ATtiny15/L

特点

1. 高性能，低功耗，8 位 AVR 结构
2. 先进的 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
   —全静态工作
3. 数据和非易失性程序内存
   —1K 字节的在线可编程 FLASH
   擦除次数：1000 次
   —64 字节在线可编程 EEPROM
   寿命：100000 次
   —程序加密位
4. 外围（Peripheral）特点
   —两个可预分频（Prescale）的 8 位定时器/计数器
   一个高速（100kHz）PWM 输出
   —4 通道 10 位 ADC
   一个具有可选 20 倍增益的差分通道
   —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
   —通过 SPI 口的 ISP
   —内外部中断源
   —低功耗空闲和掉电模式
   —低功耗，减噪和掉电模式
   —增强的上电复位电路
   —可编程的 BOD 电路
   —内部 1.6MHz 可调谐振荡器
   —内部 T/C1 25.6MHz 时钟发生器
6. I/O 和封装
   —8 脚 PDIP/SOIC：6 个可编程 I/O
7. 工作电压
   —2.7V-5.5V（ATtiny15/L1L）
   —4.0V-5.5V（ATtiny15/L）
8. 内部系统时钟
   —0.8 – 1.6MHz
9. 商业及工业温度范围

管脚配置

描述

ATtiny15/L 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，ATtiny15/L 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。ATtiny15/L 具有 4 个单端及一个 20 倍增益的差分 ADC 通道。高速 PWM 输出使得 ATtiny15/L 十分适合于电池充电器应用和电源调节电路。

图 1 ATtiny15/L 结构方框图

ATtiny15/L 具有以下特点：1K 字节 FLASH，64 字节 EEPROM，6 个通用 I/0 口，32 个通用工作寄存器，两个 8 位 T/C（一个具有 PWM 输出），内部振荡器，内外中断源，可编程的看门狗定时器，4 通道 10 位 ADC（其中之一为差分通道），以及 3 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。外部中断或硬件复位可以唤醒此状态。引脚电平变化中断的特点使得 ATtiny15/L 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。ATtiny15/L 同时还有一个 ADC 噪声抑制模式以减少 ADC 转换时的噪声。在此模式下，只有 ADC 工作。

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，ATtiny15/L 为许多嵌入式控制应用提供了灵活而低成本的方案。

ATtiny15/L 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）。PB5 是输入口。PB5 的使用由熔丝位定义。此管脚还可以作为外部中断或 ADC 输入通道 0。B 口还是模拟 I/O 引脚。

表 1 B 口的其他功能

引　脚	功　能
PB0	MOSI（SPI 数据输入） AIN0（模拟比较器输入通道 0）

	VREF（ADC 电压基准）
PB1	MISO（SPI 数据输出） AIN1（模拟比较器输入通道 1） OCP（T/C1 的 PWM 输出）
PB2	SCK（SPI 时钟输入） INT0（外部中断 0 输入） ADC1（ADC 输入通道 1） T0（T/C0 计数器外部输入）
PB3	ADC2（ADC 输入通道 2）
PB4	ADC3（ADC 输入通道 3）
PB5	RESET（外部复位输入） ADC0（ADC 输入通道 0）

内部振荡器

内部振荡器的标称频率为 1.6MHz。**内嵌的调协功能可以修正偏差**（0.8MHz - 1.6MHz）。**通过对** 8 **位** OSCCAL **寄存器的控制，可以得到小于** 1%**的调谐率。**

内部 PLL **对系统时钟进行** 16 **倍频，为** T/C1 **提供时钟信号** PCK，**最大值为** 25.6MHz。

结构纵览

图 2 ATtiny15/L/L 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。**

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所有地址空间。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。 ATtiny15/L **的堆栈为** 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 **等。** 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 3 AVR CPU **通用工作寄存器**

通

R0
R1
R2
…

用寄存器
…
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 和 R31 组成一个 16 位的指针，用于间接访问 FLASH 和寄存器文件。当访问寄存器文件时，R31 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

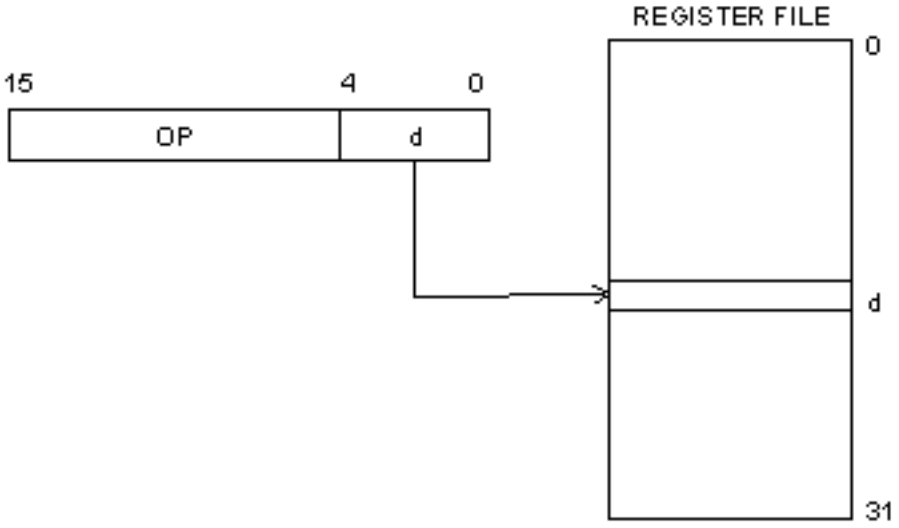
ATtiny15/L 具有 1K 字节的 FLASH。因为所有的指令为 16 位宽，故尔 FLASH 结构为 512×16。FLASH 的擦除次数至少为 1000 次。

ATtiny15/L 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 4

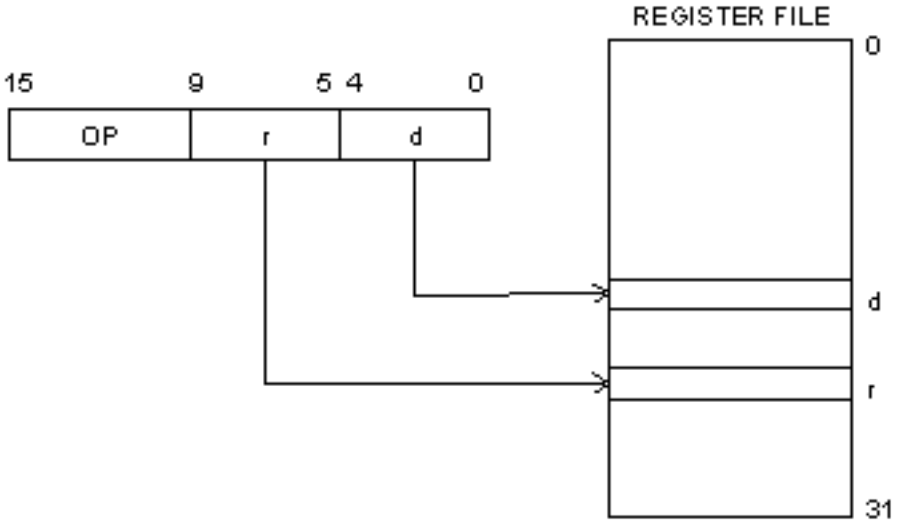


单寄存器间接寻址：

图 5

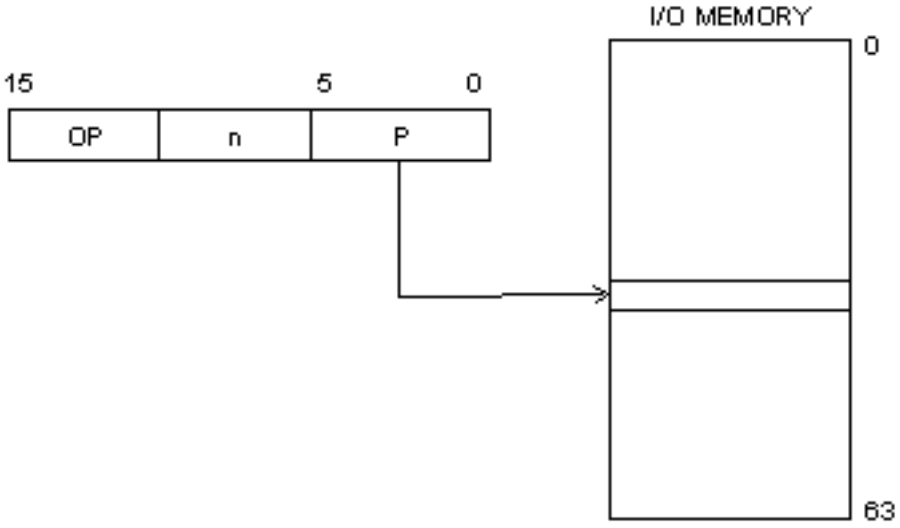
双寄存器直接寻址：

图 6



I/O 直接寻址：

图 7



程序相对寻址，RJMP 和 RCALL：

图 8

使用 LPM指令的常数寻址：

图 9

子程序和中断硬件堆栈

ATtiny15/L 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。
如果子程序或中断将 4 个返回地址 A1， A2， A3 和 A4 连续推入堆栈，则返回的将是 A4，A3，及两个 A2。

EEPROM

ATtiny15/L 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 10 取指与指令执行同时进行

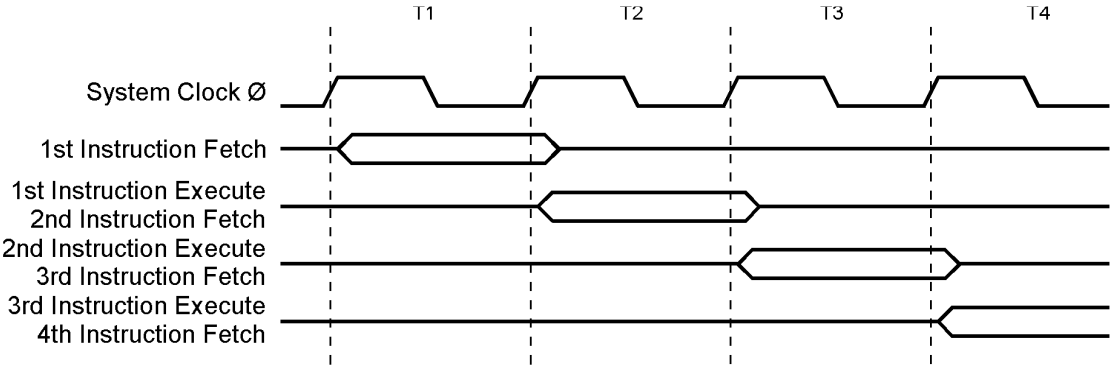
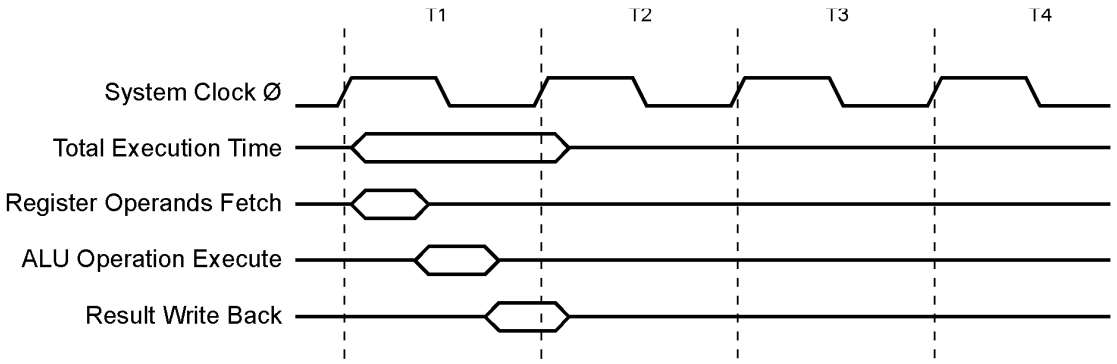


图 11 单时钟 ALU 操作



I/O 内存

表 1 ATtiny15/L 的 I/O 空间

地址	名称	功能
$3F	SREG	状态寄存器
$3B	GIMSK	通用中断屏蔽寄存器
$3A	GIFR	通用中断标志寄存器
$39	TIMSK	T/C 屏蔽寄存器
$38	TIFR	T/C 中断标志寄存器
$35	MCUCR	MCU 控制寄存器

$34	MCUSR	MCU 状态寄存器
$33	TCCR0	T/C0 控制寄存器
$32	TCNT0	T/C0（8 位）
$31	OSCCAL	振荡器标度寄存器
$30	TCCR1	T/C1 控制寄存器 A
$2F	TCNT1	T/C1
$2E	OCR1A	T/C1 输出比较寄存器 A
$2D	OCR1B	T/C1 输出比较寄存器 B
$2C	SFIOR	特殊功能 I/O 寄存器
$21	WDTCR	看门狗控制寄存器
$1E	EEAR	EEPROM 地址寄存器
$1D	EEDR	EEPROM 数据寄存器
$1C	EECR	EEPROM 控制寄存器
$18	PORTB	B 口数据寄存器
$17	DDRB	B 口数据方向寄存器
$16	PINB	B 口输入引脚
$08	ACSR	模拟比较器控制及状态寄存器
$07	ADMUX	ADC 多路选择寄存器
$06	ADCSR	ADC 控制和状态寄存器
$05	ADCH	ADC 数据寄存器高字节
$04	ADCL	ADC 数据寄存器低字节

AVRATtiny15/L 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。

一些状态标志位的清除是通过写“1”来实现的。CBI 和 SBI 指令读取已置位的标志位时，会回写“1”，因此会清除这些标志位。

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

ATtiny15/L **有** 9 **个中断源。每个中断源在程序空间都有一个独立的中断向量。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 2**。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 2　复位与中断向量

向量	程序地址	来源	定义
1	$000	RESET	**硬件管脚，上电，** BOD **和看门狗复位**
2	$001	INT0	**外部中断** 0
3	$002	I/O **口**	**引脚电平变化中断**
4	$003	TIMER1，COMP	T/C1 **比较匹配**
5	$004	TIMER1，OVF	T/C1 **溢出**
6	$005	TIMER0，OVF	T/C0 **溢出**
7	$006	EE_RDY	EEPROM **准备好**
8	$007	ANA_COMP	**模拟比较器**
9	$008	ADC	ADC **转换结束**

设置中断向量地址最典型的方法如下：

```
地址        标号        代码        ,          注释
$000                    RJMP        RESET      ；复位
$001                    RJMP        EXT_INT0   ；IRQ0
$002                    RJMP        PIN_CHANGE ；引脚电平中断
$003                    RJMP        TIM1_COMP  ；T1 比较匹配
$004                    RJMP        TIM1_OVF   ；T1 溢出
$005                    RJMP        TIM0_OVF   ；T0 溢出
$006                    RJMP        EE_RDY     ；EEP 准备好
$007                    RJMP        ANA_COMP   ；模拟比较器
$008                    RJMP        ADC        ；ADC 结束
；
$009        MAIN:       <指令>      XXX        ；主程序开始
—           —           —           —
```

复位源

ATtiny15/L **有** 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** 500ns **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**—相对跳转指令—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 12 为复位电路的逻辑图。表 3 和表 4 定义了复位电路的时序和电参数。

图 12　复位逻辑

表 3　复位电参数（V_{CC} = 5.0V）

符号	参数	条件	Min	Typ.	Max	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	1.7	2.2	2.7	V
	上电复位电压门限（下降）	BOD 禁止	0.4	0.6	0.8	V
		BOD 使能	1.7	2.2	2.7	V
V_{RST}	管脚门限电压		-	-	0.85V_{CC}	V
V_{BOT}	BOD 门限电压	(BODLEVEL = 1)	2.6	2.7	2.8	V
		(BODLEVEL = 0)	3.8	4.0	4.2	V

注：1.除非电源电压低于 V_{POT}，否则上电复位不会发生。

表 4　复位延时选择[1]

BODEN [2]	CKSEL[1:0] [2]	起动时间 t_{TOUT}，V_{CC} = 2.7V	起动时间 t_{TOUT}，V_{CC} = 5.0V	推荐用法
x	00	256ms + 1K CK	64ms + 18 CK	BOD 禁止，电源慢速稳定
x	01	256ms + 1K CK	64ms + 18 CK	BOD 禁止，电源慢速稳定
x	10	16ms + 18 CK	4ms + 18 CK	BOD 禁止，电源快速稳定
1	11	32μs + 18 CK	8μs + 18 CK	BOD 禁止
1	11	128μs + 18 CK	32μs + 18 CK	BOD 使能

注意：1、上电时，起动时间增加 0.6ms

2、“1”代表编程，“0”表示未编程

表 4 表示的是复位时的起动时间。如果 CPU 从掉电模式唤醒，则只使用时钟记数部分。看门狗振荡器用于对起动时间的实时部分的定时。看门狗振荡器溢出时间与周期的关系见表 5。看门狗振荡器的频率与电压有关。器件出厂时 CKCEL = “00”。

表 5　看门狗振荡周期数

V_{CC}	溢出时间	周期数
2.7V	32μs	8
2.7V	128μs	32
2.7V	16μs	4K
2.7V	256μs	64K
5.0V	8μs	8
5.0V	32μs	32
5.0V	4μs	4K
5.0V	64μs	64K

上电复位：

上电复位（POR）脉冲由片内检测电路产生。标称检测电平为 2.2V。当 V_{CC} 低于检测电平

时 POR 动作。POR 电路可用于触发启动复位和检测电源故障。

POR 电路保证器件在 V_{CC} 没有达到安全电平时不启动。达到安全电平后一个延时计数器产生一段时间的延迟，使得器件在 V_{CC} 上升阶段一直复位。用户可以通过 CKSEL 对延迟时间进行控制（见表 4）。当 V_{CC} 低于要求电平时，复位立即重新开始。

图 13　MCU 启动，/RESET 与 V_{CC} 相连

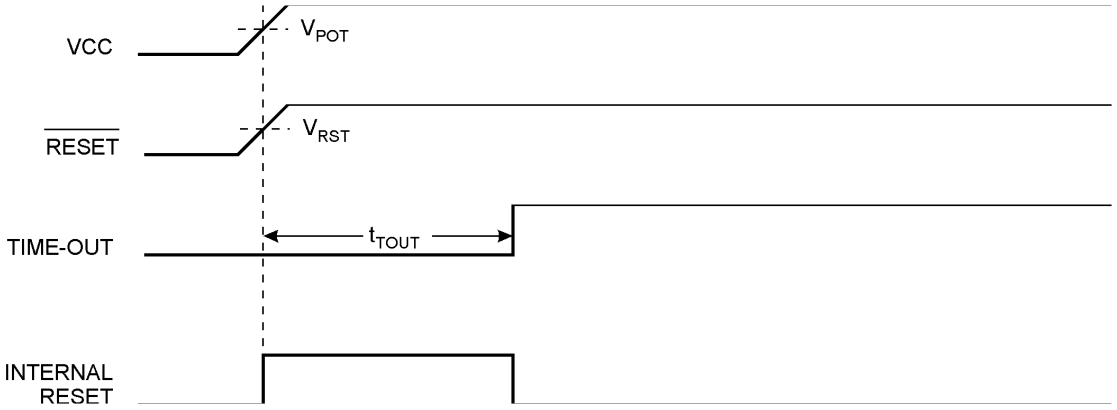
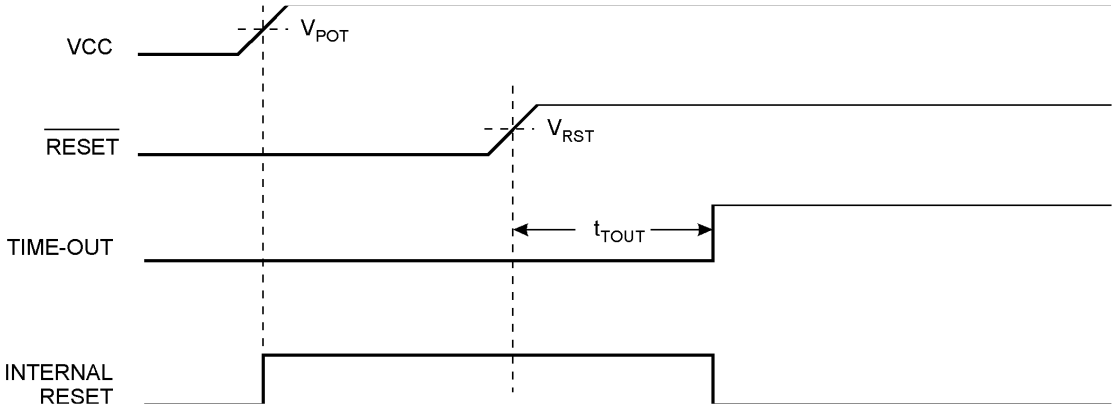


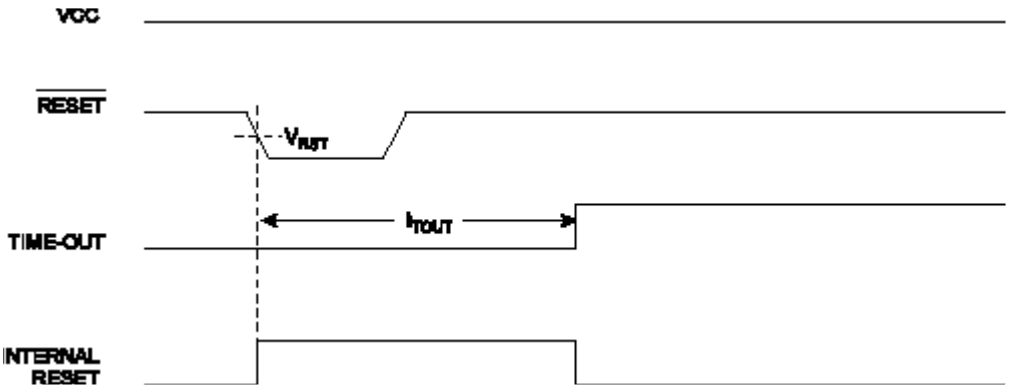
图 14　MCU 启动，/RESET 由外电路控制



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 15　工作期间的外部复位



Brown-Out 检测

ATtiny15/L 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。延迟时间与 POR 的相同（见表 3）。BOD 触发电平由 BODLEVEL 控制为 2.7V（BODLEVEL 未编程）或 4.0V（BODLEVEL 编程）。触发电平有 50mV 的冗余。

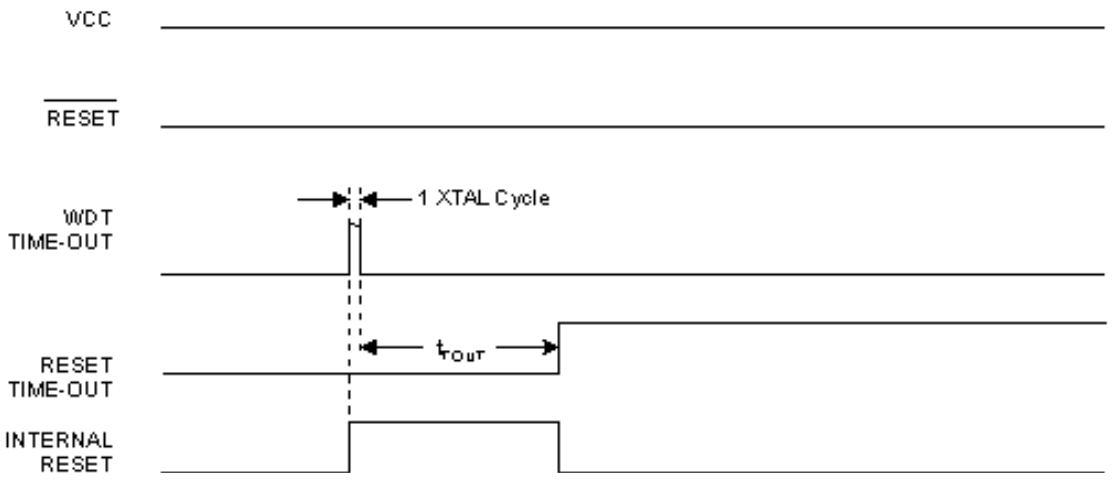
触发电平为 4.0V 时，只要 V_{CC} 低于这个值 3μs，BOD 就起作用；而当触发电平为 2.7V 时时间延长到 7μs。

图 16　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 17　工作期间的看门狗复位



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。

内部电压基准

ATtiny15/L **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。**
模拟比较器的 2.56V **基准也是由内部基准产生。**

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**）**
3、ADC **使能**
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。能隙基准源大概要消耗** 10μA**。**

中断处理

ATtiny15/L **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**”，则中断标志位挂起，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。然后被挂起的各个中断按中断优先级依次中断。**
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。**
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**（**2 **个字节**）**自动入栈，而** SP **减** 2**。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。

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	-	**OCIE1A**	-	-	-	**TOIE1**	**TOIE0**	-
读/写	R/W	R/W	R	R	R/W	R	R/W	R
初始值	0	0	0	0	0	0	0	0

位 7、5、4、3、0：保留

OCIE1A：T/C1 输出比较匹配中断使能

当 TOIE1 和 I 都为“1”时，输出比较匹配中断使能。当 T/C1 的比较匹配发生，或 TIFR 中的 OCIE1A 置位，中断例程（$003）将执行。

TOIE1：T/C1 溢出中断使能

当 TOIE1 和 I 都为“1”时，T/C1 溢出中断使能。当 T/C1 溢出，或 TIFR 中的 TOV1 位置位时，中断例程（$004）得到执行。

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（$005）得到执行。

T/C 中断标志寄存器—TIFR

BIT	7	6	5	4	3	2	1	0
$38	-	**OCF1A**	-	-	-	**TOV1**	**TOV0**	-
读/写	R/W	R/W	R	R	R/W	R	R/W	R
初始值	0	0	0	0	0	0	0	0

位 7、5、4、3、0：保留

OCF1A：输出比较标志 1A

当 T/C1 与 OCR1A 的值匹配时，OCF1A 置位。此位在中断例程里硬件清零，或者通过对其写“1”来清零。当 SREG 中的位 I、OCIE1A 和 OCF1A 一同置位时，中断例程得到执行。

TOV1：T/C1 溢出中断标志位

当 T/C1 溢出时，TOV1 置位。执行相应的中断例程后此位硬件清零。此外，TOV1 也可以通过写“1”来清零。当 SREG 中的位 I、TOIE1 和 TOV1 一同置位时，中断例程得到执行。

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。应当注意，如果中断使能，则即使 INT0 配置为输出，中断照样会被触发。此特点提供了一个产生软件中断的方法。触发方式可以为上升沿，下降沿或低电平。这些设置由 MCU 控制寄存器 MCUCR 决定。当设置为低电平触发时，只要电平为低，中断就一直触发。

引脚电平变化中断

引脚变化中断由输入口或 I/O 口电平变化触发。如果中断使能，即使管脚配置为输出，只要电平发生了变化，中断就会发生。这个特性可以用来产生软件中断。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。只有持续时间长于一个 CPU 周期的引脚变化才会产生中断，如果时间太短，中断就不一定会产生。

MCU 控制寄存器—MCUCR

BIT	7	6	5	4	3	2	1	0
$35	-	**PUR**	**SE**	**SM1**	**SM0**	-	**ISC01**	**ISC00-**
读/写	R	R/W	R/W	R/W	R/W	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

PUD 置位导致 ATtiny15/L 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。

SM1/0：休眠模式

此位用于选择休眠模式。

表 6　休眠模式

SM1	SM0	描述
0	0	空闲模式
0	1	ADC 噪声消减模式
1	0	掉电模式
1	1	保留

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

注意：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。**

休眠模式

进入休眠模式的条件是 SE **为“**1**”，然后执行** SLEEP **指令。**SM1/0 **用于控制休眠模式。使能的中断将唤醒** MCU**。**MCU **停止** 4 **个时钟周期，然后执行中断例程，完后执行** SLEEP **以后的指令。如果** MCU **是由于引脚电平变化中断从掉电模式唤醒，则** MCU **在执行完中断例程后，执行** SLEEP **以后的两条指令。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 SM1/0 **为“**00**”时，**SLEEP **指令将使** MCU **进入闲置模式。在此模式下，**CPU **停止运行，而** ADC**、模拟比较器、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** ADC **使能，则进入此模式时自动起动一次转换。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**。**

ADC 噪声消减模式

当 SM1/0 **为“**01**”时，**SLEEP **指令使得** MCU **进入** ADC **噪声消减模式。在此模式下，**CPU **停止运行，而** ADC**、外部中断、引脚变化中断及看门狗继续工作。但是请注意此时时钟系统及** PLL **都还在工作。这样可以减少** ADC **的干扰，提高** ADC **的精度。如果** ADC **使能，则进入此模式时自动起动一次转换。除了看门狗复位和外部复位，还有外部低电平复位，引脚电平变化中断和** ADC **中断可以唤醒** MCU**。**

掉电模式

当 SM1/0 **为“**10**”时，**SLEEP **指令将使** MCU **进入掉电模式。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。电平由看门狗一个周期内采样两次。如果在此期间有合适的电平，则** MCU **唤醒。如果电平保持** 2 **个以上看门狗时钟，**MCU **就可以唤醒。标称的看门狗振荡周期为** 2.9 μs，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。**

可定标内部 RC 振荡器

内部 RC 振荡器在 5V、25℃的条件下，典型振荡频率为 1.6MHz。这个时钟为系统时钟。通过改变寄存器 OSCCAL 的内容，可以对其进行定标。

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。

为产生快速外围时钟而设计的 PLL

内部 PLL 可以产生 16 倍于系统时钟的信号。此信号可以用作 T/C1 的时钟。
PLL 锁定于内部 RC 振荡器。故而对 RC 振荡器的调节（通过 OSCCAL）同样会影响 PLL。如果 RC 振荡器的频率超过 1.75MHz，T/C1 将工作不正常。因此建议不要将 RC 振荡器的频率调整到超过 1.75MHz 以保证整个器件工作正常。

定时器/计数器

ATtiny15/L 内部有两个 8 位通用定时器/计数器。T/C 从同一个 10 位的预分频定时器取得预分频的时钟。T/C0 使用片内时钟；而 T/C1 既可以使用片内时钟，也可以使用快速外围时钟（PCK）。

T/C0 预分频器

图 18 T/C0 预分频器

T/C0 的时钟可以选择 CK、CK/8、CK/64、CK/256、CK/81024 或外部输入。另外还可以由 T/C0 控制寄存器 TCCR0 来停止它。置位 SFIOR 的 PSR0 可以复位预分频器，使得用户可以操作可预期的预分频器。

T/C1 预分频器

图 19 T/C1 预分频器

T/C0 的时钟可以选择 PCK、PCK/2、PCK/4、PCK/8、CK（=PCK/16）、CK/2、CK/4、CK/8、CK/16、CK/32、CK/64、CK/128、CK/256、CK/512、CK/1024 或外部输入。另外还可以由 T/C1 控制寄存器 TCCR1 来停止它。置位 SFIOR 的 PSR1 可以复位预分频器，使得用户可以操作可预期的预分频器。

特殊功能 I/O 寄存器—SFIOR

BIT	7	6	5	4	3	2	1	0
$2C	-	-	-	-	-	**FOC1A**	**PSR1**	**PSR0**
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

FOC1A：强迫输出比较 1A

对其写“1”后 PB1（OC1A）**将按照** COM1A1 **和** COM1A0 **的设置变化。（若** FOC1A **与** COM1A1 **和** COM1A0 **一同置位，则设置不会生效，直到下一次比较匹配或强制比较匹配。）**FOC1A **置位将改变输出引脚，而无需等待定时器比较匹配发生。但不会有中断产生，在** CTC1 **置位时也不会清零** T/C1。FOC1A **的读返回值总为“**0**”。**T/C1 **工作于** PWM **模式时** FOC1A **置位与否没有作用。**

PSR1：复位 T/C1 的预分频器

读返回值总为“0**”。**

PSR0：复位 T/C0 的预分频器

读返回值总为“0**”。**

8 位 T/C0

图 20 **为** T/C0 **的框图。**

图 20 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。**TIFR **为状态标志寄存器，**TCCR0 **为控制寄存器，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 8 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 PB0（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。

8 位 T/C1

T/C1 具有高分辨率、高精度的特点，同时还可以用作高精度，高速度的 8 位 PWM，其时钟可高达 25.6MHz。PWM 由 T/C1 与输出比较寄存器构成。
图 21 为 T/C0 的框图。

图 21　　T/C1 工作框图

T/C1 的各种状态位集中于 TIFR，TCCR1 为控制寄存器，而中断使能/禁止则由 TIMSK 决定。T/C1 包括两个输出比较寄存器，OCR1A 和 OCR1B。在普通模式下，输出比较功能只涉及 OCR1A，包括在比较匹配时清除计数器的值，以及操作 PB1（OC1A）引脚的输出。
在 PWM 模式时，OCR1A 保存与 T/C1 值相比较的数值。定时器向上记数，直到 OCR1B 保存的数值，然后又从$00 开始记数。PWM 频率的选择十分灵活，表 13 给出了 PWM 频率从 10kHz 到 150kHz 时的 OCR1B 的值。

T/C1 控制寄存器—TCCR1

BIT	7	6	5	4	3	2	1	0
$30	**CTC1**	**PWM1**	**COM1A1**	**COM1A0**	**CS13**	**CS12**	**CS11**	**CS10**
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

CTC1：比较匹配时清除 T/C1
如果 CTC1 置位，则比较匹配后 T/C1 复位为$00。否则，比较匹配不影响 T/C1 的运行。
PWM1：PWM 使能
COM11，COM10：比较输出模式，位 1 和 0
COM11 和 COM10 决定 T/C1 的比较匹配发生时输出引脚的动作。这些动作都将影响管脚 PB1（OC1A）。这是 I/O 口的第二功能，相应的方向控制位要设置为“1”以便将其配置为输出。具体配置见表 9。

表 9　比较模式选择

COM1A1	COM1A0	描述
0	0	T/C1 与输出引脚 OC1 断开
0	1	OC1 输出变换
1	0	清除 OC1
1	1	置位 OC1

注意：在 PWM 模式下，这些位有不同的功能。改变 COM11/COM10 时要关闭疏输出比较中断，否则在改

变这两位时有可能引发不必要的中断。

CS13、CS12、CS11、CS10：时钟选择

表 10 T/C1 预分频选择

CS13	CS12	CS11	CS10	描述
0	0	0	0	停止
0	0	0	1	CK*16（=PCK）
0	0	1	0	CK*8
0	0	1	1	CK*4
0	1	0	0	CK*2
0	1	0	1	CK
0	1	1	0	CK/2
0	1	1	1	CK/4
1	0	0	0	CK/8
1	0	0	1	CK/16
1	0	1	0	CK/32
1	0	1	1	CK/64
1	1	0	0	CK/128
1	1	0	1	CK/256
1	1	1	0	CK/512
1	1	1	1	CK/1024

T/C1—TCNT1

BIT	7	6	5	4	3	2	1	0
$2F	**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/C1 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。

T/C1 输出比较寄存器 0—OCR1A

BIT	7	6	5	4	3	2	1	0
$2E	**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

OCR10 包含与 T/C1 连续不断地进行比较的数据。匹配后的动作由 TCCR1 指定。只有当 T/C1 记数到 OCR1A 的值时比较匹配才会发生。用软件写的方法将 TCNT1 与 OCR1A 的值设置为相同是不会引发比较匹配的。

比较匹配发生后匹配中断标志置位。

T/C1 的 PWM 模式

选择 PWM 模式后，T/C1 与 OCR10 和 OCR11 共同组成一个 8 位无尖峰的自由运行的 PWM，其输出引脚为 PB1（OC1A）。T/C1 作为向上计数器，从 0 记数到 OCR1B，然后又从$00 开始。当计数器中的数值和 OCR1A 的数值一致时，PB1（OC1A）引脚按照 COM1A0 和 COM1A1 的设置动作。

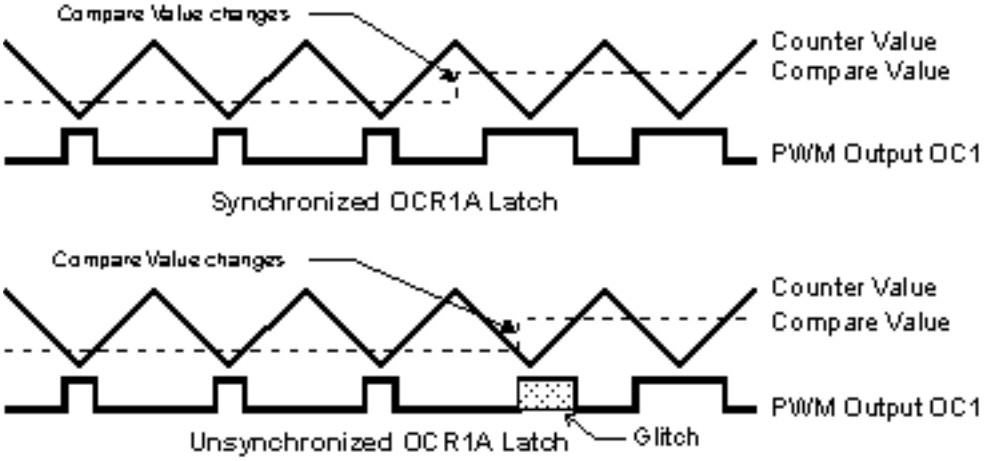
表 11 PWM 模式下的比较模式选择

COM11	COM10	OC1

0	0	不用作 PWM 功能
0	1	不用作 PWM 功能
1	0	匹配时清除 OC1；TCNT1 = $00 时置位 OC1
1	1	匹配时置位 OC1；TCNT1 = $00 时清零 OC1

注意： 在 PWM 模式下，OCR1A 首先存储在一个临时的位置，等到 T/C1 达到 OCR1B 时才真正存入 OCR1A。这样可以防止在写 OCR1A 时由于失步而出现奇数长度的 PWM 脉冲。

图 22 失步的 OCR1A 锁存



如果在执行写和锁存操作的时候读取 OCR1A，读到的是临时位置的数据。

OCR1A 的值为$00 或为 OCR1B 的值时 OC1A 的输出见表 12。

T/C1 输出比较寄存器 1—OCR1B

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

OCR1B 用于限制 T/C1 在 PWM 模式下的满度值。

表 13 OCR1A = $00 或 OCR1B 时的 PWM 输出

COM1A1	COM1A0	OCR1A	OC1A
1	0	$00	L
1	0	OCR1B	H
1	1	$00	H
1	1	OCR1B	L

在 PWM 模式下，当计数器达到 OCR11 时将置位 TOV1。此时发生的中断与正常情况下的中断是完全一样的。

PWM 的频率为定时器的时钟频率除以 OCR1B + 1 。

表 13 T/C1 时钟预分频选择

时钟选择	OCR1B	PWM 频率
CK	159	10kHz
PCK/8	159	20kHz
PCK/4	213	30kHz
PCK/4	159	40kHz
PCK/2	255	50kHz
PCK/2	213	60kHz

PCK/2	181	70kHz
PCK/2	159	80kHz
PCK/2	141	90kHz
PCK	255	100kHz
PCK	231	110kHz
PCK	213	120kHz
PCK	195	130kHz
PCK	181	140kHz
PCK	169	150kHz

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。

为了防止不小心关闭看门狗，需要有一个特定的关闭程序。

图 23　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

当 WDE 清零时此位必须为“1”才能关闭看门狗。在置位的 4 个时钟后，硬件对其清零。

WDE：看门狗使能

WDE 为“1”时，看门狗使能。只有在 WDTOE 为“1”时 WDE 才能清零。以下为关闭看门狗的步骤：

1．**在同一个指令内对 WDTOE 和 WDE 写逻辑 1，即使 WDE 已经为“1”。**

2．**在 4 个时钟之内，对 WDE 写逻辑 0。**

WDP2..0：预分频器

表 14　　看门狗定时器预分频选择

WDP2	WDP1	WDP0	振荡周期
0	0	0	16K
0	0	1	32K
0	1	0	64K
0	1	1	128K
1	0	0	256K
1	0	1	512K
1	1	0	1024K
1	1	1	2048K

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 1.1~2.1ms 之间。自定时功能可以让用户监测何时开始写下一字节。如果用户要操作 EEPROM，应当注意如下问题：在电源滤波时间常数比较大的电路中，上电/下电时 V_{CC} 上升/下降会比较慢。此时 MCU 将工作于低于晶振所要求的电源电压。在这种情况下，程序指针有可能跑飞，并执行 EEP 写指令。为了保证 EEP 的数据完整性，建议使用电压复位电路。

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：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（片内 RC 振荡器标定为 1.6MHz 时为 1.3ms 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。这可以通过内部复位电路（BOD—Brown-out Detection）来完成。
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 24 所示。

图 24 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	X	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

置位后 ATtiny15/L 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 15　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

注意：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。

模数转换器，模拟分路器和增益调节

特点：

- 10 位精度
- ±2LSB 精确度
- 0.5LSB 集成非线性度
- 可选的偏移消除
- 65 - 260μs 转换时间
- 4 个单端输入通道
- 一个差分输入通道，具有可选的 20 倍增益
- 2.56V 内部电压基准
- 差分输入范围：0 – 2.56V
- 单端输入范围：0 –VCC
- 可选的数据左对齐输出
- 自由运行模式和单次转换模式
- ADC 转换结束中断
- 睡眠模式噪声消除

ATtiny15/L 具有 10 位精度的逐次逼近型 AD 转换器。ADC 与一个 4 通道的模拟多路器相连，具有一个差分输入通道和 4 个单端输入通道，B 口作为 ADC 的输入引脚。差分输入（PB3，PB4）前端有可编程的 20 倍增益调节。单端输入相对地而言。ADC 包含一个采保放大器。ADC 框图见图 25。

内部基准电压可以通过 AREF（PB0）外接电容进行解耦。VCC 可以作为单端输入的参考电压，也可以用外部参考电压。ADMUX 的 REFSn 控制这些属性。

图 25　ADC 框图

操作

ADC 的最小值代表 GND，最大值代表所选电压基准减一。

电压基准可以通过 ADMUX 的 REFSn 选取：VCC，AREF 或内部 2.56V 基准。内部基准可以通过在 AREF 引脚外接电容以削弱噪声的影响。

ADMUX 的 MUXn 位决定输入通道。ADC2 和 ADC3 还可以分别作为差分输入的正输入及负输入。

若 ADC2 同时作为差分输入的正负极，则增益放大器及转换电路的偏移就可以作为 ADC 结果直接测量出来。后续测量应该减去这个值，将偏移误差降低到 1LSB 以下。

ADC 可以工作于两种模式一单次转换及自由运行方式。在单次转换模式下，用户必须启动每一次转换；而在自由运行方式下，ADC 会连续采样并更新 ADC 数据寄存器。ADCSR 的 ADFR 位用于选择模式。

ADC 由 ADCSR 的 ADEN 位控制使能。使能 ADC 后，第一次转换将引发一次哑转换过程以初始化 ADC，然后才真正进行 AD 转换。对用户而言，此次转换过程比其他转换过程要多 12 个 ADC 时钟周期。

ADSC 置位将启动 AD 转换。在转换过程当中 ADSC 一直保持为高；转换结束后 ADSC 硬件清零。如果在转换过程当中通道改变了，ADC 首先要完成当前的转换，然后通道才会改变。

ADC 产生 10 位的结果，ADCH 和 ADCL，ADCH 的高 6 位为无效数据（右对齐 - Right Adjusted）。用户也可以通过选择 ADMUX 的 ADLAR 来使 ADCH 的高 6 位有效（左对齐 - Left Adjusted）。

如果数据左对齐，而且只有 8 位精度，则只要读取 ADCH 就够了。否则首先要读 ADCL。一旦开始读 ADCL，ADC 对数据寄存器的访问就被禁止了。也就是说，如果读取了 ADCL，那么即使在读 ADCH 之前另一次 ADC 结束了，两个寄存器的值也不会被新的 ADC 结果更新，此次转换的数据将丢失。当读完 ADCH 之后，ADC 才能继续对 ADCH 和 ADCL 进行访问。

ADC 结束后会置位 ADIF。即使发生如上所说的由于 ADCH 未被读取而丢失转换数据的情况，ADC 结束中断仍将触发。

预分频器

图 26　ADC 预分频器

ADC 有一个预分频器，可以将系统时钟调整到可接受的 ADC 时钟（50 – 200kHz）。过高的频率将导致低的采样精度。

ADCSR 的 ADPS0 – ADPS2 用于产生合适的 ADC 时钟。一旦 ADCSR 的 ADEN 置位，预分频器就开始连续不断地记数，直到 ADEN 清零。ADSC 的作用是对 ADC 进行初始化。AD 转换在 ADC 时钟的上升沿启动。

正常转换时间为 13 个时钟。在某些特定情况下，ADC 需要更多的时钟来进行初始化和最小化偏移误差。这些扩展转换需要 25 个时钟，且只发生在如下事件发生之后的第一次转化：

- ADC 开始工作（ADEN 置位）
- 电压基准变化（REFSn 发生变化）
- 选择了差分通道（MUX2 置“1”）

实际采保发生于正常转换开始之后的 1.5 个 ADC 时钟，扩展转换起动之后的 13.5 个 ADC 时钟。转换结束后，数据进入 ADC 数据寄存器，ADIF 置位。对于单次转换模式，ADSC 在此时清零。软件可以重新置位 ADSC 启动新的转换。而在自由运行模式下，新的转换过程立即开始，ADSC 一直保持为高。工作于 200kHz 的自由运行模式具有最快的转换速度：65μs，亦即 15.2KSPS（Samplings Per Second）。转换时序见表 16。

图 27　单次转换的时序

图 28 自由运行的时序

图 29 自由运行的时序

表 16　ADC 转换时间

条件	采保（转换起动后的周期）	转换时间（周期数）	转换时间（μs）
扩展转换	13.5	25	125 – 500
普通转换	1.5	13	65 – 250

ADC 噪声抑制功能

ADC 具有消除由 CPU 核引入的噪声的功能。实现过程如下：

1、使能 ADC，选择单次转换模式，并使能转换结束中断

ADEN = 1

ADSC = 0

ADFR = 0

ADIE = 1

2、进入空闲状态。一旦 CPU 停止，ADC 将开始转换。

3、如果在 ADC 转换结束中断之前没有发生其他中断，则 ADC 转换结束中断将唤醒 MCU 并执行中断例程。

ADC 多址许则选择寄存器—ADMUX

BIT	7	6	5	4	3	2	1	0
$07	**REFS1**	**REFS0**	**ADLAR**	**-**	**-**	**MUX2**	**MUX1**	**MUX0**
读/写	R/W	R/W	R/W	R	R	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

位 4、3：保留

REFS1/0：基准选择

用于选择 ADC 的基准电压。在转换过程当中改变这两位不会影响到当前转换。而下一次转换则要花 25 个 ADC 时钟。

表 17　ADC 电压基准选择

REFS1	REFS0	选取的电压基准
0	0	VCC
0	1	外部电压基准（PB0-AREF）
1	0	内部电压基准，不外接电容，与 PB0 脱离
1	1	内部电压基准，外接电容于 PB0

ADLAR：ADC 数据左对齐

MUX2..MUX0：模拟通道选择位

用于选择 ADC 的模拟输入通道

表 18　输入通道及增益选择

MUX2..0	单端输入	差分输入正极	差分输入负极	增益
000	ADC0（PB5）	无效		
001	ADC1（PB2）			
010	ADC2（PB3）			
011	ADC3（PB4）			
100 [1)]	无效	ADC2（PB3）	ADC2（PB3）	1
101 [1)]		ADC2（PB3）	ADC2（PB3）	20
110		ADC2（PB3）	ADC3（PB4）	1
111		ADC2（PB3）	ADC3（PB4）	20

注：1、仅用于偏移标度

ADC 控制和状态寄存器—ADCSR

BIT	7	6	5	4	3	2	1	0
$06	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS 1	ADPS 0
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

ADEN：ADC 使能

ADSC：ADC 开始转换

当 ADC 工作于单次转换模式时，这一位必须设置为“1”以启动每一次转换。而对于自由运行模式，则只需在第一次转换时设置一次。不论设置动作是在 ADEN 置位之后还是一同进行，ADC 都将进行一次哑转换以初始化 ADC。

转换过程中 ADSC 保持为高。实际转换过程结束后，但在转换结果进入 ADC 数据寄存器之前（差一个 ADC 时钟），ADSC 变为低。这样就允许在当前转换完成之前（ADSC 变低之时）对下一次转换进行初始化，一旦当前转换彻底完成立即就可以进行新的一次转换过程。在哑转换过程当中，ADSC 保持为高。

对 ADSC 写零没有意义。

ADFR：ADC 自由运行模式选择

这一位置位后 ADC 工作于自由运行模式。ADC 将连续不断地进行采样和数据更新。

ADIF：ADC 中断标志

ADC 完成及数据更新完成后 ADIF 置位。如果 I 和 ADIE 置位，则 ADC 结束中断发生。在中断例程里 ADIF 硬件清零。写“1”也可以对其清零。因此要注意对 ADCSR 执行读-修改-写操作时会使挂起的中断

ADIE：ADC 中断使能

ADPS2..ADPS0：ADC 预分频器选择

表 19　ADC 预分频器选择

ADPS2	ADPS1	ADPS0	分频因子

0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

ADC 数据寄存器—ADCL 和 ADCH

ADLAR = 0：

BIT	15	14	13	12	11	10	9	8
$05	-	-	-	-	-	-	**ADC9**	**ADC8**
$04	**ADC7**	**ADC6**	**ADC5**	**ADC4**	**ADC3**	**ADC2**	**ADC1**	**ADC0**
	7	6	5	4	3	2	1	0
读/写	R	R	R	R	R	R	R	R
	R	R	R	R	R	R	R	R
初始值	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0

ADC 数据寄存器—ADCL 和 ADCH

ADLAR = 1：

BIT	15	14	13	12	11	10	9	8
$05	**ADC9**	**ADC8**	**ADC7**	**ADC6**	**ADC5**	**ADC4**	**ADC3**	**ADC2**
$04	**ADC1**	**ADC0**	-	-	-	-	-	-
	7	6	5	4	3	2	1	0
读/写	R	R	R	R	R	R	R	R
	R	R	R	R	R	R	R	R
初始值	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0

读取 ADCL 后，ADC 数据要一直等到 ADCH 也被读取才会更新。如果数据格式为左对齐，而且精度只有 8 位，则只要读取 ADCH 就可以了。必须先读 ADCL 后再读 ADCH。

扫描多个通道

由于模拟通道的转换总是要延迟到转换结束，因此自由运行模式可以用来扫描多个通道而不中断转换器。一般情况下，ADC 转换结束中断用于修改通道，但需注意：
中断在转换结果可读时触发。在自由运行模式下，下一次转换在中断触发的同时启动。在 ADC 中断触发/新一次转换开始后改变 ADMUX 将不起作用。

ADC 噪声消除技术

AT90S2333/4433 的内外部数字电路会产生 EMI，从而影响模拟测量精度。如果转换精度要求很高，则需要应用如下技术以减少噪声：

1、**ATtiny15/L 的模拟部分及其他模拟器件在 PCB 上要有独立的地线层。模拟地线与数字地线单点相连。**
2、**使模拟信号通路尽量短。要使模拟走线在模拟地上通过，并尽量远离高速数字通路。**
3、**利用 ADC 的噪声消除技术减少 CPU 引入的噪声。**

4、如果 B 口的一些引脚用作数字输出口，则在 ADC 转换过程中不要改变其状态。

ADC 特性（T_A = -40℃ - 85℃）

符号	参数	条件	Min [1]	Typ. [1]	Max [1]	单位
	精度	单端转换		10		Bits
		差分转换 1 或 20 倍增益		8		
	绝对精度	单端转换，V_{REF} = 4V ADC 时钟 = 200kHz		1	2	LSB
	绝对精度	单端转换，V_{REF} = 4V ADC 时钟 = 1MHz		4		LSB
	绝对精度	单端转换，V_{REF} = 4V ADC 时钟 = 2MHz		16		LSB
	整体非线性	V_{REF} > 2V		0.5		LSB
	差分非线性	V_{REF} > 2V		0.5		LSB
	零误差（偏移）	V_{REF} > 2V		1		LSB
	转换时间	自由运行模式	65		260	μs
	时钟频率		50		200	kHz
V_{REF}	参考电源	单端转换	GND		V_{CC}	V
		差分转换	GND		V_{CC} - 1	
R_{REF}	参考输入电阻		6	10	13	KΩ
R_{AIN}	模拟输入电阻			100		MΩ

注：1、仅为参考值，实际值待定

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..0 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或者开漏输出。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。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 20　B 口第二功能

管　脚	第二功能
PB0	MOSI（程序下载时的数据输入） AREF（ADC 电压基准） AIN0（模拟比较器正输入端）

PB1	MISO（程序下载时的数据输出） OC1A（T/C 的 PWM 输出） AIN1（模拟比较器负输入端）
PB2	SCK（程序下载时的串行时钟） INT0（外部中断 0） ADC1（ADC 输入通道 1） T0（T/C0 外部输入）
PB3	ADC2
PB4	ADC3
PB5	/RESET ADC0

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 21　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 ATtiny15/L 中，PB5 只是输入；而对于 ATtiny15/L 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 ATtiny15/L，还可以作为开漏输出。
- SCK/INT0/T0—PB2
- AIN1/OC1A/MISO—PB1
- AREF/AIN0/MOSI—PB0

程序编程

程序和数据锁定位

ATtiny15/L 具有两个锁定位，如表 22 所示。锁定位只能通过片擦除命令擦除。

表 22　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。

熔断位

ATtiny15/L 有 6 个熔断位：BODLEVEL，BODEN，SPIEN，RSTDISBL 和 CKSEL1..0。

- 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”。
- CKSEL1..0：见表 4。缺省值为“00”，64ms + 18 CK。

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。

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。

1、$000：$1E（表明是 Atmel 生产的）
2、$001：$90（1K 字节的 FLASH）
3、$002：$06（当$01 地址为$90 时，器件为 ATtiny15/L）

定标字节

ATtiny15/L 具有一个字节用来定标内部 RC 振荡器。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

ATtiny15/L
ATtiny15/L 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。
编程电压见表 23。

表 23　编程电源电压

型号	低压串行编程	高压串行编程
ATtiny15/L	2.7V – 3.6V	4.5V – 5.5V
ATtiny15	4.0V – 5.5V	4.5V – 5.5V

高压串行编程

图 30　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 30μs。
　　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 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 和 PB5；关电源。
外部 16 个时钟合一个内部时钟。数据在第 8 个外部时钟的上升沿输入/输出。

图 31　高压串行编程波形

表 24　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	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	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	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	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	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	PB0 PB1 PB2	0_0100_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_**8765**_11**43**_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 4 后等待 PB2 变高，**8..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	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 **8_765**x_x**43**x_xx		**8..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	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	

注意： **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 = RSTISBL
6 = SPIEN
7 = BODEN
8 = BODLEVEL

锁定位只能通过片擦除来清除。

高压串行编程特性

32 高压串行编程时序

表 25 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	25			ns
t_{SLSH}	SCI（PB3）脉冲（低）宽度	25			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}					ms
t_{WLWH_PFB}					ms

低压串行下载

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 33 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 ATtiny15/L，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 $t_{WD\_PROG\_FL}$/ $t_{WD\_PROG\_EE}$ 的时间（FLASH/EEPROM）。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 $t_{WD\_PROG\_FL}$/ $t_{WD\_PROG\_EE}$ 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

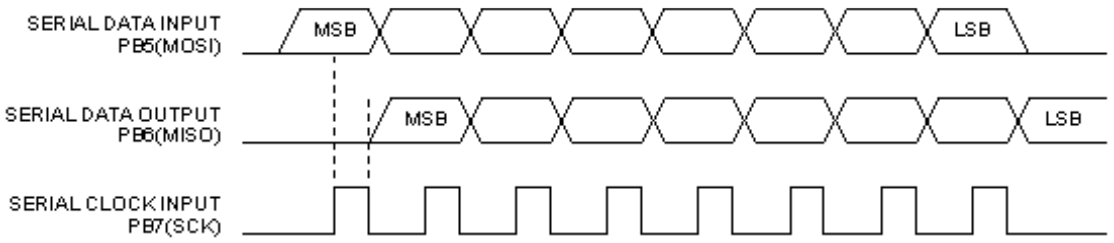


表 26　低压串行编程指令

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	**8765** 11**43**	0 代表要编程
读熔丝位	0101 0000	xxxx xxxx	xxxx xxxx	**8765** xx**43**	0 代表已编程

注意： **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 = 锁定位 1

2 = 锁定位 2

3 = CKSEL0

4 = CKSEL1

5 = RSTISBL

6 = SPIEN

7 = BODEN

8 = BODLEVEL

串行编程电特性

图 35 串行编程时序

表 27 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 – 5.5V）	0.8	1.6		MHz
t_{CLCL}	振荡周期（V_{CC} = 2.2V – 5.5V）		625	1250	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 28 擦除指令之后的最小等待时间

符号	2.7V	4.0V	5.0V

t_{WD_ERASE}	6ms	5ms	4ms

表 28 写指令之后的最小延迟时间

符号	2.7V	4.0V	5.0V
$t_{WD\_PROG\_FL}$	6ms	5ms	4ms
$t_{WD\_PROG\_EE}$	3ms	2.5ms	2ms

直流特性

T_A = -40℃到 85℃， V_{CC} = 2.7V – 5.5V（ATtiny15/L）， V_{CC} = 1.8V – 5.5V

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 PB5(ATtiny15/L)	I_{OL} =12mA， V_{CC} = 5V I_{OL} =6mA， V_{CC} = 3V			0.6 0.5	V
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

注意:

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

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 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。

5、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。

6、掉电时的最小 V_{CC} 为 1.5V（ATtiny15/L：仅当 BOD 禁止时）

ATtiny15/L 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。

定货信息

速度（MHz）	电源（V）	定货号	封装	工作范围
1.6	2.7 – 5.5	ATtiny15/L – ATtiny15/L –	8P3 8S2	商用 （0℃ - 70℃）
		ATtiny15/L – ATtiny15/L –	8P3 8S2	工业 （-40℃ - 85℃）
1.6	4.0 – 5.5	ATtiny15/L – ATtiny15/L –	8P3 8S2	商用 （0℃ - 70℃）
		ATtiny15/L – ATtiny15/L –	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.）