SONiX Technology Co., Ltd.

SN8F5701 Series Datasheet

8051-based Microcontroller

SN8F5701PG
SN8F5701SG
SN8F5701TG
SN8F5701MG
SN8F57011DG

1 产品简介

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 可编程 CPU 频率
  - 内部 32MHz 时钟发生器（IHRC）
- 4 KB Flash 存储器（IROM），支持 In-System Program 功能
- 256 字节内部 RAM（IRAM）
- 8 个中断源：可控制中断的优先等级以及独立的中断向量
  - 5 个内部中断
  - 3 个外部中断：INT0/INT1/INT2
- 1 组 DPTR
- 2 组 8/16 位定时器，包括 4 种工作模式
- 1 组 16 位带 PWM 发生器的定时器
  - 每组 PWM 都有 6 个输出通道，带独立的占空比，反相器和频率控制功能
- 6+1 通道 12 位 SAR ADC
  - 6 个外部 ADC 输入
  - 1 个内部电池检测
  - 4 个内部参考电压（VDD/4V/3V/2V）
- UART 接口
- 工作模式
  - NORMAL 模式
  - IDLE 模式
  - STOP 模式
- 片上调试
  - 单线调试接口
  - 2 个硬件断点
  - 无限制软件断点
  - ROM 数据安全保护
  - 看门狗和可编程的外部复位
- 1.8 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

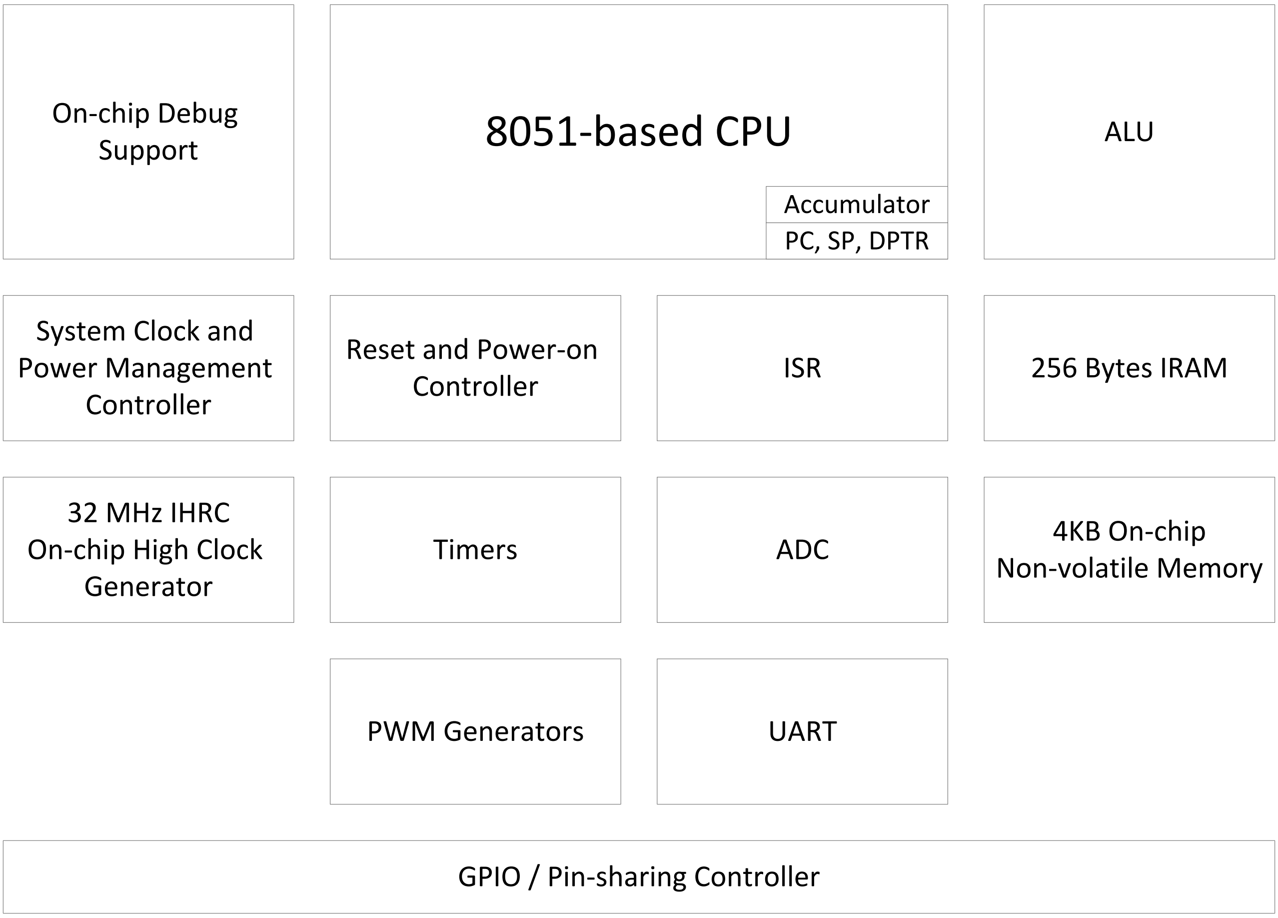
1.2 应用领域

- 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
SN8F5701PG/SG/TG/MG	6	6	-	-	V	6	-	-	3	DIP8,SOP8,TSSOP8, DFN8L
SN8F57011DG	4	4	-	-	V	4	-	-	2	SOT23-6L

1.4 框图


On-chip Debug Support
8051-based CPU
Accumulator
PC, SP, DPTR
ALU
System Clock and Power Management Controller
Reset and Power-on Controller
ISR
256 Bytes IRAM
32 MHz IHRC On-chip High Clock Generator
Timers
ADC
4KB On-chip Non-volatile Memory
PWM Generators
UART
GPIO / Pin-sharing Controller

2 目录

3 修改记录

版本	时间	修改说明
1.0	May. 2017	初版。
1.1	Jul.2017	修改 ADC VREFH 说明和电气特性章节。
1.2	Oct. 2017	1.修改订购信息； 2.增加封装信息。
1.3	Nov. 2017	1. 移除 LVD 2.4V/3.3V 低电压描述特征； 2. 更新电气特性章节。
1.4	Mar. 2019	1. 修改错误,遗漏等； 2. MP5 烧录引脚映射增加 NORMAL 模式和高速模式； 3. 修改引脚电路图部分； 4. 修改 ADC 输入补偿范围； 5. 修改上电顺序和系统时钟； 6. 修改封装信息部分； 7. 修改 T0/T1 部分。
1.5	Nov.2019	1. 修改 ADC 寄存器描述； 2. 增加 DFN8L 封装。
1.6	Apr.2024	1. 增加第 4 章引脚配置注意事项； 2. 增加 ISP 操作注意事项； 3. 修改电气特性部分； 4. 修改特性表中内存描述； 5. 修改书写错误；
1.7	Sep.2024	1. 修改订购信息。

上述事宜无关。

4 引脚配置

4.1 SN8F5701PG/SG/TG/MG (DIP8/SOP8/TSSOP8/DFN8L)

VDD	1	U	8	VSS
PWM0/INT0/AIN0/P00	2		7	P05/AIN5/PWM5
PWM1/INT1/AIN1/P01	3		6	P04/AIN4/PWM4/SWAT
URX/PWM2/INT2/RST/AIN2/P02	4		5	P03/AIN3/PWM3/UTX

4.2 SN8F57011DG (SOT23-6L)

SWAT/PWM4/AIN4/P04	1	U	6	VDD
VSS	2		5	P00/AIN0/INT0/PWM0
UTX/PWM3/AIN3/P03	3		4	P02/AIN2/RST/INT2/PWM2/URX

> ✸ *注：低功耗模式下，用户需要正确设置MCU中未封出的引脚，可以降低功耗。强烈的建议将这些引脚设置为输入上拉。*

4.3 引脚说明

电源引脚

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

P0 端口

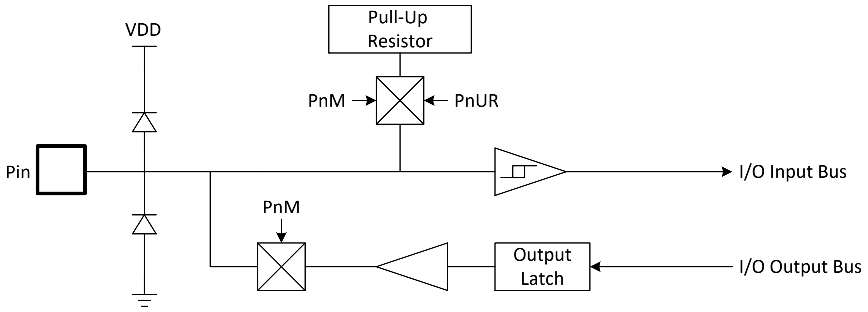
引脚名称	类型	功能说明
P0.0 AIN0 PWM0 INT0	数字 I/O 模拟输入 数字输出 数字输入	GPIO ADC：输入通道 0 PWM：可编程 PWM 输出 INT0：外部中断 0
P0.1 AIN1 PWM1 INT1	数字 I/O 模拟输入 数字输出 数字输入	GPIO ADC：输入通道 1 PWM：可编程 PWM 输出 INT1：外部中断 1
P0.2 AIN2 Reset PWM2 INT2 URX	数字 I/O 模拟输入 数字输入 数字输出 数字输入 数字输入	GPIO ADC：输入通道 2 系统复位（低电平有效） PWM：可编程 PWM 输出 INT2：外部中断 2 UART：数据输入引脚
P0.3 AIN3 PWM2 UTX	数字 I/O 模拟输入 数字输出 数字输出	GPIO ADC：输入通道 3 PWM：可编程 PWM 输出 UART：数据输出引脚
P0.4 AIN4 SWAT PWM4	数字 I/O 模拟输入 数字 I/O 数字输出	GPIO ADC：输入通道 4 单线仿真接口 PWM：可编程的 PWM 输出
P0.5 AIN5 PWM5	数字 I/O 模拟输入 数字输出	GPIO ADC：输入通道 5 PWM：可编程的 PWM 输出

4.4 引脚特性

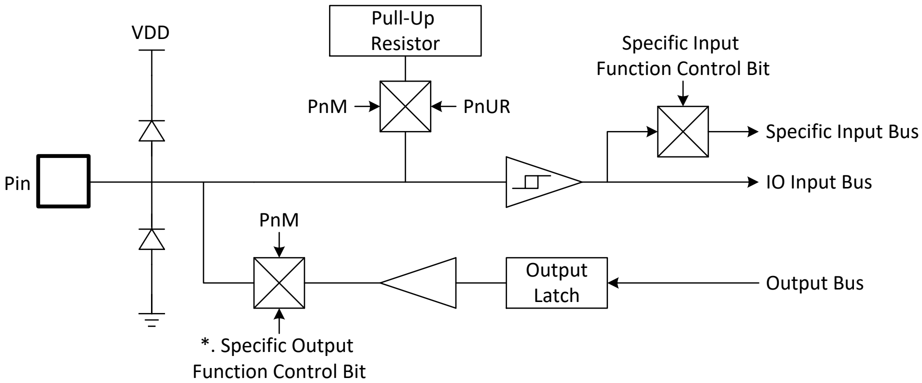
端口	开漏	外部中断	唤醒 （电平变化）	共用引脚
P0.0	V	V	V	AIN0/ PWM0/ INT0
P0.1	V	V	V	AIN1/ PWM1/ INT1
P0.2	V	V	V	AIN2/ PWM2/ INT2/ URX/ RST
P0.3	V	-	V	AIN3/ PWM3/ UTX
P0.4	V	-	V	AIN4/ PWM4/ SWAT
P0.5	V	-	V	AIN5/ PWM5

4.5 引脚电路结构图

正常双向 **I/O** 引脚：

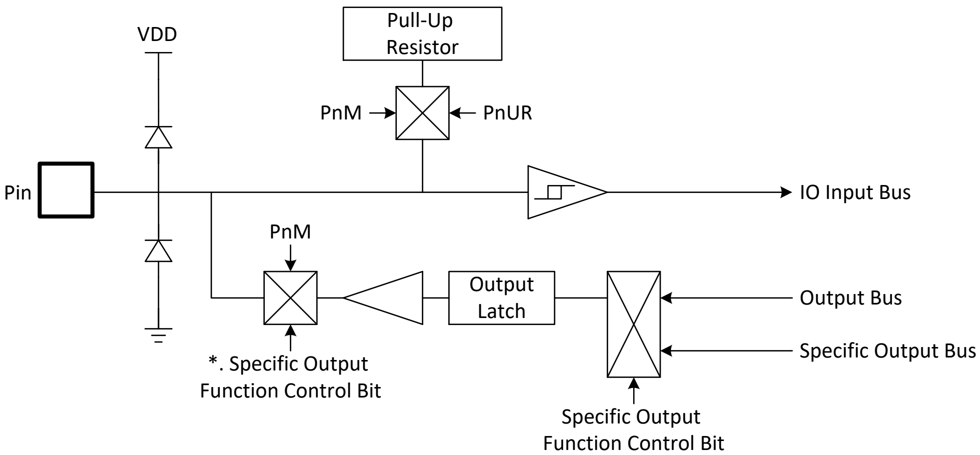


双向 **I/O** 引脚，与特殊数字输入功能引脚共用，如 **INT0**，**UART**：



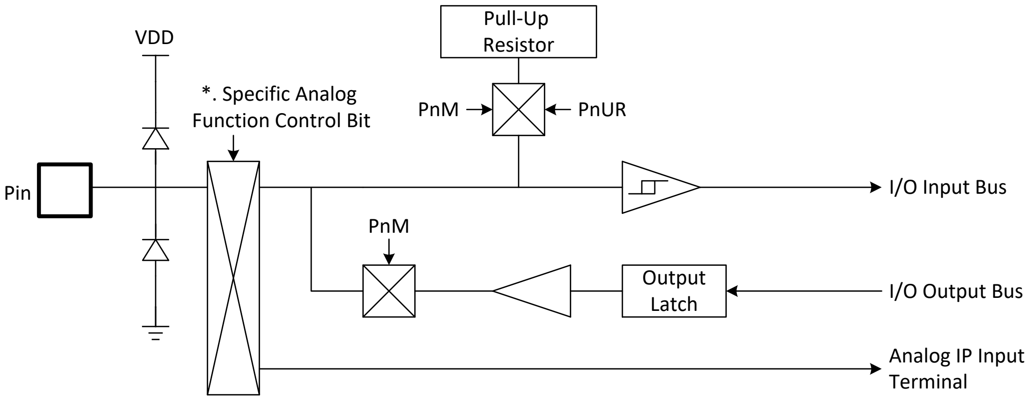
***. Some specific functions switch I/O direction directly, not through PnM register.**

双向 **I/O** 引脚，与特殊数字输出功能引脚共用，如 **T3**，**UART**：



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

双向 **I/O** 引脚，与特殊模拟输入功能引脚共用，如 **ADC**：



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

5 CPU

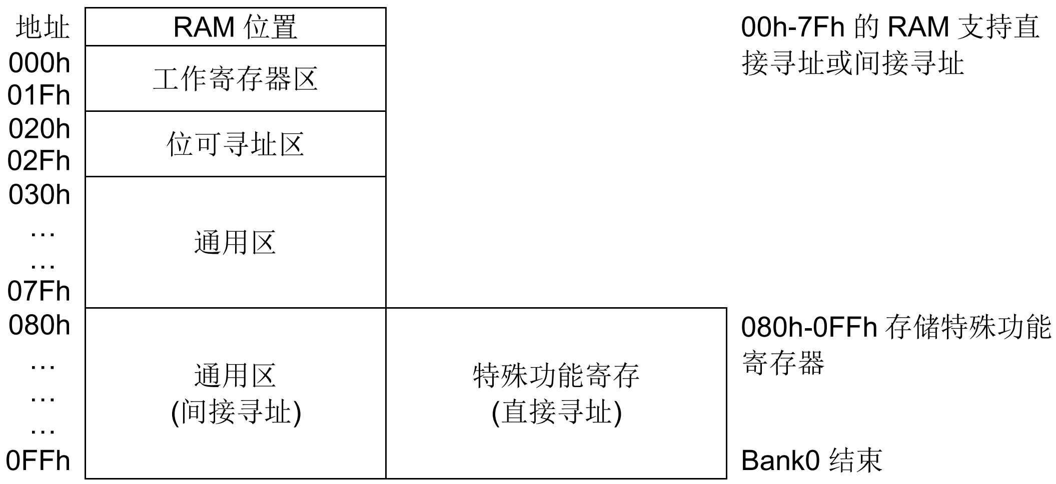
SN8F5000 系列是一颗增强型的 8051 微控制器（MCU），且完全兼容 MCS-51 指令集，因此可以使用目前流行的编译环境仿真（例如 Keil C51）。总的来说，在相同的频率下，SN8F5000 的 CPU 要比原始的 8051 快 9.4 到 12.1 倍。

5.1 存储器结构

SN8F5701 内建了两个存储器：内部 RAM（IRAM）和程序存储器（IROM）。内部 RAM 由 256 个字节组成，具有较高的存取性能（支持直接寻址和间接寻址）。程序存储器是一个 4KB 的 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)



内部数据存储器的 256 字节 RAM 是一个标准的 8051 RAM 访问配置。虽说直接寻址指令去访问内部 RAM 所花的周期比间接寻址要少，但是间接寻址可以访问内部 RAM 的所有区域。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（如 MOV A, direct）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，且以 0x0 或 0x8 结尾的特殊功能寄存器支持位寻址；

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，这 4 组工作寄存器可通过汇编指令（如 MOV A, R0）进行寻址；

- 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 堆栈

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

0x0B			0x0B			0x0B	
0x0A		PUSH →	0x0A		LCALL →	0x0A	地址
0x09		POP ←	0x09		RET ←	0x09	
0x08	(空)		0x08	数据		0x08	数据
SP = 0x07			SP = 0x08			SP = 0x0A	

默认情况下，堆栈指针（SP 寄存器）指向 0x07，就是指堆栈的区域从 IRAM 地址中的 0x08 开始。换句话说，如果计划想将堆栈区域设置为从 IRAM 的 0xC0 开始，就要在系统复位后将 SP 寄存器设置为 0xBF。

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。***

5.4 程序存储器（IROM）

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

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	
0083h	INT2	
008Bh	ADC	
00EBh	T3	
00ECh . . .	通用区域	用户程序 用户程序结束
0FF6h 0FF7h 0FFEh 0FFFh	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；
- 0x0003-0x00EB：多中断向量区域，每个中断事件都有一个唯一的中断向量；
- 0x00EC-0x0FDF：用户程序和 ISP 通用区域；
- **0x0FE0-0x0FF5：用户通用区域程序，禁止执行 ISP；**
- **0x1FF6-0x0FFF：保留区域，禁止执行 ISP；**

5.5 程序存储器安全

SN8F5701 提供的加密功能，旨在防止对 Flash 存储器中保存的信息进行非授权的访问。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 MOVC 指令时，数据指针可帮助指定 IROM 地址。该单片机有 1 组数据指针（DPH/DPL）。

5.7 堆栈和数据指针寄存器

寄存器	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

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。

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	-	-	-	-	-	PFLAG
F0	B	P0UR	FRQL	FRQH	FRQCMD	-	-	SRST
E8	-	-	-	-	-	-	-	-
E0	ACC	-	-	-	P0OC	CLKSEL	CLKCMD	TCON0
D8	S0CON2	-	-	-	-	-	-	-
D0	PSW	-	ADM	ADB	ADR	VREFH	-	-
C8	-	-	-	-	-	-	-	-
C0	-	-	-	-	-	-	-	-
B8	-	IP1	S0RELH	-	PWNV	PWO	PWCH	IRCON2
B0	-	PW3DL	PW3DH	PW4DL	PW4DH	PW5DL	PW5DH	-
A8	IEN0	IP0	S0RELL	PW1DL	PW1DH	PW2DL	PW2DH	-
A0	-	T3M	T3CL	T3CH	T3YL	T3YH	PW0DL	PW0DH
98	S0CON	S0BUF	IEN2	-	-	-	P0CON	-
90	-	-	-	-	PECMD	PEROML	PEROMH	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

0x80-0x9F 寄存器说明

寄存器	地址	说明
P0	0x80	P0 数据缓存器。
SP	0x81	堆栈指针寄存器。
DPL	0x82	数据指针 0 低字节寄存器。
DPH	0x83	数据指针 0 高字节寄存器。
-	0x84	-
-	0x85	-
WDTR	0x86	看门狗定时器清零寄存器。
PCON	0x87	系统模式寄存器。
TCON	0x88	T0/1 控制寄存器。
TMOD	0x89	T0/1 模式寄存器。
TL0	0x8A	T0 计数低字节寄存器。
TL1	0x8B	T1 计数低字节寄存器。
TH0	0x8C	T0 计数高字节寄存器。
TH1	0x8D	T1 计数高字节寄存器。
CKCON	0x8E	扩展周期寄存器。
PEDGE	0x8F	外部中断边沿控制寄存器。
-	0x90	-
-	0x91	-
-	0x92	-
-	0x93	-
PECMD	0x94	In-System Program 命令寄存器。
PEROML	0x95	In-System Program ROM 地址低字节。
PEROMH	0x96	In-System Program ROM 地址高字节。
PERAM	0x97	In-System Program RAM 分配地址。
S0CON	0x98	UART 控制寄存器。
S0BUF	0x99	UART 数据缓存器。
IEN2	0x9A	中断使能寄存器。
-	0x9B	-
-	0x9C	-
-	0x9D	-
P0CON	0x9E	P0 配置控制寄存器。
-	0x9F	-

0xA0-0xBF 寄存器说明

寄存器	地址	说明
-	0xA0	-
T3M	0xA1	T3 控制寄存器。
T3CL	0xA2	T3 计数寄存器低字节。
T3CH	0xA3	T3 计数寄存器高字节。
TC3YL	0xA4	T3 周期控制缓存器低字节。
TC3YH	0xA5	T3 周期控制缓存器高字节。
PW0DL	0xA6	PW0 占空比控制寄存器低字节。
PW0DH	0xA7	PW0 占空比控制寄存器高字节。
IEN0	0xA8	中断使能寄存器。
IP0	0xA9	中断优先权寄存器。
S0RELL	0xAA	UART 重装低字节寄存器。
PW1DL	0xAB	PW1 占空比控制寄存器低字节。
PW1DH	0xAC	PW1 占空比控制寄存器高字节。
PW2DL	0xAD	PW2 占空比控制寄存器低字节。
PW2DH	0xAE	PW2 占空比控制寄存器高字节。
-	0xAF	-
-	0xB0	-
PW3DL	0xB1	PW3 占空比控制寄存器低字节。
PW3DH	0xB2	PW3 占空比控制寄存器高字节。
PW4DL	0xB3	PW4 占空比控制寄存器低字节。
PW4DH	0xB4	PW4 占空比控制寄存器高字节。
PW5DL	0xB5	PW5 占空比控制寄存器低字节。
PW5DH	0xB6	PW5 占空比控制寄存器高字节。
-	0xB7	-
-	0xB8	-
IP1	0xB9	中断优先权寄存器。
S0RELH	0xBA	UART 重装高字节寄存器。
-	0xBB	-
PWNV	0xBC	PWM 反相控制寄存器。
PWO	0xBD	PWM 频率控制寄存器。
PWCH	0xBE	PWM 通道控制缓存器。
IRCON2	0xBF	中断请求寄存器。

0xC0-0xDF 寄存器说明

寄存器	地址	说明
-	0xC0	-
-	0xC1	-
-	0xC2	-
-	0xC3	-
-	0xC4	-
-	0xC5	-
-	0xC6	-
-	0xC7	-
-	0xC8	-
-	0xC9	-
-	0xCA	-
-	0xCB	-
-	0xCC	-
-	0xCD	-
-	0xCE	-
-	0xCF	-
PSW	0xD0	系统标志寄存器。
-	0xD1	-
ADM	0xD2	ADC 控制寄存器。
ADB	0xD3	ADC 数据缓存器。
ADR	0xD4	ADC 分辨率选择寄存器。
VREFH	0xD5	ADC 参考电压控制寄存器。
-	0xD6	-
-	0xD7	-
S0CON2	0xD8	UART 波特率控制寄存器。
-	0xD9	-
-	0xDA	-
-	0xDB	-
-	0xDC	-
-	0xDD	-
-	0xDE	-
-	0xDF	-

0xE0-0xFF 寄存器说明

寄存器	地址	说明
ACC	0xE0	ACC 寄存器。
-	0xE1	-
-	0xE2	-
-	0xE3	-
P0OC	0xE4	开漏功能控制寄存器。
CLKSEL	0xE5	时钟切换选择寄存器。
CLKCMD	0xE6	时钟切换控制寄存器。
TCON0	0xE7	T0/1 时钟控制寄存器。
-	0xE8	-
-	0xE9	-
-	0xEA	-
-	0xEB	-
-	0xEC	-
-	0xED	-
-	0xEE	-
-	0xEF	-
B	0xF0	乘法/除法指令数据缓存器。
P0UR	0xF1	P0 上拉电阻控制寄存器。
FRQL	0xF2	时钟微调控制寄存器低字节。
FRQH	0xF3	时钟微调控制寄存器高字节。
FRQCMD	0xF4	时钟微调命令寄存器。
-	0xF5	-
-	0xF6	-
SRST	0xF7	软件复位控制寄存器。
-	0xF8	-
P0M	0xF9	P0 输入/输出模式寄存器。
-	0xFA	-
-	0xFB	-
-	0xFC	-
-	0xFD	-
-	0xFE	-
PFLAG	0xFF	复位标志寄存器。

6.3 系统寄存器

寄存器	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	说明
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	说明
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	说明
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

SN8F5701 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：

```
$NOMOD51 ; Do not recongnize the 8051-specific predifined special registers.
#include <SN8F5701H>
```

使用 C 运用进行编程是，增加下面的句子：

```
#include <SN8F5701H>
```

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。

Device	Header file	Options file
SN8F5701	SN8F5701.h	OPTIONS_SN8F5701.A51
SN8F57011	SN8F57011.h	

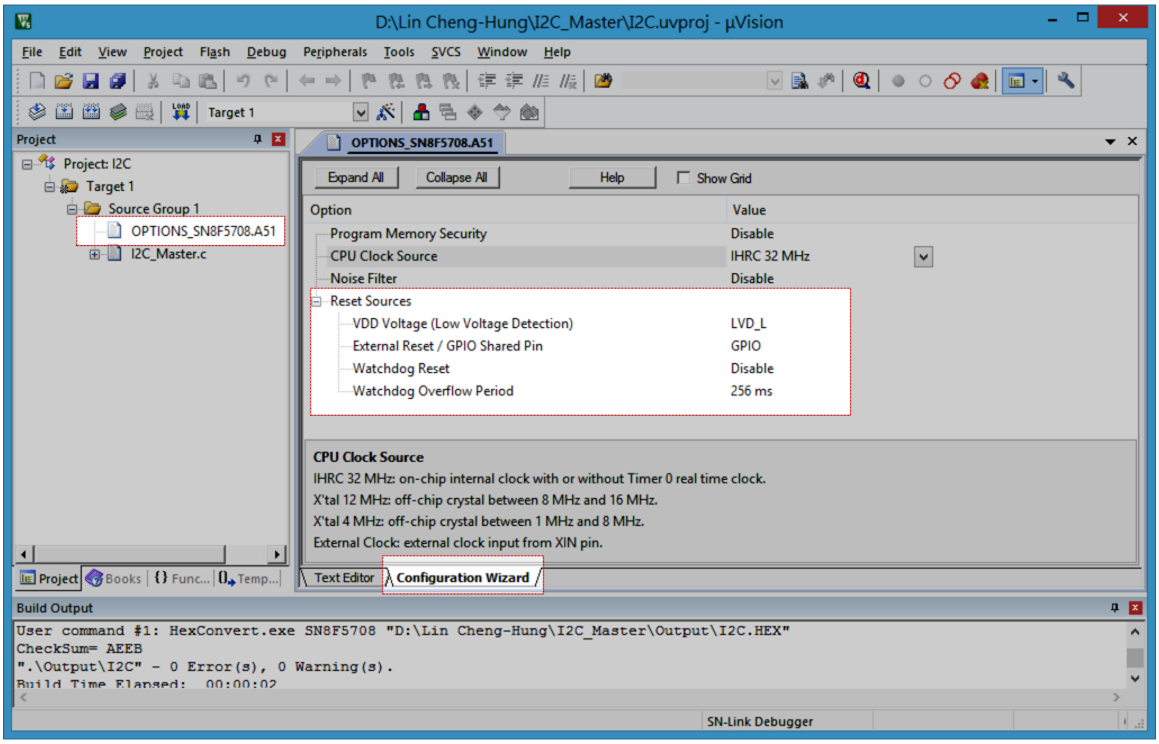
7 复位和上电控制

复位和上电控制有下列方式：低电压检测（LVD），看门狗，可编程的外部复位引脚和软件复位。前面三种方式可触发额外的上电流程，随后单片机初始化所有的寄存器，并使程序从复位向量（ROM 地址 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 OPTIONS_SN8F5701.A51 文件，该文件包含在 Keil C51.exe 的 SN-Link 驱动中（从松翰官网 www.sonix.com.tw 下载）。该选项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。

- 程序加密
- **Noise Filter**
- **CK_Fine_Tuning**
- **复位源：外部复位/GPIO 共用引脚**
- **复位源：看门狗复位＆溢出周期**

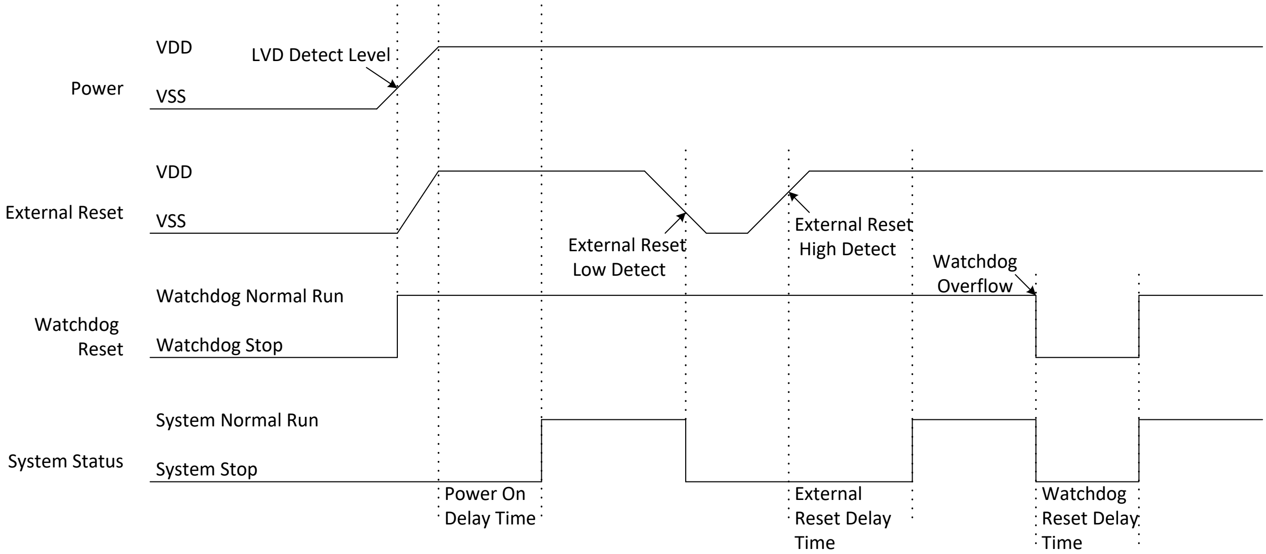


Code option 是系统硬件配置包含：晶振类型，Noise Filter 选项，看门狗时间操作，复位脚操作和 Flash ROM 加密控制。

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
Noise Filter	Disable	禁止杂讯滤波功能
	Enable	使能杂讯滤波功能
CK_Fine_Tuning	Disable	禁止 CK_Fine_Tuning
	Enable	使能 CK_Fine_Tuning
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。

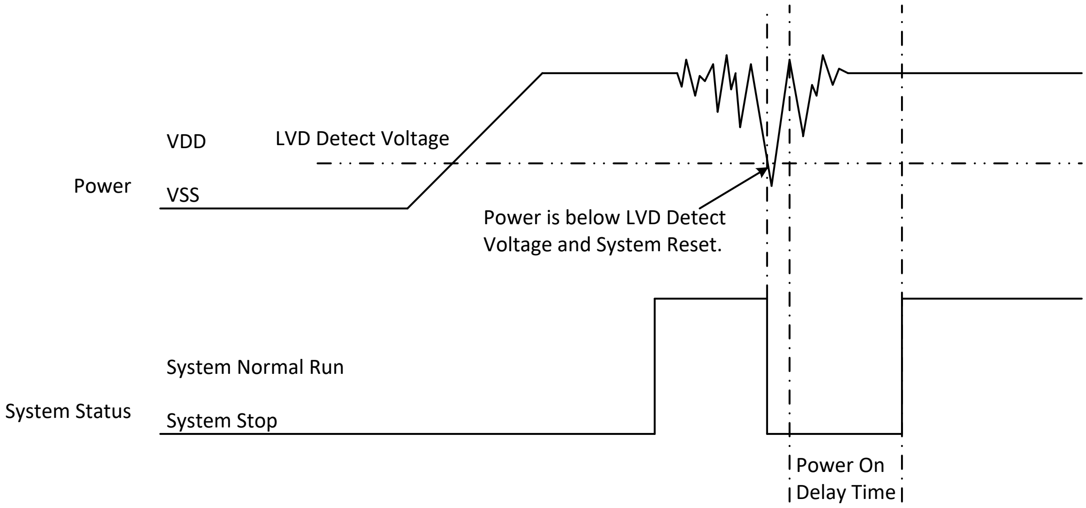


一般情况下，电源稳定期花费 4.6ms，之后单片机会自动获取 CPU 时钟源的配置。所选择的时钟源驱动起来之后，系统会计数 2048 个时钟周期以及 5 个内部低速振荡时钟以确保时钟已稳定。

> ✱ ***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，包括 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。下表显示了低电压检测配置 LVD_L，和 VDD 引脚状态的结果。



条件	LVD_L
VDD ≤ 1.8 V	复位

7.4 看门狗复位

看门狗是周期性的复位信号发生器，用于监控程序的执行流程。其内部定时器可在程序流程的检测点被清零，因此，只有在发生软件问题后才会产生实际的复位信号。通过写入 0x5A 到 WDTR 就可以在程序中设置检测点。

```
WDTR = 0x5A;
```

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

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

看门狗的工作模式可在选项文件中配置：

Always Mode：其内部定时器在 CPU 的所有工作模式（NORMAL，IDLE，SLEEP）下都在计数。

Enable Mode：其内部定时器只在 CPU 的 NORMAL 模式下计数，在 IDLE 和 SLEEP 模式下不会触发看门狗复位。

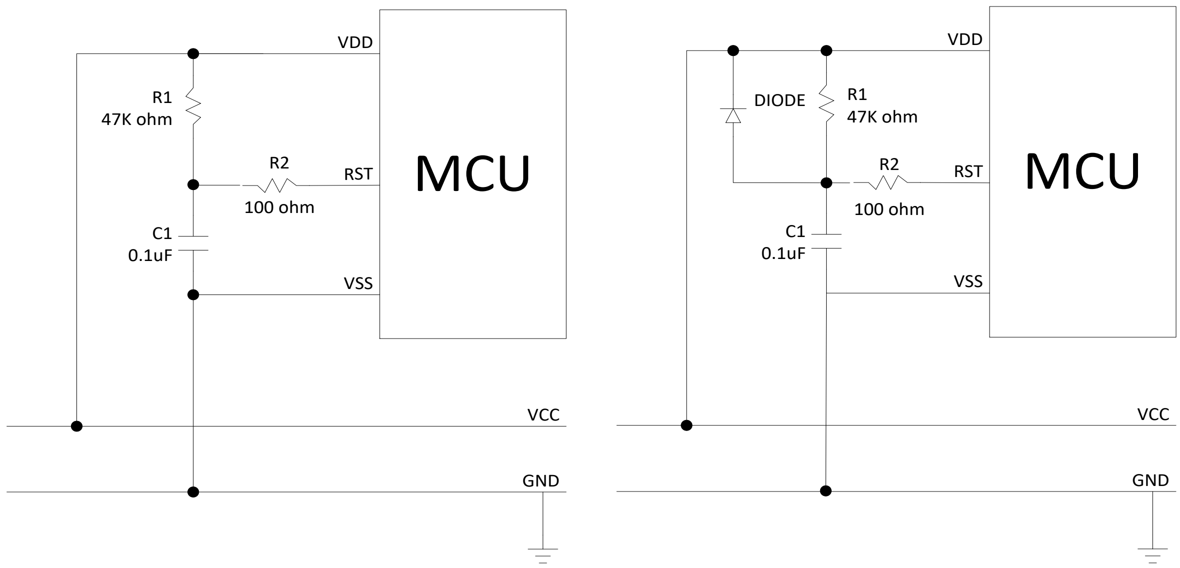
Disable Mode：其内部定时器在 CPU 的所有工作模式下都不计数，在这种模式下不会触发看门狗复位。

看门狗的工作模式为 Always on 时，系统的功耗较大。

7.5 外部复位引脚

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

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



> ✸ ***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。

```
SRST = 0x01;
SRST = 0x01;
```

7.7 复位和上电控制寄存器

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

PFLAG 寄存器

Bit	Field	Type	Initial	说明
7	POR	R	0	若单片机已经由 LVD 触发复位，则该位自动置 1。
6	WDT	R	0	若单片机已经由看门狗触发复位，则该位自动置 1。
5	RST	R	0	若单片机已经由外部复位引脚触发复位，则该位自动置 1。
4..0	Reserved	R	0	

SRST 寄存器

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

WDTR 寄存器（0x86）

Bit	Field	Type	Initial	说明
7..0	WDTR[7:0]	W	-	由 WDTR 寄存器清看门狗定时器，写入 0x5A 和到 WDTR 寄存器复位看门狗定时器。

8 系统时钟和电源管理

出于省电的目的，单片机内置 3 个不同的操作模式：NORMAL 模式，IDLE 模式和 STOP 模式。

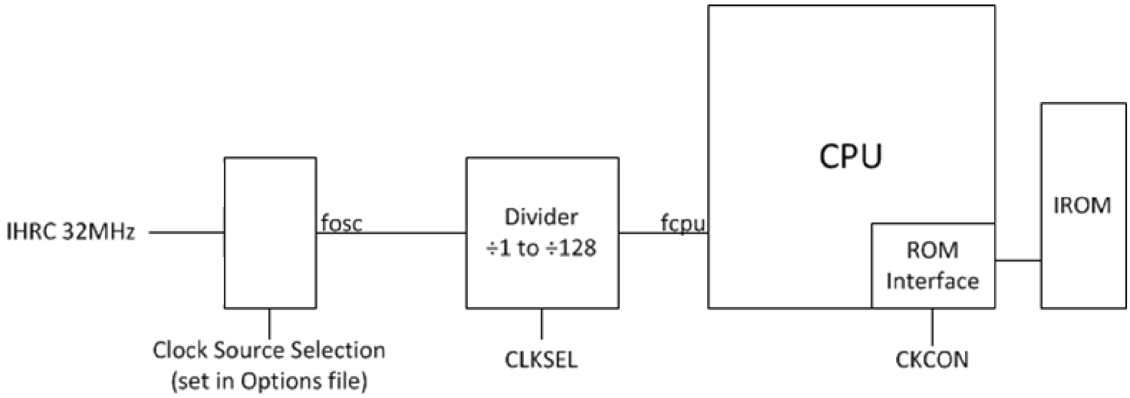
NORMAL 模式是指 CPU 和外设功能都正常工作，系统时钟是通过所选择的时钟源，程序设置的分频数，以及程序设置的 ROM 等待时间所确定的。IDLE 模式是指 CPU 时钟暂停的状态，但保留了外设功能（如定时器，PWM，和 UART）。与之相反的是，STOP 模式则禁止所有功能和时钟发生器，直至唤醒信号将系统唤醒进入 NORMAL 模式。

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。

```
CKCON = 0x70;       // For change safely the system clock
CLKSEL = 0x05;     // Set fcpu = fosc / 4
CLKCMD = 0x69;      // Apply CLKSEL's setting
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= 0x05~0x00, 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	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

8.2 高速时钟和实时时钟

高速时钟仅有内部类型。内部高速时钟 32MHz RC type。

- IHRC 32M：系统高速时钟源是内部高速 32MHz RC 型振荡器。

8.3 电源管理

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

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

若 IDLE 位设置为 1，只有 CPU 时钟源仍使能。因此，在这种状态下，外设功能（如定时器，PWM 和 UART）和时钟发生器（IHRC 32MHz）仍然正常工作。P0 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。

- 所有功能都可在 IDLE 模式下工作，只有 CPU 停止工作。
- IDLE 模式下的唤醒源为 P0 电平变换和中断。

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 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，外设和时钟发生器都处于停止状态。
- STOP 模式下的唤醒源为 P0 电平变换。

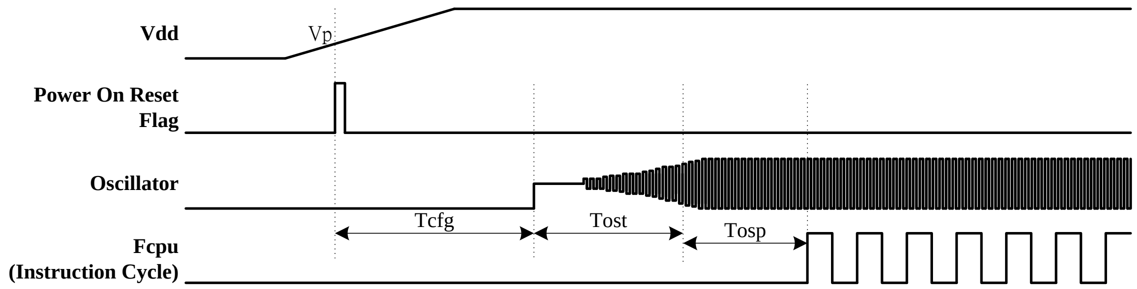
用户用 C 编译器开发时，强烈建议使用 IDLE 和 STOP 宏来控制单片机的系统模式，来代替直接设置 IDLE 和 STOP 位。

```
IDLE();
STOP();
```

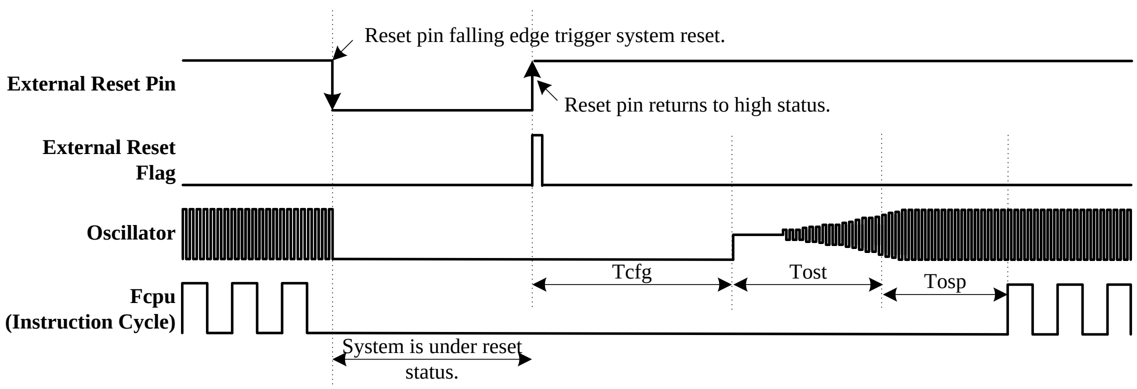
8.4 系统时间计时

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）	377us @ Fhosc = 32MHz
		掉电模式唤醒条件下的振荡器预热时间。 64*Fhosc+5*FILRC……RC 型振荡器，例如内部高速 RC 型振荡器。	RC: 315us @ Fhosc = 32MHz

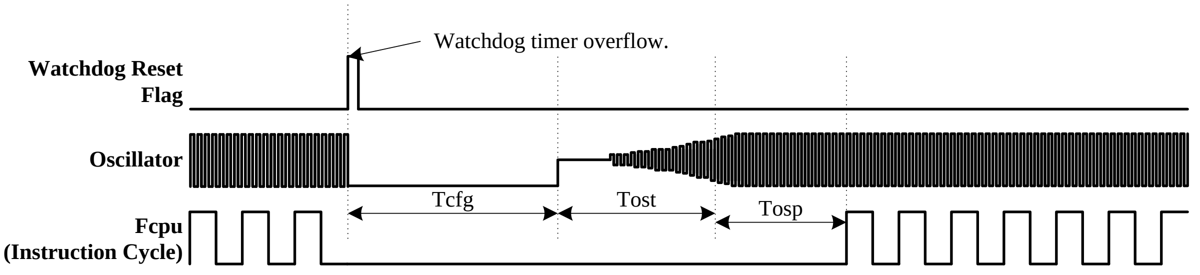
- 上电复位计时



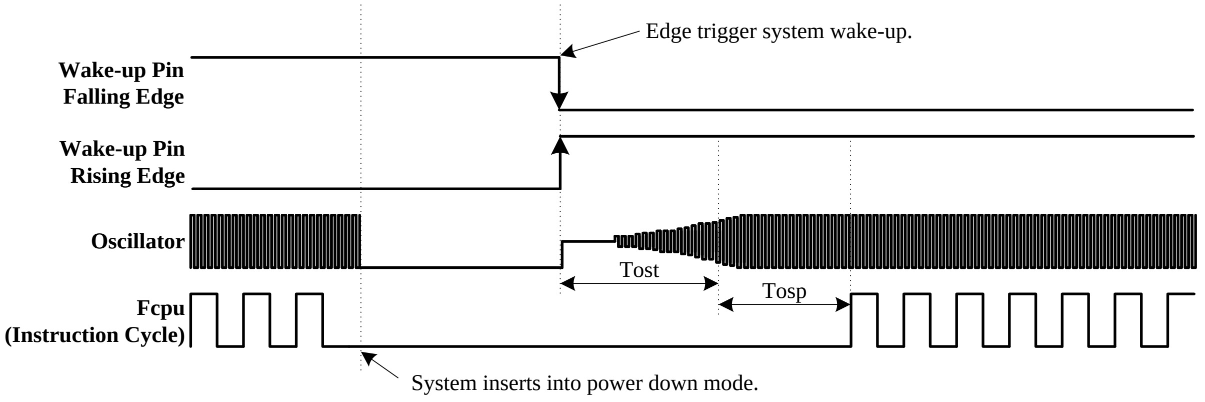
- 外部复位引脚复位计时



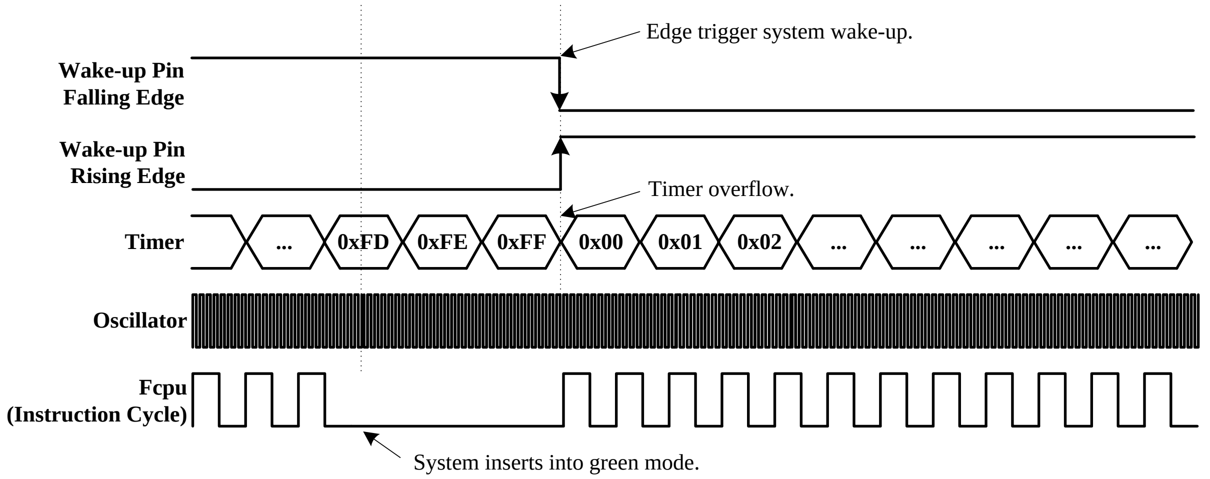
- 看门狗复位计时



- STOP 模式唤醒计时

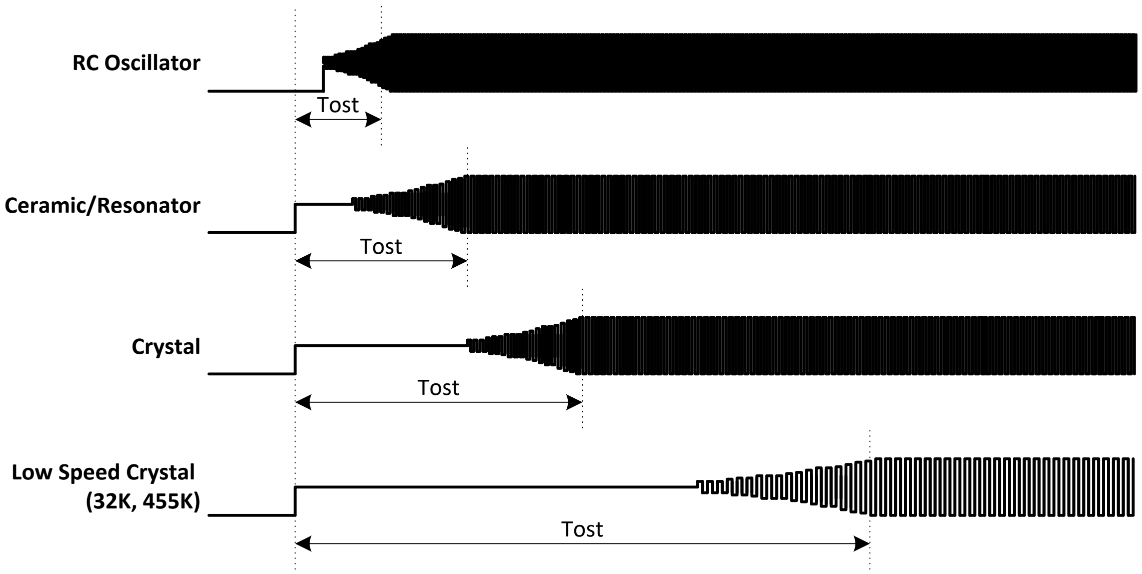


- IDLE 模式唤醒计时



- 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。



8.5 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	-	-	-	-	GF0	STOP	IDLE

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。
Else	Reserved	R	0001	

CLKSEL 寄存器（0xE5）

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

CLKCMD 寄存器（0xE6）

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

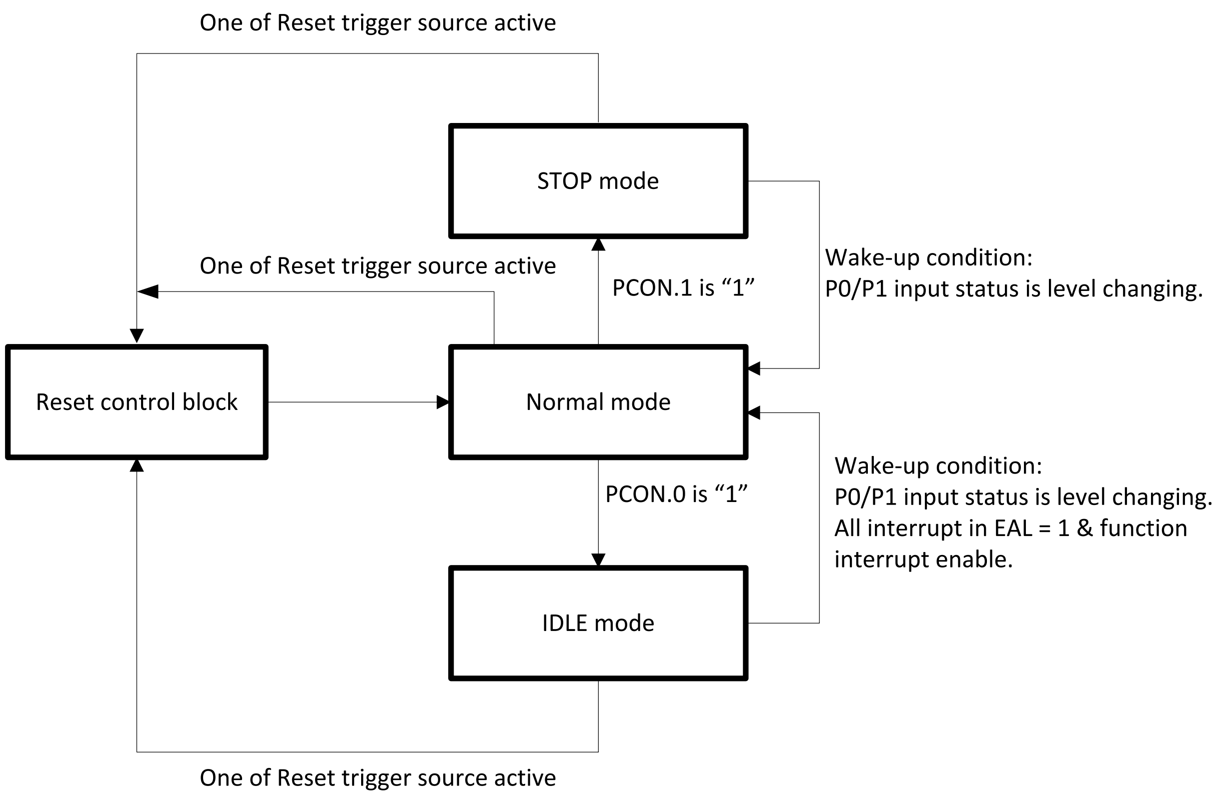
PCON 寄存器（0x87）

Bit	Field	Type	Initial	说明
7				参考其他章节
6..3	Reserved	R	0x00	
2	GF0	R/W	0	通用标志位。
1	STOP	R/W	0	1：单片机切换到 STOP 模式。
0	IDLE	R/W	0	1：单片机切换到 IDLE 模式。

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。

- NORMAL 模式：系统高速工作模式
- IDLE 模式：系统空闲模式
- STOP 模式：系统省电模式



工作模式时钟控制见下表：

工作模式	NORMAL 模式	IDLE 模式	STOP 模式
IHRC	IHRC：运行	IHRC：运行	Stop
ILRC	运行	运行	Watchdog always：运行 其它：禁止
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 3 （定时器，事件计数器）	由 T3EN 激活	由 T3EN 激活	未激活
UART	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，复位，EAL 中的所有中断 = 1 & 功能中断使能	P0，复位

- IHRC：内部高速振荡器 RC 型
- ILRC：内部低速振荡器 RC 型

9.1 NORMAL 模式

NORMAL 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 NORMAL 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 NORMAL 模式。在 NORMAL 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。
- NORMAL 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 NORMAL 模式。

9.2 STOP 模式

STOP 模式没有程序执行，也没有振荡器工作。只有内部保持电路激活，保持所有控制 IO 状态、寄存器状态和 SRAM 内容。STOP 模式由 P0/P1 电平改变触发。P0 唤醒功能通常是使能的。STOP 模式唤醒为 NORMAL 模式。进入 STOP 模式由 PCON 寄存器的停止位控制。当 PCON.1=1 时，系统进入 STOP 模式。从 STOP 模式系统唤醒后，PCON.1 自动清零。

- 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。
- 内部保持电路激活，保持所有控制 IO 状态、寄存器状态和 SRAM 内容
- 系统从 STOP 模式唤醒后进入到 NORMAL 模式。

9.3 IDLE 模式

IDLE 模式与 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 PCON.0 位控制。PCON.0=1 时，系统进入 IDLE 模式。系统从 IDLE 模式唤醒后，PCON.0 位自动清零。

- 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 NORMAL 模式进入 IDLE 模式，系统在唤醒后进入 NORMAL 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 模式或 IDLE 模式下，不执行程序。唤醒触发可以唤醒系统恢复为 NORMAL 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 NORMAL 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 模式下，唤醒系统。唤醒时间后，系统进入 NORMAL 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 模式下，唤醒系统。唤醒时间后，系统进入 NORMAL 模式。唤醒时间如下。

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

> ✱ *注：高速时钟的启动时间与 **VDD** 和振荡器类型有关。*

在 STOP 模式和 IDLE 模式下，具有唤醒功能的 I/O 端口能够唤醒系统进入 NORMAL 模式。唤醒触发边缘是在上升或下降沿边缘的电平改变。P0 和 P1 具有唤醒功能。P0 唤醒功能一直使能，但 P1 由 P1W 寄存器控制。

10 中断

SN8F5701 包含 8 个中断源（3 个外部中断和 5 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
INT2	EX2	IE2	自动清 0	2 / 0x0083
Timer 0	ET0	TF0	自动清 0	3 / 0x000B
ADC	EADC	ADCF	自动清 0y	4 / 0x008B
INT1	EX1	IE1	自动清 0y	5 / 0x0013
Timer 1	ET1	TF1	自动清 0	6 / 0x001B
UART	ES0	TI0 / RI0	By firmware	7 / 0x0023
Timer 3	ET3	TF3	自动清 0	8 /0x00EB

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（0x03-0x8B），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	INT2
Group 1	T0	ADC
Group 2	INT1	
Group 3	T1	
Group 4	UART	
Group 5	T3	

IP0, IP1 寄存器

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

IP0 寄存器（0XA9）

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

IP1 寄存器（0XB9）

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

10.3请求标志位清零

使能中断功能时，大多数中断请求标志位自动由硬件清零，而一些则需要使用程序清零。但是，关闭中断功能时，中断请求标志位只需要由程序清零。

大多数情况下由 ANL 指令清零，但某些特殊情况下，如没有同时释放相邻的中断标志，是可能被意外清零的。

```
TCON &= 0x7F;     // It is possible to cause adjacent flags to be cleared.
TF1 = 0;          // It is possible to cause adjacent flags to be cleared.
```

若想要避免上述情况，建议使用中断位特性。大多数中断请求标志位不能置 1，这样就避免将非同步的相邻的标志位清零。

例如：由 MOV 指令将 IRCON 寄存器的 bit3 清零，只需要写入 0 到 bit3，其它的位写入 1。这样就避免将非同步的相邻的标志位清零。

```
TCONCLR(bit);       // The marco can clear the flag of TCON. bit is 0~7.
IRCON2CLR(bit);     // The marco can clear the flag of IRCON2. bit is 0~7.
```

10.4 中断寄存器

寄存器	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IEN0	EAL	-	-	ES0	ET1	EX1	ET0	EX0
IEN2	-	-	-	-	-	EX2	ET3	EADC
IRCON2	-	-	-	-	-	IE2	TF3	ADCF
TCON	TF1	TR1	TF0	TR0	IE1	-	IE0	-
S0CON	SM0	SM1	SM20	REN0	TB80	RB80	TI0	RI0

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。
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	EX0	R/W	0	外部中断 P0.1（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 P0.0（INT0）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	EX2	R/W	0	外部中断 P0.2（INT2）中断控制位。 0：禁止； 1：使能。
1	ET3	R/W	0	T3 定时器中断控制位。 0：禁止； 1：使能。
0	EADC	R/W	0	ADC 中断控制位。 0：禁止； 1：使能。
Else	Reserved	R	0	

IRCON2 寄存器（0XBF）

Bit	Field	Type	Initial	说明
2	IE2	R/W	0	外部中断 P0.2（INT2）中断请求标志位。 0：无 INT2 中断请求； 1：INT2 请求中断。
1	TF3	R/W	0	T3 定时器中断请求标志位。 0：无 T3 中断请求； 1：T3 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 P0.1（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 P0.0（INT0）中断请求标志位。 0：无 INT0 中断请求； 1：INT0 请求中断。
Else	Reserved	R	0	

S0CON 寄存器（0x98）

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

11 GPIO

SN8F5701 共有 6 个双向 GPIO 引脚，不同于 8051 只有开漏输出，SN8F5701 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 寄存器控制输入输出模式。

寄存器	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	-	-	P05M	P04M	P03M	P02M	P01M	P00M
P0OC	-	-	P05OC	P04OC	P03OC	P02OC	P01OC	P00OC

P0M 寄存器（0xF9）

Bit	Field	Type	Initial	说明
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	P04M	R/W	0	P0.4 的模式选择控制位。 0：输入模式； 1：输出模式。
3	P03M	R/W	0	P0.3 的模式选择控制位。 0：输入模式； 1：输出模式。
2	P02M	R/W	0	P0.2 的模式选择控制位。 0：输入模式； 1：输出模式。
1	P01M	R/W	0	P0.1 的模式选择控制位。 0：输入模式； 1：输出模式。
0	P00M	R/W	0	P0.0 的模式选择控制位。 0：输入模式； 1：输出模式。
Else	Reserved	R	0	

P0OC 寄存器（0xE4）

Bit	Field	Type	Initial	说明
5	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。
4	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。
3	P03OC	R/W	0	P0.3 开漏输出模式控制位。 0：禁止； 1：使能，输出高电平时变为输入模式。
2	P02OC	R/W	0	P0.2 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。
Else	Reserved	R	0	

*在执行调试功能时，建议禁止 P04 的开漏输出功能。

11.2 输入数据和输出数据

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

写给 P0 寄存器的值将被马上锁存，然而，只有在 P0M 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 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	-	-	P05	P04	P03	P02	P01	P00

P0 寄存器（0x80）

Bit	Field	Type	Initial	说明
5	P05	R/W	1	读：P0.5 为逻辑电平； 写入 1/0：输出高电平/低电平（P05M=1 时使能）。
4	P04	R/W	1	读：P0.4 为逻辑电平； 写入 1/0：输出高电平/低电平（P04M=1 时使能）。
3	P03	R/W	1	读：P0.3 为逻辑电平； 写入 1/0：输出高电平/低电平（P03M=1 时使能）。
2	P02	R/W	1	读：P0.2 为逻辑电平； 写入 1/0：输出高电平/低电平（P02M=1 时使能）。
1	P01	R/W	1	读：P0.1 为逻辑电平； 写入 1/0：输出高电平/低电平（P01M=1 时使能）。
0	P00	R/W	1	读：P0.0 为逻辑电平； 写入 1/0：输出高电平/低电平（P00M=1 时使能）。
Else	Reserved	R	0	

11.3 内置上拉寄存器

P0UR 寄存器管理每个引脚的内部 100KΩ（典型值）的上拉电阻。

寄存器	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	-	-	P05UR	P04UR	P03UR	P02UR	P01UR	P00UR

P0UR 寄存器 （0xF1）

Bit	Field	Type	Initial	说明
5	P05UR	R/W	0	P0.5 的内置上拉电阻控制位。 0：禁止*； 1：使能。
4	P04UR	R/W	0	P0.4 的内置上拉电阻控制位。 0：禁止*； 1：使能。
3	P03UR	R/W	0	P0.3 的内置上拉电阻控制位。 0：禁止*； 1：使能。
2	P02UR	R/W	0	P0.2 的内置上拉电阻控制位。 0：禁止*； 1：使能。
1	P01UR	R/W	0	P0.1 的内置上拉电阻控制位。 0：禁止*； 1：使能。
0	P00UR	R/W	0	P0.0 的内置上拉电阻控制位。 0：禁止*； 1：使能。
Else	Reserved	R	0	

* 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

SN8F5701 内置模拟功能，如 ADC。若使能引脚的模拟功能，强烈建议关闭输入通道的施密特触发。

寄存器	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0CON	-	-	P0CON5	P0CON4	P0CON3	P0CON2	P0CON1	P0CON0

P2CON 寄存器（0x9E）

Bit	Field	Type	Initial	说明
5	P0CON5	R/W	0	P0.5 施密特触发控制位。 0：使能； 1：禁止。
4	P0CON4	R/W	0	P0.4 施密特触发控制位。 0：使能； 1：禁止。
3	P0CON3	R/W	0	P0.3 施密特触发控制位。 0：使能； 1：禁止。
2	P0CON2	R/W	0	P0.2 施密特触发控制位。 0：使能； 1：禁止。
1	P0CON1	R/W	0	P0.1 施密特触发控制位。 0：使能； 1：禁止。
0	P0CON0	R/W	0	P0.0 施密特触发控制位。 0：使能； 1：禁止。
Else	Reserved	R	0	

12 外部中断

外部中断源 INT0，INT1 和 INT2 内置边沿触发功能，由 PEDGE 寄存器控制。使能外部中断（EX0/EX1/EX2）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/IE2）置 1，程序计数器跳转到中断向量（ORG 0x0003/0x0013/0x0083）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

寄存器	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEDGE	-	-	EX2G1	EX2G0	EX1G1	EX1G0	EX0G1	EX0G0
IEN0	EAL	-	ET2	ES0	ET1	EX1	ET0	EX0
IEN2	-	-	-	-	-	EX2	ET3	EADC
TCON	TF1	TR1	TF0	TR0	IE1	-	IE0	-
IRCON2	-	-	-	-	-	IE2	TF3	ADCF

PEDGE 寄存器（0x8F）

Bit	Field	Type	Initial	说明
5..4	EX2G[1:0]	R/W	10	外部中断 INT2 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。
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 P0.1 中断（INT1）控制位。 0：禁止； 1：使能。
0	EX0	R/W	0	外部 P0.0 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 P0.1 中断（INT1）请求标志位。 0：禁止； 1：使能。
1	IE0	R/W	0	外部 P0.0 中断（INT0）请求标志位。 0：禁止； 1：使能。
Else				参考其它章节。

IEN0 Register (0X9A)

Bit	Field	Type	Initial	说明
2	EX2	R/W	0	外部 P0.2 中断（INT2）控制位。 0：禁止。 1：使能。
Else				参考其它章节。

IRCON2 Register (0XBF)

Bit	Field	Type	Initial	说明
2	IE2	R/W	0	外部 P0.2 中断（INT2）请求标志位。 0：禁止。 1：使能。
Else				参考其它章节。

12.2 程式范例

下面的示例代码程序演示了如何执行 INT0/INT1/INT2 断。

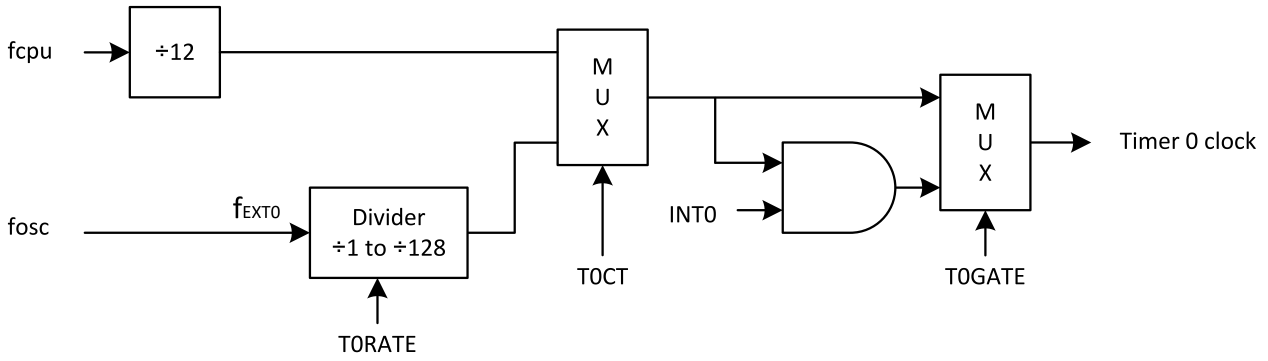
```c
#define INT0Rsing        (1 << 0)   //INT0 trigger edge is rising edge
#define INT0Falling     (2 << 0)   //INT0 trigger edge is falling edge
#define INT0LeChge       (3 << 0)   //INT0 trigger edge is level chagne
#define EINT0            (1 << 0)   //INT0 interrupt enable

#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 chagne
#define EINT1            (1 << 2)   //INT1 interrupt enable

#define INT2Rsing        (1 << 4)   //INT2 trigger edge is rising edge
#define INT2Falling     (2 << 4)   //INT2 trigger edge is falling edge
#define INT2LeChge       (3 << 4)   //INT2 trigger edge is level chagne
#define EINT2            (1 << 2)   //INT2 interrupt enable

void EnableINT(void)
{
   // INT0 rising edge, INT1 falling edge, INT2 level change
   PEDGE = INT0Rising |INT1Falling | INT2LeChge;

   // Enable INT0/INT1 interrupt
   IEN0 |= EINT0 | EINT1;
   // Enable INT2 interrupt
   IEN2 |= EINT2;
   // Enable total interrupt
   IEN0 |= 0x80;

   P0 = 0x00;
   P0M = 0x38;
}

void INT0Interrupt(void) interrupt ISRInt0   //0x03
{ //IE0 clear by hardware
   P03 = ~P03;
}

void INT1Interrupt(void) interrupt ISRInt1   //0x13
{ //IE1 clear by hardware
   P04 = ~P04;
}

void INT2Interrupt(void) interrupt ISRInt2   //0x83
{ //IE2 clear by hardware
   P05 = ~P05;
}
```

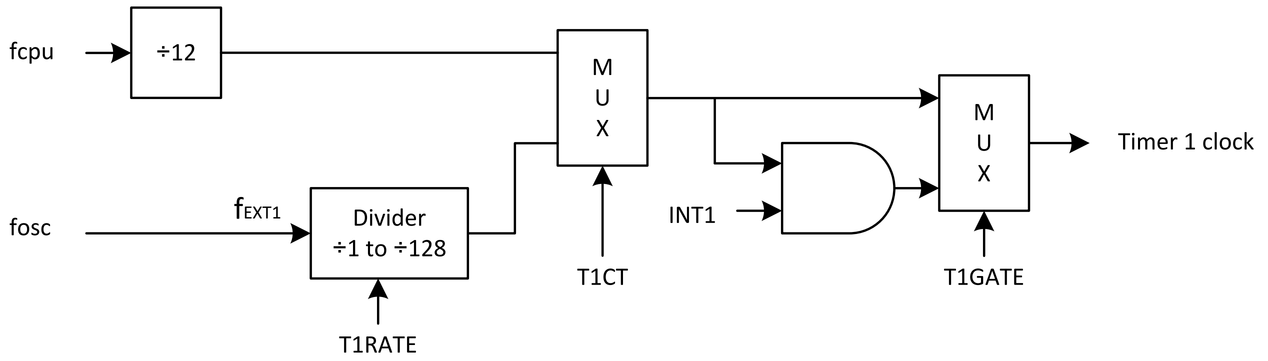
13 定时器 T0 和 T1

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

13.1 T0 和 T1 时钟选择

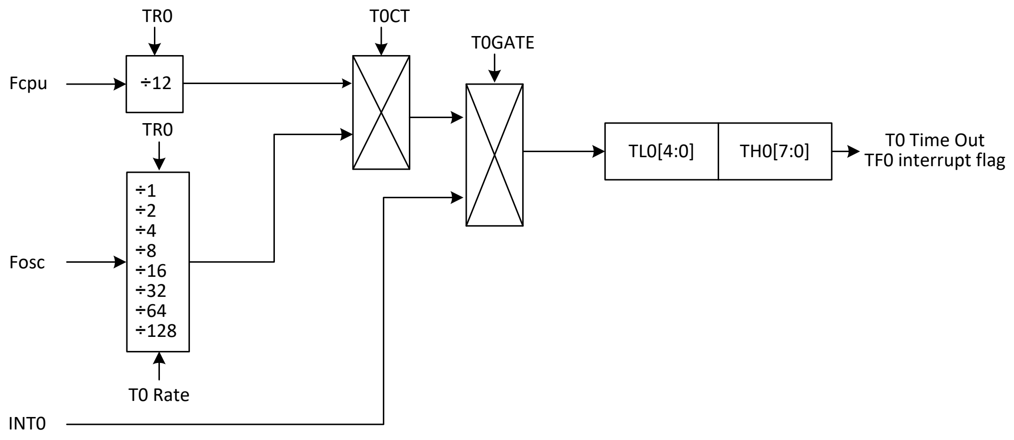
下图阐明了 T0 和 T1 的时钟选择电路。T0 有 2 组时钟源：Fcpu 和 Fosc。若使用 T0GATE，由 INT0 引脚来关闭（暂停）时钟源。而 T1 时钟源选择：Fcpu 和 Fosc。

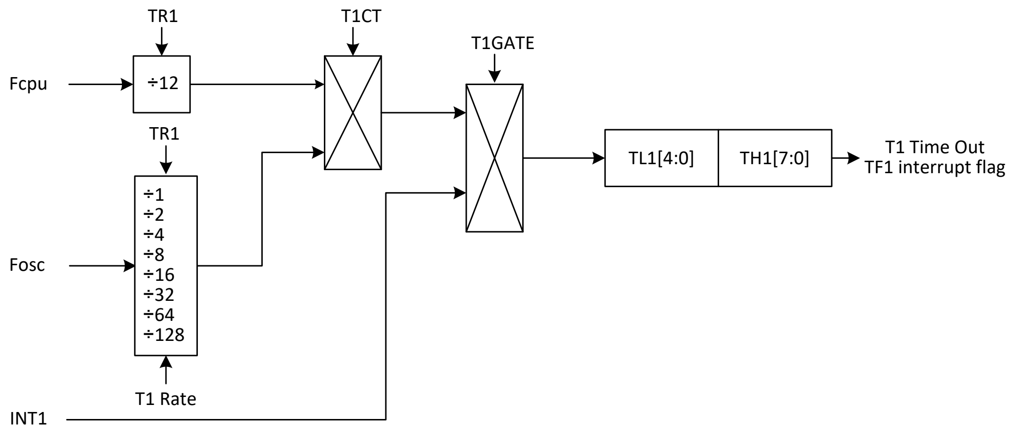




13.2 模式 0：13 位向上计数定时器

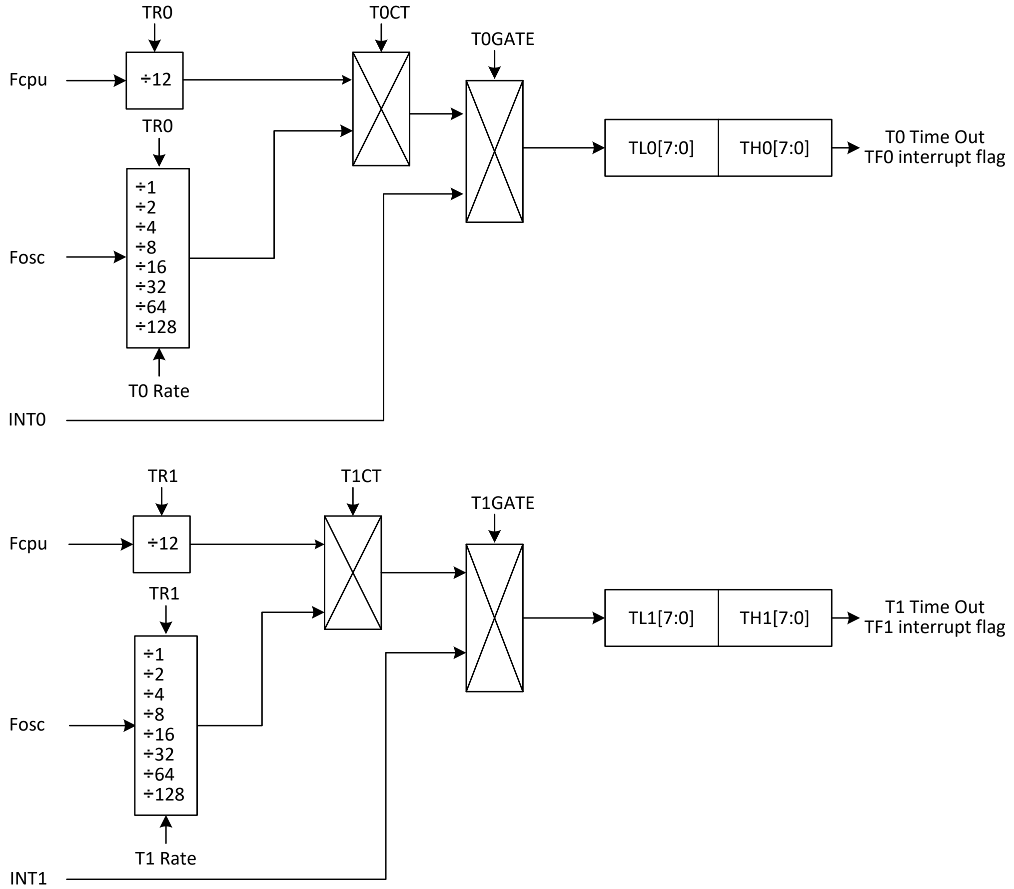
模式 0 是 13 位向上计数定时器（TL0/TL1 的高 3bit 无效）。一旦定时计数器溢出（由 0xFF1F 到 0x0000），会立即释放 TF0/TF1 标志。没有使能 ET0/ET1 时，由软件读取该标志；使能 ET0/ET1 时，则由中断控制器进行控制。





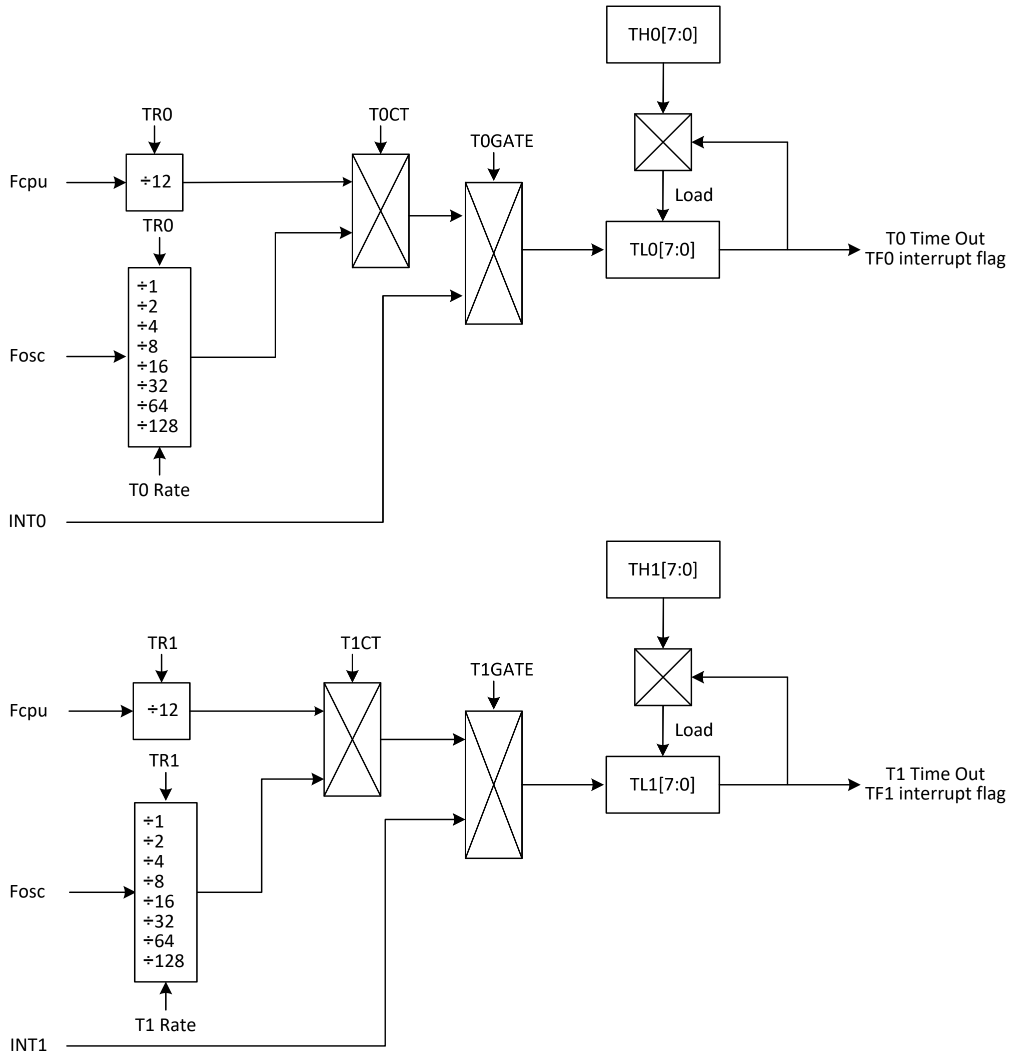
13.3 模式 1：16 位向上计数定时器

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），会释放 TF0/TF1，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（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。

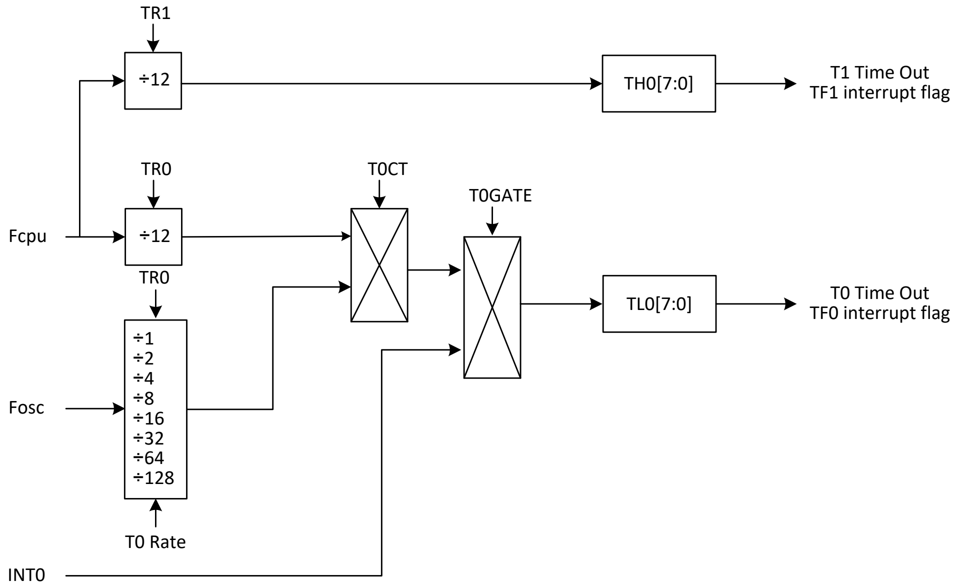


13.5 模式 3（只是 T0）：独立的 2 个 8 位向上计数定时器

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

在模式 3 下，由 TR0 使能 TL0 计数器，TL0 溢出后 TF0 置 1。TH0 计数器则由 TR1 控制，TH0 溢出后 TF1 置 1。

在该模式下，T1 不会发生溢出事件，可以看作是没有溢出标志产生计数器。



13.6 T0 和 T1 寄存器

寄存器	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TCON	TF1	TR1	TF0	TR0	IE1	-	IE0	-
TCON0	-	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	EAL	-	-	ES0	ET1	EX1	ET0	EX0

TCON 寄存器（0x88）

Bit	Field	Type	Initial	说明
7	TF1	R/W	0	T1 溢出事件显示位。 0：T1 没有溢出； 1：T1 溢出。 该位由中断助力器自动清零，或者由固件手动清零。
6	TR1	R/W	0	T1 功能控制位。 0：禁止； 1：使能。
5	TF0	R/W	0	T0 溢出事件显示位。 0：T0 没有溢出； 1：T0 溢出。 该位由中断助力器自动清零，或者由固件手动清零。
4	TR0	R/W	0	T0 功能控制位。 0：禁止； 1：使能。
3	IE1	R/W	0	参考 INT1。
2	Reserved	R	0	
1	IE0	R/W	0	参考 INT0。
0	Reserved	R	0	

IEN0 寄存器（0xA8）

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

TCON0 寄存器（0xE7）

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

TH0 / TH1 寄存器（TH0: 0x8C, TH1: 0x8D）

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

TL0 / TL1 寄存器（TL0: 0x8A, TL1: 0x8B）

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

TMOD 寄存器（0x89）

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

*(1) F_{EXT1} = fosc.
*(2) F_{EXT0} = fosc.

13.7 程式范例

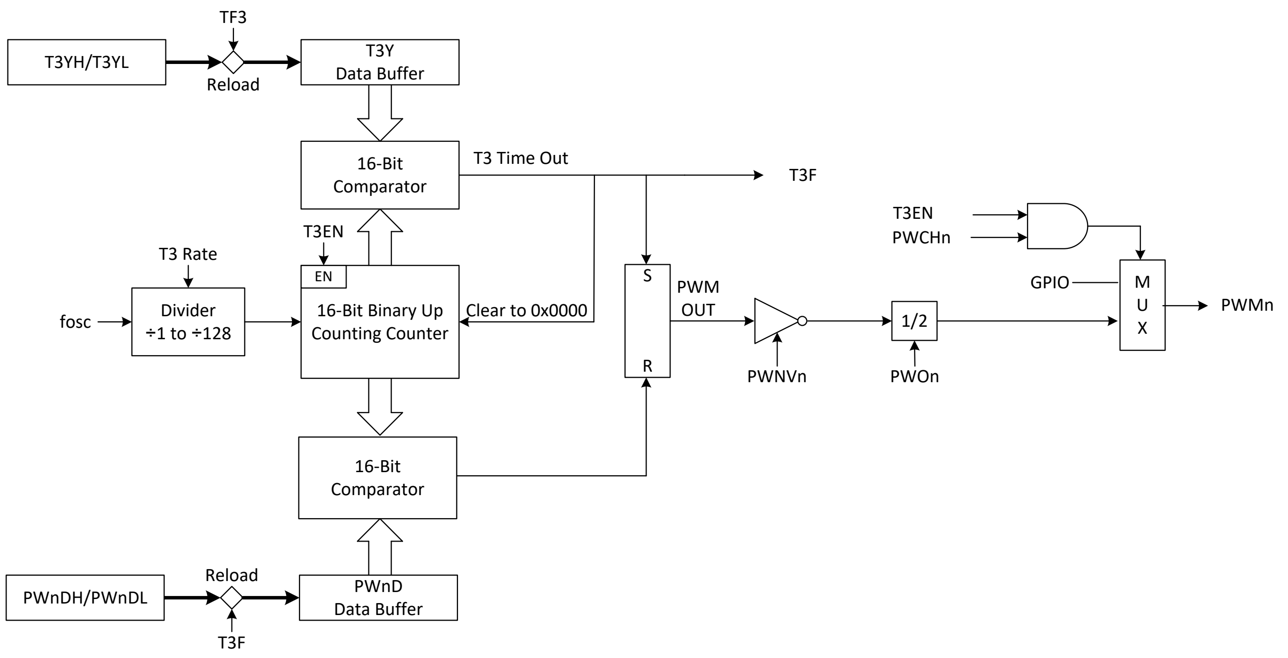
下面的示例程序代码显示了在中断下如何执行 T0/T1。

```
#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
#define T0ExtFosc      (0 << 4)   //T0 clock source from Fosc

#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 T1ClkExt       (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
  TMOD |= T0Mode0 | T0GATE | T0ClkExT;
  // T0 clock source = Fosc/1
  TCON0 |= 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 |= 0x0A;
  // 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 T3

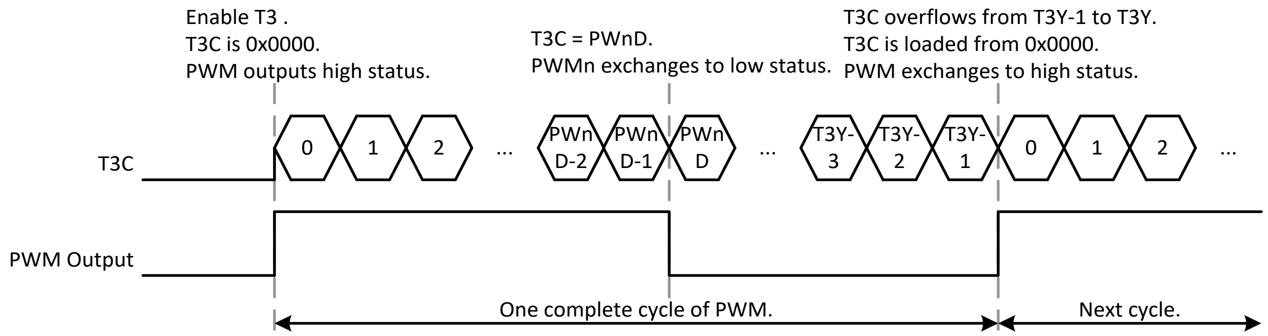
T3 是 16 位向上计数定时器，支持 6 通道 PWM 功能。计数器到达临界值（T3Y）时，计数器清零并触发中断。PWM 占空比周期由 PW0D~PW5D 寄存器控制。每个 PWM 通道都由自己的占空比控制。

PWM 功能包括 6 个可编程通道，与 GPIO 引脚共用，由 PWCH[5:0]控制。必须通过使能每个 PWCH[5:0]的位/通道执行输出操作。使能的 PWM 通道由 GPIO 切换为 PWM 输出功能。禁止 PWCH[5:0]后，PWM 通道返回上一个 GPIO 模式。若使能中断，T3 内置 IDEL 模式唤醒功能。定时器溢出时（从 T3Y-1 到 T3Y），立即释放 T3F，由软件进行读/写。由 ET3 控制 T3 的中断功能。

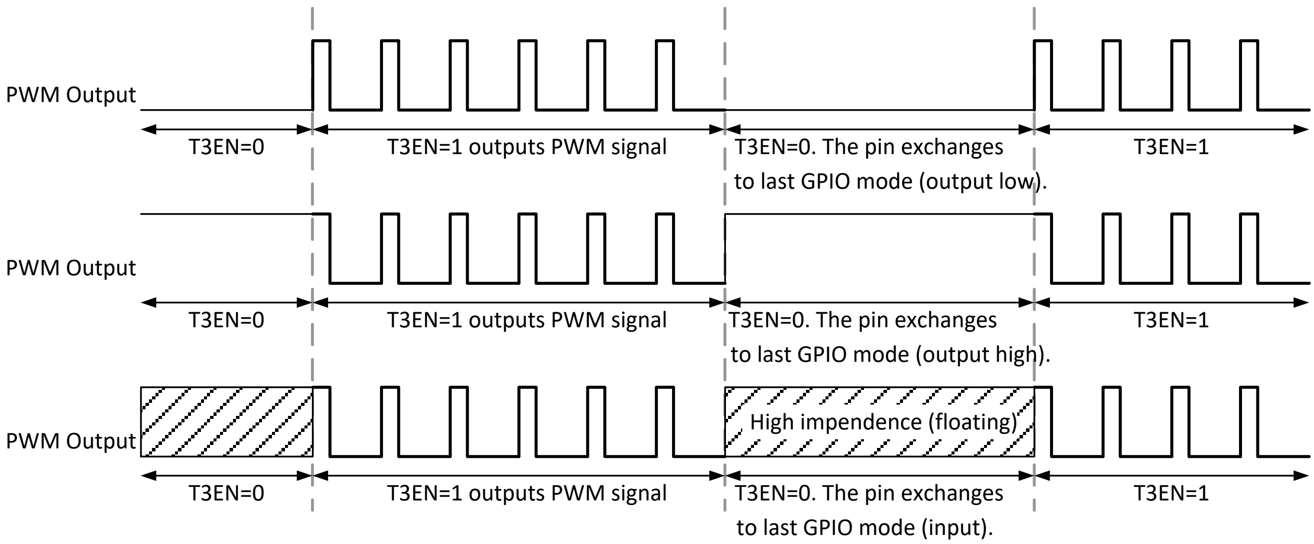


14.1 PWM

T3 内置 PWM 功能，由 T3EN 和 PWCH[5:4]位控制，PWM0~PWM5 为输出引脚，并与 GPIO 引脚共用，由 PWCH[5:0]控制。输出 PWM 时，必须设置 T3EN=1。PWM 输出信号同步完成后，PWM 通道从 GPIO 模式切换到 PWM 输出。T3EN=0 时，PWM 通道返回上一个 GPIO 模式。T3Y 和 PWnD 进行比较，其比较结果产生 PWM 信号。T3C 从 0x0000 开始计数时，PWM 输出高电平，即 PWM 的初始状态。T3 从 T3Y 寄存器加载新的值，以决定 PWM 的周期和分辨率。T3C 继续计数，系统比较 T3C 和 PWnD 的值，T3C=PWnD 时，PWM 输出低电平，T3C 继续计数，T3 定时器溢出（T3Y-1 到 0x0000）后，PWM 完成一个信号周期。T3C 自动从 0x0000 重新载入，且下一个周期 PWM 将输出高电平。PWnD 决定高电平占空比时间，T3Y 决定 PWM 的分辨率和周期。PWnD 的值不能大于 T3Y 的值，否则 PWM 信号会出错。PWM 的时钟源为 Fosc，由 T3RATE[2:0]控制：000=Fosc/128，001=Fosc/64，010=Fosc/32，011=Fosc/16，100=Fosc/8，101=Fosc/4，110=Fosc/2，111=Fosc/1。

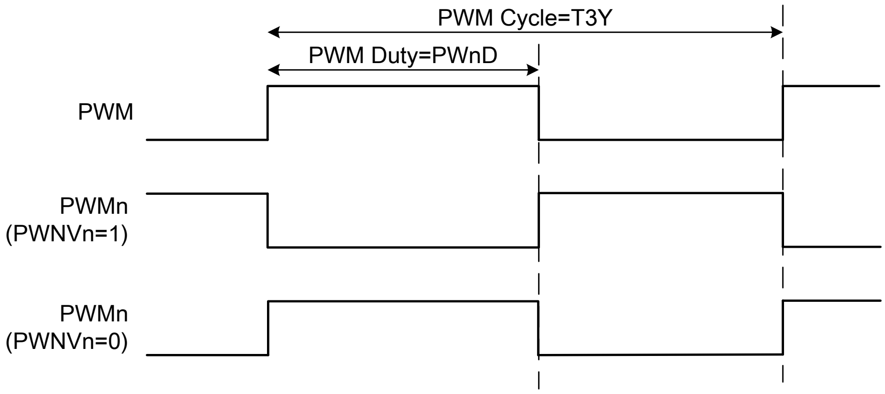


PWM 周期 = T3Y
PWM 占空比 = (PWnD):(T3Y-PWnD)

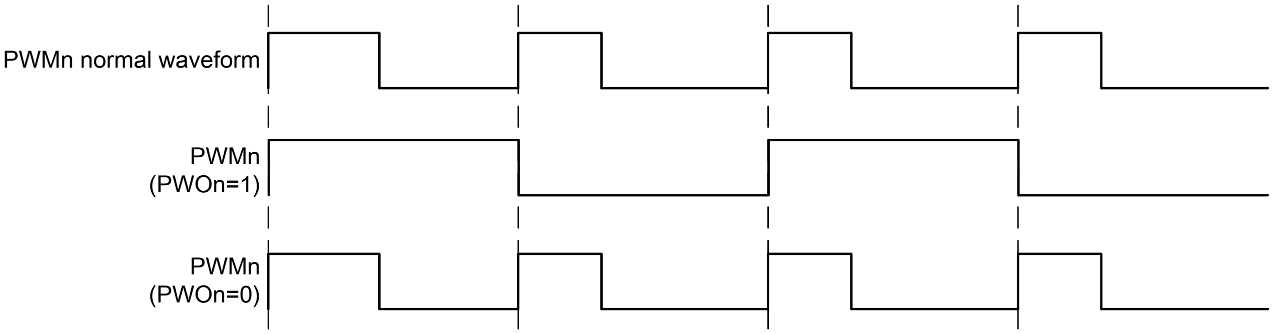


14.2 反相和频率模式

PWM 内置反相输出功能，当 PWNVn=1 时，PWM 有一个反向 PWM 信号。PWNVn=1 时，PWNVn 输出反向 PWM 信号；PWNVn=0 时，则输出正常 PWM 信号。反向 PWM 输出波形如下图所示：



PWM 具有频率模式用于更改 PWM 输出频率，频率模式由 PWO[5:0]控制。PWOn=1 时，PWMn 引脚输出 1/2*频率 PWM 信号；PWOn=0 时，PWMn 引脚输出 1*频率 PWM 信号。



14.3 PWM 寄存器

寄存器	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T3M	T3EN	T3rate2	T3rate1	T3rate0	-	-	-	-
PWNV	-	-	PWNV5	PWNV4	PWNV3	PWNV2	PWNV1	PWNV0
PWO	-	-	PWO5	PWO4	PWO3	PWO2	PWO1	PWO0
PWCH	-	-	PWCH5	PWCH4	PWCH3	PWCH2	PWCH1	PWCH0
T3CH	T3C15	T3C14	T3C13	T3C12	T3C11	T3C10	T3C9	T3C8
T3CL	T3C7	T3C6	T3C5	T3C4	T3C3	T3C2	T3C1	T3C0
T3YH	T3Y15	T3Y14	T3Y13	T3Y12	T3Y11	T3Y10	T3Y9	T3Y8
T3YL	T3Y7	T3Y6	T3Y5	T3Y4	T3Y3	T3Y2	T3Y1	T3Y0
PW0DH	PW0D15	PW0D14	PW0D13	PW0D12	PW0D11	PW0D10	PW0D9	PW0D8
PW0DL	PW0D7	PW0D6	PW0D5	PW0D4	PW0D3	PW0D2	PW0D1	PW0D0
PW1DH	PW1D15	PW1D14	PW1D13	PW1D12	PW1D11	PW1D10	PW1D9	PW1D8
PW1DL	PW1D7	PW1D6	PW1D5	PW1D4	PW1D3	PW1D2	PW1D1	PW1D0
PW2DH	PW2D15	PW2D14	PW2D13	PW2D12	PW2D11	PW2D10	PW2D9	PW2D8
PW2DL	PW2D7	PW2D6	PW2D5	PW2D4	PW2D3	PW2D2	PW2D1	PW2D0
PW3DH	PW3D15	PW3D14	PW3D13	PW3D12	PW3D11	PW3D10	PW3D9	PW3D8
PW3DL	PW3D7	PW3D6	PW3D5	PW3D4	PW3D3	PW3D2	PW3D1	PW3D0
PW4DH	PW4D15	PW4D14	PW4D13	PW4D12	PW4D11	PW4D10	PW4D9	PW4D8
PW4DL	PW4D7	PW4D6	PW4D5	PW4D4	PW4D3	PW4D2	PW4D1	PW4D0
PW5DH	PW5D15	PW5D14	PW5D13	PW5D12	PW5D11	PW5D10	PW5D9	PW5D8
PW5DL	PW5D7	PW5D6	PW5D5	PW5D4	PW5D3	PW5D2	PW5D1	PW5D0
IEN0	EAL	-	ET2	ES0	ET1	EX1	ET0	EX0
IEN2	-	-	-	-	-	EX2	ET3	EADC
IRCON2	-	-	-	-	-	IE2	TF3	ADCF

T3 寄存器（T3M: 0XA1）

Bit	Field	Type	Initial	说明
7	T3EN	R/W	0	T3 功能控制位。 0：禁止； 1：使能*。
6..4	T3RATE	R/W	000	T3 时钟源选择位。 000：Fosc / 128； 001：Fosc / 64； 010：Fosc / 32； 011：Fosc / 16； 100：Fosc / 8； 101：Fosc / 4； 110：Fosc / 2； 111：Fosc / 1。
Else	Reserved	R	0	

*周期设置为 00H 时，使能 T3 后，T3 停止工作，周期不能更新。

PWNV 寄存器（0xBC）

Bit	Field	Type	Initial	说明
7..6	Reserved	R/W	0	
5..0	PWNV[5:0]	R/W	0	PWM5~PWM0 引脚输出控制位。 0：正常输出； 1：反相输出。

PWO 寄存器（0xBD）

Bit	Field	Type	Initial	说明
7..6	Reserved	R/W	0	
5..0	PWO[5:0]	R/W	0	PWM5~PWM0 引脚输出控制位。 0：1*频率 PWM 信号； 1：1/2 *频率 PWM 信号。

PWCH 寄存器（0xBE）

Bit	Field	Type	Initial	说明
7..6	Reserved	R/W	0	
5..0	PWCH[5:0]	R/W	0	PWM5~PWM0 共用引脚控制位。 0：GPIO； 1：PWM 输出（与 P0.0~ P0.5 共用）。

T3CH/T3CL 寄存器（T3CH: 0XA3, T3CL: 0xA2）

Bit	Field	Type	Initial	说明
7..0	T3CH/L	R	0x00	16 位 T3 计数器。

T3YH/T3YL 寄存器（T3YH: 0xA5, T3YL: 0xA4）

Bit	Field	Type	Initial	说明
7..0	T3YH/L	R/W	0x00	16 位 T3 周期控制位*。

* 在开始 T3 功能之前，必须完成对周期配置的设置。

PWnDH/PWnDL 寄存器（PW0DH: 0xA7, PW0DL: 0xA6 / PW1DH: 0xAC, PW1DL: 0xAB / PW2DH: 0xAE, PW2DL: 0xAD / PW3DH: 0xB2, PW3DL: 0xB1 / PW4DH: 0xB4, PW4DL: 0xB3 / PW5DH: 0xB6, PW5DL: 0xB5）

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

IEN0 寄存器（0xA8）

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

IEN2 寄存器（0x9A）

Bit	Field	Type	Initial	说明
1	ET3	R/W	0	T3 定时器中断控制位。 0：禁止； 1：使能。
Else				参考其它章节。

IRCON2 寄存器（0xBF）

Bit	Field	Type	Initial	说明
1	TF3	R/W	0	T3 定时器中断请求标志位。 0：无 T3 中断请求； 1：T3 请求中断。
Else				参考其它章节。

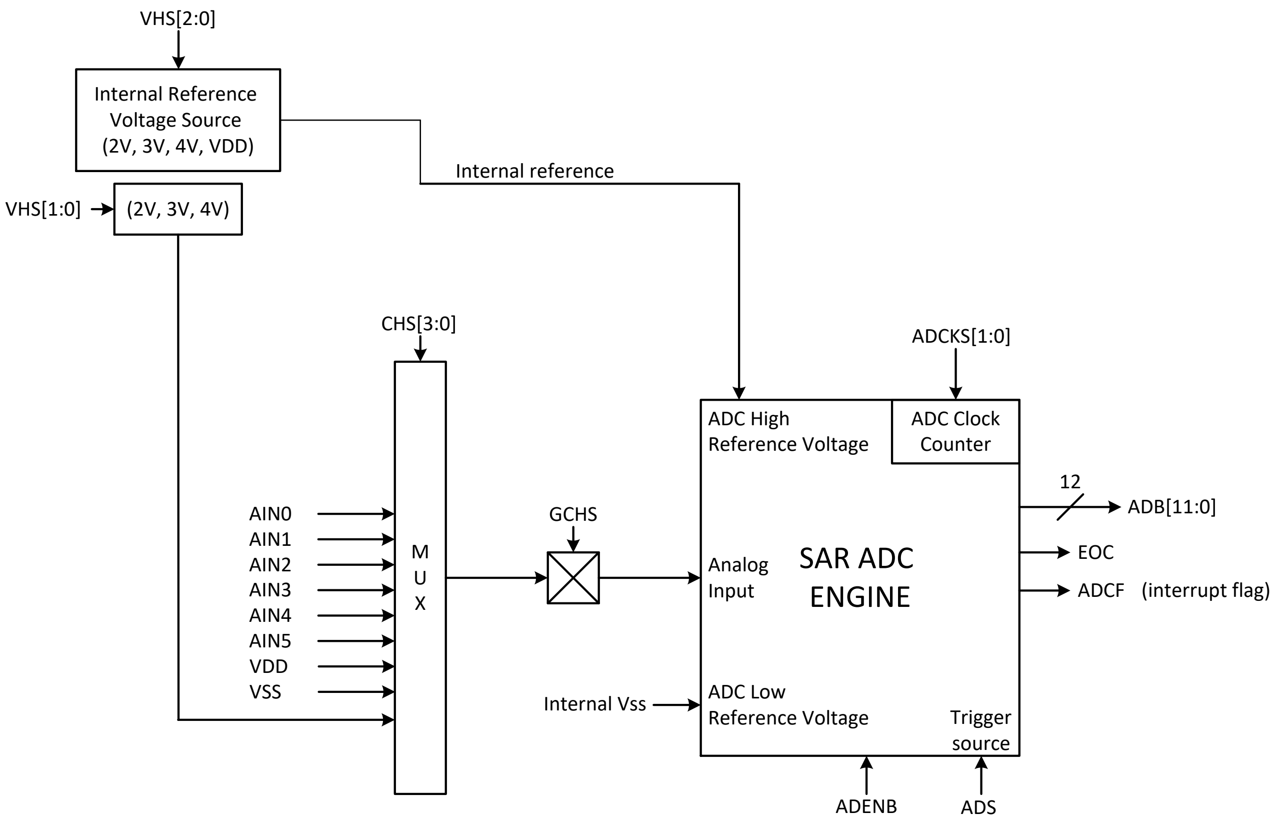
14.4程式范例

下面的示例程序代码显示了在中断下如何执行 T3。

```
#define PWMInv0        (1 << 0)   //PWM0 output inverse
#define PWMInv1        (1 << 1)   //PWM1 output inverse
#define PWMInv2        (1 << 2)   //PWM2 output inverse
#define PWMInv3        (1 << 3)   //PWM3 output inverse
#define PWMInv4        (1 << 4)   //PWM4 output inverse
#define PWMInv5        (1 << 5)   //PWM5 output inverse
#define PWM2Frq0       (1 << 0)   //PWM0 output 2*frequency PWM
#define PWM2Frq1       (1 << 1)   //PWM1 output 2*frequency PWM
#define PWM2Frq2       (1 << 2)   //PWM2 output 2*frequency PWM
#define PWM2Frq3       (1 << 3)   //PWM3 output 2*frequency PWM
#define PWM2Frq4       (1 << 4)   //PWM4 output 2*frequency PWM
#define PWM2Frq5       (1 << 5)   //PWM5 output 2*frequency PWM
#define PWM0En         (1 << 0)   //Enable PWM0 output function
#define PWM1En         (1 << 1)   //Enable PWM1 output function
#define PWM2En         (1 << 2)   //Enable PWM2 output function
#define PWM3En         (1 << 3)   //Enable PWM3 output function
#define PWM4En         (1 << 4)   //Enable PWM4 output function
#define PWM5En         (1 << 5)   //Enable PWM5 output function
#define T3En           (1 << 7)   //Enable T3 function

void InitT3(void)
{
   // T3_Initial
   T3YH = 0x80;
   T3YL = 0x00;
   PW1DH = 0x40;
   PW1DL = 0x00;

   // PWM1 channel enable
   PWCH = PWM1En;

   // T3 enable, PWM1 output inverse, clock = Fosc/32
   T3M = T3En | PWMInv1 | 0x20;

   // Enable T3 interrupt & clear T3F
   IEN2 = 0x02;
   IRCON2 = 0x00;

   // Enable total interrupt
   IEN0 |= 0x80;

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

void T3Interrupt(void) interrupt ISRT3   //0xEB
{ //T3F clear by hardware
   P00 = ~P00;
}
```

15 ADC

模拟数字转换（ADC）是一个 SAR 结构，内置 6 个输入通道，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 6 通道输入源，可以测量 10 种不同的模拟信号源，ADC 的分辨率为 12 位。ADC 共有 4 种时钟率来决定 ADC 的转换速率。ADC 参考高电压包括 4 种：4 种内部参考源（Vdd、4V、3V 和 2V）。ADC 内置 P0CON 寄存器来设置模拟输入引脚。设置好 ADENB 和 ADS 位后，ADC 开始将模拟信号转换为数字信号。ADC 可在 IDLE 模式下工作。ADC 结束后，若使能中断，则将系统从 IDLE 模式下唤醒进入 NORMAL 模式。



15.1 操作配置

ADC 开始转换前必须先设置好下列配置，具体如下：
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 高参考电压和低参考电压（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。

15.2 ADC 输入通道

SN8F5701 的 ADC 内置 6 个输入通道（AIN0~AIN5），用于测量 6 种不同的模拟信号源，由 CHS[3:0]和 GCHS 位控制。AIN8 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，AIN8 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。

ADC 输入通道

CHS[3:0]	通道	引脚名称	备注
0000	AIN0	P0.0	
0001	AIN1	P0.1	
0010	AIN2	P0.2	
0011	AIN3	P0.3	
0100	AIN4	P0.4	
0101	AIN5	P0.5	
1000	AIN8	内部 2V、3V 或 4V	电池检测器通道
1001 – 1111	-	-	保留

15.2.1 引脚配置

ADC 输入通道引脚与 P0 GPIO 引脚共用，由 CHS[3:0]选择控制。在同一时间，只能采集一路 AD，其它当做 AD 输入口的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 P0CON。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 P0 输入多个模拟信号时，也会产生额外的漏电流。S 模式下，上述漏电流会严重影响到系统的整体功耗。将 P0CON 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，P0CON 必须置 0，否则 GPIO 引脚的信号会被隔离。

15.3 参考电压

ADC 内置 4 种高参考电压，由 VREFH 寄存器控制：包括 4 个内部参考源（VDD、4V、3V、2V），由 VHS[2:0] 选择控制。VHS2 = 1 时，ADC 参考源选择 VDD；VHS[1:0] = 10 时，ADC 参考源选择 4V； VHS[1:0] = 01 时，ADC 参考源选择 3V；VHS[1:0] = 00 时，ADC 参考源选择 2V。外部参考源的限制条件为最高为 VDD，最低为内部最低电平，否则默认为 VDD。若选择 AIN8 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，不是内部 2V/3V/4V。

15.3.1 信号格式

ADC 采样电压范围为参考电压高/低电平之间。ADC 参考低电压为 VSS，高电压包括 VDD/4V/3V/2V 考电压。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。
- ADC 内部参考高电压=VDD/4V/3V/2V。

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

- ADC 参考低电压≦ ADC 采样输入信号电压≦ ADC 参考高电压

15.4 转换时间

ADC 转换时间是指从 ADS=1（开始 ADC）到 EOC=1（ADC 结束）所用的时间，由 ADC 时钟频率控制，12 位 ADC 的转换时间为 1/（ADC 时钟/4）*16 S。ADC 的时钟源为 Fosc 包括 Fosc/16、Fosc/8、Fosc/2 和 Fosc/1，由 ADCKS[1:0]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 时钟速率	fosc = 16MHz		fosc = 32MHz	
		转换时间	转换速率	转换时间	转换速率
00	fosc/16	1/(16MHz/16/4)*16 = 64us	15.625kHz	1/(32MHz/16/4)*16 = 32us	31.25kHz
01	fosc/8	1/(16MHz/8/4)*16 = 32us	31.25kHz	1/(32MHz/8/4)*16 = 16us	62.5kHz
10	fosc	1/(16MHz/4)*16 = 4us	250kHz	1/(32MHz/4)*16 = 2us	500kHz
11	fosc/2	1/(16MHz/2/4)*16 = 8us	125kHz	1/(32MHz/2/4)*16 = 4us	250kHz

15.5 数据缓存器

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

AIN 输入电压 vs. ADB 输出数据

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

15.6 ADC 寄存器

寄存器	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	-	CHS3	CHS2	CHS1	CHS0
ADB	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4
ADR	-	GCHS	ADCKS1	ADCKS0	ADB3	ADB2	ADB1	ADB0
VREFH	-	-	-	-	-	VHS2	VHS1	VHS0
P0CON	-	-	P0CON5	P0CON4	P0CON3	P0CON2	P0CON1	P0CON0
IEN2	-	-	-	-	-	EX2	ET3	EADC
IRCON2	-	-	-		-	IE2	TF3	ADCF

ADM 寄存器（0xD2）

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

*(1) AIN8 为内部 2V/3V/4V 输入通道，没有外部信号输入，ADC 参考电压必须为内部 VDD 或外部电压，不能是内部 2V/3V/4V。

ADB 寄存器（0xD3）

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

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

ADR 寄存器（0xD4）

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

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

VREFH 寄存器（0xD5）

Bit	Field	Type	Initial	说明
1..0	VHS[2:0]	R/W	00	ADC 内部参考高电压选择位。*(1) 000：2.0V； 001：3.0V； 010：4.0V； 011：保留； 100：VDD。(CHS[3：0] = 1000，AIN8=2.0V)； 101：VDD。(CHS[3：0] = 1000，AIN8=3.0V)； 110：VDD。(CHS[3：0] = 1000，AIN8=4.0V)； Others：保留。

*(1) AIN8 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。

P0CON 寄存器（0x9E）

Bit	Field	Type	Initial	说明
5..0	P0CON[7:0]	R/W	0x00	P0 配置控制位*。 0：P0 可作为模拟输入引脚（ADC 输入引脚）或者数字 GPIO 引脚； 1：P0 只能作为模拟输入引脚。
Else	Reserved	R	0	

* P0CON [5:0]配置相关的 P0 引脚为纯模拟输入引脚以避免漏电流。

IEN2 寄存器（0x9A）

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

IRCON2 寄存器（0xBF）

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

15.7 程式范例

下面的示例程序代码显示了在中断下如何设置 ADC 采集 AIN5 通道信号。

```
#define ADCAIN8_4V        (2 << 0)   //AIN8 = 4.0V
#define ADCAIN8_3V        (1 << 0)   //AIN8 = 3.0V
#define ADCAIN8_2V        (0 << 0)   //AIN8 = 2.0V
#define ADCInRefVDD       (1 << 2)   //internal reference from VDD
#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 << 1)   //enable ADC interrupt
#define ClearEOC          0xDF;

unsigned int   ADCBuffer;      // data buffer

void ADCInit(void)
{
   P0   = 0x00;
   P0M = 0x10;

   // set AIN5 pin's mode at pure analog pin
   P0CON |= 0x20;   //AIN5/P05
   P0M     &= 0xDF;      //input mode
   P0UR   &= 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 gobal interrupt
   IEN0   |= 0x80;      //enable global interrupt

   // enable ADC interrupt
   IEN2   |= EADC;

   // start ADC conversion
   ADM     |= ADCStart;
}

void ADCInterrupt(void) interrupt ISRAdc
{//ADCF clear by hardware
      P04 = ~P04;

   ADCBuffer = (ADB << 4) + (ADR & 0x0F);
   ADM &= ClearEOC;
   ADM |= ADCStart;
}
```

16 UART

UART（通用同步异步收发器）提供 1MHz 灵活的全双工传输模式。UART 共有 4 种操作模式：一种同步，3 种异步的。同步模式发送 8 位数据，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。

16.1UART 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： P03M=1 和 P03=1 URX 引脚： 发送器： P02M=1 和 P02=1 接收器： P02M=0 和 P02=1
0	1	1	异步	波特率生成器或 T1 溢出率	1	8	1	UTX 引脚： P03M=1 和 P03=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	

16.2 UART 模式 0：同步 8 位收发器

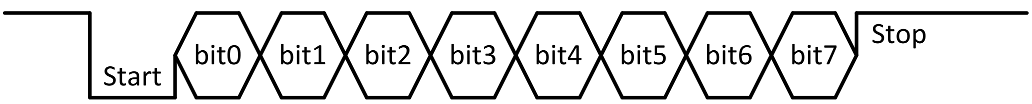
模式 0 是移位寄存器模式，作为同步收发器。UTX 引脚都输出移位时钟用于传输和接收状态。URX 引脚用于传输和接收数据。首先，发送/接收带 LSB 的 8 位数据。波特率为 Fcpu/12。通过写数据到 S0BUF 寄存器开始传输。第 8 位传输结束后，设置 TI0 标志位。通过设置 REN0 位来控制接收，清 RI0 位。REN0=1 和 RI0 由 1 到 0 时，开始传输数据，且在接收完第 8 位时，设置 RI0 标志位。

16.3 UART 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。

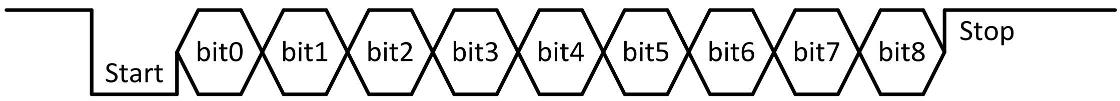


16.4 UART 模式 2：固定波特率异步 9 位收发器

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。

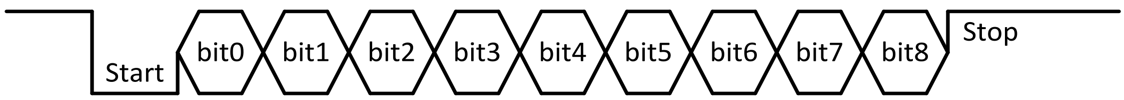


16.5 UART 模式 3：可变波特率异步 9 位收发器

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 = 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。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。



16.6 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。

16.7 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：

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

UART 波特率的设置建议如下表设置（Fcpu=32MHz）：

波特率	SMOD	S0RELH	S0RELL	Accuracy
4800	0	0x03	0x98	-0.16 %
9600	0	0x03	0xCC	-0.16 %
19200	0	0x03	0xE6	-0.16 %
38400	0	0x03	0xF3	-0.16 %
56000	1	0x03	0xEE	0.79 %
57600	1	0x03	0xEF	-2.12 %
115200	1	0x03	0xF7	3.55 %
128000	1	0x03	0xF8	2.34 %
250000	1	0x03	0xFC	0 %

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

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

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

波特率	SMOD	Timer Period	TH1/TL1	Accuracy
4800	0	6.510 us	0x30	0.16 %
9600	1	6.510 us	0x30	0.16 %
19200	1	3.255 us	0x98	0.16 %
38400	1	1.628 us	0xCC	0.16 %
56000	1	1.116 us	0xDC	-0.80 %
57600	1	1.085 us	0xDD	-0.80 %
115200	1	0.543 us	0xEF	2.08 %
128000	1	0.488 us	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。***

16.8 省电

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

16.9 UART 寄存器

寄存器	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	-	-	ES0	ET1	EX1	ET0	EX0
P0OC	-	-	P05OC	P04OC	P03OC	P02OC	P01OC	P00OC
P0M	-	-	P05M	P04M	P03M	P02M	P01M	P00M
P0	-	-	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。

*REN0=0 时，不能访问 UART 相关寄存器，模块内部时钟停止工作。

> ✱ **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
3	P03OC	R/W	0	0：切换 P0.3（UTX）为输入模式（要求）; 1：切换 P0.3（UTX）为开漏模式。
2	P02OC	R/W	0	0：切换 P0.2（URX）为推拉模式; 1：切换 P0.2（URX）为开漏模式。
Else				请参考其它章节。

*设置 P02OC 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
3	P03M	R/W	0	0：设置 P0.3（UTX）为输入模式（要求）; 1：设置 P0.3（UTX）为输出模式。
2	P02M	R/W	0	0：设置 P0.3（URX）为输入模式; 1：设置 P0.2（URX）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
3	P03	R/W	0	0：设置 P0.3（UTX）始终为低电平*; 1：P0.3（UTX）输出 UART 数据（要求）。
2	P02	R/W	0	随时读取该位以监控总线状态。
Else				请参考其它章节。

* 初始化 P03 时一定需要设置为高电平，因为 P03 只能输出低电平驱动共用引脚。

16.10 程式范例

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
   P03 = 1;
   P0M = P0M | 0x08 & ~0x04;

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

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

   // enable UART interrupt
   IEN0 |= SYSUartES0;

   // send first UTX data
   S0BUF = uartTxBuf;
}

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

17 In-System Program

SN8F5701 内置 4KB 程序存储（IROM），均分为 128 页（每页 32 字节）。In-System Program 就是使用软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软件的一个通道。

地址	程序存储器 IROM
0x0FFF 0x0FE0	**Page 127**
0x0FDF 0x0FC0	**Page 126**
	…
0x005F 0x0040	**Page 2**
0x003F 0x0020	**Page 1**
0x001F 0x0000	**Page 0**

程序存储器 IROM

17.1 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
	0300	一个 ISP 程序页
	0320	一个 ISP 程序页
	…	一个 ISP 程序页
	0700	一个 ISP 程序页
	0720	一个 ISP 程序页
	…	一个 ISP 程序页
	0FE0	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列配置设置完成：
1、保存编程数据到 IRAM，这些数据连续 32 字节。
2、在 PERAM 中设置数据位置的起始地址。
3、在 PEROM[15:5]中设置计划升级区域的起始地址（由 PEROMH/PEROML 寄存器设置）。
4、写入 0x5A 到 PECMD[7:0]开始 ISP 功能。
5、写入两次 NOP 指令。

举例来说，假设程序存储器的第 126 页（IROM，0xFC0~0xFDF）为计划升级区域，已经存入 IRAM 地址 0x60~0x7F 的内容，执行 In-System Program，只要写入开始 IROM 地址 0FC0H 到 EPROMH/EPROML 寄存器，然后指定缓存器开始地址 0x60 到 EPRAM 寄存器，随后写入 0x5A 到 PECMD[7:0]寄存器，复制缓存器的数据到 IROM 的第 126 页。

通常而言，每页的内容都可通过 In-System Program 进行修改，但是前 10 页和最后一页都不可以编程，否则会导致上电错误或复位错误。

17.2 字节编程

字节编程支持 1 个字节的存储编程，字节编程流程需要 1 个自动的 IRAM 作为它自己的数据缓存器。

在开始页编程之前必须将下列配置设置完成：
1、保存编程数据到 IRAM，数据仅为 1 个字节。
2、在 PERAM 中设置数据位置的起始地址。
3、在 PEROM[15:5]中设置计划升级区域的起始地址（由 PEROMH/PEROML 寄存器设置）。
4、写入 0x1E 到 PECMD[7:0]开始 ISP 功能。

举例来说，假设 IRPM 的地址 0x0FC5 为计划升级区域，已经寻人 IRAM 地址 0x60 的内容，执行 In-System Program，只要写入 IROM 开始地址 0x0FC5 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 1EH 到 PECM[7:0]寄存器，复制缓存器的数据到 IROM 的地址 0x0FC5 处。

✸ ***注：***
1. ***在 Flash 写（编程）操作前，必须清看门狗，否则在 ISP 运行过程中，看门狗定时器会溢出并导致系统复位；***
2. ***第一页和最后一页不要执行 ISP FLASH ROM，否则影响编程操作；***
3. ***ISP 操作(页编程)实际为后台执行 Flash 擦除和编程进程，禁止在低电压(例如 VDD<2.5V）执行 ISP Flash ROM 编程，否则可能导致掉电前 ISP 操作未完成。***

17.3 In-System Program 寄存器

寄存器	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PERAM	PERAM7	PERAM6	PERAM5	PERAM4	PERAM3	PERAM2	PERAM1	PERAM0
PEROMH	PEROM15	PEROM14	PEROM13	PEROM12	PEROM11	PEROM10	PEROM9	PEROM8
PEROML	PEROM7	PEROM6	PEROM5	PEROM4	PEROM3	PEROM2	PEROM1	PEROM0
PECMD	PECMD7	PECMD6	PECMD5	PECMD4	PECMD3	PECMD2	PECMD1	PECMD0

PERAM 寄存器（0x97）

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

PEROMH 寄存器（0x96）

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

PEROML 寄存器（0x95）

Bit	Field	Type	Initial	说明
7..0	EPRAM[7:0]	R/W	000	编程页（IROM）的地址（7^{th}– 0^{th} bit）。

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； 0x1E：开始字节编程； ~~其它值：保留*。~~

*不允许写入其它值到 PECMD 寄存器。

17.4 程式范例

```
unsigned cahr idata 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);    //Page program
}

void SYSByteIspStart(unsigned int byteAddress)
{
    BISP(byteAddress, 0xE0);    //Byte program
}
```

18 时钟微调

SN8F5701 内置时钟微调功能，可以通过软件对系统时钟频率进行微调，由 Code Option（CK_Fine_Tuning）使能控制。CK_Fine_Tuning=0 时，禁止时钟微调功能；CK_Fine_Tuning=1 时，使能时钟微调功能。系统上电后，由硬件将 10 位初始时钟调整值装入 FRQ[9:0]缓存器中。其调整值就相当于 IHRC 32MHz，更改 FRQ[9:0]的调整值，就可以调整时钟频率。

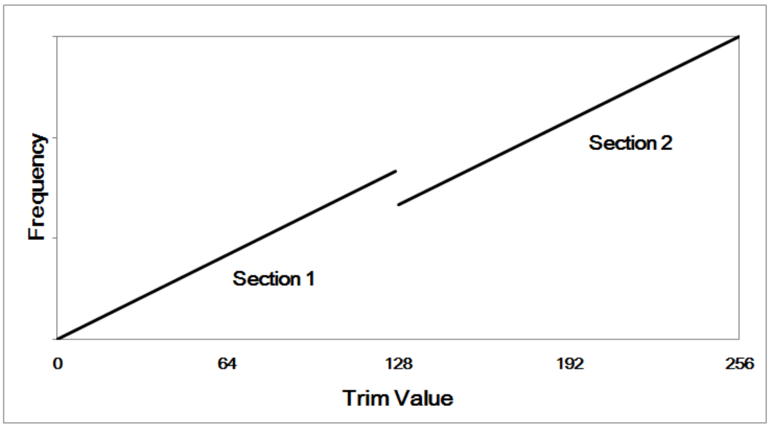
18.1 时钟调整

时钟微调包含 8 个调整段，如下表所示。每段包含 128 调整 step，在同一个调整段中，每个 step 约 32MHz*0.1％。调整值越大，频率越高。

表 17-1 时钟调整值表

Section	Trim Value	频率
1	000H ~ 07FH	低
2	080H ~ 0FFH	
3	100H ~ 17FH	
4	180H ~ 1FFH	
5	200H ~ 27FH	
6	280H ~ 2FFH	
7	300H ~ 37FH	
8	380H ~ 3FFH	高

每相邻的调整段之间都有一个频率间隙，如下图所示。因此，调整值为 127 时的频率要比调整值为 128 时高。



18.2 时钟微调步骤

在开始时钟微调功能之前，必须完成相关设置，具体步骤如下：
1、选择 Code Option CK_Fine_Tuning=1，使能设置微调功能；
2、IROM 取周期的最大值为 8MHz，建议先设置 RWSC[2:0]=7 以避免系统出错；
3、从 FRQ[9:0]中读取 10 位 32MHz 调整值；
4、从上表 17-1 中检测微调的范围；
5、写入新的时钟调整值到 FRQ[9:0]中；
6、写入 3CH 到 FRQCMD[7:0]中，触发时钟微调功能。

> ✱ ***注：请检查 IROM 取周期<8MHz，以避免系统出错。***

18.3 时钟微调寄存器

寄存器	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FRQH	-	-	-	-	-	-	FRQ9	FRQ8
FRQL	FRQ7	FRQ6	FRQ5	FRQ4	FRQ3	FRQ2	FRQ1	FRQ0
FRQCMD	FRQCMD7	FRQCMD6	FRQCMD5	FRQCMD4	FRQCMD3	FRQCMD2	FRQCMD1	FRQCMD0

FRQ 寄存器（FRQH:0xF3, FRQL:0xF2）

Bit	Field	Type	Initial	说明
9..0	FRQ[9:0]	R/W	0x00	系统时钟校准值。

FRQCMD FRQ 寄存器（0xF4）

Bit	Field	Type	Initial	说明
7..0	FRQCMD[7:0]	W	0x00	0x3C：开始时钟微调功能； 其它值：保留*(1) 。

*(1)不允许写入其它值到 FRQCMD 寄存器中。

18.4 程式范例

下面的示例程序代码显示了如何执行时钟微调功能。

```
FRQH = 0x01;
FRQL = 0x59;          // Set FRQ[9:0]= 0x159
FRQCMD = 0x3C;        // Apply CLKSEL's setting
```

19 电气特性

19.1 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

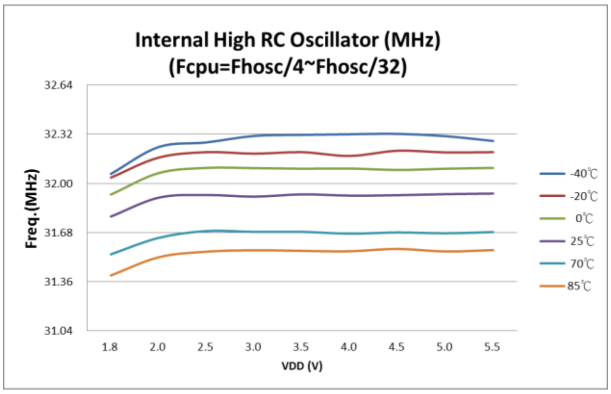
19.2 系统操作特性

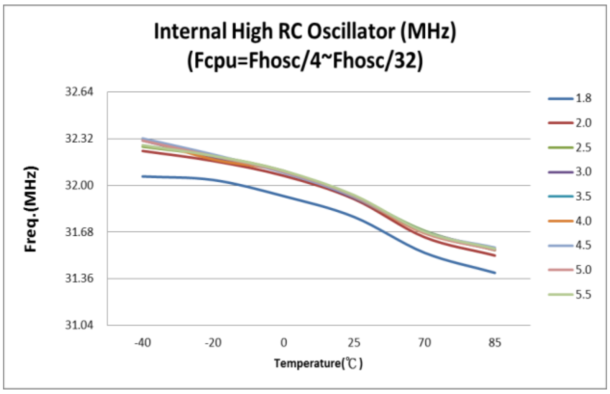
参数		测试条件	Min	TYP	MAX	单位
VDD	运行电压	Fcpu = 1MHz	1.8		5.5	V
V_{DR}	RAM 数据保持电压		1.5			V
V_{POR}	VDD 上升速率*		0.05			V/ms
I_{DD1}	NORMAL 模式供电电流	VDD = 3V, fcpu = 1MHz		2.25		mA
		VDD = 5V, fcpu = 1MHz		2.30		mA
		VDD = 3V, fcpu = 4MHz		2.60		mA
		VDD = 5V, fcpu = 4MHz		2.65		mA
		VDD = 3V, fcpu = 8MHz		3.10		mA
		VDD = 5V, fcpu = 8MHz		3.15		mA
I_{DD2}	停止模式供电电流	VDD = 3V		3.0	8.5	μA
		VDD = 5V		3.5	9.0	μA
I_{DD3}	IDLE 模式供电电流 (fcpu = 1MHz)	VDD = 3V, 32MHz IHRC		0.55		mA
		VDD = 5V, 32MHz IHRC		0.60		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, -40°C to 85°C	31.36	32	32.64	MHz
F_{ILRC}	内部低时钟发生器	VDD = 5V, 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
V_{ESD_HBM}	ESD human body mode		4000			V
V_{ESD_MM}	ESD machine mode		300			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

IHRC 图用于设计指导，没有经过测试或保证。在某些图中，所显示的数据超出了指定的工作范围。这仅供参考，并保证设备只能在指定范围内正常运行。





19.3 GPIO 特性

参数		测试条件	Min	TYP	MAX	单位
V_{IL1}	低电平输入电压(P00)		VSS		0.1VDD	V
V_{IH1}	高电平输入电压(P00)		0.5VDD		VDD	V
V_{IL2}	低电平输入电压(P01- P05)		VSS		0.3VDD	V
V_{IH2}	高电平输入电压(P01- P05)		0.7VDD		VDD	V
I_{LEKG}	I/O 口输入漏电流	VIN = VDD			2	μA
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, VO = VDD-0.5V	12	16		mA
I_{OL}	I/O sink 电流	VDD = 5V, VO = VSS+0.5V	15	20		mA

*环境温度为 25°C。

19.4 ADC 特性

参数		测试条件	Min	TYP	MAX	单位
V_{ADC}	运行电压		2.0		5.5	V
V_{AIN}	AIN 通道输入电压	VDD = 5V	0		VREFH	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		0.45		mA
		VDD = 5V		0.50		mA
f_{ADCLK}	ADC 时钟	VDD = 5V			32	MHz
f_{ADSMP}	ADC 取样率	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		±8		LSB
INL	积分非线性*	f ADSMP = 62.5kHz		±2		LSB
		f ADSMP = 250kHz		±2.5		LSB
		f ADSMP = 500kHz		±8		LSB
NMC	无遗漏码*	f ADSMP = 62.5kHz	10	11	12	Bit
		f ADSMP = 250kHz		11		Bit
		f ADSMP = 500kHz		8		Bit
V_{OFFSET}	输入偏移电压	Non-trimmed	-10	0	10	mV

* 带*号的参数：VDD = 3V，VREFH = 3V，25°C。

19.5 Flash 存储器特性

参数		测试条件	Min	TYP	MAX	单位
Vdd	供电电压		1.8		5.5	V
Ten	持续时间	25°C		*100K		cycle
Iwrt	写电流	25°C		3	4	mA
Twrt	写时间	写 1 页=32 字节，25°C		6	8	ms

*带*号的参数还未经过验证的设计参考值。

20 指令集

本章对 SN8F5701 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

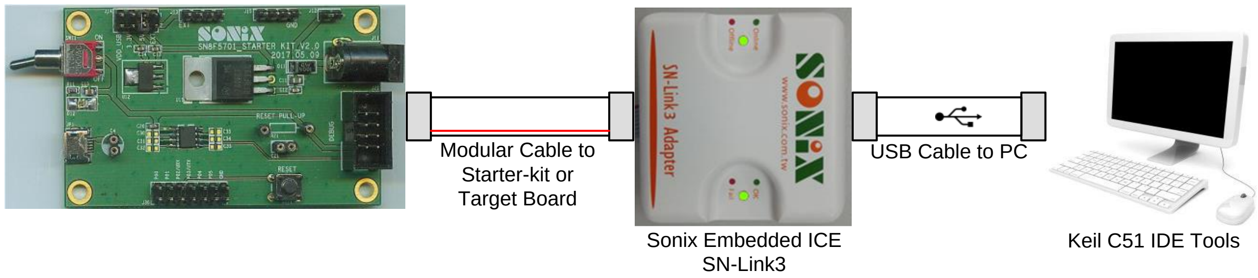
助记符	说明
CLR C	清除进位标志
CLR bit	清除直接寻址位
SETB C	设置进位标志
SETB bit	设置直接寻址位
CPL C	标志位取反
CPL bit	直接寻址位取反
ANL C, bit	直接寻址位与进位标志
ANL C, /bit	直接寻址位取反与进位
ORL C, bit	直接寻址位或进位标志
ORL C, /bit	直接寻址位取反或进位
MOV C, bit	传送直接寻址位至进位标志
MOV bit, C	传送进位标志至直接寻址位

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

21 调试界面

SONIX 提供嵌入式 ICE 仿真器系统，用于 SN8F5701 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、SN8F5701 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 SN8F5701 中，可提供真正的开发环境。



21.1 最低要求

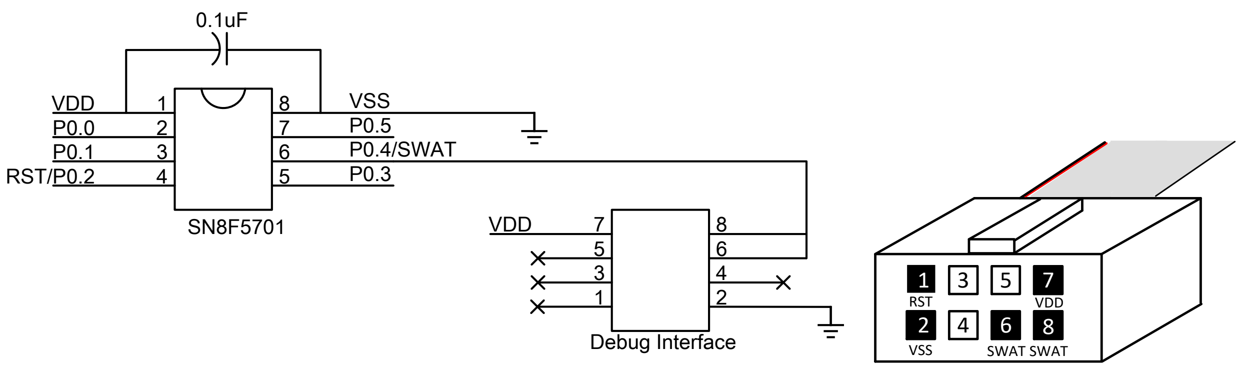
下面几项内容是建立合适的开发环境的基本要求，其兼容性已经经过验证，在后面的版本中可以很好的执行。SN-Link 的相关信息可以 SONiX 网址 www.sonix.com.tw 下载，Keil C51 可从 www.keil.com/c51 下载。

- **SN-Link3 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。

21.2 调试界面硬件

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

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

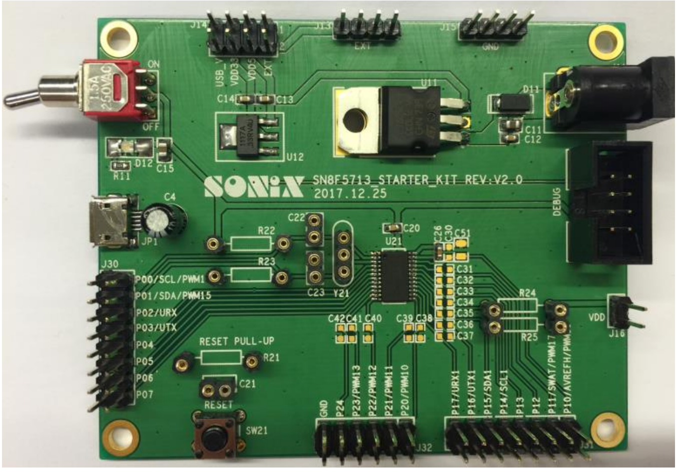


Example circuit　　SN-Link header

21.3 开发工具

SN-Link3 适配器

Starter-Kit 支持 SN8F5701, SN8F57011



MP5 烧录器

22 SN8F5701 Start-Kit

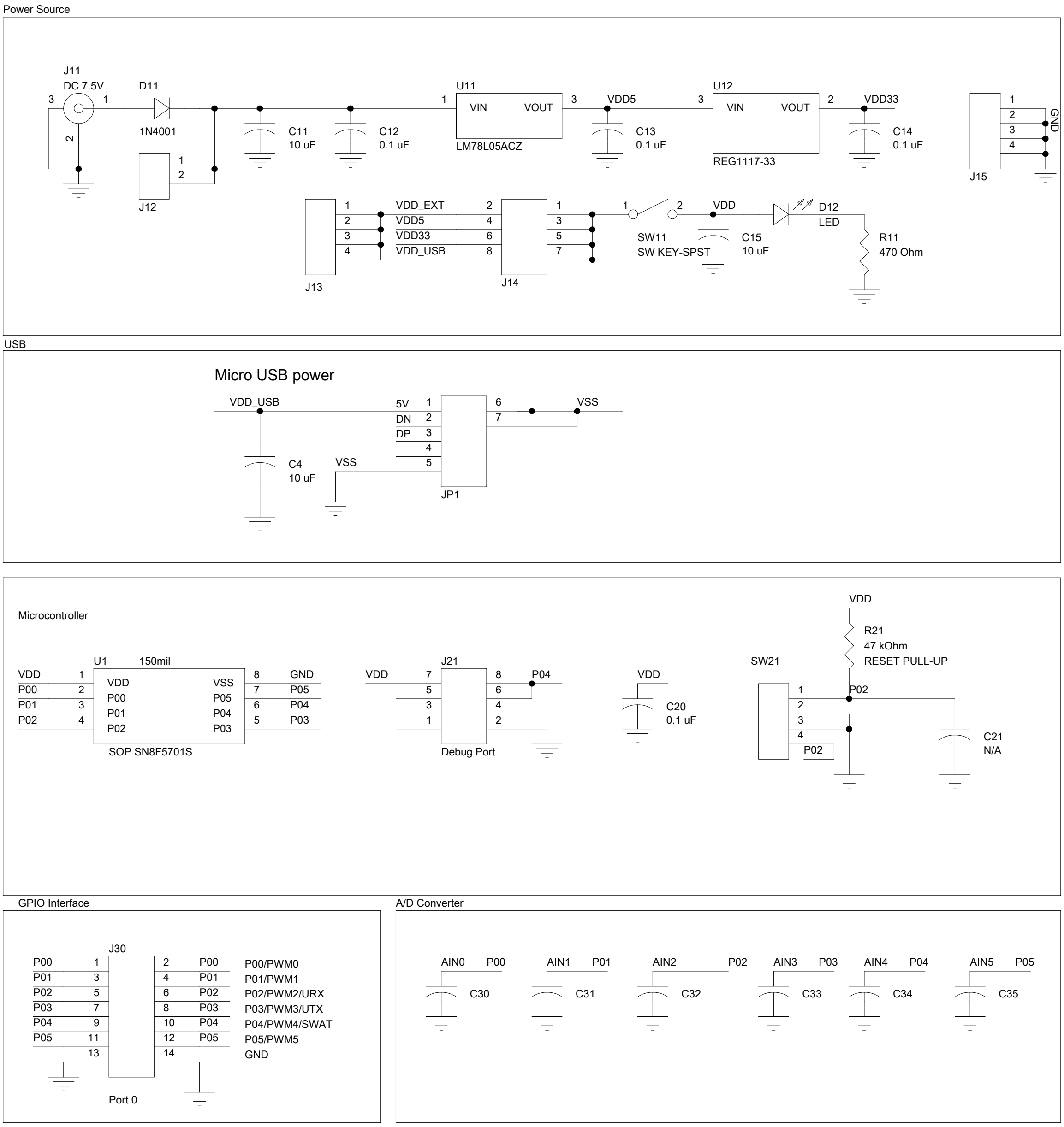
SN8F5000 Starter-Kit 提供一个简易的开发平台。它包含 SN8F5000 系列 IC 和输入信号或用户应用程序驱动设备的 IO 接口。当目标板没有准备好时，它是一个简单开发平台用于应用程序开发。Starer-Kit 可以被目标板所取代，因为 SN8F5000 系列集成了嵌入式的电路调试器电路。

22.1配置电路

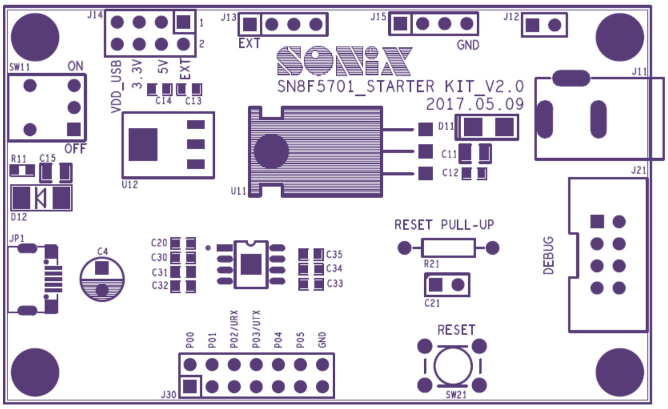
在开始利用 Start-Kit 开发前必须完成以下配置：

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. 调试端口可以连接 SN-LINK 适配器进行仿真或下载代码；
7. 当接通电源 VDD 时，MCU　LED 灯会亮起来，SN8F5000 系列 IC 将上电。

22.2电路


Power Source
J11
DC 7.5V
D11
1N4001
J12
C11
10 uF
C12
0.1 uF
U11
VIN VOUT
LM78L05ACZ
VDD5
C13
0.1 uF
U12
VIN VOUT
REG1117-33
VDD33
C14
0.1 uF
J15
GND
J13
VDD_EXT
VDD5
VDD33
VDD_USB
J14
SW11
SW KEY-SPST
VDD
C15
10 uF
D12
LED
R11
470 Ohm
USB
Micro USB power
VDD_USB
5V
DN
DP
VSS
C4
10 uF
JP1
Microcontroller
U1
150mil
VDD P00 P01 P02
VSS P05 P04 P03
GND P05 P04 P03
SOP SN8F5701S
J21
P04
Debug Port
C20
0.1 uF
R21
47 kOhm
RESET PULL-UP
SW21
P02
C21
N/A
GPIO Interface
J30
P00/PWM0
P01/PWM1
P02/PWM2/URX
P03/PWM3/UTX
P04/PWM4/SWAT
P05/PWM5
GND
Port 0
A/D Converter
AIN0 P00 C30
AIN1 P01 C31
AIN2 P02 C32
AIN3 P03 C33
AIN4 P04 C34
AIN5 P05 C35

22.3PCB 布局平面图



22.4元件描述

符号	功能描述
C30-C35	10 通道 ADC 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	IO 口
J33	T2 捕获器电容
R21，C21	外部复位上拉电阻和电容
SW11	目标电源开关
U1	SN8F5701S 实际芯片
JP1	Micro USB 口

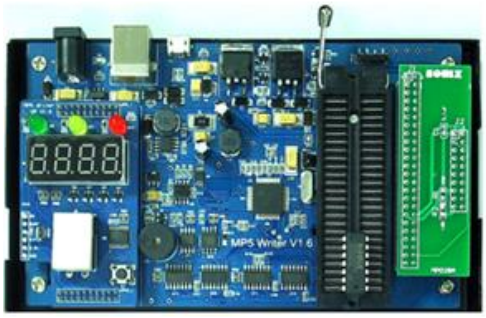
23 ROM 烧录引脚

SN-LINK 和 MP5 Writer 支持 SN8F5701 系列 Flash ROM 的擦除/烧录/校正。

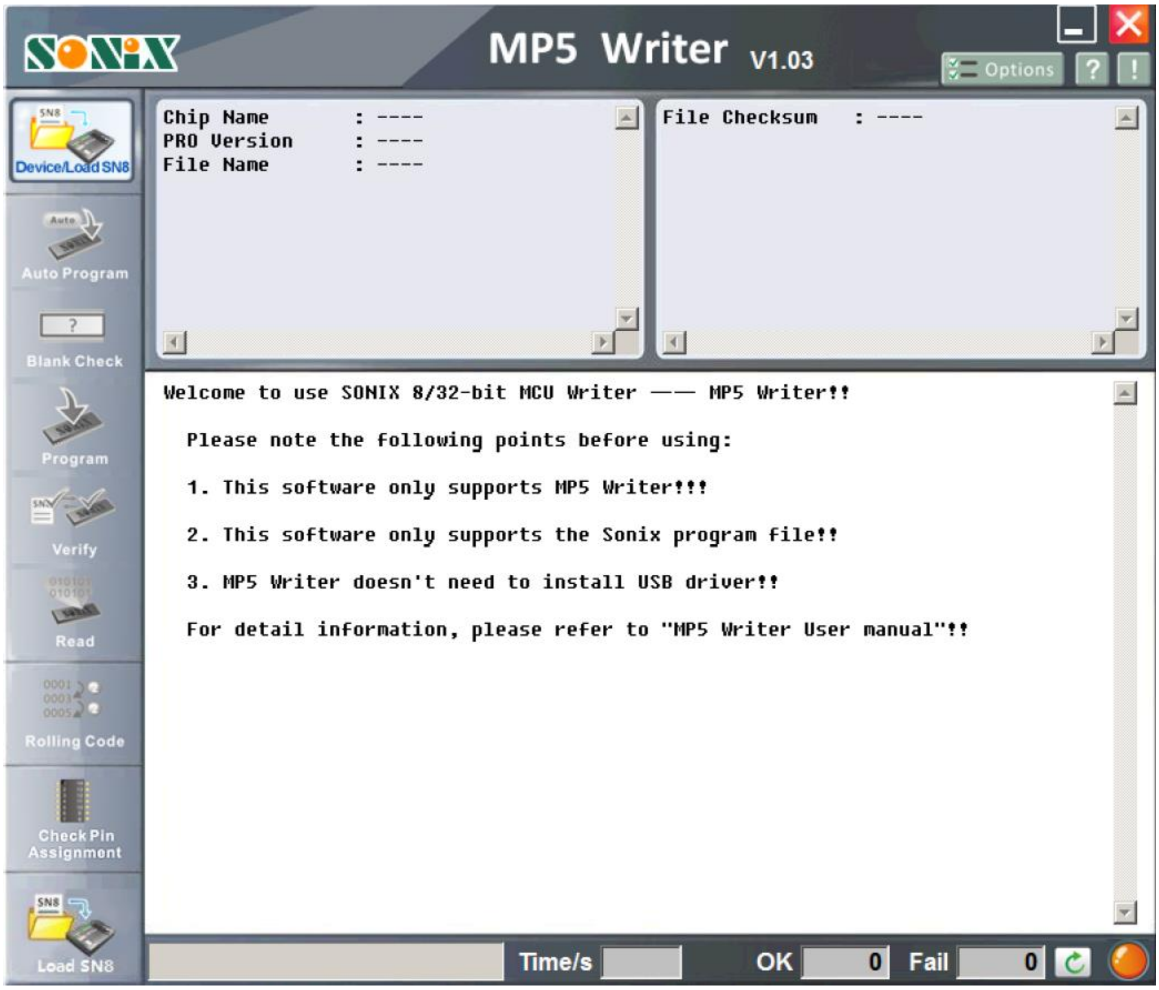
- **SN-LINK：调试界面。**
- **MP5 烧录器：SN8F5701 系列。**

23.1 MP5 硬件连接

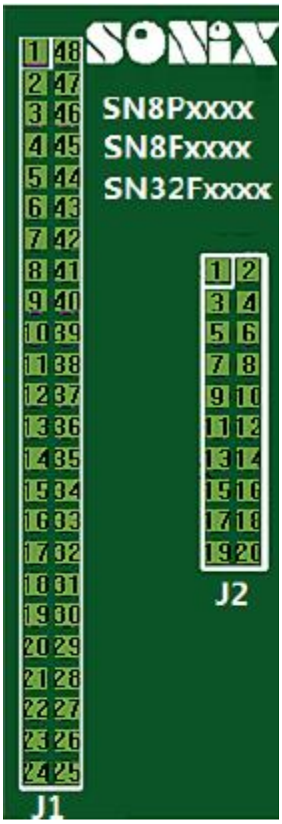
不同封装 MCU 烧录时的硬件接法如下，DIP，SOP 说明。



MP5 软件操作界面如下:



23.2 MP5 Writer 转接板引脚配置



23.3 MP5 Writer 烧录引脚配置

烧录器连接器		MCU 引脚名称	SN8F5701PG/SG/TG/MG		SN8F57011DG	
J2 引脚编号	J2 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	1	21	6	27
2	GND	VSS	8	28	2	23
7	SWAT	P0.4	6	26	1	22
9	SWAT	P0.4	6	26	1	22
20	PDB	P0.3	5	25	3	24

23.4 SN-LINK ISP 烧录

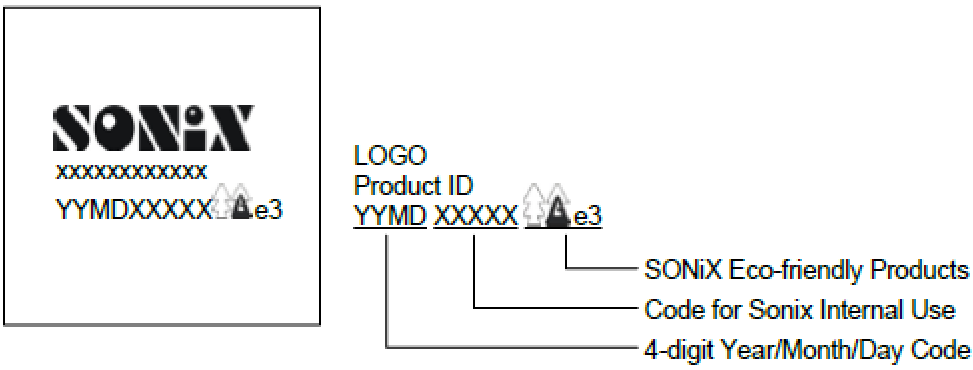
SN-Link ISP 烧录硬件和软件如下。

23.5 SN-LINK ISP 烧录引脚配置

SN-Link 连接器		MCU 引脚名称	SN8F5701PG/SG/TG/MG	SN8F57011DG
引脚编号	引脚名称		MCU 引脚编号	MCU 引脚编号
7	VDD	VDD	1	6
2	GND	VSS	8	2
6	SWAT	P0.4	6	1
8	SWAT	P0.4	6	1

24 订购信息

SONiX 的单片机的表面印有三栏信息：商标，单片机全称和日期码。



24.1 命名规则

全名	封装类型
SN8F5701W	Wafer
SN8F5701H	Dice
SN8F5701PG	PDIP，8 脚，绿色封装
SN8F5701SG	SOP，8 脚，绿色封装
SN8F5701TG	TSSOP，8 脚，绿色封装
SN8F5701MGR	DFN8L，8 脚，TR 绿色封装
SN8F57011DGR	SOT23-6L，6 脚，TR 绿色封装

注： TR 包装（卷带&编带包装）在封装名称后加“R”字符。

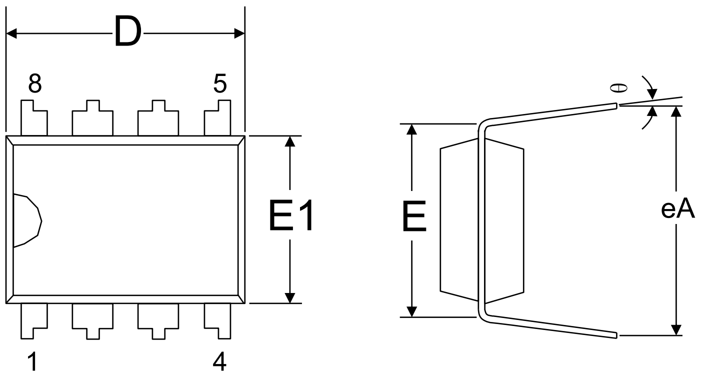
24.2 日期码

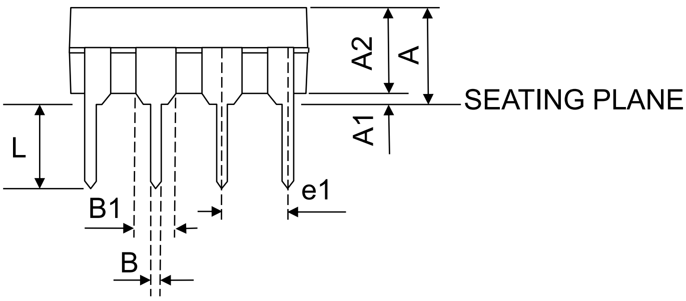
日期码包括 2 个信息：生产日期和产品序列号。生产日期是公共信息，详见下表。

Year	18：2018 19：2019 20：2020 21：2021 et cetera
Month	1：January 2：February 3：March 4：April 5：May 6：June 7：July 8：August 9：September A：October C：November D：December
Date	1：01 2：02 3：03 4：04 5：05 6：06 7：07 8：08 9：09 A：10 C：11 D：12 E：13 F：14 G：15 H：16 J：17 K：18 L：19 M：20 N：21 P：22 Q：23 R：24 S：25 T：26 U：27 V：28 W：29 X：30 Y：31

25 封装信息

25.1 P-DIP8

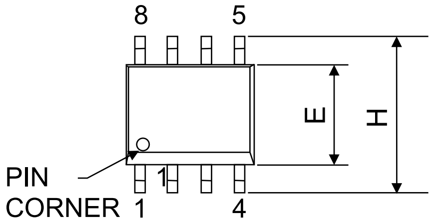




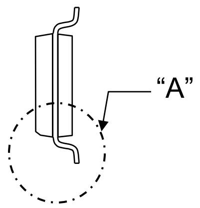
SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	5.33	--	--	0.210
A1	0.38	--	--	0.015	--	--
A2	3.18	3.30	3.43	0.125	0.130	0.135
B	0.46 typ.			0.018 typ.		
B1	1.52 typ.			0.060 typ.		
D	9.02	9.27	10.16	0.355	0.365	0.400
E	7.62 BSC.			0.300 BSC		
E1	6.22	6.35	6.47	0.245	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°

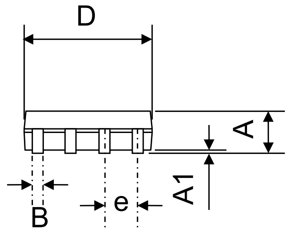
Notes：
1. JEDEC OUTLINE : MS-001 BA
2. CONTROLLING DIMENSION：inch

25.2 SOP8

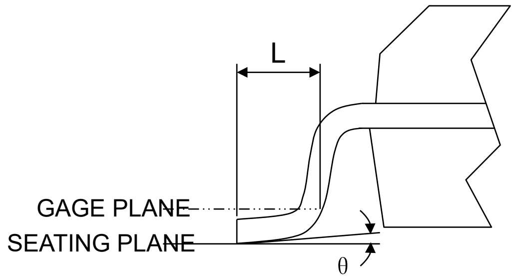


TOP VIEW





SIDE VIEW



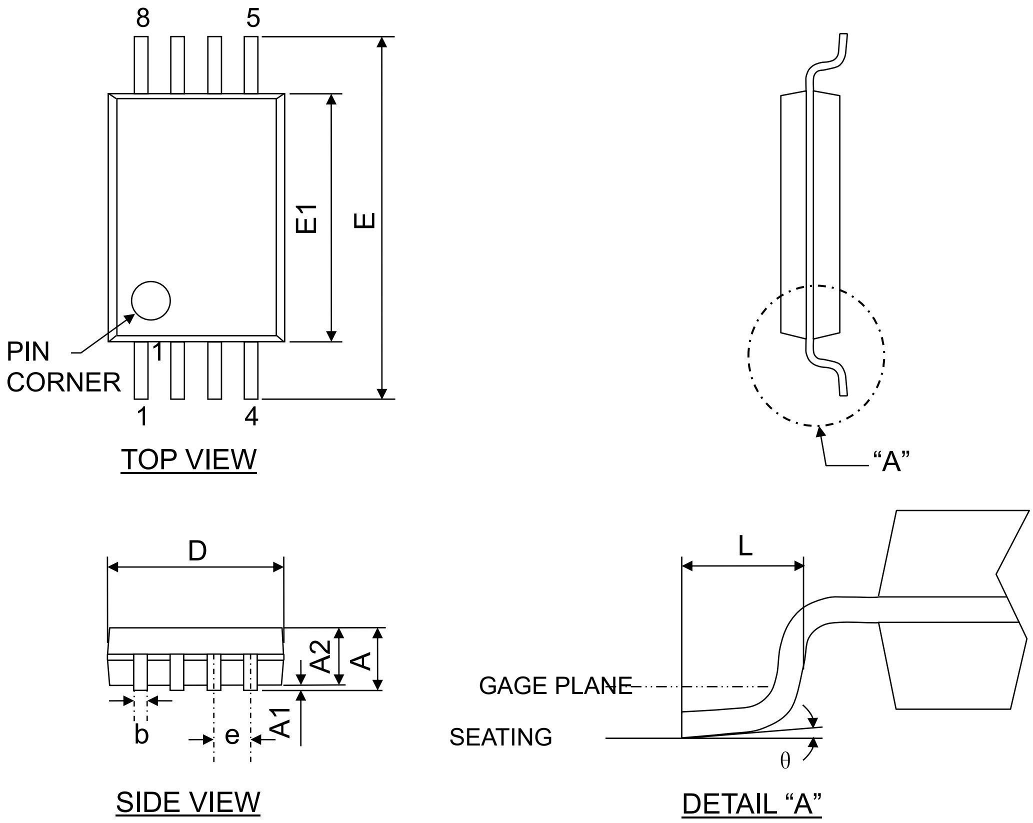
DETAIL “A”

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	--	0.51	0.012	--	0.020
D	4.90 BSC			0.193 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°	--	8°	0°	--	8°

Notes：

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

25.3 TSSOP8

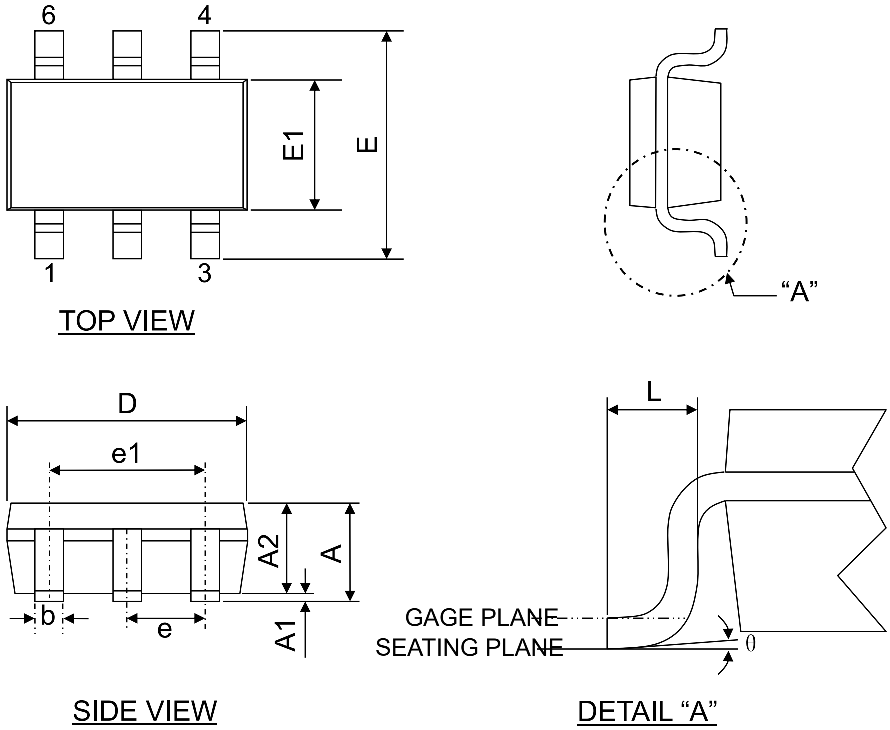


SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	1.20	--	--	0.047
A1	0.05	--	0.15	0.002	--	0.006
A2	0.80	1.00	1.05	0.031	0.039	0.041
b		0.22 typ.			0.009 typ.	
D	2.90	3.00	3.10	0.114	0.118	0.122
E		6.40 BSC.			0.252 BSC.	
E1	4.30	4.40	4.50	0.169	0.173	0.177
e		0.65 typ.			0.026 BSC.	
L	0.45	--	0.75	0.018	--	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.

25.4 SOT23-6L

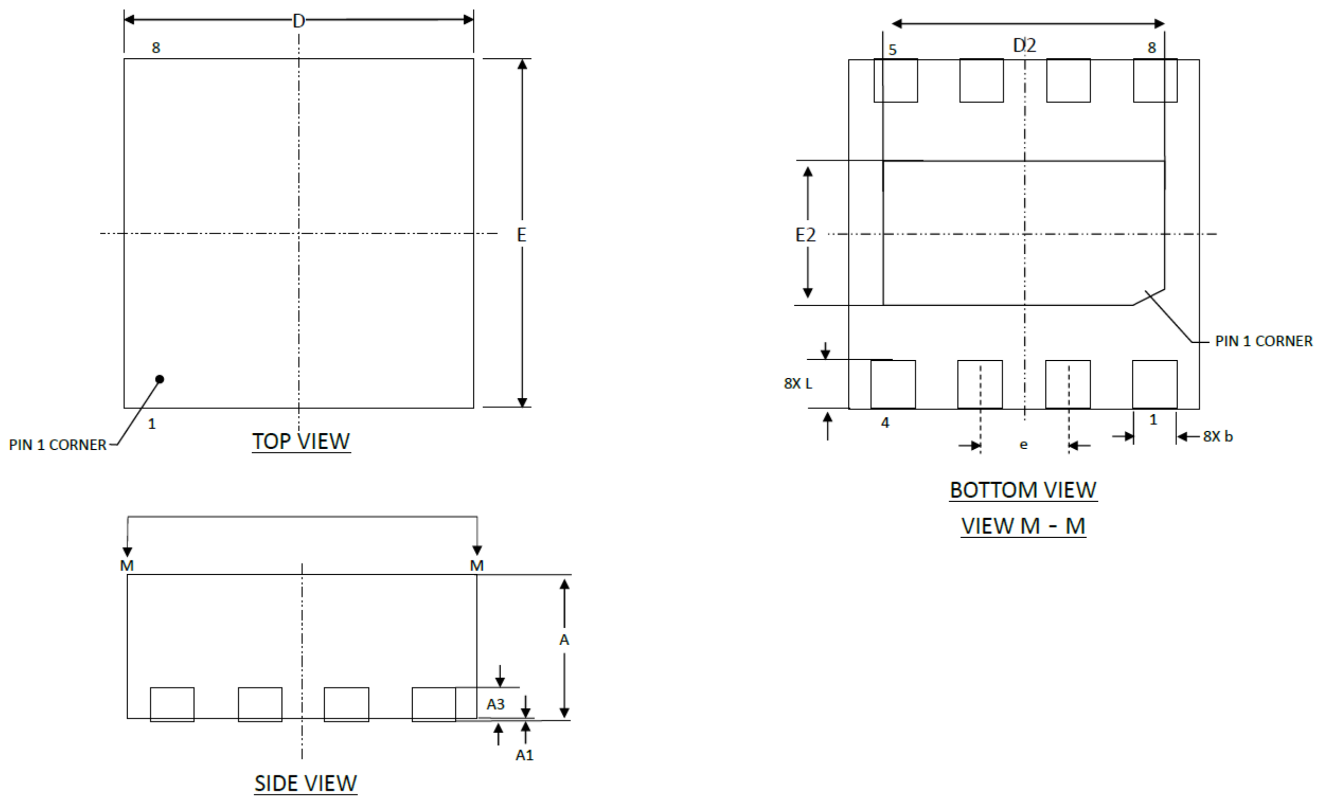


SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	1.45	--	--	0.057
A1	0.00	--	0.15	0.000	--	0.006
A2	0.90	1.15	1.30	0.035	0.045	0.051
b	0.30	--	0.50	0.012	--	0.020
D	2.90 BSC.			0.114 BSC.		
E	2.80 BSC.			0.110 BSC.		
E1	1.60 BSC.			0.063 BSC.		
e	0.95 BSC.			0.037 BSC.		
e1	1.90 BSC.			0.075 BSC.		
L	0.30	--	0.60	0.012	--	0.024
θ	0°	--	8°	0°	--	8°

Notes：

1. CONTROLLING DIMENSION：mm
2. JEDEC OUTLINE : MS-178 AB

25.5 DFN8L



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	0.20REF			0.008REF		
b	0.20	0.25	0.30	0.007	0.010	0.012
D	2.00 BSC.			0.078 BSC.		
E	2.00 BSC.			0.078 BSC.		
e	0.50 BSC.			0.020 BSC.		
D2	1.50	1.60	1.70	0.060	0.063	0.067
E2	0.75	0.85	0.95	0.030	0.033	0.037
L	0.25	0.30	0.35	0.010	0.011	0.013

Notes：

1. CONTROLLING DIMENSION：mm

26 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 系列的性能。(从 SONIX 官网：www.sonix.com.tw 下载)

文档名称	文档说明
SN8F5000 Starter-Kit 用户手册	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 系列指令集	该文档详细介绍 8051 的指令集，并进行简单示例说明。
SN8F5000 系列指令对应表	该文档介绍 8-bit Flash/OTP Type 与 8051 Flash Type 相对应的指令。
SN8F5000 封装信息	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 调试工具手册	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。

总公司：
地址：台湾新竹县竹北市台元街 36 号 10 楼之一
电话：886-3-5600-888
传真：886-3-5600-889
台北办事处：
地址：台北市松德路 171 号 15 楼之 2
电话：886-2-2759 1980
传真：886-2-2759 8180
香港办事处：
地址：香港新界沙田炭禾盛街 11 号，中建电讯大厦 26 楼 03 室
电话：852-2723 8086
传真：852-2723 9179
松翰科技（深圳）有限公司
地址：深圳市宝安区新安二路中粮紫云大厦 26F
电话：86-755-2671 9666
传真：86-755-2671 9786
技术支持：
Sn8fae@SONiX.com.tw