ATmega603/103

特点

1. AVR RISC 结构
2. AVR—高性能、低功耗 RISC 结构
   —120/121 条指令——大多数为单指令周期
   —32 个 8 位通用（工作）寄存器+外设控制寄存器
   —工作在 6MHz 时具有 6MIPS 的性能
3. 数据和非易失性程序内存
   —64K/128K 字节的在线可编程 FLASH（擦除次数：1000 次）
   —4K 字节 SRAM
   —2K/4K 字节在线可编程 EEPROM（寿命：100000 次）
   —程序加密位
   —SPI 接口，同时可用作在线下载
4. 外围（Peripheral）特点
   —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）
   —全双工 UAR
   —主/从 SPI 接口
   —自具振荡器的实时时钟 RTC
   —两个具有比较模式的可预分频（Prescale）8 位定时器/计数器
   ——一个可预分频、具有比较、捕捉和两个 8/9/10 位 PWM 功能的 16 位定时器/计数器
   —8 通道 10 位 ADC
5. 特别的 MCU 特点
   —低功耗空闲、省电和掉电模式
   —可通过软件进行选择的时钟频率
   —内外部中断源
6. 4MHz、3V、25℃条件下的功耗：
   —工作模式：5.5mA
   —空闲模式：1.6mA
   —掉电模式：<1μA
7. I/O 和封装
   —32 个可编程的 I/O 脚，8 个输出口线，8 个输入口线
   —64 脚 QFP 封装
8. 工作电压
   —2.7V-3.6V（ATmega603L 和 ATmega103L）
   —4.0V-5.5V（ATmega603 和 ATmega103）
9. 速度
   —0-4MHz（ATmega603L 和 ATmega103L）
   —0-6MHz（ATmega603 和 ATmega103）

描述

ATmega603/103 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，ATmega603/103 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。

图 1 ATmega603/103 结构方框图

ATmega603/103 具有以下特点：64K/128K 字节 FLASH，2K/4K 字节 EEPROM，4K 字节 SRAM，32 个通用 I/0 口，8 个输入口，8 个输出口 32 个通用工作寄存器，实时时钟 RTC，个具有比较模式的灵活的定时器/计数器，内外中断源，可编程的 UART，可编程的看门狗定时器，SPI 口以及三种可通过软件选择的节电模式。工作于空闲模式时，CPU 将停止运行，而寄存器、定时器/计数器、看门狗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。省电模式与掉电模式只有一点差别：省电模式下 T/C2 继续工作以维持时间基准。

器件是以 ATMEL 的高密度非易失性内存技术生产的。片内 FLASH 可以通过 SPI 接口或通用编程器多次编程。通过将增强的 RISC 8 位 CPU 与 FLASH 集成在一个芯片内，ATmega603/103 为许多嵌入式控制应用提供了灵活而低成本的方案。

ATmega603/103 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。

ATmega603 和 ATmega103 的比较

ATmega603 具有 64K 字节程序 FLASH，2K 字节 EEPROM，4K 字节 SRAM。

ATmega103 具有 128K 字节程序 FLASH，4K 字节 EEPROM，4K 字节 SRAM。

表 1 是两个器件存储器的简单比较。

表 1 存储器比较

型号	FLASH	EEPROM	SRAM
ATmega603	64K 字节	2K 字节	4K 字节
ATmega103	128K 字节	4K 字节	4K 字节

管脚配置

管脚定义

VCC、GND：电源

A 口（PA7..PA0）：

A 口是一个 8 位双向 I/O 口，每一个管脚都有内部上拉电阻。A 口的输出缓冲器能够吸收 20mA 的电流，可直接驱动 LED。当作为输入时，如果外部被拉低，由于上拉电阻的存在，管脚将输出电流。在复位过程中，A 口为三态，即使此时时钟还未起振。

在访问外部 SRAM 时 A 口作为地址/数据复用口。

B 口（PB7..PB0）:

B口是一个8位双向I/O口，每一个管脚都有内部上拉电阻。B口的输出缓冲器能够吸收20mA的电流，可直接驱动 LED。当作为输入时，如果外部被拉低，由于上拉电阻的存在，管脚将输出电流。在复位过程中，B 口为三态，即使此时时钟还未起振。

B 口作为特殊功能口的使用方方法见以后章节。

C 口（PC7..PC0）:

C 口是一个 8 位输出 I/O 口，能够吸收 20mA 的电流。

在访问外部 SRAM 时 C 口作为地址线。

在复位过程中，C 口不为三态。

D 口（PD7..PD0）:

D 口是一个带内部上拉电阻的 8 位双向 I/O 口。输出缓冲器能够吸收 20mA 的电流。当作为输入时，如果外部被拉低，由于上拉电阻的存在，管脚将输出电流。在复位过程中，D 口为三态，即使此时时钟还未起振。

D 口作为特殊功能口的使用方方法见以后章节。

E 口（PE7..PE0）:

E 口是一个带内部上拉电阻的 8 位双向 I/O 口。输出缓冲器能够吸收 20mA 的电流。当作为输入时，如果外部被拉低，由于上拉电阻的存在，管脚将输出电流。在复位过程中，E 口为三态，即使此时时钟还未起振。

E 口作为特殊功能口的使用方方法见以后章节。

F 口（PF7..PF0）:

F 口是一个 8 位输入 I/O 口，也可作为 ADC 的模拟输入。

/RESET：复位输入。超过 50ns 的低电平将引起系统复位。低于 50ns 的脉冲不能保证可靠复位。

XTAL1：振荡器放大器的输入端。

XTAL2：振荡器放大器的输出端。

TOSC1：RTC 振荡器放大器的输入端。

TOSC2：RTC 振荡器放大器的输出端。

/WR：外部 SRAM 写信号。

/RD：外部 SRAM 读信号。

ALE：访问外部存储器时的地址锁存使能信号，用于锁存低 8 位地址。

AVCC： A/D 转换器的电源。应该通过一个低通滤波器与 V_{CC} 连接。

AREF： A/D 转换器的参考电源，介于 AGND 与 AVCC 之间。

AGND：模拟地。

/PEN：串行下载的编程使能信号。

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。

图 2　振荡器连接

图 3　外部时钟驱动配置

定时器振荡器:

晶振可以直接连接到振荡器的引脚 TOSC1 和 TOSC2 而无需外部电容。振荡器已经对 32768Hz 的晶振作了优化。对外加信号的带宽为 256KHz。

结构纵览

图 4　ATmega603/103 AVR RISC **结构**

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。多数** AVR **指令都为** 16 **位长。每个程序内存地址都包含一条** 16 **位或** 32 **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。堆栈分布于通用数据 SRAM **之中，堆栈大小只受** SRAM **数量的限制。用户应该在复位例程里就初始化** SP**。**SP **为可读写的** 16 **位堆栈指针。**

4000 **字节的** SRAM **可以通过** 5 **种不同的寻址方式很容易地进行访问。**

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。**

AVR **结构的内存空间是线性的。**

通用工作寄存器文件

图 5　通用工作寄存器

	7	0	地址	
通用工作寄存器	R0		$00	
	R1		$01	
	R2		$02	
	…			
	R13		$0D	
	R14		$0E	
	R15		$0F	
	R16		$10	
	R17		$11	
	…			
	R26		$1A	X **寄存器低字节**
	R27		$1B	X **寄存器高字节**
	R28		$1C	Y **寄存器低字节**
	R29		$1D	Y **寄存器高字节**
	R30		$1E	Z **寄存器低字节**
	R31		$1F	Z **寄存器高字节**

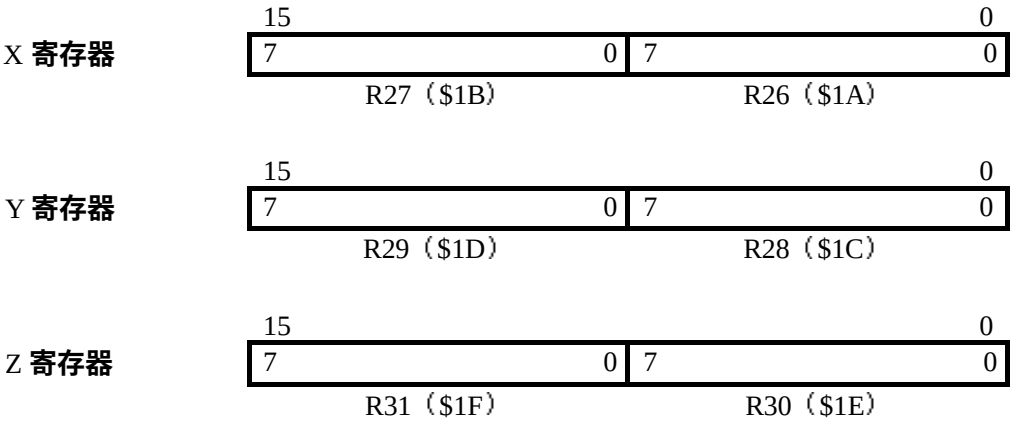
所有的寄存器操作指令都可以单指令的形式直接访问所有的寄存器。例外情况为 5 **条涉及常数操作的指令：**SBCI、SUBI、CPI、ANDI **和** ORI**。这些指令只能访问通用寄存器文件的后半部分：**R16 **到** R31**。**

如图 5 **所示，每个寄存器都有一个数据内存地址，将他们直接映射到用户数据空间的头** 32 **个地址。虽然寄存器文件的实现与** SRAM **不同，这种内存组织方式在访问寄存器方面具有极大的灵活性。**

X、Y、Z 寄存器：

寄存器 R26~R31 **除了用作通用寄存器外，还可以作为数据间接寻址用的地址指针。**

图 6　X、Y、Z 寄存器



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

ATmega603/103 具有 64K/128K 字节的 FLASH。因为所有的指令为 16 位宽，故尔 FLASH 结构为 32K×16/64K×16。FLASH 的擦除次数至少为 1000 次。

SRAM

ATmega603/103 支持两种不同的配置方法。

表 2　存储器配置

配置	内部 SRAM	外部 SRAM
A	4000	0
B	4000	最大 64K

注：外挂 64KSRAM 时，可使用 60K。

图 7　存储器配置

前 4096 个数据地址用于寻址寄存器文件，I/O 和内部数据 SRAM。起始的 96 个地址为寄存器文件+I/O，其后的 4000 个地址用于寻址 SRAM。

ATmega603/103 支持 64K 的外部 SRAM。但是要注意，低 4K 将与内部的 SRAM 相重合，只有内部的 SRAM 可用。

当访问 SRAM 的地址超出内部 SRAM 的地址时，MCU 将对外部 SRAM 寻址（指令相同）。访问内部 SRAM 时/RD 和/WR 保持无效。若要访问外部 SRAM，必须置位 MCUCR 的 SRE 位。

访问外部 SRAM 比访问内部的要多一个时钟周期，这意味着指令 LD，ST，LDS，STS，PUSH 和 POP 指令将多一个时钟周期。如果堆栈放置于外部 SRAM，则中断和函数调用将花费额外的两个时钟。如果外部 SRAM 接口使用了等待状态，则又多两个时钟周期。

数据寻址模式分为 5 种：直接，带偏移量的间接，间接，预减的间接，后加的间接。寄存器 R26 到 R31 为间接寻址的指针寄存器。

带偏移量的间接寻址模式寻址到 Y、Z 指针给定地址附近的 63 个地址。

带预减和后加的间接寻址模式要用到 X、Y、Z 指针。
32 个通用寄存器，64 个 I/O 寄存器，4000 字节的 SRAM 和最大可达 64K 的外部 SRAM 可以被所有的寻址模式访问。
程序和数据寻址模式
ATmega603/103 支持强大而有效的寻址模式。本节将要介绍各种寻址模式。
单寄存器直接寻址：

图 8

双寄存器直接寻址：

图 9

I/O 直接寻址：

图 10

数据直接寻址：

图 11

带偏移的数据间接寻址：

图 12

数据间接寻址：

图 13

带预减的数据间接寻址：

图 14

带后加的数据间接寻址：

图 15

使用 LPM、ELPM 的指令寻址常数：

图 16

直接程序寻址，JMP 和 CALL：

图 17

间接程序寻址，IJMP 和 ICALL：

图 18

相对程序寻址，RJMP 和 RCALL：

图 19

EEPROM

ATmega603/103 包含 2K/4K 字节的 EEPROM。它是作为一个独立的数据空间而存在的，可以按字节读写。EEPROM 的寿命至少为 100000 次（擦除）。EEPROM 的访问由地址寄存器，数据寄存器和控制寄存器决定。

内存访问和指令执行时序

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 20 说明了由 HARVARD 结构决定了的并行取指和执行，以及快速访问寄存器文件的概念。这是一个基本的，达到 1MIPS/MHz，具有优良的性价比、功能/时钟比、功能/功耗比的流水线概念。

图 20 并行取指与指令执行

图 21 演示的是寄存器文件内部时序。在一个时钟周期里，ALU 可以同时对两个寄存器操作数进行操作，同时将结果存回到其中的一个寄存器中去。

图 21 单时钟 ALU 操作

图 22 片内 SRAM 访问周期

I/O 内存

表 3 ATmega603/103 的 I/O 空间

地址（16 进制）	名称	功能
$3F($5F)	SREG	状态寄存器
$3E($5E)	SPH	堆栈指针高字节
$3D($5D)	SPL	堆栈指针低字节
$3C($5C)	XDIV	XTAL 分频控制寄存器
$3B($5B)	RAMPZ	RAM 页的 Z 选择寄存器
$3A($5A)	EICR	外部中断控制寄存器
$39($59)	EIMSK	外部中断屏蔽寄存器
$38($58)	EIFR	外部中断标志寄存器
$37($57)	TIMSK	T/C 中断屏蔽寄存器
$36($56)	TIFR	T/C 中断标志寄存器
$35($55)	MCUCR	MCU 控制寄存器
$34($54)	MCUSR	MCU 状态寄存器
$33($53)	TCCR0	T/C0 控制寄存器
$32($52)	TCNT0	T/C0（8 位）
$31($51)	OCR0	T/C0 输出比较寄存器
$30($50)	ASSR	异步模式状态寄存器
$2F($4F)	TCCR1A	T/C1 控制寄存器 A
$2E($4E)	TCCR1B	T/C1 控制寄存器 B
$2D($4D)	TCNT1H	T/C1 高字节
$2C($4C)	TCNT1L	T/C1 低字节
$2B($4B)	OCR1AH	T/C1 输出比较寄存器 A 高字节
$2A($4A)	OCR1AL	T/C1 输出比较寄存器 A 低字节
$29($49)	OCR1BH	T/C1 输出比较寄存器 B 高字节
$28($48)	OCR1BL	T/C1 输出比较寄存器 B 低字节
$27($47)	ICR1H	T/C1 输入捕捉寄存器高字节
$26($46)	ICR1L	T/C1 输入捕捉寄存器低字节
$25($45)	TCCR2	T/C2 控制寄存器
$24($44)	TCNT2	T/C2（8 位）
$23($43)	OCR2	T/C2 输出比较寄存器
$21($41)	WDTCR	看门狗控制寄存器
$1F($3F)	EEARH	EEPROM 高地址寄存器
$1E($3E)	EEARL	EEPROM 低地址寄存器

$1D($3D)	EEDR	EEPROM 数据寄存器
$1C($3C)	EECR	EEPROM 控制寄存器
$1B($3B)	PORTA	A 口数据寄存器
$1A($3A)	DDRA	A 口数据方向寄存器
$19($39)	PINA	A 口输入引脚
$18($38)	PORTB	B 口数据寄存器
$17($37)	DDRB	B 口数据方向寄存器
$16($36)	PINB	B 口输入引脚
$15($35)	PORTC	C 口数据寄存器
$14($34)	DDRC	C 口数据方向寄存器
$13($33)	PINC	C 口输入引脚
$12($32)	PORTD	D 口数据寄存器
$11($31)	DDRD	D 口数据方向寄存器
$10($30)	PIND	D 口输入引脚
$0F($2F)	SPDR	SPI 数据寄存器
$0E($2E)	SPSR	SPI 状态寄存器
$0D($2D)	SPCR	SPI 控制寄存器
$0C($2C)	UDR	UART 数据寄存器
$0B($2B)	USR	UART 状态寄存器
$0A($2A)	UCR	UART 控制寄存器
$09($29)	UBRR	UART 波特率寄存器
$08($28)	ACSR	模拟比较器控制及状态寄存器
$07($27)	ADMUX	ADC 多路选择寄存器
$06($26)	ADCSR	ADC 控制和状态寄存器
$05($25)	ADCH	ADC 数据寄存器高字节
$04($24)	ADCL	ADC 数据寄存器低字节
$03($23)	PORTE	E 口数据寄存器
$02($22)	DDRE	E 口数据方向寄存器
$01($21)	PINE	E 口输入引脚
$00($20)	PINF	F 口输入引脚

AVRATmega603/103 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。当使用 IN 和 OUT 指令时地址必须在$00-$3F 之间。如果要象 SRAM 一样访问 I/O 寄存器，则相应地址要加上$20。在本文档里所有 I/O 寄存器的 SRAM 地址写在括号中。

为了与后续产品兼容，保留未用的未应写“0”，而保留的 I/O 寄存器则不应访问。

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。CBI 和 SBI 指令只对$00-$1F 有效。

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（$5F）	**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。

堆栈指针 SP

BIT	15	14	13	12	11	10	9	8
$3E($5E)	**SP15**	**SP14**	**SP13**	**SP12**	**SP11**	**SP10**	**SP9**	**SP8**
$3D($5D)	**SP7**	**SP6**	**SP5**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**
	7	6	5	4	3	2	1	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	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
	0	0	0	0	0	0	0	0

堆栈指针指向位于 SRAM 的函数及中断堆栈。堆栈空间必须在调用函数或中断使能之前定义。指针必须指向高于$60 的地址。用 PUSH 指令推数据入栈时，堆栈指针将减一，而当调用函数或中断时，指针将减二。使用 POP 指令时，堆栈指针将加一，而用 RET 或 RETI 返回时，指针将加二。

内存页 Z 选择寄存器 RAMPZ

BIT	7	6	5	4	3	2	1	0
$3B（$5B）	-	-	-	-	-	-	-	**RAMPZ0**
读/写	R	R	R	R	R	R	R	R/W
初始值	0	0	0	0	0	0	0	0

RAMPZ 寄存器用于选择 Z 指针访问的是哪一个 64K 的 SRAM。由于 ATmega603/103 不支持超过 64K 的存储器，因此，RAMPZ 只用来协助 ELPM 指令决定访问哪一个程序存储器页。效果如下：

RAMPZ0 = 0：ELPM 可以访问$0000 — $7FFF（低 64K 字节）。

RAMPZ0 = 1：ELPM 可以访问$7FFF — $FFFF（高 64K 字节）。

LPM 指令不受 RAMPZ 的影响。

ATmega603 没有 RAMPZ 寄存器，也没有 ELPM 指令。因为 LPM 指令已经可以访问整个 64K 字节空间。

MCU 控制寄存器—MCUCR

BIT	7	6	5	4	3	2	1	0
$35($55)	**SRE**	**SRW**	**SE**	**SM**	-	-	-	-
读/写	R/W	R/W	R/W	R/W	R	R	R	R

初始值	0	0	0	0	0	0	0	0

SRE：外部 SRAM 使能

SRE 为“1”时，MCU 可以访问外部 SRAM，AD0-7（A 口），A8-15（C 口），/WR，/RD 引脚信号有效，且自动按照要求配置端口方向寄存器。如果 SRE 清零，外部 SRAM 无效，相关口可以当作普通 I/O 口使用。

SRW：外部 SRAM 等待状态

当 SRW 为“0”时，对外部 SRAM 的访问执行一般的 3 周期方案。若 SRW 置位，则 MCU 会自动再加入一个等待周期。

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/SM0：休眠模式

用于选择休眠模式，如表 4 所示。

表 4　休眠模式

SM1	SM0	休眠模式
0	0	空闲
0	1	保留
1	0	掉电
1	1	省电

XTAL 分频控制寄存器—XDIV

BIT	7	6	5	4	3	2	1	0
$3C($5C)	XDIVEN	XDIV6	XDIV5	XDIV4	XDIV3	XDIV2	XDIV1	XDIV0
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

XDIVEN：XTAL 分频使能

XDIVEN 为“1”时，MCU 时钟按 XDIV6-XDIV0 的值（*d*）进行分频。此位可以在运行当中进行修改。

XDIV6..XDIV0：XTAL 选择因子

分频公式为：

d 值只能在 XDIVEN 为“0”时修改才有效。由于修改的是 MCU 的主时钟，此操作将影响所有的相关外设。

复位和中断处理

ATmega603/103 有 23 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。所有的中断事件都有自己的使能位。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复位及中断向量。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

向量号	程序地址	来源	定义
1	$000	RESET	硬件管脚，上电复位和看门狗复位
2	$002	INT0	外部中断 0
3	$004	INT1	外部中断 1
4	$006	INT2	外部中断 2
5	$008	INT3	外部中断 3

6	$00A	INT4	**外部中断** 4
7	$00C	INT5	**外部中断** 5
8	$01E	INT6	**外部中断** 6
9	$010	INT7	**外部中断** 7
10	$012	TIMER2 COMP	T/C2 **比较匹配**
11	$014	TIMER2 OVF	T/C2 **溢出**
12	$016	TIMER1 CAPT	T/C1 **捕捉事件**
13	$018	TIMER1 COMPA	T/C1 **比较匹配** A
14	$01A	TIMER1 COMPB	T/C1 **比较匹配** B
15	$01C	TIMER1 OVF	T/C1 **溢出**
16	$01E	TIMER0 COMP	T/C0 **比较匹配**
17	$020	TIMER0 OVF	T/C0 **溢出**
18	$022	SPI，STC	**串行传输结束**
19	$024	UART，RX	UART **接收结束**
20	$026	UART，UDRE	UART **数据寄存器空**
21	$028	UART，TX	UART **发送结束**
22	$02A	ADC	ADC **转换结束**
23	$02C	EE READY	EEPROM **准备好**
24	$02E	ANA_COMP	**模拟比较器**

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
$002                    RJMP    EXT_INT0            ；IRQ0
$004                    RJMP    EXT_INT1            ；IRQ1
$006                    RJMP    EXT_INT2            ；IRQ2
$008                    RJMP    EXT_INT3            ；IRQ3
$00a                    RJMP    EXT_INT4            ；IRQ4
$00c                    RJMP    EXT_INT5            ；IRQ5
$00e                    RJMP    EXT_INT6            ；IRQ6
$010                    RJMP    EXT_INT7            ；IRQ7
$012                    RJMP    TIM2_COMP           ；T2 比较匹配
$014                    RJMP    TIM2_OVF            ；T2 溢出
$016                    RJMP    TIM1_CAPT           ；T1 捕捉
$018                    RJMP    TIM1_COMPA          ；T1 比较 A 匹配
$01a                    RJMP    TIM1_COMPB          ；T1 比较 B 匹配
$01c                    RJMP    TIM1_OVF            ；T1 溢出
$01e                    RJMP    TIM0_COMP           ；T0 比较匹配
$020                    RJMP    TIM0_OVF            ；T0 溢出
$022                    RJMP    SPI_STC             ；SPI 传输结束
$024                    RJMP    UART_RXC            ；UART 接收结束
$026                    RJMP    UART_DRE            ；UART 数据空
$028                    RJMP    UART_TXC            ；UART 发送结束
$02a                    RJMP    ADC                 ；AD 转换结束
$02c                    RJMP    EE_RDY              ；EEP 准备好
$02e                    RJMP    ANA_COMP            ；模拟比较器
;
```

```
$030    MAIN:   LDI     R16，HIGH(REMEND)    ；主程序开始
$022            OUT     SPH, R16
$034            LDI     R16，LOW(REMEND)
$036            RJMP
$038            <指令>   XXX
—       —       —       —
```

复位源

ATmega603/103 有 3 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。

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 JMP—绝对跳转指令—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 23 为复位电路的逻辑图。表 6 定义了复位电路的时序和电参数。

图 23　复位逻辑

表 6　复位电参数（V_{CC} = 5.0V）

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）		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}	复位引脚门限电压		-	V_{CC}/2		V
t_{TOUT}	复位延迟周期	STU = 00	11	16	21	CPU 周期
		STU = 01	0.4	0.5	0.6	ms
		STU = 10	3.2	4.0	4.8	
		STU = 11	12.8	16.0	19.2	

注：1.除非电源电压低于 V_{POT}，否则上电复位不会发生。

上电复位：

上电复位（POR）保证器件在上电时正确复位。如图 23 所示，看门狗定时器驱动一个内部定时器，此定时器保证 MCU 只有在 V_{CC} 达到 V_{POT} 且过了一定时间之后才启动。熔丝位 SUT1/0 编程后可以使 MCU 以较短的时间起动。

用户可以按照典型振荡器起振特性来选择启动时间。用于时间溢出的 WDT 振荡周期数示于表 7。看门狗振荡器的频率与工作电压有关，具体参见后续章节的典型特性。

表 7　看门狗振荡器周期数

SUT1/0	溢出时间（V_{CC}=5V）	WDT 周期数
01	0.5ms	512
10	4.0ms	4K
11	16.0ms	16K

SUT1/0 为 00 时，CPU 可以在 5 个 CPU 时钟后启动。此设置不使用 WDT 振荡器，使得 CPU 可以快速地从掉电或省电模式恢复到工作状态，如果 XTAL 是由外部直接提供给 XTAL1 的话。

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 25。

图 24　MCU 起动，/RESET 与 V_{CC} 相连

图 25　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 26　工作期间的外部复位

看门狗复位：

当看门狗定时器溢出时，将产生 1 个 XTAL 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 27　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($54)	-	-	-	-	-	-	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 8　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 9　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

中断处理：

ATmega603/103 有 2 个中断屏蔽控制寄存器 EIMSK—外部中断屏蔽寄存器和 TIMSK—T/C 中断屏蔽寄存器。其他的使能/屏蔽位位于外设控制寄存器。

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。

外部中断屏蔽寄存器—EIMSK

BIT	7	6	5	4	3	2	1	0
$39($59)	**INT7**	**INT5**	**INT5**	**INT4**	**INT3**	**INT2**	**INT1**	**INT0**
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

INT7-INT4：外部中断 7-4 请求使能

当 INTx 和 I 都为“1”时，外部引脚中断使能。外部中断控制寄存器 EICR 的中断检测控制位 ISCx1 和 ISCx0 决定是上升沿中断还是下降沿中断，或者是低电平中断。即使管脚被定义为输出，中断仍可产生。这是产生软件中断的一个途径。

INT3-INT0：外部中断 3-0 请求使能

当 INTx 和 I 都为“1”时，外部引脚中断使能。这几个中断都为低电平中断。即使管脚被定义为输出，中断仍可产生。这是产生软件中断的一个途径。只要低电平存在，中断就一直保持。

外部中断标志寄存器—EIFR

BIT	7	6	5	4	3	2	1	0
$38($58)	**INTF7**	**INTF6**	**INTF5**	**INTF4**	-	-	-	-
读/写	R/W	R/W	R/W	R/W	R	R	R	R
初始值	0	0	0	0	0	0	0	0

位 3..0：保留

INTF7-INTF4：外部中断标志 7-4

当 INTx 管脚有事件触发中断请求时，INTFx 置位（“1”）。如果 SREG 中的 I 及 EIMSK 中的 INTx 都为“1”，则 MCU 将跳转到相应的中断地址。中断例程执行后，标志被清除。另外，标志也可以通过对其写“1”来清除。

外部中断控制寄存器—EICR

BIT	7	6	5	4	3	2	1	0
$3A($5A)	**ISC71**	**ISC70**	**ISC61**	**ISC60**	**ISC51**	**ISC50**	**ISC41**	**ISC40**
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

ISCx1/0：外部中断 7-4 敏感控制位

选择 INT7-INT4 中断的边沿或电平，如下表所示：

表 10　中断 7-4 检测控制

ISCx1	ISCx0	描述
0	0	低电平中断
0	1	保留

1	0	下降沿中断
1	1	上升沿中断

注意：改变 ISCx1/ISCx0 **时，首先要禁止** INTx（**清除** EIMSK **的** INTx **位），否则可能引发不必要的中断。**

T/C 中断屏蔽寄存器—TIMSK

BIT	7	6	5	4	3	2	1	0
$37($57)	**OCIE2**	**TOIE2**	**TICIE1**	**OCIE1A**	**OCIE1B**	**TOIE1**	**OCIE0**	**TOIE0**
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

OCIE2：T/C2 输出比较匹配中断使能

当 TOIE2 **和** I **都为“**1**”时，输出比较匹配中断使能。当** T/C2 **的比较匹配发生，或** TIFR **中的** OCF2 **置位，中断例程（**$0012**）将执行。**

TOIE2：T/C2 溢出中断使能

当 TOIE2 **和** I **都为“**1**”时，**T/C2 **溢出中断使能。当** T/C2 **溢出，或** TIFR **中的** TOV2 **位置位时，中断例程（**$0014**）得到执行。**

TICIE1：T/C1 输入捕捉中断使能

当 TICIE1 **和** I **都为“**1**”时，输入捕捉中断使能。当** T/C1 **的输入捕捉事件发生（**ICP**），或** TIFR **中的** ICF1 **置位，中断例程（**$0016**）将执行。**

OCIE1A：T/C1 输出比较 A 匹配中断使能

当 TOIE1A **和** I **都为“**1**”时，输出比较** A **匹配中断使能。当** T/C1 **的比较** A **匹配发生，或** TIFR **中的** OCF1A **置位，中断例程（**$0018**）将执行。**

OCIE1B：T/C1 输出比较 B 匹配中断使能

当 TOIE1B **和** I **都为“**1**”时，输出比较** B **匹配中断使能。当** T/C1 **的比较** B **匹配发生，或** TIFR **中的** OCF1B **置位，中断例程（**$001A**）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（**$001C**）得到执行。**

OCIE0：T/C0 输出比较匹配中断使能

当 TOIE0 **和** I **都为“**1**”时，输出比较匹配中断使能。当** T/C0 **的比较匹配发生，或** TIFR **中的** OCF0 **置位，中断例程（**$001E**）将执行。**

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（**$020**）得到执行。**

T/C 中断标志寄存器—TIFR

BIT	7	6	5	4	3	2	1	0
$36($56)	**OCF2**	**TOV2**	**ICF1**	**OCF1A**	**OCF1B**	**TOV1**	**OCF0**	**TOV0**
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

OCF2：T/C2 输出比较标志

当 T/C2 **与** OCR2 **的值匹配时，**OCF2 **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**、**OCIE2 **和** OCF2 **一同置位时，中断例程得到执行。**

TOV2：T/C2 溢出中断标志位

当 T/C2 **溢出时，**TOV2 **置位。执行相应的中断例程后此位硬件清零。此外，**TOV2 **也可以**

通过写“1”来清零。当 SREG 中的位 I、TOIE2 和 TOV2 一同置位时，中断例程得到执行。在 PWM 模式中，当 T/C2 在$0000 改变记数方向时，TOV2 置位。

ICF1：输入捕捉标志位

当输入捕捉事件发生时，ICF1 置位，表明 T/C1 的值已经送到输入捕捉寄存器 ICR1。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、TICIE1A 和 ICF1 一同置位时，中断例程得到执行。

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。

OCF1B：输出比较标志 1B

当 T/C1 与 OCR1B 的值匹配时，OCF1B 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、OCIE1B 和 OCF1B 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。在 PWM 模式中，当 T/C1 在$0000 改变记数方向时，TOV1 置位。

OCF0：输出比较标志

当 T/C0 与 OCR0 的值匹配时，OCF0 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、OCIE0 和 OCF0 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。

中断响应时间

AVR 中断响应时间最少为 5 个时钟周期。在这 5 个时钟期间，PC（2 个字节）自动入栈，而 SP 减 2。在通常情况下，中断向量为一个跳转指令，此跳转要花 3 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。

休眠模式

进入休眠模式的条件是 SE 为“1”，然后执行 SLEEP 指令。具体哪一种模式由 SM1/0 决定。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 SM1/0 全为“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 SPI、UART、模拟比较器、ADC、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。如果 MCU 从闲置模式唤醒，CPU 将立即执行指令。

掉电模式：

当 SM1/0 为“10”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、看门狗复位和外部电平中断可以使 MCU 脱离掉电模式。

当使用外部电平中断方式将 MCU 从掉电模式唤醒时，必须保持外部电平一定的时间。这样可以可以减少 MCU 对噪声的敏感。看门狗振荡时钟要对此电平采样两次。如果电平为要求的电平，则 MCU 唤醒。标称的看门狗振荡器为 1μs（25℃），与电源电压有一定的关系。

从施加掉电唤醒条件到真正唤醒有一个延迟时间，此时间用于晶振重新启动并稳定下来。唤醒周期与复位周期是一样的。

唤醒条件必须保持到 MCU 真正醒过来，否则 MCU 又会回到掉电模式。

省电模式：

当 SM1/0 为“11”时，SLEEP 指令将使 MCU 进入省电模式。这一模式与掉电模式只有一点不同：

如果 T/C2 异步驱动，即 ASSR 的 AS0 置位，则 T/C2 在休眠时继续运行。除了掉电模式的唤醒方式，T/C2 的溢出中断和比较匹配中断也可以将 MCU 从休眠方式唤醒。为了保证唤醒后执行中断例程，必须置位全局中断使能位 I。

在省电模式下，如果由外部电平中断唤醒，则在中断标志更新之前 MCU 要执行两个周期；而若从 T/C2 唤醒则要执行 3 条指令后个中断标志才更新。在此期间，处理器执行指令，但是中断条件不可读，中断例程也不执行。

定时器/计数器

ATmega603/103 内部有 3 个通用定时器/计数器：两个 8 位 T/C 和一个 16 位 T/C。T/C0 可以选择异步外部时钟。这个振荡器对 32.768KHz 的晶体进行了优化，使其可用作实时时钟（RTC）。T/C0 具有自己的分频器，而 T/C1 和 T/C2 从同一个 10 位的预分频定时器取得预分频的时钟。T/C 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/C 的预分频器

图 28　T/C1 和 T/C2 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。还可以选择 CK、外部时钟，以及停止工作。

图 29　T/C0 的预分频器

T/C0 的时钟源称为 PCK0。缺省地，PCK0 与系统主时钟连接。若置位 ASSR 的 AS0，T/C0 将由 TOSC1 异步驱动，使得 T/C0 可以作为一个实时时钟。如果 AS0 置位，则 TOSC1 和 TOSC2 即可外接一个时钟晶振，作为 T/C0 的独立时钟源。

8 位 T/C0 和 T/C2

图 30 为 T/C0 的框图。

图 30　T/C0 工作框图

图 31　T/C2 工作框图

T/C0 的时钟可以选择 PCK0 或预分频的 PCK0。T/C2 的时钟可以选择 CK、预分频的 CK 或由外部引脚输入。另外还可以由 T/Cx 控制寄存器 TCCRx 来停止它。

TIFR 为状态标志寄存器，TCCR 为控制寄存器，而 TIMSK 控制 T/C 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/C 具有高分辨率和高精度的特点；而在高预分频条件下，T/C 非常适用于低速功能，如计时。
两个 T/C 都支持输出比较功能，由 OCR0 和 OCR2 控制。此功能包括匹配发生时清除计数器，以及操作 PB4（OC0/PWM0）和 PB7（PC2/PWM2）。
T/C0 及 T/C2 还支持 8 位的 PWM。

T/C0 控制寄存器—TCCR0

BIT	7	6	5	4	3	2	1	0
$33($53)	-	**PWM0**	**COM01**	**COM00**	**CTC0**	**CS02**	**CS01**	**CS00**
读/写	R	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/C2 控制寄存器—TCCR2

BIT	7	6	5	4	3	2	1	0
$25($45)	-	**PWM2**	**COM21**	**COM20**	**CTC2**	**CS22**	**CS21**	**CS20**
读/写	R	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

位 7：保留

PWM0/PWM2：PWM 使能
置位后即使能 T/C0 和 T/C2 的 PWM 功能

COM01，COM00/COM21，COM20：比较输出模式
COMn1 和 COMn0 决定 T/C 比较匹配发生时输出引脚 PB4（OC0/PWM0）和 PB7（PC2/PWM2）的动作。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 11。

表 11　比较模式选择

COMn1	COMn0	描述
0	0	T/Cn 与输出引脚 Ocn 断开
0	1	Ocn 输出变换
1	0	清除 Ocn
1	1	置位 Ocn

n = 0 或 2

CTC0/CTC2：比较匹配时清除 T/C0，T/C2
CTCn 为“1”时，比较匹配事件发生后，T/Cn 将复位为 0。若 CTCn 为“0”，则 T/Cn 将连续记数而不受比较匹配的影响。由于比较匹配事件的检测发生在匹配发生之后的一个 CPU 时钟，故而定时器的预分频比率的不同将引起此功能有不同的表现。当预分频为 1，比较匹配寄存器的值设置为 C 时，定时器的记数方式为：

..|.C-2 | C-1 | C | 0 | 1 |…

而当预分频为 8 时，定时器的记数方式则为：

..| C-2, C-2, C-2, C-2, C-2, C-2, C-2, C-2 | C-1, C-1, C-1, C-1, C-1, C-1, C-1, C-1 | C, 0, 0, 0, 0, 0, 0, 0 |…

在 PWM 模式下，这几位没有作用。

CS02、CS01、CS00/ CS22、CS21、CS20：时钟选择

表 12　T/C0 预分频选择

CS02	CS01	CS00	描述
0	0	0	停止
0	0	1	PCK0
0	1	0	PCK0/8
0	1	1	PCK0/32
1	0	0	PCK0/64
1	0	1	PCK0/128
1	1	0	PCK0/256
1	1	1	PCK0/1024

表 13 T/C2 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 T2，下降沿
1	1	1	外部引脚 T2，上升沿

停止条件提供了一个定时器使能/禁止的功能。预分频的 CK 直接由时钟振荡器分频而来。如果 T/C2 使用了外部引脚模式（[CS22，CS21] = [1，1]），则不论引脚配置如何，PD7（T2）的变化都将使 T/C2 变化。

T/C0—TCNT0

BIT	7	6	5	4	3	2	1	0
$32($52)	**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/C2—TCNT2

BIT	7	6	5	4	3	2	1	0
$22($42)	**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/Cn 是可以进行读/写访问的向上计数器。只要有时钟输入，T/Cn 就会在写入的值基础上继续记数。

T/C0 输出比较寄存器—OCR0

BIT	7	6	5	4	3	2	1	0
$31($51)	**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/C2 输出比较寄存器—OCR2

BIT	7	6	5	4	3	2	1	0
$23($43)	**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/Cn 输出比较寄存器包含与 T/Cn 值连续比较的数据。如果 T/Cn 的值与 OCRn 相等，则比

较匹配发生。用软件写操作将 TCNTn 和 OCRn 设置为相等不会引发比较匹配。
比较匹配发生后将置位相应的中断标志。

PWM 模式下的 T/C0 和 T/C2：

选择 PWM 模式后，T/C0 和 T/C2 和输出比较寄存器 OCR0/OCR2 分别组成 8 位无尖峰的、自由运行的、相位可变的 PWM，输出引脚为 PB4（OC0/PWM0）和 PB7（OC2/PWM2）。T/Cn 作为上/下计数器，从 0 记数到$FF，然后反向记数回到 0。当计数器中的数值和 OCR0/OCR2 的数值一致时，OC0/OC2 引脚按照 COM00/COM01 和 COM20/COM21 的设置动作。

表 14　PWM 模式下的比较模式选择

COMn1	COMn0	比较/PWM 引脚
0	0	不用作 PWM 功能
0	1	不用作 PWM 功能
1	0	向上记数时的匹配清除 Ocn；而向下记数时的匹配置位 Ocn（正向 PWM）
1	1	向下记数时的匹配清除 OCn；而向上记数时的匹配置位 Ocn（反向 PWM）

n = 0 或 2

注意：　在 PWM 模式下，OCR0/OCR2 首先存储在一个临时的位置，等到达到$FF 时才真正存入。这样可以防止在写 OCR0/OCR2 时由于失步而出现奇数长度的 PWM 脉冲。

图 32　失步的 OCR 锁存

如果在执行写和锁存操作的时候读取 OCR0/OCR2，读到的是临时位置的数据。
OCR0/OCR2 的值为$00 或$FF 时 OCn 的输出见表 15。

表 15　OCRn=$00 或$FF 时的 PWM 输出

COMn1	COMn0	OCRn	OCn
1	0	$00	L
1	0	$FF	H
1	1	$00	H
1	1	$FF	L

n = 0 或 2

在 PWM 模式下，当计数器达到$00 时将置位 TOV0/TOV2。此时发生的中断与正常情况下的中断是完全一样的。

PWM 的频率为时钟除以 510。

异步状态寄存器—ASSR

BIT	7	6	5	4	3	2	1	0
$30($50)	-	-	-	-	**AS0**	**TCN0UB**	**OCR0UB**	**TCR0UB**
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

AS0：异步 T/C0

当 AS0 置位时，T/C0 由 TOSC1 驱动。PC6 和 PC7 连接到晶体振荡器，不能用作普通 I/O。若 AS0 为“0”，则 T/C0 由内部系统时钟驱动。这一位变化时有可能破坏 TCNT0。

TCN0UB：T/C0 更新忙

T/C0 工作于异步模式时，写 TCNT0 将引起 TCN0UB 置位。当 TCNT0 从暂存寄存器更新完毕后 TCN0UB 由硬件清零。TCN0UB 为“0”表明 TCNT0 可以写入新值了。

OCR0UB：输出比较寄存器更新忙

T/C0 工作于异步模式时，写 OCR0 将引起 OCR0UB 置位。当 OCR0 从暂存寄存器更新完毕后 OCR0UB 由硬件清零。OCR0UB 为“0”表明 OCR0 可以写入新值了。

TCR0UB：T/C0 控制寄存器更新忙

T/C0 工作于异步模式时，写 TCCR0 将引起 TCR0UB 置位。当 TCCR0 从暂存寄存器更新完毕后 TCR0UB 由硬件清零。TCR0UB 为“0”表明 TCCR0 可以写入新值了。

如果在更新忙标志置位的时候写上述任何一个寄存器都将引起数据的破坏，并引发不必要的中断。

对 TCNT0，OCR0 和 TCCR0 进行读取的机制是不同的。读到的 TCNT0 为实际的值，而 OCR0 和 TCCR0 则是从暂存寄存器中读取的。

T/C0 的异步操作

T/C0 异步工作时要考虑如下几点。

- 警告：在同步和异步模式之间的转换有可能造成 TCNT0、OCR0、TCCR0 数据的损毁。安全的步骤应该是：
  1、关闭 T/C0 的中断 OCIE0 和 TOIE0。
  2、设置 AS0 以选择合适的时钟源。
  3、TCNT0，OCR0 和 TCCR0 写入新的数值。
  4、等待 TCN0UB，OCR0UB 和 TCR0UB 清零。
  5、必要的话，开启中断。
- 振荡器对 32768Hz 的晶振进行了优化，其对外部输入时钟信号的带宽为 256kHz。因此对外部输入的时钟信号不能高于 256kHz。另外，此信号还不能高于系统主时钟的 1/4。
- 写 TCNT0，OCR0 和 TCCR0 时数据首先传到暂存寄存器，两个 TOSC1 正跳变后才锁存。用户在数据从暂存寄存器写入目的寄存器之前不能写入新的数值。3 个寄存器具有各自独立的暂存寄存器，因此写 TCNT0 不会干扰写 OCR0。可以通过 ASSR 检查数据是否已经写入到目的寄存器。
- 如果要用 T/C0 作为 MCU 的唤醒条件，则在 TCNT0，OCR0 和 TCCR0 更新结束之前不能进入省电模式，否则 MCU 可能会在 T/C0 设置生效之前进入休眠貌模式。这对于用 T/C0 的比较匹配中断唤醒 MCU 尤其重要。因为在更新 OCR0 或 TCNT0 时比较匹配是禁止的。如果在更新过程中 MCU 进入休眠模式，则比较匹配中断永远不会发生。
- 如果要用 T/C0 作为省电模式的唤醒条件，必须注意重新进入省电模式的过程。中断逻辑需要一个 TOSC1 周期进行复位。如果从唤醒到重新进入休眠的时间小于一个 TOSC1 周期，中断将不再发生，器件再也无法唤醒。如果用户怀疑自己程序是否满足这一条件，则可以采取如下方法：
  1、对 TCNT0，OCR0 和 TCCR0 写入一个合适的值
  2、等待更新忙标志变低
  3、进入省电模式
- 若选择了异步工作模式，T/C0 的振荡器将一直工作，除非进入掉电模式。用户应该注意，此振荡器的稳定时间可能长达 1 秒钟。因此，唤醒后最好认为建议用户在器件从掉电模式唤醒或上电时至少等待 1 秒钟后再使用 T/C0。
- 省电模式唤醒过程：中断条件满足后，在下一个定时器时钟里唤醒过程启动。在 MCU 时钟起动后的 3 个周期，中断标志置位。在此期间，MCU 执行其他指令，但中断条件还不可读，中断例程也不会执行。
- 在异步模式下，中断标志的同步需要 3 个处理器周期加一个定时器周期。输出比较引脚的变化与定时器时钟同步，而不是处理器时钟。

16 位 T/C1

图 33 为 T/C1 的框图。

图 33　T/C1 工作框图

T/C1 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/C1 控制寄存器 TCCR1A 和 TCCR1B 来停止它。TIFR 为状态标志寄存器，而 TIMSK 控制 T/C1 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/C1 具有高分辨率和高精度的特点；而在高预分频条件下，T/C1 非常适用于低速功能，如计时。
利用输出比较寄存器 OCR1A/OCR1B 作为数据源，T/C1 还可以实现输出比较的功能。此功能包括比较匹配 A 发生时清除计数器和比较输出引脚的动作。
T/C1 还可以用作 8，9 或 10 位的 PWM 调制器。在此模式下，计数器和 OCR1A/OCR1B 寄存器用于两个无尖峰干扰的中心对称的 PWM。
当输入捕捉引脚 ICP 发生相应事件时，T/C1 的值将被传到输入捕捉寄存器 ICR1。捕捉事件的设置由 TCCR1B 控制。此外，模拟比较器也可以设置为触发输入捕捉。ICP 引脚逻辑见图 32。

图 32　ICP 引脚原理图

如果噪声抑制功能使能，则触发信号要进行 4 次采样。只有当 4 个采样值都相等时，才会触发捕捉标志。

T/C1 控制寄存器 A—TCCR1A

BIT	7	6	5	4	3	2	1	0
$2F($4F)	COM1A1	COM1A0	COM1B1	COM1B0	-	-	PWM11	PWM10
读/写	R/W	R/W	R/W	R/W	R	R	R/W	R/W
初始值	0	0	0	0	0	0	0	0

位 3、2：保留

COM1A1，COM1A0：比较输出模式 1A，位 1 和 0

COM1A1 和 COM1A0 决定 T/C1 的比较匹配发生时输出引脚 OC1A 的动作。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 12。

COM1B1，COM1B0：比较输出模式 1B，位 1 和 0

COM1B1 和 COM1B0 决定 T/C1 的比较匹配发生时输出引脚 OC1B 的动作。

表 16　比较 1 模式选择

COM1X1	COM1X0	描述
0	0	T/C1 与输出引脚 OC1X 断开
0	1	OC1X 输出变换
1	0	清除 OC1X
1	1	置位 OC1X

X = A 或 B

在 PWM 模式，这些位具有不同的功能。细节见表 17。

PWM11，PWM10：PWM 选择

表 17　PWM 模式选择

PWM11	PWM10	描述

0	0	T/C1 的 PWM 操作无效
0	1	T/C1 为 8 位 PWM
1	0	T/C1 为 9 位 PWM
1	1	T/C1 为 10 位 PWM

T/C1 控制寄存器 B—TCCR1B

BIT	7	6	5	4	3	2	1	0
$2E($4E)	**ICNC1**	**ICES1**	**-**	**-**	**CTC1**	**CS12**	**CS11**	**CS10**
读/写	R/W	R/W	R	R	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

位 5、4：保留

ICNC1：输入捕捉抑制器（4 个时钟）

ICNC1 **高有效。输入捕捉在** ICP-**输入捕捉引脚的第一个上升/下降边沿触发。当** ICNC1 **为“**1**”时，**ICP **信号要进行** 4 **次连续采样，只有** 4 **个采样值都有效时输入捕捉标志才置位。采样频率为** XTAL **时钟。**

ICES1：输入捕捉 1 边沿选择

当 ICES1 **位为“**0**”时，**T/C1 **的值在** ICP **引脚电平的下降沿被传送到输入捕捉寄存器** ICR1**。若** ICES1 **位为“**1**”，则** T/C1 **的值在** ICP **引脚电平的上升沿被传送到** ICR1**。**

CTC1：比较匹配时清除 T/C1

CTC1 **为“**1**”时，比较** A **匹配事件发生后，**T/C1 **将复位为** 0**。若** CTC1 **为“**0**”，则** T/C1 **将连续记数而不受比较匹配的影响。由于比较匹配事件的检测发生在匹配发生之后的一个** CPU **时钟，故而定时器的预分频比率的不同将引起此功能有不同的表现。当预分频为** 1**，**A **比较匹配寄存器的值设置为** C **时，定时器的记数方式为：**

..|.C-2 | C-1 | C | 0 | 1 |…

而当预分频为 8 **时，定时器的记数方式则为：**

..| C-2, C-2, C-2, C-2, C-2, C-2, C-2, C-2 | C-1, C-1, C-1, C-1, C-1, C-1, C-1, C-1 | C, 0, 0, 0, 0, 0, 0, 0 |…

在 PWM **模式下，这几位没有作用。**

CS12、CS11、CS10：时钟选择

表 14 T/C1 **预分频选择**

CS12	CS11	CS10	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** T1**，下降沿**
1	1	1	**外部引脚** T1**，上升沿**

T/C1—TCNT1H 和 TCNT1L

BIT	15	14	13	12	11	10	9	8
$2D($4D)	**MSB**							
$2C($4C)								**LSB**
	7	6	5	4	3	2	1	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	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
	0	0	0	0	0	0	0	0

此 16 位寄存器包含了 T/C1 的值。当 CPU 访问这两个寄存器时，为了保证高字节和低字节能够同时读写，要用到一个 8 位的临时寄存器 TEMP。此寄存器在访问 OCR1A、OCR1B 和 ICR1 的时候也要用到。如果主程序和中断程序在访问寄存器时都要用到 TEMP，那么在适当的时候需要关闭中断使能防止出错。

- 写 TCNT1：
  当 CPU 写 TCNT1H 时，数据将被放置在 TEMP 寄存器。当 CPU 写低字节 TCNT1L 时，此数据及 TEMP 中的数据一并写入 TCNT1。因此，在写 TCNT1（16 位）时，首先要写 TCNT1H。
- 读 TCNT1：
  当 CPU 读取 TCNT1L 时，TCNT1L 的数据将送入 CPU，同时，TCNT1H 将送入 TEMP 寄存器。等到 CPU 读取 TCNT1H 时，TEMP 中的数据送入 CPU。因此，在读 16 位的 TCNT1 时，首先要读 TCNT1L。

T/C1 是向上记数器或上/下记数器（在 PWM 模式下）。若 T/C1 被置数，则 T/C1 将在预置数的基础上记数。

T/C1 输出比较寄存器—OCR1AH 和 OCR1AL

BIT	15	14	13	12	11	10	9	8
$2B($4B)	**MSB**							
$2A($4A)								**LSB**
	7	6	5	4	3	2	1	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	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
	0	0	0	0	0	0	0	0

T/C1 输出比较寄存器—OCR1BH 和 OCR1BL

BIT	15	14	13	12	11	10	9	8
$29($49)	**MSB**							
$28($48)								**LSB**
	7	6	5	4	3	2	1	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	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
	0	0	0	0	0	0	0	0

T/C1 输出比较寄存器包含与 T/C1 值连续比较的数据。如果 T/C1 的值与 OCR 相等，则比较匹配发生。用软件写操作将 TCNT1 和 OCR1A 或 OCR1B 设置为相等不会引发比较匹配。置位 PORTD5 和 PORTD4 将置位相应的 OC1A 和 OC1B。

由于 OCR1A/OCR1B 为 16 位寄存器，所以在访问时要用到 TEMP 寄存器以保证两个字节的同步更新。其读写过程与读写 TCNT1 相同。

访问 TCNT1 和 ICR1 同样要用到 TEMP 寄存器。如果主程序和中断例程都要用到 TEMP，则在主程序访问这些寄存器时要禁止中断。

T/C1 输入捕捉寄存器—ICR1H 和 ICR1L

BIT	15	14	13	12	11	10	9	8
$27($47)	**MSB**							
$26($46)								**LSB**
	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

按照 ICES1 的设定，输入捕捉引脚 ICP 发生上跳变或下跳变时，TCNT1 被送入 ICR1。同时，ICF1 置位。

由于 ICR1 为 16 位寄存器，所以在访问时要用到 TEMP 寄存器以保证同时读取两个字节。读写过程与读写 TCNT1 相同。

PWM 模式下的 T/C1：

选择 PWM 模式后，T/C1 和输出比较寄存器 OCR1A/OCR1B 共同组成两个 8，9 或 10 位的无尖峰的自由运行的 PWM。T/C1 作为上/下计数器，从 0 记数到 TOP，然后反向记数回到 0。当计数器中的数值和 OCR1A/OCR1B 的数值（低 8，9 或 10 位）一致时，OC1A/OC1B 引脚按照 COM1A0/COM1A1 和 COM1B0/COM1B1 的设置动作。

表 19　TOP 值及 PWM 频率

PWM 分辨率	TOP 的值	频率
8 位	$00FF（255）	$F_{TC1}/510$
9 位	$01FF（511）	$F_{TC1}/1022$
10 位	$03FF（1023）	$F_{TC1}/2046$

表 16　PWM 模式下的比较 1 模式选择

COM1X1	COM1X0	OC1
0	0	不用作 PWM 功能
0	1	不用作 PWM 功能
1	0	向上记数时的匹配清除 OC1；而向下记数时的匹配置位 OC1（正向 PWM）
1	1	向下记数时的匹配清除 OC1；而向上记数时的匹配置位 OC1（反向 PWM）

X = A 或 B

注意： 在 PWM 模式下，OCR1A/OCR1B 的低 10 位首先存储在一个临时的位置，等到 T/C1 达到 TOP 时才真正存入 OCR1A/OCR1B。这样可以防止在写 OCR1A/OCR1B 时由于失步而出现奇数长度的 PWM 脉冲。

图 33　失步的 OCR1 锁存

如果在执行写和锁存操作的时候读取 OCR1A/OCR1B，读到的是临时位置的数据。

OCR1 的值为$0000 或 TOP 时 OC1 的输出见表 13。

表 21　OCR1X=$0000 或 TOP 时的 PWM 输出

COM1X1	COM1X0	OCR1X	OC1X
1	0	$0000	L
1	0	TOP	H
1	1	$0000	H
1	1	TOP	L

X = A 或 B

在 PWM 模式下，当计数器达到$0000 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。

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。ATmega603/103 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 36　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($41)	-	-	-	**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 22　　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}=3V	典型溢出时间 V_{CC}=5V
0	0	0	16K	47ms	15ms
0	0	1	32K	94ms	30ms
0	1	0	64K	0.19s	60ms
0	1	1	128K	0.38s	0.12s
1	0	0	256K	0.75s	0.24s
1	0	1	512K	1.5s	0.49s
1	1	0	1024K	3.0s	0.97s
1	1	1	2048K	6.0s	1.9s

注意：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。

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。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—EEARH 和 EEARL

BIT	15	14	13	12	11	10	9	8
$1F($3F)	-	-	-	-	**EEAR1**	**EEAR1**	**EEAR9**	**EEAR8**

					1	0		
$1E($3E)	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0
	7	6	5	4	3	2	1	0
读/写	R	R	R	R	R/W	R/W	R/W	R/W
	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	X
	X	X	X	X	X	X	X	X

EEARH 和 EEARL 指定 ATmega603/103 的 2K/4K 字节的 EEPROM。对于 ATmega603 则没有 EEAR11。EEPROM 的地址是线性的，从 0 到 2047/4095。

EEPROM数据寄存器—EEDR

BIT	7	6	5	4	3	2	1	0
$1D($3D)	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	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

位 7.3：保留

EERIE：EEPROM准备好中断使能

当 I 和 EERIE 置位时，EEPROM 准备好中断使能。EEWE 为“0”时，EEPROM 准备好中断将产生（constant）中断。

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

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。

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。

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。

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。
用户在读取 EEP 时应该检测 EEWE。如果一个写操作正在进行，写 EEAR 和 EEDR 将中断 EEP 的写入，使得结果无法预测。

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。

串行外设接口—SPI

串行外设接口 SPI 允许 ATmega603/103 和外设之间进行高速的同步数据传输。ATmega603/103 SPI 的特点如下：

- 全双工，3 线同步数据传输
- 主从操作
- LSB 在先或 MSB 在先
- 4 种可编程的比特率
- 传输结束中断
- 写碰撞标志检测
- 可以从闲置模式唤醒（作为从机工作时）

图 37 SPI 方框图

主从 CPU 的 SPI 连接见图 38。PB1（SCK）为主机的时钟输出，从机的时钟输出。把数据写入主机 SPI 数据寄存器的操作将启动 SPI 时钟产生器，数据从主机的 PB2（MOSI）移出，从从机的 PB2（MOSI）移入。移完一个字节后，SPI 时钟停止，并设置发送结束标志。此时如果 SPCR 的 SPIE（SPI 中断使能）置位，则引发中断。若要选择某器件为从机，需要将从机选择输入，PB4（/SS），拉低。主从机的移位寄存器可以看成一个分布式的 16 位循环移位寄存器。当数据从主机移向从机的同时，数据也从从机移向主机。这说明在移位过程当中，主从机进行了数据交换。

图 38 SPI 主从连接

SPI 在发送方向有一个缓冲器，而在接收方向有两个缓冲器。这意味着在移位周期没有完全结束之前，新的数据不能写到 SPI 数据寄存器。而在新的数据完全移入 SPI 数据寄存器之前，旧的数据必须读出。
当 SPI 功能使能后，MOSI、MISO、SCK 和/SS 引脚被自动配置成如下：

表 16　SPI 引脚配置

引脚	方向（主机）	方向（从机）
MOSI	用户定义	输入
MISO	输入	用户定义
SCK	用户定义	输入
/SS	用户定义	输入

/SS 引脚功能

当 SPI 配置为主机时（SPCR 的 MSTR 置位），用户可以决定/SS 引脚的方向。若/SS 配置为输出，则此引脚可以用作普通的 I/O 口而不影响 SPI。如果/SS 配置为输入，则/SS 必须保持为高以保证 SPI 的正常工作。若系统配置为主机，/SS 为输入，但被外设拉低，则 SPI 控制器会将此低电平解释为有一个外部主机将自己选择为从机。为了防止总线冲突，SPI 系统遵循以下规则：

1、如果 SPCR 的 MSTR 位为“0”，则 SPI 为从机，MOSI 和 SCK 为输入。

2、如果 SPSR 的 SPIF 置位，且 SPI 中断和全局中断开放，则中断例程将得到执行。

因此，在 SPI 主机使用中断方式进行数据发送时，/SS 有可能被拉低。中断例程必须要检测 MSTR 置位。如果检测到 MSTR 被清零，用户必须置位 MSTR，以便重新进入主机模式。

如果 SPI 配置为从机，则/SS 一直为输入。当/SS 为低时，SPI 功能激活，MISO 成为输出引脚而其他成为输入。如果/SS 为高，则所有相关引脚都为输入，SPI 不接收任何数据。要注意的是若/SS 拉高，SPI 逻辑将复位。如果在发送过程中/SS 被拉高，则数据传输马上中断，数据丢失。

数据模式

SCK 的相位、极性与数据间有 4 种组合。CPHA 和 CPOL 控制组合的方式。SPI 数据传输格式见图 39 和 40。

图 39　SPI 传输格式（CPHA=0，DORD=0）

图 40　SPI 传输格式（CPHA=1，DORD=0）

SPI 控制寄存器—SPCR

BIT	7	6	5	4	3	2	1	0
$0D($2D)	**SPIE**	**SPE**	**DORD**	**MSTR**	**CPOL**	**CPHA**	**SPR1**	**SPR0**
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

SPIE：SPI 中断使能

SPE：SPI 使能

DORD：数据次序

DORD 为“1”，LSB 先发送；DORD 为“0”，MSB 先发送。

MSTR：主从选择

MSTR 置位时选择主机模式，否则为从机。如果 MSTR 为“1”，/SS 为输入，但被拉低，则 MSTR 被清零，SPIF 置位。用户必须重新设置 MSTR 进入主机模式。

CPOL：时钟极性

CPOL 置位表明总线空闲时 SCK 为高。

CPHA：时钟相位

参见图 39 和 40。

SPR1，SPR0：SPI 时钟速率选择位

表 24　SCK 和时钟频率之间的关系

SPR1	SPR0	SCK
0	0	$f_{CL}/4$
0	1	$f_{CL}/16$
1	0	$f_{CL}/64$
1	1	$f_{CL}/128$

SPI 状态寄存器—SPSR

BIT	7	6	5	4	3	2	1	0
$0E($2E)	**SPIF**	**WCOL**	-	-	-	-	-	-
读/写	R/W	R/W	R	R	R	R	R	R
初始值	0	0	0	0	0	0	0	0

位 5..0：保留

SPIF：SPI 中断标志

串行发送结束后，SPIF **置位，即使此时/SS 被拉低（作为输入口）。进入中断例程后** SPIF **自动复位。或者可以通过先读** SPSR，**紧接着读** SPDR **来对** SPIF **清零。**

WCOL：写碰撞标志

在 SPI **发送当中对** SPI **数据寄存器** SPDR **写数据将置位** WCOL。WCOL **可以通过先读** SPSR，**紧接着读** SPDR **来清零。**

SPI 数据寄存器—SPDR

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

UART

ATmega603/103 **具有全双工通用异步收发器。其主要特点为：**

- **波特率发生器可以产生大量的波特率（**bps）
- **在低时钟下仍然可以得到高的波特率**
- 8 **或** 9 **位数据**
- **噪声滤波**
- **过速检测**
- **帧错误检测**
- **错误起始位检测**
- 3 **个独立的中断：发送结束，发送数据寄存器空，接收结束**

数据发送

图 41 **为** UART **发送器的原理图。**

把待发送的数据写入 UART **数据寄存器** UDR **将起动数据发送。在如下情况下** UDR **的数据进入发送移位寄存器：**

- **若前一个字符的停止位已经移出移位寄存器，则** UDR **的数据立即送入移位寄存器。**
- **若前一个字符的停止位还没有移出移位寄存器，则要等到停止位移出后，**UDR **的数据才送入移位寄存器。**

图 41　UART **发送器**

如果 10（11）位收发器移位寄存器为空，UDR 中的数据将传送到移位寄存器。此时 UDRE 置位，表明 UART 可以接收下一个数据。当数据送入移位寄存器的时候，移位寄存器的 0 位（起始位）自动清零，而位 9 和 10（停止位）置位。如果选择了 9 位数据格式（UART 的 CHR9 置位），则 UCR 中的 TXB8 将送到移位寄存器的位 9。

UART 首先在 TXD 引脚送出起始位，然后是数据，低位在前。如果 UDR 里有新数据，则 UART 会在停止位发送完毕只有自动加载数据。在加载数据的同时，UDRE 置位，并一直保持到有新数据写入 UDR。如果 UDR 没有新数据，而且停止位也已经输出一个 bit 的长度，则发送结束标志 TXC 置位。

UCR 的 TXEN 使能 UART 发送器。当 TXEN 为“0”时，PD1 可用作普通 I/O 口。当 TXEN 为“1”时，UART 输出自动连接到 PD1，强迫其为输出，而不管方向寄存器的设置。

数据接收

图 42　UART 接收器

接收器的前端以 16 倍于波特率的频率对 RXD 引脚进行采样。如果在此管脚处于空闲状态的时候检测到低电平，则认为这是起始位的下降沿，起始位检测序列开始。假定采样 1 为第一次检测到低电平的时刻。CPU 会在采样 8，9 和 10 对 RXD 进行 3 次连续采样。如果有两个或全部采样值为高，则认为当前信号是一个虚假的起始位，要丢弃。MCU 将开始等待新一次的 1 到 0 的转换。

如果检测到了一个有效的起始位，MCU 就会开始采样数据。数据位的检测同样是在采样 8，9 和 10。两个或 3 个相同的值被认为是当前位的值。图 43 说明了采样过程。

图 43　采样接收到的数据

当停止位进入接收器时，3 个采样值当中必须要有两个以上为高。否则，USR 中的帧错误标志位 FE 置位。在读 UDR 之前，用户应该检查 FE。

不管停止位有效与否，接收到的数据都将被送入 UDR，USR 的 RXC 置位。UDR 实际上是两个物理上分离的寄存器，一个用于发送，一个用于接收。读取 UDR 时，访问的是接收 UDR，而在写 UDR 时，访问的是发送 UDR。如果数据格式为 9 位，则 UCR 的 RXB8 在数据传送到 UDR 时加载到发送移位寄存器的位 9。

如果在读取 UDR 之前，UART 又接收到一个数据，则 USR 的 OR 置位。这表明数据无法转移到 UDR 而丢失了。OR 一直保持到 UDR 被读取。因此，如果波特率比较高，或者 CPU 负载比较重，用户应该在读 UDR 时首先检测 OR 标志。

如果 RXEN 为“0”，则接收器不工作。PD0 可以用作普通 I/O 口。而若 RXEN 置位，则 UART 接收器连接到 PD0，强迫其作为输入而不管方向寄存器的设置。

当 UCR 的 CHR9 为“1”时，收发的数据格式为 9 位。要发送的第 9 位是 UCR 的 TXB8（要在写 UDR 之前设置），而接收到的第 9 位是 RXB8。

UART 控制

UART数据寄存器—UDR

BIT	7	6	5	4	3	2	1	0
$0C($2C)	**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

UDR 实际上是两个物理上分离的寄存器，一个用于发送，一个用于接收。读取 UDR 时，访问的是接收 UDR，而在写 UDR 时，访问的是发送 UDR。

UART状态寄存器—USR

BIT	7	6	5	4	3	2	1	0
$0B($2B)	**RXC**	**TXC**	**UDRE**	**FE**	**OR**	-	-	-
读/写	R	R/W	R	R	R	R	R	R
初始值	0	0	1	0	0	0	0	0

位 2..0：保留

RXC：UART接收结束

RXC 置位表示接收到的数据已经从接收移位寄存器传送到 UDR，但是不管数据是否有误。如果 RXCIE 为“1”，则 RXC 置位时将引起接收结束中断。RXC 在读 UDR 时自动被清除。如果采用中断方式，则中断例程必须读一次 UDR 以清除 RXC，否则中断结束后又会引发中断。

TXC：UART发送结束

TXC 置位表示数据已经从发送移位寄存器发送出去，且 UDR 中没有新的要发送的数据。在半双工通信应用当中，由于发送器在发送完数据之后要立即转换到接收模式，所以这个标志位特别有用。

如果 TXCIE 已置位，则 TXC 置位将引发发送结束中断。进入中断例程后 TXC 自动清零，或者用户可以对其写“1”以达到清零的目的。

UDRE：UART数据寄存器空

当数据从 UDR 传送到发送移位寄存器后，UDRE 置位，表明发送器已经准备好接收新的要发送的数据。

当 UDRIE 置位，则只要 UDRE 为“1”，UART 发送结束中断就可以执行。写 UDR 将复位 UDRE。如果利用中断方式发送数据，则在 UART 数据寄存器空中断例程里必须写 UDR 以清除 UDRE，否则中断将连续发生。

复位后 UDRE 的初始值为“1”，表明发送器就绪。

FE：帧错误

MCU 检测到帧错误（如：检测到停止位为“0”）时 FE 置位。

当检测到数据停止位为“1”时 FE 复位。

OR：过速

如果 UDR 未读，而新的数据又已进入移位寄存器，则 OR 置位。

UDR 数据移入后 OR 清零。

UART控制寄存器—UCR

BIT	7	6	5	4	3	2	1	0
$0A($2A)	**RXCIE**	**TXCIE**	**UDRIE**	**RXEN**	**TXEN**	**CHR9**	**RXB8**	**TXB8**
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R	W
初始值	0	0	0	0	0	0	1	0

RXCIE：接收结束中断使能

TXCIE：发送结束中断使能

UDRIE：UART数据寄存器空中断使能

RXEN：接收使能

置位时使能 UART 接收。接收禁止将导致 TXC，OR 和 FE 无法置位，同时也不能复位已经置位的标志位。

TXEN：发送使能

TXEN 清零不能立即关闭发送器，如果发送器当前正在发送的话。只有等到发送完毕后，发送器才真正关闭。

CHR9：9 位字符

置位后 MCU 将接收和发送 9 位字符。第 9 位字符的读/写分别由 RXB8/TXB8 控制。第 9 位字符可用作停止位或奇偶校验。

RXB8：接收的第 9 位

接收到的字符的第 9 位

TXB8：发送的第 9 位

发送字符的第 9 位

波特率产生器

波特率计算公式为：

$$BAUD = \frac{f_{CK}}{16(BURR + 1)}$$

式中，BAUD = 波特率，

f_{CK} = 晶振时钟频率

UBRR = UART 波特率寄存器的内容（0-255）

表 18 给出了在某些晶振下波特率对应的 UBRR 的值。

表 18　不同频率下 UBRR 的设定值

波特率	1MHz	%误差	1.8432 MHz	%误差	2MHz	%误差	2.4576 MHz	%误差
2400	UBRR = 25	0.2	47	0.0	51	0.2	63	0.0
4800	12	0.2	23	0.0	25	0.2	31	0.0
9600	6	7.5	11	0.0	12	0.2	15	0.0
14400	3	7.8	7	0.0	8	3.7	10	3.1
19200	2	7.8	5	0.0	6	7.5	7	0.0
28800	1	7.8	3	0.0	3	7.8	4	6.3
38400	1	22.9	2	0.0	2	7.8	3	0.0
57600	0	7.8	1	0.0	1	7.8	2	12.5
76800	0	22.9	1	33.3	1	22.9	1	0.0
115200	0	84.3	0	0.0	0	7.8	0	25.0

波特率	3.2768 MHz	%误差	3.6864 MHz	%误差	4MHz	%误差	4.608MHz	%误差
2400	UBRR = 84	0.4	95	0.0	103	0.2	119	0.0
4800	42	0.8	47	0.0	51	0.2	59	0.0
9600	20	1.6	23	0.0	25	0.2	29	0.0
14400	13	1.6	15	0.0	16	2.1	19	0.0
19200	10	3.1	11	0.0	12	0.2	14	0.0
28800	6	1.6	7	0.0	8	3.7	9	0.0
38400	4	6.3	5	0.0	6	7.5	7	6.7
57600	3	12.5	3	0.0	3	7.8	4	0.0
76800	2	12.5	2	0.0	2	7.8	3	6.7
115200	1	12.5	1	0.0	1	7.8	2	20.0

波特率	7.3728 MHz	%误差	8MHz	%误差	9.216MHz	%误差	11.059 MHz	%误差
2400	UBRR = 191	0.0	207	0.2	299	0.0	287	-
4800	95	0.0	103	0.2	119	0.0	143	0.0
9600	47	0.0	51	0.2	59	0.0	71	0.0
14400	31	0.0	34	0.8	39	0.0	47	0.0
19200	23	0.0	25	0.2	29	0.0	35	0.0
28800	15	0.0	16	2.1	19	0.0	23	0.0
38400	11	0.0	12	0.2	14	0.0	17	6.7
57600	7	0.0	8	3.7	9	0.0	11	0.0
76800	5	0.0	6	7.5	7	6.7	8	6.7
115200	3	0.0	3	7.8	4	0.0	5	20.0

UART波特率寄存器—UBRR

BIT	7	6	5	4	3	2	1	0
$09($29)	**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

模拟比较器

模拟比较器比较正输入端 PB2（AIN0）**和负输入端** PB3（AIN1）**的值。如果** PB2（AIN0）**的电压高于** PB3（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），也可以触发** T/C1 **的输入捕捉功能。其框图如图** 44 **所示。**

图 44 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($28)	**ACD**	-	**ACO**	**ACI**	**ACIE**	**ACIC**	**ACIS1**	**ACIS0**
读/写	R/W	R	R	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

位 6：**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。**

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。**

ACIC：模拟比较器输入捕捉使能

ACIC 为“1”时，T/C1 的输入捕捉功能由比较器中断触发。此时，比较器的输出与 T/C1 的输入捕捉前端直接相连，T/C1 的输入捕捉噪声抑制和边沿选择仍然适用。如果 ACIC 为“0”，则模拟比较器与 T/C1 没有关联。为了使能比较器驱动的 T/C1 输入捕捉中断，TICIE1 必须置位。

ACIS1，ACIS0：模拟比较器中断模式选择

表 16 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

特点：

- 10 位精度
- ±2LSB 精确度
- 0.5LSB 集成非线性度
- 70 - 280μs 转换时间
- 8 通道
- 轨到轨输入范围
- 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

ATmega603/103 具有 10 位精度的逐次逼近型 AD 转换器。ADC 与一个 8 通道的模拟多路器相连，这样就允许 A 口作为 ADC 的输入引脚。ADC 包含一个采保放大器。ADC 框图见图 45。

ADC 具有两个电源引脚 AV_{CC} 和 AGND。AGND 必须与 GND 相连，AV_{CC} 与 V_{CC} 的差别不能大于±0.3V。

AREF 为外部参考电压输入端。此电压介于 AGND - AV_{CC}。

图 45 ADC 框图

操作

ADC 工作于单次转换模式。用户必须启动每一次转换。

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 13 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。为了保证正确读取数据，系统采用了如下保护逻辑：

读数据时，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 46　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。采样-保持要花费一个 ADC 时钟。在第 13 个时钟 ADC 转换结束，数据进入 ADC 数据寄存器。在进行下一次转换时需要两个额外的 ADC 时钟，如图 47 所示。然后转换可继续进行，如果此时 ADSC 为“1”的话。转换时序见表 27。

图 47　首次转换的时序

表 27　ADC 时序

条件	采样周期	得到结果的周期	总的转换周期数	总的转换时间*（μs）
第一次转换	14	26	28	140 - 560
单次转换模式	1	13	15	75 – 300

图 48　单次转换的时序

图 49 自由运行的时序

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、通过清除 ADEN 关闭 ADC

2、使能 ADC，置位 ADEN 和 ADSC，启动哑转换和实际转换

3、在 14 个 ADC 时钟内进入空闲状态

4、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($27)	-	-	-	-	-	**MUX2**	**MUX1**	**MUX0**
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

MUX2.MUX0：模拟通道选择位

用于选择 ADC 的模拟输入通道 0 – 7

ADC 控制和状态寄存器—ADCSR

BIT	7	6	5	4	3	2	1	0
$06($26)	**ADEN**	**ADSC**	**-**	**ADIF**	**ADIE**	**ADPS 2**	**ADPS 1**	**ADPS 0**
读/写	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
初始值	0	0	0	0	0	0	0	0

位 5：**保留。写** ADCSR **时这一位必须写“**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。**

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 28 ADC **预分频器选择**

ADPS2	ADPS1	ADPS0	分频因子
0	0	0	**无效**
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

BIT	15	14	13	12	11	10	9	8
$05($25)	-	-	-	-	-	-	**ADC9**	**ADC8**
$04($24)	**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

ADCL **要先于** ADCH **读取。**

ADC 噪声消除技术

ATmega603/103 **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、ATmega603/103 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、AV_{CC} 要有自己的解耦电容，如图 49 所示。
4、利用 ADC 的噪声消除技术减少 CPU 引入的噪声。
5、如果 F 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。

图 49　ADC 电源连接

ADC 特性（T_A = -40℃ - 85℃）

符号	参数	条件	Min	典型值	Max	单位
	精度			10		Bit
	绝对精度	V_{REF} = 4V，V_{cc} = 4V ADC 时钟 = 200kHz		1	2	LSB
	绝对精度	V_{REF} = 4V，V_{cc} = 4V ADC 时钟 = 1MHz		4		LSB
	绝对精度	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）			1		LSB
	转换时间		70		280	μs
	时钟频率		50		200	kHz
AV_{CC}	模拟电源		V_{CC} - 0.3(1)		V_{CC} + 0.3(2)	V
V_{REF}	参考电源		AGND		AV_{CC}	V
R_{REF}	参考输入电阻		6	10	13	KΩ
R_{AIN}	模拟输入电阻			100		MΩ

注：1、AV_{CC} 的最小值为 2.7V
　　2、AV_{CC} 的最大值为 6.0V

外部 SRAM 接口

SRAM 接口包括：

- A 口：低字节地址与数据总线
- C 口：高字节地址总线
- ALE：地址锁存信号
- /RD 和/WR：读/写信号

外部 SRAM 使能通过 MCUCR 的 SRE 位设定。SRE 置位后，A 口的端口配置将自动改变。若 SRE 为“0”，则外部 SRAM 禁止，相应端口用作普通 I/O 口。

当 ALE 从高变为低时，A 口的数据为有效的地址。在数据传输过程中 ALE 保持为低。/RD 和/WR 仅在访问外部 SRAM 时有效。

外部 SRAM 使能后，访问内部 SRAM 时在 ALE 引脚可能出现短脉冲，但是在访问外部 SRAM 的过程中 ALE 信号是稳定的。

图 50 为 AVR 与外部 SRAM 的连接图。所用锁存器为 8 位，G 为高时选通。

如图 51 所示，在缺省状态下，访问外部 SRAM 需要 3 个时钟。如果需要一个额外的等待周期，则需要周围 MCUCR 的 SRW 位。结果见图 52。在两种情况下，A 口作为数据总线的时间都是一个周期。一旦数据接收完毕，A 口立即变成低字节地址总线。
具体时序参见图 78，表 46/47/48。

图 50　AVR 与外部 SRAM 的连接

图 51　没有等待周期的外部 SRAM 访问周期

图 52　有等待周期的外部 SRAM 访问周期

I/O 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）。

A 口

A 口是 8 位双向 I/O 口。
A 口有 3 个 I/O 地址：数据寄存器—PORTA，$1B（$3B），数据方向寄存器—DDRA，$1A（$3A）和输入引脚—PINA，$19（$39）。PORTA 和 DDRA 可读可写，PINA 只可读。
所有的管脚都可以单独选择上拉电阻。引脚缓冲器可以吸收 20mA 的电流，能够直接驱动 LED。当管脚被拉低时，如果上拉电阻已经激活，则引脚会输出电流。
A 口的第二功能是访问外部 SRAM 的地址/数据总线。

A 口数据寄存器—PORTA

BIT	7	6	5	4	3	2	1	0
$1B($3B)	**PORTA7**							**PORTA0**
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

A 口数据方向寄存器—DDRA

BIT	7	6	5	4	3	2	1	0
$1A($3A)	**DDA7**							**DDA0**
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

A 口输入引脚地址—PINA

BIT	7	6	5	4	3	2	1	0
$19($39)	**PINA7**							**PINA0**
读/写	R	R	R	R	R	R	R	R
初始值	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z

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

A 口用作通用数字 I/O

作为通用数字 I/O 时，A 口的 8 个管脚具有相同的功能。

PAn，通用 I/O 引脚：DDRA 中的 DDAn 选择引脚的方向。如果 DDAn 为“1”，则 PAn 为输出脚；如果 DDAn 为“0”，则 PAn 为输入脚。在复位期间，A 口为三态口。

表 29　A 口的配置

DDAn	PORTAn	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 0 输出
1	1	输出	N	推挽 1 输出

n：7，6..0，引脚号

A 口示意图

图 53　A 口示意图（PA0-PA7）

B 口

B 口是 8 位双向 I/O 口。

B 口有 3 个 I/O 地址：数据寄存器—PORTB，$18（$38），数据方向寄存器—DDRB，$17（$37）和输入引脚—PINB，$16（$36）。PORTB 和 DDRB 可读可写，PINB 只可读。

所有的管脚都可以单独选择上拉电阻。引脚缓冲器可以吸收 20mA 的电流，能够直接驱动 LED。当管脚被拉低时，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 30　B 口第二功能

管脚	第二功能
PB0	/SS（SPI 从机选择）
PB1	SCK（串行时钟）
PB2	MOSI（程序下载时的数据输入线）
PB3	MISO（程序下载时的数据输出线）
PB4	OC0/PWM0（T/C0 的比较输出及 PWM 输出）
PB5	OC1A/PWM1A（T/C1 的 A 比较输出及 PWM 输出）
PB6	OC1B/PWM1B（T/C1 的 B 比较输出及 PWM 输出）
PB7	OC2/PWM2（T/C2 的比较输出及 PWM 输出）

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($38)	**PORTB7**							**PORTB0**
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

B 口数据方向寄存器—DDRB

BIT	7	6	5	4	3	2	1	0
$17($37)	**DDB7**							**DDB0**

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

B 口输入引脚地址—PINB

BIT	7	6	5	4	3	2	1	0
$16($36)	**PINB7**							**PINB0**
读/写	R	R	R	R	R	R	R	R
初始值	Hi-Z	Hi-Z	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 8 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 31　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 0 输出
1	1	输出	N	推挽 1 输出

n：7，6..0，引脚号

B 口的第二功能

- OC2/PWM2—PB7
  T/C2 工作于 PWM 模式时的输出
- OC1A/PWM1A—PB6
  T/C1A 工作于 PWM 模式时的输出
- OC1B/PWM1B—PB5
  T/C1B 工作于 PWM 模式时的输出
- OC0/PWM0—PB4
  T/C0 工作于 PWM 模式时的输出
- MISO—PB3
  SPI 的主机数据输入，从机数据输出。配置为主机时，此引脚配置为输入而不管 DDB6 的值。而当 SPI 为主机时，MISO 的配置由 PB6 和 DDB6 控制。
- MOSI—PB2
  SPI 的主机数据输出，从机数据输入。配置为从机时，此引脚配置为输入而不管 DDB5 的值。而当 SPI 为主机时，MOSI 的配置由 PB5 和 DDB5 控制。
- SCK—PB1
  SPI 的主机时钟输出，从机时钟输入。配置为从机时，此引脚配置为输入而不管 DDB7 的值。而当 SPI 为主机时，SCK 的配置由 PB7 和 DDB7 控制。
- /SS—PB0
  从机选择信号。配置为从机时，此引脚配置为输入而不管 DDB4 的值。/SS 为低将激活 SPI。配置为主机时，此引脚的方向由 DDB4 控制。如果 DDB4 为“1”，则上拉仍然可

以由 PORTB4 控制。

- AIN1—PB3
  当配置为输入（DDB2=3），无上拉电阻（PB3=0）时，为模拟比较器的负输入端
- AIN0—PB2
  当配置为输入（DDB2=0），无上拉电阻（PB2=0）时，为模拟比较器的正输入端
- T1—PB1
  T/C1 的外部记数输入
- T0—PB0
  T/C0 的外部记数输入

B 口示意图

图 54　B 口示意图（PB0）

图 55　B 口示意图（PB1）

图 56　B 口示意图（PB2）

图 57　B 口示意图（PB3）

图 58　B 口示意图（PB4）

图 59　B 口示意图（PB5/6）

图 60　B 口示意图（PB7）

C 口

C 口是 8 位输出口。

C 口可作为访问外部 SRAM 的高阶地址线。复位时不为三态。但是在两个稳定的示众后，C 口为初始值。

C 口数据寄存器—PORTC

BIT	7	6	5	4	3	2	1	0
$15($35)	**PORTC7**							**PORTC0**
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

图 60　C 口示意图（PC0-PC7）

D 口

D **口是** 8 **位双向** I/O **口，具有内部上拉电阻。**

D **口有** 3 **个** I/O **地址：数据寄存器—**PORTD，$12（$32），**数据方向寄存器—**DDRD，$11（$31）**和输入引脚—**PIND，$10（$30）。PORTD **和** DDRD **可读可写，**PIND **只可读。**

D **口的引脚缓冲器可以吸收** 20mA **的电流，能够直接驱动** LED**。当管脚被拉低时，如果上拉电阻已经激活，则引脚会输出电流。**

D **口的第二功能如下表所示：**

表 33　**D **口第二功能

管脚	**第二功能**
PD0	/INT0（**外部中断** 0 **输入**）
PD1	/INT1（**外部中断** 1 **输入**）
PD2	/INT2（**外部中断** 2 **输入**）
PD3	/INT3（**外部中断** 3 **输入**）
PD4	IC1（T/C1 **输入捕捉**）
PD6	T1（T/C1 **外部记数输入**）
PD7	T2（T/C2 **外部记数输入**）

D 口数据寄存器—PORTD

BIT	7	6	5	4	3	2	1	0
$12($32)	**PORTD7**							**PORTD0**
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

D 口数据方向寄存器—DDRD

BIT	7	6	5	4	3	2	1	0
$11($31)	**DDD7**							**DDB0**
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

D 口输入引脚地址—PIND

BIT	7	6	5	4	3	2	1	0
$10($30)	**PIND7**							**PINB0**
读/写	R	R	R	R	R	R	R	R
初始值	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z

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

D 口用作通用数字 I/O

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

表 34　**D **口的配置

DDDn	**PORTDn**	**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 0 输出
1	1	输出	N	推挽 1 输出

n：7,6..0，引脚号

D 口的第二功能

- /INT0-/INT3—PD0-3
  外部中断源 0-3。
- IC1—PD4
  T/C1 的输入捕捉引脚。当引脚上发生正（或负，可选）跳变时，TCNT1 的值将被送入 T/C1 的输入捕捉寄存器。此时引脚必须配置为输入。
- T1—PD6
  T/C1 的时钟源
- T2—PD7
  T/C2 的时钟源

D 口示意图：

图 62　D 口示意图（PD0/1/2/3）

图 63　D 口示意图（PD4）

图 64　D 口示意图（PD5）

图 65　D 口示意图（PD6 和 PD7）

E 口

E 口是 8 位双向 I/O 口，具有内部上拉电阻。
E 口有 3 个 I/O 地址：数据寄存器—PORTE，$03（$23），数据方向寄存器—DDRE，$02（$22）和输入引脚—PINE，$01（$21）。PORTE 和 DDRE 可读可写，PINE 只可读。
E 口的引脚缓冲器可以吸收 20mA 的电流，能够直接驱动 LED。当管脚被拉低时，如果上拉电阻已经激活，则引脚会输出电流。
E 口的第二功能如下表所示：

表 33　E 口第二功能

管脚	第二功能
PE0	PDI/RXD（可编程数据输入或 UART 接收）
PE1	PD0/TXD（可编程数据输出或 UART 发送）
PE2	AC+（模拟比较器正输入端）
PE3	AC-（模拟比较器负输入端）

PE4	INT4（外部中断 4 输入）
PE5	INT5（外部中断 5 输入）
PE6	INT6（外部中断 6 输入）
PE7	INT7（外部中断 7 输入）

E 口数据寄存器—PORTE

BIT	7	6	5	4	3	2	1	0
$03($23)	**PORTE7**							**PORTE0**
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

E 口数据方向寄存器—DDRE

BIT	7	6	5	4	3	2	1	0
$02($22)	**DDE7**							**DDE0**
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

E 口输入引脚地址—PINE

BIT	7	6	5	4	3	2	1	0
$01($21)	**PINE7**							**PINE0**
读/写	R	R	R	R	R	R	R	R
初始值	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z

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

E 口用作通用数字 I/O

PEn，通用 I/O 引脚：DDRE 中的 DDEn 选择引脚的方向。如果 DDEn 为“1”，则 PEn 为输出脚；如果 DDEn 为“0”，则 PEn 为输入脚。在复位期间，E 口为三态口。

表 35　E 口的配置

DDEn	PORTEn	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 0 输出
1	1	输出	N	推挽 1 输出

n：7,6..0，引脚号

E 口的第二功能

- PDI/RXD—PE0
  PDI，串行下载的数据输入端。
  RXD，UART 的数据接收端
- PDOI/TXD—PE1
  PDO，串行下载的数据输出端。
  TXD，UART 的数据发送端
- AC+—PE2
  模拟比较器的正输入端
- AC–PE3

模拟比较器的负输入端

- INT4-INT7—PE4-7
  外部中断源 4-7。

E 口示意图：

图 66　E 口示意图（PE0）

图 67　E 口示意图（PE1）

图 68　E 口示意图（PE2）

图 69　E 口示意图（PE3）

图 70　E 口示意图（PE4/5/6/7）

F 口

F 口是 8 位输入口。

F 口输入引脚地址—PINF

BIT	7	6	5	4	3	2	1	0
$00($20)	PINF7							PINF0
读/写	R	R	R	R	R	R	R	R
初始值	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z

PINF 不是一个寄存器，这个地址用来访问 F 口的物理值。

F 口示意图：

图 71　F 口示意图（PF0-7）

编程

程序和数据锁定位

ATmega603/103 具有两个锁定位，如表 36 所示。锁定位只能通过片擦除命令擦除为“1”。

表 36　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、在并行编程模式下，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

ATmega603/103 有 4 个熔断位：SPIEN、SUT1、SUT0、EESAVE。

- SPIEN 编程（为“0”）后，串行下载程序使能。缺省值为“0”。
- EESAVE 编程后，执行片擦除时保留 EEPROM 中的内容。缺省值为“1”。
- SUT1/0 决定 MCU 的起动时间。参见表 6。缺省值为“11”。

串行下载程序时不能访问熔断位，只能在并行下载程序时访问。芯片擦除命令不影响熔断位。

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。其位置为：

对于 ATmega603：

1、$000：$1E（表明是 Atmel 生产的）

2、$001：$96（64K 字节的 FLASH）

3、$002：$01（当$01 地址为$99 时，器件为 603）

对于 ATmega103：

1、$000：$1E（表明是 Atmel 生产的）

2、$001：$97（128K 字节的 FLASH）

3、$002：$01（当$01 地址为$97 时，器件为 103）

编程 FLASH 和 EEPROM

ATmega603/103 具有 64K/128K 字节的片内可编程 FLASH 和 2K/4K 字节的 EEPROM，在出厂时已经被擦除为“1”。器件支持+12V 高压并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。

ATmega603/103 的 FLASH 结构为 256/512 字节/页。编程 FLASH 时，数据首先保存在页缓冲区里。

在两种编程模式下，EEPROM 是以字节的形式写入的。片内集成了自擦和自定时除功能。在编程时，要注意电源电压要满足要求。

表 37 编程电源电压

型号	串行编程	并行编程
ATmega603	4.0V – 5.0V	4.0V – 5.0V
ATmega603L	3.4V – 3.6V	3.4V – 5.0V
ATmega103	4.0V – 5.0V	4.0V – 5.0V
ATmega103L	3.4V – 3.6V	3.4V – 5.0V

并行编程

本节介绍如何在并行模式下对 FLASH、EEPROM、熔断位和锁定位进行编程。

图 72 并行编程

信号命名：

ATmega603/103 的某些管脚在本节将以并行编程的信号来命名。XA1/XA0 决定了当 XTAL1 引脚上出现正脉冲时要进行的动作。

当驱动/WE 或/OE 时，执行加载的命令。

表 38　管脚命名

编程时的信号名称	管脚	I/O	功能
RDY/BSY	PD1	O	0：器件忙 1：可以接受新命令
/OE	PD2	I	输出使能（低电平）
/WR	PD3	I	写脉冲（低电平）
BS1	PD4	I	字节选择 1（“0”：低字节 “1”：高字节）
XA0	PD5	I	见表 39
XA1	PD6	I	见表 39
BS2	PD7	I	字节选择 2，总为低电平
PAGEL	PA0	I	程序存储器页加载
DATA	PB0-7	I/O	双向数据总线（/OE 为低时输出）

表 39　XA1 和 XA0 编码

XA1	XA0	XTAL1 给正脉冲后的操作
0	0	加载 FLASH 或 EEPROM 的地址（BS 决定是高位地址还是低位地址）
0	1	加载数据（BS 决定是高位地址还是低位地址）
1	0	加载命令
1	1	无操作

表 40　命令字节编码

命令字节	执行的命令
1000 0000	芯片擦除
0100 0000	写熔断位
0010 0000	写锁定位
0001 0000	写 FLASH
0001 0001	写 EEPROM
0000 1000	读厂标
0000 0100	读熔断位和锁定位
0000 0010	读 FLASH
0000 0011	读 EEPROM

进入编程模式：

以下步骤使器件进入并行编程模式：

1、按表 37 在 V_{CC} 和 GND 之间加上电源电压

2、拉低/RESET 和 BS1，等待至少 100ns

3、给/RESET 引脚加上 11.5~12.5V 的电压。如果 BS1 在/RESET 加上+12V 之后 100ns 之内发生动作，将导致器件无法进入编程模式。

芯片擦除：

此命令擦除所有的 FLASH 和 EEPROM，以及锁定位。在 FLASH 和 EEPROM 完全擦除之前，锁定位不会擦除。擦除过程不影响熔断位。擦除命令必须在对 FLASH 和 EEPROM 重新编程之前执行。

加载“擦除”命令：

1、设置 XA1，XA0 为“10”—使能命令加载

2、设置 BS1 为“0”。

3、设置 DATA 为“1000 0000”—擦除命令
4、给 XTAL1 一个正脉冲—加载命令
5、给/WE 施加一个 t_{WLWH_CE} 的负脉冲。擦除命令在 RDY/BSY 引脚没有反馈。

表 41　片擦除时/WR 的最小脉冲宽度

符号	3.2V	3.6V	4.0V	5.0V
t_{WLWH_CE}	56ms	43ms	35ms	22ms

FLASH 编程：
FLASH 组织为 256 字节/页，总共 256/512 页。数据首先保存在页缓冲区里。
A：加载命令
1、设置 XA1，XA0 为“10”—使能命令加载
2、设置 BS1 为“0”。
3、设置 DATA 为“0001 0000”—写 FLASH 命令
4、给 XTAL1 一个正脉冲—加载命令
B：加载低位地址
1、设置 XA1，XA0 为“00”—使能地址加载
2、设置 BS1 为“0”— 选择地址的低位字节
3、设置 DATA = 地址的低位字节（$00~$FF）
4、给 XTAL1 一个正脉冲—加载地址的低位字节
C：加载数据的低位字节
1、设置 BS1 为“0”— 选择数据的低位字节
2、设置 XA1，XA0 为“01”—使能数据加载
3、设置 DATA = 数据的低位字节（$00~$FF）
5、给 XTAL1 一个正脉冲—加载数据的低位字节
D：锁存数据的低位字节
　在 PAGEL 上施加一个正脉冲。参见图 73。
E：加载数据的高位字节
1、设置 BS1 为“1”— 选择数据的高位字节
2、设置 XA1，XA0 为“01”—使能数据加载
3、设置 DATA = 数据的高位字节（$00~$FF）
4、给 XTAL1 一个正脉冲—加载数据的低位字节
F：锁存数据的高位字节
　在 PAGEL 上施加一个正脉冲。
G：重复 128 次 B 到 F 的过程，以填满一个页缓冲区。
　在 PAGEL 上施加一个正脉冲。参见图 73。
H：加载高位地址
1、设置 XA1，XA0 为“00”—使能地址加载
2、设置 BS1 为“1”— 选择地址的高位字节
3、设置 DATA = 地址的高位字节（$00 - $7F/$FF）
4、给 XTAL1 一个正脉冲—加载地址的高位字节
I：编程页
1、在/WR 上施加一个负脉冲。RDY/BSY 将变低。
2、一直等到 RDY/BSY 变高。参见图 74。

J：结束编程
1、设置 XA1，XA0 为“10”—使能命令加载
2、设置 DATA = ‘0000 0000’，这是“无操作”的命令。
3、给 XTAL1 一个正脉冲
K：重复 A 到 J，直到所有数据都写入。

图 73 编程波形一

图 74 编程波形二

读 FLASH：
1、A：加载命令“0000 0010”
2、H：加载地址的高位字节（$00~$7F/$FF）
3、B：加载地址的低位字节（$00~$FF）
4、设置/OE 和 BS1 为“0”。此时可以从 DATA 总线读 FLASH 数据的低位字节。
5、设置 BS1 为“1”。 此时可以读 FLASH 数据的高位字节。
6、设置/OE 为“1”。

编程 EEPROM：
1、A：加载命令“0001 0001”
2、H：加载地址的高位字节（$00~$07/$0F）
3、B：加载地址的低位字节（$00~$FF）
4、E：加载数据的低位字节（$00~$FF）
L：写数据的低位字节
1、设置 BS1 为“0”— 选择数据的低位字节
2、在/WR 上施加负脉冲。RDY/BSY 将变低。
3、一直等到 RDY/BSY 变高。参见图 75。
加载后的命令和地址在编程过程中维持不变。为了提高效率，应考虑如下因素：
- 写/读多个存储器地址时，仅需加载一次命令。
- 仅需在编程新的一页（256 个 WORD）时加载一次地址高字节。
- 如果执行了片擦除命令，则无需编程数据$FF。

这些原则同样适用于读取 FLASH、EEPROM 和厂标。

图 75 编程 EEPROM

读 EEPROM：
1、A：加载命令“0000 0011”
2、H：加载地址的高位字节（$00~$07/$0F）
3、B：加载地址的低位字节（$00~$FF）
4、设置/OE 和 BS1 为“0”。此时可以从 DATA 总线读取数据的低位字节。
5、设置/OE 为“1”。

编程熔断位：

1、A：**加载命令**“0100 0000”

2、C：**加载数据的低位字节。**Bit n =“0”**代表要编程。**

Bit 5 = SPIEN

Bit 2 = EESAVE

Bit 2 = **总为**“1”

Bit 1 = SUT1

Bit 0 = SUT0

Bit 7,6,4,2 = “1”**。这些位是保留位。**

3、**在**/WR **上施加** t_{WLWH_PFB} **的负脉冲。此命令在** RDY/BSY **引脚没有反馈。**

编程锁定位：

1、A：**加载命令**“0010 0000”

2、D：**加载数据的低位字节。**Bit n =“0”**代表要编程。**

Bit 2 = Lock Bit2

Bit 1 = Lock Bit1

Bit 7- 3, 0 = “1”**。这些位是保留位。**

3、L：**写数据的低位字节**

锁定位只能在芯片擦除上时清除。

读熔断位和锁定位：

1、 A：**加载命令**“0000 0100”

2、 **设置**/OE **为**“0”，BS1 **为**“0”。 **此时可以从** DATA **总线读取数据：**

Bit 5 = SPIEN

Bit 2 = EESAVE

Bit 2 = **总为**“1”

Bit 1 = SUT1

Bit 0 = SUT0

设置/OE **为**“0”，BS1 **为**“0”。 **此时可以从** DATA **总线读取数据：**

Bit 2 = Lock Bit1

Bit 1 = Lock Bit2

3、 **设置**/OE **为**“1”**。**

读厂标：

1、 A：**加载命令**“0000 1000”

C：**加载数据的低位字节**（$00~$02）**。**

设置/OE **为**“0”，BS1 **为**“0”。 **此时可以从** DATA **总线读取到数据。**

2、 **设置**/OE **为**“1”**。**

并行编程特性

图 76 并行编程时序

表 42　并行编程特性　T_A = 25℃±10%，V_{CC} = 5V±10%

符号	参数	最小值	典型值	最大值	单位
V_{PP}	编程使能电压	11.5		12.5	V
I_{PP}	编程使能电流			250	μA
t_{DVXH}	Data & Control Setup before XTAL1 High	67			ns
t_{XHXL}	XTAL2 脉宽	67			ns
t_{XLDX}	Data & Control Hold after XTAL1 Low	67			ns
t_{XLWL}	XTAL1 Low to /WR Low	67			ns
t_{BVWL}	BS1 Valid to /WR Low	67			ns
t_{RHBX}	BS1 Hold after RDY/BSY High	67			ns
t_{WLWH}	/WR Pluse Width Low(1)	67			ns
t_{WHRL}	/WR High to RDY/BSY Low(2)		20		ns
t_{WLRH}	/WR Low to RDY/BSY Gigh(2)	0.5	0.7	0.9	ms
t_{XLOL}	XTAL1 Low to /OE Low	67			ns
t_{OLDV}	/OE Low to DATA Valid		20		ns
t_{OHDZ}	/OE High to DATA Tri-stated			20	ns
t_{WLWH_CE}	/WR Pulse Width Low for Chip Erase	5	10	15	ms
t_{WLWH_PFB}	/WR Pulse Width Low for Programming the Fuse Bits	1.0	1.5	1.8	ms

注意：1、在芯片擦除时使用 t_{WLWH_CE}，而在编程锁定位是使用 t_{WLWH_PFB}。

2、如果 t_{WLWH} 保持地比 t_{WLRH} 长，则看不见 RDY/BSY 脉冲。

串行下载

当/RESET 或/PEN 拉到地时，FLASH 和 EEPROM 可以进行串行下载。串行接口包括 SCK，TXD/PDI（输入）和 RXD/PDO（输出）。/RESET 拉低后，在进行编程/擦除之前首先要执行编程使能指令。

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。

对于 ATmega103，FLASH 的范围是$0000~$FFFF，EEPROM 的范围是$000~$0FFF。

对于 ATmega603，FLASH 的范围是$0000~$7FFF，EEPROM 的范围是$000~$07FF。

时钟可以从 XTAL1 引脚输入，或是将晶振接到 XTAL1 和 XTAL2。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 ATmega603/103，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。在整个编程过程当中，/RESET 要一直保持为低。在某些情况下，编程器可能无法保证在上电过程中保持 SCK 为低。此时，可以先拉低 SCK，然后在/RESET 上施加一个持续至少两个 XTAL 时钟的正脉冲。

作为另一种选择，/RESET 可以用/PEN 代替。但是，如果编程器无法保证在上电过程中保持 SCK 为低，那么这种方法不能使用。

2、至少等待 20ms。 然后在 PDI/RXD（PE0）加载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 $2t_{WD\_ERASE}$，然后在/RESET 上施加正脉冲，回到第二步。

5、FLASH 是以页为单位写入的。加载页命令同时完成数据和地址最低 7 位的加载。而写页命令则加载地址的高 9 位。写操作起动后，必须等待至少 t_{WD_FLASH} 的时间才能继续下一页的加载。

6、EEPROM 是一个字节一个字节编程的。在新数据写入之前原先内容将被自动擦除。发送完写指令后要等待 t_{WD_EEPROM} 的时间。对于擦除过的器件，数据$FF 就用不着再写了。

7、任意一个内存地址都可以用读指令在 PDO/RXD（PE1）读出。

8、编程结束后，把/RESET 拉高，进入正常工作模式。

9、下电过程（如果需要的话）：

将 XTAL1 拉低（如果没有用外部晶振）。

把/RESET 拉高。

关掉电源。

EEPROM 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 P1；自擦除过程结束后，器件则返回 P2（P1 和 P2 的定义见表 44）。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。但是对于特定的数据 P1 和 P2，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_EEPROM} 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。

FLASH 不支持数据检测功能！

表 43　写 FLASH 或 EEPROM 时的最小等待时间

符号	3.2V	3.6V	4.0V	5.0V
t_{WD_FLASH}	56ms	43ms	35ms	22ms
t_{WD_EEPROM}	9ms	7ms	6ms	4ms

表 44　EEPROM 数据检测返回值

型号	P1	P2
ATmega603	$80	$7F
ATmega103	$80	$7F

表 45　ATmega603/103 的串行编程指令

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 H000	aaaa aaaa	bbbb bbbb	oooo oooo	从字地址 a:b 读取 H（高或低）字节 o
加载 FLASH 页	0100 H000	xxxx xxxx	xbbb bbbb	iiii iiii	在字地址 b 写 H（高或低）字节 i

写FLASH页	0100 1100	aaaa aaaa	bxxx xxxx	xxxx xxxx	将缓冲区页写到地址 a:b
读 EEPROM	1010 0000	0000 aaaa	bbbb bbbb	oooo oooo	从地址 a:b 读取数据 o
写 EEPROM	1100 0000	0000 aaaa	bbbb bbbb	iiii iiii	写数据 i 到地址 a:b
读锁定位	0101 1000	xxxx xxxx	xxxx xxxx	xxxx x21x	“0”代表已编程
写锁定位	1010 1100	1111 1211	xxxx xxxx	xxxx xxxx	写锁定位
读熔丝位	0101 0000	xxxx xxxx	xxxx xxxx	xx5x 6143	“0”代表已编程
写熔丝位	1010 1100	1011 6143	xxxx xxxx	xxxx xxxx	“0”代表要编程
读厂标	0011 0000	Xxxx xxxx	xxxx xxbb	oooo oooo	从地址 b 读厂标 o

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 = Lock Bit1

2 = Lock Bit2

3 = SUT0

4 = SUT1

5 = SPIEN

6 = EESAVE

图 77 串行编程波形

直流特性

T_A = -40℃到 85℃，V_{CC} = 2.7V – 3.6V 和 4.0V – 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 XTAL1	-0.5		0.3 V_{CC} (1)	V
V_{IL1}	输入低电压	XTAL1	-0.5		0.1(1)	V
V_{IH}	输入高电压	除了 XTAL1 和 /RESET	0.6 V_{CC} (2)		V_{CC} +0.5	V
V_{IH1}	输入高电压	XTAL1	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) A,B,C,D,E 口	I_{OL} =20mA，V_{CC} = 5V I_{OL} =10mA，V_{CC} = 3V			0.6 0.5	V
V_{OH}	输出高电压 (4) A,B,C,D,E 口	I_{OH} =-3mA，V_{CC} = 5V I_{OH} =-1.5mA,V_{CC} = 3V	4.3 2.2			V
I_{IL}	输入泄露电流 I/O 脚	V_{CC} = 6V，pin low			8.0	μA
I_{IH}	输入泄露电流 I/O 脚	V_{CC} = 6V，pin low			8.0	μA
RRST	复位上拉电阻		100		500	kΩ
$R_{I/O}$	I/O 口的上拉电阻		35		120	kΩ
I_{CC}	电源电流	工作状态，V_{CC} = 3V，4MHz			5.0	mA

		闲置状态，V_{CC} = 3V，4MHz			2.0	mA
I_{CC}	掉电模式[5]	WDT 使能，V_{CC} = 3V			40.0	μA
		WDT 禁止，V_{CC} = 3V			25.0	μA
I_{CC}	省电模式[5]	WDT 禁止，V_{CC} = 3V			35.0	μA
V_{ACIO}	模拟比较器输入偏置电流	V_{CC} = 5V			40	mV
I_{ACLK}	模拟比较器输入泄露电流	V_{CC}=5V V_{IN} =V_{CC}/2	-50		50	nA
t_{ACPD}	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**

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** IOL **之和不能超过** 400mA

2]A **口、** ALE、C3-C7 **的** IOL **之和不能超过** 100mA

3]C0-C2、/RD、/WR、D0-D7、XTAL2 **的** IOL **之和不能超过** 100mA

4]B0-B7 **的** IOL **之和不能超过** 100mA

5]E0-E7 **的** IOL **之和不能超过** 100mA

如果 IOL **超过测试条件，则** VOL **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：**

6]**所有** I/O **口的** IOH **之和不能超过** 400mA

7]A **口、** ALE、C3-C7 **的** IOH **之和不能超过** 100mA

8]C0-C2、/RD、/WR、D0-D7、XTAL2 **的** IOH **之和不能超过** 100mA

9]B0-B7 **的** IOH **之和不能超过** 100mA

10] E0-E7 **的** IOH **之和不能超过** 100mA

如果 IOH **超过测试条件，则** VOH **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** V_{CC} **最小不能低于** 2V

外部 SRAM 时序

表 46 外部 SRAM **特性（**4.0V – 6.0V，**无等待周期）**

	符号	参数	6M 晶振		其他晶振		单位
			最小值	最大值	最小值	最大值	
0	$1/t_{CLCL}$	振荡频率			0.0	6.0	MHz
1	t_{LHLL}	ALE 脉冲宽度	48.3		0.5 t_{CLCL} -35.0[1]		ns
2	t_{AVLL}	A口地址有效到 ALE低	43.3		0.5 t_{CLCL} –40.0[1]		ns
3a	t_{LLAX_ST}	ALE低之后地址保持时间，ST/STD/STS 指令	77.3		0.5 t_{CLCL} -10.0[2]		ns
3b	t_{LLAX_LD}	ALE低之后地址保持时间，LD/LDD/LDS 指令	15.0		15.0		ns

4	t_{AVLLC}	C口地址有效到 ALE低	43.3		0.5 t_{CLCL} –40.0(1)		ns
5	t_{AVRL}	地址有效到 RD 低	136.7		1.0 t_{CLCL} –30.0		ns
6	t_{AVWL}	地址有效到 WR 低	215.0		1.5 t_{CLCL} -35.0(1)		ns
7	t_{LLWL}	ALE 低到 WR 低	146.7	145	1.0 t_{CLCL} -20.0	1.0 t_{CLCL} +20.0	ns
8	t_{LLRL}	ALE 低到 RD 低	146.7	82.5	0.5 t_{CLCL} -20.0(2)	0.5 t_{CLCL} +20.0(2)	ns
9	t_{DVRH}	数据设定到 RD 高	65.0		65.0		ns
10	T_{RLDV}	RD 低到数据有效		70.0		1.0 t_{CLCL} -30.0	ns
11	T_{RHDX}	RD 高之后数据保持	0.0		0.0		ns
12	T_{RLRH}	RD 脉冲宽度	146.7		1.0 t_{CLCL} -20.0		ns
13	T_{DVWL}	数据设定到 WR 低	53.3		0.5 t_{CLCL} –30.0(2)		ns
14	T_{WHDX}	WR 高之后数据保持	0.0		0.0		ns
15	T_{DVWH}	数据有效到 WR 高	146.7		1.0 t_{CLCL} -20.0		ns
16	T_{WLWH}	WR 脉冲宽度	63.3		0.5 t_{CLCL} -20.0(1)		ns

表 47　外部 SRAM 特性（4.0V – 6.0V，一个等待周期）

	符号	参数	6M 晶振		其他晶振		单位
			最小值	最大值	最小值	最大值	
0	$1/t_{CLCL}$	振荡频率			0.0	6.0	MHz
10	T_{RLDV}	RD 低到数据有效		303.4		2.0 t_{CLCL} –30.0	ns
12	T_{RLRH}	RD 脉冲宽度	313.4		2.0 t_{CLCL} –20.0		ns
15	T_{DVWH}	数据有效到 WR 高	313.4		2.0 t_{CLCL} –30.0		ns
16	T_{WLWH}	WR 脉冲宽度	230.0		1.5 t_{CLCL} –20.0(2)		ns

注意： 1、假定为 50%的占空比。半周期（half period）实际为 XTAL1 的高电平时间。
2、假定为 50%的占空比。半周期（half period）实际为 XTAL1 的低电平时间。

表 48　外部 SRAM 特性（2.7V – 3.6V，无等待周期）

	符号	参数	4M 晶振		其他晶振		单位
			最小值	最大值	最小值	最大值	
0	$1/t_{CLCL}$	振荡频率			0.0	4.0	MHz
1	t_{LHLL}	ALE 脉冲宽度	65.0		0.5 t_{CLCL} –60.0(1)		ns
2	t_{AVLL}	A口地址有效到 ALE低	75.0		0.5 t_{CLCL} –50.0(1)		ns
3a	t_{LLAX_ST}	ALE低之后地址保持时间，ST/STD/STS 指令	125.0		0.5 t_{CLCL} (2)		ns
3b	t_{LLAX_LD}	ALE低之后地址保持时间，LD/LDD/LDS 指令	15.0		15.0		ns
4	t_{AVLLC}	C口地址有效到 ALE低	75.0		0.5 t_{CLCL}		ns

					$-50.0^{(1)}$		
5	t_{AVRL}	地址有效到 RD 低	205.0		1.0 t_{CLCL} −50.0		ns
6	t_{AVWL}	地址有效到 WR 低	325.0		1.5 t_{CLCL} $-50.0^{(1)}$		ns
7	t_{LLWL}	ALE 低到 WR 低	230.0	270.0	1.0 t_{CLCL} −20.0	1.0 t_{CLCL} +20.0	ns
8	t_{LLRL}	ALE 低到 RD 低	105.0	145.0	0.5 t_{CLCL} $-20.0^{(2)}$	0.5 t_{CLCL} $+20.0^{(2)}$	ns
9	t_{DVRH}	数据设定到 RD 高	110.0		110.0		ns
10	T_{RLDV}	RD 低到数据有效		170.0		1.0 t_{CLCL} -40.0	ns
11	T_{RHDX}	RD 高之后数据保持	0.0		0.0		ns
12	T_{RLRH}	RD 脉冲宽度	230.0		1.0 t_{CLCL} −20.0		ns
13	T_{DVWL}	数据设定到 WR 低	90.0		0.5 t_{CLCL} $-35.0^{(2)}$		ns
14	T_{WHDX}	WR 高之后数据保持	0.0		0.0		ns
15	T_{DVWH}	数据有效到 WR 高	220.0		1.0 t_{CLCL} −20.0		ns
16	T_{WLWH}	WR 脉冲宽度	100.0		0.5 t_{CLCL} $-25.0^{(1)}$		ns

表 49　外部 SRAM 特性（2.7V – 3.6V，**一个等待周期**）

	符号	参数	4M 晶振		其他晶振		单位
			最小值	最大值	最小值	最大值	
0	$1/t_{CLCL}$	振荡频率			0.0	4.0	MHz
10	T_{RLDV}	RD 低到数据有效		460.0		2.0 t_{CLCL} −40.0	ns
12	T_{RLRH}	RD 脉冲宽度	480.0		2.0 t_{CLCL} −20.0		ns
15	T_{DVWH}	数据有效到 WR 高	480.0		2.0 t_{CLCL} −40.0		ns
16	T_{WLWH}	WR 脉冲宽度	350.0		1.5 t_{CLCL} $-25.0^{(2)}$		ns

注意： 1、**假定为** 50%**的占空比。半周期**（half period）**实际为** XTAL1 **的高电平时间。**

2、**假定为** 50%**的占空比。半周期**（half period）**实际为** XTAL1 **的低电平时间。**

图 78　外部 SRAM **时序图**

外部时钟驱动波形

图 79　外部时钟

外部时钟

符号	参数	V_{CC}=2.7V~3.6V		V_{CC}=4.0V~5.5V		单位
		最小值	最大值	最小值	最大值	
$1/t_{CLCL}$	振荡频率	0	4	0	6	MHz
t_{CLCL}	时钟周期	250		176		ns
t_{CHCX}	高电平时间	100		67		ns
t_{CLCX}	低电平时间	100		67		ns
t_{CLCH}	上升时间		1.6		0.5	μs

t_{CHCL}	下降时间		1.6		0.5	μs

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
4	2.7 – 3.6	ATmega603L – 4AC	64A	商用 （0℃ - 70℃）
		ATmega603L – 4AI	64A	工业 （-40℃ - 85℃）
6	4.0 – 5.5	ATmega603 – 6AC	64A	商用 （0℃ - 70℃）
		ATmega603 – 6AI	64A	工业 （-40℃ - 85℃）
4	2.7 – 3.6	ATmega103L – 4AC	64A	商用 （0℃ - 70℃）
		ATmega103L – 4AI	64A	工业 （-40℃ - 85℃）
6	4.0 – 5.5	ATmega103 – 6AC	64A	商用 （0℃ - 70℃）
		ATmega103 – 6AI	64A	工业 （-40℃ - 85℃）

封装类型	
64A	64 脚，TQFP

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.）