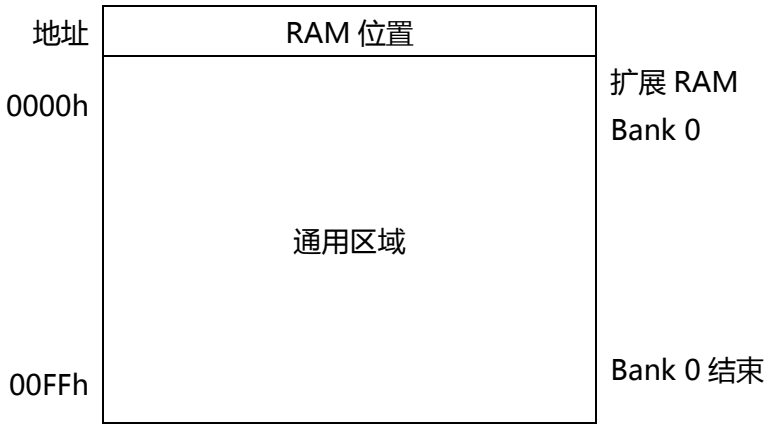
  在位可寻址区域，用户可以使用唯一的地址来读或写任何一个位。支持 16 字节的位可寻址存储区域，提供 128 个可寻址位。在 0x00 到 0x7F 区域内每个位都有单独的地址，因此都可直接位寻址。字节 0x20 内存的 BIT0 位地址为 0x00，BIT7 位地址为 0x07。字节 0x2F 内存的 BIT0 位地址为 0x78，BIT7 位地址为 0x7F。当设置“SETB 42H”，意思是设置字节 0x28 的 BIT2 位地址。

	Byte Address	Bite 0	Bite 1	Bite 2	Bite 3	Bite 4	Bite 5	Bite 6	Bite 7
Bit Addressable Area	**0x20**	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
	0x21	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
	0x22	0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17
	0x23	0x18	0x19	0x1A	0x1B	0x1C	0x1D	0x1E	0x1F
	0x24	0x20	0x21	0x22	0x23	0x24	0x25	0x26	0x27
	0x25	0x28	0x29	0x2A	0x2B	0x2C	0x2D	0x2E	0x2F
	0x26	0x30	0x31	0x32	0x33	0x34	0x35	0x36	0x37
	0x27	0x38	0x39	0x3A	0x3B	0x3C	0x3D	0x3E	0x3F
	0x28	0x40	0x41	0x42	0x43	0x44	0x45	0x46	0x47
	0x29	0x48	0x49	0x4A	0x4B	0x4C	0x4D	0x4E	0x4F
	0x2A	0x50	0x51	0x52	0x53	0x54	0x55	0x56	0x57
	0x2B	0x58	0x59	0x5A	0x5B	0x5C	0x5D	0x5E	0x5F
	0x2C	0x60	0x61	0x62	0x63	0x64	0x65	0x66	0x67
	0x2D	0x68	0x69	0x6A	0x6B	0x6C	0x6D	0x6E	0x6F
	0x2E	0x70	0x71	0x72	0x73	0x74	0x75	0x76	0x77
	0x2F	0x78	0x79	0x7A	0x7B	0x7C	0x7D	0x7E	0x7F

5.3 外部RAM（XRAM）

256 x 8bit XRAM(扩展数据存储器)。

外部 RAM 增加了变量的容量，但跟内部 RAM 相比，它的访问能力是最低的，这是因为频繁地使用变量和本地变量是为了存入内部 RAM，而外部 RAM 的绝大多数用法都是特定的。外部 RAM 可以作为优先级别较低的，或者在 ROM 中预装查表以加速访问周期的变量存储区域。



256 字节的外部 RAM 支持传统访问外部 RAM 的方法。使用 MOVX A,@Ri 或 MOVX @Ri,A 来访问外部低 256 字节 RAM；用 MOVX A,@DPTR 或 MOVX @DPTR,A 来访问外部 256 字节。

5.4 程序存储器（IROM）

程序存储器是 FLASH 存储器，可以存储程序代码，ROM 查表数据以及其它的临时修改数据。可通过调试工具如 SN-Link3 进行升级，也可以通过在线程序处理来自行更新（参考在线编程章节）。

地址	RAM	注释
0000h	复位向量	复位向量
0001h 0002h	通用区域	用户程序
0003h	INT0	中断向量
000Bh	T0	
0013h	INT1	
001Bh	T1	
0023h	UART	
002Bh	T2	
0043h	IIC	
004Bh	SPI	
0053h	T2COM0	
005Bh	T2COM1	
0063h	T2COM2	
006Bh	T2COM3	
0083h	PWM1	
008Bh	ADC	
0093h	比较器	
0094h	通用区域	用户程序
.		
.		用户程序
.		结束
1FF6h 1FF7h 1FFEh 1FFFh	保留	

ROM 包含复位向量，中断向量，通用区域和保留区域。复位向量是程序的起始地址，中断向量是当任何中断发生时的中断服务程序索引，通用区域是主程序区域包含主循环，子程序，数据表。

- 0x0000 复位向量：程序指针指向 0x0000 当发生复位事件后(上电复位 ,外部复位 ,看门狗复位 ,LVD 复位...)；
- 0x0001-0x0002：通用区域处理系统复位操作；
- 0x0003-0x0093：多中断向量区域，每个中断事件都有一个唯一的中断向量；
- 0x0094-0x1FDF：用户程序和 ISP 通用区域；
- **0x1FE0-0x1FF5：用户通用区域程序，禁止执行 ISP；**
- **0x1FF6-0x1FFF：保留区域，禁止执行 ISP；**

5.5 程序存储器安全

SN8F5703 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 0x00 的数据。

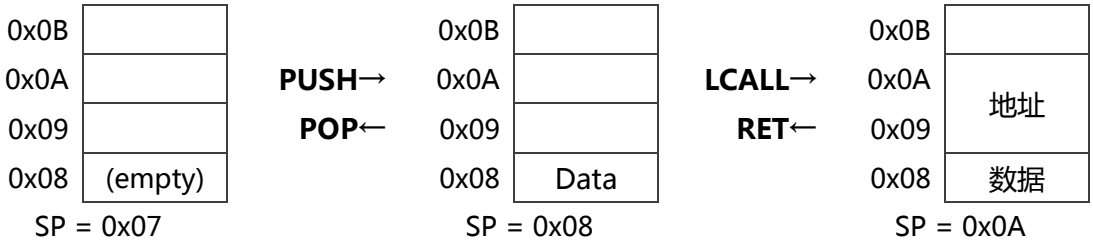
5.6 数据指针

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其它 RAM 的变量重叠。堆栈溢出或者下溢会导致其它 RAM 的变量重写错误，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。

✷ **注：堆栈与内部RAM(IRAM)共用相同的内存空间，当程序占用太多的IRAM空间而造成堆栈不够时Keil C51编译器不会提示"错误"或"警告"，因此用户必须注意这种情况。**

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：禁止 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	P0OC	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	CMPT	-
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	OPM	CMPM	-	P2CON	-
90	P1	P1W	-	DPC	PECMD	PEROML	PERONH	PERAM
88	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON	PEDGE
80	P0	SP	DPL	DPH	-	-	WDTR	PCON

□*注：最左边列所有的特殊功能寄存器都是位可寻址（每个**0x0/0x8**结尾的**SFR**地址都是位可寻址） 。

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	-
DPC	093H	数据指针控制寄存器。
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。
OPM	09BH	OP-AMP 控制寄存器。
CMPM	09CH	比较器 CMP 控制寄存器。
-	09DH	-
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。
PW1CH	0BEH	PW1 通道使能寄存器
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。
CMPT	0CEH	比较器 CMP0/1 带 PWM 触发选择寄存器。
-	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

SN8F5703 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 <SN8F5703.H>
```

使用 C 运用进行编程是，增加下面的句子：

```
1 #include <SN8F5703.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
SN8F5703	SN8F5703.H	OPTIONS_SN8F5703.A51
SN8F570320	SN8F570320.H	
SN8F570321	SN8F570321.H	
SN8F570310	SN8F570310.H	
SN8F570311	SN8F570311.H	

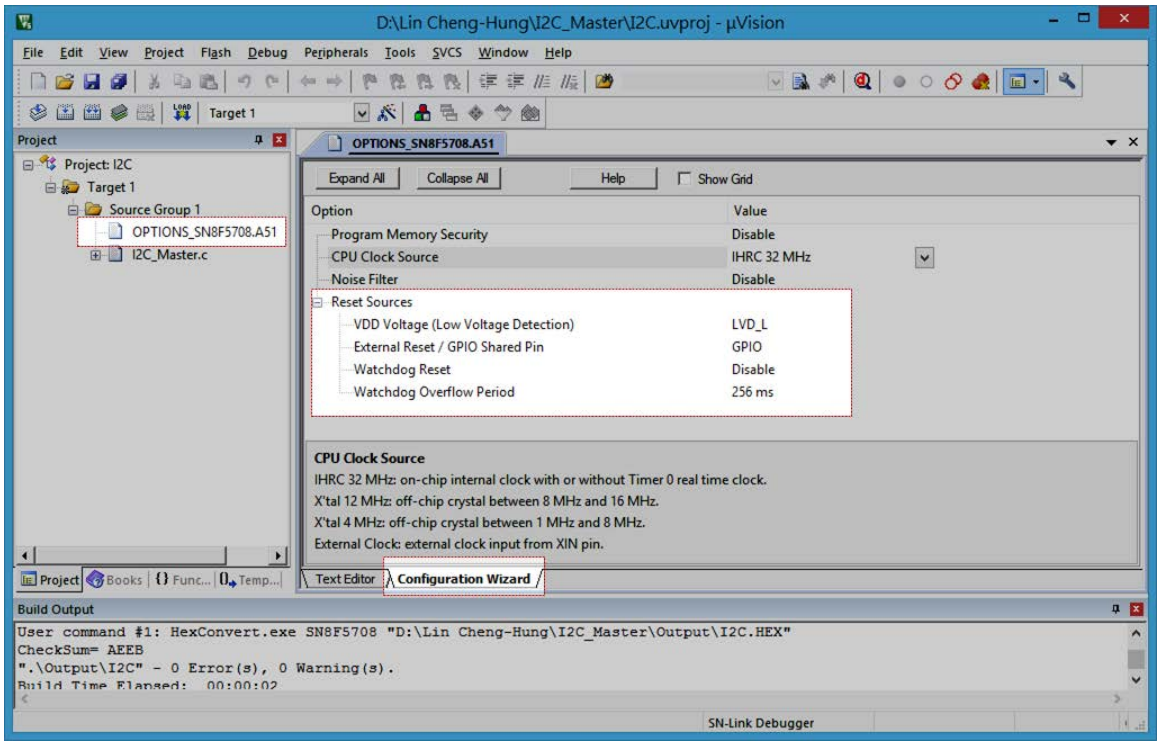
7 复位和上电控制

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，同时程序从复位向量（ROM 地址 0x0000）处重新开始执行。

7.1 复位配置和上电控制

SONiX 发布了一个 SN8F5703_OPTIONS.A51 文件，该文件包含在 SN8F5703 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。

- 程序内存加密
- 时钟源
- Noise Filter
- 复位源：VDD 电压（低电压检测）
- 复位源：外部复位/ GPIO 共用引脚
- 复位源：看门狗复位&溢出周期

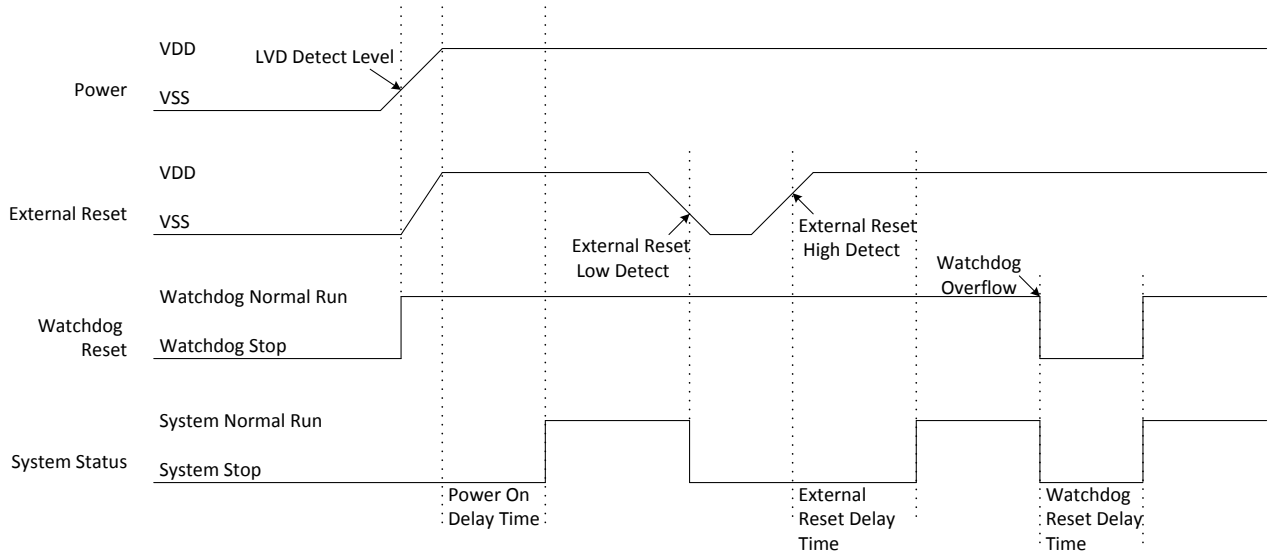


Code option 是系统硬件配置包含：晶振类型，Noise Filter 选项，看门狗时间操作，LVD 选项和 Flash ROM 加密控制。

Code option	选项	功能描述
Program Memory Security	Security Disable	禁用 ROM 代码加密功能
	Security Enable	使能 ROM 代码加密功能
CPU Clock Source	IHRC 32MHz	内部高速 32M RC，XIN/XOUT 为普通双向 IO
	IHRC 32MHz with RTC	内部高速 32M RC 带低速晶振（例：32.768khz）。低速晶振用于定时器 T0 的实时时钟
	X’tal 12MHz□□□□	高速晶振（例：12MHz）外部高速晶振/谐振器
	X’tal 4MHz	标准晶振（例：4MHz）外部高速晶振/谐振器
	External Clock	XIN 引脚连接外部时钟（1M~32M） XOUT 引脚为双向 GPIO 模式
Noise Filter	Disable	禁止杂讯滤波功能
	Enable	使能杂讯滤波功能
LVD	LVD_L	如果 VDD 低于 1.8V，LVD 复位
External Reset	Reset with De-bounce	使能外部复位引脚，带 De-bounce
	Reset without De-bounce	使能外部复位引脚，不带 De-bounce
	GPIO with P02	使能内部复位功能，P02 为普通双向 IO
Watchdog Reset	Always	始终开启看门狗定时器，即使在 STOP 模式和 ILDE 模式下也处于开始状态
	Enable	开启看门狗定时器，但在 STOP 和 ILDE 模式时关闭
	Disable	关闭看门狗定时器
Watchdog Overflow Period	64ms	看门狗定时器时钟源 FILRC /4
	128ms	看门狗定时器时钟源 FILRC /8
	256ms	看门狗定时器时钟源 FILRC /16
	512ms	看门狗定时器时钟源 FILRC /32

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。

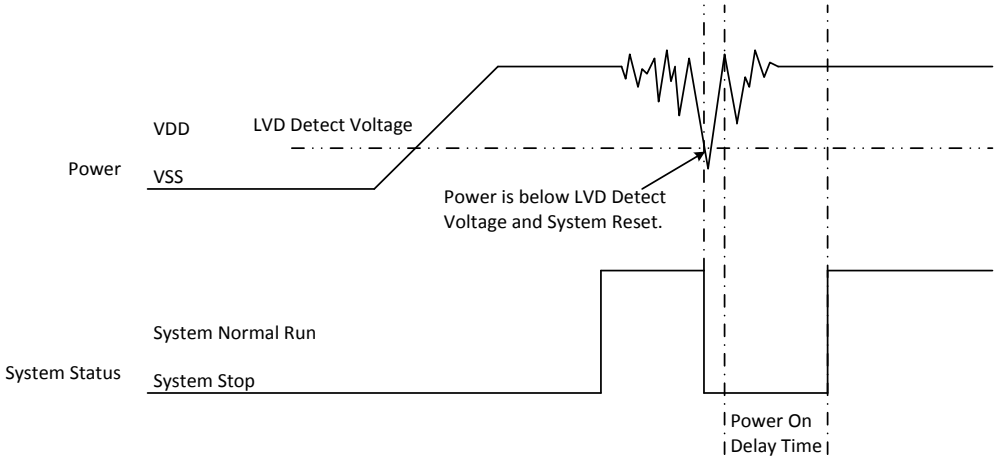


一般情况下，电源稳定期花费 4.6ms，之后单片机会自动获取 CPU 时钟源的配置。所选择的时钟源驱动起来之后，系统会计数 2048 个时钟周期以及 5 个内部低速振荡时钟以确保时钟已稳定。

*** 注：在强干扰环境下，用户可以接一个 10 欧姆电阻在 0.1uF 电容和 VDD 引脚之前，这样形成 RC 滤波电路，可以有效的抑制高压脉冲干扰，避免 IC 受损。**

7.3 LVD复位

低电压检测监控 VDD 引脚的电压，仅有 1 个电压点：1.8V。根据 LVD 的配置，比较结果可作为系统的复位信号。下表显示了低电压配置，LVD_L，以及 VDD 引脚的状态。



Condition	LVD_L
VDD ≤ 1.8 V	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。因此通过写入 5AH 到 WDTR 是一个很好的在程序中放置测试点的方式。

```
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

看门狗的工作模式如下所示：

Always mode：其内部定时器在 CPU 的所有工作模式（MORMAL，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。

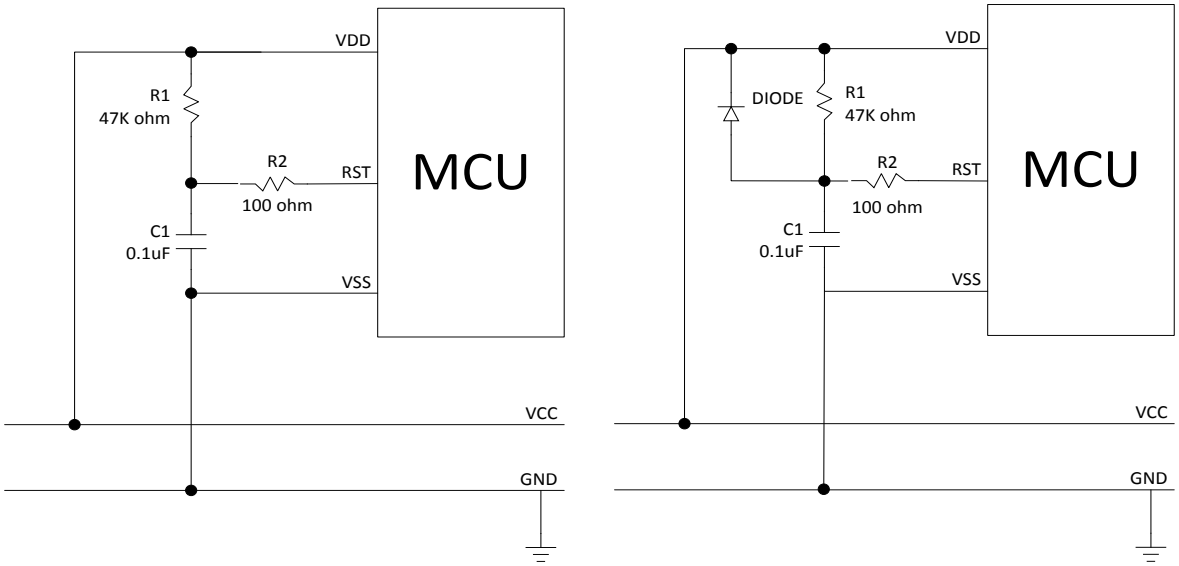
Disable mode：其内部定时器在 CPU 的所有工作模式下都不计数，在这种模式下不会触发看门狗复位。

当看门狗为 Always mode 时,系统将产生额外的功耗.

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。



*** 注：**

1 左边的复位电路接法，针对异常电源跌落或掉电复位的情况没有任何保护。

2 右图 R2 100ohm 电阻的”简单复位电路”和”二极管&RC 复位电路”是必要的，它可限制电流从外部电容 C 灌入复位引脚,导致 ESD 与 EOS 损坏 MCU 引脚。

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	-	若单片机已经由 LVD 触发复位，则该位自动置 1
6	WDT	R	-	若单片机已经由看门狗触发复位，则该位自动置 1
5	RST	R	-	若单片机已经由外部复位引脚触发复位，则该位自动置 1
4..3	Reserved	R	0	
0...2	Reserved	R	0	

SRST 寄存器

Bit	Field	Type	Initial	说明
7..1	Reserved	R	0	
0	SRSTREQ	R/W	-	Read：当单片机由软件复位,则该位自动置 1. Writer：连续设置该位 2 次触发软件复位.

WDTR 寄存器

Bit	Field	Type	Initial	说明
7..0	WDTR[7:0]	W	-	操作 WDTR 寄存器清看么狗，在寄存器中写 0x5A 将复位看门狗计时器

8 系统时钟和电源管理

为了省电，单片机内置 3 个不同的操作模式：NORMAL 模式，IDLE 模式和 STOP 模式。

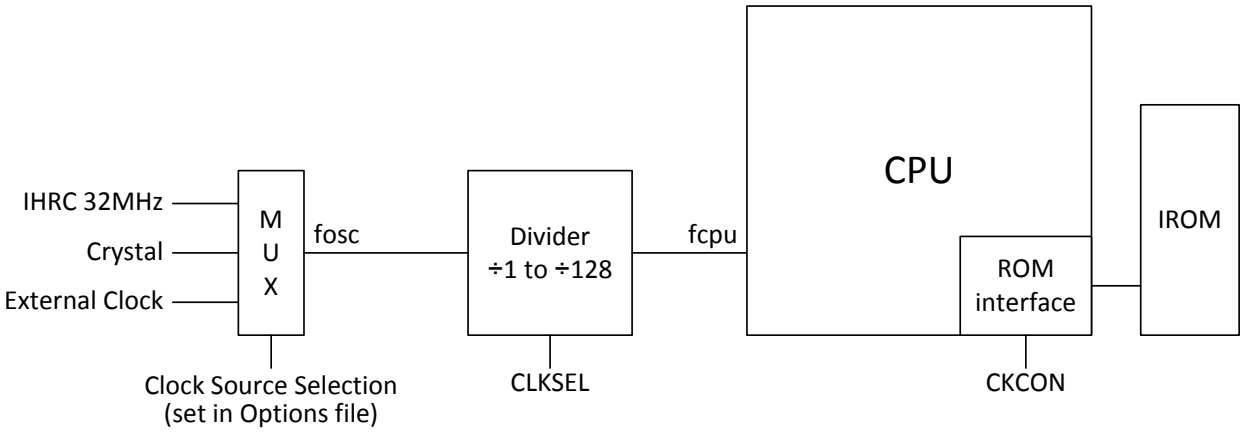
NORMAL 模式是指 CPU 和外设功能都正常工作，系统时钟取决于所选择的时钟源、时钟分频、以及程序存储器等待状态。IDLE 模式是指 CPU 时钟暂停的状态，但保留了外设功能（如定时器，PWM，SPI，UART 和 I2C）。与之相反的是，STOP 模式则禁止所有功能和时钟发生器，直至唤醒信号将系统唤醒进入 NORMAL 模式。

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。在复位或上电过程中，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 0x69 到 CLKCMD 寄存器时设置 CLKSEL。

```
1    CLKSEL = 0x05;    // Set fcpu = fosc / 4
2    CLKCMD = 0x69;    // Apply CLKSEL's setting
3    CKCON = 0x00;     // IROM fetch = fcpu / 1
```



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。

IROM fetching cycle (Instruction cycle) ≦8MHz

✷ **注：对于用户在 C 语言或汇编语言中开发的程序，程序的第一行“必须设置”CKCON=0x70,然后设置 CLKSEL= 0x07~0x00, CLKMD= 0x69,CKCON=0x00~0x70，这个优先级不能被修改。**

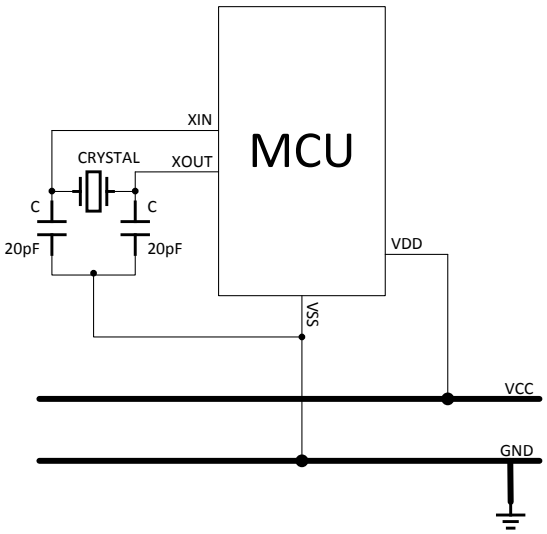
系统时钟速率和程序内存扩展周期限制见下表。

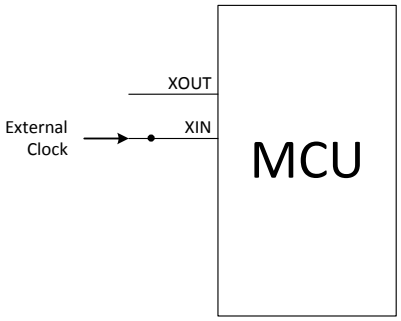
代码选项 CPU 时钟源	Fcpu = CLKSEL[2:0]	IROM Fetch = CKCON[6:4]
IHRC 32M IHRC 32M with RTC External Clock (16-32MHz)	**仅支持** 000 = Fosc/128 001 = Fosc/64 010 = Fosc/32 011 = Fosc/16 100 = Fosc/8 101 = Fosc/4	**仅支持** **000 = Fcpu/1 =>建议！** 001 = Fcpu/2 010 = Fcpu/3 011 = Fcpu/4 100 = Fcpu/5 101 = Fcpu/6 110 = Fcpu/7 111 = Fcpu/8
X’tal 12M(Crystal 8-16MHz) External Clock (8-16MHz)	**仅支持** 000 = Fosc/128 001 = Fosc/64 010 = Fosc/32 011 = Fosc/16 100 = Fosc/8 101 = Fosc/4 110 = Fosc/2	
X’tal 12M(Crystal 4-8MHz) X’tal 4M(Crystal 1-4MHz) External Clock (1-8MHz)	**仅支持** 000 = Fosc/128 001 = Fosc/64 010 = Fosc/32 011 = Fosc/16 100 = Fosc/8 101 = Fosc/4 110 = Fosc/2 111 = Fosc/1	

8.2 高速时钟和实时时钟

高速时钟分为内部和外部两种类型。外部高速时钟包括 4MHz、12 MHz crystal/ceramic 和外部时钟输入模式。内部高速振荡器是 32MHz RC 类型。这些高速振荡器由 SN8F5703_OPTIONS.A51 选择。

- IHRC 32M: 系统高速时钟源是内部高速 32MHz RC 型振荡器。在此模式下，XIN 和 XOUT 引脚是双向 GPIO 模式，且不与任何外部振荡器设备连接。
- IHRC 32M，带 RTC: 系统高速时钟源是内部高速 32MHz RC 型振荡器。RTC 时钟源是外部低速 32768Hz 晶振。在此模式下，XIN 和 XOUT 引脚限定为驱动外部 32768Hz 晶振，且禁用 GPIO 功能。
- X’tal 12M:系统高速时钟源是外部高速 crystal/ceramic。振荡器带宽是 4MHz~16 MHz，用 20 pf 电容器连接到 XIN/ XOUT 引脚接地。
- X’tal 4M:系统高速时钟源是外部高速 crystal/resonator。振荡器带宽是 1MHz~4MHz，用 20 pf 电容器连接到 XIN/ XOUT 引脚接地。
- 外部时钟：系统高速时钟源为外部时钟输入模式。输入信号只连接到 XIN 引脚，而 XOUT 引脚是双向 GPIO 模式。





SN8F5703 支持外部低速时钟（F_{RTC}）用于 T0 实时时钟。在 IHRC 32M 带 RTC 模式下，XIN/XOUT 引脚切为晶振模式驱动外部 32.768KHz 晶振。这个晶振接在 XIN/XOUT 引脚间，且连接 20pF 电容到地。

8.3 Noise Filter

Noise Filter 功能由 Noise Filter 选项控制，是一个低通滤波器，并支持晶振模式。这个选项的目的在于滤除外部高速震荡源上的高频噪声干扰。高干扰环境下，强烈建议使能 Noise Filter 以减少噪音干扰。

8.4 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。总之，CPU 和所有外设在此种情况下以 NORMAL 模式开始工作。

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 位已由程序设置为 1，只有 CPU 时钟源被关闭。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/晶振驱动器）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。

- IDLE 模式下，所有功能都能工作，仅 CPU 暂停工作。
- IDLE 模式的唤醒源为 P0/P1 电平变换触发和中断触发唤醒。

反之，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。

- CPU、外设功能和时钟发生器都暂停工作。
- STOP 模式的唤醒源为 P0/P1 电平变换触发唤醒。

用户在 C 编译中进行程序开发时，强烈建议使用 IDLE 和 STOP 宏指令来控制单片机的系统模式，不要直接设置 IDLE 和 STOP 位。

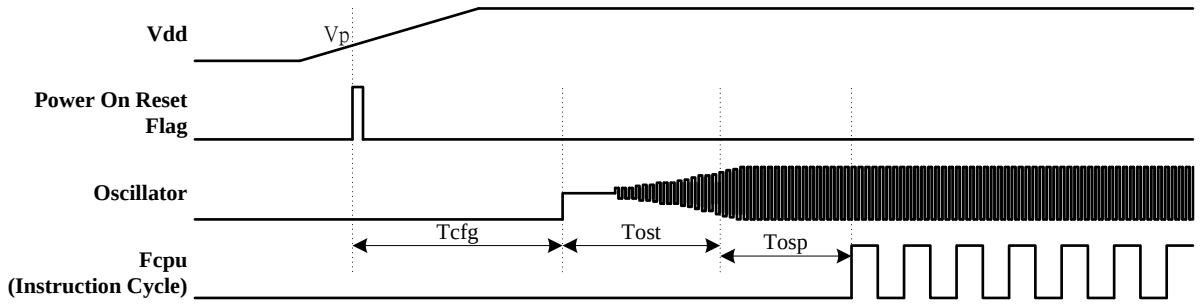
```
1    IDLE();
2    STOP();
```

✷ **注：通过“汇编语言”进入 IDLE 模式或 STOP 模式必须使用 MOV 指令。**

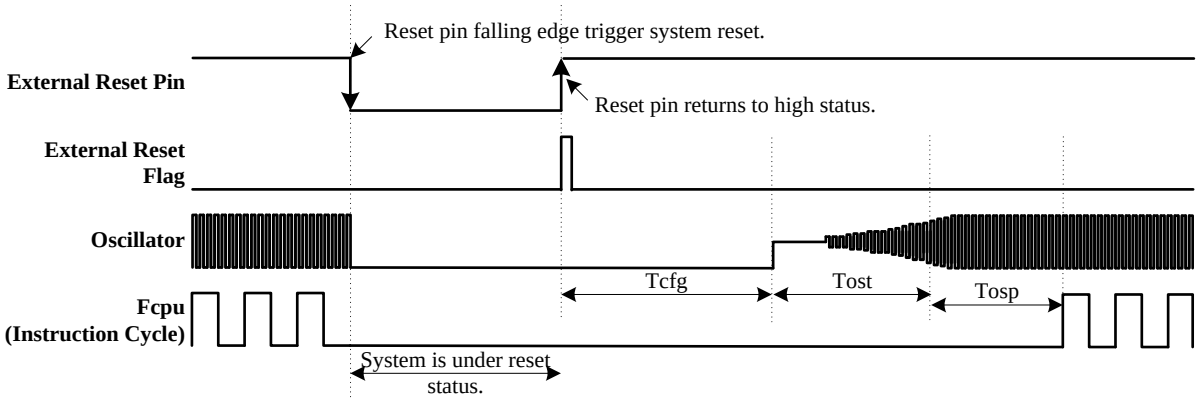
8.5 系统时间计时

参数	符号	说明	典型值
硬件配置时间	Tcfg	8 * FILRC +2^{17} * FIHRC	4.6ms F_{ILRC} = 16kHz & F_{IHRC} = 32MHz
振荡器启动时间	Tost	启动时间取决于振荡器的材料、制造厂和架构。通常情况下，低速振荡器的启动时间低于高速振荡器的。RC 型振荡器的启动时间快于晶体型振荡器。	-
振荡器预热时间	Tosp	复位条件下的振荡器预热时间。 2048*Fhosc+ 5*F_{ILRC} （上电复位，LVD 复位，看门狗复位，外部复位引脚激活）	825us @ Fhosc = 4MHz 441us @ Fhosc = 16MHz 377us @ Fhosc = 32MHz
		掉电模式唤醒条件下的振荡器预热时间。 2048*Fhosc+5*FILRC ……晶体/共鸣器型振荡器，例如 32768Hz crystal, 4MHz crystal, 16MHz crystal… 64*Fhosc+5*FILRC……RC 型振荡器，例如内部高速 RC 型振荡器。	X’ tal: 825us @ Fhosc = 4MHz 441us @ Fhosc = 16MHz RC: 315us @ Fhosc = 32MHz

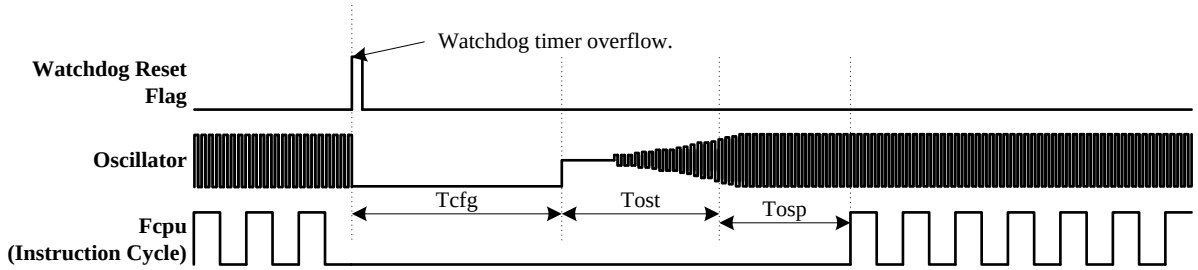
- 上电复位计时



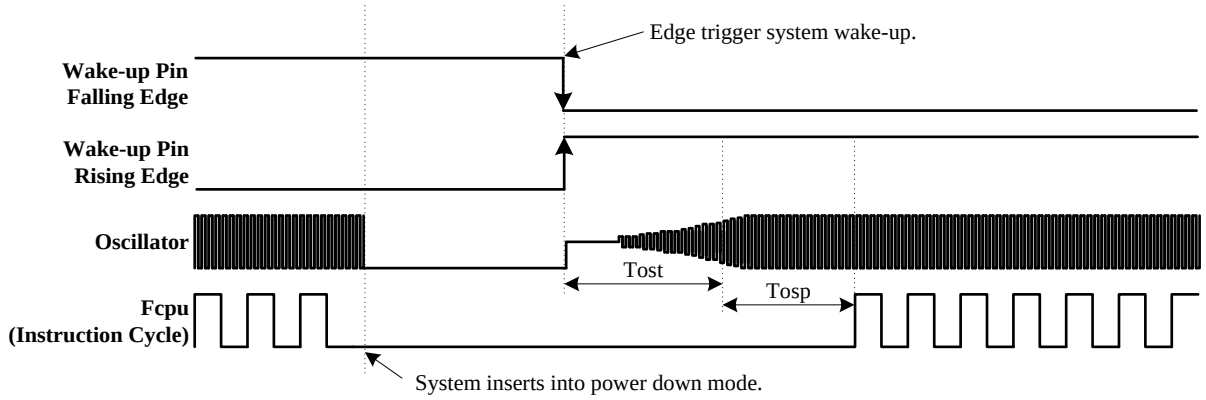
- 外部复位引脚复位计时



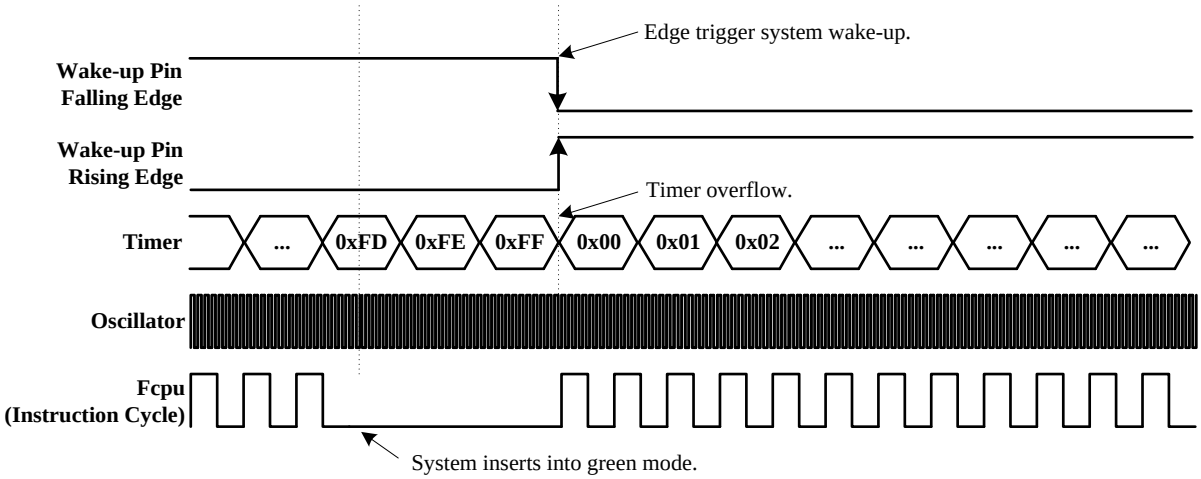
- 看门狗复位计时



- STOP 模式唤醒计时

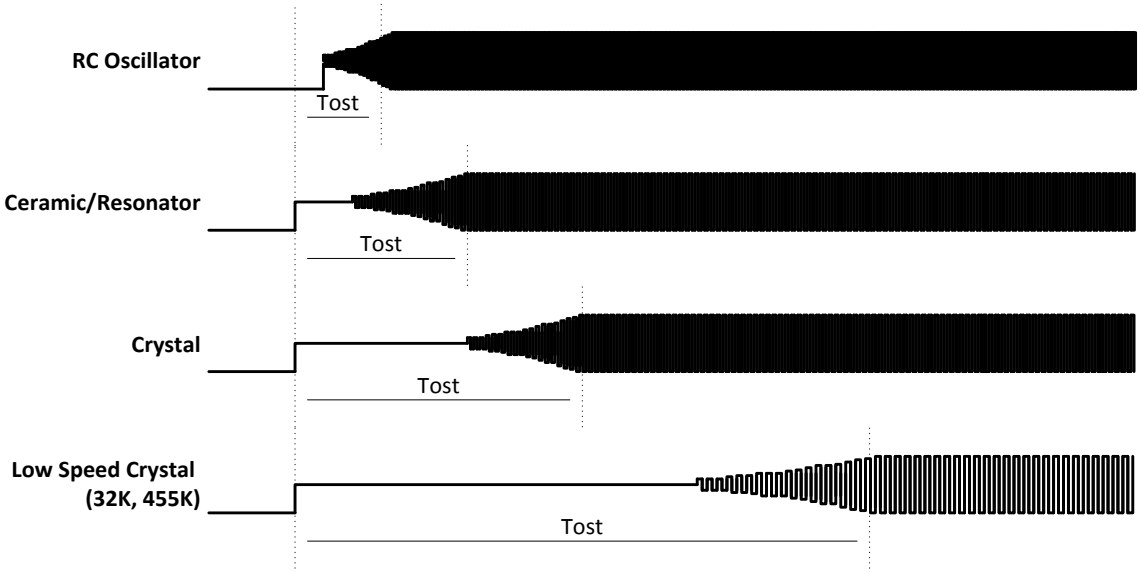


- IDLE 模式唤醒计时



- 振荡器启动时间

启动时间取决于振荡器的材料、制造厂和架构。通常情况下，低速振荡器的启动时间低于高速振荡器的。



8.6 系统时钟和电源管理寄存器

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

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。

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 0x69 来应用 CLKSEL 的设置。

PCON 寄存器（0x87）

Bit	Field	Type	Initial	说明
7				参考其它章节。
6..4	Reserved	R	0x00	
3	P2SEL	R/W	1	高阶地址字节配置位。在 MOVX @Ri 操作期间选择地址的高字节（“XRAM[15：8]”） 0:“XRAM[15：8]”=“P2REG”。“P2REG”是端口 2 输出寄存器的内容。 1:“XRAM[15：8]”= 0x00。
2	GF0	R/W	0	通用标志位
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。

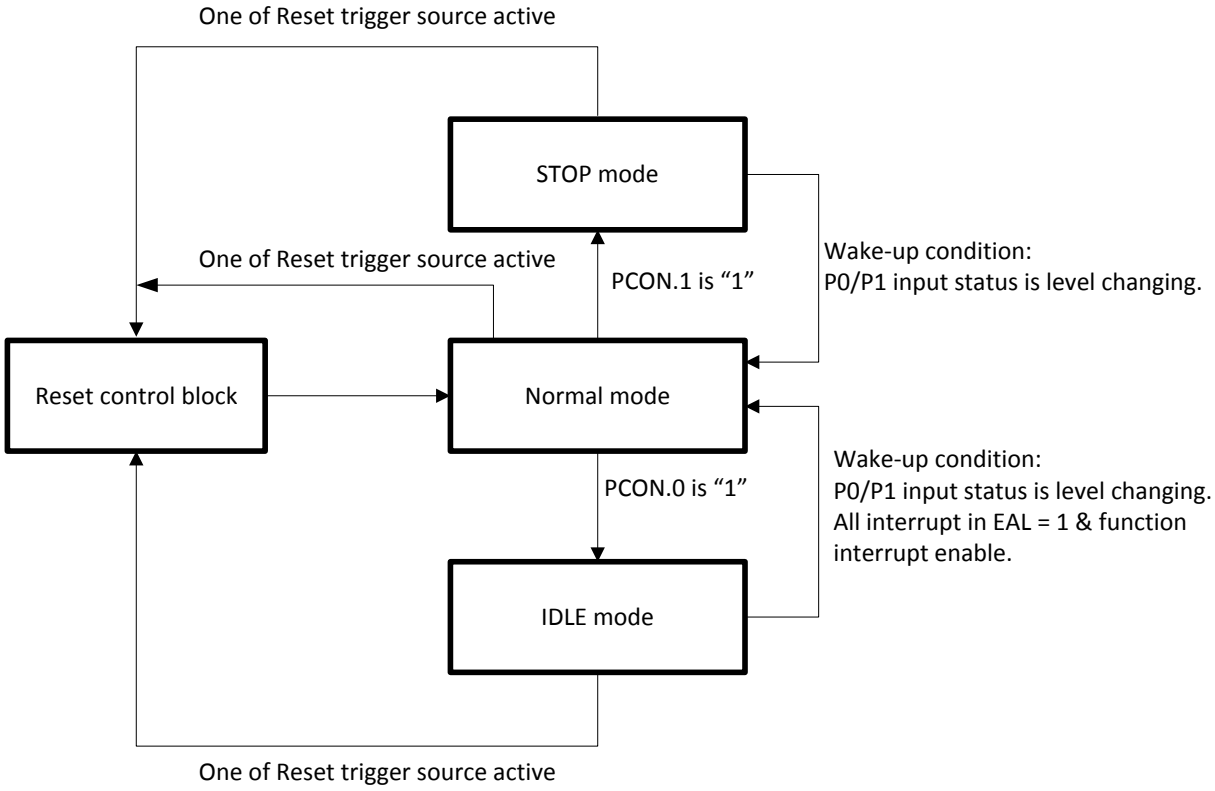
P1W 寄存器（0x91）

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 系统操作模式

芯片在不同的时钟频率和不同的省电原因下有三种操作模式。这些模式控制振荡器、op-code 操作和模拟外围设备的操作。

- 普通模式：系统高速工作模式
- IDLE 模式：系统空闲模式（绿色模式）
- STOP 模式：系统省电模式（睡眠模式）



工作模式时钟控制见下表：

工作模式	普通模式	IDLE 模式	STOP 模式
IHRC	IHRC: 运行 Ext. OSC: 禁止	IHRC: 运行 Ext. OSC: 禁止	Stop
ILRC	运行	运行	Watchdog always: 运行 其它：禁止
Ext. OSC	IHRC: 禁止 Ext. OSC：运行	IHRC: 禁止 Ext. OSC：运行	Stop
CPU 指令	执行	Stop	Stop
定时器 0 （定时器，事件计数器）	由 TR0 激活	由 TR0 激活	未激活
定时器 1 （定时器，事件计数器）	由 TR1 激活	由 TR1 激活	未激活
定时器 2 （定时器，事件计数器）	设置为使能时有效	设置为使能时有效	未激活
PWM	设置为使能时有效	设置为使能时有效	未激活
UART	设置为使能时有效	设置为使能时有效	未激活
SPI	设置为使能时有效	设置为使能时有效	未激活
I2C	设置为使能时有效	设置为使能时有效	未激活
ADC	设置为使能时有效	设置为使能时有效	未激活
看门狗定时器	通过看门狗 代码选项	通过看门狗 代码选项	通过看门狗 代码选项
内部中断	全部激活	全部激活	全部未激活
外部中断	全部激活	全部激活	全部未激活
唤醒源	-	P0, P1, 复位, EAL 中的所有中断 = 1 & 功能中断使能	P0, P1, 复位

- Ext. OSC：外部高速振荡器 (XIN/XOUT).
- IHRC：内部高速振荡器 RC 型
- ILRC：内部低速振荡器 RC 型

9.1 普通模式

普通模式是系统高速时钟工作模式。系统时钟源来自高速振荡器。执行程序。上电以及复位触发释放后，系统进入到普通模式执行程序。当从停止/空闲模式下唤醒系统，系统也将进入普通模式。在普通模式下，高速振荡器激活，功耗是所有工作模式中最大的。

- 执行程序，且所有功能是可控的。
- 系统高速运行。
- 高速振荡器和内部低速 RC 振荡器激活。
- 普通模式可以通过 PCON 寄存器切换到其他操作模式。
- STOP/IDLE 模式可唤醒为普通模式。

9.2 STOP模式

STOP 模式系统的理想状态。没有程序执行，也没有振荡器工作。只有内部稳压器激活，保持所有控制栅状态、寄存器状态和 SRAM 内容。STOP 模式由 P0/P1 硬件电平改变触发。P0 唤醒功能通常是使能的。STOP 模式唤醒为普通模式。进入 STOP 模式由 PCON 寄存器的停止位控制。当 stop=1 时，系统进入 STOP 模式。从 STOP 模式系统唤醒后，自动禁用停止位 (零状态)。

- 程序停止执行，且禁用所有功能。
- 振荡器包括外部高速振荡器，内部高速振荡器和内部低速振荡器停止。
- 内部稳压器激活，保持所有控制栅状态、寄存器状态和 SRAM 内容
- 系统从 STOP 模式唤醒后进入到普通模式。
- STOP 模式唤醒源是 P0/P1 硬件电平改变触发。

9.3 IDLE模式

IDLE 模式是另一个系统的理想状态，但与 STOP 模式不同。在 STOP 模式下，所有功能和硬件设备都被禁用。但在 IDLE 模式下，系统时钟源持续运行，所以 IDLE 模式的功耗大于 STOP 模式的功耗。在 IDLE 模式下，程序不执行 ,但定时器唤醒功能是激活启用的 ,定时器时钟源是不间断的系统时钟。IDLE 模式有 2 个唤醒源。一个是 P0/P1 电平改变触发唤醒。另一个是 EAL=1 中断&功能中断使能。这意味着用户可以用中断使能设置功能，而系统直到中断发出才被唤醒。进入 IDLE 模式由 PCON 寄存器的 idle 位控制。idle=1 时，系统进入 IDLE 模式。系统从 IDLE 模式唤醒后，idle 位自动禁用（零状态）。

- 停止执行程序，且禁用全部功能。
- 定时器溢出中断可以唤醒系统。
- 作为系统时钟源的振荡器保持运行，而其他振荡器是否工作取决于系统操作模式配置。
- 如果从普通模式进入 IDLE 模式，系统在唤醒后进入普通模式。
- IDLE 模式的唤醒源是 P0/P1 电平改变触发。
- 时钟源来自于系统时钟的硬件模块 ,如果有被使能 ,在 ILDE 模式下都处于工作状态 ,例如 :定时器、PWM、事件计数器…。
- 在有打开总中断和使能中断功能时，任何中断事件都会唤醒 IDLE 模式。

9.4 唤醒

STOP 模式（睡眠模式）或 IDLE 模式下，不执行程序。唤醒触发可以唤醒系统恢复为普通模式。唤醒触发来源是外部触发(P0/P1 电平改变)和内部触发（EAL= 1 &功能使能）。唤醒功能内置有中断操作，发出请求标志位，触发系统执行中断服务程序，将系统唤醒。

当系统在 STOP 模式下，高时钟振荡器停止。当从 STOP 模式唤醒时，MCU 等待 2048 个外部高速振荡时钟+ 5 个内部低速振荡时钟，以及 64 个内部高速振荡器时钟+ 5 个内部低速振荡时钟的唤醒时间，以稳定振荡电路。唤醒时间后，系统进入普通模式。

外部高速时钟振荡器唤醒时间值如下。

唤醒时间 = 1/Fosc * 2048 (sec) + 1/Flosc * 5 + 高速时钟启动时间

例：在 STOP 模式（睡眠模式）下，唤醒系统。唤醒时间后，系统进入普通模式。唤醒时间如下。

唤醒时间= 1/Fosc * 2048 + 1/Flosc * 5 = 0.825 ms (Fosc = 4MHz)

总唤醒时间 = 0.825 ms + 振荡器启动时间

RC 型内部高速时钟振荡器唤醒时间值如下。

唤醒时间= 1/Fosc * 64 (sec) + 1/Flosc * 5 +高速时钟启动时间

例：在 STOP 模式（睡眠模式）下，唤醒系统。唤醒时间后，系统进入普通模式。唤醒时间如下。

唤醒时间= 1/Fosc * 64 +1/Flosc * 5= 315 us (Fhosc = 32MHz)

> ✷ **注：高速时钟启动时间取决于高速时钟的 VDD 和振荡器类型。**

在 STOP 模式和绿色模式下，具有唤醒功能的 I/O 端口能够唤醒系统进入普通模式。唤醒触发边缘是在上升或下降沿边缘的电平改变。端口 0 和端口 1 具有唤醒功能。端口 0 唤醒功能通常是启用的，但端口 1 由 P1W 寄存器控制。

P1W 寄存器(0x91)

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

10 中断

该单片机包含 15 个中断源（2 个外部中断和 13 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 / MODF	软件清 0	6 / 0x004B
INT1	EX1	IE1	自动清 0	7 / 0x0013
Comparator	ECMP	CM0F	软件清 0	8 / 0x0093
T2COM0	ET2C0	TF2C0	自动清 0	9 / 0x0053
Timer 1	ET1	TF1	自动清 0	10 / 0x001B
T2COM1	ET2C1	TF2C1	自动清 0	11 / 0x005B
UART	ES0	TI0 / RI0	软件清 0	12 / 0x0023
T2COM2	ET2C1	TF2C2	自动清 0	13 / 0x0063
Timer 2	ET2 / ET2RL	TF2 / TF2RL	软件清 0	14 / 0x002B
T2COM3	ET2C3	TF2C3	自动清 0	15 / 0x006B

* 注：当中断请求标志可以通过硬件自动清除时，不要在通过软件清除。

10.1 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（0x03 – 0x93），系统执行相对应的中断服务程序 ISR。

10.2 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。

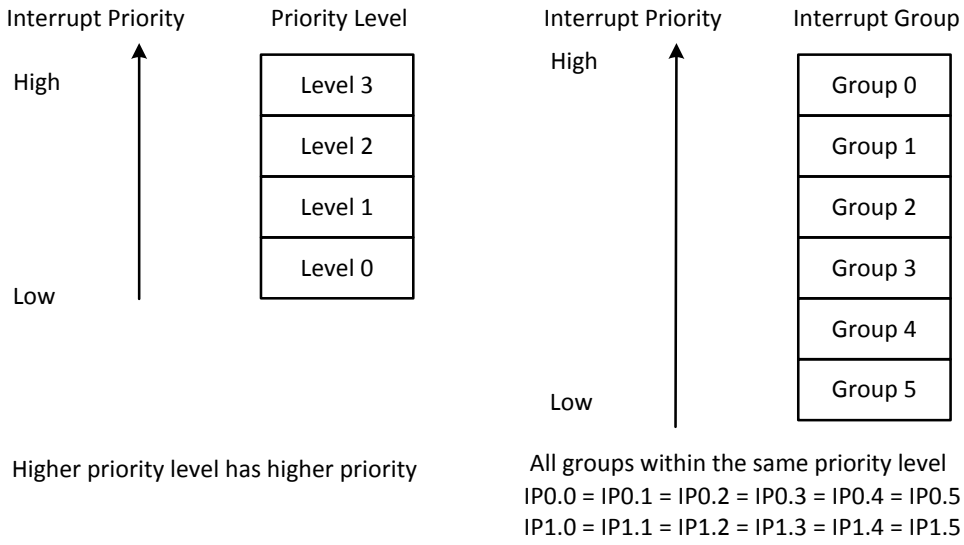
优先级别	IP1.x	IP0.x
Level 0	0	0
Level 1	0	1
Level 2	1	0
Level 3	1	1

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	INT1	Comparator	T2 COM0	
Group 3	T1		T2 COM1	
Group 4	UART		T2 COM2	
Group 5	T2		T2 COM3	

当多个中断请求发生时，必须先执行最高优先级的请求。根据自然优先权和优先级，选择最高优先级的请求。步骤如下:

1．选择所有组之间优先级最高的组。
2．在最高优先级组之间选择自然优先权最高的组。
3．选择最高优先级组内的有最高自然优先权的 ISR。

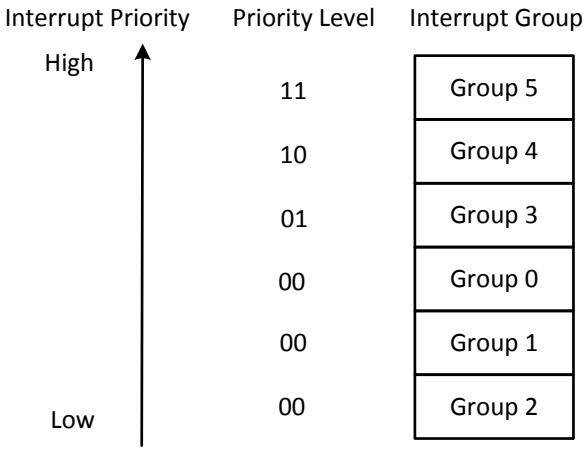


Higher priority level has higher priority

All groups within the same priority level
IP0.0 = IP0.1 = IP0.2 = IP0.3 = IP0.4 = IP0.5
IP1.0 = IP1.1 = IP1.2 = IP1.3 = IP1.4 = IP1.5

例，group5 具有最高优先级，而 group0 ~ group2 具有最低优先级。这意味着 group5 中的中断向量具有最高的中断优先权，group4 具有第二中断优先权，而 group3 具有第三中断优先权。Group0 ~ group2 具有相同的优先级，因此遵循自然优先规则。故而中断优先权 :group5 > group4 > group3 > group0 > group1 > group2。

```
MOV     IP0, #00101000B     ; 设置 group0 - group5 的不同优先级。
MOV     IP0, #00110000B
```



Priority Level: GP5>GP4>GP3=GP2=GP1=GP0

Interrupt Priority: GP5>GP4>GP3>GP0>GP1>GP2

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	IP0[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	

10.3 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	EX1	ET0	EX0
IEN1	ET2RL	-	ET2C3	ET2C2	ET2C1	ET2C0	ESPI	EI2C
IEN2	-	-	-	-	-	ECMP	EADC	-
IEN4	EPWM1	-	-	-	PWM1F	-	-	-
IRCON	TF2RL	TF2	TF2C3	TF2C2	TF2C1	TF2C0	-	-
IRCON2	-	-	-	-	-	-	CMPF	ADCF
TCON	TF1	TR1	TF0	TR0	IE1	-	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。
2	EX1	R/W	0	外部中断 P0.2（INT1）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 P2.0（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
2	ECMP	R/W	0	比较器 CMP 中断控制位。 0：禁止； 1：使能。
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
1	CMPF	R/W	0	比较器 CMP 中断请求标志位。 0：无 CMP 中断请求； 1：CMP 请求中断。
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。
3	IE1	R/W	0	外部中断 P0.2（INT1）中断请求标志位。 0：无 INT1 中断请求； 1：INT1 请求中断。
1	IE0	R/W	0	外部中断 P2.0（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，显示 UART 串行接收的完成状态。在模式 0 的第 8 位结束时，或者其它模式的停止位中间时，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。

10.4 10.4 示例代码

例：定义中断向量。中断服务程序在用户程序之后。

```
                ORG     0               ;  0000H
                JMP     START           ; 跳转至用户程序地址。
                ...
                ORG     0X000B          ; 跳转至中断服务程序地址。
                JMP     ISR_T0
                ORG     0X0013
                JMP     ISR_INT1
                ...
                ORG     0X008B
                JMP     ISR_ADC
                ...
                ORG     0X00EC
START:                                  ; 00ECH，用户程序头。
                ...                     ; 用户程序。
                ...
                ...
                JMP     START           ; 用户程序结束。
                ...
ISR_ T0:                                ; 中断服务程序头。
                PUSH    ACC             ; ACC 存入堆栈缓存。
                PUSH    PSW             ; PSW 存入堆栈缓存。
                ...
                POP     PSW             ; 从堆栈缓存加载 PSW。
                POP     ACC             ; 从堆栈缓存加载 ACC。
                RETI                    ; 中断服务程序结束。
ISR_ADC:                                ;
                PUSH    ACC             ; ACC 存入堆栈缓存。
                PUSH    PSW             ; PSW 存入堆栈缓存。
                ...
                POP     PSW             ; 从堆栈缓存加载 PSW。
                POP     ACC             ; 从堆栈缓存加载 ACC。
                RETI                    ; 中断服务程序结束。
                ...
ISR_INT1                                ;
```

```
PUSH    ACC         ; ACC 存入堆栈缓存。
PUSH    PSW         ; PSW 存入堆栈缓存。
...
POP     PSW         ; 从堆栈缓存加载 PSW。
POP     ACC         ; 从堆栈缓存加载 ACC。
RETI                ; 中断服务程序结束。


END                 ; 程序结束。
```

11 GPIO

该单片机共有 22 个 GPIO 引脚，不同于 8051 只有开漏输出，SN8F5703 还内置推挽式输出结构，以增强其驱动能力。

11.1 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	-	-	P25M	P24M	P23M	P22M	P21M	P20M
P0OC	-	-	-	P06OC	P05OC	P04OC	P01OC	P00OC

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				等等

P0OC 寄存器（0Xe4）

Bit	Field	Type	Initial	说明
Else	保留	R	0	
4	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。
3	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。
2	P04OC	R/W	0	P0.4 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。
1	P01OC	R/W	0	P0.1 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	P00OC	R/W	0	P0.0 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。

11.2 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，当该引脚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	-	-	P25	P24	P23	P22	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				等等

11.3 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	-	-	P25UR	P24UR	P23UR	P22UR	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				等等

* 如果引脚为输出模式或者模拟功能，建议禁止上拉电阻。

11.4 与模拟功能共用的引脚

SN8F5703 内置模拟功能，如 ADC，OPA 和比较器。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	-
P2CON	-	-	P2CON5	P2CON4	P2CON3	P2CON2	P2CON1	P2CON0

P1CON: 0Xd6，P2CON: 0x9E

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				等等

12 外部中断

外部中断源 INT0 和 INT1 内置边沿触发功能，由 PEDGE 寄存器控制。使能外部中断（EX0/EX1）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/IE1）置 1，程序计数器跳转到中断向量（ORG 0x0003/0x0013）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。

12.1 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	-	-	-	-	EX1G1	EX1G0	EX0G1	EX0G0
IEN0	EAL	-	ET2	ES0	ET1	EX1	ET0	EX0
TCON	TF1	TR1	TF0	TR0	IE1	-	IE0	-

PEDGE 寄存器（0X8F）

Bit	Field	Type	Initial	说明
3..2	EX1G[1:0]	R/W	10	外部中断 INT1 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。
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	

IEN0 Register (0XA8)

Bit	Field	Type	Initial	说明
7	EAL	R/W	0	使能所有中断控制位。 0：禁止。 1：使能。
2	EX1	R/W	0	外部 P0.2 中断（INT1）控制位。 0：禁止。 1：使能。
0	EX0	R/W	0	外部 P2.0 中断（INT0）控制位。 0：禁止。 1：使能。
Else				参考其它章节。

TCON Register (0X88)

Bit	Field	Type	Initial	说明
3	IE1	R/W	0	外部 P0.2 中断（INT1）请求标志位。 0：禁止。 1：使能。
1	IE0	R/W	0	外部 P2.0 中断（INT0）请求标志位。 0：禁止。 1：使能。
Else				参考其它章节。

* 注：当通过手动软件清除 TF0，TF1，IE0，IE1 时，用户必须确保 TCON 寄存器中没有其它中断请求。

12.2 示例代码

下面的示例代码程序演示了如何执行 INT0/INT1 中断。

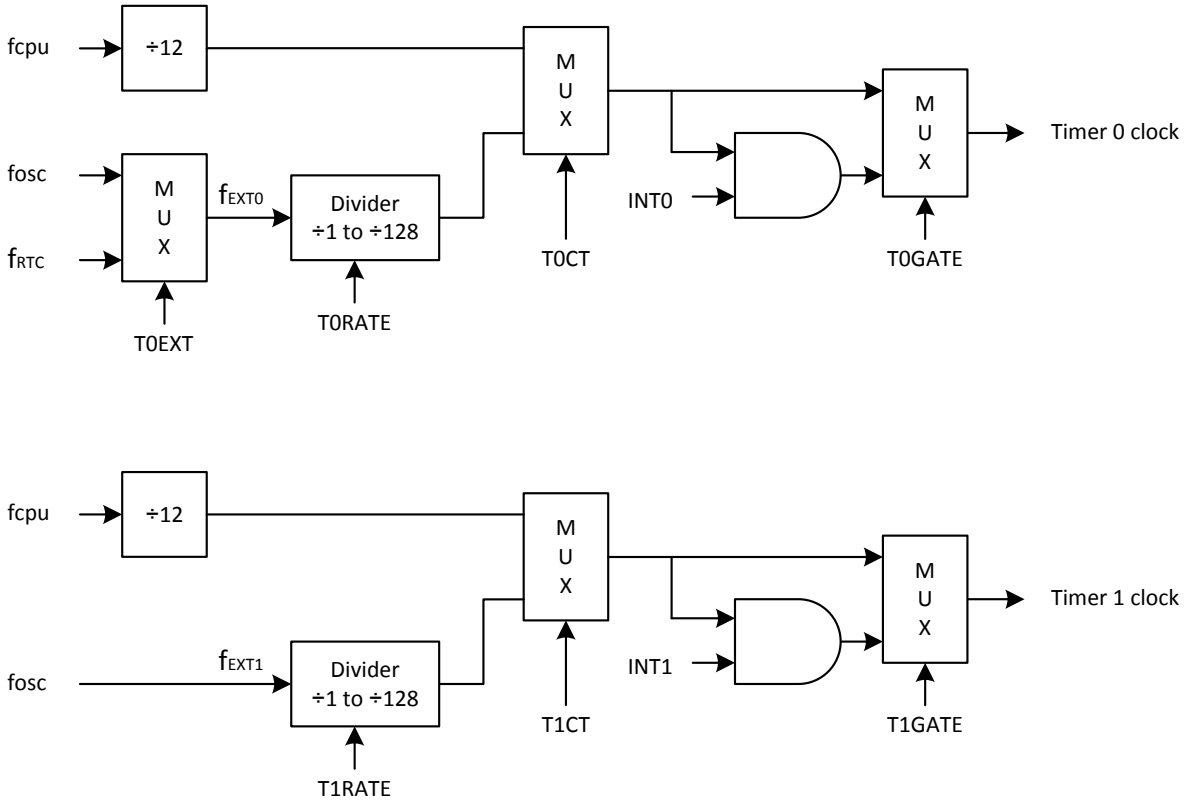
```
#define INT0Rsing           (1 << 0)    //INT0 trigger edge is rising edge
#define INT0Falling         (2 << 0)    //INT0 trigger edge is falling edge
#define INT0LeChge          (3 << 0)    //INT0 trigger edge is level change
#define EINT0               (1 << 0)    //INT0 interrupt enable

#define INT1Rsing           (1 << 2)    //INT1 trigger edge is rising edge
#define INT1Falling         (2 << 2)    //INT1 trigger edge is falling edge
#define INT1LeChge          (3 << 2)    //INT1 trigger edge is level change
#define EINT1               (1 << 2)    //INT1 interrupt enable

void EnableINT(void)
{
   //INT0 rising edge, INT1 falling edge
   PEDGE = INT0Rising | INT1Falling

   //Enable INT0/INT1 interrupt
   IEN0 | = EINT0 | EINT1;
   //Enable total interrupt
   IEN0 | = 0x80;

   P0 = 0x00;
   P0M = 0X03;
}

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

void INT1Interrupt(void) interrupt ISRInt1   //0x13
{   //IE1 clear by hardware
    P01 = ~P01;
}
```

13 定时器 T0 和 T1

T0 和 T1 是 2 个独立的二进制计数定时器。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，并指定重装值；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。

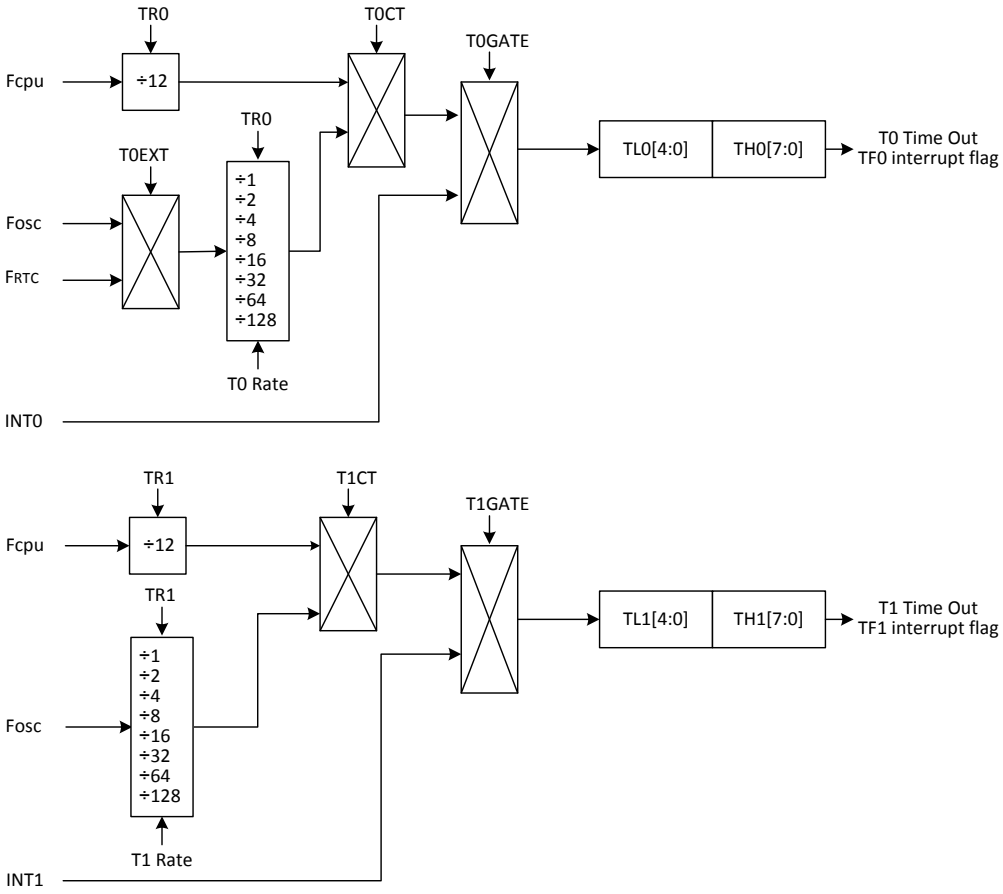
13.1 T0 和T1 时钟选择

下图阐明了 T0 和 T1 的时钟选择电路。T0 有三个时钟源选择：fcpu, fosc, and f_{RTC}。如果应用了 T0GATE,所有时钟源都可由 INT0 引脚关闭(暂停)。T1 时钟源选择:fcpu 和 fosc。如果应用了 T1GATE,所有时钟源都可由 INT1 引脚关闭(暂停)。总的来说，T0 和 T1 这两个计数器最主要的区别在于，T0 支持 f_{RTC} 时钟源(实时计数器，RTC)，这个功能需要 MCU 的时钟源选择为 IHRC 32MHz witch RTC (参考复位和上电控制章节)，同时硬件上需要外接一个 32KHz 的晶振。



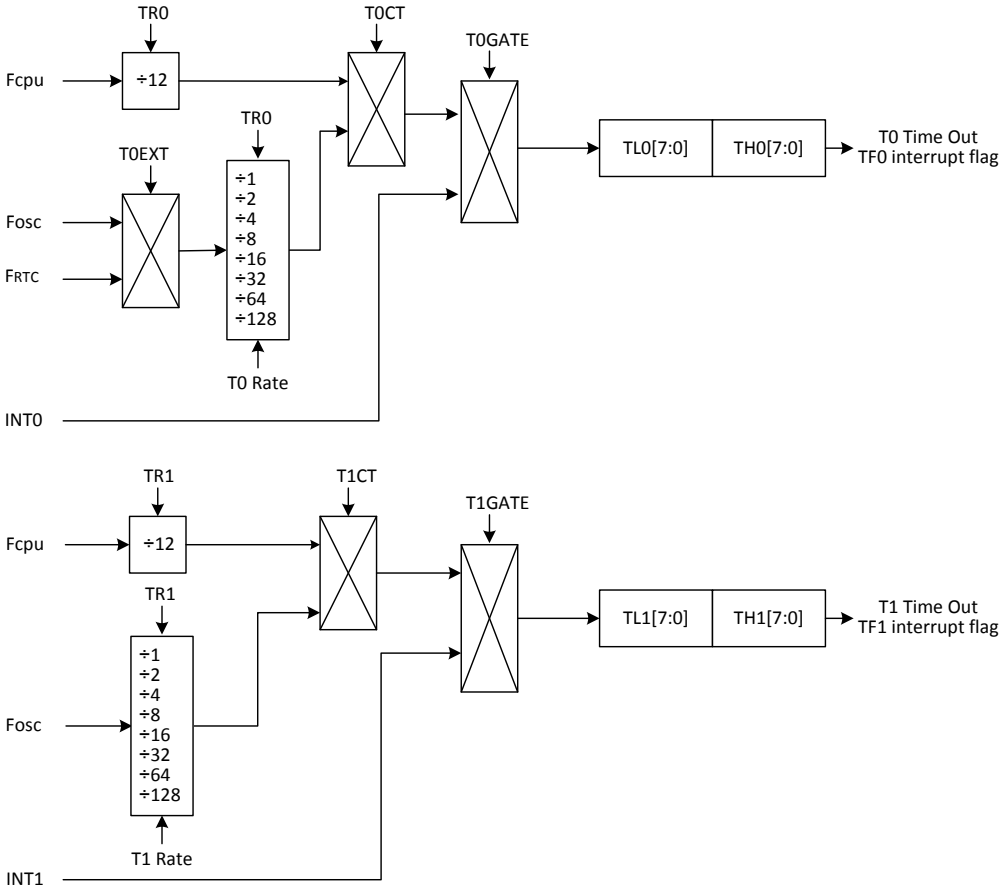
13.2 模式 0：13 位向上计数定时器

模式 0 是 13 位向上计数定时器（TL0 的高 3 位无效）。一旦定时器的计数器溢出（0xFF1F 到 0x0000），会立即设置 TF0/TF1 标志位。没有使能 ET0/ET1 时，由软件读取该标志，否则，在使能 ET0/ET1 时，由中断控制器进行控制。



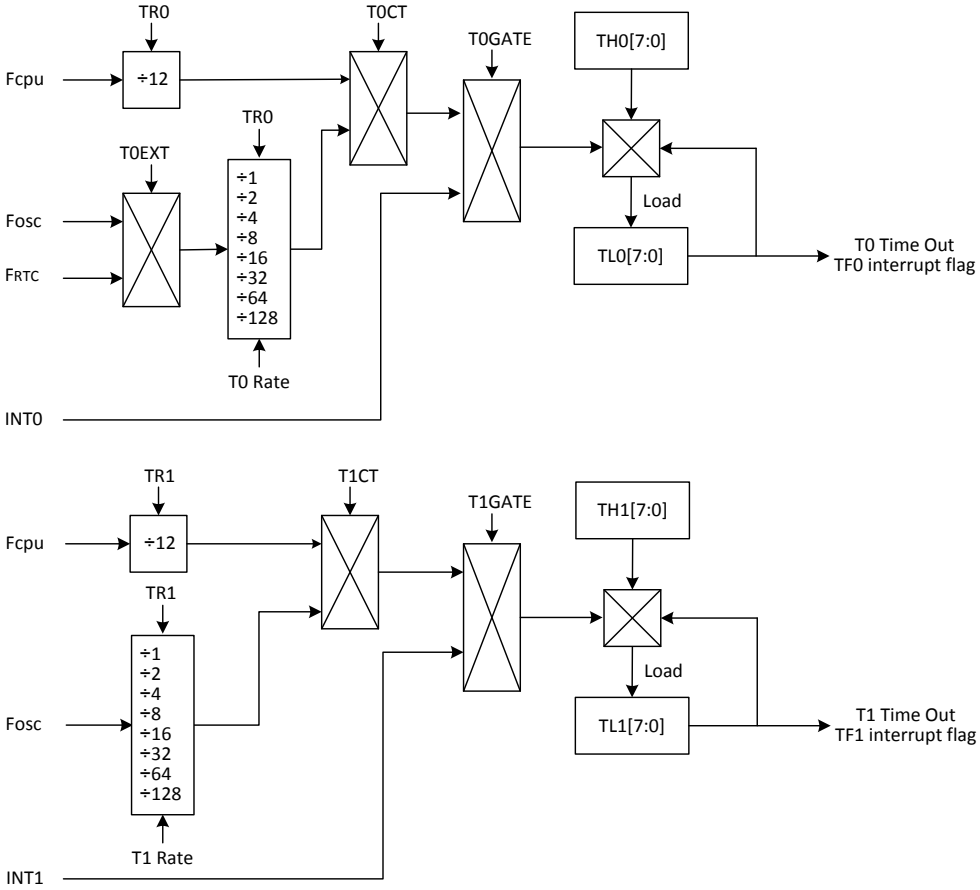
13.3 模式 1：16 位向上计数定时器

模式 1 是 16 位向上计数定时器。一旦定时器的计数器溢出（从 0xFFFF 到 0x0000），会立即置位 TF0/TF1；可由软件读取标志；使能 ET0/ET1 时，则由中断控制器进行控制。



13.4 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（TL0/TL1 计数从 0xFF 到 0x00）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 0xFF 计数到 TH0/TH1 的值。

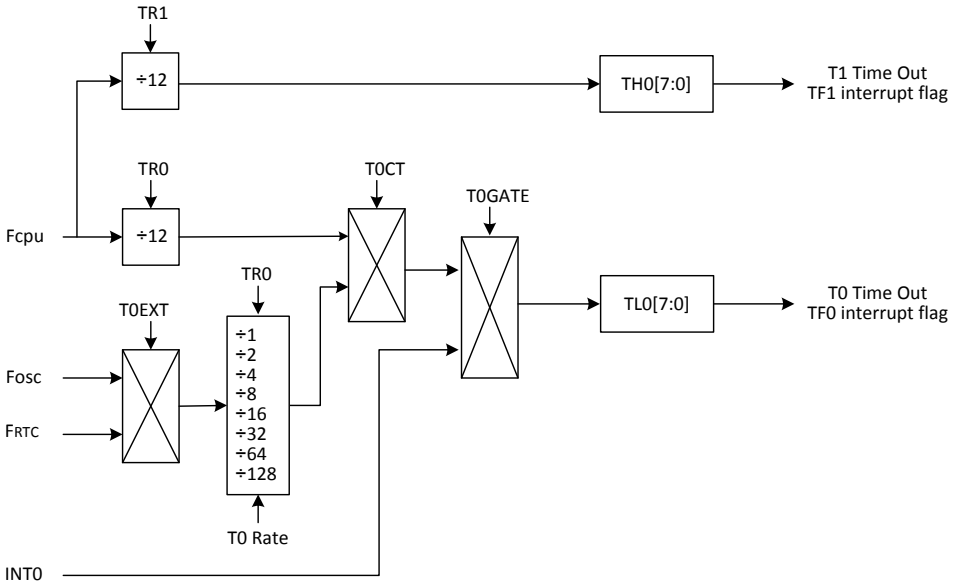


13.5 模式 3（仅T0）：独立的 2 个 8 位向上计数定时器

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。当应用了 T0GATE 时，TL0 的时钟源可以由 INT0 引脚控制开关。

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的自计数定时器。



13.6 T0 和T1 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	EX1	ET0	EX0
TCON	TF1	TR1	TF0	TR0	IE1	-	IE0	-
TCON0	T0EXT	T0RATE2	T0RATE1	T0RATE0	-	T1RATE2	T1RATE1	T1RATE0
TMOD	T1GATE	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 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。

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	IE1	R/W	0	参考 INT1。
2	Reserved	R	0	
1	IE0	R/W	0	参考 INT0。
0	Reserved	R	0	

＊ 注：在通过软件手动清除 TF0，TF1，IE0 或 IE1 中任意一个时，用户必须确认其它的中断请求标志不为 1。

TCON0 寄存器（0Xe7）

Bit	Field	Type	Initial	说明
7	T0EXT	R/W	0	T0 实时时钟计数器控制位。 0：禁止； 1：使能*。
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。

* T0EXT = 1 仅仅当选择为 ‘IHRC 32 MHz with RTC’ 才能进行设置; 当选择其他时钟源时，请设置为 0.

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。

*(1) f_{EXT1} = fosc.

*(2) f_{EXT0} = fosc or f_{RTC}.

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。

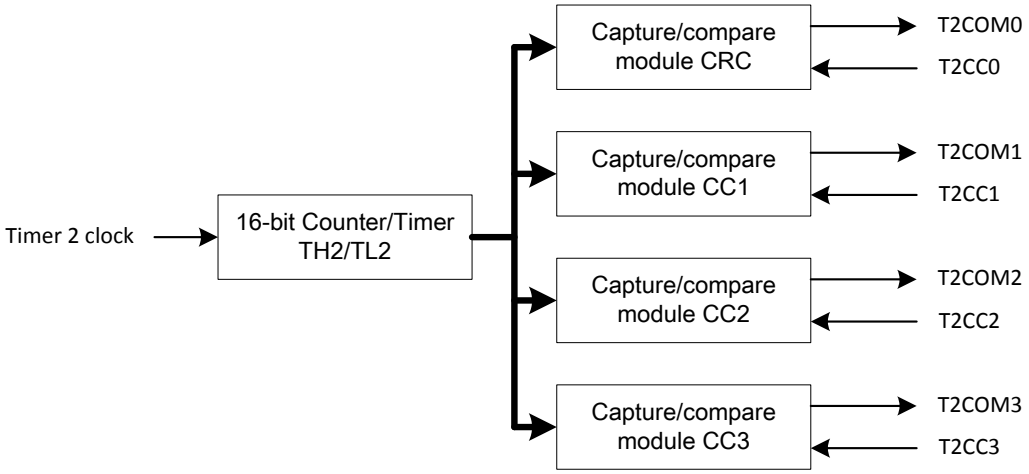
13.7 示例程序代码

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;
}
```

14 定时器 T2

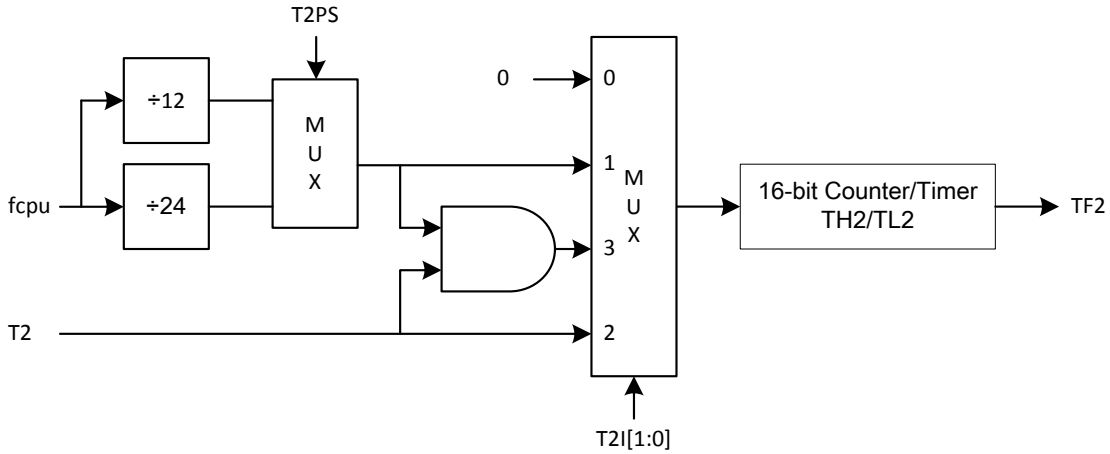
T2 是 16 位向上计数定时器，具有好几个可选的扩展功能：指定重装值，比较器输出（PWM）和捕捉功能。T2 由 1 个 16 位计数/定时器和 4 个 16 位捕捉/比较器模块组成：每个捕捉/比较器模块在使能状态时都有自己相关的 IO 引脚；且每个捕捉/比较器模块都可以作为下列模式独立工作：比较器，上升沿时捕捉，以及写入寄存器时捕捉。



14.1 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 0xFFFF 到 0x0000），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 引脚下降沿（T2 引脚与 P1.1 共有，时钟 rate≤ 0.5 * fcpu）

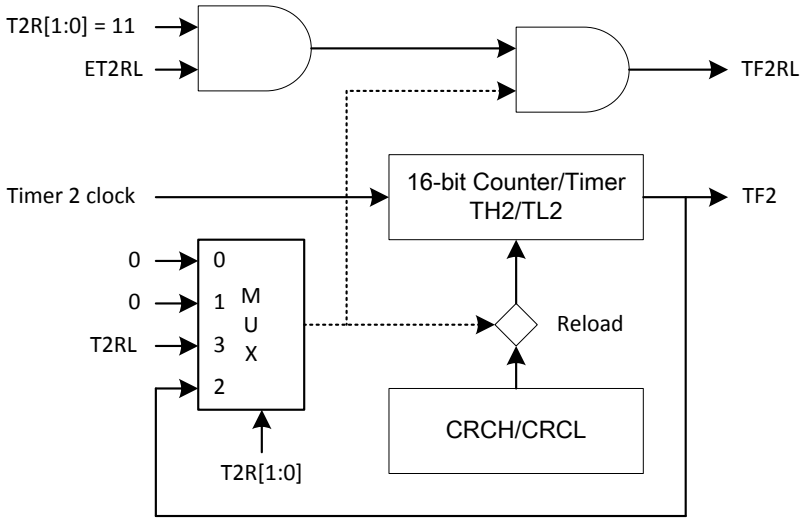


14.2 指定T2 重装值

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 0xFFFF。

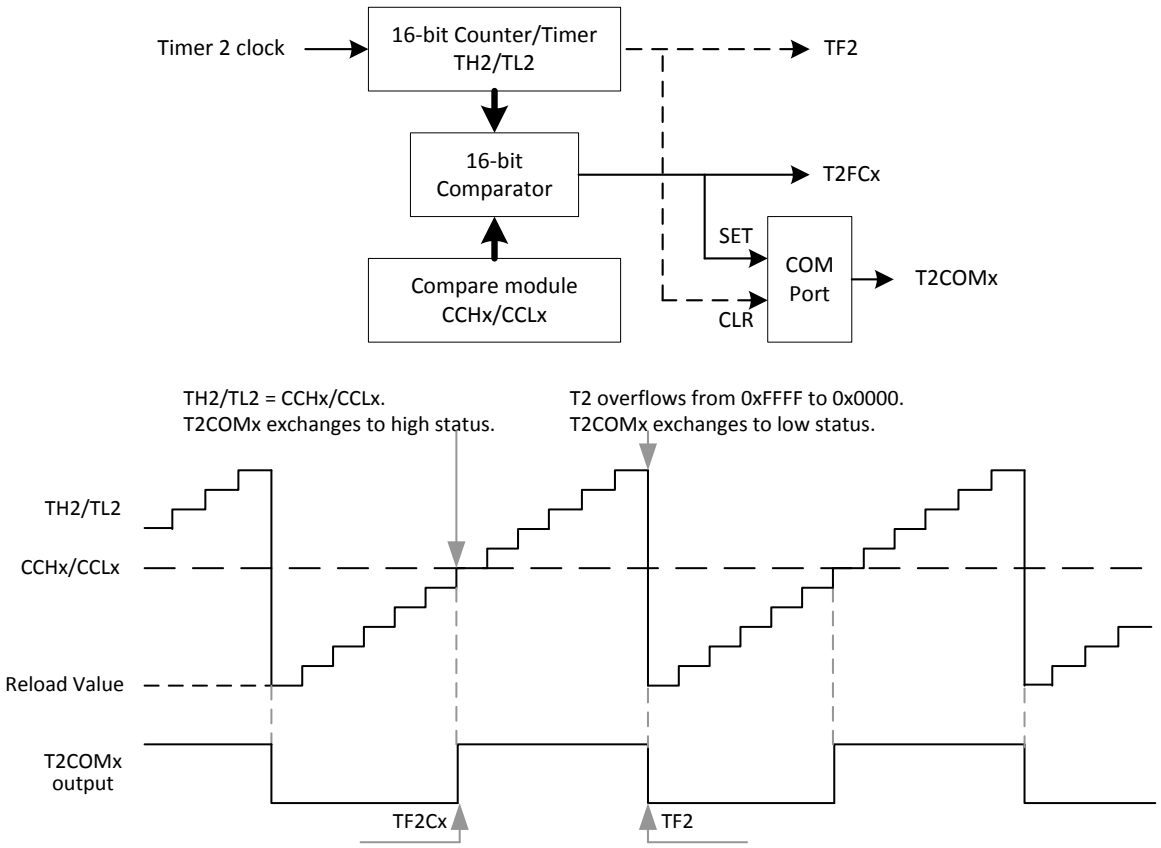
从另一方面来说，外部引脚 T2RL（与 P1.2 共用）的下降沿可以选择为重装信号。在这种情况下，若 T2RL 引脚保持稳定，T2 从 0000H 开始计数直到 0FFFFH，但只要 T2RL 引脚有下降信号，CRCH/CRCL 的值随时可能替换计数器的值。然后 T2 继续从 CRCH/CRCL 的值开始计数，若使能外部重装中断（ET2RL 和 ET2 都置 1），则置位外部重装中断标志（TF2RL）。外部中断向量与 T2 中断向量共用，由软件去判断。



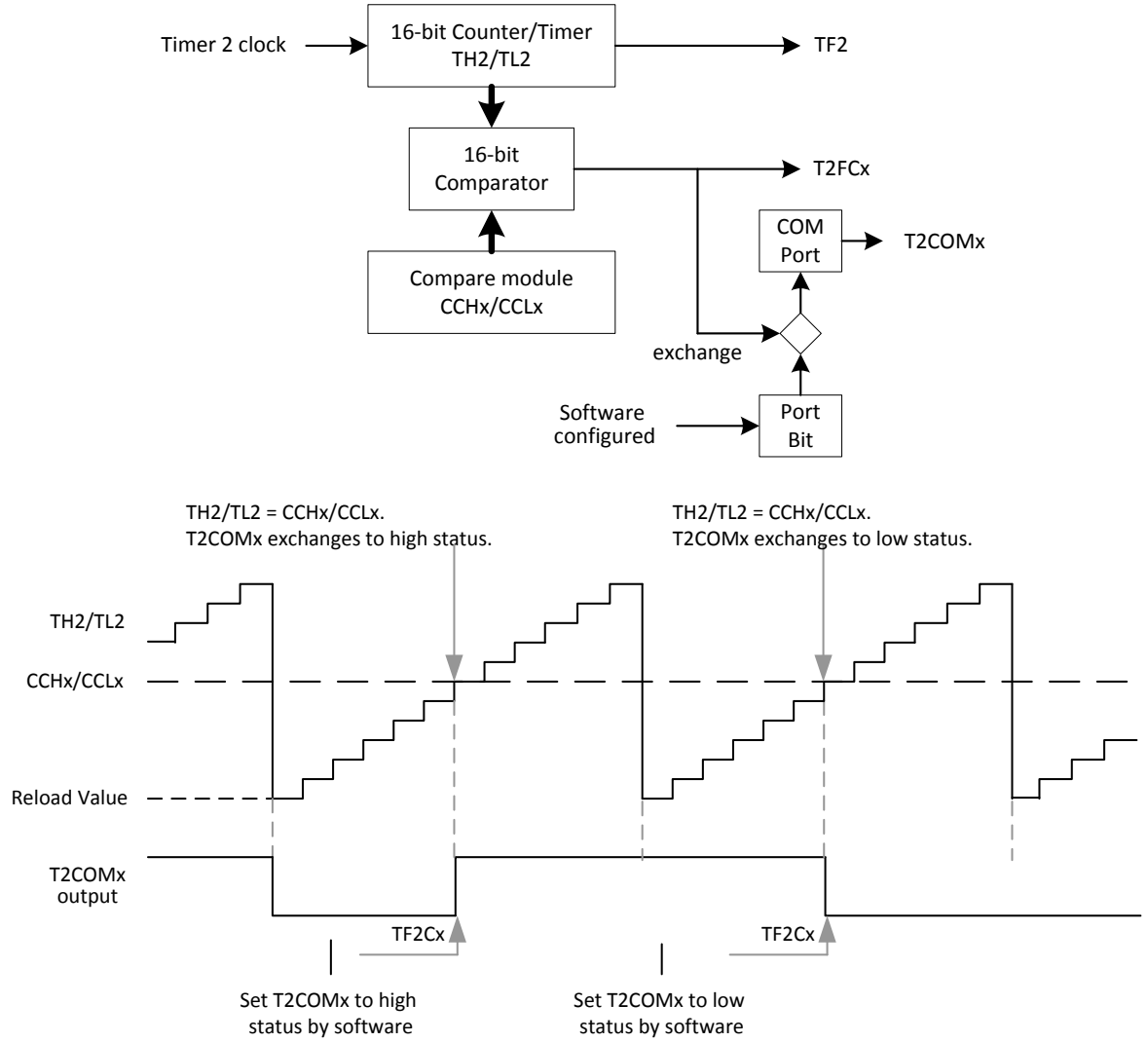
14.3 比较输出（PWM）

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（与 P0.3-P0.6 共用）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。

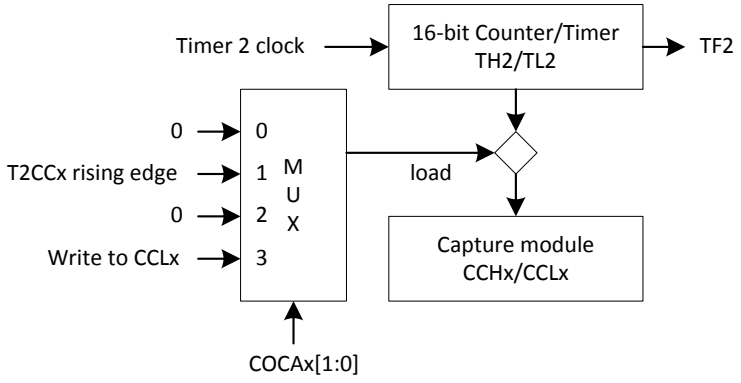


相比之下，间接输出是指保持相对应引脚先前的输出设置状态直到 T2 计数器的值超过 CRC/CC1/CC2/CC3 寄存器的值。在该模式下，可以由软件来控制输出信号的转换，换句话说，就是 TH2/TL2 和 CRC 寄存器的值相等时，P0.3 寄存器位会影响 T2COM0/P0.3 引脚。而 T2 的溢出则不会导致输出变化。

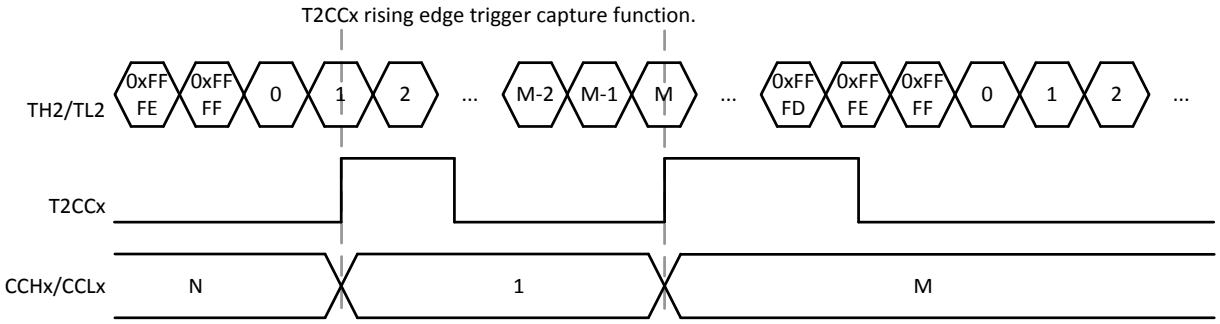


14.4 捕捉功能

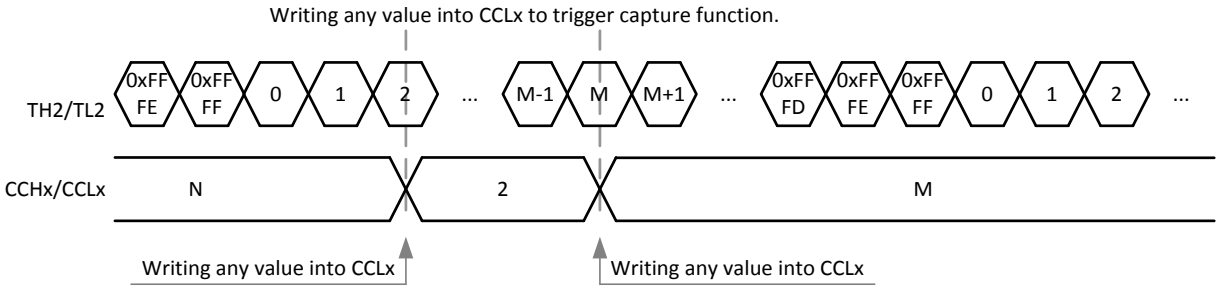
捕捉功能类似于秒表的 split/lap 按键，当 T2 计数器（TH2/TL2）开始向上计数时，一个 split 事件在 CRC/CC1/CC2/CC3 寄存器中记录计数器的值。



split 事件可由硬件或软件提供，T2CC0 引脚（与 P0.0 共用）可触发硬件 split 事件，复制 TH2/TL2 的值到 CRCH/CRCL 寄存器，而 T2CC1（P0.1）、T2CC2（P0.7）和 T2CC3（P1.0）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。



14.5 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	EX1	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	T2 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..0	TH2/TL2	R/W	0x00	16 位 T2 定时器寄存器。

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。
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。

14.6 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。

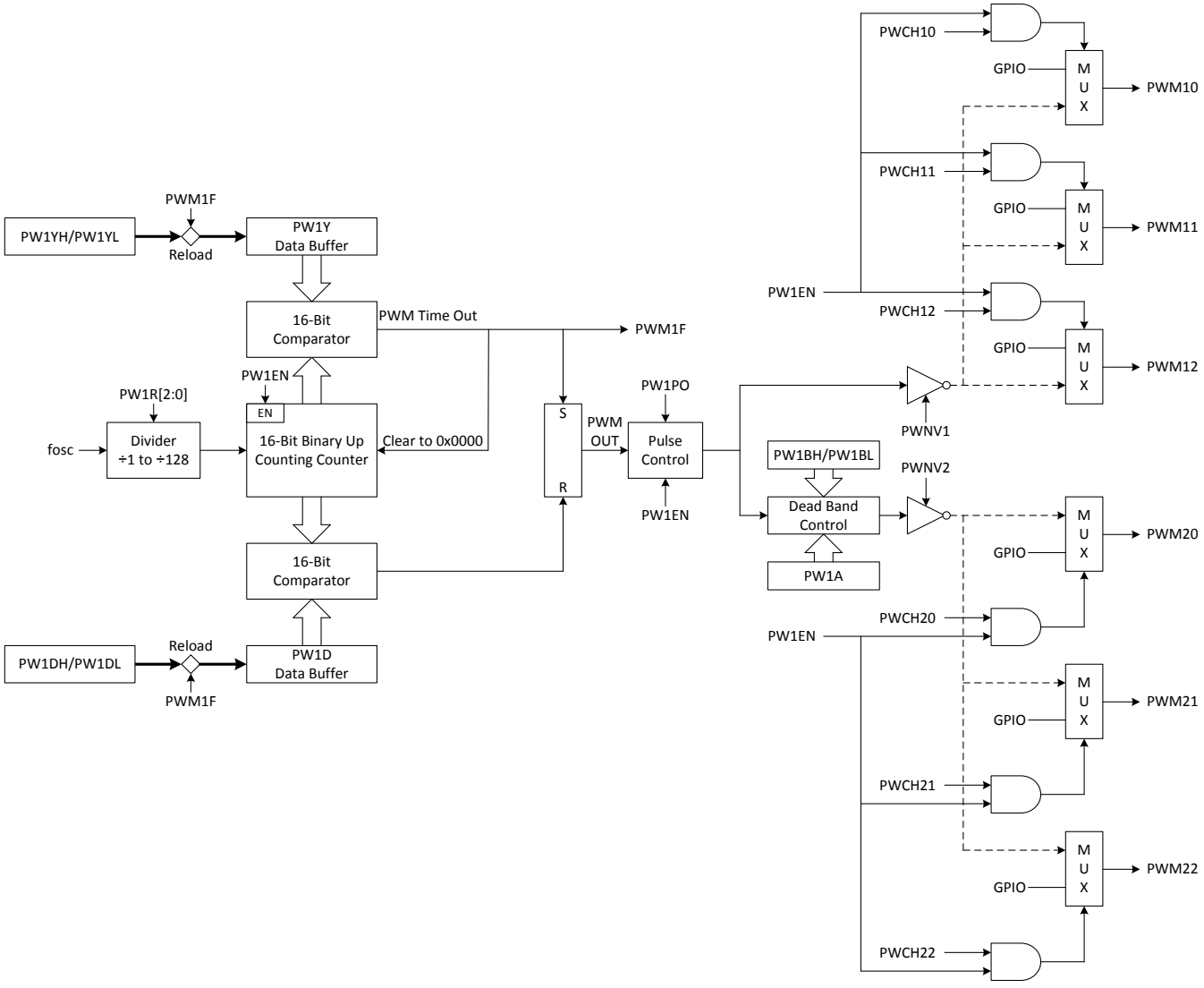
```
#define T2ClkFcpu     (1 << 0) // T2 clock from Fcpu
#define T2ClkPin      (2 << 0) // T2 clock from T2 pin
#define T2ClkGate     (3 << 0) // T2 clock from Fcpu with T2 pin gating
#define T2Fcpu12      (0<<7) // T2 clock = Fcpu/12
#define T2Fcpu24      (1 << 7) // T2 clock = Fcpu/24
#define T2RLMMode0    (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;

 //P11(T2)/P12(T2RL) is input mode with pull-high resister
   P1M &= 0Xf9;
 P1UR&= 0x06;

// Enable T2RL/T2COM0/1/2/3 interrupt
 IEN1| = 0Xbc;
// Enable total/Timer2 interrupt
 IEN0 | = 0Xa0;

   P0 = 0x00;
   P0M = 0x3F
}
Void T2Interrupt(void) interrupt ISRTimer2// 0x2B
 {// TF2/TF2RL clear by software
If ((IRCON & 0x40) ==0x40){
```

```
55      IRCON & = 0Xbf;   //Clear TF2
56    P20 = ~P20;
57    }
58  If ((IRCON & 0x80 == 0x80) {
59      IRCON & = 0x7F;   //Clear TF2RL
60    P21 = ~P21;
61      }
62  }
63  Void T2COM0Interrupt(void) interrupt ISRCom1     // 0x53
64  {// TF2C0 clear by hardware
65    P22 = ~P22;
66  }
67    Void T2COM1Interrupt(void) interrupt ISRCom2     // 0x5B
68  {// TF2C1 clear by hardware
69    P23 = ~P23;
70  }
71    Void T2COM2Interrupt(void) interrupt ISRCom3     // 0x63
72  {// TF2C2 clear by hardware
73    P24 = ~P24;
74  }
75    Void T2COM3Interrupt(void) interrupt ISRCom4     // 0x6B
76  {// TF2C3 clear by hardware
77    P25 = ~P25;
78  }
```

15 PWM

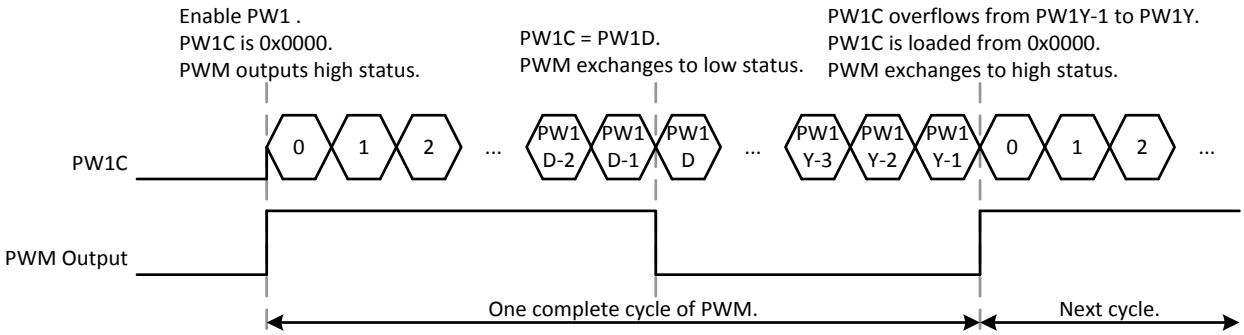
PW1 定时器各自包含一个 16 位二进制 6 通道 PWM，一个单脉冲 PWM。计数器计数到上限值（PW1Y）后，计数器清零并触发一个中断信号。每个 PWM 的占空比周期由 PW1D 控制。

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 6 个可编程控制的 PWM 通道，与 GPIO 引脚共用，由 PW1CH 位控制。通过使能 PW1CH 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 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。PWM 时钟源为 Fosc，可以分频为 Fosc/1~Fosc/128，由 PW1R[2:0]控制位。由 EPWM1 控制 PW1 的中断功能。



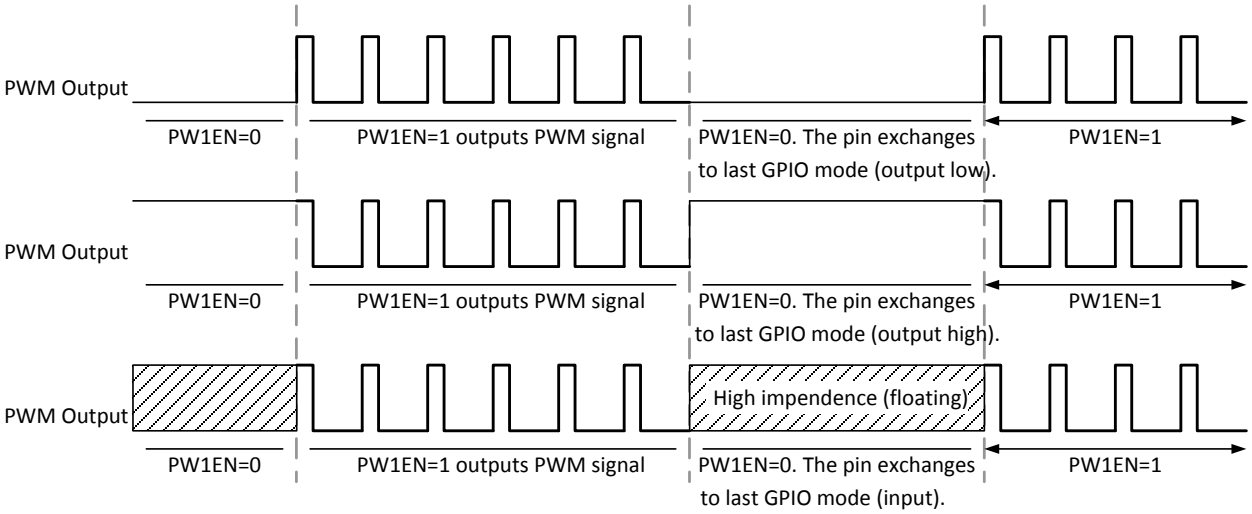
15.1 普通PWM

PW1 定时器内置 PWM 功能 ,由 PW1EN 和 PW1CH 控制 ,PWM10 ,PWM11 ,PWM12，PWM20 ,PWM21，PWM22 为输出引脚，并与 GPIO 引脚(P2.2，P2.4，P1.7，P2.3，P2.5，P1.6)共用，由 PW1CH 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（PW1Y-1 到 PW1Y）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。



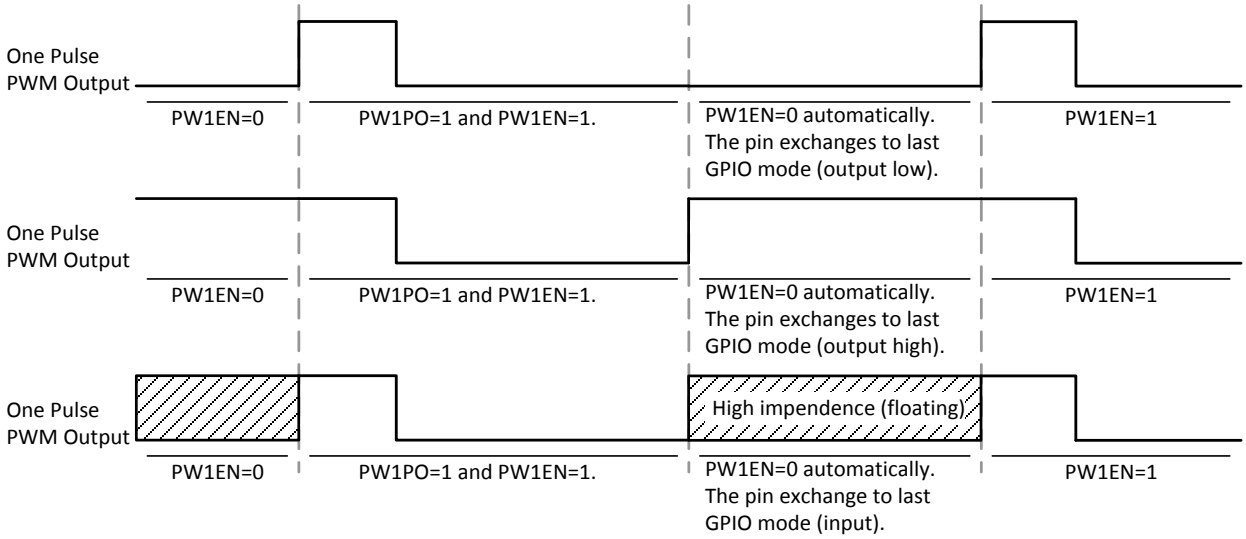
PWM 周期 = PW1Y

PWM 占空比 = (PW1D): (PW1Y-PW1D)



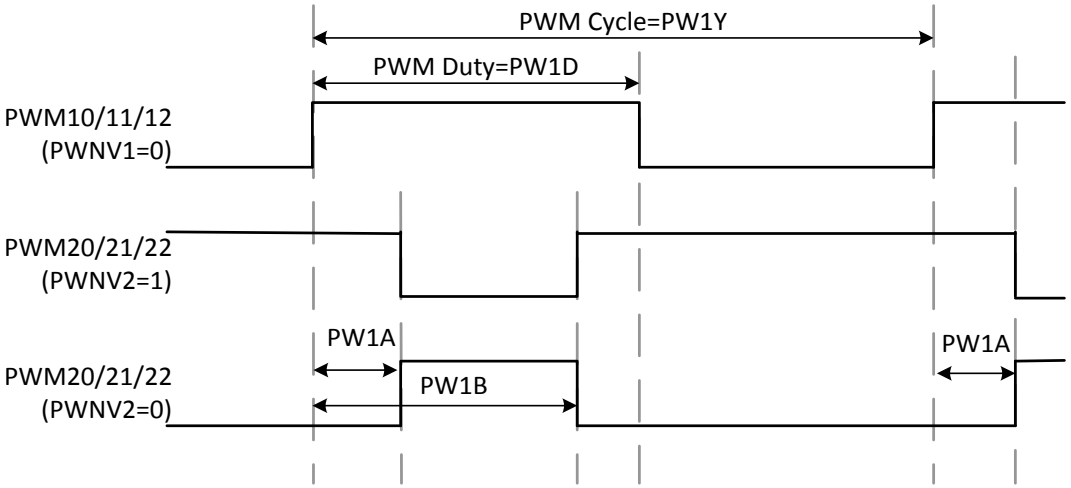
15.2 单脉冲PWM

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 进行选择。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。



15.3 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。

15.4 PWM寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1M	PW1EN	PW1R2	PW1R1	PW1R0	PWNV2	PWNV1	PW1CM	PW1PO
PW1CH	-	PWCH22	PWCH21	PWCH20	-	PWCH12	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
IEN0	EAL	-	ET2	ES0	ET1	EX1	ET0	EX0
IEN4	EPWM1	-	-	-	PWM1F	-	-	-

PW1CH 寄存器（0xBE）

Bit	Field	Type	Initial	说明
7	Reserved	R/W	0	
6	PWCH22	R/W	0	PWM1 共用引脚控制位。
5	PWCH21	R/W	0	0：GPIO 引脚；
4	PWCH20	R/W	0	1：PWM 输出引脚(共用 P1.6/P2.5/P2.3)。
3	Reserved	R/W	0	
2	PWCH12	R/W	0	PWM1 共用引脚控制位。
1	PWCH11	R/W	0	0：GPIO 引脚；
0	PWCH1	R/W	0	1：PWM 输出引脚(共用 P1.7/P2.4/P2.2)。

PW1M 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	PW1R[2:0]	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	PWM20/21/22 引脚输出控制位。 0：正常输出； 1：反向输出。
2	PWNV1	R/W	0	PWM10/11/12 引脚输出控制位。 0：正常输出； 1：反向输出。
1	PW1CM	R/W	0	PWM1 输出和 CM[触发同步控制位。 0：禁止； 1：使能。
0	Pw1PO	R/W	0	单脉冲功能。 0：禁止； 1：使能。

*周期设置为 0000H 时，使能 PWM 后，PWM 会停止工作，其周期不能更新。

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。

*在开始 PWM 功能之前，必须将周期配置设置完成。

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	PW1A	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	中断使能控制位。参考中断章节。
Else				参考其他章节。

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	

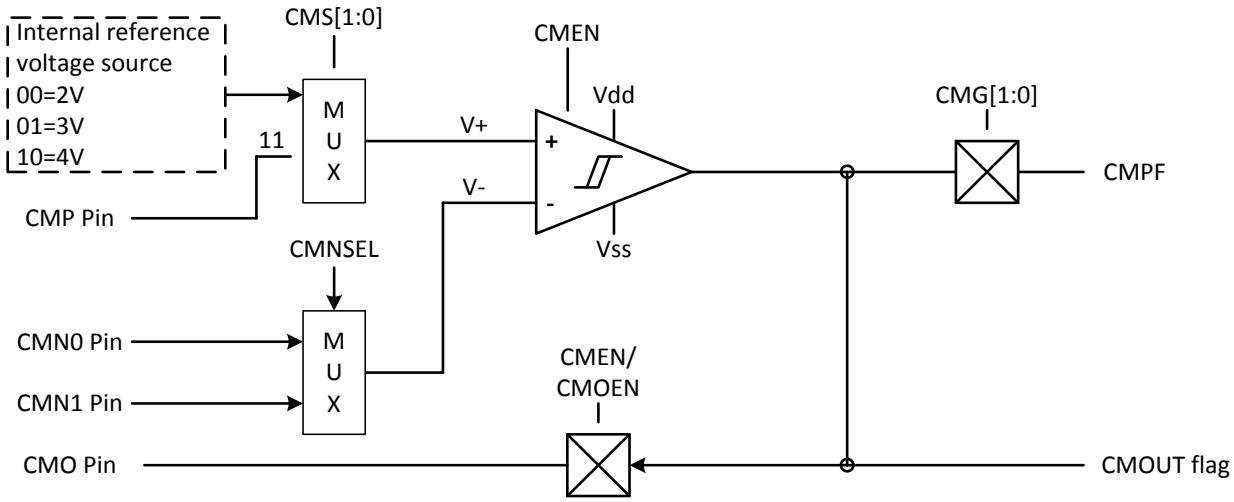
15.5 示例程序代码

下面的示例程序代码显示了在中断下如何执行 PW1 中断比较功能。

```
#define PW1Inv1         (1 << 2)      //PWM10/11/12 output inverse
#define PW1Inv2         (1 << 3)      //PWM20/21/22 output inverse
#define PW1CMPTri       (1 << 1)     //PW1 output Control by Comparator trigger
#define PW1OnePu          (1 << 0)    //Enable PW1 pulse output function
#define PWM10En           (1 << 0)    //Enable PWM10 output function
#define PWM11En           (2 << 0)    //Enable PWM11 output function
#define PWM12En           (4 << 0)    //Enable PWM12 output function
#define PWM20En           (1 << 4)    //Enable PWM20 output function
#define PWM21En           (2 << 4)    //Enable PWM21 output function
#define PWM22En           (4 << 4)    //Enable PWM22 output function
#define PW1En             (1 << 5)    //Enable PWM1 output function

void InitPWM(void)
{
   //PWM1_Initial
   PW1YH = 0x80;
   PW1YL = 0x00;
   PW1DH = 0x40;
   PW1DL = 0x00;
   PW1BH = 0x60;
   PW1BL = 0x00;
   PW1A = 0x80;

   //PW10/11/12/20/21/22 channel enable
   PW1CH = PWM10En | PWM11En | PWM12En | PWM20En | PWM21En | PWM22En;

   //PWM1 enable,P10/11/12 output inverse, clock = Fosc/32
   PW1M = PW10En | PW21Inv1 | 0x20;

  // Enable PWM1 interrupt & clear PWM1F
   IEN4 | = 0x80;

   //Enable total interrupt
   IEN0 | = 0x80;

   P0 = 0x00;
   P0M | = 0x01;
}

void PW1Interrupt(void) interrupt ISRPwm1     //0x83
{ //PWM1F clear by software
   if ((IEN4 & 0x08) == 0x08)   {
  IEN4 & = 0Xf7;       //Clear PWM1F
  P00 = ~P00
  }
}
```

16 比较器

SN8F5703 内置 1 个比较器功能。当正极输入电压大于负极输入电压时，比较器输出高电平；当正极输入电压小于负极输入电压时，比较器输出低电平。比较器的正极输入电源为内部 2V/3V/4V 或者 CMP，可以通过程序设置比较器的有效触发沿，可用于控制 PWM 的关闭和打开。在不同的应用下，比较器具有状态指示功能，中断功能和 IDLE 模式唤醒功能。



16.1 操作配置

比较器引脚与 GPIO 引脚共用，由 CMEN 位控制。CMEN=1 时，使能 CMN/CMP 引脚，连接到比较器负极。CMOEN 控制比较器输出 ,连接到 GPIO 或不连接到 GPIO 引脚。CMOEN=1 时 ,比较器输出端连接到 GPIO 引脚，并屏蔽 GPIO 功能。

内部参考电压包括 2V/3V/4V，由 CMS[1:0]控制。当 CMEN=1，如果 CMNSEL=0，CMN0(P1.5)是比较器的负极引脚，而 CMN1(P1.6)为 GPIO 引脚。反之 CMN1(P1.6)为比较器的负极引脚，CMN0(P1.5)是 GPIO 引脚。比较器引脚的配置如下表所示：

CMEN	CMNSEL	比较器负极引脚	比较器正极引脚(CMnS[1:0])				比较器输出引脚(CMnOEN)	
			00	01	10	11	0	1
CMEN=0	CMNSEL=X	所有引脚为 GPIO 模式，禁止比较器功能。						
CMEN=1	CMNSEL=0	CMN0	2V	3V	4V	CMP	GPIO	CMO
	CMNSEL=1	CMN1						

16.2 比较器输出功能

比较器输出信号就是其输出功能源，比较器输出功能包括：

- CMOUT 输出标志：比较器输出信号直接连接到 CMOUT 标志，CMOUT 位立即指示比较器的状态，程序从 CMOUT 位读取比较器的状态。
- 比较器外部引脚输出功能：比较器的输出状态可从 CMO(共用 P1.3)引脚输出，由 CMOEN 位控制。CMOEN=0 时，比较器输出引脚为 GPIO 模式；CMOEN=1 时，CMO 引脚输出比较器的输出状态，请屏蔽 GPIO 功能。
- 比较器边沿触发和中断功能：比较器内置中断功能，和可编程控制的边沿触发功能。CMG[1:0]位控制比较器的触发沿方向。比较器边沿触发时 ,CMPF 立即置为 1 ,必须通过程序来将 CMPF 位清零。若 CMEN=1，CMPF=1 时，程序计数器指向中断向量，系统执行中断服务程序。当触发沿方向与中断触发条件相同时，系统会立即执行中断操作。
- 比较器 IDLE 模式唤醒功能：比较器的唤醒功能只在 IDLE 模式下（需要使能中断）有效，在 STOP 模式下是无效的。若检测到边沿触发（比较器输出状态改变）时，系统会从 IDLE 模式下被唤醒。由于在中断触发条件下设置 CMPF 为 1，因此使能中断时就执行中断程序。

16.3 PWM输出控制

比较器的结果可以用来控制 PWM 输出，用户可通过 CMPT 寄存器来选择合适的控制模式。下表是控制模式列表：

CMT[1:0]	PWM 同步触发操作
00	CMP 与 PWM 输出无关
01	CMPF = 1 ➔ PWM 停止
10	CMP > CMN（上升沿触发）➔ PWM 输出 CMP < CMN（下降沿触发）➔ PWM 停止
11	CMP < CMN（下降沿触发）➔ PWM 输出 CMP > CMN（上升沿触发）➔ PWM 停止

比较器可通知 PWM 输出，取决于 PW1CM 位的设置。详细说明请参考 PWM 章节 PW1M 寄存器 bit 1 ：PW1CM。

16.4 比较器寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMPM	CMEN	CMNSEL	CMS1	CMS0	CMOEN	CMOUT	CMG1	CMG0
CMPT	-	-	-	-	-	-	CMT1	CMT0
P1CON	P1CON7	P1CON6	P1CON5	P1CON4	P1CON3	P1CON2	P1CON1	-
IEN0	EAL	-	ET2	ES0	SE1	EX1	ET0	EX0
IEN2	-	-	-	-	-	ECMP	EADC	-
IRCON2	-	-	-	-	-	-	CMPF	ADCF

CMPM 寄存器（0x9C）

Bit	Field	Type	Initial	说明
7	CMEN	R/W	0	比较器 CMP 控制位。 0：禁止，CMP/CMN 引脚为 GPIO 模式； 1：使能，CMP/CMN 引脚为 CMP 输入引脚。
6	CMNSEL	R/W	0	比较器 CMP 负极引脚选择位。 0：CMN 引脚为 CMN0（P1.5）； 1：CMN 引脚为 CMN1（P1.6）。
5..4	CMS[1:0]	R/W	00	CMP 正极输入电压控制位。 00：2.0V；01：3.0V；10：4.0V；11：CM0P(共用 P1.4)。
3	CMOEN	R/W	0	比较器 CMP 输出引脚控制位。 0：禁止；CMO(P1.3)为 GPIO 模式； 1：使能，CMO(P1.3)为比较器输出引脚，屏蔽 GPIO 功能。
2	CMOUT	R/W	0	比较器 CMP 输出标志位。 0：CMP 电压小于 CMN 电压； 1：CMP 电压大于 CMN 电压。
1..0	CMG[1:0]	R/W	00	比较器中断触发沿控制位。 00：保留； 01：上升沿触发，CMP > CMN； 10：下降沿触发，CMP < CMN； 11：上升下降沿触发。

CMPT 寄存器（0Xce）

Bit	Field	Type	Initial	说明
7..2	Reserved	R	0	
1..0	CMT[1:0]	R/W	00	CMP 带 PWM 触发选择位。 00：CMP 与 PWM 输出无关； 01：CMPF = 1➔ PWM 停止； 10：CMP > CMN➔PWM 输出；CMP < CMN ➔ PWM 停止； 11：CMP < CMN➔PWM 输出；CMP > CMN➔PWM 停止。

P1CON 寄存器（0Xd6）

Bit	Field	Type	Initial	说明
6..4	P1CON[6:4]	R/W	0x00	P1 配置控制位*。 0：P1 为模拟输入引脚（CMP 输入引脚）或数字 GPIO 引脚。 1：P1 为单纯的模拟输入引脚，不能作为数字 GPIO 引脚。
Else				请参考其它章节。

* P1CON [6:4]可以配置相关的 P1 引脚为单纯的模拟输入引脚以避免漏电流的发生。

IEN0 寄存器（0Xa8）

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

IEN2 寄存器（0x9A）

Bit	Field	Type	Initial	说明
2	ECMP	R/W	0	比较器 CMP 中断控制位。 0：禁止 CMP 中断； 1：使能 CMP 中断。
Else				请参考其它章节。

IRCON2 寄存器（0Xbf）

Bit	Field	Type	Initial	说明
1	CMP1F	R/W	0	比较器 CMP 中断请求标志位。 0：无 CMP 中断请求； 1：CMP 请求中断。
Else				请参考其它章节。

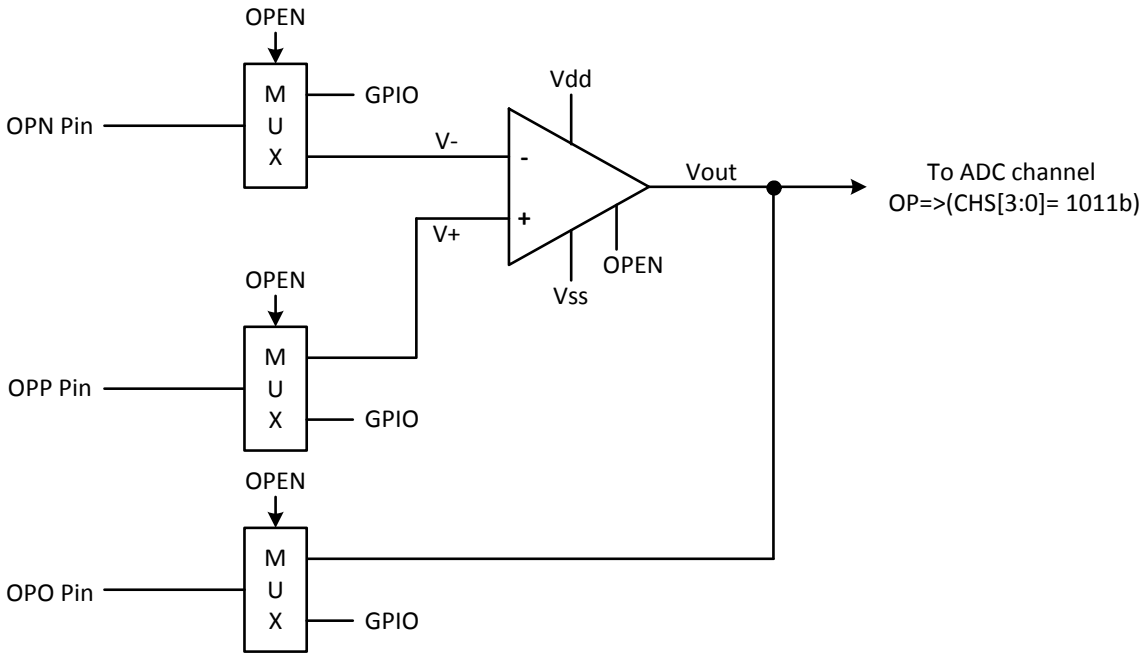
16.5 示例程序代码

下面的示例程序代码显示了如何执行带中断的 CMP。

```
#define LevelChange    (3 << 0)   // CMP > CMN or CMP < CMN
#define CMNGreCMP       (2 << 0)   // CMP < CMN
#define CMPGreCMN       (1 << 0)   // CMP > CMN
#define CMPOEN           (1 << 3)   // CMP output pin enable
#define CMPVin2V    (0 << 4)   // CMP positive Vin connect 2.0V
#define CMPVin3V    (1 << 4)   // CMP positive Vin connect 3.0V
#define CMPVin4V    (2 << 4)   // CMP positive Vin connect 4.0V
#define CMPVINP          (3 << 4)   // CMP positive Vin connect CMP
#define CMPEN            (1 << 7)   //enable CMP
#define ECMP         (1 << 4)   //enable CMP interrupt

void CMP0Init(void)
{
 P1   = 0x00;
 P1M = 0x80;

 // set CMP pins'   mode at pure analog pin
 P1CON |= 0x1C;    //P15/P14/P13
 // configure CMP positive Vin and interrupt trigger.
 // enable CMP and output pin
 CMPM = CMPEN | CMPVin4V | CMPOEN | CMPPGreCMN;

 // enable CMP interrupt
 IEN0 |= 0x80;      //enable global interrupt
 IEN2 |= ECMP;
}

void CMPInterrupt(void) interrupt ISRCmp
{
 if ((IRCON2 &0x08) == 0x08) {
     P17 = ~P17;
     IRCON2 &= 0Xfd;    //Clear CMPF
 }
}
```

17 OPA

SN8F5703 内置 1 个可操作的放大器 (OP1。OP-Amp 的电源范围为 VSS-VDD，OP-Amp 的输入信号和输出电压都在该电压范围内。OP-Amp 的输出引脚可编程控制，与 ADC 输入通道连接，用于电压测量。



17.1 操作配置

OP-AMP 引脚与 GPIO(P1.3，P1.5)引脚共用，由 OPEN 位控制。OPEN=0 时，OP AMP 引脚为 GPIO 引脚；OPEN=1 时，GPIO 引脚切换为 OP-AMP 引脚并屏蔽 GPIO 功能。OP-AMP 引脚的选择列表如下所示：

OPEN	OP 正极引脚	OP 负极引脚	OP 输出引脚
OPEN = 0	所有引脚都为 GPIO 模式。		
OPEN = 1	OPP (Vin+)	OPN (Vin-)	OPO (Vout)

OP 输出引脚也连接到 ADC 内部 AIN11 通道=> CHS[3:0]。详见 ADC 章节内容。

17.2 OPA寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPM	-	-	-	-	-	-	-	OPEN
P1CON	P1CON7	P1CON6	P1CON5	P1CON4	P1CON3	P1CON2	P1CON1	-

OPM 寄存器（0x9B）

Bit	Field	Type	Initial	说明
Else	Reserned	R	0	
0	OPEN	R/W	0	OP-Amp 使能位。 0：禁止 OP-Amp，OPO/OPP/OPN 引脚为 GPIO 模式。 1：使能 OP-Amp，OPO/OPP/OPN 为 OP-Amp 输入输出引脚。

*OPO/P1.3，OPP/P1.4，OPN/P1.5

P1CON 寄存器（0XD6）

Bit	Field	Type	Initial	说明
5..3	P1CON[5:3]	R/W	0x00	P1 配置控制位*。 0：P1 为模拟输入引脚（OP 输入引脚）或数字 GPIO 引脚； 1：P1 为单纯的模拟输入引脚，不能作为数字 GPIO 引脚。
Else	Reserved	R	0	

* P1CON [5:3]可以配置相关的 P3 引脚为单纯的模拟输入引脚以避免漏电流的发生。

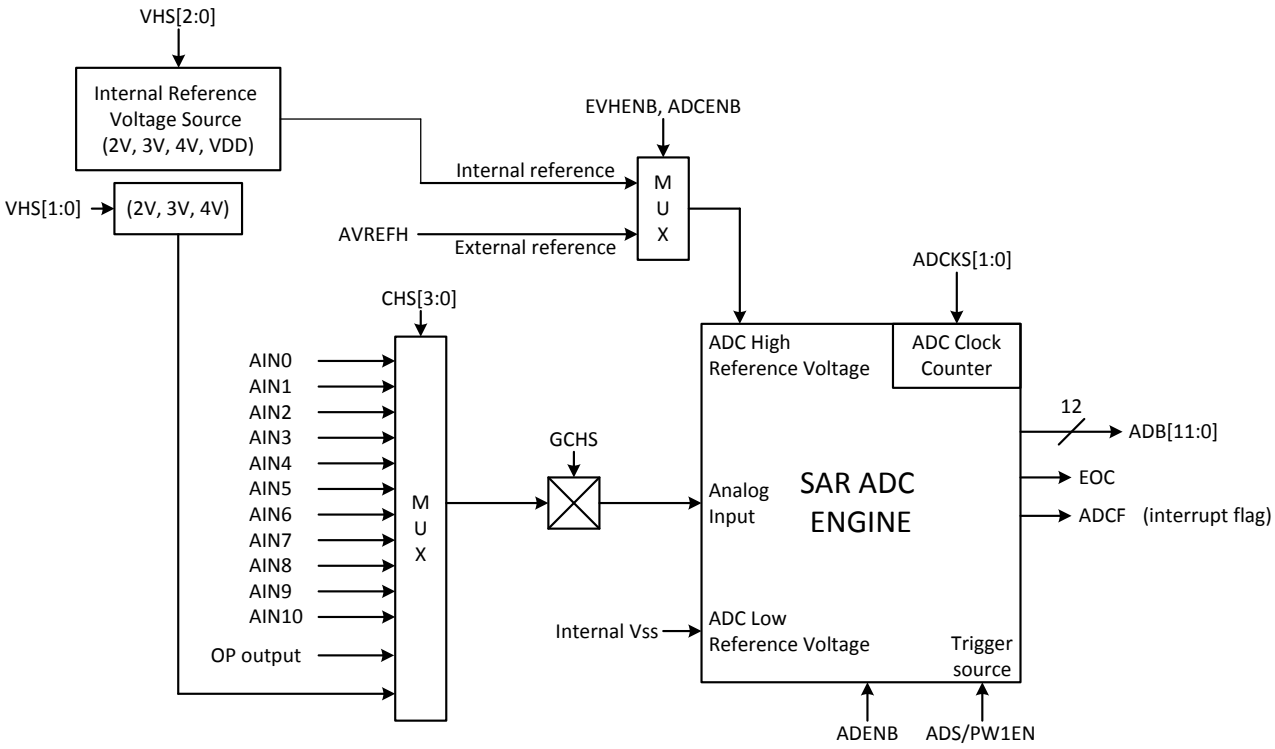
17.3 示例程序代码

下面的示例程序代码显示了如何执行 OP0。

```
#define OPEN          (1 << 0)

void OPEn(void)
{
   // set OP0 pins'   mode at pure analog pin
   P1CON = 0x38;

   // enable OP
   OPM |= OP0EN;
}
```

18 ADC

模拟数字转换（ADC）是一个 SAR 结构，内置 11 个输入通道（AIN0~AIN10），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 11 个模拟通道可以测量 11 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 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。



18.1 操作配置

ADC 开始转换前必须先设置好下列配置，具体如下：

1 选择和使能 ADC 输入通道（由 CHS[3:0]位和 GCHS 位设置）；
2 将 ADC 输入通道设置为输入模式（由 PnM 寄存器设置）；
3 必须禁止 ADC 输入通道的内部上拉电阻（由 PnUR 寄存器设置）；
4 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）；
5 选择 ADC 高参考电压（由 VREFH 寄存器设置）；
6 选择 ADC 时钟频率（由 ADCKS[1:0]位设置）；
7 设置 ADENB 位后，ADC 准备开始转换。

由 ADENB 位使能 ADC 功能时，要保证由程序启动 ADC，写入 1 到 ADM 寄存器的 ADS 位。设置 ADENB 和 ADS 位后，ADC 开始将模拟信号转换为数字信号。转换结束后，ADS 复位为逻辑 0，ADC 电路将 EOC 和 ADCF 位置 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。ADPWS=1 时，若使能 PWM 作为转换源，则 ADC 继续转换直至禁止 PWM。

18.2 ADC输入通道

SN8F5703 的 ADC 内置 11 个输入通道（AIN0~AIN10），用于测量 11 种不同的模拟信号源，由 CHS[3:0]和 GCHS 位控制。AIN11 通道为 OP –Amp 输出端；AIN12 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，AIN12 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。

CHS[3:0]	Channel	Pin name	备注
0000	AIN0	P2.0	
0001	AIN1	P2.1	
0010	AIN2	P2.2	
0011	AIN3	P2.3	
0100	AIN4	P2.4	
0101	AIN5	P2.5	
0110	AIN6	P1.7	
0111	AIN7	P1.6	
1000	AIN8	P1.5	
1001	AIN9	P1.4	
1010	AIN10	P1.3	
1011	AIN11	OP	OP-AMP 输出端

1100	AIN12	Internal 2V or 3V or 4V	电池检测器通道
1101-1111	-	-	保留

18.2.1 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 ADC 通道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。

18.3 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/P2.0），此时必须设置 P2.0 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。内部参考电压的局限性是不能超过 VDD，最高等于 VDD。若选择 AIN12 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。

18.3.1 信号格式

ADC 采样电压范围为参考电压高/低电平之间。ADC 参考低电压固定为 VSS。高电压包括 VDD/4V/3V/2V 和外部参考电压（由 P2.0/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 参考低电压为 VSS=0V，故 ADC 参考高电压范围为 2V-VDD。外部参考电压需在此范围之内。

- ADC 内部参考低电压=0V。（EVHENB = 0）
- ADC 内部参考高电压=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

18.4 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

18.5 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

18.6 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	TCHEN	GCHS	ADCKS1	ADCKS0	ADB3	ADB2	ADB1	ADB0
VREFH	EVHENB	-	-	ADPWS	-	VHS2	VHS1	VHS0
P1CON	P1CON7	P1CON6	P1CON5	P1CON4	P1CON3	-	-	-
P2CON	-	-	P2CON5	P2CON4	P2CON3	P2CON2	P2CON1	P2CON0
IEN0	EAL	-	ET2	ES0	Et1	Ex1	Et0	Ex0
IEN2	-	-	-	-	-	EPWM	EADC	-
IRCON2	-	-	-	-	-		CMPF	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）。
4	Reserved	R	0	
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；1011：AIN11*(1) ； 1100：AIN12*(2)；others：保留；

*(1) AIN11 通道为 OP-Amp 的输出端。

*(2) AIN12 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*[11:4]。

* 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
7	Reserved	R/W	0	
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/P2.0 为 GPIO 引脚。 1：禁止 ADC 内部 VREFH 功能，AVREFH/P2.0 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 内部参考高电压选择位(AIN12 为内部 2V/3V/4V 输入通道)。 0：由 VHS[1:0]*(3)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 内部参考高电压选择位。 00：2.0V； 01：3.0V； 10：4.0V； 11：VDD。
Else	Reserved	R	0	

*(1) AVREFH 的电压值必须在 VDD 到 2.0V 范围内。

*(2) AIN12 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[7:1]	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 [7:1]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
5..0	P2CON[5:0]	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。

IEN0 寄存器（0Xa8）

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

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。

18.7 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 ADCAIN12_VDD     (3 << 0)   //AIN12 = VDD
#define ADCAIN12_4V      (2 << 0)   //AIN12 = 4.0V
#define ADCAIN12_3V      (1 << 0)   //AIN12 = 3.0V
#define ADCAIN12_2V      (0 << 0)   //AIN12 = 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
 P2CON |= 0x20;     //AIN5/P25
 P2M     &= 0Xdf;      //input mode
 P2UR   &= 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;
```

```
54   }
55 }
```

19 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，模式 0 是移位寄存器模式，作为同步收发器。模式 1-模式 3 下，UART 是 8 位或 9 位数据的异步收发器。传输格式有起始位、8 位/ 9 位数据和停止位。通过写 S0BUF 寄存器开始传输。接收后，S0BUF 寄存器完成接收后，输入数据可用。TB80/RB80 位可用作 9 位 UART 模式下传输和接收的第九位。可编程的波特率支持不同速度的外围设备。

UART 特征如下：

- 全双工，双线同步/异步数据传输。
- 可编程的波特率。
- 8 位移位寄存器：同步收发器。
- 8 位/ 9 位 UART：异步收发器，带 8 或 9 位数据位和可编程的波特率。

19.1 UART 操作

UART UTX 和 URX 引脚与 GPIO 共用。在同步模式下，UTX/URX 共用引脚必须由软件设置为输出高。在异步模式下(8 位/ 9 位 UART)，UTX 共用引脚必须由软件设置为输出高， URX 设置为输入高。因此，URX/UTX 引脚将转为 UART 用途。禁用 UART 时，UART 引脚返回 GPIO 的上一个状态。

UTX/URX 引脚也支持开漏结构。开漏选项由 PnOC 位控制。PnOC = 0 时，禁用 UTX/URX 开漏结构。PnOC = 1 时，启用 UTX/URX 开漏结构。如果启用开漏结构，UTX/URX 引脚必须设定为高电平(IO 模式控制将被忽略)，需要外部上拉电阻。

UART 支持中断功能。ES0 是 UART0 传输中断功能控制位。ES0 控制 UART 的发送和接收功能。ES0 = 0，禁用收发器的中断功能。ES0 = 1 ，启用 UART 收发器的中断功能。UART 的发送和接收中断功能共用中断向量 0x0023。UART 中断功能使能时，UART 操作之后 程序计数器指向中断向量执行 UART 中断服务程序。当禁用中断时，TI0/RI0 是 UART0 中断请求标志，也是 UART 操作状态指示符。TI0 和 RI0 必须由软件清零。

UART 提供四个操作模式 (一个同步和三个异步), 均由 S0CON 寄存器控制。这些模式可以支持不同的波特率和通信协议。

SM0	SM1	模式	同步	时钟率	开始位	数据位	停止位	UART 引脚的模式和数据
0	0	0	同步	Fcpu/12	X	8	X	UTX 引脚： P00M=1 和 P00=1 URX 引脚： 发送器： P01M=1 和 P01=1 接收器： P01M=0 和 P01=1
0	1	1	异步	波特率生成器或 T1 溢出率	1	8	1	UTX 引脚： P00M=1 和 P00=1 URX 引脚： P01M=0
1	0	2	异步	Fcpu/64 或 Fcpu/32	1	9	1	
1	1	3	异步	波特率生成器或 T1 溢出率	1	9	1	

19.2 模式 0：同步 8 位收发器

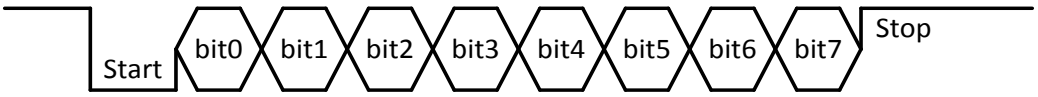
模式 0 是移位寄存器模式，作为同步收发器。UTX 引脚都输出移位时钟用于传输和接收状态。URX 引脚用于传输和接收数据。首先，发送/接收带 LSB 的 8 位数据。波特率为 Fcpu/12。通过写数据到 S0BUF 寄存器开始传输。第 8 位传输结束后，设置 TI0 标志位。通过设置 REN0 位来控制接收，清 RI0 位。REN0=1 和 RI0 由 1 到 0 时，开始传输数据，且在接收完第 8 位时，设置 RI0 标志位。

19.3 模式 1：可变波特率异步 8 位收发器

模式 1 支持可变波特率的异步 8 位 UART。传输格式包括 1 位起始位、8 位数据位(LSB 在前)和 1 位停止位。通过 UTX 引脚传输数据，通过 URX 引脚接收数据。波特率时钟源可以是波特率生成器或 BD 位控制的 T1 溢出。BD = 0 时，波特率时钟源来自 T1 溢出。BD =1 时，波特率时钟源是波特率生成器，由 S0RELH 和 S0RELL 控制。此外，波特率可以通过 SMOD 位翻倍。

由 REN0 位控制数据传输。传输配置之后，加载传输数据到 S0BUF，然后 UART 开始传输数据包。然后，在停止位的开始处设置 TI0 标志位。

由 REN0 位控制数据接收。REN0 = 1 时，数据接收功能启用。对于主终端，通过接收起始位开始数据接收，URX 检测到起始位的下降沿，然后在停止位的中间设置 RI0 标志位。直到接收完成，输入数据存储在 S0BUF 寄存器中，停止位存储在 RB80 中。



19.4 UART模式 2：固定波特率异步 9 位收发器

模式 2 支持固定波特率的异步 9 位 UART。传输格式包括 1 位起始位、9 位数据位(LSB 在前)和 1 位停止位。通过 UTX 引脚传输数据，通过 URX 引脚接收数据。波特率时钟源可以是波特率生成器或 BD 位控制的 T1 溢出。波特率时钟源固定为 fcpu/64 或 fcpu/32，且由 SMOD 位控制。SMOD=0 时，波特率是 fcpu/64。SMOD=1 时，波特率是 fcpu/32。

由 REN0 位控制数据传输。传输配置之后，加载传输数据到 S0BUF，然后 UART 开始传输数据包。从 TB80 获取第 9 位数据位。在停止位的开始处设置 TI0 标志位。

由 REN0 位控制数据接收。REN0 = 1 时，数据接收功能启用。对于主终端，通过接收起始位开始数据接收，URX 检测到起始位的下降沿 然后在停止位的中间设置 RI0 标志位。直到接收完成 输入数据的低 8 位存储在 S0BUF 寄存器中，第 9 位存储在 RB80 中。

19.5 UART模式 3：可变波特率异步 9 位收发器

模式 3 支持可变波特率的异步 9 位 UART。传输格式包括 1 位起始位、9 位数据位(LSB 在前)和 1 位停止位。通过 UTX 引脚传输数据 ,通过 URX 引脚接收数据。模式 2 和模式 3 的不同之处在于波特率的选择。在模式 3 条件下，波特率时钟源可以是波特率生成器或 BD 位控制的 T1 溢出。BD = 0 时，波特率时钟源来自 T1 溢出。BD = 0 时，波特率时钟源是波特率生成器，由 S0RELH 和 S0RELL 控制。此外，波特率可以通过 SMOD 位翻倍。

由 REN0 位控制数据传输。传输配置之后，加载传输数据到 S0BUF，然后 UART 开始传输数据包。从 TB80 获取第 9 位数据位。在停止位的开始处设置 TI0 标志位。

由 REN0 位控制数据接收。REN0 = 1 时，数据接收功能启用。对于主终端，通过接收起始位开始数据接收，URX0 检测到起始位的下降沿 ,然后在停止位的中间设置 RI0 标志位。直到接收完成 ,低 8 位输入数据存储在 S0BUF 寄存器中，第 9 位存储在 RB80 中。

19.6 多处理器通信

模式 2 和模式 3(9-bit UART)下，UART 支持主设备和一个或多个从设备之间的多处理器通信。主标识符通过使用第 9 位数据位纠正从标志符。通信开始时，主设备发送特定地址字节，且第 9 位设为“1”，以选择的从设备，然后在接下来的传输中发送一个数据字节，且第 9 位设为“0”。

多处理器通信由 SM20 位控制。当 SM20 =0 时，禁用多处理器通信。当 SM20 = 1，启用多处理器通信。如果设置了 SM20，只在第 9 位接收到的位为“1” (RB80) 时生成 UART 接收中断。从设备将通过软件比较接收的数据和已有的地址数据。如果地址字节匹配，在接下来的数据传输中从设备将清零 SM20，以启用中断功能。如果地址不匹配，则 SM20 保持为“1”， 在接下来的数据传输中不会产生中断。

19.7 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=8MHz）:

波特率	SMOD	S0RELH	S0RELL	Accuracy
4800 Hz	0	0x03	0xE6	0.16 %
9600 Hz	0	0x03	0xF3	0.16 %
19200 Hz	1	0x03	0xF3	0.16 %
38400 Hz	1	0x03	0xF9	-6.99 %
56000 Hz	1	0x03	0xFB	-10.71 %
57600 Hz	1	0x03	0xFC	8.51 %
115200 Hz	1	0x03	0xFE	8.51 %
128 kHz	1	0x03	0xFE	-2.34 %
250 kHz	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。

$$\text{Baud Rate} = 2^{\text{SMOD}} \times \frac{\text{T1 clock rate}}{32 \times (256 - \text{TH1})}$$

建议设置 T1 溢出间隔时间（T1 时钟=32M），用于普通 UART 波特率（Fcpu=8MHz）

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 %

✷ 注：

1. 波特率产生器源是 T1 溢出率，最大计数值是 0xFB（只支持 0x00~0xFB）。

2. 波特率产生器源是 T1 溢出率，系统时钟 fcpu 必须大于等于 T1 时钟率。

19.8 省电

UART 模块具有时钟门控功能，用于省电。REN0=0 时，停止 UART 模块内部时钟以省电，UART 相关寄存器（S0CON，S0CON2，S0BUF，SORELL，S0RELH 和 SMOD 位）都停止工作。

反之，REN0=1 时，UART 内部时钟工作，相关寄存器也可以访问。在初始化 UART 之前，REN0 必须设置为 1。

19.9 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
P0OC	-	-	-	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。

*当 REN0 位为 0 时，UART 相关寄存器无法访问，模块内部时钟停止。

✷ **注：中断使能，通过软件清空 TI0 和 RI0 。**

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。

P0OC 寄存器（0Xe4）

Bit	Field	Type	Initial	说明
1	P01OC	R/W	0	0：切换 P0.1（URX）为输入模式（要求）； ~~1：切换 P0.1（URX）为开漏模式*。~~
0	P00OC	R/W	0	0：切换 P0.0（UTX）为推拉模式； 1：切换 P0.0（UTX）为开漏模式。

Else				请参考其它章节。

*设置 P01OC 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
1	P01M	R/W	0	0：设置 P0.1（URX）为输入模式（要求）; ~~1：设置 P0.1（URX）为输出模式*。~~
0	P00M	R/W	0	~~0：设置 P0.0（UTX）为输入模式*；~~ 1：设置 P0.0（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
1	P01	R/W	0	随时读取该位以监控总线状态。
0	P00	R/W	0	~~0：设置 P0.0（UTX）始终为低电平*；~~ 1：P0.0（UTX）输出 UART 数据（要求）。
Else				请参考其它章节。

* 初始化 P00 时一定需要设置为高电平，因为 UART 模块只能驱动共用引脚的低电平信号。

19.10 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
   P00 = 1;
   P0M = P0M | 0x01& ~0x02;

   // 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;
  }
}
```

20 SPI

SPI 是一个串行通信接口，用于数据交换从一个 MCU 到另一个 MCU 或其他硬件外围设备。它是一个简单的 8 位接口，对协议、数据包或控制位没有主要定义。SPI 收发器包括三个引脚在主机与从机间发送数据：时钟线（SCK），数据输入和数据输出（MISO/ MOSI）。从动模式下由寄存器使能片选引脚 SSN。SPI 接口内置 4 种模式，时钟空闲状态和时钟脉冲状态。

- 全双工，3 线同步数据传输。
- 主控（SCK 为时钟输出）或 从动（SCK 为时钟输出）操作。
- 7 种 SPI 主控波特率。
- 从动时钟率高达 fcpu/8.
- 8 位数据传输(MSB 先，LSB 最后)。
- 串行时钟具有可编程极性和相位
- MCU 中断功能置主控模式故障错误标志。
- 写冲突标志保护。

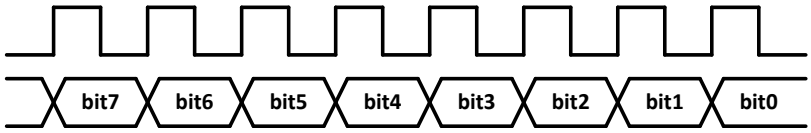
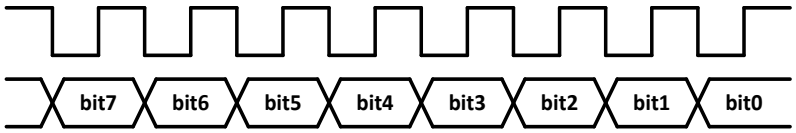
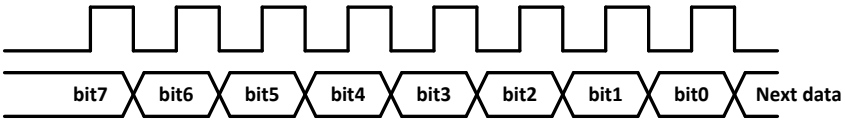
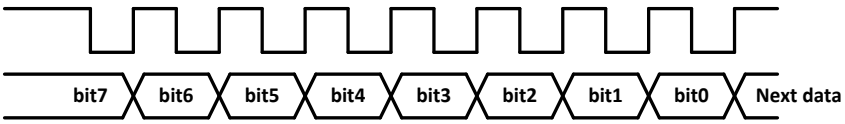
20.1 SPI操作

SPCON 寄存器控制 SPI 操作功能，比如：发送/接收，时钟率，数据传输方向， SPI 时钟空闲状态和时钟脉冲和使能这个电路。SPI 电路通过设置 SPCON 寄存器中的 SPEN 和写/读 SPDAT 寄存器来自动传输或接收 8 位数据。

CPOL 位用于控制 SPI 时钟空闲状态。CPHA 位用于控制数据接收的时钟沿方向。 CPOL 和 CPHA 位决定 SPI 的格式。SPI 数据传输方向是 MSB 位到 LSB 位。

SPI 支持由 CPOL 和 CPHA 位控制的 4 种模式格式。边沿方向是“数据传输沿”。当设置上升沿时，意味着在 SCK 上升的边缘接收和传输一个 BIT 数据，数据转换处于 SCK 的下降沿。当设置下降沿时，意味着在 SCK 下降的边缘接收和传输一个 BIT 数据，数据转换处于 SCK 的上升沿。

"CPHA"是时钟相位 BIT，控制采样数据的时钟相位。当 CPHA=1, SCK 的第一个边沿是数据转换，接收和传输数据在 SCK 第二边沿。当 CPHA=0，第 1 位数据已经固定了，SCK 的第一个边沿就接收和传输数据。 SPI 数据传输时序如下图：

CPOL	CPHA	Diagrams	Description
0	1		SCK 低电平空闲 MSB 为传输数据第一 BIT SCK 下降沿传输数据
1	1		SCK 高电平空闲 MSB 为传输数据第一 BIT SCK 上升沿传输数据
0	0		SCK 低电平空闲 MSB 为传输数据第一 BIT SCK 上升沿传输数据
1	0		SCK 高电平空闲 MSB 为传输数据第一 BIT SCK 下降沿传输数据

SPI 支持中断功能。ESPI 是 SPI 中断功能控制位。 ESPI=0，禁止 SPI 中断功能，ESPI=1，使能 SPI 中断功能。当 SPI 中断功能使能，SPI 操作后程序计数器指向 SPI 中断向量执行 SPI 中断例程。 SPIF 为 SPI 中断请求标志位，当 ESPI=0 时，也可以是 SPI 操作状态指示器，但是可通过读取 SPSTA，SPDAT 寄存器来清除。

SPI 在芯片选择功能中实现 SPI 多设备模式。在 SPI 总线中，一个主机与几个从属设备通信，而芯片选择决定指定的设备。芯片选择引脚是 SSN。

SPI 引脚也支持开漏结构。开漏选项由 PnOC 位控制。当 PnOC=1，使能 SPI 开漏结构。如果使能开漏结构，SPI 引脚必须设置为输入模式且需要外部上拉电阻。

20.2 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 P0.6 共用）输出，其 IDLE 状态由 CPOL 控制。

输入输出数据的相位由 CPHA 寄存器自动指定。主控模式下，MOSI 引脚（与 P0.5 共用）负责输出数据，MISO 引脚（与 P0.4 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。

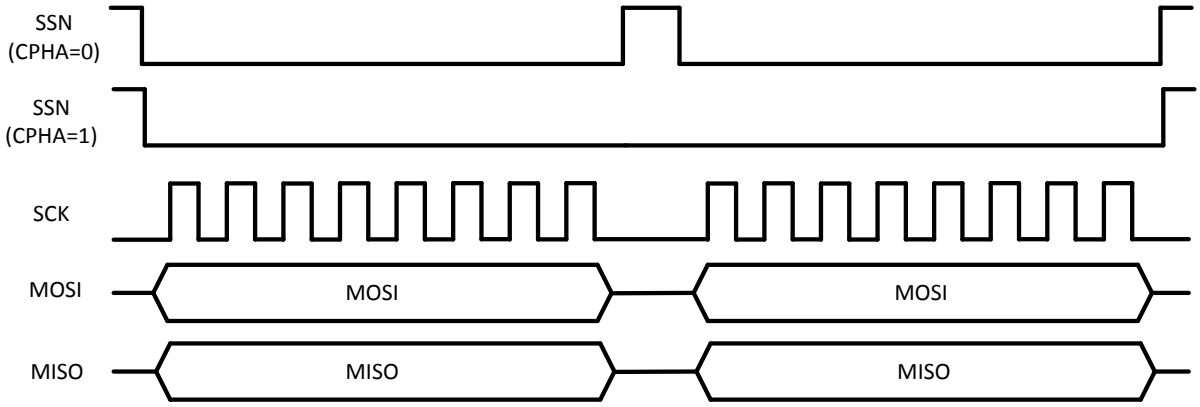
主控模式下，有 2 种包含中断功能的状态标志：

- 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 P0.3 共用）低电平状态释放 MODF，通过设置 SSDIS 位屏蔽中断源。

20.3 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。但当 CPHA= 0 时，SSN 必须严格遵循每个 8 位数据需要包含下降沿和上升沿，而 CPHA=1 则没有限制。



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。

20.4 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。

20.5 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	-	-	-	P06OC	P05OC	P04OC	P01OC	P00OC
P0M	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M

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。

*当 SPEN 位为 0 时，SPI 相关寄存器无法访问，模块内部时钟停止。

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	P06OC	R/W	0	0：切换 P0.6（SCK）为输入或输出模式； 1：切换 P0.6（SCK）为开漏模式。
3	P05OC	R/W	0	0：切换 P0.5（MOSI）为输入或输出模式； 1：切换 P0.5（MOSI）为开漏模式。
2	P04OC	R/W	0	0：切换 P0.4（MISO）为输入或输出模式； 1：切换 P0.4（MISO）为开漏模式。
Else				请参考其它章节。

P0M 寄存器（0Xf9）

Bit	Field	Type	Initial	说明
6	P06M	R/W	0	0：设置 P0.6（SCK）为输入模式[从动]； 1：设置 P0.6（SCK）为输出模式[主控]。
5	P05M	R/W	0	0：设置 P0.5（MOSI）为输入模式[从动]； 1：设置 P0.5（MOSI）为输出模式[主控]。
4	P04M	R/W	0	0：设置 P0.4（MISO）为输入模式[主控]； 1：设置 P0.4（MISO）为输出模式[从动]。
3	P03M	R/W	0	0：设置 P0.3(SSN) 为输入模式*； 1：设置 P0.3(SSN) 为输出模式*。
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。

*若从动模式下使能 SSN 功能：本就是要设置 SSN 为输入模式。

20.6 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 | = 0x0A;
 //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;
  }
}
```

21 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）。

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。

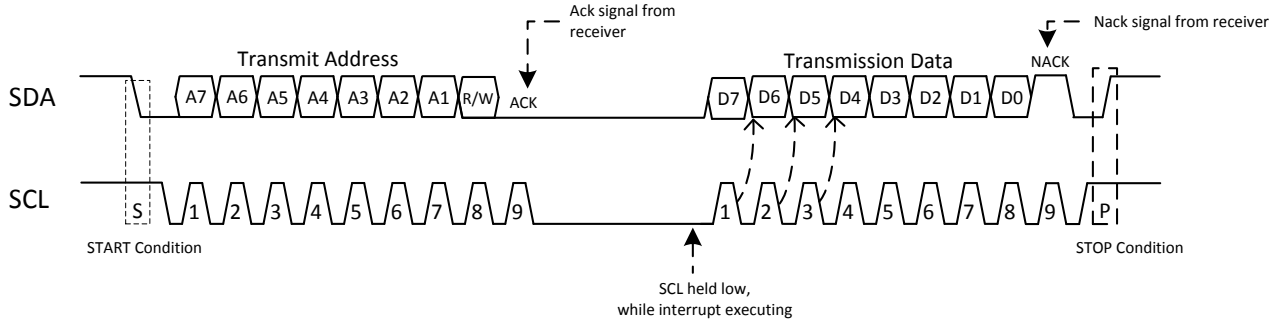
- 主 Tx, Rx 模式
- 从 Tx, Rx 模式（通用地址调用），以在单主机情况下进行 multiplex slave
- 双线同步数据收发器
- 支持 100K/400K 时钟率

21.1 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。

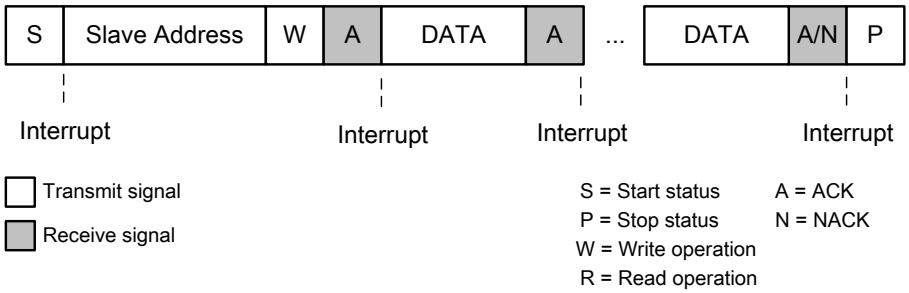


21.2 I2C传输模式

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。

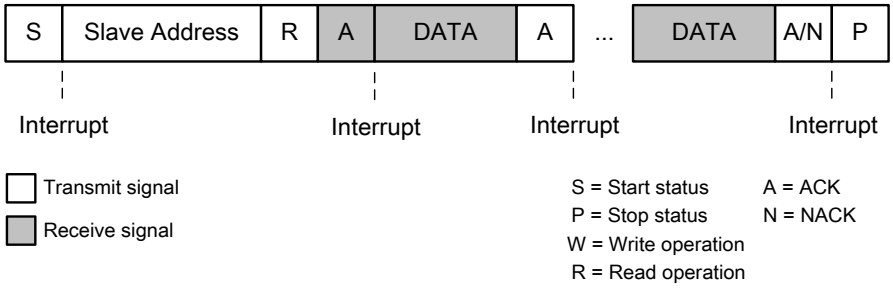
21.2.1 主机发送模式

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。



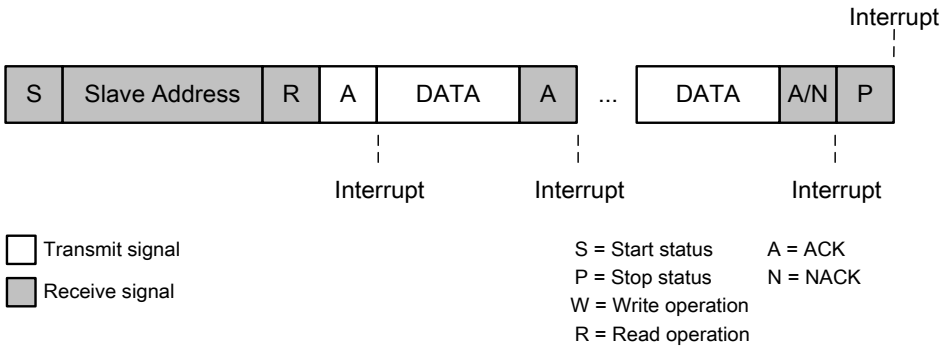
21.2.2 主机接收模式

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。



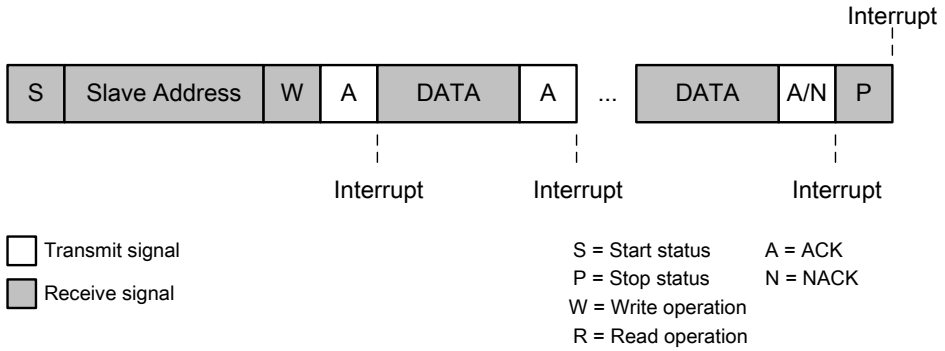
21.2.3 从机发送模式

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。



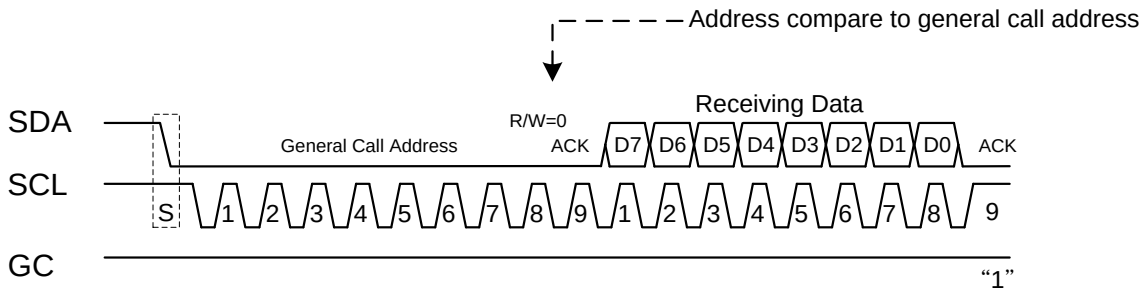
21.2.4 从机接收模式

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。



21.3 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。



21.4 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	16MHz
0	0	0	256	23	31
0	0	1	224	27	36
0	1	0	192	31	42
0	1	1	160	37	50
1	0	0	960	6.25	8
1	0	1	120	50	67
1	1	0	60	100	133
1	1	1	(Timer 1 overflow rate)/8		

✷ 注：

1. I2C 操作的第一步是设置 I2C 引脚的模式。在 SDA/SCL 引脚必须设置为“输入模式”。

2. 当时钟源是 T1 溢出率时，最大的计数器值为 0xFB(仅支持 0x00~0xFB)。此时，如果 T1 时钟率是 IHRC_32MHz，SCL 最大时钟率是 800kHz。

3. 若用户想产生 100KHz/400KHz 的 SCL 时钟速率，需提前设置 T1 计数器的值为 0xD8/0xF6。

21.5 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。

21.6 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=8MHz	
		十进制	十六进制
Tmext	5ms	39	27
Tsext	25ms	195	C3
Tout	35ms	273	111

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

21.7 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。

21.8 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

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。

* ENS1=0 时，禁止 I2C 相关寄存器，停止模块内部时钟。

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。
	58H	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	
主机接收器	60H	已接收到对应的 SLA+W ；	无动作	X	0	0	0/1	接收数据字节；返回 NACK/ACK

		已返回 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 时仲裁丢失；并接收到广播呼叫地址（00H）；已返回 ACK	无动作	X	0	0	0/1	接收数据字节；返回 NACK/ACK
	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	1	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 设为输入模式。

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 设为输入模式。

21.9 示例代码

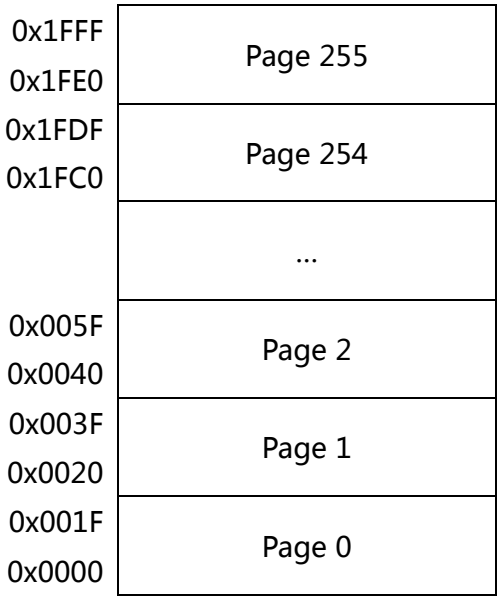
下面的示例代码程序演示了如何执行 I2C

```
Unsigned int I2Caddr;
Unsigned int I2C_TXData0;
Unsigned int I2C_TXDatan;
Unsigned int I2C_RXData0;
Unsigned int I2C_RXDatan;

void I2Cinit(void)
{
   P1M & = 0Xcf;         //P14 & P15 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
```

```
55          case 0x58:                          //read last byte & stop
56              I2C_RXData0 = I2CDAT;
57              I2CCON | = 0x10;                //STOP (STO)
58            break;
59       default:
60            I2CCON | = 0x10;                //STOP (STO)
61    }
62
63    I2CCON & = 0Xf7;                        //Clear I2C flag (SI)
64  }
```

22 在线编程

SN8F5703 内置 8KB 程序存储（IROM），均分为 256 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 Flash 存储器或者现场升级固件的一个通道。



程序存储器 IROM

22.1 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。

ISP ROM MAP		ROM 地址 bit0~bit4 (hex) =0
ROM address bit5~bit15 (hex)	0000	这些页包括复位向量和中断向量。强烈建议保留该区，不做 ISP 擦除动作。
	0020	
	0040	
	...	
	00C0	
	00E0	
	0100	一个 ISP 程序页
	0120	一个 ISP 程序页
	...	一个 ISP 程序页
	1000	一个 ISP 程序页
	1020	一个 ISP 程序页
	...	一个 ISP 程序页
	1700	一个 ISP 程序页

	1720	一个 ISP 程序页
	...	一个 ISP 程序页
	1FE0	该页包括 ROM 保留区。强烈建议保留该区，不做 ISP 擦除动作。

在开始页编程之前，必须将下面的内容设置完成。具体如下所示：

1、保存程序数据到 IRAM，数据继续 32 字节。

2、设置内容的开始地址到 PERAM。

3、设置计划升级区域的开始地址到 PEROM[15:5]。（PEROMH/PEROML 寄存器）

4、写入 0xA5A 到 PECMD[11:0]以触发 ISP 功能。写入 0x5A 到 PECMD[7:0]之前，必须先写入 0xA 到 PECMD[11:8]。

举例来说，假设程序存储器的第 254 页（IROM，1FC0H~1FDFH）为计划升级区域，已经存入 IRAM 地址 0x60~0x7F 的内容，执行在线编程，只要写入开始 IROM 地址 1FC0H 到 EPROMH/EPROML 寄存器，然后指定缓存器开始地址 0x60 到 EPRAM 寄存器，随后写入 0xA5A 到 PECMD[11:0]寄存器，复制缓存器的数据到 IROM 的第 254 页。

通常而言，每页的内容都可通过在线编程进行修改，但是第一页和最后一页（page0 和 page255）分别的存储复位向量和上电控制信息，都不可以编程，否则会导致上电错误或复位错误。

✷ **注：**

1. **在线编程操作前应该清看门狗，否则 ISP 操作过程中看门狗溢出将复位系统。**
2. **第一页和最后一页不要执行 ISP FLASH ROM，否则影响编程操作。**

22.2 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（15th– 8th 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（7th– 5th）。
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*。~~

*写入 0x5A 到 PECMD[7:0]之前，必须先写入 0xA 到 PECMD[11:8]。

*不允许写入其它值到 PECMD 寄存器。

22.3 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)
{
   ISP(pageAddress, 0Xe0);
}
```

23 电气特性

23.1 极限参数

VDD 至 VSS 的电压 ..... - 0.3V~6.0V
任意引脚至 VSS 的电压 ..... - 0.3V~VDD+0.3V
运行环境温度 ..... -40°C~85°C
存储环境温度 ..... -40°C~125°C
结温 ..... -40°C to 125°C

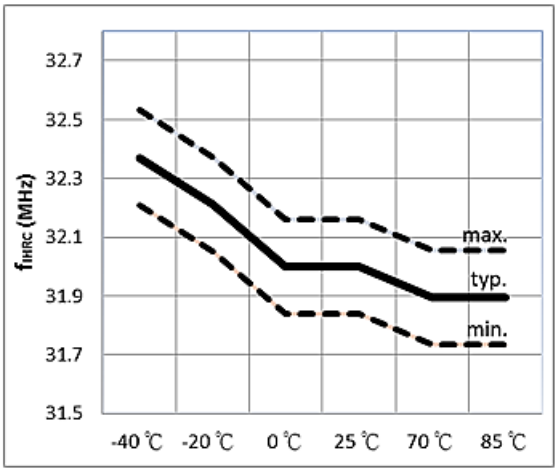
23.2 系统操作特性

参数		测试条件	Min	TYP	MAX	单位
VDD	工作电压	fcpu = 1MHz	1.8		5.5	V
V_{DR}	RAM 数据保持电压		1.5			V
V_{POR}	VDD 上升速率*		0.05			V/ms
I_{DD1}	正常模式供电电流 (CKCON=0x00,32MHz IHRC)	VDD = 3V, fcpu = 1MHz		2.19		mA
		VDD = 5V, fcpu = 1MHz		2.20		mA
		VDD = 3V, fcpu = 4MHz		2.65		mA
		VDD = 5V, fcpu = 4MHz		2.66		mA
		VDD = 3V, fcpu = 8MHz		3.26		mA
		VDD = 5V, fcpu = 8MHz		3.3		mA
	正常模式供电电流 (CKCON=0x00,16MHz IHRC)	VDD = 3V, fcpu = 1MHz		2.15		mA
		VDD = 5V, fcpu = 1MHz		2.74		mA
		VDD = 3V, fcpu = 4MHz		2.69		mA
		VDD = 5V, fcpu = 4MHz		3.28		mA
		VDD = 3V, fcpu = 8MHz		3.42		mA
		VDD = 5V, fcpu = 8MHz		3.98		mA
	正常模式供电电流 (CKCON=0x00,4MHz IHRC)	VDD = 3V, fcpu = 1MHz		1.86		mA
		VDD = 5V, fcpu = 1MHz		2.05		mA
		VDD = 3V, fcpu = 4MHz		2.40		mA
		VDD = 5V, fcpu = 4MHz		2.58		mA
I_{DD2}	STOP 模式供电电流	VDD = 3V		2.5	8.5	uA
		VDD = 5V		3.2	9.0	uA
I_{DD3}	IDLE 模式供电电流 (Fcpu=1MHz)	VDD = 3V, 32MHz IHRC		0.56		mA
		VDD = 5V, 32MHz IHRC		0.57		mA
		VDD = 3V, 16MHz Crystal		0.53		mA

		VDD = 5V, 16MHz Crystal		1.12		mA
		VDD = 3V, 4MHz Crystal		0.25		mA
		VDD = 5V, 4MHz Crystal		0.44		mA
F_{IHRC}	内部高时钟发生器	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	
		VDD = 1.8V to 5.5V, -40°C to 25°C	32.31	-	32.64	MHz
F_{ILRC}	内部低时钟发生器	VDD = 5.0V, 25°C	12	16	24	KHz
V_{LVD18}	LVD18 检测电压	25°C	1.7	1.8	1.9	V
		-40°C to 85°C	1.6	1.8	2.0	V

* 带*号的参数还未经过验证的设计参考值。环境温度为 25°C。

- **IHRC 频率–温度图**



23.3 GPIO特性

参数		测试条件	Min	TYP	MAX	单位
V_{IL}	低电平输入电压		VSS		0.3VDD	V
V_{IH}	高电平输入电压		0.7VDD		VDD	V
I_{LEKG}	I/O 口输入漏电流	V_{IN} = VDD			2	uA
R_{UP}	上拉电阻	VDD = 3V	100	200	300	KΩ
		VDD = 5V	50	100	150	KΩ
I_{OH}	I/O 输出 source 电流	VDD = 5V, V_O = VDD-0.5V	12	16		mA
I_{OL1}	I/O sink 电流 (P11 - P17 , P2)	VDD = 5V, V_O = VSS+0.5V	15	20		mA
I_{OL2}	I/O sink 电流 (P0, P10)	VDD = 5V, V_O = VSS+1.5V	80	100		mA

* 环境温度为 25°C.

23.4 ADC特性

参数		测试条件	Min	TYP	MAX	单位
V_{ADC}	工作电压		2.0		5.5	V
V_{AIN}	AIN 通道输入电压	VDD = 5V	0		V_{REFH}	V
V_{REFH}	AVREFH 引脚输入电压	VDD = 5V	2		VDD	V
V_{IREF}	内部 VDD 参考电压	VDD = 5V		VDD		V
	内部 4V 参考电压	VDD = 5V	3.92	4	4.08	V
	内部 3V 参考电压	VDD = 5V	2.94	3	3.06	V
	内部 2V 参考电压	VDD = 5V	1.96	2	2.04	V
I_{AD}	ADC 电流消耗	VDD = 3V		0.67		mA
		VDD = 5V		0.74		mA
f_{ADCLK}	ADC 时钟	VDD = 5V			32	MHz
f_{ADSMP}	ADC 时钟率	VDD = 5V			500	KHz
t_{ADEN}	ADC 功能使能周期	VDD = 5V	100			μs
DNL	微分非线性*	f_{ADSMP} = 62.5kHz		±1		LSB
		f_{ADSMP} = 250kHz		±1		LSB
		f_{ADSMP} = 500kHz		±3.5		LSB
INL	积分非线性*	f_{ADSMP} = 62.5kHz		±2		LSB
		f_{ADSMP} = 250kHz		±2		LSB
		f_{ADSMP} = 500kHz		±4		LSB
NMC	无遗漏码*	f_{ADSMP} = 62.5kHz	10	11	12	Bit
		f_{ADSMP} = 250kHz		10		Bit
		f_{ADSMP} = 500kHz		9		Bit
V_{OFFSET}	输入偏移电压	Non-trimmed	-10	0	10	mV

* 带*号的参数：VDD = 5V, V_{REFH} = 2.4V, 25°C.

23.5 OPA特性

参数		测试条件	Min	TYP	MAX	单位
V_{OPA}	工作电压		2.0		5.5	V
I_{OPA}	OPA 功耗	VDD = 3V		90		mA
		VDD = 5V		100		mA
V_{CM}	Common 模式输入范围	VDD = 5V	VSS		VDD	V
V_{OFFSET}	输入补偿电压	VDD = 5V	-15		15	mV
PSRR	电源抑制比*	V_{CM} = VSS	50		70	Db
CMRR	Common 模式抑制比*	V_{CM} = -0.3V to 5V, VDD = 5V	50			Db
A_{OL}	开环增益*	V_O = 0.2V to VDD-0.2V, V_{CM} = VSS	90			Db
V_{OS}	输出电压幅动	V_{OPP} = 2.5V	VSS+15		VDD-15	mV
I_{SC}	开短路功耗*(2)			4		mA
T_{OSR}	输出转换率	VDD = 5V, V_O rising		5		μs
		VDD = 5V, V_O falling		5		μs

*带*号的参数还未经过验证的设计参考值。

*(2) Unit Gain Buffer, Vi=Vdd~Vss, Vo=Vss~Vdd, Vdd=5V. (Vdd-0.5V and Vss+0.5V)

23.6 比较器特性

参数		测试条件	Min	TYP	MAX	单位
V_{CMP}	工作电压		2.0		5.5	V
I_{CMP}	功耗*	VDD = 5V		100		mA
V_{OFFSET}	输入补偿电压	VDD = 5V, V_{CM} = 0.5VDD	-15		15	mV
T_{RS}	响应时间	VDD = 5V, V_O rising		120		ns
		VDD = 5V, V_O falling		100		ns
T_{OSR}	输出转换率	VDD = 5V, V_O rising		100		ns
		VDD = 5V, V_O falling		100		ns
V_{IREF}	内部 4V 参考电压	VDD = 5V	3.92	4	4.08	V
	内部 3V 参考电压	VDD = 5V	2.94	3	3.06	V
	内部 2V 参考电压	VDD = 5V	1.96	2	2.04	V
V_{CMR}	共模输出电压	VDD = 5V	VSS+0.5		VDD-0.5	V

*带*号的参数还未经过验证的设计参考值。

23.7 Flash存储器特性

参数		测试条件	Min	TYP	MAX	单位
V_{dd}	供电电压		1.8		5.5	V
T_{en}	持续时间	25°C		*100K		Cycle
Iwrt	写电流	25°C		3	4	mA
T_{wrt}	写时间	写 1 页= 32 字节，25℃		6	8	ms

*带*号的参数还未经过验证的设计参考值。

24 指令集

本章对 SN8F5703 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

符号	说明
Rn	工作寄存器 R0 - R7
direct	128 个内部 RAM 地址之一或其它特殊功能寄存器
@Ri	寄存器 R0 或 R1 寻址的间接内部或外部 RAM 地址
#data	8 位常量（立即操作数）
#data16	16 位常量（立即操作数）
bit	内部 RAM 中的 128 个软件标志之一，或其它位-可寻址特殊功能寄存器的标志
addr16	LCALL 或 LJMP 的目标地址，可以是程序存储器地址空间的 64K 字节页面内的任一地址
addr11	ACALL 或 AJMP 的目标地址，程序存储器地址的同一个 2K 字节页面内，作为下一指令的第一个字节
rel	SJMP 和所有条件跳移，包括一个 8 位偏移字节。其范围是+127/-128 字节，相对于下一指令的第一个字节
A	累加器

运算指令

助记符	说明
ADD A, Rn	加寄存器到累加器
ADD A, direct	加直接寻址数据到累加器
ADD A, @Ri	加间接寻址数据到累加器
ADD A, #data	加立即数据到累加器
ADDC A, Rn	加寄存器到累加器，带进位
ADDC A, direct	加直接寻址数据到累加器，带进位
ADDC A, @Ri	加间接寻址数据到累加器，带进位
ADDC A, #data	加立即数据到累加器，带进位
SUBB A, Rn	从累加器减去寄存器，带借位
SUBB A, direct	从累加器减去直接寻址数据，带借位
SUBB A, @Ri	从累加器减去间接寻址数据，带借位
SUBB A, #data	从累加器减去立即数据，带借位
INC A	累加器递增
INC Rn	寄存器递增
INC direct	直接寻址地址递增
INC @Ri	间接寻址地址递增

INC DPTR	数据指针递增
DEC A	累加器递减
DEC Rn	寄存器递减
DEC direct	直接寻址地址递减
DEC @Ri	间接寻址地址递减
MUL AB	A 乘以 B
DIV	A 除以 B
DA A	十进制调整累加器

逻辑运算指令

助记符	说明
ANL A, Rn	寄存器与累加器
ANL A, direct	直接寻址数据与累加器
ANL A, @Ri	间接寻址数据与累加器
ANL A, #data	立即数据与累加器
ANL direct, A	累加器与直接寻址地址
ANL direct, #data	累加器与间接寻址地址
ORL A, Rn	寄存器或累加器
ORL A, direct	直接寻址数据或累加器
ORL A, @Ri	间接寻址数据或累加器
ORL A, #data	立即数据或累加器
ORL direct, A	累加器或直接寻址地址
ORL direct, #data	立即数据或直接寻址地址
XRL A, Rn	寄存器异或累加器
XRL A, direct	直接寻址数据异或累加器
XRL A, @Ri	间接寻址数据异或累加器
XRL A, #data	立即数据异或累加器
XRL direct, A	累加器异或间接寻址地址
XRL direct, #data	累加器异或直接寻址地址
CLR A	清零累加器
CPL A	累加器取反
RL A	循环左移累加器
RLC A	带进位符循环左移累加器
RR A	循环右移累加器
RRC A	带进位符循环右移累加器
SWAP A	累加器高低四位互换

数据传输指令

助记符	说明
MOV A, Rn	传送寄存器至累加器
MOV A, direct	传送直接寻址数据至累加器
MOV A, @Ri	传送间接寻址数据至累加器
MOV A, #data	传送立即数据至累加器
MOV Rn, A	传送累加器至寄存器
MOV Rn, direct	传送直接寻址数据至寄存器
MOV Rn, #data	传送立即数据至寄存器
MOV direct, A	传送累加器至直接寻址数据
MOV direct, Rn	传送寄存器至直接寻址数据
MOV direct1, direct2	传送直接寻址数据至直接寻址地址
MOV direct, @Ri	传送间接寻址数据至直接寻址地址
MOV direct, #data	传送立即数据至直接寻址地址
MOV @Ri, A	传送累加器至间接寻址地址
MOV @Ri, direct	传送直接寻址数据至间接寻址地址
MOV @Ri, #data	传送立即数据至间接寻址地址
MOV DPTR, #data16	加载带 16 位立即数据的数据指针
MOVC A, @A+DPTR	加载带相对于 DPTR 的代码字节的累加器
MOVC A, @A+PC	加载带相对于 PC 的代码字节的累加器
MOVX A, @Ri	传送外部 RAM（8 位地址）至累加器
MOVX A, @DPTR	传送外部 RAM（16 位地址）至累加器
MOVX @Ri, A	传送累加器至外部 RAM（8 位地址）
MOVX @DPTR, A	传送累加器至外部 RAM（16 位地址）
PUSH direct	直接寻址数据入栈
POP direct	直接寻址地址出栈
XCH A, Rn	寄存器与累加器交换
XCH A, direct	直接寻址地址与累加器交换
XCH A, @Ri	间接 RAM 与累加器交换
XCHD A, @Ri	间接地址操作数低半字节与累加器低半字节内容互换

布尔运算指令

助记符	说明
CLR C	清除进位标志
CLR bit	清除直接寻址位
SETB C	设置进位标志
SETB bit	设置直接寻址位
CPL C	标志位取反
CPL bit	直接寻址位取反
ANL C, bit	直接寻址位与进位标志

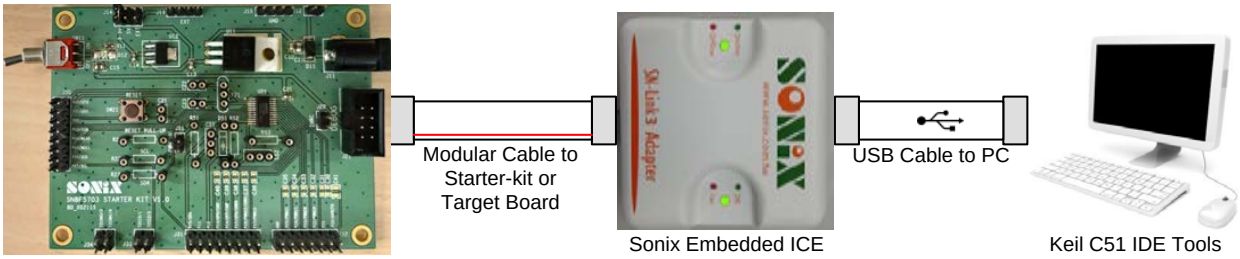
ANL C, /bit	直接寻址位取反与进位
ORL C, bit	直接寻址位或进位标志
ORL C, /bit	直接寻址位取反或进位
MOV C, bit	传送直接寻址位至进位标志
MOV bit, C	传送进位标志至直接寻址位

程序跳转指令

助记符	说明
ACALL addr11	绝对调用 addr11
LCALL addr16	长调用 addr16
RET	返回下一条指令处
RETI	返回自中断至下一条指令处
AJMP addr11	短跳转 addr11
LJMP addr16	长跳转 addr16
SJMP rel	相对跳转（相对地址）
JMP @A+DPTR	散转移
JZ rel	判断 0 转移
JNZ rel	判断非 0 转移
JC rel	判断进位标志为 1 时，跳转
JNC rel	判断进位标志为 0 时，跳转
JB bit, rel	判断直接寻址位为 1 时，跳转
JNB bit, rel	判断直接寻址位为 0 时，跳转
JBC bit, rel	判断直接寻址位为 1 时，跳转并定位清零
CJNE A, direct, rel	比较直接寻址数据和累加器，如果不相等，则跳转
CJNE A, #data, rel	比较立即数据和累加器，如果不相等，则跳转
CJNE Rn, #data, rel	比较直接寻址数据和寄存器，如果不相等，则跳转
CJNE @Ri, #data, rel	比较立即数据和间接寻址数据，如果不相等，则跳转
DJNZ Rn, rel	寄存器减一，不为 0 跳转
DJNZ direct, rel	直接寻址地址减一，不为 0 跳转
NOP	一个周期的空操作

25 开发环境

SONIX 提供嵌入式 ICE 仿真器系统，用于 SN8F5703 的 FW 开发。该平台是内电路调试器，并由 Microsoft Windows 平台上的 Keil C51 IDE 软件控制。该平台包括 SN-Link3、SN8F5703 starter-kit 和 Keil C51 IDE 软件，是高速、低成本、功能强大和多任务的开发环境，包括仿真器、调试器和程序员。执行仿真如同在真正的芯片上，因为仿真器电路集成在 SN8F5703 中，可提供真正的开发环境。



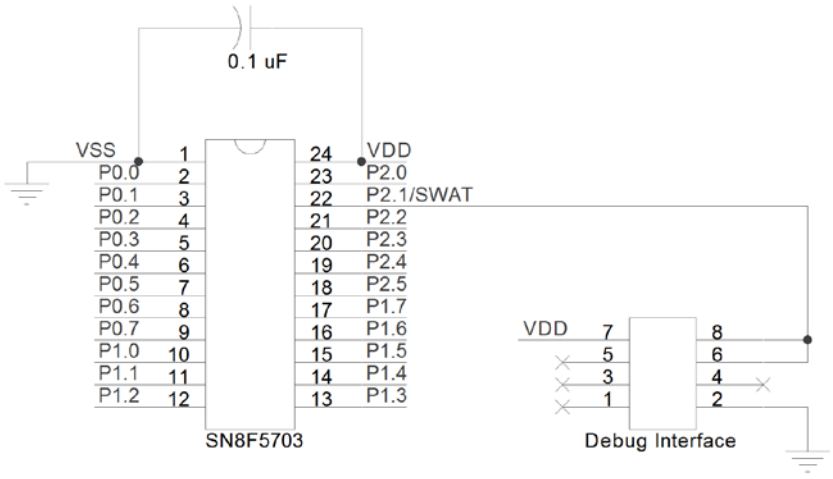
25.1 最低要求

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。

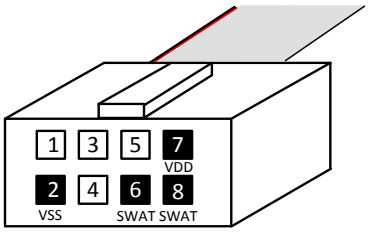
- **SN-Link 3 Adapter**：更新到 V1.02。
- **SN-Link Driver for Keil C51：**V1.00.317。
- **Keil C51**：V 9.50a 和 9.54a 或更高。

25.2 硬件调试界面

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 3。
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

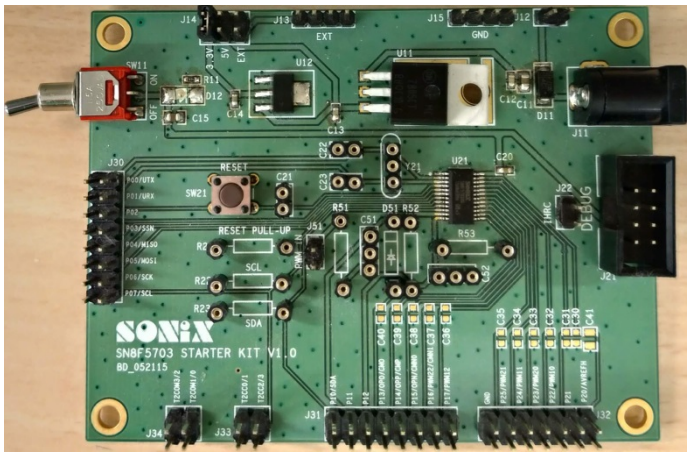


SN-Link header

25.3 开发工具

SN-Link3 适配器

Starter-Kit s 支持 SN8F5703, SN8F570320/321, SN8F570310/311



MP5 烧录器

26 SN8F5703 Start-Kit

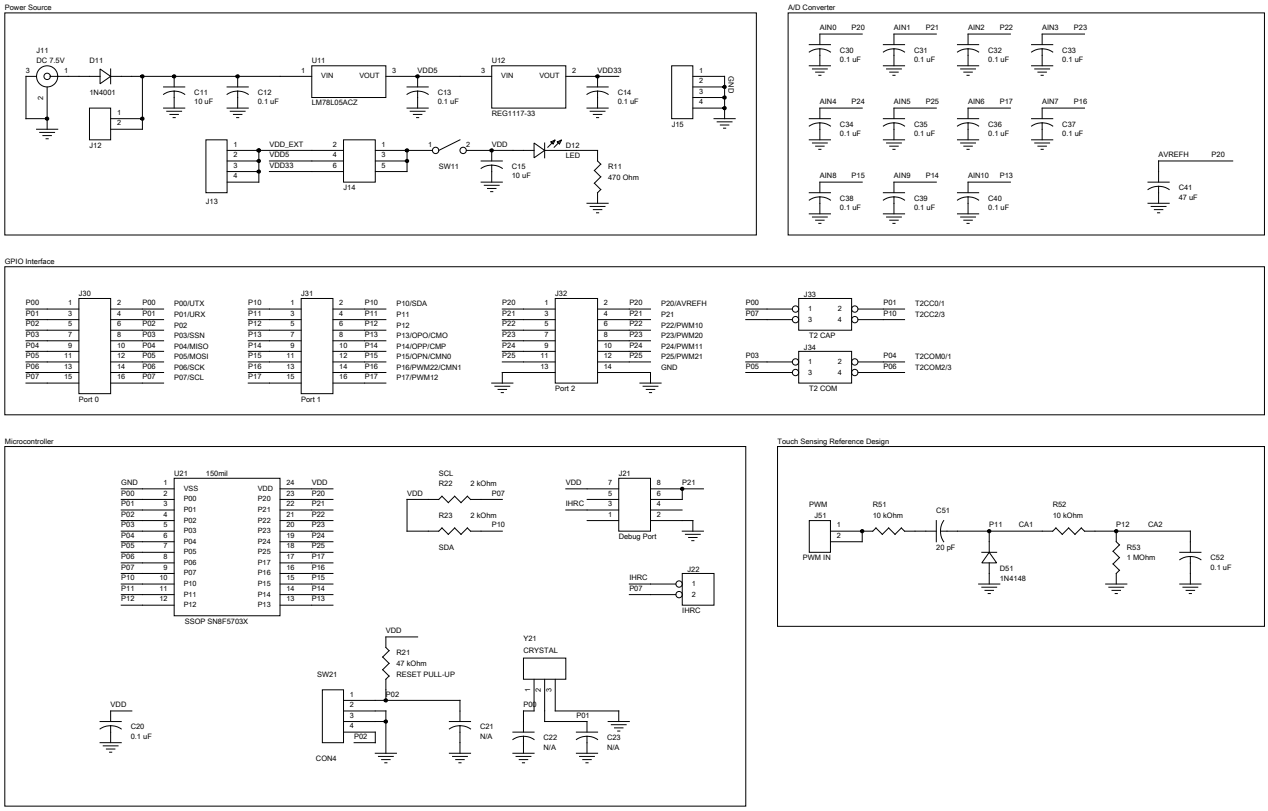
SN8F5000 Starter-Kit 提供一个简易的开发平台。它包含 SN8F5000 系列 IC 和输入信号或用户应用程序驱动设备的 IO 接口。当目标板没有准备好时，它是一个简单开发平台用于应用程序开发。Starer-Kit 可以被目标板所取代，因为 SN8F5000 系列集成了嵌入式的电路调试器电路。

26.1 配置电路

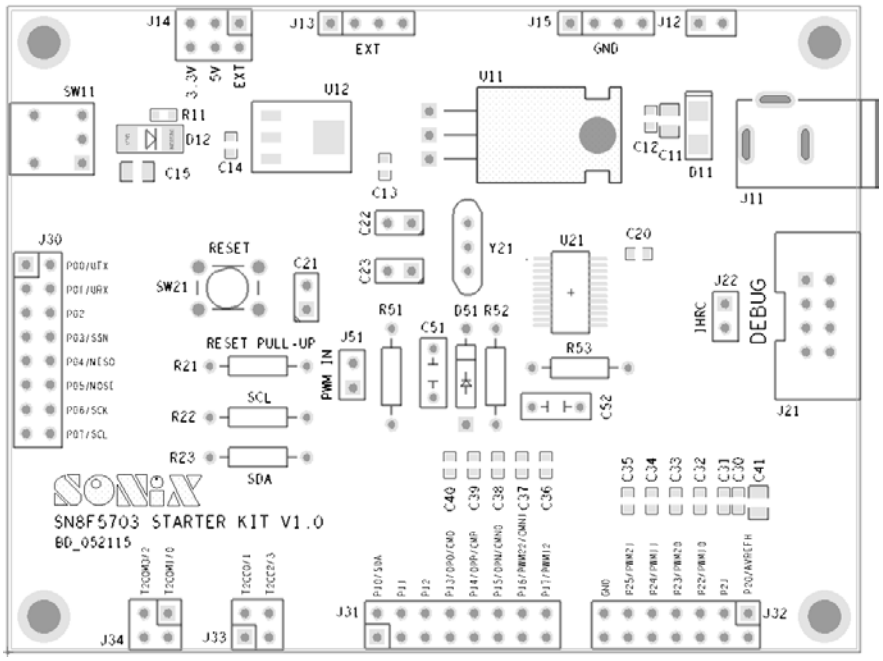
在开始利用 Start-Kit 开发前必须完成以下配置：

1. 确认电路板元件是否完整；
2. starkit 电路的电源是从 5.0 V，3.3 V，外部电源或 MINI USB 接口中选择的；
3. 如果电源来自 5.0 V 或 3.3 V，必须连接 7.5 V 直流电源适配器；
4. 如果电源来自外部电源，则外部电源连接到 EXT；
5. 当选择外部复位时，“RST”引脚需要将上拉电阻与 VDD 连接起来；
6. 当系统时钟设置晶振或 RTC 模式时，“XIN”和“XOUT”引脚需要接晶振；
7. 当系统时钟设置外部时钟输入模式时，“XIN”需要连接外部时钟源；
8. 调试端口可以连接 SN-LINK 适配器进行仿真或下载代码；
9. 当接通电源 VDD 时，MCU　LED 灯会亮起来，SN8F5000 系列 IC 将上电。

26.2 电路



26.3 PCB布局平面图



26.4 元件描述

符号	功能描述
C30-C40	11 通道 ADC 电容
C41	AVREFH 电容
D12	MCU LED
J11	DC 7.5V 电源适配器
J13/J15	外部电源
SW21	外部复位触发源
J14	VDD 电源选择为 5.0V，3.3V 或外部电源
J21	仿真接口
J30-J32	IO 口
J33	T2 捕获器电容
J34	T2 比较器电容
R21，C21	外部复位上拉电阻和电容
R22，R23	IIC 上拉电阻
SW11	目标电源开关
U21	SN8F5703X 实际芯片
Y21，C22，C23	外部晶振/谐振器部分

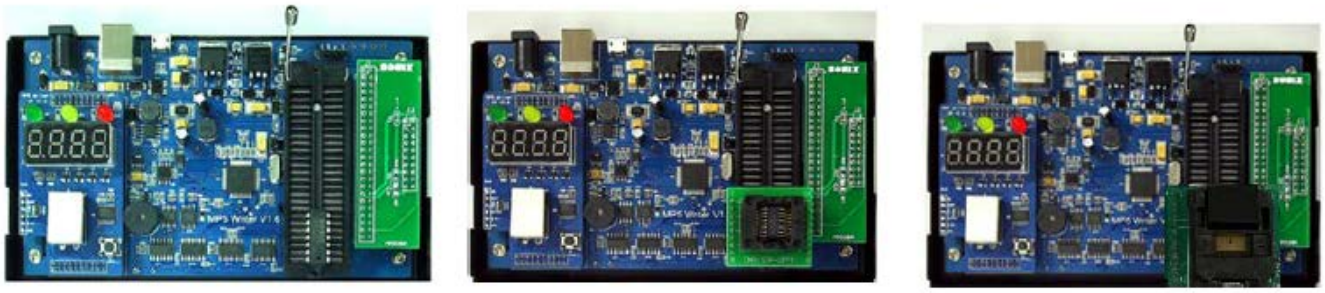
27 ROM 烧录引脚

SN-LINK 和 MP5 Writer 支持 SN8F5703 系列 Flash ROM 的擦除/烧录/校正。

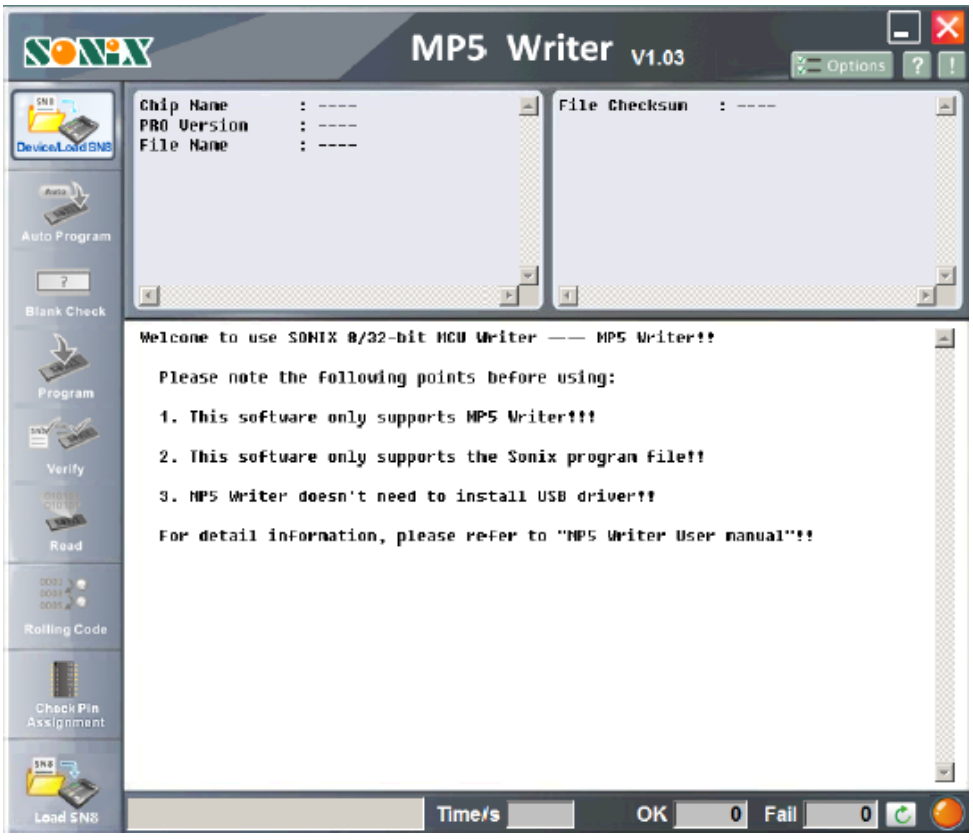
- **SN-LINK：调试界面。**
- **MP5 Writer：SN8F5703 系列。**

27.1 MP5 Writer转接板引脚配置

不同封装 MCU 烧录时的硬件接法如下，DIP，SOP，SSOP，TSSOP 和 QFN 说明。

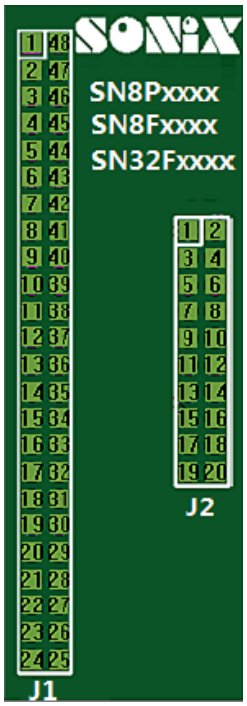


MP5 软件操作界面如下:



27.2 MP5 Writer转接板引脚配置

MP5 烧录器转接板：



27.3 MP5 Writer烧录引脚配置

烧录器连接器		MCU 引脚编号	SN8F5703S/X/T		SN8F5703J		SN8F570320P/S/T		SN8F570321J	
J2 引脚编号	J2 引脚名称		MCU 引脚编号	J1 引脚编号	MCU 引脚编号	J1 引脚编号	MCU 引脚编号	J1 引脚编号	MCU 引脚编号	J1 引脚编号
1	VDD	VDD	24	36	21	33	20	34	17	31
2	GND	VSS	1	13	22	34	1	15	18	32
7	SWAT	P2.1	22	34	19	31	18	32	15	29
9	SWAT	P2.1	22	34	19	31	18	32	15	29
20	PDB	P0.7	9	21	6	18	6	20	6	20

烧录器连接器		MCU 引脚编号	SN8F570310P/S		SN8F570311J					
J2 引脚编号	J2 引脚名称		MCU 引脚编号	J1 引脚编号	MCU 引脚编号	J1 引脚编号				
1	VDD	VDD	16	32	13	29				
2	GND	VSS	1	17	14	30				
7	SWAT	P2.1	14	30	12	28				
9	SWAT	P2.1	14	30	12	28				
20	PDB	P0.7	5	21	5	21				

27.4 SN-LINK ISP烧录

SN-Link ISP 烧录硬件和软件如下：

27.5 SN-LINK ISP烧录引脚图

SN-LINK 连接器		MCU 引脚编号	SN8F5703S/X/T	SN8F5703J	SN8F570320P/S/T	SN8F570321J
引脚编号	引脚名称		MCU 引脚编号	MCU 引脚编号	MCU 引脚编号	MCU 引脚编号
7	VDD	VDD	24	21	20	17
2	GND	VSS	1	22	1	18
6	SWAT	P2.1	22	19	18	15
8	SWAT	P2.1	22	19	18	15

SN-LINK 连接器		MCU 引脚编号	SN8F570310P/S	SN8F570311J		
引脚编号	引脚名称		MCU 引脚编号	MCU 引脚编号		
7	VDD	VDD	16	13		
2	GND	VSS	1	14		
6	SWAT	P2.1	14	12		
8	SWAT	P2.1	14	12		

28 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	5703	KG
8-bit MCU	Device Series	Package

Date Code

15	5	J	AEB11
Year	Mo.	Date	Internal Usage

28.1 命名规则

单片机全称	封装类型
SN8F5703W	Wafer
SN8F5703H	Dice
SN8F7503SG	SOP，24 脚，绿色封装
SN8F5703XG	SSOP，24 脚，绿色封装
SN8F5703TG	TSSOP，24 脚，绿色封装
SN8F5703JG	QFN，24 脚，绿色封装
SN8F570320PG	PDIP，20 脚，绿色封装
SN8F570320SG	SOP，20 脚，绿色封装
SN8F570320TG	TSSOP，20 脚，绿色封装
SN8F570321JG	QFN，20 脚，绿色封装
SN8F570310PG	PDIP，16 脚，绿色封装
SN8F570310SG	SOP，16 脚，绿色封装
SN8F570311JG	QFN，16 脚，绿色封装

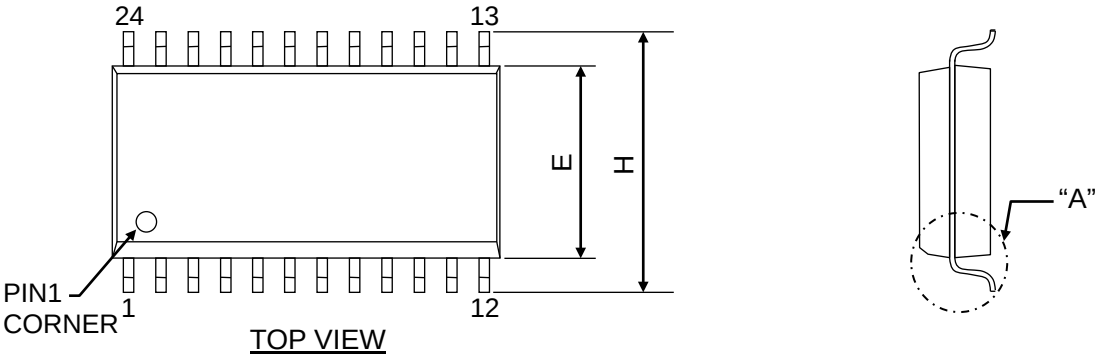
28.2 日期码

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

29 封裝信息

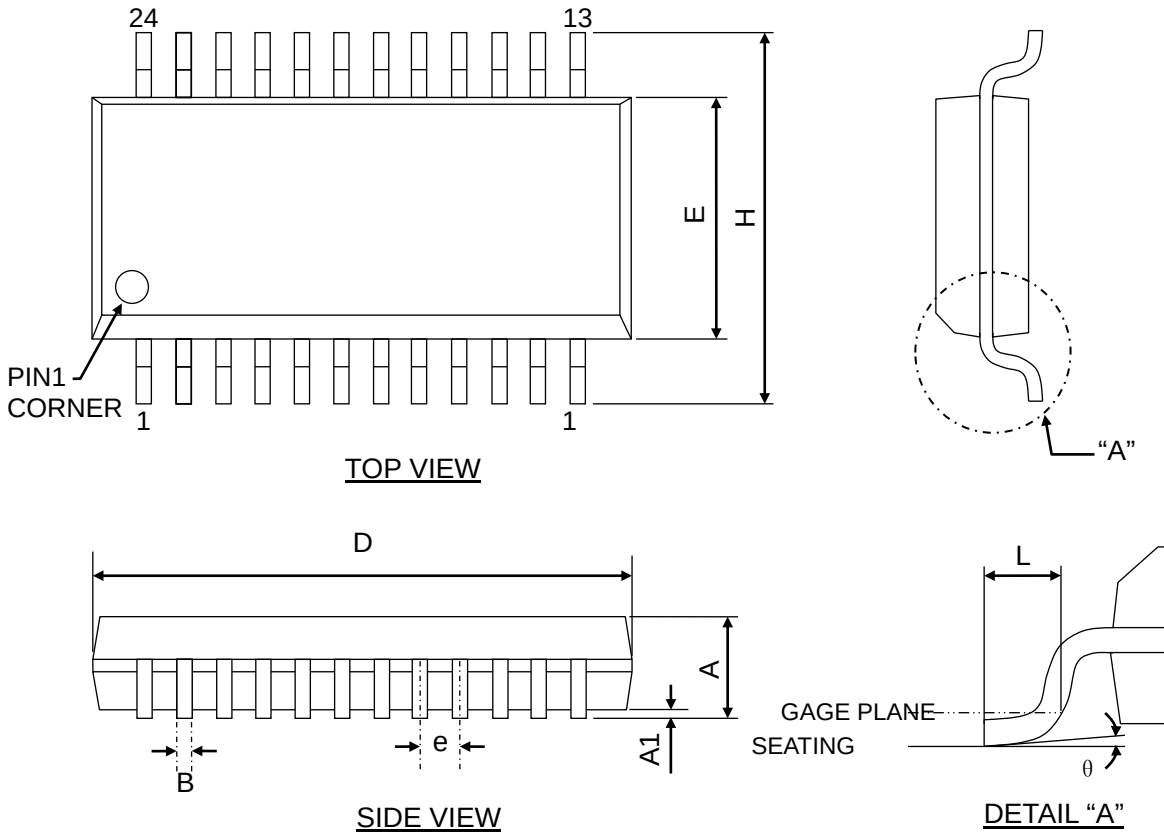
29.1 SOP24



SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	2.65	--	--	0.104
A1	0.10	--	0.30	0.004	--	0.011
B	0.31	0.41	0.51	0.012	0.016	0.020
D	15.30	15.50	15.70	0.602	0.618	0.618
E	7.50 BSC			0.295 BSC		
e	1.27 BSC			0.050 BSC		
H	10.30 BSC			0.405 BSC		
L	0.4	--	1.27	0.015	--	0.05
θ	0°	4°	8°	0°	4°	8°

Notes:
1. CONTROLLING DIMENSION: mm
2. JEDEC OUTLINE : MO-119 AA

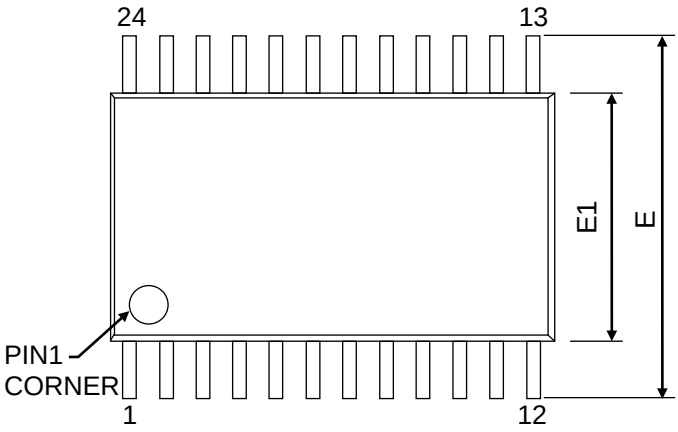
29.2 SSOP24



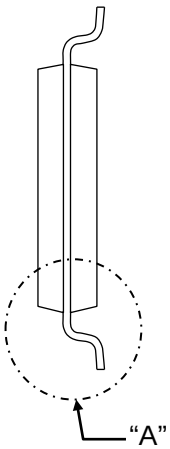
SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	1.75	--	--	0.069
A1	0.10	0.15	0.25	0.004	0.006	0.010
B	0.20	--	0.30	0.008	--	0.012
D	8.55	8.65	8.75	0.337	0.341	0.344
E	3.80	3.90	4.0	0.150	0.154	0.157
e	0.635 BSC.			0.025 BSC.		
H	5.80	6.00	6.20	0.228	0.236	0.244
L	0.41	0.64	1.27	0.016	0.025	0.050
θ	0°	--	8°	0°	--	8°

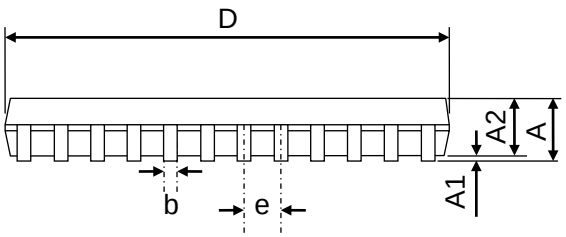
Notes:
1. CONTROLLING DIMENSION: INCH
2. JEDEC OUTLINE : MO-137 AE

29.3 TSSOP24

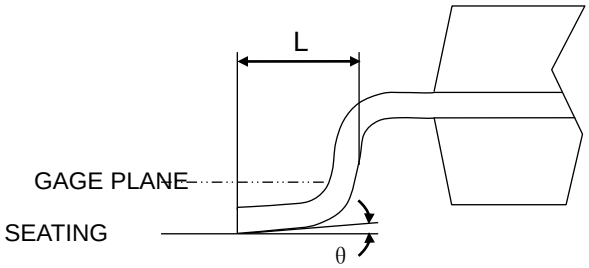


TOP VIEW





SIDE VIEW



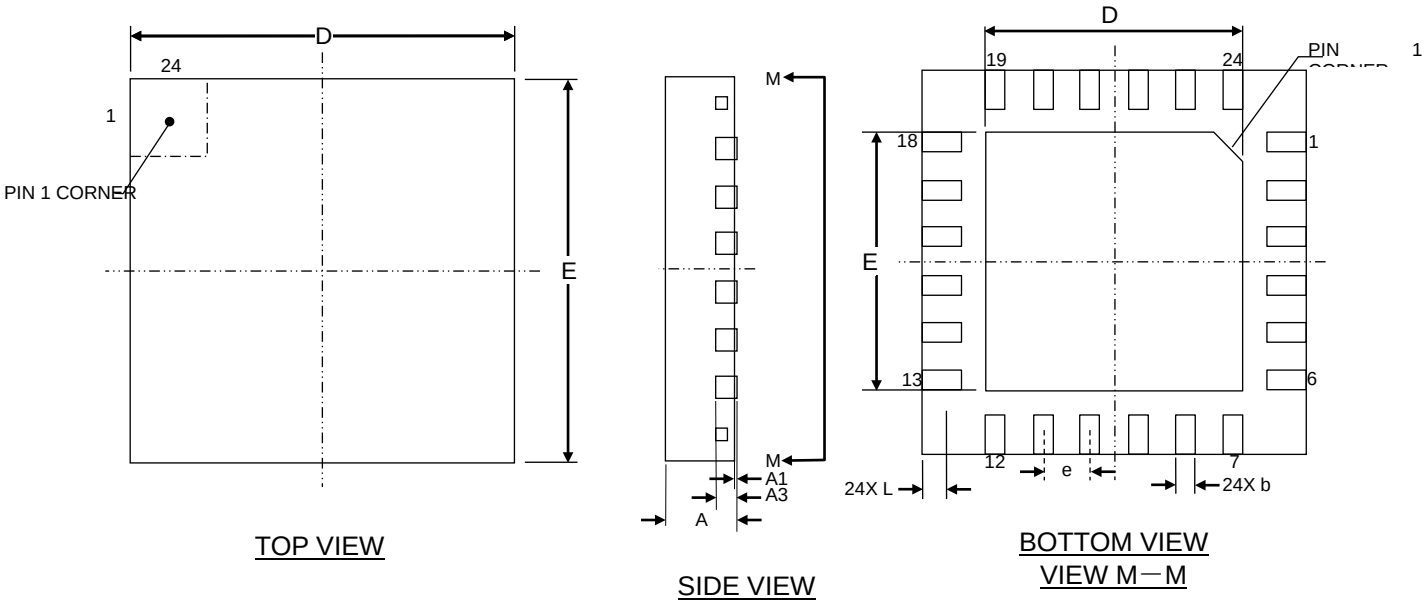
DETAIL "A"

SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	1.20	--	--	0.047
A1	0.00	--	0.15	0.000	--	0.006
A2	0.80	1.00	1.05	0.031	0.039	0.041
b	0.19	--	0.30	0.007	--	0.012
D	7.70	7.80	7.90	0.303	0.307	03.11
E	6.40 BSC.			0.252 BSC.		
E1	4.30	4.40	4.50	0.169	0.173	0.177
e	0.65 BSC.			0.026 BSC.		
L	0.45	0.60	0.75	0.018	0.024	0.030
θ	0°	--	8°	0°	--	8°

Notes:

1. CONTROLLING DIMENSION: mm
2. JEDEC OUTLINE : MO-153
3. DIMENSION 'D' DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BERRES.
4. DIMENSION 'E1' DOES NOT INCLUDE INTERLEAD FLASH OR PROTRUSION.
5. DIMENSION 'b' DOES NOT INCLUDE DAMBAR PROTRUSION.

29.4 QFN24 4X4



TOP VIEW

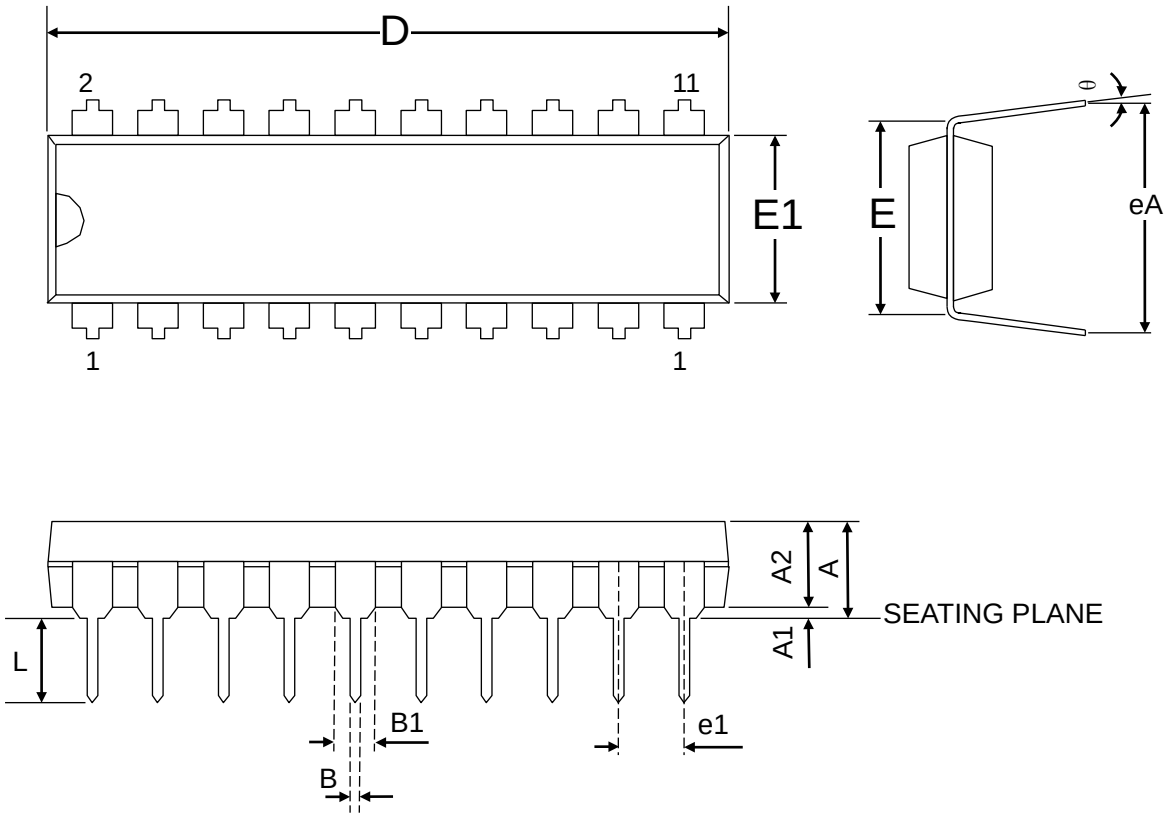
SIDE VIEW

BOTTOM VIEW
VIEW M－M

SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	0.70	0.80	0.90	0.028	0.031	0.035
A1	0.00	0.02	0.05	0.000	0.001	0.002
A3	0.203 REF			0.008 REF		
b	0.15	0.25	0.30	0.007	0.010	0.012
D	4.00 BSC			0.157 BSC		
E	4.00 BSC			0.157 BSC		
e	0.50 BSC			0.020 BSC		
D2	1.90	2.35	2.80	0.075	0.093	0.110
E2	1.90	2.35	2.80	0.075	0.093	0.110
L	0.30	0.40	0.50	0.012	0.016	0.020

Notes：
1. CONTROLLING DIMENSION：MILLIMETER (mm)

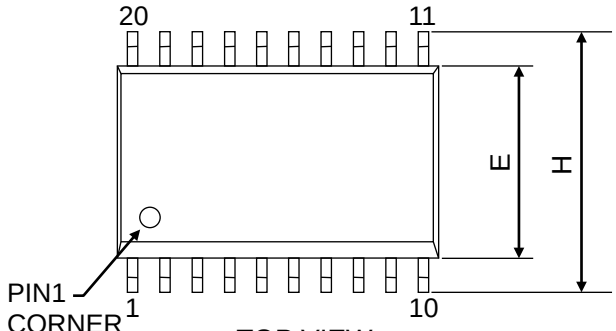
29.5 DIP20



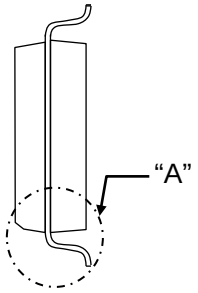
SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	4.45	--	--	0.175
A1	0.35	--	--	0.015	--	--
A2	3.18	3.30	3.43	0.125	0.130	0.135
B	0.46 typ.			0.018 typ.		
B1	1.52 typ.			0.060 typ.		
D	25.70	26.06	26.42	1.012	1.026	1.040
E	7.62 BSC.			0.300 BSC.		
E1	6.05	6.35	6.65	0.238	0.250	0.261
e1	2.54 typ.			0.100 typ.		
L	3.05	3.30	3.56	0.120	0.130	0.140
eA	7.62	9.02	9.53	0.300	0.355	0.375
θ	0˚	7˚	15˚	0˚	7˚	15˚

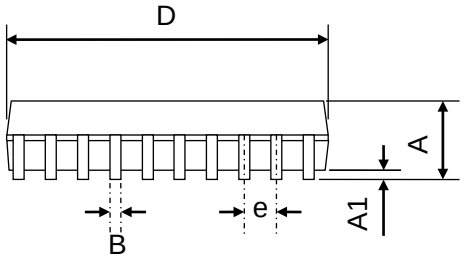
Notes：
1. JEDEC OUTLINE : MS-001 AD
2. CONTROLLING DIMENSION：inch

29.6 SOP20

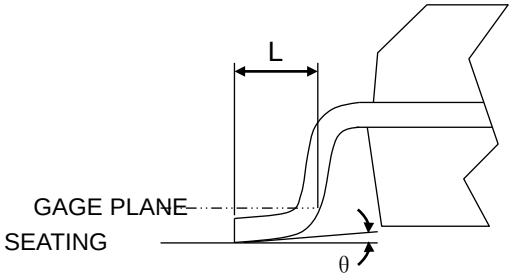


TOP VIEW





SIDE VIEW

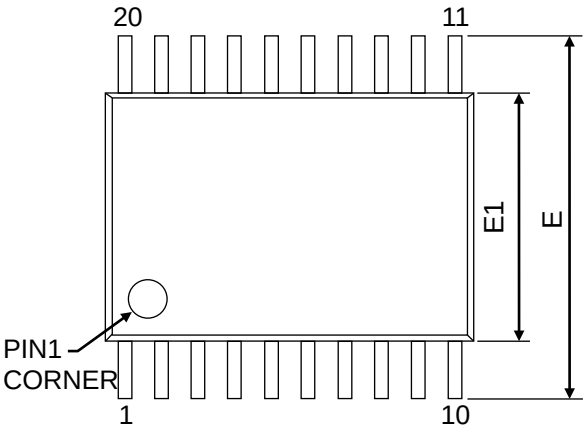


DETAIL "A"

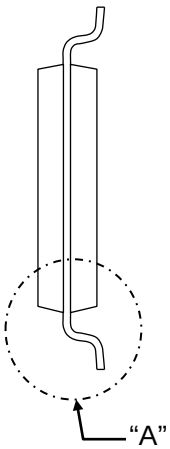
SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	2.65	--	--	0.104
A1	0.10	--	0.30	0.004	--	0.012
B	0.31	0.41	0.51	0.012	0.016	0.020
D	12.80 BSC			0.503		
E	7.50 BSC			0.295		
e	1.27 BSC			0.050 BSC		
H	10.30 BSC			0.405		
L	0.40	--	1.27	0.016	--	0.050
θ	0°	4°	8°	0°	4°	8°

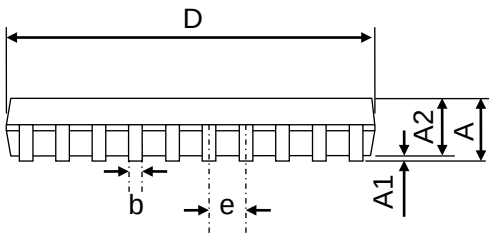
Notes:
1. CONTROLLING DIMENSION: mm
2. JEDEC OUTLINE : MO-013 AC

29.7 TSSOP20

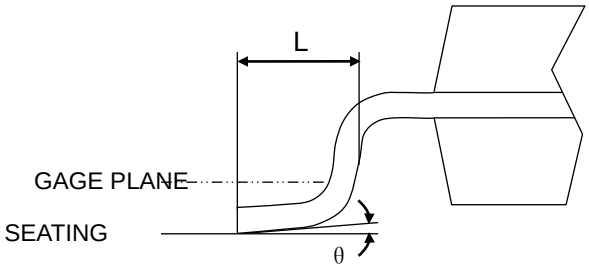


TOP VIEW





SIDE VIEW

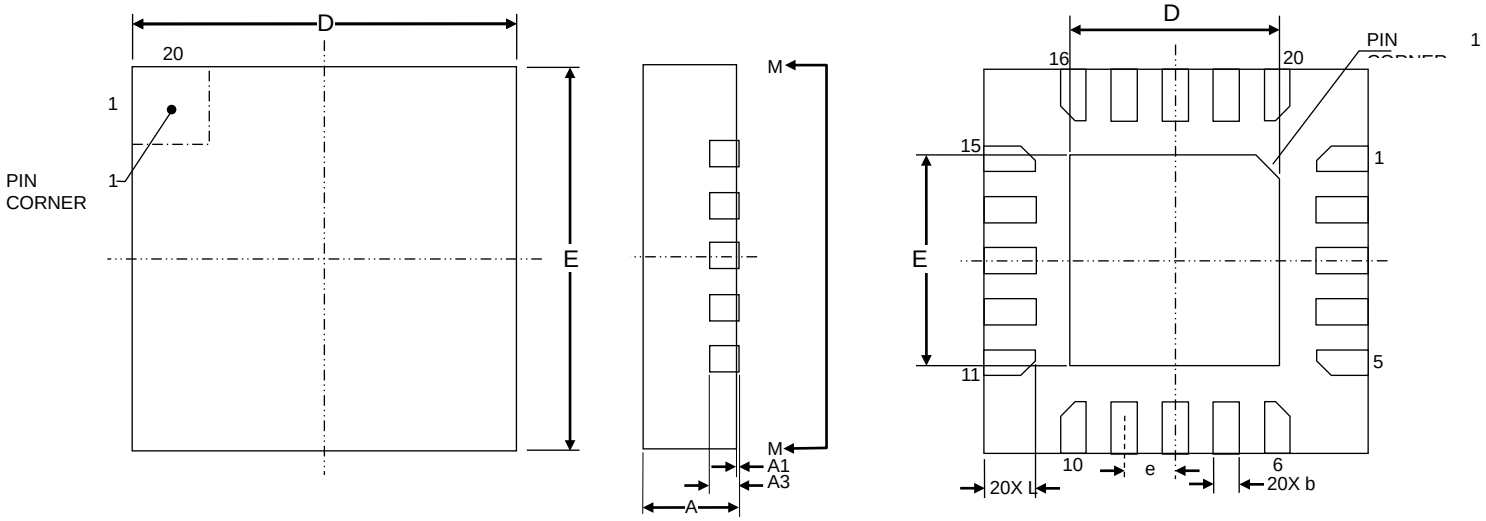


DETAIL "A"

SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	1.20	--	--	0.047
A1	0.05	--	0.15	0.002	--	0.006
A2	0.80	--	1.05	0.031	--	0.041
b	0.19	--	0.30	0.007	--	0.012
D	6.40	6.50	6.60	0.252	0.256	0.260
E	6.40 BSC.			0.252 BSC.		
E1	4.30	4.40	4.50	0.169	0.173	0.177
e	0.65 BSC.			0.026 BSC.		
L	0.45	0.60	0.75	0.018	0.024	0.030
θ	0°	--	8°	0°	--	8°

Notes:
1. CONTROLLING DIMENSION: mm
2. JEDEC OUTLINE : MO-153
3. DIMENSION 'D' DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BERRES.
4. DIMENSION 'E1' DOES NOT INCLUDE INTERLEAD FLASH OR PROTRUSION.
5. DIMENSION 'b' DOES NOT INCLUDE DAMBAR PROTRUSION.

29.8 QFN20 3X3

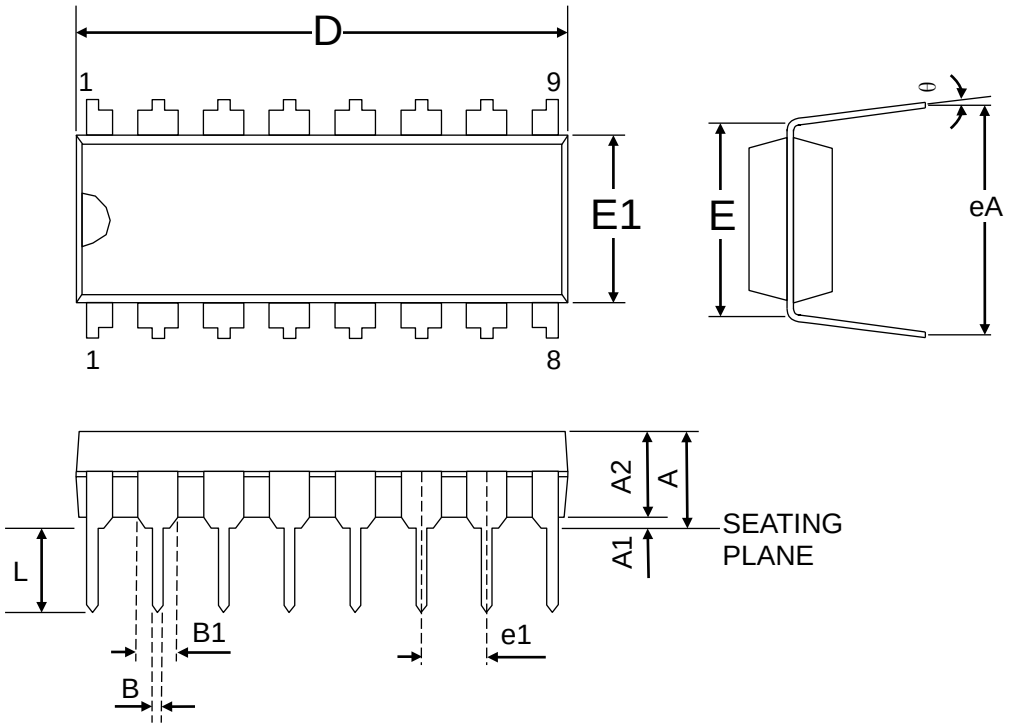


SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	0.70	0.80	0.90	0.028	0.031	0.035
A1	0.00	0.02	0.05	0.000	0.001	0.002
A3	0.203 REF			0.008 REF		
b	0.15	0.20	0.25	0.006	0.008	0.010
D	3.00 BSC			0.118 BSC		
E	3.00 BSC			0.118 BSC		
e	0.40 BSC			0.016 BSC		
D2	1.55	1.65	1.75	0.61	0.65	0.69
E2	1.55	1.65	1.75	0.61	0.65	0.69
L	0.30	0.40	0.50	0.012	0.016	0.020

Notes：

1. CONTROLLING DIMENSION：MILLIMETER (mm)

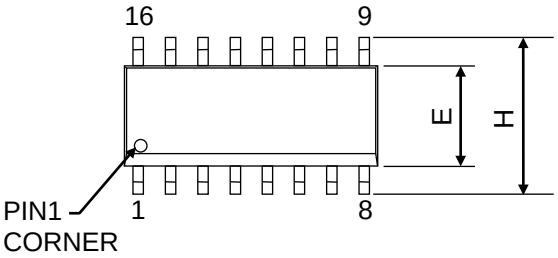
29.9 DIP16



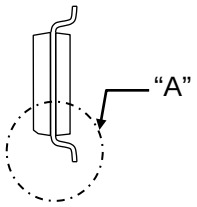
SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	5.33	--	--	0.210
A1	0.38	--	--	0.015	--	--
A2	3.18	3.30	3.43	0.125	0.130	0.135
B		0.46 typ.			0.018 typ.	
B1		1.52 typ.			0.060 typ.	
D	18.67	19.18	19.69	0.735	0.755	0.775
E		7.62 BSC.			0.300 BSC	
E1	6.22	6.35	6.48	0.245	0.250	0.255
e1		2.54 typ.			0.100 typ.	
L	2.92	3.30	3.81	0.115	0.130	0.150
eA	7.62	9.02	9.53	0.300	0.355	0.375
θ	0˚	7˚	15˚	0˚	7˚	15˚

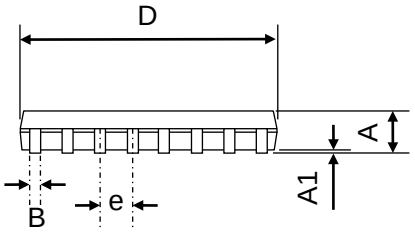
Notes：
1. JEDEC OUTLINE : MS-001 BB
2. CONTROLLING DIMENSION：inch

29.10 SOP16

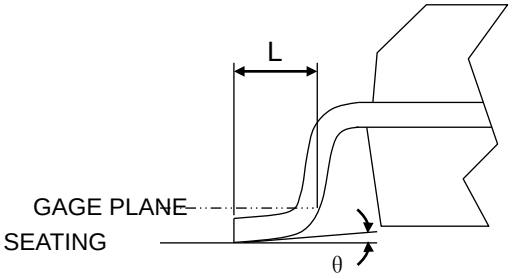


TOP VIEW





SIDE VIEW

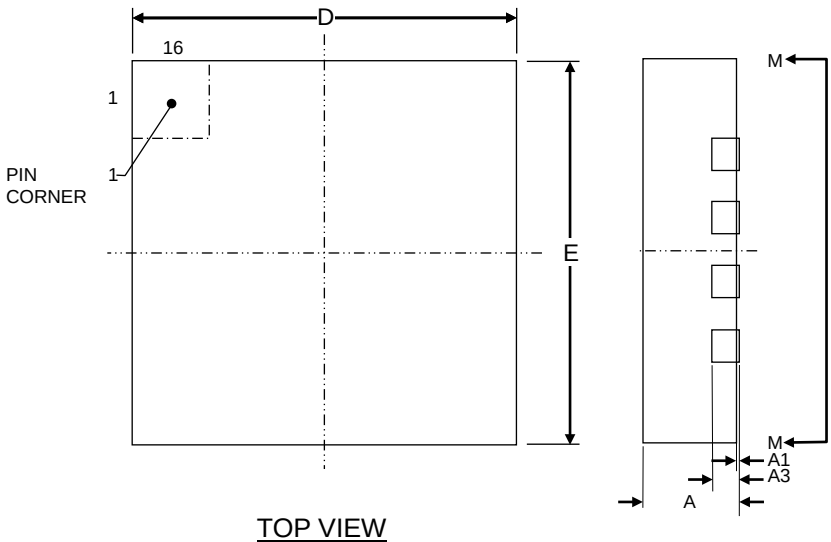


DETAIL "A"

SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	1.75	--	--	0.069
A1	0.10	--	0.25	0.004	--	0.010
B	0.31	0.41	0.51	0.012	0.016	0.020
D	9.90 BSC			0.389 BSC		
E	3.90 BSC			0.153 BSC		
e	1.27 BSC			0.050 BSC		
H	6.00 BSC			0.236 BSC		
L	0.40	--	1.27	0.016	--	0.050
θ	0°	4°	8°	0°	4°	8°

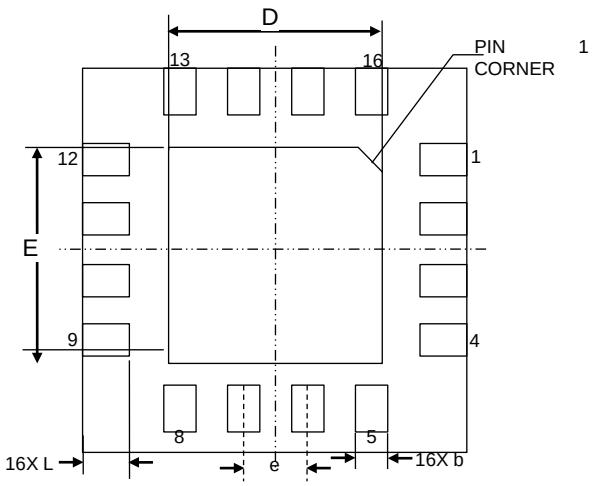
Notes:
1. CONTROLLING DIMENSION: mm
2. JEDEC OUTLINE : MS-012 AC

29.11 QFN16 3X3



TOP VIEW

SIDE VIEW



BOTTOM VIEW
VIEW M—M

SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	0.70	0.80	0.90	0.028	0.031	0.035
A1	0.00	0.02	0.05	0.000	0.001	0.002
A3	0.20 REF			0.008 REF		
b	0.18	0.25	0.30	0.007	0.010	0.012
D	3.00 BSC			0.118 BSC		
E	3.00 BSC			0.118 BSC		
e	0.50 BSC			0.020 BSC		
D2	1.40	1.60	1.80	0.055	0.063	0.070
E2	1.40	1.60	1.80	0.055	0.063	0.070
L	0.25	0.35	0.45	0.010	0.014	0.018

Notes：

1. CONTROLLING DIMENSION：MILLIMETER (mm)

30 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。

SN8F5703 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