SN8F27E60 系列

用户参考手册
Version 1.2

SN8F27E65
SN8F27E64
SN8F27E62

SN8F27E65L
SN8F27E64L
SN8F27E62L

SONiX 8 位单片机

修正记录

版本	时间	修正记录
VER 1.0	2010/03	初版。
VER 1.1	2010/08	1．修改电气特性章节的部分内容。 2．修改 MSP 章节的部分内容。 3．调整 QFN 4*4 封装的部分尺寸。
	2011/07	1、修改 ROM 的烧录引脚。 2、修改 QFN 4*4 封装的尺寸表格。 3、SN8F27E64 和 SN8F26E62 增加 AVREFH 引脚。 4、修改 SN8F27E65 Starter Kit 的内容。 5、增加 SDIP 的封装形式。
VER 1.2	2012/06	增加 SN8F27E65 Starter Kit 的原理图。

目 录

1 产品简介

8 位单片机 SN8F27E60 系列是一款全新的产品，利用先进的半导体技术设计的 Flash ROM 结构。通过 Flash ROM 平台，SN8F27E60 系列内置在线烧录（ISP）功能，可扩展为 EEPROM 应用和嵌入式 ICE 功能。SN8F27E60 系列提供高性能的 12 通道 10 位 ADC，3 组独立的可编程的 PWM， 3 种串行接口和灵活的操作模式。强大的功能、高可靠性和低功耗可以在 AC 场合和电池场合轻松应用。

1.1 功能特性

- **存储器配置**
  Flash ROM：6K x 16 位。包括 EEROM 仿真（在线烧录）
  RAM：512 x 8 位。

- **8 层堆栈缓存器**
- **13 个中断源**
  11 个内部中断：T0、TC0、TC1、TC2、T1、ADC、SIO、MSP、UTX（UART TX）、URX（UART RX）、 WAKE。
  2 个外部中断：INT0、INT1。
- **多中断向量结构**
  每个中断源都对应有一个独立的中断向量。

- **I/O 引脚配置**
  双向输入输出端口：P0、P1、P4、P5。
  唤醒功能端口：P0、P1 电平变换。
  具有上拉电阻的端口：P0、P1、P4、P5。
  外部中断触发引脚：P0.0、P0.1。
  ADC 输入引脚：AIN0~AIN11。

- **Fcpu（指令周期）**
  Fcpu = Fhosc/1、Fhosc/2、Fhosc/4、Fhosc/8、Fhosc/16、Fhosc/32、Fhosc/64、Fhosc/128。
- **内置看门狗定时器和时钟源**
- **1.8V/2.4V/3.3V 3 层 LVD**

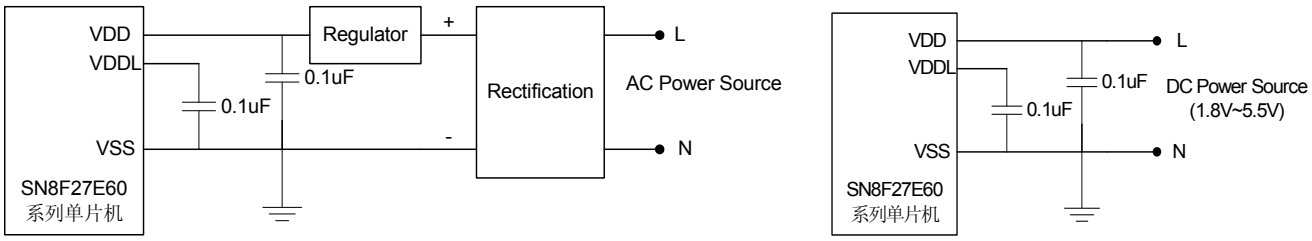
- **功能强大的指令集**
  指令的长度为一个字长。
  大多数指令的周期为一个时钟周期。
  JMP 指令可在整个 ROM 区运行。
  查表指令（MOVC）可寻址整个 ROM 区。

- **4 个 8 位定时器（T0、TC0、TC1、TC2）**
  T0：基本定时器。
  TC0：定时器/计数器/PWM0。
  TC1：定时器/计数器/PWM1。
  TC2：定时器/计数器/PWM2。
- **3 通道占空比/周期可编程控制的 PWM 可用作普通**
- **PWM，Buzzer 和 IR 载波信号（PWM0~2）**
- **1 个 16 位定时器（T1），具有捕捉定时器功能**
- **12 通道 10 位 SAR ADC**
- **串行接口：SIO，UART，MSP**
- **内置嵌入式 ICE 功能**

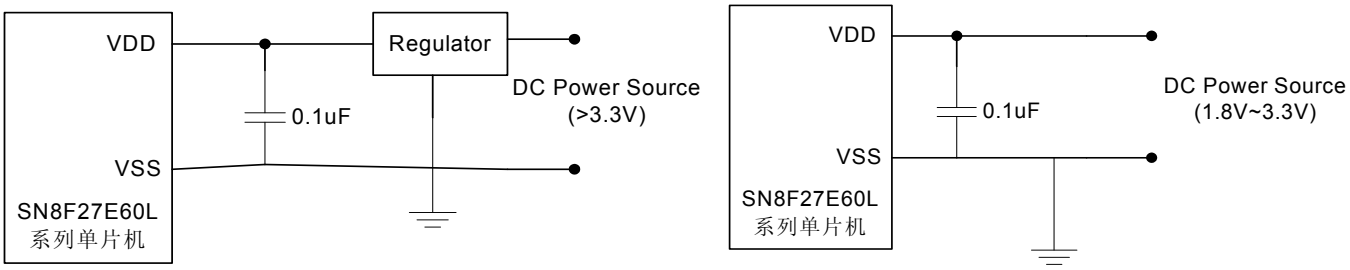
- **4 个系统时钟**
  外部高速时钟：RC，高达 10MHz。
  外部高速时钟：晶振，高达 16MHz。
  内部高速时钟：RC，高达 16MHz。
  内部低速时钟：RC，16KHz。
- **4 种操作模式**
  普通模式：高、低速时钟都正常运行。
  低速模式：只有低速时钟正常运行。
  睡眠模式：高、低速时钟都停止工作。
  绿色模式：由定时器周期性的唤醒。
- **封装形式**
  PDIP 32 pin
  LQFP 32 pin
  QFN 32 pin
  SDIP 32 pin
  SKDIP 28 pin
  SOP 28 pin
  SSOP 28 pin
  QFN 28 pin
  DIP 20 pin
  SOP 20 pin

SN8F27E60 系列单片机有两种不同的引脚配置，分别应用于不同的电源场合。

应用于 AC（交流电源）和 DC 高压（≤5.5V）场合时，电源引脚有 VDD 和 VDDL。VDD 引脚连接到由 DC-DC 变压器或者调整器产生的 DC 电源，且和 VSS 引脚（GND）之间须连接一个 0.1uF 的电容。VDDL 引脚是内部的终端电源，不需要在和任何电源连接，只需要和 VSS 引脚（GND）之间连接一个 0.1uF 的电容。该引脚配置具有高抗干扰能力，但静态电流较高。适用于家电、马达等控制领域。



应用于 DC（电池电源）场合时，电源引脚为 VDD，连接到由电池提供的电源上，且须和 VSS 引脚（GND）之间连接一个 0.1uF 的电容。该引脚配置的抗干扰能力较低，但静态电流也很低。适用于便携、无线产品等领域。



☞ **产品选型表**

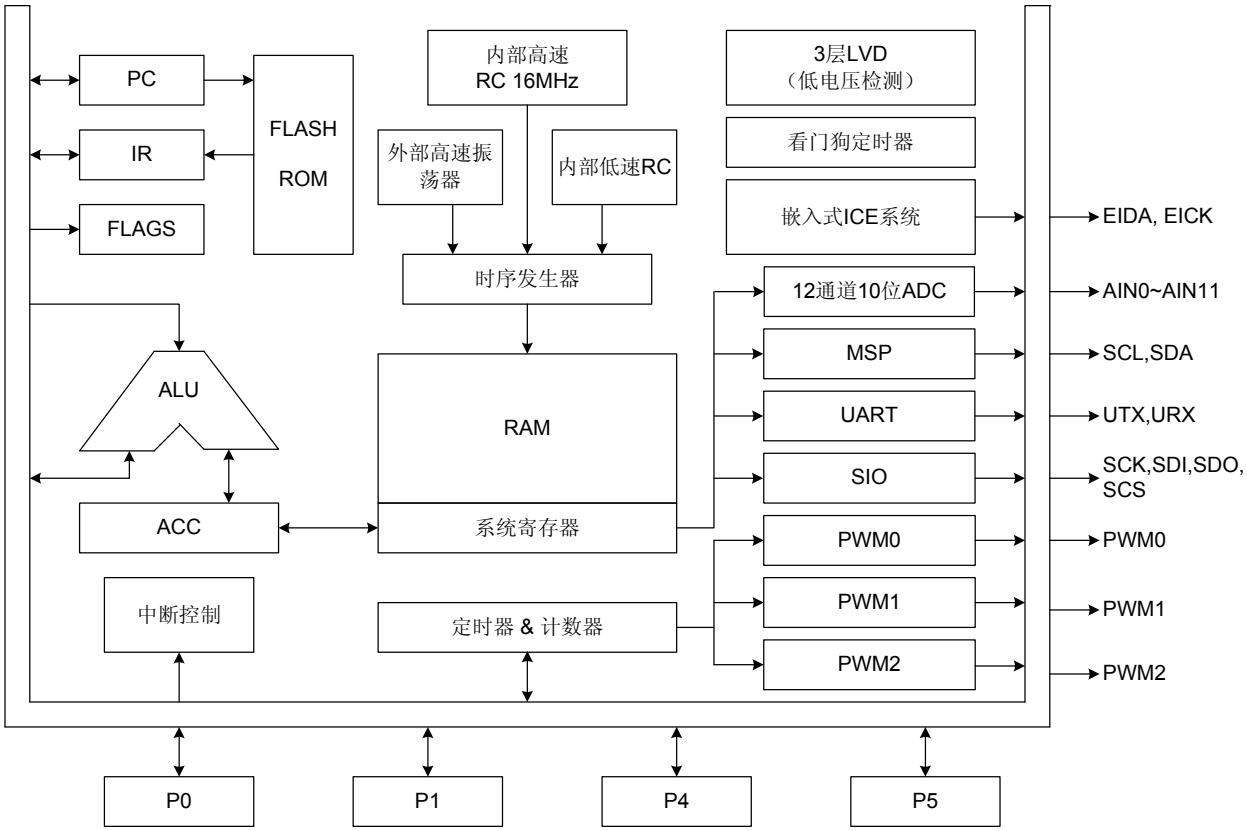
SN8F27E60 系列

单片机名称	ROM	RAM	堆栈	定时器	I/O	PWM	ADC	SIO	UART	MSP	Ext. INT	ISP/嵌入式 ICE	工作电压范围	封装形式
SN8F27E65	6K*16	512	8	8-bit*4 16-bit*1	27	3-ch	12-ch	V	V	V	2	V	1.8V~5.5V	DIP32 LQFP32 QFN32 SDIP32
SN8F27E64	6K*16	512	8	8-bit*4 16-bit*1	25	3-ch	11-ch	V	V	V	2	V	1.8V~5.5V	SKDIP28 SOP28 SSOP28 QFN28
SN8F27E62	6K*16	512	8	8-bit*4 16-bit*1	17	3-ch	9-ch	-	V	-	1	V	1.8V~5.5V	DIP20 SOP20

SN8F27E60L 系列

单片机名称	ROM	RAM	堆栈	定时器	I/O	PWM	ADC	SIO	UART	MSP	Ext. INT	ISP/嵌入式 ICE	工作电压范围	封装形式
SN8F27E65L	6K*16	512	8	8-bit*4 16-bit*1	27	3-ch	12-ch	V	V	V	2	V	1.8V~3.3V	DIP32 LQFP32 QFN32 SDIP32
SN8F27E64L	6K*16	512	8	8-bit*4 16-bit*1	25	3-ch	11-ch	V	V	V	2	V	1.8V~3.3V	SKDIP28 SOP28 SSOP28
SN8F27E62L	6K*16	512	8	8-bit*4 16-bit*1	17	3-ch	9-ch	-	V	-	1	V	1.8V~3.3V	DIP20 SOP20

1.2 系统框图


PC
IR
FLAGS
FLASH
ROM
内部高速
RC 16MHz
外部高速振
荡器
内部低速RC
时序发生器
3层LVD
（低电压检测）
看门狗定时器
嵌入式ICE系统
12通道10位ADC
MSP
UART
SIO
PWM0
PWM1
PWM2
EIDA, EICK
AIN0~AIN11
SCL,SDA
UTX,URX
SCK,SDI,SDO,
SCS
PWM0
PWM1
PWM2
ALU
RAM
ACC
系统寄存器
中断控制
定时器 & 计数器
P0
P1
P4
P5

1.3 引脚配置

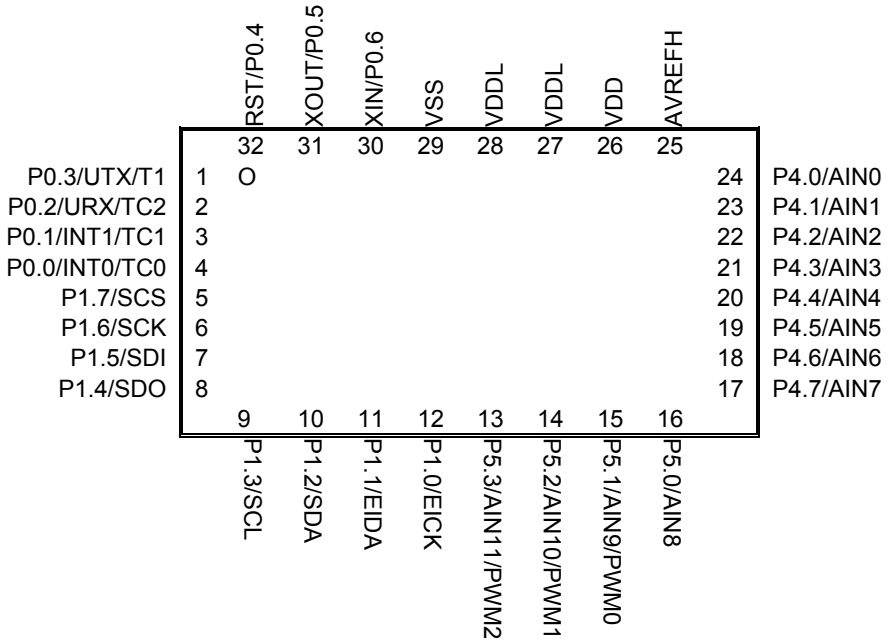
SN8F27E65P (AC field, DIP 32 Pin):
SN8F27E65P (AC field, SDIP 32 Pin):

VSS	1	U	32	VDDL
XIN/P0.6	2		31	VDDL
XOUT/P0.5	3		30	VDD
RST/P0.4	4		29	AVREFH
P0.3/UTX/T1	5		28	P4.0/AIN0
P0.2/URX/TC2	6		27	P4.1/AIN1
P0.1/INT1/TC1	7		26	P4.2/AIN2
P0.0/INT0/TC0	8		25	P4.3/AIN3
P1.7/SCS	9		24	P4.4/AIN4
P1.6/SCK	10		23	P4.5/AIN5
P1.5/SDI	11		22	P4.6/AIN6
P1.4/SDO	12		21	P4.7/AIN7
P1.3/SCL	13		20	P5.0/AIN8
P1.2/SDA	14		19	P5.1/AIN9/PWM0
P1.1/EIDA	15		18	P5.2/AIN10/PWM1
P1.0/EICK	16		17	P5.3/AIN11/PWM2

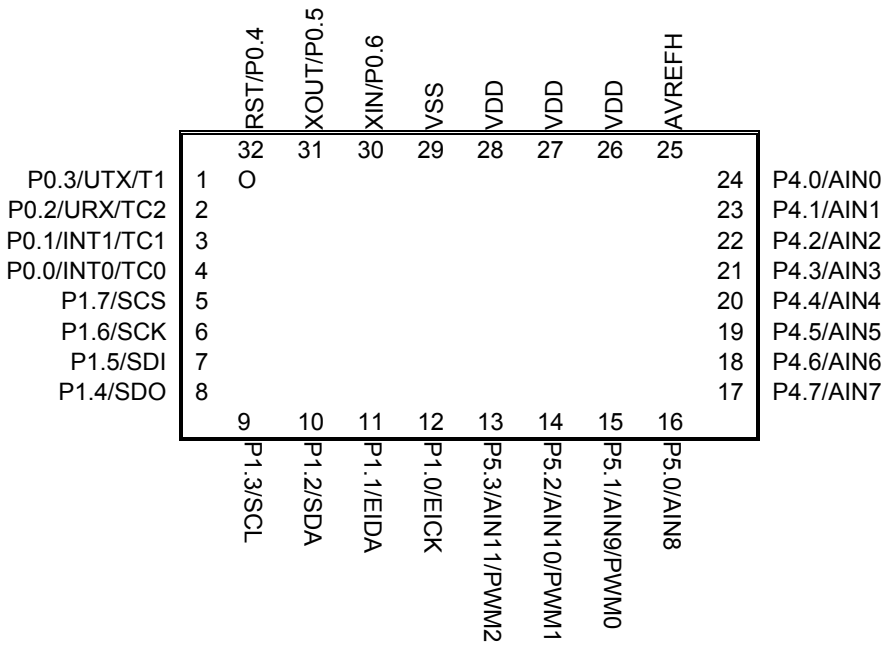
SN8F27E65LP (DC field, DIP 32 Pin):
SN8F27E65LP (DC field, SDIP 32 Pin):

VSS	1	U	32	VDD
XIN/P0.6	2		31	VDD
XOUT/P0.5	3		30	VDD
RST/P0.4	4		29	AVREFH
P0.3/UTX/T1	5		28	P4.0/AIN0
P0.2/URX/TC2	6		27	P4.1/AIN1
P0.1/INT1/TC1	7		26	P4.2/AIN2
P0.0/INT0/TC0	8		25	P4.3/AIN3
P1.7/SCS	9		24	P4.4/AIN4
P1.6/SCK	10		23	P4.5/AIN5
P1.5/SDI	11		22	P4.6/AIN6
P1.4/SDO	12		21	P4.7/AIN7
P1.3/SCL	13		20	P5.0/AIN8
P1.2/SDA	14		19	P5.1/AIN9/PWM0
P1.1/EIDA	15		18	P5.2/AIN10/PWM1
P1.0/EICK	16		17	P5.3/AIN11/PWM2

SN8F27E65F (AC field, LQFP 32 Pin):
SN8F27E65J (AC field, QFN 5x5 32Pin):



SN8F27E65LF (DC field, LQFP 32 Pin):
SN8F27E65LJ (DC field, QFN 5x5 32 Pin):



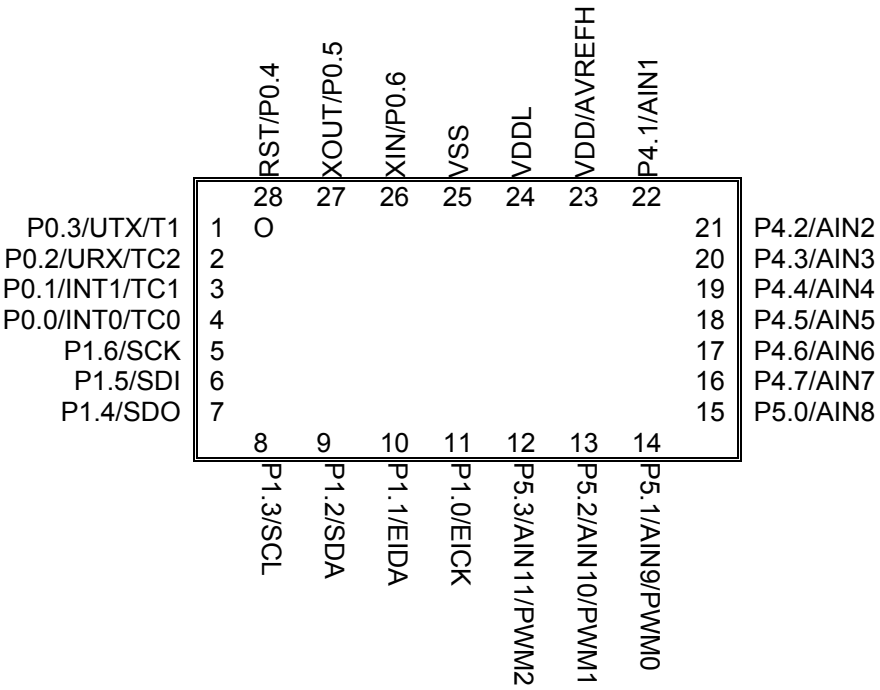
SN8F27E64K (AC field, SKDIP 28 Pin):
SN8F27E64S (AC field, SOP 28 Pin):
SN8F27E64X (AC field, SSOP 28 Pin):

VSS	1	U	28	VDDL
XIN/P0.6	2		27	VDD/AVREFH
XOUT/P0.5	3		26	P4.1/AIN1
RST/P0.4	4		25	P4.2/AIN2
P0.3/UTX/T1	5		24	P4.3/AIN3
P0.2/URX/TC2	6		23	P4.4/AIN4
P0.1/INT1/TC1	7		22	P4.5/AIN5
P0.0/INT0/TC0	8		21	P4.6/AIN6
P1.6/SCK	9		20	P4.7/AIN7
P1.5/SDI	10		19	P5.0/AIN8
P1.4/SDO	11		18	P5.1/AIN9/PWM0
P1.3/SCL	12		17	P5.2/AIN10/PWM1
P1.2/SDA	13		16	P5.3/AIN11/PWM2
P1.1/EIDA	14		15	P1.0/EICK

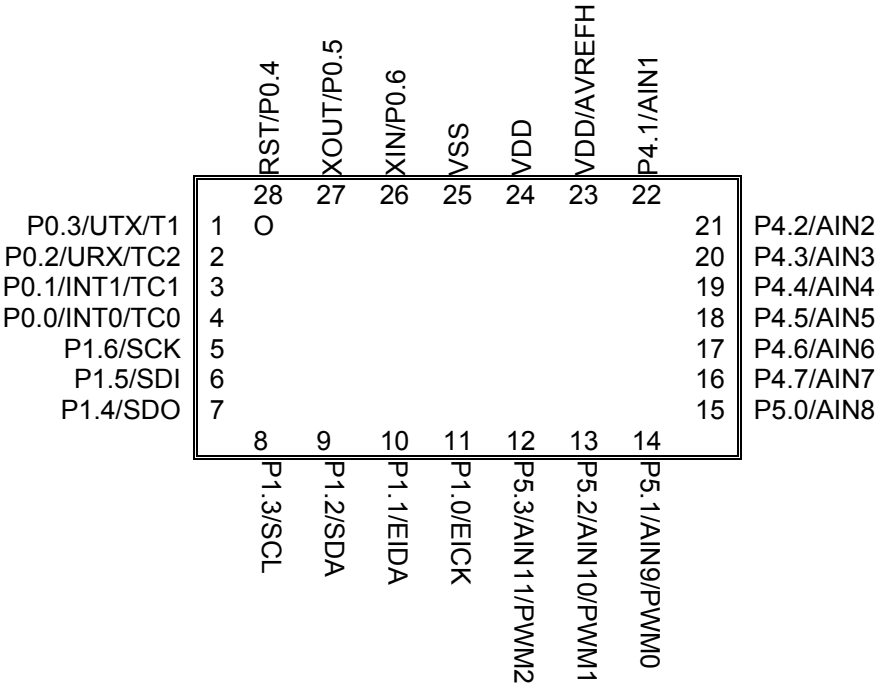
SN8F27E64LK (DC field, SKDIP 28 Pin):
SN8F27E64LS (DC field, SOP 28 Pin):
SN8F27E64LX (DC field, SSOP 28 Pin):

VSS	1	U	28	VDD
XIN/P0.6	2		27	VDD/AVREFH
XOUT/P0.5	3		26	P4.1/AIN1
RST/P0.4	4		25	P4.2/AIN2
P0.3/UTX/T1	5		24	P4.3/AIN3
P0.2/URX/TC2	6		23	P4.4/AIN4
P0.1/INT1/TC1	7		22	P4.5/AIN5
P0.0/INT0/TC0	8		21	P4.6/AIN6
P1.6/SCK	9		20	P4.7/AIN7
P1.5/SDI	10		19	P5.0/AIN8
P1.4/SDO	11		18	P5.1/AIN9/PWM0
P1.3/SCL	12		17	P5.2/AIN10/PWM1
P1.2/SDA	13		16	P5.3/AIN11/PWM2
P1.1/EIDA	14		15	P1.0/EICK

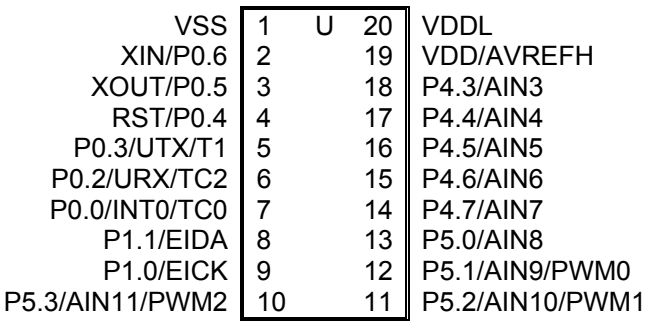
SN8F27E64J (AC field, QFN 4x4 28 Pin):



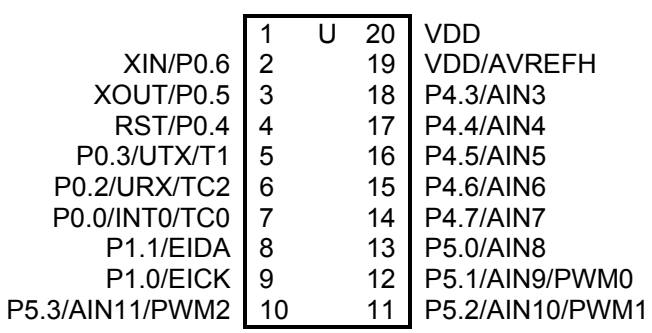
SN8F27E64LJ (DC field, QFN 4x4 28 Pin):



SN8F27E62K (AC field, DIP 20 Pin):
SN8F27E62S (AC field, SOP 20 Pin):



SN8F27E62LK (DC field, DIP 20 Pin):
SN8F27E62LS (DC field, SOP 20 Pin):



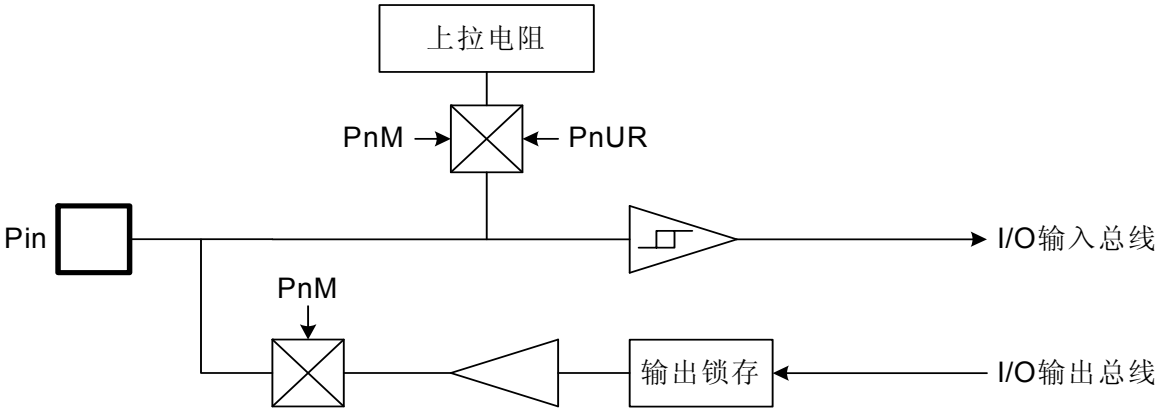
1.4 引脚说明

引脚名称	类型	功能说明
VDD, VSS	P	电源输入引脚。
VDDL	P	低电源引脚，和 VSS 引脚之间连接一个 0.1uF 的电容。
AVREFH	P	ADC 参考电压的高电平输入引脚。
RST/P0.4	I/O	RST：系统复位输入引脚，施密特触发，低电平有效，通常保持高电平。
		P0.4：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时具有唤醒功能。
XIN/P0.6	I/O	XIN：外部振荡器输入引脚。
		P0.6：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时具有唤醒功能。
XOUT/P0.5	I/O	XOUT：外部振荡器输出引脚。
		P0.5：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时具有唤醒功能。
P0.0/INT0/ TC0	I/O	P0.0：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时具有唤醒功能。
		INT0：外部中断 INT0 输入引脚。
		TC0：事件计数器 TC0 输入引脚。
P0.1/INT1/ TC1	I/O	P0.1：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时具有唤醒功能。
		INT1：外部中断 INT1 输入引脚。
		TC1：事件计数器 TC1 输入引脚。
P0.2/URX/ TC2	I/O	P0.2：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时具有唤醒功能，可编程漏极开路引脚。
		TC2：事件计数器 TC2 输入引脚。
		URX：UART 接收输入引脚。
P0.3/UTX/T1	I/O	P0.3：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时具有唤醒功能，可编程漏极开路引脚。
		UTX：UART 发送输出引脚。
		T1：时间计数器 T1 输入引脚。
P1.0/EICK	I/O	P1.0：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时具有唤醒功能，可编程漏极开路引脚。
		EICK：嵌入式 ICE 时钟引脚。
P1.1/EIDA	I/O	P1.1：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时具有唤醒功能，可编程漏极开路引脚。
		EIDA：嵌入式 ICE 数据引脚。
P1.2/SDA	I/O	P1.2：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时具有唤醒功能，可编程漏极开路引脚。
		SDA：MSP 数据引脚。
P1.3/SCL	I/O	P1.3：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时具有唤醒功能，可编程漏极开路引脚。
		SCL：MSP 时钟引脚。
P1.4/SDO	I/O	P1.4：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时具有唤醒功能，可编程漏极开路引脚。
		SDO：SIO 数据输出引脚。
P1.5/SDI	I/O	P1.5：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时具有唤醒功能，可编程漏极开路引脚。
		SDI：SIO 数据输出引脚。
P1.6/SCK	I/O	P1.6：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时具有唤醒功能，可编程漏极开路引脚。
		SCK：SIO 时钟引脚。
P1.7/SCS	I/O	P1.7：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时具有唤醒功能，可编程漏极开路引脚。
		SCS：SIO 总线控制引脚。
P4.0/AIN0	I/O	P4.0：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻。
		AIN0：ADC 输入通道 AIN0。
P4.1/AIN1	I/O	P4.1：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻。
		AIN1：ADC 输入通道 AIN1。
P4.2/AIN2	I/O	P4.2：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻。
		AIN2：ADC 输入通道 AIN2。
P4.3/AIN3	I/O	P4.3：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻。
		AIN3：ADC 输入通道 AIN3。
P4.4/AIN4	I/O	P4.4：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻。

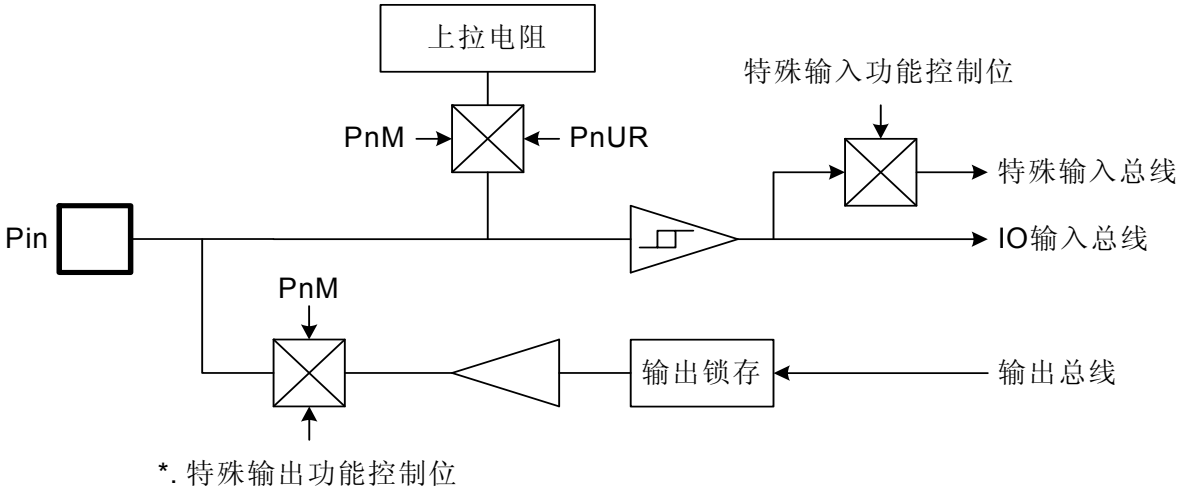
		AIN4：ADC 输入通道 AIN4。
P4.5/AIN5	I/O	P4.5：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻。
		AIN5：ADC 输入通道 AIN5。
P4.6/AIN6	I/O	P4.6：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻。
		AIN6：ADC 输入通道 AIN6。
P4.7/AIN7	I/O	P4.7：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻。
		AIN7：ADC 输入通道 AIN7。
P5.0/AIN8	I/O	P5.0：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻。
		AIN8：ADC 输入通道 AIN8。
P5.1/AIN9/ PWM0	I/O	P5.1：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻。
		AIN9：ADC 输入通道 AIN9。
		PWM0：PWM0 输出引脚。
P5.2/AIN10/ PWM1	I/O	P5.2：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻。
		AIN10：ADC 输入通道 AIN10。
		PWM1：PWM1 输出引脚。
P5.3/AIN11/ PWM2	I/O	P5.3：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻。
		AIN11：ADC 输入通道 AIN11。
		PWM2：PWM2 输出引脚。

1.5 引脚电路结构图

普通双向 I/O 引脚：

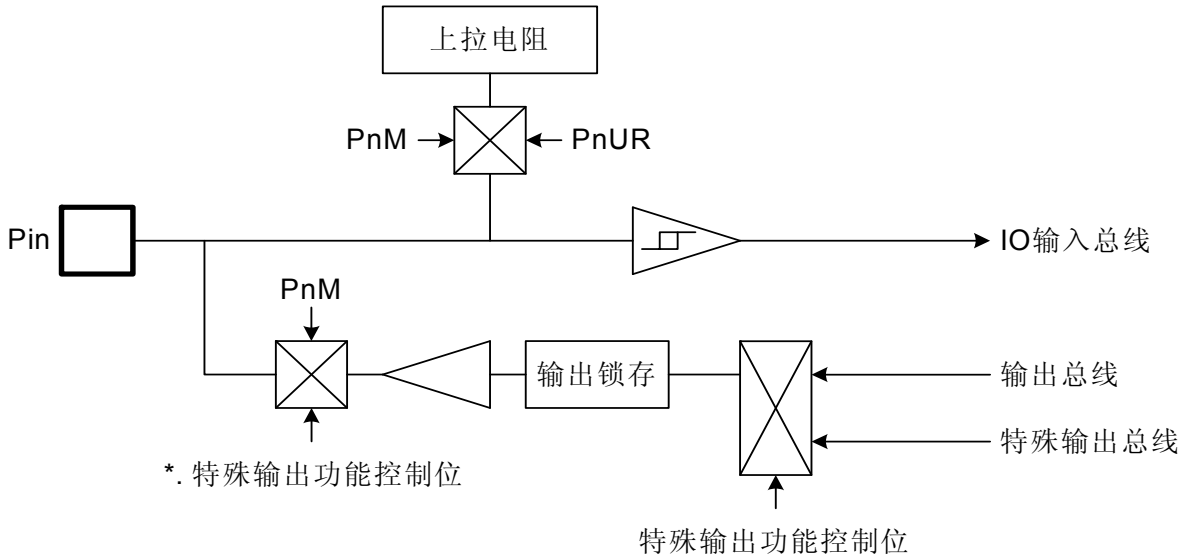


双向 I/O 引脚，与特殊数字输入引脚复用，如：INT0、事件计数器、SIO、MSP、UART……



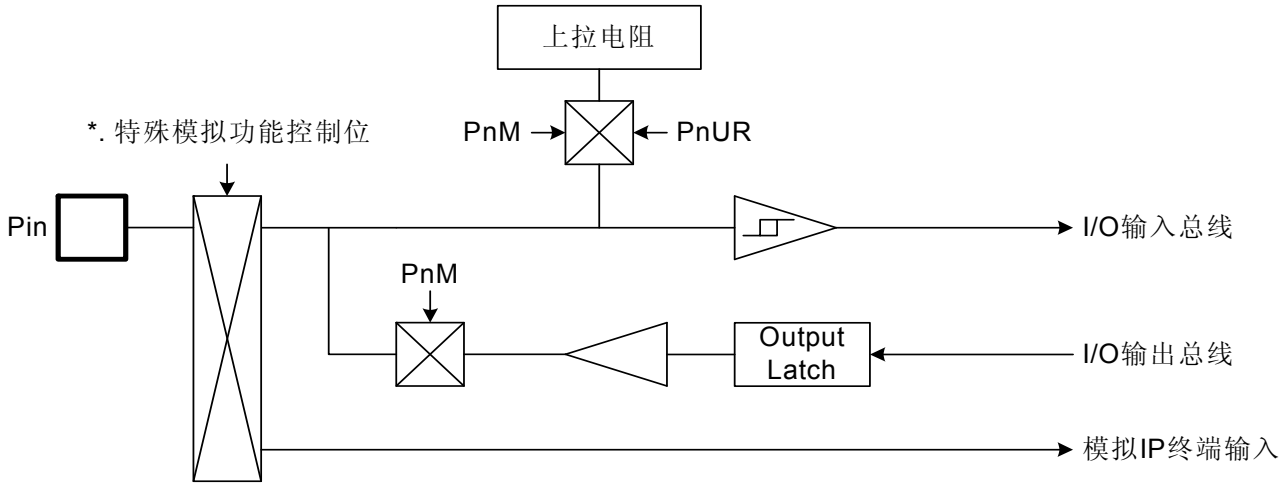
***. 部分特殊功能直接通过I/O方向进行切换，而无需通过PnM寄存器。**

双向 I/O 引脚，与特殊数字输出引脚复用，如：PWM、SIO、MSP、UART……



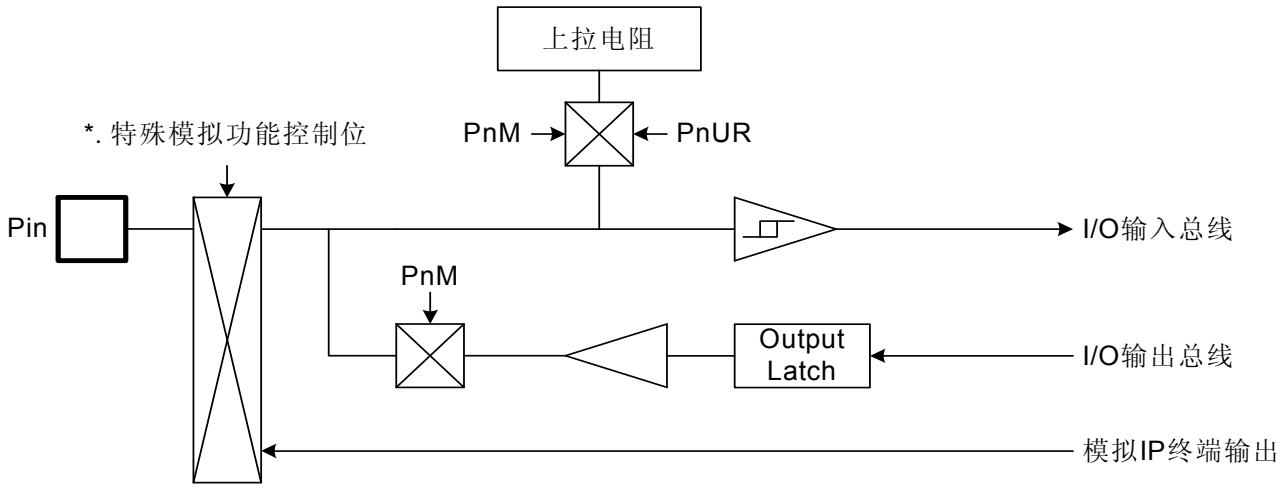
***. 部分特殊功能直接通过I/O方向进行切换，而无需通过PnM寄存器。**

双向 I/O 引脚，与特殊模拟输入引脚复用，如：XIN、ADC……



***. 部分特殊功能直接通过I/O方向进行切换，而无需通过PnM寄存器。**

双向 I/O 引脚，与特殊模拟输出引脚复用，如 XOUT……



***. 部分特殊功能直接通过I/O方向进行切换，而无需通过PnM寄存器。**

2 中央处理器（CPU）

2.1 程序存储器（FLASH ROM）

☞ **FLASH ROM：6K**

地址	ROM	注释
0000H	**复位向量**	复位向量
0001H . 0007H	**通用存储区**	用户程序
0008H	**WAKE 中断向量**	中断向量
0009H	**INT0 中断向量**	
000AH	**INT1 中断向量**	
000BH	**T0 中断向量**	
000CH	**TC0 中断向量**	
000DH	**TC1 中断向量**	
000EH	**TC2 中断向量**	
000FH	**T1 中断向量**	
0010H	**ADC 中断向量**	
0011H	**SIO 中断向量**	
0012H	**MSP 中断向量**	
0013H	**UART RX 中断向量**	
0014H	**UART TX 中断向量**	
0015H	**通用存储区**	用户程序 用户程序结束
17FCH 17FDH 17FEH 17FFH	**系统保留**	

ROM 包括复位向量，中断向量，通用存储区和系统保留区。复位系列是程序的起始地址，中断向量是响应中断时，中断程序的起始地址，通用存储区是用户程序的存储区，包括主程序循环和子程序及数据表格。

- **0000H**：复位向量。复位（上电复位、复位引脚复位、看门狗复位和 LVD 复位等）后，程序计数器 PC 指向 00H。
- **0001H~0007H**：通用存储区，处理系统复位操作。
- **0008H~0014H**：多个中断向量区域。每个中断都相应对应唯一的中断向量。
- **0015H~177FH**：通用存储区。用来存储用户程序，ISP（EEPROM 功能）。
- **1780H~17F7H**：通用存储区。用来存储用户程序，不执行 ISP。
- **17F8H~17FFH**：系统保留区。不执行 ISP。
- **ROM 的安全规则**：偶数地址的 ROM 数据保护和输出 00H。

2.1.1 复位向量（0000H）

具有一个字长的系统复位向量（0000H）。

- **上电复位（POR=1）；**
- **看门狗复位（WDT=1）；**
- **外部复位（RST=1）。**

发生上述任一种复位后，程序将从 0000H 处重新开始执行，系统寄存器也都将恢复为默认值。根据 PFLAG 寄存器中的 POR、WDT 和 RST 标志位的内容可以判断系统复位方式。下面一段程序演示了如何定义 ROM 中的复位向量。

➢ **例：定义复位向量。**

```
                ORG     0               ;
                JMP     START           ; 跳至用户程序。
                …

                ORG     15H
START:                                  ; 用户程序起始地址。
                …                       ; 用户程序。
                …
                ENDP                    ; 程序结束。
```

✷ **注：用户程序的起始地址应该跳过中断向量区域，以免引起程序执行错误。**

2.1.2 中断向量（0008H~0014H）

13 字的中断向量地址用来响应中断请求，任意中断发生时，程序计数器 PC 的值存入到堆栈缓存器并跳转至程序存储器的 08H~14H 处执行中断。27E60 系列的多中断对应唯一的中断向量。用户必须定义中断向量，下面的程序演示了如何定义中断向量。

✷ **注：PUSH、POP 操作无需通过 PUSH、POP 指令执行，由硬件自动保存和恢复 ACC 和工作寄存器（80H~8FH）。**

	ROM	优先级别
0008H	**WAKE 中断向量**	1
0009H	**INT0 中断向量**	2
000AH	**INT1 中断向量**	3
000BH	**T0 中断向量**	4
000CH	**TC0 中断向量**	5
000DH	**TC1 中断向量**	6
000EH	**TC2 中断向量**	7
000FH	**T1 中断向量**	8
0010H	**ADC 中断向量**	9
0011H	**SIO 中断向量**	10
0012H	**MSP 中断向量**	11
0013H	**UART RX 中断向量**	12
0014H	**UART TX 中断向量**	13

当有中断发生时，程序计数器跳至相对应的中断向量处地址，然后执行中断。如发生 WAKE 中断时，程序计数器跳至 ORG 8 处，发生 INT0 中断时，程序计数器跳至 0RG 9 处。通常情况下，同一时间会发生多个中断，故必须设置中断的优先权，否则系统不知道应该先处理哪一个中断。中断的优先顺序为：ORG 8，ORG 9，依次往下。

举例说明：如果 WAKE、T0、TC2、T1 和 SIO 同时请求中断，则系统处理中断的顺序为：WAKE、T0、TC2、T1，最后为 SIO，即系统最先处理 WAKE 中断，然后处理 T0 中断，最后处理 SIO 中断。

➢ **例：**

下列中断请求响应（2~8 中断是在执行 WAKE 中断时请求响应的）。

1	2	3	4	5	6	7	8
WAKE	ADC	TC1	T0	SIO	INT0	T1	UART RX

处理上表中断的顺序：

1	2	3	4	5	6	7	8
WAKE	INT0	T0	TC1	T1	ADC	SIO	UART RX

➢ **例：定义中断向量，中断服务程序紧随用户程序之后。**

```
.CODE
                ORG     0               ; 0000H
                JMP     START           ; 跳到用户程序。
                …
                ORG     8               ; 中断向量，0008H.
                JMP     ISR_WAKE        ; 跳转到中断服务程序。
                JMP     ISR_INT0
                JMP     ISR_INT1
                JMP     ISR_T0
                JMP     ISR_TC0
                JMP     ISR_TC1
                JMP     ISR_TC2
                JMP     ISR_T1
                JMP     ISR_ADC
                JMP     ISR_SIO
                JMP     ISR_MSP
                JMP     ISR_UART_RX
                JMP     ISR_UART_TX

                ORG     15H
START:                                  ; 0015H，用户程序开始。
                …                       ; 用户程序。
                …
                …
                JMP     START           ; 用户程序结束。
                …
ISR_WAKE:                               ; 中断服务程序开始。
                                        ; 保存 ACC 和 80H~8FH 工作寄存器。
                …
                                        ; 恢复 ACC 和 80H~8FH 工作寄存器。
                RETI                    ; 中断服务程序结束。
ISR_INT0:                               ;
                                        ; 保存 ACC 和 80H~8FH 工作寄存器。
                …
                                        ; 恢复 ACC 和 80H~8FH 工作寄存器。
                RETI                    ; 中断服务程序结束。
                …
                …
                …
                …
                …
ISR_UART_TX:                            ;
                                        ; 保存 ACC 和 80H~8FH 工作寄存器。
                …
                                        ; 恢复 ACC 和 80H~8FH 工作寄存器。
                RETI                    ; 中断服务程序结束。


                ENDP                    ; 程序结束。
```

✷ **注：从上面的示例程序可以看出 SONIX 的编程规则，有以下几点：**
1、地址 0000H 处的 JMP 指令是整个程序的开始地址；
2、地址 0008H~0014H 为中断向量地址；
3、用户主程序为一个循环。

2.1.3 查表

在 SONiX 单片机中，对 ROM 区中的数据进行查找，查表指针存放在寄存器 Y，Z 中。寄存器 Y 指向所找数据地址的中间字节（bit8~bit15），寄存器 Z 指向所找数据地址的低字节（bit0~bit7）。执行完 MOVC 指令后，所查找数据低字节内容被存入 ACC 中，而数据高字节内容被存入 R 寄存器。

➢ **例：查找 ROM 地址为“TABLE1”的值。**

```
        B0MOV   Y, #TABLE1$M    ; 设置 table1 的中间字节地址。
        B0MOV   Z, #TABLE1$L    ; 设置 table1 的低位字节地址。
        MOVC                    ; 查表，R = 00H，ACC = 35H。

                                ; 查找下一地址。
        INCMS   Z               ; Z+1。
        JMP     @F              ; Z 没有溢出。
        INCMS   Y               ; Z 溢出，Y=Y+1。
        JMP     @F              ; Y 没有溢出。
        NOP
                                ;
@@:     MOVC                    ; 查表，R = 51H，ACC = 05H。
        …                       ;

TABLE1: DW      0035H           ; 定义数据表数据（16-bit）。
        DW      5105H
        DW      2012H
        …
```

✷ **注：当寄存器 Z 溢出（从 FFH 变成 00H）时，Y 寄存器并不会自动加 1。用户必须注意这种情况以免查表错误。若 Z 溢出，Y 必须由程序加 1，下面的宏指令 INC_YZ 可以对 Y 和 Z 寄存器自动处理。**

➢ **例：宏指令 INC_YZ。**

➢ **例：宏指令 INC_YZ。**

```
INC_YZ  MACRO
        INCMS   Z               ; Z+1
        JMP     @F              ; 没有溢出。

        INCMS   Y               ; Y+1
        NOP                     ; 没有溢出。
@@:
        ENDM
```

➢ **例：通过宏指令 INC_YZ 对上例优化。**

```
        B0MOV   Y, #TABLE1$M    ; 设置 table1 的中间字节地址。
        B0MOV   Z, #TABLE1$L    ; 设置 table1 的低位字节地址。
        MOVC                    ; 查表，R = 00H，ACC = 35H。

        INC_YZ                  ; 查找下一地址。
                                ;
@@:     MOVC                    ; 查表，R = 51H，ACC = 05H。
        …                       ;

TABLE1: DW      0035H           ; 定义数据表数据（16-bit）。
        DW      5105H
        DW      2012H
        …
```

下面的程序通过累加器对 Y 和 Z 寄存器进行处理来实现查表功能，但需要特别注意进位时的处理。

➢ **例：通过指令 B0ADD/ADD 实现查表功能。**

```
                B0MOV     Y, #TABLE1$M    ; 设置 table1 的中间字节地址。
                B0MOV     Z, #TABLE1$L    ; 设置 table1 的低位字节地址。

                B0MOV     A, BUF          ; Z = Z + BUF。
                B0ADD     Z, A

                B0BTS1    FC              ; 检查进位标志。
                JMP       GETDATA         ; FC = 0。
                INCMS     Y               ; FC = 1，Y+1。
                NOP

GETDATA:                                  ;
                MOVC                      ; 存储数据，如果 BUF = 0，数据为 35H。
                                          ; 如果 BUF = 1，数据＝5105H。
                                          ; 如果 BUF = 2，数据＝2012H
                …

TABLE1:         DW        0035H           ; 定义数据表数据（16-bit）。
                DW        5105H
                DW        2012H
                …
```

2.1.4 跳转表

跳转表能够实现多地址跳转功能。由于 PCL 和 ACC 的值相加即可得到新的 PCL，因此，可以通过对 PCL 加上不同的 ACC 值来实现多地址跳转。ACC 值若为 n，PCL+ACC 即表示当前地址加 n，执行完当前指令后 PCL 值还会自加 1，可参考以下范例。如果 PCL＋ACC 后发生溢出，PCH 则自动加 1。由此得到的新的 PC 值再指向跳转指令列表中新的地址。这样，用户就可以通过修改 ACC 的值轻松实现多地址的跳转。

✷ **注：PCH 只支持 PC 增量运算，而不支持 PC 减量运算。当 PCL+ACC 后如有进位，PCH 的值会自动加 1。PCL-ACC 后若有借位，PCH 的值将保持不变，用户在设计应用时要加以注意。**

➢ **例：跳转表。**

```
        ORG     0100H           ; 跳转表从 ROM 前端开始。

        B0ADD   PCL, A          ; PCL = PCL + ACC，PCL 溢出时 PCH 加 1。
        JMP     A0POINT         ; ACC = 0，跳至 A0POINT。
        JMP     A1POINT         ; ACC = 1，跳至 A1POINT。
        JMP     A2POINT         ; ACC = 2，跳至 A2POINT。
        JMP     A3POINT         ; ACC = 3，跳至 A3POINT。
```

SONiX 单片机提供一个宏以保证可靠执行跳转表功能，它会自动检测 ROM 边界并将跳转表移至适当的位置。但采用该宏程序会占用部分 ROM 空间。

➢ **例：如果跳转表跨越 ROM 边界，将引起程序错误。**

```
@JMP_A  MACRO   VAL
        IF      (($+1) !& 0XFF00) !!= (($+(VAL)) !& 0XFF00)
        JMP     ($ | 0XFF)
        ORG     ($ | 0XFF)
        ENDIF
        B0ADD   PCL, A
        ENDM
```

✷ **注：“VAL”为跳转表列表中列表个数。**

➢ **例：宏“MACRO3.H”中，“@JMP_A”的应用。**

```
        B0MOV   A, BUF0         ; “BUF0”从 0 至 4。
        @JMP_A  5               ; 列表个数为 5。
        JMP     A0POINT         ; ACC = 0，跳至 A0POINT。
        JMP     A1POINT         ; ACC = 1，跳至 A1POINT。
        JMP     A2POINT         ; ACC = 2，跳至 A2POINT。
        JMP     A3POINT         ; ACC = 3，跳至 A3POINT。
        JMP     A4POINT         ; ACC = 4，跳至 A4POINT。
```

如果跳转表恰好位于 ROM BANK 边界处（00FFH~0100H），宏指令“@JMP_A”将调整跳转表到适当的位置（0100H）。

➢ **例：“@JMP_A”运用举例**

```
; 编译前
ROM 地址
                B0MOV     A, BUF0       ; “BUF0”从 0 到 4。
                @JMP_A    5             ; 列表个数为 5。
00FDH           JMP       A0POINT       ; ACC = 0，跳至 A0POINT。
00FEH           JMP       A1POINT       ; ACC = 1，跳至 A1POINT。
00FFH           JMP       A2POINT       ; ACC = 2，跳至 A2POINT。
0100H           JMP       A3POINT       ; ACC = 3，跳至 A3POINT。
0101H           JMP       A4POINT       ; ACC = 4，跳至 A4POINT。

; 编译后
ROM 地址
                B0MOV     A, BUF0       ; “BUF0”从 0 到 4。
                @JMP_A    5             ; 列表个数为 5。
0100H           JMP       A0POINT       ; ACC = 0，跳至 A0POINT。
0101H           JMP       A1POINT       ; ACC = 1，跳至 A1POINT。
0102H           JMP       A2POINT       ; ACC = 2，跳至 A2POINT。
0103H           JMP       A3POINT       ; ACC = 3，跳至 A3POINT。
0104H           JMP       A4POINT       ; ACC = 4，跳至 A4POINT。
```

2.1.5 CHECKSUM计算

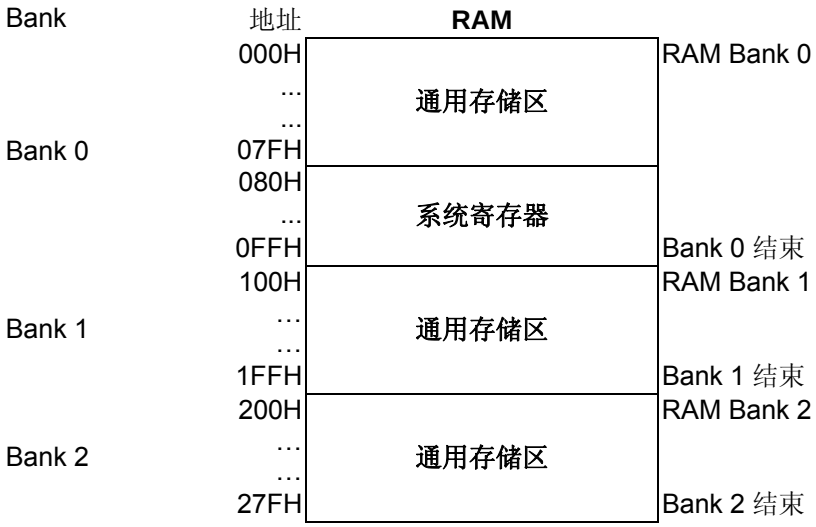
ROM 区末端位置的几个字限制使用，进行 Checksum 计算时，用户应避免对该单元的访问。

➢ **例：示例程序演示了如何对 00H 到用户程序结束进行 Checksum 计算。**

```
                MOV     A,#END_USER_CODE$L
                B0MOV   END_ADDR1, A        ; 用户程序结束地址低位地址存入 end_addr1。
                MOV     A,#END_USER_CODE$M
                B0MOV   END_ADDR2, A        ; 用户程序结束地址中间地址存入 end_addr2。
                CLR     Y                   ; 清 Y。
                CLR     Z                   ; 清 Z。
@@:
                MOVC
                B0BCLR  FC                  ; 清标志位 C。
                ADD     DATA1, A            ;
                MOV     A, R
                ADC     DATA2, A            ;
                JMP     END_CHECK           ; 检查 YZ 地址是否为代码的结束地址。
AAA:
                INCMS   Z
                JMP     @B                  ; 若 Z != 00H，进行下一个计算。
                JMP     Y_ADD_1             ; 若 Z = 00H，Y+1。
END_CHECK:
                MOV     A, END_ADDR1
                CMPRS   A, Z                ; 检查 Z 地址是否为用户程序结束地址低位地址。
                JMP     AAA                 ; 否，则进行 Checksum 计算。
                MOV     A, END_ADDR2
                CMPRS   A, Y                ; 是则检查 Y 的地址是否为用户程序结束地址中间地址。
                JMP     AAA                 ; 否，则进行 Checksum 计算。
                JMP     CHECKSUM_END        ; 是则 Checksum 计算结束。

Y_ADD_1:
                INCMS   Y                   ;
                NOP
                JMP     @B                  ; 跳转到 Checksum 计算。
CHECKSUM_END:
                …
                …
END_USER_CODE:                              ; 程序结束。
```

2.2 数据存储器（RAM）

☞ **RAM：512 X 8-bit**



512 字节的 RAM 分为 Bank0，Bank1 和 Bank2，通过设置寄存器 RBANK 在三个 RAM 中转换。当 RBANK=0 时，程序直接在 BANK0 中运行；RBANK=1 时，直接在 BANK1 中运行；RBANK=2 时，则直接在 BANK2 中运行。当在其中一个 BANK 中运行，而需要访问另一个 BANK 时，必须设置 RBANK。中断发生时，保存 RBANK，即保存最后一个 BANK 的工作状态。用户在处理中断的过程中，可以通过设置 RBANK 选择 RAM BANK。执行 RETI，即退出中断时，恢复 RBANK，RAM BANK 恢复到上一个 RAM 工作状态。SONIX 提供 BANK0 型的指令（如 B0MOV、B0ADD、B0BTS1、B0BSET 等），可以在非 0 RAM BANK 区域直接控制 BANK0 的工作状态。

➢ **例：在 Bank1 中访问 Bank0，将 Bank0 RAM（WK00）的值一点到 Bank1 RAM（WK01）中。**

```
; Bank 1 (RBANK = 1)
        B0MOV   A, WK00     ; 使用 Bank0 型指令访问 Bank0 RAM。
        MOV     WK01,A
```

✷ **注：对于多个 BANK RAM 的操作，用户必须注意 RBANK 的设置以选择 RAM BANK，尤其在中断时的 RAM BANK 操作。RAM BANK 切换到 BANK0 时，系统不会自动保存 RBANK，必须通过程序保存，最好使用 BANK0 型指令来进行该操作。**

2.2.1 系统寄存器

2.2.1.1系统寄存器列表

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	L	H	R	Z	Y	X	PFLAG	RBANK	W0	W1	W2	W3	W4	W5	W6	W7
9	@HL	@YZ	-	PCL	PCH	OSCM	WDTR	INTRQ0	INTRQ1	-	INTEN0	INTEN1	P0OC	P1OC	P1W	PEDGE
A	P0M	P1M	-	-	P4M	P5M	P0	P1	-	-	P4	P5	P0UR	P1UR	-	-
B	P4UR	P5UR	T0M	T0C	TC0M	TC0C	TC0R	TC0D	TC1M	TC1C	TC1R	TC1D	TC2M	TC2C	TC2R	TC2D
C	T1M	T1CL	T1CH	CPTM	CPTCL	CPTCH	P4CON	P5CON	ADM	ADB	ADR	ADT	-	-	-	-
D	-	-	-	-	-	-	-	-	-	-	-	PECMD	PE ROML	PE ROMH	PE RAML	PERAM CNT
E	SIOM	SIOR	SIOB	SIOC	URTX	URRX	URCR	UTXD	URXD	-	MSP STAT	MSPM1	MSPM2	MSP BUF	MSP ADR	STKP
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.2.1.2系统寄存器说明

H, L = 工作寄存器，@HL 间接寻址寄存器
R = 工作寄存器，ROM 查表数据缓存器
X = 工作寄存器，ROM 地址寄存器
RBANK = RAM bank 选择寄存器
P1W = P1 唤醒功能寄存器
PEDGE = P0.0，P0.1 触发方向控制寄存器
URTX = UART 发送控制寄存器
URRX = UART 接收控制寄存器
URCR = UART 波特率控制寄存器
UTXD = UART 发送数据缓存器.
T1CH, L = T1 计数寄存器
ADM = ADC 模式寄存器
ADR = ADC 精度选择寄存器
PEDGE = P0.0，P0.1，P0.2 触发方向控制寄存器
INTEN0,1 = 中断使能寄存器
PnM = Pn 输入/输出模式寄存器
PnUR = Pn 上拉电阻控制寄存器
PCH, PCL = 程序计数器
T0C = T0 计数寄存器
TCnC = TCn 计数寄存器
TCnD= TCn 占空比控制寄存器
CPTCL,H= 捕捉定时器计数寄存器
MSPBUF= MSP 缓存器
MSPADR= MSP 地址寄存器
PECMD= ISP 命令寄存器
PEROM= ISP ROM 地址
@HL = 间接寻址寄存器
STKP = 堆栈指针寄存器

Y, Z = 工作寄存器，@YZ 间接寻址寄存器，ROM 地址寄存器
PFLAG = 特殊标志寄存器
W0~W7= 工作寄存器
P0OC,P1OC = 漏极开路控制寄存器
SIOM = SIO 模式控制寄存器
SIOR = SIO 时钟控制寄存器
SIOB = SIO 数据缓存器
SIOC = SIO 控制寄存器
T1M = T1 模式寄存器
URXD = UART 接收数据缓存器
P4CON,P5CON = P4，P5 配置寄存器
ADB = ADC 数据缓存器
ADT = ADC 偏移校准寄存器
INTRQ0,1 = 中断请求寄存器
WDTR = 看门狗清零寄存器
Pn = Pn 数据缓存器
OSCM = 时钟模式寄存器
T0M = T0 模式寄存器
TCnM = TCn 模式寄存器
TCnR = TCn 自动重装数据缓存器
CPTM= 捕捉定时器控制寄存器
MSPSTAT= MSP 状态寄存器
MSPM1= MSP 模式寄存器
MSPM2= MSP 模式寄存器 2
PERAM= ISP RAM 分配地址
PERAMCNT= ISP RAM 编程计数寄存器
@YZ = 间接寻址寄存器
STK0~STK7 = 堆栈缓存器

2.2.1.3寄存器的位定义

地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	注释
080H	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0	R/W	L
081H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0	R/W	H
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
085H	XBIT7	XBIT6	XBIT5	XBIT4	XBIT3	XBIT2	XBIT1	XBIT0	R/W	X
086H	POR	WDT	RST	STKOV		C	DC	Z	R/W	PFLAG
087H							RBANKS1	RBANKS0	R/W	RBANK
088H	W0BIT7	W0BIT6	W0BIT5	W0BIT4	W0BIT3	W0BIT2	W0BIT1	W0BIT0	R/W	W0
089H	W1BIT7	W1BIT6	W1BIT5	W1BIT4	W1BIT3	W1BIT2	W1BIT1	W1BIT0	R/W	W1
08AH	W2BIT7	W2BIT6	W2BIT5	W2BIT4	W2BIT3	W2BIT2	W2BIT1	W2BIT0	R/W	W2
08BH	W3BIT7	W3BIT6	W3BIT5	W3BIT4	W3BIT3	W3BIT2	W3BIT1	W3BIT0	R/W	W3
08CH	W4BIT7	W4BIT6	W4BIT5	W4BIT4	W4BIT3	W4BIT2	W4BIT1	W4BIT0	R/W	W4
08DH	W5BIT7	W5BIT6	W5BIT5	W5BIT4	W5BIT3	W5BIT2	W5BIT1	W5BIT0	R/W	W5
08EH	W6BIT7	W6BIT6	W6BIT5	W6BIT4	W6BIT3	W6BIT2	W6BIT1	W6BIT0	R/W	W6
08FH	W7BIT7	W7BIT6	W7BIT5	W7BIT4	W7BIT3	W7BIT2	W7BIT1	W7BIT0	R/W	W7
090H	@HL7	@HL6	@HL5	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL
091H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
093H	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
094H				PC12	PC11	PC10	PC9	PC8	R/W	PCH
095H				CPUM1	CPUM0	CLKMD	STPHX		R/W	OSCM
096H	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
097H	ADCIRQ	T1IRQ	TC2IRQ	TC1IRQ	TC0IRQ	T0IRQ	P01IRQ	P00IRQ	R/W	INTRQ0
098H				MSPIRQ	UTXIRQ	URXIRQ	SIOIRQ	WAKEIRQ	R/W	INTRQ1
09AH	ADCIEN	T1IEN	TC2IEN	TC1IEN	TC0IEN	T0IEN	P01IEN	P00IEN	R/W	INTEN0
09BH				MSPIEN	UTXIEN	URXIEN	SIOIEN	WAKEIEN	R/W	INTEN1
09CH							P03OC	P02OC	R/W	P0OC
09DH	P17OC	P16OC	P15OC	P14OC	P13OC	P12OC	P11OC	P10OC	R/W	P1OC
09EH	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W	R/W	P1W
09FH					P01G1	P01G0	P00G1	P00G0	R/W	PEDGE
0A0H		P06M	P05M	P04M	P03M	P02M	P01M	P00M	R/W	P0M
0A1H	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0A4H	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M	R/W	P4M
0A5H					P53M	P52M	P51M	P50M	R/W	P5M
0A6H		P06	P05	P04	P03	P02	P01	P00	R/W	P0
0A7H	P17	P16	P15	P14	P13	P12	P11	P10	R/W	P1
0AAH	P47	P46	P45	P44	P43	P42	P41	P40	R/W	P4
0ABH					P53	P52	P51	P50	R/W	P5
0ACH		P06UR	P05UR	P04UR	P03UR	P02UR	P01UR	P00UR	R/W	P0UR
0ADH	P17UR	P16UR	P15UR	P14UR	P13UR	P12UR	P11UR	P10UR	R/W	P1UR
0B0H	P47UR	P46UR	P45UR	P44UR	P43UR	P42UR	P41UR	P40UR	R/W	P4UR
0B1H					P53UR	P52UR	P51UR	P50UR	R/W	P5UR
0B2H	T0ENB	T0rate2	T0rate1	T0rate0				T0TB	R/W	T0M
0B3H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0B4H	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS1	TC0CKS0		PWM0OUT	R/W	TC0M
0B5H	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0B6H	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0	W	TC0R
0B7H	TC0D7	TC0D6	TC0D5	TC0D4	TC0D3	TC0D2	TC0D1	TC0D0	R/W	TC0D
0B8H	TC1ENB	TC1rate2	TC1rate1	TC1rate0	TC1CKS1	TC1CKS0		PWM1OUT	R/W	TC1M
0B9H	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0	R/W	TC1C
0BAH	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0	W	TC1R
0BBH	TC1D7	TC1D6	TC1D5	TC1D4	TC1D3	TC1D2	TC1D1	TC1D0	R/W	TC1D
0BCH	TC2ENB	TC2rate2	TC2rate1	TC2rate0	TC2CKS1	TC2CKS0		PWM2OUT	R/W	TC2M
0BDH	TC2C7	TC2C6	TC2C5	TC2C4	TC2C3	TC2C2	TC2C1	TC2C0	R/W	TC2C
0BEH	TC2R7	TC2R6	TC2R5	TC2R4	TC2R3	TC2R2	TC2R1	TC2R0	W	TC2R
0BFH	TC2D7	TC2D6	TC2D5	TC2D4	TC2D3	TC2D2	TC2D1	TC2D0	R/W	TC2D
0C0H	T1ENB	T1rate2	T1rate1	T1rate0	T1CKS				R/W	T1M
0C1H	T1C7	T1C6	T1C5	T1C4	T1C3	T1C2	T1C1	T1C0	R/W	T1CL
0C2H	T1C15	T1C14	T1C13	T1C12	T1C11	T1C10	T1C9	T1C8	R/W	T1CH
0C3H	CPTEN				CPTMD	CPTStart	CPTG1	CPTG0	R/W	CPTM
0C4H	CPTC7	CPTC6	CPTC5	CPTC4	CPTC3	CPTC2	CPTC1	CPTC0	R/W	CPTCL
0C5H	CPTC15	CPTC14	CPTC13	CPTC12	CPTC11	CPTC10	CPTC9	CPTC8	R/W	CPTCH
0C6H	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0	R/W	P4CON
0C7H					P5CON3	P5CON2	P5CON1	P5CON0	R/W	P5CON
0C8H	ADENB	ADS	EOC	GCHS	CHS3	CHS2	CHS1	CHS0	R/W	ADM
0C9H	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	R	ADB
0CAH		ADCKS1	ADLEN	ADCKS0			ADB1	ADB0	R/W	ADR
0CBH	ADTS1	ADTS0		ADT4	ADT3	ADT2	ADT1	ADT0	R/W	ADT
0DBH	PECMD7	PECMD6	PECMD5	PECMD4	PECMD3	PECMD2	PECMD1	PECMD0	R/W	PECMD
0DCH	PEROML7	PEROML6	PEROML5	PEROML4	PEROML3	PEROML2	PEROML1	PEROML0	R/W	PEROML

0DDH	PEROMH7	PEROMH6	PEROMH5	PEROMH4	PEROMH3	PEROMH2	PEROMH1	PEROMH0	R/W	PEROMH
0DEH	PERAML7	PERAML6	PERAML5	PERAML4	PERAML3	PERAML2	PERAML1	PERAML0	R/W	PERAML
0DFH	PERAMCNT7	PERAMCNT6	PERAMCNT5	PERAMCNT4	PERAMCNT3		PERAML9	PERAML8	R/W	PERAMCNT
0E0H	SENB	START	SRATE1	SRATE0	MLSB	SCLKMD	CPOL	CPHA	R/W	SIOM
0E1H	SIOR7	SIOR6	SIOR5	SIOR4	SIOR3	SIOR2	SIOR1	SIOR0	W	SIOR
0E2H	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0	R/W	SIOB
0E3H						SIOBZ	SCSEN	SCSP	R/W	SIOC
0E4H	UTXEN	UTXPEN	UTXPS	UTXBRK	URXBZ	UTXBZ			R/W	URTX
0E5H	URXEN	URXPEN	URXPS	URXPC	UFMER	URS2	URS1	URS0	R/W	URRX
0E6H	URCR7	URCR6	URCR5	URCR4	URCR3	URCR2	URCR1	URCR0	R/W	URCR
0E7H	UTXD7	UTXD6	UTXD5	UTXD4	UTXD3	UTXD2	UTXD1	UTXD0	R/W	UTXD
0E8H	URXD7	URXD6	URXD5	URXD4	URXD3	URXD2	URXD1	URXD0	R/W	URXD
0EAH	-	CKE	D_A	P	S	RED_WRT	-	BF	R	MSPSTAT
0EBH	WCOL	MSPOV	MSPENB	CKP	SLRXCKP	MSPWK	-	MSPC	R/W	MSPM1
0ECH	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	R/W	MSPM2
0EDH	MSPBUF7	MSPBUF6	MSPBUF5	MSPBUF4	MSPBUF3	MSPBUF2	MSPBUF1	MSPBUF0	R/W	MSPBUF
0EEH	MSPADR7	MSPADR6	MSPADR5	MSPADR4	MSPADR3	MSPADR2	MSPADR1	MSPADR0	R/W	MSPADR
0EFH	GIE	LVD24	LVD33			STKPB2	STKPB1	STKPB0	R/W	STKP
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H				S7PC12	S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H				S6PC12	S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H				S5PC12	S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H				S4PC12	S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H				S3PC12	S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH				S2PC12	S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH				S1PC12	S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH				S0PC12	S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

✷ 注：
1、为了避免系统错误，在初始化时，请将上表所有寄存器的位都按照设计要求设置为确定的"1"或者"0"；
2、所有寄存器名都已在 SN8ASM 编译器中做了宣告；
3、用户使用 SN8ASM 编译器对寄存器的位进行操作时，须在该寄存器的位前加“F”；
4、指令“b0bset”，“b0bclr”，“bset”，“bclr”只能用于可读写的寄存器（“R/W”）。

2.2.2 累加器ACC

8 位数据寄存器 ACC 用来执行 ALU 与数据存储器之间数据的传送操作。如果操作结果为零（Z）或有进位产生（C 或 DC），程序状态寄存器 PFLAG 中相应位会发生变化。

ACC 并不在 RAM 中，因此在立即寻址模式中不能用“B0MOV”指令对其进行读写。

➢ **例：读/写 ACC。**

```
; 将立即数写入 ACC。
        MOV     A, #0FH

;把 ACC 中的数据存入 BUF 中。
        MOV     BUF, A
        B0MOV   BUF, A

; 把 BUF 中的数据送到 ACC 中。
        MOV     A, BUF
        B0MOV   A, BUF
```

中断时硬件自动保存 ACC 和工作寄存器（80H~8FH）。

➢ **例：ACC 和工作寄存器中断保护操作。**

```
INT_SERVICE:
                                ; 保存 ACC。
                                ; 保存工作寄存器。
        …
        …                       ; 恢复工作寄存器。
                                ; 恢复 ACC。

        RETI                    ; 退出中断。
```

2.2.3 程序状态寄存器PFLAG

寄存器 PFLAG 中包含 ALU 运算状态信息、系统复位状态信息和 LVD 低电压检测状态信息。其中，POR、WDT 和 RST 位显示系统的复位状态信息，包括上电复位、LVD 复位，外部复位和看门狗复位。C、DC 和 Z 位显示 ALU 的运算信息。LVD24、LVD33 位显示 LVD 检测低电压的状态。

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	POR	WDT	RST	STKOV	-	C	DC	Z
读/写	R	R	R	R	-	R/W	R/W	R/W
复位后	-	-	-	-	-	0	0	0

Bit 7 **POR**：上电复位和掉电复位显示位。
0 = 没有复位；
1 = 复位，LVD 显示复位标志。

Bit 6 **WDT**：看门狗复位显示位。
0 = 没有复位；
1 = 复位，看门狗显示复位标志。

Bit 5 **RST**：外部复位显示位。
0 = 没有复位；
1 = 复位，外部复位显示复位标志。

Bit 4 **STKOV**：堆栈溢出显示位。
0 = 没有溢出；
1 = 堆栈溢出。

Bit 2 **C**：进位标志。
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；
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。

Bit 1 **DC**：辅助进位标志。
1 =加法运算时低四位有进位，或减法运算后没有向高四位借位；
0 =加法运算时低四位没有进位，或减法运算后有向高四位借位。

Bit 0 **Z**：零标志。
1 =算术/逻辑/分支转移运算的结果为零；
0 =算术/逻辑/分支转移运算的结果非零。

0EFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	LVD24	LVD33	-	-	STKPB2	STKPB1	STKPB0
读/写	R/W	R	R	-	-	R/W	R/W	R/W
复位后	0	-	-	-	-	1	1	1

Bit 6 **LVD24**：LVD24 低电压检测指示位。
0 = Vdd > LVD24 检测电平；
1 = Vdd < LVD24 检测电平。

Bit 5 **LVD33**：LVD33 低电压检测指示位。
0 = Vdd > LVD33 检测电平；
1 = Vdd < LVD33 检测电平。

✷ **注：关于标志位 C、DC 和 Z 的更多信息请参阅指令集相关内容。**

2.2.4 程序计数器PC

程序计数器 PC 是一个 13 位二进制程序地址寄存器，分高 5 位和低 8 位。专门用来存放下一条需要执行指令的地址。通常，程序计数器会随程序中指令的执行自动增加。

若程序执行 CALL 和 JMP 指令时，PC 指向特定的地址。

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC	-	-	-	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
复位后	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
	PCH								PCL							

☞ **单地址跳转**

在 SONiX 单片机里面，有 9 条指令（CMPRS、INCS、INCMS、DECS、DECMS、BTS0、BTS1、B0BTS0 和 B0BTS1）可完成单地址跳转功能。如果这些指令执行结果为真，那么 PC 值加 2 以跳过下一条指令。

如果位测试为真，PC 加 2。

```
                B0BTS1      FC          ; 若 Carry_flag = 1 则跳过下一条指令。
                JMP         C0STEP      ; 否则执行 C0STEP。
                …
C0STEP:         NOP

                B0MOV       A, BUF0     ; BUF0 送入 ACC。
                B0BTS0      FZ          ; Zero flag = 0 则跳过下一条指令。
                JMP         C1STEP      ; 否则执行 C1STEP。
                …
C1STEP:         NOP
```

如果 ACC 等于指定的立即数则 PC 值加 2，跳过下一条指令。

```
                CMPRS       A, #12H
                JMP         C0STEP
                …
C0STEP:         NOP
```

执行加 1 指令后，结果为零时，PC 的值加 2，跳过下一条指令。

INCS：

```
                INCS        BUF0
                JMP         C0STEP      ;如果 ACC 不为“0”，则跳至 C0STEP。
                …
C0STEP:         NOP
```

INCMS：

```
                INCMS       BUF0
                JMP         C0STEP      ;如果 BUF0 不为“0”，则跳至 C0STEP。
                …
C0STEP:         NOP
```

执行减 1 指令后，结果为零时，PC 的值加 2，跳过下一条指令。

DECS：

```
                DECS        BUF0
                JMP         C0STEP      ;如果 ACC 不为“0”，则跳至 C0STEP。
                …
C0STEP:         NOP
```

DECMS：

```
                DECMS       BUF0
                JMP         C0STEP      ;如果 BUF0 不为“0”，则跳至 C0STEP。
                …
C0STEP:         NOP
```

☞ **多地址跳转**

执行 JMP 或 ADD M,A（M=PCL）指令可实现多地址跳转。执行 ADD M, A、ADC M, A 或 B0ADD M, A 后，若 PCL 溢出，PCH 会自动进位。对于跳转表及其它应用，用户可以通过上述 3 条指令计算 PC 的值而不需要担心 PCL 溢出的问题。

✷ **注：PCH 仅支持 PC 的递增运算而不支持递减运算。当 PCL+ACC 执行完 PCL 有进位时，PCH 会自动加 1；但执行 PCL-ACC 有借位发生，PCH 的值会保持不变。**

➢ **例：PC = 0323H（PCH = 03H，PCL = 23H）。**

```
; PC = 0323H
                MOV         A, #28H
                B0MOV       PCL, A          ; 跳到地址 0328H。
                …

; PC = 0328H
                MOV         A, #00H
                B0MOV       PCL, A          ; 跳到地址 0300H。
                …
```

➢ **例：PC = 0323H（PCH = 03H，PCL = 23H）。**

```
; PC = 0323H
                B0ADD       PCL, A          ; PCL = PCL + ACC，PCH 的值不变。
                JMP         A0POINT         ; ACC = 0，跳到 A0POINT。
                JMP         A1POINT         ; ACC = 1，跳到 A1POINT。
                JMP         A2POINT         ; ACC = 2，跳到 A2POINT。
                JMP         A3POINT         ; ACC = 3，跳到 A3POINT。
                …
                …
```

2.2.5 H, L寄存器

寄存器 H 和 L 都是 8 位缓存器，主要有以下两个功能：

- **通用工作寄存器；**
- **RAM 数据寻址指针@HL。**

081H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

080H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
L	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

➢ **例：用 H、L 作为数据指针，访问 bank0 中 020H 处的内容。**

```
B0MOV   H, #00H
B0MOV   L, #20H
B0MOV   A, @HL
```

➢ **例：对 bank 0 中的数据进行清零处理。**

```
        CLR     H           ; H = 0，指向 bank 0。
        B0MOV   L, #7FH     ; L = 7FH。
CLR_HL_BUF:
        CLR     @HL         ; @HL 清零。
        DECMS   L           ; L – 1，如果 L = 0，程序结束。
        JMP     CLR_HL_BUF

        CLR     @HL
END_CLR:
        …
        …
```

2.2.6 X寄存器

8 位缓存器 X 寄存器是工作寄存器。

085H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	XBIT7	XBIT6	XBIT5	XBIT4	XBIT3	XBIT2	XBIT1	XBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

2.2.7 Y, Z寄存器

寄存器 Y 和 Z 都是 8 位缓存器，主要用途如下：

- **通用工作寄存器；**
- **RAM 数据寻址指针@YZ；**
- **配合指令 MOVC 对 ROM 数据进行查表。**

084H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Y	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

083H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Z	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

➢ **例：用 Y、Z 作为数据指针，访问 bank0 中 025H 处的内容。**

```
B0MOV   Y, #00H     ; Y 指向 RAM bank 0。
B0MOV   Z, #25H     ; Z 指向 25H。
B0MOV   A, @YZ      ; 数据送入 ACC。
```

➢ **例：利用数据指针@YZ 对 RAM 数据清零。**

```
        B0MOV   Y, #0       ; Y = 0，指向 bank 0。
        B0MOV   Z, #7FH     ; Z = 7FH，RAM 区的最后单元。

CLR_YZ_BUF:
        CLR     @YZ         ; @YZ 清零。

        DECMS   Z           ;
        JMP     CLR_YZ_BUF  ; 不为零。

        CLR     @YZ
END_CLR:                    ;
        …
```

2.2.8 R寄存器

8 位缓存器 R 主要有以下两个功能：

- **通用工作寄存器使用；**
- **存储执行查表指令后的高字节数据。**（执行 MOVC 指令，指定 ROM 单元的高字节数据会被存入 R 寄存器而低字节数据则存入 ACC。）

082H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

✷ **注：关于 R 寄存器的查表功能，请参考“查表”章节的说明。**

2.2.9 W寄存器

8 位缓存器 W 寄存器包括 W0~W7，主要有以下 2 个功能：

- **汇编语言时作为通用工作寄存器；**
- **C 语言时作为程序缓存器。**

088H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W0	W0BIT7	W0BIT6	W0BIT5	W0BIT4	W0BIT3	W0BIT2	W0BIT1	W0BIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

089H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W1	W1BIT7	W1BIT6	W1BIT5	W1BIT4	W1BIT3	W1BIT2	W1BIT1	W1BIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

08AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W2	W2BIT7	W2BIT6	W2BIT5	W2BIT4	W2BIT3	W2BIT2	W2BIT1	W2BIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

08BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W3	W3BIT7	W3BIT6	W3BIT5	W3BIT4	W3BIT3	W3BIT2	W3BIT1	W3BIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

08CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W4	W4BIT7	W4BIT6	W4BIT5	W4BIT4	W4BIT3	W4BIT2	W4BIT1	W4BIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

08DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W5	W5BIT7	W5BIT6	W5BIT5	W5BIT4	W5BIT3	W5BIT2	W5BIT1	W5BIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

08EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W6	W6BIT7	W6BIT6	W6BIT5	W6BIT4	W6BIT3	W6BIT2	W6BIT1	W6BIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

08FH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W7	W7BIT7	W7BIT6	W7BIT5	W7BIT4	W7BIT3	W7BIT2	W7BIT1	W7BIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

✷ 注：
1、汇编语言时，W0~W7 作为通用工作寄存器；
2、C 语言时，W0~W7 被保留，强烈建议不要通过程序访问 W0~W7。

2.3 寻址模式

2.3.1 立即寻址

将立即数直接送入 ACC 或指定的 RAM 单元。

- **例：立即数 12H 送入 ACC。**

```
MOV     A, #12H
```

- **例：立即数 12H 送入寄存器 R。**

```
B0MOV   R, #12H
```

✷ **注：立即寻址模式中，指定的 RAM 单元必须是 80H~8FH 的工作寄存器。**

2.3.2 直接寻址

通过 ACC 对 RAM 单元数据进行操作。

- **例：地址 12H 处的内容送入 ACC。**

```
B0MOV   A, 12H
```

- **例：ACC 中数据写入 RAM 中 12H 单元。**

```
B0MOV   12H, A
```

2.3.3 间接寻址

通过数据指针（H/L，Y/Z）对数据存储单元进行读写。

- **例：通过指针@HL 间接寻址。**

```
B0MOV   H, #0       ; 清“H”以寻址 RAM bank 0。
B0MOV   L, #12H     ; 设定寄存器地址。
B0MOV   A, @HL
```

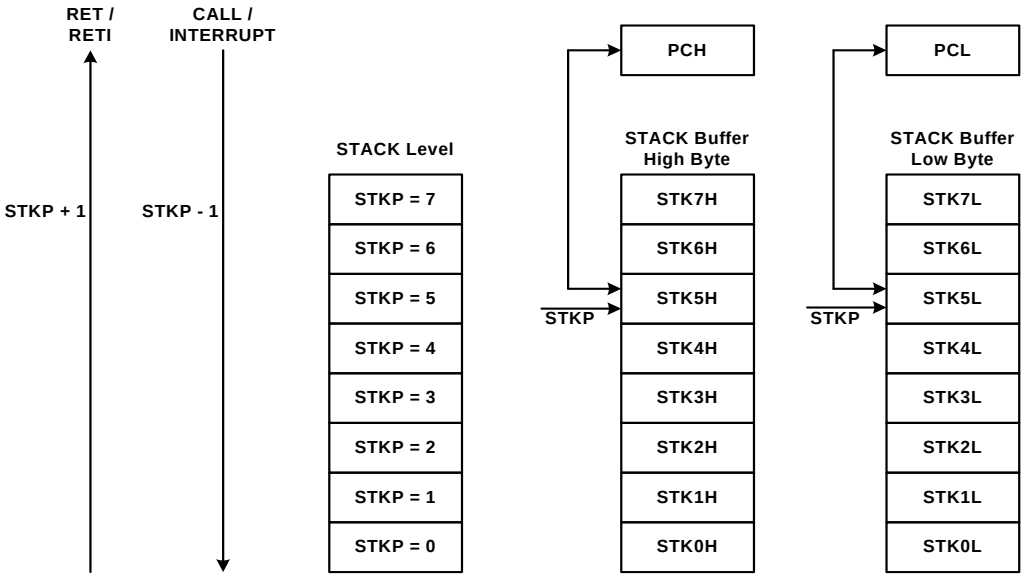
- **例：通过指针@YZ 间接寻址。**

```
B0MOV   Y, #0       ; 清“Y”以寻址 RAM bank 0。
B0MOV   Z, #12H     ; 设定寄存器地址。
B0MOV   A, @YZ
```

2.4 堆栈操作

2.4.1 概述

SN8F27E60 系列的堆栈缓存器共有 8 层，程序进入中断或执行 CALL 指令时，用来存储程序计数器 PC 的值。寄存器 STKP 为堆栈指针，指向堆栈缓存器顶层，STKnH 和 STKnL 分别是各堆栈缓存器高、低字节。



2.4.2 堆栈指针

堆栈指针 STKP 是一个 3 位寄存器，存放被访问的堆栈单元地址，13 位数据存储器 STKnH 和 STKnL 用于暂存堆栈数据。堆栈操作遵循后进先出（LIFO）的原则，入栈时堆栈指针 STKP 的值减 1，出栈时 STKP 的值加 1，这样，STKP 总是指向堆栈缓存器顶层单元，写入最后的 PC 值到堆栈缓存器。

0EFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	LVD24	LVD33	-	-	STKPB2	STKPB1	STKPB0
读/写	R/W	R	R	-	-	R/W	R/W	R/W
复位后	0	-	-	-	-	1	1	1

Bit[2:0] **STKPBn**：堆栈指针（n = 0 ~ 2）。

Bit 7 **GIE**：全局中断控制位。
0 = 禁止；
1 = 允许。

➢ **例：系统复位时，堆栈指针寄存器内容为默认值，但强烈建议在程序初始部分重新设定，如下面所示：**

```
MOV     A, #00000111B
B0MOV   STKP, A
```

2.4.3 堆栈缓存器

在执行 CALL 指令或者中断服务程序之前，先将程序计数器 PC 的值存入堆栈缓存器。堆栈操作遵循后进先出（LIFO）的原则，堆栈指针（STKP）和堆栈缓存器（STKnH 和 STKnl）都位于 Bank0 区域的系统寄存器内。

0F0H~0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnH	-	-	-	SnPC12	SnPC11	SnPC10	SnPC9	SnPC8
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	0	0	0

0F0H~0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnL	SnPC7	SnPC6	SnPC5	SnPC4	SnPC3	SnPC2	SnPC1	SnPC0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

STKn = STKnH , STKnL (n = 7 ~ 0)

2.4.4 堆栈溢出指示

堆栈指针正常工作时，表示程序正常运行；若堆栈溢出，则表示程序运行错误。STKOV 位是堆栈指针的溢出指示位，用量监控堆栈指针的工作状态。若 STKOV=0，则堆栈指针正常工作；若 STKOV=1，则堆栈溢出，程序运行错误。

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	POR	WDT	RST	STKOV	-	C	DC	Z
读/写	R	R	R	R	-	R/W	R/W	R/W
复位后	-	-	-	-	-	0	0	0

Bit 4 **STKOV**：堆栈溢出指示位。
0 = 堆栈没有溢出；
1 = 堆栈溢出。

✷ **注：STKOV 显示堆栈的溢出状态，仅在系统复位时才会清 STKOV 位，如看门狗定时器溢出、外部复位好 LVD 复位。**

➢ **例：通过看门狗复位保护堆栈溢出，必须使能看门狗定时器。**

```
MAIN:
                …
StackChk:
                B0BTS1      STKOV
                JMP         MAIN        ; STKOV=0，程序继续运行。
                JMP         $           ; STKOV=1，堆栈溢出，使用“jump here”命令。
                                        ；看门狗定时器溢出触发系统复位。
```

➢ **例：通过外部复位保护堆栈溢出，必须使能外部复位功能，将一个 GPIO 引脚（输出模式）连接到外部复位引脚。**

```
MAIN:
                …
StackChk:
                B0BTS1      STKOV
                JMP         MAIN        ; STKOV=0，程序继续运行。
                B0BCLR      P1.0        ; STKOV=1，堆栈溢出，设置 P1.0 输出低状态，强制复位引脚输出低以
                                        触发系统复位。
```

2.4.5 堆栈操作举例

执行程序调用指令 CALL 和响应中断服务时，堆栈指针 STKP 的值减 1，指针指向下一个堆栈缓存器。同时，对程序计数器 PC 的内容进行入栈保护。入栈操作如下表所示：

堆栈层数	STKP 寄存器			堆栈缓存器		STKOV	说明
	STKPB2	STKPB1	STKPB0	高字节	低字节		
0	1	1	1	Free	Free	0	-
1	1	1	0	STK0H	STK0L	0	-
2	1	0	1	STK1H	STK1L	0	-
3	1	0	0	STK2H	STK2L	0	-
4	0	1	1	STK3H	STK3L	0	-
5	0	1	0	STK4H	STK4L	0	-
6	0	0	1	STK5H	STK5L	0	-
7	0	0	0	STK6H	STK6L	0	-
8	1	1	1	STK7H	STK7L	0	-
> 8	1	1	0	-	-	1	堆栈溢出，出错

对应每个入栈操作，都有一个出栈操作来恢复程序计数器 PC 的值。RETI 指令用于中断服务程序中，RET 用于子程序调用。出栈时，STKP 加 1 并指向下一个空闲堆栈缓存器。堆栈恢复操作如下表所示：

堆栈层数	STKP 寄存器			堆栈缓存器		STKOV	说明
	STKPB2	STKPB1	STKPB0	高字节	低字节		
8	1	1	1	STK7H	STK7L	0	-
7	0	0	0	STK6H	STK6L	0	-
6	0	0	1	STK5H	STK5L	0	-
5	0	1	0	STK4H	STK4L	0	-
4	0	1	1	STK3H	STK3L	0	-
3	1	0	0	STK2H	STK2L	0	-
2	1	0	1	STK1H	STK1L	0	-
1	1	1	0	STK0H	STK0L	0	-
0	1	1	1	Free	Free	0	-

✷ **注：堆栈溢出时，系统会检测到并设置 STKOV 标志位（逻辑 1），程序不能将 STKOV 位清零。**

2.5 编译选项列表（CODE OPTION）

编译选项（CODE OPTION）是一种系统的硬件配置，包括振荡器的类型，杂讯滤波器的选项，看门狗定时器的操作，LVD 选项，复位引脚选项以及 Flash ROM 的安全控制。如下表所示：

编译选项	配置项目	功能说明
High_Clk	IHRC_16M	内部高速 16MHz RC 振荡器，XIN/XOUT 为普通的 GPIO 引脚。
	IHRC_RTC	高速 16MHz RC 振荡器，XIN/XOUT 引脚连接外部 32768Hz 晶振。
	RC	外部高速振荡器采用低成本的 RC 振荡电路，XIN 引脚连接 RC 电路，XOUT 引脚为 GPIO 引脚。
	32K X'tal	外部高速振荡器采用低频振荡器（如 32.768KHz）。
	12M X'tal	外部高速振荡器采用高速陶瓷/石英振荡器（如 12MHz）。
	4M X'tal	外部高速振荡器采用标准陶瓷/石英振荡器（如 4MHz）。
High_Fcpu	Fhosc/1	普通模式下指令周期为 1 个高速振荡时钟周期。
	Fhosc/2	普通模式下指令周期为 2 个高速振荡时钟周期。
	Fhosc/4	普通模式下指令周期为 4 个高速振荡时钟周期。
	Fhosc/8	普通模式下指令周期为 8 个高速振荡时钟周期。
	Fhosc/16	普通模式下指令周期为 16 个高速振荡时钟周期。
	Fhosc/32	普通模式下指令周期为 32 个高速振荡时钟周期。
	Fhosc/64	普通模式下指令周期为 64 个高速振荡时钟周期。
	Fhosc/128	普通模式下指令周期为 128 个高速振荡时钟周期。
Low_Fcpu	Flosc/1	低速模式学指令周期为 1 个低速振荡时钟周期。
	Flosc/2	低速模式学指令周期为 2 个低速振荡时钟周期。
	Flosc/4	低速模式学指令周期为 4 个低速振荡时钟周期。
	Flosc/8	低速模式学指令周期为 8 个低速振荡时钟周期。
Noise_Filter	Enable	开启杂讯滤波器。
	Disable	关闭杂讯滤波器。
WDT_CLK	Flosc/4	看门狗定时器时钟源为 Flosc/4。
	Flosc/8	看门狗定时器时钟源为 Flosc/8。
	Flosc/16	看门狗定时器时钟源为 Flosc/16。
	Flosc/32	看门狗定时器时钟源为 Flosc/32。
Watch_Dog	Always_On	始终开启看门狗定时器，睡眠模式和绿色模式下也不例外。
	Enable	开启看门狗定时器，但在睡眠模式和绿色模式会处于关闭状态。
	Disable	关闭看门狗定时器。
Reset_Pin	Reset	使能外部复位引脚。
	P04	使能 P0.4。
Security	Enable	ROM 代码加密。
	Disable	ROM 代码不加密。
LVD	LVD_L	VDD 低于 1.8V 时，LVD 复位系统。
	LVD_M	VDD 低于 1.8V 时，LVD 复位系统； 使能 PFLAG 寄存器的 LVD24 标志位以开启 2.4V 的低电压检测功能。
	LVD_H	VDD 低于 2.4V 时，LVD 复位系统； 使能 PFLAG 寄存器的 LVD33 标志位以开启 3.3V 的低电压检测功能。
	LVD_MAX	VDD 低于 3.3V 时，LVD 复位系统。

2.5.1 Fcpu编译选项

Fcpu 指在高速/低速操作模式下的指令周期。High_Fcpu 和 Low_Fcpu 编译选项选择指令周期的分频等级，以决定指令周期的分频。普通模式（高速模式）下，系统始终源由高速振荡器提供，Fcpu 共有 8 个选项：Fhosc/1，Fhosc/2，Fhosc/4，Fhosc/8，Fhosc/16，Fhosc/32，Fhosc/64，Fhosc/128。低速模式下，系统时钟源由内部低速 RC 振荡电路提供，Fcpu 有 4 个选项：Flosc/1，Flosc/2，Flosc/4，Flosc/8。

2.5.2 Reset_Pin编译选项

复位引脚与双向输入输出引脚共用，由编译选项控制。

- **Reset**：使能外部复位引脚功能。当下降沿触发时，系统复位。
- **P04**：使能 P0.4 为双向输入输出引脚。此时禁止外部复位引脚功能。

2.5.3 Security编译选项

Security 编译选项是对 Flash ROM 的一种保护，当使能 Secuity 编译选项，ROM 代码加密，可以保护 ROM 的内容。

2.5.4 Noise Filter编译选项

Noise Filter 编译选项是强杂讯滤除功能以减少杂讯对系统时钟的影响。如使能杂讯滤波器，在高干扰环境下，使能看门狗定时器且选择一个合适的 LVD 选项可以使整个系统更好的工作。

3 复位

3.1 概述

SN8F27E60 系列的单片机有以下几种复位方式：

- **上电复位；**
- **看门狗复位；**
- **掉电复位；**
- **外部复位（使能外部复位引脚时有效）。**

上述任一种复位发生时，所有的系统寄存器恢复默认状态，程序停止运行，同时程序计数器 PC 清零。复位结束后，系统从向量 0000H 处重新开始运行。PFLAG 寄存器的 POR、WDT 和 RST 标志位可以显示系统复位状态的信息。用户可以根据 POR、WDT 和 RST 的状态，编程控制系统的运行路径。

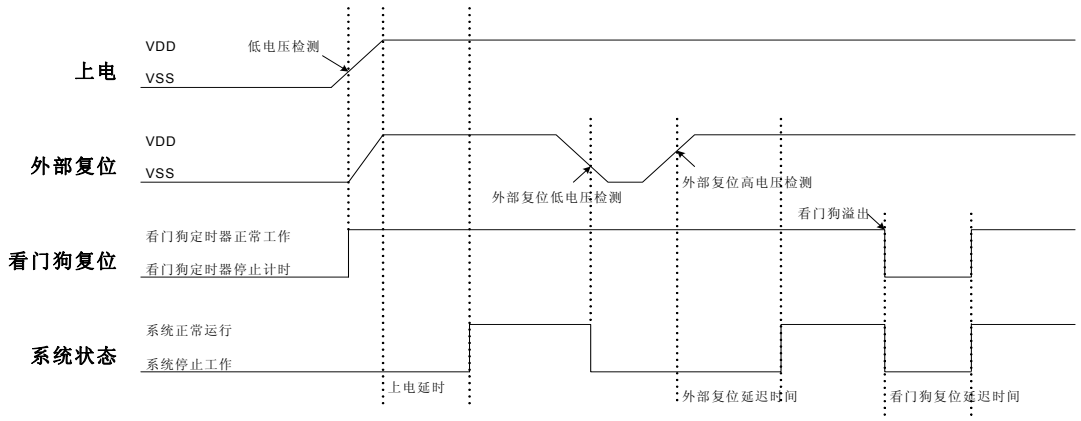
086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	POR	WDT	RST	STKOV	-	C	DC	Z
读/写	R	R	R	R	-	R/W	R/W	R/W
复位后	-	-	-	-	-	0	0	0

Bit 7 **POR**：上电复位和掉电复位显示位。
0 = 没有复位；
1 = 复位，LVD 显示复位标志。

Bit 6 **WDT**：看门狗复位显示位。
0 = 没有复位；
1 = 复位，看门狗显示复位标志。

Bit 5 **RST**：外部复位显示位。
0 = 没有复位；
1 = 复位，外部复位显示复位标志。

任何一种复位方式都需要一定的响应时间，系统提供完善的复位流程以保证复位动作的顺利进行。对于不同类型的振荡器，完成复位所需要的时间也不同。因此，VDD 的上升速度和不同晶振的起振时间都不固定。RC 振荡器的起振时间最短，晶体振荡器的起振时间则较长。在用户使用的过程中，应考虑系统对上电复位时间的要求。系统复位时序图如下：



3.2 上电复位

上电复位与 LVD 操作密切相关。系统上电过程呈逐渐上升的曲线形式，需要一定时间才能达到正常电平值。下面给出上电复位的正常时序：

- **上电：**系统检测到电源电压上升并等待其稳定；
- **外部复位（使能外部复位引脚时才有效）：**系统检测外部复位引脚状态。如果不为高电平，系统保持复位状态直到外部复位引脚的复位结束。
- **系统初始化：**所有的系统寄存器被置为默认状态；
- **振荡器开始工作：**振荡器开始提供系统时钟；
- **执行程序：**上电结束，程序开始运行。

3.3 看门狗复位

看门狗复位是系统的一种保护设置。在正常状态下，由程序将看门狗定时器清零。若出错，系统处于未知状态，看门狗定时器溢出，此时系统复位。看门狗复位后，系统重启进入正常状态。看门狗复位的时序如下：

- **看门狗定时器状态：**系统检测看门狗定时器是否溢出，若溢出，则系统复位；
- **系统初始化：**所有的系统寄存器被置为默认状态；
- **振荡器开始工作：**振荡器开始提供系统时钟；
- **执行程序：**上电结束，程序开始运行。

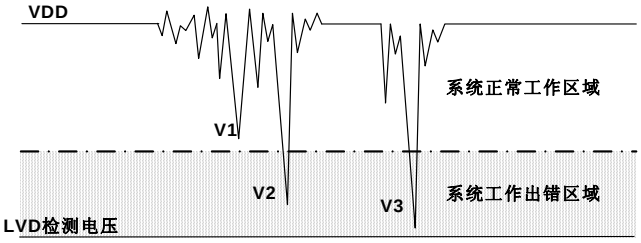
看门狗定时器应用注意事项如下：

- 对看门狗清零之前，检查 I/O 口的状态和 RAM 的内容可增强程序的可靠性；
- 不能在中断中对看门狗清零，否则无法侦测到主程序跑飞的状况；
- 程序中应该只在主程序中有一次清看门狗的动作，这种架构能够最大限度的发挥看门狗的保护功能。

✷ **注：关于看门狗定时器的详细内容，请参阅“看门狗定时器”有关章节。**

3.4 掉电复位

掉电复位针对外部因素引起的系统电压跌落情形（例如：干扰或外部负载的变化），掉电复位可能会引起系统工作状态不正常或程序执行错误。



掉电复位示意图

电压跌落可能会进入系统死区。系统死区意味着电源不能满足系统的最小工作电压要求。上图是一个典型的掉电复位示意图。图中，VDD 受到严重的干扰，电压值降的非常低。虚线以上区域系统正常工作，在虚线以下的区域内，系统进入未知的工作状态，这个区域称作死区。当 VDD 跌至 V1 时，系统仍处于正常状态；当 VDD 跌至 V2 和 V3 时，系统进入死区，则容易导致出错。以下情况系统可能进入死区：

DC 运用中：

DC 运用中一般都采用电池供电，当电池电压过低或单片机驱动负载时，系统电压可能跌落并进入死区。这时，电源不会进一步下降到 LVD 检测电压，因此系统维持在死区。

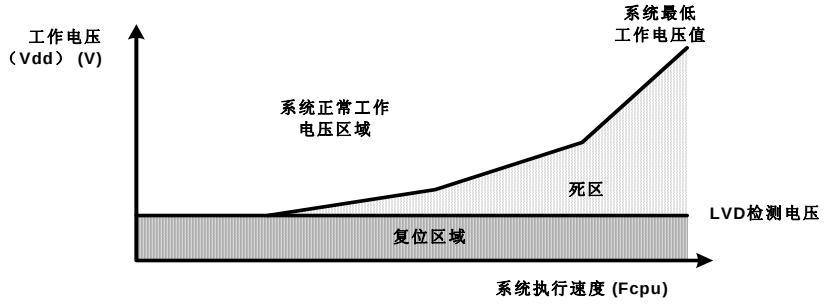
AC 运用中：

系统采用 AC 供电时，DC 电压值受 AC 电源中的噪声影响。当外部负载过高，如驱动马达时，负载动作产生的干扰也影响到 DC 电源。VDD 若由于受到干扰而跌落至最低工作电压以下时，则系统将有可能进入不稳定工作状态。

在 AC 运用中，系统上、下电时间都较长。其中，上电时序保护使得系统正常上电，但下电过程却和 DC 运用中情形类似，AC 电源关断后，VDD 电压在缓慢下降的过程中易进入死区。

3.4.1 系统最低工作电压

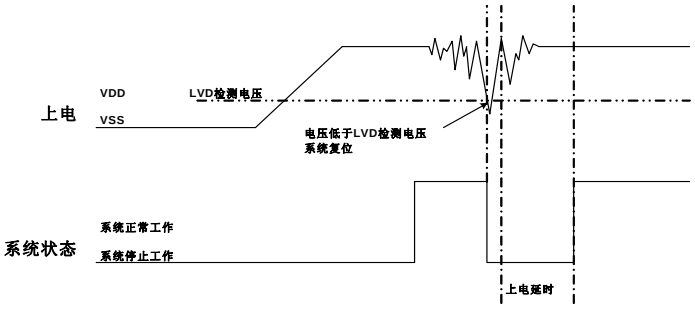
为了改善系统掉电复位的性能，首先必须明确系统具有的最低工作电压值。系统最低工作电压与系统执行速度有关，不同的执行速度下最低工作电压值也不同。



系统工作电压与执行速度关系图

如上图所示，系统正常工作电压区域一般高于系统复位电压，同时复位电压由低电压检测（LVD）电平决定。当系统执行速度提高时，系统最低工作电压也相应提高，但由于系统复位电压是固定的，因此在系统最低工作电压与系统复位电压之间就会出现一个电压区域，系统不能正常工作，也不会复位，这个区域即为死区。

3.4.2 低电压检测（LVD）



低电压检测（LVD）是 SONiX 8 位单片机内置的掉电复位保护装置，当 VDD 跌落并低于 LVD 检测电压值时，LVD 被触发，系统复位。不同的单片机有不同的 LVD 检测电平，LVD 检测电平值仅为一个电压点，并不能覆盖所有死区范围。因此采用 LVD 依赖于系统要求和环境状况。电源变化较大时，LVD 能够起到保护作用，如果电源变化触发 LVD，系统工作仍出错，则 LVD 就不能起到保护作用，就需要采用其它复位方法。

LVD 设计为三层结构（1.8V/2.4V/3.3V），由 LVD 编译选项控制。对于上电复位和掉电复位，1.8V LVD 始终处于使能状态；2.4V LVD 具有 LVD 复位功能，并能通过标志位显示 VDD 状态；3.3V LVD 具有标记功能，可显示 VDD 的工作状态。LVD 标志功能只是一个低电压检测装置，标志位 LVD24 和 LVD33 给出 VDD 的电压情况。对于低电压检测应用，只需查看 LVD24 和 LVD33 的状态即可检测电池状况。

0EFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	LVD24	LVD33	-	-	STKPB2	STKPB1	STKPB0
读/写	R/W	R	R	-	-	R/W	R/W	R/W
复位后	0	-	-	-	-	1	1	1

Bit 5 **LVD33**：LVD 3.3V 工作电压标志。
0 = Vdd > LVD33；
1 = Vdd < LVD33。

Bit 6 **LVD24**：LVD 2.4V 工作电压标志。
0 = Vdd > LVD24；
1 = Vdd < LVD24。

LVD	LVD Code Option			
	LVD_L	LVD_M	LVD_H	LVD_MAX
1.8V 复位	有效	有效	有效	有效
2.4V 标志	-	有效	-	-
2.4V 复位	-	-	有效	有效
3.3V 标志	-	-	有效	-
3.3V 复位	-	-	-	有效

LVD_L
如果 VDD < 1.8V，系统复位；
LVD24 和 LVD33 标志位无意义。
LVD_M
如果 VDD < 1.8V，系统复位；
LVD24：如果 VDD > 2.4V，LVD24 ＝0；如果 VDD <= 2.4V，LVD24＝1；
LVD33 标志位无意义。
LVD_H
如果 VDD < 2.4V，系统复位；
LVD33：如果 VDD > 3.3V，LVD33＝0；如果 VDD <= 3.3V，LVD33＝1；
LVD24 标志位无意义。
LVD_MAX
如果 VDD<3.3V，系统复位；
LVD24、LVD33 标志位无意义。

✷ 注：
a）LVD复位结束后，LVD24和LVD33都将被清零；
b）LVD 2.4V和LVD3.3V检测电平值仅作为设计参考，不能用作芯片工作电压值的精确检测。

3.4.3 掉电复位性能改进

如何改善系统掉电复位性能，有以下几点建议：

- **LVD 复位；**
- **看门狗复位；**
- **降低系统工作速度；**
- **采用外部复位电路（稳压二极管复位电路，电压偏移复位电路，外部 IC 复位电路）。**

✷ **注：“稳压二极管复位电路”、“电压偏移复位电路”和“外部 IC 复位电路”能够完全避免掉电复位出错。**

看门狗复位：

看门狗定时器用于保证系统正常工作。通常，会在主程序中将看门狗定时器清零，但不要在多个分支程序中清看门狗。若程序正常运行，看门狗不会复位。当系统进入死区或程序运行出错的时候，看门狗定时器继续计数直至溢出，系统复位。如果看门狗复位后电源仍处于死区，则系统复位失败，保持复位状态，直到系统工作状态恢复到正常值。

降低系统工作速度：

系统工作速度越快最低工作电压值越高，从而加大工作死区的范围，因此降低系统工作速度不失为降低系统进入死区几率的有效措施。所以，可选择合适的工作速度以避免系统进入死区，这个方法需要调整整个程序使其满足系统要求。

附加外部复位电路：

外部复位也能够完全改善掉电复位性能。有三种外部复位方式可改善掉电复位性能：稳压二极管复位电路，电压偏移复位电路和外部 IC 复位电路。它们都采用外部复位信号控制单片机可靠复位。

3.5 外部复位

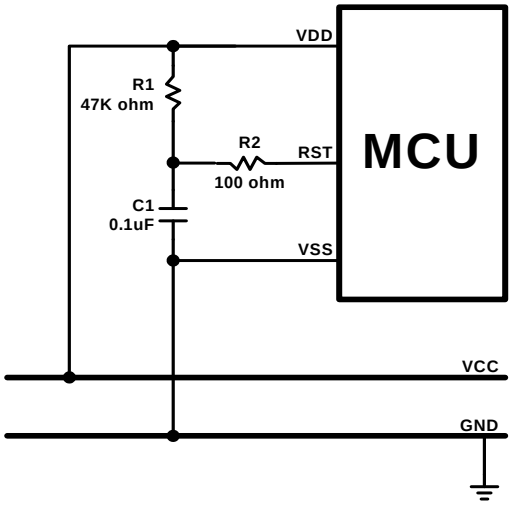
外部复位功能由编译选项“Reset_Pin”控制。将该编译选项置为“Reset”，可使能外部复位功能。外部复位引脚为施密特触发结构，低电平有效。复位引脚处于高电平时，系统正常运行。当复位引脚输入低电平信号时，系统复位。外部复位操作在上电和正常工作模式时有效。需要注意的是，在系统上电完成后，外部复位引脚必须输入高电平，否则系统将一直保持在复位状态。外部复位的时序如下：

- **外部复位（当且仅当外部复位引脚为使能状态）：**系统检测复位引脚的状态，如果复位引脚不为高电平，则系统会一直保持在复位状态，直到外部复位结束；
- **系统初始化：**所有的系统寄存器被置为初始状态；
- **振荡器开始工作：**振荡器开始提供系统时钟；
- **执行程序：**上电结束，程序开始运行。

外部复位可以在上电过程中使系统复位。良好的外部复位电路可以保护系统以免进入未知的工作状态，如 AC 应用中的掉电复位等。

3.6 外部复位电路

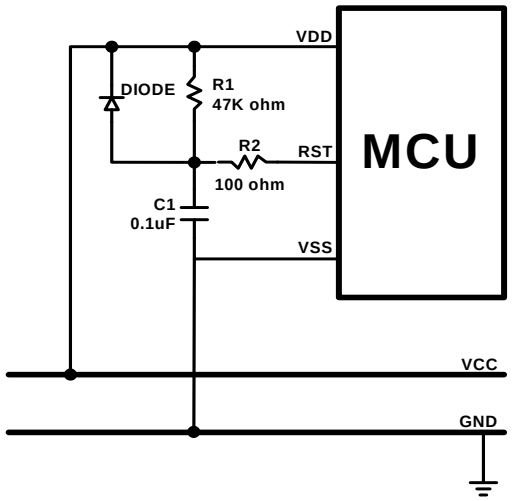
3.6.1 基本RC复位电路



上图为一个由电阻 R1 和电容 C1 组成的基本 RC 复位电路，它在系统上电的过程中能够为复位引脚提供一个缓慢上升的复位信号。这个复位信号的上升速度低于 VDD 的上电速度，为系统提供合理的复位时序，当复位引脚检测到高电平时，系统复位结束，进入正常工作状态。

✷ **注：此 RC 复位电路不能解决非正常上电和掉电复位问题。**

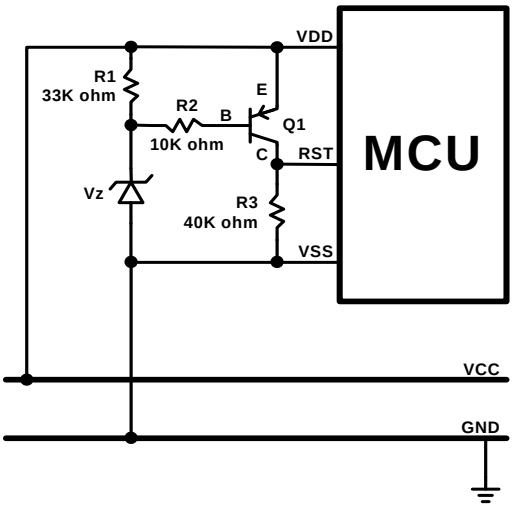
3.6.2 二极管&RC复位电路



上图中，R1 和 C1 同样是为复位引脚提供输入信号。对于电源异常情况，二极管正向导通使 C1 快速放电并与 VDD 保持一致，避免复位引脚持续高电平、系统无法正常复位。

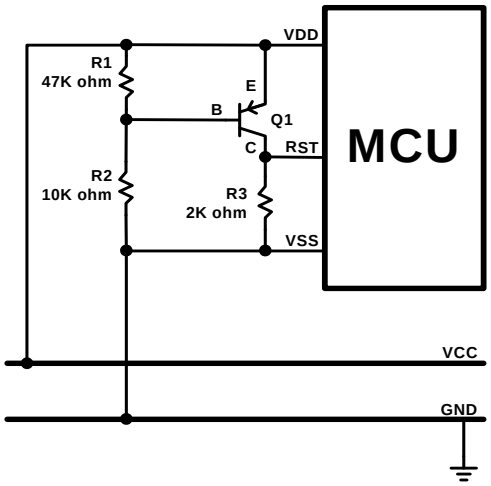
✷ **注：“基本 RC 复位电路”和“二极管及 RC 复位电路”中的电阻 R2 都是必不可少的限流电阻，以避免复位引脚 ESD（Electrostatic Discharge）或 EOS（Electrical Over-stress）击穿 。**

3.6.3 齐纳二极管复位电路



稳压二极管复位电路是一种简单的 LVD 电路，基本上可以完全解决掉电复位问题。如上图电路中，利用稳压管的击穿电压作为电路复位检测值，当 VDD 高于“Vz + 0.7V”时，三极管集电极输出高电平，单片机正常工作；当 VDD 低于“Vz + 0.7V”时，三极管集电极输出低电平，单片机复位。稳压管规格不同则电路复位检测值不同，根据电路的要求选择合适的二极管。

3.6.4 电压偏置复位电路

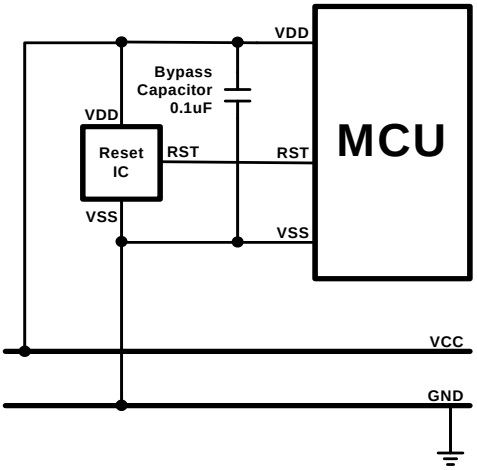


电压偏置复位电路是一种简单的 LVD 电路，基本上可以完全解决掉电复位问题。与稳压二极管复位电路相比，这种复位电路的检测电压值的精确度有所降低。电路中，R1 和 R2 构成分压电路，当 VDD 高于和等于分压值“0.7V x (R1 + R2) / R1”时，三极管集电极 C 输出高电平，单片机正常工作；VDD 低于“0.7V x (R1 + R2) / R1”时，集电极 C 输出低电平，单片机复位。

对于不同应用需求，选择适当的分压电阻。单片机复位引脚上电压的变化与 VDD 电压变化之间的差值为 0.7V。如果 VDD 跌落并低于复位引脚复位检测值，那么系统将被复位。如果希望提升电路复位电平，可将分压电阻设置为 R2>R1，并选择 VDD 与集电极之间的结电压高于 0.7V。分压电阻 R1 和 R2 在电路中要耗电，此处的功耗必须计入整个系统的功耗中。

✷ **注：在电源不稳定或掉电复位的情况下，“稳压二极管复位电路”和“偏压复位电路”能够保护电路在电压跌落时避免系统出错。当电压跌落至低于复位检测值时，系统将被复位。从而保证系统正常工作。**

3.6.5 外部IC复位电路



外部复位也可以选用 IC 进行外部复位，但是这样一来系统成本将会增加。针对不用的应用要求选择适当的复位 IC，如上图所示外部 IC 复位电路，能够有效的降低电源变化对系统的影响。

4 系统时钟

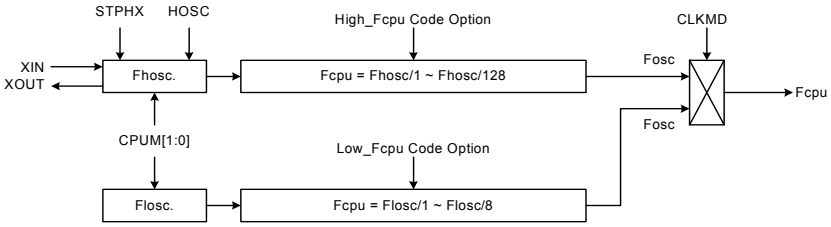
4.1 概述

SN8F27E60 系列单片机内置双时钟系统：高速时钟和低速时钟。高速时钟由内部高速振荡时钟和外部高速振荡时钟提供，外部高速振荡时钟由编译选项“High_CLK”控制。低速时钟由内部低速振荡器提供，由 OSCM 寄存器的 CLKMD 位控制，高、低速时钟都可以作为系统时钟源。

- **高速振荡器**
  内部高速振荡器：RC，高达 16MHz，称为 IHRC 和 IHRC_RTC；
  外部高速振荡器：包括石英/陶瓷振荡器（4MHz，12MHz 和 32KHz）和 RC 振荡器。

- **低速振荡器**
  内部低速振荡器：RC，16KHz，称为 ILRC。

- **系统时钟框图**



- HOSC：High_Clk 编译选项。
- Fhosc：外部高速时钟/内部高速 RC 时钟。
- Flosc：内部低速 RC 时钟频率（包括 16KHz@3V 和@5V）。
- Fosc：系统时钟频率。
- Fcpu：指令周期。

4.2 指令周期（FCPU）

系统时钟速率，即指令周期（Fcpu），从系统时钟源分离出来，决定系统的工作速率。Fcpu 的速率由 High_Fcpu 编译选项决定，正常模式下，Fcpu=Fhosc/1~Fhosc/128。当外部时钟选择 4MHz，且 High_Fcpu 编译选项选择 Fhosc/4 时，则 Fcpu 频率为 4MHz/4=1MHz。低速模式下，Fcpu=Flosc/1~Flosc/8，由 Low_Fcpu 编译选项决定。如果 Low_Fcpu 编译选项选择 Flosc/4，则 Fcpu 的频率为 16KHz/4=4KHz。

4.3 杂讯滤波器（NOISE FILTER）

杂讯滤波器（由编译选项“Noise_Filter”控制）是一个低通滤波器，支持外部振荡器，包括 RC 和晶体模式。杂讯滤波器可以滤除来自外部振荡器的高干扰信号。

在高干扰环境下，强烈建议开启杂讯滤波器以减少干扰的影响。

4.4 系统高速时钟

系统高速时钟包括外部高速时钟和内部高速时钟。外部高速时钟又包括 4MHz、12MHz、32KHz 晶体/陶瓷和 RC 振荡器，高速时钟振荡器由编译选项 High_CLK 选择。内部高速时钟支持实时时钟（RTC）功能，在 IHRC_RTC 模式下，内部高速时钟和外部 32KHz 振荡器有效，内部高速时钟作为系统时钟源，而外部 32KHz 振荡器为 RTC 时钟源，提供一个精确的实时时钟频率。

4.4.1 HIGH_CLK编译选项

对应不同的时钟功能，SONIX 提供多种高速时钟选项，由 High_CLK 选项控制。High_CLK 选项可以选择 IHRC_16M、IHRC_RTC、RC、32K X'tal、12M X'tal 和 4M X'tal，以支持不同带宽的振荡器。

- **IHRC_16M**：系统高速时钟源来自内部高速 16MHz RC 振荡器，XIN/XOUT 作为普通的 I/O 引脚，不连接任何外部振荡设备。
- **IHRC_RTC**：系统高速时钟源来自内部高速 16MHz RC 振荡器，RTC 时钟源为外部低速 32768Hz 振荡器。XIN/XOUT 连接外部 32768Hz 晶振，其 I/O 功能被禁止。
- **RC**：系统高速时钟源来自廉价的 RC 振荡电路，RC 振荡电路只需要和 XIN 引脚连接，XOUT 作为普通的 I/O 引脚。
- **32K X'tal**：系统高速时钟源来自外部低频 32768Hz 振荡器。该选项仅支持 32768Hz 晶体振荡器，RTC 正常工作。
- **12M X'tal**：系统高速时钟源来自外部高频晶体/陶瓷振荡器，其带宽为 10MHz~16MHz。
- **4M X'tal**：系统高速时钟源来自外部高频晶体/陶瓷振荡器，其带宽为 1MHz~10MHz。

关于功耗，选择 IHRC_RTC 选项时，绿色模式下内部高速振荡器和内部低速振荡器都停止工作，仅外部 32768Hz 晶振正常工作，此时，看门狗定时器不能选择 Always_On 选项，否则内部低速振荡器会正常工作。

4.4.2 内部高速RC振荡器（IHRC）

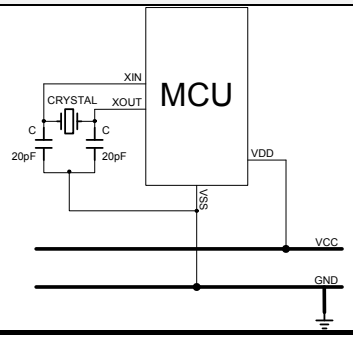
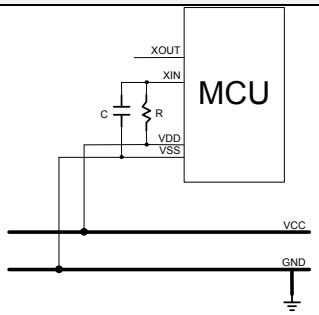
内部高速 16MHz RC 振荡器，普通环境下精确度为±2%，当选择 IHRC_16M 或者 IHRC_RTC 时，使能内部高速振荡器。

- **IHRC_16M**：系统高速时钟为内部 16MHz RC 振荡器，XIN/XOUT 为普通 I/O 引脚。
- **IHRC_RTC**：系统高速时钟为内部 16MHz RC 振荡器，外部 32768Hz 晶振作为实时时钟的时钟源，XIN/XOUT 连接外部 32768Hz 晶振。

4.4.3 外部高速振荡器

外部高速振荡器包括 4MHz，12MHz，32KHz 和 RC。4M、12M 和 32K 可以使用晶体和陶瓷振荡器，XIN/XOUT 和 GND 之间需连接一个 20pF 的电容。廉价的 RC 振荡电路只需要和 XIN 引脚连接，电容的容值不能低于 100pF，电阻的阻值由频率决定。

4.4.4 外部振荡应用电路

晶体/陶瓷	RC
	

✷ **注：晶体/陶瓷和电容 C 要尽可能的靠近单片机的 XIN/XOUT/VSS；电阻 R 和电容 C 要尽可能的靠近单片机的 VDD。**

4.5 系统低速时钟

系统低速时钟源即内置的低速振荡器，采用 RC 振荡电路。低速时钟的输出频率受系统电压和环境温度的影响，通常为输出 16KHZ。

低速时钟可作为看门狗定时器的时钟源。由 CLKMD 控制系统低速工作模式。

- **Flosc = 内部低速 RC 振荡器（16KHz）。**
- **低速模式 Fcpu = Flosc / 1~Flosc/8，由编译选项 Low_Fcpu 控制。**

在睡眠模式下并禁止看门狗可以停掉内部低速 RC。

➢ **例：在睡眠模式下，禁止看门狗定时器，停止内部低速振荡器。**

```
B0BSET      FCPUM0
```

4.6 OSCM寄存器

寄存器 OSCM 控制振荡器的状态和系统的工作模式。

095H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OSCM	0	0	0	CPUM1	CPUM0	CLKMD	STPHX	0
读/写	-	-	-	R/W	R/W	R/W	R/W	-
复位后	-	-	-	0	0	0	0	-

Bit 1 **STPHX**：高速振荡器控制位。
0 = 高速时钟正常运行；
1 = 高速振荡器停止，内部低速 RC 振荡器运行。

Bit 2 **CLKMD**：系统高/低速时钟模式控制位。
0 = 普通模式，系统采用高速时钟；
1 = 低速模式，系统采用内部低速时钟。

Bit[4:3] **CPUM[1:0]**：单片机工作模式控制位。
00 = 普通模式；
01 = 睡眠模式；
10 = 绿色模式；
11 = 系统保留。

STPHX 位为内部高速 RC 振荡器和外部振荡器的模式控制位。当 STPHX=0，外部振荡器和内部高速 RC 振荡器正常运行；当 STPHX=1，外部振荡器或内部高速 RC 振荡器停止运行。不同的高速时钟选项决定不同的 STPHX 功能。

- **IHRC_16M：STPHX=1，禁止内部高速 RC 振荡器；**
- **IHRC_RTC：STPHX=1，禁止内部高速 RC 振荡器，外部 32768Hz 振荡器保持运行；**
- **RC, 4M, 12M, 32K：STPHX=1，禁止外部振荡器。**

4.7 系统时钟测试

在设计过程中，用户可通过软件指令周期对系统时钟速度进行测试。

➢ **例：外部振荡器的 Fcpu 指令周期测试。**

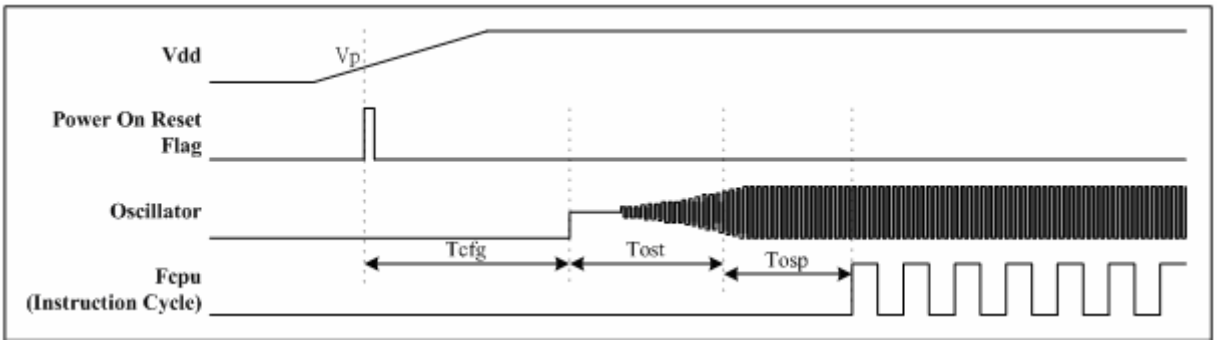
```
        B0BSET      P0M.0           ; P0.0 置为输出模式以输出 Fcpu 的触发信号。

@@:
        B0BSET      P0.0
        B0BCLR      P0.0
        JMP         @B
```

✷ **注：不能直接从 XIN 引脚测试 RC 振荡频率，因为探针的连接会影响测试的准确性。**

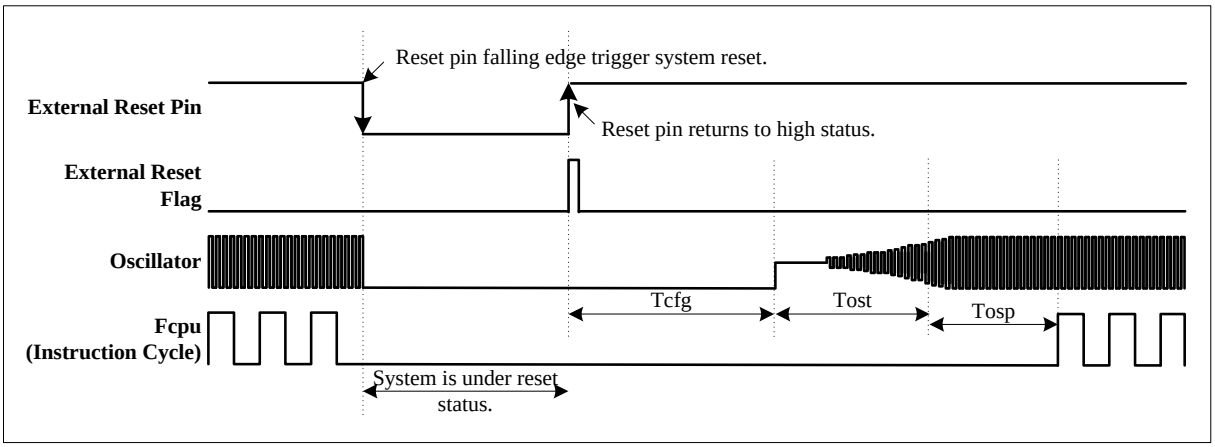
4.8 系统时钟时序

参数	符号	说明	典型值
硬件配置时间	Tcfg	2048*FILRC	64ms @ FILRC = 32KHz 128ms @ FILRC = 16KHz
振荡器启动时间	Tost	启动时间取决于振荡器的材料、工艺等。通常情况下，低速振荡器的启动时间要比高速振荡器的启动时间慢，RC 振荡器的启动时间要比晶体/陶瓷振荡器的启动时间快。	-
振荡器起振时间	Tosp	复位情况下的振荡器起振时间为 2048*Fhosc （使能上电复位，LVD 复位，看门狗复位，外部复位引脚）	64ms @ Fhosc = 32KHz 512us @ Fhosc = 4MHz 128us @ Fhosc = 16MHz
		睡眠模式唤醒情况的振荡器起振时间为： 2048*Fhosc ……晶体/陶瓷振荡器，如 32768Hz 晶振，4MHz 晶振，16MHz 晶振等； 32*Fhosc……RC 振荡器，如外部 RC 振荡电路，内部高速 RC 振荡器。	X'tal: 64ms @ Fhosc = 32KHz 512us @ Fhosc = 4MHz 128us @ Fhosc = 16MHz RC: 8us @ Fhosc = 4MHz 2us @ Fhosc = 16MHz

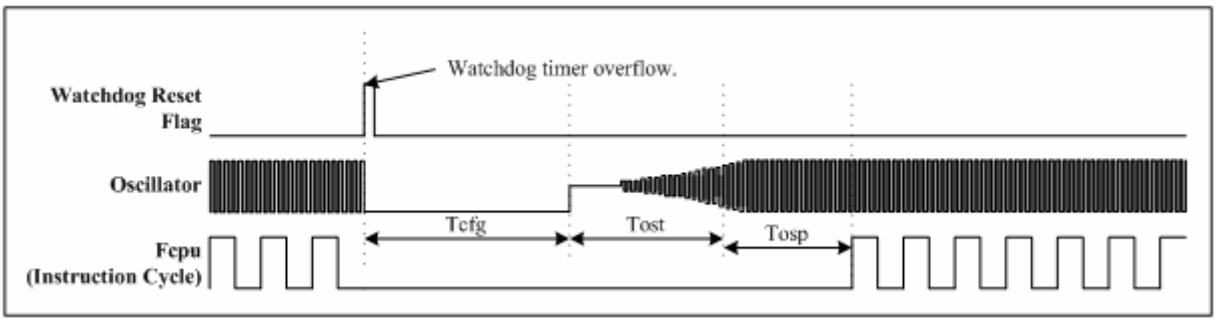
- 上电复位时序



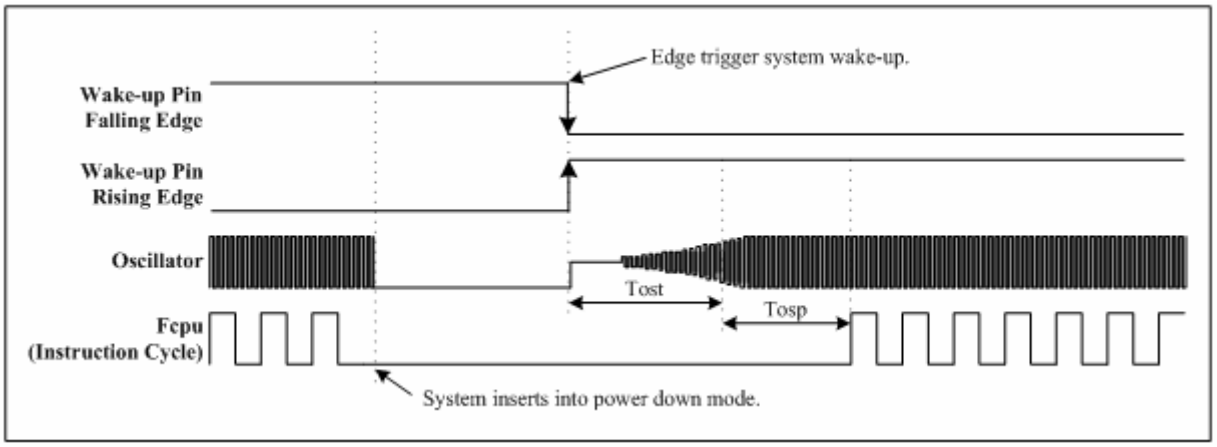
- 外部复位引脚复位时序



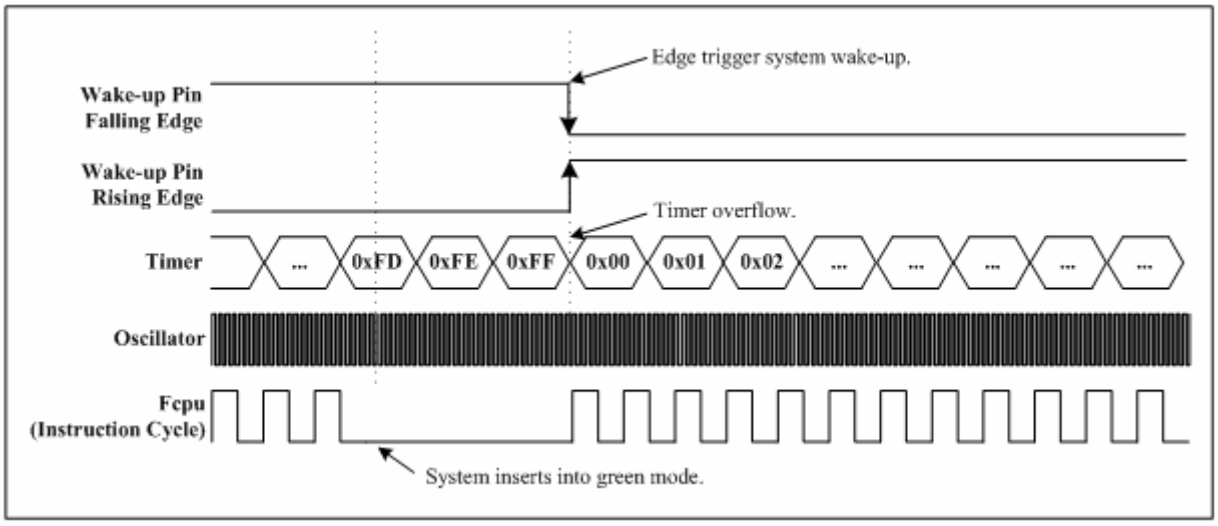
- 看门狗复位时序



- **睡眠模式唤醒时序**

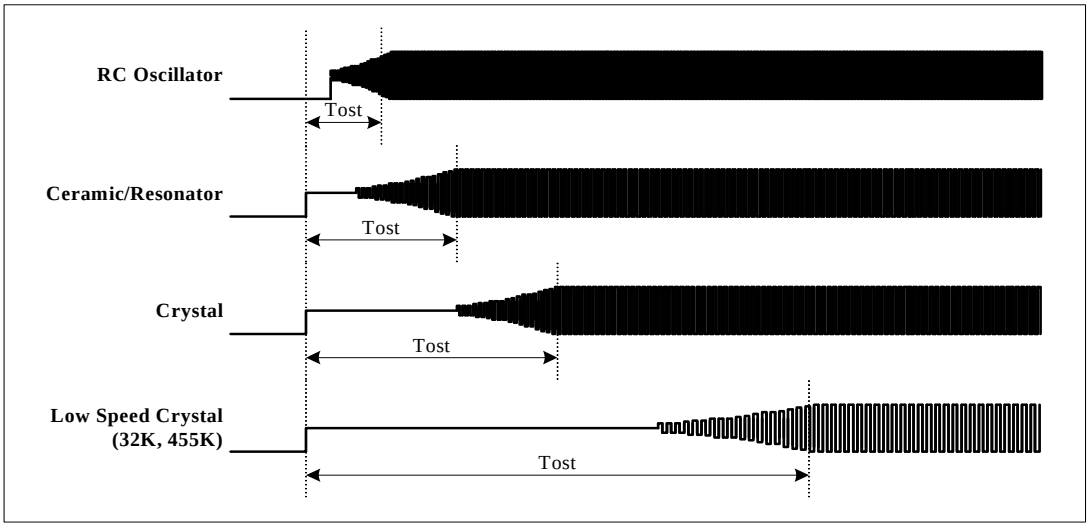


- **绿色模式唤醒时序**



- **振荡器启动时间**

启动时间取决于振荡器的材料、工艺等。通常情况下，低速振荡器的启动时间要比高速振荡器的启动时间慢，RC 振荡器的启动时间要比晶体/陶瓷振荡器的启动时间快。



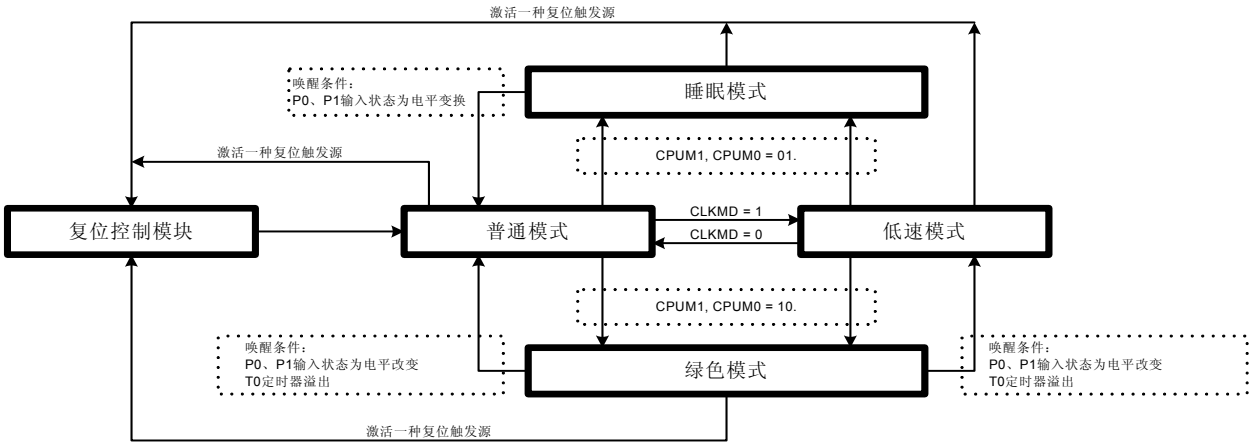
5 系统操作模式

5.1 概述

SN8F27E60 系列单片机可以在 4 种工作模式下以不同的时钟频率工作，这些模式可以控制振荡器的工作、程序的执行已经模拟电路的功能损耗。

- **普通模式：**系统高速工作模式；
- **低速模式：**系统低速工作模式；
- **省电模式：**系统省电模式（睡眠模式）；
- **绿色模式：**系统理想模式。

工作模式控制框图



工作模式时钟控制表

工作模式	普通模式	低速模式	绿色模式	睡眠模式
IHRC	IHRC/IHRC_RTC：运行 Ext.OSC：禁止	IHRC/IHRC_RTC：STPHX Ext.OSC：禁止	IHRC/IHRC_RTC： STPHX Ext.OSC：禁止	停止
ILRC	运行	运行	运行	停止
Ext.OSC	IHRC：禁止 IHRC_RTC/Ext.OSC：运行	IHRC：禁止 IHRC_RTC：运行 Ext.OSC：STPHX	IHRC：STPHX IHRC_RTC：运行 Ext.OSC：STPHX	停止
CPU 指令	执行	执行	停止	停止
T0 定时器	T0ENB	T0ENB	T0ENB	无效
TC0 定时器（定时器、事件计数器和 PWM）	TC0ENB	TC0ENB	TC0ENB	无效
TC1 定时器（定时器、事件计数器和 PWM）	TC1ENB	TC1ENB	TC1ENB	无效
TC2 定时器（定时器、事件计数器和 PWM）	TC2ENB	TC2ENB	TC2ENB	无效
T1 定时器（定时器、事件计数器）	T1ENB	T1ENB	T1ENB	无效
SIO	使能时有效	无效	无效	无效
MSP	使能时有效	无效	无效	无效
UART	使能时有效	无效	无效	无效
ADC	使能时有效	无效	无效	无效
看门狗定时器	Watch_Dog 编译选项	Watch_Dog 编译选项	Watch_Dog 编译选项	Watch_Dog 编译选项
内部中断	全部有效	全部有效	全部有效	全部无效
外部中断	全部有效	全部有效	全部有效	全部无效
唤醒功能	-	-	P0, P1, T0，复位	P0, P1，复位

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

✷ 注：
1、低速模式和绿色模式下，SIO、MSP 和 UART 功能无效，是因为此时时钟源不存在。在系统进入低速模式和绿色模式之前，利用韧件禁止 SIO、MSP 和 UART 功能。
2、IHRC_RTC 模式下，STPHX 只控制 IHRC，而不控制 Ext.32K。STPHX=0，IHRC 有效；STPHX=1，IHRC 停止工作。

5.2 普通模式

普通模式是系统高速时钟正常工作模式，系统时钟源由高速振荡器提供。程序被执行。上电复位或任意一种复位触发后，系统进入普通模式执行程序。当系统从睡眠模式被唤醒后进入普通模式。普通模式下，高速振荡器正常工作，功耗最大。

- 程序被执行，所有的功能都可控制。
- 系统速率为高速。
- 高速振荡器和内部低速 RC 振荡器都正常工作。
- 通过 OSCM 寄存器，系统可以从普通模式切换进入其他工作模式。
- 系统从睡眠模式唤醒后进入普通模式。
- 低速模式可以切换到普通模式。
- 从普通模式切换到绿色模式，唤醒后返回到普通模式。

5.3 低速模式

低速模式为系统低速时钟正常工作模式。系统时钟源由内部低速 RC 振荡器提供。低速模式由 OSCM 寄存器的 CLKMD 位控制。当 CLKMD=0 时，系统为普通模式；当 CLKMD=1 时，系统进入低速模式。切换进入低速模式后，不能自动禁止高速振荡器，必须通过 SPTHX 位来禁止以减少功耗。低速模式下，系统速率可以通过编译选项从 Flosc/1，Flosc/2，Flosc/4，Flosc/8（Flosc 为内部低速 RC 振荡器频率）进行选择。

- 程序被执行，所有的功能都可控制。
- 系统速率位低速（Flosc/1，Flosc/2，Flosc/4，Flosc/8，由编译选项控制）。
- 内部低速 RC 振荡器正常工作，高速振荡器由 STPHX=1 控制。低速模式下，强烈建议停止高速振荡器。
- 通过 OSCM 寄存器，低速模式可以切换进入其它的工作模式。
- 从低速模式切换到睡眠模式，唤醒后返回到普通模式。
- 普通模式可以切换进入低速模式。
- 从低速模式切换到绿色模式，唤醒后返回到低速模式。

5.4 睡眠模式

睡眠模式是系统的理想状态，不执行程序，振荡器也停止工作。只有内部基准工作以保持门状态、寄存器状态和 SRAM 内容。睡眠模式可以由 P0、P1 的电平变换触发唤醒。P0 的唤醒功能一直有效，P1 的唤醒功能由 P1W 寄存器控制。从任何工作模式进入睡眠模式，被唤醒后都返回到普通模式。由 OSCM 寄存器的 CPUM0 位控制是否进入睡眠模式，当 CPUM0=1，系统进入睡眠模式。当系统从睡眠模式被唤醒后，CPUM0 被自动禁止（0 状态），WAKE 位为 1。

- 程序停止执行，所有的功能被禁止。
- 所有的振荡器，包括外部高速振荡器、内部高速振荡器和内部低速振荡器都停止工作。
- 系统从睡眠模式被唤醒后进入普通模式。
- 睡眠模式的唤醒源为 P0 和 P1 电平变换触发。
- 系统从睡眠模式被唤醒后，WAKE 位置 1，由程序清除。
- 如果唤醒源为外部中断，WAKE 位不能置 1，外部中断请求标志位被置 1。系统给出外部中断请求并执行中断服务程序。

✷ **注：普通模式下，设置 STPHX=1 禁止高速时钟振荡器，这样，无系统时钟在执行，可以确保系统进入睡眠模式，可以由 P0、P1 电平变换触发唤醒。**

5.5 绿色模式

绿色模式是另外的一种理想状态。在睡眠模式下，所有的功能和硬件设备都被禁止，但在绿色模式下，系统时钟保持工作，绿色模式下的功耗大于睡眠模式下的功耗。绿色模式下，不执行程序，但具有唤醒功能的定时器仍正常工作，定时器的时钟源为仍在工作的系统时钟。绿色模式下，有 2 种方式可以将系统唤醒：1、P0 和 P1 电平变换触发；2、具有唤醒功能的定时器溢出，这样，用户可以给定时器设定固定的周期，系统就在溢出时被唤醒。由 OSCM 寄存器 CPUM1 为决定是否进入绿色模式，当 CPUM1=1，系统进入绿色模式。当系统从绿色模式下被唤醒后，自动禁止 CPUM1（0 状态），WAKE 位被置 1。

- 程序停止执行，所有的功能被禁止。
- 具有唤醒功能的定时器正常工作。
- 作为系统时钟源的振荡器正常工作，其它的振荡器工作状态取决于系统工作模式的配置。
- 由普通模式切换到绿色模式，被唤醒后返回到普通模式。
- 由低速模式切换到绿色模式，被唤醒后返回到低速模式。
- 绿色模式下的唤醒方式为 P0、P1 电平变换触发唤醒或者指定的定时器溢出。
- 如果唤醒源为外部中断，WAKE 位不能置 1，外部中断请求标志位被置 1。系统给出外部中断请求并执行中断服务程序。
- 绿色模式下 PWM 和 Buzzer 功能仍然有效，但是定时器溢出时不能唤醒系统。

✷ **注：sonix 提供宏“GreenMode”来控制绿色模式的工作状态，必要时使用宏“GreeMode”进绿色模式。该宏共有 3 条指令。但在使用 BRANCH 指令（如 BTS0、BTS1、B0BTS0、B0BTS1、INCS、INCMS、DECS、DECMS、CMPRS、JMP）时必须注意宏的长度，否则程序会出错。**

5.6 工作模式控制宏

Sonix 提供工作模式控制宏以方便系统工作模式的切换。

宏名称	长度	说明
SleepMode	1-word	系统进入睡眠模式。
GreenMode	3-word	系统进入绿色模式。
SlowMode	2-word	系统进入低速模式并停止高速振荡器。
Slow2Normal	5-word	系统从低速模式返回到普通模式。该宏包括工作模式的切换，使能高速振荡器，高速振荡器唤醒延迟时间。

➢ **例：从普通模式/低速模式切换进入睡眠模式。**

```
SleepMode                       ; 直接宣告“SleepMode”宏。
```

➢ **例：从普通模式切换进入低速模式。**

```
SlowMode                        ; 直接宣告“SlowMode”宏。
```

➢ **例：从低速模式切换进入普通模式（外部高速振荡器停止工作）。**

```
Slow2Normal                     ; 直接宣告“Slow2Normal”宏。
```

➢ **例：从普通/低速模式切换进入绿色模式。**

```
GreenMode                       ; 直接宣告“GreenMode”宏。
```

➢ **例：从普通/低速模式切换进入绿色模式，并使能 T0 唤醒功能。**

```
; 设置定时器 T0 的唤醒功能。
        B0BCLR      FT0IEN      ; 禁止 T0 中断。
        B0BCLR      FT0ENB      ; 禁止 T0 定时器。
        MOV         A,#20H      ;
        B0MOV       T0M,A       ; 设置 T0 时钟= Fcpu / 64。
        MOV         A,#74H
        B0MOV       T0C,A       ; 设置 T0C 的初始值= 74H（设置 T0 间隔值 = 10 ms）。
        B0BCLR      FT0IEN      ; 禁止 T0 中断。
        B0BCLR      FT0IRQ      ; 清 T0 中断请求。
        B0BSET      FT0ENB      ; 使能 T0 定时器。

; 进入绿色模式。
        GreenMode               ; 直接宣告“GreenMode”宏。
```

➢ **例：从普通/低速模式切换进入绿色模式，并使能 T0 的唤醒功能，带有 RTC 功能。**

```
        CLR         T0C         ; 清 T0 计数器。
        B0BSET      FT0TB       ; 使能 T0 RTC 功能。
        B0BSET      FT0ENB      ; 使能 T0 定时器。

; 进入绿色模式。
        GreenMode               ; 直接宣告“GreenMode”宏。
```

5.7 系统唤醒

5.7.1 概述

睡眠模式和绿色模式下，系统并不执行程序。唤醒触发信号可以将系统唤醒进入普通模式或低速模式。唤醒触发信号包括：外部触发信号（P0、P1 的电平变换）和内部触发（T0 定时器溢出）。唤醒功能内置中断请求标志位，触发后执行中断服务程序。

- 从睡眠模式唤醒后只能进入普通模式，且将其唤醒的触发只能是外部触发信号（P0、P1 电平变化）；
- 如果是将系统由绿色模式唤醒返回到上一个工作模式（普通模式或低速模式），则可以是外部触发信号（P0、P1 电平变换）和内部触发信号（T0 溢出）。
- 系统从睡眠模式或绿色模式下被唤醒时，Wakeup 中断功能给出 WAKEIRQ 中断请求。若 WAKEIEN 为 1，触发中断，执行中断服务程序。

✷ **注：如果唤醒源为外部中断，WAKE 位不能置 1，外部中断请求标志位被置 1。系统给出外部中断请求并执行中断服务程序。**

5.7.2 唤醒时间

系统进入睡眠模式后，高速时钟振荡器停止运行。把系统从睡眠模式唤醒时，单片机需要等待 2048 个外部高速时钟周期和 32 个内部高速时钟周期的时间，以等待振荡电路稳定工作。唤醒时间结束后，系统进入普通模式。

✷ **注：从绿色模式下唤醒系统不需要唤醒时间，因为系统时钟在绿色模式下仍然正常工作。**

唤醒时间的计算如下：

唤醒时间 = 1/Fosc * 2048（sec）+ 高速时钟启动时间

➢ **例：睡眠模式下，系统被唤醒进入普通模式。唤醒时间的计算如下：**

唤醒时间 = 1/Fosc * 2048 = 0.512 ms（Fosc = 4MHz）
总的唤醒时间 = 0.512ms + 振荡器启动时间

内部高速 RC 时钟的唤醒时间的计算如下：

唤醒时间 = 1/Fosc * 32 (sec) + 高速时钟启动时间

➢ **例：睡眠模式下，系统被唤醒进入普通模式。唤醒时间的计算如下：**

唤醒时间 = 1/Fosc * 32 = 2 us（Fosc = 16MHz）

✷ **注：高速时钟的启动时间决定于 VDD 和振荡器的类型。**

5.7.3 P1W唤醒控制寄存器

在绿色模式和睡眠模式下，有唤醒功能的 I/O 口能够将系统唤醒到普通模式。P0 和 P1 都具有唤醒功能，二者区别在于：P0 的唤醒功能始终有效，而 P1 的唤醒功能则由寄存器 P1W 控制。

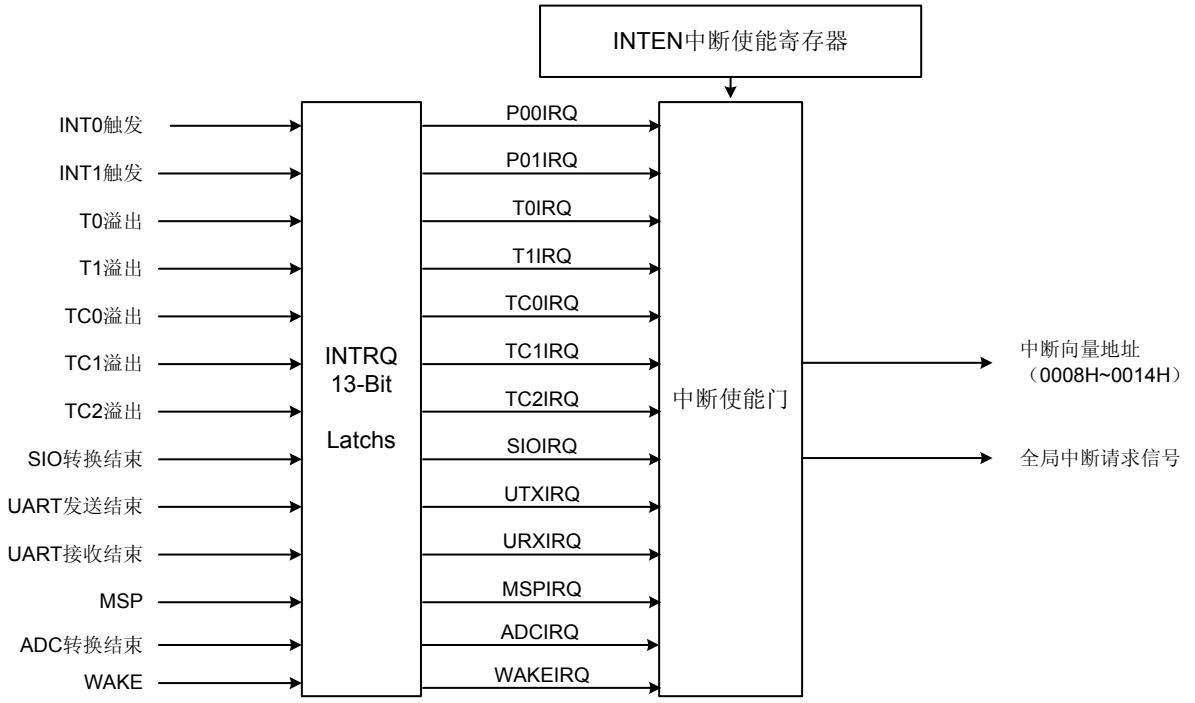
09EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **P10W~P17W**：P1 唤醒功能控制位。
0 = 禁止；
1 = 使能。

6 中断

6.1 概述

SN8F27E60 系列单片机共提供 13 种中断源：11 个内部中断（T0/T1/TC0/TC1/TC2/SIO/MSP/UTX/URX/WAKE/ADC）和 2 个外部中断（INT0/INT1）。外部中断可以将系统从睡眠模式唤醒进入高速模式，在返回高速模式前，外部中断请求被锁定。一旦程序进入中断，寄存器 STKP 的位 GIE 将被硬件自动清零以避免再次响应其它中断。系统退出中断，即执行完 RETI 指令后，硬件自动将 GIE 置“1”，以响应下一个中断。中断请求存放在寄存器 INTRQ 中。



✷ **注：程序响应中断时，位 GIE 必须处于有效状态。**

6.2 中断操作

中断由 IRQ 和 IEN 位控制，IRQ 为中断源的事件标志位，与是否使能中断无关；IEN 控制系统中断的执行。当 IEN=0，系统不会跳至中断向量处执行中断；当 IEN=1，且其中一个 IRQ 标志位有效时，系统才会执行中断。

- **IEN = 1 且 IRQ = 1，系统跳至中断向量处开始执行中断。**

当任意一个中断发生时，系统会跳至对应的中断向量处开始执行该中断。中断开始的第一个操作为 PUSH，中断结束时的最后一个操作为 POP，PUSH 和 POP 操作由硬件自动执行，而无需通过 PUSH、POP 指令。

- **PUSH 操作：保存 ACC 的内容和工作寄存器（80H~8FH）到硬件缓存器中，在系统跳至中断向量之前执行 PUSH 操作。RAM bank 保持为主程序状态，并不会自动切换到 bank 0。由程序选择 RAM bank。**
- **POP 操作：将 ACC 的内容和工作寄存器（80H~8FH）从硬件缓存器中恢复出来，在执行 RETI 指令后执行 POP 操作。恢复 RBANK 的内容后，RAM bank 切换到主程序的上一个状态下。**
- **80H~8FH 工作寄存器包括 L、H、R、Z、Y、X、PFLAG、RBANK、W0~W7。**

6.3 中断使能寄存器INTEN

中断请求控制寄存器 INTEN 包括所有中断的使能控制位。INTEN 的有效位被置为“1”，则系统进入该中断服务程序，程序计数器入栈，程序转至 0008H~0014H 即中断程序。程序运行到指令 RETI 时，中断结束，系统退出中断服务。

09AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN0	ADCIEN	T1IEN	TC2IEN	TC1IEN	TC0IEN	T0IEN	P01IEN	P00IEN
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 0 **P00IEN**：P0.0 外部中断（INT0）控制位。
0 = 禁止；
1 = 使能。

Bit 1 **P01IEN**：P0.1 外部中断（INT1）控制位。
0 = 禁止；
1 = 使能。

Bit 2 **T0IEN**：T0 中断控制位。
0 = 禁止；
1 = 使能。

Bit 3 **TC0IEN**：TC0 中断控制位。
0 = 禁止；
1 = 使能。

Bit 4 **TC1IEN**：TC1 中断控制位。
0 = 禁止；
1 = 使能。

Bit 5 **TC2IEN**：TC2 中断控制位。
0 = 禁止；
1 = 使能。

Bit 6 **T1IEN**：T1 中断控制位。
0 = 禁止；
1 = 使能。

Bit 7 **ADCIEN**：ADC 中断控制位。
0 = 禁止；
1 = 使能。

09BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN1	-	-	-	MSPIEN	UTXIEN	URXIEN	SIOIEN	WAKEIEN
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	0	0	0

Bit 0 **WAKEIEN**：Wakeup 中断控制位。
0 = 禁止；
1 = 使能。

Bit 1 **SIOIEN**：SIO 中断控制位。
0 = 禁止；
1 = 使能。

Bit 2 **URXIEN**：UART 接收中断控制位。
0 = 禁止；
1 = 使能。

Bit 3 **UTXIEN**：UART 发送中断控制位。
0 = 禁止；
1 = 使能。

Bit 4 **MSPIEN**：MSP 中断控制位。
0 = 禁止；
1 = 使能。

6.4 中断请求寄存器INTRQ

中断请求寄存器 INTRQ 中存放各中断请求标志。一旦有中断请求发生，INTRQ 中的相应位将被置“1”，该请求被响应后，程序应将该标志位清零。根据 INTRQ 的状态，程序判断是否有中断发生，并执行相应的中断服务。

097H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ0	ADCIRQ	T1IRQ	TC2IRQ	TC1IRQ	TC0IRQ	T0IRQ	P01IRQ	P00IRQ
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 0 **P00IRQ**：外部 P0.0 中断（INT0）请求标志位。
0 = 无中断请求；
1 = 请求中断。

Bit 1 **P01IRQ**：外部 P0.1 中断（INT1）请求标志位。
0 = 无中断请求；
1 = 请求中断。

Bit 2 **TC0IRQ**：T0 中断请求标志位
0 = 无中断请求；
1 = 请求中断。

Bit 3 **TC0IRQ**：TC0 中断请求标志位。
0 = 无中断请求；
1 = 请求中断。

Bit 4 **TC1IRQ**：TC1 中断请求标志位。
0 = 无中断请求；
1 = 请求中断。

Bit 5 **TC2IRQ**：TC2 中断请求标志位。
0 = 无中断请求；
1 = 请求中断。

Bit 6 **T1IRQ**：T1 中断请求标志位。
0 = 无中断请求；
1 = 请求中断。

Bit 7 **ADCIRQ**：ADC 中断请求标志位。
0 = 无中断请求；
1 = 请求中断。

098H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ1				MSPIRQ	UTXIRQ	URXIRQ	SIOIRQ	WAKEIRQ
读/写				R/W	R/W	R/W	R/W	R/W
复位后				0	0	0	0	0

Bit 0 **WAKEIRQ**：Wakeup 中断请求标志位。
0 = 无中断请求；
1 = 请求中断。

Bit 1 **SIOIRQ**：SIO 中断请求标志位。
0 = 无中断请求；
1 = 请求中断。

Bit 2 **URXIRQ**：UART 接收中断请求标志位。
0 = 无中断请求；
1 = 请求中断。

Bit 3 **UTXIRQ**：UART 发送中断请求标志位。
0 = 无中断请求；
1 = 请求中断。

Bit 4 **MSPIRQ**：MSP 中断请求标志位。
0 = 无中断请求；
1 = 请求中断。

6.5 全局中断操作控制位GIE

只有当全局中断控制位 GIE 置“1”的时候程序才能响应中断请求。 一旦有中断发生，程序计数器（PC）指向中断向量地址（ORG8~14），堆栈层数加 1。

0EFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	LVD24	LVD33	-	-	STKPB2	STKPB1	STKPB0
读/写	R/W	R	R	-	-	R/W	R/W	R/W
复位后	0			-	-	1	1	1

Bit 7 **GIE**：全局中断控制位。
0 = 禁止全局中断；
1 = 使能全局中断。

➢ **例：设置全局中断控制位（GIE）。**

```
B0BSET      FGIE        ; 使能 GIE。
```

✷ **注：在所有中断中，GIE 都必须处于使能状态。**

6.6 外部中断（INT0~INT1）

SN8F27E60 系列单片机提供 2 个外部中断 INT0/INT1。当发生外部触发，且使能外部中断控制位时，外部中断请求位置 1。如果禁止外部中断控制位，即使发生外部触发，外部中断请求位也不会请求中断。当使能外部中断控制位，且触发外部中断时，程序计数器会跳至中断向量 ORG 0009H/ORG 000AH 并执行外部中断。

外部中断内置唤醒锁存功能，即系统从睡眠模式触发唤醒时，唤醒源为外部中断源（P0.0 或者 P0.1），其唤醒触发方向与中断触发方向统一时，唤醒触发方向被锁定，即系统被唤醒后先执行外部中断。

09FH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEDGE	-	-	-	-	P01G1	P01G0	P00G1	P00G0
读/写	-	-	-	-	R/W	R/W	R/W	R/W
复位后	-	-	-	-	0	0	0	0

Bit[1:0] **P00G[1:0]**：P0.0 中断触发控制位。
00 = 保留；
01 = 上升沿触发；
10 = 下降沿触发；
11 = 上升/下降沿触发（电平变换触发）。

Bit[3:2] **P01G[1:0]**：P0.1 中断触发控制位。
00 = 保留；
01 = 上升沿触发；
10 = 下降沿触发；
11 = 上升/下降沿触发（电平变换触发）。

➢ **例：INT0 中断请求设置，电平触发。**

```
        MOV     A, #03H
        B0MOV   PEDGE, A        ; 设置 INT0 为电平触发。

        B0BCLR  FP00IRQ         ; 清 INT0 中断请求标志。
        B0BSET  FP00IEN         ; 允许 INT0 中断。
        B0BSET  FGIE            ; 使能 GIE。
```

➢ **例：INT0 中断。**

```
        ORG     9H              ;
        JMP     INT_SERVICE
INT_SERVICE:

        …                       ; 保存 ACC 和 PFLAG。

        B0BTS1  FP00IRQ         ; 检查是否有 P00 中断请求标志。
        JMP     EXIT_INT        ; P00IRQ = 0，退出中断。

        B0BCLR  FP00IRQ         ; 清 P00IRQ。
        …                       ; INT0 中断服务程序。
        …
EXIT_INT:
        …                       ; 恢复 ACC 和 PFLAG。

        RETI                    ; 退出中断。
```

6.7 T0 中断

T0C 溢出时，无论 T0IEN 处于何种状态，T0IRQ 都会置“1”。若 T0IEN 和 T0IRQ 都置“1”，系统就会响应 T0 中断；若 T0IEN = 0，则无论 T0IRQ 是否置“1”，系统都不会响应 T0 中断。尤其需要注意多种中断下的情形。

➢ **例：设置 T0 中断。**

```
        B0BCLR      FT0IEN          ; 禁止 T0 中断。
        B0BCLR      FT0ENB          ; 关闭 T0 定时器。
        MOV         A, #20H         ;
        B0MOV       T0M, A          ; 设置 T0 时钟= Fcpu / 64。
        MOV         A, #74H         ; 初始化 T0C = 74H。
        B0MOV       T0C, A          ; 设置 T0 间隔时间= 10 ms。

        B0BCLR      FT0IRQ          ; T0IRQ 清零。
        B0BSET      FT0IEN          ; 使能 T0 中断。
        B0BSET      FT0ENB          ; 开启定时器 T0。

        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：T0 中断服务程序。**

```
        ORG         0BH
        JMP         INT_SERVICE
INT_SERVICE:

        …                           ; 保存 ACC 和 PFLAG。

        B0BTS1      FT0IRQ          ; 检查是否有 T0 中断请求标志。
        JMP         EXIT_INT        ; T0IRQ = 0，退出中断。

        B0BCLR      FT0IRQ          ; 清 T0IRQ。
        MOV         A, #74H
        B0MOV       T0C, A          ;
        …
        …
EXIT_INT:
        …                           ; 恢复 ACC 和 PFLAG。

        RETI                        ; 退出中断。
```

✷ **注：RTC 模式下，不能在中断时复位 T0C。**

6.8 TC0 中断

TC0C 溢出时，无论 TC0IEN 处于何种状态，TC0IRQ 都会置“1”。若 TC0IEN 和 TC0IRQ 都置“1”，系统就会响应 TC0 中断；若 TC0IEN = 0，则无论 TC0IRQ 是否置“1”，系统都不会响应 TC0 中断。尤其需要注意多种中断下的情形。

➢ **例：设置 TC0 中断。**

```
        B0BCLR      FTC0IEN     ; 禁止 TC0 中断。
        B0BCLR      FTC0ENB     ; 关闭 TC0 定时器。
        MOV         A, #10H     ;
        B0MOV       TC0M, A     ; 设置 TC0 时钟= Fcpu / 64。
        MOV         A, #74H     ; 设置 TC0C 的初始值为 74H。
        B0MOV       TC0C, A     ; 设置 TC0 的间隔时间为 10ms。

        B0BCLR      FTC0IRQ     ; 清 TC0IRQ。
        B0BSET      FTC0IEN     ; 使能 TC0 中断。
        B0BSET      FTC0ENB     ; 开启 TC0 定时器。

        B0BSET      FGIE        ; 使能 GIE。
```

➢ **例：TC0 中断。**

```
        ORG         0CH         ; 中断向量。
        JMP         INT_SERVICE
INT_SERVICE:

        …                       ; 保存 ACC 和 PFLAG。

        B0BTS1      FTC0IRQ     ; 检查是否有 TC0 中断请求。
        JMP         EXIT_INT    ; TC0IRQ = 0，退出中断。

        B0BCLR      FTC0IRQ     ; 清 TC0IRQ。
        MOV         A, #74H
        B0MOV       TC0C, A     ; 初始化 TC0C。
        …                       ; TC0 中断程序。
        …
EXIT_INT:
        …                       ; 恢复 ACC 和 PFLAG。

        RETI                    ; 退出中断。
```

6.9 TC1 中断

TC1C 溢出时，无论 TC1IEN 处于何种状态，TC1IRQ 都会置“1”。若 TC1IEN 和 TC1IRQ 都置“1”，系统就会响应 TC1 中断；若 TC1IEN = 0，则无论 TC1IRQ 是否置“1”，系统都不会响应 TC1 中断。尤其需要注意多种中断下的情形。

➢ **例：设置 TC1 中断。**

```
        B0BCLR      FTC1IEN         ; 禁止 TC1 中断。
        B0BCLR      FTC1ENB         ; 关闭 TC1 定时器。
        MOV         A, #10H         ;
        B0MOV       TC1M, A         ; 设置 TC1 时钟= Fcpu / 64
        MOV         A, #74H         ; 设置 TC1C 初始值为 74H。
        B0MOV       TC1C, A         ; 设置 TC1 间隔时间为 10ms。

        B0BCLR      FTC1IRQ         ; 清 TC1IRQ。
        B0BSET      FTC1IEN         ; 使能 TC1 中断。
        B0BSET      FTC1ENB         ; 开启 TC1 定时器。

        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：TC1 中断。**

```
        ORG         0DH             ; 中断向量。
        JMP         INT_SERVICE
INT_SERVICE:

        …                           ; 保存 ACC 和 PFLAG。

        B0BTS1      FTC1IRQ         ; 检查是否有 TC1 中断请求。
        JMP         EXIT_INT        ; TC1IRQ = 0，退出中断。

        B0BCLR      FTC1IRQ         ; 清 TC1IRQ。
        MOV         A, #74H
        B0MOV       TC1C, A         ; 初始化 TC1C。
        …                           ; TC1 中断程序。
        …
EXIT_INT:
        …                           ; 恢复 ACC 和 PFLAG。

        RETI                        ; 退出中断。
```

6.10 TC2 中断

TC2C 溢出时，无论 TC2IEN 处于何种状态，TC2IRQ 都会置“1”。若 TC2IEN 和 TC2IRQ 都置“1”，系统就会响应 TC2 中断；若 TC2IEN = 0，则无论 TC2IRQ 是否置“1”，系统都不会响应 TC2 中断。尤其需要注意多种中断下的情形。

➢ **例：设置 TC2 中断。**

```
        B0BCLR      FTC2IEN         ; 禁止 TC2 中断。
        B0BCLR      FTC2ENB         ; 关闭 TC2 定时器。
        MOV         A, #10H         ;
        B0MOV       TC2M, A         ; 设置 TC2 时钟= Fcpu / 64
        MOV         A, #74H         ; 设置 TC2C 初始值为 74H。
        B0MOV       TC2C, A         ; 设置 TC2 间隔时间为 10ms。

        B0BCLR      FTC2IRQ         ; 清 TC2IRQ。
        B0BSET      FTC2IEN         ; 使能 TC2 中断。
        B0BSET      FTC2ENB         ; 开启 TC2 定时器。

        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：TC2 中断。**

```
        ORG         0EH             ; 中断向量。
        JMP         INT_SERVICE
INT_SERVICE:

        …                           ; 保存 ACC 和 PFLAG。

        B0BTS1      FTC2IRQ         ; 检查是否有 TC2 中断请求。
        JMP         EXIT_INT        ; TC2IRQ = 0，退出中断。

        B0BCLR      FTC2IRQ         ; 清 TC2IRQ。
        MOV         A, #74H
        B0MOV       TC2C, A         ; 初始化 TC2C。
        …                           ; TC2 中断程序。
        …
EXIT_INT:
        …                           ; 恢复 ACC 和 PFLAG。

        RETI                        ; 退出中断。
```

6.11 T1 中断

T1C（T1CH、T1CL）溢出时，无论 T1IEN 处于何种状态，T1IRQ 都会置“1”。若 T1IEN 和 T1IRQ 都置“1”，系统就会响应 T1 中断；若 T1IEN = 0，则无论 T1IRQ 是否置“1”，系统都不会响应 T1 中断。尤其需要注意多种中断下的情形。

➢ **例：设置 T1 中断。**

```
        B0BCLR      FT1IEN          ; 禁止 T1 中断。
        B0BCLR      FT1ENB          ; 关闭 T1 定时器。
        MOV         A, #20H         ;
        B0MOV       T1M, A          ; 设置 T1 时钟= Fcpu / 32，下降沿触发。
        CLR         T1CH
        CLR         T1CL

        B0BCLR      FT1IRQ          ; 清 T1IRQ。
        B0BSET      FT1IEN          ; 使能 T1 中断。
        B0BSET      FT1ENB          ; 开启 T1 定时器。

        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：T1 中断。**

```
        ORG         0FH             ; 中断向量。
        JMP         INT_SERVICE
INT_SERVICE:

        B0BTS1      FT1IRQ          ; 检查是否有 T1 中断请求。
        JMP         EXIT_INT        ; T1IRQ = 0，退出中断。

        B0BCLR      FT1IRQ          ; 清 T1IRQ。
        B0MOV       A, T1CL
        B0MOV       T1CLBUF,A
        B0MOV       A,T1CH
        B0MOV       T1CHBUF,A       ; 保存脉宽。
        CLR         T1CH
        CLR         T1CL
        …                           ; T1 中断程序。
        …
EXIT_INT:
        RETI                        ; 退出中断。
```

6.12 ADC中断

当 ADC 转换完成后，无论 ADCIEN 是否使能，ADCIQR 都会置“1”。若 ADCIEN 和 ADCIQR 都置“1”，那么系统就会响应 ADC 中断。若 ADCIEN = 0，不管 ADCIRQ 是否置“1”，系统都不会进入 ADC 中断。用户应注意多种中断下的处理。

➢ **例：ADC 中断设置。**

```
        B0BCLR      FADCIEN             ; 禁止 ADC 中断。

        MOV         A, #10110000B       ;
        B0MOV       ADM, A              ; 允许 P4.0 ADC 输入，使能 ADC 功能。
        MOV         A, #00000000B       ; 设置 AD 转换速率 = Fcpu/16。
        B0MOV       ADR, A

        B0BCLR      FADCIRQ             ; 清除 ADC 中断请求标志。
        B0BSET      FADCIEN             ; 使能 ADC 中断。
        B0BSET      FGIE                ; 使能 GIE。

        B0BSET      FADS                ; 开始 AD 转换。
```

➢ **例：ADC 中断服务程序。**

```
        ORG         10H                 ; 中断向量地址。
        JMP         INT_SERVICE
INT_SERVICE:

        …                               ; 保存 ACC 和 PFLAG。

        B0BTS1      FADCIRQ             ; 检查是否有 ADC 中断。
        JMP         EXIT_INT            ; ADCIRQ = 0，退出中断。

        B0BCLR      FADCIRQ             ; 清 ADCIRQ。
        …                               ; ADC 中断服务程序。
        …
EXIT_INT:
        …                               ; 恢复 ACC 和 PFLAG。

        RETI                            ; 退出中断。
```

6.13 SIO中断

SIO 成功传送后，无论 SIOIEN 处于何种状态，SIOIRQ 都会置“1”。若 SIOIEN 和 SIOIRQ 都置“1”，系统就会响应 SIO 中断；若 SIOIEN = 0，则无论 SIOIRQ 是否置“1”，系统都不会响应 SIO 中断。尤其需要注意多种中断下的情形。

➢ **例：设置 SIO 中断。**

```
        B0BSET      FSIOIEN     ; 使能 SIO 中断。
        B0BCLR      FSIOIRQ     ; 清 SIO 中断请求标志。
        B0BSET      FGIE        ; 使能 GIE。
```

➢ **例：SIO 中断。**

```
        ORG         11H         ; 中断向量。
        JMP         INT_SERVICE
INT_SERVICE:
        …                       ; 保存 ACC 和 PFLAG。

        B0BTS1      FSIOIRQ     ; 检查是否有 SIO 中断请求。
        JMP         EXIT_INT    ; SIOIRQ = 0，退出中断。

        B0BCLR      FSIOIRQ     ; 清 SIOIRQ。
        …                       ; SIO 中断程序。
        …
EXIT_INT:
        …                       ; 恢复 ACC 和 PFLAG。

        RETI                    ; 退出中断。
```

6.14 UART中断

UART 成功传送后，无论 URXIEN/URXIEN 处于何种状态，URXIRQ/UTXIQR 都会置 1。若 URXIEN/UTXIEN 和 URXIRQ/UTXIRQ 都置 1 时，系统才会响应 URX/UTX 中断。若 URXIEN/UTXIEN=0，则无论 URXIRQ/UTXIRQ 是否置 1，系统都不会响应 URX/UTX 中断。尤其需要注意多种中断下的情形。

➢ **例：设置 UART 接收和发送中断。**

```
        B0BSET      FURXIEN     ; 使能 UART 接收中断。
        B0BCLR      FURXIRQ     ; 清 UART 接收中断请求标志。

        B0BSET      FUTXIEN     ; 使能 UART  发送中断。
        B0BCLR      FUTXIRQ     ; 清 UART 发送中断请求标志。
        B0BSET      FGIE        ; 使能 GIE。
```

➢ **例：UART 接收中断。**

```
        ORG         13H         ; 中断向量。
        JMP         INT_SERVICE
INT_SERVICE:
        …                       ; 保存 ACC 和 PFLAG。

        B0BTS1      FURXIRQ     ; 检查是否有 URXIRQ 中断请求。
        JMP         EXIT_INT    ; URXIRQ = 0，退出中断。

        B0BCLR      FURXIRQ     ; 清 URXIRQ。
        …                       ; UART 接收中断程序。
        …
EXIT_INT:
        …                       ; 恢复 ACC 和 PFLAG。

        RETI                    ; 退出中断。
```

6.15 多中断操作

在同一时刻，系统中可能出现多个中断请求。处理这种多中断情况时，必须预先对各中断请求设置不同的优先级别。中断请求标志 IRQ 由中断事件触发，当 IRQ 处于有效值“1”时，系统并不一定会响应该中断。各中断触发事件列表如下：

中断名称	触发事件说明
WAKEIRQ	系统从睡眠模式或者绿色模式下被唤醒。
P00IRQ	P0.0 触发，由 PEDGE 控制。
P01IRQ	P0.1 触发，由 PEDGE 控制。
T0IRQ	T0C 溢出。
TC0IRQ	TC0C 溢出。
TC1IRQ	TC1C 溢出。
TC2IRQ	TC2C 溢出。
T1IRQ	T1CH、T1CL 溢出。
ADCIRQ	ADC 转换结束。
SIOIRQ	SIO 传送成功。
MSPIRQ	MSP 发送成功。
RXIRQ	UART 发送成功。
TXIRQ	UART 接收成功。

多个中断同时发生时，需要注意的是：1、该单片机具有多个中断向量，每个中断对应唯一的中断向量地址；2、用户必须定义中断向量。下面的示例程序中显示了如何在程序存储器中定义中断向量。

➢ **例：多中断操作下检查中断请求。**

```
        ORG     8               ; 中断向量。
        JMP     ISR_WAKE
        JMP     ISR_INT0
        JMP     ISR_INT1
        JMP     ISR_T0
        JMP     ISR_TC0
        JMP     ISR_TC1
        JMP     ISR_TC2
        JMP     ISR_T1
        JMP     ISR_ADC
        JMP     ISR_SIO
        JMP     ISR_MSP
        JMP     ISR_UART_RX
        JMP     ISR_UART_TX

ISR_WAKE:                       ; WAKE-UP 中断复位程序。

        RETI                    ; 退出中断。
ISR_INT0:                       ; INT0 中断服务程序。
                                ;
        RETI                    ; 退出中断。
ISR_INT1:                       ; INT1 中断服务程序。

        RETI                    ; 退出中断。
        …
        …

ISR_UART_TX:                    ; UART_TX 中断服务程序。

        RETI                    ; 退出中断。
```

7 I/O端口

7.1 概述

SN8F27E60 系列单片机共有 27 个 I/O 口，大多数 I/O 口模拟引脚即特殊功能的引脚共用。I/O 口共用情况如下表所示：

I/O 引脚		共用引脚		共用引脚控制条件
名称	类型	名称	类型	
P0.0	I/O	INT0	DC	P00IEN=1
		TC0	DC	TC0CKS=1，TC0ENB=1
P0.1	I/O	INT1	DC	P01IEN=1
		TC1	DC	TC1CKS=1，TC1ENB=1
P0.2	I/O	URX	DC	URXEN=1
		TC2	DC	TC2CKS=1，TC2ENB=1
P0.3	I/O	UTX	DC	UTXEN=1
		T1	DC	T1CKS=1，T1ENB=1
P0.4	I/O	RST	DC	Reset_Pin 编译选项 = Reset
P0.5	I/O	XOUT	AC	High_CLK 编译选项 = IHRC_RTC，32K，4M，12M
P0.6	I/O	XIN	AC	High_CLK 编译选项 = IHRC_RTC，RC，32K，4M，12M
P1.0	I/O	EICK	DC	嵌入式 ICE 模式
P1.1	I/O	EIDA	DC	嵌入式 ICE 模式
P1.2	I/O	SDA	DC	MSPENB=1
P1.3	I/O	SCL	DC	MSPENB=1
P1.4	I/O	SDO	DC	SENB=1
P1.5	I/O	SDI	DC	SENB=1
P1.6	I/O	SCK	DC	SENB=1
P1.7	I/O	SCS	DC	SENB=1
P4[7:0]	I/O	AIN[7:0]	AC	ADENB=1，GCHS=1，CHS[3:0]=0000b~0111b
P5.0	I/O	AIN[8]	AC	ADENB=1，GCHS=1，CHS[3:0]=1000b
P5.1	I/O	AIN[9]	AC	ADENB=1，GCHS=1，CHS[3:0]=1001b
		PWM0	DC	TC0ENB=1，PWM0OUT=1
P5.2	I/O	AIN[10]	AC	ADENB=1，GCHS=1，CHS[3:0]=1010b
		PWM1	DC	TC0ENB=1，PWM1OUT=1
P5.3	I/O	AIN[11]	AC	ADENB=1，GCHS=1，CHS[3:0]=1011b
		PWM2	DC	TC0ENB=1，PWM2OUT=1

*** DC：数字；AC：模拟；HV：高压**

7.2 I/O口模式

寄存器 PnM 的设置决定各 I/O 口的数据传送方向，所有的 I/O 口必须设置相应的输入、输出方向。

0A0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	-	P06M	P05M	P04M	P03M	P02M	P01M	P00M
读/写	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	0	0	0	0	0	0	0

0A1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1M	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0A4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4M	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0A5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	-	-	-	-	P53M	P52M	P51M	P50M
读/写	-	-	-	-	R/W	R/W	R/W	R/W
复位后	-	-	-	-	0	0	0	0

Bit[7:0] **PnM[7:0]**：Pn 模式控制位（n = 0~5）。
0 = 输入模式；
1 = 输出模式。

✷ **注： 可通过位控制指令 B0BSET、B0BCLR 设置模式寄存器。**

➢ **例：I/O 模式设置。**

```
CLR     P0M             ; 设置为输入模式。
CLR     P4M
CLR     P5M

MOV     A, #0FFH        ; 设置为输出模式。
B0MOV   P0M, A
B0MOV   P4M, A
B0MOV   P5M, A

B0BCLR  P4M.0           ; 设置为输入模式。

B0BSET  P4M.0           ; 设置为输出模式。
```

7.3 I/O口上拉电阻寄存器

I/O 引脚内置上拉电阻，只支持输入模式。内部上拉电阻有 PnUR 寄存器编程控制。PnUR 为 0 时，禁止上拉电阻；PnUR 为 1 时，使能上拉电阻。

0ACH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	-	P06R	P05R	P04R	P03R	P02R	P01R	P00R
读/写	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	0	0	0	0	0	0	0

0ADH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1UR	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0B0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4UR	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5UR					P53R	P52R	P51R	P50R
读/写					R/W	R/W	R/W	R/W
复位后					0	0	0	0

➢ **例：I/O 口上拉电阻。**

```
MOV     A, #0FFH        ; 使能 P0、P4 和 P5 的上拉电阻。
B0MOV   P0UR, A         ;
B0MOV   P4UR, A
B0MOV   P5UR, A
```

7.4 I/O口数据寄存器

0A6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	-	P06	P05	P04	P03	P02	P01	P00
读/写	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	0	0	0	0	0	0	0

0A7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1	P17	P16	P15	P14	P13	P12	P11	P10
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0AAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4	P47	P46	P45	P44	P43	P42	P41	P40
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0ABH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5					P53	P52	P51	P50
读/写					R/W	R/W	R/W	R/W
复位后					0	0	0	0

✷ **注：由编译选项使能外部服务时，P04 保持为 1。**

➢ **例：读取输入口的数据。**

```
B0MOV   A, P0       ; 读取 P0 口的数据。
B0MOV   A, P4       ; 读取 P4 口的数据。
B0MOV   A, P5       ; 读取 P5 口的数据。
```

➢ **例：写入数据到输出口。**

```
MOV     A, #0FFH    ; 写入数据 0FFH 到 P0、P4 和 P5。
B0MOV   P0, A
B0MOV   P4, A
B0MOV   P5, A
```

➢ **例：写入 1 位数据到输出口。**

```
B0BSET  P4.0        ; 设置 P4.0 和 P5.3 为 1。
B0BSET  P5.3

B0BCLR  P4.0        ; 清 P4.0 和 P5.3。
B0BCLR  P5.3
```

7.5 P4、P5 与ADC共用引脚

P4、P5 口和 ADC 的输入口共用，非施密特触发。同一时间只能设置 P4、P5 口的一个引脚作为 ADC 的测量信号输入口（通过 ADM 寄存器来设置），其它引脚则作为普通 I/O 使用。具体应用中，当输入一个模拟信号到 CMOS 结构端口，尤其当模拟信号为 1/2 VDD 时，将可能产生额外的漏电流。同样，当 P4、P5 口外接多个模拟信号时，也会产生额外的漏电流。在睡眠模式下，上述漏电流会严重影响到系统的整体功耗。P4CON 为 P4 口的配置寄存器，P5CON 为 P5 口的配置寄存器。将 P4CON.n 或者 P5CON.n 置 1 时，其对应的 P4、P5 口的引脚将被设置为纯模拟信号输入口，从而避免上述漏电流的情况。

0C6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4CON	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **P4CON[7:0]：** P4.n 控制位。
0 = P4.n 作为模拟信号输入引脚（ADC 输入引脚）或普通 I/O 引脚；
1 = P4.n 只能作为模拟信号输入引脚，不能作为普通 I/O 引脚。

0C7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5CON					P5CON3	P5CON2	P5CON1	P5CON0
读/写					R/W	R/W	R/W	R/W
复位后					0	0	0	0

Bit[3:0] **P5CON[3:0]：** P4.n 控制位。
0 = P5.n 作为模拟信号输入引脚（ADC 输入引脚）或普通 I/O 引脚；
1 = P5.n 只能作为模拟信号输入引脚，不能作为普通 I/O 引脚。

✷ **注：当 P4.n/P5.n 作为普通 I/O 口而不是 ADC 输入引脚时，P4CON.n/P5CON.n 必须置为 0，否则 P4.n/P5.n 的普通 I/O 信号会被隔离开来。**

P4/P5 的 ADC 模拟输入由寄存器 ADM 的 GCHS 和 CHSn 位控制，若 GCHS = 0，P4.n/P5.n 为普通的 I/O 引脚，若 GCHS = 1，CHSn 所对应的 P4.n/P5.n 用作 ADC 模拟信号输入引脚。

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	CHS3	CHS2	CHS1	CHS0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 4 **GCHS：** ADC 输入通道控制位。
0 = 禁止 AIN 通道；
1 = 开启 AIN 通道。

Bit[3:0] **CHS[3:0]：** ADC 输入通道选择位。
0000 = AIN0；0001 = AIN1；0010 = AIN2；0011 = AIN3；0100 = AIN4；0101 = AIN5；0110 = AIN6；
0111 = AIN7；1000 = AIN8；1001 = AIN9；1010 = AIN10；1011 = AIN11。

✷ **注：在设置 P4.n/P5.n 为普通的 I/O 引脚时，必须保证 P4.n/P5.n 的 ADC 功能已经被禁止。否则 GCHS=1，CHS[3:0]指向 P4.n/P5.n 时，会自动将 P4.n/P5.n 设置为 ADC 模拟输入引脚。**

➢ **例：设置 P4.1 为普通的输入引脚，P4CON.1 必须置为 0。**

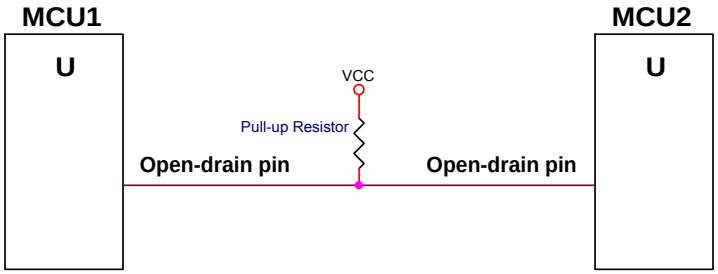
```
; 检查 GCHS 和 CHS[3:0]的状态。
        B0BCLR      FGCHS       ; 若 CHS[3:0]指向 P4.1（CHS[3:0] = 0001B），GCHS=0。
                                ; 若 CHS[3:0]没有指向 P4.1，则忽略 GCHS 的状态。
; 清 P4CON。
        B0BCLR      P4CON.1     ; 使能 P4.1 的普通 I/O 功能。
; P4.1 设为输入模式。
        B0BCLR      P4M.1       ; 设置 P4.1 为输入模式。
```

➢ **例：设置 P4.1 为普通的输出模式，P4CON.1 必须置为 0。**

```
; 检查 GCHS 和 CHS[3:0]的状态。
        B0BCLR      FGCHS       ; 若 CHS[3:0]指向 P4.1（CHS[3:0] = 0001B），GCHS=0。
                                ; 若 CHS[3:0]没有指向 P4.1，则忽略 GCHS 的状态。
; 清 P4CON。
        B0BCLR      P4CON.1     ; 使能 P4.1 的普通 I/O 功能。
; 设置 P4.1 为输出模式以避免误操作。
        B0BSET      P4.1        ; 设置 P4.1 为 1。
; 或
        B0BCLR      P4.1        ; 设置 P4.1 为 0。
; P4.1 设为输出模式。
        B0BSET      P4M.1       ; 设置 P4.1 为输出模式。
```

7.6 漏极开路寄存器

P0.2/P0.3/P1.0~P1.7 内置漏极开路功能，当使能该功能时，P0.2/P0.3/P1.0~P1.7 必须设置为输出模式。漏极开路的外部电路如下图所示：



上图中的上拉电阻必不可少，漏极开路的输出高电平由上拉电阻驱动，输出低电平由灌电流驱动。

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0OC	-	-	-	-	-	-	P03OC	P02OC
读/写	-	-	-	-	-	-	R/W	R/W
复位后	-	-	-	-	-	-	0	0

Bit [1:0] **P02OC, P03OC**：P0.2, P0.3 漏极开路控制位。
0 = 禁止漏极开路功能；
1 = 开启漏极开路功能。

09DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1OC	P17OC	P16OC	P15OC	P14OC	P13OC	P12OC	P11OC	P10OC
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **P10OC~P17OC**：P1.0~P1.7 漏极开路控制位。
0 = 禁止漏极开路功能；
1 = 开启漏极开路功能。

➢ **例：开启 P1.0 的漏极开路功能并输出高电平。**

```
B0BSET      P1.0        ; P1.0 置高。

B0BSET      P10M        ; P1.0 设为输出模式。
B0MOV       P1OC, A     ; 开启 P1.0 的漏极开路功能。
```

➢ **例：禁止漏极开路功能。**

```
B0BCLR      P10OC       ; 禁止 P1.0 的漏极开路功能。
```

✷ **注：禁止漏极开路功能后，恢复到之前的 I/O 模式。**

8 定时器

8.1 看门狗定时器

看门狗定时器 WDT 是一个 4 位二进制计数器，用于监控程序的正常执行。如果由于干扰，程序进入了未知状态，看门狗定时器溢出，系统复位。看门狗的分频选择由编译选项 WDT_CLK 控制，其时钟源由内部低速 16KHz RC 振荡器提供。

看门狗间隔时间 = 256 * 1/（内部低速振荡器频率/WDT 分频）…sec = 256/（16KHz/WDT 分频）…sec

内部低速振荡器	WDT 分频	看门狗间隔时间
Flosc=16KHz	Flosc/4	256/(16000/4)=64ms
	Flosc/8	256/(16000/8)=128ms
	Flosc/16	256/(16000/16)=256ms
	Flosc/32	256/(16000/32)=512ms

看门狗定时器有三种工作模式，由编译选项“WatchDog”控制。

- **Disable**：关闭看门狗定时器。
- **Enable**：开启看门狗定时器，但在睡眠模式和绿色模式下会关闭看门狗定时器。
- **Always_On**：始终开启看门狗定时器，即使在睡眠模式和绿色模式下也处于开启状态。

✷ **注：在高干扰环境下，强烈建议将看门狗定时器设置为“Always_On”选项，保证系统进入错误环境时复位以重新启动系统。**

看门狗清零的方法是对看门狗计数器清零寄存器 WDTR 写入清零控制字 5AH。

096H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WDTR	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

➢ **例：下面是对看门狗定时器操作的示例程序，在主程序的开始清看门狗定时器。**

```
Main:
        MOV     A, #5AH         ; 清看门狗定时器。
        B0MOV   WDTR, A
        …
        CALL    SUB1
        CALL    SUB2
        …
        JMP     Main
```

➢ **例：用宏指令@RST_WDT 清看门狗定时器。**

```
Main:
        @RST_WDT                ; 清看门狗定时器。
        …
        CALL    SUB1
        CALL    SUB2
        …
        JMP     Main
```

看门狗定时器应用注意事项如下：

- 对看门狗清零之前，检查 I/O 口的状态和 RAM 的内容可增强程序的可靠性；
- 不能在中断中对看门狗清零，否则无法侦测到主程序跑飞的状况；
- 程序中应该只在主程序中有一次清看门狗的动作，这种架构能够最大限度的发挥看门狗的保护功能。

➢ **例：下面是对看门狗定时器操作的示例程序，在主程序的开始清看门狗定时器。**

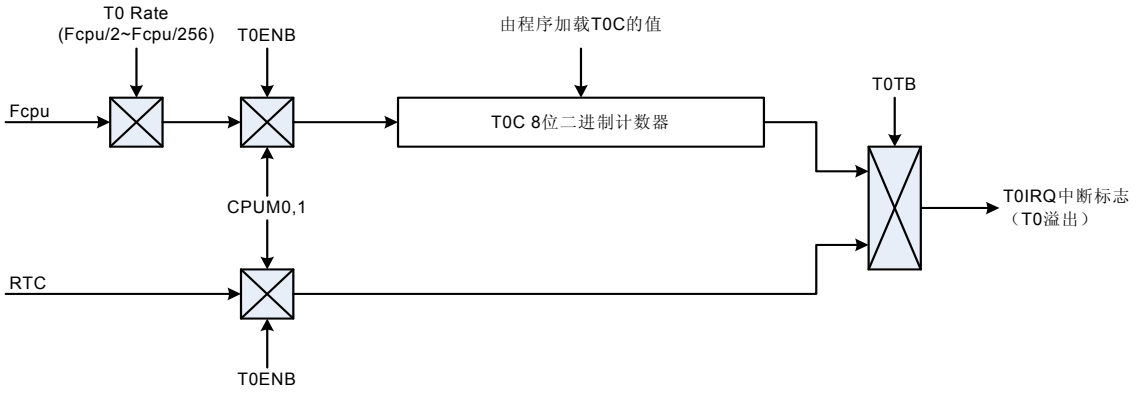
```
Main:
                …                               ; 检查 I/O 口的状态。
                …                               ; 检查 RAM 的内容。
Err:            JMP $                           ; I/O 口或 RAM 出错，不清看门狗等看门狗计时溢出。

Correct:                                        ; I/O 口和 RAM 都正确，清看门狗定时器。
                                                ;
                MOV         A,#5AH              ; 清看门狗定时器。
                B0MOV       WDTR,A
                …
                CALL        SUB1
                CALL        SUB2
                …
                JMP         Main
```

8.2 8 位基本定时器T0

8.2.1 概述

8 位二进制基本定时器 T0，支持标志指示（T0IRQ）和中断操作（中断向量）。可以通过 T0M 和 T0C 寄存器控制间隔时间，具有 RTC 功能和在绿色模式下唤醒功能。在绿色模式下，T0 溢出，则将系统唤醒返回到上一个工作模式。

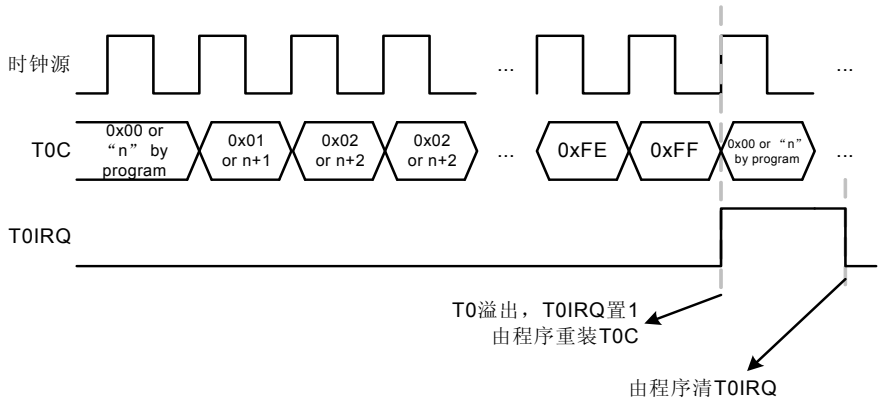
- **8 位可编程计数定时器：**根据选择的时钟频率周期性的产生中断请求。
- **中断功能：**T0 定时器支持中断功能，当 T0 溢出，T0IRQ 有效，程序计数器跳到中断向量地址执行中断。
- **RTC 功能：**T0 支持 RTC 功能。T0TB=1 时，RTC 时钟源由外部低速 32K 振荡器提供。RTC 功能仅在 High_Clk 选择 IHRC_RTC 时有效。
- **绿色模式唤醒功能：**T0 定时器在绿色模式下正常工作，溢出时将系统从绿色模式下唤醒。



✷ **注：RTC 模式下，T0 的间隔时间固定为 0.5S，T0C 的值为 256。**

8.2.2 T0 操作

T0 定时器由 T0ENB 控制。当 T0ENB=0 时，T0 停止工作；当 T0ENB=1 时，T0 开始计数。T0C 溢出（从 0FFH 到 00H）时，T0IRQ 置 1 显示溢出状态并由程序清零。T0 溢出时由程序加载新值给 T0C，以选定合适的间隔时间。如果使能 T0 中断（T0IEN=1），T0 溢出后系统执行中断服务程序，在中断下必须由程序清 T0IRQ。T0 可以在普通模式、低速模式和绿色模式下工作，绿色模式下，T0 溢出时 T0IRQ 置 1，系统被唤醒。



T0 的时钟源为 Fcpu（指令周期），由 T0Rate[2:0]决定。详见下表：

T0rate[2:0]	T0 时钟	T0 间隔时间					
		Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4		IHRC_RTC mode	
		max. (ms)	Unit (us)	max. (ms)	Unit (us)	max. (sec)	Unit (ms)
000b	Fcpu/256	16.384	64	65.536	256	-	-
001b	Fcpu/128	8.192	32	32.768	128	-	-
010b	Fcpu/64	4.096	16	16.384	64	-	-
011b	Fcpu/32	2.048	8	8.192	32	-	-
100b	Fcpu/16	1.024	4	4.096	16	-	-
101b	Fcpu/8	0.512	2	2.048	8	-	-
110b	Fcpu/4	0.256	1	1.024	4	-	-
111b	Fcpu/2	0.128	0.5	0.512	2	-	-
-	32768Hz/64	-	-	-	-	0.5	1.953

8.2.3 T0M模式寄存器

模式寄存器 T0M 设置 T0 的工作模式，包括 T0 前置分频器、时钟源等，这些设置必须在使能 T0 定时器之前完成。

0B2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	-	-	-	T0TB
读/写	R/W	R/W	R/W	R/W	-	-	-	R/W
复位后	0	0	0	0	-	-	-	0

Bit 0 **T0TB**：RTC 时钟源控制位。
0 = 禁止 RTC（T0 时钟源来自 Fcpu）；
1 = 使能 RTC。

Bit [6:4] **T0RATE[2:0]**：T0 分频选择位。
000 = fcpu/256；
001 = fcpu/128；
…；
110 = fcpu/4；
111 = fcpu/2。

Bit 7 **T0ENB**：T0 启动控制位。
0 = 禁止；
1 = 使能。

✷ **注：RTC 模式下，T0RATE 处于无效状态。T0 的间隔时间固定为 0.5S。**

8.2.4 T0C计数寄存器

8 位计数器 T0C 溢出时，T0IRQ 置 1 并由程序清零，用来控制 T0 的中断间隔时间。必须保证写入正确的值到 T0C 寄存器，然后使能 T0 定时器以保证第一个周期准确无误。T0 溢出后，由程序加载一个正确的值到 T0C 寄存器。

0B3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0C	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

T0C 初始值的计算公式如下：

T0C 初始值 = 256 -（T0 中断间隔时间 * T0 时钟 Rate）

➢ **例：计算 T0C，T0 的间隔时间为 10ms，T0 时钟源 Fcpu = 4MHz/4=1MHz，T0RATE = 001（Fcpu/128）。**
T0 间隔时间=10ms，T0 时钟 Rate=4MHz/4/128

T0C 初始值 = 256 -（T0 中断间隔时间 * 输入时钟）
= 256 -（10ms * 4MHz / 4 / 128）
= 256 -（10-2 * 4 * 106 / 4 /128）
= B2H

✷ **注：RTC 模式下，T0C 为 256，T0 的间隔时间为 0.5S。不能在 RTC 模式下修改 T0C 的值。**

8.2.5 T0 操作举例

- **T0 定时器：**

```
; 复位 T0 定时器。
        MOV     A,#00H          ; 清 T0M。
        B0MOV   T0M,A

; 设置 T0 时钟源和 T0Rate。
        MOV     A, #0nnn0000b
        B0MOV   T0M, A

; 设置 T0C 寄存器获取 T0 间隔时间。
        MOV     A, #value
        B0MOV   T0C, A

; 清 T0IRQ。
        B0BCLR  FT0IRQ

; 使能 T0 定时器和中断功能。
        B0BSET  FT0IEN          ; 使能 T0 中断。
        B0BSET  FT0ENB          ; 使能 T0 定时器。
```

- **T0 在 RTC 模式下工作：**

```
; 复位 T0 定时器。
        MOV     A, #00H         ; 清 T0M。
        B0MOV   T0M, A

; 设置 T0 RTC 功能。
        B0BSET  FT0TB

; 清 T0C。
        CLR     T0C

; 清 T0IRQ。
        B0BCLR  FT0IRQ

; 使能 T0 定时器和中断功能。
        B0BSET  FT0IEN          ; 使能 T0 中断。
        B0BSET  FT0ENB          ; 使能 T0 定时器。
```

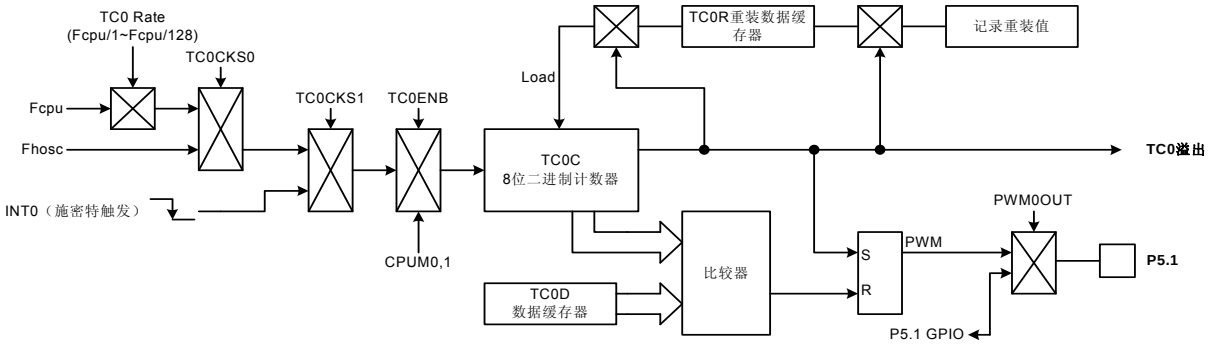
8.3 8 位定时/计数器TC0

8.3.1 概述

8 位二进制定时/计数器具有基本定时器、事件计数器和 PWM 功能。基本定时器功能可以支持标志显示（TC0IRQ）和中断操作（中断向量）。由 TC0M、TC0C、TC0R 寄存器控制 TC0 的中断间隔时间。事件计数器可以将 TC0 时钟源由系统时钟更改为外部时钟信号（如连续的脉冲、R/C 振荡信号等）。TC0 作为计数器时记录外部时钟数目以进行测量应用。TC0 还内置周期/占空比可编程控制的 PWM 功能，PWM 的周期和分辨率由 TC0 时钟 Rate、TC0R 和 TC0D 寄存器控制，故具有良好性能的 PWM 可以处理 IR 载波信号，马达控制和光度调节等。

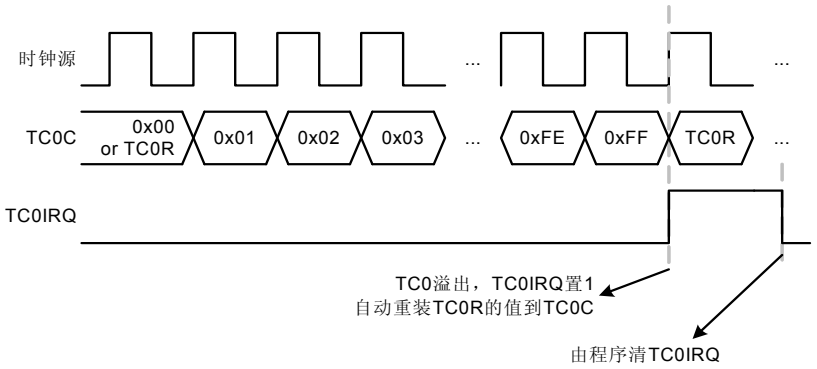
TC0 的主要用途如下：

- **8 位可编程定时器：**根据选择的时钟信号，产生周期中断；
- **中断功能：**TC0 定时器支持中断，当 TC0 溢出时，TC0IRQ 置 1，系统执行中断；
- **外部事件计数器：**对外部事件计数；
- **PWM 输出：**由 TC0D 和 TC0R 寄存器控制占空比/周期；
- **绿色模式功能：**绿色模式下，TC0 正常工作，但无唤醒功能。



8.3.2 TC0 操作

TC0 定时器由 TC0ENB 控制。当 TC0ENB=0 时，TC0 停止工作；当 TC0ENB=1 时，TC0 开始计数。使能 TC0 之前，先要设定好 TC0 的功能模式，如基本定时器、TC0 中断等。TC0C 溢出（从 0FFH 到 00H）时，TC0IRQ 置 1 以显示溢出状态并由程序清零。在不同的功能模式下， TC0C 不同的值对应不同的操作，若改变 TC0C 的值影响到操作，会导致功能出错。TC0 内置双重缓存器以避免此种状况的发生。在 TC0C 计数的过程中不断的刷新 TC0C，保证将最新的值存入 TC0R（重装缓存器）中，当 TC0 溢出后，TC0R 的值由自动存入 TC0C。进入下一个周期后，TC0 按新的配置工作。使能 TC0 时，自动使能 TC0 的自动重装功能。如果使能 TC0 中断功能（TC0IEN=1），在 TC0 溢出时系统执行中断服务程序，在中断时必须由程序清 TC0IRQ。TC0 可以在普通模式、低速模式和绿色模式下工作。但在绿色模式下，TC0 虽继续工作，但不能唤醒系统。



TC0 根据不同的时钟源选择不同的应用模式，TC0 的时钟源由 Fcpu（指令周期）、Fhosc（高速振荡时钟）和外部引脚输入（P0.0）提供，由 TC0CKS[1:0]控制。TC0CKS0 选择时钟源来自 Fcpu 或者 Fhosc，当 TC0CKS0=0 时，TC0 时钟源来自 Fcpu，可以由 TC0Rate[2:0]选择不同的分频。当 TC0CKS0=1 时，TC0 时钟源来自 Fhosc，可以由 TC0Rate[2:0]选择不同的分频。TC0CKS1 决定时钟源由外部引脚输入或者由 TC0CKS0 控制，TC0CKS1=0 时，TC0 的时钟源由 TC0CKS0 控制，TC0CKS1=1 时，TC0 时钟源由外部输入引脚提供，此时使能外部事件计数功能。TC0CKS1=1 时，TC0Rate[2:0]处于无效状态。TC0 为 8 位定时器（256 阶），每经过一个 TC0 时钟周期计一个点。

TC0CKS0	TC0rate[2:0]	TC0 时钟	TC0 间隔时间			
			Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
			max. (ms)	Unit (us)	max. (ms)	Unit (us)
0	000b	Fcpu/128	8.192	32	32.768	128
0	001b	Fcpu/64	4.096	16	16.384	64
0	010b	Fcpu/32	2.048	8	8.192	32
0	011b	Fcpu/16	1.024	4	4.096	16
0	100b	Fcpu/8	0.512	2	2.048	8
0	101b	Fcpu/4	0.256	1	1.024	4
0	110b	Fcpu/2	0.128	0.5	0.512	2
0	111b	Fcpu/1	0.064	0.25	0.256	1
1	000b	Fhosc/128	2.048	8	8.192	32
1	001b	Fhosc/64	1.024	4	4.096	16
1	010b	Fhosc/32	0.512	2	2.048	8
1	011b	Fhosc/16	0.256	1	1.024	4
1	100b	Fhosc/8	0.128	0.5	0.512	2
1	101b	Fhosc/4	0.064	0.25	0.256	1
1	110b	Fhosc/2	0.032	0.125	0.128	0.5
1	111b	Fhosc/1	0.016	0.0625	0.064	0.25

8.3.3 TC0M模式寄存器

模式寄存器 TC0M 控制 TC0 的工作模式，包括 TC0 分频、时钟源和 PWM 功能等。 这些设置必须在使能 TC0 定时器之前完成。

0B4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS1	TC0CKS0	-	PWM0OUT
读/写	R/W	R/W	R/W	R/W	R/W	R/W	-	R/W
复位后	0	0	0	0	0	0	-	0

Bit 0 **PWM0OUT**：PWM 输出控制位。
0 = 禁止 PWM 输出，P5.1 为普通 I/O 引脚；
1 = 允许 PWM 输出，P5.1 输出 PWM 信号。

Bit 2 **TC0CKS 0**：TC0 时钟源选择位。。
0 = Fcpu；
1 = Fhosc。

Bit 3 **TC0CKS1**：TC0 时钟源选择位。
0 = 内部时钟（Fcpu 或者 Fhosc，由 TC0CKS0 控制）；
1 = 外部时钟信号（P0.0/INT0），使能事件计数器功能，TC0Rate[2:0]处于无效状态。

Bit [6:4] **TC0RATE[2:0]**：TC0 分频选择位。
TC0CKS0=0 -> 000 = Fcpu/128；001 = Fcpu/64；010 = Fcpu/32；011 = Fcpu/16；100 = Fcpu/8；
101 = Fcpu/4；110 = Fcpu/2；111 = Fcpu/1。
TC0CKS0=1 -> 000 = Fhosc/128；001 = Fhosc/64；010 = Fhosc/32；011 = Fhosc/16；100 = Fhosc/8；
101 = Fhosc/4；110 = Fhosc/2；111 = Fhosc/1。

Bit 7 **TC0ENB**：TC0 启动控制位。
0 = 关闭 TC0 定时器；
1 = 开启 TC0 定时器。

8.3.4 TC0C计数寄存器

8 位计数器 TC0C 溢出时，TC0IRQ 置 1 并由程序清零，用来控制 TC0 的中断间隔时间。首先须写入正确的值到 TC0C 和 TC0R 寄存器，并使能 TC0 定时器以保证第一个周期正确。TC0 溢出后，TC0R 的值自动装入 TC0C。

0B5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0C	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

TC0C 初始值的计算公式如下：

TC0C 初始值 = 256 -（TC0 中断间隔时间 * TC0 时钟 Rate）

8.3.5 TC0R自动重装寄存器

TC0 内置自动重装功能，TC0R 寄存器存储重装值。TC0C 溢出时，TC0R 的值自动装入 TC0C 中。TC0 定时器工作在计时模式时，要通过修改 TC0R 寄存器来修改 TC0 的间隔时间，而不是通过修改 TC0C 寄存器。在 TC0 定时器溢出后，新的 TC0C 值会被更新，TC0R 会将新的值装载到 TC0C 寄存器中。但在初次设置 TC0M 时，必须要在开启 TC0 定时器前把 TC0C 以及 TC0R 设置成相同的值。

TC0 为双重缓存器结构。若程序对 TC0R 进行了修改，那么修改后的 TC0R 值首先被暂存在 TC0R 的第一个缓存器中，TC0 溢出后，TC0R 的新值就会被存入 TC0R 缓存器中，从而避免 TC0 中断时间出错以及 PWM 误动作。

0B6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

TC0R 初始值计算公式如下：

TC0R 初始值 = 256 -（TC0 中断间隔时间 * TC0 时钟 Rate）

- **例：计算 TC0C 和 TC0R 的值，TC0 的间隔时间为 10ms，时钟源来自 Fcpu= 16MHz/16=1MHz，TC0RATE=000（Fcpu/128）。**

TC0 间隔时间为 10ms，TC0 时钟 Rate=16MHz/16/128

TC0C/TC0R 初始值 = 256 -（TC0 中断间隔时间 * 输入时钟）
= 256 -（10ms * 16MHz / 16 / 128）
= 256 -（10-2 * 16 * 106 / 16 / 128）
= B2H

8.3.6 TC0D PWM占空比寄存器

TC0D 寄存器用来控制 PWM 的占空比。PWM 模式下，TC0R 控制 PWM 的周期，TC0D 控制 PWM 的占空比。TC0C=TC0D 时，PWM 切换为低电平。这样在应用中易于设置 TC0D 以选择合适的 PWM 占空比。

0B7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0D	TC0D7	TC0D6	TC0D5	TC0D4	TC0D3	TC0D2	TC0D1	TC0D0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

TC0D 初始值的计算方法如下：

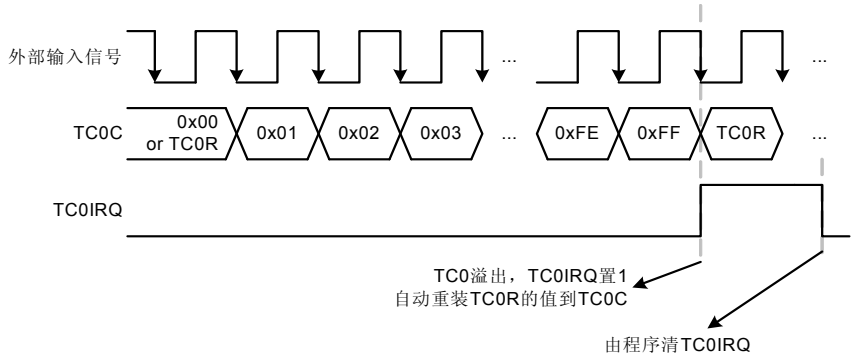
TC0D 初始值 = TC0R +（PWM 脉冲高电平宽度周期 / TC0 时钟 rate）

- **例：计算 TC0D 的值。1/3 占空比 PWM，TC0 时钟源 Fcpu=16MHz/16=1MHz，TC0RATE=000（Fcpu/128）。TC0R = B2H，TC0 间隔时间=10ms，PWM 周期频率为 100Hz，1/3 占空比条件下，PWM 高电平的宽度值约为 3.33ms。**

TC0D 初始值 = B2H +（PWM 脉冲高电平宽度值/TC0 时钟 Rate）
= B2H + (3.33ms * 16MHz / 16 / 128)
= B2H + 1AH
= CCH

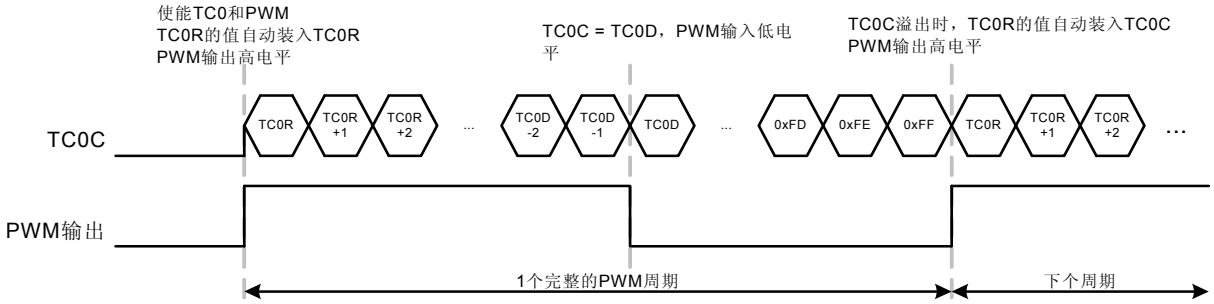
8.3.7 TC0 事件计数器

TC0 作为外部事件计数器时，其时钟源由外部输入引脚（P0.0）提供。当 TC0CKS1=1 时，TC0 的时钟源由外部输入引脚（P0.0）提供，下降沿触发。TC0C 溢出（从 FFH 到 00H）时， TC0 触发事件计数器溢出。使能外部事件计数功能，同时外部输入引脚的唤醒功能被禁止以避免外部事件的触发信号将系统唤醒而耗电。此时，P0.0 的外部中断功能也被禁止，即 P00IRQ=0。外部事件计数器通常用来测量外部连续信号的比率，如连续的脉冲信号，R/C 振荡信号等，外部信号的相位与 MCU 时钟的相位并不同步，通过 TC0 事件计数器的测量和计算以达到不同的应用。

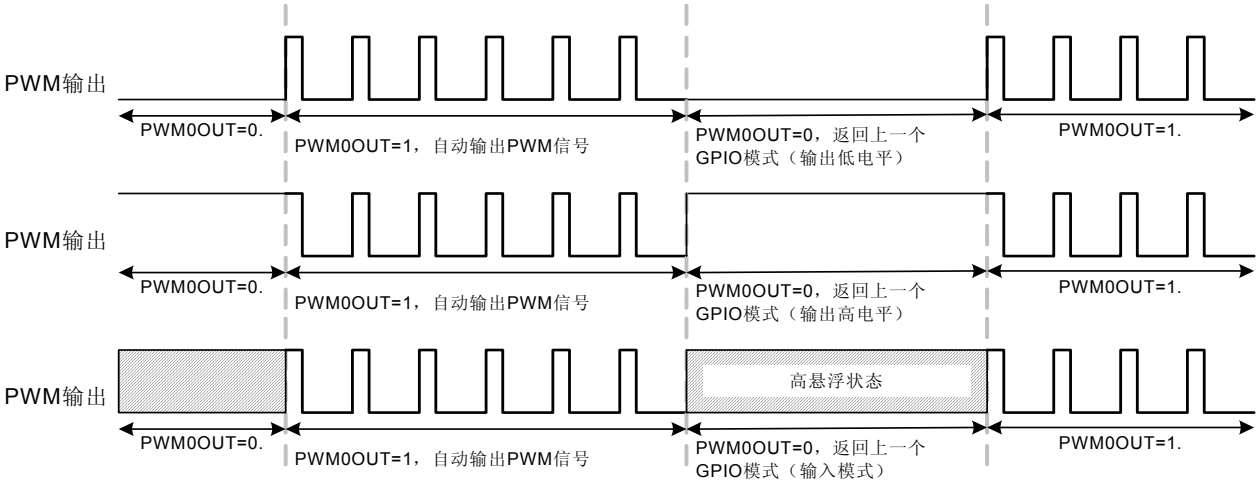


8.3.8 脉冲宽度调制（PWM）

可编程控制占空比/周期的 PWM 可以提供不同的 PWM 信号。使能 TC0 定时器且 PWM0OUT=1 时，由 PWM 输出引脚（P5.1）输出 PWM 信号。PWM 首先输出高电平，然后输出低电平。TC0R 寄存器控制 PWM 的周期，TC0D 控制 PWM 的占空比（脉冲高电平的长度）。开启 TC0 定时器且定时器溢出后，TC0R 重装载 TC0C 的初始值。当 TC0C=TC0D 时，PWM 输出低电平；TC0 溢出时（TC0C 的值从 0FFH 到 00H），整个 PWM 周期完成，并进入下一个周期。TC0 溢出时，TC0R 的值自动装入 TC0C，PWM 的一个周期完成，以保持 PWM 的连贯性。在 PWM 输出的过程由程序更改 PWM 的占空比，则在下一个周期开始输出新的占空比的 PWM 信号。



PWM 的分辨率由 TC0R 决定，而 TC0R 的范围值为 00H~0FFH。当 TC0R=00H 时，PWM 的分辨率为 1/256；TC0R=80H 时，PWM 的分辨率为 1/128。TC0D 控制 PWM 高电平脉冲的宽度，即 PWM 的占空比。当 TC0C=TC0D 时，PWM 输出低电平，TC0D 的值必须大于 TC0R 的值，否则 PWM 的信号保持低电平状态。PWM 输出过程中，TC0 溢出时，TC0IRQ 有效，TC0IEN=1 时，则使能 TC0 中断。但强烈建议小心同时使用 PWM 和 TC0 定时器功能，保证两种功能都能正常工作。PWM 的输出引脚与 GPIO 共用，PWM0OUT=1 时，自动输出 PWM 信号；PWM0OUT=0，即禁止 PWM 时，该引脚自动返回到上一个 GPIO 模式。这样有利于处理 ON/OFF 操作的载波信号，而不控制 TC0ENB 位。



8.3.9 TC0 操作举例

- **TC0 定时器**

```
; 复位 TC0。
        CLR     TC0M            ; 清 TC0M。

; 设置 TC0 时钟源和 TC0Rate。
        MOV     A, #0nnn0n00b
        B0MOV   TC0M, A

; 设置 TC0C 和 TC0R 获得 TC0 的间隔时间。
        MOV     A, #value
        B0MOV   TC0C, A
        B0MOV   TC0R, A

; 清 TC0IRQ。
        B0BCLR  FTC0IRQ

; 使能 TC0 定时器和中断功能。
        B0BSET  FTC0IEN         ; 使能 TC0 中断。
        B0BSET  FTC0ENB         ; 使能 TC0 定时器。
```

- **TC0 事件计数器**

```
; 复位 TC0。
        CLR     TC0M            ; 清 TC0M。

; 使能 TC0 事件计数器。
        B0BSET  FTC0CKS1        ; 设置 TC0 的时钟源由外部输入引脚（P0.0）提供。

; 设置 TC0C 和 TC0R 寄存器获得 TC0 的间隔时间。
        MOV     A, #value       ; TC0C 必须和 TC0R 相等。
        B0MOV   TC0C, A
        B0MOV   TC0R, A

; 清 TC0IRQ。
        B0BCLR  FTC0IRQ

; 使能 TC0 定时器和中断功能。
        B0BSET  FTC0IEN         ; 使能 TC0 中断。
        B0BSET  FTC0ENB         ; 使能 TC0 定时器。
```

- **TC0 PWM**

```
; 复位 TC0。
        CLR     TC0M            ; 清 TC0M。

; 设置 TC0 时钟源和 TC0Rate。
        MOV     A, #0nnn0n00b
        B0MOV   TC0M, A

; 设置 TC0C 和 TC0R 寄存器获得 PWM 周期。
        MOV     A, #value1
        B0MOV   TC0C, A
        B0MOV   TC0R, A

; 设置 TC0D 寄存器获得 PWM 占空比。
        MOV     A, #value2      ; TC0D 的值必须大于 TC0R 的值。
        B0MOV   TC0D, A

; 使能 PWM 和 TC0 定时器。
        B0BSET  FTC0ENB         ; 使能 TC0 定时器。
        B0BSET  FPWM0OUT        ; 使能 PWM。
```

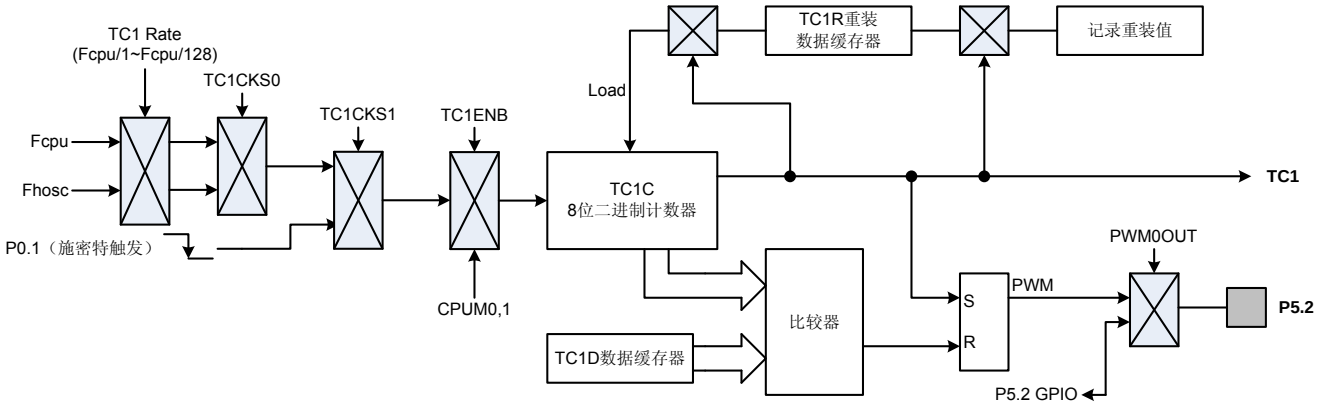
8.4 8 位定时/计数器TC1

8.4.1 概述

8 位二进制定时器 TC1 具有基本定时器、事件计数器和 PWM 功能。基本定时器功能可以支持中断请求标志的显示（TC1IRQ）和中断操作（中断向量）。由 TC1M、TC1C、TC1R 寄存器控制 TC1 的中断间隔时间。事件计数器可以将 TC1 时钟源由系统时钟（Fcpu/Fhosc）更改为外部时钟信号（如连续的脉冲、R/C 振荡信号等）。TC1 作为计数器时记录外部时钟数目以进行测量应用。TC1 还内置周期/占空比可编程控制的 PWM 功能，PWM 的周期和分辨率由 TC1 时钟周期、TC1R 和 TC1D 寄存器控制，故具有良好性能的 PWM 可以处理 IR 载波信号，马达控制和光度调节等。

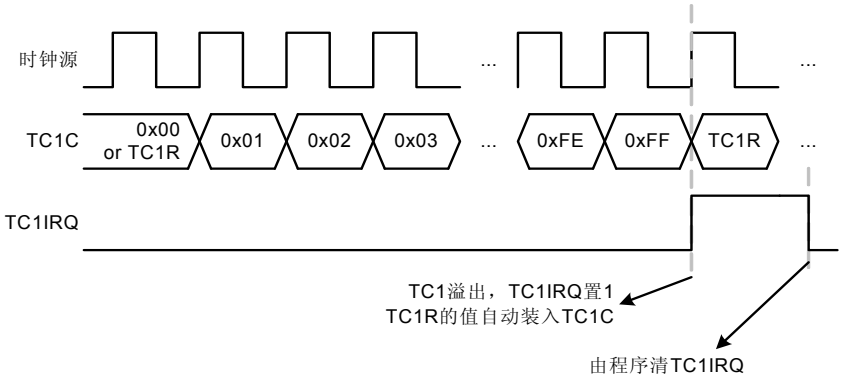
TC1 的主要用途如下：

- **8 位可编程定时器：**根据选择的时钟信号，产生周期性中断；
- **中断功能：**TC0 定时器支持中断，当 TC0 溢出时，TC0IRQ 置 1，系统执行中断；
- **外部事件计数器：**对外部事件计数；
- **可编程控制占空比/周期的 PWM 输出：**由 TC1R 和 TC1D 寄存器控制占空比/周期；
- **绿色模式功能：**绿色模式下，TC1 正常工作，但无唤醒功能。



8.4.2 TC1 操作

TC1 定时器由 TC1ENB 控制。当 TC1ENB=0 时，TC1 停止工作；当 TC1ENB=1 时，TC1 开始计数。使能 TC1 之前，先要设定好 TC1 的功能模式，如基本定时器、TC1 中断等。TC1C 溢出（从 0FFH 到 00H）时，TC1IRQ 置 1 以显示溢出状态并由程序清零。在不同的功能模式下，TC1C 不同的值对应不同的操作，若改变 TC1C 的值影响到操作，会导致功能出错。TC1 内置双重缓存器以避免此种状况的发生。在 TC1C 计数的过程中不断的刷新 TC1C，保证将最新的值存入 TC1R（重装缓存器）中，当 TC1 溢出后，新的 TC1R 值将自动装载到 TC1C。进入下一个周期后，TC1 按新的配置工作。定时/计数器模式时，使能 TC1 时，自动使能自动重装功能。如果使能 TC1 中断功能（TC1IEN=1），在 TC1 溢出时系统执行中断服务程序，在中断时必须由程序清 TC1IRQ。TC1 可以在普通模式、低速模式和绿色模式下工作。但在绿色模式下，TC1 虽继续工作，但不能唤醒系统。



TC1 根据不同的时钟源选择不同的应用模式，TC1 的时钟源由 Fcpu（指令周期）、Fhosc（高速振荡时钟）和外部引脚输入（P0.1）提供，由 TC1CKS[1:0]控制。TC1CKS0 选择时钟源来自 Fcpu 或者 Fhosc，当 TC1CKS0=0 时，TC1 时钟源来自 Fcpu，可以由 TC1Rate[2:0]选择不同的分频。当 TC1CKS0=1 时，TC1 时钟源来自 Fhosc，可以有 TC1Rate[2:0]选择不同的分频。TC1CKS1 决定时钟源由外部引脚输入或者由 TC1CKS0 控制，TC1CKS1=0 时，TC1 的时钟源由 TC1CKS0 控制，TC1CKS1=1 时，TC1 时钟源由外部输入引脚提供，此时使能外部事件计数功能。TC1CKS1=1 时，TC1Rate[2:0]处于无效状态。TC1 为 8 位定时器（256 阶），每经过一个 TC1 时钟周期计一个点。

TC1CKS0	TC1rate[2:0]	TC1 时钟	TC1 间隔时间			
			Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
			max. (ms)	Unit (us)	max. (ms)	Unit (us)
0	000b	Fcpu/128	8.192	32	32.768	128
0	001b	Fcpu/64	4.096	16	16.384	64
0	010b	Fcpu/32	2.048	8	8.192	32
0	011b	Fcpu/16	1.024	4	4.096	16
0	100b	Fcpu/8	0.512	2	2.048	8
0	101b	Fcpu/4	0.256	1	1.024	4
0	110b	Fcpu/2	0.128	0.5	0.512	2
0	111b	Fcpu/1	0.064	0.25	0.256	1
1	000b	Fhosc/128	2.048	8	8.192	32
1	001b	Fhosc/64	1.024	4	4.096	16
1	010b	Fhosc/32	0.512	2	2.048	8
1	011b	Fhosc/16	0.256	1	1.024	4
1	100b	Fhosc/8	0.128	0.5	0.512	2
1	101b	Fhosc/4	0.064	0.25	0.256	1
1	110b	Fhosc/2	0.032	0.125	0.128	0.5
1	111b	Fhosc/1	0.016	0.0625	0.064	0.25

8.4.3 TC1M模式寄存器

模式寄存器 TC1M 控制 TC1 的工作模式，包括 TC1 分频、时钟源和 PWM 功能等。 这些设置必须在使能 TC1 定时器之前完成。

0B8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1M	TC1ENB	TC1rate2	TC1rate1	TC1rate0	TC1CKS1	TC1CKS0	-	PWM1OUT
读/写	R/W	R/W	R/W	R/W	R/W	R/W	-	R/W
复位后	0	0	0	0	0	0	-	0

Bit 0 **PWM1OUT**：PWM 输出控制位。
0 = 禁止 PWM 输出，P5.2 为普通 I/O 引脚；
1 = 允许 PWM 输出，P5.2 输出 PWM 信号。

Bit 2 **TC1CKS 0**：TC1 时钟源选择位。
0 = Fcpu；
1 = Fhosc。

Bit 3 **TC1CKS1**：TC1 时钟源选择位。
0 = 内部时钟（Fcpu 或者 Fhosc，由 TC1CKS0 控制）；
1 = 外部时钟信号（P0.1/INT1），使能事件计数器功能，TC1Rate[2:0]处于无效状态。

Bit [6:4] **TC1RATE[2:0]**：TC1 分频选择位。
TC1CKS0=0 -> 000 = Fcpu/128；001 = Fcpu/64；010 = Fcpu/32；011 = Fcpu/16；100 = Fcpu/8；
101 = Fcpu/4；110 = Fcpu/2；111 = Fcpu/1。
TC1CKS0=1 -> 000 = Fhosc/128；001 = Fhosc/64；010 = Fhosc/32；011 = Fhosc/16；100 = Fhosc/8；
101 = Fhosc/4；110 = Fhosc/2；111 = Fhosc/1。

Bit 7 **TC1ENB**：TC1 启动控制位。
0 = 关闭 TC1 定时器；
1 = 开启 TC1 定时器。

8.4.4 TC1C计数寄存器

8 位计数器 TC1C 溢出时，TC1IRQ 置 1 并由程序清零，用来控制 TC1 的中断间隔时间。首先须写入正确的值到 TC1C 和 TC1R 寄存器，并使能 TC1 定时器以保证第一个周期正确。TC1 溢出后，TC1R 的值自动装入 TC1C。

0B9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1C	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

TC1C 初始值的计算公式如下：

TC1C 初始值 = 256 -（TC1 中断间隔时间 * TC1 时钟 Rate）

8.4.5 TC1R自动重装寄存器

TC1 内置自动重装功能，TC1R 寄存器存储重装数据。当 TC1C 溢出时，TC1R 的值自动装入 TC1C 中。TC1 定时器工作在计时模式时，要通过修改 TC1R 寄存器来修改 TC1 的间隔时间，而不是通过修改 TC1C 寄存器。在 TC1 定时器溢出后，新的 TC1C 值会被更新，TC1R 会将新的值装载到 TC1C 寄存器中。但在初次设置 TC1M 时，必须要在开启 TC1 定时器前把 TC1C 以及 TC1R 设置成相同的值。

TC1 为双重缓存器结构。若程序对 TC1R 进行了修改，那么修改后的 TC1R 值首先被暂存在 TC1R 的第一个缓存器中，TC1 溢出后，TC1R 的新值就会被存入 TC1R 缓存器中，从而避免 TC1 中断时间出错以及 PWM 误动作。

0BAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1R	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

TC1R 初始值计算公式如下：

TC1R 初始值 = 256 -（TC1 中断间隔时间 * TC1 时钟 Rate）

- **例：计算 TC1C 和 TC1R 的值，TC1 的间隔时间为 10ms，时钟源来自 Fcpu= 16MHz/16=1MHz，TC1RATE=000（Fcpu/128）。**

TC1 间隔时间为 10ms，TC1 时钟 Rate=16MHz/16/128

TC1C/TC1R 初始值 = 256 -（TC1 中断间隔时间 * 输入时钟）
= 256 -（10ms * 16MHz / 16 / 128）
= 256 -（10-2 * 16 * 106 / 16 / 128）
= B2H

8.4.6 TC1D PWM占空比寄存器

TC1D 寄存器用来控制 PWM 的占空比。PWM 模式下，TC1R 控制 PWM 的周期，TC1D 控制 PWM 的占空比。TC1C=TC1D 时，PWM 切换为低电平。这样在应用中易于设置 TC1D 以选择合适的 PWM 占空比。

0BBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1D	TC1D7	TC1D6	TC1D5	TC1D4	TC1D3	TC1D2	TC1D1	TC1D0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

TC1D 初始值的计算方法如下：

TC1D 初始值 = TC1R +（PWM 脉冲高电平宽度周期 / TC1 时钟 rate）

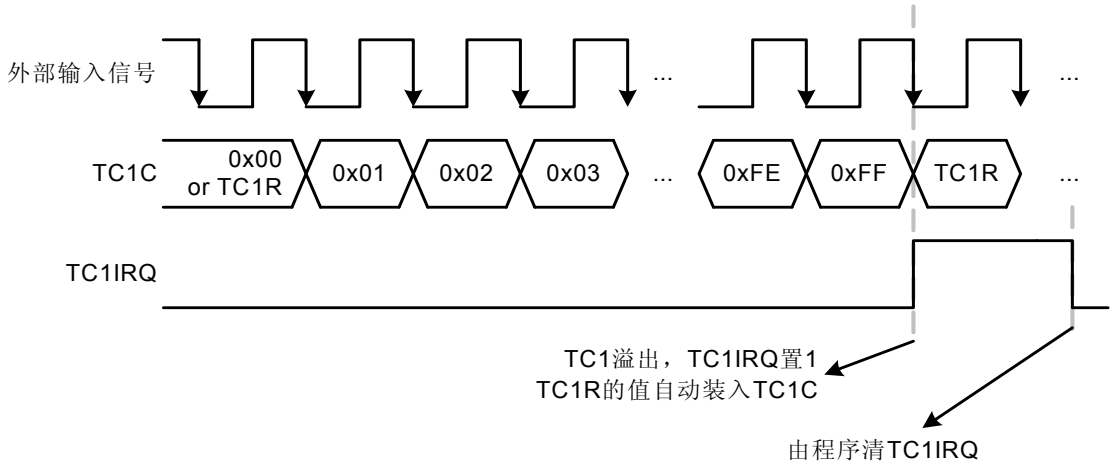
- **例：计算 TC1D 的值。1/3 占空比 PWM，TC1 时钟源 Fcpu=16MHz/16=1MHz，TC1RATE=000（Fcpu/128）。**

TC1R = B2H，TC1 间隔时间=10ms，PWM 周期频率为 100Hz，1/3 占空比条件下，PWM 高电平的宽度值约为 3.33ms。

TC1D 初始值 = B2H +（PWM 脉冲高电平宽度值/TC0 时钟 Rate）
= B2H + (3.33ms * 16MHz / 16 / 128)
= B2H + 1AH
= CCH

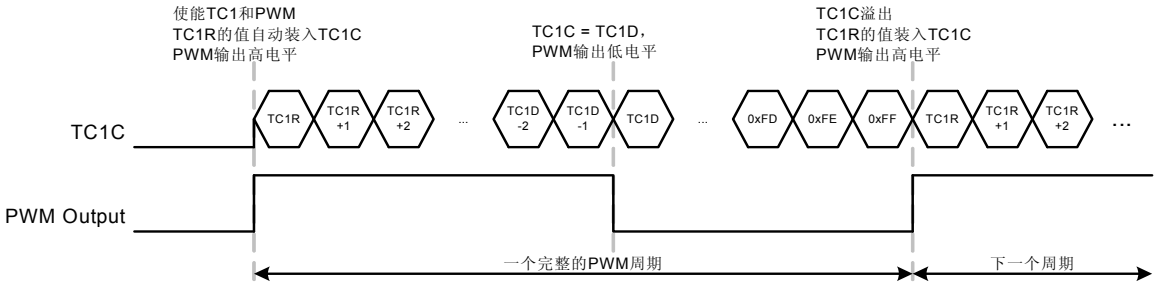
8.4.7 TC1 事件计数器

TC1 作为外部事件计数器时，其时钟源由外部输入引脚（P0.1）提供。当 TC1CKS1=1 时，TC1 的时钟源由外部输入引脚（P0.1）提供，下降沿触发。TC1C 溢出（从 FFH 到 00H）时， TC1 触发事件计数器溢出。使能外部事件计数功能，同时禁止外部输入引脚的唤醒功能以避免外部事件的触发信号将系统唤醒而耗电。此时，P0.1 的外部中断功能也被禁止，即 P01IRQ=0。外部事件计数器通常用来测量外部连续信号的比率，如连续的脉冲信号，R/C 振荡信号等，外部信号的相位与 MCU 时钟的相位并不同步，通过 TC1 事件计数器的测量和计算以达到不同的应用。

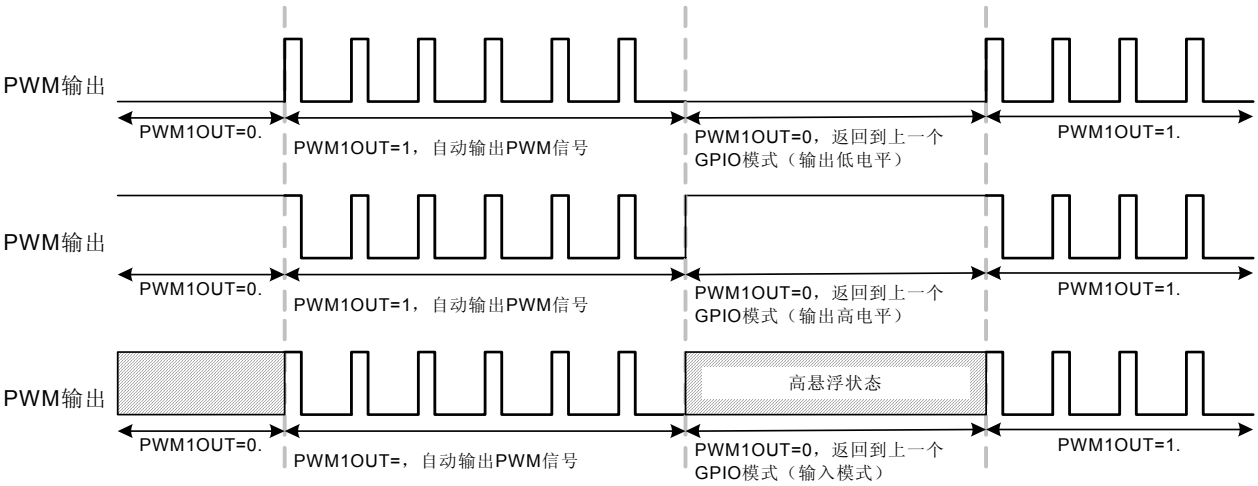


8.4.8 脉冲宽度调制（PWM）

可编程控制占空比/周期的 PWM 可以提供不同的 PWM 信号。使能 TC1 定时器且 PWM1OUT=1 时，由 PWM 输出引脚（P5.2）输出 PWM 信号。PWM 首先输出高电平，然后输出低电平。TC1R 寄存器控制 PWM 的周期，TC1D 控制 PWM 的占空比（脉冲高电平的长度）。开启 TC1 定时器且定时器溢出后，TC1R 重装载 TC1C 的初始值。当 TC1C=TC1D 时，PWM 输出低电平；TC1 溢出时（TC1C 的值从 0FFH 到 00H），整个 PWM 周期完成，并进入下一个周期。TC1 溢出时，TC1R 的值自动装入 TC1C，PWM 的一个周期完成，以保持 PWM 的连贯性。在 PWM 输出的过程由程序更改 PWM 的占空比，则在下一个周期开始输出新的占空比的 PWM 信号。



PWM 的分辨率由 TC1R 决定，而 TC1R 的范围值为 00H~0FFH。当 TC1R=00H 时，PWM 的分辨率为 1/256；TC1R=80H 时，PWM 的分辨率为 1/128。TC1D 控制 PWM 高电平脉冲的宽度，即 PWM 的占空比。当 TC1C=TC1D 时，PWM 输出低电平，TC1D 的值必须大于 TC1R 的值，否则 PWM 的信号保持低电平状态。PWM 输出过程中，TC1 溢出时，TC1IRQ 有效，TC1IEN=1 时，则使能 TC1 中断。但强烈建议小心同时使用 PWM 和 TC1 定时器功能，保证两种功能都能正常工作。PWM 的输出引脚与 GPIO 共用，PWM1OUT=1 时，自动输出 PWM 信号；PWM1OUT=0，即禁止 PWM 时，该引脚自动返回到上一个 GPIO 模式。这样有利于处理 ON/OFF 操作的载波信号，而不控制 TC1ENB 位。



8.4.9 TC1 操作举例

- **TC1 定时器**

```
; 复位 TC1。
        CLR         TC1M            ; 清 TC1M。

; 设置 TC1 时钟源和 TC1Rate。
        MOV         A, #0nnn0n00b
        B0MOV       TC1M, A

; 设置 TC1C 和 TC1R 获得 TC1 的间隔时间。
        MOV         A, #value       ;
        B0MOV       TC1C, A
        B0MOV       TC1R, A

; 清 TC1IRQ。
        B0BCLR      FTC1IRQ

; 使能 TC1 定时器和中断功能。
        B0BSET      FTC1IEN         ; 使能 TC1 中断。
        B0BSET      FTC1ENB         ; 使能 TC1 定时器。
```

- **TC1 事件计数器**

```
; 复位 TC1。
        CLR         TC1M            ; 清 TC1M。

; 使能 TC1 事件计数器。
        B0BSET      FTC1CKS1        ; 设置 TC1 的时钟源由外部输入引脚（P0.1）提供。

; 设置 TC1C 和 TC1R 寄存器获得 TC1 的间隔时间。
        MOV         A, #value       ; TC1C 必须和 TC1R 相等。
        B0MOV       TC1C, A
        B0MOV       TC1R, A

; 清 TC1IRQ。
        B0BCLR      FTC1IRQ

; 使能 TC1 定时器和中断功能。
        B0BSET      FTC1IEN         ; 使能 TC1 中断。
        B0BSET      FTC1ENB         ; 使能 TC1 定时器。
```

- **TC1 PWM**

```
; 复位 TC1。
        CLR         TC1M            ; 清 TC1M。

; 设置 TC1 时钟源和 TC1Rate。
        MOV         A, #0nnn0n00b
        B0MOV       TC1M, A

; 设置 TC1C 和 TC1R 寄存器获得 PWM 周期。
        MOV         A, #value1      ; TC1C 必须和 TC1R 相等。
        B0MOV       TC1C, A
        B0MOV       TC1R, A

; 设置 TC1D 寄存器获得 PWM 占空比。
        MOV         A, #value2      ; TC1D 的值必须大于 TC1R 的值。
        B0MOV       TC1D, A

; 使能 PWM 和 TC1 定时器。
        B0BSET      FTC1ENB         ; 使能 TC1 定时器。
        B0BSET      FPWM1OUT        ; 使能 PWM。
```

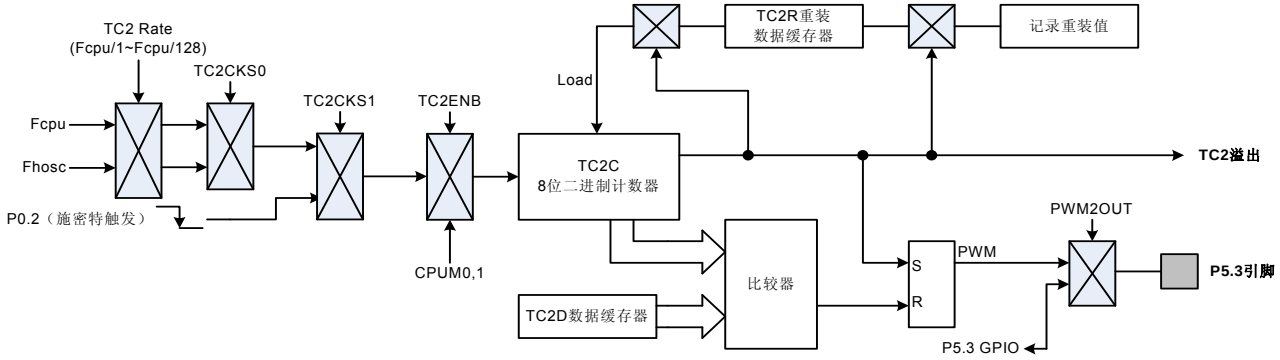
8.5 8 位定时/计数器TC2

8.5.1 概述

8 位二进制定时器 TC2 具有基本定时器、事件计数器和 PWM 功能。基本定时器功能可以支持中断请求标志的显示（TC2IRQ）和中断操作（中断向量）。由 TC2M、TC2C、TC2R 寄存器控制 TC2 的中断间隔时间。事件计数器可以将 TC2 时钟源由系统时钟（Fcpu/Fhosc）更改为外部时钟信号（如连续的脉冲、R/C 振荡信号等）。TC2 作为计数器时记录外部时钟数目以进行测量应用。TC2 还内置周期/占空比可编程控制的 PWM 功能，PWM 的周期和分辨率由 TC2 时钟周期、TC2R 和 TC2D 寄存器控制，故具有良好性能的 PWM 可以处理 IR 载波信号，马达控制和光度调节等。

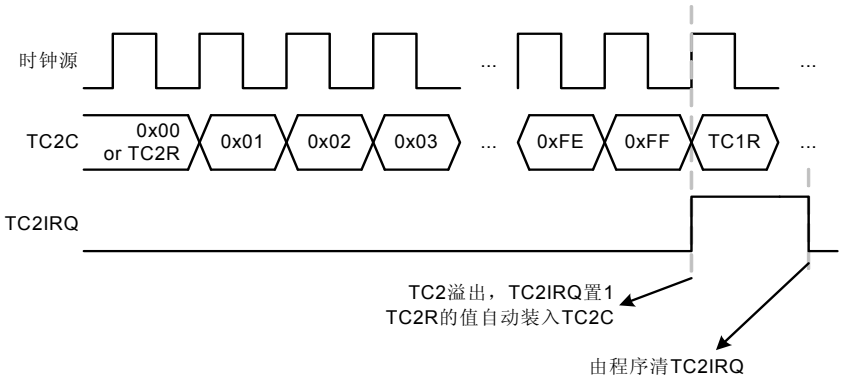
TC2 的主要用途如下：

- **8 位可编程定时器：**根据选择的时钟信号，产生周期性中断；
- **中断功能：**TC2 定时器支持中断，当 TC2 溢出时，TC2IRQ 置 1，系统执行中断；
- **外部事件计数器：**对外部事件计数；
- **可编程控制占空比/周期的 PWM 输出：**由 TC2R 和 TC2D 寄存器控制占空比/周期；
- **绿色模式功能：**绿色模式下，TC2 正常工作，但无唤醒功能。



8.5.2 TC2 操作

TC2 定时器由 TC2ENB 控制。当 TC2ENB=0 时，TC2 停止工作；当 TC2ENB=1 时，TC2 开始计数。使能 TC2 之前，先要设定好 TC2 的功能模式，如基本定时器、TC2 中断等。TC2C 溢出（从 0FFH 到 00H）时，TC2IRQ 置 1 以显示溢出状态并由程序清零。在不同的功能模式下，TC2C 不同的值对应不同的操作，若改变 TC2C 的值影响到操作，会导致功能出错。TC2 内置双重缓存器以避免此种状况的发生。在 TC2C 计数的过程中不断的刷新 TC2C，保证将最新的值存入 TC2R（重装缓存器）中，当 TC2 溢出后，新的 TC2R 值将自动装载到 TC2C。进入下一个周期后，TC2 按新的配置工作。定时/计数器模式时，使能 TC2 时，自动使能自动重装功能。如果使能 TC2 中断功能（TC2IEN=1），在 TC2 溢出时系统执行中断服务程序，在中断时必须由程序清 TC2IRQ。TC2 可以在普通模式、低速模式和绿色模式下工作。但在绿色模式下，TC2 虽继续工作，但不能唤醒系统。



TC2 根据不同的时钟源选择不同的应用模式，TC2 的时钟源由 Fcpu（指令周期）、Fhosc（高速振荡时钟）和外部引脚输入（P0.2）提供，由 TC2CKS[1:0]控制。TC2CKS0 选择时钟源来自 Fcpu 或者 Fhosc，当 TC2CKS0=0 时，TC2 时钟源来自 Fcpu，可以由 TC2Rate[2:0]选择不同的分频。当 TC2CKS0=1 时，TC2 时钟源来自 Fhosc，可以有 TC2Rate[2:0]选择不同的分频。TC2CKS1 决定时钟源由外部引脚输入或者由 TC2CKS0 控制，TC2CKS1=0 时，TC2 的时钟源由 TC2CKS0 控制，TC2CKS1=1 时，TC2 时钟源由外部输入引脚提供，此时使能外部事件计数功能。TC2CKS1=1 时，TC2Rate[2:0]处于无效状态。TC2 为 8 位定时器（256 阶），每经过一个 TC2 时钟周期计一个点。

TC2CKS0	TC2rate[2:0]	TC2 时钟	TC2 间隔时间			
			Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
			max. (ms)	Unit (us)	max. (ms)	Unit (us)
0	000b	Fcpu/128	8.192	32	32.768	128
0	001b	Fcpu/64	4.096	16	16.384	64
0	010b	Fcpu/32	2.048	8	8.192	32
0	011b	Fcpu/16	1.024	4	4.096	16
0	100b	Fcpu/8	0.512	2	2.048	8
0	101b	Fcpu/4	0.256	1	1.024	4
0	110b	Fcpu/2	0.128	0.5	0.512	2
0	111b	Fcpu/1	0.064	0.25	0.256	1
1	000b	Fhosc/128	2.048	8	8.192	32
1	001b	Fhosc/64	1.024	4	4.096	16
1	010b	Fhosc/32	0.512	2	2.048	8
1	011b	Fhosc/16	0.256	1	1.024	4
1	100b	Fhosc/8	0.128	0.5	0.512	2
1	101b	Fhosc/4	0.064	0.25	0.256	1
1	110b	Fhosc/2	0.032	0.125	0.128	0.5
1	111b	Fhosc/1	0.016	0.0625	0.064	0.25

8.5.3 TC2M模式寄存器

模式寄存器 TC2M 控制 TC2 的工作模式，包括 TC2 分频、时钟源和 PWM 功能等。 这些设置必须在使能 TC2 定时器之前完成。

0BCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC2M	TC2ENB	TC2rate2	TC2rate1	TC2rate0	TC2CKS1	TC2CKS0	-	PWM2OUT
读/写	R/W	R/W	R/W	R/W	R/W	R/W	-	R/W
复位后	0	0	0	0	0	0	-	0

Bit 0 **PWM2OUT**：PWM 输出控制位。
0 = 禁止 PWM 输出，P5.3 为普通 I/O 引脚；
1 = 允许 PWM 输出，P5.3 输出 PWM 信号。

Bit 2 **TC2CKS 0**：TC2 时钟源选择位。。
0 = Fcpu；
1 = Fhosc。

Bit 3 **TC2CKS1**：TC2 时钟源选择位。
0 = 内部时钟（Fcpu 或者 Fhosc，由 TC2CKS0 控制）；
1 = 外部时钟信号（P0.2/INT2），使能事件计数器功能，TC2Rate[2:0]处于无效状态。

Bit [6:4] **TC2RATE[2:0]**：TC2 分频选择位。
TC2CKS0=0 -> 000 = Fcpu/128；001 = Fcpu/64；010 = Fcpu/32；011 = Fcpu/16；100 = Fcpu/8；
101 = Fcpu/4；110 = Fcpu/2；111 = Fcpu/1。
TC2CKS0=1 -> 000 = Fhosc/128；001 = Fhosc/64；010 = Fhosc/32；011 = Fhosc/16；100 = Fhosc/8；
101 = Fhosc/4；110 = Fhosc/2；111 = Fhosc/1。

Bit 7 **TC2ENB**：TC2 启动控制位。
0 = 关闭 TC2 定时器；
1 = 开启 TC2 定时器。

8.5.4 TC2C计数寄存器

8 位计数器 TC2C 溢出时，TC2IRQ 置 1 并由程序清零，用来控制 TC2 的中断间隔时间。首先须写入正确的值到 TC2C 和 TC2R 寄存器，并使能 TC2 定时器以保证第一个周期正确。TC2 溢出后，TC2R 的值自动装入 TC2C。

0BDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC2C	TC2C7	TC2C6	TC2C5	TC2C4	TC2C3	TC2C2	TC2C1	TC2C0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

TC2C 初始值的计算公式如下：

TC2C 初始值 = 256 -（TC2 中断间隔时间 * TC2 时钟 Rate）

8.5.5 TC2R自动重装寄存器

TC2 内置自动重装功能，TC2R 寄存器存储重装数据。当 TC2C 溢出时，TC2R 的值自动装入 TC2C 中。TC2 定时器工作在计时模式时，要通过修改 TC2R 寄存器来修改 TC2 的间隔时间，而不是通过修改 TC2C 寄存器。在 TC2 定时器溢出后，新的 TC2C 值会被更新，TC2R 会将新的值装载到 TC2C 寄存器中。但在初次设置 TC2M 时，必须要在开启 TC2 定时器前把 TC2C 以及 TC2R 设置成相同的值。

TC2 为双重缓存器结构。若程序对 TC2R 进行了修改，那么修改后的 TC2R 值首先被暂存在 TC2R 的第一个缓存器中，TC2 溢出后，TC2R 的新值就会被存入 TC2R 缓存器中，从而避免 TC2 中断时间出错以及 PWM 误动作。

0BEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC2R	TC2R7	TC2R6	TC2R5	TC2R4	TC2R3	TC2R2	TC2R1	TC2R0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

TC2R 初始值计算公式如下：

TC2R 初始值 = 256 -（TC2 中断间隔时间 * TC2 时钟 Rate）

➢ **例：计算 TC2C 和 TC2R 的值，TC2 的间隔时间为 10ms，时钟源来自 Fcpu= 16MHz/16=1MHz，TC2RATE=000（Fcpu/128）。**

TC2 间隔时间为 10ms，TC2 时钟 Rate=16MHz/16/128

TC2C/TC2R 初始值 = 256 -（TC1 中断间隔时间 * 输入时钟）
= 256 -（10ms * 16MHz / 16 / 128）
= 256 -（10-2 * 16 * 106 / 16 / 128）
= B2H

8.5.6 TC2D PWM占空比寄存器

TC2D 寄存器用来控制 PWM 的占空比。PWM 模式下，TC2R 控制 PWM 的周期，TC2D 控制 PWM 的占空比。TC2C=TC2D 时，PWM 切换为低电平。这样在应用中易于设置 TC2D 以选择合适的 PWM 占空比。

0BFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC2D	TC2D7	TC2D6	TC2D5	TC2D4	TC2D3	TC2D2	TC2D1	TC2D0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

TC2D 初始值的计算方法如下：

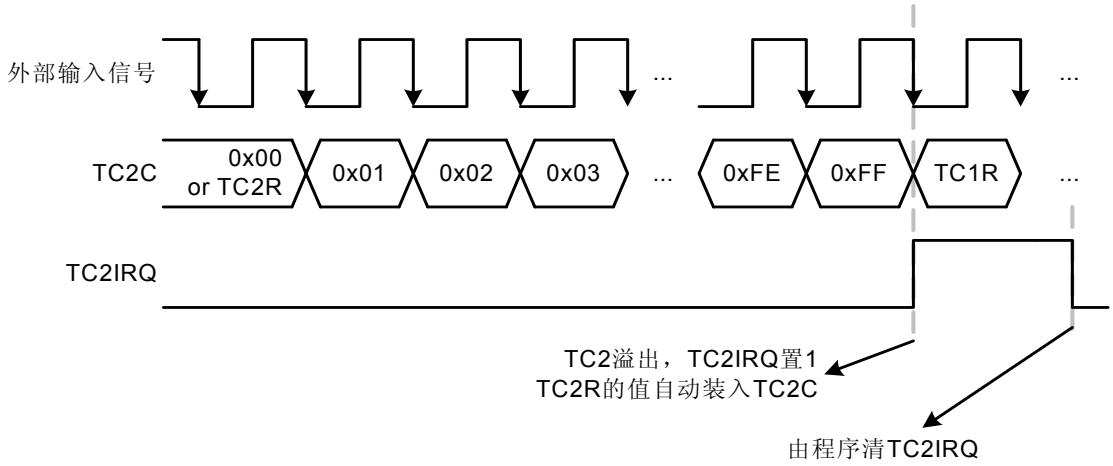
TC2D 初始值 = TC2R +（PWM 脉冲高电平宽度周期 / TC2 时钟 rate）

➢ **例：计算 TC2D 的值。1/3 占空比 PWM，TC2 时钟源 Fcpu=16MHz/16=1MHz，TC2RATE=000（Fcpu/128）。TC2R = B2H，TC2 间隔时间=10ms，PWM 周期频率为 100Hz，1/3 占空比条件下，PWM 高电平的宽度值约为 3.33ms。**

TC2D 初始值 = B2H +（PWM 脉冲高电平宽度值/TC0 时钟 Rate）
= B2H + (3.33ms * 16MHz / 16 / 128)
= B2H + 1AH
= CCH

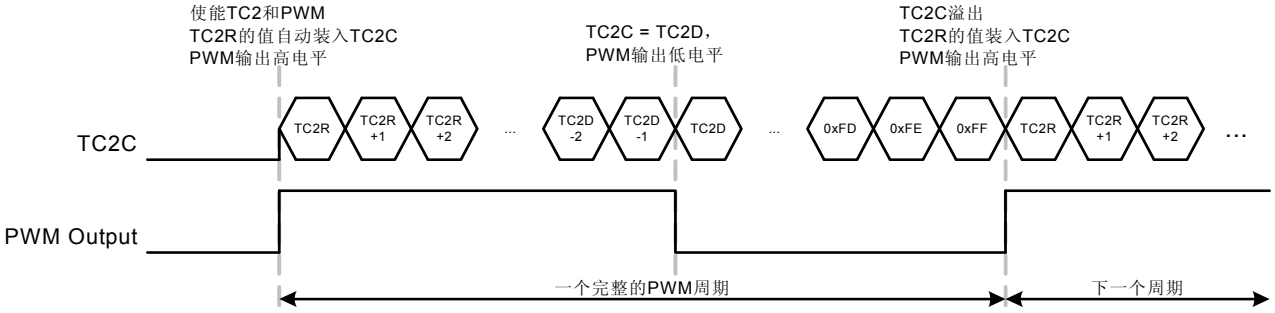
8.5.7 TC2 事件计数器

TC2 作为外部事件计数器时，其时钟源由外部输入引脚（P0.2）提供。当 TC2CKS1=1 时，TC2 的时钟源由外部输入引脚（P0.2）提供，下降沿触发。TC2C 溢出（从 FFH 到 00H）时， TC2 触发事件计数器溢出。使能外部事件计数功能，同时禁止外部输入引脚的唤醒功能以避免外部事件的触发信号将系统唤醒而耗电。此时，P0.2 的外部中断功能也被禁止，即 P02IRQ=0。外部事件计数器通常用来测量外部连续信号的比率，如连续的脉冲信号，R/C 振荡信号等，外部信号的相位与 MCU 时钟的相位并不同步，通过 TC2 事件计数器的测量和计算以达到不同的应用。

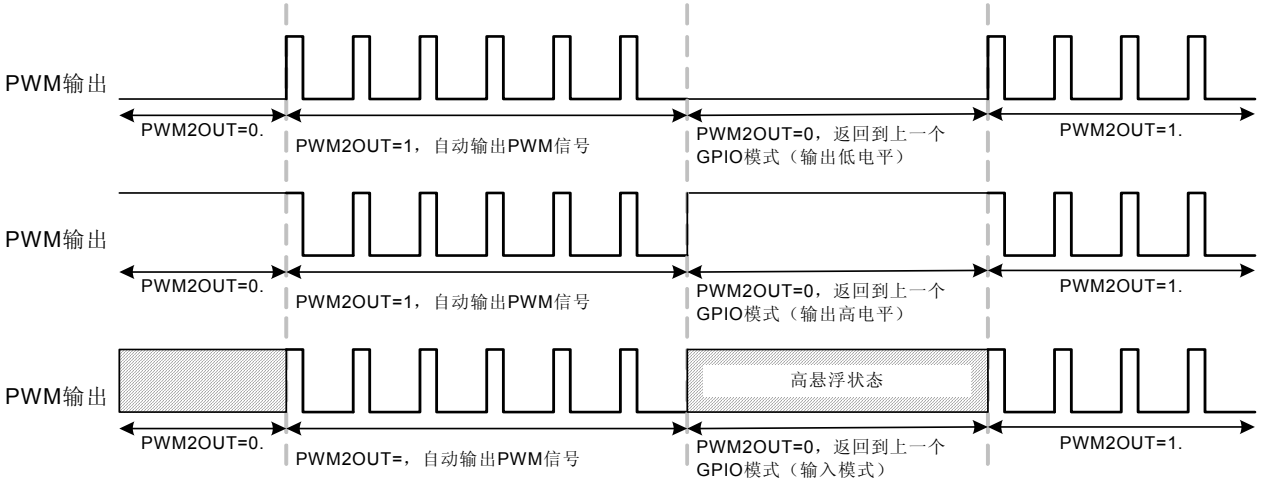


8.5.8 脉冲宽度调制（PWM）

可编程控制占空比/周期的 PWM 可以提供不同的 PWM 信号。使能 TC2 定时器且 PWM2OUT=1 时，由 PWM 输出引脚（P5.3）输出 PWM 信号。PWM 首先输出高电平，然后输出低电平。TC2R 寄存器控制 PWM 的周期，TC2D 控制 PWM 的占空比（脉冲高电平的长度）。开启 TC2 定时器且定时器溢出后，TC2R 重装载 TC2C 的初始值。当 TC2C=TC2D 时，PWM 输出低电平；TC2 溢出时（TC2C 的值从 0FFH 到 00H），整个 PWM 周期完成，并进入下一个周期。TC2 溢出时，TC2R 的值自动装入 TC2C，PWM 的一个周期完成，以保持 PWM 的连贯性。在 PWM 输出的过程由程序更改 PWM 的占空比，则在下一个周期开始输出新的占空比的 PWM 信号。



PWM 的分辨率由 TC2R 决定，而 TC2R 的范围值为 00H~0FFH。当 TC2R=00H 时，PWM 的分辨率为 1/256；TC2R=80H 时，PWM 的分辨率为 1/128。TC2D 控制 PWM 高电平脉冲的宽度，即 PWM 的占空比。当 TC2C=TC2D 时，PWM 输出低电平，TC2D 的值必须大于 TC2R 的值，否则 PWM 的信号保持低电平状态。PWM 输出过程中，TC2 溢出时，TC2IRQ 有效，TC2IEN=1 时，则使能 TC2 中断。但强烈建议小心同时使用 PWM 和 TC2 定时器功能，保证两种功能都能正常工作。PWM 的输出引脚与 GPIO 共用，PWM2OUT=1 时，自动输出 PWM 信号；PWM2OUT=0，即禁止 PWM 时，该引脚自动返回到上一个 GPIO 模式。这样有利于处理 ON/OFF 操作的载波信号，而不控制 TC2ENB 位。



8.5.9 TC2 操作举例

- **TC2 定时器**

```
; 复位 TC2。
        CLR         TC2M            ; 清 TC2M。

; 设置 TC2 时钟源和 TC2Rate。
        MOV         A, #0nnn0n00b
        B0MOV       TC2M, A

; 设置 TC2C 和 TC2R 获得 TC2 的间隔时间。
        MOV         A, #value
        B0MOV       TC2C, A
        B0MOV       TC2R, A

; 清 TC2IRQ。
        B0BCLR      FTC2IRQ

; 使能 TC2 定时器和中断功能。
        B0BSET      FTC2IEN         ; 使能 TC2 中断。
        B0BSET      FTC2ENB         ; 使能 TC2 定时器。
```

- **TC2 事件计数器**

```
; 复位 TC2。
        CLR         TC2M            ; 清 TC2M。

; 使能 TC2 事件计数器。
        B0BSET      FTC2CKS1        ; 设置 TC2 的时钟源由外部输入引脚（P0.2）提供。

; 设置 TC2C 和 TC2R 寄存器获得 TC2 的间隔时间。
        MOV         A, #value       ; TC2C 必须和 TC2R 相等。
        B0MOV       TC2C, A
        B0MOV       TC2R, A

; 清 TC2IRQ。
        B0BCLR      FTC2IRQ

; 使能 TC2 定时器和中断功能。
        B0BSET      FTC2IEN         ; 使能 TC2 中断。
        B0BSET      FTC2ENB         ; 使能 TC2 定时器。
```

- **TC2 PWM**

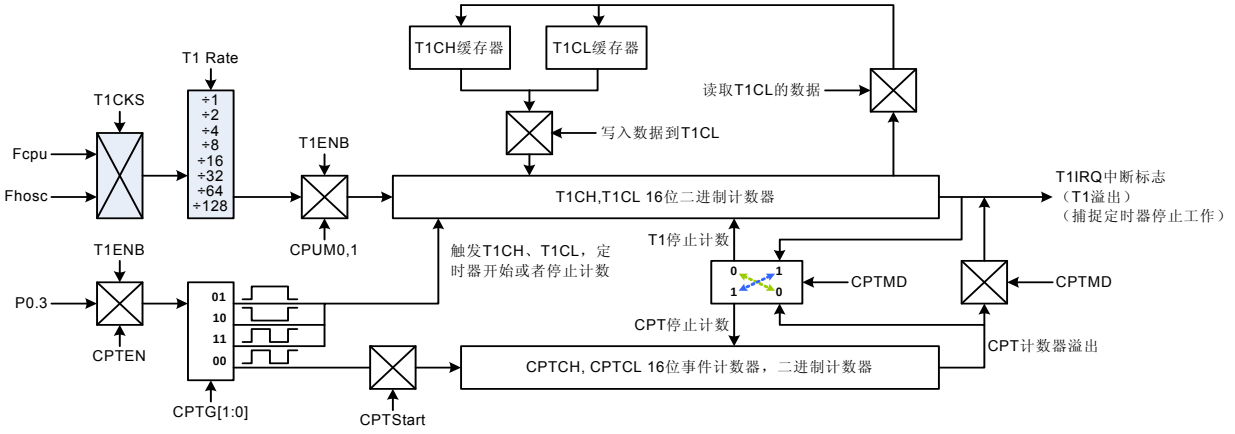
```
; 复位 TC2。
        CLR         TC2M            ; 清 TC2M。

; 设置 TC2 时钟源和 TC2Rate。
        MOV         A, #0nnn0n00b
        B0MOV       TC2M, A

; 设置 TC2C 和 TC2R 寄存器获得 PWM 周期。
        MOV         A, #value1
        B0MOV       TC2C, A
        B0MOV       TC2R, A

; 设置 TC2D 寄存器获得 PWM 占空比。
        MOV         A, #value2      ; TC2D 的值必须大于 TC2R 的值。
        B0MOV       TC2D, A

; 使能 PWM 和 TC2 定时器。
        B0BSET      FTC2ENB         ; 使能 TC2 定时器。
        B0BSET      FPWM2OUT        ; 使能 PWM。
```

8.6 16 位定时器T1

8.6.1 概述

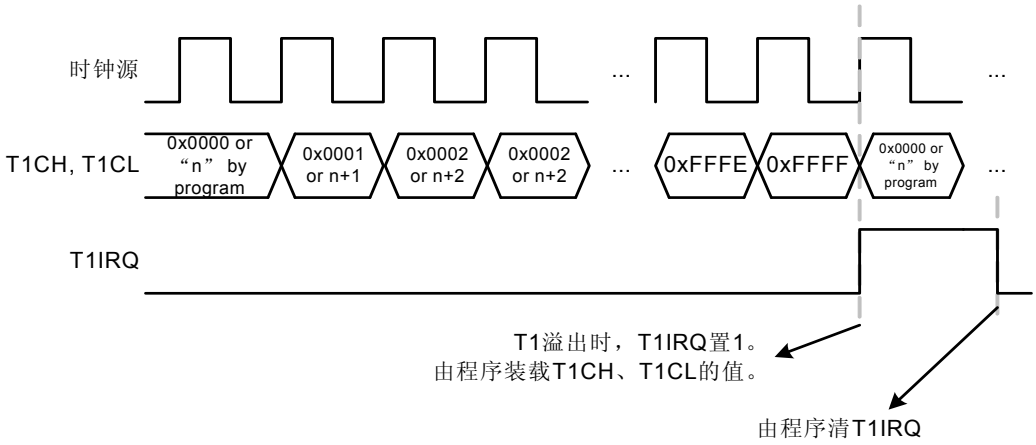
16 位二进制定时器 T1 具有基本定时器和捕捉定时器功能：基本定时器支持中断请求标志的显示（T1IRQ）和中断操作（中断向量），中断间隔时间由 T1M、T1CH/T1CL 计数寄存器控制；捕捉定时器支持脉冲高电平宽度测量、低电平宽度测量和周期测量（由 P0.3 提供）。T1 作为定时器使用时用来记录外部信号时间参数以进行测量应用。T1 的主要功能如下所示：

- **16 位可编程的计数定时器：**根据选择的时钟频率周期性的产生中断请求。
- **16 位测量：**测量输入信号的脉冲宽度和周期，由根据决定捕捉定时器的分辨率的 T1 时钟决定。T1 内置可编程的触发边沿选择装置以决定开始-停止触发事件。
- **16 位捕捉定时器：** 16 位事件计数器检测事件源以积累捕捉时间功能。
- **中断功能：**T1 定时器和捕捉定时器支持中断功能。当 T1 溢出，T1IRQ 有效，程序计数器跳到中断向量地址执行中断。
- **绿色模式功能：**T1 定时器在绿色模式下正常工作，但不能将系统从绿色模式下唤醒。当发生 IRQ 触发条件时（如定时器溢出等），IRQ 有效。



8.6.2 T1 操作

T1 定时器由 T1ENB 控制。当 T1ENB=0 时，T1 停止工作；当 T1ENB=1 时，T1 开始计数。使能 T1 之前，先设置 T1 的功能模式，如基本定时器、中断功能等。16 位计数器 T1（T1CH、T1CL）溢出（从 0FFFFH 到 0000H）时，T1IRQ 置 1 显示溢出状态并由程序清零，并由程序装载 T1CH、T1CL 的值以确定合适的中断间隔时间。若使能 T1 中断（T1IEN=1），T1 溢出时，程序计数器跳到中断向量地址开始执行中断服务程序，在中断下必须由程序清 T1IRQ。T1 可以在普通模式、低速模式和绿色模式下工作。



T1 根据不同的时钟源选择不同的应用模式，T1 的时钟源由 Fcpu（指令周期）和 Fhosc（高速振荡时钟）提供，由 T1CKS 控制。T1CKS 选择时钟源来自 Fcpu 或者 Fhosc，当 T1CKS=0 时，T1 时钟源来自 Fcpu，可以由 T1Rate[2:0] 选择不同的分频。当 T1CKS=1 时，T1 时钟源来自 Fhosc，可以由 TC0Rate[2:0]选择不同的分频。

T1CKS	T1rate[2:0]	T1 时钟	T1 间隔时间			
			Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
			max. (ms)	Unit (us)	max. (ms)	Unit (us)
0	000b	Fcpu/128	2097.152	32	8388.608	128
0	001b	Fcpu/64	1048.576	16	4194.304	64
0	010b	Fcpu/32	524.288	8	2097.152	32
0	011b	Fcpu/16	262.144	4	1048.576	16
0	100b	Fcpu/8	131.072	2	524.288	8
0	101b	Fcpu/4	65.536	1	262.144	4
0	110b	Fcpu/2	32.768	0.5	131.072	2
0	111b	Fcpu/1	16.384	0.25	65.536	1
1	000b	Fhosc/128	524.288	8	2097.152	32
1	001b	Fhosc/64	262.144	4	1048.576	16
1	010b	Fhosc/32	131.072	2	524.288	8
1	011b	Fhosc/16	65.536	1	262.144	4
1	100b	Fhosc/8	32.768	0.5	131.072	2
1	101b	Fhosc/4	16.384	0.25	65.536	1
1	110b	Fhosc/2	8.192	0.125	32.768	0.5
1	111b	Fhosc/1	4.096	0.0625	16.384	0.25

8.6.3 T1M模式寄存器

模式寄存器 T1M 设置 T1 的工作模式，包括 T1 前置分频器、时钟源和捕捉参数等，这些设置必须在使能 T1 定时器之前完成。

0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1M	T1ENB	T1rate2	T1rate1	T1rate0	T1CKS			
读/写	R/W	R/W	R/W	R/W	R/W			
复位后	0	0	0	0	0			

Bit 7 **T1ENB**：T1 启动控制位。
0 = 禁止；
1 = 使能。

Bit [6:4] **T1RATE[2:0]**：T1 分频选择位。
T1CKS=0 -> 000 = Fcpu/128；001 = Fcpu/64；010 = Fcpu/32；011 = Fcpu/16；100 = Fcpu/8；
101 = Fcpu/4；110 = Fcpu/2；111 = Fcpu/1。
T1CKS=1 -> 000 = Fhosc/128；001 = Fhosc/64；010 = Fhosc/32；011 = Fhosc/16；100 = Fhosc/8；
101 = Fhosc/4；110 = Fhosc/2；111 = Fhosc/1。

Bit 3 **T1CKS**：T1 时钟源控制位。
0 = Fcpu；
1 = Fhosc。

8.6.4 T1CH, T1CL 16 位计数寄存器

16 位计数器 T1CH、T1CL 溢出时，T1IRQ 置 1 并由程序清零，用来控制 T1 的中断间隔时间。首先须写入正确的值到 T1CH 和 T1CL 寄存器，并使能 T1 定时器以保证第一个周期正确。T1 溢出后，由程序自动装入正确的值到 T1CH 和 T1CL 寄存器。

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1CL	T1CL7	T1CL6	T1CL5	T1CL4	T1CL3	T1CL2	T1CL1	T1CL0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0C2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1CH	T1CH7	T1CH6	T1CH5	T1CH4	T1CH3	T1CH2	T1CH1	T1CH0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

16 位定时计数器 T1 为双重缓存器设计，但核心总线只有 8 位，故处理 16 位数据时须锁存标志，以免瞬间状态影响到 16 位数据而出错。写入数据时，写 T1CL 是锁存控制标志；读取数据时，读 T1CL 是锁存控制标志。故写入数据到 T1 时，要先写入数据到 T1CH，再写入数据到 T1CL，即执行写入数据到 T1CL 时，所有数据都已经写入 16 位计数器 T1 中。反之，读取 T1 的数据时，要先读取 T1CL 的数据，再读取 T1CH 的数据，即执行读取 T1CH 时，T1 中的所有数据都已经被读取完毕。

- **读取 T1 计数缓存器中的数据时，要先读取 T1CL 中的数据，再读取 T1CH 中的数据。**
- **写入数据到 T1 计数缓存器中时，要先写入数据到 T1CH，再写入数据到 T1CL。**

16 位计数器 T1（T1CH，T1CL）初始值的计算公式如下：

T1CH，T1 初始值 = 65536 -（T1 中断间隔时间 * T1 时钟比率 T1Rate）

- **例：通过计算 T1CH 和 T1CL 的值，获得 T1 的中断间隔时间为 500ms，T1 时钟源为 Fcpu =16MHz/16=1MHz，T1RATE = 000（Fcpu/128）。**

T1 中断间隔时间=500ms，T1Rate=16MHz/16/128

T1CH、T1CL 初始值 = 65536 -（T1 中断间隔时间 * 输入时钟）
= 65536 -（500ms * 16MHz / 16 / 128）
= 65536 -（500*10-3 * 16 * 106 / 16 / 128）
= F0BDH（T1CH = F0H，T1CL = BDH）

8.6.5 T1 捕捉定时器

16 位捕捉定时器由 CPTEN 位控制，但必须使能 T1。T1ENB=1，且 CPTEN=1 时，使能捕捉定时器功能。捕捉定时器仅是一个单纯的计数器，并无时钟源来决定间隔时间。捕捉定时器的输入源为 P0.3，CPTG[1:0]控制捕捉定时器的功能。

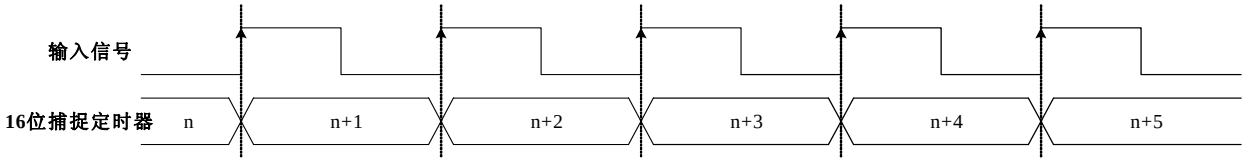
CPTG[1:0] = 00：捕捉定时器功能。
CPTG[1:0] = 01：测量 P0.3 高电平的脉冲宽度。
CPTG[1:0] = 10：测量 P0.3 低电平的脉冲宽度。
CPTG[1:0] = 11：测量 P0.3 的周期。

捕捉定时器的所有功能都必须通过 T1 来实现，捕捉定时器可以测量输入脉冲的高电平宽度，输入脉冲的低电平宽度和输入脉冲的周期以及输入信号（P0.3）的捕捉持续时间，由 CPTG[1:0]控制。CPTStart 位执行捕捉定时器功能，CPTStart= 1 时，捕捉定时器等待合适的触发沿，开始工作；等到第二个合适的触发沿到来时，捕捉定时器停止工作，此时 CPTStart 位被清零，T1IRQ 被置 1。

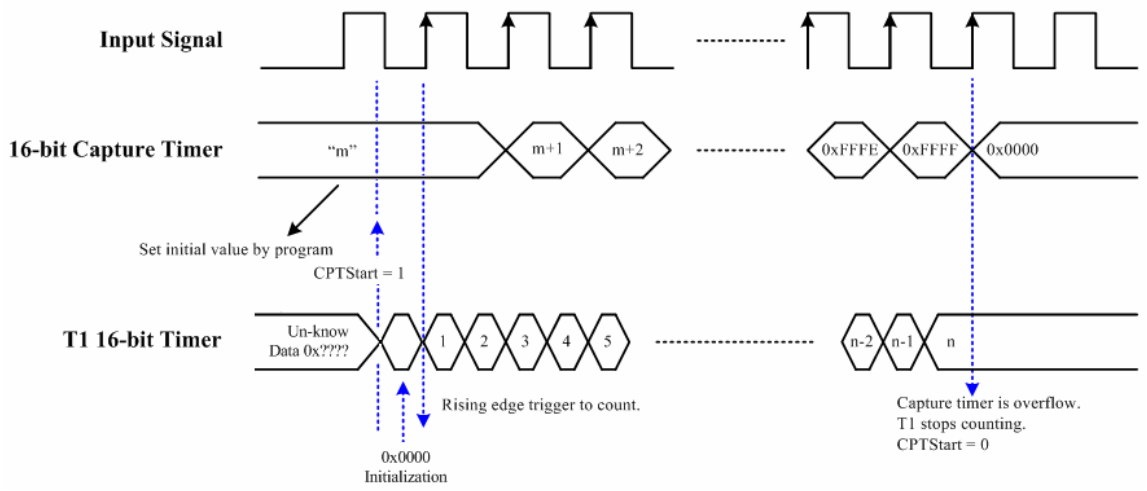
8.6.5.1捕捉定时器

捕捉定时器功能由 CPTG[1:0]控制，CPTG[1:0]=00 时，使能捕捉定时器功能。捕捉定时器用来测量连续信号的周期。捕捉定时器功能包括两种模式对应不同速度的信号，由 CPTMD 控制。设置 CPTStart 为 1 时，捕捉定时器开始工作，首先是 P0.3 输入信号的上升沿触发。在上升沿触发之前，捕捉定时器和 T1 定时器保持理想的状态以等待上升沿的到来。等到第一个合适的触发沿后，捕捉定时器和 T1 定时器开始工作，每次溢出（由 CPTMD 控制）发生时，捕捉定时器和 T1 定时器就停止计数，CPTStart 位被清零，T1IRQ 被置 1。若 T1IEN=1，系统执行 T1 中断服务程序。

- **捕捉定时器的计数触发源为输入信号的上升沿。**

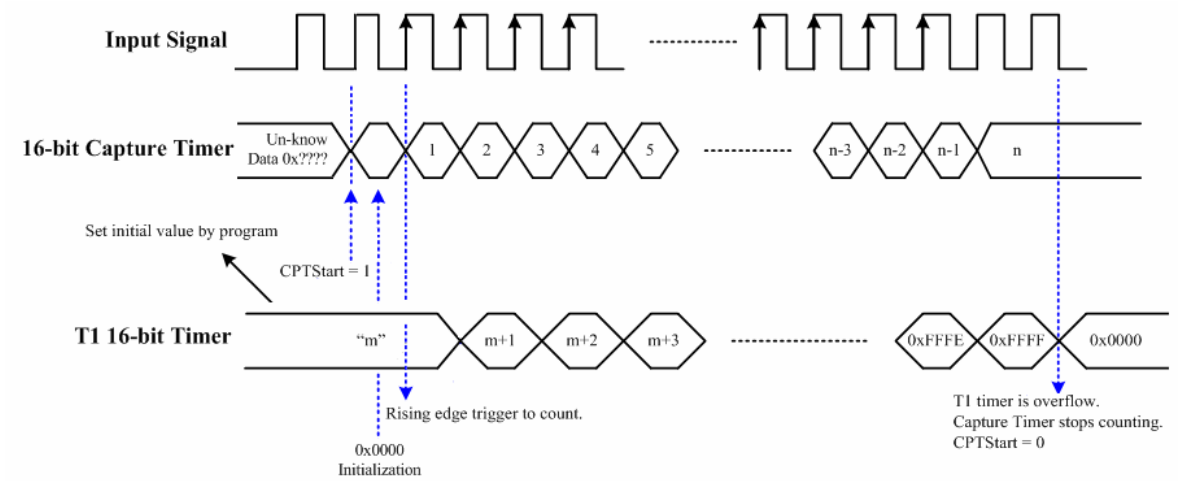


- **CPTMD = 0，低速模式（T1ENB = 1，CPTEN = 1，CPTG[1:0] = 00）。**



输入信号 rate<T1 定时器 rate，利用 T1 定时器测量输入信号的持续时间。设置捕捉定时器的初始值（CPTCH、CPTCL = “m”）并由程序清 T1 定时器（T1CH、T1CL = 0000H），CPTStart 位置 1，捕捉定时器开始工作，输入信号的第一个上升沿到达时，捕捉定时器和 T1 定时器开始计数，捕捉定时器溢出（从 0FFFFH 到 0000H）时，T1 停止计数，CPTStart 位自动清零，T1IRQ 置 1，16 位计数器 T1 的值（T1CH、T1CL=“n”）就是输入信号的持续时间。

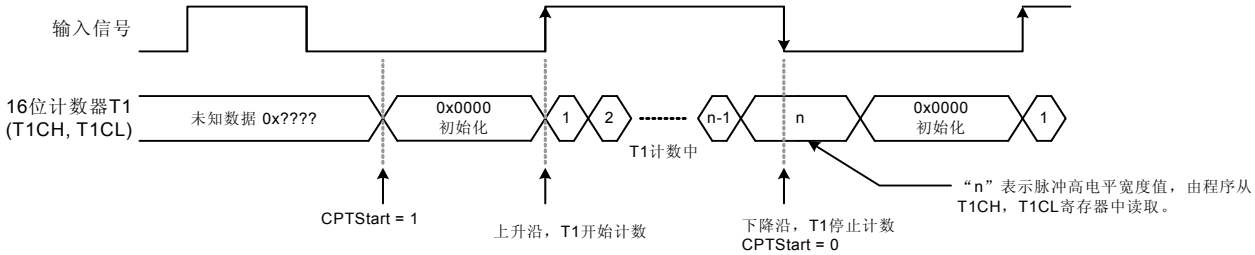
- **CPTMD = 1，高速模式（T1ENB = 1，CPTEN = 1，CPTG[1:0] = 00）。**



输入信号 rate>T1 定时器 rate，利用 T1 定时器设置一个唯一的定时器，来测量输入信号的计数。设置 T1 定时器的初始值（T1CH、T1CL =“m”）并由程序清捕捉定时器（CPTCH、CPTCL = 0000H），CPTStart 位置 1，捕捉定时器开始工作，输入信号的第一个上升沿到达时，捕捉定时器和 T1 定时器开始计数，T1 定时器溢出（从 0FFFFH 到 0000H）时，捕捉停止计数，CPTStart 位自动清零，T1IRQ 置 1，16 位捕捉定时器的值（CPTCH、CPTCL=“n”）就是输入信号的计数。

8.6.5.2测量脉冲高电平宽度

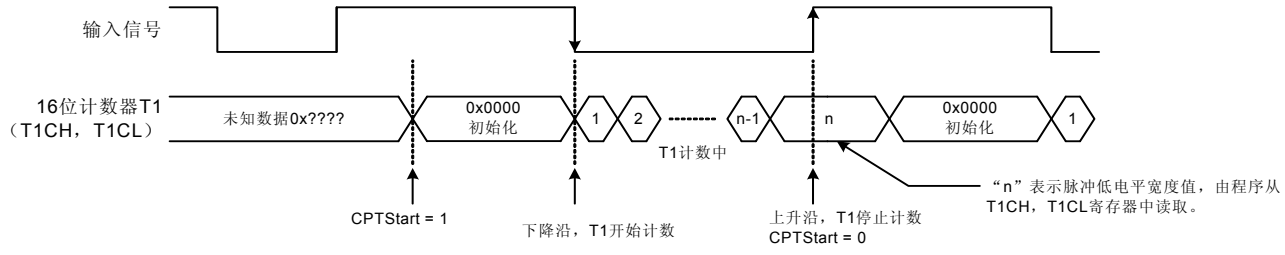
T1ENB = 1，CPTEN = 1，CPTG[1:0] = 01。



测量脉冲高电平宽度：上升沿时，T1 开始计数；下降沿时，T1 停止计数。如果在高电平期间设置 CPTStart 位，捕捉定时器会在下个上升沿时重新测量高电平宽度。在下降沿时 T1 停止计数后，16 位计数器 T1CH 和 T1CL 寄存器中的值则为脉冲高电平的宽度值。

8.6.5.3测量脉冲低电平宽度

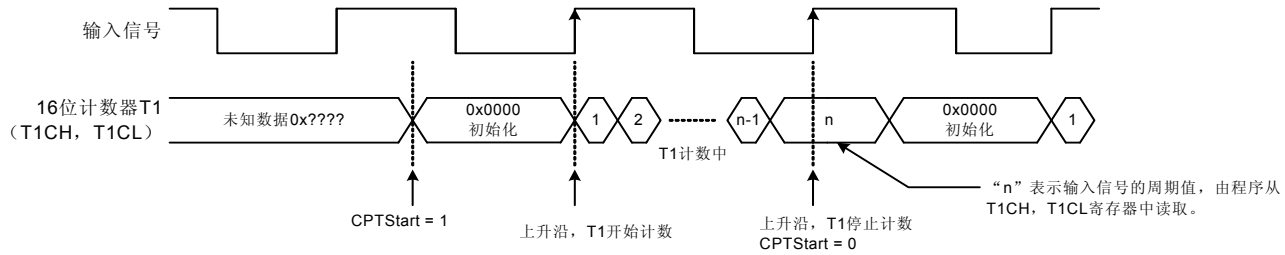
T1ENB = 1，CPTEN = 1，CPTG[1:0] = 10。



测量脉冲低电平宽度：下降沿时，T1 开始计数；上升沿时，T1 停止计数。如果在低电平期间设置 CPTStart 位，捕捉定时器会在下个下降沿时重新测量低电平宽度。在上升沿时 T1 停止计数后，16 位计数器 T1CH 和 T1CL 寄存器中的值则为脉冲低电平的宽度值。

8.6.5.4测量输入信号的周期

T1ENB = 1，CPTEN = 1，CPTG[1:0] = 11。



测量输入信号的周期：上升沿时，T1 开始计数；下一个上升沿时，T1 停止计数。如果在高电平和低电平期间设置 CPTStart 位，捕捉定时器会在下个上升沿时重新开始测量。在上升沿时 T1 停止计数后，16 位计数器 T1CH 和 T1CL 寄存器中的值则为脉冲的周期值。

8.6.6 捕捉定时器控制寄存器

C3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CPTM	CPTEN				CPTMD	CPTStart	CPTG1	CPTG0
读/写	R/W				R/W	R/W	R/W	R/W
复位后	0				0	0	0	0

Bit 7 **CPTEN**：捕捉定时器功能控制位。
0 = 禁止；
1 = 使能。**必须使能 T1EN。**

Bit 3 **CPTMD**：捕捉定时器模式控制位。
0 = CPT 溢出模式；
1 = T1 溢出模式。

Bit 2 **CPTStart**：捕捉定时器计数控制位。
0 = 计数完成；
1 = 计数过程中。

Bit [1:0] **CPTG[1:0]**：捕捉定时器功能控制位。
00 = 捕捉定时器功能；
01 = 测量脉冲高电平宽度；
10 = 测量脉冲低电平宽度；
11 = 测量输入信号周期。

C4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CPTCL	CPTC7	CPTC6	CPTC5	CPTC4	CPTC3	CPTC2	CPTC1	CPTC0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CPTCH	CPTC15	CPTC14	CPTC13	CPTC12	CPTC11	CPTC10	CPTC9	CPTC8
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

16 位捕捉定时计数器 CPTCH 和 CPTCL 为双重缓存器设计，但核心总线只有 8 位，故处理 16 位数据时须锁存标志，以免瞬间状态影响到 16 位数据而出错。写入数据时，写 CPTCL 是锁存控制标志；读取数据时，读 CPTCL 是锁存控制标志。故写入数据到捕捉定时器时，要先写入数据到 CPTCH，再写入数据到 CPTCL，即执行写入数据到 CPTCL 时，所有数据都已经写入 16 位捕捉定时器中。反之，读取捕捉定时器的数据时，要先读取 CPTCL 的数据，再读取 CPTCH 的数据，即执行读取 CPTCH 时，捕捉定时器中的所有数据都已经被读取完毕。

- **读取捕捉定时计数缓存器中的数据时，要先读取 CPTCL 中的数据，再读取 CPTCH 中的数据。**
- **写入数据到捕捉定时计数缓存器中时，要先写入数据到 CPTCH，再写入数据到 CPTCL。**

8.6.7 T1 操作举例

- **T1 定时器：**

```
; 复位 T1。
        MOV         A, #00H             ; 清 T1M。
        B0MOV       T1M, A

; 设置 T1Rate。
        MOV         A, #0nnn0000b
        B0MOV       T1M, A

; 设置 T1CH，T1CL 寄存器获取 T1 中断间隔时间。
        MOV         A, #value1          ; 先设置高字节。
        B0MOV       T1CH, A
        MOV         A, #value2          ; 设置低字节。
        B0MOV       T1CL, A

; 清 T1IRQ。
        B0BCLR      FT1IRQ

; 使能 T1 定时器和中断功能。
        B0BSET      FT1IEN              ; 使能 T1 中断功能。
        B0BSET      FT1ENB              ; 使能 T1 定时器。
```

- **T1 捕捉定时器，测量连续信号。**

```
; 复位 T1。
        CLR         T1M                 ; 清 T1M。

; 设置 T1 时钟 rate，选择/使能 T1 捕捉定时器功能。
        MOV         A, #0nnnm000b       ; “nnn”代表 T1rate[2:0]，选择 T1 时钟 Rate。
        B0MOV       T1M, A              ; “m”为时钟源控制位。
        MOV         A, #000000mmb       ; “mm”代表 CPTG[1:0]，选择 T1 捕捉定时器的功能。
        B0MOV       CPTM, A             ; CPTG[1:0] = 00b，使能 T1 捕捉定时器功能。
                                        ; CPTG[1:0] = 01b/10b/11b，测量使能脉冲宽度和周期。

; 选择捕捉定时器高速/低速模式。
        B0BCLR      FCPTMD              ; CPT 溢出模式。
; or
        B0BSET      FCPTMD              ; T1 溢出模式。

; 清 T1CH，T1CL。
        CLR         T1CH                ; 先清除高字节。
        CLR         T1CL                ; 再清除低字节。

; 设置 16 位定时器 CPTCH、CPTCL，测量连续信号。
        MOV         A, #value1          ; 先设置高字节。
        B0MOV       CPTCH, A
        MOV         A, #value2          ; 再设置低字节。
        B0MOV       CPTCL, A

; 清 T1IRQ。
        B0BCLR      FT1IRQ

; 使能 T1 定时器，中断功能和 T1 捕捉定时器功能。
        B0BSET      FT1IEN              ; 使能 T1 中断功能。
        B0BSET      FT1ENB              ; 使能 T1 定时器。
        B0BSET      FCPTEN              ; 使能 T1 捕捉定时器功能。

; 设置捕捉定时器开始位。
        B0BSET      FCPTStart
```

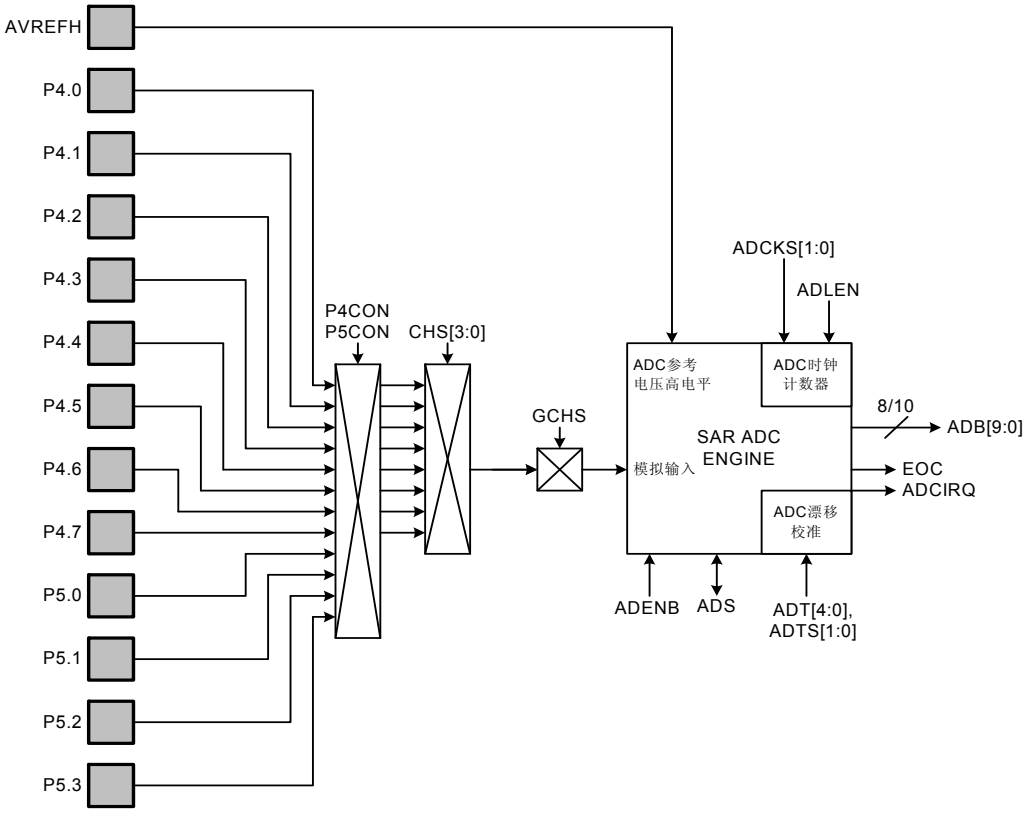
- **T1 捕捉定时器，测量信号周期。**

```
; 复位 T1。
        MOV         A, #00H             ; 清 T1M。
        B0MOV       T1M, A

; 设置 T1Rate，选择输入源，选择/使能 T1 捕捉定时器功能。
        MOV         A, #0nnnm000b       ; “nnn”代表 T1rate[2:0]，选择 T1 时钟 Rate。
        B0MOV       T1M, A              ; “m”为时钟源控制位。
        MOV         A,#000000mmb        ; “mm”代表 CPTG[1:0]，选择 T1 捕捉定时器的功能。
        B0MOV       CPTM,A
                                        ; CPTG[1:0] = 00b，捕捉定时器功能。
                                        ; CPTG[1:0] = 01b，测量脉冲的高电平宽度。
                                        ; CPTG[1:0] = 10b，测量脉冲的低电平宽度。
                                        ; CPTG[1:0] = 11b，测量脉冲的周期。

; 清 T1CH，T1CL。
        CLR         T1CH                ; 先清除高字节。
        CLR         T1CL                ; 再清除低字节。

; 清 T1IRQ。
        B0BCLR      FT1IRQ

; 使能 T1 定时器、中断功能和捕捉定时器功能。
        B0BSET      FT1IEN              ; 使能 T1 中断功能。
        B0BSET      FT1ENB              ; 使能 T1 定时器。
        B0BSET      FCPTEN              ; 使能 T1 捕捉定时器功能。

; 设置捕捉定时器开始位。
        B0BSET      FCPTStart
```

9 12 通道AD转换（ADC）

9.1 概述

模拟数字转换（ADC）是一个 SAR 结构，内置 12 个模拟通道（AIN0~AIN11），高达 1024 阶的分辨率，能将一个模拟信号转换成相应的 10 位数字信号。ADC 内置的 12 个模拟通道可以测量 12 种不同的模拟信号源，由 CHS[3:0]和 GCHS 位控制。通过 ADLEN 位可以选择 ADC 的分辨率为 8 位或者 10 位；可以通过 ADCKS[1:0]位选择 ADC 的转换速率以决定 ADC 的转换时间。ADC 参考电压的高电平来自 AVREFH 引脚。ADC 内置 P4CON 和 P5CON 寄存器来设置模拟输入引脚，必须由程序将 P4 和 P5 设为不带上拉电阻的输入引脚。设置好 ADENB 和 ADS 位后，ADC 开始转换，转换结束时，ADC 电路将 EOC 和 ADCIRQ 置 1，并将转换结果存入 ADB 和 ADR 寄存器中。若 ADCIEN=1，ADC 请求中断，AD 转换完成后，ADCIRQ=1 时，程序计数器跳转中断向量地址（ORG 0010H）执行中断服务程序。中断时必须由程序将 ADCIRQ 清零。



9.2 ADC模式寄存器

ADC 模式寄存器 ADM 设置 ADC 的相关配置：包括 ADC 启动，ADC 通道选择和 ADC 处理状态显示等。必须在 AD 开始转换前将这些配置设置完毕。

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	CHS3	CHS2	CHS1	CHS0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **ADENB**：ADC 控制位。**省电模式下，禁止 ADC 以省电。**
0 = 禁止 ADC；
1 = 使能 ADC。

Bit 6 **ADS**：ADC 启动控制位。**AD 转换完成后自动将 ADS 位清零。**
0 = 停止 AD 转换；
1 = 开始 AD 转换。

Bit 5 **EOC**：ADC 状态位。**ADC 开始之前必须由程序将 EOC 位清零。**
0 = AD 转换中；
1 = AD 转换结束，ADS 位复位。

Bit 4 **GCHS**：ADC 全局通道控制位。
0 = 禁止 AIN 通道；
1 = 使能 AIN 通道。

Bit [3:0] **CHS[3:0]**：ADC 输入通道选择位。
0000 = AIN0；0001 = AIN1；0010 = AIN2；0011 = AIN3；0100 = AIN4；0101 = AIN5；0110 = AIN6；
0111 = AIN7；1000 = AIN8；1001 = AIN9；1010 = AIN10；1011 = AIN11；1100~1111 = 系统保留。

ADR 寄存器包括 ADC 模式控制和 ADC 低字节数据缓存器，ADC 配置包括 ADC 时钟速率和 ADC 分辨率。必须在启动 ADC 之前设置好这些配置。

0CAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR	-	ADCKS1	ADLEN	ADCKS0	-	-	ADB1	ADB0
读/写	-	R/W	R/W	R/W	-	-	R	R
复位后	-	0	0	0	-	-	-	-

Bit 6,4 **ADCKS [1:0]**：ADC 时钟 rate 选择位。
00 = Fcpu/16；01 = Fcpu/8；10 = Fcpu/1；11 = Fcpu/2。

Bit 5 **ADLEN**：ADC 分辨率选择位。
0 = 8 位；
1 = 10 位。

9.3 ADC数据缓存器

ADC 数据缓存器共 10 位，用来存储 AD 转换结果，ADB 存放结果的高字节（bit2~bit9），ADR（ADR[1:0]）存放低字节（bit0~bit1）。ADC 数据缓存器是只读寄存器，系统复位后处于未知状态。

ADB[9:2]：8 位 ADC 模式下，ADC 数据存放于 ADB 寄存器中。
ADB[9:0]：10 位 ADC 模式下，ADC 数据存放于 ADB 和 ADR 寄存器中。

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADB	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2
读/写	R	R	R	R	R	R	R	R
复位后	-	-	-	-	-	-	-	-

Bit[7:0] **ADB[7:0]**：8 位 ADC 数据缓存器，存放 10 位 ADC 的高字节数据。

0CAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR	-	ADCKS1	ADLEN	ADCKS0			ADB1	ADB0
读/写	-	R/W	R/W	R/W			R	R
复位后	-	0	0	0			-	-

Bit [1:0] **ADB [1:0]**：10 位 ADC 的低字节数据缓存器。

AIN 输入电压 v.s. ADB 输出数据

AIN n	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
0/1024*VREFH	0	0	0	0	0	0	0	0	0	0
1/1024*VREFH	0	0	0	0	0	0	0	0	0	1
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
1022/1024*VREFH	1	1	1	1	1	1	1	1	1	0
1023/1024*VREFH	1	1	1	1	1	1	1	1	1	1

针对不同的应用，用户可能需要精度介于 8 位到 10 位之间的 AD 转换器。对于这种情况，可以通过对保存在 ADR 和 ADB 中的转换结果进行处理得到。首先，用户必须选择 10 位分辨率的模式，进行 AD 转换，然后在转换结果中去掉最低的几位得到需要的结果。如下表所示：

ADC Resolution	ADB								ADR	
	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
8-bit	O	O	O	O	O	O	O	O	x	x
9-bit	O	O	O	O	O	O	O	O	O	x
10-bit	O	O	O	O	O	O	O	O	O	O

✷ **注：ADC 数据缓存器包括 ADB 和 ADR 低字节，系统复位后，ADC 数据缓存器的值是未知的。**

9.4 ADC操作说明和注意事项

9.4.1 ADC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，高电压为 AVREFH。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。
ADC 外部参考电压=2V~VDD。

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 采用输入信号电压 ≦ ADC 参考高电压**

9.4.2 AD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 分辨率和 ADC 时钟 Rate 控制，10 位 ADC 的转换时间为 1/（ADC 时钟/4）*14 S；8 位 ADC 的转换时间为 1/（ADC 时钟/4）*12 S。ADC 的时钟源为 Fcpu，包括 Fcpu/1，Fcpu/2，Fcpu/8，Fcpu/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 Rate 的模拟信号，必须要选择一个高 Rate 的 ADC 转换 Rate。如果 ADC 的转换时间比模拟信号的转换 Rate 慢，则 ADC 的结果出错。故选择合适的 ADC 时钟 Rat 和 ADC 分辨率才能得到合适的 ADC 转换 Rate。

10 位 ADC 转换时间 = 1/(ADC 时钟 Rate/4)*14 sec

ADLEN	ADCKS1, ADCKS0	ADC 时钟 Rate	Fcpu=4MHz		Fcpu=16MHz	
			AD 转换时间	AD 转换 Rate	AD 转换时间	AD 转换 Rate
1 (10-bit)	00	Fcpu/16	1/(4MHz/16/4)*14 = 224 us	4.464KHz	1/(16MHz/16/4)*14 = 56 us	17.857KHz
	01	Fcpu/8	1/(4MHz/8/4)*14 = 112 us	8.929KHz	1/(16MHz/8/4)*14 = 28 us	35.71KHz
	10	Fcpu	1/(4MHz/4)*14 = 14 us	71.43KHz	1/(16MHz/4)*14 = 3.5 us	286KHz
	11	Fcpu/2	1/(4MHz/2/4)*14 = 28 us	35.71KHz	1/(16MHz/2/4)*14 = 7 us	143KHz

8 位 ADC 转换时间 = 1/(ADC 时钟 Rate/4)*12 sec

ADLEN	ADCKS1, ADCKS0	ADC 时钟 Rate	Fcpu=4MHz		Fcpu=16MHz	
			AD 转换时间	AD 转换 Rate	AD 转换时间	AD 转换 Rate
0 (8-bit)	00	Fcpu/16	1/(4MHz/16/4)*12 = 192 us	5.208KHz	1/(16MHz/16/4)*12 = 48 us	20.833KHz
	01	Fcpu/8	1/(4MHz/8/4)*12 = 96 us	10.416KHz	1/(16MHz/8/4)*12 = 24 us	41.667KHz
	10	Fcpu	1/(4MHz/4)*12 = 12 us	83.333KHz	1/(16MHz/4)*12 = 3 us	333.333KHz
	11	Fcpu/2	1/(4MHz/2/4)*12 = 24 us	41.667KHz	1/(16MHz/2/4)*12 = 6 us	166.667KHz

9.4.3 ADC引脚配置

ADC 输入引脚与 P4 和 P5 口共用，ADC 输入通道的选择由 ADCHS[3:0]控制，ADCCHS[3:0]=0000 时选择 AIN0，ADCCHS[3:0]=0001 时选择 AIN1……同一时间设置 P4 或者 P5 口的一个引脚作为 ADC 的输入引脚，该引脚必须设置为输入引脚，禁止内部上拉，并首先由程序使能 P4CON 和 P5CON 寄存器。通过 ADCHS[3:0]选择好 ADC 输入通道后，GCHS 置 1 以使能 ADC 功能。

- **ADC 输入引脚为 GPIO 引脚时必须设为输入模式。**
- **必须禁止 ADC 输入引脚的内部上拉电阻。**
- **ADC 输入通道的 P4CON 和 P5CON 位必须置 1。**

ADC 输入引脚与普通 I/O 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 P4 和 P5 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。P4CON/P5CON 为 P4/P5 口的配置寄存器，将 P4CON[7:0]/P5CON[3:0]置 1，其对应的 P4/P5 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。

0C6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4CON	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **P4CON[7:0]**：P4.n 配置控制位。
0 = P4.n 可以作为模拟输入（ADC 输入）引脚或者 GPIO 引脚；
1 = P4.n 只能作为模拟输入引脚，不能作为 GPIO 引脚。

0C7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5CON					P5CON3	P5CON2	P5CON1	P5CON0
读/写					R/W	R/W	R/W	R/W
复位后					0	0	0	0

Bit[3:0] **P5CON[3:0]**：P5.n 配置控制位。
0 = P5.n 可以作为模拟输入（ADC 输入）引脚或者 GPIO 引脚；
1 = P5.n 只能作为模拟输入引脚，不能作为 GPIO 引脚。

✷ **注：P4.n/P5.n 作为 GPIO 引脚而不是 ADC 输入引脚时，P4CON.n/P5CON.n 必须置 0，否则 P4.n/P5.n 的普通 I/O 信号会被隔离。**

9.4.4 ADC操作举例

- **ADC：**

```
; 复位 ADC。
        CLR     ADM             ; 清 ADM 寄存器。

; 设置 ADC 时钟 Rate 和 ADC 分辨率。
        MOV     A, #0nmn0000b   ; nn DCKS[1:0]代表 ADC 时钟 Rate。
        B0MOV   ADR, A          ; m 代表 ADC 分辨率。

; 设置 ADC 输入通道。
        MOV     A, #value1      ; 设置 P4CON 选择 ADC 输入通道。
        B0MOV   P4CON, A
        MOV     A, #value2      ; 设置 ADC 输入通道为输入模式。
        B0MOV   P4M, A
        MOV     A, #value3      ; 禁止 ADC 输入通道的内部上拉电阻。
        B0MOV   P4UR, A

; 使能 ADC。
        B0BSET  FADENB

; 执行 ADC 100us 启动时间延迟循环。
        CALL    100usDLY        ; 100us 延迟循环。

; 选择 ADC 输入通道。
        MOV     A, #value       ; 设置 ADCHS[3:0]选择 ADC 输入通道。
        OR      ADM, A

; 使能 ADC 输入通道。
        B0BSET  FGCHS

; 使能 ADC 中断功能。
        B0BCLR  FADCIRQ         ; 清 ADC 中断请求。
        B0BSET  FADCIEN         ; 使能 ADC 中断功能。

; 开始 AD 转换。
        B0BSET  FADS
```

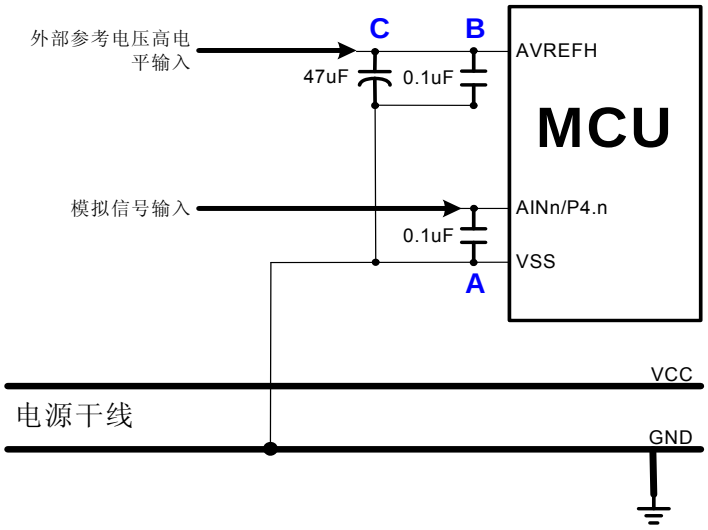
✷ 注：
1、使能 ADENB 后（不是使能 ADS），系统必须延迟 100us 等待启动 ADC，然后设置 ADS 开始 AD 转换，否则 ADC 的结果出错。系统正常运行时，设置 ADENB 一次，延时一次。
2、睡眠模式和绿色模式下禁止 ADC 以省电。

ADC 转换：

```
; 禁止 ADC 中断模式。
@@:
        B0BTS1      FEOC            ; 检查 ADC 状态标志。
        JMP         @B              ; EOC=0：AD 转换中。
        B0MOV       A, ADB          ; EOC=1：AD 转换结束，处理 AD 转换结果。
        B0MOV       BUF1,A
        MOV         A, #00000011b
        AND         A, ADR
        B0MOV       BUF2,A
        …                           ; AD 转换结果处理完成。
        CLR         FEOC            ; 清除 ADC 状态标志以准备下一次 ADC。

; 使能 ADC 中断模式。
        ORG         10H             ; 中断向量。
INT_SR:                             ; 中断服务程序。
        B0BTS1      FADCIRQ         ; 检查 ADC 中断标志。
        JMP         EXIT_INT        ; ADCIRQ=0：没有 ADC 中断发生。
        B0MOV       A, ADB          ; ADCIRQ=1：AD 转换结束，处理 AD 转换结果。
        B0MOV       BUF1,A
        MOV         A, #00000011b
        AND         A, ADR
        B0MOV       BUF2,A
        …                           ; AD 转换结果处理完成。
        CLR         FEOC            ; 清除 ADC 状态标志以准备下一次 ADC。
        JMP         INT_EXIT

INT_EXIT:
        RETI                        ; 退出中断。
```

✷ **注：AD 转换结束时自动将 ADS 清零，EOC 即时反映 ADC 的转换状态，ADS=1 时自动清除 EOC，用户无需通过程序来清零。**

9.5 ADC应用电路



模拟信号从 ADC 输入引脚 AINn/P4.n 输入。在 ADC 输入引脚和 VSS 之间（A）必须连接一个 0.1uF 的电容，且要尽可能的靠近 ADC 输入引脚。不能将电容的 GND 直接连接到电源干线上的 GND，必须通过 VSS 引脚。该电容可以减少电源干扰对模拟信号的影响。

在 AVREFH 引脚和 VSS 之间连接电容，首先在图中 C 处连接一个 47uF 的电解电容，再在 B 处连接一个 0.1uF 的电容，且要尽可能的靠近 AVREFH 引脚。不能将电容的 GND 直接连接到电源干线上的 GND，必须通过 VSS 引脚。

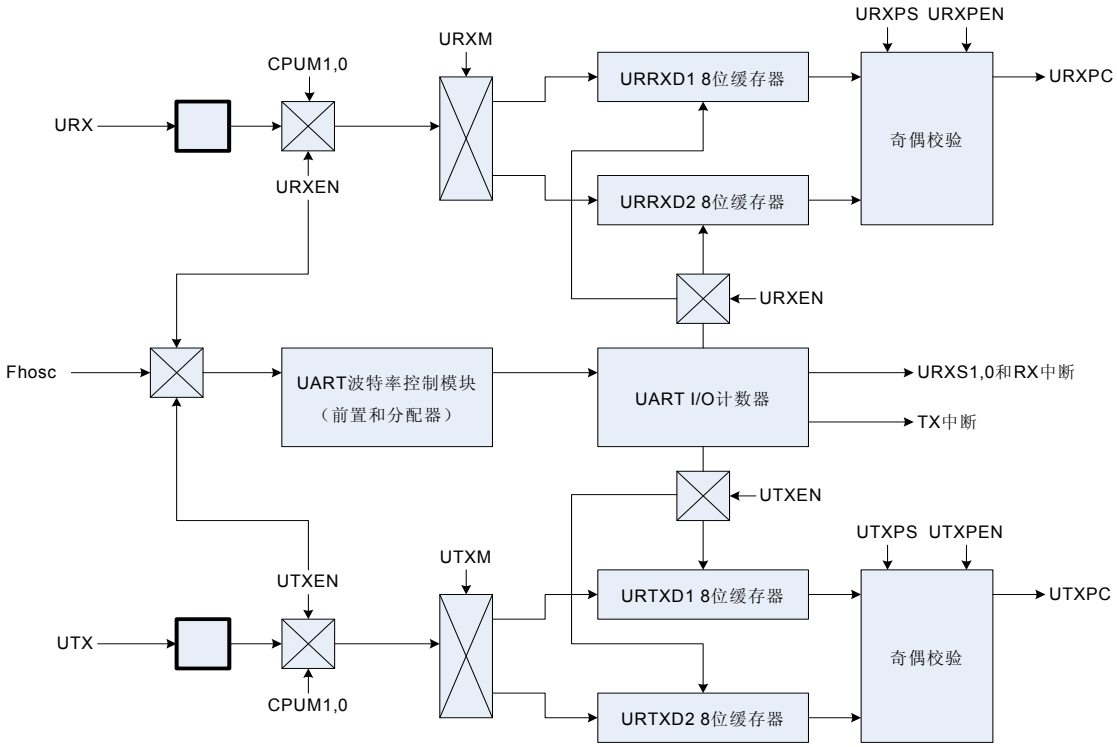
10 通用异步收发器（UART）

10.1 概述

UART（通用异步收发器）可提供来传送低速数据及与低速外部设备通信。 SN8F27E60 系列单片机支持 RS232 规范，可传输 1 字节的数据，数据传送格式为：起始位，8 位数据位，校验位及停止位，并能设置各种波特率，在使用上灵活方便。UART I/O 引脚支持推挽式和漏极开路结构，由寄存器控制。

UART 特性包括：

- **全双工，2 线异步数据传输；**
- **可编程波特率；**
- **支持 8 位数据长度；**
- **奇偶校验位；**
- **传输结束中断；**
- **支持 DMX512 协议；**
- **支持 Break pocket 功能；**
- **支持大范围的波特率。**



UART 接口示意图

10.2 UART操作

UART RX 和 TX 引脚与普通 I/O 引脚共用。使能 UART 功能时（RXDEN=1, TXDEN=1），共用引脚为 UART 功能引脚并自动禁止该引脚的 GPIO 功能。禁止 UART 功能时，该引脚返回到普通 I/O 模式，引脚状态改变为 UART 模式前状态。UART 数据缓存器支持 1 字节。

UART 支持中断功能，UTRXIEN/UTTXIEN 为传送数据中断功能控制位。UTRXIEN=0 时，禁止 UART 接收中断功能；UTTXIEN=0 时，禁止 UART 发送中断功能；UTRXIEN=1 时，使能 UART 接收中断功能；UTTXIEN=1 时，使能 UART 发送中断功能；UART 中断功能使能时，程序计数器指向中断向量（ORG 0013H/0014H）处，在 UART 操作完成后执行 UART 中断服务程序，UTRXIRQ 和 UTTXIRQ 是 UART 的中断请求标志位，在 UTRXIEN=0 或 UTTXIEN=0 使能时也可以显示 UART 操作状态，但必须由程序清零。UART 操作结束时，UTRXIRQ/UTTXIRQ 被置“1”。

UART 内置“Busy Bit”显示 UART 总线的状态。URXBZ 是 UART RX 的操作指示位，UTXBZ 是 UART TX 操作指示位。若总线处于工作状态时，Busy bit 显示 1；总线完成工作或者处于空闲状态是，Busy bit 显示 0。

UART TX 操作由加载 UTXD 数据缓存器控制，UART TX 配置完成后，加载需要发送的数据到 UTXD 缓存器中，然后 UART 开始发送。

UART RX 操作由接收来自主机终端的起始位控制，UART RX 配置完成后，URX 引脚检测到起始位的下降沿，UART 开始接收来自主机终端的数据。

UART 提供 URXPC 位和 UFMER 位来检测接收到的数据。URXPC 检测接收到奇偶位，如果接收到的奇偶位错误，URXPC 位置 1，若 URXPC 位为 0，则表示接收到的奇偶位正确。UFMER 位检测接收到的数据串，数据串的错误情形包括起始位错误、停止位错误、stream 长度错误，UART 波特率错误等等，上述的每一种情形都会使 UFMER 位显示为 1。

10.3 UART波特率

UART 时钟为 2-stage 的结构，包括前置分频器和 8 位缓存器。UART 时钟源由系统振荡器 Fhosc 提供，Fhosc 通过 UART 前置分频器以获得 UART 主时钟 Fuart。UART 前置分频器共有 8 个选项（Fhosc/1、Fhosc/2、Fhosc/4、Fhosc/8、Fhosc/16、Fhosc/32、Fhosc/64、Fhosc/128），以及 3 位控制位（URS[2:0]。UART 主时钟（Fuart）是 front-end 时钟，通过 UART 8 位缓存器（URCR）获得 UART 处理时钟及决定 UART 波特率。

UART 前置分频选项 URS[2:0]	UART 主时钟 Rate	Fuart (Fhosc=16MHz)
000b	Fhosc/1	16MHz
001b	Fhosc/2	8MHz
010b	Fhosc/4	4MHz
011b	Fhosc/8	2MHz
100b	Fhosc/16	1MHz
101b	Fhosc/32	0.5MHz
110b	Fhosc/64	0.25MHz
111b	Fhosc/128	0.125MHz

0E6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URCR	URCR7	URCR6	URCR5	URCR4	URCR3	URCR2	URCR1	URCR0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

UART 波特率时钟源为 Fhosc，可由预分频器分频后选择，计算方式如下：

UART 波特率 = 1/2 *(Fuart * 1/(256 - URCR))…bps

Fhosc = 16MHz

波特率	UART 前置分频	URS[2:0]	URCR (Hex)	精确度(%)
1200	Fhosc/32	101b	30	-0.16%
2400	Fhosc/32	101b	98	-0.16%
4800	Fhosc/32	101b	CC	-0.16%
9600	Fhosc/32	101b	E6	-0.16%
19200	Fhosc/32	101b	F3	-0.16%
38400	Fhosc/1	000b	30	-0.16%
51200	Fhosc/1	000b	64	-0.16%
57600	Fhosc/1	000b	75	0.08%
102400	Fhosc/1	000b	B2	-0.16%
115200	Fhosc/1	000b	BB	-0.64%
128000	Fhosc/1	000b	C1	0.80%
250000	Fhosc/1	000b	E0	0.00%

✷ 注：
1、我们我们强烈建议不要设置 URCR=FFH，否则 UART 会发生错误；
2、如果 Noise_Filter 选择为 Enable，我们强烈要求设置 Fcpu 为 Fhosc/2~Fhosc/16，否则 UART 会发生错误；Noise_Filter 选择为 Disable，则 Fcpu 的选择没有限制；

10.4 UART发送格式

UART 发送数据格式包括：空闲状态，起始位，8 位数据位，奇偶校验位和停止位，如下图所示：

UART 传送格式（带奇偶校验位）

UART 传送格式（无奇偶校验位）

总线空闲状态

总线空闲状态即总线上无任何操作的状态。单片机 UART 接收总线空闲状态为悬浮状态，由终端发送设备置高，发送总线空闲状态为高。UART 总线由 URXEN 和 UTXEN 使能后设置。

起始位

UART 为异步通信，因而需要向接收器提供一个标志位以说明传送开始的信息，起始位起到这样的作用，数据格式从高变低即标志传送开始，该信息位持续一位的时间。接收器可以通过起始位识别到将有数据传送来。

8 位数据位

数据格式为 8 位长度，发送起始位后，接着发送最低位（LSB）。数据位持续的时间为 UART 波特率一个单元的时间，由寄存器控制。

奇偶校验位

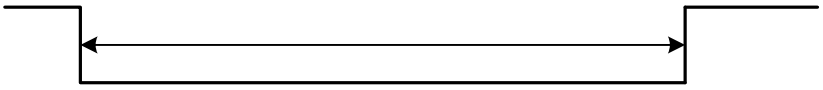
奇偶校验位用来检测数据传送是否出错，附加在需要发送数据之后。奇偶校验位的数据由校验方法得到，且由 URXPS/UTXPS 位控制。接收到数据及其校验位后，程序自动检查奇偶校验位是否正确，由 UPXPC 位显示校验结果。奇偶校验位功能由 URXPEN/UTXPEN 位控制。如果奇偶校验位功能禁止，UART 传送时将不传送奇偶检验位，发送完数据后接着发送停止位。

停止位

与起始位格式相同，停止位用来指示 UART 已传送结束。停止位以电平由低变高为准，并且持续一位的时间。

10.5 BREAK POCKET

break pocket 是一种空闲的 stream，可以复位 UART 总线。Break pocket 是一个长时间的 0pocket，周期为 88us~1s。



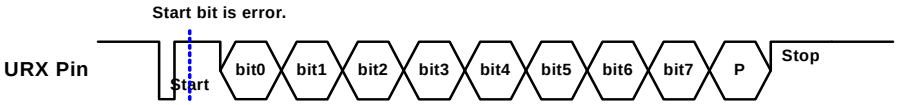
TX Break Pocket：UART 内置 UTXBRK 位发送 Break pocket。当 UTXEN = 1（使能 UART TX 功能）时，设置 UTXBRK 位发送 Break pocket。当 Break pocket 完成发送后，UTXIRQ 置 1，UTXBRK 被自动清零。发送 break pocket 的周期为 25 个 UART 波特率时钟。假如 UART 的波特率为 250000bps，则 break pocket 的周期为 100us。

UART TX Break Pocket 周期 = 25/UART 波特率…sec

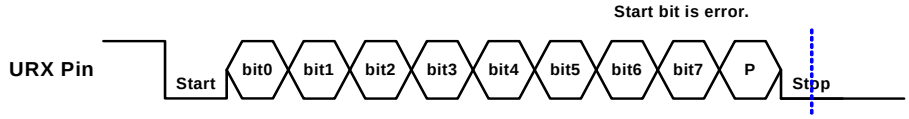
RX Break Pocket：UART 接收 break pocket 会得到一个错误的 frame 信号，这是由于数据周期比 UART 的典型持续时间长，UART 不能接收一个完整的数据 pocket。接收完一个 UART pocket 后，break pocket 仍然输出低电平。UART 的 frame 错误标志（URMEN=1）和 URXIRQ。奇偶校验位可能在校验模式下出错，UART 会将其返回到初始值直到检测到下一个起始位。

10.6 异常POCKET

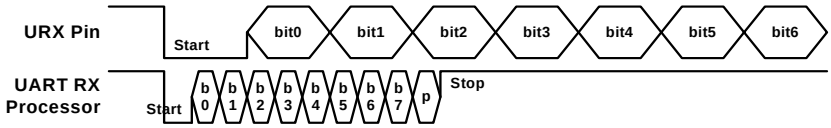
UART RX 模式下会出现异常 pocket 的情况，Break pocket 就是异常 pocket 的一种。异常 pocket 包括 stream 周期错误，起始位错误，停止位错误等。当 UART 接收到异常 pocket，UFMER 位则会置 1，释放 URXIRQ。系统通过韧件发现异常 pocket，UART 会改变初始值直到检测到下一个起始位。



UART 检测到错误的起始位，释放 UFMER 标志，但 UART 仍然接收 pocket 直至完成。



UART 检测到错误的停止位，释放 UFMER 标志，但 UART 仍然接收 pocket 直至完成。



若主机的 UART 波特率与接收终端的不匹配，则接收到 pocket 错误。但也很难区分接收到的 pocket 是否正确，因为接收到的错误 pocket 有可能符合 UART 规则，但数据不对，这就需要检测 UFMER 位和 URXPC 位的状态来决定 stream。如果两种情况看起来都是正确的，但其实是异常的 pocket，则 UART 就需将该 pocket 当作正确的来接受。

10.7 UART接收控制寄存器

0E5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URRX	URXEN	URXPEN	URXPS	URXPC	UFMER	URS2	URS1	URS0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **URXEN**：UART RX 控制位。
0 = 禁止 UART RX，URX 引脚为 GPIO 引脚，返回到上一个 GPIO 状态；
1 = 使能 UART RX，URX 引脚为 UART RX 引脚。

Bit 6 **URXPEN**：UART RX 奇偶校验控制位。
0 = 禁止 UART RX 奇偶校验位功能，数据 stream 不包括奇偶校验位；
1 = 使能 UART RX 奇偶校验位功能，数据 stream 包括奇偶校验位。

Bit 5 **URXPS**：UART RX 奇偶校验位格式控制位。
0 = UART RX 奇偶校验位的格式为偶数校验；
1 = UART RX 奇偶校验位的格式为奇数校验。

Bit 4 **URXPC**：UART RX 奇偶校验位检测标志。
0 = 奇偶校验位正确，无奇偶校验功能；
1 = 奇偶校验位错误。

Bit 3 **UFMER**：UART RX stream frame 错误标志位。
0 = 收集 UART frame；
1 = UART frame 错误，包括起始位、停止位和 stream 长度。

Bit [2:0] **URS[2:0]**：UART 前置分频器选择位。
000 = Fhosc/1；001 = Fhosc/2；010 = Fhosc/4；011 = Fhosc/8；100 = Fhosc/16；101 = Fhosc/32；
110 = Fhosc/64；111 = Fhosc/128。

10.8 UART发送控制寄存器

0E4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URTX	UTXEN	UTXPEN	UTXPS	UTXBRK	URXBZ	UTXBZ	-	-
读/写	R/W	R/W	R/W	R/W	R	R	-	-
复位后	0	0	0	0	0	0	-	-

Bit 7 **UTXEN**：UART TX 控制位。
0 = 禁止 UART TX，UTX 引脚为 GPIO 引脚，返回到上一个 GPIO 状态；
1 = 使能 UART TX，UTX 引脚为 UART TX 引脚，空闲时为高。

Bit 6 **UTXPEN**：UART TX 奇偶校验控制位。
0 = 禁止 UART TX 奇偶校验位功能，数据 stream 不包括奇偶校验位；
1 = 使能 UART TX 奇偶校验位功能，数据 stream 包括奇偶校验位。

Bit 5 **UTXPS**：UART TX 奇偶校验位格式控制位。
0 = UART TX 奇偶校验位格式为偶数校验；
1 = UART TX 奇偶校验位格式为奇数校验。

Bit 4 **UTXBRK**：UART TX BREAK pocket 控制位。
0 = UART BREAK pocket 发送完成；
1 = UART BREAK pocket 发送开始。

Bit 3 **URXBZ**：UART RX 操作状态标志位。
0 = UART RX 操作完成，处于空闲状态；
1 = UART RX 操作过程中。

Bit 2 **UTXBZ**：UART TX 操作状态标志位。
0 = UART TX 操作完成，处于空闲状态；
1 = UART TX 操作过程中。

✷ **注：URXBZ 和 UTXBZ 指示 UART 的工作状态，在设置 UART 发送或者接收数据后，必须要延时(2*Fcpu/Fuart)个 NOP 指令后，才能通过检测 URXBZ 和 UTXBZ 标志来判断 UART 的状态。**

10.9 UART数据缓存器

0E7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UTXD	UTXD7	UTXD6	UTXD5	UTXD4	UTXD3	UTXD2	UTXD1	UTXD0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **UTXD**：UART 发送数据缓存器。

0E8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URXD	UTXD27	UTXD26	UTXD25	UTXD24	UTXD23	UTXD22	UTXD21	UTXD20
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **URXD**：UART 接收数据缓存器。

10.10 UART操作举例

- **UART TX：**

```
; 选择奇偶校验位功能。
        B0BCLR      FUTXPEN     ; 禁止 UART TX 奇偶校验位功能。
;or
        B0BSET      FUTXPEN     ; 使能 UART TX 奇偶校验位功能。

; 选择奇偶校验位格式。
        B0BCLR      FUTXPS      ; 偶数校验。
;or
        B0BSET      FUTXPS      ; 奇数校验。

; 设置 UART 波特率。
        MOV         A, #value1  ; 设置 UART 前置分频器 URS[2:0]。
        B0MOV       URRX, A
        MOV         A, #value2  ; 设置 UART 波特率 8 位缓存器。
        B0MOV       URCR, A

; 使能 UART TX 引脚。
        B0BSET      FUTXEN      ; 使能 UART TX 功能和引脚。

; 使能 UART TX 中断功能。
        B0BCLR      FUTXIRQ     ; 清 UART TX 中断标志。
        B0BSET      FUTXIEN     ; 使能 UART TX 中断功能。

; 加载 TX 数据缓存器，执行 TX 发送。
        MOV         A, #value3  ; 加载 8 位数据到 UTXD 数据缓存器。
        B0MOV       UTXD, A
                                ; 加载完成后，UART TC 开始发送。
        NOP
; 检查 TX 操作状态。
        B0BTS0      FUTXBZ      ; 检查 UTXBZ 位。
        JMP         CHKTX       ; UTXBZ=1，TX 发送过程中。
        JMP         ENDTX       ; UTXBZ=0，TX 发送完成。
```

✷ **注：UART TX 通过加载 UTXD 数据缓存器开始发送。**

- **发送 Break Pocket：**

```
; 选择奇偶校验位功能。
        B0BCLR      FUTXPEN     ; 禁止 UART TX 奇偶校验位功能。
;or
        B0BSET      FUTXPEN     ; 使能 UART TX 奇偶校验位功能。

; 选择奇偶校验位格式。
        B0BCLR      FUTXPS      ; 偶数校验。
;or
        B0BSET      FUTXPS      ; 奇数校验。

; 设置 UART 波特率。
        MOV         A, #value1  ; 设置 UART 前置分频器 URS[2:0]。
        B0MOV       URRX, A
        MOV         A, #value2  ; 设置 UART 波特率 8 位缓存器。
        B0MOV       URCR, A

; 使能 UART TX 引脚。
        B0BSET      FUTXEN      ; 使能 UART TX 功能和引脚。

; 使能 UART TX 中断功能。
        B0BCLR      FUTXIRQ     ; 清 UART TX 中断标志。
        B0BSET      FUTXIEN     ; 使能 UART TX 中断功能。

; 开始 UART break pocket。
        B0BSET      FUTXBRK     ; 发送 UART break pocket。
        NOP

; 检查 TX 操作。
        B0BTS0      FUTXBZ      ; 检查 UTXBZ 位。
        JMP         CHKTX       ; UTXBZ=1，TX 发送过程中。
        JMP         ENDTX       ; UTXBZ=0，TX 发送完成。
```

✷ **注：UART TX break pocket 由 UTXBRK 位控制，不加载数据到 UTXD 缓存器中。**

- **UART RX：**

```
; 选择奇偶校验位功能。
        B0BCLR      FURXPEN     ; 禁止 UART RX 奇偶校验位功能。
;or
        B0BSET      FURXPEN     ; 使能 UART RX 奇偶校验位功能。

; 选择奇偶校验位格式。
        B0BCLR      FURXPS      ; 偶数校验。
;or
        B0BSET      FURXPS      ; 奇数校验。

; 设置 UART 波特率。
        MOV         A, #value1  ; 设置 UART 前置分频器 URS[2:0]。
        B0MOV       URRX, A
        MOV         A, #value2  ; 设置 UART 波特率 8 位缓存器。
        B0MOV       URCR, A

; 使能 UART RX 引脚。
        B0BSET      FURXEN      ; 使能 UART RX 功能和引脚。

; 使能 UART RX 中断功能。
        B0BCLR      FURXIRQ     ; 清 UART RX 中断标志。
        B0BSET      FURXIEN     ; 使能 UART RX 中断功能。
        NOP

; 检查 RX 操作。
        B0BTS0      FURXBZ      ; 检查 URXBZ 位。
        JMP         CHKRX       ; URXBZ=1，TX 接收过程中。
        JMP         ENDRX       ; URXBZ=0，TX 接收完成。
```

✷ **注：UART RX 在主机发送起始位时开始接收。**

11 串行输入/输出收发器（SIO）

11.1 概述

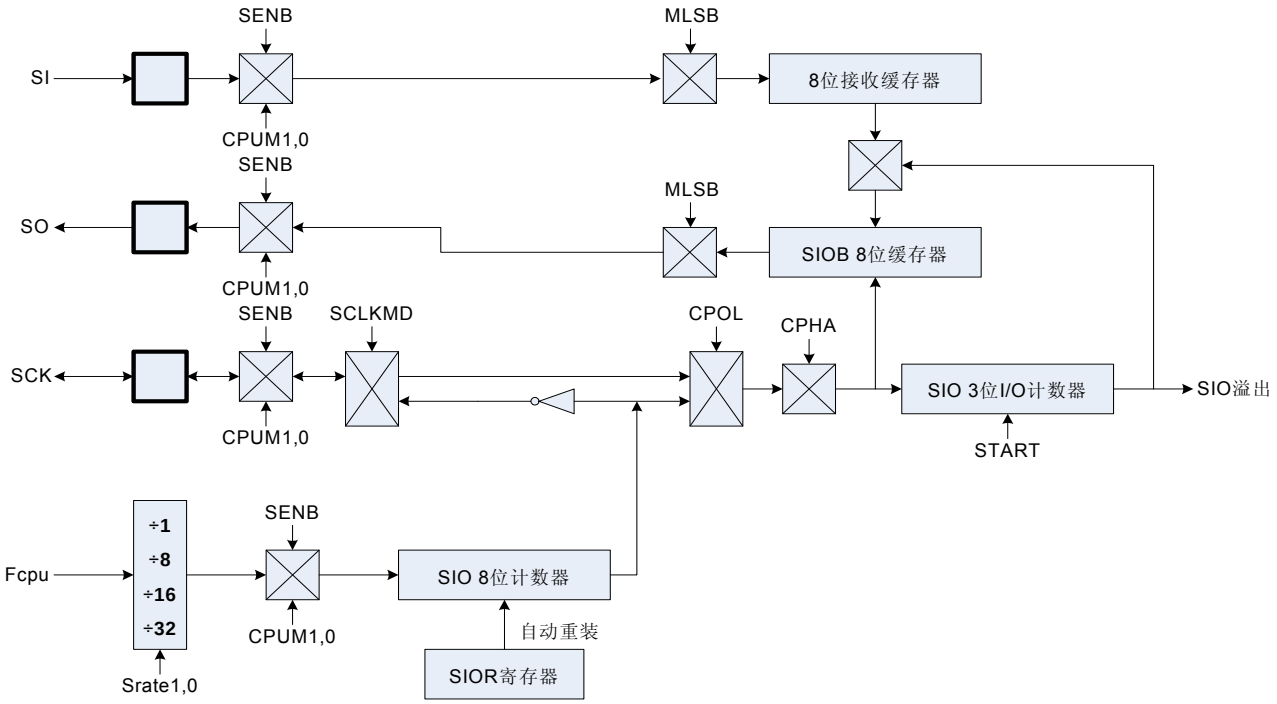
串行输入/输出收发器 SIO 允许数据在单片机与单片机，或单片机与其它外设之间进行传送。它为一个简单的 8 位接口，无规定的协议，封包及控制位。串行收发器模块有 3 个引脚，时钟引脚（SCK），数据输入引脚（SI），数据输出引脚（SO），这几个引脚使得数据可以在主机和从机之间传送。串行收发器具有 8 种数据发送格式，由 3 个位控制：时钟闲置状态，时钟相位和起始位优先发送。其支持大多数 SIO/SPI 通讯规范。

SIO 特性如下：

- **全双工 3 线同步传输；**
- **主控模式 (SCK 为时钟输出) 或从动模式 (SCK 为时钟输入) 操作；**
- **MSB/LSB 起始位传送；**
- **可以通过寄存器控制采样数据在第一个时钟相位有效或者第二个时钟相位有效；**
- **在多路从动装置应用时，SCK, SI, SO 均是可编程漏极开路输出引脚；**
- **主控模式时可设置数据传输速率；**
- **传送结束时产生 SIO 中断。**

11.2 SIO操作

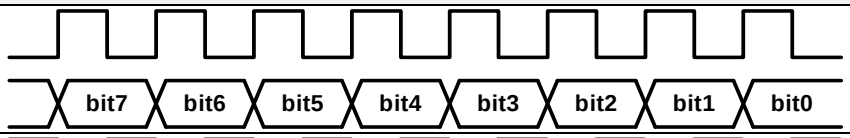
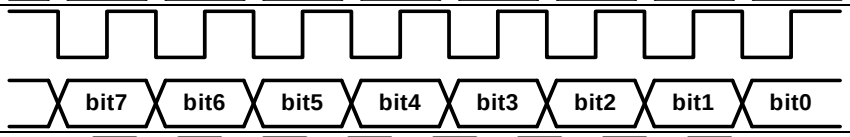
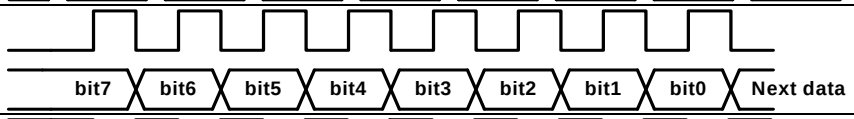
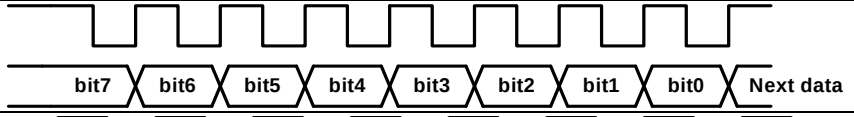
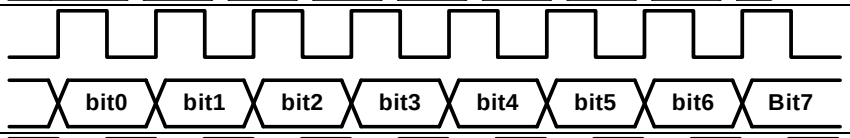
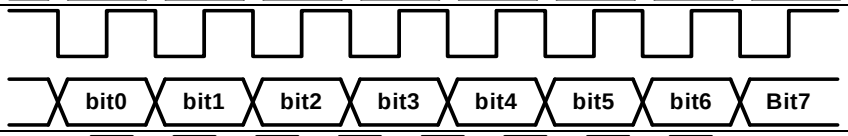
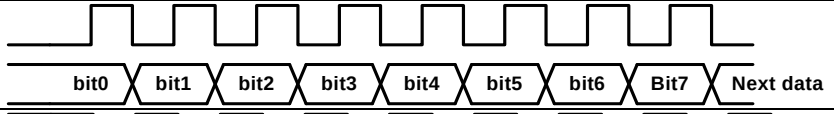
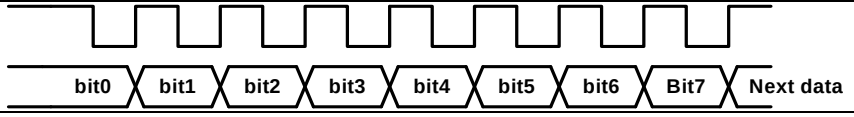
寄存器 SIOM 用来控制 SIO 功能，如发送/接收、时钟速率、触发边沿，时钟闲置状态，时钟相位，起始位优先发送权等。通过设置寄存器 SIOM 的 SENB 和 START 位，SIO 就可自动发送和接收 8 位数据。SIOB 是一个 8 位双层数据缓存器，用于存储发送/接收的数据，SIOC 和 SIOR 具有自动装载功能，能够产生 SIO 的时钟源。3 位的 I/O 计数器可以监控 SIO 的操作，每接收/发送 8 位数据后，会产生一个中断请求。一次发送或接收结束后，SIO 电路将自动禁止，可以通过重新编程 SIOM 寄存器启动下一次的数据传输。CPOL 位用来控制时钟闲置状态，CPHA 位用来控制数据接收的时钟沿方向，这两个位控制着串行收发器的数据的收发格式。起始位发送的优先级由 MLSB 位控制，可以先从低位发送，也可以先从高位开始发送。



SIO 接口电路示意图

串行收发器具有 8 种数据发送格式，由 3 个位控制：MLSB，CPOL 和 CPHA 。发送数据时由“数据传送边沿”控制。当设置为上升沿时，数据的位在 SCK 上升沿发送和接收，在 SCK 下降沿发生数据转换；当设置为下降沿时，数据的位在 SCK 下降沿发送和接收，在 SCK 上升沿发生数据转换。

“CPHA”为时钟相位控制位，它控制有效数据采样的时钟。当 CPHA=1 时，在第一个 SCK 边沿转换数据，在第二个 SCK 边沿发送和接收数据；当 CPHA=0 时，第一个位被锁定，且在 SCK 第一个边沿发送和接收数据。SIO 数据传送时序图如下所示：

MLSB	CPOL	CPHA	Diagrams	Description
0	0	1		SCK 空闲状态 = Low 起始优先发送位 = MSB SCK 数据传送边沿 = 下降沿
0	1	1		SCK 空闲状态 = High 起始优先发送位 = MSB SCK 数据传送边沿 =上升沿
0	0	0		SCK 空闲状态 = Low 起始优先发送位 = MSB SCK 数据传送边沿 = 上升沿
0	1	0		SCK 空闲状态 = High 起始优先发送位 = MSB SCK 数据传送边沿 = 下降沿
1	0	1		SCK 空闲状态 = Low 起始优先发送位 = LSB SCK 数据传送边沿 = 下降沿
1	1	1		SCK 空闲状态 = High 起始优先发送位 = LSB SCK 数据传送边沿 = 上升沿
1	0	0		SCK 空闲状态 = Low 起始优先发送位 = LSB SCK 数据传送边沿 = 上升沿
1	1	0		SCK 空闲状态 = High 起始优先发送位 = LSB SCK 数据传送边沿 = 下降沿

SIO 数据传送时序图

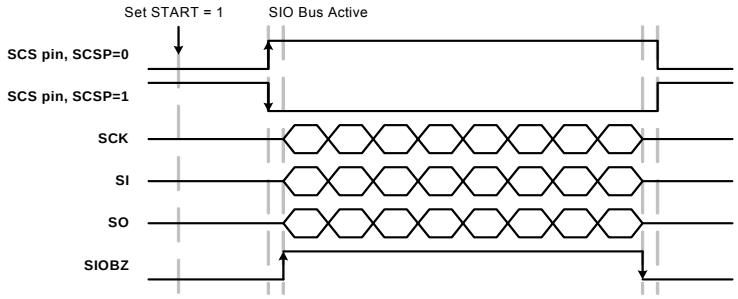
SIO 支持中断功能。SIOIEN 为 SIO 中断允许控制位。SIOIEN=0 时，禁止 SIO 中断功能；SIOIEN=1 时，使能 SIO 中断功能。SIO 产生中断时，程序计数器指向中断向量（ORG 11）处，进而去执行相应的中断服务程序。SIOIRQ 为 SIO 中断请求标志位，在 SIOIEN = 0 时也可以指示 SIO 的执行状态，但是必须由程序清零。SIO 操作完成后，SIOIRQ 置 1，是与 SIO “START” 控制位相反的状态位。

SIOIRQ 和 SIO 起始位指示 8 位数据传送后 SIO 的状态，由于从 SIO 传送结束到 SIOIRQ/START 有效的时间大约为**“1/2*SIO 时钟周期”**，这也就意味着，SIO 传送结束的指示标志位不能立即显示出来。

✷ **注：SIO 操作的第一步是先设置 SIO 各引脚的模式，使能 SENB，选择 CPOL 和 CPHA 位，这些位将控制 SIO 各引脚的模式。**

SIO 内置芯片选择功能，以完善 SIO 多路装置模式。当主机和几个从机之间通过 SIO 总线进行通讯时，芯片的选择决定了到底和其中的哪一个从机进行通讯。芯片的选择引脚为 SCS 引脚，有 SCSEN 位控制。SCS 功能仅在从动模式（SCKMD=1）下有效，SCS 包括两个相位：高电平有效和低电平有效，由 SCSP 位控制。SCSP=1 时，SCS 置高，低电平有效；SCSP=0 时，SCS 置低，高电平有效。SIO 操作有 START 位控制，使能 SCS 功能时，START 置 1，并不表示使能 SIO。SCS 的条件必须满足，若 SCS 的状态不存在，则 SIO 总线保持空闲状态直至 SCS 的条件得以满足。

SIO 内置 SIOBZ 位，以显示 SIO 的操作状态。SIOBZ=1，表示 SIO 工作中；SIOBZ=0，则表示 SIO 处于空闲状态，或者 SIO 处理完成。当 SIO 开始处理时，SIOBZ 变为逻辑高状态；当 SIO 处理完成时，SIOBZ 置为逻辑低。不同模式下的 SIOBZ 状态如下图所示：



11.3 SIOM模式寄存器

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOM	SENB	START	SRATE1	SRATE0	MLSB	SCKMD	CPOL	CPHA
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **SENB**：SIO 功能控制位。
0 = 禁止 SIO 功能，SIO 引脚作为普通输入输出引脚；
1 = 使能 SIO 功能，GPIO 引脚为 SIO 引脚，**SIO 引脚可以为推拉式结构和由 P1OC 寄存器控制的漏极开路结构**。

Bit 6 **START**：SIO 状态控制位。
0 = 传送结束；
1 = 传送过程中。

Bit [5:4] **SRATE1,0**：SIO 传送时钟分频位，当 SCKMD=1，这两个位的功能是无效的。
00 = Fcpu；
01 = Fcpu/32；
10 = Fcpu/16；
11 = Fcpu/8。

Bit 3 **MLSB**：MSB/LSB 优先传送控制位。
0 = MSB 优先发送；
1 = LSB 优先发送。

Bit 2 **SCKMD**：SIO 时钟模式选择位。
0 = 内部时钟（主控模式）；
1 = 外部时钟（从动模式）。

Bit 1 **CPOL**：时钟（SCK）闲置控制位。
0 = SCK 闲置输出低；
1 = SCK 闲置输出高。

Bit 0 **CPHA**：采样数据时钟相位控制位。
0 =在第一个时钟相位接收数据；
1 =在第二个时钟相位接收数据。

0E3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOC	-	-	-	-	-	SIOBZ	SCSEN	SCSP
读/写	-	-	-	-	-	R	R/W	R/W
复位后	-	-	-	-	-	0	0	0

Bit 2 **SIOBZ**：SIO 处理状态标志位。
0 = SIO 处理完成或者处于空闲状态；
1 = SIO 处理过程中。

Bit 1 **SCSEN**：SIO 芯片选择功能控制位。
0 = 禁止芯片选择功能，SCS 返回到上一个 GPIO 模式；
1 = 使能芯片选择功能，SCKMD=1 时，SCS 为 SIO 芯片选择引脚，否则保持 GPIO 模式。

Bit 0 **SCSP**：SIO 芯片选择方向控制位。
0 = 低电平空闲，高电平有效；
1 = 高电平空闲，低电平有效。

SIO 功能是与 GPIO 引脚共用： 下表列出了在使能或禁止 SIO 功能时，SIO 引脚的模式状态和设置。

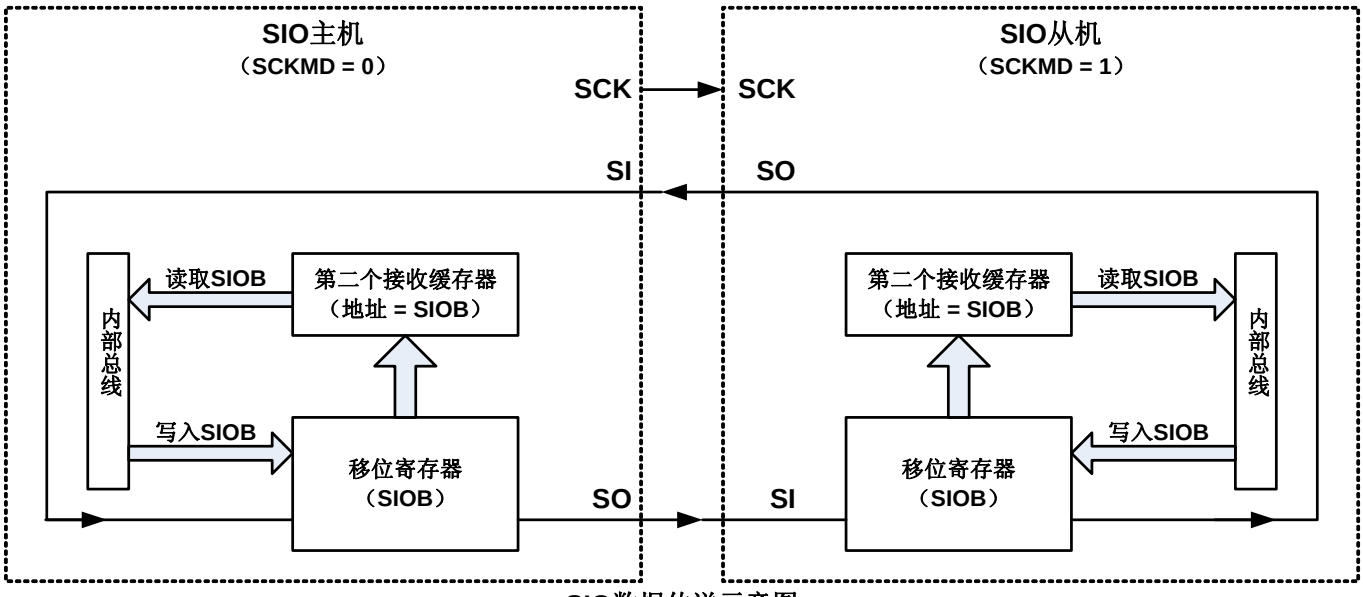
SENB=1 (SIO 使能 SIO 功能**)**		
SCK	(SCKMD=1) 时钟来自外部时钟	GPIO 被自动设为输入模式，不管 PnM 如何设置
	(SCKMD=0) SIO 时钟源来自内部时钟	GPIO 被自动设为输出模式，不管 PnM 如何设置
SI	GPIO 必须设为输入模式，否则 SIO 功能会出错	
SO	SIO =发送/接收	P5.2 被自动设为输出模式，不管 PnM 如何设置
SENB=0 (禁止 SIO 功能)		
GPIO	禁止 SIO 功能时，自动由 PnM 完全控制 GPIO]的 I/O 模式	

- **注 1：在外部时钟模式时，若 SCKMD=1，则 SIO 处于从动模式；内部时钟时，若 SCKMD = 0 则为主控模式。**
- **注 2：不能同时设置 SENB 和 START 位为 1，否则会导致 SIO 传送出错。**
- **注 3：SIO 引脚可为推拉式结构，也可以为漏极开路结构，由 P10C 寄存器控制。**
- **注 4：只在 SCKMD=1 和 SCSEN=1 时才能使能 SCS 引脚，若 SCKMD=0，SCSEN=1，SCS 引脚仍然为 GPIO 引脚。**

11.4 SIOB数据缓存器

0E2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOB	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

SIOB 为 SIO 的数据缓存寄存器，它存储发送/接收的数据。系统在传送时为一次缓存，而在接收时为两级缓存。也就是说在整个移位周期结束前，新的数据不能写入SIOB数据寄存器中；而在接收数据时，在新的数据完全移入前，必须从SIOB数据寄存器中读出接收的数据，否则，前一个数据将会丢失。下图是一个典型的单片机之间的数据通信。由主控单片机发送SCK启动数据传输，两个单片机必须有相同的时钟沿触发方式，并将在同一时刻发送和接收数据。



SIO数据传送示意图

11.5 SIOR寄存器说明

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOR	SIOR7	SIOR6	SIOR5	SIOR4	SIOR3	SIOR2	SIOR1	SIOR0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

寄存器SIOR用于SIO的自动装载，类似于后置分频器，能够提高SIO的时钟精度。用户可对SIOR进行写操作，从而控制SIO的通信速率。SIO的时钟频率计算如下：

SCK 频率 = （SIO 速率 /（256 - SIOR））/2

SIOR = 256 - （1 / 2*（ SCK 频率） * SIO 速率）

- **例：设置 SIO 时钟频率为 5KHz，Fhosc = 4MHz，SIO 的速率 = Fcpu = Fhosc/4.**

SIOR = 256 –（1/（2*5KHz） * 4MHz/4）
= 256 –（0.0001*1000000）
= 256 – 100
= 156

12 MSP

12.1 概述

MSP 是一个串行的通信接口，用来进行几个单片机之间或者外围器件之间的数据交换。外围设备可以是串行 EEPROM、ADC，显示设备等。

MSP 模块可以在下面模式下进行操作：

- **主控 Tx，Rx 模式；**
- **从动 Tx，Rx 模式（带通用地址调用），支持一主机对多从机。**

MSP 的主要性能如下：

- **2 线同步数据发送/接收；**
- **主控（SCL 为时钟输出）/从动（SCL 为时钟输入）操作；**
- **SCL、SDA 为可编程漏极开路输出引脚，可适合多种从动设备的应用；**
- **支持 400K 的时钟频率@Fcpu=4MIPs；**
- **发送/接收结束时产生 MSP 中断。**

12.2 MSP状态寄存器

0EAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPSTAT	-	CKE	D_A	P	S	RED_WRT	-	BF
读/写	-	R/W	R	R	R	R	-	R
复位后	-	0	0	0	0	0	-	0

Bit 6 **CKE：**从机边沿控制位。
从动模式下接收地址或数据字节。
0 = SCL 上升沿锁存数据（默认）；
1 = SCL 下降沿锁存数据。

✷ **注 1：从动模式下发送数据时，接收地址取决于 CKE 的设置；在 SCL 下降沿发送数据。**
✷ **注 2：从动模式下接收数据时，接收的地址和数据都由 CKE 设置。**

Bit 5 **D_A：**数据/地址标志位。
0 = 表示接收或者发送的上一字节是地址；
1 = 表示接收或者发送的上一字节是数据。

Bit 4 **P：**停止标志位。
0 = 没有检测到停止位；
1 = 检测到停止位。

✷ **注 1：检测到起始位后将停止位清零。**

Bit 3 **S：**起始标志位。
0 = 没有检测到起始位；
1 = 检测到起始位。

✷ **注 1：检测到停止位将起始位清零。**

Bit 2 **RED_WRT：**读/写标志位。该标志位是和上一个地址信息的读写标志位相匹配的，只在该区间有效，当下一个 START 命令、STOP 命令或者 NO ACK 命令出现时，自动无效。

从动模式下：0 = 写；1 = 读。
主控模式下：0 = 没有发送；1 = 发送过程中。
该位和 SEN、RSEN、PEN、RCEN、ACKEN 位指明 MSP 是否在 IDLE 状态。

Bit 0 **BF：**缓存器状态位。
接收：1 = 接收完成，MSPBUF 填满；
0 = 接收没有完成，MSPBUF 空置。
发送：1 = 正在发送数据（不包括 ACK 和停止位），MSPBUF 填满；
0 = 完成发送数据（不包括 ACK 和停止位），MSPBUF 空置。

12.3 MSP模式寄存器 1

0EBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPM1	WCOL	MSPOV	MSPENB	CKP	SLRXCKP	MSPWK	-	MSPC
读/写	R/W	R/W	R/W	R/W	R/W	R/W	-	R/W
复位后	0	0	0	0	0	0	-	0

Bit 7 **WCOL**：写数据检测标志位。
主控模式：0 = 未检测到有数据写入 MSPBUF；
1 = I2C 通讯还未开始，已经写入数据到 MSPBUF 寄存器，此位被置 1。
从动模式：0 = 未检测到有数据写入 MSPBUF；
1 = 正在发送上一个数据时，有写入新的数据到 MSPBUF。（必须由软件清零）

Bit 6 **MSPOV**：接收溢出显示位。
0 = 没有溢出；
1 = 当 MSPBUF 仍在处理前一个数据时，接收到新数据，此时此位被置 1，发送模式下 MSPOV 可以忽略，任一模式下都必须在软件下将该位清零。

Bit 5 **MSPENB**：MSP 通信使能位。
0 = 禁止 MSP，相关引脚为普通 I/O 引脚；
1 = 使能 MSP，相关引脚为 SCL、SDA 串口引脚。

注：禁止 MSP 后，清 MSP 状态寄存器。故用户在使能 MSP 之前需重新设置 MSP 寄存器。

```
Ex:B0bclr FMSPENB
Call MSP_init_setting
     B0bset FMSPENB
```

Bit 4 **CKP**：SCL 时钟优先控制位。
MSP 从动模式：
0 = SCL 保持低电平（保证数据设置时间和从动设备准备就绪）；
1 = 释放 SCL 时钟（从动发送模式时时钟使能 CKP 功能，从动接收模式时由 SLRXCKP 控制 CKP 功能）。
MSP 主控模式：无效。

Bit 3 **SLRXCKP**：从动接收模式时 SCL 时钟优先控制位。
MSP 从动接收模式：
0 = 禁止 CKP 功能；
1 = 使能 CKP 功能。
MSP 主控和从动发送模式：无效。

Bit 2 **MSPWK**：MSP 唤醒显示位。
0 = MSP 没有唤醒单片机；
1 = MSP 唤醒单片机。

✷ **注：在进入睡眠模式之前将 MSPWK 清零，以便在唤醒后知道是否由 MSP 唤醒。**

Bit 0 **MSPC**：MSP 模式控制位。
0 = MSP 在从动模式下工作，7 位地址；
1 = MSP 在主控模式下工作。

12.4 MSP模式寄存器 2

0ECH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPM2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **GCEN**：通用地址应答使能位（仅在从动模式下有效）。
0 = 禁止通用地址应答功能；
1 = 使能通用地址应答功能和中断功能。

Bit 6 **ACKSTAT**：应答状态位（仅在主控模式下有效）。
主控发送模式下：
0 = 接收到从动端的应答；
1 = 没有接收到从动端的应答。

Bit 5 **ACKDT**：应答数据位（仅在主控模式下有效）。
主控接收模式下：在接收到一个字节数据后，执行应答操作时，此位的数据将被发送。
0 = 应答；
1 = 无应答。

Bit 4 **ACKEN**：应答流程使能位（仅在主控模式下有效）。
主控接收模式下：
0 = 应答流程空闲；
1 = 在 SDA 和 SCL 引脚开始应答流程，发送 ACKDT 位，自动由硬件清零。

Bit 3 **RCEN**：接收使能位（仅在主控模式下有效）。
0 = 接收空闲；
1 = 使能 MSP 接收模式。

Bit 2 **PEN**：停止条件使能位（仅在主控模式下有效）。
0 = 停止条件空闲；
1 =使能 SCL/SDA 发送停止信号，由硬件自动清零。

Bit 1 **RSEN**：重复起始条件使能位（仅在主控模式下有效）。
0 = 重复起始条件空闲；
1 = SCL/SDA 重新发送 START 信号，由硬件自动清零。

Bit 0 **SEN**：起始条件使能位（仅在主控模式下有效）。
0 = 起始条件空闲；
1 =使能 SCL/SDA 发送 START 信号，由硬件自动清零。

12.5 MSP MSPBUF寄存器

MSPBUF 初始值 = 0000 0000

0EDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPBUF	MSPBUF7	MSPBUF6	MSPBUF5	MSPBUF4	MSPBUF3	MSPBUF2	MSPBUF1	MSPBUF0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

12.6 MSP MSPADR寄存器

MSPADR 初始值 = 0000 0000

0EEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPADR	MSPADR7	MSPADR6	MSPADR5	MSPADR4	MSPADR3	MSPADR2	MSPADR1	MSPADR0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:1] 7 位地址。

Bit 0 Tx/Rx 模式控制位。
0=Tx 模式；
1=Rx 模式。

12.7 从机模式操作

当地址数据被发送或地址接收到，硬件将自动产生 ACK 信号，将接收到的数据从 MSPSR 装载到 MSPBUF（MSP 缓存寄存器）中。

在下列情形下，MSP 功能不会应答 ACK_信号：

- **数据缓存器全满：接收到另一组发送的内容，BF = 1（MSPSTAT bit 0）。**
- **数据溢出：接收到另一组发送的内容，MSP0V=1（MSPM1 bit 6）。**

BF=1，即 MCU 还未读取 MSPBUF 的数据，故 MSPSR 中的数据还未装入 MSPBUF，但此时 MSPIRQ 和 MSPOV 位仍置 1，当读取 MSPBUF 寄存器后，BF 自动被清零，MSPOV 必须由软件清零。

12.7.1 寻址

使能 MSP 从动功能时，需等待产生一个 START 信号，START 信号产生后，8 位地址装入 MSPSR 寄存器，MSPSR[7:1] 的数据与 MSPADR 寄存器在 SCL 脉冲的第 8 个下降沿进行比较，若地址相同，BF 和 MSPOV 都清零，产生下列事件：

- 在 SCL 的第 8 个下降沿，MSPSR 寄存器的数据装入 MSPBUF。
- 在 SCL 的第 8 个下降沿，缓存器状态位 BF 置 1。
- 产生一个 ACK_信号。
- 在 SCL 的第 9 个下降沿，MSP 中断请求 MSPIRQ 置 1。

接收数据时的状态		MSPSR→ MSPBUF	响应 ACK_信号	设置 MSPIRQ
BF	MSPOV			
0	0	是	是	是
*0	*1	是	否	是
1	0	否	否	是
1	1	否	否	是

接收数据效果表

✷ **注：BF=0，MSPOV=1 表示软件没有清除数据溢出标志位。**

12.7.2 从机接收

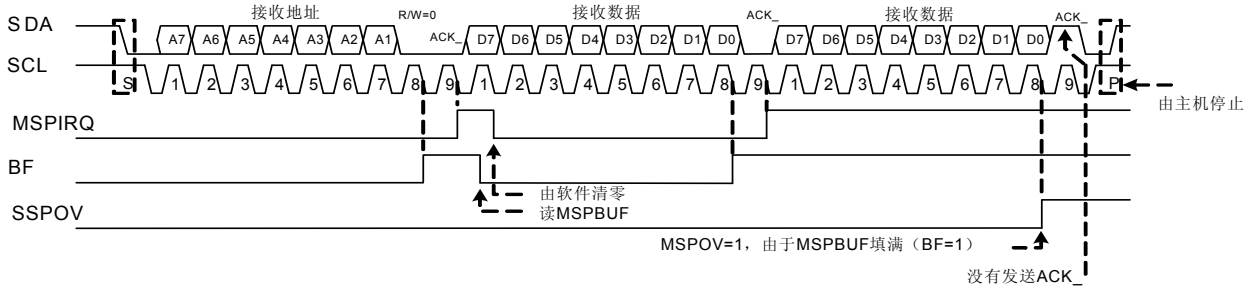
当硬件地址匹配，且 R/W 位=0，那么 MSPSTAT 中的 RED_WRT 标志位被清零。同时地址载入 MSPBUF，在给出应答信号后，MSP 将每 8 个 CLK 接收一次数据，由 SLRXCKP 位控制是否使能 CKP 功能，由 CKE 位控制数据的边沿锁存功能－上升沿或下降沿。

当溢出时，不管 BF=1 还是 MSPOV=1，都没有应答信号。

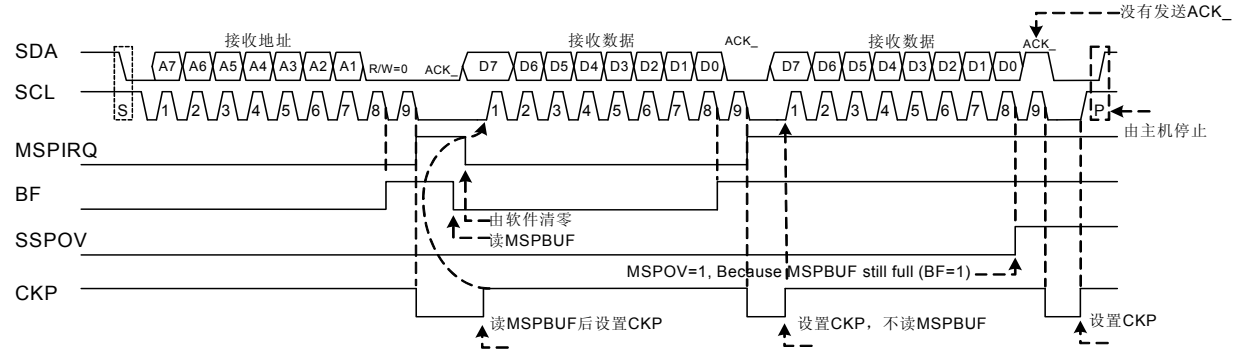
每发送一次数据会产生一次 MSP 中断，必须由软件清 MSPIRQ。

从动模式接收示意图如下所示：

SLRXCKP=0



SLRXCKP=1

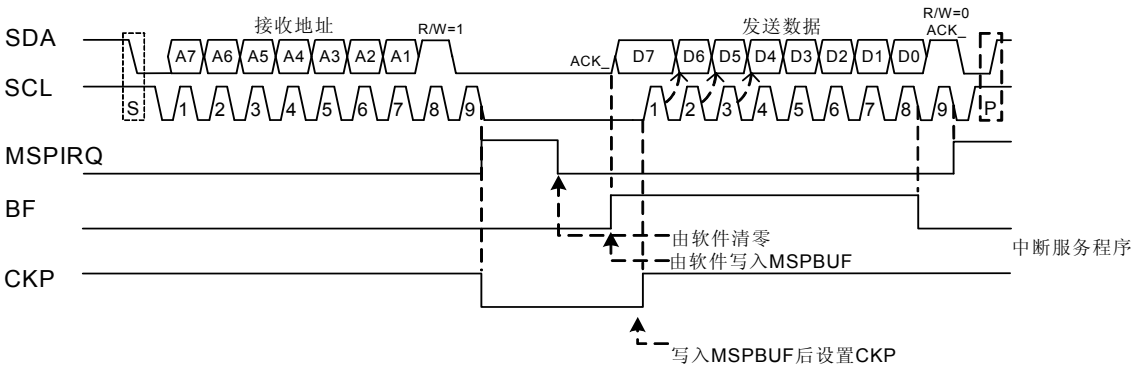


12.7.3 从机发送

匹配地址后，设置 R/W 位，MSPSTAT bit 2 RED_WRT 置 1，接收到的地址装入 MSPBUF，然后在第 9 个时钟发送一个 ACK 信号，SCL 保持低电平。发送的数据装入 MSPBUF，MSPBUF 装入 MSPSR 寄存器。主控装置监控 SCL 引脚信号，从动设备保持 CKP 低电平。装入 MSPBUF 后，设置 CKP 位，MSPBUF 数据在 SCL 信号的下降沿溢出，确保 SDA 信号在 SCL 高电平状态是有效的。

每个字节发送时产生 MSP 中断，中断请求 MSPIRQ 在第 9 个 SCL 时钟置 1，由软件清零。MSPSTAT 寄存器可以监控数据发送的状态。

从动发送模式下，从主机接收的 ACK_信号在第 9 个 SCL 时钟的上升沿被锁存，如 ACK_为高电平，发送完成，从动设备逻辑复位并等待另一个 START 信号。如 ACK_为低电平，从设备应该重新导入 MSPBUF，设置 CKP=1，重新开始发送数据。



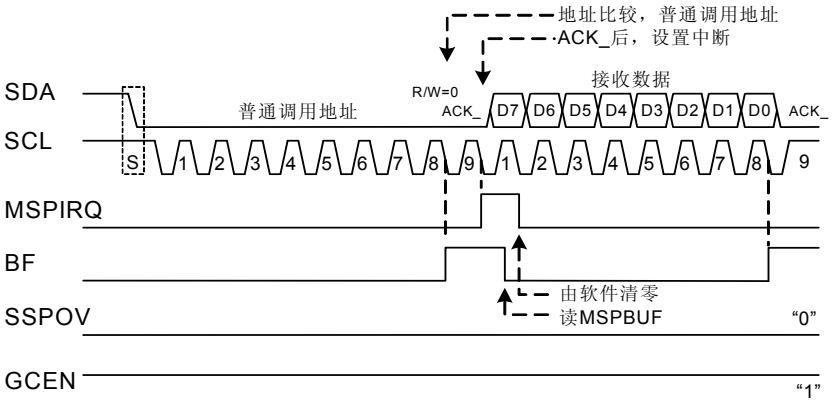
MSP 从动模式发送时序图

12.7.4 通用地址调用

MSP 总线的第一个字节的前 7 位为从动地址，只有该地址和 MSPADR 的数据相符时才会响应一个 ACK_信号。现有一个通用地址可以作为所有从动设备的硬件地址，发生该地址时，所有的从动设备都响应一个应答信号。

通用地址是一个全为 0 的特殊地址。通用地址功能由 GCEN 位控制。该位置 1 则使能通用地址功能，反之则禁止。当 GCEN=1，START 信号后，8 位地址装入 MSPSR，并与 MSPADR 进行比较，且通用地址由硬件固定。

如通用地址匹配，MSPSR 数据发送至 MSPBUF，BF 置 1，在第 9 个时钟（ACK_）下降沿请求 MSP 中断，执行中断时，读取 MSPBUF，检测该地址是否为通用地址或者特殊地址。

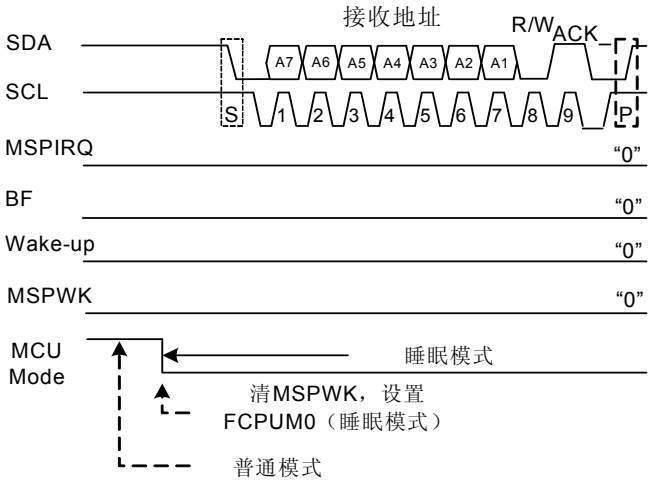


通用地址调用时序图

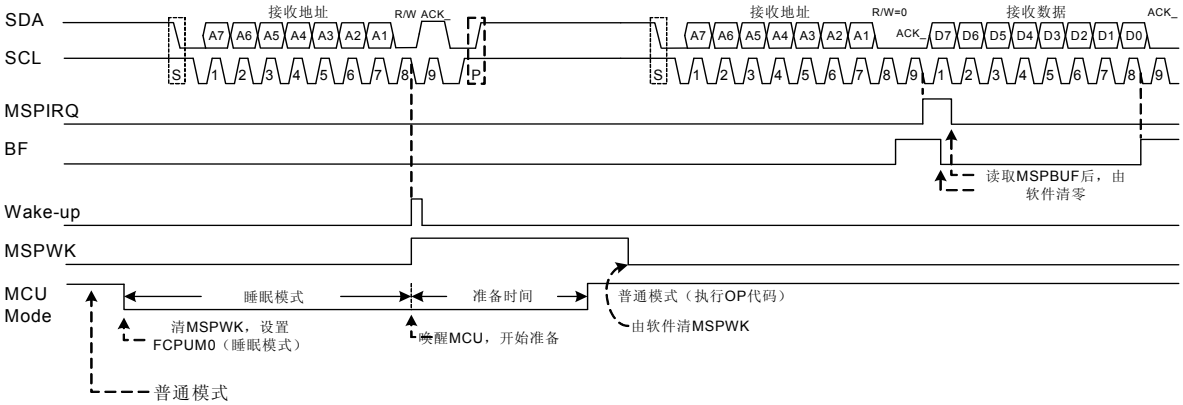
12.7.5 从机模式唤醒

睡眠模式下，若 MSPENB 置 1，则匹配的设备地址可以将单片机唤醒。

8 位 MSP 总线地址跟随 START 之后，装入 MSPSR，若地址匹配，在第 9 个 SCL 时钟时响应一个 NO ACK 信号，单片机被唤醒，MSPWK 置位，开始唤醒处理，但 MSPIRQ 不会置 1，且 MSPSR 数据也不会装入 MSPBUF。单片机被唤醒后，MSP 处于空闲状态，并等待主机的 START 信号。进入睡眠模式之前控制寄存器 BF、MSPIRQ、MSPOV 和 MSPBUF 的状态/数据相同。若地址不匹配，在第 9 个 SCL 时钟时发送一个 NO ACK 信号，但不会唤醒单片机，系统仍处于睡眠状态。



MSP 唤醒时序示意图：地址不匹配



MSP 唤醒时序示意图：地址匹配

➢ **例：**

```
B0BSET      FCPUM0
B0BCLR      FMSPENB
NOP
B0BSET      FMSPENB
MOV         A, #0Xnn
B0MOV       MSPADR, A       ; 重新写入 I2C 的从动地址。
```

✷ **注 1：MSP 功能只能在普通模式下工作，当系统从睡眠模式被唤醒后，单片机必须在主机发送 START 信号之前进入普通模式下进行工作。**

✷ **注 2：MSP 唤醒时，若地址不匹配，则单片机仍处于睡眠模式。**

✷ **注 3：在进入睡眠模式之前，先由软件清 MSPWK，以显示唤醒状态。**

12.8 主控模式

MSP 主控模式的操作从 START 信号开始，STOP 信号结束。
系统复位或者禁止 MSP 功能时，START（S）和 STOP（P）被清零。
主控模式下，SCL 和 SDA 由 MSP 硬件控制。

MSPIEN 置 1，请求中断时，若发生下列情形的事件，则 MSP 中断请求（MSPIRQ）可以置 1：

- **START 情形。**
- **STOP 情形。**
- **发送/接收数据字节。**
- **发送应答信号。**
- **重复 START。**

12.8.1 主控模式支持格式

MSPC 和 MSPENB 置位时使能 MSP 主控模式，有 6 种情形：

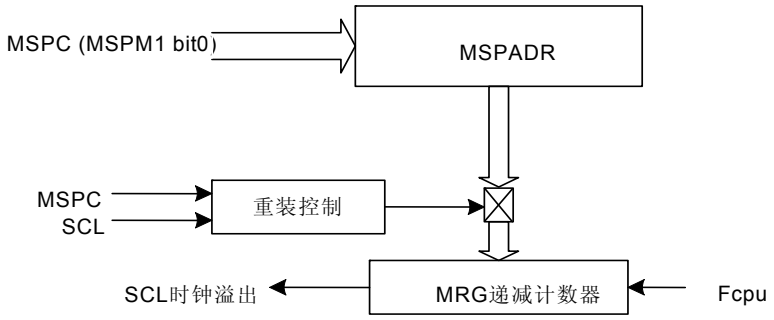
- 在 SCL 和 SDA 线上发送一个 START 信号。
- 在 SCL 和 SDA 线上发送一个重复的 START 信号。
- 将发送的数据和地址字节写入 MSPBUF 寄存器中。
- 在 SCL 和 SDA 线上发送一个 STOP 信号。
- 配置 MSP 端口以接收数据。
- 在接收数据的末端发送一个应答信号。

12.8.2 MSP速率产生器

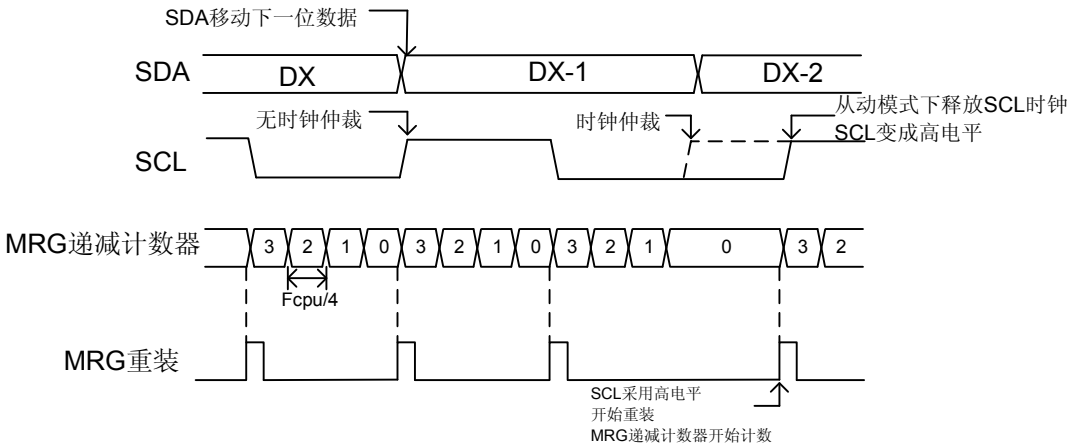
MSP 模式下，MSP 速率产生器的重装值位于 MSPADR 寄存器的低 7 位地址。当 MRG 装入寄存器，MRG 计数到 0，到另一个重装值时停止。在 MSP 主控模式下，MRG 自动重装 MSPADR 的值。若时钟仲裁发送（如 SCL 引脚由从动设备保持低电平），当 SCL 引脚检测到高电平时 MRG 重新装载。

SCL 时钟速率 = Fcpu/(MSPADR)*2

例如：设置 400KHz 4MHz Fcpu，MSPADR 必须设置为 05H
MSPADR=4Mhz/400Khz*2=5



MSP 速率产生器示意框图



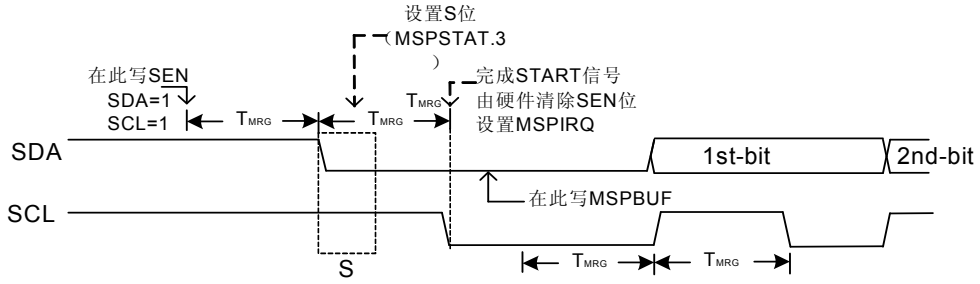
MRG 时序图，有无时钟仲裁（MSPADR=03H）

12.8.3 MSP主控模式START操作

为产生一个 START 信号，用户可设置 SEN 位（MSPM2.0），当 SDA 和 SCL 都为高时， MSP 速率产生器将重装 MSPADR[6:0]然后开始递减计数，当 SDA 和 SCL 都为高且 MRG 溢出时，SDA 从高变为低，此时位 S (MSPSTAT.3)将被置 1。MRG 重新装载和计数，当 MRG 溢出时，位 SEN 将自动清零，MRG 挂起，SDA 保持低电平，完成一次完整的 START。

12.8.3.1WCOL状态标志

若用户在处理 START 时写入数据到 MSPBUF，WCOL 置位，MSPBUF 的数据不作任何改变（写动作无效）。



START 操作时序示意图

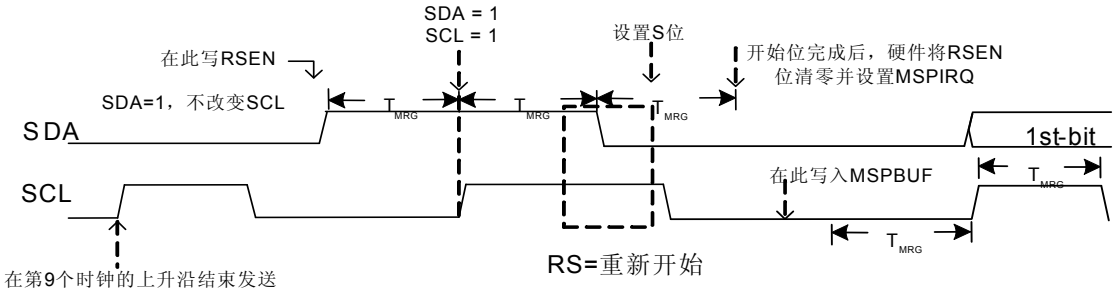
12.8.4 MSP主控模式重复START操作

当 MSP 逻辑模块空闲且 RSEN 置 1 时，重复 START 操作。RSEN 置 1 且 SCL 引脚为低电平时，MSPADR[6:0]数据重装到 MSP 速率产生器中并开始往下计数。SDA 引脚在 MSP 速率发生计数器（T_{MRG}）释放高电平。当 MRG 溢出，且 SDA 为高电平时，SCL 引脚输出高电平。当 SCL 为高电平，MSPADR 重装到 MRG 并开始往下计数时，SDA 和 SCL 必须在 T_{MRG} 期间保持高电平。在下一个 T_{MRG} 期间，SCL 采用高电平时，SDA 输出低电平，RSEN 由硬件自动清零，MRG 不会重装，SDA 保持低电平。一旦检测到 SDA 和 SCL 发生 START 操作，S 位被置 1（MSPSTAT.3），MRG 溢出时，MSPIRQ 置 1。

- **注 1：在处理过程中发生任何其它的事件，设置 RSEN 以保证不造成影响。**
- **注 2：在重复 START 操作时会有总线冲突；SCL 开始输出高电平时，SDA 为低电平。**

12.8.4.1WCOL状态标志

若用户在重复 START 时写入数据到 MSPBUF，WCOL 置位，MSPBUF 的数据不作任何改变（写动作无效）。



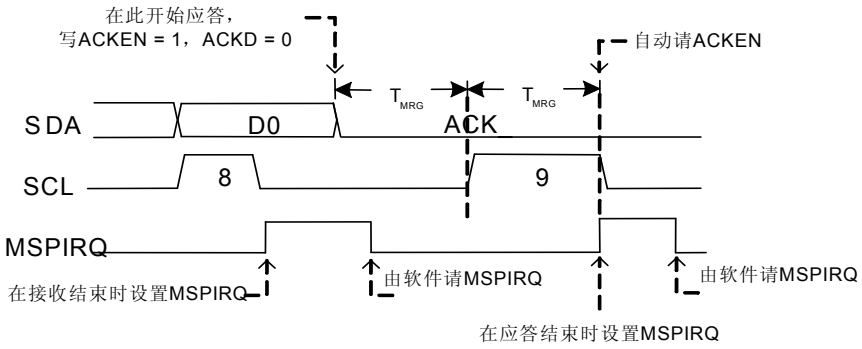
重复 START 操作时序示意图

12.8.5 应答流程时序

设置 ACKEN（MSPM2.4）时使能应答流程，SCL 输出低电平，应答数据位在 SDA 引脚为目前状态。若用户想要响应一个应答信号，需将 ACKDT 位清零，如若不然，在开始应答流程之前，置 ACKDT 位。MSP 速率发生器溢出时，释放 SCL 引脚（输出高电平）。SCL 为高电平时，MSP 速率发生器开始 T_{MRG} 周期，往下计数。此段周期结束后，SCL 输出低电平，ACKEN 位被硬件自动清零。MRG 再次溢出时，MSP 进入空闲模式。

12.8.5.1WCOL状态标志

若用户在处理应答流程时写入数据到 MSPBUF，WCOL 置位，MSPBUF 的数据不作任何改变（写动作无效）。



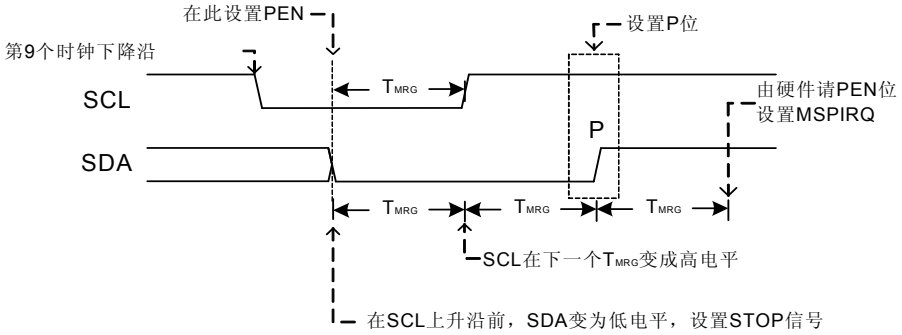
应答流程时序示意图

12.8.6 STOP操作时序

发送/接收结束时，通过设置 STOP 位 PEN（MSPM2.2），在 SDA 引脚出现 STOP 信号。发送/接收结束时，SCL 在第 9 个时钟的下降沿开始输出低电平。位 PEN 为 1 时，SDA 输出低，此时 MSP 速率发生器重装并递减计数。MRG 溢出时，SCL 引脚输出高电平。一个 T_{MRG} 周期结束后，SDA 输出高电平。SCL 输出高电平时，SDA 为高电平，P 位置 1，下一个 T_{MRG} 周期结束后，PEN 位被清零，MSPIRQ 置 1。

12.8.6.1WCOL状态标志

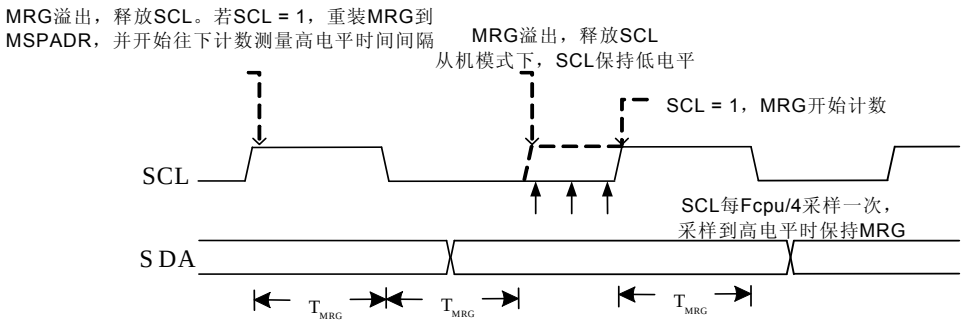
若用户在处理 STOP 时写入数据到 MSPBUF，WCOL 置位，MSPBUF 的数据不作任何改变（写动作无效）。



STOP 操作流程时序示意图

12.8.7 时钟仲裁

当主机接收/发送任何数据或者重复 START、STOP 信号时，产生时钟仲裁，此时 SCL 引脚处于高悬浮状态，MSP 速率发生器（MRG）一直挂起，直到 SCL 为高电平。MRG 重装 MSPADR[6:0]，并开始往下计数，保证 SCL 高电平时间至少为一个 MRG 溢出时间，时钟由外部设备控制保持为低电平。



时钟仲裁流程时序示意图

12.8.8 主控模式发送

完全写入 MSPBUF 寄存器，表示一个数据字节、7 位地址或者 8 位数据发送完成，该操作设置缓存器 BF 标志，并允许 SMP 速率发生器开始计数。

写入 MSPBUF 后，地址的每一位都将在 SCL 的下降沿移出，直到 7 位地址和 R/W 位都完全移出。在时钟的第 8 个下降沿，主机将 SDA 拉低并给从机一个应答信号。在时钟的第 9 个下降沿，采样的 SDA 显示已经由从机接收的地址。ACK 的状态装入 ACKSTAT，MSPIRQ 置 1，BR 清零，MRG 持续直到另外的数据写入 MSPBUF，SCL 保持低电平，并允许 SDA 悬浮。

12.8.8.1BF状态标志

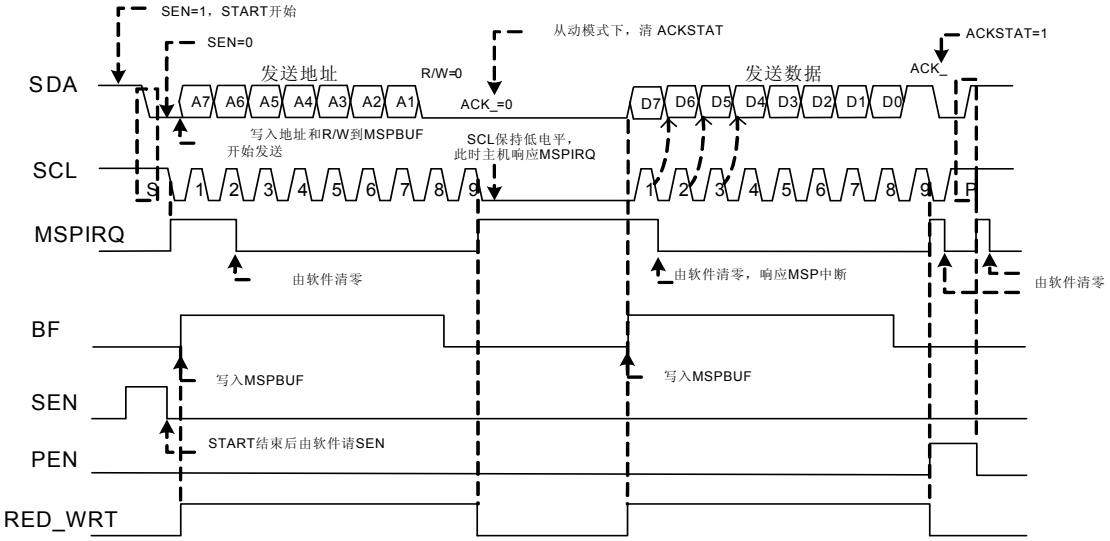
发送模式下，BF 位在数据写入 MSPBUF 时置 1，8 位数据全部移出时自动清零。

12.8.8.2WCOL标志

若用户在处理发送流程时写入数据到 MSPBUF，WCOL 置位，MSPBUF 的数据不作任何改变。

12.8.8.3ACKSTAT状态标志

发送模式下，从机发送应答信号后（ACK_=0）ACKSTAT 清零，没有发送应答信号（ACK_=1），ACKSTAT 则置 1。识别地址（包括普通调用地址）时，或者已经完全接收数据后，从机发送一个应答信号。



MSP 主机发送模式时序示意图

12.8.9 主控模式接收

主控接收模式由 RCEN 位设置。

SCL 为高电平时，MRG 开始计数，数据存入 MSPSR。在第 9 个时钟的下降沿时，接收使能位 RECN 被自动清零，MSP 的内容转托 MSPBUF，BF 位置 1，MSPIRQ 置 1，MRG 计数器挂起，SCL 保持低电平。MSP 处于空闲状态，并等待下一个操作命令。当由软件读取 MSPBUF 的数据后，BF 被自动清零。通过设置 ACKEN 位，用户可以在接收结束时发送应答信号。

12.8.9.1BF状态标志

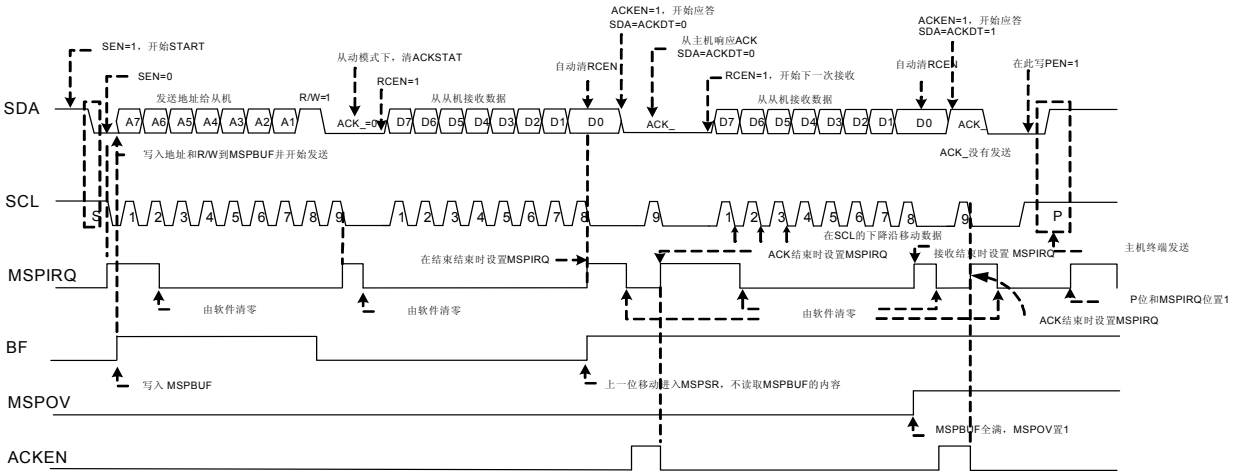
接收模式下，BF 在地址或者数据字节装入 MSPBUF 后被置 1，MSPBUF 的内容被读取后则自动清零。

12.8.9.2MSPOV标志

接收模式下，接收到另一个 8 位数据装入 MSPSR 后，MSPOV 置 1，BF 从前一次接收就一直置 1。

12.8.9.3WCOL标志

若用户在处理已经接收到的数据时写入数据到 MSPBUF，WCOL 置位，MSPBUF 的数据不作任何改变。



MSP 主机接收模式时序示意图

13 FLASH ROM在线烧录

13.1 概述

SN8F27E65 系列单片机内置可编程（ISP）的 FLASH 存储器以方便 CODE 升级。FLASH 存储器可经由 SONiX 8 位单片机编程界面或者应用 code 进行编程。SN8F27E65 系列单片机提供安全的选项以保护用户代码的安全性。ISP Flash ROM 给用户提供了一种存储数据到 Flash ROM 的简单方法，就是将 RAM 缓存器中的内容转移到 Flash ROM 中。选择 ROM/RAM 地址，执行 ROM 编程命令-PECMD，编程字（由 PERAMCNT 控制）后，PERAML/PERAMCNT 数据都被送入了地址 PEROML/PEROMH 处。

	RAM (byte)			Flash ROM (word)		
RAM 地址	bit7 ~ bit0		ROM 地址	bit15 ~ bit8	bit7 ~ bit0
X	DATA0		Y	DATA1	DATA0
X+1	DATA1		Y+1	DATA3	DATA2
X+2	DATA2	=>	Y+2		
X+3	DATA3		Y+3		
…			…		
…			…		
X+N	DATAN		Y+M	DATAN	DATAN-1

在 Flash 编程或擦除时，即使外围设备（定时器、WDT、I/O、PWM 等）都在正常工作，单片机也会停止工作。当 PECMD 寄存器置 1 执行 ISP 编程或擦除动作时，程序计数器停止计数，OP-Code 不能存入 Flash ROM，指令停止工作，程序停止执行。此时根据 ISP 操作（Flash ROM 擦除或者编程）自动决定硬件。ISP 操作完成后，硬件释放出系统时钟使程序计数器开始工作，系统返回到上一个工作模式，并执行下一条指令。建议在执行 ISP 操作前添加两条“NOP”指令。

- **ISP flash ROM 擦除时间 = 27ms……1-page, 128-word.**
- **ISP flash ROM 编程时间 = 28us……1-word.**
- **ISP flash ROM 编程时间 = 56us……2-word.**
  …
- **ISP flash ROM 编程时间 = 448us……16-word.**
  …
- **ISP flash ROM 编程时间 = 896us……32-word.**

✷ 注：
1、需要在执行 Flash ISP 操作（编程或擦除）前清看门狗定时器，否则看门狗会在 ISP 操作期间溢出导致系统复位。
2、除了编程操作之外，其他的功能（如定时器、ADC、SIO、UART、MSP 等）都在 ISP 操作期间正常工作。所有的中断触发事件有效，自动锁存中断标志。如果在 ISP 操作期间，有中断请求发生，会在 ISP 完成后响应该中断。

13.2 ISP FLASH ROM擦除操作

ISP flash ROM 擦除操作指将 Flash ROM 的内容清空为 1，擦除 ROM 的长度为 128 字，有 ROM 页的限制。ISP flash ROM 擦除 ROM 如下所示：

ISP ROM MAP		ROM 地址 bit0~bit6 (hex)														
		0000	0001	0002	…	0010	0011	…	0050	0051	…	0070	0071	…	007E	007F
ROM address bit7~bit15 (hex)	0000	**该页包括复位向量和中断向量。强烈建议将该页保留而不要在该页进行 ISP 擦除操作。**														
	0080	ISP 擦除页														
	0100	ISP 擦除页														
	0180	ISP 擦除页														
	0200	ISP 擦除页														
	0280	ISP 擦除页														
	…	ISP 擦除页														
	0F00	ISP 擦除页														
	0F80	ISP 擦除页														
	1000	ISP 擦除页														
	1080	ISP 擦除页														
	1100	ISP 擦除页														
	1180	ISP 擦除页														
	…	ISP 擦除页														
	1600	ISP 擦除页														
	1680	ISP 擦除页														
	1700	ISP 擦除页														
	1780	**该页包括 ROM 保留区域。强烈建议将该页保留而不要在该页进行 ISP 擦除操作。**														

ISP flash ROM 擦除密度为 128 字，限定了擦除页的边界。Flash ROM 的第一个 128 字（0000H~0007FH）包括复位向量和中断向量，这里关系到基本的程序操作；flash ROM 的最后 128 字（1780H~17FFH）包括系统保留 ROM 区域，强烈建议不要在这两页执行 ISP Flash ROM 擦除操作。Flash ROM 区域（0008H~177FH）包括 46 页，可以执行 ISP flash ROM 擦除操作。

执行 ISP flash ROM 擦除操作的第一步是指定 ROM-page 地址，地址必须要在每页的开始地址，如 80H、100H、180H、…1600H、1680H 和 1700H。PEROML[7:0]和 PEROMH[7:0]定义 flash ROM 的开始地址[15:0]。写入开始地址到 PEROML 和 PEROMH 寄存器，设置 PECMD 寄存器为 0C3H，系统开始执行 ISP flash ROM 擦除操作。

➢ **例：执行 ISP flash ROM 擦除操作，清除 flash ROM 0080H~00FFH 地址的内容。**

```
; 设置擦除的开始地址为 0080H。
        MOV     A, #80H
        B0MOV   PEROML, A       ; 转移低字节地址 80H 到 PEROML。
        MOV     A, #00H
        B0MOV   PEROMH, A       ; 转移高字节地址 00H 到 PEROMH。

; 清看门狗定时器。
        MOV     A,#5AH
        B0MOV   WDTR,A

; 开始执行 ISP flash ROM 擦除操作。
        MOV     A,#0C3H         ; 开始擦除。
        B0MOV   PECMD, A
        NOP                     ;
        NOP
                                ; ISP flash ROM 擦除结束。
```

ISP flash ROM 擦除操作结束后，两条 NOP 指令可以提供一个短暂的延迟时间等待系统稳定。

✷ **注：不能在 flash ROM 的第一页和最后一页执行 ISP flash ROM 擦除操作，否则会影响程序的执行。**

13.3 ISP FLASH ROM编程操作

ISP flash ROM 编程操作是指将数据通过程序写入 flash ROM 中，不能限制写入数据的地址和长度，但是限定了每页 32 字的密度。执行 ISP flash ROM 编程操作时，可以每次写入 1 字~32 字，但必须在同一页中。ISP flash ROM 编程 ROM 如下所示：

ISP ROM MAP		ROM 地址 bit0~bit4 (hex)								
		0000	0001	0002	…	000F	0010	…	001E	001F
ROM address bit5~bit15 (hex)	0000	该页包括复位向量和中断向量。强烈建议将该页保留而不要在该页进行 ISP 擦除操作。								
	0020	ISP 编程页								
	0040	ISP 编程页								
	0060	ISP 编程页								
	0080	ISP 编程页								
	00A0	ISP 编程页								
	00C0	ISP 编程页								
	00E0	ISP 编程页								
	0100	ISP 编程页								
	0120	ISP 编程页								
	…	ISP 编程页								
	1000	ISP 编程页								
	1020	ISP 编程页								
	…	ISP 编程页								
	1700	ISP 编程页								
	1720	ISP 编程页								
	…	ISP 编程页								
	17E0	该页包括 ROM 保留区域。强烈建议将该页保留而不要在该页进行 ISP 擦除操作。								

ISP flash ROM 编程密度为 32 字，限定了编程页的边界。Flash ROM 的第一个 32 字（0000H~0001FH）包括复位向量和中断向量，这里关系到基本的程序操作；flash ROM 的最后 32 字（17E0H~17FFH）包括系统保留 ROM 区域，强烈建议不要在这两页执行 ISP Flash ROM 编程操作。Flash ROM 区域（0020H~17DFH）包括 190 页，可以执行 ISP flash ROM 编程操作。

ISP flash ROM 编程操作是一个简单的存储器映射操作。执行 ISP flash ROM 编程操作的第一步是计划一个 RAM 区域，用来存储需要编程的数据，并保持 RAM 地址为 ISP RAM 地址；第二步是指定一个 ROM 区域，将需要编程的数据从 RAM 区域转移到 flash ROM 中，执行 ISP flash ROM 编程操作。通过 PERAML[9:0] 10 位缓存器决定 RAM 的开始地址，RAM 的数据存储流程为 down-up 结构。RAM 的第一个数据为 Flash ROM 的第一个字的低字节，RAM 的第二个数据为 flash ROM 的第一个字的高字节，以此类推。

ISP的编程长度为1字~32字，由PERAMCNT[7:3]控制。在执行ISP ROM编程操作之前，由程序设置好编程长度。PEROML[7:0]和PEROMH[7:0] 定义flash ROM的开始地址[15:0]。写入开始地址到PEROML和PEROMH寄存器，设置PECMD寄存器为5AH，系统开始执行ISP flash ROM编程操作。如果编程长度超过的ISP Flash ROM编程页的边界，在完成ISP flash ROM编程页的最后一个字的编程动作后，硬件立即停止随后的编程动作。故在编程前必须计划合适的ROM地址和编程长度。

- **Case 1：**32字ISP程序，RAM缓存器长度为64字节，RAM地址为X~X+63。PERAMCNT[7:3]=11111B，表示完成flash ROM的一页32字。Flash ROM的页地址为Y~Y+31，Y指开始地址，设置PEROML、PEROMH寄存器。

RAM (byte)

RAM 地址 64 字节	bit7 ~ bit0
X	DATA0
X+1	DATA1
X+2	DATA2
X+3	DATA3
…	…
X+62	DATA62
X+63	DATA63

=>

Flash ROM (word)

ROM 地址 32 字	bit15 ~ bit8	bit7 ~ bit0	
Y	DATA1	DATA0	页的开始处。ISP 的开始地址。
Y+1	DATA3	DATA2	
Y+2	…	…	
Y+3	…	…	
…	…	…	
…	…	…	
Y+31	DATA63	DATA62	页的结束处。ISP 的结束地址。

- **Case 2:** 16字ISP程序，RAM缓存器长度为32字节。PERAMCNT[7:3]=01111B，表示完成flash ROM的一页16字。Flash ROM的页地址为Y~Y+31，但开始地址并不位于页的开始处。定义开始地址为Y+10，设置PEROML、PEROMH寄存器，可编程的ROM区域为Y+10~Y+25。

RAM (byte)

RAM 地址 32 字节	bit7 ~ bit0
X	DATA0
X+1	DATA1
X+2	DATA2
X+3	DATA3
…	…
X+30	DATA30
X+31	DATA31

=>

Flash ROM (word)

ROM 地址 32 字	bit15 ~ bit8	bit7 ~ bit0	
Y			页的开始处。
Y+1			
…			
Y+10	DATA1	DATA0	ISP 的开始地址。
Y+11	DATA3	DATA2	
…	…	…	
Y+25	DATA31	DATA30	ISP 的结束地址。
…			
Y+30			
Y+31			页的结束处。

- **Case 3：**在上面Case的基础上，更改ROM的开始地址为Y+20。可编程的flash ROM地址为Y+20~Y+35，ROM的范围超过的编程页的边界。ISP flash ROM操作完成后，最后的4字不能成功的写入flash ROM中。编程长度超过了ISP flash ROM编程页的边界，在完成ISP flash ROM编程页的最后一个字（Y+31）的编程动作后，硬件立即停止随后的编程动作。

RAM (byte)

RAM 地址 32 字	bit7 ~ bit0
X	DATA0
X+1	DATA1
X+2	DATA2
X+3	DATA3
…	…
X+30	DATA30
X+31	DATA31

=>

Flash ROM (word)

ROM 地址 32 字	bit15 ~ bit8	bit7 ~ bit0	
Y			页的开始处。
Y+1			
…			
Y+20	DATA1	DATA0	ISP 的开始地址。
Y+21	DATA3	DATA2	
…	…	…	
Y+30	DATA21	DATA20	
Y+31	DATA23	DATA22	页的结束处。 ISP 的结束地址。

➢ **例：执行 ISP flash ROM 编程操作，将 32 字数据写入 flash ROM（如 case1）。设置 RAM 缓存器的开始地址为 10H，flash ROM 编程的开始地址为 20H。**

```
; 加载数据到 64 字节 RAM 缓存器。
        …
        …

; 设置 64 字节 RAM 缓存器的开始地址。
        MOV     A, #10H
        B0MOV   PERAML, A           ; 设置 PERAML[7:0]为 10H。
        MOV     A, #00H
        B0MOV   PERAMCNT, A         ; 设置 PERAML[9:8]为 00H。

; 设置 ISP 编程长度为 32 字。
        MOV     A, #11111000b
        OR      PERAMCNT, A         ;

; 设置 flash ROM 编程的开始地址为 20H。
        MOV     A, #20H
        B0MOV   PEROML, A           ; 移动低字节地址 20H 到 PEROML。
        MOV     A, #0x00
        B0MOV   PEROMH, A           ; 移动高字节地址 00H 到 PEROMH。

; 清看门狗定时器。
        MOV     A,#5AH
        B0MOV   WDTR,A

; 开始执行 ISP flash ROM 编程操作。
        MOV     A,#5AH              ; 开始 flash ROM 编程。
        B0MOV   PECMD, A
        NOP                         ;
        NOP
                                    ; ISP flash ROM 编程结束。
```

ISP flash ROM 编程操作结束后，两条 NOP 指令可以提供一个短暂的延迟时间等待系统稳定。

✷ **注：不能在 flash ROM 的第一页和最后一页执行 ISP flash ROM 编程操作，否则会影响程序的执行。**

13.4 ISP编程/擦除控制寄存器

0DBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PECMD	PECMD7	PECMD6	PECMD5	PECMD4	PECMD3	PECMD2	PECMD1	PECMD0
读/写	W	W	W	W	W	W	W	W
复位后	-	-	-	-	-	-	-	-

Bit [7:0] **PECMD [7:0]**：ISP 操作控制寄存器。
5AH：编程页（32 字/页）；
0C3H：擦除页（128 字/页）；
其他：系统保留。

✷ **注：执行 ISP 编和擦除操作之前，必须先清 PECMD 寄存器；ISP 相关设置完成后，在指令“MOV A,I“和”B0MOV M,A”指令中设置 ISP 操作代码开始 ISP 操作。**

13.5 ISP ROM地址寄存器

ISP ROM地址的长度为16位，分别位于PEROML和PEROMH寄存器中。执行ISP之前，用程序设置ISP ROM的开始地址。

0DCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEROML	PEROML7	PEROML6	PEROML5	PEROML4	PEROML3	PEROML2	PEROML1	PEROML0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **PEROML[7:0]**：ISP ROM 地址的低字节缓存器。

0DDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEROMH	PEROMH7	PEROMH6	PEROMH5	PEROMH4	PEROMH3	PEROMH2	PEROMH1	PEROMH0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **PEROMH[7:0]]**：ISP ROM 地址的高字节缓存器。

13.6 ISP RAM地址寄存器

ISP RAM地址的长度为10位，分别位于PERAML寄存器和PERAMCNT[1:0]中。执行ISP之前，用程序设置ISP RAM的开始地址。

0DEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PERAML	PERAML7	PERAML6	PERAML5	PERAML4	PERAML3	PERAML2	PERAML1	PERAML0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **PERAML[7:0]**：ISP RAM 地址[7:0]。

0DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PERAMCNT	PERAMCNT7	PERAMCNT6	PERAMCNT5	PERAMCNT4	PERAMCNT3	-	PERAML9	PERAML8
读/写	R/W	R/W	R/W	R/W	R/W	-	R/W	R/W
复位后	0	0	0	0	0	-	0	0

Bit [1:0] **PERAMCNT[1:0]**：ISP RAM 地址[9:8]。

13.7 ISP ROM编程长度寄存器

ISP编程长度为1字~32字，由PERAMCNT[7:3]（5位格式）控制。执行ISP ROM编程之前，由程序设置编程长度。

0DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PERAMCNT	PERAMCNT7	PERAMCNT6	PERAMCNT5	PERAMCNT4	PERAMCNT3	-	PERAML9	PERAML8
读/写	R/W	R/W	R/W	R/W	R/W	-	R/W	R/W
复位后	0	0	0	0	0	-	0	0

Bit [7:3] **PERAMCNT[7:3]**：ISP ROM 编程长度控制寄存器。

ISP 编程长度 = PERAMCNT[7:3] + 1

PERAMCNT[7:3]=0：ISP 编程长度为 1 字。
PERAMCNT[7:3]=1：ISP 编程长度为 2 字。
…
…
PERAMCNT[7:3]=30：ISP 编程长度为 31 字。
PERAMCNT[7:3]=31：ISP 编程长度为 32 字。

✷ **注：定义需要编程的字的编号，PERAMENT[7:3]最大为 01FH，可以写入 32 字（64 字节 RAM）到 flash ROM 中，PERAMCNT[7:3]最小为 00H，可以写入 1 字到 Flash ROM 中。**

14 指令集

Field	Mnemonic		指令说明	C	DC	Z	Cycle
	MOV	A,M	A ← M	-	-	√	1
M	MOV	M,A	M ← A	-	-	-	1
O	B0MOV	A,M	A ← M (bank 0)	-	-	√	1
V	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
E	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I，“M”指 80H~87H 的寄存器（如 PFLAG、R、Y、Z …）。	-	-	-	1
	XCH	A,M	A ←→M	-	-	-	1+N
	B0XCH	A,M	A ←→M (bank 0)	-	-	-	1+N
	MOVC		R, A ← ROM [Y,Z]	-	-	-	2
	ADC	A,M	A ← A + M + C，若有进位，则 C=1，否则 C=0。	√	√	√	1
A	ADC	M,A	M ← A + M + C，若有进位，则 C=1，否则 C=0。	√	√	√	1+N
R	ADD	A,M	A ← A + M，若有进位，则 C=1，否则 C=0。	√	√	√	1
I	ADD	M,A	M ← A + M，若有进位，则 C=1，否则 C=0。	√	√	√	1+N
T	B0ADD	M,A	M (bank 0) ← M (bank 0) + A，若有进位，则 C=1，否则 C=0。	√	√	√	1+N
H	ADD	A,I	A ← A + I，若有进位，则 C=1，否则 C=0。	√	√	√	1
M	SBC	A,M	A ← A - M - /C，若有借位，则 C=0，否则 C=1。	√	√	√	1
E	SBC	M,A	M ← A - M - /C，若有借位，则 C=0，否则 C=1。	√	√	√	1+N
T	SUB	A,M	A ← A - M，若有借位，则 C=0，否则 C=1。	√	√	√	1
I	SUB	M,A	M ← A - M，若有借位，则 C=0，否则 C=1。	√	√	√	1+N
C	SUB	A,I	A ← A - I，若有借位，则 C=0，否则 C=1。	√	√	√	1
	DAA		将 ACC 中的数据由十六进制转换成十进制格式。	√	-	-	1
	MUL	A,M	R, A ← A * M，结果的 LB 存入 ACC，HB 存入 R 寄存器，ZF 受 ACC 的影响。	-	-	√	2
	AND	A,M	A ← A 与 M。	-	-	√	1
L	AND	M,A	M ← A 与 M。	-	-	√	1+N
O	AND	A,I	A ← A 与 I。	-	-	√	1
G	OR	A,M	A ← A 或 M。	-	-	√	1
I	OR	M,A	M ← A 或 M。	-	-	√	1+N
C	OR	A,I	A ← A 或 I。	-	-	√	1
	XOR	A,M	A ← A 异或 M。	-	-	√	1
	XOR	M,A	M ← A 异或 M。	-	-	√	1+N
	XOR	A,I	A ← A 异或 I。	-	-	√	1
	COM	M	A ← M (1’s complement).	-	-	√	1
	COMM	M	M ← M (1’s complement).	-	-	√	1
	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)	-	-	-	1
P	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)	-	-	-	1+N
R	RRC	M	A ← RRC M	√	-	-	1
O	RRCM	M	M ← RRC M	√	-	-	1+N
C	RLC	M	A ← RLC M	√	-	-	1
E	RLCM	M	M ← RLC M	√	-	-	1+N
S	CLR	M	M ← 0	-	-	-	1
S	BCLR	M.b	M.b ← 0	-	-	-	1+N
	BSET	M.b	M.b ← 1	-	-	-	1+N
	B0BCLR	M.b	M(bank 0).b ← 0	-	-	-	1+N
	B0BSET	M.b	M(bank 0).b ← 1	-	-	-	1+N
	CMPRS	A,I	ZF,C ← A – I，若 A=1，则跳到下一条指令。	√	-	√	1 + S
B	CMPRS	A,M	ZF,C ← A – M 若 A=M，则跳到下一条指令。	√	-	√	1 + S
R	INCS	M	A ← M + 1，若 A=0，则跳到下一条指令。	-	-	-	1+ S
A	INCMS	M	M ← M + 1，若 M = 0，则跳到下一条指令。	-	-	-	1+N+S
	INC	M	A ← M + 1.	-	-	√	1
	INCM	M	M ← M + 1.	-	-	√	1+N
N	DECS	M	A ← M – 1，若 A = 0，则跳到下一条指令。	-	-	-	1+ S
C	DECMS	M	M ← M – 1，若 M = 0，则跳到下一条指令。	-	-	-	1+N+S
	DEC	M	A ← M – 1.	-	-	√	1
	DECM	M	M ← M – 1.	-	-	√	1+N
H	BTS0	M.b	M.b = 0，则跳到下一条指令。	-	-	-	1 + S
	BTS1	M.b	M.b = 1，则跳到下一条指令。	-	-	-	1 + S
	B0BTS0	M.b	M(bank 0).b = 0，则跳到下一条指令。	-	-	-	1 + S
	B0BTS1	M.b	M(bank 0).b = 1，则跳到下一条指令。	-	-	-	1 + S
	TS0M	M	M = 0, Z = 1，否则 Z = 0。	-	-	√	1
	JMP	d	PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALL	d	Stack ← PC15~PC0, PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALLHL		Stack ← PC15~PC0, PC15~PC8 ← H register, PC7~PC0 ← L register	-	-	-	2
	CALLYZ		Stack ← PC15~PC0, PC15~PC8 ← Y register, PC7~PC0 ← Z register	-	-	-	2
M	RET		PC ← Stack	-	-	-	2
I	RETI		PC ← Stack，使能全局中断。	-	-	-	2
S	RETLW	I	PC ← Stack，加载 I 的数据到 ACC。	-	-	√	2
C	NOP		空操作。	-	-	-	1

注： **1. M 指系统寄存器或 RAM，若 M 指系统寄存器，则 N=0，否则 N=1。**
2. 若条件为真则 S=1，否则 S=0。

15 电气特性

15.1 极限参数

Supply voltage (Vdd)..... - 0.3V ~ 6.0V
Supply voltage (Vdd).....-0.3V ~ 3.6V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8F27E65L, SN8F27E64L, SN8F27E62L..... -40°C ~ + 85°C
SN8F27E65, SN8F27E64, SN8F27E62… ..... -40°C ~ + 85°C
Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

15.2 电气特性

SN8F27E60 Series DC CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 16MHz, ambient temperature is 25°C unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	-40°C~85°C, Fcpu = 16MHz, ISP is inactive.		1.8	-	5.5	V
		-40°C~85°C, Fcpu = 16MHz, ISP actives.		2.5		5.5	V
RAM Data Retention voltage	Vdr			1.5	-	-	V
*Vdd rise rate	Vpor	Vdd rise rate to ensure internal power-on reset		0.05	-	-	V/ms
Input Low Voltage	ViL	All input ports, Reset pin, XIN/XOUT pins.		Vss	-	0.3Vdd	V
Input High Voltage	ViH	All input ports, Reset pin, XIN/XOUT pins.		0.7Vdd	-	Vdd	V
Output Low Voltage	VoL	IoL1=15mA, IoL2=23mA.		Vss		Vss+0.5	V
Output High Voltage	VoH	IoH1=10mA, IoH2=12mA.		Vdd-0.5		Vdd	V
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O port pull-up resistor	Rup1	Vin = Vss, Vdd = 3V, XIN/XOUT pins		120	240	360	KΩ
		Vin = Vss, Vdd = 5V, XIN/XOUT pins		60	120	180	
	Rup2	Vin = Vss , Vdd = 3V, P0/P1/P4/P5 pins		100	200	300	
		Vin = Vss , Vdd = 5V, P0/P1/P4/P5 pins		50	100	150	
I/O output source current	IoH1	Vop = Vdd – 0.5V, XIN/XOUT pins.		5	10	-	mA
	IoH2	Vop = Vdd – 0.5V, P0/P1/P4/P5 pins.		5	13	-	
I/O output sink current	IoL1	Vop = Vss + 0.5V, XIN/XOUT pins.		8	15	-	
	IoL2	Vop = Vss + 0.5V, P0/P1/P4/P5 pins.		8	23	-	
*INTn trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
Supply Current (Disable ADC)	Idd1	Run Mode (No loading)	Vdd= 3V, Fcpu = 16MHz	-	6.8	-	mA
			Vdd= 5V, Fcpu = 16MHz	-	7	-	mA
			Vdd= 3V, Fcpu = 4MHz	-	2.1	-	mA
			Vdd= 5V, Fcpu = 4MHz	-	2.2	-	mA
			Vdd= 3V, Fcpu = 1MHz	-	0.85	-	mA
			Vdd= 5V, Fcpu = 1MHz	-	0.87	-	mA
			Vdd= 3V, Fcpu = 32KHz/4	-	120	-	uA
			Vdd= 5V, Fcpu = 32KHz/4	-	140	-	uA
	Idd2	Slow Mode (Internal low RC, Stop high clock)	Vdd= 3V, ILRC=16KHz	-	110	-	uA
			Vdd= 5V, ILRC=32KHz	-	130	-	uA
	Idd3	Sleep Mode	Vdd= 3V	-	90	-	uA
			Vdd= 5V	-	100	-	uA
	Idd4	Green Mode (No loading, Watchdog Disable)	Vdd= 3V, IHRC=16MHz	-	450	-	uA
			Vdd= 5V, IHRC=16MHz	-	500	-	uA
			Vdd= 3V, Ext. 32KHz X'tal	-	110	-	uA
			Vdd= 5V, Ext. 32KHz X'tal	-	130	-	uA
			Vdd= 3V, ILRC=16KHz	-	110	-	uA
			Vdd= 5V, ILRC=32KHz	-	120	-	uA
Internal High Oscillator Freq.	Fihrc	Internal Hihg RC (IHRC)	25°C, Vdd=2.4V~ 5.5V	15.68	16	16.32	MHz
			-40°C~85°C,Vdd=2.4V~ 5.5V	15.4	16	16.5	MHz
LVD Voltage	Vdet0	Low voltage reset level. 25°C		1.7	1.8	1.9	V
		Low voltage reset level. -40°C~85°C		1.6	1.8	2.0	V
	Vdet1	Low voltage reset/indicator level. 25°C		2.3	2.4	2.5	V
		Low voltage reset/indicator level. -40°C~85°C		2.2	2.4	2.6	V
	Vdet2	Low voltage reset/indicator level. 25°C		3.2	3.3	3.4	V
		Low voltage reset/indicator level. -40°C~85°C		3.1	3.3	3.5	V

" *" These parameters are for design reference, not tested.

SN8F27E60L Series DC CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 3.0V, Fosc = 16MHz, ambient temperature is 25°C unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	-40°C ~ + 85°C, Fcpu = 16MHz, ISP is inactive.		1.8	3.0	3.3	V
		-40°C ~ + 85°C, Fcpu = 16MHz, ISP actives.		2.5	3.0	3.3	V
RAM Data Retention voltage	Vdr			1.5	-	-	V
*Vdd rise rate	Vpor	Vdd rise rate to ensure internal power-on reset		0.05	-	-	V/ms
Input Low Voltage	ViL	All input ports, Reset pin, XIN/XOUT pins.		Vss	-	0.3Vdd	V
Input High Voltage	ViH	All input ports, Reset pin, XIN/XOUT pins.		0.7Vdd	-	Vdd	V
Output Low Voltage	VoL	IoL1=15mA, IoL2=23mA.		Vss		Vss+0.5	V
Output High Voltage	VoH	IoH1=10mA, IoH2=12mA.		Vdd-0.5		Vdd	V
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O port pull-up resistor	Rup1	Vin = Vss, XIN/XOUT pins.		120	240	360	KΩ
	Rup2	Vin = Vss, P0/P1/P4/P5 pins.		100	200	300	
I/O output source current	IoH1	Vop = Vdd – 0.5V, XIN/XOUT pins.		3	7	-	mA
	IoH2	Vop = Vdd – 0.5V, P0/P1/P4/P5 pins.		4	8	-	
I/O output sink current	IoL1	Vop = Vss + 0.5V, XIN/XOUT pins.		4	9	-	
	IoL2	Vop = Vss + 0.5V, P0/P1/P4/P5 pins.		7	14	-	
*INTn trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
Supply Current (Disable ADC)	Idd1	Run Mode (No loading)	Vdd= 3V, Fcpu = 16MHz	-	7	-	mA
			Vdd= 3V, Fcpu = 4MHz	-	1.9	-	mA
			Vdd= 3V, Fcpu = 1MHz	-	0.73	-	mA
			Vdd= 3V, Fcpu = 32KHz/4	-	35	-	uA
	Idd2	Slow Mode (Internal low RC, Stop high clock)	Vdd= 3V, ILRC=16KHz	-	25	-	uA
	Idd3	Sleep Mode	Vdd= 3V	-	1	3	uA
	Idd4	Green Mode (No loading, Watchdog Disable)	Vdd= 3V, IHRC=16MHz	-	400	-	uA
			Vdd= 3V, Ext. 32KHz X'tal	-	20	-	uA
			Vdd= 3V, ILRC=16KHz	-	5	-	uA
Internal High Oscillator Freq.	Fihrc	Internal Hihg RC (IHRC)	25°C, Vdd=2.4V~ 5.5V	15.68	16	16.32	MHz
			-40°C ~ + 85°C, Vdd=2.4V~ 5.5V	15.4	16	16.5	
LVD Voltage	Vdet0	Low voltage reset level. 25°C		1.7	1.8	1.9	V
		Low voltage reset level. -40°C ~ + 85°C		1.6	1.8	2.0	
	Vdet1	Low voltage reset/indicator level. 25°C		2.3	2.4	2.5	V
		Low voltage reset/indicator level. -40°C ~ + 85°C		2.2	2.4	2.6	
	Vdet2	Low voltage reset/indicator level. 25°C		3.2	3.3	3.4	V
		Low voltage reset/indicator level. -40°C ~ + 85°C		3.1	3.3	3.5	

" *" These parameters are for design reference, not tested.

ADC CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 4MHz,Fcpu=1MHz,ambient temperature is 25°C unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
AIN0 ~ AIN11 input voltage	Vani	Vdd = 5.0V	0	-	Avrefh	V
ADC reference Voltage	Vref		2	-	-	V
*ADC enable time	Tast	Ready to start convert after set ADENB = "1"	100	-	-	us
*ADC current consumption	IADC	Vdd=5.0V	-	0.6	-	mA
		Vdd=3.0V	-	0.4	-	mA
ADC Clock Frequency	FADCLK	VDD=5.0V	-	-	8M	Hz
		VDD=3.0V	-	-	5M	Hz
ADC Conversion Cycle Time	FADCYL	VDD=2.4V~5.5V	64	-	-	1/FADCLK
ADC Sampling Rate (Set FADS=1 Frequency)	FADSMP	VDD=5.0V	-	-	125	K/sec
		VDD=3.0V	-	-	80	K/sec
Differential Nonlinearity	DNL	VDD=5.0V , AVREFH=3.2V, FADSMP =7.8K	-1	-	+1	LSB
Integral Nonlinearity	INL	VDD=5.0V , AVREFH=3.2V, FADSMP =7.8K	-1	-	+1	LSB
No Missing Code	NMC	VDD=5.0V , AVREFH=3.2V, FADSMP =7.8K	9	-	10	Bits
ADC offset Voltage	VADCoffset	Non-trimmed	-10	0	+10	mV
		Trimmed	-2	0	+2	mV

" *" These parameters are for design reference, not tested.

FLASH MENORY CHARACTERISTIC

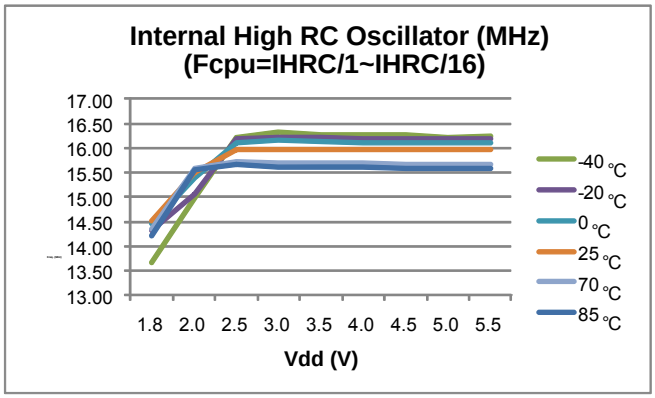
(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 4MHz,Fcpu=1MHz,ambient temperature is 25°C unless otherwise note.)

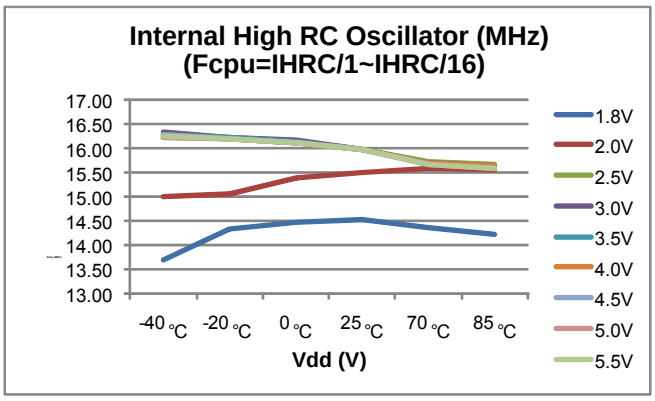
PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
Supply Voltage	Vdd1	Read mode	1.8		Vdd	V
		Erase/Program	2.5		Vdd	
Flash Endurance time	Ten1	Erase + Program, -10°C ~ + 85°C	20K	*100K	-	Cycle
	Ten2	Erase + Program, -40°C ~ 10°C	20K	*70K	-	Cycle
Page erase current	Ier	Vdd1 = 2.5V	-	2.5	5	mA
Program current	Ipg	Vdd1 = 2.5V	-	3.5	7	mA
Page erase time	Ter	Vdd1 = 2.5V, 1-page (128-word).	-	-	30	ms
Program time	Tpg	Vdd = 2.5V, ISP setup time.	-	-	380	us
		Vdd1 = 2.5V, 1-word program.	-	-	30	

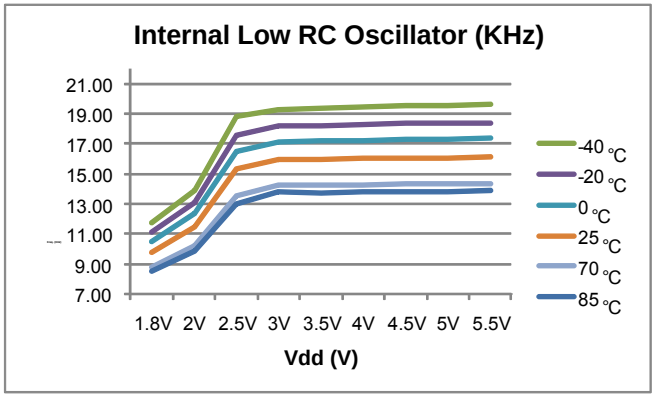
" *" These parameters are for design reference, not tested.

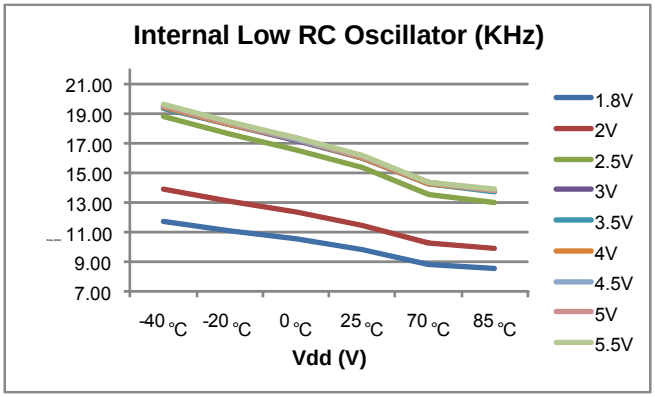
15.3 特性曲线

本章所列的各曲线图仅作设计参考，其中给出的部分数据可能超出了芯片指定的工作范围，为保证芯片的正常工作，请严格参照电气特性说明。





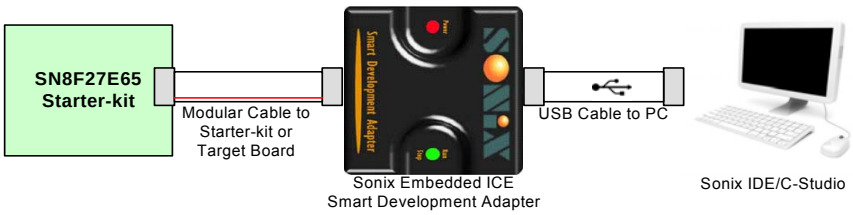




16 开发工具

SONIX 提供了一款嵌入式的 ICE 用来仿真 SN8F27E65 系列的韧件开发。其平台为在线调试器，由 SONIX M2IDE 软件控制，包括 Smart Development Adapter（SDA）、SN8F27E65 Starter-kit 和 M2IDE 软件，建立一个高速、低廉、功能强大和多任务同时执行的开发环境：包括仿真器、调试器和编程器。可以仿真如同实际芯片工作一样，这是因为仿真器的电路集成到 SN8F27E65 中，提供了一个理想真实的开发环境。

SN8F27E65 嵌入式 ICE 仿真系统：



SN8F27E65 嵌入式 ICE 仿真器包括：
Smart Development Adapter（SDA）；
USB 线，连接 SDA 和 PC 进行通讯；
SN8F27E65 Starter-Kit.
10PIN 排线，连接 SDA 和 SN8F27E65 Starter-kit 或者客户的目标板；
CD-ROM，包含 M2IDE 软件（M2IDE V124 或更新的版本）。

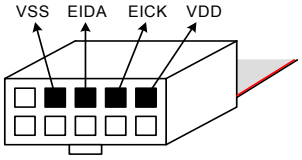
SN8F27E65 嵌入式 ICE 仿真器特性：
- 客户的目标板的工作电压：1.8V~5.5V；
- 高达 6 个硬件断点；
- 系统时钟速率高达 16MHz（Fcpu=16mips）；
- 振荡器支持内部高速 RC、内部低速 RC、外部晶体/陶瓷振荡电路和外部 RC 电路。

SN8F27E65 嵌入式 ICE 仿真器限制：
- EIDA 和 EICK 引脚与 GPIO 共用，嵌入式 ICE 模式下，共用的 GPIO 引脚不能工作。 强烈建议将这两个引脚设置为简单的功能，可以在没有调试平台的情况下进行校验。

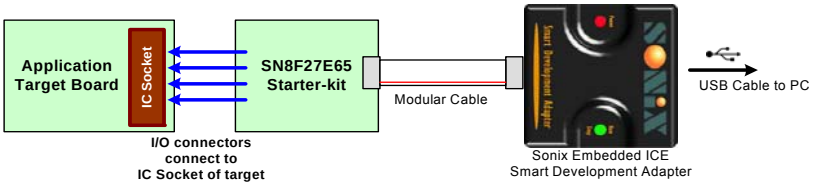
16.1 SMART DEVELOPMENT ADAPTER（SDA）

高速仿真器 Smart Development Adapter（SDA）是 SONIX 嵌入式 ICE 的一种，可以调试和编程 SONIX flash ROM 单片机，还可以在 M2IDE 和 SONIX Flash 单片机之间通过 USB 接口传送单片机的系统状态、RAM 数据和系统寄存器。SDA 的另一端通过 4 线串行接口连接 SN8F27E65 Starter-kit 或者客户的目标板。除了调试功能外，SDA 还可以用在编程器，将 PC 中的韧件传入单片机中。

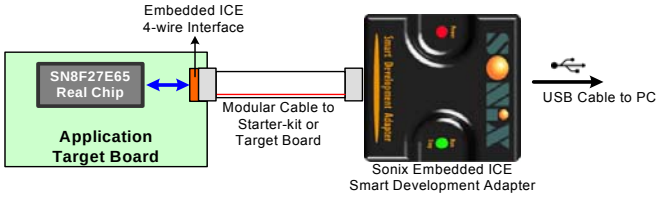
SDA 和 SN8F27E65 flash 单片机通过 4 线总线进行通讯，4 线总线的引脚定义如下：



排线可以插入 SN8F27E65 Starter-kit 上的目标座，或者客户目标板上相匹配的目标座上。

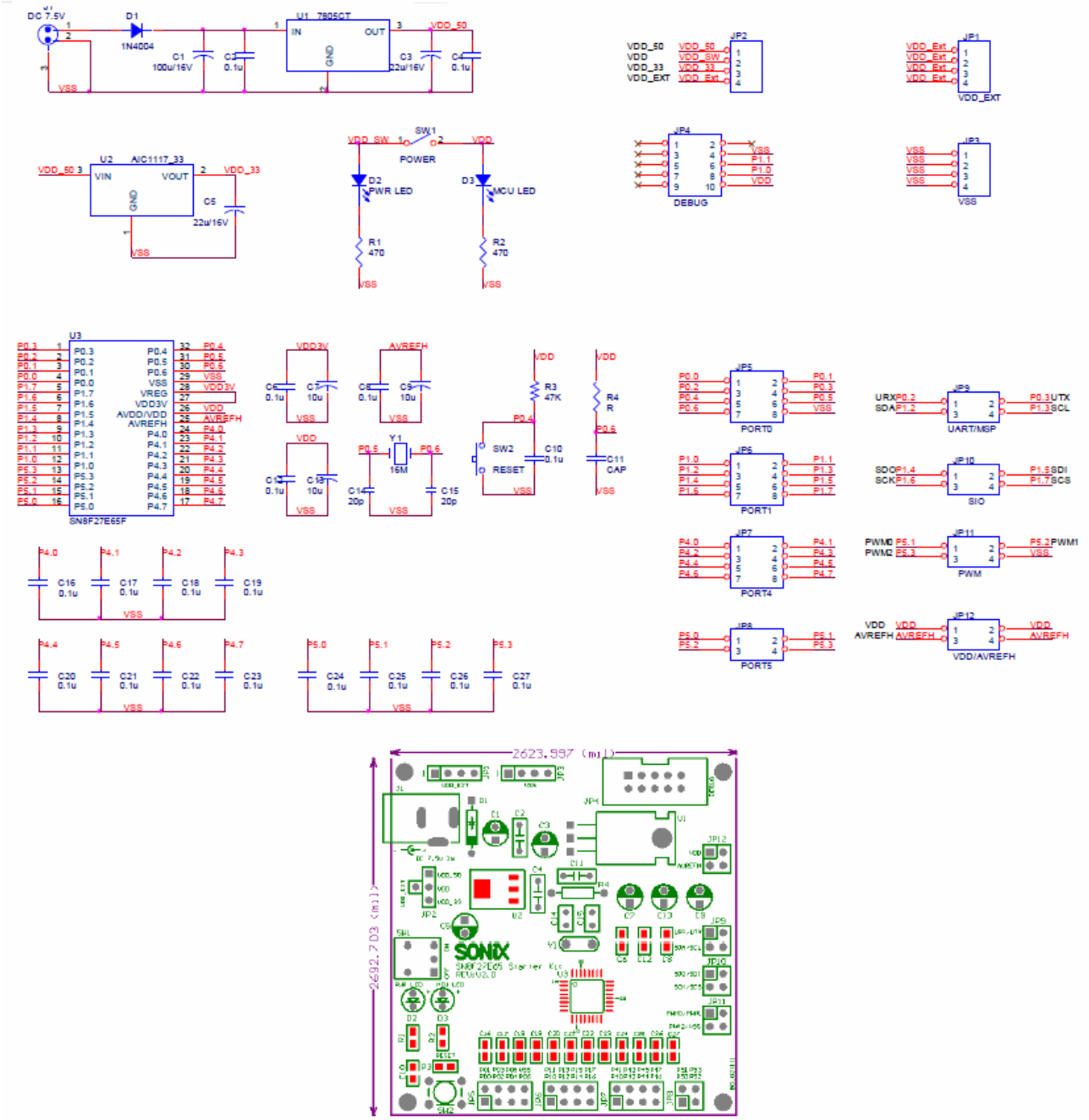


如果应用目标板已经设计并准备好，可以直接将排线总线插入目标板上，4 线总线连接到 SN8F27E65 实际芯片，建立一个实际的应用环境，在此模式下，必须将 SN8F27E65 IC 放置在目标板，否则会仿真出错。



16.2 SN8F27E65 STARTER-KIT

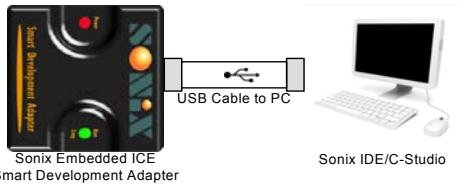
SN8F27E65 Starter-kit 是一个简易的开发平台，包括 SN8F27E65 实际芯片和 I/O 接口，用来输入信号或者驱动用户应用的外部设备。在用户目标板还没有准备好时，替代目标板进行仿真，这是由于 SN8F27E65 集成了嵌入式 ICE 的在线调试器电路。SN8F27E65 Starter-kit 的 PCB 图如下所示：



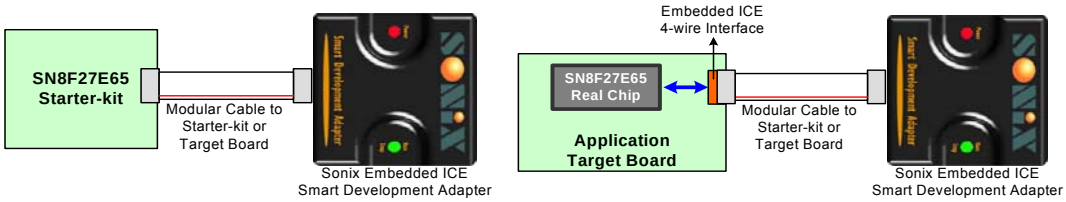
- J1：电源开关。
- JP2：VDD 电压源为 5.0V/3.3V/外部电源。
- JP1/JP3：外部电源。
- SW1：目标电源开关。
- U3：SN8F27E65F（SONIX 标准选项）。
- D2：电源指示灯。
- D3：单片机指示灯。
- C16~C27：12 通道 ADC 电容。
- SW2：外部复位触发源。
- JP5~JP11：I/O 接口。
- JP10：VDD 测试点。
- Y1, C14, C15：外部晶体/陶瓷振荡电路元件。
- R4, C11：外部 RC 振荡电路元件。
- JP12：VDD 测试点和 AVREFH 接口。

16.3 安装仿真器/调试器

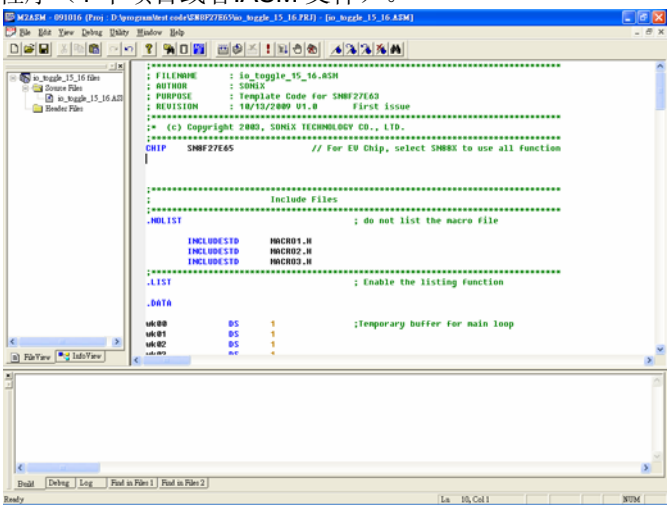
- 安装 M2IDE 软件（V124 或者更新的版本）。
- 通过 USB 线连接 SDA 和电脑。



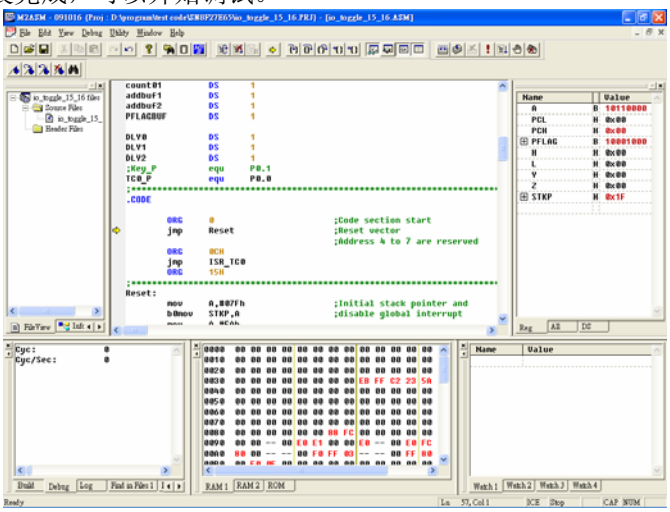
- 通过排线连接 SDA 和 SN8F27E65 Starter-kit 或者用户目标板。



- 连接电源到 SN8F27E65 Starter-kit 或者用户目标板，然后打开电源开关。
- 打开 M2IDE 软件，下载韧件程序（1 个项目或者.ASM 文件）。

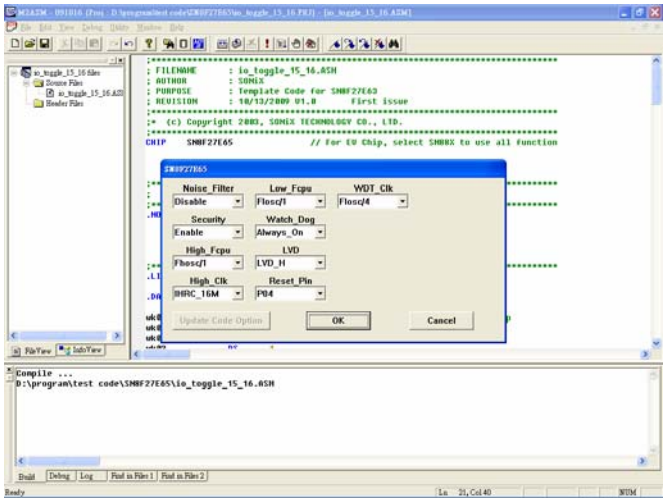


- 打开 SN8F27E65 Starter-kit 或者用户目标板上的电源开关。
- 嵌入式 ICE 仿真平台已经安装完成，可以开始调试。

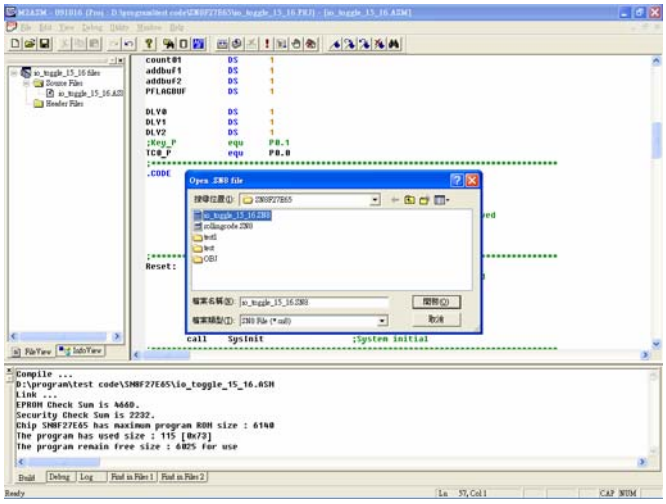


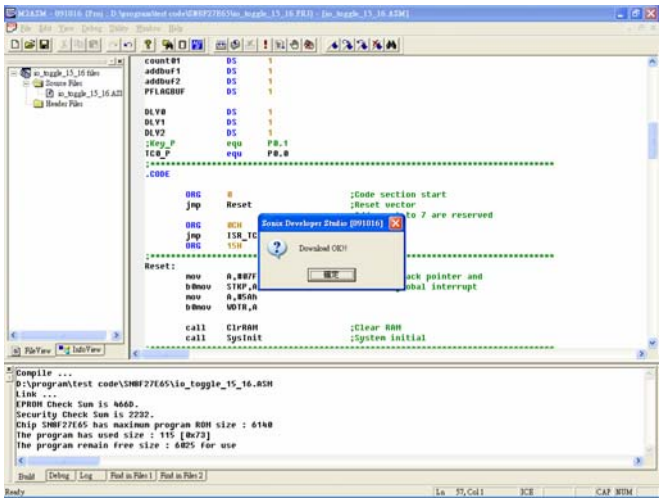
16.4 烧录配置

- 首先设置/调试环境。
- 编译韧件程序并生成.SN8 文件。



- 执行 M2IDE 的下载功能（F8）。
- 打开一个.SN8 文件，按下“ENTER”，下载韧件到 SN8F27E65 Starter-kit 或者用户目标板。





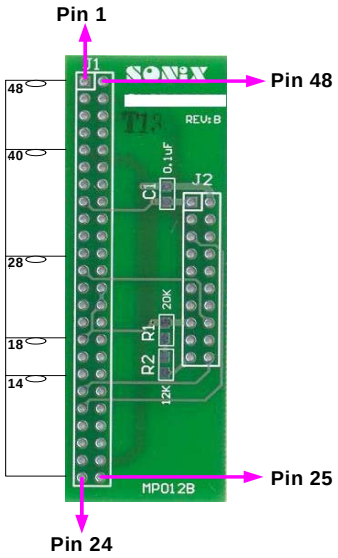
- 关闭 SN8F27E65 Starter-kit 或者用户目标板上的电源。
- 断开 SN8F27E65 Starter-kit 或者用户目标板与 SDA 的连接。
- 打开 SN8F27E65 Starter-kit 或者用户目标板的上的电源，单片机独立工作。

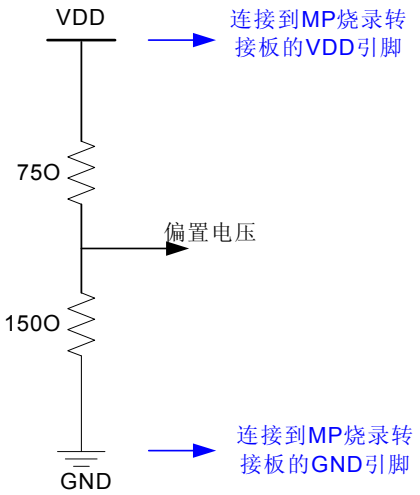
17 Flash烧录引脚

可以通过 SDA、MP Pro Writer 和 MPIII Writer 对 SN8F27E60 系列单片机 Flash ROM 进行擦除/烧录/校验动作。

- **SDA**：嵌入式 ICE。
- **MP Pro Writer**：直接放置 SN8F27E60 系列单片机进行烧录。
- **MPIII Writer**：针对“L”系列，必须在烧录转接板上设置偏压电路。

17.1 烧录转接板引脚配置





JP3（48-pin text 工具引脚分配）

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

Writer JP1/JP2

VDD	1	2	VSS
CLK/PGCLK	3	4	CE
PGM/OTPCLK	5	6	OE/ShiftDat
D1	7	8	D0
D3	9	10	D2
D5	11	12	D4
D7	13	14	D6
VDD	15	16	VPP
HLS	17	18	RST
-	19	20	ALSB/PDB

JP1 连接烧录转接板
JP2 连接 Dice 和>48 pin 封装形式的 IC

17.2 烧录引脚配置

SN8F27E65 系列单片机烧录引脚信息							
单片机名称		SN8F27E65P(DIP)			SN8F27E65LP(DIP)		
烧录器接口		IC 和 JP3 48-pin text tool 引脚配置					
JP1/JP2 引脚编号	JP1/JP2 引脚名称	IC 引脚编号	IC 引脚名称	JP3 引脚编号	IC 引脚编号	IC 引脚名称	JP3 引脚编号
1	VDD	30	VDD	38	31	VDD	39
2	GND	1	VSS	9	1	VSS	9
3	CLK	23	P4.5	31	23	P4.5	31
4	CE	-	-	-	-	-	-
5	PGM	22	P4.6	30	22	P4.6	30
6	OE	21	P4.7	29	21	P4.7	29
7	D1	-	-	-	-	-	-
8	D0	-	-	-	-	-	-
9	D3	-	-	-	-	-	-
10	D2	-	-	-	-	-	-
11	D5	-	-	-	-	-	-
12	D4	-	-	-	-	-	-
13	D7	-	-	-	-	-	-
14	D6	-	-	-	-	-	-
15	VDD	-	-	-	-	-	-
16	VPP						
17	HLS	-	-	-	-	-	-
18	RST	-	-	-	-	-	-
19	-	-	-	-	-	-	-
20	ALSB/PDB	20	P5.0	28	20	P5.0	28
-	Bias Voltage	-	-	-	32	VDD	40

SN8F27E65 系列单片机烧录引脚信息							
单片机名称		SN8F27E65F(LQFP) SN8F27E65J (QFN)			SN8F27E65LF(LQFP) SN8F27E65LJ (QFN)		
烧录器接口		IC 和 JP3 48-pin text tool 引脚配置					
JP1/JP2 引脚编号	JP1/JP2 引脚名称	IC 引脚编号	IC 引脚名称	JP3 引脚编号	IC 引脚编号	IC 引脚名称	JP3 引脚编号 r
1	VDD	26	VDD	34	27	VDD	35
2	GND	29	VSS	37	29	VSS	37
3	CLK	19	P4.5	27	19	P4.5	27
4	CE	-	-	-	-	-	-
5	PGM	18	P4.6	26	18	P4.6	26
6	OE	17	P4.7	25	17	P4.7	25
7	D1	-	-	-	-	-	-
8	D0	-	-	-	-	-	-
9	D3	-	-	-	-	-	-
10	D2	-	-	-	-	-	-
11	D5	-	-	-	-	-	-
12	D4	-	-	-	-	-	-
13	D7	-	-	-	-	-	-
14	D6	-	-	-	-	-	-
15	VDD	-	-	-	-	-	-
16	VPP						
17	HLS	-	-	-	-	-	-
18	RST	-	-	-	-	-	-
19	-	-	-	-	-	-	-
20	ALSB/PDB	16	P5.0	24	16	P5.0	24
-	Bias Voltage	-	-	-	28	VDD	36

SN8F27E65 系列单片机烧录引脚信息							
单片机名称		SN8F27E64K/S/X(SKDIP/SOP/SSOP)			SN8F27E64LK/S/X(SKDIP/SOP/SSOP)		
烧录器接口		IC 和 JP3 48-pin text tool 引脚配置					
JP1/JP2 引脚编号	JP1/JP2 引脚名称	IC 引脚编号	IC 引脚名称	JP3 引脚编号	IC 引脚编号	IC 引脚名称	JP3 引脚编号
1	VDD	27	VDD	37	27	VDD	37
2	GND	1	VSS	11	1	VSS	11
3	CLK	22	P4.5	32	22	P4.5	32
4	CE	-	-	-	-	-	-
5	PGM	21	P4.6	31	21	P4.6	31
6	OE	20	P4.7	30	20	P4.7	30
7	D1	-	-	-	-	-	-
8	D0	-	-	-	-	-	-
9	D3	-	-	-	-	-	-
10	D2	-	-	-	-	-	-
11	D5	-	-	-	-	-	-
12	D4	-	-	-	-	-	-
13	D7	-	-	-	-	-	-
14	D6	-	-	-	-	-	-
15	VDD	-	-	-	-	-	-
16	VPP						
17	HLS	-	-	-	-	-	-
18	RST	-	-	-	-	-	-
19	-	-	-	-	-	-	-
20	ALSB/PDB	19	P5.0	29	19	P5.0	29
-	Bias Voltage	-	-	-	28	VDD	38

SN8F27E65 系列单片机烧录引脚信息							
单片机名称		SN8F27E64J (QFN)			SN8F27E64LJ (QFN)		
烧录器接口		IC 和 JP3 48-pin text tool 引脚配置					
JP1/JP2 引脚编号	JP1/JP2 引脚名称	IC 引脚编号	IC 引脚名称	JP3 引脚编号	IC 引脚编号	IC 引脚名称	JP3 引脚编号
1	VDD	23	VDD	33	23	VDD	33
2	GND	25	VSS	35	25	VSS	35
3	CLK	18	P4.5	28	18	P4.5	28
4	CE	-	-	-	-	-	-
5	PGM	17	P4.6	27	17	P4.6	27
6	OE	16	P4.7	26	16	P4.7	26
7	D1	-	-	-	-	-	-
8	D0	-	-	-	-	-	-
9	D3	-	-	-	-	-	-
10	D2	-	-	-	-	-	-
11	D5	-	-	-	-	-	-
12	D4	-	-	-	-	-	-
13	D7	-	-	-	-	-	-
14	D6	-	-	-	-	-	-
15	VDD	-	-	-	-	-	-
16	VPP						
17	HLS	-	-	-	-	-	-
18	RST	-	-	-	-	-	-
19	-	-	-	-	-	-	-
20	ALSB/PDB	15	P5.0	25	15	P5.0	25
-	Bias Voltage	-	-	-	24	VDD	34

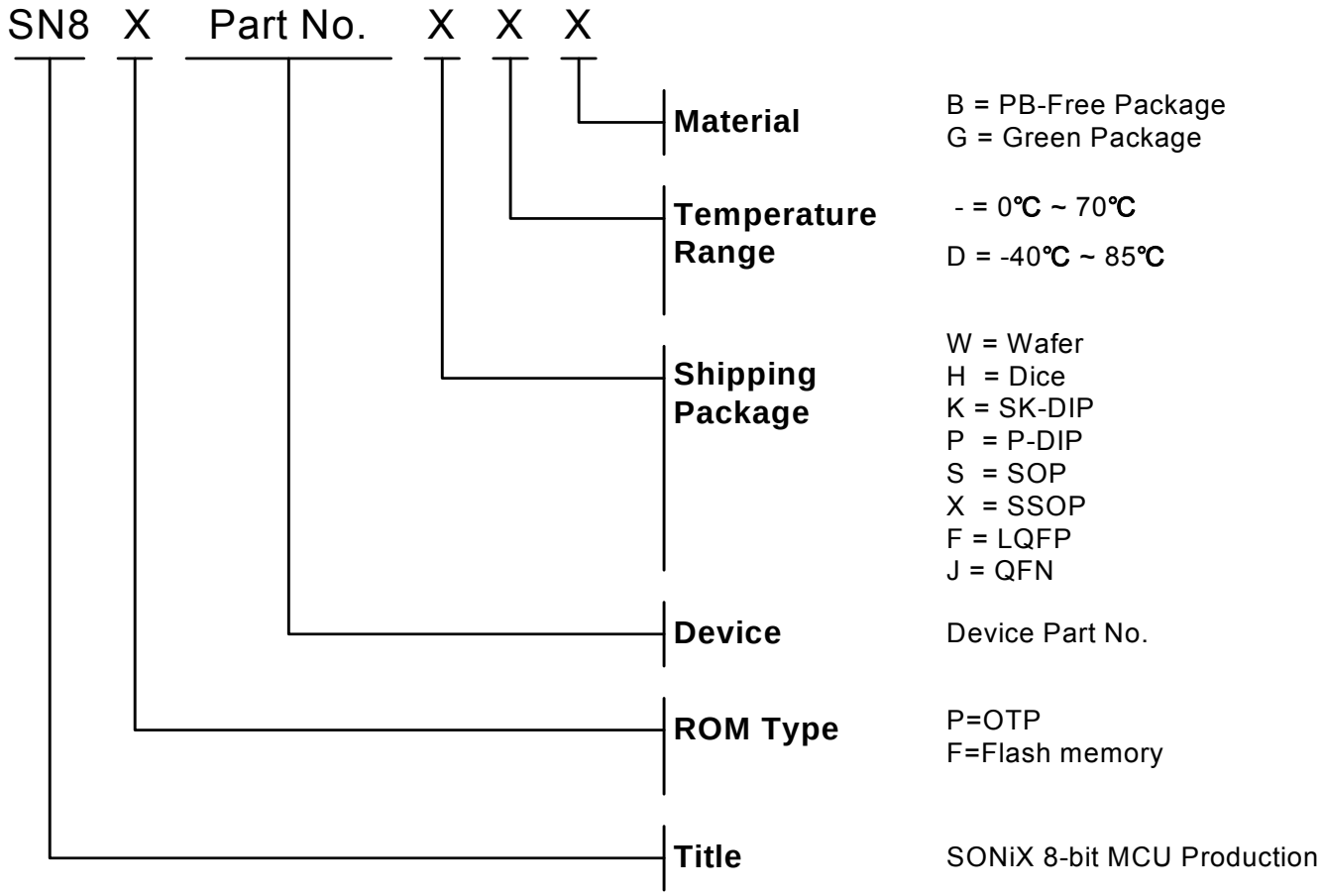
SN8F27E65 系列单片机烧录引脚信息							
单片机名称		SN8F27E62P/S(PDIP/SOP)			SN8F27E62LP/S(PDIP/SOP)		
烧录器接口		IC 和 JP3 48-pin text tool 引脚配置					
JP1/JP2 引脚编号	JP1/JP2 引脚名称	IC 引脚编号	IC 引脚名称	JP3 引脚编号	IC 引脚编号	IC 引脚名称	JP3 引脚编号
1	VDD	19	VDD	33	19	VDD	33
2	GND	1	VSS	15	1	VSS	15
3	CLK	16	P4.5	30	16	P4.5	30
4	CE	-	-	-	-	-	-
5	PGM	15	P4.6	29	15	P4.6	29
6	OE	14	P4.7	28	14	P4.7	28
7	D1	-	-	-	-	-	-
8	D0	-	-	-	-	-	-
9	D3	-	-	-	-	-	-
10	D2	-	-	-	-	-	-
11	D5	-	-	-	-	-	-
12	D4	-	-	-	-	-	-
13	D7	-	-	-	-	-	-
14	D6	-	-	-	-	-	-
15	VDD	-	-	-	-	-	-
16	VPP						
17	HLS	-	-	-	-	-	-
18	RST	-	-	-	-	-	-
19	-	-	-	-	-	-	-
20	ALSB/PDB	13	P5.0	27	13	P5.0	27
-	Bias Voltage	-	-	-	20	VDD	34

18 芯片正印命名规则

18.1 概述

SONiX 8 位单片机产品具有多种型号，本章将给出所有 8 位单片机分类命名规则。

18.2 芯片型号说明



18.3 命名举例

- **Wafer，Dice：**

单片机名称	ROM 类型	器件名称（Device）	封装形式	温度范围	封装材料
S8F27E65W	FLASH	27E65	Wafer	-40℃~85℃	-
SN8F27E65H	FLASH	27E65	Dice	-40℃~85℃	-

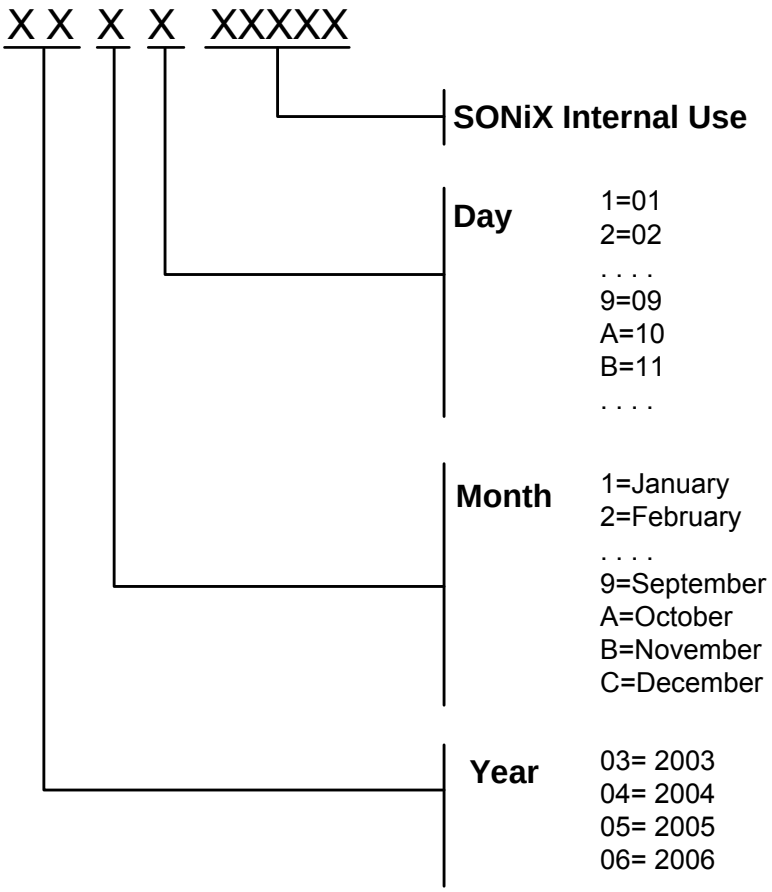
- 绿色封装：

单片机名称	ROM 类型	器件名称（Device）	封装形式	温度范围	封装材料
SN8F27E65PG	FLASH	27E65	P-DIP	-40℃~85℃	绿色封装
SN8F27E65FG	FLASH	27E65	LQFP	-40℃~85℃	绿色封装
SN8F37E65JG	FLASH	27E65	QFN	-40℃~85℃	绿色封装
SN8F27E65LPG	FLASH	27E65	P-DIP	0℃~70℃	绿色封装
SN8F27E65LFG	FLASH	27E65	LQFP	0℃~70℃	绿色封装
SN8F27E65LJG	FLASH	27E65	QFN	0℃~70℃	绿色封装
SN8F27E64KG	FLASH	27E65	SK-DIP	-40℃~85℃	绿色封装
SN8F27E64SG	FLASH	27E65	SOP	-40℃~85℃	绿色封装
SN8F27E64XG	FLASH	27E65	SSOP	-40℃~85℃	绿色封装
SN8F27E64JG	FLASH	27E65	QFN	-40℃~85℃	绿色封装
SN8F27E64LKG	FLASH	27E65	SK-DIP	0℃~70℃	绿色封装
SN8F27E64LSG	FLASH	27E65	SOP	0℃~70℃	绿色封装
SN8F27E64LXG	FLASH	27E65	SSOP	0℃~70℃	绿色封装
SN8F27E64LJG	FLASH	27E65	QFN	0℃~70℃	绿色封装
SN8F27E62PG	FLASH	27E65	P-DIP	-40℃~85℃	绿色封装
SN8F27E62SG	FLASH	27E65	SOP	-40℃~85℃	绿色封装
SN8F27E62LPG	FLASH	27E65	P-DIP	0℃~70℃	绿色封装
SN8F27E62LSG	FLASH	27E65	SOP	0℃~70℃	绿色封装

- 无铅封装：

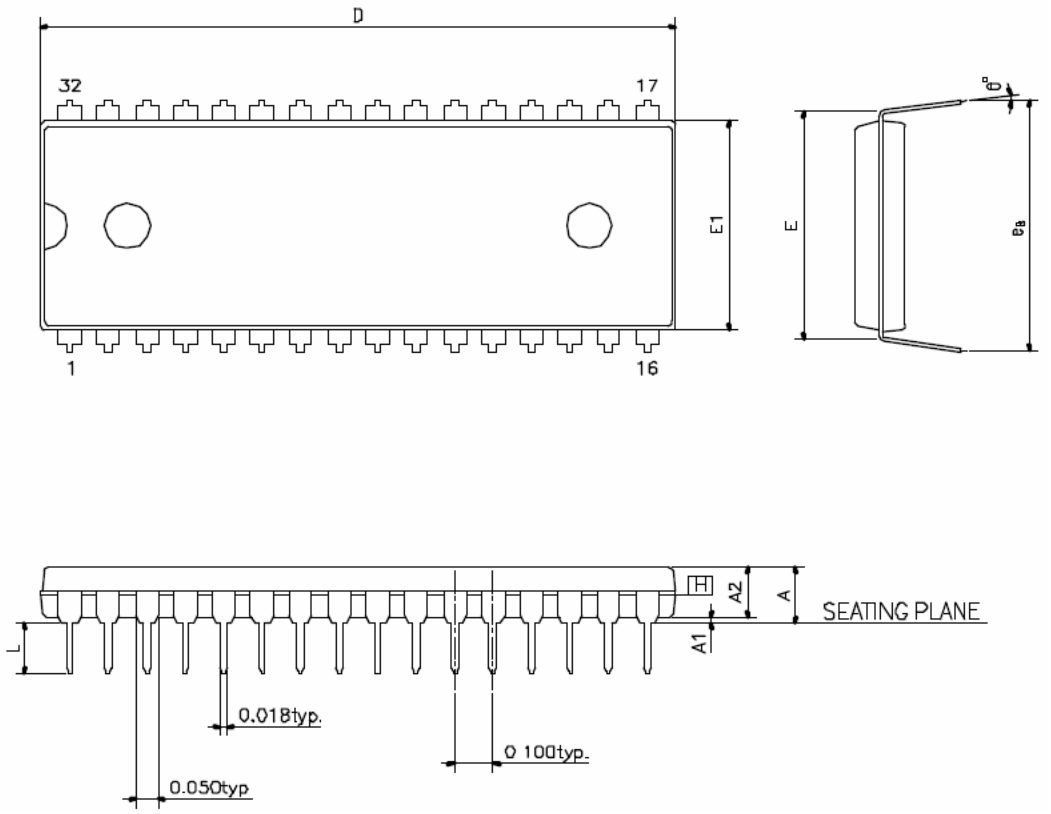
单片机名称	ROM 类型	器件名称（Device）	封装形式	温度范围	封装材料
SN8F27E65PB	FLASH	27E65	P-DIP	-40℃~85℃	无铅封装
SN8F27E65FB	FLASH	27E65	LQFP	-40℃~85℃	无铅封装
SN8F37E65JB	FLASH	27E65	QFN	-40℃~85℃	无铅封装
SN8F27E65LPB	FLASH	27E65	P-DIP	0℃~70℃	无铅封装
SN8F27E65LFB	FLASH	27E65	LQFP	0℃~70℃	无铅封装
SN8F27E65LJB	FLASH	27E65	QFN	0℃~70℃	无铅封装
SN8F27E64KB	FLASH	27E65	SK-DIP	-40℃~85℃	无铅封装
SN8F27E64SB	FLASH	27E65	SOP	-40℃~85℃	无铅封装
SN8F27E64XB	FLASH	27E65	SSOP	-40℃~85℃	无铅封装
SN8F27E64JB	FLASH	27E65	QFN	-40℃~85℃	无铅封装
SN8F27E64LKB	FLASH	27E65	SK-DIP	0℃~70℃	无铅封装
SN8F27E64LSB	FLASH	27E65	SOP	0℃~70℃	无铅封装
SN8F27E64LXB	FLASH	27E65	SSOP	0℃~70℃	无铅封装
SN8F27E64LJB	FLASH	27E65	QFN	0℃~70℃	无铅封装
SN8F27E62PB	FLASH	27E65	P-DIP	-40℃~85℃	无铅封装
SN8F27E62SB	FLASH	27E65	SOP	-40℃~85℃	无铅封装
SN8F27E62LPB	FLASH	27E65	P-DIP	0℃~70℃	无铅封装
SN8F27E62LSB	FLASH	27E65	SOP	0℃~70℃	无铅封装

18.4 日期码规则


XX X X XXXXX
SONiX Internal Use
Day
1=01
2=02
. . . .
9=09
A=10
B=11
. . . .
Month
1=January
2=February
. . . .
9=September
A=October
B=November
C=December
Year
03= 2003
04= 2004
05= 2005
06= 2006
. . . .

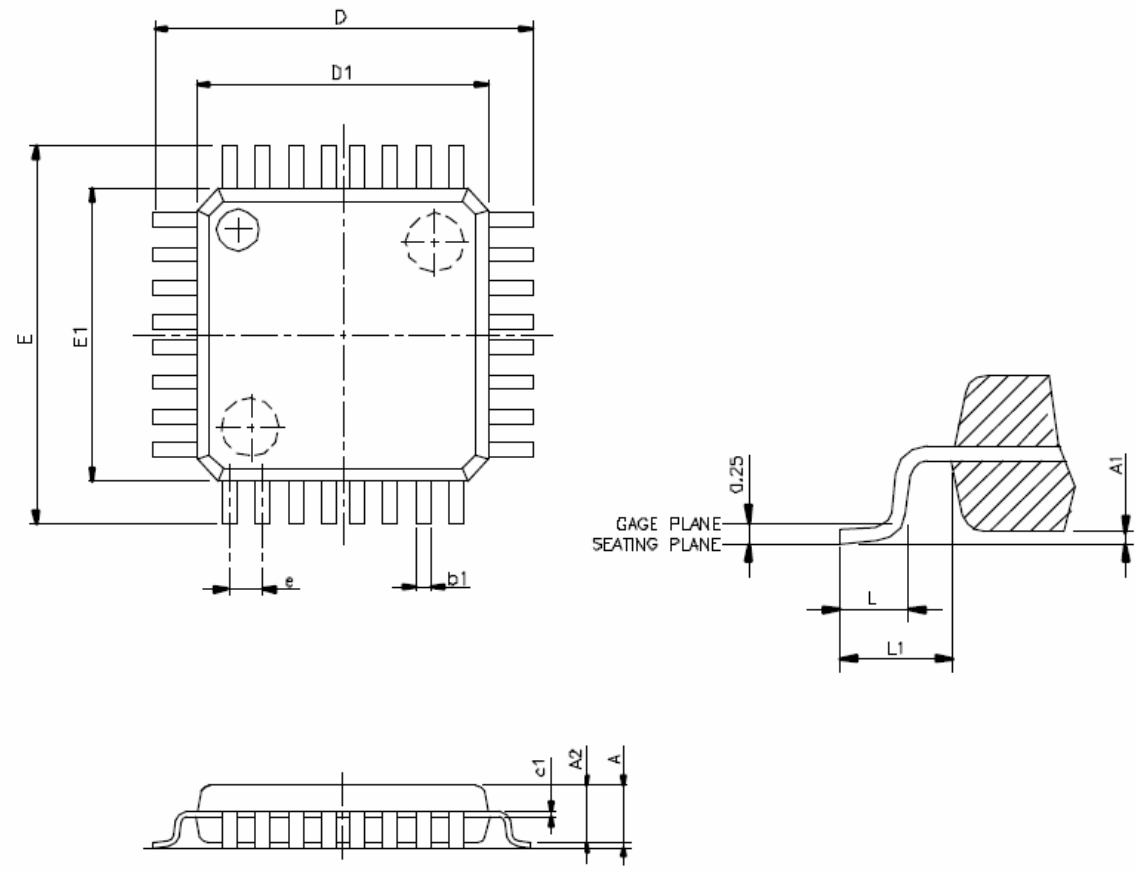
19 封裝信息

19.1 P-DIP 32 PIN



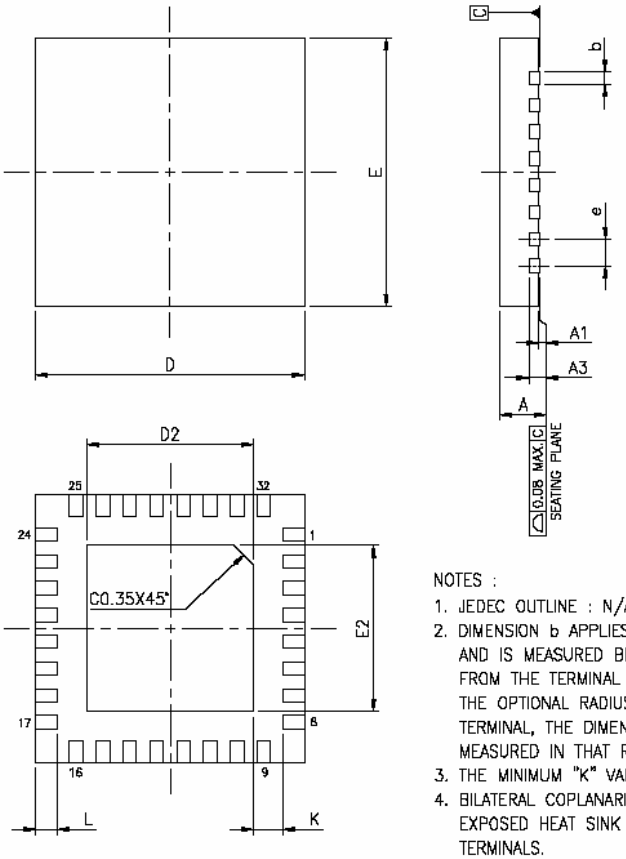
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.220	-	-	5.588
A1	0.015	-	-	0.381	-	-
A2	0.150	0.155	0.160	3.81	3.937	4.064
D	1.645	1.650	1.660	41.783	41.91	42.164
E	0.600 BSC			15.24 BSC		
E1	0.540	0.545	0.550	13.716	13.843	13.97
L	0.115	0.130	0.150	2.921	3.302	3.81
e_B	0.630	0.650	0.670	16.002	16.51	17.018
θ°	0°	7°	15°	0°	7°	15°

19.2 LQFP 32 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.063	-	-	1.6
A1	0.002	0.004	0.006	0.05	0.1	0.15
A2	0.053	0.055	0.057	1.35	1.4	1.45
c1	0.004	0.005	0.006	0.09	0.125	0.16
D	0.354 BSC			9 BSC		
D1	0.276 BSC			7 BSC		
BSC E	0.354 BSC			9 BSC		
E1	0.276 BSC			7 BSC		
e	0.031 BSC			0.8 BSC		
b	0.012	0.015	0.018	0.3	0.375	0.45
L	0.018	0.024	0.030	0.45	0.6	0.75
L1	0.039 REF			1 REF		

19.3 QFN 5X5 32 PIN

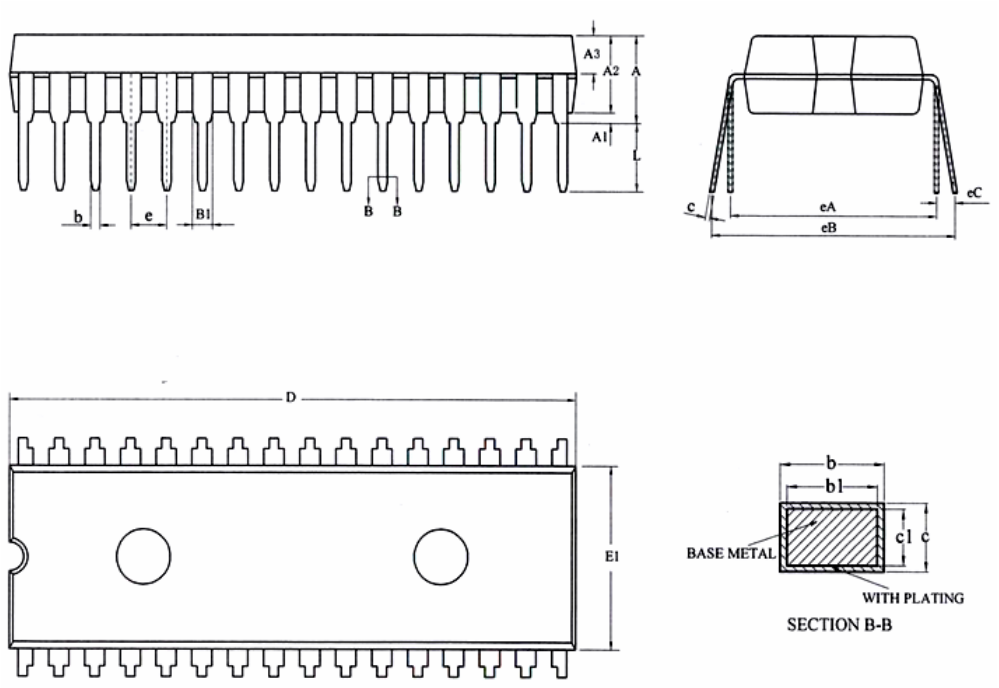


NOTES :
1. JEDEC OUTLINE : N/A.
2. DIMENSION b APPLIES TO METALLIZED TERMINAL AND IS MEASURED BETWEEN 0.15mm AND 0.30mm FROM THE TERMINAL TIP. IF THE TERMINAL HAS THE OPTIONAL RADIUS ON THE OTHER END OF THE TERMINAL, THE DIMENSION b SHOULD NOT BE MEASURED IN THAT RADIUS AREA.
3. THE MINIMUM "K" VALUE OF 0.20mm APPLIES.
4. BILATERAL COPLANARITY ZONE APPLIES TO THE EXPOSED HEAT SINK SLUG AS WELL AS THE TERMINALS.

SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.003	0.030	0.031	0.070	0.750	0.800
A1	0.000	0.001	0.002	0.000	0.020	0.050
A3	0.008 REF.			0.203 REF.		
b	0.007	0.010	0.012	0.180	0.250	0.300
D	0.20 BSC			5.00 BSC		
E	0.20 BSC			5.00 BSC		
e	0.02 BSC			0.50 BSC		
L	0.014	0.016	0.018	0.350	0.400	0.450
K	0.008	-	-	0.20	-	-

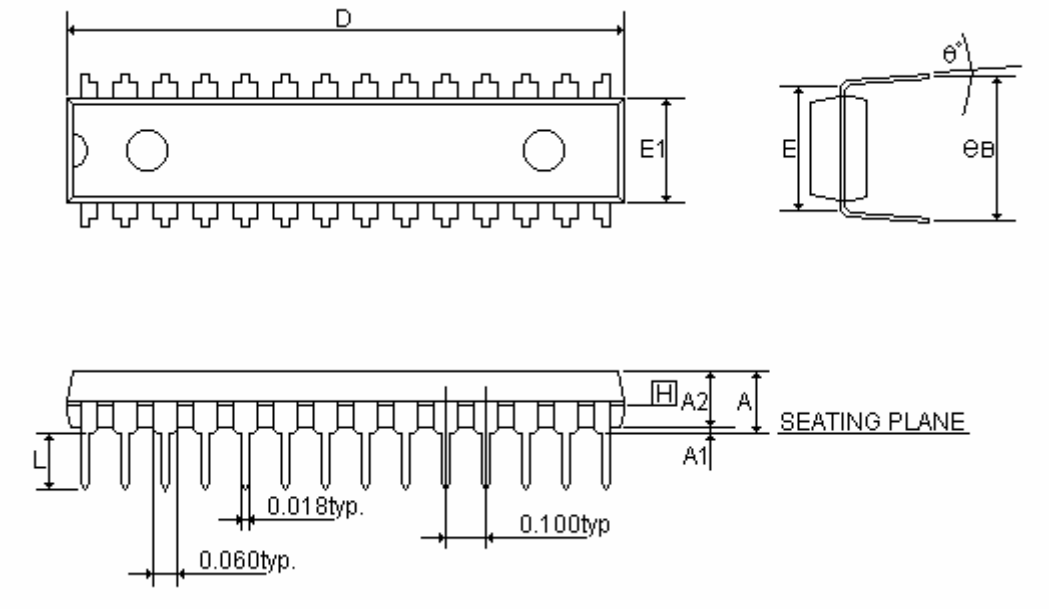
PAD SIZE	D2 (mm)			E2 (mm)		
	MIN	NOR	MAX	MIN	NOR	MAX
114x114 MIL	2.60	2.70	2.75	2.60	2.70	2.75
134x134 MIL	3.10	3.20	3.25	3.10	3.20	3.25

19.4 S-DIP 32 PIN



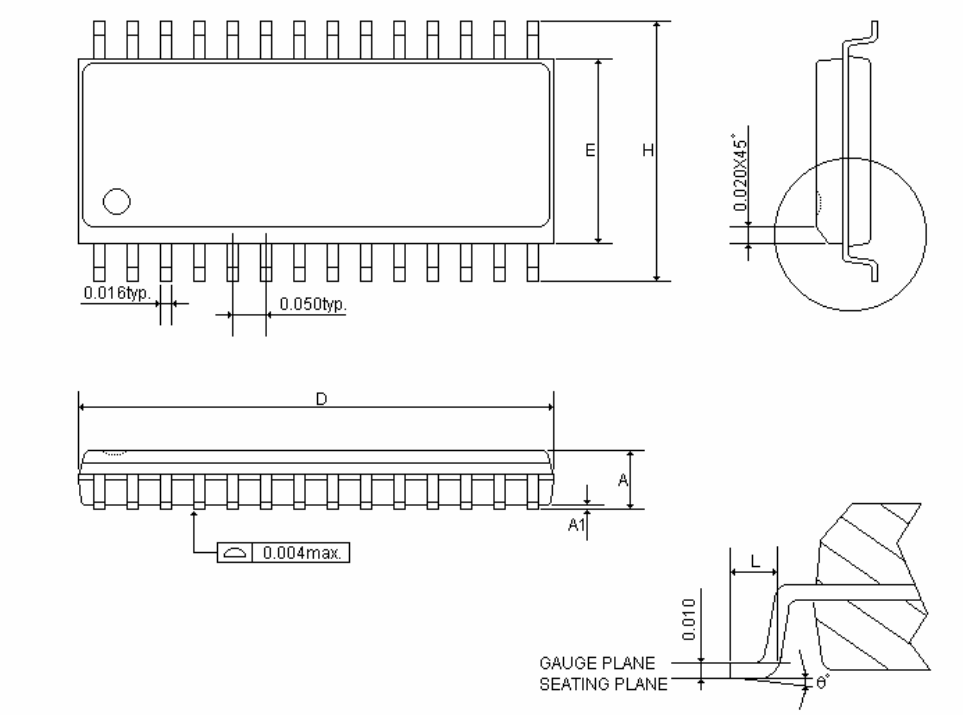
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.165	0.173	0.181	4.20	4.40	4.60
A1	0.043	-	-	1.10	-	-
A2	0.126	0.130	0.134	3.20	3.30	3.40
A3	0.058	0.060	0.062	1.47	1.52	1.57
b	0.017	-	0.021	0.44	-	0.53
b1	0.017	0.018	0.019	0.43	0.46	0.48
B1		0.039BSC			1.00BSC	
c	0.010	-	0.012	0.25	-	0.31
c1	0.009	0.010	0.010	0.24	0.25	0.26
D	1.094	1.102	1.110	27.8	28.00	28.20
E1	0.343	0.350	0.358	8.70	8.90	9.10
e		0.07BSC			1.778BSC	
eA		0.4BSC			10.16BSC	
eB	0.400	-	0.466	10.16	-	11.84
eC	0.000	-	0.033	0	-	0.84
L	0.118	-	-	3.00	-	-

19.5 SK-DIP 28 PIN



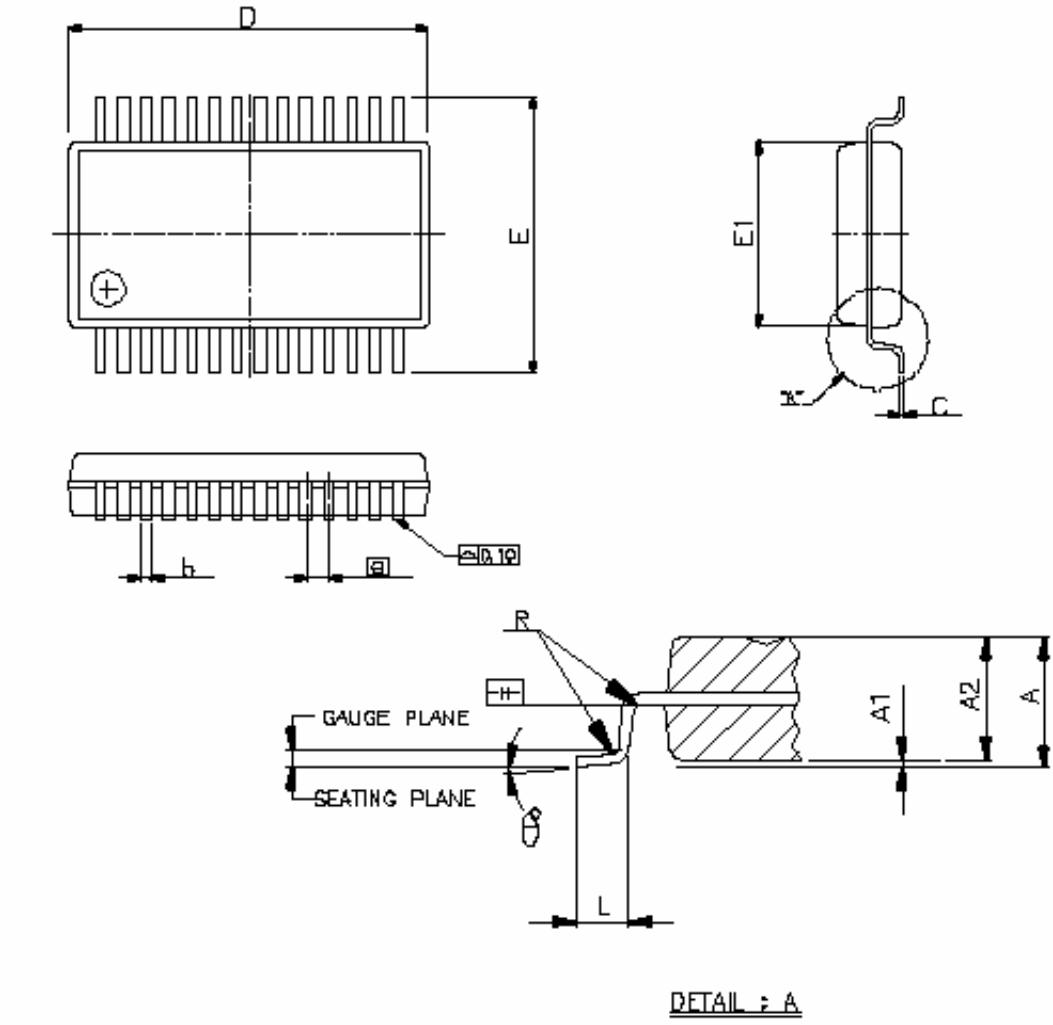
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-	-	0.381	-	-
A2	0.114	0.130	0.135	2.896	3.302	3.429
D	1.390	1.390	1.400	35.306	35.306	35.560
E		0.310			7.874	
E1	0.283	0.288	0.293	7.188	7.315	7.442
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.330	0.350	0.370	8.382	8.890	9.398
θ°	0°	7°	15°	0°	7°	15°

19.6 SOP 28 PIN



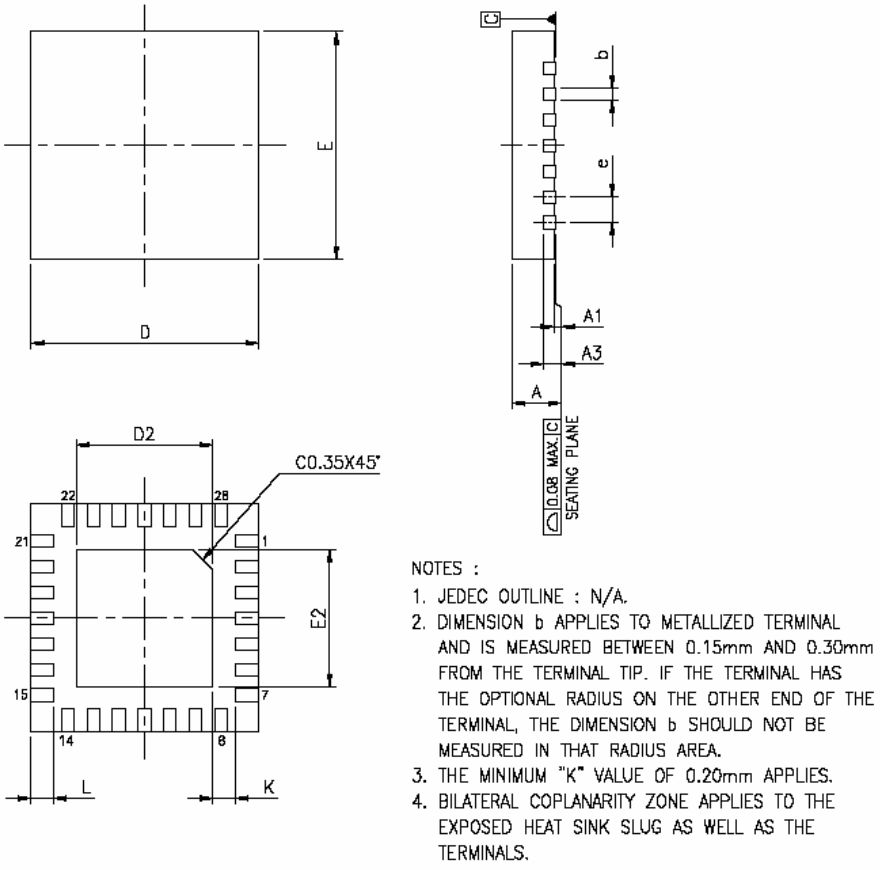
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.093	0.099	0.104	2.362	2.502	2.642
A1	0.004	0.008	0.012	0.102	0.203	0.305
D	0.697	0.705	0.713	17.704	17.907	18.110
E	0.291	0.295	0.299	7.391	7.493	7.595
H	0.394	0.407	0.419	10.008	10.325	10.643
L	0.016	0.033	0.050	0.406	0.838	1.270
θ°	0°	4°	8°	0°	4°	8°

19.7 SSOP 28 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.08	-	-	2.13
A1	0.00	-	0.01	0.05	-	0.25
A2	0.06	0.07	0.07	1.63	1.75	1.88
b	0.01	-	0.01	0.22	-	0.38
C	0.00	-	0.01	0.09	-	0.20
D	0.39	0.40	0.41	9.90	10.20	10.50
E	0.29	0.31	0.32	7.40	7.80	8.20
E1	0.20	0.21	0.22	5.00	5.30	5.60
[e]	0.0259BSC			0.65BSC		
L	0.02	0.04	0.04	0.63	0.90	1.03
R	0.00	-	-	0.09	-	-
θ°	0°	4°	8°	0°	4°	8°

19.8 QFN 4X4 28 PIN

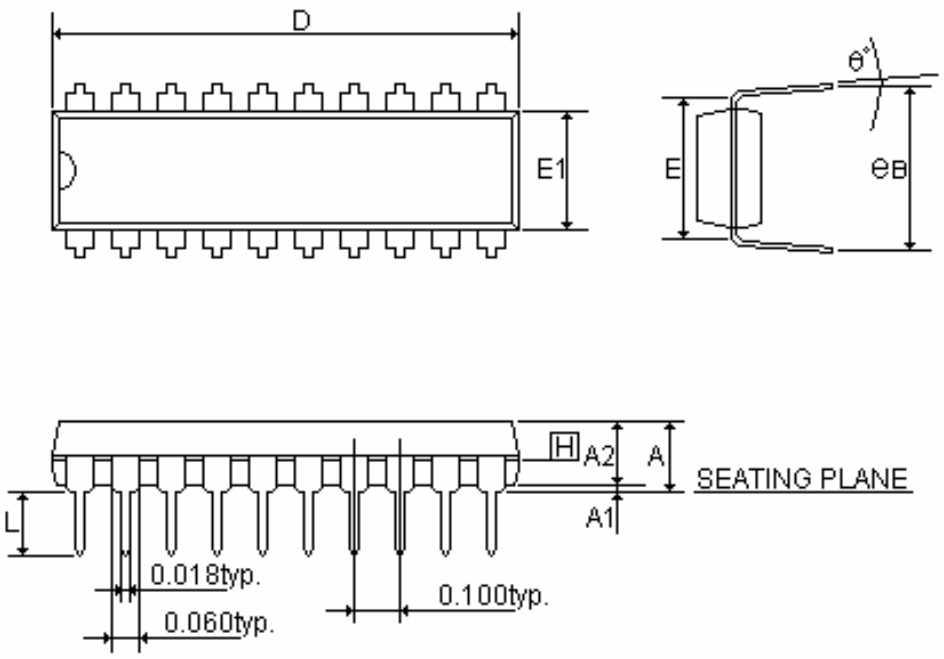


NOTES :
1. JEDEC OUTLINE : N/A.
2. DIMENSION b APPLIES TO METALLIZED TERMINAL AND IS MEASURED BETWEEN 0.15mm AND 0.30mm FROM THE TERMINAL TIP. IF THE TERMINAL HAS THE OPTIONAL RADIUS ON THE OTHER END OF THE TERMINAL, THE DIMENSION b SHOULD NOT BE MEASURED IN THAT RADIUS AREA.
3. THE MINIMUM "K" VALUE OF 0.20mm APPLIES.
4. BILATERAL COPLANARITY ZONE APPLIES TO THE EXPOSED HEAT SINK SLUG AS WELL AS THE TERMINALS.

SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.003	0.030	0.031	0.07	0.75	0.80
A1	0.000	0.001	0.002	0.00	0.02	0.05
A3	0.008 REF.			0.20 REF.		
b	0.006	0.008	0.010	0.15	0.20	0.25
e	0.016 BSC			0.40 BSC		
D	0.16 BSC			4.00 BSC		
E	0.16 BSC			4.00 BSC		
L	0.014	0.016	0.018	0.35	0.40	0.45
K	0.008	-	-	0.20	-	-

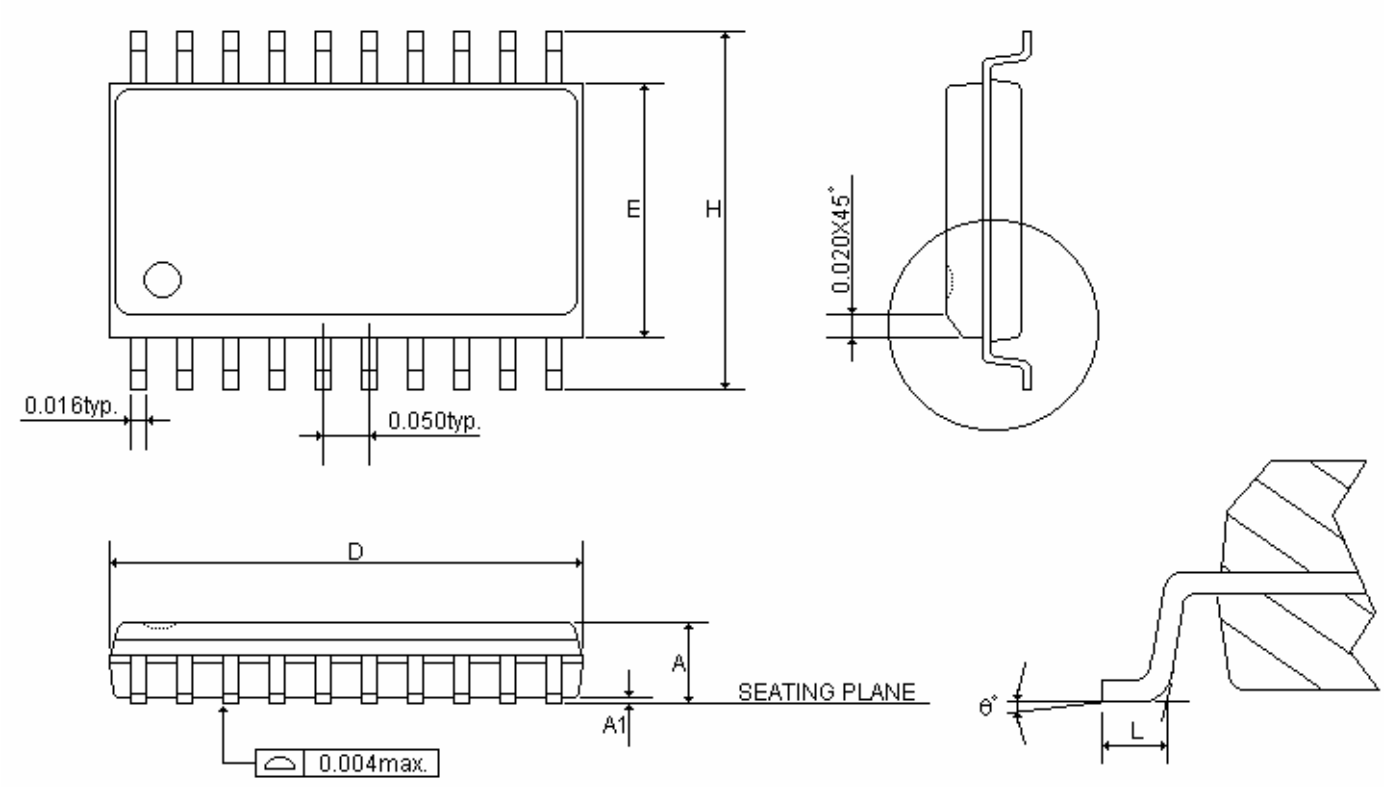
PAD SIZE	D2 (mm)			E2 (mm)		
	MIN	NOR	MAX	MIN	NOR	MAX
115 * 115 MIL	2.50	2.60	2.65	2.50	2.60	2.65

19.9 P-DIP 20 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-	-	0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	0.980	1.030	1.060	24.892	26.162	26.924
E		0.300			7.620	
E1	0.245	0.250	0.255	6.223	6.350	6.477
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.335	0.355	0.375	8.509	9.017	9.525
θ°	0°	7°	15°	0°	7°	15°

19.10 SOP 20 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.093	0.099	0.104	2.362	2.502	2.642
A1	0.004	0.008	0.012	0.102	0.203	0.305
D	0.496	0.502	0.508	12.598	12.751	12.903
E	0.291	0.295	0.299	7.391	7.493	7.595
H	0.394	0.407	0.419	10.008	10.325	10.643
L	0.016	0.033	0.050	0.406	0.838	1.270
θ°	0°	4°	8°	0°	4°	8°

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。

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：香港九龙湾宏开道 8 号其士商业中心 15 楼 1519 室
电话：852-2723 8086
传真：852-2723 9179
松翰科技（深圳）有限公司
地址：深圳市南山区高新技术产业园南区 T2-B 栋 2 层
电话：86-755-2671 9666
传真：86-755-2671 9786
技术支持：
Sn8fae@SONiX.com.tw