ATmega88包含512字节的EEPROM数据存储器。它是作为一个独立的数据 EEPROM 的寿命至少为 100,000 次擦除周期。 EEPROM 的访问由地址寄存器，数据寄存器和控制寄存器决定。

具体的 SPI 及并行下载 EEPROM 数据请参见 P254“存储器编程”。

**EEPROM 读/ 写访问**

EEPROM 读/ 写访问EEPROM 的访问寄存器位于I/O 空间。

EEPROM的写访问时间由Table 3给出。自定时功能可以让用户软件监测何时可以开始写下一字节。用户操作EEPROM 需要注意如下问题：在电源滤波时间常数比较大的电路中，上电/ 下电时VCC 上升/ 下降速度会比较慢。此时CPU 可能工作于低于晶振所要求的电源电压。请参见 P20“ 防止EEPROM 数据丢失” 以避免出现EEPROM 数据丢失的问题。

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EEPROM控制寄存器的内容。

执行EEPROM 读操作时， CPU 会停止工作4 个周期，然后再执行后续指令；执行EEPROM 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**



· Bits 15..9 – Res: 保留
保留位，读操作返回值为零。

· Bits 8..0 – EEAR8..0: EEPROM 地址
EEPROM 地址寄存器 – EEARH和EEARL指定了 512 字节的EEPROM。
EEPROM 地址是线性的，从 0 到 511 。 EEAR 的初始值没有定义。 在访问EEPROM 之前必须为其赋予正确的数据。
EEAR8 在 ATmega48 中为无效位，必须始终将其赋值为 ”0”。

**EEPROM 数据寄存器 － EEDR**



· Bits 7..0 – EEDR7.0: EEPROM 数据
对于EEPROM 写操作， EEDR 是需要写到EEAR 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**



· Bits 7..4 – Res: 保留
保留位，读操作返回值为零。

· Bit 5, 4 – EEPM1 与 EEPM0: EEPROM 编程模式位
EEPROM编程模式位的设置决定对EEPE写入后将触发什么编程方式。EEPROM的编程可以作为一个原子操作来实现擦除老的数据并写入新的数据，也可以将擦除与写操作 分为两步进行。不同编程模式的时序请见 Table2。 EEPE 置位时，对 EEPMn 的任何写操作都将会被忽略。在复位过程中，除非 EEPROM 处于编程状态，EEPMn 位将被设置 为 0b00。



· Bit 3 – EERIE: 使能 EEPROM 准备好中断
若 SREG 的 I 为 "1"，则置位 EERIE 使能 EEPROM 准备好中断。清零 EERIE 则禁止此中断。当 EEPE 清零时 EEPROM准备好中断即可发生。

· Bit 2 – EEMPE: EEPROM 写使能
EEMPE决定设置EEPE为"1“是否可以启动EEPROM写操作。当EEMPE为"1“时，在4个时钟周期内置位 EEPE 将把数据写入 EEPROM 的指定地址；若 EEMPE 为 "0“，则EEPE不起作用。EEMPE置位后4个周期，硬件对其清零。见EEPROM写过程对EEPE位的描述。
· Bit 1 – EEPE: EEPROM 写使能
写使能信号 EEPE 是 EEPROM 的写入选通信号。当 EEPROM 数据和地址设置好之后，需置位 EEPE 以便将数据写入 EEPROM。此时 EEMPE 必须置位，否则 EEPROM 写操作将不会发生。写时序如下 ( 第 3 和第 4 步的次序可更改 )：
1. 等待EEPE 位变为零
2. 等待SPMCSR 中的SPMEN 位变为零
3. 将新的EEPROM 地址写入EEAR ( 可选)
4. 将新的EEPROM 数据写入EEDR ( 可选)
5. 对EECR 寄存器的EEMPE 写"1"，同时清零EEPE
6. 在置位EEMPE 的4 个周期内，置位EEPE

在 CPU 写 Flash 存储器的时候不能对 EEPROM 进行编程。 在启动 EEPROM写操作之前软件必须要检查Flash写操作是否已经完成。第二步仅在软件包含引导程序，允许CPU对 Flash 进行编程时才有用。如果 CPU 永远都不会写 Flash，则第二步可以忽略。请参考 P240“Boot Loader 支持 RWW 自编程， ATmega88 与 ATmega168”。（注意：ATmega48 无Boot Loader）

注意 : 如有中断发生于步骤 5 和 6 之间将导致写操作失败。 因为此时 EEPROM 写使能操作将超时。如果一个操作EEPROM的中断打断了另一个EEPROM操作，EEAR或EEDR寄存器可能被修改，引起 EEPROM 操作失败。建议此时关闭全局中断标志 I。

经过写访问时间之后，EEPE硬件清零。用户可以凭此位判断写时序是否已经完成。EEPE置位后， CPU 要停止两个时钟周期才会运行下一条指令。

· Bit 0 – EERE: EEPROM 读使能
读使能信号 EERE是 EEPROM的写入选通信号。 当 EEPROM地址设置好之后，需置位EERE以便将数据读入EEAR。EEPROM数据的读取只需要一条指令。读取EEPROM时 CPU 要停止 4 个时钟周期然后才能执行下一条指令。

用户在读取EEPROM 时应该检测EEPE。如果一个写操作正在进行，就无法读取EEPROM，也无法改变寄存器EEAR。 经过校准的片内振荡器用于EEPROM定时。

标定的振荡器用于 EEPROM 访问定时。 Table3 为 CPU 访问 EEPROM的典型时间。



下面的代码分别用汇编和C函数说明如何实现EEPROM的写操作。在此假设中断不会在执行这些函数的过程当中发生。例子同时还假设软件没有引导程序。若引导程序存在，则 EEPROM 写函数还需要等待正在进行的 SPM 命令的结束。

汇编代码例程
EEPROM\_write:
; 等待上一次写操作结束
sbic EECR,EEPE
rjmp EEPROM\_write
; 设置地址寄存器 (r18:r17)
out EEARH, r18
out EEARL, r17
; 将数据写入数据寄存器(r16)
out EEDR,r16
; 置位EEMPE
sbi EECR,EEMPE
; 置位EEPE 以启动写操作
sbi EECR,EEPE
ret
C 代码例程
void EEPROM\_write(unsigned int uiAddress, unsigned char ucData)
{
/\* 等待上一次写操作结束 \*/
while(EECR & (1<<EEPE))
;
/\* 设置地址和数据寄存器\*/
EEAR = uiAddress;
EEDR = ucData;
/\* 置位EEMPE \*/
EECR |= (1<<EEMPE);
/\* 置位EEPE 以启动写操作\*/
EECR |= (1<<EEPE);
}

下面的例子说明如何用汇编和C 函数来读取EEPROM，在此假设中断不会在执行这些函数的过程当中发生。

汇编代码例程
EEPROM\_read:
; 等待上一次写操作结束
sbic EECR,EEPE
rjmp EEPROM\_read
; 设置地址寄存器 (r18:r17)
out EEARH, r18
out EEARL, r17
; 设置EERE 以启动读操作
sbi EECR,EERE
; 自数据寄存器读取数据
in r16,EEDR
ret
C 代码例程
unsigned char EEPROM\_read(unsigned int uiAddress)
{
/\* 等待上一次写操作结束 \*/
while(EECR & (1<<EEPE))
;
/\* 设置地址寄存器\*/
EEAR = uiAddress;
/\* 设置EERE 以启动读操作\*/
EECR |= (1<<EERE);
/\* 自数据寄存器返回数据 \*/
return EEDR;
}

**在掉电休眠模式下的EEPROM写操作**

若程序执行掉电指令时EEPROM的写操作正在进行，EEPROM的写操作将继续，并在指定的写访问时间之前完成。 但写操作结束后，振荡器还将继续运行，单片机并非处于完全的掉电模式。因此在执行掉电指令之前应结束EEPROM 的写操作。

**防止EEPROM 数据丢失**

若电源电压过低，CPU 和EEPROM 有可能工作不正常，造成EEPROM 数据的毁坏( 丢失)。这种情况在使用独立的EEPROM 器件时也会遇到。因而需要使用相同的保护方案。

由于电压过低造成EEPROM 数据损坏有两种可能：一是电压低于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方法解决：
当电压过低时保持AVR RESET信号为低。这可以通过使能芯片的掉电检测电路BOD来实现。如果BOD 电平无法满足要求则可以使用外部复位电路。若写操作过程当中发生了复位，只要电压足够高，写操作仍将正常结束。