ATmega83/163

特点

1. 高性能，低功耗的 AVR 结构
2. 先进的 RISC 结构
   —ATmega83：128 条指令——大多数为单指令周期
   —Atmega163：130 条指令——大多数为单指令周期
   —32 个 8 位通用（工作）寄存器
   —工作在 8MHz 时具有 8MIPS 的性能
   —2 个周期的硬件乘法器
3. 数据和非易失性程序内存
   —ATmega83：8K 字节的在线可编程 FLASH（擦除次数：1000 次）
   —ATmega163：16K 字节的在线可编程 FLASH（擦除次数：1000 次）
   —Boot 代码区具有独立的 Lock Bits，In-System-Programming 可通过 Boot 代码完成（自编程的概念）
   —512 字节在线可编程 EEPROM（寿命：100，000 次）
   —ATmega83：512 字节 SRAM
   —ATmega163：1024 字节 SRAM
   —程序加密位
4. 外围（Peripheral）特点
   —两个具有比较模式的可预分频（Prescale）8 位定时器/计数器
   —1 个可预分频、具有比较、捕捉功能的 16 位定时器/计数器
   —具有独立振荡器的实时时钟
   —3 个 PWM 通道
   —8 通道，10 位 ADC
      8 个单端通道
      7 个差分通道
      2 个具有可编程增益（1，10，200）的差分通道
   —I^2C 接口
   —可编程的 UART
   —主/从 SPI 接口
   —可编程的看门狗定时器（由片内振荡器生成）
   —片内模拟比较器
5. 特别的 MCU 特点
   —上电复位及可编程的掉电检测电路
   —可标度的内部 RC 振荡器
   —内外部中断源
   —4 种睡眠模式：空闲、ADC 噪声抑制、省电和掉电模式
6. I/O 和封装
   —32 个可编程的 I/O 脚
   —40 脚 PDIP、44 脚 PLCC 和 TQFP 封装
7. 工作电压

—2.7V-5.5V（ATmega83L/163L）
—4.0V-5.5V（ATmega83/163）

8. 速度
—0-4MHz（ATmega83L/163L）
—0-8MHz（ATmega83/163）

描述

ATmega83/163 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，ATmega83/163 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 ATmega83/163 结构方框图

ATmega83/163具有以下特点：8K/16K字节在线编程/自编程的FLASH，512字节EEPROM，512/1024字节存储器，32个通用I/0口，32个通用工作寄存器，实时时钟RTC，3个具有比较模式的灵活的定时器/计数器，内外中断源，8位的I^2C总线接口，8通道10位ADC（1个为差分输入，增益可调），可编程的看门狗定时器，SPI口以及四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SRAM、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。在ADC噪声抑制模式下CPU及其他I/O模块停止，只有ADC和异步定时器T/C2工作，以减小ADC转换过程当中的开关噪声。
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、通用编程器及 BOOT 程序进行编程。BOOT 程序可以利用任一接口来下载应用程序到应用 FLASH 区。通过将增强的 RISC 8位 CPU 与 FLASH 集成在一个芯片内，ATmega83/163 为许多嵌入式控制应用提供了灵活而低成本的方案。
ATmega83/163 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。

ATmega83 与 ATmega163 的比较

ATmega83具有8K字节的系统内可编程Flash和512字节的SRAM。ATmega163具有16K字节的系统内可编程 Flash 和 1024字节的 SRAM。二者都有 512 字节的EEPROM。由于ATmega163的FLASH大于8K，故而需要JMP和CALL指令。由于JMP指令占用两个字，因此，ATmega163的中断向量为两个字长，而ATmega83的中断向量为一个字长。
表1列出了两者的异同。

Table 1 存储器大小及特点

Feature	ATmega83	ATmega163
Flash	8K字节	16K字节
EEPROM	512字节	512字节
SRAM	512字节	1024字节
JMP/CALL	不支持	支持

中断向量	1个字	2个字

管脚配置

管脚定义

VCC、GND：电源

A 口（PA7..PA0）：

A 口为 ADC 的模拟输入。如果 AD 功能禁止，则 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。

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位双向I/O口，每一个管脚都有内部上拉电阻。C口的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。C 口的两个引脚还可以作为 T/C2 的振荡器引脚。在复位过程中，C 口为三态，即使此时时钟还未起振。

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。

/RESET： 复位输入。在振荡器起振的前提下，超过两个时钟的低电平将产生复位信号。

XTAL1： 振荡器放大器的输入端。

XTAL2： 振荡器放大器的输出端。

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。

时钟选择

在熔丝位的控制下，器件可有如下的时钟选项：

表2 时钟选项

时钟选项	CKSEL3..0
外部晶振/陶瓷振荡器	1111 - 1010
外部低频晶振	1001 - 1000
外部RC振荡器	0111 - 0101
内部RC振荡器	0100 - 0010
外部时钟	0001 – 0000

注意：“1” 表示未编程，“0” 表示编程。

不同的选项决定了不同的起动时间。

内部 RC 振荡器

内部 RC 振荡器的标称频率为 1 MHz。选择之后，器件无需外部元件就可以工作。

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

外部时钟

XTAL1 必须如下图连接。

图 3　外部时钟驱动配置

外部 RC 振荡器

对时间不敏感的应用可以使用下图的外部 RC 振荡器方法。

图 4　外部 RC 振荡器

时钟振荡器：

晶振可以直接连接到振荡器的引脚 PC6（TOSC1）和 PC7（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 5　ATmega83/163 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。

寄存器文件中的 6 个可以组成 3 个 16 位用于数据寻址的间接寻址寄存器指针，以提高地址运算能力。这些增加的功能寄存器为 X、Y 和 Z 寄存器。

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。

除了寄存器操作模式，通常的内存访问模式也适用于寄存器文件。这是因为 ATmega83/163 为寄存器文件分配了 32 个最低的数据空间地址（$00 - $1F），允许其象普通内存地址一样访问。

I/O 内存空间包括 64 个地址作为 CPU 外设的控制寄存器，T/C，以及其他 I/O 功能。I/O 内存可以直接访问，也可以作为数据地址（$20 - $5F）来访问。

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。程序存储器为可自编程的 FLASH 存储器。

相对跳转和相对调用指令可以直接访问 8K/16K 地址空间。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。

FLASH 空间分为两部分：BOOT 区（256~2048 字节）和应用程序区。两部分都有独立的锁定位。编程指令 SPM 只能在 BOOT 区执行。

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存储器之中，堆栈大小只受存储器数量的限制。用户应该在复位例程里就初始化 SP。SP 为可读写的 10/11

位堆栈指针。

512/1024 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。

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。

图 6　存储器配置

通用工作寄存器文件

图 7　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

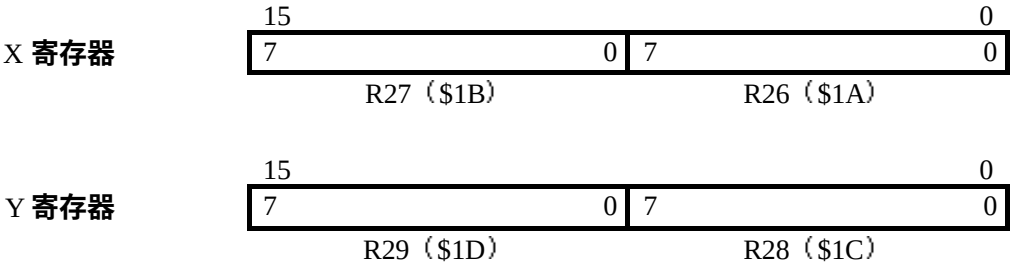
所有的寄存器操作指令都可以单指令的形式直接访问所有的寄存器。例外情况为 5 条涉及常数操作的指令：SBCI、SUBI、CPI、ANDI 和 ORI。这些指令只能访问通用寄存器文件的后半部分：R16 到 R31。

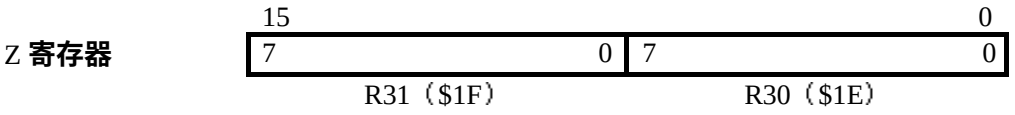
如图 7 所示，每个寄存器都有一个数据内存地址，将他们直接映射到用户数据空间的头 32 个地址。虽然寄存器文件的实现与存储器不同，这种内存组织方式在访问寄存器方面具有极大的灵活性。

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 7　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。ATmega83/163 还支持硬件乘法功能：有/无符号乘法及分数格式乘法。

自编程 FLASH

ATmega83/163 具有 8K/16K 字节可自编程/在线编程的 FLASH。因为所有的指令为 16（或 32）位宽，故尔 FLASH 结构为 4K/8K×16。FLASH 的擦除次数至少为 1000 次。PC 为 12/13 位宽，可以寻址到 4096/8192 个存储器地址。对 BOOT 区及相应锁定位的操作见后。常数表可以分布于整个存储区，用 LPM 指令寻址。

存储器

图 9　存储器配置

低 608/1120 个数据地址用于寻址寄存器文件，I/O 和内部数据存储器。起始的 96 个地址为寄存器文件+I/O，其后的 512/1024 个地址用于寻址存储器。
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，512/1024 字节的存储器可以被所有的寻址模式访问。
程序和数据寻址模式
ATmega83/163 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 10

双寄存器直接寻址：

图 11

I/O 直接寻址：

图 12

数据直接寻址：

图 13

带偏移的数据间接寻址：

图 14

数据间接寻址：

图 15

带预减的数据间接寻址：

图 16

带后加的数据间接寻址：

图 17

使用 LPM 的指令寻址常数：

图 18

间接程序寻址，IJMP 和 ICALL：

图 19

相对程序寻址，RJMP 和 RCALL：

图 20

EEPROM

ATmega83/163 包含 512 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 21 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 21 并行取指与指令执行

图 22 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 22 单时钟 ALU 操作

图 23 片内存储器访问周期

I/O 内存

表 3 ATmega83/163 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
$3B($5B)	GIMSK	通用中断屏蔽寄存器
$3A($5A)	GIFR	通用中断控制寄存器
$39($59)	TIMSK	T/C 中断屏蔽寄存器

$38($58)	TIFR	T/C 中断标志寄存器
$37($57)	SPMCR	存储程序存储器控制寄存器
$36($56)	I2CR	I^2C 控制寄存器
$35($55)	MCUCR	MCU 通用控制寄存器
$34($54)	MCUSR	MCU 通用状态寄存器
$33($53)	TCCR0	T/C0 控制寄存器
$32($52)	TCNT0	T/C0（8 位）
$31($51)	OSCCAL	振荡器标度寄存器
$30($50)	SFIOR	特殊功能 I/O 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)	TCCR2	T/C2 控制寄存器
$26($46)	ICR1H	T/C1 输入捕捉寄存器高字节
$25($45)	ICR1L	T/C1 输入捕捉寄存器低字节
$24($44)	TCNT2	T/C2（8 位）
$23($43)	OCR2	T/C2 输出比较寄存器
$22($42)	ASSR	异步模式状态寄存器
$21($41)	WDTCR	看门狗控制寄存器
$20($40)	UBRRHI	UART 波特率寄存器高字节
$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)	UCSRA	UART 控制/状态寄存器 A
$0A($2A)	UCSRB	UART 控制/状态寄存器 B
$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 通道选择器
$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)	I2DR	I^2C I/O 数据寄存器
$02($22)	I2AR	I^2C I/O（从机）地址寄存器
$01($21)	I2SR	I^2C I/O 状态寄存器
$00($20)	I2BR	I^2C I/O 比特率寄存器

AVRATmega83/163 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 寄存器的存储器地址写在括号中。

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)	-	-	-	-	-	**SP10** [1]	**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

注： 1. ATmega163

堆栈指针指向位于存储器的函数及中断堆栈。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。

复位和中断处理

ATmega83/163 有 17 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 4。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 4　复位与中断向量

向量号	程序地址		来　　源	定　　义
	163	83		
1	$000	$000	RESET	硬件管脚，上电复位和看门狗复位
2	$002	$001	INT0	外部中断 0
3	$004	$002	INT1	外部中断 1
4	$006	$003	TIMER2 COMP	T/C2 比较匹配
5	$008	$004	TIMER2 OVF	T/C2 溢出
6	$00A	$005	TIMER1 CAPT	T/C1 捕捉事件
7	$00C	$006	TIMER1 COMPA	T/C1 比较匹配 A
8	$01E	$007	TIMER1 COMPB	T/C1 比较匹配 B
9	$010	$008	TIMER1 OVF	T/C1 溢出
10	$012	$009	TIMER0 OVF	T/C0 溢出
11	$014	$00A	SPI，STC	串行传输结束
12	$016	$00B	UART，RX	UART 接收结束
13	$018	$00C	UART，UDRE	UART 数据寄存器空
14	$01A	$00D	UART，TX	UART 发送结束
15	$01C	$00E	ADC	ADC 转换结束
16	$01E	$00F	EE READY	EEPROM 准备好
17	$020	$010	ANA_COMP	模拟比较器
18	$022	$011	I^2C	I^2C 串行总线接口

设置中断向量地址最典型的方法如下：

ATmega83：

地址	标号	代码	注释
$000		rjmp RESET ;	复位
$001		rjmp EXT_INT0 ;	IRQ0
$002		rjmp EXT_INT1 ;	IRQ1
$003		rjmp TIM2_COMP ;	Timer2 Compare
$004		rjmp TIM2_OVF ;	Timer2 Overflow
$005		rjmp TIM1_CAPT ;	Timer1 Capture
$006		rjmp TIM1_COMPA ;	Timer1 CompareA
$007		rjmp TIM1_COMPB ;	Timer1 CompareB
$008		rjmp TIM1_OVF ;	Timer1 Overflow

```
$009                    rjmp TIM0_OVF ;          Timer0 Overflow
$00a                    rjmp SPI_STC; ;          SPI Transfer Complete
$00b                    rjmp UART_RXC ;          UART RX Complete
$00c                    rjmp UART_DRE ;               UDR Empty
$00d                    rjmp UART_TXC ;               UART TX Complete
$00e                    rjmp ADC ;               ADC Conversion Complete
$00f                    rjmp EE_RDY ;            EEPROM Ready
$010                    rjmp ANA_COMP ;          Analog Comparator
$011                    rjmp I2C ;               I2C Interrupt
;
$012        MAIN:        ldi r16, high(RAMEND);  主程序开始
$013                    out SPH, r16 ;                设置堆栈指针
$014                  ldi r16, low(RAMEND)
$015                  out SPL, r16
...                     ...                      ...
ATmega163：
$000                    rjmp RESET ;                  复位
$002                    rjmp EXT_INT0 ;          IRQ0
$004                    rjmp EXT_INT1 ;          IRQ1
$006                    rjmp TIM2_COMP ;         Timer2 Compare
$008                    rjmp TIM2_OVF ;          Timer2 Overflow
$00a                    rjmp TIM1_CAPT ;         Timer1 Capture
$00c                    rjmp TIM1_COMPA ;        Timer1 CompareA
$00e                    rjmp TIM1_COMPB ;        Timer1 CompareB
$010                    rjmp TIM1_OVF ;          Timer1 Overflow
$012                    rjmp TIM0_OVF ;          Timer0 Overflow
$014                    rjmp SPI_STC; ;          SPI Transfer Complete
$016                    rjmp UART_RXC ;          UART RX Complete
$018                    rjmp UART_DRE ;               UDR Empty
$01a                    rjmp UART_TXC ;               UART TX Complete
$01c                    rjmp ADC ;               ADC Conversion Complete
$01e                    rjmp EE_RDY ;            EEPROM Ready
$020                    rjmp ANA_COMP ;          Analog Comparator
$022                    rjmp I2C ;               I2C Interrupt
;
$024        MAIN:        ldi r16, high(RAMEND);  主程序开始
$025                    out SPH, r16 ;                设置堆栈指针
$026                  ldi r16, low(RAMEND)
$027                  out SPL, r16
...                     ...                      ...
```

复位源

ATmega83/163 **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。
- 电压检测复位。电源电压降低到某个特定值时。

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/JMP—相对跳转/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 24 为复位电路的逻辑图。表 5/6 定义了复位电路的时序和电参数。

图 24　复位逻辑

here　here　here

表 5　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}	上电复位电压门限（上升）	No BOD	1.0	1.4	1.8	V
		BOD	1.7	2.2	2.7	V
	上电复位电压门限（下降）	No BOD	0.4	0.6	0.8	V
		BOD	1.7	2.2	2.7	V
V_{RST}	复位引脚门限电压		-	-	.85V_{CC}	V
V_{BOT}	电平检测复位门限	BODLEVEL=1	2.6	2.7	2.8	V
		BODLEVEL=0	3.8	4.0	4.2	

注：除非电源电压低于 V_{POT}，否则上电复位不会发生。

表 6　复位延时选择

CKSEL[2:0]	起动时间 t_{TOUT}，V_{CC} = 2.7V	起动时间 t_{TOUT}，V_{CC} = 4.0V	推荐用法 [1]
000	4.2ms + 6 CK	5.8ms + 6 CK	外部时钟，电源快速上升
001	30μs + 6 CK	10μs + 6 CK	外部时钟，BOD 使能 [2] [3]
010	67ms + 16K CK	92ms + 16K CK	晶振
011	4.2ms + 16K CK	5.8ms + 16K CK	晶振，电源慢速上升
100	30μs + 16K CK	10μs + 16K CK	晶振，BOD 使能 [2] [3]
101	67ms + 1K CK	92ms + 1K CK	陶瓷/外时钟，电源慢速上升
110	4.2ms + 1K CK	5.8ms + 1K CK	陶瓷振荡器，电源快速上升
111	30μs + 1K CK	10μs + 1K CK	陶瓷振荡器，BOD 使能 [2] [3]

注意：1、CKSEL 只控制启动时间。上电时，实时时钟的启动时间典型值为 0.6ms。

2、或外部上电复位

3、实时时钟则为 50μs

表 4 显示了复位后的启动时间。从休眠唤醒时则只计算时钟计时的部分。看门狗溢出见表 5。

表 5　看门狗振荡器周期数

BODLEVEL	溢出时间	时钟数
未编程	4.2ms（V_{CC} = 2.7V）	1K
未编程	67ms（V_{CC} = 2.7V）	16K
已编程	5.8ms（V_{CC} = 4.0V）	4K
已编程	92ms（V_{CC} = 4.0V）	64K

注意：即使禁止电平检测（BODEN 不编程），BODLEVEL 也可用于选择启动时间。

看门狗的振荡频率与电压有关。芯片出厂时 CKSEL = 010。

上电复位

上电复位（POR）脉冲有片内检测电路产生。标称检测电平为 1.4V。当 V_{CC} 低于此电压时复位过程就会激活。

上电复位保证器件在上电时正确复位。外电压达到复位门限后激发一个延时计数器，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。延时时间可由 CKSEL 确定，见表 4。

图 25　MCU 起动，/RESET 与 V_{CC} 相连

图 26　MCU 启动，/RESET 由外电路控制

外部复位

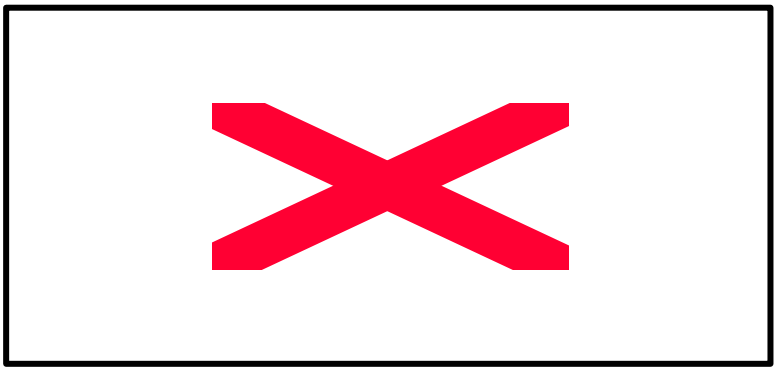
外部复位由外加于/RESET 引脚的低电平产生。大于 500ns 的复位脉冲将造成芯片复位。施加短脉冲不能保证可靠复位。当外加信号达到复位门限电压 V_{RST}（上升沿）时，t_{TOUT} 延时周期开始。然后，MCU 启动。

图 27　工作期间的外部复位

Brown-Out 检测

mega83/163 具有片内 BOD 检测电路用以监测 V_{CC}。**使用 BOD 功能时要用 74nF 到 100nF 电容对 V_{CC} 进行解耦。**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 4。BOD 触发电平由 BODLEVEL 控制为 2.7V（BODLEVEL 未编程）或 4.0V（BODLEVEL 已编程）。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 9μs，BOD 就起作用；而当触发电平为 2.7V 时时间延长到 21μs。

图 28　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 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 29　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)	-	-	-	-	**WDRF**	**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

外部复位时置位。上电复位或对其写“0”将使其清零。

PORF：上电复位标志

由上电复位置位。对其写“0”将使其清零。

如果要利用这些中断标志来识别复位条件，用户软件要尽早对其完成读和清零。如果在下一个复位发生前完成清零工作，则通过检查复位复位标志就可以找出复位源。

中断处理

ATmega83/163 有 2 个 8 位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。

中断发生后状态寄存器的保存由软件完成。

通用中断屏蔽寄存器—GIMSK

BIT	7	6	5	4	3	2	1	0
$3B($5B)	**INT1**	**INT0**	**INT2**	-	-	-	-	-
读/写	R/W	R/W	R/W	R	R	R	R	R
初始值	0	0	0	0	0	0	0	0

位 4..0：保留。读出值为 0。

INT1：外部中断 1 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（ISC11 和 ISC10）定义中断 1 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。

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。

INT2：外部中断 2 请求使能

当 INT2 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（ISC02）定义中断 2 是上升沿中断还是下降沿中断。即使管脚被定义为输出，中断仍可产生。

通用中断标志寄存器—GIFR

BIT	7	6	5	4	3	2	1	0
$3A($5A)	**INTF1**	**INTF0**	**INTF2**	-	-	-	-	-
读/写	R/W	R/W	R/W	R	R	R	R	R
初始值	0	0	0	0	0	0	0	0

位 4..0：保留。读出值为 0。

INTF1：外部中断标志 1

当 INT1 管脚有事件触发中断请求时，INTF1 置位（“1”）。如果 SREG 中的 I 及 GIMSK 中的 INT1 都为“1”，则 MCU 将跳转到中断地址$004。中断例程执行后，此标志被清除。另外，标志也可以通过对其写“1”来清除。

INTF0：外部中断标志 0

当 INT0 管脚有事件触发中断请求时，INTF0 置位（“1”）。如果 SREG 中的 I 及 GIMSK 中的 INT0 都为“1”，则 MCU 将跳转到中断地址$002。中断例程执行后，此标志被清除。另外，标志也可以通过对其写“1”来清除。

INTF2：外部中断标志 2

当 INT2 管脚有事件触发中断请求时，INTF2 置位（“1”）。如果 SREG 中的 I 及 GIMSK 中的 INT2 都为“1”，则 MCU 将跳转到中断地址$006。中断例程执行后，此标志被清除。另外，标志也可以通过对其写“1”来清除。

T/C 中断屏蔽寄存器—TIMSK

BIT	7	6	5	4	3	2	1	0
$39($59)	**TOIE1**	**OCIE1A**	**OCIE1B**	**TOIE2**	**TICIE1**	**TICIE2**	**TOIE0**	**OCIE0**
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

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（$0012）得到执行。

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（$00E）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（$010）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（$00A）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（$00C）将执行。

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（$008）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（**$016**）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（**$014**）将执行。**

T/C 中断标志寄存器—TIFR

BIT	7	6	5	4	3	2	1	0
$38($58)	**TOV1**	**OCF1A**	**OCF1B**	**TOV2**	**ICF1**	**OCF2**	**TOV0**	**OCF0**
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

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。**

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。**

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。**

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。**

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。**

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。**

外部中断

外部中断由 INT0/1/2 **触发。即使这些引脚配置为输出，只要中断使能，引脚电平的变化仍然可以引发中断。这是产生软件中断的一个方法。外部中断可以由上升/下降沿或低电平（除了** INT2**）触发—取决于** MCUCR（INT0/1）**和** EMCUCR（INT2）**。**

MCU控制寄存器—MCUCR

BIT	7	6	5	4	3	2	1	0
$35($55)	**SRE**	**SRW10**	**SE**	**SM1**	**ISC11**	**ISC10**	**ISC01**	**ISC00**
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

SRE：外部存储器使能

SRE **为“1”时，** MCU **可以访问外部存储器，** AD0-7（A **口），** A8-15（C **口），** ALE，/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，外部存储器无效，相关口可以当作普通** I/O **口使用。**

SRW10：外部存储器等待状态

当访问外部存储器时， SRW10 **用于设置等待周期。**

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：休眠模式

与 EMCUCR **的** SM0 **位一起用于选择休眠模式，如表** 6 **所示。**

表 6　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

ISC11，ISC10：中断检测控制 1 的位 1 和位 0

选择 INT1 **中断触发信号的边沿或电平，如下表所示：**

表 7　中断 1 **检测控制**

ISC11	ISC10	描述
0	0	低电平中断
0	1	任何逻辑变化都将产生中断请求
1	0	下降沿中断
1	1	上升沿中断

注：改变 ISC11/ISC10 **时，首先要禁止** INT1**（清除** GIMSK **的** INT1 **位），否则可能引发不必要的中断。**

ISC01，ISC00：中断检测控制 0 的位 1 和位 0

选择 INT0 **中断触发信号的边沿或电平，如下表所示：**

表 8　中断 1 **检测控制**

ISC01	ISC00	描述
0	0	低电平中断
0	1	任何逻辑变化都将产生中断请求
1	0	下降沿中断
1	1	上升沿中断

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

扩展的 MCU控制寄存器—EMCUCR

BIT	7	6	5	4	3	2	1	0
$36($56)	**SM0**	**SRL2**	**SRL1**	**SRL0**	**SRW01**	**SRW00**	**SRW11**	**ISC2**
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

SM0：休眠模式
与 SM1 **一起决定休眠模式。**
SRL2/1/0：外部存储器限制
对于不同的外部存储器区可以选择不同的等待状态。
SRW01/00/11：外部存储器等待状态选择
设置外部存储器的等待状态。
ISC2：中断检测控制位
为“0**”时，**INT2 **在下降沿触发；为“**1**”时，**INT2 **在上升沿触发。**

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和** INT2 **的沿中断可以使** MCU **脱离掉电模式。**
当使用 INT2 **的沿中断方式将** MCU **从掉电模式唤醒时，** MCU **可以记住沿变化直至唤醒。**
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**。**

定时器/计数器

ATmega83/163 **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/C2 **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/C2 **具有自己的分频器，而** T/C0 **和** T/C1 **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 30 T/C0 和 T/C1 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。寄存器 SFIOR 的 PSR10 用于复位预分频器，便于用户清楚了解预分频器的工作情况。

图 31 T/C2 的预分频器

T/C2 的时钟源称为 PCK2。缺省地，PCK2 与系统主时钟连接。若置位 ASSR 的 AS2，T/C2 将由 TOSC1 异步驱动，使得 T/C2 可以作为一个实时时钟。如果 AS2 置位，则 TOSC1 和 TOSC2 即可外接一个时钟晶振，作为 T/C2 的独立时钟源。T/C2 对 32.768K 晶振做了特殊的优化。T/C2 的时钟频率不能高于 CPU 频率的 1/4，也不能高于 256KHz。置位 SFIOR 的 PSR2 将复位其分频器。

特殊功能 IO 寄存器—SFIOR

BIT	7	6	5	4	3	2	1	0
$30($50)	-	-	-	-	-	-	**PSR2**	**PSR10**
读/写	R	R	R	R	R	R	R/W	R/W
初始值	0	0	0	0	0	0	0	0

位 7..2：保留

PSR2：T/C2 分频器复位

置位后 T/C2 的预分频器复位。操作完成后硬件对其清零。软件写零并不真正执行清零工作。如果 T/C2 由内部 CPU 时钟驱动，读取这一位的结果为零。如果 T/C2 工作于异步模式（由自己的晶振驱动），则这一位将一直保持为“1”，直到异步操作完成。

PSR10：T/C1/0 分频器复位

置位后 T/C2 的预分频器复位。操作完成后硬件对其清零。软件写零并不真正执行清零工作。读取这一位的结果为零。

8 位 T/C0 和 T/C2

图 32 为 T/C0 的框图。

图 32 T/C0 工作框图

图 33 T/C2 工作框图

T/C0 的时钟可以选择 CK、预分频的 CK 或由外部引脚输入。

T/C2 的时钟可以选择 CK、预分频的 CK 或由外部引脚 TOSC1 输入。

两个 T/C 可以通过 TCCR0/2 控制寄存器而停止。

TIFR 为状态标志寄存器，TCCR0/2 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 PWM 功能。

T/C0 控制寄存器—TCCR0

BIT	7	6	5	4	3	2	1	0
$33($53)	**FOC0**	**PWM0**	**COM01**	**COM00**	**CTC0**	**CS02**	**CS01**	**CS00**
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 控制寄存器—TCCR2

BIT	7	6	5	4	3	2	1	0
$27($47)	**FOC2**	**PWM2**	**COM21**	**COM20**	**CTC2**	**CS22**	**CS21**	**CS20**
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

FOC0/2：强迫输出比较

写“1”后，按照 COMn1/0 **的设置，在引脚** PB0（T/C0）**和** PB1（T/C1）**强迫产生一次比较匹配输出。如果** COM **的值与** FOC **的值在同一写入周期更新，本次功能将被忽略。由于** FOC **置位产生的比较匹配不会产生中断。读** FOC **的返回值总为零。在** PWM **模式下，**FOC **没有任何意义。**

PWM0/PWM2：PWM 使能

置位后使能 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）**和** PB1（PC2）**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** 9**。**

表 9　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	CK0
0	1	0	CK0/8
0	1	1	CK0/64

1	0	0	CK0/265
1	0	1	CK0/1024
1	1	0	**外部引脚** T2，**下降沿**
1	1	1	**外部引脚** T2，**上升沿**

表 13 T/C2 **预分频选择**

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

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/C0 **使用了外部引脚模式**（[CS02，CS01] = [1，1]），**则不论引脚配置如何**，PB0（T0）**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
$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 **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
$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 **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/C **可作为上升/下降或回卷（从**$FF **直接回到** 0）**功能。**

选择上升/下降模式后，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 PB0（OC0/PWM0）和 PB1（OC2/PWM2）。选择回卷模式后，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，输出频率为上升/下降模式的 2 倍。

PWM 模式（上升/下降及回卷）

通过对 CTC0/2 的设置，可以得到两种 PWM 模式。

若 CTC0/2 为零，则 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。

若 CTC0/2 为“1”，则 T/Cn 作为回卷计数器，从 0 记数到$FF，然后直接回到 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

CTCn	COMn1	COMn0	引脚动作	频率
0	0	0	不用作 PWM 功能	
0	0	1	不用作 PWM 功能	
0	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）	$f_{TCK0/2}$/510
0	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）	$f_{TCK0/2}$/510
1	0	0	不用作 PWM 功能	
1	0	1	不用作 PWM 功能	
1	1	0	比较匹配时清除 Ocn；溢出时置位 Ocn	$f_{TCK0/2}$/256
1	1	1	溢出时置位 Ocn；比较匹配时清除 Ocn	$f_{TCK0/2}$/256

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 34　失步的 OCR 锁存（上升/下降模式）

图 35　失步的 OCR 锁存（回卷模式）

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

在上升/下降模式下，当计数器达到$00 时将置位 TOV0/TOV2。而在回卷模式下发生的中断与定时/记数模式下的中断是完全一样的。

异步状态寄存器—ASSR

BIT	7	6	5	4	3	2	1	0

$26($46)	-	-	-	-	AS2	TCN2UB	OCR2UB	TCR2UB
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

AS2：异步 T/C2

当 AS2 置位时，T/C2 由 TOSC1 驱动。PD4 和 PD5 连接到晶体振荡器，不能用作普通 I/O。若 AS2 为“0”，则 T/C2 由内部系统时钟驱动。这一位变化时有可能破坏 TCNT2、OCR2 和 TCCR2。

TCN2UB：T/C2 更新忙

T/C2 工作于异步模式时，写 TCNT2 将引起 TCN2UB 置位。当 TCNT2 从暂存寄存器更新完毕后 TCN2UB 由硬件清零。TCN2UB 为“0”表明 TCNT2 可以写入新值了。

OCR2UB：输出比较寄存器更新忙

T/C2 工作于异步模式时，写 OCR2 将引起 OCR2UB 置位。当 OCR2 从暂存寄存器更新完毕后 OCR2UB 由硬件清零。OCR2UB 为“0”表明 OCR2 可以写入新值了。

TCR2UB：T/C2 控制寄存器更新忙

T/C2 工作于异步模式时，写 TCCR2 将引起 TCR2UB 置位。当 TCCR2 从暂存寄存器更新完毕后 TCR2UB 由硬件清零。TCR2UB 为“0”表明 TCCR2 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 TCNT2，OCR2 和 TCCR2 进行读取的机制是不同的。读到的 TCNT2 为实际的值，而 OCR2 和 TCCR2 则是从暂存寄存器中读取的。

T/C2 的异步操作

T/C2 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 TCNT2、OCR2、TCCR2 数据的损毁。安全的步骤应该是：
  1、关闭 T/C2 的中断 OCIE2 和 TOIE2。
  2、设置 AS2 以选择合适的时钟源。
  3、TCNT2，OCR2 和 TCCR2 写入新的数值。
  4、等待 TCN2UB，OCR2UB 和 TCR2UB 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 TCNT2，OCR2 和 TCCR2 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 TCNT2 不会干扰写 OCR2。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/C2 作为 MCU 的唤醒条件，则在 TCNT2，OCR2 和 TCCR2 更新结束之前不能进入省电模式，否则 MCU 可能会在 T/C2 设置生效之前进入休眠貌模式。这对于用 T/C2 的比较匹配中断唤醒 MCU 尤其重要。因为在更新 OCR2 或 TCNT2 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/C2 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 TCNT2，OCR2 和 TCCR2 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/C2 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/C2。
- 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 36 为 T/C1 的框图。

图 36　　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 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/C1 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/非中心对称的 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 37。

图 37　　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**	**FOC1A**	**FOC1B**	**PWM11**	**PWM10**
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

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 14。

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 14　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 18。

FOC1A：强迫输出比较 A

写“1”后，按照 COM1A0/1 的设置，在引脚 PD5 强迫产生一次比较匹配输出。如果 COM 的值与 FOC 的值在同一写入周期更新，本次功能将被忽略。强迫输出比较功能使得 MCU 可以不用等待比较匹配的发生就变换引脚的输出。由于 FOC 置位产生的比较匹配不会引发中断，也不会清除 TCNT1。读 FOC 的返回值总为零。在 PWM 模式下，FOC 没有定义。

FOC1B：强迫输出比较 B

写“1”后，按照 COM1B0/1 的设置，在引脚 PE2 强迫产生一次比较匹配输出。如果 COM 的值与 FOC 的值在同一写入周期更新，本次功能将被忽略。强迫输出比较功能使得 MCU 可以不用等待比较匹配的发生就变换引脚的输出。由于 FOC 置位产生的比较匹配不会引发中断，也不会清除 TCNT1。读 FOC 的返回值总为零。在 PWM 模式下，FOC 没有任何意义。

PWM11，PWM10：PWM 选择

表 15　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 16 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。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
$25($45)	**MSB**							
$24($44)								**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 17 TOP **值及** PWM **频率**

CTC1	PWM11	PWM10	WPM分辨率	TOP值	频率
0	0	1	8 位	$00FF（255）	f_{TCK1}/510
0	1	0	9 位	$01FF（511）	f_{TCK1}/1022
0	1	1	10 位	$03FF（1023）	f_{TCK1}/2046
1	0	0	8 位	$00FF（255）	f_{TCK1}/256
1	1	1	9 位	$01FF（511）	f_{TCK1}/512
1	1	0	10 位	$03FF（1023）	f_{TCK1}/1024

如上表所示，PWM **工作于** 8/9/10 **位分辨率。**OCR1A/B **中未使用的位由硬件自动写为零。**

表 18 PWM **模式下的比较** 1 **模式选择**

CTC1	COM1X1	COM1X0	OC1
0	0	0	**不用作** PWM **功能**
0	0	1	**不用作** PWM **功能**

0	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）
0	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）
1	0	0	不用作 PWM 功能
1	0	1	不用作 PWM 功能
1	1	0	比较匹配时清除 Ocn；溢出时置位 Ocn
1	1	1	溢出时置位 Ocn；比较匹配时清除 Ocn

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 38　失步的 OCR1 锁存

图 38　失步的 OCR1 锁存（回卷模式）

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 19　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

在上升/下降模式下，当计数器达到$0000 时将置位 TOV1。而在回卷模式下发生的中断与定时/记数模式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。ATmega83/163 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 40　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 20　　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)	-	-	-	-	-	-	-	**EEAR8**
$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	R	R	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** ATmega83/163 **的** 512 **字节的** EEPROM**。**EEPROM **的地址是线性的，从** 0 **到** 511**。**

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 ATmega83/163 和外设之间进行高速的同步数据传输。ATmega83/163 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
- 7 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）
- 具有双速模式

图 41　SPI 方框图

主从 CPU 的 SPI 连接见图 42。PB7（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 PB5（MOSI）移出，从从机的 PB5（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 42　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 21　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 43 SPI 传输格式（CPHA=0，DORD=0）

图 44 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 43 和 44。

SPR1，SPR0：SPI 时钟速率选择位

表 22 SCK 和时钟频率之间的关系

SPI2X	SPR1	SPR0	SCK
0	0	0	$f_{CL}/4$
0	0	1	$f_{CL}/16$
0	1	0	$f_{CL}/64$
0	1	1	$f_{CL}/128$
1	0	0	$f_{CL}/2$
1	0	1	$f_{CL}/8$
1	1	0	$f_{CL}/32$
1	1	1	$f_{CL}/64$

SPI 状态寄存器—SPSR

BIT	7	6	5	4	3	2	1	0
$0E($2E)	**SPIF**	**WCOL**	-	-	-	-	-	**SPI2X**
读/写	R/W	R/W	R	R	R	R	R	R/W
初始值	0	0	0	0	0	0	0	0

位 5..1：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。

SPI2X：双速

置位后 SPI 的速度加倍。若作为主机，则 SPI 频率可达 CPU 频率的一半。若为从机，只能保证 $f_{CL}/4$。

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

UARTs

ATmega83/163 有两个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不同的波特率（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
- 双速 UART 模式

数据发送

图 45 为 UART 发送器的原理图。

图 45　UART 发送器

把待发送的数据写入 UART 数据寄存器 UDRn 将起动数据发送。在如下情况下 UDRn 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 UDRn 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，UDRn 的数据才送入移位寄存器。

如果 10（11）位收发器移位寄存器为空，UDRn 中的数据将传送到移位寄存器。此时 UDREn 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 CHR9n 置位），则 UCSRnB 中的 TXB8 将送到移位寄存器的位 9。

UART 首先在 TXDn 引脚送出起始位，然后是数据，低位在前。如果 UDRn 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，UDREn 置位，并一直保持到有新数据写入 UDRn。如果 UDRn 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 TXCn 置位。

UCSRnB 的 TXENn 使能 UART 发送器。当 TXENn 为“0”时，PD1（UART0）或 PB3（UART1）可用作普通 I/O 口。当 TXENn 为“1”时，UART 输出自动连接到 PD1（UART0）或 PB3（UART1），强迫其为输出，而不管方向寄存器的设置。要注意的是，PB3 还作为模拟比较器的一个输入。两个功能不能同时实现。

数据接收

图 46　UART 接收器

接收器的前端以 16 倍于波特率的频率对 RXDn 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 RXDn 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 47 说明了采样过程。

图 47　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，UCSRnA 中的帧错误标志位 FEn 置位。在读 UDRn 之前，用户应该检查 FEn。

不管停止位有效与否，接收到的数据都将被送入 UDRn，UCSRnA 的 RXCn 置位。UDRn 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 UDRn 时，访问的是接收 UDRn，而在写 UDRn 时，访问的是发送 UDRn。如果数据格式为 9 位，则 UCSRnB 的 RXB8n 在数据传送到 UDRn 时加载到发送移位寄存器的位 9。

如果在读取 UDRn 之前，UART 又接收到一个数据，则 UCSRnB 的 ORn 置位。这表明数据无法转移到 UDRn 而丢失了。ORn 一直保持到 UDRn 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 UDRn 时首先检测 ORn 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 或 PB2，强迫其作为输入而不管方向寄存器的设置。此时，PORTD0 与 PORTB2 仍然可以选择内部上拉是否有效。

PB2（UART1）还用作模拟比较器的一个输入。因此，如果要使用模拟比较器的话，UART1 就无法使用了。

当 UCSRnB 的 CHR9n 为“1”时，收发的数据格式为 9 位。要发送的第 9 位是 UCSRnB 的 TXB8n（要在写 UDRn 之前设置），而接收到的第 9 位是 RXB8n。

处理器通讯模式

多处理器通讯模式实现了多个从机与一个主机之间的通讯。首先，一个地址字节用来找出哪一个 MCU 是被寻址的器件。这个特定的 MCU 将可以接收后续的数据，而未被寻址的 MCU 则忽略这些数据。

主机 MCU 的发送模式必须为 9 位（置位 UCSRnB 的 CHR9n）。当第 9 位为“1”时，表明此次发送的字节为地址；否则，表明发送的是数据。

从机可以工作于 8 位或 9 位模式。8 位模式时，如果停止位为“1”，则表明接收到的是地址，否则为数据。9 位模式时，停止位始终为“1”。第 9 位用来识别是地址还是数据：为“1”时表明接收到的是地址，为“0”时表明接收到的是数据。

下述过程用来实现多机通讯：

1、所有的从机处于多机通讯模式（置位 UCSRnA 的 MPCMn）

2、主机发送地址字节，所有的从机都将接收到次字节，且 UCSRnA 的 RXCn 置位。

3、从机读取 UDRn 并与自己的地址进行比较。如果相同，则它清除 MPCMn 以接收后续数据；否则继续等待下一个地址字节。

4、接收从机将按照普通单机通讯模式接收数据。但是如果接收模式为 8 位，由于停止位为“0”，所以在接收到数据的同时，帧错误 FEn 亦将置位。对于其他 MPCMn 保持置位的 MCU 则简单地忽略数据字节，不产生任何动作。

5、最后一个字节发送完毕后，流程回到第二步。

UART 控制

UART0 数据寄存器—UDR0

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

UART1 数据寄存器—UDR1

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

UDRn 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 UDRn 时，访问的是接收 UDRn，而在写 UDRn 时，访问的是发送 UDRn。

UART0 控制和状态寄存器—UCSR0A

BIT	7	6	5	4	3	2	1	0
$0B($2B)	**RXC0**	**TXC0**	**UDRE0**	**FE0**	**OR0**	**-**	**U2X0**	**MPCM0**
读/写	R	R	R	R	R	R	R/W	R/W
初始值	0	0	1	0	0	0	0	0

UART0 控制和状态寄存器—UCSR1A

BIT	7	6	5	4	3	2	1	0
$0B($2B)	**RXC1**	**TXC1**	**UDRE1**	**FE1**	**OR1**	**-**	**U2X1**	**MPCM1**
读/写	R	R	R	R	R	R	R/W	R/W
初始值	0	0	1	0	0	0	0	0

位 2：保留

RXC0/1：UART 接收结束

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

TXC0/1：UART 发送结束

TXCn 置位表示数据已经从发送移位寄存器发送出去，且 UDRn 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 TXCIEn 已置位，则 TXCn 置位将引发发送结束中断。进入中断例程后 TXCn 自动清零，或者用户可以对其写“1”以达到清零的目的。

UDREn：UART数据寄存器空

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

当 UDRIEn **置位，则只要** UDREn **为“**1**”，** UART **发送结束中断就可以执行。写** UDRn **将复位** UDREn**。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** UDRn **以清除** UDREn**，否则中断将连续发生。**

复位后 UDREn **的初始值为“**1**”，表明发送器就绪。**

FE0/1：帧错误

MCU **检测到帧错误（如：检测到停止位为“**0**”）时** FEn **置位。**

当检测到数据停止位为“1**”时** FEn **复位。**

OR0/1：过速

如果 UDRn **未读，而新的数据又已进入移位寄存器，则** ORn **置位。**

UDRn **数据移入后** ORn **清零。**

U2X0/1：双速模式

置位后， UART **的速度将比普通模式快一倍。**

MPCM0/1：多处理器通讯模式

当从机等待地址字节时应置位此位；被寻址后要清除此位以接收数据。

UART0 控制和状态寄存器—UCSR0B

BIT	7	6	5	4	3	2	1	0
$0A($2A)	**RXCIE0**	**TXCIE0**	**UDRIE0**	**RXEN0**	**TXEN0**	**CHR90**	**RXB80**	**TXB80**
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

UART1 控制和状态寄存器—UCSR1B

BIT	7	6	5	4	3	2	1	0
$0A($2A)	**RXCIE1**	**TXCIE1**	**UDRIE1**	**RXEN1**	**TXEN1**	**CHR91**	**RXB81**	**TXB81**
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

RXCIE0/1：接收结束中断使能

TXCIE0/1：发送结束中断使能

UDRIE0/1：UART数据寄存器空中断使能

RXEN0/1：接收使能

使能 UART **接收。接收禁止将导致** TXC0/1**，** OR0/1 **和** FE0/1 **无法置位，同时也不能复位已经置位的标志位。**

TXEN0/1：发送使能

TXEN0/1 **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。**

CHR90/1：9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** RXB80/TXB80、RXB81/TXB81 **控制。第** 9 **位字符可用作停止位或奇偶校验。**

RXB80/1：接收的第9位

接收到的字符的第 9 **位**

TXB80/1：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**（UBRRHI **和** UBRRn）**的内容**（0-4095）

此公式不使用于双速模式。

表 23 **给出了在某些晶振下波特率对应的** UBRR **的值。**

表 23　不同频率下 UBRR **的设定值**

波特率	1MHz	%误差	1.8432 MHz	%误差	2MHz	%误差
2400	UBRR = **25**	0.2	**47**	0.0	**51**	0.2
4800	12	0.2	**23**	0.0	**25**	0.2
9600	6	7.5	**11**	0.0	**12**	0.2
14400	3	7.8	**7**	0.0	8	3.7
19200	2	7.8	**5**	0.0	6	7.5
28800	1	7.8	**3**	0.0	3	7.8
38400	1	22.9	**2**	0.0	2	7.8
57600	0	7.8	**1**	0.0	1	7.8
76800	0	22.9	1	33.3	1	22.9
115200	0	84.3	**0**	0.0	0	7.8

波特率	3.2768 MHz	%误差	3.6864 MHz	%误差	4MHz	%误差
2400	UBRR = 84	0.4	**95**	0.0	**83/163**	0.2
4800	42	0.8	**47**	0.0	**51**	0.2
9600	20	1.6	**23**	0.0	**25**	0.2
14400	13	1.6	**15**	0.0	16	2.1
19200	10	3.1	**11**	0.0	**12**	0.2
28800	6	1.6	**7**	0.0	8	3.7
38400	4	6.3	**5**	0.0	6	7.5
57600	3	12.5	**3**	0.0	3	7.8
76800	2	12.5	**2**	0.0	2	7.8
115200	1	12.5	**1**	0.0	1	7.8

波特率	7.3728 MHz	%误差	8MHz	%误差	9.216MHz	%误差
2400	**UBRR = 191**	0.0	**207**	0.2	**299**	0.0
4800	**95**	0.0	**83/163**	0.2	**119**	0.0
9600	**47**	0.0	**51**	0.2	**59**	0.0
14400	**31**	0.0	34	0.8	**39**	0.0
19200	**23**	0.0	*25*	0.2	**29**	0.0
28800	**15**	0.0	16	2.1	**19**	0.0
38400	**11**	0.0	**12**	0.2	**14**	0.0

57600	**7**	0.0	8	3.7	**9**	0.0
76800	**5**	0.0	6	7.5	7	6.7
115200	**3**	0.0	3	7.8	**4**	0.0

UART0/1 波特率高字节寄存器—UBRRHI

BIT	7	6	5	4	3	2	1	0
$20($40)	**MSB1**			**LSB1**	**MSB0**			**LSB0**
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

波特率寄存器为 12 **位寄存器。高** 4 **位位于** UBRRHI**。**UART0 **与** UART1 **共享此寄存器。**0 —~~3~~ **属于** UART0，4—~~7~~ **属于** UART1**。**

UART0 波特率低字节寄存器—UBRR0

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

UART1 波特率低字节寄存器—UBRR1

BIT	7	6	5	4	3	2	1	0
$00($20)	**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

双速传输

通过置位 UCSRnA **的** U2X，83/163 **提供了另一个** UART **模式以允许用户进行双速通讯。接收器的前端以** 8 **倍于波特率的频率对** RXDn **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** 4，5 **和** 6 **对** RXDn **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** 4，5 **和** 6**。两个或** 3 **个相同的值被认为是当前位的值。图** 48 **说明了采样过程。**

图 48　采样接收到的数据

波特率产生器

波特率计算公式为：

$$BAUD = \frac{f_{CK}}{8(BURR + 1)}$$

式中，BAUD = **波特率，**

f_{CK} = **晶振时钟频率**

UBRR = UART **波特率寄存器（**UBRRHI **和** UBRRn**）的内容（**0-4095**）**

表 24 **给出了在某些晶振下波特率对应的** UBRR **的值。**

表 24　不同频率下 UBRR **的设定值**

波特率	1MHz	%误差	1.8432 MHz	%误差	2MHz	%误差
2400	UBRR = **5 1**	0.2	**95**	0.0	**103**	0.2
4800	**25**	0.2	**47**	0.0	**51**	0.2
9600	**12**	0.2	**23**	0.0	**25**	0.2
14400	8	3.7	**15**	0.0	16	2.1
19200	6	7.5	**11**	0.0	12	0.2
28800	3	7.8	**7**	0.0	8	3.7
38400	2	7.8	**5**	0.0	6	7.5
57600	1	7.8	**3**	0.0	3	7.8
76800	1	22.9	2	33.3	2	7.8
115200	0	84.3	**1**	0.0	1	7.8
230400	-	-	**0**	0.0	0	84.3

波特率	3.2768 MHz	%误差	3.6864 MHz	%误差	4MHz	%误差
2400	UBRR = 1 7 0	0.2	**191**	0.0	**207**	0.2
4800	84	0.4	**95**	0.0	**103**	0.2
9600	42	0.8	**47**	0.0	**51**	0.2
14400	27	1.6	**31**	0.0	34	0.8
19200	20	1.6	**23**	0.0	**25**	0.2
28800	13	1.6	**15**	0.0	16	2.1
38400	10	3.1	**11**	0.0	**12**	0.2
57600	6	1.6	**7**	0.0	8	3.7
76800	4	6.2	**5**	0.0	6	7.5
115200	3	12.5	**3**	0.0	3	7.8
230400	1	12.5	**1**	0.0	1	7.8
460800	0	12.5	**0**	0.0	0	7.8
912600	-	-	**-**	-	0	84.3

波特率	7.3728 MHz	%误差	8MHz	%误差	9.216MHz	%误差
2400	**UBRR = 3 8 3**	0.0	**416**	0.1	**479**	0.0
4800	**191**	0.0	**207**	0.2	**239**	0.0
9600	**95**	0.0	**103**	0.2	**119**	0.0
14400	**63**	0.0	68	0.6	**79**	0.0
19200	**47**	0.0	**51**	0.2	**59**	0.0
28800	**31**	0.0	34	0.8	**39**	0.0
38400	**23**	0.0	**25**	0.2	**29**	0.0
57600	**15**	0.0	16	2.1	**19**	0.0
76800	**11**	0.0	**12**	0.2	**14**	0.0
115200	**7**	0.0	8	3.7	**9**	0.0
230400	**3**	0.0	3	7.8	**4**	0.0
460800	**1**	0.0	1	7.8	2	20.0
912600	**0**	0.0	0	7.8	0	20.0

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 49 所示。

图 49 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	AINGB	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

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

置位后内部 1.22±0.05V 能隙基准源取代输入管脚连接到 AIN0。否则，PB2 连接到比较器的正输入端。

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 25 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。

模拟比较器的输入引脚（PB2 和 PB3）还作为 UART1 的 TXD1 及 RXD1 使用。如果 UART1 功能使能，则 UART1 将按照其自身设定改变 DDRB 的值，即使此时模拟比较器的功能已经打开。因此，要避免两种功能同时使用。

内部电压基准

ATmega83/163 有一个内部基准电压源，标称值为 1.22V。此基准电压源用于电压检测及模拟比较器。

基准电压使能信号及启动时间

基准电压有一个启动时间 TBD。为了节省功耗，只有在下述情况下基准电压才会打开：

1、BOD 使能（熔丝位 BODEN 被编程）

2、基准源连接到模拟比较器（ACSR 的 AINBG 置位）

因此，如果 BOD 禁止，则 AINBG 置位后，用户需要等待基准电压启动，然后才能利用模拟比较器的输入。基准电压大概要消耗 10μA 的电流。

外部存储器接口

利用外部存储器接口，用户可以操作存储器件存储器或 FLASH，以及外设 LCD、ADC、DAC 等。控制接口的寄存器有 MCUCR 及 EMCUCR。

MCU控制寄存器—MCUCR

BIT	7	6	5	4	3	2	1	0
$35($55)	**SRE**	**SRW10**	**SE**	**SM1**	**ISC11**	**ISC10**	**ISC01**	**ISC00**
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

扩展 MCU控制寄存器—EMCUCR

BIT	7	6	5	4	3	2	1	0
$36($56)	**SM0**	**SRL2**	**SRL1**	**SRL0**	**SRW01**	**SRW00**	**SRW11**	**ISC20**
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

SRE：外部存储器使能

SRE 为“1”时，MCU 可以访问外部存储器，AD0-7（A 口），A8-15（C 口），ALE，/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，外部存储器无效，相关口可以当作普通 I/O 口使用。

SRL2/1/0：等待状态页面限制

Mega83/163 可以为不同的外部存储器设置不同的等待状态。外部存储器可以被区分为 2 页。3 位 SRL 用来将存储区分割为不同的页面。具体参见表 27 和图 50。SRL 的缺省值为 0，代表整个外部存储区为一个整体，此时的等待状态仅由 SRW11 和 SRW10 控制。

SRW11、SRW10：外部存储器高页面等待状态

SRW11/10 用来控制外部存储器高页面的等待状态。具体参见表 26。当 SRL 为全“0”，这两位将控制所有外部存储器的等待状态。

SRW01、SRW00：外部存储器低页面等待状态

SRW01/00 用来控制外部存储器低页面的等待状态。具体参见表 26。当 SRL 为全“0”，这两位将控制所有外部存储器的等待状态。

表 25　等待状态

SRWn1	SRWn0	等待状态
0	0	无等待状态
0	1	读/写时等待一个时钟

1	0	读/写时等待二个时钟
1	1	读/写时等待二个时钟，输出新地址时等待一个时钟

n = 0 或 1

请参看图 51-54 以获取进一步信息。

表 27　页面限制

SRL2	SRL1	SRL0	页 面 限 制
0	0	0	低页面 = N/A 高页面 = $0460 - $FFFF
0	0	1	低页面 = $0460 - $1FFF 高页面 = $2000 - $FFFF
0	1	0	低页面 = $0460 - $4FFF 高页面 = $4000 - $FFFF
0	1	1	低页面 = $0460 - $5FFF 高页面 = $6000 - $FFFF
1	0	0	低页面 = $0460 - $7FFF 高页面 = $8000 - $FFFF
1	0	1	低页面 = $0460 - $9FFF 高页面 = $A000 - $FFFF
1	1	0	低页面 = $0460 - $BFFF 高页面 = $C000 - $FFFF
1	1	1	低页面 = $0460 - $DFFF 高页面 = $E000 - $FFFF

图 50　带页面选择的外部存储器

图 51　没有等待状态的外部数据存储器周期（SRWn1 = 0，SRWn0 = 0）

注意：　n = 1 或 0

位于 T4 的 ALE 仅当下一个指令访问 RAM（内/外部）时才出现。ALE 出现后，数据和地址总线才会于 T4 改变。

图 52　SRWn1 = 0，SRWn0 = 1 时的外部数据存储器周期

注意：　n = 1 或 0

位于 T5 的 ALE 仅当下一个指令访问 RAM（内/外部）时才出现。ALE 出现后，数据和地址总线才会于 T5 改变。

图 53　SRWn1 = 1，SRWn0 = 0 时的外部数据存储器周期

注意：　n = 1 或 0

位于 T6 的 ALE 仅当下一个指令访问 RAM（内/外部）时才出现。ALE 出现后，数据和地址总线才会于 T6 改变。

图 54　SRWn1 = 1，SRWn0 = 1 时的外部数据存储器周期

注意：　n = 1 或 0

位于 T7 的 ALE 仅当下一个指令访问 RAM（内/外部）时才出现。ALE 出现后，数据和地址总线才会于 T7 改变。

使用外部存储器接口

存储器接口包括：

- 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

外部存储器使能通过 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”，则外部存储器禁止，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 仅在访问外部存储器时有效。

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

图 55 为 AVR 与外部存储器的连接图。所用锁存器为 8 位，G 为高时选通。

图 55　AVR 与外部存储器的连接

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存储器的地址/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 28　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 56　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	OC0/T0（T/C0 的比较输出/计数器 0 输入）
PB1	OC2/T1（T/C2 的比较输出/计数器 1 输入）
PB2	RXD1（UART1 输入）/AIN0（模拟比较器正输入端）
PB3	TXD1（UART1 输出）/AIN1（模拟比较器负输入端）
PB4	/SS（SPI 从机选择）
PB5	MOSI（SPI 主机输出/从机输入）
PB6	MISO（SPI 主机输入/从机输出）
PB7	SCK（串行时钟）

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

- SCK—PB7
  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。**
- MISO—PB6
  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—PB5
  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。**
- /SS—PB4
  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。**
- TXD1/AIN1—PB3

AIN1，模拟比较器负输入端。

TXD1，UART1 的数据发送引脚。UART1 发送器使能时，引脚自动配置为输出。

- RXD1/AIN0—PB2

  AIN0，模拟比较器正输入端。

  RXD1，UART1 的数据发送引脚。UART1 发送器使能时，引脚自动配置为输入。如果需要上拉，则还需置位 PORTB2。

- OC2/T1—PB1

  T1，T/C1 做计数器时的计数事件输入。

  OC2，T/C2 的输出比较匹配输出信号。输出比较匹配功能使能后，此引脚需配置为输出。

- OC0/T0—PB0

  T0，T/C0 做计数器时的计数事件输入。

  OC0，T/C0 的输出比较匹配输出信号。输出比较匹配功能使能后，此引脚需配置为输出。

B 口示意图

图 57　B 口示意图（PB0 和 PB1）

图 58　B 口示意图（PB2）

图 59　B 口示意图（PB3）

图 60　B 口示意图（PB4）

图 61　B 口示意图（PB5）

图 62　B 口示意图（PB6）

图 63　B 口示意图（PB7）

C 口

C 口是 8 位双向 I/O 口。

C 口有 3 个 I/O 地址：数据寄存器—PORTC，$15（$35），数据方向寄存器—DDRC，$14（$34）和输入引脚—PINC，$13（$33）。PORTC 和 DDRC 可读可写，PINC 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。

C 口具有与访问外部 SRAM 相关的第二功能。此时 C 可以配置为高字节地址输出。这一功能由 SRE 控制。

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

C 口数据方向寄存器—DDRC

BIT	7	6	5	4	3	2	1	0
$14($34)	DDC7							DDC0
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

C 口输入引脚地址—PINC

BIT	7	6	5	4	3	2	1	0
$13($33)	PINC7							PINC0
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

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

C 口用作通用数字 I/O

作为通用数字 I/O 时，C 口的 8 个管脚具有相同的功能。

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

表 23　C 口的配置

DDCn	PORTCn	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

图 64　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 32　D 口第二功能

管脚	第二功能
PD0	RXD0（UART0 接收引脚）
PD1	TXD0（UART0 发送引脚）
PD2	INT0（外部中断 0 输入）
PD3	INT1（外部中断 1 输入）
PD4	TOSC1（PTC 振荡器 T/C2）
PD5	TOSC2（PTC 振荡器 T/C2）/OC1A（T/C1 输出比较 A 匹配输出）
PD6	/WR
PD7	/RD

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

- /RD—PD7
- /WR—PD6
- OC1A—PD5

  PD5 可以用作 T/C1 比较匹配的外部输出。此时 PD5 必须配置为输出。OC1A 也是 PWM

的输出引脚。

- INT1—PD3
  外部中断源 1
- INT0—PB2
  外部中断源 0
- TXD0—PD1
  发送数据引脚。发送器使能后，引脚自动配置为输出而不管 DDR1。
- RXD0—PD0
  接收数据引脚。接收器使能后，引脚自动配置为输入而不管 DDR0。若此时 PORTD0 为“1”，则上拉有效。

D 口示意图：

图 65　D 口示意图（PD0）

图 66　D 口示意图（PD1）

图 67　D 口示意图（PD2/3）

图 68　D 口示意图（PD4）

图 69　D 口示意图（PD5）

图 70　D 口示意图（PD6）

图 71　D 口示意图（PD7）

E 口

E 口是 3 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，$07（$27），数据方向寄存器—DDRE，$06（$26）和输入引脚—PINE，$05（$25）。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 34　E 口第二功能

管脚	第二功能
PE0	ICP（T/C1 输入捕捉引脚）/INT2（外部中断 2 输入）

PE1	OC1B（T/C1 输出比较匹配 B）
PE2	ALE

E 口数据寄存器—PORTE

BIT	7	6	5	4	3	2	1	0
$07($27)						**PORTE2**		**PORTE0**
读/写	R	R	R	R	R	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
$06($26)						**DDE2**		**DDE0**
读/写	R	R	R	R	R	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
$05($25)						**PINE2**		**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

- OC1B—PE2
  PE2 **可以用作**T/C1 **比较匹配** B **的外部输出。此时** PE2 **必须配置为输出。**OC1B **也是** PWM **的输出引脚。**
- ALE—PE1
- ICP/INT2—PE0
  ICP，T/C1 **的输入捕捉引脚**
  INT2，**外部中断信号输入**

E **口示意图：**

图 72　**E **口示意图（PE0**）**

图 73　E 口示意图（PE1）

图 74　E 口示意图（PE2）

编程

BOOT 加载器

ATmega83/163 支持程序自我加载。
ATmega83/163 的 FLASH 存储区分为两个部分：
1．应用代码区（$0000 - $1DFF）
2．BOOT 区（$1E00 - $1FFF）

图 75 存储区

BOOT 加载程序可以利用任何可用的数据接口，如 UART，来传送程序代码，并将代码写入 FLASH。
程序 FLASH 以页区分，每页 128 字节。BOOT 区占据从$1E00 - $1FFF 的 8 个页。
SPM（Store Program Memory）指令可以访问整个 FLASH 区，但是只能在 BOOT 范围内执行。如果用户不用 BOOT 功能，则应用代码可证据全部 FLASH。BOOT 区具有两个独立的锁定位，以方便用户选择不同级别的保护。用户可以选择：

- 保护整个 FLASH，禁止 BOOT 更新程序
- 仅保护 BOOT 程序自身免于被自己更新
- 仅保护应用程序免于被 BOOT 程序更新
- 允许软件更新整个 FLASH 区

具体请参见表 36/37。

表 36　BOOT 锁定位 0 保护模式（应用代码区）

BLB0 模式	BLB01	BLB02	保　护
1	1	1	SPM，LPM 可访问应用代码区（$0000 - $1DFF）
2	0	1	SPM 不能访问应用代码区（$0000 - $1DFF）
3	0	0	SPM，LPM 不能访问应用代码区（$0000 - $1DFF）
4	1	0	LPM 不能访问应用代码区（$0000 - $1DFF）

注："1" 代表未编程

表 37　BOOT 锁定位 1 保护模式（BOOT 代码区）

BLB1 模式	BLB11	BLB12	保　护
1	1	1	SPM，LPM 可访问 BOOT 区（$1E00 - $1FFF）
2	0	1	SPM 不能访问 BOOT 区（$1E00 - $1FFF）
3	0	0	SPM，LPM 不能访问 BOOT 区（$1E00 - $1FFF）

4	1	0	LPM 不能访问 BOOT 区（$1E00 - $1FFF）

注："1" 代表未编程

进入 BOOT 加载程序

通过跳转或从应用程序调用即可进入 BOOT 程序区。这些条件可以籍由 UART 或 SPI 接收到的命令触发。此外，也可以将 BOOTRST 熔丝位编程，使得上电后直接跳转到$1E00。要注意 MCU 自己并不能改变熔丝位。也就是说，一旦 BOOTRST 被编程，MCU 的复位向量就一直指向 BOOT 加载程序，除非通过串行/并行编程来改变。BOOTRST 由 LB1 锁定。如果 LB1 被编程，则只有片擦除后才可以改变 BOOTRST。

表 38　BOOT 复位熔丝位 BOOTRST

BOOTRST	复 位 地 址
1	复位向量 = 应用复位（地址$0000）
0	复位向量 = BOOT 复位（地址$1E00）

注："1" 代表未编程

BOOT 加载器的功能

BOOT 加载区的程序可以读/写整个 FLASH 区，包括 BOOT 程序自身。因此，用户可以更新应用程序代码以及负责软件更新的 BOOT 程序本身。要注意的是，如果 BOOT 锁定位 11 未编程，则对 BOOT 程序的不经意的写操作将破坏软件更新。因此，如果不需要更新 BOOT 程序自身，最好将 BOOT 锁定位 11 编程。

FLASH 自编程

写 FLASH 以页为单位，一次操作完成一页。编程前先要擦除。通用写锁定位 2 不控制由 SPM 指令引发的编程操作；同样，读/写锁定位 1 也不控制由 LPM/SPM 引发的操作。

程序存储器只能一页一页地更新，而不是以字为单位。一页有 128 字节（64 个字）。编程过程包括：首先执行页擦除，然后以 SPM 指令一次填充一个字到临时页缓冲区，最后执行编程指令。如果仅需要更新一页当中的一部分，则在擦除前先要将无需改变的内容存储到其他地方，如临时页缓冲区，然后重写。临时页缓冲区可以随机顺序访问。在擦除和编程过程当中，CPU 停止。

设置 SPM 的 BOOT 加载器锁定位

设置 BOOT 加载器锁定位首先要将相应的数据写到 R0，写"1001"到 SPMCR，然后在 SPMCR 操作后的 4 个时钟之内执行 SPM。唯一可以访问的锁定位是 BOOT 锁定位，用来防止应用程序代码和 BOOT 区程序被修改。具体参见表 36 和 37。

BIT	7	6	5	4	3	2	1	0
$05($25)	-	-	**BLB12**	**BLB11**	**BLB02**	**BLB01**	-	-

如果 R0 的位 5-2 为 0，且 SPMCR 的 SPMEN 及 BLBSET 置位，则 SPMCR 操作后的 4 个时钟之内执行的 SPM 指令将对 4 个锁定位进行编程。

页擦除

进行页擦除时，首先要将地址写入 Z 寄存器，然后写"0011"到 SPMCR。在 SPMCR 操作后的 4 个时钟之内执行 SPM。R1 和 R0 中的数据不影响页擦除。页地址必须写入 Z13:Z7，其他位不影响页擦除。

填充临时缓冲区

首先要在 Z 寄存器中设置地址，在 R1:R0 里写入数据，然后写"0001"到 SPMCR。在 SPMCR

操作后的 4 个时钟之内执行 SPM。Z6:Z1 用于寻址位于临时缓冲区的数据，而 Z13:Z7 必须指向要写入的页的地址。

页写入

首先在 Z 寄存器中设置地址，然后写“0101”到 SPMCR。在 SPMCR 操作后的 4 个时钟之内执行 SPM。R1 和 R0 中的数据不影响页写。页地址必须写入 Z13:Z7。Z6:Z0 一定要保持为“0”以保证页写正确。

在自编程过程中寻址 FLASH

Z 寄存器用于 SPM 命令的寻址。

BIT	7	6	5	4	3	2	1	0
$1F($1F)	**Z15**	**Z14**	**Z13**	**Z12**	**Z11**	**Z10**	**Z9**	**Z8**
$1E($1E)	**Z7**	**Z6**	**Z5**	**Z4**	**Z3**	**Z2**	**Z1**	**Z0**

Z15:Z14：没有用

Z13:Z7：页选（页擦除及页写）

Z6:Z1：填充临时缓冲区时用于字选（在页写时必须为 0）

Z0：用于 SPM 时必须为 0。在 LPM 指令里作为字节选取。

不需 Z 寄存器的操作是设置 BOOT 加载器锁定位操作。此时 Z 寄存器对操作没有任何影响。页擦除和页写是独立寻址的。所以要当心不要将地址搞错了。

LPM 指令也使用 Z 寄存器。由于它是一个字节一个字节寻址的，因此还用上了 Z0。

SPM 使能时间窗用来保护不经意地执行 SPM 指令。所有的操作都首先设置 I/O 位，然后在 4 个时钟之内执行 SPM。控制 SPM 的 I/O 寄存器为：

保存程序存储器控制寄存器—SPMCR

BIT	7	6	5	4	3	2	1	0
$37($57)	-	-	-	-	**BLBSET**	**PGWRT**	**PGERS**	**SPMEN**
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

BLBSET：置位 BOOT 锁定位

若此位与 SPMEN 一起置位，则在其后 4 个时钟内执行的 SPM 将 R0 的数据写入到 BOOT 锁定位。R1 及 Z 寄存器的数据不参与此项操作。锁定位改变完成后，或者在 4 个时钟内 SPM 未执行，BLBSET 自动清零。锁定位改变过程当中 CPU 停止。只有片擦除操作才可以清除锁定位。

BLBSET 和 SPMEN 置位后 4 个时钟内执行的 LPM 将把锁定位或熔丝位（依赖于 Z 寄存器的 Z0）读到目的寄存器。

PGWRT：页写

若此位与 SPMEN 一起置位，则在其后 4 个时钟内执行的 SPM 将把临时缓冲区的数据写入到 FLASH。页地址位于 Z 寄存器的高位。页写完成后，或者在 4 个时钟内 SPM 未执行，PGWRT 自动清零。页写过程当中 CPU 停止。

PGERS：页擦除

若此位与 SPMEN 一起置位，则在其后 4 个时钟内执行的 SPM 将擦除一页。页地址位于 Z 寄存器的高位。页擦除完成后，或者在 4 个时钟内 SPM 未执行，PGERS 自动清零。页擦除过程当中 CPU 停止。

SPMEN：保存程序存储器使能

使能其后 4 个时钟内执行的 SPM。如果与 BLBSET，PGWRT 或 PGERS 的其中之一一起置

位，则 SPM 将执行特殊命令。如果仅置位 SPMEN，则其后的 SPM 指令仅把 R1:R0 的值保存到由 Z 寄存器指定的临时缓冲区。SPM 完成后，或者在 4 个时钟内 SPM 未执行，SPMEN 自动清零。

如果在低 4 位写入除“1001”，“0101”，“0011”，“0001”之外的值，或是在任一位已置位后再对 SPMCR 操作将无效。

写 EEPROM 阻止写 SPMCR

写 EEPROM 将阻止对 FLASH 的编程以及读熔丝位和锁定位。所以在写 SPMCR 之前最好先检查 EECR 的 EEWE 位。等 EEWE 为 0 后再操作 SPMCR。

在软件里读熔丝位和锁定位

读锁定位：设置 Z 寄存器为$0001，置位 BLBSET 和 SPMEN，然后在 3 个时钟内执行 LMP，将锁定位的内容传到目的寄存器。读取锁定位完成后，或者在 3/4 个时钟内 LPM/SPM 未执行，BLBSET 和 SPMEN 自动清零。BLBSET 和 SPMEN 清零后，LPM 恢复为通常用途。

BIT	7	6	5	4	3	2	1	0
	-	-	**BLB12**	**BLB11**	**BLB02**	**BLB01**	**LB2**	**LB1**

读熔丝位的算法与读锁定位的类似，但是 Z 寄存器加载的不是$0001，而是$0000。

BIT	7	6	5	4	3	2	1	0
	-	**BOOTRST**	**SPIEN**	**BODLEVEL**	**BODEN**	**CKSEL[2]**	**CKSEL[1]**	**CKSEL[0]**

已编程的位其值为 0。

程序和数据锁定位

ATmega83/163 具有 6 个锁定位，如表 39 所示。锁定位只能通过片擦除命令擦除为“1”。

表 39　锁定保护模式

存储器锁定位			保护类型
LB 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** FLASH **和** EEPROM [1)]
BLB0 模式	**BLB01**	**BLB02**	
1	1	1	SPM，LPM **可访问应用代码区**（$0000 - $1DFF）
2	0	1	SPM **不能访问应用代码区**（$0000 - $1DFF）
3	0	0	SPM，LPM **不能访问应用代码区**（$0000 - $1DFF）
4	1	0	LPM **不能访问应用代码区**（$0000 - $1DFF）
BLB1 模式	**BLB11**	**BLB12**	
1	1	1	SPM，LPM **可访问** BOOT **区**（$1E00 - $1FFF）
2	0	1	SPM **不能访问** BOOT **区**（$1E00 - $1FFF）
3	0	0	SPM，LPM **不能访问** BOOT **区**（$1E00 - $1FFF）
4	1	0	LPM **不能访问** BOOT **区**（$1E00 - $1FFF）

注意：1、在并行编程模式下，熔丝位编程也被禁止。要先编程熔丝位，然后编程锁定位。

熔丝位

ATmega83/163 有 7 个熔丝位：BOOTRST、SPIEN、BODLEVEL、BODEN 和 CKSEL[2:0]。

- BOOTRST 编程（“0”）后复位向量设置为$1E00—BOOT 加载器的入口地址。若未编程（“1”），则复位向量设置为$0000。
- SPIEN 编程（为“0”）后，串行下载程序使能。缺省值为“0”。串行下载模式不能访问

这一位。

- BODLEVEL 选择掉电检测电平及改变启动时间。缺省值为 1。
- BODEN 编程（“0”）后，掉电检测使能。
- CKSEL2..0：复位延时选择。缺省值为“011”。

芯片擦除命令不影响熔丝位。要先编程熔丝位，然后编程锁定位。

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 ATmega83/163：

1、$000：$1E（表明是 Atmel 生产的）

2、$001：$94（16K 字节的 FLASH）

3、$002：$01（当$01 地址为$94 时，器件为 ATmega83/163）

编程 FLASH 和 EEPROM

ATmega83/163 具有 16K 字节的片内可编程 FLASH 和 512 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。

ATmega83/163 的 FLASH 结构为 128 字节/页，共 128 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 40 编程电源电压

型号	串行编程	并行编程
ATmega83/163L	2.7V – 5.5V	4.5V – 5.5V
ATmega83/163	4.0V – 5.5V	4.5V – 5.5V

并行编程

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

信号命名：

图 76 并行编程

ATmega83/163 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 41 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 42
XA1	PD6	I	见表 42
PAGEL	PD7	I	程序存储器页加载

BS2	PA0	I	字节选择 2，总为低电平
DATA	PB0-7	I/O	双向数据总线（/OE 为低时输出）

表 42　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（BS1 决定是高位地址还是低位地址）
0	1	加载数据（BS1 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 43　命令字节编码

命令字节	执行的命令
1000 0000	芯片擦除
0100 0000	写熔丝位
0010 0000	写锁定位
0001 0000	写 FLASH
0001 0001	写 EEPROM
0000 1000	读厂标
0000 0100	读熔丝位和锁定位
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 40 在 V_{CC} 和 GND 之间加上电源电压

2、拉低/RESET 和 BS，等待至少 500ns

3、给/RESET 引脚加上 11.5~12.5V 的电压，等待至少 500ns

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。擦除过程不影响熔丝位。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/WE 施加一个负脉冲。RDY/BSY 变低。

5、等待 RDY/BSY 变高。

FLASH 编程：

FLASH 组织为 128 字节/页，总共 128 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 77。
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 64 次 B 到 F 的过程，以填满一个页缓冲区。
寻址一个页面需要 7 位。前 5 位从地址高位字节得到。后 2 位是 B 步加载的地址低字节的高 2 位（7，6）。
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 128 次 A 到 J，直到所有数据都写入。

图 77 编程波形一

图 78 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~$1F）
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~$01）
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 BS 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 79 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~$01）
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”。

编程熔丝位：
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 6 = BOOTRST
Bit 5 = SPIEN
Bit 4 = BODLEVEL
Bit 3 = BODEN
Bit 2..0 = CKSEL2..0
Bit 7 = “1”。此位是保留位。
3、在/WR 上施加负脉冲，并一直等到 RDY/BSY 变高。

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 5 = Lock Bit12

Bit 4 = Lock Bit11

Bit 3 = Lock Bit02

Bit 2 = Lock Bit01

Bit 1 = Lock Bit2

Bit 0 = Lock Bit1

Bit 7- 6 = “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。

读熔丝位和锁定位：

1、 A：加载命令“0000 0100”

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

Bit 6 = BOOTRST

Bit 5 = SPIEN

Bit 4 = BODLEVEL

Bit 3 = BODEN

Bit 2..0 = CKSEL2..0

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

Bit 5 = Lock Bit12

Bit 4 = Lock Bit11

Bit 3 = Lock Bit02

Bit 2 = Lock Bit01

Bit 1 = Lock Bit2

Bit 0 = Lock Bit1

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”，BS 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 80 并行编程时序

表 44 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_{BVXH}	BS1 Valid before XTAL1 High	67			ns
t_{PHPL}	PAGEL Pulse Width High	67			ns
t_{PLBX}	BS1 Hold after RDY/BSY High	67			ns
t_{PLWL}	PAGEL 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	67			ns
t_{WLRL}	/WR Low to RDY/BSY Low	0		25	μs
t_{WLRH}	/WR Low to RDY/BSY High(1)	0.5	0.7	0.9	ms
t_{WLRH_CE}	/WR Low to RDY/BSY High for Chip Erase(2)	10	14	18	ms
t_{WLRH_FLASH}	/WR Low to RDY/BSY High for Write Flash(3)	5	7	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

注意： 1、t_{WLRH} **仅在写** EEPROM **，熔丝位和锁定位时有效**

2、t_{WLRH_CE} **仅在片擦除时有效**

3、t_{WLRH_FLASH} **仅在写** FLASH **时有效**

串行下载

当/RESET **拉到地时，**FLASH **和** EEPROM **可以进行串行下载。串行接口包括** SCK，MOSI **（输入）和** MISO**（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 **的范围是**$0000~$1FFF**，** EEPROM **的范围是**$000~$01FF**。**

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** ATmega83/163**，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。**

2、**至少等待** 20ms**。 然后在** MOSI**（**PB5**）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**，给**/RESET **施加一个正脉冲并重发编程使能指令。**

4、**如果执行了擦除指令，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低** 6 **位的加载。而写页命**

令则加载地址的高 7 位。如果不用数据检测功能，则必须等待至少 t_{WD_FLASH} 的时间才能继续下一页的加载。否则有可能破坏数据。

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。如果不用数据检测功能，则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 MISO（PB6）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。

FLASH 数据检测

写 FLASH 时，如果内部写过程没有结束，读正在写的地址会得到$FF；当写过程结束后，读取的数据则为写入的数据。用这种方法可以确定何时可以写入新数据。但是对于特定的数据$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_FLASH} 的时间。如果芯片在编程之前已经进行过芯片擦除，则数据$FF 就可以不用再编程了。

EEPROM 数据检测

写 EEPROM 时，如果内部过程没有结束，读正在写的地址会得到$FF；当写过程结束后，读取的数据则为写入的数据。用这种方法可以确定何时可以写入新数据。但是对于特定的数据$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_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。

表 45　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}	28ms	22ms	15ms	11ms
t_{WD_EEPROM}	9ms	7ms	6ms	4ms

表 46　ATmega83/163 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	xxx**a 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	xx**bb bbbb**	**iiii iiii**	在字地址 b 写 H（高或低）字节 i
写 FLASH 页	0100 1100	xxx**a aaaa**	**bb**xx xxxx	iiii iiii	将缓冲区页写到地址 a:b
读 EEPROM	1010 0000	xxxx xxx**a**	**bbbb bbbb**	**oooo oooo**	从地址 a:b 读取数据 o
写 EEPROM	1100 0000	xxxx xxx**a**	**bbbb bbbb**	**iiii iiii**	写数据 i 到地址 a:b
读锁定位	0101 1000	xxxx xxxx	xxxx xxxx	xx**65 4321**	写锁定位
写锁定位	1010 1100	111x xxxx	xxxx xxxx	11**65 4321**	“0”代表已编程

读厂标	0101 0000	xxxx xxxx	xxxx xxxx	oooo oooo	从地址 b 读厂标 o
写熔丝位	1010 1100	101x xxxx	xxxx xx**bb**	1**DCB A987**	“0”代表要编程
读熔丝位	0011 0000	Xxxx xxxx	xxxx xxxx	x**DCB A987**	“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 = Lock Bit1

2 = Lock Bit2

3 = BOOT Lock Bit01

4 = BOOT Lock Bit02

5 = BOOT Lock Bit11

6 = BOOT Lock Bit12

7 = CKSEL0

8 = CKSEL1

9 = CKSEL2

A = BODEN

B = BODLEVEL

C = SPIEN

D = BOOTRST

图 81　串行编程波形

表 47　串行编程电特性，T_A = -40℃**到** 85℃，V_{CC} = 2.7V – 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} = 2.7V – 5.5V）	0		4	MHz
t_{CLCL}	**振荡周期**（V_{CC} = 2.7V – 5.5V）	250			ns
1/ t_{CLCL}	**振荡频率**（V_{CC} = 4.0V – 5.5V）	0		8	MHz
t_{CLCL}	**振荡周期**（V_{CC} = 4.0V – 5.5V）	125			ns
t_{SHSL}	SCK **高**	2 t_{CLCL}			ns
t_{SLSH}	SCK **低**	2t_{CLCL}			ns
t_{OVSH}	MOSI Setup to SCK High	t_{CLCL}			ns
t_{SHOX}	MOSI Hold after SCK High	2t_{CLCL}			ns
t_{SLIV}	SCK Low to MISO Valid	10	16	32	ns

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		0.2 V_{CC} (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	0.8 V_{CC} (2)		V_{CC} +0.5	V
V_{IH2}	**输入高电压**	/RESET	0.9 V_{CC} (2)		V_{CC} +0.5	V

V_{OL}	输出低电压[3] A,B,C,D 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 口	I_{OH} =-3mA，V_{CC} = 5V I_{OH} =-1.5mA,V_{CC} = 3V	4.2 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			980	nA
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			3.0	mA
		闲置状态，V_{CC} = 3V，4MHz			1.2	mA
I_{CC}	掉电模式[5]	WDT 使能，V_{CC} = 3V			15.0	μA
		WDT 禁止，V_{CC} = 3V		9	2.0	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 200mA

2]A 口、ALE、OC1B 和 C0-C7 的 IOL 之和不能超过 100mA

3]B 口、D 口和 XTAL2 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：

1]所有 I/O 口的 IOH 之和不能超过 200mA

2] B 口、D 口和 XTAL2 的 IOH 之和不能超过 100mA

3] B 口、D 口和 XTAL2 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

外部时钟驱动波形

图 83　外部时钟

表 48　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	8	MHz

t_{CLCL}	时钟周期	250		125		ns
t_{CHCX}	高电平时间	100		50		ns
t_{CLCX}	低电平时间	100		50		ns
t_{CLCH}	上升时间		1.6		0.5	μs
t_{CHCL}	下降时间		1.6		0.5	μs

外部存储器时序

表 49　外部存储器特性（4.0V – 5.5V，**无等待周期**）

	符号	参数	8M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					ns
2	t_{AVLL}	A口地址有效到 ALE低					ns
3a	t_{LLAX_ST}	ALE低之后地址保持时间，ST/STD/STS 指令					ns
3b	t_{LLAX_LD}	ALE低之后地址保持时间，LD/LDD/LDS 指令					ns
4	t_{AVLLC}	C口地址有效到 ALE低					ns
5	t_{AVRL}	地址有效到 RD 低					ns
6	t_{AVWL}	地址有效到 WR 低					ns
7	t_{LLWL}	ALE 低到 WR 低					ns
8	t_{LLRL}	ALE 低到 RD 低					ns
9	t_{DVRH}	数据设定到 RD 高					ns
10	T_{RLDV}	RD 低到数据有效					ns
11	T_{RHDX}	RD 高之后数据保持					ns
12	T_{RLRH}	RD 脉冲宽度					ns
13	T_{DVWL}	数据设定到 WR 低					ns
14	T_{WHDX}	WR 高之后数据保持					ns
15	T_{DVWH}	数据有效到 WR 高					ns
16	T_{WLWH}	WR 脉冲宽度					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 50　外部存储器特性（4.0V – 5.5V，**一个等待周期**）

	符号	参数	8M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	8.0	MHz
10	T_{RLDV}	RD 低到数据有效					ns
12	T_{RLRH}	RD 脉冲宽度					ns
15	T_{DVWH}	数据有效到 WR 高					ns
16	T_{WLWH}	WR 脉冲宽度					ns

表 51　外部存储器特性（4.0V – 5.5V，SRWn1 = 1，SRWn0 = 0）

	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	8.0	MHz

10	T_{RLDV}	RD 低到数据有效					ns
12	T_{RLRH}	RD 脉冲宽度					ns
15	T_{DVWH}	数据有效到 WR 高					ns
16	T_{WLWH}	WR 脉冲宽度					ns

表 52　外部存储器特性（4.0V – 5.5V，SRWn1 = 1，SRWn0 = 1）

	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	8.0	MHz
10	T_{RLDV}	RD 低到数据有效					ns
12	T_{RLRH}	RD 脉冲宽度					ns
15	T_{DVWH}	数据有效到 WR 高					ns
16	T_{WLWH}	WR 脉冲宽度					ns

表 53　外部存储器特性（2.7V – 5.5V，无等待周期）

	符号	参数	8M 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					ns
2	t_{AVLL}	A口地址有效到 ALE低					ns
3a	t_{LLAX_ST}	ALE低之后地址保持时间，ST/STD/STS 指令					ns
3b	t_{LLAX_LD}	ALE低之后地址保持时间，LD/LDD/LDS 指令					ns
4	t_{AVLLC}	C口地址有效到 ALE低					ns
5	t_{AVRL}	地址有效到 RD 低					ns
6	t_{AVWL}	地址有效到 WR 低					ns
7	t_{LLWL}	ALE 低到 WR 低					ns
8	t_{LLRL}	ALE 低到 RD 低					ns
9	t_{DVRH}	数据设定到 RD 高					ns
10	T_{RLDV}	RD 低到数据有效					ns
11	T_{RHDX}	RD 高之后数据保持					ns
12	T_{RLRH}	RD 脉冲宽度					ns
13	T_{DVWL}	数据设定到 WR 低					ns
14	T_{WHDX}	WR 高之后数据保持					ns
15	T_{DVWH}	数据有效到 WR 高					ns
16	T_{WLWH}	WR 脉冲宽度					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 54　外部存储器特性（2.7V – 5.5V，一个等待周期）

	符号	参数	8M 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	8.0	MHz
10	T_{RLDV}	RD 低到数据有效					ns
12	T_{RLRH}	RD 脉冲宽度					ns
15	T_{DVWH}	数据有效到 WR 高					ns
16	T_{WLWH}	WR 脉冲宽度					ns

表 55　外部存储器特性（2.7V – 5.5V，SRWn1 = 1，SRWn0 = 0）

	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	8.0	MHz
10	T_{RLDV}	RD 低到数据有效					ns
12	T_{RLRH}	RD 脉冲宽度					ns
15	T_{DVWH}	数据有效到 WR 高					ns
16	T_{WLWH}	WR 脉冲宽度					ns

表 56　外部存储器特性（2.7V – 5.5V，SRWn1 = 1，SRWn0 = 1）

	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	8.0	MHz
10	T_{RLDV}	RD 低到数据有效					ns
12	T_{RLRH}	RD 脉冲宽度					ns
15	T_{DVWH}	数据有效到 WR 高					ns
16	T_{WLWH}	WR 脉冲宽度					ns

图 84　外部存储器时序图（SRWn1 = 0，SRWn0 = 0）

图 85　外部存储器时序图（SRWn1 = 0，SRWn0 = 1）

图 86　外部存储器时序图（SRWn1 = 1，SRWn0 = 0）

图 87　外部存储器时序图（SRWn1 = 1，SRWn0 = 1）

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。

指　　令

Rd**——目的寄存器**　　Rr**——源寄存器**

K——常数　　k**——常数表示的地址**

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

Z、Y、X——R31:R26

A——I/O 地址　　q——偏移量（6 位）

助记符	操作数	描述	操作	受影响的标志	时钟数
算术及逻辑操作					
ADD	Rd，Rr	**不带进位位加**	Rd ← Rd + Rr	Z,C,N,V,S,H	1
ADC	Rd，Rr	**带进位位加**	Rd ← Rd + Rr + C	Z,C,N,V,S,H	1
ADIW	Rd，**K**	**给字加立即数**	Rd+1:Rd← Rd+1:Rd+**K**	Z,C,N,V,S	2
SUB	Rd，Rr	**不带进位位减**	Rd ← Rd - Rr	Z,C,N,V,S,H	1
SUBI	Rd，**K**	**减立即数**	Rd ← Rd - **K**	Z,C,N,V,S,H	1
SBC	Rd，Rr	**带进位位减**	Rd ← Rd – Rr - C	Z,C,N,V,S,H	1
SBC I	Rd，**K**	**带进位减立即数**	Rd ← Rd – **K** - C	Z,C,N,V,S,H	1
SBIW	Rd，**K**	**给字减立即数**	Rd+1:Rd← Rd+1:Rd-**K**	Z,C,N,V,S	2
AND	Rd，Rr	**逻辑与**	Rd ← Rd ▪ Rr	Z, N,V,S	1
ANDI	Rd，**K**	**与立即数**	Rd ← Rd ▪ **K**	Z, N,V,S	1
OR	Rd，Rr	**逻辑或**	Rd ← Rd ∨ Rr	Z, N,V,S	1
ORI	Rd，**K**	**或立即数**	Rd ← Rd ∨ **K**	Z, N,V,S	1
EOR	Rd，Rr	**异或**	Rd ← Rd O± Rr	Z, N,V,S	1
COM	Rd	1	Rd ← $FF - Rd	Z,C,N,V,S	1
NEG	Rd	2 **的补码**	Rd ← $00 - Rd	Z,C,N,V,S,H	1
SBR	Rd，**K**	**寄存器的位置位**	Rd ← Rd ∨ **K**	Z, N,V,S	1
CBR	Rd，**K**	**寄存器的位清零**	Rd ← Rd ▪（$FF - **K**）	Z, N,V,S	1
INC	Rd	**加一**	Rd ← Rd + 1	Z, N,V,S	1
DEC	Rd	**减一**	Rd ← Rd - 1	Z, N,V,S	1
TST	Rd	**零和负测试**	Rd ← Rd ▪ Rd	Z, N,V,S	1
CLR	Rd	**清除寄存器**	Rd ← Rd O± Rd	Z, N,V,S	1
SER	Rd	**置位寄存器**	Rd ← $FF	-	1
MUL	Rd，Rr	**无符号数乘**	R1:R0 ← Rd×Rr(UU)	Z,C	2
MULS	Rd，Rr	**有符号数乘**	R1:R0 ← Rd×Rr(SS)	Z,C	2
MULSU	Rd，Rr	**有/无符号数乘**	R1:R0 ← Rd×Rr(SU)	Z,C	2
FMUL	Rd，Rr	**无符号分数乘**	R1:R0 ← Rd × Rr<<1 (UU)	Z,C	2
FMULS	Rd，Rr	**有符号分数乘**	R1:R0 ← Rd × Rr<<1 (SS)	Z,C	2
FMULSU	Rd，Rr	**有/无符号分数乘**	R1:R0 ← Rd × Rr<<1 (SU)	Z,C	2
跳转指令					
RJMP	k	**相对跳转**	PC ← PC + k + 1	-	2
IJMP		**间接跳转**（Z）	PC(15:0)←Z, PC(21:16)←0	-	2
JMP	k	**绝对跳转**	PC ← k	-	3
RCALL	k	**相对调用**	PC ← PC + k + 1	-	3/4
ICALL		**间接调用**（Z）	PC(15:0)←Z, PC(21:16)←0	-	3/4
CALL	k	**绝对调用**	PC ← k	-	4/5
RET		**调用返回**	PC ← **堆栈**	-	4/5
RETI		**中断返回**	PC ← **堆栈**	I	4/5
CPSE	Rd，Rr	**比较，相等则跳**	**若**（Rd=Rr）PC←PC+2 **或** 3	-	1/2/3
CP	Rd，Rr	**比较**	Rd - Rr	Z,C,N,V,S,H	1
CPC	Rd，Rr	**带进位位比较**	Rd – Rr - C	Z,C,N,V,S,H	1

CPI	Rd，**K**	**与立即数比较**	Rd – **K**	Z,C,N,V,S,H	1
SBRC	Rd，b	**寄存器位清零即跳**	**若**（Rd（b） = 0）PC←PC+2 **或** 3	-	1/2/3
SBRS	Rd，b	**寄存器位置位即跳**	**若**（I/O（A，b） = 1）PC←PC+2 **或** 3	-	1/2/3
SBIC	A，b	I/O **寄存器位清零即跳**	**若**（I/O（A，b） = 0）PC←PC+2 **或** 3	-	1/2/3
SBIS	A，b	I/O **寄存器位置位即跳**	**若**（Rd（b） = 1）PC←PC+2 **或** 3	-	1/2/3
BRBS	s，k	**状态标志置位跳**	**若**（SREG（s）=1）PC←PC+k+1	-	1/2
BRBC	s，k	**状态标志清零跳**	**若**（SREG（s）=0）PC←PC+k+1	-	1/2
BREQ	k	**相等即跳**	**若**（Z=1）PC←PC+k+1	-	1/2
BRNE	k	**不等即跳**	**若**（Z=0）PC←PC+k+1	-	1/2
BRCS	k	**进位即跳**	**若**（C=1）PC←PC+k+1	-	1/2
BRCC	k	**没有进位即跳**	**若**（C=0）PC←PC+k+1	-	1/2
BRSH	k	**大于等于即跳（无符号）**	**若**（C=0）PC←PC+k+1	-	1/2
BRLO	k	**小于即跳（无符号）**	**若**（C=1）PC←PC+k+1	-	1/2
BRMI	k	**负即跳**	**若**（N=1）PC←PC+k+1	-	1/2
BRPL	k	**正即跳**	**若**（N=0）PC←PC+k+1	-	1/2
BRGE	k	**大于等于即跳（有符号）**	**若**（NO+V=1）PC←PC+k+1	-	1/2
BRLT	k	**小于即跳（有符号）**	**若**（NO+V=0）PC←PC+k+1	-	1/2
BRHS	k	H **位置位即跳**	**若**（H=1）PC←PC+k+1	-	1/2
BRHC	k	H **位清零即跳**	**若**（H=0）PC←PC+k+1	-	1/2
BRTS	k	T **置位即跳**	**若**（T=1）PC←PC+k+1	-	1/2
BRTC	k	T **清零即跳**	**若**（T=0）PC←PC+k+1	-	1/2
BRVS	k	V **置位即跳**	**若**（V=1）PC←PC+k+1	-	1/2
BRVC	k	V **清零即跳**	**若**（V=0）PC←PC+k+1	-	1/2
BRIE	k	**中断使能即跳**	**若**（I=1）PC←PC+k+1	-	1/2
BRID	k	**中断禁止即跳**	**若**（I=0）PC←PC+k+1	-	1/2
数据传输指令					
MOV	Rd，Rr	**拷贝寄存器**	Rd ← Rr	-	1
MOVW	Rd，Rr	**拷贝一对寄存器**	Rd+1:Rd ← Rr+1:Rr	-	1
LDI	Rd，**K**	**取立即数**	Rd ← **K**	-	1
LDS	Rd，k	**从数据区取数**	Rd ←（k）	-	2
LD	Rd，X	**间接取数**	Rd ←（X）	-	2
LD	Rd，X+	**间接取数，后加**	Rd ←（X），X←X+1	-	2
LD	Rd，-X	**间接取数，先减**	X←X-1，Rd ←（X）	-	2
LD	Rd，Y	**间接取数**	Rd ←（Y）	-	2
LD	Rd，Y+	**间接取数，后加**	Rd ←（Y），Y←Y+1	-	2
LD	Rd，-Y	**间接取数，先减**	Y←Y-1，Rd ←（Y）	-	2
LDD	Rd,Y+q	**带偏移间接取数**	Rd ←（Y + q）	-	2
LD	Rd，Z	**间接取数**	Rd ←（Z）	-	2
LD	Rd，Z+	**间接取数，后加**	Rd ←（Z），Z←Z+1	-	2

LD	Rd，-Z	**间接取数，先减**	Z←Z-1，Rd ←（Z）	-	2
LDD	Rd,Z+q	**带偏移间接取数**	Rd ←（Z + q）	-	2
STS	k ，Rr	**存数于数据区**	（k）← Rr	-	2
ST	X，Rr	**间接存数**	（X）← Rr	-	2
ST	X+，Rr	**间接存数，后加**	（X）← Rr，X←X+1	-	2
ST	X-，Rr	**间接存数，先减**	X←X-1，（X）← Rr	-	2
ST	Y，Rr	**间接存数**	（Y）← Rr	-	2
ST	Y+，Rr	**间接存数，后加**	（Y）← Rr，Y←Y+1	-	2
ST	Y-，Rr	**间接存数，先减**	Y←Y-1，（Y）← Rr	-	2
STD	Y+q,Rr	**带偏移间接存数**	（Y + q）← Rr	-	2
ST	Z，Rr	**间接存数**	（Z）← Rr	-	2
ST	Z+，Rr	**间接存数，后加**	（Z）← Rr，Z←Z+1	-	2
ST	Z-，Rr	**间接存数，先减**	Z←Z-1，（Z）← Rr	-	2
STD	Z+q,Rr	**带偏移间接存数**	（Z + q）← Rr	-	2
LPM		**在程序区取数**	R0 ←（Z）	-	3
LPM	Rd，Z	**在程序区取数**	Rd ←（Z）	-	3
LPM	Rd，Z+	**在程序区取数，后加**	Rd ←（Z），Z←Z+1	-	3
SPM		**存数于程序区**	（Z）← R1:R0	-	-
IN	Rd，A	**从 I/O 取数**	Rd ← I/O（A）	-	1
OUT	A，Rr	**存数于 I/O**	I/O（A）← Rr	-	1
PUSH	Rr	**入栈**	STACK ← Rr	-	2
POP	Rd	**出栈**	Rd ← STACK	-	2
位及位测试指令					
LSL	Rd	**逻辑左移**	Rd(n + 1) ← Rd(n)，Rd(n)←0，C←Rd(7)	Z,C,N,V,H	1
LSR	Rd	**逻辑右移**	Rd(n) ← Rd(n + 1)，Rd(7)←0，C←Rd(0)	Z,C,N,V	1
ROL	Rd	**带进位位左移**	Rd(0)←C，Rd(n + 1) ← Rd(n)，C←Rd(7)	Z,C,N,V,H	1
ROR	Rd	**带进位位右移**	Rd(7)←C ，Rd(n) ← Rd(n + 1)，C←Rd(0)	Z,C,N,V	1
ASR	Rd	**算术右移**	Rd(n) ← Rd(n + 1)，n=6..0	Z,C,N,V	1
SWAP	Rd	**高低位交换**	Rd(3..0) ←→ Rd(7..4)	-	1
BSET	s	**设置标志**	SREG（s）← 1	SREG（s）	1
BCLR	s	**清除标志**	SREG（s）← 0	SREG（s）	1
SBI	A，b	**设置 I/O 寄存器的位**	I/O（A，b）← 1	-	2
CBI	A，b	**清除 I/O 寄存器的位**	I/O（A，b）← 0	-	2
BST	Rr，b	Rr **的** b **位到** T	T ← Rr（b）	T	1
BLD	Rd，b	T **到** Rr **的** b **位**	Rr（b）← T	-	1
SEC		**置位** C	C ← 1	C	1
CLC		**清零** C	C ← 0	C	1
STN		**置位** N	N ← 1	N	1
CLN		**清零** N	N ← 0	N	1
SEZ		**置位** Z	Z ← 1	Z	1
CLZ		**清零** Z	Z ← 0	Z	1

SEI		置位 I	I ← 1	I	1
CLI		清零 I	I ← 0	I	1
SES		置位 S	S ← 1	S	1
CLS		清零 S	S ← 0	S	1
SEV		置位 V	V ← 1	V	1
CLV		清零 V	V ← 0	V	1
SET		置位 T	T ← 1	T	1
CLT		清零 T	T ← 0	T	1
SEH		置位 H	H ← 1	H	1
CLH		清零 H	H ← 0	H	1
NOP		空操作		-	1
SLEEP		休眠指令		-	3
WDR		看门狗复位		-	1

定货信息

速度（MHz）	电源（V）	定货号	封装	工作范围
4	2.7 – 5.5	ATmega83/163 – 4AC ATmega83/163 – 4JC ATmega83/163 – 4PC	44A 44J 40P6	商用 （0℃ - 70℃）
		ATmega83/163 – 4AI ATmega83/163 – 4JI ATmega83/163 – 4PI	44A 44J 40P6	工业 （-40℃ - 85℃）
8	4.0 – 5.5	ATmega83/163 – 8AC ATmega83/163 – 8JC ATmega83/163 – 8PC	44A 44J 40P6	商用 （0℃ - 70℃）
		ATmega83/163 – 8AI ATmega83/163 – 8JI ATmega83/163 – 8PI	44A 44J 40P6	工业 （-40℃ - 85℃）

封 装 类 型	
44A	44 脚，TQFP
44J	44 脚，PLCC
44P6	40 脚，PDIP

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.）