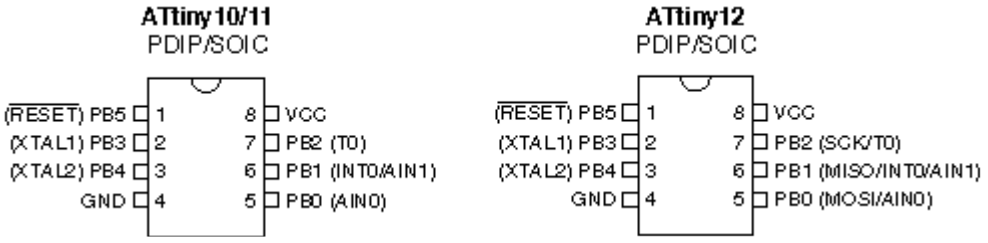
ATtiny10/11/12

特点

1. AVR RISC 结构
2. AVR—高性能、低功耗 RISC 结构
   —90 条指令——大多数为单指令周期
   —32 个 8 位通用（工作）寄存器
   —工作在 8MHz 时具有 8MIPS 的性能
3. 数据和非易失性程序内存
   —1K 字节的 FLASH
   QuickFlashTM（ATtiny10）
   ISP（ATtiny12）
   擦除次数：1000 次（ATtiny11/12）
   —64 字节在线可编程 EEPROM（ATtiny12）
   寿命：100000 次
   —程序加密位
4. 外围（Peripheral）特点
   —引脚电平变化中断及唤醒
   ——一个可预分频（Prescale）的 8 位定时器/计数器
   —片内模拟比较器
   —可编程的看门狗定时器（由片内振荡器生成）
5. 特别的 MCU 特点
   —低功耗空闲和掉电模式
   —内外部中断源
   —通过 SPI 口的 ISP（ATtiny12）
   —增强的上电复位电路（ATtiny12）
   —可标度的片内 RC 振荡器（）
6. 规范（Specification）
   —低功耗、高速 CMOS 工艺
   —全静态工作
7. 4MHz、3V、25℃条件下的功耗：
   —工作模式：2.2mA
   —空闲模式：0.5mA
   —掉电模式：<1μA
8. I/O 和封装
   —8 脚 PDIP 和 SOIC 封装
9. ATtiny10 是 ATtiny11 的 QuickFlash 版本
10. 工作电压
    —1.8V-5.5V（ATtiny12V-1）
    —2.7V-5.5V（ATtiny11L-2 和 ATtiny11L-4）
    —4.0V-5.5V（ATtiny11-6 和 ATtiny12-8）
11. 速度

—0-1 MHz（ATtiny12V-1）
—0-2 MHz（ATtiny11L-2）
—0-4 MHz（ATtiny12L-4）
—0-6MHz（ATtiny11-6）
—0-8MHz（ATtiny12-8）

管脚配置



描述

ATtiny10/11/12 是一款基于 AVR RISC 的低功耗 CMOS 的 8 位单片机。通过在一个时钟周期内执行一条指令， ATtiny10/11/12 可以取得接近 1MIPS/MHz 的性能，从而使得设计人员可以在功耗和执行速度之间取得平衡。

AVR 核将 32 个工作寄存器和丰富的指令集联结在一起。所有的工作寄存器都与 ALU（算逻单元）直接相连，允许在一个时钟周期内执行的单条指令同时访问两个独立的寄存器。这种结构提高了代码效率，使 AVR 得到了比普通 CISC 单片机高将近 10 倍的性能。

表 1　器件描述

器件	FLASH	EEPROM	寄存器	电压范围	频率
ATtiny10/11L	1K	-	32	2.7V-5.5V	0-2 MHz
ATtiny10/11	1K	-	32	4.0V-5.5V	0-6 MHz
ATtiny12V	1K	64B	32	1.8V-5.5V	0-1 MHz
ATtiny12L	1K	64B	32	2.7V-5.5V	0-4 MHz
ATtiny12	1K	64B	32	4.0V-5.5V	0-8 MHz

ATtiny10/11 具有以下特点：1K 字节 FLASH，多达 5 个通用 I/0 口，1 个输入口，32 个通用工作寄存器，一个 8 位 T/C，内外中断源，可编程的看门狗定时器，以及两种可通过软件选择的省电模式。工作于空闲模式时，CPU 将停止运行，而定时器/计数器和中断系统继续工作；掉电模式时振荡器停止工作，所有功能都被禁止，而寄存器内容得到保留。只有中断或硬件复位才可以退出此状态。引脚电平变化中断的特点使得 ATtiny10/11 对外部事件有很高的响应性，同时具有掉电模式的低功耗的优点。

器件是以 ATMEL 的高密度非易失性内存技术生产的。片内 FLASH 允许多次编程。通过将增强的 RISC 8 位 CPU 与 FLASH 集成在一个芯片内，ATtiny10/11 为许多嵌入式控制应用提供了灵活而低成本的方案。

ATtiny10/11 具有一整套的编程和系统开发工具：宏汇编、调试/仿真器、在线仿真器和评估板。

图 1　　ATtiny10/11 **结构方框图**

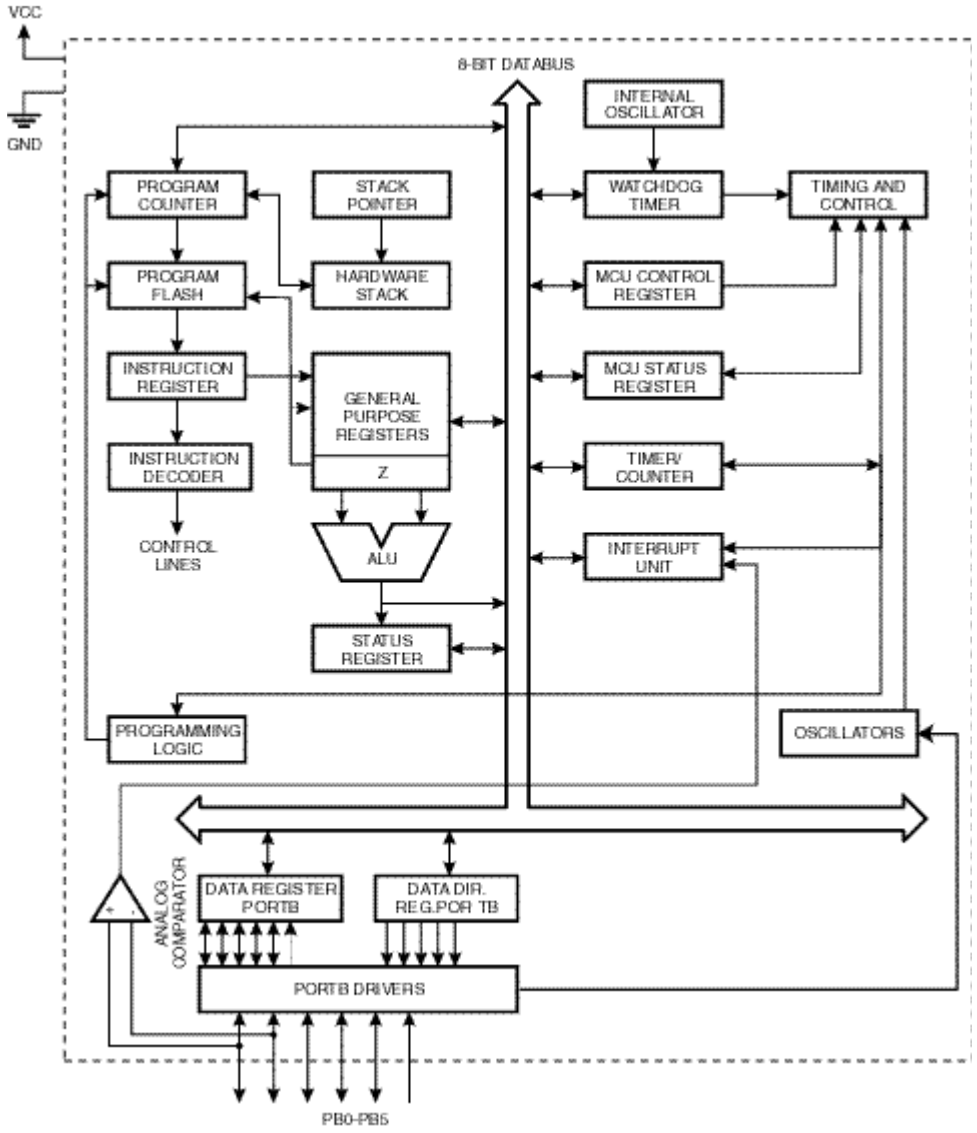
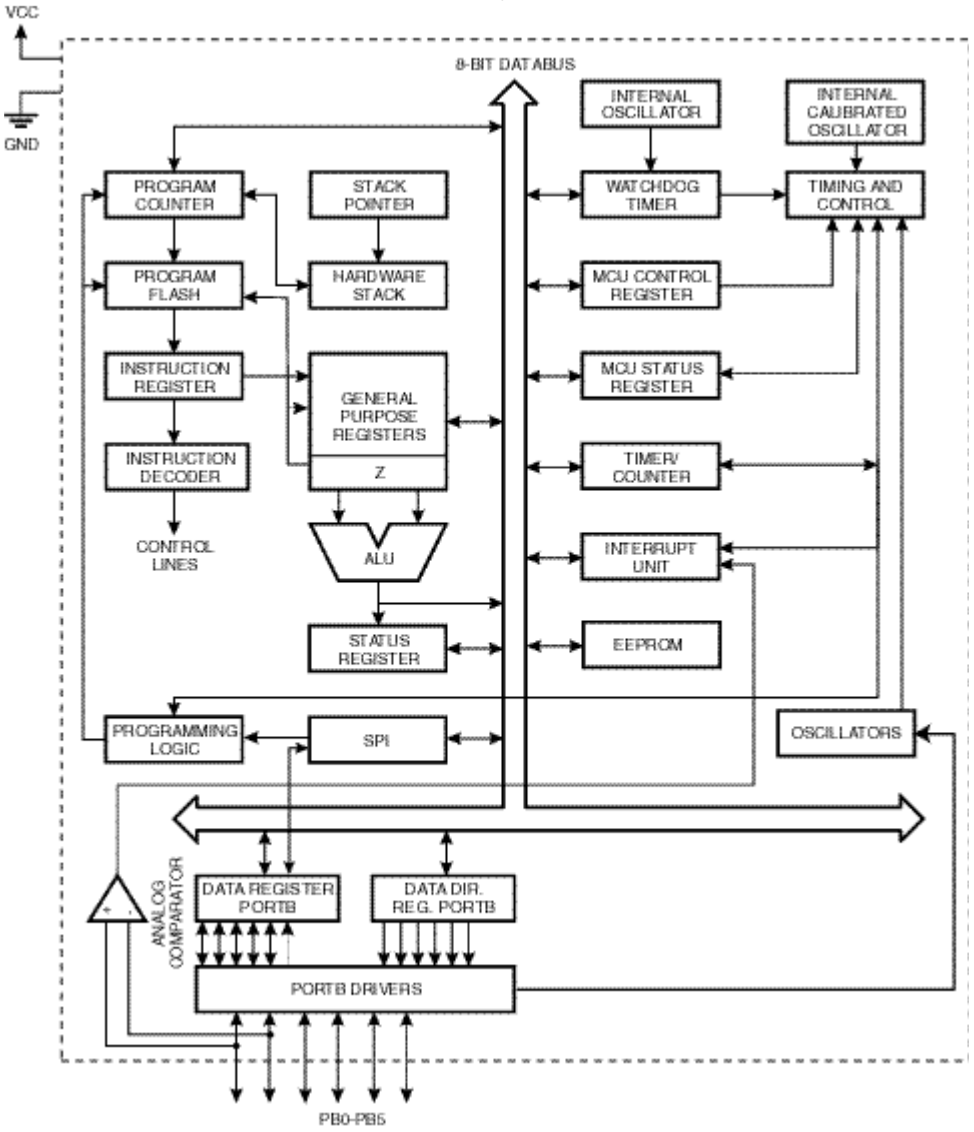


图 2　　ATtiny12 **结构方框图**



ATtiny12 **具有以下特点：** 1K **字节** FLASH，64 **字节** EEPROM，**多达** 6 **个通用** I/0 **口，** 32 **个通用工作寄存器，一个** 8 **位** T/C，**可编程的看门狗定时器，以及两种可通过软件选择的省电模式。工作于空闲模式时，** CPU **将停止运行，而定时器/计数器和中断系统继续工作；掉电模式时振荡器停止工作，所有功能都被禁止，而寄存器内容得到保留。只有外部中断或硬件复位才可以退出此状态。引脚电平变化中断的特点使得** ATtiny10/11 **对外部事件有很高的响应性，同时具有掉电模式的低功耗的优点。**

器件是以 ATMEL **的高密度非易失性内存技术生产的。片内** FLASH **允许多次编程。通过将增强的** RISC 8**位** CPU **与** FLASH **集成在一个芯片内，** ATtiny12 **为许多嵌入式控制应用提供了灵活而低成本的方案。**

ATtiny12 **具有一整套的编程和系统开发工具：宏汇编、调试/仿真器、在线仿真器和评估板。**

管脚定义

VCC、GND：电源

B 口（PB5..PB0）:

B **口是一个** 6 **位** I/O **口，**PB4 ..PB0 **有内部上拉电阻（可单独选择）。对于** ATtiny10/11，PB5 **是输入口，而对于** ATtiny12，PB5 **可以是输入口或是开漏输出口。在复位过程中，**B **口为三**

态，即使此时时钟还未起振。PB5 ..PB3 是用作输入还是 I/O 取决于复位和时钟设置，如下表所示。

表 2　PB5..PB3 功能与时钟设定的关系

时钟选择	PB5	PB4	PB3
外部复位使能	已用	-	-
外部复位禁止	输入/I/O [1)]	-	-
外部晶振	-	已用	已用
外部低频晶振	-	已用	已用
外部陶瓷振荡器	-	已用	已用
外部 RC 振荡器	-	I/O	已用
外部时钟	-	I/O	已用
内部 RC 振荡器	-	I/O	I/O

（1）对于 ATtiny10/11，PB5 是输入口，而对于 ATtiny12，PB5 可以是输入口或是开漏输出口。

XTAL1：振荡器放大器的输入端。

XTAL2：振荡器放大器的输出端。

晶体振荡器：

XTAL1 和 XTAL2 分别是片内振荡器的输入、输出端，可使用晶体振荡器或是陶瓷振荡器。当使用外部时钟时，XTAL2 应悬空。

时钟选择

器件具有多种时钟选择，由熔丝位控制。

表 3　器件时钟选择

器件时钟选择	ATtiny10/11 CKSEL2..0	ATtiny12 CKSEL3..0
外部晶振/陶瓷振荡器	111	1111 - 1010
外部低频晶振	110	1001 - 1000
外部 RC 振荡器	101	0111 - 0101
内部 RC 振荡器	100	0100 - 0010
外部时钟	000	0001 - 0000
保留	其他选择	-

注："1" 代表未编程，"0" 代表已编程。

内部 RC 振荡器

内部 RC 振荡器的频率固定为 1MHz。器件出厂时已经选择了这项功能。对于 ATtiny10/11，看门狗振荡器用作为时钟，而对于 ATtiny12 则使用了单独的可标度振荡器。

晶振

XTAL1 和 XTAL2 分别为片内反向放大器的输入和输出，如图 3 所示。

图 3　振荡器连接

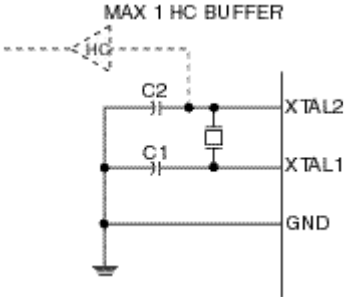
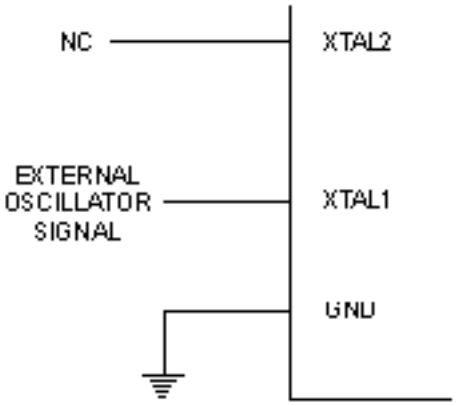


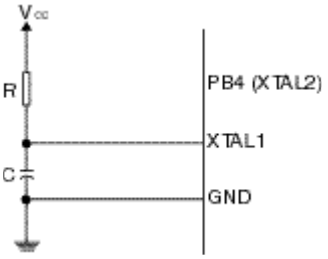
图 4　外部时钟驱动配置



外部 RC 振荡器

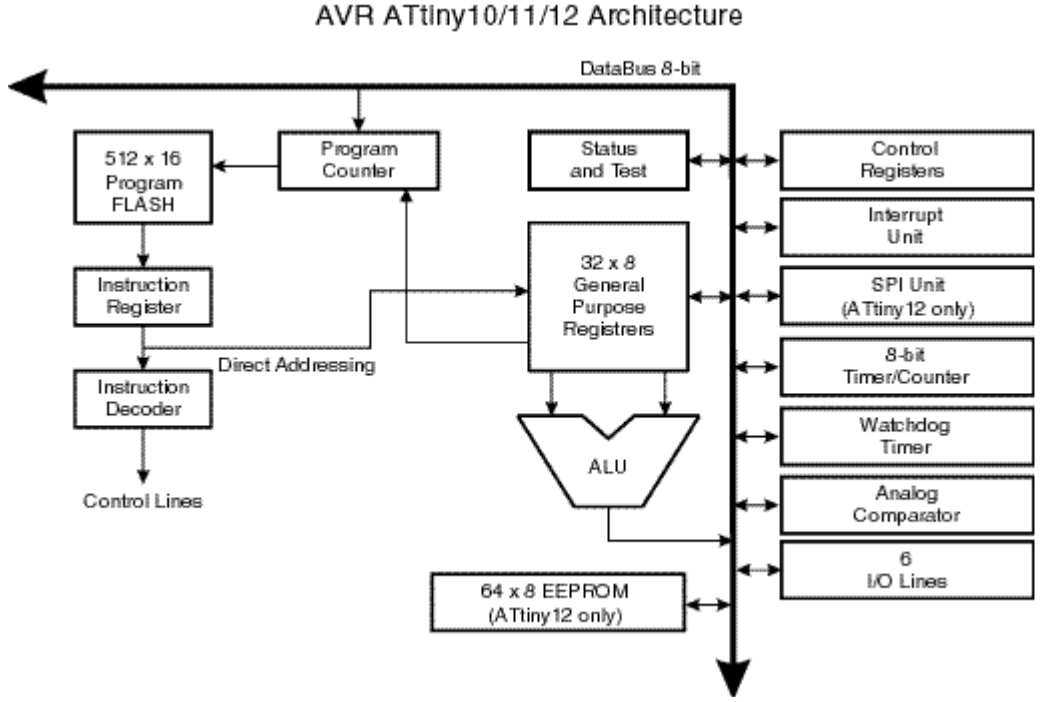
如果对时序不敏感，可以使用外部 RC **振荡器，如图** 5 **所示。**

图 5　外部 RC **配置**



结构纵览

图 6　ATtiny10/11/12 AVR RISC　**结构**



快速访问寄存器文件包含 32 个 8 位可单周期访问的通用寄存器。这意味着在一个时钟周期内，ALU 可以完成一次如下操作：读取寄存器文件中的两个操作数，执行操作，将结果存回到寄存器文件。
2 个寄存器可以用作访问存储区的 16 位指针，叫做 Z 指针。
ALU 支持两个寄存器之间、寄存器和常数之间的算术和逻辑操作，以及单寄存器的操作。
AVR 采用了 HARVARD 结构：程序和数据总线分离。程序内存通过两段式的管道（Pipeline）进行访问：当 CPU 在执行一条指令的同时，就去取下一条指令。这种预取指的概念使得指令可以在一个时钟完成。
相对跳转和相对调用指令可以直接访问 512 个地址空间。所有的 AVR 指令都为 16 位长，也就是说，每一个程序内存地址都包含一条 16 位的指令。
当执行中断和子程序调用时，返回地址存储于堆栈中。堆栈为 3 级硬件堆栈。
I/O 内存空间包含 64 个 CPU 外围的地址，如控制寄存器，T/C，和其他 I/O 功能。AVR 结构的内存空间是线性的。
中断模块由 I/O 空间中的控制寄存器和状态寄存器中的全局中断使能位组成。每个中断都具有一个中断向量，由中断向量组成的中断向量表位于程序存储区的最前面。中断向量地址低的中断具有高的优先级。

通用工作寄存器文件

图 7　AVR CPU 通用工作寄存器

通用寄存器	R0
	R1
	R2
	…
	…
	R28
	R29
	R30（Z 寄存器低字节）
	R31（Z 寄存器高字节）

所有的寄存器操作指令都可以单指令的形式直接访问所有的寄存器。例外情况为 5 条涉及常数操作的指令：SBCI、SUBI、CPI、ANDI 和 ORI。这些指令只能访问通用寄存器文件的后半部分：R16 到 R31。
寄存器 R30 及 R31 组成一个 16 位指针（Z 指针），用来间接访问程序区和寄存器文件。在访问寄存器文件时，R31 将被 CPU 忽略。

ALU

AVR ALU 与 32 个通用工作寄存器直接相连。ALU 操作分为 3 类：算术、逻辑和位操作。

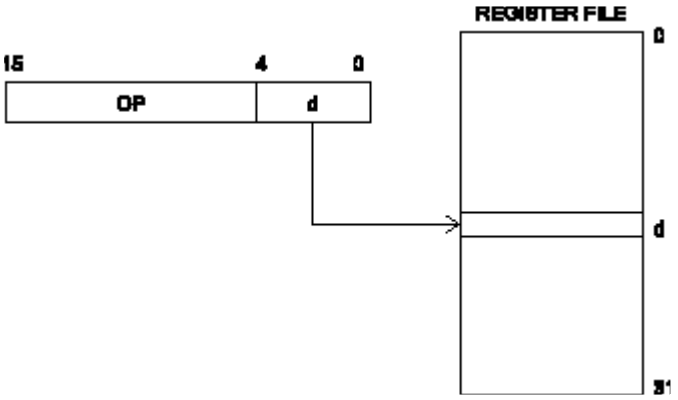
在线可编程 FLASH

ATtiny10/11/12 具有 1K 字节的 FLASH。因为所有的指令为 16 位宽，故尔 FLASH 结构为 512×16。FLASH 的擦除次数至少为 1000 次。
ATtiny10/11/12 的程序计数器（PC）为 9 位宽，可以寻址到 512 个字的 FLASH 程序区。

程序和数据寻址模式

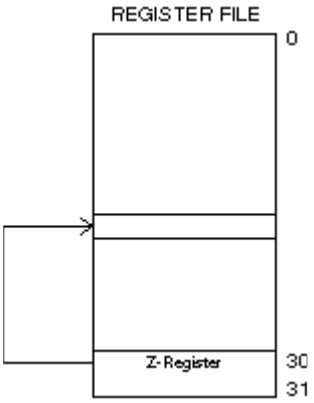
单寄存器直接寻址：

图 8



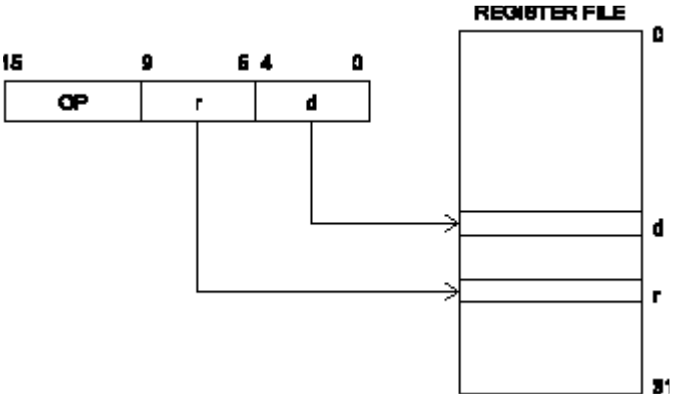
单寄存器间接寻址：

图 9



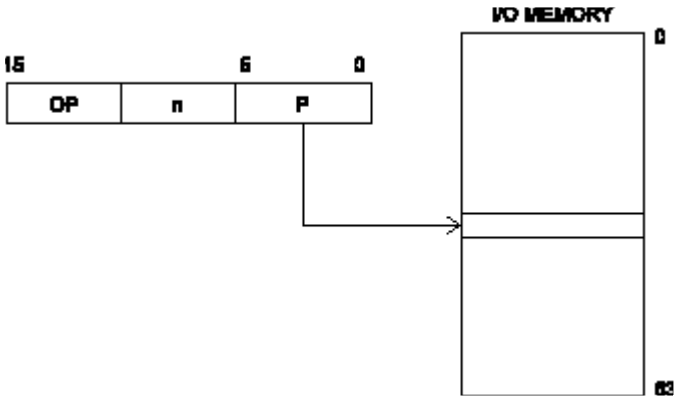
双寄存器直接寻址：

图 10



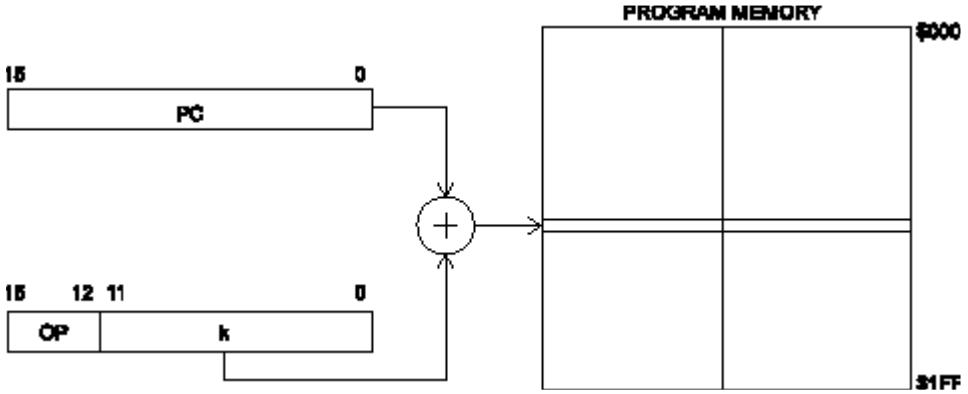
I/O 直接寻址：

图 11



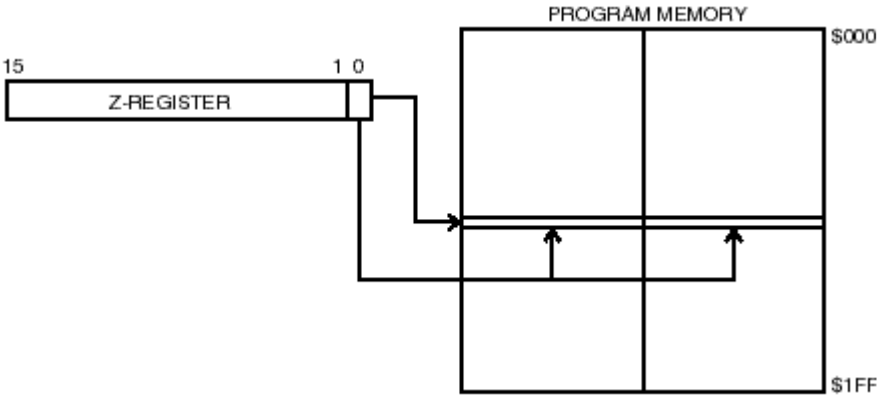
程序相对寻址：

图 12



利用 LPM指令的常数寻址：

图 13



常数字节地址由 Z **寄存器指定。前** 15 **位指定字地址，**LSB **指定低（**LSB=0**）或高（**LSB=1**）字节。**

子程序和中断硬件堆栈

ATtiny10/11/12 **使用** 3 **级硬件堆栈，宽度为** 9 **位。**

RCALL **指令和中断将** PC **推入堆栈** 0**，而原有的堆栈** 0 **和** 1 **的内容进入深一级的位置。执行** RET **或** RETI **后堆栈** 0 **的** PC **将出栈，而堆栈** 1 **和** 2 **的内容上升一级。**

如果有多于 3 **个的子程序或中断连续（嵌套）发生，则第一个进栈的内容将丢失。**

EEPROM

ATtiny12 包含 64 字节的 EEPROM，其写寿命为 100000 次。具体操作见后续章节。

指令操作时序

这一节介绍指令执行和内存访问时序。
AVR CPU 由系统时钟Φ驱动。此时钟由外部晶体直接产生。

图 14　取指与指令执行同时进行

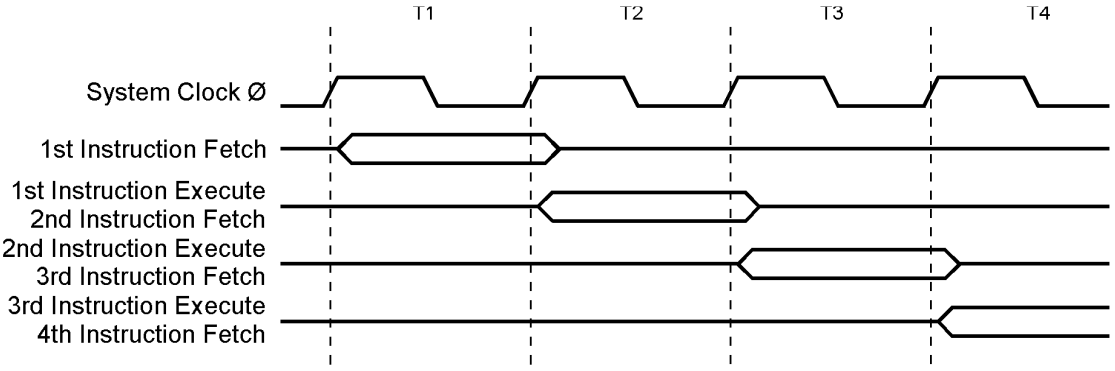
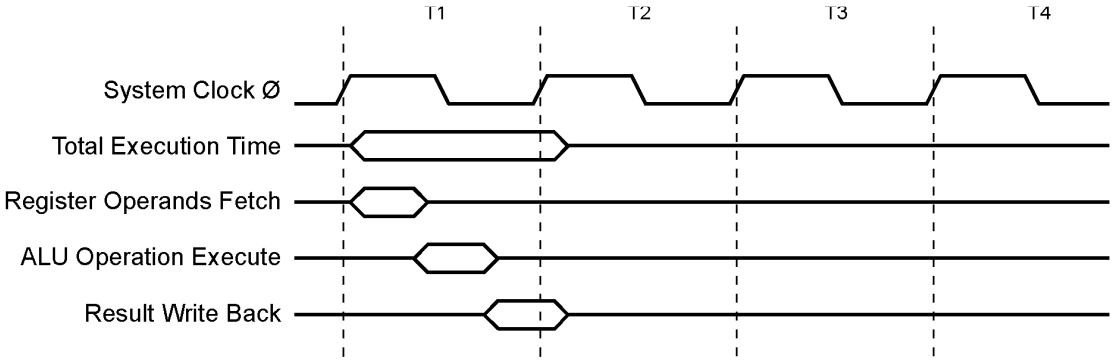


图 15　单时钟 ALU 操作



I/O 内存

表 4　ATtiny10/11/12 的 I/O 空间

地址	名称	器件	功能
$3F	SREG	ATtiny10/11/12	状态寄存器
$3B	GIMSK	ATtiny10/11/12	通用中断屏蔽寄存器
$3A	GIFR	ATtiny10/11/12	通用中断标志寄存器
$39	TIMSK	ATtiny10/11/12	T/C 屏蔽寄存器
$38	TIFR	ATtiny10/11/12	T/C 中断标志寄存器
$35	MCUCR	ATtiny10/11/12	MCU 控制寄存器
$34	MCUSR	ATtiny10/11/12	MCU 状态寄存器
$33	TCCR0	ATtiny10/11/12	T/C0 控制寄存器
$32	TCNT0	ATtiny10/11/12	T/C0（8 位）

$31	OSCCAL	ATtiny12	振荡器标度寄存器
$21	WDTCR	ATtiny10/11/12	看门狗控制寄存器
$1E	EEAR	ATtiny12	EEPROM 地址寄存器
$1D	EEDR	ATtiny12	EEPROM 数据寄存器
$1C	EECR	ATtiny12	EEPROM 控制寄存器
$18	PORTB	ATtiny10/11/12	B 口数据寄存器
$17	DDRB	ATtiny10/11/12	B 口数据方向寄存器
$16	PINB	ATtiny10/11/12	B 口输入引脚
$08	ACSR	ATtiny10/11/12	模拟比较器控制状态寄存器

AVRATtiny10/11/12 的所有 I/O 和外围都被放置在 I/O 空间。IN 和 OUT 指令用来访问不同的 I/O 地址，以及在 32 个通用寄存器之间传输数据。地址为$00-$1F 的 I/O 寄存器还可用 SBI 和 CBI 指令进行位寻址，而 SIBC 和 SIBS 则用来检查单个位置位与否。
为了与后续产品兼容，保留未用的位应写“0”，而保留的 I/O 寄存器则不应写。
I/O 寄存器和外围控制寄存器在后续章节介绍。

状态寄存器 **SREG**（Status Register）

BIT	7	6	5	4	3	2	1	0
$3F	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

I：全局中断使能
置位时使能全局中断。单独的中断使能由个独立控制寄存器控制。如果 I 清零，则不论单独中断标志置位与否，都不会产生中断。I 在复位时清零，RETI 指令执行后置位。
T：位拷贝存储
位拷贝指令 BLD 和 BST 利用 T 作为目的或源地址。BST 把寄存器的某一位拷贝到 T，而 BLD 把 T 拷贝到寄存器的某一位。
H：半加标志位
S：符号位
总是 N 与 V 的异或。
V：二进制补码溢出标志位
N：负数标志位
Z：零标志位
C：进位标志位
状态寄存器在进入中断和退出中断时并不自动进行存储和恢复。这项工作由软件完成。

复位和中断处理

ATtiny10/11 有 4 个中断源，而 ATtiny12 有 5 个。每个中断源在程序空间都有一个独立的中断向量。这 3 个中断事件有自己的使能位。当使能位置位，且 I 也置位的情况下，中断可以发生。
器件复位后，程序空间的最低位置自动定义为中断向量。完整的中断表见表 5。在中断向量表中处于低地址的中断具有高的优先级。所以，RESET 具有最高的优先级。

表 5　复位与中断向量

向量	器件	程序地址	来源	定义
1	ATtiny10/11	$000	RESET	硬件管脚，上电复位和看门狗复位
1	ATtiny12	$000	RESET	硬件管脚，上电、BOD 及看门狗复位
2	ATtiny10/11/12	$001	INT0	外部中断 0
3	ATtiny10/11/12	$002	I/O 口	引脚电平变化中断
4	ATtiny10/11/12	$003	TIMER0，OVF0	T/C0 溢出
5	ATtiny10/11	$004	ANA_COMP	模拟比较器
5	ATtiny12	$004	EE_RDY	EEPROM 准备好
6	ATtiny12	$005	ANA_COMP	模拟比较器

设置中断向量地址（ATtiny10/11）最典型的方法如下：

```
地址        标号        代码                    注释
$000                    RJMP    RESET           ；复位
$001                    RJMP    EXT_INT0        ；IRQ0
$002                    RJMP    PIN_CHANGE      ；引脚变化
$003                    RJMP    TIM0_OVF        ；T0 溢出
$004                    RJMP    ANA_COMP        ；模拟比较器
；
$005        MAIN:       <指令>  XXX             ；主程序开始
—           —           —       —
```

设置中断向量地址（ATtiny12）最典型的方法如下：

```
地址        标号        代码                    注释
$000                    RJMP    RESET           ；复位
$001                    RJMP    EXT_INT0        ；IRQ0
$002                    RJMP    PIN_CHANGE      ；引脚变化
$003                    RJMP    TIM0_OVF        ；T0 溢出
$004                    RJMP    EE_RDY          ；EEP 准备好
$005                    RJMP    ANA_COMP        ；模拟比较器
；
$006        MAIN:       <指令>  XXX             ；主程序开始
—           —           —       —
```

复位源

ATtiny10/11/12 有 3/4 个复位源：

- 上电复位。当电源电压低于上电门限 V_{POT} 时 MCU 复位。
- 外部复位。当/RESET 引脚上的低电平超过 50ns 时 MCU 复位。
- 看门狗复位。看门狗定时器超时后 MCU 复位。
- BROWN-OUT 复位。当电源电压 V_{CC} 低于一个特定值时 MCU 复位。

在复位期间，所有的 I/O 寄存器被设置为初始值，程序从地址$000 开始执行。$000 地址中放置的指令必须为 RJMP一相对跳转指令一跳转到复位处理例程。若程序永远不需中断，则中断向量就可放置通常的程序代码。图 16 为 ATtiny10/11 复位电路的逻辑图，图 17 为 ATtiny12 复位电路的逻辑图。表 6 定义了 ATtiny10/11 的复位电路的时序和电参数，表 8 定义了 ATtiny12 的复位电路的时序和电参数。

图 16　ATtiny10/11 的复位逻辑

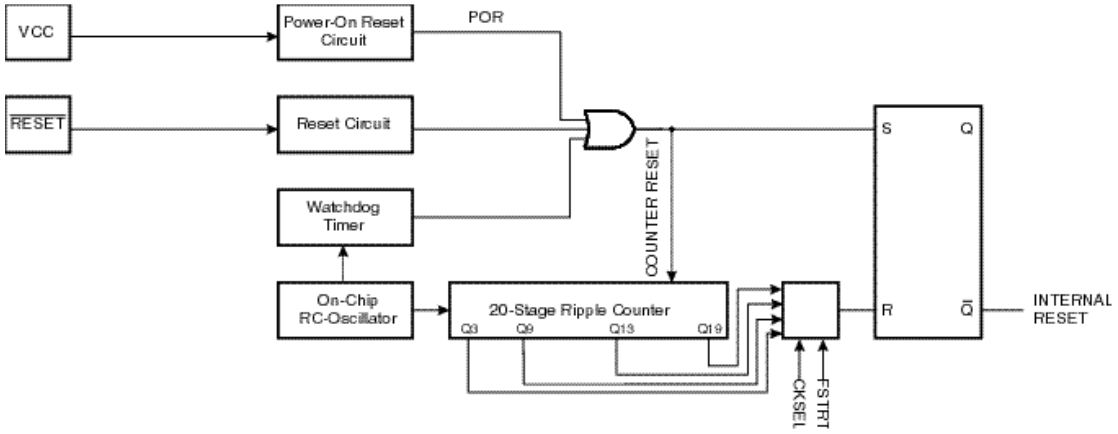


表 6　复位电参数（ATtiny10/11）

符号	参数	最小值	典型值	最大值	单位
V_{POT}[(1)]	上电复位电压门限（上升）	1.0	1.4	1.8	V
	上电复位电压门限（下降）	0.4	0.6	0.8	V
V_{RST}	管脚门限电压	-	$0.6V_{CC}$	-	V

注：1.除非电源电压低于 V_{POT}，否则上电复位不会发生。

ATtiny10/11 的上电复位

上电复位（POR）保证器件在上电时正确复位。如图 16 所示，在电源电压达到 V_{POT}后，片内定时器将启动延时 t_{TOUT}起动。FSTRT 编程后可以缩短起动时间。不同时钟选择的起动时间见表 7。看门狗振荡器用来定时起动时间。

表 7　ATtiny10/11 的起动时间（V_{CC} = 2.7V）

时钟	起动时间 t_{TOUT}	
	FSTRT 未编程	FSTRT 已编程
外部晶振	67ms	4.2ms
外部陶瓷振荡器	67ms	4.2ms
外部低频晶振	4.2s	4.2s
外部 RC 振荡器	4.2ms	67μs
内部 RC 振荡器	4.2ms	67μs
外部时钟	4.2ms	复位时需 5 个时钟 掉电唤醒时需 3 个时钟

如果内置于片内的启动时间足够的话，/RESET 可以与 V_{CC}直接相连，或是外接上拉电阻。如果在加上 V_{CC}的同时保持/RESET 为低，则可以延长复位周期。例子可参看图 19。

图 17　ATtiny12 的复位逻辑

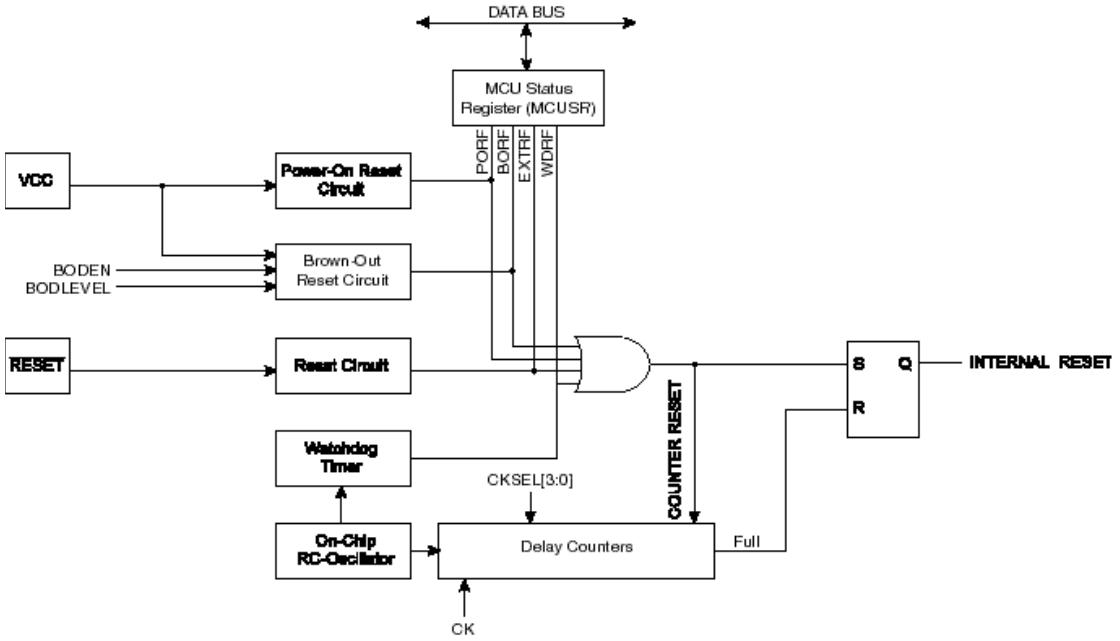


表 8　复位电参数（ATtiny12）

符号	参数	条件	最小值	典型值	最大值	单位
V_{POT} (1)	上电复位电压门限（上升）	BOD 禁止	1.0	1.4	1.8	V
		BOD 使能	0.6	1.2	1.8	V
	上电复位电压门限（下降）	BOD 禁止	0.4	0.6	0.8	V
		BOD 使能	0.6	1.2	1.8	V
V_{RST}	管脚门限电压		-	$0.6V_{CC}$	-	V
V_{BOT}	上电复位电压门限（上升）	（BODLEVEL=1）	1.7	1.8	1.9	V
		（BODLEVEL=1）	2.6	2.7	2.8	V

注：1.**除非电源电压低于** V_{POT}，**否则上电复位不会发生。**

表 9　ATtiny12 **的起动时间**

CKSEL3..0	时钟	起动时间，V_{CC} =1.8V, BODLEVEL 未编程	起动时间，V_{CC} =2.7V, BODLEVEL 已编程
1111	外部晶振/陶瓷振荡器 [1]	1K CK	1K CK
1110	外部晶振/陶瓷振荡器 [1]	3.6ms + 1K CK	4.2ms + 1K CK
1101	外部晶振/陶瓷振荡器 [1]	57ms + 1K CK	67ms + 1K CK
1100	外部晶振/陶瓷振荡器	16K CK	16K CK
1011	外部晶振/陶瓷振荡器	3.6ms + 16K CK	4.2ms + 16K CK
1010	外部晶振/陶瓷振荡器	57ms + 16K CK	67ms + 16K CK
1001	外部低频晶振	57ms + 1K CK	67ms + 1K CK
1000	外部低频晶振	57ms + 32K CK	67ms + 32K CK
0111	外部 RC 振荡器	6 CK	6 CK
0110	外部 RC 振荡器	3.6ms + 6 CK	4.2ms + 6 CK
0101	外部 RC 振荡器	57ms + 6 CK	67ms + 6 CK
0100	内部 RC 振荡器	6 CK	6 CK
0011	内部 RC 振荡器	3.6ms + 6 CK	4.2ms + 6 CK
0010	内部 RC 振荡器	57ms + 6 CK	67ms + 6 CK
0001	外部时钟	6 CK	6 CK
0000	外部时钟	3.6ms + 6 CK	4.2ms + 6 CK

注意：由于起动阶段有限的时钟周期，推荐使用陶瓷振荡器。

此表说明的是复位起动时间。而休眠唤醒时间只用到了起动时间的记数部分。看门狗计数器用来完成记数工作。具体见表 10。

表 10　看门狗振荡器周期数

BODLEVEL	溢出时间	周期数
未编程	3.6ms（V_{CC} =1.8V）	256
未编程	57ms（V_{CC} =1.8V）	4K
已编程	4.2ms（V_{CC} =2.7V）	1K
已编程	67ms（V_{CC} =2.7V）	16K

看门狗振荡器的频率与电源电压有关。

即使 BOD 功能禁止，BODLEVEL 也可以用于选择起动时间。

出厂时 CKSEL3..0 = 0010。

ATtiny12 的上电复位

POR 由片内检测电路产生。标称检测电平为 1.4V。只要 V_{CC} 低于检测电平 POR 就工作。上电复位（POR）保证器件在上电时正确复位。如图 16 所示，在电源电压达到 V_{POT} 后，片内定时器将启动延时 t_{TOUT} 起动。FSTRT 编程后可以缩短起动时间。不同时钟选择的起动时间见表 9。看门狗振荡器用来定时起动时间。

如果内置于片内的启动时间足够的话，/RESET 可以与 V_{CC} 直接相连，或是外接上拉电阻。如果在加上 V_{CC} 的同时保持/RESET 为低，则可以延长复位周期。例子可参看图 19。

图 18　MCU 启动，/RESET 与 V_{CC} 相连

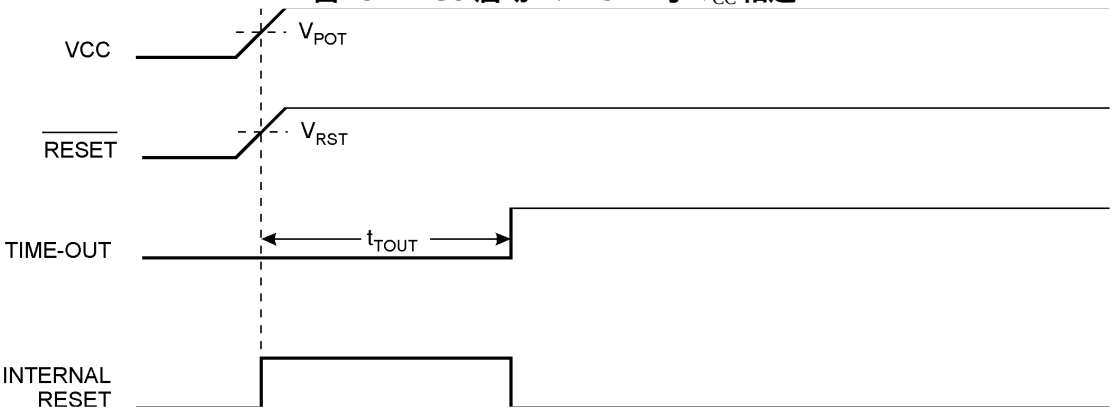
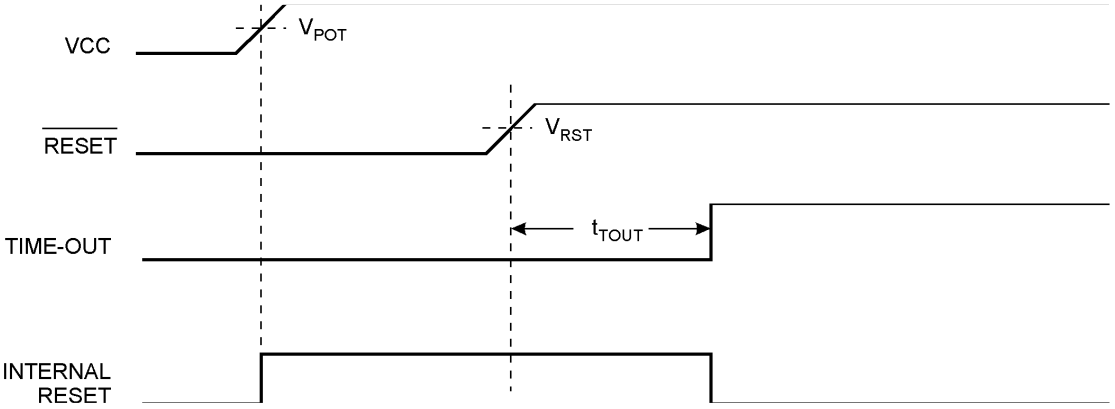


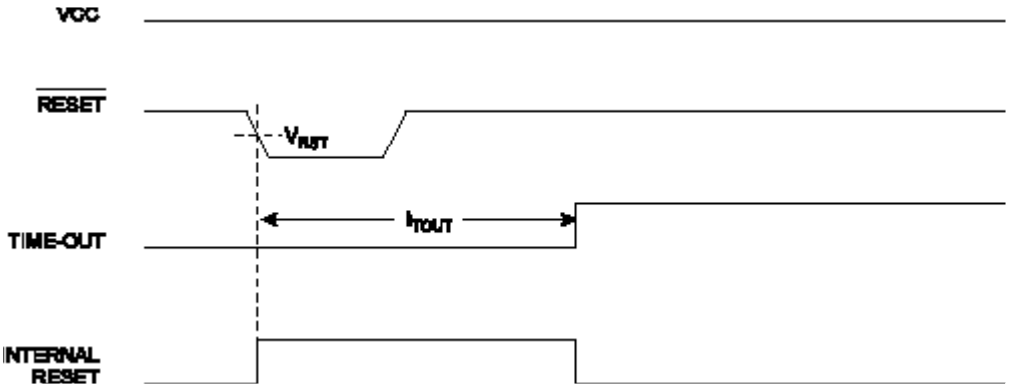
图 19　MCU 启动，/RESET 由外电路控制



外部复位

外部复位由外加于/RESET **引脚的低电平产生。大于** 50ns **的复位脉冲将造成芯片复位。施加短脉冲不能保证可靠复位。当外加信号达到复位门限电压** V_{RST}（**上升沿**）**时，** t_{TOUT} **延时周期开始。然后，** MCU **启动。**

图 20　工作期间的外部复位

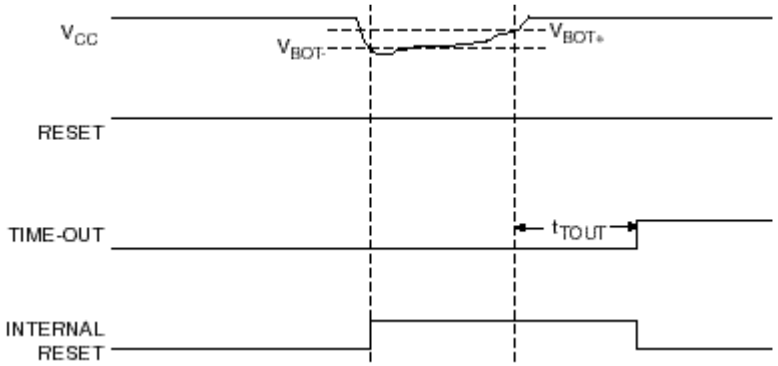


Brown-Out 检测（ATtiny12）

ATtiny12 **具有片内** BOD **检测电路用以监测** V_{CC}**。**BOD **可由熔丝位** BODEN **控制。如果** BODEN **使能（被编程），则只要** V_{CC} **低于触发电平，**BOD **就立即工作。等到电压回升到触发电平后，器件等待一段时间，然后** BOD **复位。延迟时间由表** 9 **决定。** BOD **触发电平由** BODLEVEL **控制为** 1.8V（BODLEVEL **未编程）或** 2.7V（BODLEVEL **已编程）。触发电平有** 50mV **的冗余。**

触发电平为 2.7V **时，只要** V_{CC} **低于这个值** 7μs，BOD **就起作用；而当触发电平为** 1.8V **时时间延长到** 24μs**。**

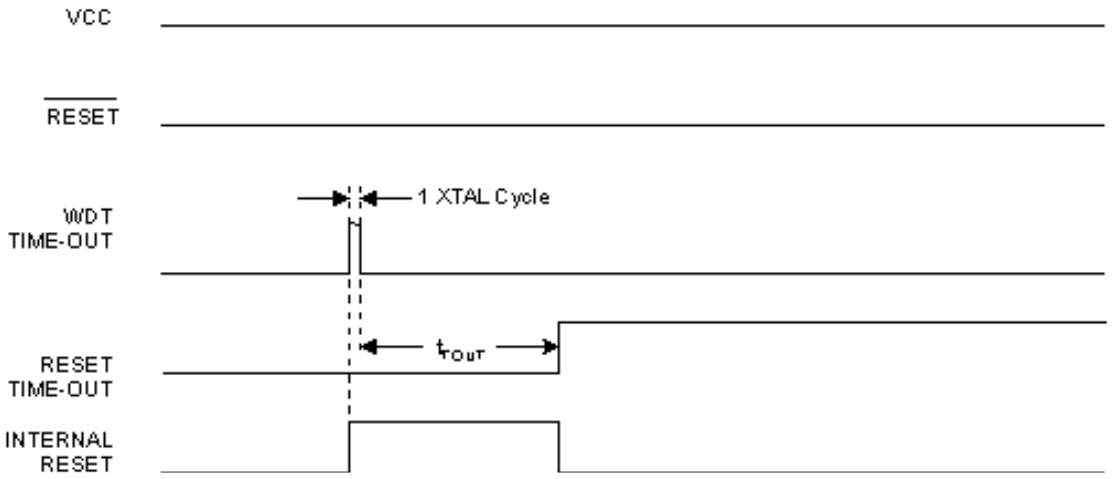
图 21　工作期间的 BOD **复位**



看门狗复位

当看门狗定时器溢出时，将产生 1 **个** XTAL **周期的复位脉冲。在脉冲的下降沿，延时定时器开始对** t_{TOUT} **记数。**

图 22　工作期间的看门狗复位



ATtiny10/11 的 MCU 状态寄存器—MCUSR

BIT	7	6	5	4	3	2	1	0
$34	-	-	-	-	-	-	**EXTRF**	**PORF**
读/写	R	R	R	R	R	R	R/W	R/W
初始值	0	0	0	0	0	0		

位 7..2：保留

EXTRF：外部复位标志

上电复位时这一位没有定义（X）。外部复位时置位。看门狗复位对其没有影响。

PORF：上电复位标志

由上电复位置位。看门狗复位或外部复位对其没有影响。

表 11　复位后的 PORF 和 EXTFR

复位源	PORF	EXTRF
上电复位	1	没有定义
外部复位	不变化	1
看门狗复位	不变化	不变化

如果要利用 PORF 和 EXTRF 来识别复位条件，用户软件要尽早对其清零。检查 PORF 和 EXTRF 的语句在对其清零之前执行。如果某一位在外部复位或看门狗复位之前清零，则复位可以通过如下真值表找出来：

表 12　复位源鉴别

PORF	EXTRF	复位源
0	0	看门狗复位
0	1	外部复位
1	0	上电复位
1	1	上电复位

ATtiny12 的 MCU 状态寄存器—MCUSR

BIT	7	6	5	4	3	2	1	0
$34	-	-	-	-	**WDFR**	**BORF**	**EXTRF**	**PORF**
读/写	R	R	R	R	R/W	R/W	R/W	R/W
初始值	0	0	0	0				

位 7..4：保留

WDRF：看门狗复位标志

看门狗复位时这一位置位。上电复位或对其写“0”将使其清零。

BORF：BOD 复位标志

BOD 复位时这一位置位。上电复位或对其写“0”将使其清零。

EXTRF：外部复位标志

外部复位时这一位置位。上电复位或对其写“0”将使其清零。

PORF：上电复位标志

上电复位时这一位置位。对其写“0”将使其清零。

如果要利用这些复位标志来识别复位条件，用户软件要尽早对其清零。如果寄存器在其他复位发生前清零，则以后的复位源可以通过检查这些标志位找出来。

中断处理：

ATtiny10/11/12 有 2 个中断屏蔽控制寄存器 GIMSK—通用中断屏蔽寄存器和 TIMSK—T/C 中断屏蔽寄存器。

一个中断产生后，全局中断使能位 I 将被清零，后续中断被屏蔽。用户可以在中断例程里对 I 置位，从而开放中断。执行 RETI 后 I 重新置位。

当程序计数器指向实际中断向量开始执行相应的中断例程时，硬件清除对应的中断标志。一些中断标志位也可以通过软件写“1”来清除。

当一个符合条件的中断发生后，如果相应的中断使能位为“0”，则中断标志位置位，并一直保持到中断执行，或者被软件清除。

如果全局中断标志被清零，则所有的中断都不会被执行，直到 I 置位。

注意：外部电平中断没有中断标志位，因此当电平变为非中断电平后，中断条件即终止。

进入中断和退出中断时 MCU 不会自动保存或恢复状态寄存器，故尔需由软件处理。

通用中断屏蔽寄存器—GIMSK

BIT	7	6	5	4	3	2	1	0
$3B	-	**INT0**	**PCIE**	-	-	-	-	-
读/写	R	R/W	R/W	R	R	R	R	R
初始值	0	0	0	0	0	0	0	0

位 7、4..0：保留

INT0：外部中断 0 请求使能

当 INT0 和 I 都为“1”时，外部引脚中断使能。MCU 通用控制寄存器（MCUCR）中的中断检测控制位 1/0（ISC01 和 ISC00）定义中断 0 是上升沿/下降沿中断、电平变化中断，还是低电平中断。即使管脚被定义为输出，中断仍可产生。

PCIE：引脚电平变化中断

当 PCIE 和 I 都为“1”时，引脚上的电平变化将触发中断。

通用中断标志寄存器—GIFR

BIT	7	6	5	4	3	2	1	0
$3A	-	**INTF0**	**PCIF**	-	-	-	-	-
读/写	R	R/W	R/W	R	R	R	R	R
初始值	0	0	0	0	0	0	0	0

位 7、4..0：保留

INTF0：外部中断 0 标志

如果 SREG 的位 I 及 GIMSK 的位 INT0 都为“1”，则 MCU 跳转到$001 执行 INT0 中断。执行中断例程时此位硬件清零，也可以对其写“1”清零。如果 INT0 配置为电平中断，则这一位一直为“0”。

PCIF：引脚电平变化中断标志

如果 SREG **的位** I **及** GIMSK **的位** PCIE **都为“**1**”，则** MCU **跳转到**$002 **执行电平变化中断。执行中断例程时此位硬件清零，也可以对其写“**1**”清零。**

T/C 中断屏蔽寄存器—TIMSK

BIT	7	6	5	4	3	2	1	0
$39	-	-	-	-	-	-	**TOIE0**	-
读/写	R	R	R	R	R	R	R/W	R
初始值	0	0	0	0	0	0	0	0

位 7..2、0：**保留**

TOIE0：T/C0 溢出中断使能

当 TOIE0 **和** I **都为“**1**”时，**T/C0 **溢出中断使能。当** T/C0 **溢出，或** TIFR **中的** TOV0 **位置位时，中断例程（**$002**）得到执行。**

T/C 中断标志寄存器—TIFR

BIT	7	6	5	4	3	2	1	0
$38	-	-	-	-	-	-	**TOV0**	-
读/写	R	R	R	R	R	R	R/W	R
初始值	0	0	0	0	0	0	0	0

位 7..2、0：**保留**

TOV0：T/C0 溢出中断标志位

当 T/C0 **溢出时，**TOV0 **置位。执行相应的中断例程后此位硬件清零。此外，**TOV0 **也可以通过写“**1**”来清零。当** SREG **中的位** I、TOIE0 **和** TOV0 **一同置位时，中断例程得到执行。**

外部中断：

外部中断由 INT0 **引脚触发。触发方式可以为上升沿，下降沿、低电平或电平变化。这些设置由** MCU **控制寄存器** MCUCR **决定。当** INT0 **设置为低电平触发时，只要电平为低，中断就一直挂起。**

即使 INT0 **配置为输出中断也会发生。这种特性可以用来实现软件中断。**

引脚电平变化中断：

此中断由任何输入或 I/O **口上的电平变化触发。**PB2..0 **上的电平变化总是会引起中断。而**PB5..3 **要配置为输入或** I/O **口才可以。要即使管脚配置为输出中断也会发生。这种特性可以用来实现软件中断。还要注意引脚变化中断即使在某一个引脚的活动触发其他中断时也会发生，例如外部中断。这说明一个事件可能触发好几个中断。**

中断响应时间：

AVR **中断响应时间最少为** 4 **个时钟周期。在这** 4 **个时钟期间，**PC **自动入栈。在通常情况下，中断向量为一个相对跳转指令，此跳转要花** 2 **个时钟周期。如果中断在一个多周期指令执行期间发生，则在此多周期指令执行完后** MCU **才会执行中断程序。**

中断返回亦需 4 **个时钟。在此期间，**PC **将被弹出栈，**SREG **的位** I **被置位。如果在中断期间发生了其他中断，则** AVR **在退出中断程序后，要执行一条主程序指令之后才能再响应被挂起的中断。**

要注意 ATtiny10/11/12 **只有一个** 3 **级硬件堆栈。如果** 3 **个以上例程嵌套发生，则只有最后的** 3 **个返回地址得到保留，其他的将丢失。**

MCU控制寄存器—MCUCR

BIT	7	6	5	4	3	2	1	0
$35	-	(PUR)	SE	SM	-	-	ISC01	ISC00-
读/写	R	R(/W)	R/W	R/W	R	R	R/W	R/W
初始值	0	0	0	0	0	0	0	0

位 7、6、3、2：**保留**

PUD：上拉禁止（ATtiny12）

PUD **置位导致** ATtiny12 **所有上拉都失效。如果这一位为“**0**”，则上拉可以单独选择。**

SE：休眠使能

执行 SLEEP **指令时，** SE **必须置位才能使** MCU **进入休眠模式。为了防止无意间使** MCU **进入休眠，建议与** SLEEP **指令相连使用。**

SM：休眠模式

此位用于选择休眠模式。 SM **为“**0**”时为闲置模式；** SM **为“**1**”时为掉电模式。**

ISC01，ISC00：中断检测控制位

选择 INT0 **中断的边沿或电平，如下表所示：**

表 13 中断 0 **检测控制**

ISC01	ISC00	描述
0	0	低电平中断
0	1	电平变化中断
1	0	下降沿中断
1	1	上升沿中断

注：改变 ISC01/ISC00 **时，首先要禁止** INT0**（清除** GIMSK **的** INT0 **位），否则可能引发不必要的中断。**

INT0 **引脚的电平在检测边沿之前采样。如果边沿中断使能，则大于一个** MCU **时钟的脉冲将触发中断。如果选择了低电平触发，则此电平必须保持到当前执行的指令结束。**

ATtiny10/11 的休眠模式

进入休眠模式的条件是 SE **为“**1**”，然后执行** SLEEP **指令。** SM **用于控制休眠模式。使能的中断将唤醒** MCU**。完成中断例程后，** MCU **执行** SLEEP **以后的指令。在休眠期间，寄存器文件及** I/O **内存的内容不会丢失。如果在休眠模式下复位，则** MCU **从** RESET **向量（**$000**）处开始运行。**

闲置模式：

当 SM **为“**0**”时，** SLEEP **指令将使** MCU **进入闲置模式。在此模式下，** CPU **停止运行，而定时器/计数器、看门狗和中断系统继续工作。内外部中断都可以唤醒** MCU**。如果不需要从模拟比较器中断唤醒** MCU**，为了减少功耗，可以切断比较器的电源。方法是置位** ACSR **的** ACD**。**

掉电模式：

当 SM **为“**1**”时，** SLEEP **指令将使** MCU **进入掉电模式。在此模式下，外部晶振停振，而外部中断及看门狗（在使能的前提下）继续工作。只有外部复位、看门狗复位、外部电平中断（**INT0**）和电平变化中断可以使** MCU **脱离掉电模式。**

使用外部电平中断或电平变化中断唤醒 MCU **时要注意保持电平大于** T_{TOUT} **的时间，否则** MCU **继续保持掉电模式。**

ATtiny12 的休眠模式

进入休眠模式的条件是 SE **为“**1**”，然后执行** SLEEP **指令。** SM **用于控制休眠模式。使能的**

中断将唤醒 MCU。完成中断例程后，MCU 执行 SLEEP 以后的指令。在休眠期间，寄存器文件及 I/O 内存的内容不会丢失。如果在休眠模式下复位，则 MCU 从 RESET 向量（$000）处开始运行。

闲置模式

当 SM 为“0”时，SLEEP 指令将使 MCU 进入闲置模式。在此模式下，CPU 停止运行，而定时器/计数器、看门狗和中断系统继续工作。内外部中断都可以唤醒 MCU。如果不需要从模拟比较器中断唤醒 MCU，为了减少功耗，可以切断比较器的电源。方法是置位 ACSR 的 ACD。

掉电模式

当 SM 为“1”时，SLEEP 指令将使 MCU 进入掉电模式。在此模式下，外部晶振停振，而外部中断及看门狗（在使能的前提下）继续工作。只有外部复位、看门狗复位、外部电平中断（INT0）和电平变化中断可以使 MCU 脱离掉电模式。

使用外部电平中断或电平变化中断唤醒 MCU 时要注意保持电平一段时间，这样可以使 MCU 免受噪声干扰。唤醒周期与复位周期的时钟记数部分相同（见表 9）。如果电平保持 2 个以上看门狗周期，MCU 就可以唤醒。如果唤醒周期少于 2 个看门狗周期，那么电平也要保持那么长的时间才会唤醒 MCU。如果在复位周期结束之前要求的电平变化了，则 MCU 也会唤醒，但不会执行相应的中断例程。标称的看门狗振荡周期为 2.7μs，3.0V，25℃。

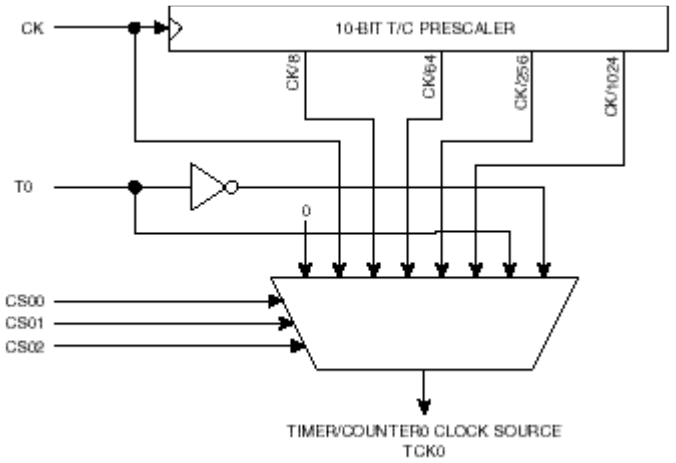
从掉电模式唤醒时有一个延迟时间，此时间用于时钟的重新起动及稳定。唤醒周期由 CKSEL 定义。

定时器/计数器 0（T/C0）

ATtiny10/11/12 内部有一个 8 位通用定时器/计数器。T/C0 从 10 位预分频定时器取得预分频的时钟。T/C0 既可用作使用片内时钟的定时器，也可用作对外部触发信号记数的计数器。

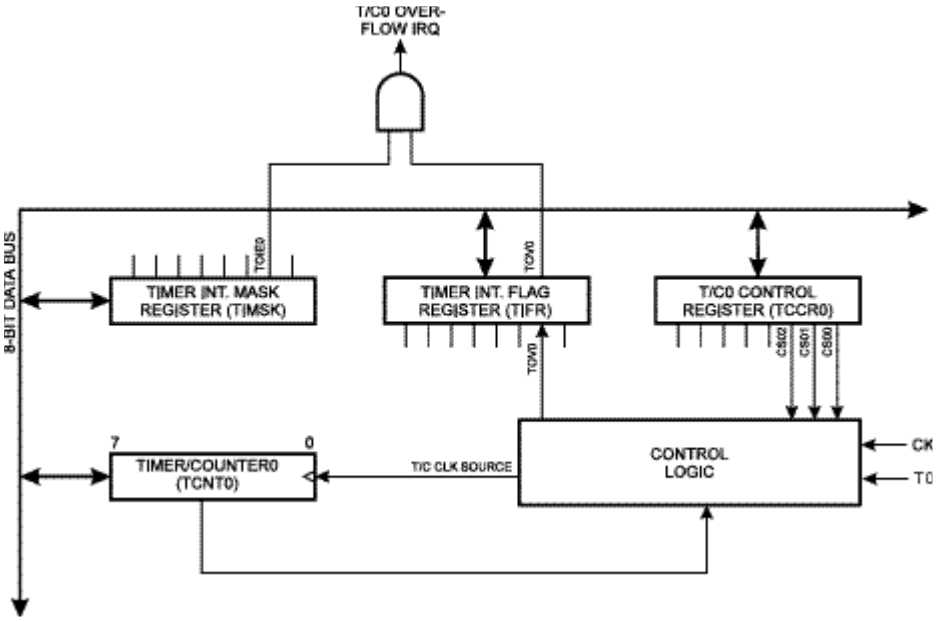
T/C0 的预分频器

图 23　T/C0 的预分频器



4 种可选的预分频时钟为：CK/8、CK/64、CK/256 和 CK/1024。CK 为振荡器时钟。对于 T/C0 还可以选择 CK、外部时钟，以及停止工作。图 24 显示了 T/C0 的工作框图。

图 24　T/C0 工作框图



T/C0 **的时钟可以选择** CK、**预分频的** CK **或外部引脚输入。另外还可以由** T/C0 **控制寄存器** TCCR0 **来停止它。溢出状态标志位在** TIFR，TCCR0 **是控制寄存器，而** TIMSK **控制** T/C0 **的中断屏蔽。**

当 T/C0 **由外部时钟信号驱动时，为了保证** CPU **对信号的正确采样，要保证外部信号的转换时间至少为一个** CPU **时钟周期。** MCU **在内部** CPU **时钟的上升沿对外部信号进行采样。**

在低预分频条件下， T/C0 **具有高分辨率和高精度的特点；而在高预分频条件下，** T/C0 **非常适用于低速功能，如计时。**

T/C0 控制寄存器—TCCR0

BIT	7	6	5	4	3	2	1	0
$33	-	-	-	-	-	**CS02**	**CS01**	**CS00**
读/写	R	R	R	R	R	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

位 7..3：**保留**

CS02、CS01、CS00：时钟选择

表 5　T/C0 预分频选择

CS02	CS01	CS00	**描述**
0	0	0	**停止**
0	0	1	CK
0	1	0	CK/8
0	1	1	CK/64
1	0	0	CK/256
1	0	1	CK/1024
1	1	0	**外部引脚** T0，**下降沿**
1	1	1	**外部引脚** T0，**上升沿**

当 T/C0 **由外部引脚** T0 **驱动时，即使** PD4（T0）**配置为输出，管脚上的信号变化照样可以使计数器发生相应的变化。这就为用户提供了一个软件控制的方法。**

T/C0—TCNT0

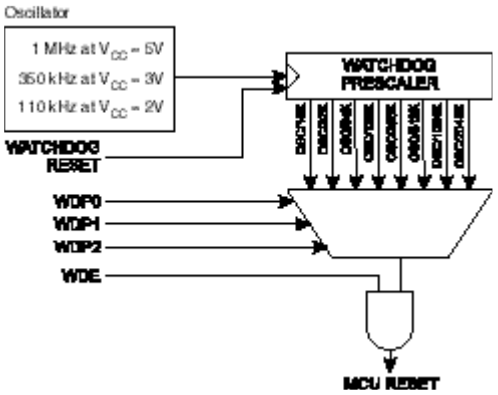
BIT	7	6	5	4	3	2	1	0
$32	**MSB**							**LSB**
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

T/C0 **是可以进行读/写访问的向上计数器。只要有时钟输入，**T/C0 **就会在写入的值基础上向上记数。**

看门狗定时器

看门狗定时器由片内独立的振荡器驱动。在 V_{CC}=5V **的条件下，典型振荡频率为** 1MHz**。通过调整定时器的预分频因数（**8 **种），可以改变看门狗复位时间间隔。看门狗复位指令是** WDT**。如果定时时间已经到，而且没有执行** WDT **指令，则看门狗将复位** MCU**。**MCU **从复位地址重新开始执行。**

图 25　看门狗定时器



看门狗定时器控制寄存器—WDTCR

BIT	7	6	5	4	3	2	1	0
$21	-	-	-	**WDTOE**	**WDE**	**WDP2**	**WDP1**	**WDP0**
读/写	R	R	R	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

位 7..5**：保留**

WDTOE：看门狗关闭使能

WDE **清零后** WDTOE **必须置位，否则看门狗不会关闭。在置位后的** 4 **个周期硬件清零。**

WDE：看门狗使能

置位时使能看门狗。为了关闭看门狗，要遵循如下步骤：

1、**同时置位** SDTOE **和** WDE**，即使** WDE **已经为“**1**”。**

2、**在随后的** 4 **个周期内，清零** WDE**。**

WDP2..0：预分频器

表 15　　看门狗定时器预分频选择

WDP2	WDP1	WDP0	振荡周期	典型溢出时间 V_{CC}=2V	典型溢出时间 V_{CC}=3V	典型溢出时间 V_{CC}=5V
0	0	0	16K	0.15s	47ms	15ms
0	0	1	32K	3.0s	94ms	30ms
0	1	0	64K	6.0s	0.19s	60ms
0	1	1	128K	1.2s	0.38s	0.12s
1	0	0	256K	2.4s	0.75s	0.24s

1	0	1	512K	4.8s	1.5s	0.49s
1	1	0	1024K	9.6s	3.0s	0.97s
1	1	1	2048K	19s	6.0s	1.9s

注：看门狗的振荡频率于电压有关。

WDT 应该在看门狗使能之前执行一次。如果看门狗在复位之前使能，则看门狗定时器有可能不是从 0 开始记数。

ATtiny12 可定标内部 RC 振荡器

ATtiny12 的内部振荡器在 5V、25℃的条件下，典型振荡频率为 1MHz。这个时钟可以用作系统时钟。通过改变寄存器 OSCCAL 的内容，可以对其进行定标。将其用作系统主时钟时仍然是看门狗的时钟。

振荡器定标寄存器—OSCCAL

BIT	7	6	5	4	3	2	1	0
$31	**CAL7**							**CAL0**
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

用于调节内部振荡器以消除生产工艺对振荡频率的影响。OSCCAL 为 0 时频率最低，为$FF 时频率最高。

ATtiny12 的 EEPROM 读/写

EEPROM 访问寄存器位于 I/O 空间。
写 EEP 的时间与电压有关，大概在 2.5~4ms 之间。自定时功能可以让用户监测何时开始写下一字节。EEPROM 准备好中断可以用来指明 EEP 什么时候可以接收新数据。
为了防止无意识的 EEPROM 写操作，需要执行一个特定的写时序。具体参看后续内容。
当执行 EEPROM 读/写操作时，CPU 会停止工作 2 个周期，然后再执行后续指令。

EEPROM 地址寄存器—EEAR

BIT	7	6	5	4	3	2	1	0
$1E	-	-	**EEAR5**	**EEAR4**	**EEAR3**	**EEAR2**	**EEAR1**	**EEAR0**
读/写	R	R	R/W	R/W	R/W	R/W	R/W	R/W
初始值	0	0	X	X	X	X	X	X

位 7..6：保留

EEAR5..EEAR0：
EEPROM 的地址是线性的。

EEPROM 数据寄存器—EEDR

BIT	7	6	5	4	3	2	1	0
$1D	**MSB**							**LSB**
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

EEDR7..EEDR0：EEPROM 数据
对于 EEPROM 写操作，EEDR 是需要写到 DDAR 单元的数据；对于读操作，EEDR 是从地

址 EEAR 读取的数据。

EEPROM控制寄存器—EECR

BIT	7	6	5	4	3	2	1	0
$1E	-	-	-	-	**EERIE**	**EEMWE**	**EEWE**	**EERE**
读/写	R	R	R	R	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

位 7..4：保留

EERIE：EEPROM准备好中断使能

清零时此中断禁止。

EEMWE：EEPROM主写使能

EEMWE 决定是否设置 EEWE 为“1”以写 EEPROM。当 EEMWE 为“1”时，置位 EEWE 将把数据写入 EEPROM 的指定地址；若 EEMWE 为“0”，则 EEWE 不起作用。EEMWE 置位后 4 个周期，硬件对其清零。

EEWE：EEPROM写使能

EEWE：EEPROM写使能

当 EEP 数据和地址设置好之后，需置位 EEWE 以便将数据写入 EEPROM。写时序如下（第 2 和第 3 步不是必须的）：

1. 等待 EEWE 为 0；
2. 将 EEP 的新地址写入 EEAR；
3. 将新数据写入 EEDR；
4. 置位 EEMWE；
5. 在置位 EEMWE 的 4 个周期内，对 EEWE 写逻辑 1。

注意：发生在步骤 4 和 5 之间的中断将导致写操作失败。如果一个操作 EEP 的中断打断了 EEP 操作，RRAR 或 EEDR 寄存器可能被修改，引起 EEP 操作失败。建议此时关闭全局中断标志 I。

经过写访问时间（V_{CC}=2.7V 时为 4ms 左右，V_{CC}=5V 时为 2.5ms 左右）之后，EEWE 硬件清零。用户可以凭此位判断写时序是否已经完成。EEWE 置位后，CPU 要停止 2 个周期。

EERE：EEPROM读使能

当 EEP 地址设置好之后，需置位 EERE 以便将数据读入 EEDR。EERE 清零表示 EEPROM 的数据已经读入 EEDR。EEPROM 数据的读取只需要一条指令，且无需等待。EERE 置位后，CPU 要停止 2 个周期。

在读之前用户应该检查 EEWE。如果写过程没有结束，而新数据及地址就写入了相应的 I/O 寄存器，则写操作会被打断，其结果将不可预测。

防止 EEPROM数据毁坏：

由于电源电压过低，CPU 和 EEPROM 有可能工作不正常，造成 EEPROM 数据的毁坏。这种情况在使用独立的 EEPROM 器件时也会遇到。

由于电压过低造成 EEPROM 数据损坏有两种可能：一是电压低至 EEPROM 写操作所需要的最低电压；二是 CPU 本身已经无法正常工作。

EEPROM 数据损坏的问题可以通过以下 3 种方法解决：

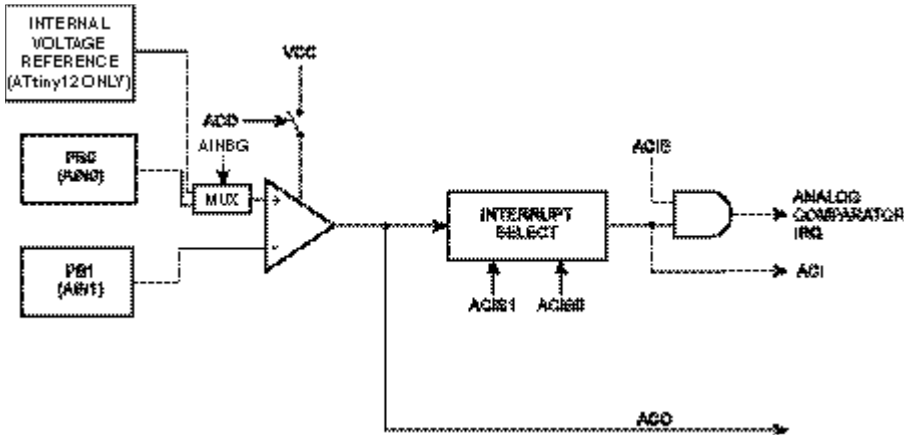
1、 当电压过低时保持/RESET 信号为低。这可以通过外加复位电路（BOD—Brown-out Detection）来完成。有些 AVR 产品本身就内含 BOD 电路。详情请看有关数据手册。
2、 当 V_{CC} 过低时使 AVR 内核处于掉电休眠状态。这可以防止 CPU 对程序解码和执行代码，有效防止对 EEPROM 的误操作。

3、将那些不需修改的常数存储于 FLASH 之中。

模拟比较器

模拟比较器比较正输入端 PB0（AIN0）和负输入端 PB1（AIN1）的值。如果 PB0（AIN0）的电压高于 PB1（AIN1）的值，比较器的输出 ACO 将置位。此输出可用来触发模拟比较器中断（上升沿、下降沿或电平变换）。其框图如图 21 所示。

图 26 模拟比较器框图



模拟比较器控制和状态寄存器—ACSR

BIT	7	6	5	4	3	2	1	0
$08	**ACD**	**(AINBG)**	**ACO**	**ACI**	**ACIE**	**-**	**ACIS1**	**ACIS0**
读/写	R/W	R/W	R	R/W	R/W	R	R/W	R/W
初始值	0	0	0	0	0	0	0	0

位 2：保留

ACD：模拟比较器禁止

当 ACD 为“1”时模拟比较器的电源将切断。可以在任何时候对其置位以关闭模拟比较器。这样可以减少器件的功耗。改变 ACD 时要注意禁止模拟比较器的中断，否则有可能引发不必要的中断。

AINBG：模拟比较器能隙基准源选择（ATtiny12）

置位后 ATtiny12 的内部 1.22±0.05V 能隙基准源取代输入管脚连接到 AIN0。

ACO：模拟比较器输出

ACO 与比较器的输出直接相连。

ACI：模拟比较器中断标志位

当比较器输出触发中断时 ACI 将置位。中断方式由 ACIS1 和 ACIS0 决定。如果 ACI 和 I 都为“1”，则 CPU 执行比较器中断例程。进入中断例程后，ACI 被硬件清零。此外，ACI 也可以通过对此位写“1”来达到清零的目的。要注意的是，如果 ACSR 的另一些位被 SBI 或 CBI 指令修改时，ACI 亦被清零。

ACIE：模拟比较器中断使能

ACIE 为“1”时，比较器中断使能。

ACIS1，ACIS0：模拟比较器中断模式选择

表 7 ACIS1/ACIS0 设置

ACIS1	ACIS0	中断模式

0	0	电平变换引发中断
0	1	保留
1	0	（ACO）下降沿中断
1	1	（ACO）上升沿中断

注：改变 ACIS1/ACIS0 时要注意禁止模拟比较器的中断，否则有可能引发不必要的中断。

ATtiny12 的内部电压基准

ATtiny12 具有标称值为 1.22V 的内部基准源。此基准源用于 BOD 及模拟比较器。

电压基准使能信号及起动时间

最小起动时间为 TBD。为了节省功耗，电压基准不是总是打开的。在如下条件下基准开启：

1、BOD 使能（BODEN 编程）

2、连接到模拟比较器（置位 AINBG）

因此，如果 BOD 功能禁止，则置位 AINBG 之后，用户要等待一段起动时间，然后再使用模拟比较器的输出。

I/O B 口

所有的 AVR I/O 端口都具有真正的读-修改-写功能。这意味着用 SBI 或 CBI 指令改变某些管脚的方向（值、禁止/使能、上拉）时不会无意地改变其他管脚的方向（值、禁止/使能、上拉）。

B 口是 6 位双向 I/O 口。

B 口有 3 个 I/O 地址：数据寄存器—PORTB（$18），数据方向寄存器—DDRB（$17）和输入引脚—PINB（$16）。PORTB 和 DDRB 可读可写，PINB 只可读。

PB5..3 具有特殊功能。如果 PB5 没有被配置为外部复位，则作为没有上拉的输入口。在 ATtiny12 中还可以作为开漏输出。PB4 和 PB3 如果没有用作时钟功能，则可作为 I/O 口使用。所有的 I/O 口都有单独可选的上拉。

PB0 到 PB4 输出缓冲器可以吸收 20mA 的电流，能够直接驱动 LED。ATtiny12 的 PB5 则可以吸收 12mA。当 PB0..PB4 被拉低时，如果上拉电阻已经激活，则引脚会输出电流。

B 口的第二功能如下表所示：

表 17　B 口第二功能

管　脚	第二功能	器　件
PB0	AIN0（模拟比较器正输入端）	ATtiny10/11/12
	MOSI（程序下载时的数据输入）	ATtiny12
PB1	INT0	ATtiny10/11/12
	AIN1（模拟比较器负输入端）	ATtiny10/11/12
	MISO（程序下载时的数据输出）	ATtiny12
PB2	T0	ATtiny10/11/12
	SCK（程序下载时的串行时钟）	ATtiny12
PB3	XTAL1	ATtiny10/11/12
PB4	XTAL2	ATtiny10/11/12
PB5	/RESET	ATtiny10/11/12

当使用 B 口的第二功能时，DDRB 和 PORTB 要设置成对应的值。

B 口数据寄存器—PORTB

BIT	7	6	5	4	3	2	1	0
$18	-	-	-	**PORTB4**				**PORTB0**
读/写	R	R	R	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

B 口数据方向寄存器—DDRB

BIT	7	6	5	4	3	2	1	0
$17	-	-	(DDB5)	**DDB4**				**DDB0**
读/写	R	R	R(/W)	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

B 口输入引脚地址—PINB

BIT	7	6	5	4	3	2	1	0
$16	-	-	**PINB5**					**PINB0**
读/写	R	R	R	R	R	R	R	R
初始值	0	0	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z

PINB **不是一个寄存器，这个地址用来访问** B **口的物理值。读取** PORTB**时，读到的是** B **口锁存的数据；而读取** PINB **时，读到的是施加于引脚上的逻辑数值。**

B 口用作通用数字 I/O

作为通用数字 I/O **时，**B **口的** 5 **个管脚具有相同的功能。**

PBn**，通用** I/O **引脚：**DDRB **中的** DDBn **选择引脚的方向。如果** DDBn **为“**1**”，则** PBn **为输出脚；如果** DDBn **为“**0**”，则** PBn **为输入脚。在复位期间，**B **口为三态口。**

表 18　**B **口的配置

DDBn	PORTBn	I/O	**上拉**	**注释**
0	0	**输入**	N	**三态（高阻）**
0	1	**输入**	Y	**外部拉低时会输出电流**
1	0	**输出**	N	**推挽输出** 0
1	1	**输出**	N	**推挽输出** 1

n：4，3..0，**引脚号**

注意：在 ATtiny10/11 **中，**PB5 **只是输入；而对于** ATtiny12 **则可以是输入或是开漏输出。由于这个管脚用于** 12V **编程，因此没有** ESD **保护二极管，在使用过程当中要注意保证此引脚电压不超过** V_{CC} + 1 V**。否则容易使** MCU **复位或进入编程模式。**

B 口的第二功能

- /RESET—PB5
  RSTDISBL **未编程时作为外部复位；若** RSTDISBL **已编程，则引脚可作为输入。对于** ATtiny12**，还可以作为开漏输出。**
- XTAL2—PB4
- XTAL1—PB3
- T0/SCK—PB2
  计数器外部时钟输入。对于 ATtiny12 **还是串行编程的时钟输入。**
- INT0/AIN1/MISO—PB1

可以用作外部中断 0 输入，模拟比较器负极信号输入。对于 ATtiny12 还是程序下载时的输出数据。

- AIN0/MOSI—PB0
  模拟比较器正极信号输入。对于 ATtiny12 还是下载程序时的数据。

程序编程

程序和数据锁定位

ATtiny10/11/12 具有两个锁定位，如表 19 所示。锁定位只能通过片擦除命令擦除。

表 19　锁定保护模式

程序锁定位			保护类型
模式	LB1	LB2	
1	1	1	无锁定功能
2	0	1	禁止编程 [1)]
3	0	0	禁止校验

注意：1、在高压串行编程模式下，熔断位编程也被禁止。要先编程熔断位，然后编程锁定位。

ATtiny10/11 的熔断位

ATtiny10/11 有 5 个熔断位：FSTRT，RSTDISBL 和 CKSEL2..0。

- FSTRT：见表 7。缺省值为“1”。
- RSTDISBL 编程后 PB5 的外部复位功能被禁止[注意)]。缺省值为“1”。
- CKSEL2..0：见表 3。缺省值为“100”，内部 RC 振荡器。

芯片擦除命令不影响熔断位。

注意：RSTDISBL 编程后，在对其下载程序时，要在上电复位的同时在 PB5 引脚加+12V。

ATtiny12 的熔断位

ATtiny12 有 8 个熔断位：BODLEVEL，BODEN，SPIEN，RSTDISBL 和 CKSEL3..0，且可以在高低压串行编程模式下编程。

- BODLEVEL：选择 BOD 检测电压。缺省已编程（“0”）。
- BODEN：编程后使能 BOD 功能。
- SPIEN：编程后使能低压串行编程。缺省已编程（“0”）。
- RSTDISBL 编程后 PB5 的外部复位功能被禁止[注意)]。缺省值为“1”。
- CKSEL3..0：见表 3 及表 9。缺省值为“0010”，内部 RC 振荡器，长起动时间。

芯片擦除命令不影响熔断位。

注：RSTDISBL 编程后，在对其下载程序时，要在上电复位的同时在 PB5 引脚加+12V。

厂标

所有的 Atmel 微处理器都有 3 字节的厂标，用以识别器件。此代码在串行或并行模式下都可以访问。

对于 ATtiny10，其位置为：
1、$000：$1E（表明是 Atmel 生产的）
2、$001：$90（1K 字节的 FLASH）
3、$002：$03（当$01 地址为$90 时，器件为 ATtiny10）
对于 ATtiny11，其位置为：
1、$000：$1E（表明是 Atmel 生产的）
2、$001：$90（1K 字节的 FLASH）
3、$002：$04（当$01 地址为$90 时，器件为 ATtiny11）
对于 ATtiny12[注意]，其位置为：
1、$000：$1E（表明是 Atmel 生产的）
2、$001：$90（1K 字节的 FLASH）
3、$002：$05（当$01 地址为$90 时，器件为 ATtiny12）
注：在锁定保护模式 3 有效时，厂标不能以串行模式读出。其返回值将为$00，$01 和$02。

ATtiny12 的定标字节

ATtiny12 具有一个字节用来定标内部 RC 振荡器（1MHz）。此字节位于厂标地址空间$000 的高字节。如果要利用这个字节，应该将此信息读出并写入 FLASH。程序代码再将其写入 OSCCAL。

编程 FLASH 和 EEPROM

ATtiny10/11
ATtiny10/11 具有 1K 字节的片内可编程 FLASH，在出厂时已经被擦除为“1”。
器件支持+12V 高压串行编程。+12V 只用来使能高压编程，不会有明显的电流流过。
FLASH 以字节的形式写入。

ATtiny12
ATtiny12 具有 1K 字节的片内可编程 FLASH 和 64 字节的 EEPROM，在出厂时已经被擦除为“1”。
器件支持+12V 高压串行编程和低压串行编程。+12V 只用来使能高压编程，不会有明显的电流流过。
FLASH 和 EEPROM 以字节的形式写入。

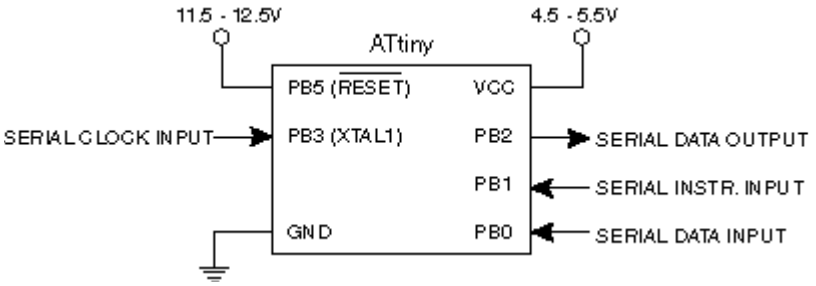
ATtiny10/11/12
编程电压见表 20。

表 20　编程电源电压

型号	低压串行编程	高压串行编程
ATtiny10/11L	不支持	4.5V – 5.5V
ATtiny10/11	不支持	4.5V – 5.5V
ATtiny12V	2.2V – 5.5V	4.5V – 5.5V
ATtiny12L	2.7V – 5.5V	4.5V – 5.5V
ATtiny12	4.0V – 5.5V	4.5V – 5.5V

高压串行编程

图 27　高压串行编程



编程算法：

以下步骤使器件进入高压串行编程模式：

1、**上电序列：在** V_{CC} **和** GND **之间加上** 4.5 – 5.5V **电压。拉低** PB5 **和** PB0，**并保持至少** 100ns。**改变** PB3 **的电平至少** 4 **次，脉宽至少为** 100ns。**将** PB3 **拉低，等待至少** 100ns。**给** PB5（/RESET）**加上** 12V **的电压，并在** PB0 **变化之前保持至少** 100ns。**在继续后续指令前等待至少** 8μs。

2、FLASH **的编程以字节为单位。首先加地址，然后是数据。**PB2（RDY/BSY）**变高说明写入过程结束。**

3、EEPROM（ATtiny12）**的编程以字节为单位。首先加地址，然后是数据。**PB2（RDY/BSY）**变高说明写入过程结束。**

4、**任何地址的内容可通过读指令由** PB2 **读出。**

5、**下电序列：拉低** PB3；**置位** PB5（“1”）；**关电源。**

数据在时钟的上升沿输入/输出。

图 28　高压串行编程波形

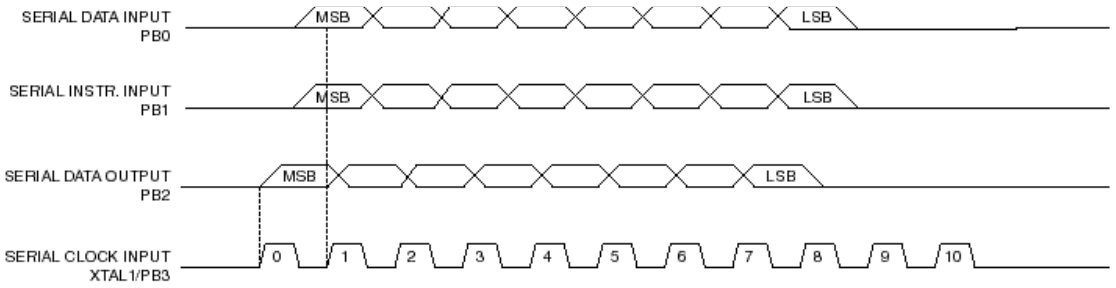


表 21　高压串行编程指令集

指令		**指令格式**				**注释**
		指令 1	**指令 2**	**指令 3**	**指令 4**	
片擦除（ATtiny 10/11）	PB0 PB1 PB2	0_1000_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	**执行完指令** 3 **后要等待** t_{WLWH_CE}
片擦除（ATtiny 12）	PB0 PB1 PB2	0_1000_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	**执行完指令** 4 **后等待** PB2 **变高**
写 FLASH **高低地址**	PB0 PB1 PB2	0_0001_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_000**a**_00 0_0001_1100_00 x_xxxx_xxxx_xx	0_**bbbb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx		**指令** 2 **每** 256 **字节执行一**

						次，指令 3 则每个地址执行一次。
写 FLASH 低字节	PB0 PB1 PB2	0_**iiii_iiii**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 0_0000_0000_00		指令 3 后要等待 PB2 变高。
写 FLASH 高字节	PB0 PB1 PB2	0_**iiii_iiii**_00 0_0011_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 0_0000_0000_00		指令 3 后要等待 PB2 变高。
读 FLASH 高低地址	PB0 PB1 PB2	0_0000_0010_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_000**a**_00 0_0001_1100_00 x_xxxx_xxxx_xx	0_**bbbb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx		每个地址都需执行指令 2 和 3
读 FLASH 低字节	PB0 PB1 PB2	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **o_oooo_oooo**_xx			每个地址都需执行指令 1 和 2
读 FLASH 高字节	PB0 PB1P PB2	0_0000_0000_00 0_0111_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 **o_oooo_oooo**_xx			每个地址都需执行指令 1 和 2
写 EEP 低地址（ATtiny 12）	PB0 PB1 PB2	0_0001_0001_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_00**bb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx			每个地址都需执行指令 2
写 EEP 字节（ATtiny 12）	PB0 PB1 PB2	0_**iiii_iiii**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 0_0000_0000_00		指令 3 后要等待 PB2 变高。
读 EEP 低地址（ATtiny 12）	PB0 PB1 PB2	0_0000_0011_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_00**bb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx			每个地址都需执行指令 2
读 EEP 字节（ATtiny 12）	PB0 PB1 PB2	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **o_oooo_oooo**_xx			每个地址都需执行指令 2
写熔丝位（ATtiny 10/11）	PB0 PB1 PB2	0_0100_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_000**7_6543**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx	执行完指令 3 后要等待 t_{WLWH_PFB}，7..3 为“0”代表编程
写熔丝位（ATtiny 12）	PB0 PB1 PB2	0_0100_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_**CBA9_8543**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx	执行完指令 3 后要等待 t_{WLWH_PFB}，C..3 为“0”代表编程
写锁定位	PB0 PB1 PB2	0_0010_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0**21**0_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx	指令 4 后要等待 PB2 变高。2，1= “0” 表示编程
读熔丝位（ATtiny 10/11）	PB0 PB1 PB2	0_0000_0100_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xx**76_543**x_xx		若 7..3 为“0”表示已编程
读熔丝位（ATtiny 12）	PB0 PB1 PB2	0_0000_0100_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **C_BA98_543**x_xx		若 C..3 为“0”表示已编程
读锁定位	PB0 PB1 PB2	0_0000_0100_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_**21**xx_xx		若 1，2 为“0”表示已编程

读厂标	PB0 PB1 PB2	0_0000_1000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_00**bb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **o_oooo_ooo**x_xx	**重复指令** 2-4 **以读取其他字节**
读定标字节（Attiny12）	PB0 PB1 PB2	0_0000_1000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 **o_oooo_ooo**x_xx	

注： a = 地址的高比特位

b = 地址的低比特位

i = 输入的数据

o = 输出的数据

x = **不用管**

1 = 锁定位 1

2 = 锁定位 2

3 = CKSEL0

4 = CKSEL1

5 = CKSEL2

9 . 6 = RSTISBL

7 = FSTRT

8 = CKSEL3

A = SPIEN

B = BODEN

C = BODLEVEL

高压串行编程特性

图 29 高压串行编程时序

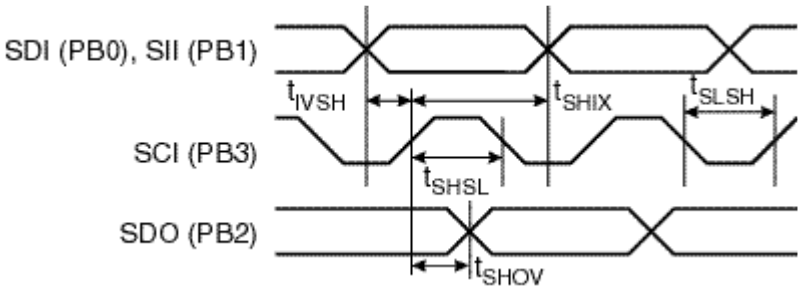


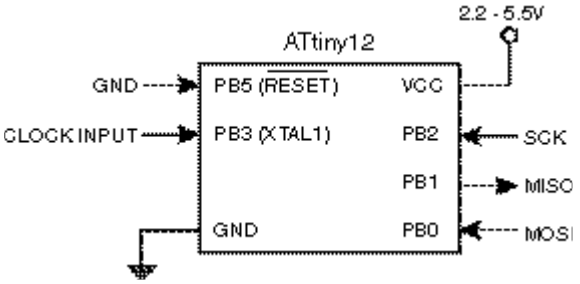
表 22 高压串行编程特性 T_A = 25℃±10%，V_{CC} = 5V±10%

符号	参数	最小值	典型值	最大值	单位
t_{SHSL}	SCI（PB3）**脉冲（高）宽度**	100			ns
t_{SLSH}	SCI（PB3）**脉冲（低）宽度**	100			ns
t_{IVSH}	SDI（PB0），SII（PB1）**有效到** SCI **高**	50			ns
t_{SHIX}	SCI **高后** SDI（PB0），SII（PB1）**保持**	50			ns
t_{SHOV}	SCI（PB3）**高到** SDO（PB2）**有效**	10	16	32	ns
t_{WLWH_CE}	**片擦除时指令** 3 **等待时间**	5	10	15	ms
t_{WLWH_PFB}	**写熔丝位时指令** 3 **等待时间**	1.0	1.5	1.8	ms

低压串行下载（仅对 ATtiny12 有效）

当/RESET 拉到地时，FLASH 和 EEPROM 可以利用 SPI 总线进行串行下载。串行接口包括 SCK，MOSI 和 MISO。/RESET 拉低后，在进行编程/擦除之前首先要执行编程使能指令。

图 30　串行编程和校验



对于 EEPROM，由于其本身有自动擦除功能和自定时功能，因此更新时无需擦除。擦除指令将使 FLASH 和 EEPROM 的内容全部变为$FF。

FLASH 和 EEPROM 的地址是分离的。FLASH 的范围是$0000~$01FF，EEPROM 的范围是$000~$03F。

时钟可以从 XTAL1 引脚输入，或者使用连接到 XTAL1 和 XTAL2 的晶振。SCK 脉冲的最小高低电平时间为：

低：> 2 个 XTAL1 时钟

高：> 2 个 XTAL1 时钟

串行编程算法

进行串行编程时，数据在 SCK 的上升沿输入 ATtiny12，在 SCK 的下降沿输出。

编程算法如下：

1、上电过程：
   在/RESET 和 SCK 拉低的同时在 V_{CC} 和 GND 之间加上电源电压。
2、至少等待 20ms。 然后在 MOSI（PB0）串行输入编程使能指令。
3、如果通讯失步则串行编程将失败。如果同步，则在写编程使能命令第 3 个字节的时候器件会响应第二个字节（$53）。不论响应正确与否，4 字节指令必须发完。如果响应不是$53，则要产生一个 SCK 正脉冲并发送编程使能指令。如果发送编程使能指令 32 次都没有正确的响应就说明外部器件不存在或器件已坏。
4、如果此时执行了擦除指令，则须等待 t_{WD_ERASE}，然后在/RESET 上施加正脉冲，回到第二步。
5、FLASH 和 EEPROM 是一个字节一个字节编程的。发送完写指令后要等待 t_{WD_PROG} 的时间。对于擦除过的器件，数据$FF 就用不着再写了。
6、任意一个内存地址都可以用读指令在 MISO（PB1）读出。
7、编程结束后，可以把/RESET 拉高，进入正常工作模式。
8、下电过程（如果需要的话）：
   将 XTAL1 拉低（如果没有用外部晶振，或者用的是内部 RC 振荡器）。
   把/RESET 拉高。
   关掉电源。

数据检测

写 EEPROM 时，如果内部的自擦除过程没有结束，读正在写的地址会得到$FF。当写过程结束后，读取的数据则为写入的数据。用这种方法可以确定何时可以写入新数据。但是对于$FF，就不可以用这种方法了。此时应当在编程新数据之前至少等待 t_{WD_PROG} 的时间。如果芯片在编程 EEPROM 之前已经进行过芯片擦除，则数据$FF 就可以不用再编程了。

图 31　串行编程波形

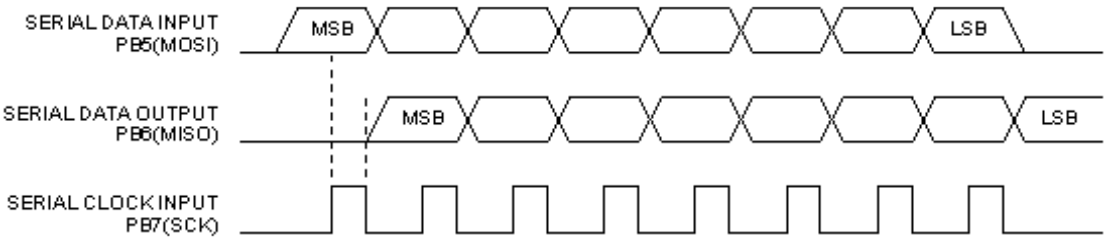


表 23　低压串行编程指令

指令	指令格式				操作
	字节 1	字节 2	字节 3	字节 4	
编程使能	1010 1100	0101 0011	Xxxx xxxx	xxxx xxxx	/RESET 为低时使能串行编程
芯片擦除	1010 1100	100x xxxx	Xxxx xxxx	xxxx xxxx	擦除 FLASH 和 EE
读 FLASH	0010 **H**000	0000 000**a**	**bbbb bbbb**	**oooo oooo**	从字地址 a:b 读取 H（高或低）字节 o
写 FLASH	0100 **H**000	0000 000**a**	**bbbb bbbb**	**iiii iiii**	写 H（高或低）字节 i 到字地址 a:b
读 EEPROM	1010 0000	0000 0000	xx**bb bbbb**	**oooo oooo**	从地址 b 读取数据 o
写 EEPROM	1100 0000	0000 0000	xx**bb bbbb**	**iiii iiii**	写数据 i 到地址 b
读锁定位	0101 1000	xxxx xxxx	xxxx xxxx	xxxx x**21**x	“0”代表编程
写锁定位	1010 1100	1111 1***21***1	xxxx xxxx	xxxx xxxx	写锁定位
读厂标	0011 0000	xxxx xxxx	xxxx xx**bb**	**oooo oooo**	从地址 b 读厂标 o(1)
读定标字节	0011 1000	xxxx xxxx	0000 0000	**oooo oooo**	
写熔丝位	1010 1100	101x xxxx	xxxx xxxx	**A987 6543**	0 代表要编程
读熔丝位	0101 0000	xxxx xxxx	xxxx xxxx	**A987 6543**	0 代表已编程

注：**a** = 地址高 Bit

b = 地址低 Bit　　**H** = 0：低地址；1：高地址

o = 输出数据　　**i** = 输入数据

x = 任意

1 = Lock Bit1　　***2*** = Lock Bit2

1 = 锁定位 1

2 = 锁定位 2

3 = CKSEL0

4 = CKSEL1

5 = CKSEL2

6 = CKSEL3

7 = RSTDISBL

8 = SPIEN

9 = BODEN

A = BODLEVEL

1、**厂标不能在锁定模式** 3 **下读出。**

串行编程电特性

图 32 串行编程时序

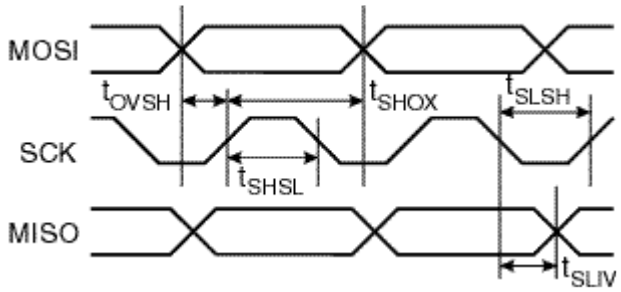


表 24 串行编程电特性，T_A = -40℃**到** 85℃，V_{CC} = 2.2V – 5.5V

符号	参数	最小值	典型值	最大值	单位
1/ t_{CLCL}	**振荡频率**（V_{CC} = 2.2V – 2.7V）	0		1	MHz
t_{CLCL}	**振荡周期**（V_{CC} = 2.2V – 2.7V）	1000			ns
1/ t_{CLCL}	**振荡频率**（V_{CC} = 2.7V – 4.0V）	0		4	MHz
t_{CLCL}	**振荡周期**（V_{CC} = 2.7V – 4.0V）	250			ns
1/ t_{CLCL}	**振荡频率**（V_{CC} = 4.0V – 5.5V）	0		8	MHz
t_{CLCL}	**振荡周期**（V_{CC} = 4.0V – 5.5V）	125			ns
t_{SHSL}	SCK **高**	2 t_{CLCL}			ns
t_{SLSH}	SCK **低**	2t_{CLCL}			ns
t_{OVSH}	MOSI Setup to SCK High	t_{CLCL}			ns
t_{SHOX}	MOSI Hold after SCK High	2t_{CLCL}			ns
t_{SLIV}	SCK Low to MISO Valid	10	16	32	ns

表 25 擦除指令之后的最小等待时间

符号	2.2V	2.7V	4.0V	5.0V
t_{WD_ERASE}	7ms	6ms	5ms	4ms

表 26 写指令之后的最小延迟时间

符号	2.2V	2.7V	4.0V	5.0V
t_{WD_PROG}	7ms	6ms	5ms	4ms

直流特性

T_A = -40℃**到** 85℃，V_{CC} = 2.7V – 5.5V（ATtiny10/11），V_{CC} = 1.8V – 5.5V（ATtiny12）

符号	参数	条件	最小值	典型值	最大值	单位
V_{IL}	**输入低电压**	**除了** XTAL	-0.5		0.3 V_{CC} (1)	V
V_{IL1}	**输入低电压**	XTAL	-0.5		0.1V_{CC} (1)	V
V_{IH}	**输入高电压**	**除了** XTAL **和**/RESET	0.6 V_{CC} (2)		V_{CC} +0.5	V
V_{IH1}	**输入高电压**	XTAL	0.7 V_{CC} (2)		V_{CC} +0.5	V
V_{IH2}	**输入高电压**	/RESET	0.85 V_{CC} (2)		V_{CC} +0.5	V
V_{OL}	**输出低电压** [3] B **口**	I_{OL} =20mA，V_{CC} = 5V I_{OL} =10mA，V_{CC} = 3V			0.6 0.5	V
V_{OL}	**输出低电**	I_{OL} =12mA，V_{CC} = 5V			0.6	V

	PB5(ATtiny12)	I_{OL} =6mA，V_{CC} = 3V			0.5	
V_{OH}	**输出高电压** [4] **B 口**	I_{OH} = -3mA，V_{CC} = 5V I_{OH} = -1.5mA，V_{CC} = 3V	4.3 2.3			V
I_{IL}	**输入泄露电流 I/O 脚**	V_{CC} = 5.5V，pin low			8.0	μA
I_{IH}	**输入泄露电流 I/O 脚**	V_{CC} = 5.5V，pin low			8.0	μA
$R_{I/O}$	**I/O 口的上拉电阻**		35		122	kΩ
I_{CC}	**电流**	**工作**，4MHz，V_{CC} =3V			3.0	mA
		空闲，4MHz，V_{CC} =3V		1.0	1.2	mA
		掉电，4MHz [5]，V_{CC} =3V，**看门狗使能**		9.0	15	μA
		掉电，4MHz [5]，V_{CC} =3V，**看门狗关闭**		< 1	2	μA
Vacio	**模拟比较器输入偏置电压**	V_{CC} = 5V			40	mV
Iaclk	**模拟比较器输入偏置电流**	V_{CC} = 5V V_{IN} = V_{CC}/2	-50		50	nA
Tacpd	**模拟比较器传输延迟**	V_{CC} = 2.7V V_{CC} = 4.0V		750 500		ns

注：1、**“最大值”代表保证可以“0”读取时的最高电压**

2、**“最小值”代表保证可以“1”读取时的最低电压**

3、 **虽然每个** I/O **口在常态下可以吸收超过测试条件（** V_{CC} = 5V **为** 20mA，V_{CC} = 3V **为** 10mA **）的电流，但需遵守如下条件：**

1] **所有** I/O **口的** I_{OL} **之和不能超过** 100mA

如果 I_{OL} **超过测试条件，则** V_{OL} **也将超过相关的指标。超过测试标准使用没有保证。**

4、 **虽然每个** I/O **口在常态下可以吸收超过测试条件（** V_{CC} = 5V **为** 3mA，V_{CC} = 3V **为** 1.5mA **）的电流，但需遵守如下条件：**

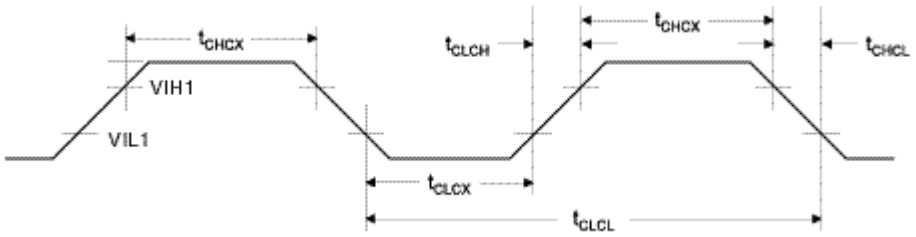
1]**所有** I/O **口的** I_{OH} **之和不能超过** 100mA

如果 I_{OH} **超过测试条件，则** V_{OH} **也将超过相关的指标。超过测试标准使用没有保证。**

5、 **掉电时的最小** V_{CC} **为** 1.5V（ATtiny12：**仅当** BOD **禁止时）**

外部时钟驱动波形

图 33　外部时钟



外部时钟

符号	参数	V_{CC}=1.8V~2.7V		V_{CC}=2.7V~4.0V		V_{CC}=4.0V~5.5V		单位
		Min	Max	Min	Max	Min	Max	
1/ t_{CLCL}	振荡频率	0	1	0	4	0	8	MHz
t_{CLCL}	时钟周期	1000		250		125		ns
t_{CHCX}	高电平时间	400		100		50		ns
t_{CLCX}	低电平时间	400		100		50		ns
t_{CLCH}	上升时间		1.6		1.6		0.5	μs
t_{CHCL}	下降时间		1.6		1.6		0.5	μs

表 27　外部 RC 振荡器典型频率

R[kΩ]	C[pF]	f
100	70	100kHz
31.5	20	1.0MHz
6.5	20	4.0MHz

注意：R 的范围应该在 3 – 100kΩ 之间，而 C 不能小于 20pF。上表给出的 C 包含了引脚电容。不同的封装具有不同的电容值。

ATtiny11 典型特性

后续图表表明了器件的典型特点。这些数据并没有进行 100%的测试。功耗测量的条件为所有 I/O 引脚配置为输入（有上拉）。正弦波发生器作为时钟。
掉电模式的功耗与时钟的选择无关。
器件功耗受以下因素影响：工作电压，工作频率，I/O 口的加载，I/O 口变换频率，执行的代码以及工作温度。主要因素是工作电压和频率。
容性负载的功耗可由公式 C_L*V_{CC}*f 进行计算。式中，C_L 为负载电容，V_{CC} =工作电压，f=I/O 引脚的平均开关频率。
器件曲线标度高于测试的极限。使用时一定要按照订购器件的指标来使用。
工作于掉电模式时，看门狗使能及禁止两条曲线之差即表示了看门狗的功耗。

图 34　工作电流与频率的关系

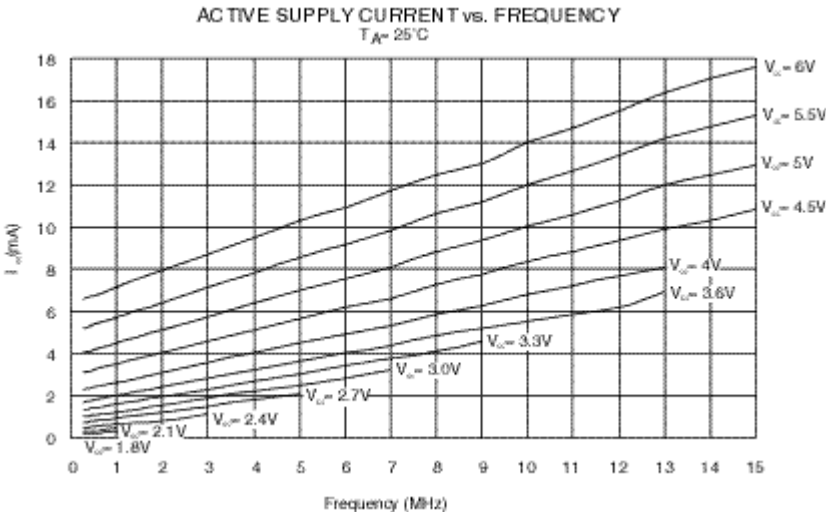


图 35　工作电流与电压的关系

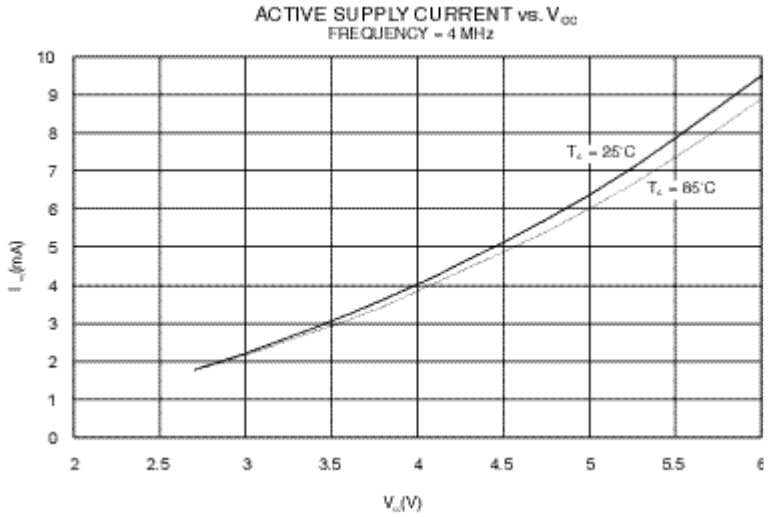


图 36　工作电流与频率的关系，闲置模式

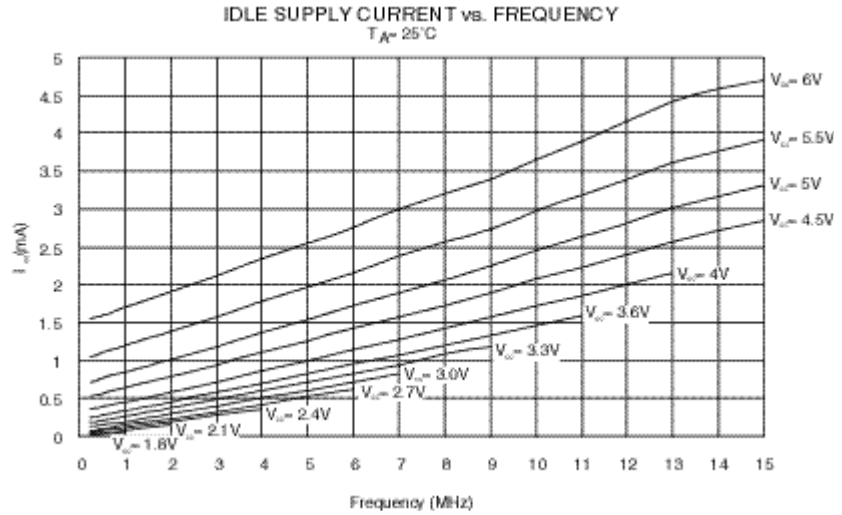


图 37　工作电流与电压的关系，闲置模式

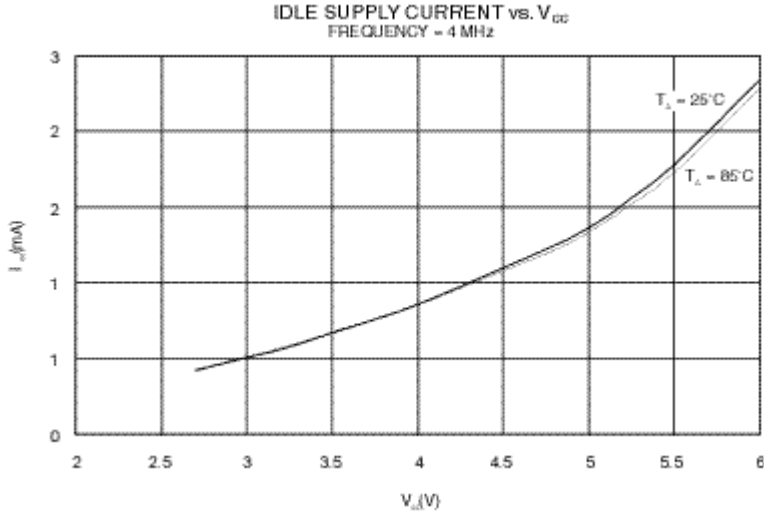


图 38　工作电流与电压的关系，掉电模式，看门狗禁止

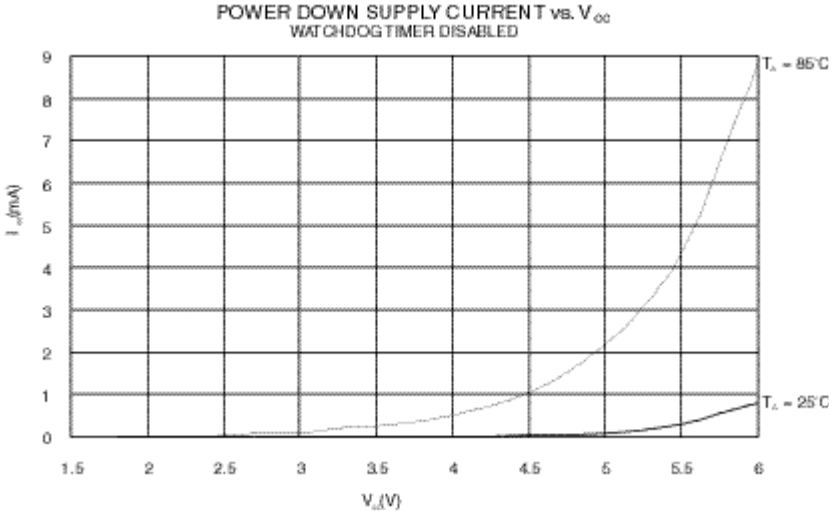


图 39　工作电流与电压的关系，掉电模式，看门狗使能

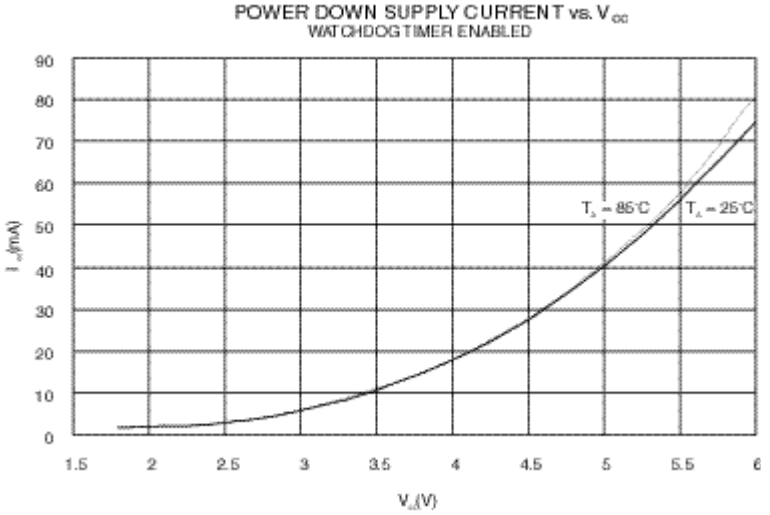


图 40　模拟比较器电流与电压的关系

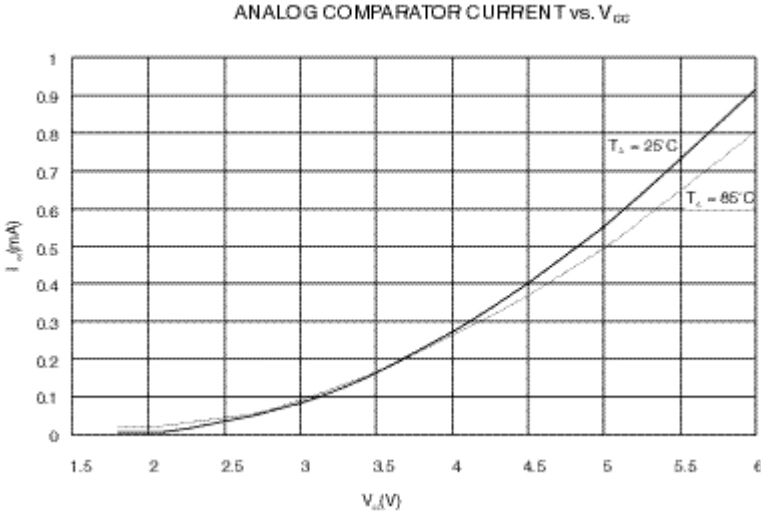


图 41　模拟比较器偏置电压与共模电压的关系，V_{CC}=5V

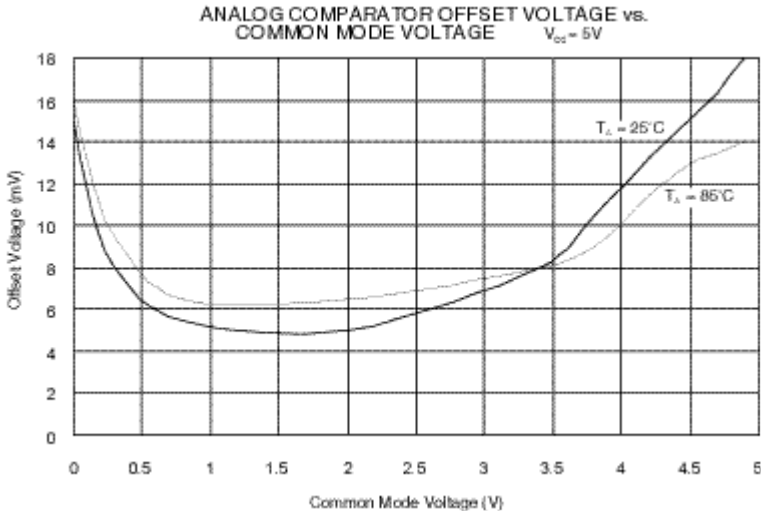


图 42　模拟比较器偏置电压与共模电压的关系

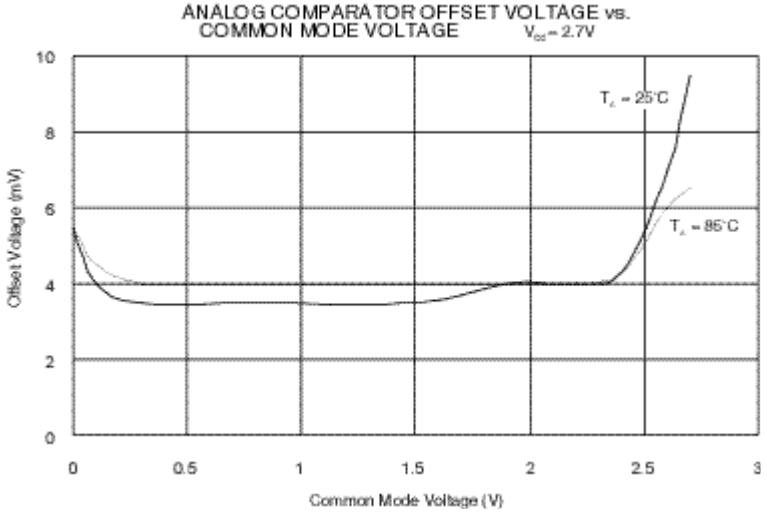


图 43　模拟比较器输入泄漏电流

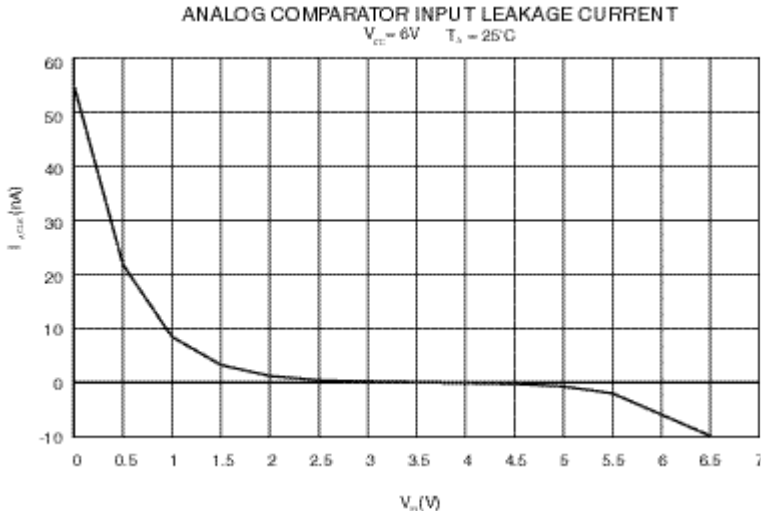


图 44　看门狗振荡频率与电压的关系

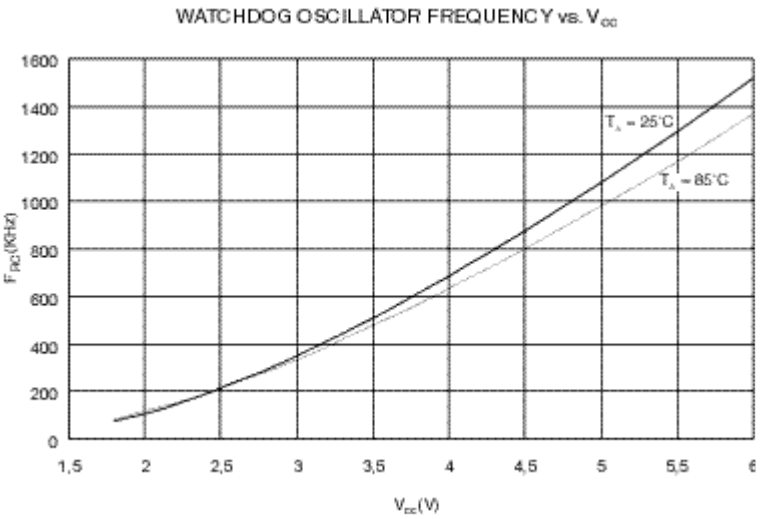


图 45　上拉电阻电流与输入电压的关系

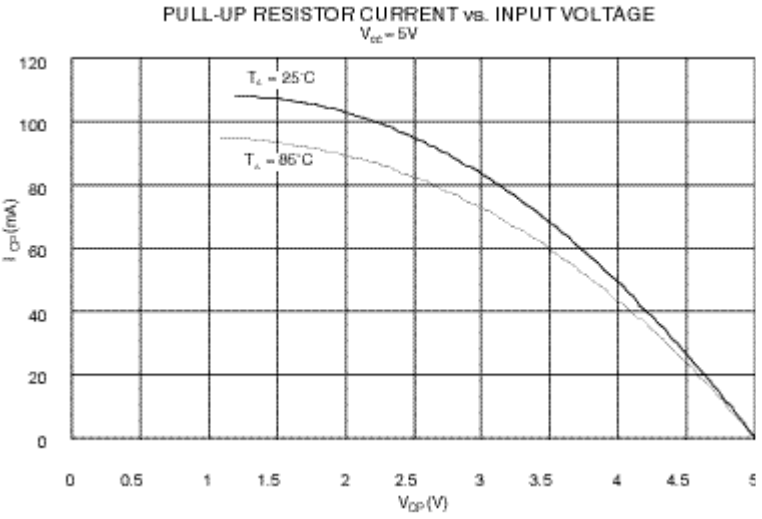


图 46　上拉电阻电流与输入电压的关系

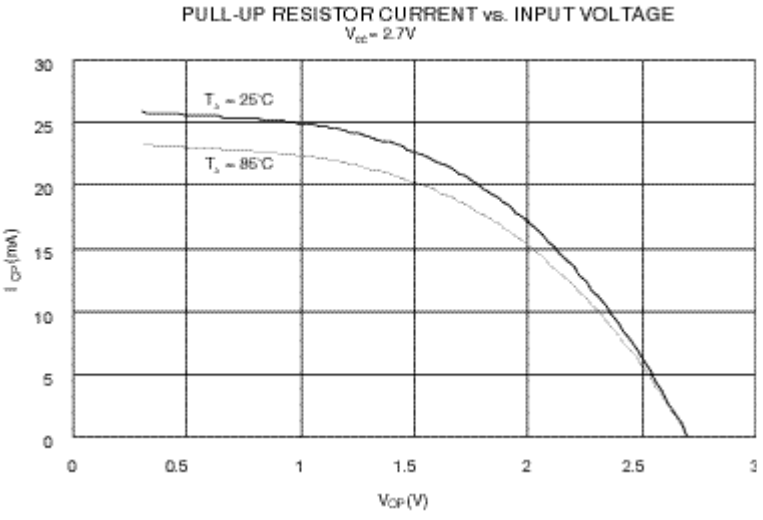


图 47　I/O 引脚吸入电流与输出电压的关系

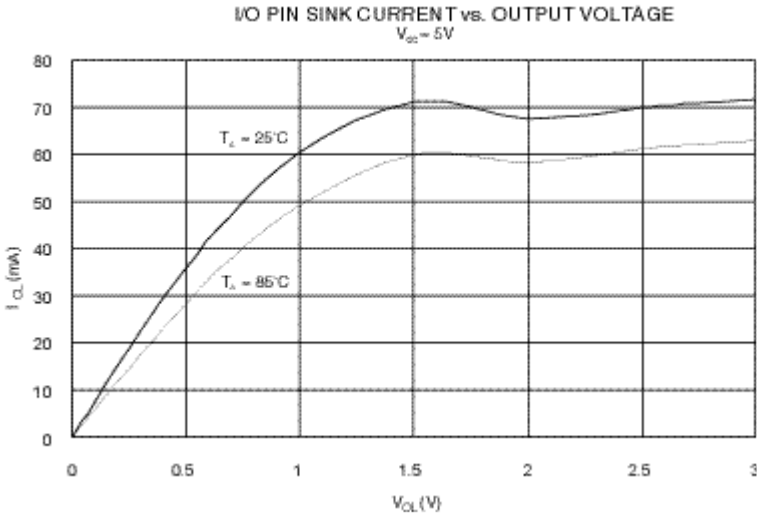


图 48　I/O 引脚吸入电流与输出电压的关系

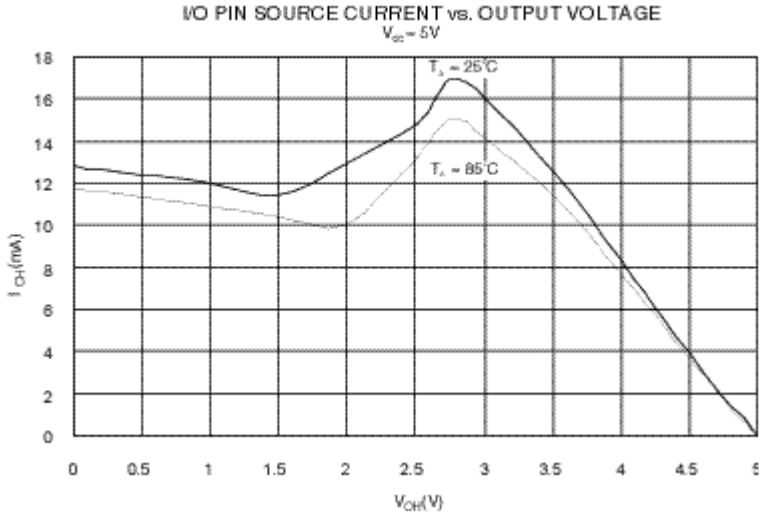


图 49　I/O 引脚吸入电流与输出电压的关系

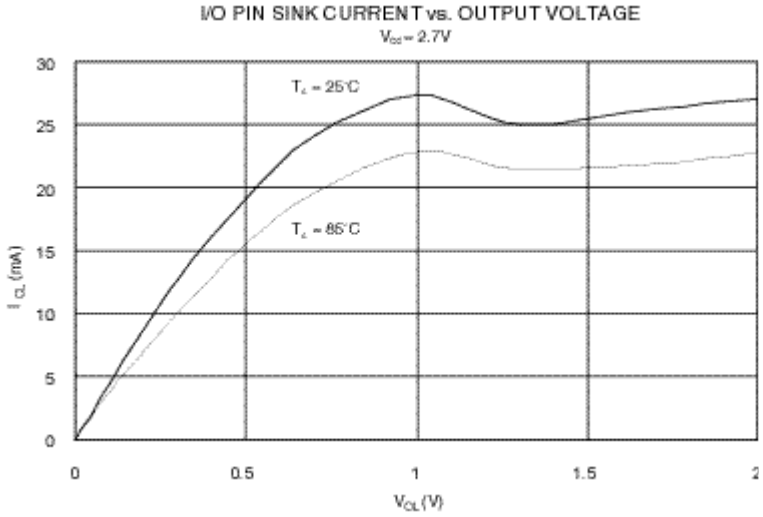


图 50　I/O 引脚输出电流与输出电压的关系

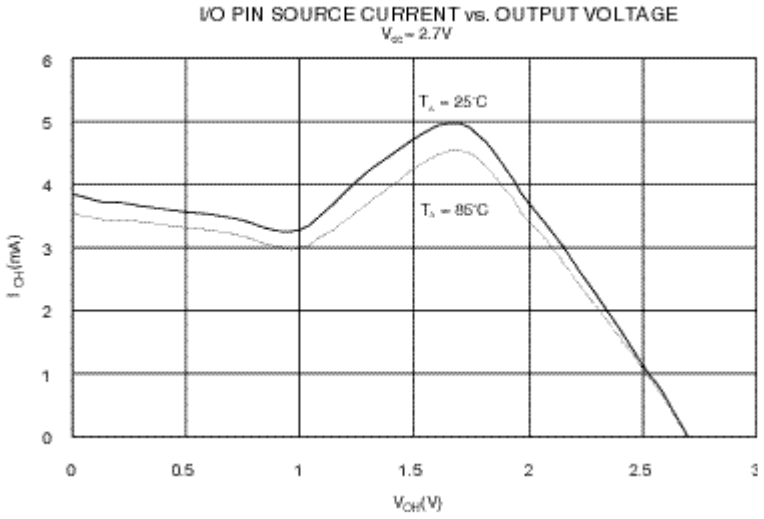


图 51　I/O 引脚输出门限电压与电压的关系

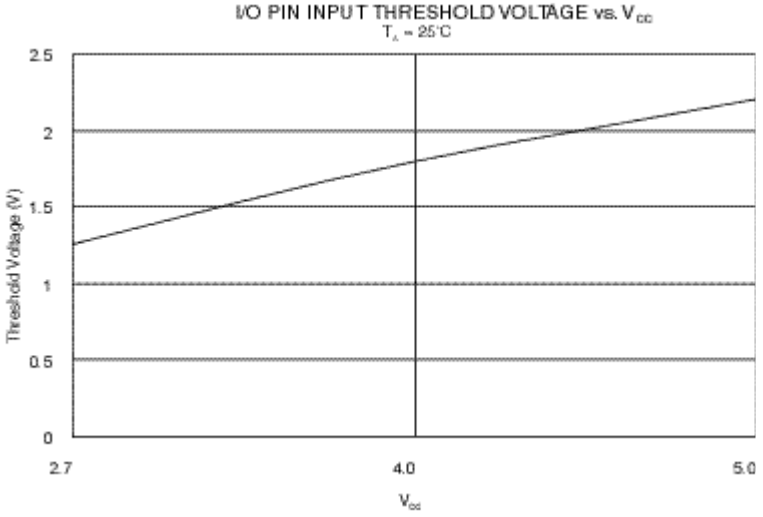
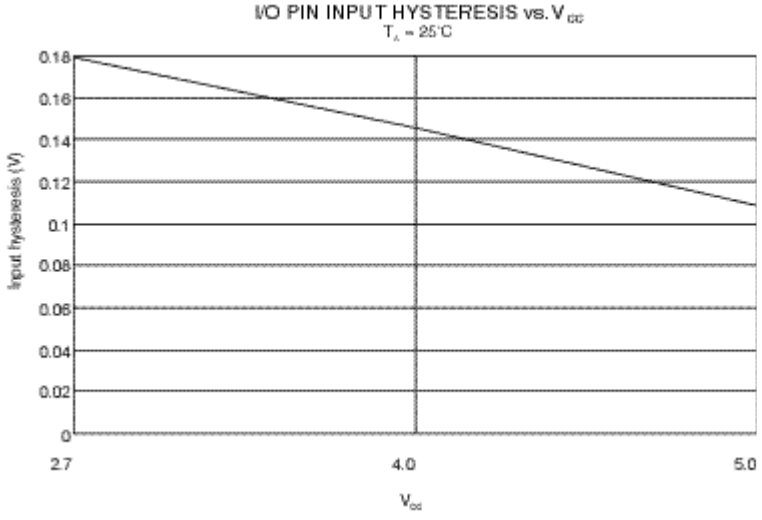


图 52　I/O 引脚输入容限与电压的关系



ATtiny12 典型特性

后续图表表明了器件的典型特点。这些数据并没有进行 100%的测试。功耗测量的条件为所有 I/O 引脚配置为输入（有上拉）。正弦波发生器作为时钟。
掉电模式的功耗与时钟的选择无关。
器件功耗受以下因素影响：工作电压，工作频率，I/O 口的加载，I/O 口变换频率，执行的代码以及工作温度。主要因素是工作电压和频率。
容性负载的功耗可由公式 $C_L*V_{CC}*f$ 进行计算。式中，C_L 为负载电容，V_{CC} =工作电压，f=I/O 引脚的平均开关频率。
器件曲线标度高于测试的极限。使用时一定要按照订购器件的指标来使用。
工作于掉电模式时，看门狗使能及禁止两条曲线之差即表示了看门狗的功耗。

图 53　片内 RC 振荡器频率与电压的关系

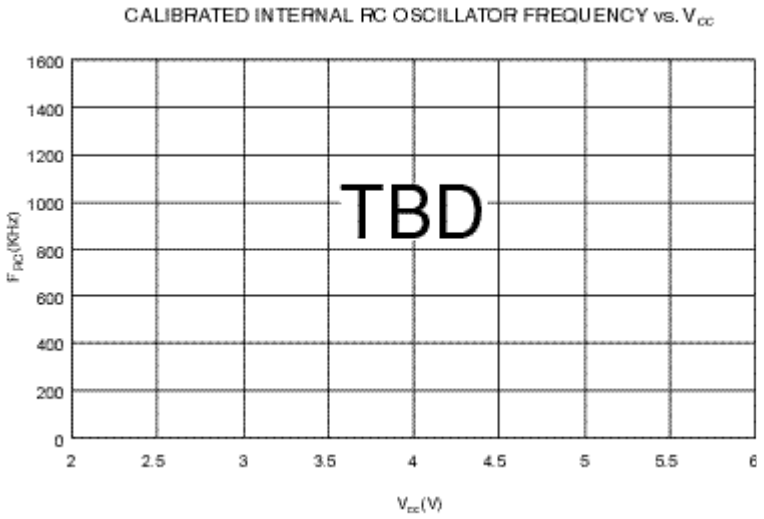


图 54　模拟比较器偏置电压与共模电压的关系

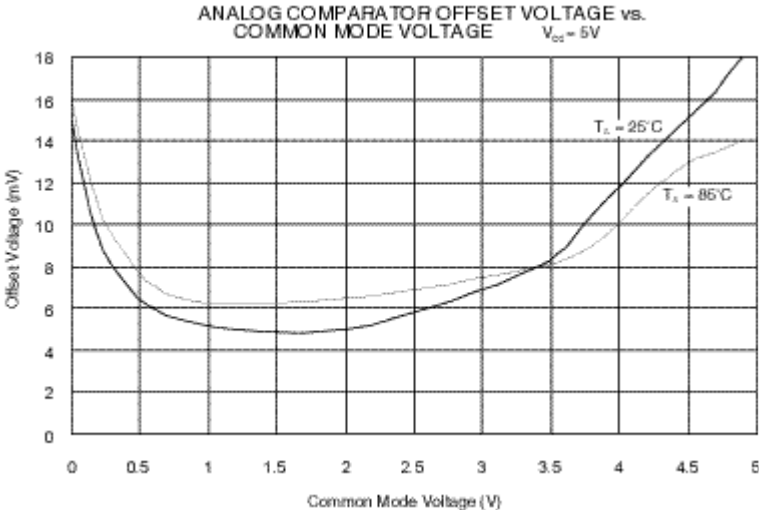


图 55　模拟比较器偏置电压与共模电压的关系

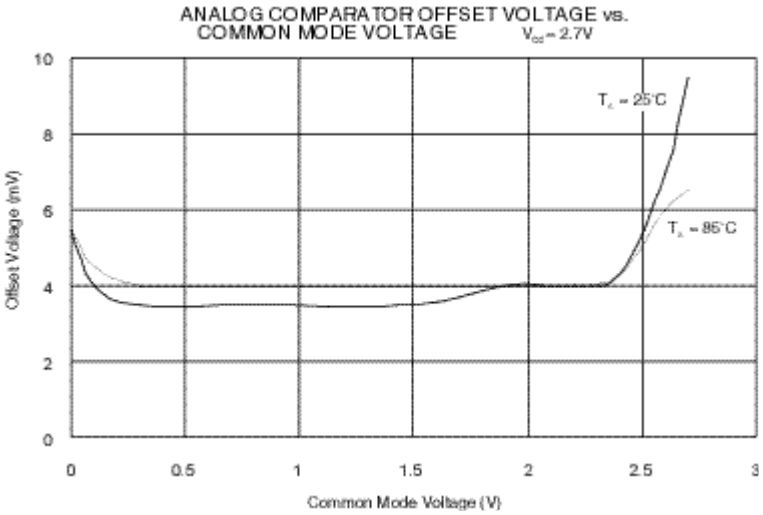


图 56　模拟比较器输入泄漏电流

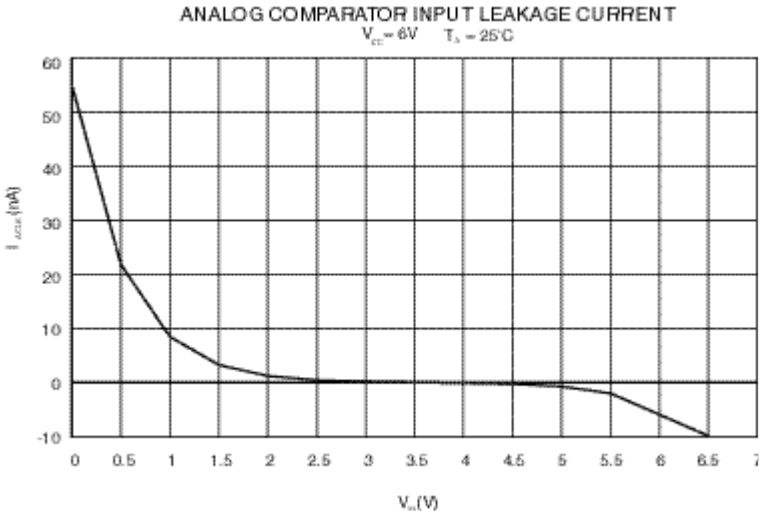


图 57　看门狗振荡频率与电压的关系

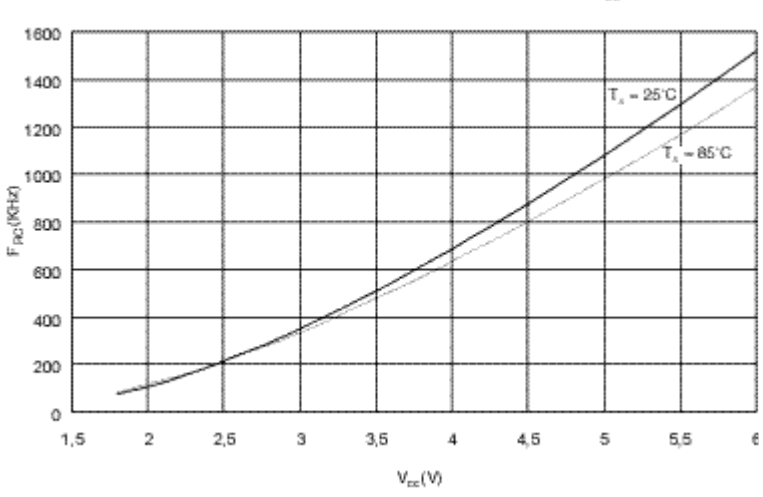


图 58　上拉电阻电流与输入电压的关系

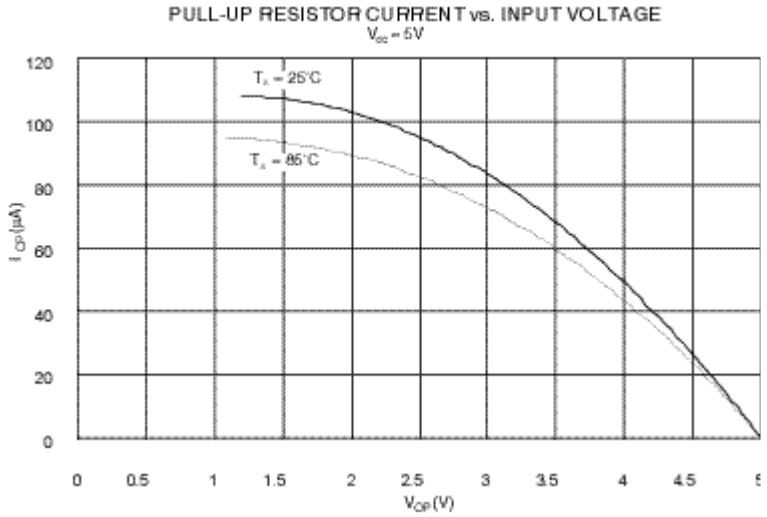


图 59　上拉电阻电流与输入电压的关系

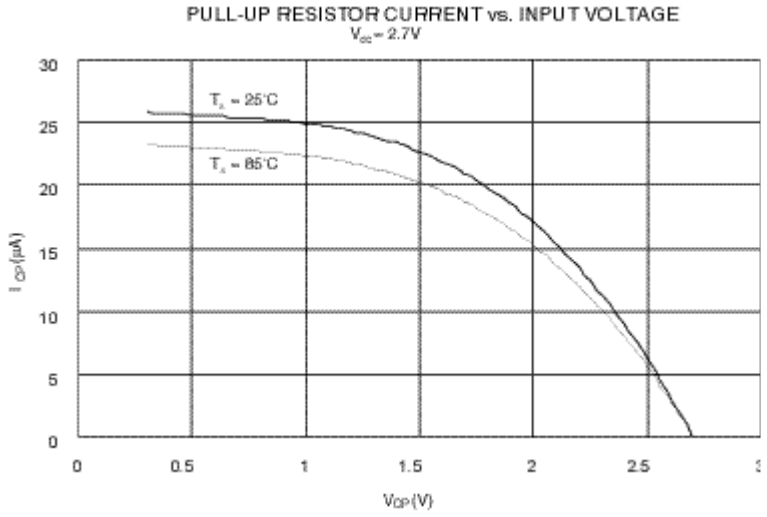


图 60　I/O 引脚吸入电流与输出电压的关系

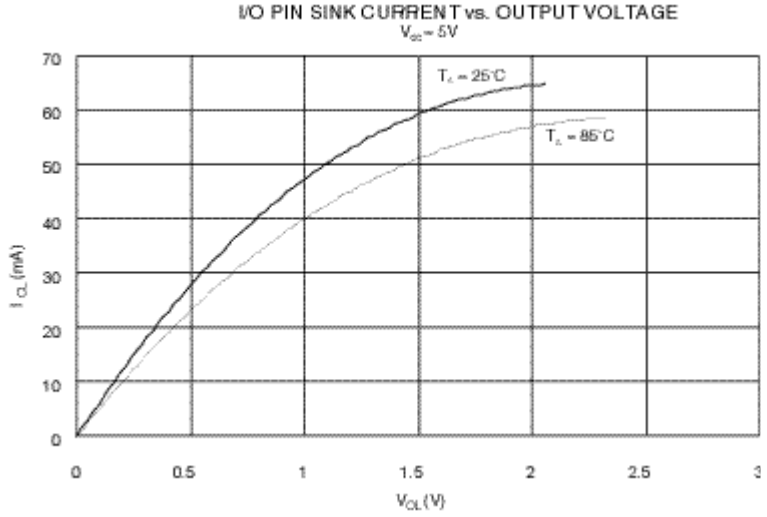


图 61　I/O 引脚输出电流与输出电压的关系

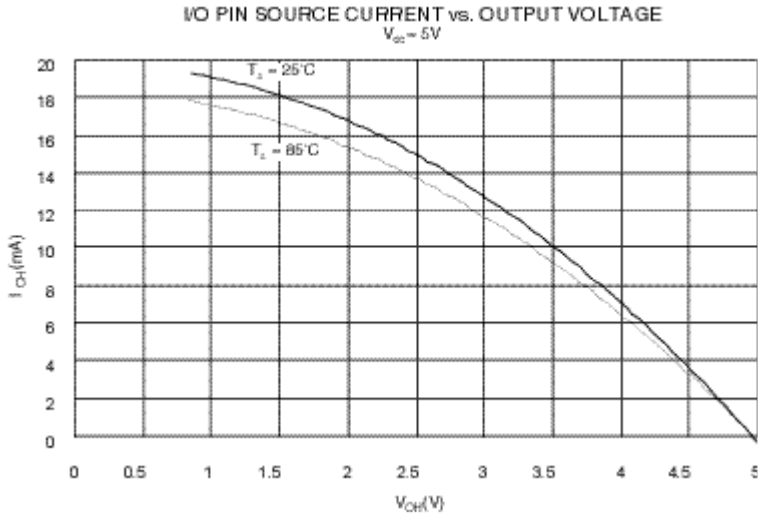


图 62　I/O 引脚吸入电流与输出电压的关系

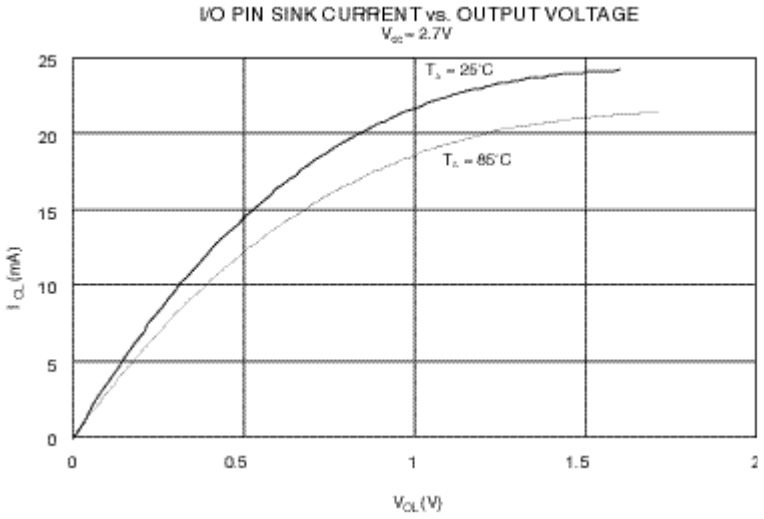


图 63　I/O 引脚输出电流与输出电压的关系

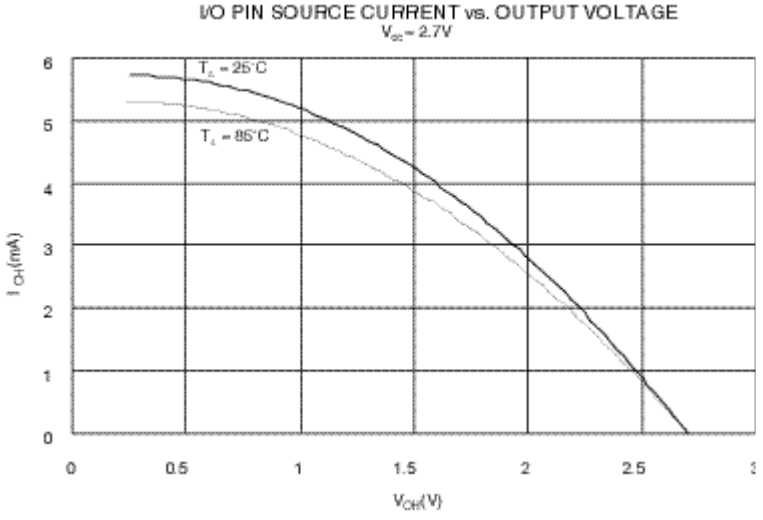


图 64　I/O 引脚输出门限电压与电压的关系

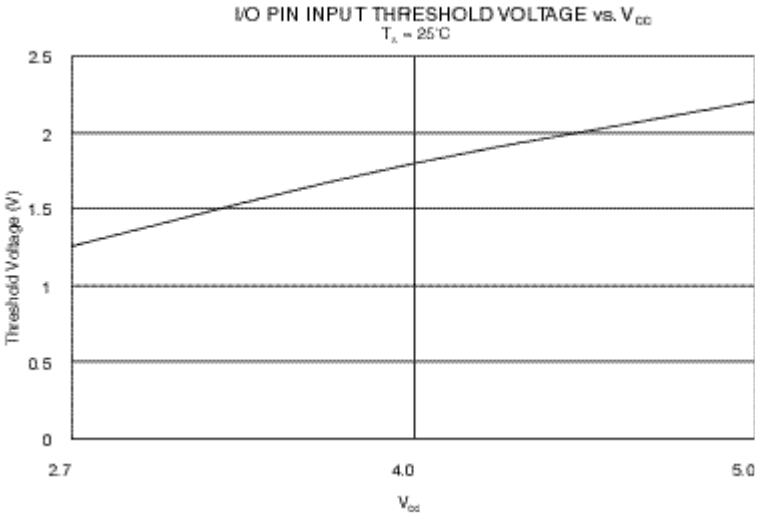
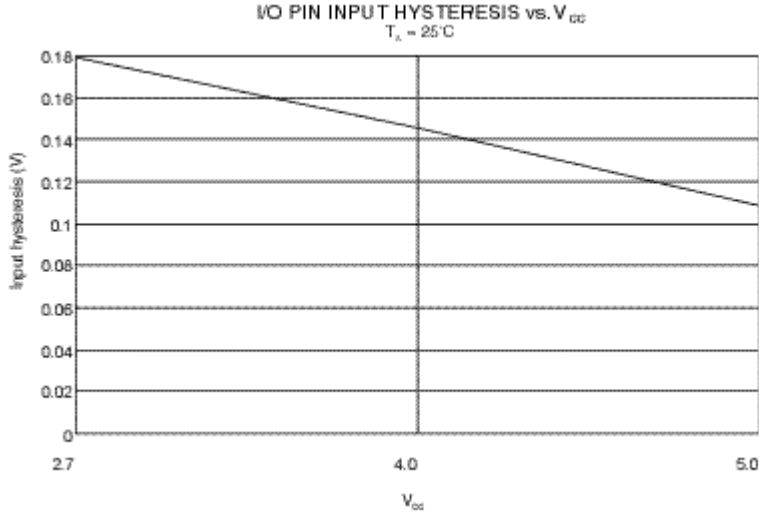


图 65　I/O 引脚输入容限与电压的关系



指　　令

Rd——**目的寄存器**　　Rr——**源寄存器**

K——**常数**　　k——**常数表示的地址**

b——**寄存器文件或** I/O **寄存器的位**　　s——**状态寄存器中的位**

Z、Y、X——R31:R26

A——I/O **地址**　　q——**偏移量**（6 **位**）

助记符	**操作数**	**描述**	**操作**	**受影响的标志**	**时钟数**
算术及逻辑操作					
ADD	Rd，Rr	**不带进位位加**	Rd ← Rd + Rr	Z,C,N,V,S,H	1
ADC	Rd，Rr	**带进位位加**	Rd ← Rd + Rr + C	Z,C,N,V,S,H	1
SUB	Rd，Rr	**不带进位位减**	Rd ← Rd - Rr	Z,C,N,V,S,H	1
SUBI	Rd，**K**	**减立即数**	Rd ← Rd - **K**	Z,C,N,V,S,H	1
SBC	Rd，Rr	**带进位位减**	Rd ← Rd – Rr - C	Z,C,N,V,S,H	1

SBC I	Rd，**K**	**带进位减立即数**	Rd ← Rd – **K** - C	Z,C,N,V,S,H	1
AND	Rd，Rr	**逻辑与**	Rd ← Rd ▪ Rr	Z, N,V,S	1
ANDI	Rd，**K**	**与立即数**	Rd ← Rd ▪ **K**	Z, N,V,S	1
OR	Rd，Rr	**逻辑或**	Rd ← Rd ∨ Rr	Z, N,V,S	1
ORI	Rd，**K**	**或立即数**	Rd ← Rd ∨ **K**	Z, N,V,S	1
EOR	Rd，Rr	**异或**	Rd ← Rd O± Rr	Z, N,V,S	1
COM	Rd	1	Rd ← $FF - Rd	Z,C,N,V,S	1
NEG	Rd	2 **的补码**	Rd ← $00 - Rd	Z,C,N,V,S,H	1
SBR	Rd，**K**	**寄存器的位置位**	Rd ← Rd ∨ **K**	Z, N,V,S	1
CBR	Rd，**K**	**寄存器的位清零**	Rd ← Rd ▪ （$FF - **K**）	Z, N,V,S	1
INC	Rd	**加一**	Rd ← Rd + 1	Z, N,V,S	1
DEC	Rd	**减一**	Rd ← Rd - 1	Z, N,V,S	1
TST	Rd	**零和负测试**	Rd ← Rd ▪ Rd	Z, N,V,S	1
CLR	Rd	**清除寄存器**	Rd ← Rd O± Rd	Z, N,V,S	1
SER	Rd	**置位寄存器**	Rd ← $FF	-	1
跳转指令					
RJMP	k	**相对跳转**	PC ← PC + k + 1	-	2
RCALL	k	**相对调用**	PC ← PC + k + 1	-	3/4
RET		**调用返回**	PC ← **堆栈**	-	4/5
RETI		**中断返回**	PC ← **堆栈**	I	4/5
CPSE	Rd，Rr	**比较，相等则跳**	**若**（Rd=Rr）PC←PC+2 **或** 3	-	1/2/3
CP	Rd，Rr	**比较**	Rd - Rr	Z,C,N,V,S,H	1
CPC	Rd，Rr	**带进位位比较**	Rd – Rr - C	Z,C,N,V,S,H	1
CPI	Rd，**K**	**与立即数比较**	Rd – **K**	Z,C,N,V,S,H	1
SBRC	Rd，b	**寄存器位清零即跳**	**若**（Rd（b） = 0）PC←PC+2 **或** 3	-	1/2/3
SBRS	Rd，b	**寄存器位置位即跳**	**若**（I/O（A，b） = 1）PC←PC+2 **或** 3	-	1/2/3
SBIC	A，b	I/O **寄存器位清零即跳**	**若**（I/O（A，b） = 0）PC←PC+2 **或** 3	-	1/2/3
SBIS	A，b	I/O **寄存器位置位即跳**	**若**（Rd（b） = 1）PC←PC+2 **或** 3	-	1/2/3
BRBS	s，k	**状态标志置位跳**	**若**（SREG（s）=1）PC←PC+k+1	-	1/2
BRBC	s，k	**状态标志清零跳**	**若**（SREG（s）=0）PC←PC+k+1	-	1/2
BREQ	k	**相等即跳**	**若**（Z=1）PC←PC+k+1	-	1/2
BRNE	k	**不等即跳**	**若**（Z=0）PC←PC+k+1	-	1/2
BRCS	k	**进位即跳**	**若**（C=1）PC←PC+k+1	-	1/2
BRCC	k	**没有进位即跳**	**若**（C=0）PC←PC+k+1	-	1/2
BRSH	k	**大于等于即跳（无符号）**	**若**（C=0）PC←PC+k+1	-	1/2
BRLO	k	**小于即跳（无符号）**	**若**（C=1）PC←PC+k+1	-	1/2
BRMI	k	**负即跳**	**若**（N=1）PC←PC+k+1	-	1/2
BRPL	k	**正即跳**	**若**（N=0）PC←PC+k+1	-	1/2
BRGE	k	**大于等于即跳（有符号）**	**若** （ NO+V=1 ）PC←PC+k+1	-	1/2

BRLT	k	小于即跳（有符号）	若 （ NO+V=0 ） PC←PC+k+1	-	1/2
BRHS	k	H 位置位即跳	若（H=1）PC←PC+k+1	-	1/2
BRHC	k	H 位清零即跳	若（H=0）PC←PC+k+1	-	1/2
BRTS	k	T 置位即跳	若（T=1）PC←PC+k+1	-	1/2
BRTC	k	T 清零即跳	若（T=0）PC←PC+k+1	-	1/2
BRVS	k	V 置位即跳	若（V=1）PC←PC+k+1	-	1/2
BRVC	k	V 清零即跳	若（V=0）PC←PC+k+1	-	1/2
BRIE	k	中断使能即跳	若（I=1）PC←PC+k+1	-	1/2
BRID	k	中断禁止即跳	若（I=0）PC←PC+k+1	-	1/2
数据传输指令					
LD	Rd，Z	间接取数	Rd ← （Z）	-	2
ST	Z，Rr	间接存数	（Z）← Rr	-	2
MOV	Rd，Rr	拷贝寄存器	Rd ← Rr	-	1
LDI	Rd，**K**	取立即数	Rd ← **K**	-	1
IN	Rd，A	从 I/O 取数	Rd ← I/O（A）	-	1
OUT	A，Rr	存数于 I/O	I/O（A）← Rr	-	1
LPM		在程序区取数	R0 ← （Z）	-	3
位及位测试指令					
SBI	A，b	设置 I/O 寄存器的位	I/O（A，b）← 1	-	2
CBI	A，b	清除 I/O 寄存器的位	I/O（A，b）← 0	-	2
LSL	Rd	逻辑左移	Rd(n + 1) ← Rd(n)，Rd(n)←0，C←Rd(7)	Z,C,N,V,H	1
LSR	Rd	逻辑右移	Rd(n) ← Rd(n + 1)，Rd(7)←0，C←Rd(0)	Z,C,N,V	1
ROL	Rd	带进位位左移	Rd(0)←C，Rd(n + 1) ← Rd(n)，C←Rd(7)	Z,C,N,V,H	1
ROR	Rd	带进位位右移	Rd(7)←C，Rd(n) ← Rd(n + 1)，C←Rd(0)	Z,C,N,V	1
ASR	Rd	算术右移	Rd(n) ← Rd(n + 1)，n=6..0	Z,C,N,V	1
SWAP	Rd	高低位交换	Rd(3..0) ←→ Rd(7..4)	-	1
BSET	s	设置标志	SREG（s）← 1	SREG（s）	1
BCLR	s	清除标志	SREG（s）← 0	SREG（s）	1
BST	Rr，b	Rr 的 b 位到 T	T ← Rr（b）	T	1
BLD	Rd，b	T 到 Rr 的 b 位	Rr（b）← T	-	1
SEC		置位 C	C ← 1	C	1
CLC		清零 C	C ← 0	C	1
STN		置位 N	N ← 1	N	1
CLN		清零 N	N ← 0	N	1
SEZ		置位 Z	Z ← 1	Z	1
CLZ		清零 Z	Z ← 0	Z	1
SEI		置位 I	I ← 1	I	1
CLI		清零 I	I ← 0	I	1
SES		置位 S	S ← 1	S	1
CLS		清零 S	S ← 0	S	1
SEV		置位 V	V ← 1	V	1

CLV		清零 V	V ← 0	V	1
SET		置位 T	T ← 1	T	1
CLT		清零 T	T ← 0	T	1
SHE		置位 H	H ← 1	H	1
CLH		清零 H	H ← 0	H	1
NOP		空操作		-	1
SLEEP		休眠指令		-	3
WDR		看门狗复位		-	1

定货信息

速度（MHz）	电源（V）	定货号	封装	工作范围
2	2.7 – 5.5	ATtiny11L – 2PC ATtiny11L – 2SC	8P3 8S2	商用 （0℃ - 70℃）
		ATtiny11L – 2PI ATtiny11L – 2SI	8P3 8S2	工业 （-40℃ - 85℃）
6	4.0 – 5.5	ATtiny11 – 6PC ATtiny11 – 6SC	8P3 8S2	商用 （0℃ - 70℃）
		ATtiny11 – 6PI ATtiny11 – 6SI	8P3 8S2	工业 （-40℃ - 85℃）
1	1.8 – 5.5	ATtiny11L – 2PC ATtiny11L – 2SC	8P3 8S2	商用 （0℃ - 70℃）
		ATtiny12V – 2PI ATtiny12V – 2SI	8P3 8S2	工业 （-40℃ - 85℃）
4	2.7 – 5.5	ATtiny12L – 4PC ATtiny12L – 4SC	8P3 8S2	商用 （0℃ - 70℃）
		ATtiny12L – 4PI ATtiny12L – 4SI	8P3 8S2	工业 （-40℃ - 85℃）
8	4.0 – 5.5	ATtiny12 – 8PC ATtiny12 – 8SC	8P3 8S2	商用 （0℃ - 70℃）
		ATtiny12 – 8PI ATtiny12 – 8SI	8P3 8S2	工业 （-40℃ - 85℃）

封装类型	
8P3	8 脚，0.300’’，塑料双列直插（PDIP）
8S2	8 脚，0.200’’，塑料 Gull-Wing 小尺寸（EIAJ SOIC）

开发过程及所需工具

汇编软件

AVR ASM（**免费软件，可从**www.atmel.com**或**WWW.SL.COM.CN**下载**）

仿真器

AVR ICE（STDPOD 或 ADCPOD），或 ICE 200。调试软件为 AVR STUDIO（免费软件，可从www.atmel.com或WWW.SL.COM.CN下载）

下载器

START KIT 200 或第三方厂商（如：广州天河双龙电子有限公司 SL-AVR,LT-48.）