产品特性

- 使用 AVR® RISC 结构
- AVR – 高性能、低功耗的 RISC 结构
  - 120 条指令 – 大多数指令执行时间为单个时钟周期
  - 32 个 8 位通用工作寄存器
  - 全静态工作
  - 工作于 20 MHz 时性能高达 20 MIPS
- 数据与非易失性程序和数据存储器
  - 2K 字节的系统内可编程 Flash
    擦写寿命 : 10,000 次
  - 128 字节的系统内可编程 EEPROM
    擦写寿命 : 100,000 次
  - 128 字节的片内 SRAM
  - 可以对锁定位进行编程以及实现 EEPROM 数据的加密
- 外设特点
  - 具有独立预分频器及比较模式的 8 位定时器 / 计数器
  - 具有独立预分频器及比较、捕获模式的 16 位定时器 / 计数器
  - 四路 PWM 通道
  - 片内模拟比较器
  - 具有片内振荡器的可编程看门狗定时器
  - USI – 全局串行接口
  - 全双工 USART
- 特殊的处理器特点
  - debugWIRE 片上调试
  - 通过 SPI 端口在系统内可编程
  - 片内 / 片外中断源
  - 低功耗空闲模式、掉电模式、 Standby 模式
  - 增强型上电复位
  - 可编程的掉电检测
  - 片内标定振荡器
- I/O 和封装
  - 18 可编程 I/O 线
  - 20 引脚 PDIP, 20 引脚 SOIC 与 32 引脚 MLF
- 工作电压
  - 1.8 - 5.5V (ATtiny2313V)
  - 2.7 - 5.5V (ATtiny2313)
- 速度等级
  - ATtiny2313V: 0 - 4 MHz @ 1.8 - 5.5V, 0 - 10 MHz @ 2.7 - 5.5V
  - ATtiny2313: 0 - 10 MHz @ 2.7 - 5.5V, 0 - 20 MHz @ 4.5 - 5.5V
- 功耗估计
  - 正常模式 :
    1 MHz, 1.8V: 300 µA
    32 kHz, 1.8V: 20 µA (包括振荡器)
  - 掉电模式 :
    < 0.2 µA at 1.8V

具有 2KB 系统内可编程 Flash 的 8 位 AVR® 微控制器

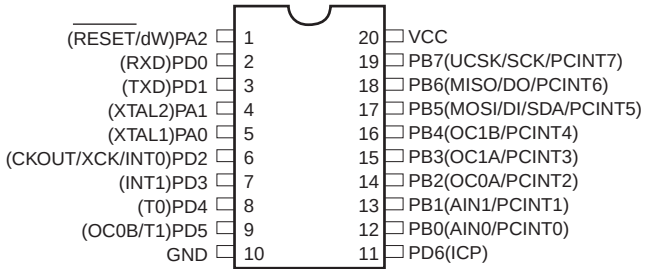
ATtiny2313/V

初稿

Rev. 2543F–AVR–07/04

引脚配置

Figure 1. ATtiny2313 引脚配置

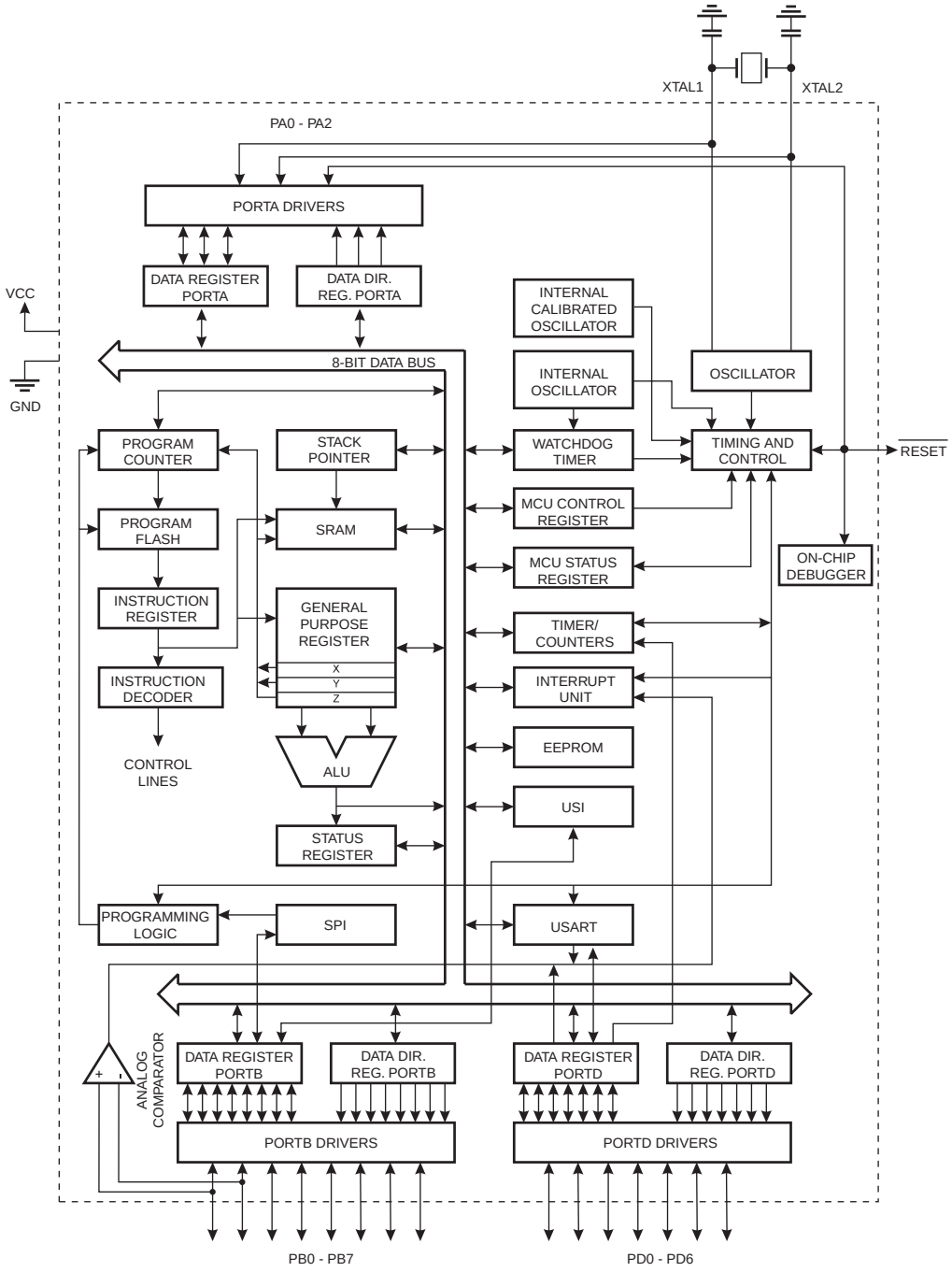


综述

ATtiny2313是基于增强的AVR RISC结构的低功耗8位CMOS微控制器。由于其先进的指令集以及单时钟周期指令执行时间， ATtiny2313 的数据吞吐率高达 1 MIPS/MHz，从而可以缓减系统在功耗和处理速度之间的矛盾。

方框图

Figure 2. 方框图



AVR 内核具有丰富的指令集和 32 个通用工作寄存器。所有的寄存器都直接与算逻单元 (ALU) 相连接，使得一条指令可以在一个时钟周期内同时访问两个独立的寄存器。这种结构大大提高了代码效率，并且具有比普通的 CISC 微控制器最高至 10 倍的数据吞吐率。

ATtiny2313有2K字节系统内可编程Flash，128字节EEPROM，128字节SRAM，18 个通用 I/O 口线， 32 个通用工作寄存器，对片内调试的单线接口， 2 个具有比较模式的灵活的定时器 / 计数器，片内 / 外中断，串行可编程 USART，有启动状态检测器的通用串行接口，含片内振荡器的可编程看门狗定时器，以及三种可以通过软件进行选择的省电模式。工作于空闲模式时 CPU 停止工作，而 SRAM、 T/C 以及中断系统继续工作；掉电模式时晶体振荡器停止振荡，所有功能除了中断和硬件复位之外都停止工作； Standby 模式下只有晶体振荡器运行，使得器件只消耗极少的电流，同时具有快速启动能力。

本芯片是以 Atmel 高密度非易失性存储器技术生产的。通过 SPI 串行接口或非易失性存储器编程器可对程序存储器进行系统内编程。通过将 8 位 RISC CPU 与系统内可编程的 Flash 集成在一个芯片内，ATtiny2313 成为一个功能强大的单片机，为许多嵌入式控制应用提供了灵活而低成本的解决方案。

ATtiny2313 AVR 具有一整套的编程与系统开发工具，包括：C 语言编译器、宏汇编、程序调试器 / 软件仿真器、仿真器及评估板。

引脚说明

VCC 数字电路的电源。

GND 地

端口 A(PA2..PA0) 端口 A 为 3 位双向 I/O 口，具有可编程的内部上拉电阻。其输出缓冲器具有对称的驱动特性，可以输出和吸收大电流。作为输入使用时，若内部上拉电阻使能，端口被外部电路拉低时将输出电流。在复位过程中，即使系统时钟还未起振，端口 A 处于高阻状态。

端口 A 也可以用做其他不同的特殊功能，请参见 P51。

端口 B(PB7..PB0) 端口 B 为 8 位双向 I/O 口，具有可编程的内部上拉电阻。其输出缓冲器具有对称的驱动特性，可以输出和吸收大电流。作为输入使用时，若内部上拉电阻使能，端口被外部电路拉低时将输出电流。在复位过程中，即使系统时钟还未起振，端口 B 处于高阻状态。

端口 B 也可以用做其他不同的特殊功能，请参见 P51。

端口 D(PD6..PD0) 端口 D 为 7 位双向 I/O 口，具有可编程的内部上拉电阻。其输出缓冲器具有对称的驱动特性，可以输出和吸收大电流。作为输入使用时，若内部上拉电阻使能，端口被外部电路拉低时将输出电流。在复位过程中，即使系统时钟还未起振，端口 D 处于高阻状态。

端口 D 也可以用做其他不同的特殊功能，请参见 P54。

$\overline{\text{RESET}}$ 复位输入引脚。持续时间超过最小门限时间的低电平将引起系统复位。门限时间见 P32Table 15。持续时间小于门限时间的脉冲不能保证可靠复位。其第二功能为PA2与dW的复位输入。

XTAL1 反向振荡放大器与片内时钟操作电路的输入端。XTAL1 是 PA0 的第二功能。

XTAL2 反向振荡放大器的输出端。 XTAL2 是 PA1 的第二功能。

代码例子

本数据手册包含了一些简单的代码例子以说明如何使用芯片各个不同的功能模块。这些例子都假定在编译之前已经包含了正确的头文件。有些 C 编译器在头文件里并没有包含位定义，而且各个 C 编译器对中断处理有自己不同的处理方式。请注意查阅相关文档以获取具体的信息。

声明

本数据手册的典型值来源于对器件的仿真，以及其他基于相同产生工艺的 AVR 微控制器的标定特性。本器件经过特性化之后将给出实际的最大值和最小值。

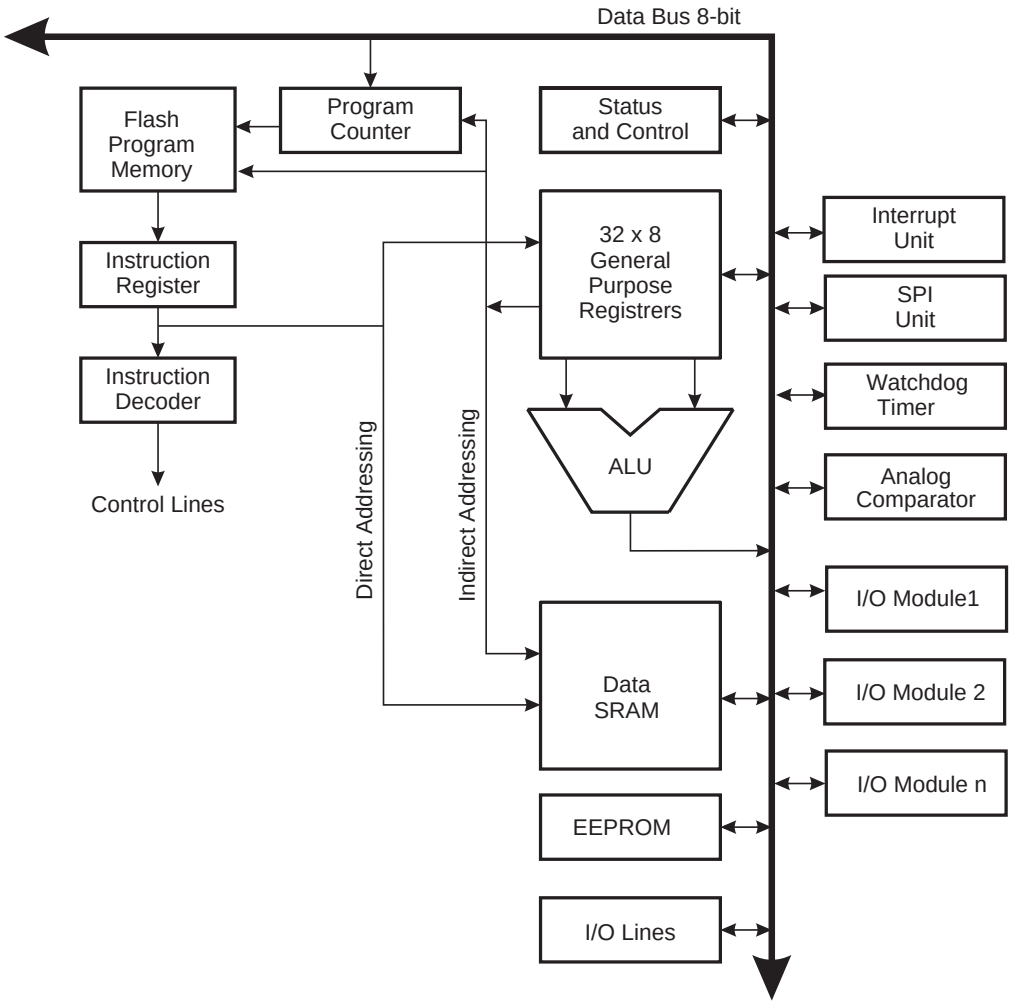
AVR CPU 内核

介绍

本节从总体上讨论 AVR 内核的结构。CPU 的主要任务是保证程序的正确执行。因此它必须能够访问存储器、执行运算、控制外设以及处理中断。

结构综述

Figure 3. AVR 结构的方框图



为了获得最高的性能以及并行性， AVR 采用了 Harvard 结构，具有独立的数据和程序总线。程序存储器里的指令通过一级流水线运行。 CPU 在执行一条指令的同时读取下一条指令（在本文称为预取）。这个概念实现了指令的单时钟周期运行。程序存储器是可以在线编程的 Flash。

快速访问寄存器文件包括 32 个 8 位通用工作寄存器，访问时间为一个时钟周期。从而实现了单时钟周期的 ALU 操作。在典型的 ALU 操作中，两个位于寄存器文件中的操作数同时被访问，然后执行运算，结果再被送回到寄存器文件。整个过程仅需一个时钟周期。

寄存器文件里有 6 个寄存器可以用作 3 个 16 位的间接寻址寄存器指针以寻址数据空间，实现高效的地址运算。其中一个指针还可以作为程序存储器查询表的地址指针。这些附加的功能寄存器即为 16 位的 X、 Y、 Z 寄存器。

ALU支持寄存器之间以及寄存器和常数之间的算术和逻辑运算。ALU也可以执行单寄存器操作。运算完成之后状态寄存器的内容得到更新以反映操作结果。

程序流程通过有 / 无条件的跳转指令和调用指令来控制，从而直接寻址整个地址空间。大多数指令长度为 16 位，亦即每个程序存储器地址都包含一条 16 位或 32 位的指令。

在中断和调用子程序时返回地址的程序计数器 (PC) 保存于堆栈之中。堆栈位于通用数据 SRAM，因此其深度仅受限于 SRAM 的大小。在复位例程里用户首先要初始化堆栈指针 SP。这个指针位于 I/O 空间，可以进行读写访问。数据 SRAM 可以通过 5 种不同的寻址模式进行访问。

AVR 存储器空间为线性的平面结构。

AVR有一个灵活的中断模块。控制寄存器位于I/O空间。状态寄存器里有全局中断使能位。每个中断在中断向量表里都有独立的中断向量。各个中断的优先级与其在中断向量表的位置有关，中断向量地址越低，优先级越高。

I/O存储器空间包含64个可以直接寻址的地址，作为CPU外设的控制寄存器以及其他I/O功能。映射到数据空间即为寄存器文件之后的地址 0x20 - 0x5F。

ALU - 算术逻辑单元

AVR ALU 与 32 个通用工作寄存器直接相连。寄存器与寄存器之间、寄存器与立即数之间的 ALU 运算只需要一个时钟周期。 ALU 操作分为 3 类：算术、逻辑和位操作。此外还提供了支持无 / 有符号数和分数乘法的乘法器。具体请参见指令集。

状态寄存器

状态寄存器包含了最近执行的算术指令的结果信息。这些信息可以用来改变程序流程以实现条件操作。如指令集所述，所有 ALU 运算都将影响状态寄存器的内容。这样，在许多情况下就不需要专门的比较指令了，从而使系统运行更快速，代码效率更高。

在进入中断服务程序时状态寄存器不会自动保存，中断返回时也不会自动恢复。这些工作需要软件来处理。

AVR 中断寄存器 SREG 定义如下：

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
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	

- **Bit 7 – I: 全局中断使能**

I 置位时使能全局中断。单独的中断使能由其他独立的控制寄存器控制。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。任意一个中断发生后 I 清零，而执行 RETI 指令后 I 恢复置位以使能中断。 I 也可以通过 SEI 和 CLI 指令来置位和清零。

- **Bit 6 – 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。

- **Bit 5 – H: 半进位标志**

半进位标志 H 表示算术操作发生了半进位。此标志对于 BCD 运算非常有用。详见指令集的说明。

- **Bit 4 – S: 符号位 , S = N ⊕ V**

S 为负数标志 N 与 2 的补码溢出标志 V 的异或。详见指令集的说明。

- **Bit 3 – V: 2 的补码溢出标志**

支持 2 的补码运算。详见指令集的说明。

- **Bit 2 – N: 负数标志**

表明算术或逻辑操作结果为负。详见指令集的说明。

- **Bit 1 – Z: 零标志**

表明算术或逻辑操作结果为零。详见指令集的说明。

- **Bit 0 – C: 进位标志**

表明算术或逻辑操作发生了进位。详见指令集的说明。

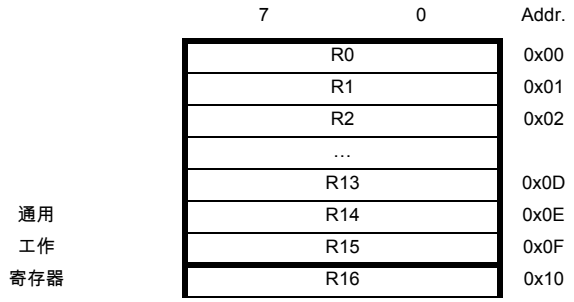
通用寄存器文件

寄存器文件针对 AVR 增强型 RISC 指令集做了优化。为了获得需要的性能和灵活性，寄存器文件支持以下的输入 / 输出方案：

- 输出一个 8 位操作数，输入一个 8 位结果
- 输出两个 8 位位操作数，输入一个 8 位结果
- 输出两个 8 位位操作数，输入一个 16 位结果
- 输出一个 16 位位操作数，输入一个 16 位结果

Figure 4 为 CPU 32 个通用工作寄存器的结构。

Figure 4. AVR CPU 通用工作寄存器



R17	0x11	
…		
R26	0x1A	X 寄存器，低字节
R27	0x1B	X 寄存器，高字节
R28	0x1C	Y 寄存器，低字节
R29	0x1D	Y 寄存器，高字节
R30	0x1E	Z 寄存器，低字节
R31	0x1F	Z 寄存器，高字节

大多数操作寄存器文件的指令都可以直接访问所有的寄存器，而且多数这样的指令的执行时间为单个时钟周期。

如 Figure 4 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。虽然寄存器文件的物理实现不是 SRAM，这种内存组织方式在访问寄存器方面具有极大的灵活性，因为 X、 Y、 Z 寄存器可以设置为指向任意寄存器的指针。

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。这三个间接寻址寄存器示于 Figure 5。

Figure 5. X、 Y、 Z 寄存器

在不同的寻址模式中，这些地址寄存器可以实现固定偏移量，自动加一和自动减一功能。具体细节请参见指令集。

堆栈指针

堆栈指针主要用来保存临时数据、局部变量和中断 / 子程序的返回地址。堆栈指针总是指向堆栈的顶部。要注意 AVR 的堆栈是向下生长的，即新数据推入堆栈时，堆栈指针的数值将减小。

堆栈指针指向数据 SRAM 堆栈区。在此聚集了子程序堆栈和中断堆栈。调用子程序和使能中断之前必须定义堆栈空间，且堆栈指针必须指向高于 0x60 的地址空间。使用 PUSH 指令将数据推入堆栈时指针减一；而子程序或中断返回地址推入堆栈时指针将减二。使用 POP 指令将数据弹出堆栈时，堆栈指针加一；而用 RET 或 RETI 指令从子程序或中断返回时堆栈指针加二。

AVR的堆栈指针由I/O空间中的两个8位寄存器实现。实际使用的位数与具体器件有关。请注意某些 AVR 器件的数据区太小，用 SPL 就足够了。此时将不给出 SPH 寄存器。

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	–	–	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
读 / 写	R	R	R	R	R	R	R	R	
	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	

指令执行时序

这一节介绍指令执行过程中的访问时序。AVR CPU 由系统时钟 clk_{CPU} 驱动。此时钟直接来自选定的时钟源。芯片内部不对此时钟进行分频。

Figure 6 说明了由 Harvard 结构决定的并行取指和指令执行，以及可以进行快速访问的寄存器文件的概念。这是一个基本的流水线概念，性能高达 1 MIPS/MHz，具有优良的性价比、功能 / 时钟比、功能 / 功耗比。

Figure 6. 并行取指和指令执行

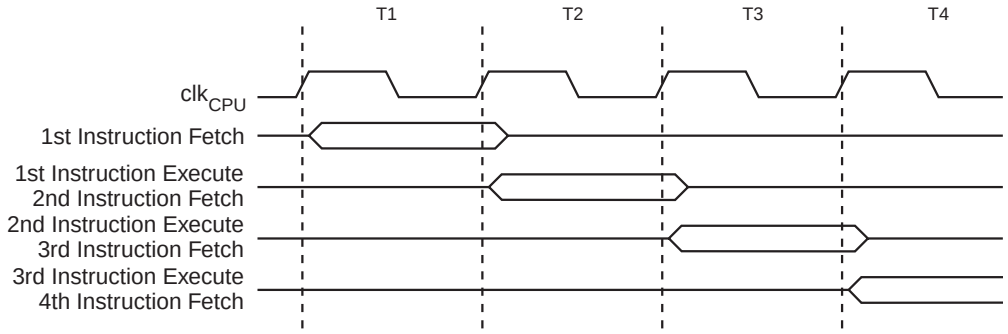
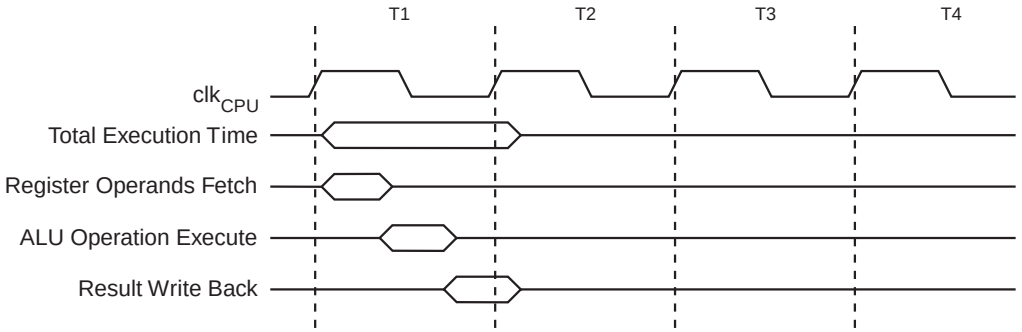


Figure 7 演示的是寄存器文件内部访问时序。在一个时钟周期里，ALU 可以同时对两个寄存器操作数进行操作，同时将结果保存到目的寄存器中去。

Figure 7. 单时钟周期 ALU 操作



复位与中断处理

AVR 有不同的中断源。每个中断和复位在程序空间都有独立的中断向量。所有的中断事件都有自己的使能位。当使能位置位，且状态寄存器的全局中断使能位 I 也置位时，中断可以发生。

程序存储区的最低地址缺省为复位向量和中断向量。完整的向量列表请参见 P42“ 中断 ”。列表也决定了不同中断的优先级。向量所在的地址越低，优先级越高。 RESET 具有最高的优先级，第二个为 INT0 – 外部中断请求 0，详见 P42“ 中断 ”。

任一中断发生时全局中断使能位 I 被清零，从而禁止了所有其他的中断。用户软件可以在中断程序里置位 I 来实现中断嵌套。此时所有的中断都可以中断当前的中断服务程序。执行 RETI 指令后 I 自动置位。

从根本上说有两种类型的中断。第一种由事件触发并置位中断标志。对于这些中断，程序计数器跳转到实际的中断向量以执行中断处理程序，同时硬件将清除相应的中断标志。中断标志也可以通过对其写 ”1” 的方式来清除。当中断发生后，如果相应的中断使能位为 "0"，则中断标志位置位，并一直保持到中断执行，或者被软件清除。类似的，如果全局中断标志被清零，则所有已发生的中断都不会被执行，直到 I 置位。然后挂起的各个中断按中断优先级依次执行。

第二种类型的中断则是只要中断条件满足，就会一直触发。这些中断不需要中断标志。若中断条件在中断使能之前就消失了，中断不会被触发。

AVR 退出中断后总是回到主程序并至少执行一条指令才可以去执行其他被挂起的中断。

要注意的是，进入中断服务程序时状态寄存器不会自动保存，中断返回时也不会自动恢复。这些工作必须由用户通过软件来完成。

使用 CLI 指令来禁止中断时，中断禁止立即生效。没有中断可以在执行 CLI 指令后发生，即使它是在执行 CLI 指令的同时发生的。下面的例子说明了如何在写 EEPROM 时使用这个指令来防止中断发生以避免对 EEPROM 内容的可能破坏。

汇编代码例程

```
in  r16, SREG     ; 保存SREG 值
cli    ; 禁止中断
sbi EECR, EEMWE   ; 启动 EEPROM 写操作
sbi EECR, EEWE
out SREG, r16     ; 恢复SREG (I 位)
```

C 代码例程

```
char cSREG;
cSREG = SREG;/* 保存SREG 值 */
/* 禁止中断 */
__disable_interrupt();
EECR |= (1<<EEMWE); /* 启动 EEPROM 写操作 */
EECR |= (1<<EEWE);
SREG = cSREG; /* 恢复SREG (I 位)*/
```

使用SEI指令使能中断时,紧跟其后的第一条指令在执行任何中断之前一定会首先得到执行。

汇编代码例程

```
sei   ; 置位全局中断使能标志
sleep ; 进入休眠模式，等待中断发生
; 注意： 在执行任何被挂起的中断之前 MCU 将首先进入休眠模式
```

C 代码例程

```
_SEI(); /* 置位全局中断使能标志 */
_SLEEP(); /* 进入休眠模式，等待中断发生 */
/* 注意： 在执行任何被挂起的中断之前 MCU 将首先进入休眠模式 */
```

中断响应时间

AVR 中断响应时间最少为 4 个时钟周期。4 个时钟周期后,程序跳转到实际的中断处理例程。在这 4 个时钟期期间 PC 自动入栈。在通常情况下,中断向量为一个跳转指令,此跳转需要 3 个时钟周期。如果中断在一个多时钟周期指令执行期间发生,则在此多周期指令执行完毕后 MCU 才会执行中断程序。若中断发生时 MCU 处于休眠模式,中断响应时间还需增加 4 个时钟周期。此外还要考虑到不同的休眠模式所需要的启动时间。这个时间不包括在前面提到的时钟周期里。

中断返回需要 4 个时钟。在此期间 PC(两个字节) 将被弹出栈,堆栈指针加二,状态寄存器 SREG 的 I 置位。

AVR ATtiny2313 存储器

本节讲述 ATtiny2313 的存储器。AVR 结构具有两个主要的存储器空间 : 数据存储器空间和程序存储器空间。此外， ATtiny2313 还有 EEPROM 存储器以保存数据。这三个存储器空间都为线性的平面结构。

系统内可编程的 Flash 程序存储器

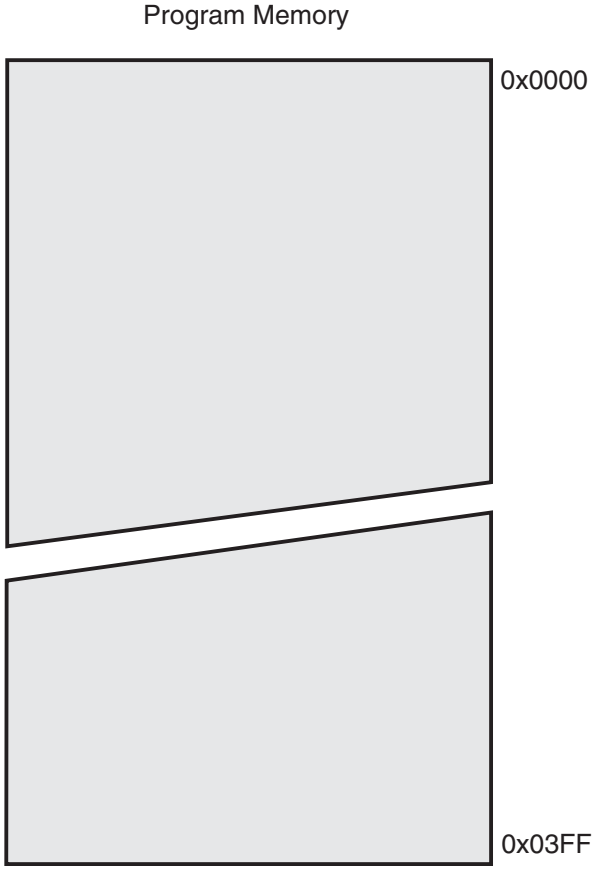
ATtiny2313 具有 2K 字节的片上系统内编程 Flash，用于存放程序指令代码。因为所有的 AVR 指令为 16 位或 32 位，故而 Flash 组织成 1K x 16 位的形式。

Flash存储器至少可以擦写10,000次。ATtiny2313的程序计数器(PC)为10位，因此可以寻址 1K 的程序存储器空间。P150“ 存储器编程 ” 详述了如何用 SPI 接口实现对 Flash 的串行下载。

常数可以保存于整个程序存储器地址空间 (参考 LPM 加载程序存储器指令的说明)。

取指与执行时序图请参见 P10“ 指令执行时序 ” 。

Figure 8. 程序存储器映像



SRAM 数据存储器

Figure 9 给出了 ATtiny2313 SRAM 空间的组织结构。

前 224 个数据存储器包括了寄存器文件、I/O 存储器、扩展 I/O 存储器及内部数据 SRAM。起始的32个地址为寄存器文件与64个标准I/O存储器，接着是128字节的内部数据SRAM。

数据存储器的寻址方式分为 5 种：直接寻址、带偏移量的间接寻址、间接寻址、带预减量的间接寻址和带后增量的间接寻址。寄存器文件中的寄存器 R26 到 R31 为间接寻址的指针寄存器。

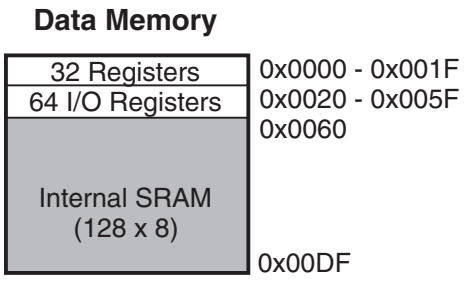
直接寻址范围可达整个数据区。

带偏移量的间接寻址模式能够寻址到由寄存器 Y 和 Z 给定的基址附近的 63 个地址。

在自动预减和后加的间接寻址模式中，寄存器 X、 Y 和 Z 自动增加或减少。

ATtiny2313的全部32个通用寄存器、64个I/O寄存器及128个字节的内部数据SRAM可以通过所有上述的寻址模式进行访问。寄存器文件的描述见 P8“ 通用寄存器文件 ”。

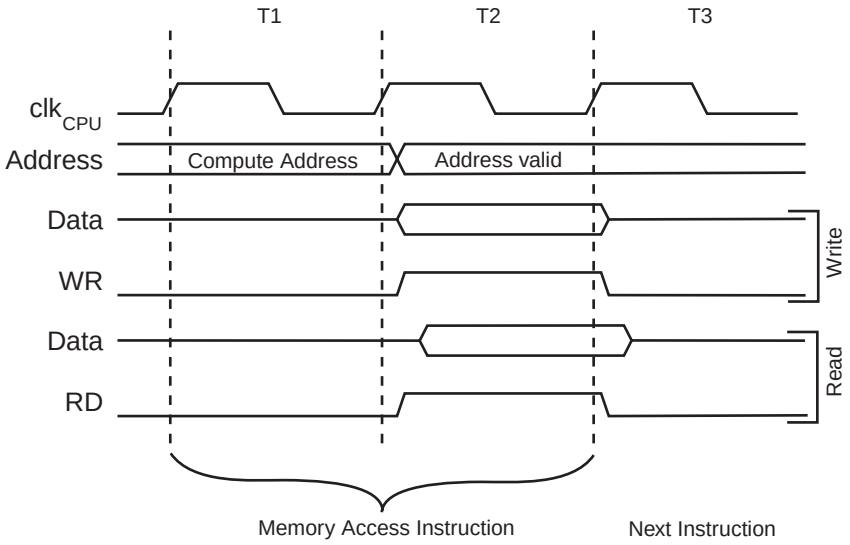
Figure 9. 数据存储器映像



数据存储器访问时间

本节说明访问内部存储器的时序。如 Figure 10 所示，内部数据 SRAM 访问时间为两个 clk_{CPU} 时钟。

Figure 10. 片上 SRAM 存取周期



EEPROM 数据存储器

ATtiny2313 包含 128 字节的 EEPROM 数据存储器。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 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。详见 EEPROM 的串行数据下载。

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 1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 V_{CC} 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 P19 “ 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。详见关于 EEPROM 控制寄存器的说明。

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

Bit	7	6	5	4	3	2	1	0	
	–	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEAR
读 / 写	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	X	X	X	X	X	X	X	

- **Bit 7 – Res: 保留**

保留位，读操作返回值为零。

- **Bits 6..0 – EEAR6..0: EEPROM 地址**

EEPROM地址寄存器EEARL指定了128 字节的EEPROM空间。EEPROM地址是线性的，从 0 到 127。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。

EEPROM 数据寄存器 - EEDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	EEDR
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	

- **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 EEARL 读取的数据。

EEPROM 控制寄存器 - EECR

Bit	7	6	5	4	3	2	1	0	
	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	EECR
读 / 写	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	X	X	0	0	X	0	

- **Bits 7..6 – Res: 保留**

保留位，读操作返回值为零。

- **Bits 5, 4 – EEPM1 与 EEPM0: EEPROM 编程模式位**

设置编程模式位定义当对 EEPE 写入时触发哪种编程方式。可能在一个时钟周期中数据编程 (擦除旧值写入新值) 或将擦除与写操作分为两步。不同模式的编程时间见 Table 1。当 EEPE 置位，对 EEPMn 的写操作将忽略。复位时，除非 EEPROM 处于编程状态，否则 EEPMn 位将复位为 0b00。

Table 1. EEPROM 模式位

EEPM1	EEPM0	编程时间	操作
0	0	3.4 ms	擦除与写入操作在一个时钟周期中完成
0	1	1.8 ms	只擦除
1	0	1.8 ms	只写入
1	1	–	保留

- **Bit 3 – EERIE: EEPROM 就绪中断使能**

若 SREG 的 I 为 "1"，则置位 EERIE 将使能 EEPROM 就绪中断。清零 EERIE 则禁止此中断。当非易失性存储器编程就绪时，EEPROM 就绪中断位产生中断。

- **Bit 2 – EEMPE: EEPROM 主机编程使能**

EEMPE 位决定 EEPE 写入 "1” 是否有效。

当 EEMPE 为 "1" 时，在四个时钟周期内设置 EEPE 将会在 EEPROM 指定的位置编程；若 EEMPE 为 "0“，设置 EEPE 无效。当 EEMPE 由软件写入 "1”，则在四个时钟周期后由硬件清零。

- **Bit 1 – EEPE: EEPROM 编程使能**

EEPE 为 EEPROM 的编程使能信号。当 EEPE 为 "1"，通过 EEPMn 位的设置，将会对 EEPROM编程。在EEPE写入逻辑"1"前，EEMPE位必须写入"1"，否则不会出现EEPROM写操作。当写访问时间结束， EEPE 位由硬件清零。当 EEPE 置位，CPU 在执行指令前终止两个时钟周期。

- **Bit 0 – EERE: EEPROM 读使能**

EERE为EEPROM读操作的使能信号。当EEPROM地址设置好之后，需置位EERE以便将数据读入 EEARL。EEPROM 数据的读取只需要一条指令，且无需等待。读取 EEPROM 后 CPU 要停止 4 个时钟周期才可以执行下一条指令。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。

基本字节编程

使用基本字节编程是最简单的模式。当对 EEPROM 写入一个字节，用户必须将地址写入 EEARL 寄存器，将数据写入 EEDR 寄存器。若 EEPMn 位为零，对 EEPE 的写操作 (在对 EEMPE 写完后的四个时钟周期内) 将触发擦除 / 写入操作。擦除与写入操作在一个时钟周期内完成，整个编程时间见 Table 1。 EEPE 位会保持置位，直到擦除与写入操作完成。而当芯片处于编程状态时，不会进行其他 EEPROM 操作。

分离字节编程

可以将擦除与写入操作分为两个周期。若系统需要对一些有限的时间缩短访问时间 (尤其若电源电压下降) 该方式有效。使用该方式时，必须在写入操作前先进行擦除操作。但由于擦除与写入操作是分离的，有可能当系统允许进行时间临界操作时 (尤其在掉电后) 进行擦除操作。

擦除

擦除一个字节，地址必须写入 EEARL。若 EEPMn 为 0b01，对 EEPE 写入 (在对 EEMPE 写完后的四个时钟周期内) 将只触发擦除操作 (编程时间见 Table 1)。 EEPE 位会保持到擦除操作完成。而当芯片处于编程状态时，不会进行其他 EEPROM 操作。

写入

写入时，用户必须将地址写入 EEARL，将数据写入 EEDR。若 EEPMn 为 0b10，对 EEPE 写入 (在对 EEMPE 写完后的四个时钟周期内) 将只触发写入操作 (编程时间见 Table 1)。EEPE 位会保持到擦除操作完成。若在写入前数据没有擦除，则认为写入数据丢失。当芯片处于编程状态时，不会进行其他 EEPROM 操作。

EEPROM访问使用标定振荡器定时。振荡器频率见P25“振荡器标定寄存器 - OSCCAL” 。

下面的代码分别用汇编和 C 函数说明如何实现 EEPROM 的擦除、写入或基本写入。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_write:
  ; 等待上一次写操作结束
  sbic EECR,EEPE
  rjmp EEPROM_write
  ; 设置编程模式
  ldi  r16, (0<<EEPM1)|(0<<EEPM0)
  out  EECR, r16
  ; 设置地址寄存器 r17
  out EEARL, r17
  ; 将数据写入数据寄存器 (r16)
  out EEDR,r16
  ; 置位 EEMWE
  sbi EECR,EEMWE
  ; 置位 EEWE 以启动写操作
  sbi EECR,EEWE
  ret
```

C 代码例程

```
void EEPROM_write(unsigned char ucAddress, unsigned char ucData)
{
  /* 等待上一次写操作结束 */
  while(EECR & (1<<EEPE))
    ;
  /* 设置编程模式 */
  EECR = (0<<EEPM1)|(0>>EEPM0)
  /* 设置地址与数据寄存器 */
  EEARL = ucAddress;
  EEDR = ucData;
  /* 置位 EEMWE */
  EECR |= (1<<EEMWE);
  /* 置位 EEWE 以启动写操作 */
  EECR |= (1<<EEWE);
}
```

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 r17
  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 char ucAddress)
{
  /* 等待上一次写操作结束 */
  while(EECR & (1<<EEPE))
    ;
  /* 设置地址寄存器 */
  EEARL = ucAddress;
  /* 设置 EERE 以启动读操作 */
  EECR |= (1<<EERE);
  /* 自数据寄存器返回数据 */
  return EEDR;
}
```

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。

I/O 存储器

ATtiny2313 的 I/O 空间定义见 P202“ 寄存器概述 ”。

ATtiny2313所有的I/O及外设都被放置于I/O空间。所有的I/O位置都可以通过LD/LDS/LDD与 ST/STS/STD 指令来访问，在 32 个通用工作寄存器和 I/O 之间传输数据。地址为 0x00 - 0x1F的I/O寄存器还可用SBI和CBI指令直接进行位寻址，而SBIS和SBIC则用来检查某一位的值。更多内容请参见指令集。使用 IN 和 OUT 指令时地址必须在 0x00 - 0x3F 之间。如果要象 SRAM 一样通过 LD 和 ST 指令访问 I/O 寄存器，相应的地址要加上 0x20。

为了与后续产品兼容，保留未用的未应写 "0"，而保留的 I/O 寄存器则不应进行写操作。

一些状态标志位的清除是通过写 "1" 来实现的。要注意的是，与其他大多数 AVR 不同，CBI 和 SBI 指令只能对某些特定的位进行操作，因而可以用于包含这些状态标志的寄存器。 CBI 与 SBI 指令只对 0x00 到 0x1F 的寄存器有效。

I/O 和外设控制寄存器在后续其他章节进行介绍。

通用功能 I/O 寄存器

ATtiny2313包含3个通用功能I/O寄存器。这些寄存器可用来存储信息，特别是可用来存储全局变量及状态标志。在 0x00 - 0x1F 范围内的通用功能 I/O 寄存器可使用 SBI、 CBI、SBIS 与 SBIC 指令直接对位进行操作。

通用功能 I/O 寄存器 2 - GPIOR2

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	GPIOR2
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	

通用功能 I/O 寄存器 1 - GPIOR1

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	GPIOR1
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	

通用功能 I/O 寄存器 0 - GPIOR0

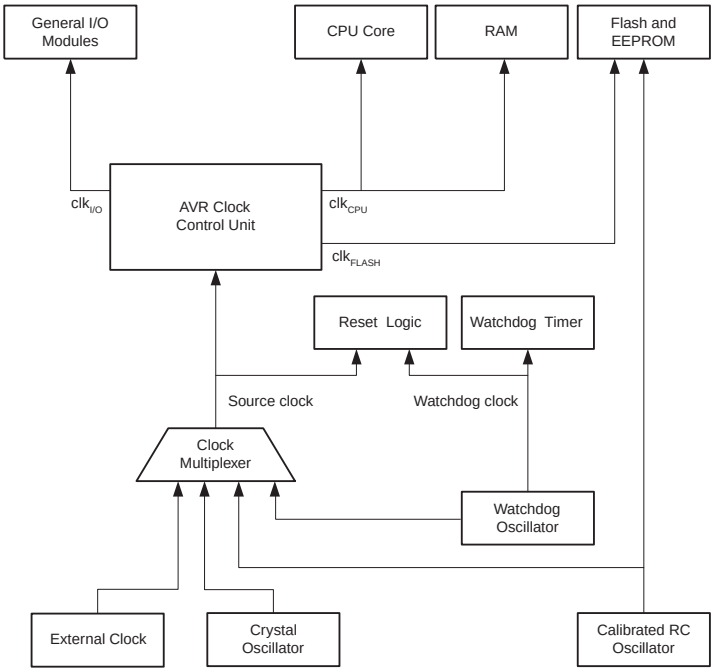
Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	GPIOR0
读 / 写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

系统时钟及时钟选项

时钟系统及其分布

Figure 11为AVR的主要时钟系统及其分布。这些时钟并不需要同时工作。为了降低功耗，可以通过使用不同的睡眠模式来禁止无需工作的模块的时钟，详见 P29“ 电源管理及睡眠模式 ”。时钟系统详见 Figure 11。

Figure 11. 时钟分布



CPU 时钟 - clk_{CPU}

CPU时钟与操作AVR内核的子系统相连，如通用寄存器文件、状态寄存器及保存堆栈指针的数据存储器。终止 CPU 时钟将使内核停止工作和计算。

I/O 时钟 - $clk_{I/O}$

I/O 时钟用于主要的 I/O 模块，如定时器 / 计数器、USART。I/O 时钟还用于外部中断模块。要注意的是有些外部中断由异步逻辑检测，因此即使 I/O 时钟停止了这些中断仍然可以得到监控。同时要注意当 $clk_{I/O}$ 停止， USI 模块的启动状态检测为异步执行，在所有睡眠模式下使能 USI 启动状态检测。

Flash 时钟 - clk_{FLASH}

Flash 时钟控制 Flash 接口的操作。此时钟通常与 CPU 时钟同时挂起或激活。

时钟源

ATtiny2313芯片有如下几种通过Flash熔丝位进行选择的时钟源。时钟输入到AVR时钟发生器，再分配到相应的模块。

Table 2. 时钟源选择 (1)

器件时钟选项	CKSEL3..0
外部时钟	0000
标定为 4MHz 的内部 RC 振荡器	0010
标定为 8MHz 的内部 RC 振荡器	0100

Table 2. 时钟源选择 (1)

器件时钟选项	CKSEL3..0
外部时钟	0000
128kHz 的看门狗振荡器	0110
外部振荡器	1000 - 1111
保留	0001/0011/0101/0111

Note: 1. 对于所有的熔丝位，“1” 表示未编程，“0” 代表已编程。

不同的时钟选项将在后续部分进行介绍。当 CPU 自掉电模式唤醒之后，被选择的时钟源用来为启动过程定时，保证振荡器在开始执行指令之前进入稳定状态。当 CPU 从复位开始工作时，还有额外的延迟时间以保证在 MCU 开始正常工作之前电源达到稳定电平。这个启动时间的定时由看门狗振荡器完成。看门狗溢出时间所对应的 WDT 振荡器周期数列于 Table 3。看门狗振荡器的频率是压控的，见 P171“ATtiny2313 典型特性 ”。

Table 3. 看门狗振荡器周期数

典型的溢出时间 (V_{CC} = 5.0V)	典型的溢出时间 (V_{CC} = 3.0V)	时钟周期数
4.1 ms	4.3 ms	4K (4,096)
65 ms	69 ms	64K (65,536)

缺省时钟源

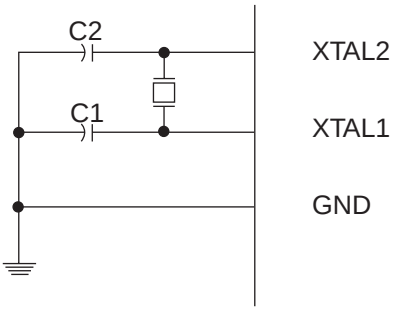
器件出厂时 CKSEL = “0010”， SUT = “10”，且 CKDIV8 编程。这个缺省设置的时钟源是内部 RC 振荡器，初始系统时钟预分频为 8，启动时间为最长。这种设置保证用户可以通过 ISP 或并行编程器得到所需的时钟源。

晶体振荡器

XTAL1 与 XTAL2 分别为用作片内振荡器的反向放大器的输入和输出，如 P22Figure 12 所示。这个振荡器可以使用石英晶体，也可以使用陶瓷谐振器。

C1 和 C2 的数值要一样。最佳的数值与使用的晶体或谐振器有关，还与杂散电容和环境的电磁噪声有关。P23Table 4 给出了针对晶体选择电容的一些指南。对于陶瓷谐振器，应该使用厂商提供的数值。

Figure 12. 晶体振荡器连接图



振荡器可以工作于三种不同的模式，每一种都有一个优化的频率范围。工作模式通过熔丝位 CKSEL3..1 来选择，如 Table 4 所示。

Table 4. 晶体振荡器工作模式

CKSEL3..1	频率范围 (1) (MHz)	使用晶体时电容 C1 和 C2 的推荐范围 (pF)
100(2)	0.4 - 0.9	–
101	0.9 - 3.0	12 - 22
110	3.0 - 8.0	12 - 22
111	8.0 -	12 - 22

Notes: 1. 频率范围仅是初步值，实际值在测试中。
2. 此选项不适用于晶体，只能用于陶瓷谐振器。

如 Table 5 所示，熔丝位 CKSEL0 以及 SUT1..0 用于选择启动时间。

Table 5. 晶体振荡器时钟选项对应的启动时间

CKSEL0	SUT1..0	掉电与节电模式下的启动时间	复位时额外的延迟时间 (V_{CC} = 5.0V)	推荐用法
0	00	258 CK(1)	14CK + 4.1 ms	陶瓷谐振器，电源快速上升
0	01	258 CK(1)	14CK + 65 ms	陶瓷谐振器，电源缓慢上升
0	10	1K CK(2)	14CK	陶瓷谐振器， BOD 使能
0	11	1K CK(2)	14CK + 4.1 ms	陶瓷谐振器，电源快速上升
1	00	1K CK(2)	14CK + 65 ms	陶瓷谐振器，电源缓慢上升
1	01	16K CK	14CK	石英振荡器，BOD 使能
1	10	16K CK	14CK + 4.1 ms	石英振荡器，电源快速上升
1	11	16K CK	14CK + 65 ms	石英振荡器，电源慢速上升

Notes: 1. 这些选项只能用于工作频率不太接近于最大频率，而且启动时的频率稳定性对于应用而言不重要的情况。
2. 这些选项是为陶瓷谐振器设计的，可以保证启动时频率足够稳定。若工作频率不太接近于最大频率，而且启动时的频率稳定性对于应用而言不重要时也适用于晶体。

标定的片内 RC 振荡器

标定的片内 RC 振荡器提供了固定的 8.0 MHz 的时钟。这些频率都是 3V、 25°C 下的标称数值。若频率超出器件标称值，必须对 CKDIV8 熔丝位编程，以在启动阶段对内部频率 8分频，器件在出厂时已经对CKDIV8熔丝位编程。这个时钟也可以作为系统时钟，只要按照 Table 6 对熔丝位 CKSEL 进行编程即可。选择这个时钟之后就无需外部器件了。复位时硬件将标定字节加载到 OSCCAL 寄存器，自动完成对 RC 振荡器的标定。在 3V、25°C 时，这种标定可以提供标称频率 ± 10% 的精度。使用在 www.atmel.com/avr 中所述的标定方法，可能会在任何电压及任何温度下使精度达到 ± 2%。当使用这个振荡器作为系统时钟时，看门狗仍然使用自己的看门狗定时器作为溢出复位的依据。更多的有关标定数据的信息请参见 P152“ 校准字节 ”。

Table 6. 片内标定的 RC 振荡器工作模式

CKSEL3..0	标称频率
0010 - 0011	4.0 MHz(1)
0100 - 0101	8.0 MHz

Note: 1. 出厂时的设置。

选择了这个振荡器之后，启动时间由熔丝位 SUT 确定，如 Table 7 所示。

Table 7. 内部标定 RC 振荡器的启动时间

SUT1..0	掉电模式与省电模式的启动时间	复位时的额外延迟时间 (V_{CC} = 5.0V)	推荐用法
00	6 CK	14CK	BOD 使能

Table 7. 内部标定 RC 振荡器的启动时间

SUT1..0	掉电模式与省电模式的启动时间	复位时的额外延迟时间 (V_{CC} = 5.0V)	推荐用法
01	6 CK	14CK + 4.1 ms	电源快速上升
10(1)	6 CK	14CK + 65 ms	电源缓慢上升
11	保留		

Note: 1. 出厂时的设置。

振荡器标定寄存器 - OSCCAL

Bit	7	6	5	4	3	2	1	0	
	–	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	OSCCAL
读 / 写	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	标定数据								

- **Bits 6..0 – CAL6..0: 振荡器标定值**

将标定数据写入这个地址可以对内部振荡器进行调节以消除由于生产工艺所带来的振荡器频率偏差。这在复位时自动完成。当 OSCCAL 为零时振荡器以最低频率工作。当对其写如不为零的数据时内部振荡器的频率将增长。写入 0x7F 即得到最高频率。标定的振荡器用来为访问 EEPROM 和 Flash 定时。有写 EEPROM 和 Flash 的操作时不要将频率标定到超过标称频率的 10% ,否则写操作有可能失败。要注意振荡器只对 8.0/4.0 MHz 这两种频率进行了标定，其他频率则无法保证，见 Table 8。

为保证 MCU 稳定工作，当标定内部 RC 振荡器时避免大幅度改变标称值。工作频率突变超过 2% 将会产生异常现象。每次对 OSCCAL 寄存器中值的改变不应超过 0x20。

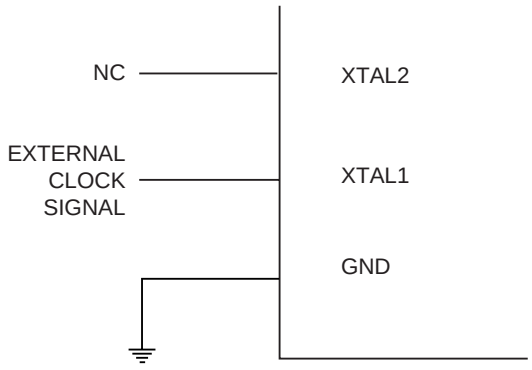
Table 8. 内部 RC 振荡器频率范围

OSCCAL 数值	最小频率，标称频率的百分比	最大频率，标称频率的百分比
0x00	50%	100%
0x3F	75%	150%
0x7F	100%	200%

外部时钟

为了从外部时钟源驱动芯片， XTAL1 必须如 Figure 13 所示的进行连接。同时，熔丝位 CKSEL 必须编程为 “0000”。

Figure 13. 外部时钟配置图



选择了这个振荡器之后，启动时间由熔丝位 SUT 确定，如 Table 10 所示。

Table 9. 晶振时钟频率

CKSEL3..0	频率范围
0000 - 0001	0 - 16 MHz

Table 10. 外部时钟的启动时间

SUT1..0	掉电模式和省电模式的启动时间	复位时的额外延迟时间 (V_{CC} = 5.0V)	推荐用法
00	6 CK	14CK	BOD 使能
01	6 CK	14CK + 4.1 ms	电源快速上升
10	6 CK	14CK + 65 ms	电源缓慢上升
11	保留		

为了保证 MCU 能够稳定工作，不能突然改变外部时钟源的振荡频率。工作频率突变超过 2% 将会产生异常现象。应该在 MCU 保持复位状态时改变外部时钟的振荡频率。

注意，系统时钟预分频器可用来实现内部时钟频率运行时间改变且保证稳定工作。

128 kHz 片内振荡器

The 128 kHz Internal Oscillator is a low power Oscillator providing a clock of 128 kHz. The frequency is nominal at 3 V and 25°C. This clock may be selected as the system clock by programming the CKSEL Fuses to “0110 - 0111”.128 kHz 片内振荡器为提供时钟频率为 128 kHz 的低功耗振荡器。该频率为在 3V、25°C 下的标定值。通过将 CKSEL 熔丝位编程为 “0110 - 0111”。

当选择该时钟源，启动时间由 Table 11 中所示的 SUT 熔丝位决定。

Table 11. 128 kHz 片内振荡器启动时间

SUT1..0	掉电模式和省电模式的启动时间	复位时的额外延迟时间	推荐用法
00	6 CK	14CK	BOD 使能
01	6 CK	14CK + 4 ms	电源快速上升
10	6 CK	14CK + 64 ms	电源缓慢上升
11	保留		

时钟预分频寄存器 - CLKPR

Bit	7	6	5	4	3	2	1	0	
	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	CLKPR
读 / 写	R/W	R	R	R	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	见位说明				

- **Bit 7 – CLKPCE: 时钟预分频器变化使能**

CLKPCE 位必须置"1”使能 CLKPS位。只有当CLKPR寄存器的其他位同时写"0”时，CLKPCE 位改变。CLKPCE 在写入四个周期后或当 CLKPS 位写入后由硬件清零。在暂停周期中重新写 CLKPCE 位，既不扩展暂停周期，也不清除 CLKPCE 位。

- **Bits 3..0 – CLKPS3..0: 时钟预分频器选择位 3 - 0**

这几位定义所选时钟源与内部系统时钟所分频因子。这几位写入运行时间来改变时钟频率以适应运行需要。当作为 MCU 主时钟输入分频器，使用分频因子时，所有同步外设速度将会下降。分频因子见 Table 12。

为避免时钟频率的无意改变，对 CLKPS 位的写入必须按照如下步骤进行：

1. 将 CLKPCE 位写 "1”，而 CLKPR 寄存器的其他位写 "0”。
2. 在四个时钟周期内，将期望值写入 CLKPS，并在 CLKPCE 位写 "0”。

在改变预分频器设置时必须禁止中断，以保证在写入过程中不会出现中断。

CKDIV8熔丝位决定CLKPS位的初始值。若CKDIV8未编程，CLKPS位复位为“0000”；若CKDIV8 已编程，CLKPS 位复位为 “0011”，给出启动时分频因子为 8。若所选时钟源频率大于当前工作状态下器件最大频率时，应利用该特性分频。注意， CLKPS 位写入值不受CKDIV8 熔丝位设置影响。若所选时钟源频率大于当前工作状态下器件最大频率，应用程序必须保证选择一个足够大的分频因子。芯片出厂时 CKDIV8 熔丝位已编程。

Table 12. 时钟预分频器选择

CLKPS3	CLKPS2	CLKPS1	CLKPS0	时钟分频因子
0	0	0	0	1
0	0	0	1	2
0	0	1	0	4
0	0	1	1	8
0	1	0	0	16

Table 12. 时钟预分频器选择

CLKPS3	CLKPS2	CLKPS1	CLKPS0	时钟分频因子
0	1	0	1	32
0	1	1	0	64
0	1	1	1	128
1	0	0	0	256
1	0	0	1	保留
1	0	1	0	保留
1	0	1	1	保留
1	1	0	0	保留
1	1	0	1	保留
1	1	1	0	保留
1	1	1	1	保留

电源管理及睡眠模式

睡眠模式可以使应用程序关闭 MCU 中没有使用的模块，从而降低功耗。 AVR 具有不同的睡眠模式，允许用户根据自己的应用要求实施剪裁。

进入睡眠模式的条件是置位寄存器 MCUCR 的 SE，然后执行 SLEEP 指令。具体哪一种模式 (空闲模式、掉电模式 或 Standby 模式) 由 MCUCR 的 SM1 和 SM0 决定 , 如 Table 13所示。使能的中断可以将进入睡眠模式的MCU唤醒。经过启动时间 , 外加4个时钟周期后， MCU 就可以运行中断例程了。然后返回到 SLEEP 的下一条指令。唤醒时不会改变寄存器文件和 SRAM 的内容。如果在睡眠过程中发生了复位 , 则 MCU 唤醒后从中断向量开始执行。

P21Figure 11 介绍了 ATtiny2313 不同的时钟系统及其分布。此图在选择合适的睡眠模式时非常有用。

MCU 控制寄存器 - MCUCR

睡眠模式控制寄存器就电源管理包含以下控制位。

Bit	7	6	5	4	3	2	1	0	
	PUD	SM1	SE	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
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	

- **Bits 6, 4 – SM1..0: 休眠模式选择位 1 与 0**

如 Table 13 所示，这些位用于选择具体的休眠模式。

Table 13. 休眠模式选择

SM1	SM0	休眠模式
0	0	空闲模式
0	1	掉电模式
1	1	掉电模式
1	0	Standby 模式

Note: 1. 建议 Standby 模式只在外部晶体或谐振器中使用。

- **Bit 5 – SE: 休眠使能**

为了使 MCU 在执行 SLEEP 指令后进入休眠模式， SE 必须置位。为了确保进入休眠模式是程序员的有意行为 , 建议仅在 SLEEP 指令的前一条指令置位 SE。MCU 一旦唤醒立即清除 SE。

空闲模式

当 SM1..0 为 00 时， SLEEP 指令将使 MCU 进入空闲模式。在此模式下， CPU 停止运行 , 而模拟比较器、ADC、定时器 / 计数器、看门狗和中断系统继续工作。这个休眠模式只停止了 clk_{CPU} 和 clk_{FLASH}，其他时钟则继续工作。

象定时器溢出等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。方法是置位模拟比较器控制和状态寄存器 ACSR 的 ACD。如果 ADC 使能，进入此模式后将自动启动一次转换。

掉电模式

当 SM1..0 为 01 或 11 时， SLEEP 指令将使 MCU 进入掉电模式。在此模式下，外部晶体停振，而外部中断、 USI 启动状态检测及看门狗 (如果使能的话) 继续工作。只有外部复位、看门狗复位、 BOD 复位、 USI 启动状态检测、外部电平中断 INT0 或引脚变化中断可以使 MCU 脱离掉电模式。这个睡眠模式停止了所有的时钟，只有异步模块可以继续工作。

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。具体请参见 P57“ 外部中断 ”。

从施加掉电唤醒条件到真正唤醒有一个延迟时间，此时间用于时钟重新启动并稳定下来。唤醒周期与由熔丝位 CKSEL 定义的复位周期是一样的，如 P21“ 时钟源 ” 所示。

Standby 模式

当 SM1..0 为 10 且选择外部晶振或谐振器时钟时， SLEEP 指令将使 MCU 进入 Standby 模式。这一模式与掉电模式唯一的不同之处在于振荡器继续工作。其唤醒时间只需要 6 个时钟周期。

Table 14. 在不同睡眠模式下活动的时钟以及唤醒源

	工作的时钟			振荡器	唤醒源			
睡眠模式	clk_{CPU}	clk_{FLASH}	clk_{IO}	使能	INT0, INT1 与引脚变化	USI 启动状态	SPM/EEPROM 就绪	其他 I/O
空闲模式			X	X	X	X	X	X
掉电模式					X(2)	X		
Standby(1)				X	X(2)	X		

Notes: 1. 时钟源为外部晶体或谐振器。
2. 电平中断 INT0

最小化功耗

试图降低 AVR 控制系统的功耗时需要考虑几个问题。一般来说，要尽可能利用睡眠模式，并且使尽可能少的模块继续工作。不需要的功能必须禁止。下面的模块需要特殊考虑以达到尽可能低的功耗。

模拟比较器

在空闲模式时，如果没有使用模拟比较器，可以将其关闭。在其他睡眠模式模拟比较器是自动关闭的。如果模拟比较器使用了内部电压基准源，则不论在什么睡眠模式下都需要关闭它。否则内部电压基准源将一直使能。请参见 P141“ 模拟比较器 ” 以了解如何配置模拟比较器。

掉电检测 BOD

如果系统没有利用掉电检测器 BOD，这个模块也可以关闭。如果熔丝位 BODLEVEL 被编程，从而使能了 BOD 功能，它将在各种休眠模式下继续工作。在深层次的休眠模式下，这个电流将占总电流的很大比重。请参看 P33“ 掉电检测 ” 以了解如何配置 BOD。

片内基准电压

使用 BOD、模拟比较器时可能需要内部电压基准源。若这些模块都禁止了，则基准源也可以禁止。重新使能后用户必须等待基准源稳定之后才可以使用它。如果基准源在休眠过程中是使能的，其输出立即可以使用，详见 P36“ 片内基准电压 ” 。

看门狗定时器

如果系统无需利用看门狗，这个模块也可以关闭。若使能，则在任何休眠模式下都持续工作，从而消耗电流。在深层次的睡眠模式下，这个电流将占总电流的很大比重。请参看 P42“ 中断 ” 以了解如何配置看门狗定时器。

端口引脚

进入休眠模式时，所有的端口引脚都应该配置为只消耗最小的功耗。最重要的是避免驱动电阻性负载。在休眠模式下 I/O 时钟 $clk_{I/O}$ 被停止了，输入缓冲器也禁止了，从而保证输入电路不会消耗电流。在某些情况下输入逻辑是使能的，用来检测唤醒条件。用于此功能的具体引脚请参见 P48“ 数字输入使能和休眠模式 ” 。如果输入缓冲器是使能的，此时输入不能悬空，信号电平也不应该接近 $V_{CC}/2$，否则输入缓冲器会消耗额外的电流。

对于模拟输入引脚，数字输入缓冲应始终禁用。即使在工作模式下，在输入引脚的接近 $V_{CC}/2$ 的模拟信号电平会带来明显的电流 。数字输入缓冲可通过写 DIDR 来禁用，参见 P142“ 数字输入禁用寄存器 - DIDR” 。

系统控制和复位

复位 AVR

复位时所有的 I/O 寄存器都被设置为初始值，程序从复位向量处开始执行。复位向量处的指令必须是绝对跳转 JMP 指令，以使程序跳转到复位处理例程。如果程序永远不利用中断功能，中断向量可以由一般的程序代码所覆盖。Figure 14 为复位逻辑的电路图。Table 15 则定义了复位电路的电气参数。

复位源有效时 I/O 端口立即复位为初始值。此时不要求任何时钟处于正常运行状态。

所有的复位信号消失之后，芯片内部的一个延迟计数器被激活，将内部复位的时间延长。这种处理方式使得在 MCU 正常工作之前有一定的时间让电源达到稳定的电平。延迟计数器的溢出时间通过熔丝位 SUT 与 CKSEL 设定。延迟时间的选择请参见 P21“ 时钟源 ”。

复位源

The ATtiny2313 有 4 个复位源：

- 上电复位。电源电压低于上电复位门限 V_{POT} 时， MCU 复位。
- 外部复位。引脚 $\overline{\text{RESET}}$ 上的低电平持续时间大于最小脉冲宽度时 MCU 复位。
- 看门狗复位。看门狗使能并且看门狗定时器溢出时复位发生。
- 掉电检测复位。掉电检测复位功能使能，且电源电压低于掉电检测复位门限 V_{BOT} 时 MCU 即复位。

Figure 14. 复位逻辑

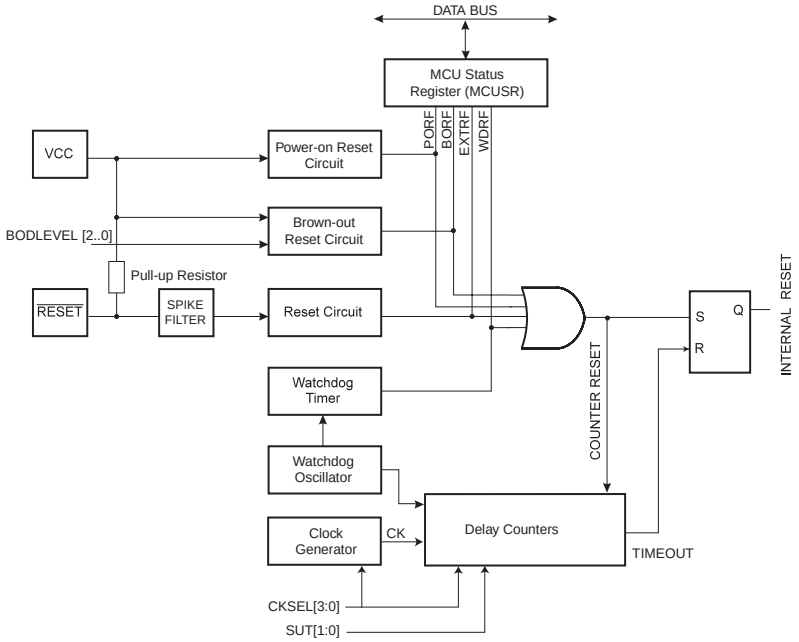


Table 15. 复位特性

符号	参数	条件	最小值 (1)	典型值 (1)	最大值 (1)	单位
V_{POT}	上电复位门限电压 (电压由低到高上升)	T_A = -40 - 85°C		1.2		V
	上电复位门限电压 (电压由高到低跌落)(2)	T_A = -40 - 85°C		1.1		V
V_{RST}	$\overline{\text{RESET}}$ 门限电压	V_{CC} = 1.8 - 5.5V	0.1 V_{CC}		0.9 V_{CC}	V
t_{RST}	$\overline{\text{RESET}}$ 最小脉冲宽度	V_{CC} = 1.8 - 5.5V			2.5	µs

Notes: 1. 这些值仅作为参考，实际值待测试。
2. 电压下降时，只有电压低于 V_{POT} 时复位才会发生。

上电复位

上电复位 (POR) 脉冲由片内检测电路产生。检测电平请参见 Table 15。无论何时 V_{CC} 低于检测电平 POR 即发生。 POR 电路可以用来触发启动复位，或者用来检测电源故障。

POR 电路保证器件在上电时复位。V_{CC} 达到上电门限电压后触发延迟计数器。在计数器溢出之前器件一直保持为复位状态。当 V_{CC} 下降时，只要低于检测门限，RESET 信号立即生效。

Figure 15. MCU 启动过程，$\overline{\text{RESET}}$ 连接到 V_{CC}

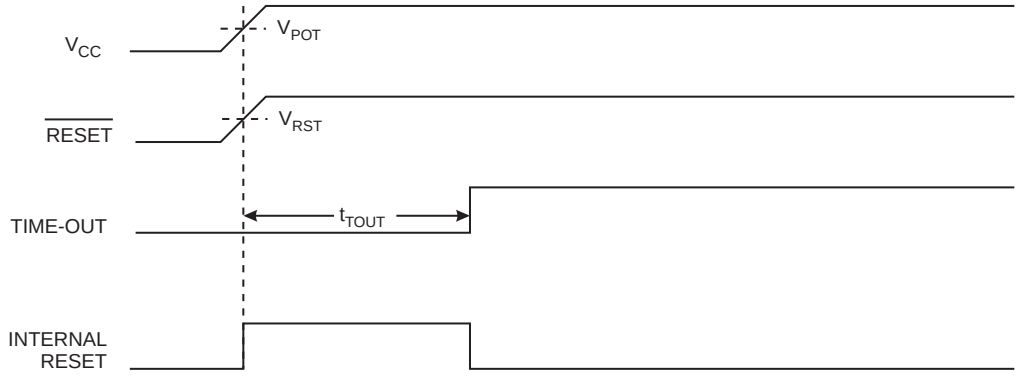
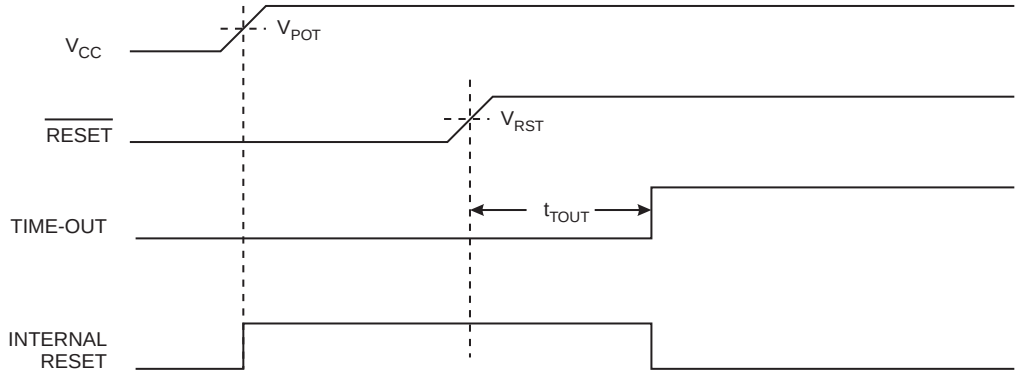


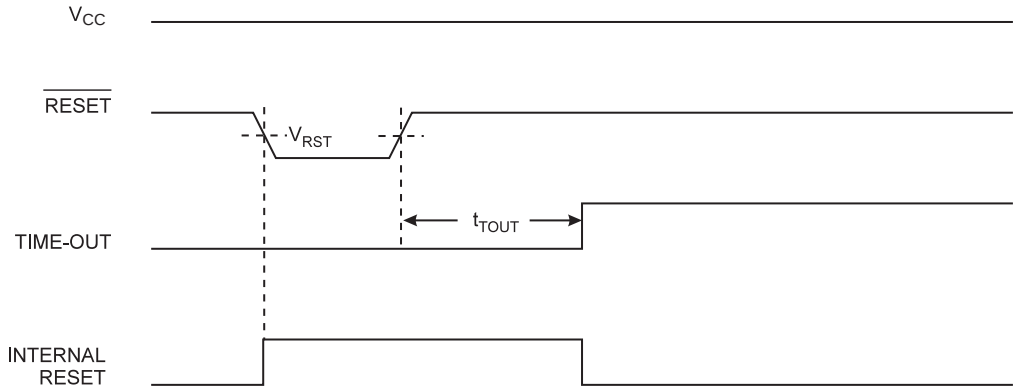
Figure 16. MCU 启动过程，$\overline{\text{RESET}}$ 由外电路控制



外部复位

外部复位由外加于 $\overline{\text{RESET}}$ 引脚的低电平产生。当复位低电平持续时间大于最小脉冲宽度时 (参见 Table 15) 即触发复位过程，即使此时并没有时钟信号在运行。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。延时结束后 MCU 即启动。

Figure 17. 工作过程中发生外部复位



掉电检测

ATtiny2313 具有片内 BOD(Brown-out Detection) 电路，通过与固定的触发电平的对比来检测工作过程中 V_{CC} 的变化。此触发电平通过熔丝位 BODLEVEL 来设定。 BOD 的触发电平具有迟滞功能以消除电源尖峰的影响。这个迟滞功能可以解释为 $V_{BOT+} = V_{BOT} + V_{HYST}/2$ 以及 $V_{BOT-} = V_{BOT} - V_{HYST}/2$。

Table 16. BODLEVEL 熔丝位编码 (1)

BODLEVEL [1..0] 熔丝位	最小 V_{BOT}	典型 V_{BOT}	最大 V_{BOT}	单位
111	BOD 禁用			
110		1.8		V
101		2.7		
100		4.3		
011	保留			
010				
001				
000				

Note: 1. 对某些芯片 V_{BOT} 可能低于其标称的最小电压。对这些芯片，在产品测试时，必须有 $V_{CC} = V_{BOT}$ ，以保证电压低于正常工作电压 V_{CC} 时会有掉电复位。对 ATtiny2313V BODLEVEL = 110，对 ATtiny2313L BODLEVEL = 101 时执行测试。

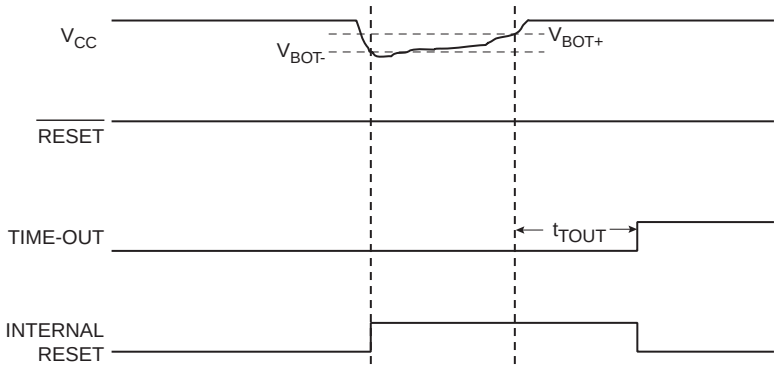
Table 17. 掉电特性

符号	参数	最小值	典型值	最大值	单位
V_{HYST}	BOD 迟滞		50		mV
t_{BOD}	掉电复位最小脉宽		2		µs

当 BOD 使能，一旦 V_{CC} 下降到触发电平以下 (V_{BOT-}，Figure 18)， BOD 复位立即被激发。当 V_{CC} 上升到触发电平以上时 (V_{BOT+}，Figure 18)，延时计数器开始计数，一旦超过溢出时间 t_{TOUT}， MCU 即恢复工作。

如果 V_{CC} 一直低于触发电平并保持如 Table 15 所示的时间 t_{BOD}， BOD 电路将只检测电压跌落。

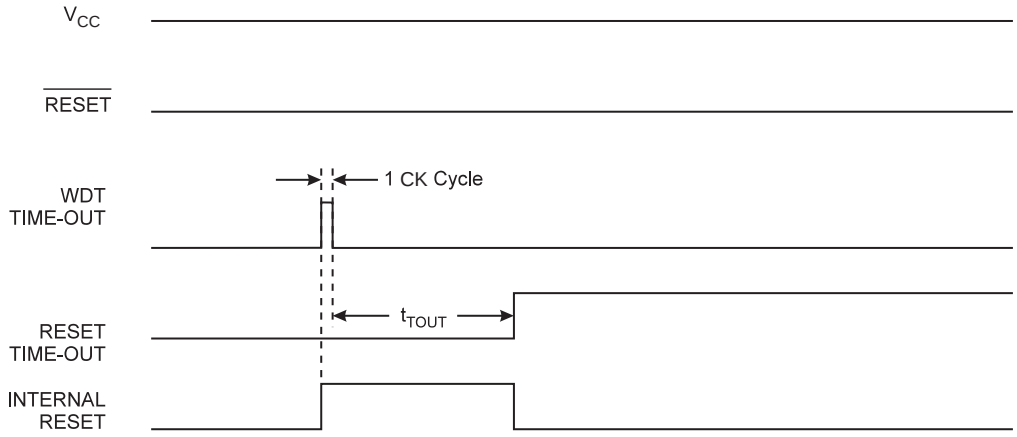
Figure 18. 工作过程中的掉电复位



看门狗复位

看门狗定时器溢出时将产生持续时间为 1 个 CK 周期的复位脉冲。在脉冲的下降沿，延时定时器开始对 t_{TOUT} 记数。请参见看门狗定时器以了解其具体操作过程。

Figure 19. 工作过程中发生看门狗复位



MCU 状态寄存器 - MCUSR

MCU 状态寄存器提供了有关引起 MCU 复位的复位源的信息。

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	WDRF	BORF	EXTRF	PORF	MCUSR
读 / 写	R	R	R	R	R/W	R/W	R/W	R/W	
初始值	0	0	0		参见位说明				

- **Bit 3 – WDRF: 看门狗复位标志**

看门狗复位发生时置位。上电复位将使其清零，也可以通过写 ”0” 来清除。

- **Bit 2 – BORF: 掉电检测复位标志**

掉电检测复位发生时置位。上电复位将使其清零，也可以通过写 ”0” 来清除。

- **Bit 1 – EXTRF: 外部复位标志**

外部复位发生时置位。上电复位将使其清零，也可以通过写 ”0” 来清除。

- **Bit 0 – PORF: 上电复位标志**

上电复位发生时置位。只能通过写 ”0” 来清除。

为了使用这些复位标志来识别复位条件，用户应该尽早读取此寄存器的数据，然后将其复位。如果在其他复位发生之前将此寄存器复位，则后续复位源可以通过检查复位标志来了解。

片内基准电压

ATtiny2313 具有片内能隙基准源，用于掉电检测，或者是作为模拟比较器的输入。

基准电压使能信号和启动时间

电压基准的启动时间可能影响其工作方式。启动时间列于 Table 18.。为了降低功耗，可以控制基准源仅在如下情况打开：

1. BOD 使能 (熔丝位 BODLEVEL [2..0] 被编程)。
2. 能隙基准源连接到模拟比较器 (ACSR 寄存器的 ACBG 置位)。

因此，当 BOD 被禁止时，置位 ACBG 后要在模拟比较器使用前启动基准源。为了降低掉电模式的功耗，用户可以禁止上述三种条件，并在进入掉电模式之前关闭基准源。

Table 18. 内部电压基准源的特性 (1)

符号	参数	条件	最小值	典型值	最大值	单位
V_{BG}	能隙基准源电压	V_{CC} = 2.7V, T_A = 25°C	1.0	1.1	1.2	V
t_{BG}	能隙基准源启动时间	V_{CC} = 2.7V, T_A = 25°C		40	70	µs
I_{BG}	能隙基准源功耗	V_{CC} = 2.7V, T_A = 25°C		15		µA

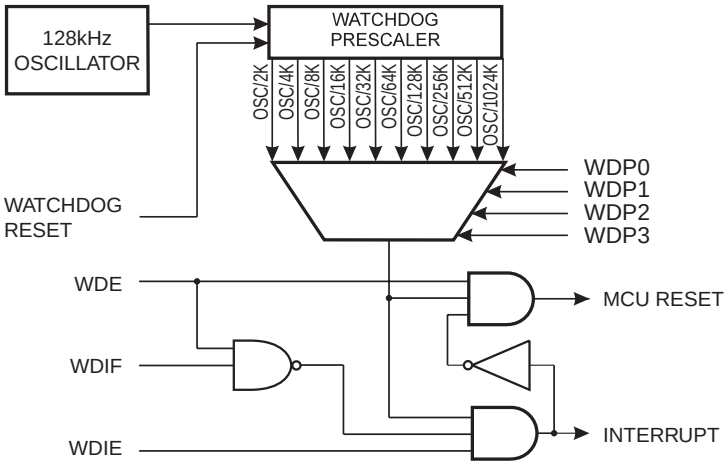
Note: 1. 这些值仅作为参考，实际值待测试。

看门狗定时器

ATtiny2313 有一个增强型的看门狗定时器 (WDT)，其主要特征为：

- **独立的片上振荡器提供时钟**
- **3 种工作模式**
  - **– 中断**
  - **– 系统复位**
  - **– 中断与系统复位**
- **暂停时间从 16ms 到 8s 可选**
- **看门狗熔丝始终处于故障保险模式。**

Figure 20. 看门狗定时器



看门狗定时器由独立的 128 kHz 片内振荡器驱动。当计数器达到给定的溢出值时，WDT 发出中断或系统复位。在正常工作模式下，在计数器达到溢出值前，它需要系统使用看门狗定时器复位指令来重启计数器。若系统没有重启计数器，则会出现中断或系统复位。

在中断模式下，当定时器结束 WDT 发出一个中断。该中断可将芯片从休眠状态中唤醒，也可作为一个通用系统定时器。例如限制最大工作时间，当工作时间超出期望值时发出中断。在系统复位模式下，当定时器结束 WDT 发出复位信号。这是为防止由于错误代码所引起的系统挂起的典型使用。第三种模式，中断与系统复位模式，结合两种模式，首先给出中断，然后转换到系统复位模式。使用该模式，可在系统复位前通过保存临界参数来安全关闭。

WDTON 熔丝位编程将使看门狗定时器进入系统复位模式。对其编程时，系统复位模式位 (WDE) 与中断模式位 (WDTIE) 分别为 1 和 0。

为保证编程安全，必须按照下面顺序来改变看门狗设置：

1. 在一步操作中，同时对 WDCE 位与 WDE 写 "1”。无论 WDE 的初始值是多少，在此必须对其写逻辑 ”1”。
2. 在接着的四个时钟周期内，在 WDE 与 WDP 中写入期望值，但同时要清除 WDCE 位。

下面的例子分别用汇编和 C 语言实现了关闭 WDT 的操作。在此假定中断处于用户控制之下（比如禁止全局中断），因而在执行下面程序时中断不会发生。

汇编代码例程 (1)

```
WDT_off:
  ; 关闭全局中断
  cli
  ; WDT 复位
  wdr
  ; 清除 MCUSR 寄存器中 WDRF
  in    r16, MCUSR
  andi  r16, (0xff & (0<<WDRF))
  out   MCUSR, r16
  ; 在 WDCE 与 WDE 中写逻辑 1
  ; 保持旧预分频器设置防止无意暂停
  in    r16, WDTCR
  ori   r16, (1<<WDCE) | (1<<WDE)
  out   WDTCR, r16
  ; 关闭 WDT
  ldi   r16, (0<<WDE)
  out   WDTCR, r16
  ;开启全局中断
  sei
  ret
```

C 代码例程 (1)

```
void WDT_off(void)
{
  __disable_interrupt();
  __watchdog_reset();
  /* 清除 MCUSR 寄存器中 WDRF*/
  MCUSR &= ~(1<<WDRF);
  /* 在 WDCE 与 WDE 中写逻辑 1*/
  /* 保持旧预分频器设置防止无意暂停 */
  WDTCR |= (1<<WDCE) | (1<<WDE);
  /* 关闭 WDT */
  WDTCR = 0x00;
  __enable_interrupt();
}
```

Note: 1. 代码例程假设包括所需头文件。

注意：若看门狗由于错误指针或掉电状态等使看门狗出现意外使能，芯片将复位看门狗定时器将保持使能。如果编码没有设置处理看门狗，则可能导致溢出复位出现死循环。为避免出现这种状况，即使没有使用看门狗，应用程序在初始化时应对 WDRF 与 WDE 控制位清零。

下面的例子分别用汇编和 C 语言实现了看门狗定时器溢出值的改变。

汇编代码例程 (1)

```
WDT_Prescaler_Change:
  ; 关闭全局中断
  cli
  ; 复位看门狗定时器
  wdr
  ; 启动时序
  in    r16, WDTCR
  ori   r16, (1<<WDCE) | (1<<WDE)
  out   WDTCR, r16
  ; --  从此时在四个周期中设置新值 -
  ; 设置新预分频器值 = 64K 周期 (~0.5 s)
  ldi   r16, (1<<WDE) | (1<<WDP2) | (1<<WDP0)
  out   WDTCR, r16
  ; --  完成新值设置，使用 2 周期 -
  ; 开启全局中断
  sei
  ret
```

C 代码例程 (1)

```
void WDT_Prescaler_Change(void)
{
  __disable_interrupt();
  __watchdog_reset();
  /* 启动时序 */
  WDTCR |= (1<<WDCE) | (1<<WDE);
  /* 设置新预分频器值 = 64K 周期 (~0.5 s) */
  WDTCR = (1<<WDE) | (1<<WDP2) | (1<<WDP0);
  __enable_interrupt();
}
```

Note: 1. 代码例程假设包括所需头文件。

注意：看门狗定时器应在 WDP 位改变前复位，因为当改变 WDP 转换到一个短溢出周期时可能会导致暂停。

看门狗定时器控制寄存器 - WDTCSR

Bit	7	6	5	4	3	2	1	0	
	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	WDTCSR
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	X	0	0	0	

- **Bit 7 - WDIF: 看门狗定时器中断标志**

当看门狗定时器出现溢出且看门狗定时器配置为中断时，该位置位。当执行相应的中断程序时， WDTIF 由硬件清除；或者在该位写入逻辑 "1" 来清除。当 SREG 寄存器的 I 位与 WDTIE 置位，执行看门狗溢出中断。

- **Bit 6 - WDIE: 看门狗定时器中断使能**

当该位与 SREG 寄存器的 I 位置位，看门狗中断使能。如果同时也将 WDE 清除，看门狗定时器进入中断模式，一旦看门狗定时器程序暂停，则执行相应的中断。

若 WDE 置位，则看门狗定时器处于中断与系统复位模式。看门狗定时器的第一次溢出将设置 WDIF。执行相应的中断向量将会由硬件最大清除 WDIE 与 WDIF(看门狗进入系统复位模式)。这种方式会保证使用中断时看门狗定时器的安全性。在中断与系统复位模式下，WDIE 在每次中断后必须设置。然而它不能在中断服务子程序中执行，因为这可能会损害看门狗系统复位模式的安全性。如果在下一次溢出前没有程序中断，则进入系统复位模式。

Table 19. 看门狗定时器配置

WDTON	WDE	WDIE	模式	暂停活动
0	0	0	停止	无
0	0	1	中断模式	中断
0	1	0	系统复位模式	复位
0	1	1	中断与系统复位模式	中断，然后进入系统复位模式
1	x	x	系统复位模式	复位

- **Bit 4 - WDCE: 看门狗修改使能**

该位用在改变 WDE 与预分频位的时序中。WDCE 置位来清除 WDE 位，与 / 或改变预分频位。

一旦置 "1"，硬件将在四个时钟周期后对 WDCE 清零。

- **Bit 3 - WDE: 看门狗系统复位使能**

WDE 由 MCUSR 寄存器的 WDRF 决定。这就是说当 WDRF 设置时，WDE 也设置。要清除 WDE，必须先清除 WDRF。这一特性保证状态出错时的多重复位，及出错后的安全启动。

- **Bit 5, 2..0 - WDP3..0: 看门狗定时器预分频器 3, 2, 1 和 0**

WDP3..0 决定看门狗定时器的预分频器。如 P40Table 20 所示。

Table 20. 看门狗定时器预分频器选项

WDP3	WDP2	WDP1	WDP0	看门狗振荡器周期	V_{CC} = 5.0V 时典型的溢出周期
0	0	0	0	2K (2048) 周期	16 ms
0	0	0	1	4K (4096) 周期	32 ms
0	0	1	0	8K (8192) 周期	64 ms

Table 20. 看门狗定时器预分频器选项

WDP3	WDP2	WDP1	WDP0	看门狗振荡器周期	V_{CC} = 5.0V 时典型的溢出周期
0	0	1	1	16K (16384) 周期	0.125 s
0	1	0	0	32K (32768) 周期	0.25 s
0	1	0	1	64K (65536) 周期	0.5 s
0	1	1	0	128K (131072) 周期	1.0 s
0	1	1	1	256K (262144) 周期	2.0 s
1	0	0	0	512K (524288) 周期	4.0 s
1	0	0	1	1024K (1048576) 周期	8.0 s
1	0	1	0	保留	
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

中断

本节描述ATtiny2313的中断处理。更一般的AVR中断处理请参见P11“复位与中断处理”。

ATtiny2313 的中断向量

Table 21. 复位和中断向量

向量号	程序地址	中断源	中断定义
1	0x0000	RESET	外部引脚电平引发的复位，上电复位，掉电检测复位，看门狗复位
2	0x0001	INT0	外部中断请求 0
3	0x0002	INT1	外部中断请求 1
4	0x0003	TIMER1 CAPT	定时器 / 计数器 1 捕获
5	0x0004	TIMER1 COMPA	定时器 / 计数器 1 比较匹配 A
6	0x0005	TIMER1 OVF	定时器 / 计数器 1 溢出
7	0x0006	TIMER0 OVF	定时器 / 计数器 0 溢出
8	0x0007	USART0, RX	USART0 接收结束
9	0x0008	USART0, UDRE	USART0 数据寄存器空
10	0x0009	USART0, TX	USART0 发送结束
11	0x000A	ANALOG COMP	模拟比较器
12	0x000B	PCINT	引脚变化中断
13	0x000C	TIMER1 COMPB	定时器 / 计数器 1 比较匹配 B
14	0x000D	TIMER0 COMPA	定时器 / 计数器 0 比较匹配 A
15	0x000E	TIMER0 COMPB	定时器 / 计数器 0 比较匹配 B
16	0x000F	USI START	USI 启动状态
17	0x0010	USI OVERFLOW	USI 溢出
18	0x0011	EE READY	EEPROM 就绪
19	0x0012	WDT OVERFLOW	看门狗定时器溢出

ATtiny2313 复位与中断向量地址典型设置为：

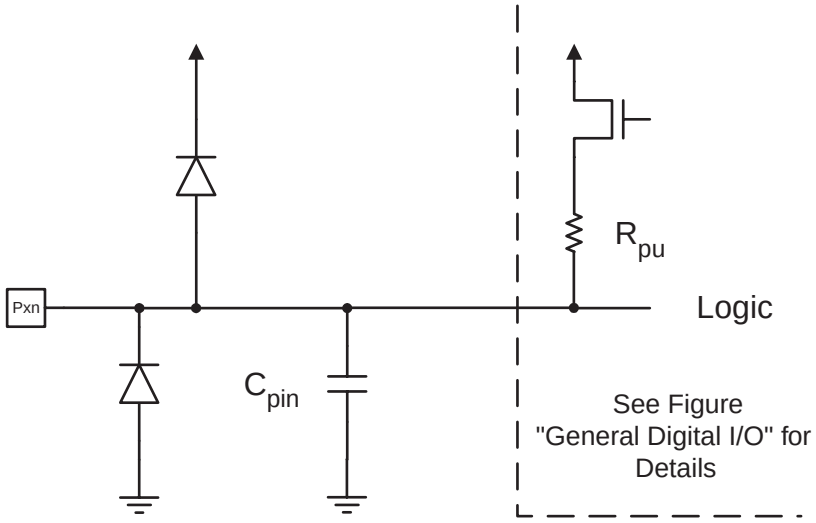
```
地址      符号    代码                    说明
0x0000            rjmp   RESET            ; 复位句柄
0x0001            rjmp   INT0             ; 外部中断 0 句柄
0x0002            rjmp   INT1             ; 外部中断 1 句柄
0x0003            rjmp   TIM1_CAPT        ; 定时器 1 捕获句柄
0x0004            rjmp   TIM1_COMPA       ; 定时器 1 比较 A 句柄
0x0005            rjmp   TIM1_OVF         ; 定时器 1 溢出句柄
0x0006            rjmp   TIM0_OVF         ; 定时器 0 溢出句柄
0x0007            rjmp   USART0_RXC       ; USART0 接收结束句柄
0x0008            rjmp   USART0_DRE       ; USART0,UDR 空句柄
0x0009            rjmp   USART0_TXC       ; USART0 发送结束句柄
0x000A            rjmp   ANA_COMP         ; 模拟比较器句柄
0x000B            rjmp   PCINT            ; 引脚变化中断
0x000C            rjmp   TIMER1_COMPB     ; 定时器 1 比较 B 句柄
0x000D            rjmp   TIMER0_COMPA     ; 定时器 0 比较 A 句柄
0x000E            rjmp   TIMER0_COMPB     ; 定时器 0 比较 B 句柄
0x000F            rjmp   USI_START        ; USI 启动句柄
0x0010            rjmp   USI_OVERFLOW     ; USI 溢出句柄
0x0011            rjmp   EE_READY         ; EEPROM 就绪句柄
0x0012            rjmp   WDT_OVERFLOW     ; 看门狗溢出句柄

;
0x0013  RESET: ldi    r16, low(RAMEND); 主程序
0x0014         out    SPL,r16           设置堆栈指针为 RAM 的顶部
0x0015         sei                      ; 使能中断
0x0016         <instr>  xxx
  ...     ...     ...  ...
```

I/O 端口

介绍

作为通用数字 I/O 使用时，所有 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时不会无意地改变其他管脚的方向。输出缓冲器具有对称的驱动能力，可以输出或吸收大电流，直接驱动 LED。所有的端口引脚都具有与电压无关的上拉电阻。并有保护二极管与 V_{CC} 和地相连，如 Figure 21 所示。请参见 P168“电气特性”以了解完整的参数列表。

Figure 21. I/O 引脚等效原理图



本节所有的寄存器和位以通用格式表示：小写的 “x” 表示端口的序号，而小写的 “n” 代表位的序号。但是在程序里要写完整。例如， PORTB3 表示端口 B 的第 3 位，而本节的通用格式为 PORTxn。物理 I/O 寄存器和位定义列于 P56“I/O 端口寄存器说明”。

每个端口都有三个 I/O 存储器地址：数据寄存器 – PORTx、数据方向寄存器 – DDRx 和端口输入引脚 – PINx。数据寄存器和数据方向寄存器为读 / 写寄存器，而端口输入引脚为只读寄存器。但是需要特别注意的是，对 PINx 寄存器某一位写入逻辑 "1“ 将造成数据寄存器相应位的数据发生 "0“ 与 “1“ 的交替变化。当寄存器 MCUCR 的上拉禁止位 PUD 置位时所有端口引脚的上拉电阻都被禁止。

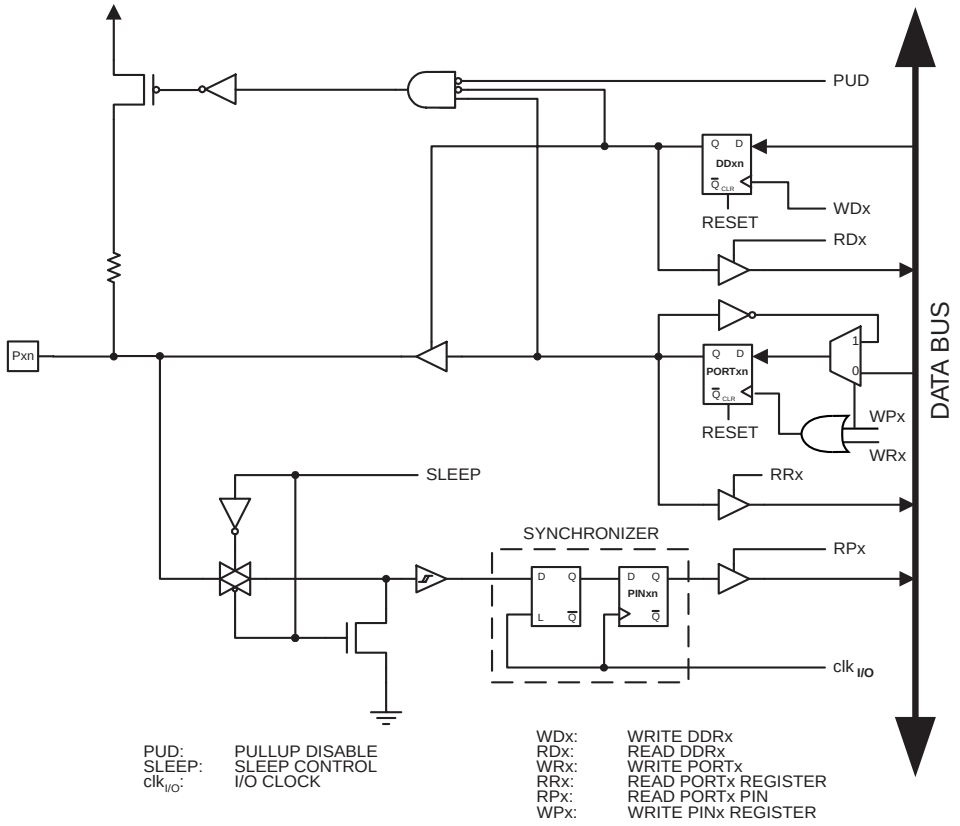
作为通用数字 I/O 时的端口请参见 P44“作为通用数字 I/O 的端口”。多数端口引脚是与第二功能复用的，如 P48“端口的第二功能”所示。请参见各个模块的具体说明以了解引脚的第二功能。

使能某些引脚的第二功能不会影响其他属于同一端口的引脚用于通用数字 I/O 目的。

作为通用数字 I/O 的端口

端口为具有可选上拉电阻的双向 I/O 端口。 Figure 22 为一个 I/O 端口引脚的说明。

Figure 22. 通用数字 I/O[1]



Note: 1. WRx, WPx, WDx, RRx, RPx 和 RDx 对于同一端口的所有引脚都是一样的。$clk_{I/O}$, SLEEP 和 PUD 则对所有的端口都是一样的。

配置引脚

每个端口引脚都具有三个寄存器位：DDxn、PORTxn 和 PINxn，如 P56“I/O 端口寄存器说明 ” 所示。DDxn 位于 DDRx 寄存器，PORTxn 位于 PORTx 寄存器，PINxn 位于 PINx 寄存器。

DDxn 用来选择引脚的方向。 DDxn 为 "1“ 时， Pxn 配置为输出，否则配置为输入。

引脚配置为输入时，若 PORTxn 为 "1“，上拉电阻将使能。如果需要关闭这个上拉电阻，可以将 PORTxn 清零，或者将这个引脚配置为输出。复位时各引脚为高阻态，即使此时并没有时钟在运行。

当引脚配置为输出时，若 PORTxn 为 "1“，引脚输出高电平 ("1“)，否则输出低电平 (“0“)。

转换引脚

在 PINxn 写逻辑 "1” 转换 PORTxn 值，与 DDRxn 值无关。注意，SBI 指令可用来端口的一位。

输入输出间转换

在 (高阻态) 三态 ({DDxn, PORTxn} = 0b00) 输出高电平 ({DDxn, PORTxn} = 0b11) 两种状态之间进行切换时，上拉电阻使能 ({DDxn, PORTxn} = 0b01) 或输出低电平 ({DDxn, PORTxn} = 0b10) 这两种模式必然会有一个发生。通常，上拉电阻使能是完全可以接受的，因为高阻环境不在意是强高电平输出还是上拉输出。如果使用情况不是这样子，可以通过置位 MCUCR 寄存器的 PUD 来禁止所有端口的上拉电阻。

在上拉输入和输出低电平之间切换也有同样的问题。用户必须选择高阻态 ({DDxn, PORTxn} = 0b00) 或输出高电平 ({DDxn, PORTxn} = 0b10) 作为中间步骤。

Table 22 总结了引脚的控制信号。

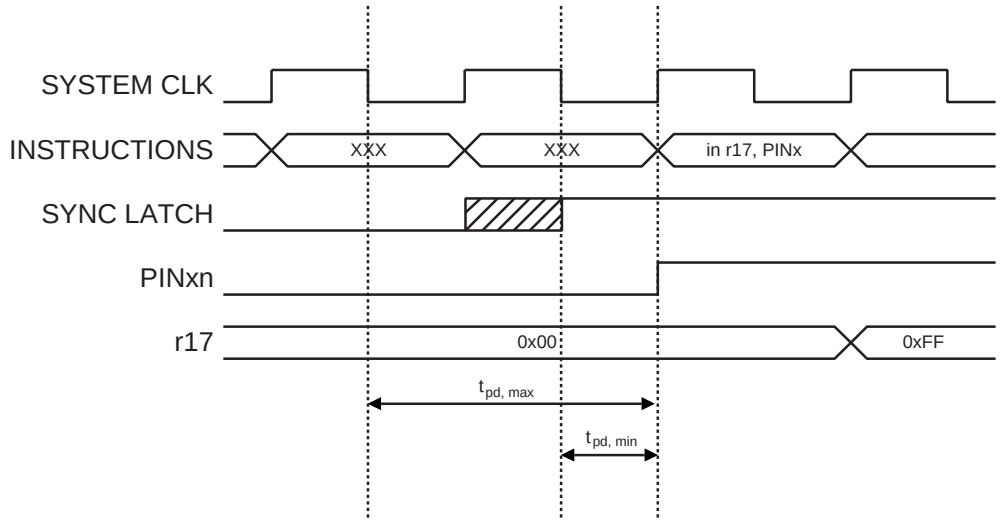
Table 22. 端口引脚配置

DDxn	PORTxn	PUD (MCUCR2 中)	I/O	上拉电阻	说明
0	0	X	输入	No	高阻态 (Hi-Z)
0	1	0	输入	Yes	被外部电路拉低时将输出电流
0	1	1	输入	No	高阻态 (Hi-Z)
1	0	X	输出	No	输出低电平 (吸收电流)
1	1	X	输出	No	输出高电平 (输出电流)

读取引脚上的数据

不论如何配置 DDxn ,都可以通过读取 PINxn 寄存器来获得引脚电平。如 Figure 22 所示，PINxn 寄存器的各个位与其前面的锁存器组成了一个同步器。这样就可以避免在内部时钟状态发生改变的短时间范围内由于引脚电平变化而造成的信号不稳定。其缺点是引入了延迟。Figure 23 为读取引脚电平时同步器的时序图。最大和最小传输延迟分别为 $t_{pd,max}$ 和 $t_{pd,min}$。

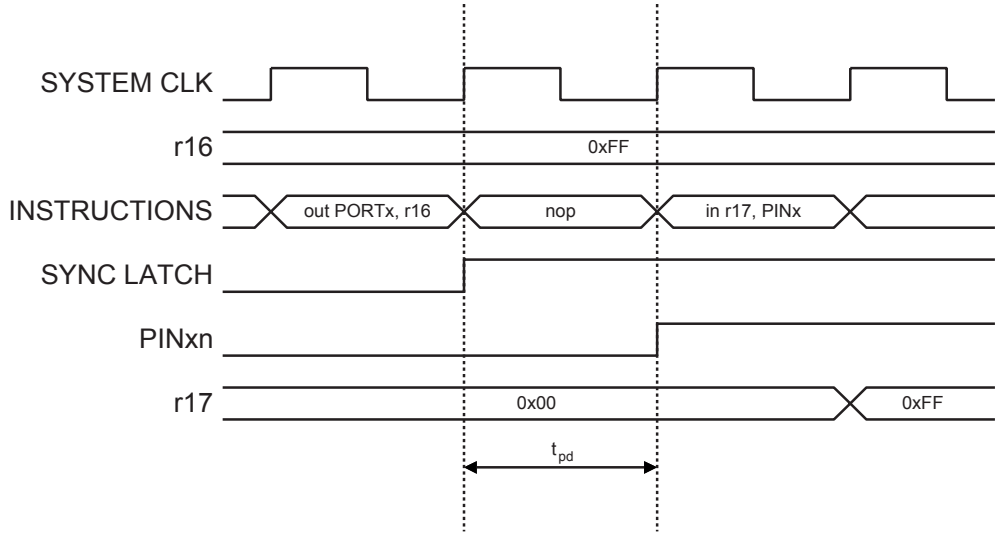
Figure 23. 读取引脚数据时的同步



下面考虑第一个系统时钟下降沿之后起始的时钟周期。当时钟信号为低时锁存器是关闭的；而时钟信号为高时信号可以自由通过，如图中 SYNC LATCH 信号的阴影区所示。时钟为低时信号即被锁存，然后在紧接着的系统时钟上升沿锁存到 PINxn 寄存器。如 $t_{pd,max}$ 和 $t_{pd,min}$ 所示，引脚上的信号转换延迟界于 ½ ~ 1½ 个系统时钟。

如 Figure 24 所示，读取软件赋予的引脚电平时需要在赋值指令 *out* 和读取指令 *in* 之间有一个时钟周期的间隔，如 *nop* 指令。*out* 指令在时钟的上升沿置位 SYNC LATCH 信号。此时同步器的延迟时间 t_{pd} 为一个系统时钟。

Figure 24. 读取软件赋予的引脚电平的同步



下面的例子演示了如何置位端口 B 的引脚 0 和 1，清零引脚 2 和 3，以及将引脚 4 到 7 设置为输入，并且为引脚 6 与 7 设置上拉电阻。然后将各个引脚的数据读回来。如前面讨论的那样，我们在输出和输入语句之间插入了一个 *nop* 指令。

汇编代码例程 (1)

```
...
; 定义上拉电阻和设置高电平输出
; 为端口引脚定义方向
ldi  r16,(1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0)
ldi  r17,(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
out  PORTB,r16
out  DDRB,r17
; 为了同步插入 nop 指令
nop
; 读取端口引脚
in   r16,PINB
...
```

C 代码例程

```
unsigned char i;
...
/* 定义上拉电阻和设置高电平输出 */
/* 为端口引脚定义方向 */
PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
/* 为了同步插入 nop 指令 */
__no_operation();
/* 读取端口引脚 */
i = PINB;
...
```

Note: 1. 在汇编程序里使用了两个暂存器。其目的是为了使整个操作过程的时间最短。通过拉高引脚 0、1、6、7，直到方向位设置正确，定义位 2、3 为低，且重新定义为 0 与 1 为强驱动。

数字输入使能和休眠模式

如 Figure 22 所示，数字输入信号 (施密特触发器的输入) 可以钳位到地。图中的 SLEEP 信号由 MCU 休眠控制器在各种掉电模式、省电模式以及 Standby 模式下设置，以防止在输入悬空或模拟输入电平接近 $V_{CC}/2$ 时消耗太多的电流。

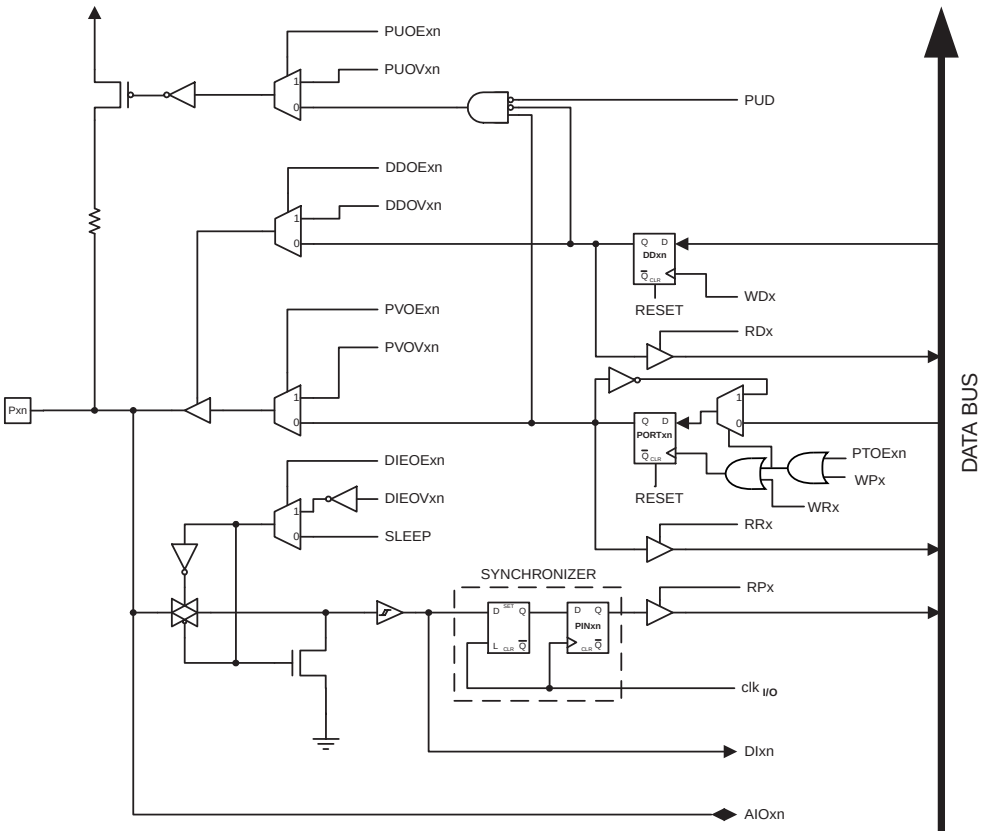
引脚作为外部中断输入时 SLEEP 信号无效。但若外部中断没有使能， SLEEP 信号仍然有效。引脚的第二功能使能时 SLEEP 也让位于第二功能，如 P48“ 端口的第二功能 ” 里描述的那样。

如果逻辑高电平 (“1”) 出现在一个被设置为 " 上升沿、下降沿或任何逻辑电平变化都引起中断 " 的外部异步中断引脚上，即使该外部中断未被使能，但从上述休眠模式唤醒时，相应的外部中断标志位仍会被置 "1”。这是因为引脚电平在休眠模式下被钳位到 "0” 电平。唤醒过程造成了引脚电平从 "0” 到 ”1” 的变化。

端口的第二功能

除了通用数字 I/O 功能之外，大多数端口引脚都具有第二功能。Figure 25 说明了由 Figure 22 简化得出的端口引脚控制信号是如何被第二功能取代的。这些被重载的信号不会出现在所有的端口引脚，但本图可以看作是适合于AVR系列处理器所有端口引脚的一般说明。

Figure 25. 端口的第二功能 (1)



PUOExn:	Pxn PULL-UP OVERRIDE ENABLE	PUD:	PULLUP DISABLE
PUOVxn:	Pxn PULL-UP OVERRIDE VALUE	WDx:	WRITE DDRx
DDOExn:	Pxn DATA DIRECTION OVERRIDE ENABLE	RDx:	READ DDRx
DDOVxn:	Pxn DATA DIRECTION OVERRIDE VALUE	RRx:	READ PORTx REGISTER
PVOExn:	Pxn PORT VALUE OVERRIDE ENABLE	WRx:	WRITE PORTx
PVOVxn:	Pxn PORT VALUE OVERRIDE VALUE	RPx:	READ PORTx PIN
DIEOExn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE ENABLE	WPx:	WRITE PINx
DIEOVxn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE VALUE	$clk_{I/O}$:	I/O CLOCK
SLEEP:	SLEEP CONTROL	DIxn:	DIGITAL INPUT PIN n ON PORTx
PTOExn:	Pxn, PORT TOGGLE OVERRIDE ENABLE	AIOxn:	ANALOG INPUT/OUTPUT PIN n ON PORTx

Note: 1. WPx, WDx, RLx, RPx和RDx对于同一个端口的所有引脚都是一样的。$clk_{I/O}$, SLEEP和PUD 则对所有的端口都是一样的。其他信号只对某一个引脚有效。

Table 23 为重载信号的简介。表中没有给出 Figure 25 的引脚和端口索引。这些重载信号是由第二功能模块产生的。

Table 23. 第二功能重载信号的一般说明

信号名称	全称	说明
PUOE	上拉电阻 重载使能	若此信号置位，上拉电阻使能将受控于 PUOV ；若此信号清零，则 {DDxn, PORTxn, PUD} = 0b010 时上拉电阻使能。
PUOV	上拉电阻 重载使能	若 PUOE 置位，则 PUOV 置位 / 清零时上拉电阻使能 / 禁止，而不管 DDxn、PORTxn 和 PUD 寄存器各个位的设置如何。
DDOE	数据方向 重载使能	如果此信号置位，则输出驱动使能由 DDOV 控制；若此信号清零，输出驱动使能由 DDxn 寄存器控制。
DDOV	数据方向 重载使能	若 DDOE 置位，则 DDOV 置位 / 清零时输出驱动使能 / 禁止，而不管 DDxn 寄存器的设置如何。
PVOE	端口数据 重载使能	如果这个信号置位，且输出驱动使能，端口数据由 PVOV 控制；若 PVOE 清零，且输出驱动使能，端口数据由寄存器 PORTxn 控制。
PVOV	端口数据 重载使能	若 PVOE 置位，端口值设置为 PVOV，而不管寄存器 PORTxn 如何设置。
PTOE	端口信号切换 重载使能	若 PTOE 置位，端口寄存器位取反。
DIEOE	数字输入 使能覆盖使能	如果这个信号置位，数字输入使能由 DIEOV 控制；若 DIEOE 清零，数字输入使能由 MCU 的状态确定 (正常模式，睡眠模式)。
DIEOV	数字输入 使能覆盖使能	若 DIEOE 置位， DIEOV 置位 / 清零时数字输入使能 / 禁止，而不管 MCU 的状态如何 (正常模式，睡眠模式)。
DI	数字输入	此信号为第二功能的数字输入。在图中，这个信号与施密特触发相连，并且在同步器之前。除非数字输入用作时钟源，否则第二功能模块将使用自己的同步器。
AIO	模拟信号 输入 / 输出	模拟输入 / 输出。信号直接与引脚接点相连，而且可以用作双向端口。

下面的几小节将简单地说明每个端口的第二功能以及相关的信号。具体请参考有关第二功能的说明。

MCU 控制寄存器 - MCUCR

Bit	7	6	5	4	3	2	1	0	
	PUD	SM1	SE	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
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	

- **Bit 7 – PUD: 上拉禁用**

当该位为 1，即使 DDxn 与 PORTxn 寄存器配置为使能上拉电阻 ({DDxn, PORTxn} = 0b01)，I/O 端口上拉电阻禁用，详见 P45“ 配置引脚 ” 。

端口 A 的第二功能

端口 A 的第二功能列于 Table 5。

Table 24. 端口 A 的第二功能

端口引脚	第二功能
PA2	RESET, dW
PA1	XTAL2
PA0	XTAL1

端口 B 的第二功能

端口 B 的第二功能列于 Table 25。

Table 25. 端口 B 的第二功能

端口引脚	第二功能
PB7	USCK/SCL/PCINT7
PB6	DO/PCINT6
PB5	DI/SDA/PCINT5
PB4	OC1B/PCINT4
PB3	OC1A/PCINT3
PB2	OC0A/PCINT2
PB1	AIN1/PCINT1
PB0	AIN0/PCINT0

第二功能引脚配置如下：

- **USCK/SCL/PCINT7 - 端口 B, Bit 7**

USCK：三线模式通用串行接口时钟。

SCL：USI 两线模式串行时钟。

PCINT7：引脚变换中断源 7。 PB7 引脚作为外部中断源。

- **DO/PCINT6 - 端口 B, Bit 6**

DO : 三线模式通用串行接口数据输出。三线模式数据输出覆盖 PORTB6 值且当数据方向位 DDB6 置 1 时驱动端口。但若方向为输入且 PORTB6 为 1，PORTB6 位仍控制上拉电阻使能。

PCINT6：引脚变换中断源 6。 PB6 引脚作为外部中断源。

- **DI/SDA/PCINT5 - 端口 B, Bit 5**

DI : 三线模式通用串行接口数据输入。三线模式没有覆盖普通端口功能，因此引脚必须配置为输入。

SDA：两线模式串行接口数据。

PCINT5：引脚变换中断源 5。 PB5 引脚作为外部中断源。

- **OC1B/PCINT4 – 端口 B, Bit 4**

OC1B:输出比较匹配B输出:PB4引脚可以做为T/C1输出比较B的外部输出。引脚配置为输出 (DDB6 置 1) 。 OC1B 引脚也做为 PWM 模式定时器功能的输出引脚。

PCINT4：引脚变换中断源 4。 PB4 引脚作为外部中断源。

- **OC1A/PCINT3 – 端口 B, Bit 3**

OC1A:输出比较匹配A输出:PB3引脚可以做为T/C1输出比较A的外部输出。引脚配置为输出 (DDB3 置 1) 。 OC1A 引脚也做为 PWM 模式定时器功能的输出引脚。

PCINT3：引脚变换中断源 3。 PB3 引脚作为外部中断源。

- **OC0A/PCINT2 – 端口 B, Bit 2**

OC0A:输出比较匹配A输出:PB2引脚可以做为T/C0输出比较A的外部输出。引脚配置为输出 (DDB2 置 1) 。 OC0A 引脚也做为 PWM 模式定时器功能的输出引脚。

PCINT2：引脚变换中断源 2。 PB2 引脚作为外部中断源。

- **AIN1/PCINT1 – 端口 B, Bit 1**

AIN1：模拟比较器负极输入。配置端口引脚作为输入时要将内部上拉电阻切断，以避免数字端口功能妨碍模拟比较器的功能。

PCINT1：引脚变换中断源 1。 PB1 引脚作为外部中断源。

- **AIN0/PCINT0 – 端口 B, Bit 0**

AIN0：模拟比较器正极输入。配置端口引脚作为输入时要将内部上拉电阻切断，以避免数字端口功能妨碍模拟比较器的功能。

PCINT0：引脚变换中断源 0。 PB0 引脚作为外部中断源。

Table 26与Table 27端口B的第二功能的重载信号见 P49Figure 25。SPI MSTR INPUT与SPI SLAVE OUTPUT构成MISO信号，而MOSI分为SPI MSTR OUTPUT与SPI SLAVE INPUT。

Table 26. PB7..PB4 重载信号

信号名称	PB7/USCK/SCL/PCINT7	PB6/DO/PCINT6	PB5/SDA/DI/PCINT5	PB4/OC1B/PCINT4
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	USI_TWO_WIRE	0	USI_TWO_WIRE	0
DDOV	(USI_SCL_HOLD + $\overline{\text{PORTB7}}$)•DDB7	0	($\overline{\text{SDA}}$ + $\overline{\text{PORTB5}}$)• DDRB5	0
PVOE	USI_TWO_WIRE • DDRB7	USI_THREE_WIRE	USI_TWO_WIRE • DDRB5	OC1B_PVOE
PVOV	0	DO	0	0OC1B_PVOV
PTOE	USI_PTOE	0	0	0
DIEOE	(PCINT7•PCIE) +USISIE	(PCINT6•PCIE)	(PCINT5•PCIE) + USISIE	(PCINT4•PCIE)
DIEOV	1	1	1	1
DI	PCINT7 输入 USCK 输入 SCL 输入	PCINT6 输入	PCINT5 输入 SDA 输入 DI 输入	PCINT4 输入
AIO	–	–	–	–

Table 27. PB3..PB0 重载信号

信号名称	PB3/OC1A/PCINT3	PB2/OC0A/PCINT2	PB1/AIN1/PCINT1	PB0/AIN0/PCINT0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC1A_PVOE	OC0A_PVOE	0	0
PVOV	OC1A_PVOV	OC0A_PVOV	0	0
PTOE	0	0	0	0
DIEOE	(PCINT3 • PCIE)	(PCINT2 • PCIE)	(PCINT1 • PCIE)	(PCINT0 • PCIE)
DIEOV	1	1	1	1
DI	PCINT7 输入	PCINT6 输入	PCINT5 输入	PCINT4 输入
AIO	–	–	AIN1	AIN0

端口 D 的第二功能

端口 D 的第二功能列于 Table 28。

Table 28. 端口 D 的第二功能

端口引脚	第二功能
PD6	ICP
PD5	OC0B/T1
PD4	T0
PD3	INT1
PD2	INT0/XCK/CKOUT
PD1	TXD
PD0	RXD

第二功能引脚配置如下：

- **ICP – 端口 D, Bit 6**

ICP：T/C1 输入捕获引脚。 PD6 引脚可做为 T/C1 输入捕获引脚。

- **OC1B/T1 – 端口 D, Bit 5**

OC0B: : 输出比较匹配 B 输出: PD5 引脚可以做为 T/C0 输出比较 B 的外部输出。引脚配置为输出 (DDB5 置 1) 。 OC0B 引脚也做为 PWM 模式定时器功能的输出引脚。

T1: 通过置位 T/C1 控制寄存器 TCCR1 的 CS02 与 CS01 位 , T/C1 外部计数器时钟输入。

- **T0 – 端口 D, Bit 4**

T0: 通过置位 T/C0 控制寄存器 TCCR0 的 CS02 与 CS01 位 , T/C0 外部计数器时钟输入。

- **INT1 – 端口 D, Bit 3**

INT0：外部中断源 0。 PD3 引脚作为 MCU 的外部中断源。

- **INT0/XCK/CKOUT – 端口 D, Bit 2**

INT1：外部中断源 1。 PD2 引脚作为 MCU 的外部中断源。

XCK：同步传送模式下使用 USART 传送时钟。

CKOUT：系统时钟输出。

- **TXD – 端口 D, Bit 1**

TXD：UART 数据传送器。

- **RXD – 端口 D, Bit 0**

RXD：UART 数据计数器。

Table 29 与 Table 30 端口 D 的第二功能的重载信号见 P49Figure 25。

Table 29. PD7..PD4 重载信号

信号名称	PD6/ICP	PD5/OC1B/T1	PD4/T0
PUOE	0	0	0
PUOV	0	0	0
DDOE	0	0	0
DDOV	0	0	0
PVOE	0	OC1B_PVOE	0
PVOV	0	OC1B_PVOV	0
PTOE	0	0	0
DIEOE	ICP 使能	T1 使能	T0 使能
DIEOV	1	1	1
DI	ICP 输入	T1 输入	T0 输入
AIO	–	–	AIN1

Table 30. PD3..PD0 重载信号

信号名称	PD3/INT1	PD2/INT0/XCK/ CKOUT	PD1/TXD	PD0/RXD
PUOE	0	0	TXD_OE	RXD_OE
PUOV	0	0	0	PORTD0 • $\overline{PUD}$
DDOE	0	0	TXD_OE	RXD_EN
DDOV	0	0	1	0
PVOE	0	XCKO_PVOE	TXD_OE	0
PVOV	0	XCKO_PVOV	TXD_PVOV	0
PTOE	0	0	0	0
DIEOE	INT1 使能	INT0 使能 / XCK 输入使能	0	0
DIEOV	1	1	0	0
DI	INT1 输入	INT0 输入 / XCK 输入	–	RXD 输入
AIO	–	–	–	–

I/O 端口寄存器说明

端口 A 数据寄存器 - PORTA

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	PORTA2	PORTA1	PORTA0	PORTA
读 / 写	R	R	R	R	R	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

端口 A 数据方向寄存器 - DDRA

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	DDA2	DDA1	DDA0	DDRA
读 / 写	R	R	R	R	R	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

端口 A 输入引脚地址 - PINA

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	PINA2	PINA1	PINA0	PINA
读 / 写	R	R	R	R	R	R/W	R/W	R/W	
初始值	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

端口 B 数据寄存器 - PORTB

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
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	

端口 B 数据方向寄存器 - DDRB

Bit	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
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	

端口 B 输入引脚地址 - PINB

Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
读 / 写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

端口 D 数据寄存器 - PORTD

Bit	7	6	5	4	3	2	1	0	
	–	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
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	

端口 D 数据方向寄存器 - DDRD

Bit	7	6	5	4	3	2	1	0	
	–	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
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	

端口 D 输入引脚地址 - PIND

Bit	7	6	5	4	3	2	1	0	
	–	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
读 / 写	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

外部中断

外部中断通过引脚 INT0、 INT1 或 PCINT7..0 触发。只要使能了中断，即使引脚 INT0、INT1 或 PCINT7..0 配置为输出，只要电平发生了合适的变化，中断也会触发。这个特点可以用来产生软件中断。使能 PCINT7..0 引脚转换将触发引脚变化 PCI0。哪个引脚作为引脚变化中断由 PCMSK1 与 PCMSK0 寄存器控制。 PCINT7..0 的引脚变化中断是异步检测的。也就是说，这些中断可以用来将器件从睡眠模式唤醒。

通过设置外部中断控制寄存器 A – EICRA，中断可以由下降沿、上升沿，或者是低电平触发。当外部中断使能并且配置为电平触发，只要引脚电平为低，中断就会产生。若要求 INT0 或 INT1 在信号下降沿或上升沿触发，I/O 时钟必须工作，如 P21“时钟系统及其分布”说明的那样。 INT0 与 INT1 的中断条件检测是异步的。也就是说，这些中断可以用来将器件从睡眠模式唤醒。在睡眠过程 (除了空闲模式) 中 I/O 时钟是停止的。

注意，通过电平方式触发中断，从而将 MCU 从掉电模式唤醒时，要保证电平保持一定的时间。若电平在启动完成前结束， MCU 被唤醒，但不会产生中断。启动时间由 P21“ 时钟系统及其分布 ” 所示的 SUT 与 CKSEL 熔丝位定义。

MCU 控制寄存器 - MCUCR

外部中断控制寄存器 A 包含中断敏感控制的控制位。

Bit	7	6	5	4	3	2	1	0	
	PUD	SM1	SE	SMD	ISC11	ISC10	ISC01	ISC00	MCUCR
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	

- **Bit 3, 2 – ISC11, ISC10: 中断敏感方式控制 1 Bit 1 与 Bit 0**

如果 SREG 寄存器的 I 标志位和相应的中断屏蔽位置位的话，外部中断 1 由引脚 INT1 激发。触发方式如 Table 32 所示。在检测边沿前 MCU 首先采样 INT1 引脚上的电平。如果选择了边沿触发方式或电平变化触发方式，那么持续时间大于一个时钟周期的脉冲将触发中断，过短的脉冲则不能保证触发中断。如果选择低电平触发方式，那么低电平必须保持到当前指令执行完成。

Table 31. 中断 1 敏感控制

ISC11	ISC10	说明
0	0	INT1 为低电平时产生中断请求
0	1	INT1 引脚上任意的逻辑电平变化都将引发中断
1	0	INT1 的下降沿产生异步中断请求
1	1	INT1 的上升沿产生异步中断请求

- **Bit 1, 0 – ISC01, ISC00: 中断敏感方式控制 0 Bit1 与 Bit 0**

如果 SREG 寄存器的 I 标志位和相应的中断屏蔽位置位的话，外部中断 0 由引脚 INT0 激发。触发方式如 Table 32 所示。在检测边沿前 MCU 首先采样 INT0 引脚上的电平。如果选择了边沿触发方式或电平变化触发方式，那么持续时间大于一个时钟周期的脉冲将触发中断，过短的脉冲则不能保证触发中断。如果选择低电平触发方式，那么低电平必须保持到当前指令执行完成。

Table 32. 中断 0 敏感控制

ISC01	ISC00	说明
0	0	INT0 为低电平时产生中断请求
0	1	INT0 引脚上任意的逻辑电平变化都将引发中断
1	0	INT0 的下降沿产生异步中断请求
1	1	INT0 的上升沿产生异步中断请求

通用中断屏蔽寄存器 - GIMSK

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	PCIE	–	–	–	–	–	GIMSK
读 / 写	R/W	R/W	R/W	R	R	R	R	R	
初始值	0	0	0	0	0	0	0	0	

- **Bit 7 – INT1: 外部中断请求 1 使能**

当 INT1 为 '1'，而且状态寄存器 SREG 的 I 标志置位，相应的外部引脚中断就使能了。中断敏感电平控制 1 位 1/0 (ISC11 与 ISC10) 决定中断是由上升沿、下降沿，还是 INT1 电平触发的。只要使能，即使 INT1 引脚被配置为输出，只要引脚电平发生了相应的变化，中断可将产生。

- **Bit 6 – INT0: 外部中断请求 0 使能**

当 INT0 为 '1'，而且状态寄存器 SREG 的 I 标志置位，相应的外部引脚中断就使能了。中断敏感电平控制 0 位 1/0 (ISC01 与 ISC00) 决定中断是由上升沿、下降沿，还是 INT0 电平触发的。只要使能，即使 INT0 引脚被配置为输出，只要引脚电平发生了相应的变化，中断可将产生。

- **Bit 5 – PCIE: 引脚变化中断使能**

当 PCIE 位为 '1'，而且状态寄存器 SREG 的 I 标志置位，引脚变化中断使能。使能的 PCINT7..0 引脚上电平变化将造成中断。执行相应的 PCI 中断程序。PCINT7..0 各引脚由 PCMSK 寄存器单独使能。

外部中断标志寄存器 - EIFR

Bit	7	6	5	4	3	2	1	0	
	INTF1	INTF0	PCIF	–	–	–	–	–	EIFR
读 / 写	R/W	R/W	R/W	R	R	R	R	R	
初始值	0	0	0	0	0	0	0	0	

- **Bit 7 – INTF1: 外部中断标志 1**

INT1 引脚电平发生跳变时触发中断请求，并置位相应的中断标志 INTF1。如果 SREG 的位 I 以及 GIMSK 寄存器相应的中断使能位 INT1 为 ”1”，MCU 即跳转到相应的中断向量。进入中断服务程序之后该标志自动清零。此外，标志位也可以通过写入 ”1” 来清零。当 INT1 配置为电平中断时，该标志始终清零。

- **Bit 6 – INTF0: 外部中断标志 0**

INT0 引脚电平发生跳变时触发中断请求，并置位相应的中断标志 INTF0。如果 SREG 的位 I 以及 GIMSK 寄存器相应的中断使能位 INT0 为 ”1”，MCU 即跳转到相应的中断向量。进入中断服务程序之后该标志自动清零。此外，标志位也可以通过写入 ”1” 来清零。当 INT0 配置为电平中断时，该标志始终清零。

- **Bit 5 – PCIF: 引脚变化中断标志**

当 PCINT7..0 任意引脚电平变化触发中断请求，PCIF 置 "1”。若 SREG 的位 I 与 GIMSK 寄存器的 PCIE 位为 "1”，MCU 即跳转到相应的中断向量。进入中断服务程序之后该标志自动清零。此外，标志位也可以通过写入 ”1” 来清零。

引脚变化屏蔽寄存器 - PCMSK

Bit	7	6	5	4	3	2	1	0	
	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK
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	

- **Bit 7..0 – PCINT7..0: 引脚变化使能屏蔽 15..8**

每个 PCINT7..0 位选择是否使能相应的 I/O 引脚变化中断。若 PCINT7..0 与 PCIE 位设置，使能相应的引脚变化中断。若 PCINT7..0 清除，则禁用相应的引脚变化中断。

具有 PWM 功能的 8 位定时器 / 计时器 0

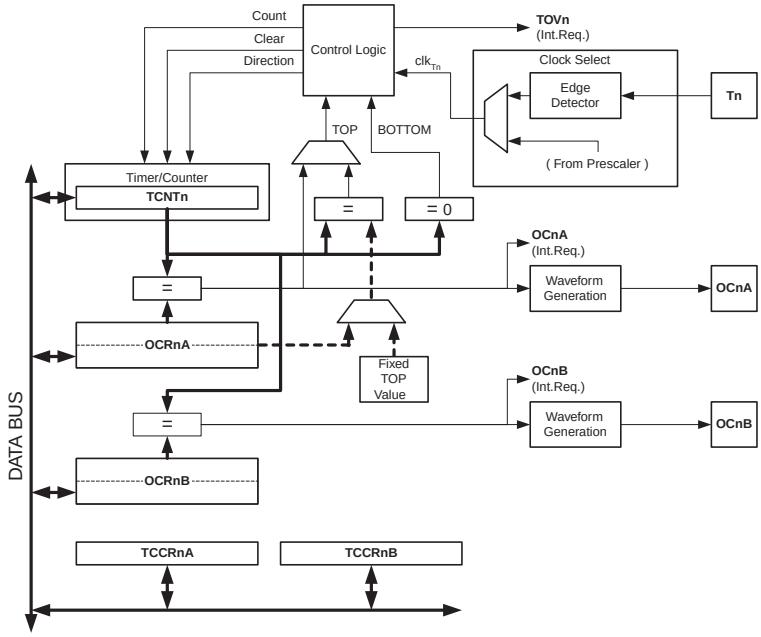
T/C0 是通用 8 位定时器 / 计数器模块 , 有两个独立的输出比较单元 , 且支持 PWM 功能。它提供精确的执行时序与波形产生。其主要特点如下：

- **两个独立输出比较单元**
- **双缓冲输出比较寄存器**
- **比较匹配发生时清除定时器 (自动加载)**
- **无干扰脉冲，相位正确的 PWM**
- **可变 PWM 周期**
- **频率发生器**
- **三个独立中断源 (TOV0, OCF0A 及 OCF0B)**

综述

Figure 26为8位定时器/计数器的简化框图。实际引脚排列请参考 P2“ATtiny2313引脚配置 ”。CPU 可以访问的 I/O 寄存器 , 包括位和引脚 , 以粗体显示。I/O 寄存器和位的位置列于 P69“8 位 定时器 / 计数器寄存器的说明 ”。

Figure 26. 8 位 T/C 方框图



寄存器

T/C(TCNT0) 和输出比较寄存器 (OCR0A 与 OCR0B) 为 8 位寄存器。中断请求 (图中简写为 Int.Req.) 信号在定时器中断标志寄存器 TIFR0 都有反映。所有中断都可以通过定时器中断屏蔽寄存器 TIMSK 单独进行屏蔽。图中没有给出 TIFR 和 TIMSK。

T/C 可以通过预分频器由内部时钟源驱动 , 或者是通过 T0 引脚的外部时钟源来驱动。时钟选择逻辑模块控制使用哪一个时钟源与什么边沿来增加 (或降低)T/C 的数值。如果没有选择时钟源 T/C 就不工作。时钟选择模块的输出定义为定时器时钟 clk_{T0}。

双缓冲的输出比较寄存器 (OCR0A 与 OCR0B) 一直与 T/C 的数值进行比较。比较的结果可用来产生 PWM 波，或在输出比较引脚 OC0 上产生变化频率的输出，如 P62 “ 输出比较单元 ” 说明的那样。比较匹配事件还将置位比较标志 (OCF0A 或 OCF0B)。此标志可以用来产生输出比较中断请求。

定义

本文的许多寄存器及其各个位以通用的格式表示。小写的 “n” 取代了 T/C 的序号，在此即为 0。小写的 “x” 取代了输出比较单元通道，在此即为通道 A 或 B。但是在写程序时要使用精确的格式，例如使用 TCNT0 来访问 T/C0 计数器值，等等。

Table 33 的定义适用于全文。

Table 33. 定义

BOTTOM	计数器计到 0x00 时即达到 BOTTOM。
MAX	计数器计到 0xFF (十进制的 255) 时即达到 MAX。
TOP	计数器计到计数序列的最大值时即达到 TOP。TOP 值可以为固定值 0xFF (MAX)，或是存储于寄存器 OCR0A 里的数值，具体由工作模式确定

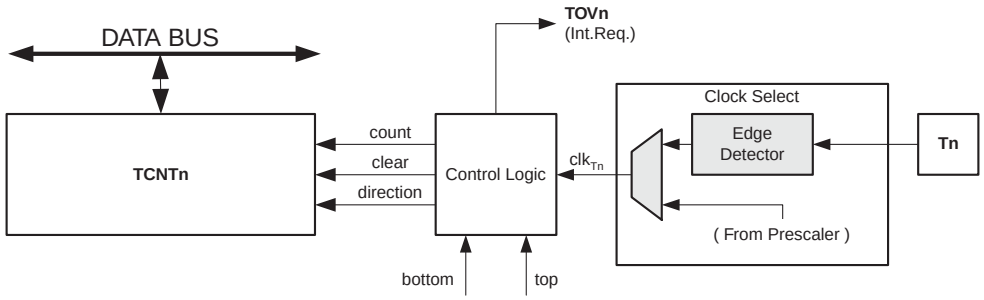
T/C 的时钟源

T/C 可以由内部同步时钟或外部时钟驱动。时钟源是由时钟选择逻辑决定的，而时钟选择逻辑是由位于 T/C 控制寄存器 TCCR0B 的时钟选择位 CS02:0 控制的。P75“T/C0 与 T/C1 预分频器 ” 对时钟源与预分频有详尽的描述。

计数器单元

8 位 T/C 的主要部分为可编程的双向计数单元。Figure 27 即为计数器和周边电路的框图。

Figure 27. 计数器单元方框图



信号说明 (内部信号) :

count 使 TCNT0 加 1 或减 1。

direction 选择加操作或减操作。

clear 清除 TCNT0 (将所有的位清零)。

clk_{Tn} T/C 的时钟，clk_{T0} 。

top 表示 TCNT0 已经达到了最大值。

bottom 表示 TCNT0 已经达到了最小值 (0)。

根据不同的工作模式，计数器针对每一个 clk_{T0} 实现清零、加一或减一操作。clk_{T0} 可以由内部时钟源或外部时钟源产生，具体由时钟选择位 CS02:0 确定。没有选择时钟源时 (CS02:0 = 0) 定时器即停止。但是不管有没有 clk_{T0}，CPU 都可以访问 TCNT0。CPU 写操作比计数器其他操作 (如清零、加减操作) 的优先级高。

计数序列由 T/C 控制寄存器 (TCCR0A) 的 WGM01 和 WGM00 及 (TCCR0B) 的 WGM02 位决定。计数器计数行为与输出比较 OC0A 的波形有紧密的关系。有关计数序列和波形产生的详细信息请参考 P87“ 工作模式 ” 。

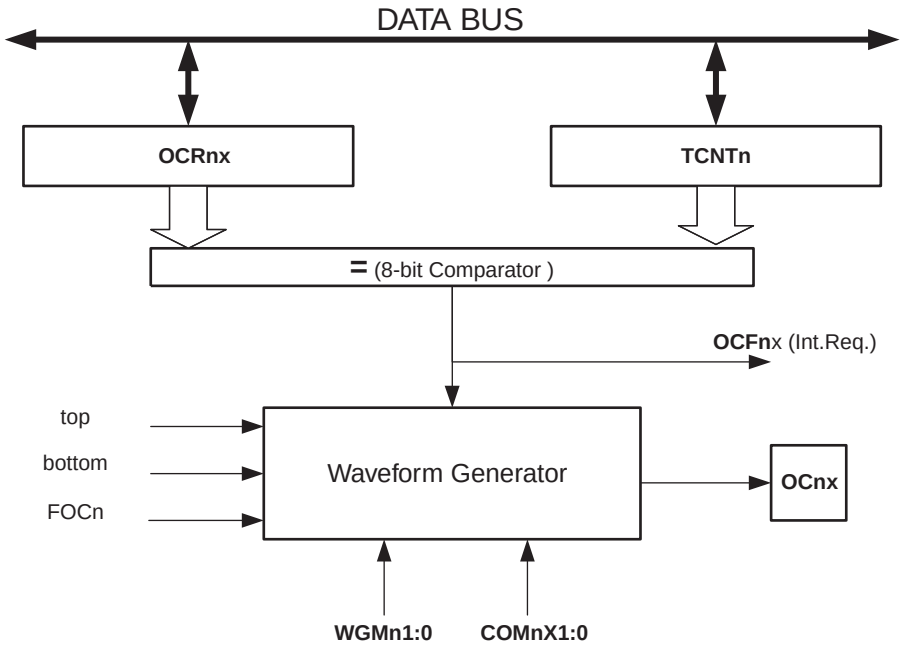
T/C溢出中断标志TOV0根据WGM01:0 设定的工作模式来设置。TOV0可以用于产生CPU中断。

输出比较单元

8位比较器持续对TCNT0和输出比较寄存器OCR0A(与OCR0B)进行比较。一旦TCNT0等于 OCR0A 或 OCR0B，比较器就给出匹配信号。在匹配发生的下一个定时器时钟周期输出比较标志 OCF0A(或 OCF0B) 置位。若此时相应中断使能， CPU 将产生输出比较中断。执行中断服务程序时 OCF0A(或 OCF0B) 自动清零，或者通过软件写 ”1” 的方式来清零。根据由 WGM2:0 和 COM0x1:0 设定的不同的工作模式，波形发生器利用匹配信号产生不同的波形。同时，波形发生器还利用 max 和 bottom 信号来处理极值条件下的特殊情况 (P87“ 工作模式 ”)。

Figure 28 为输出比较单元的方框图。

Figure 28. 输出比较单元方框图



使用 PWM 模式时 OCR0x 寄存器为双缓冲寄存器；而在正常工作模式和匹配时清零模式双缓冲功能是禁止的。双缓冲可以将更新 OCR0x 寄存器与 top 或 bottom 时刻同步起来，从而防止产生不对称的 PWM 脉冲，消除了干扰脉冲。

访问 OCR0x 寄存器看起来很复杂，其实不然。使能双缓冲功能时，CPU 访问的是 OCR0x 缓冲寄存器；禁止双缓冲功能时 CPU 访问的则是 OCR0x 本身。

强制输出比较

工作于非 PWM 模式时，可以通过对强制输出比较位 FOC0x 写 ”1” 的方式来产生比较匹配。强制比较匹配不会置位 OCF0x 标志，也不会重载 / 清零定时器，但是 OC0x 引脚将被更新，好象真的发生了比较匹配一样 (COM01:0 决定 OC0x 是置位、清零，还是 "0“-”1” 交替变化)。

写 TCNT0 操作将阻止比较匹配

CPU 对 TCNT0 寄存器的写操作会在下一个定时器时钟周期阻止比较匹配的发生，即使此时定时器已经停止了。这个特性可以用来将 OCR0x 初始化为与 TCNT0 相同的数值而不触发中断。

使用输出比较单元

由于在任意模式下写 TCNT0 都将在下一个定时器时钟周期里阻止比较匹配，在使用输出比较时改变 TCNT0 就会有风险，不论 T/C 此时是否在运行与否。如果写入的 TCNT0 的数值等于 OCR0x，比较匹配就被丢失了，造成不正确的波形发生结果。类似地，在计数器进行降序计数时不要对 TCNT0 写入等于 BOTTOM 的数据。

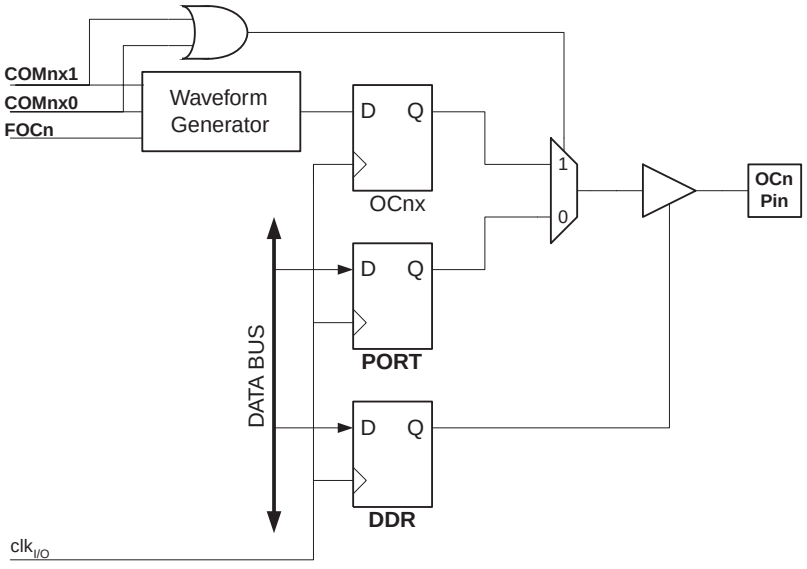
OC0x 的设置应该在设置数据方向寄存器之前完成。最简单的设置 OC0x 的方法是在普通模式下利用强制输出比较 FOC0x。即使在改变波形发生模式时 OC0x 寄存器也会一直保持它的数值。

注意 COM0x1:0 和比较数据都不是双缓冲的。 COM0x1:0 的改变将立即生效。

比较匹配输出单元

比较匹配模式控制位 COM0x1:0 具有双重功能。波形发生器利用 COM0x1:0 来确定下一次比较匹配发生时的输出比较状态 (OC0x) ； COM0x1:0 还控制 OC0x 引脚输出信号的来源。 Figure 29 为受 COM0x1:0 设置影响的简化逻辑框图。 I/O 寄存器、 I/O 位和 I/O 引脚以粗体表示。图中只给出了受 COM0x1:0 影响的通用 I/O 端口控制寄存器 (DDR 和 PORT)。谈及OC0x状态时指的是内部OC0x寄存器，而不是OC0x引脚。系统复位时OC0x寄存器清零。

Figure 29. 比较匹配输出单元原理图



如果COM0x1:0不全为零，通用I/O口功能将被波形发生器的输出比较功能取代。但OC0x引脚为输入还是输出仍然由数据方向寄存器 DDR 控制。在使用 OC0x 功能之前首先要通过数据方向寄存器的 DDR_OC0x 位将此引脚设置为输出。端口功能与波形发生器的工作模式无关。

输出比较逻辑的设计允许 OC0x 状态在输出之前首先进行初始化。要注意某些 COM0x1:0 设置保留给了其他操作类型，详见 P69 “8 位 定时器 / 计数器寄存器的说明 ”。

比较输出模式和波形产生

波形发生器利用 COM0x1:0 的方法在普通模式、 CTC 模式和 PWM 模式下有所区别。对于所有的模式，设置 COM0x1:0 = 0 表明比较匹配发生时波形发生器不会操作 OC0x 寄存器。非PWM模式的比较输出请参见 P62Figure 28；快速PWM的比较输出示于 P53Table 26 ；相位修正 PWM 的比较输出在 P53Table 27 有描述。

改变 COM0x1:0 将影响写入数据后的第一次比较匹配。对于非 PWM 模式，可以通过使用 FOC0x 来立即产生效果。

工作模式

工作模式 - T/C 和输出比较引脚的行为 - 由波形发生模式 (WGM02:0) 及比较输出模式 (COM0x1:0) 的控制位决定。比较输出模式对计数序列没有影响，而波形产生模式对计数序列则有影响。 COM0x1:0 控制 PWM 输出是否为反极性。非 PWM 模式时 COM0x1:0 控制输出是否应该在比较匹配发生时置位、清零，或是电平取反 (P63 “ 比较匹配输出单元 ”)。

具体的时序信息请参考 P67“T/C 时序图 ” 的 Figure 33、 Figure 34、 Figure 35 与 Figure 36。

普通模式

普通模式 (WGM02:0 = 0) 为最简单的工作模式。在此模式下计数器不停地累加。计到 8 比特的最大值后 (TOP = 0xFF)，由于数值溢出计数器简单地返回到最小值 0x00 重新开始。在 TCNT0 为零的同一个定时器时钟里 T/C 溢出标志 TOV0 置位。此时 TOV0 有点象第 9 位，只是只能置位，不会清零。但由于定时器中断服务程序能够自动清零 TOV0，因此可以通过软件提高定时器的分辨率。在普通模式下没有什么需要特殊考虑的，用户可以随时写入新的计数器数值。

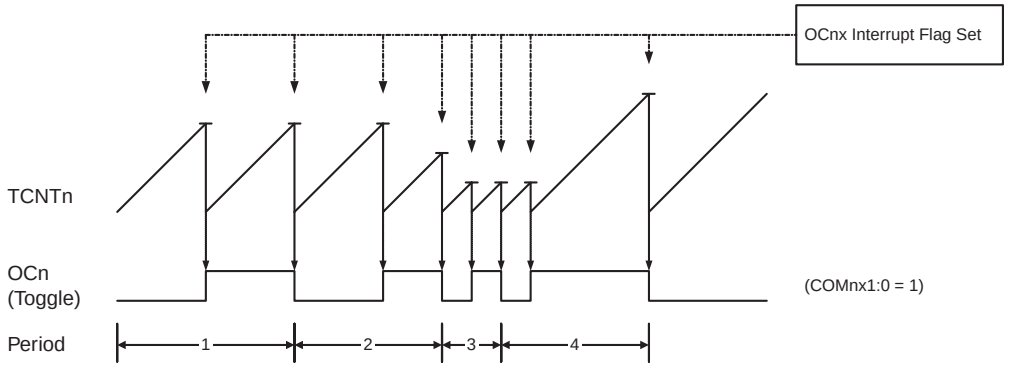
输出比较单元可以用来产生中断。但是不推荐在普通模式下利用输出比较来产生波形，因为这会占用太多的 CPU 时间。

CTC(比较匹配时清零定时器) 模式

在 CTC 模式 (WGM02:0 = 2) 下 OCR0A 寄存器用于调节计数器的分辨率。当计数器的数值 TCNT0 等于 OCR0A 时计数器清零。OCR0A 定义了计数器的 TOP 值，亦即计数器的分辨率。这个模式使得用户可以很容易地控制比较匹配输出的频率，也简化了外部事件计数的操作。

CTC模式的时序图为Figure 30。计数器数值TCNT0一直累加到TCNT0与OCR0A匹配，然后 TCNT0 清零。

Figure 30. CTC 模式的时序图



利用 OCF0A 标志可以在计数器数值达到 TOP 时产生中断。在中断服务程序里可以更新TOP的数值。由于CTC模式没有双缓冲功能,在计数器以无预分频器或很低的预分频器工作的时候将 TOP 更改为接近 BOTTOM 的数值时要小心。如果写入的 OCR0A 数值小于当前 TCNT0 的数值，计数器将丢失一次比较匹配。在下一次比较匹配发生之前，计数器不得不先计数到最大值 0xFF，然后再从 0x00 开始计数到 OCF0A。

为了在CTC模式下得到波形输出,可以设置OC0A在每次比较匹配发生时改变逻辑电平。这可以通过设置 COM0A1:0 = 1 来完成。在期望获得 OC0A 输出之前，首先要将其端口设置为输出。波形发生器能够产生的最大频率为 $f_{OC0} = f_{clk\_I/O}/2$ (OCR0 A= 0x00)。频率由如下公式确定：

$$f_{OCnx} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnx)}$$

变量 N 代表预分频因子 (1、 8、 64、 256 或 1024)。

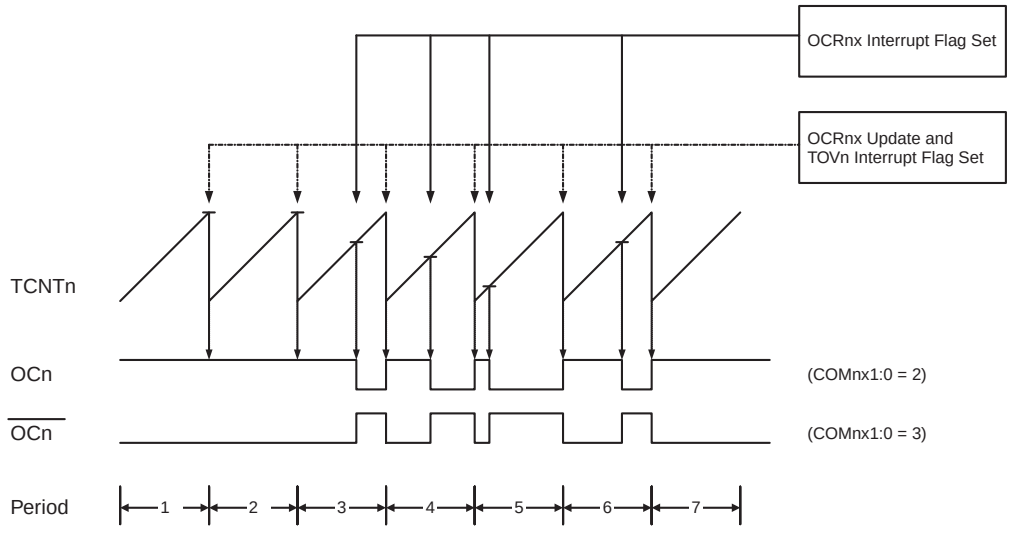
在普通模式下， TOV0 标志的置位发生在计数器从 MAX 变为 0x00 的定时器时钟周期。

快速 PWM 模式

快速 PWM 模式 (WGM02:0 = 3 或 7) 可用来产生高频的 PWM 波形。快速 PWM 模式与其他 PWM 模式的不同之处是其单斜坡工作方式。计数器从 BOTTOM 计到 TOP，然后立即回到 BOTTOM 重新开始。当 WGM2:0 = 3 时，TOP 值为 0xFF ；当 WGM2:0 = 7 时，TOP值为OCR0A。对于普通的比较输出模式,输出比较引脚OC0x在TCNT0与OCR0x匹配时清零，在 BOTTOM 时置位；对于反向比较输出模式， OC0x 的动作正好相反。由于使用了单斜坡模式,快速 PWM 模式的工作频率比使用双斜坡的相位修正 PWM 模式高一倍。此高频操作特性使得快速 PWM 模式十分适合于功率调节，整流和 DAC 应用。高频可以减小外部元器件 (电感，电容) 的物理尺寸，从而降低系统成本。

工作于快速 PWM 模式时，计数器的数值一直增加到 TOP，然后在后面的一个时钟周期清零。具体的时序图为 Figure 29。图中柱状的 TCNT0 表示这是单边斜坡操作。方框图同时包含了普通的 PWM 输出以及反向 PWM 输出。TCNT0 斜坡上的短水平线表示 OCR0x 和 TCNT0 的比较匹配。

Figure 31. 快速 PWM 模式时序图



计时器数值达到 TOP 时 T/C 溢出标志 TOV0 置位。如果中断使能，在中断服务程序可以更新比较值。

工作于快速PWM模式时,比较单元可以在OC0x引脚上输出PWM波形。设置COM0x1:0为 2 可以产生普通的 PWM 信号；为 3 则可以产生反向 PWM 波形 : 若 WGM02 位设置，

设置 COM0A1:0 位为 "1" 允许 AC0A 引脚转换到比较匹配。该选项对 OC0B 引脚无效 (参见 P53Table 26)。要想在引脚上得到输出信号还必须将OC0x的数据方向设置为输出。产生 PWM 波形的机理是 OC0x 寄存器在 OCR0x 与 TCNT0 匹配时置位 (或清零) , 以及在计数器清零 (从 TOP 变为 BOTTOM) 的那一个定时器时钟周期清零 (或置位)。

输出的 PWM 频率可以通过如下公式计算得到：

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

变量 N 代表分频因子 (1、8、64、256 或 1024)。

OCR0A 寄存器为极限值时表示快速 PWM 模式的一些特殊情况。若 OCR0A 等于 BOTTOM , 输出为出现在第MAX+1个定时器时钟周期的窄脉冲；OCR0A为MAX时 , 根据 COM0A1:0 的设定，输出恒为高电平或低电平。

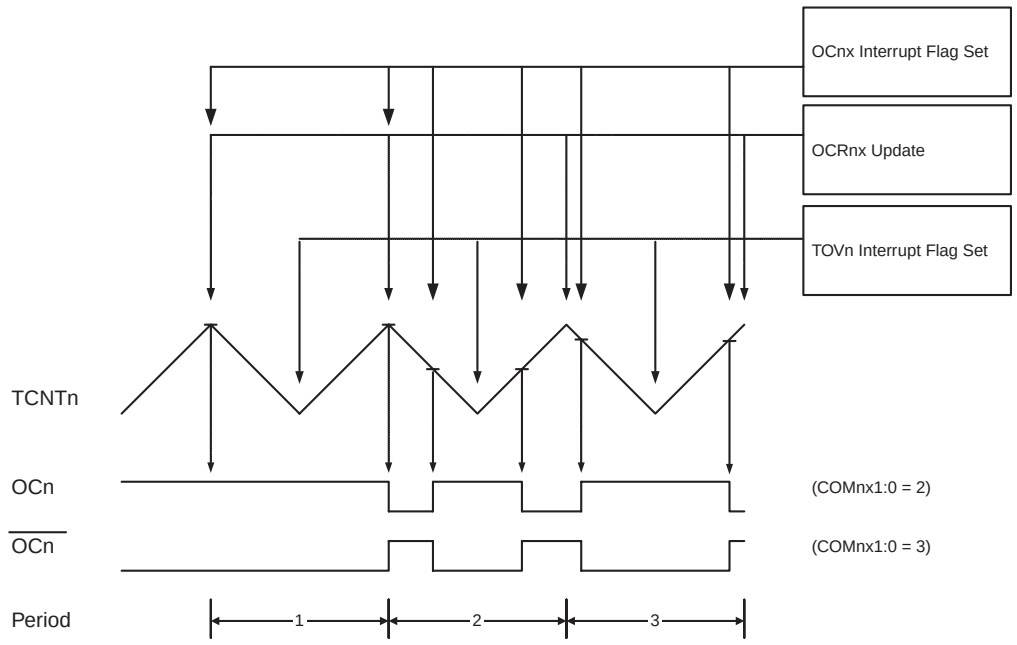
通过设定OC0x在比较匹配时进行逻辑电平取反(COM0x1:0 = 1) , 可以得到占空比为50% 的周期信号。 OCR0A 为 0 时信号有最高频率 $f_{oc0} = f_{clk\_I/O}/2$。这个特性类似于 CTC 模式下的 OC0A 取反操作，不同之处在于快速 PWM 模式具有双缓冲。

相位修正 PWM 模式

相位修正 PWM 模式 (WGM02:0 = 1 或 5) 为用户提供了一个获得高精度相位修正 PWM 波形的方法。此模式基于双斜坡操作。计时器重复地从 BOTTOM 计到 TOP , 然后又从 TOP 倒退回到 BOTTOM。当 WGM2:0 = 1 时， TOP 值为 0xFF ；当 WGM2:0 = 5 时， TOP 值为 OCR0A 。在一般的比较输出模式下，当计时器往 TOP 计数时若发生了 TCNT0 与 OCR0x 的匹配 , OC0x 将清零为低电平；而在计时器往 BOTTOM 计数时若发生了 TCNT0 与 OCR0x 的匹配， OC0x 将置位为高电平。工作于反向输出比较时则正好相反。与单斜坡操作相比 , 双斜坡操作可获得的最大频率要小。但由于其对称的特性，十分适合于电机控制。

计时器不断地累加直到 TOP , 然后开始减计数。在一个定时器时钟周期里 TCNT0 的值等于 TOP。时序图可参见 Figure 32。图中 TCNT0 的数值用柱状图表示，以说明双斜坡操作。本图同时说明了普通 PWM 的输出和反向 PWM 的输出。 TCNT0 斜坡上的小横条表示 OCR0x 与 TCNT0 的比较匹配。

Figure 32. 相位修正 PWM 模式的时序图



当计时器达到 BOTTOM 时 T/C 溢出标志位 TOV0 置位。此标志位可用来产生中断。

工作于相位修正 PWM 模式时，比较单元可以在 OC0x 引脚产生 PWM 波形：将 COM0x1:0 设置为 2 产生普通相位的 PWM，设置 COM0x1:0 为 3 产生反向 PWM 信号 ：若 WGM02 位设置，设置 COM0A0 位为 "1" 允许 OC0A 引脚转换到比较匹配。该选项对 OC0B 引脚无效 (参见 P53Table 27)。要想在引脚上得到输出信号还必须将 OC0x 的数据方向设置为输出。OCR0x 和 TCNT0 比较匹配发生时 OC0x 寄存器将产生相应的清零或置位操作，从而产生 PWM 波形。工作于相位修正模式时 PWM 频率可由下式公式获得：

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

变量 N 表示预分频因子 (1、8、64、 256 或 1024)。

OCR0A 寄存器处于极值代表了相位修正 PWM 模式的一些特殊情况。在普通 PWM 模式下，若 OCR0A 等于 BOTTOM，输出一直保持为低电平；若 OCR0A 等于 MAX，则输出保持为高电平。反向 PWM 模式则正好相反。

在 Figure 32 的第 2 个周期，虽然没有发生比较匹配， OCn 也出现了一个从高到低的跳变。其目的是保证波形在 BOTTOM 两侧的对称。没有比较匹配时有两种情况会出现跳变

- 如 Figure 32 所示，OCR0A 的值从 MAX 改变为其他数据。当 OCR0A 值为 MAX 时，引脚 OCn 的输出应该与前面降序记数比较匹配的结果相同。为了保证波形在 BOTTOM 两侧的对称，当 T/C 的数值为 MAX 时，引脚 OCn 的输出又必须符合后面升序记数比较匹配的结果。
- 定时器从一个比 OCR0A 高的值开始记数，并因而丢失了一次比较匹配。系统因此引入发生 OCn 却仍然有跳变的现象。

T/C 时序图

T/C 是同步电路，因此其时钟 clk_{T0} 可以表示为时钟使能信号，如下图所示。图中还说明了中断标志设置的时间。 Figure 33 给出了基本的 T/C 工作时序，以及除了相位修正 PWM 模式之外其他模式接近 MAX 时的记数序列。

Figure 33. T/C 时序图，无预分频器

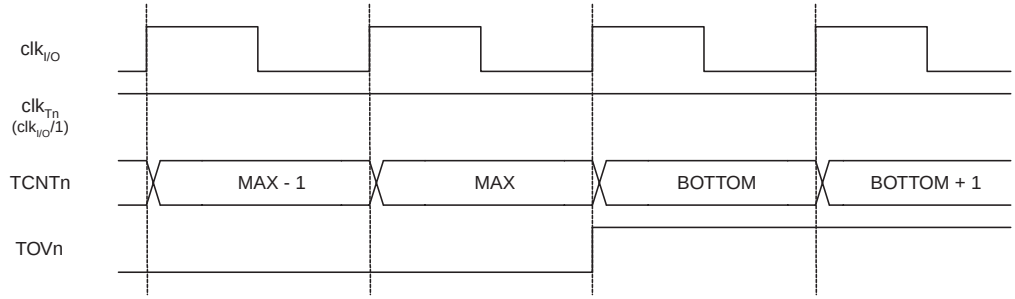


Figure 33 所示为相同的工作时序，但有预分频。

Figure 34. T/C 时序图，预分频器为 $f_{clk\_I/O}/8$

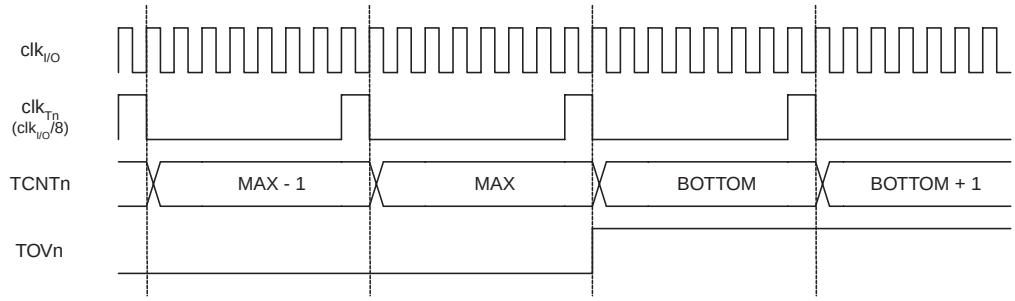


Figure 35 给出了各种模式下 OCF0B 以及（除了 CTC 模式)OCF0A 的置位情况，其中 OCR0A 为 TOP。

Figure 35. T/C 时序图， OCF0x 置位，预分频器为 $f_{clk\_I/O}/8$

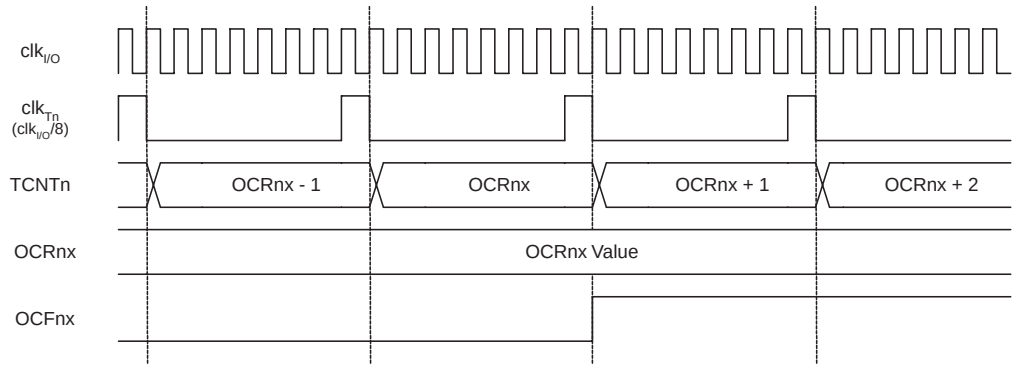
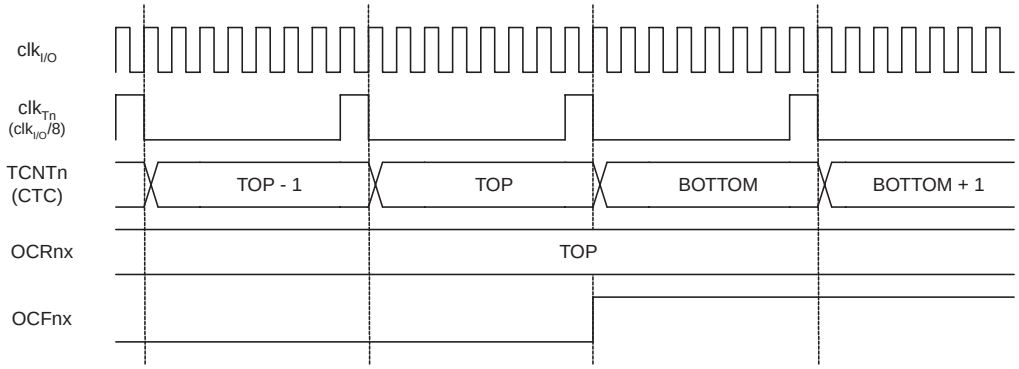


Figure 36 给出了 CTC 模式与快速 PWM 模式下 OCF0A 置位和 TCNT0 清除的情况，其中 OCR0A 为 TOP。

Figure 36. T/C 时序图， CTC 模式，预分频器为 $f_{clk\_I/O}/8$



8 位定时器 / 计数器寄存器的说明

T/C 控制寄存器 A - TCCR0A

Bit	7	6	5	4	3	2	1	0	
	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
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	

- **Bits 7:6 – COM0A1:0: 强制输出比较 A**

这两位控制输出比较引脚 (OC0A) 状态。如果 COM0A1:0 中至少有一位置位，OC0A 输出替代与其相联的 I/O 口的普通端口功能。但要注意必须设置与 OC0A 相应的 DDR 位，来使能输出驱动。

当 OC0A 与引脚相联，COM0A1:0 位的功能由 WGM02:0 位的设置决定。Table 34 给出当 WGM02:0 位设为普通模式或 CTC 模式 (非 PWM) 时 COM0A1:0 位的功能。

Table 34. 比较输出模式，非 PWM 模式

COM0A1	COM0A0	说明
0	0	正常的端口操作，不与 OC0A 相连接
0	1	比较匹配发生时 OC0A 取反
1	0	比较匹配发生时 OC0A 清零
1	1	比较匹配发生时 OC0A 置位

Table 35 给出了当 WGM01:0 设置为快速 PWM 模式时 COM01:0 的功能。

Table 35. 比较输出模式，快速 PWM 模式 (1)

COM0A1	COM0A0	说明
0	0	正常的端口操作，不与 OC0A 相连接
0	1	WGM02 = 0: 正常的端口操作，不与 OC0A 相连接 WGM02 = 1: 比较匹配发生时 OC0A 取反
1	0	比较匹配发生时 OC0A 清零，计数到 TOP 时 OC0A 置位
1	1	比较匹配发生时 OC0A 置位，计数到 TOP 时 OC0A 清零

Note: 1. 一个特殊情况是 OCR0A 等于 TOP , 且 COM01 置位。此时比较匹配将被忽略 , 而计数到 TOP 时 OC0 的动作继续有效。详细信息请参见 P65“ 快速 PWM 模式 ”。

Table 36 给出了当 WGM02:0 设置为相位修正 PWM 模式时 COM0A1:0 的功能。

Table 36. 比较输出模式，相位修正 PWM 模式 (1)

COM0A1	COM0A0	说明
0	0	正常的端口操作，不与 OC0A 相连接
0	1	WGM02 = 0: 正常的端口操作，不与 OC0A 相连接 WGM02 = 1: 比较匹配发生时 OC0A 取反
1	0	在升序计数时发生比较匹配将清零 OC0A ；降序计数时发生比较匹配将置位 OC0A
1	1	在升序计数时发生比较匹配将置位 OC0A ；降序计数时发生比较匹配将清零 OC0A

Note: 1. 一个特殊情况是 OCR0A 等于 TOP , 且 COM0A1 置位。此时比较匹配将被忽略 , 而计数到 TOP 时 OC0A 的动作继续有效。详细信息请参见 P66“ 相位修正 PWM 模式 ”。

- **Bits 5:4 – COM0B1:0: 比较匹配输出模式 B**

这些位决定了比较匹配发生时输出引脚 OC0B 的电平。如果 COM0B1:0 中的一位或全部都置位，OC0B 以比较匹配输出的方式进行工作。同时其方向控制位要设置为 1 以使能输出驱动器。

当 OC0B 连接到物理引脚上时，COM0B1:0 的功能依赖于 WGM01:0 的设置。Table 37 给出了当 WGM02:0 设置为普通模式或 CTC 模式时 COM0B1:0 的功能。

Table 37. 比较输出模式，非 PWM 模式

COM0B1	COM0B0	说明
0	0	正常的端口操作，不与 OC0B 相连接
0	1	比较匹配发生时 OC0B 取反
1	0	比较匹配发生时 OC0B 清零
1	1	比较匹配发生时 OC0B 置位

Table 38 给出了当 WGM02:0 设置为快速 PWM 模式时 COM0B1:0 的功能。

Table 38. 比较输出模式，快速 PWM 模式 (1)

COM0B1	COM0B0	说明
0	0	正常的端口操作，不与 OC0B 相连接
0	1	保留
1	0	比较匹配发生时 OC0B 清零，计数到 TOP 时 OC0B 置位
1	1	比较匹配发生时 OC0B 置位，计数到 TOP 时 OC0B 清零

Note: 1. 一个特殊情况是OCR0B等于TOP，且COM0B1置位。此时比较匹配将被忽略，而计数到 TOP 时 OC0B 的动作继续有效。详细信息请参见 P65“ 快速 PWM 模式 ”。

Table 39 给出了当 WGM02:0 设置为相位修正 PWM 模式时 COM0B1:0 的功能。

Table 39. 比较输出模式，相位修正 PWM 模式 (1)

COM0B1	COM0B0	说明
0	0	正常的端口操作，不与 OC0B 相连接
0	1	保留
1	0	在升序计数时发生比较匹配将清零 OC0B ；降序计数时发生比较匹配将置位 OC0B
1	1	在升序计数时发生比较匹配将置位 OC0B ；降序计数时发生比较匹配将清零 OC0B

Note: 1. 一个特殊情况是OCR0B等于TOP，且COM0B1置位。此时比较匹配将被忽略，而计数到 TOP 时 OC0B 的动作继续有效。详细信息请参见 P66“ 相位修正 PWM 模式 ”。

- **Bits 3, 2 – Res: 保留**

保留位，该操作返回值为零。

- **Bits 1:0 – WGM01:0: 波形产生模式**

这两位与 TCCR0B 寄存器中的 WGM02 位一起控制计数器的计数顺序，计数器的 TOP 值及产生哪种波形，见 Table 40。T/C 单元支持的工作模式有：正常模式 (计数器)，CTC 模式，及两种 PWM 模式 (见 P87“ 工作模式 ”)。

Table 40. 波形产生位说明

Mode	WGM2	WGM1	WGM0	T/C 工作模式	TOP	OCRx 更新	TOV 标志设置 (1)(2)
0	0	0	0	正常	0xFF	立即	MAX
1	0	0	1	相位修正 PWM	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	立即	MAX
3	0	1	1	快速 PWM	0xFF	TOP	MAX
4	1	0	0	保留	–	–	–
5	1	0	1	相位修正 PWM	OCRA	TOP	BOTTOM
6	1	1	0	保留	–	–	–
7	1	1	1	快速 PWM	OCRA	TOP	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

T/C 控制寄存器 B - TCCR0B

Bit	7	6	5	4	3	2	1	0	
	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
读 / 写	W	W	R	R	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC0A: 强制输出比较 A**

FOC0A 仅在 WGM 指明非 PWM 模式时才有效。

但是，为了保证与未来器件的兼容性，在使用 PWM 时，写 TCCR0B 要对其清零。对其写 1 后，波形发生器将立即进行比较操作。比较匹配输出引脚 OC0A 将按照 COM0A1:0 的设置输出相应的电平。要注意 FOC0A 类似一个锁存信号，真正对强制输出比较起作用的是 COM0A1:0 的设置。

FOC0A不会引发任何中断，也不会在利用OCR0A作为TOP的CTC模式下对定时器进行清零的操作。

读 FOC0A 的返回值永远为 0。

- **Bit 6 – FOC0B: 强制输出比较 B**

FOC0B 仅在 WGM 指明非 PWM 模式时才有效。

但是，为了保证与未来器件的兼容性，在使用 PWM 时，写 TCCR0B 要对其清零。对其写 1 后，波形发生器将立即进行比较操作。比较匹配输出引脚 OC0B 将按照 COM0B1:0 的设置输出相应的电平。要注意 FOC0B 类似一个锁存信号，真正对强制输出比较起作用的是 COM0B1:0 的设置。

FOC0B不会引发任何中断，也不会在利用OCR0B作为TOP的CTC模式下对定时器进行清零的操作。

读 FOC0B 的返回值永远为 0。

- **Bits 5:4 – Res: 保留**

保留位，返回值为零。

- **Bit 3 – WGM02: 波形产生模式**

见 P69“T/C 控制寄存器 A - TCCR0A” 中说明。

- **Bits 2:0 – CS02:0: 时钟选择**

这三位用于选择 T/C 的时钟源，见 P73Table 41。

Table 41. 时钟选择位说明

CS02	CS01	CS00	说明
0	0	0	无时钟， T/C 不工作
0	0	1	$clk_{I/O}$/1 (没有预分频)
0	1	0	$clk_{I/O}$/8 (来自预分频器)
0	1	1	$clk_{I/O}$/64 (来自预分频器)
1	0	0	$clk_{I/O}$/256 (来自预分频器)
1	0	1	$clk_{I/O}$/1024 (来自预分频器)
1	1	0	时钟由 T0 引脚输入，下降沿触发
1	1	1	时钟由 T0 引脚输入，上升沿触发

如果 T/C0 使用外部时钟，即使 T0 被配置为输出，其上的电平变化仍然会驱动记数器。利用这一特性可通过软件控制记数。

T/C 寄存器 - TCNT0

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
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	

通过 T/C 寄存器可以直接对计数器的 8 位数据进行读写访问。对 TCNT0 寄存器的写访问将在下一个时钟阻止比较匹配。在计数器运行的过程中修改 TCNT0 的数值有可能丢失一次 TCNT0 和 OCR0x 的比较匹配。

输出比较寄存器 A - OCR0A

Bit	7	6	5	4	3	2	1	0	
	OCR0A[7:0]								OCR0A
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	

输出比较寄存器包含一个 8 位的数据，不间断地与计数器数值 TCNT0 进行比较。匹配事件可以用来产生输出比较中断，或者用来在 OC0A 引脚上产生波形。

输出比较寄存器 B - OCR0B

Bit	7	6	5	4	3	2	1	0	
	OCR0B[7:0]								OCR0B
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	

输出比较寄存器包含一个 8 位的数据，不间断地与计数器数值 TCNT0 进行比较。匹配事件可以用来产生输出比较中断，或者用来在 OC0B 引脚上产生波形。

T/C 中断屏蔽寄存器 - TIMSK

Bit	7	6	5	4	3	2	1	0	
	TOIE1	OCIE1A	OCIE1B	–	ICIE1	OCIE0B	TOIE0	OCIE0A	TIMSK
读 / 写	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

- **Bit 4 – Res: 保留**

保留位，返回值为零。

- **Bit 2 – OCIE0B: T/C0 输出比较匹配 B 中断使能**

当 OCIE0B 和状态寄存器的全局中断使能位 I 都为 ”1” 时，T/C 的输出比较匹配 B 中断使能。当 T/C 的比较匹配发生，即 TIFR 中的 OCF0B 置位时，中断服务程序得以执行。

- **Bit 1 – TOIE0: T/C0 溢出中断使能**

当 TOIE0 和状态寄存器的全局中断使能位 I 都为 ”1” 时，T/C0 的溢出中断使能。当 T/C0 发生溢出，即 TIFR 中的 TOV0 位置位时，中断服务程序得以执行。

- **Bit 0 – OCIE0A: T/C0 输出比较匹配 A 中断使能**

当 OCIE0A 和状态寄存器的全局中断使能位 I 都为 ”1” 时， T/C0 的输出比较匹配 A 中断使能。当 T/C0 的比较匹配发生，即 TIFR 中的 OCF0A 置位时，中断服务程序得以执行。

T/C 中断标志寄存器 - TIFR

Bit	7	6	5	4	3	2	1	0	
	TOV1	OCF1A	OCF1B	–	ICF1	OCF0B	TOV0	OCF0A	TIFR
读 / 写	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

- **Bit 4 – Res: 保留**

保留位，返回值为零。

- **Bit 2 – OCF0B: 输出比较标志 0 B**

当 T/C 与 OCR0B(输出比较寄存器 0B) 的值匹配时， OCF0B 置位。此位在中断服务程序里硬件清零，也可以对其写 1 来清零。当 SREG 中的位 I、 OCIE0B(T/C0 比较 B 匹配中断使能) 和 OCF0B 都置位时，中断服务程序得到执行。

- **Bit 1 – TOV0: T/C0 溢出标志**

当 T/C0 溢出时， TOV0 置位。执行相应的中断服务程序时此位硬件清零。此外， TOV0 也可以通过写 1 来清零。当 SREG 中的位 I、 TOIE0(T/C0 溢出中断使能) 和 TOV0 都置位时，中断服务程序得到执行。

该标志的设置决定于 WGM02:0 位的设置，参见 Table 40，P71“ 波形产生位说明 ” 。

- **Bit 0 – OCF0A: 输出比较标志 0 A**

当 T/C0 与 OCR0A(输出比较寄存器 0) 的值匹配时，OCF0A 置位。此位在中断服务程序里硬件清零，也可以对其写 1 来清零。当 SREG 中的位 I、 OCIE0A(T/C0 比较匹配中断使能) 和 OCF0A 都置位时，中断服务程序得到执行。

T/C0 与 T/C1 预分频器

T/C1 与 T/C0 共用一个预分频模块，但可以有不同的预分频设置。下面的说明对 T/C1 与 T/C0 均适用。

内部时钟源

当 CSn2:0 = 1 时，系统内部时钟直接作为 T/C 的时钟源，这也是 T/C 最高频率的时钟源 $f_{CLK\_I/O}$，与系统时钟频率相同。预分频器可以输出 4 个不同的时钟信号 $f_{CLK\_I/O}/8$、$f_{CLK\_I/O}/64$、$f_{CLK\_I/O}/256$ 或 $f_{CLK\_I/O}/1024$。

预分频器复位

预分频器是独立运行的。也就是说，其操作独立于 T/C 的时钟选择逻辑，且由 T/C1 与 T/C0 共用。由于预分频器不受 T/C 时钟选择的影响，预分频器的状态需要包含预分频时钟被用到何处这样的信息。一个典型的例子发生在定时器使能并由预分频器驱动 (6 > CSn2:0 > 1)的时候：从计时器使能到第一次开始计数可能花费 1 到N+1个系统时钟周期，其中 N 等于预分频因子 (8、 64、 256 或 1024)。

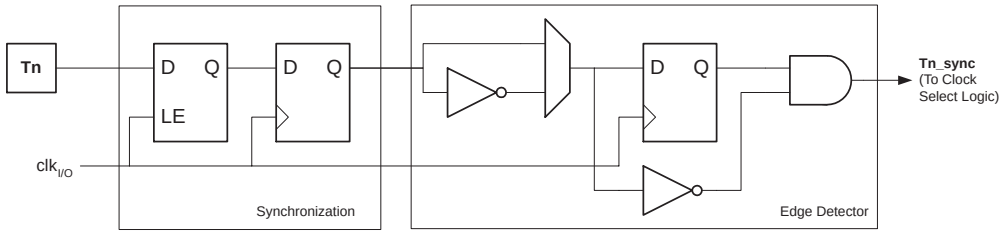
通过复位预分频器来同步 T/C 与程序运行是可能的。但必须注意另一个共用预分频器的 T/C。预分频器复位会影响所有与其连接的 T/C 周期。

外部时钟源

由 T1/T0 引脚提供的外部时钟源可以用作 T/C 时钟 clk_{T1}/clk_{T0}。引脚同步逻辑在每个系统时钟周期对引脚 T1/T0 进行采样。然后将同步（采样）信号送到边沿检测器。 Figure 37 给出了 T1/T0 同步采样与边沿检测逻辑的功能等效方框图。寄存器由内部系统时钟 $clk_{I/O}$ 的上跳沿驱动。当内部时钟为高时，锁存器可以看作时透明的。

CSn2:0 = 7 时边沿检测器检测到一个正跳变产生一个 clk_{T1} 脉冲； CSn2:0 = 6 时一个负跳变就产生一个 clk_{T1}/clk_{T0} 脉冲。

Figure 37. T1/T0 引脚采样



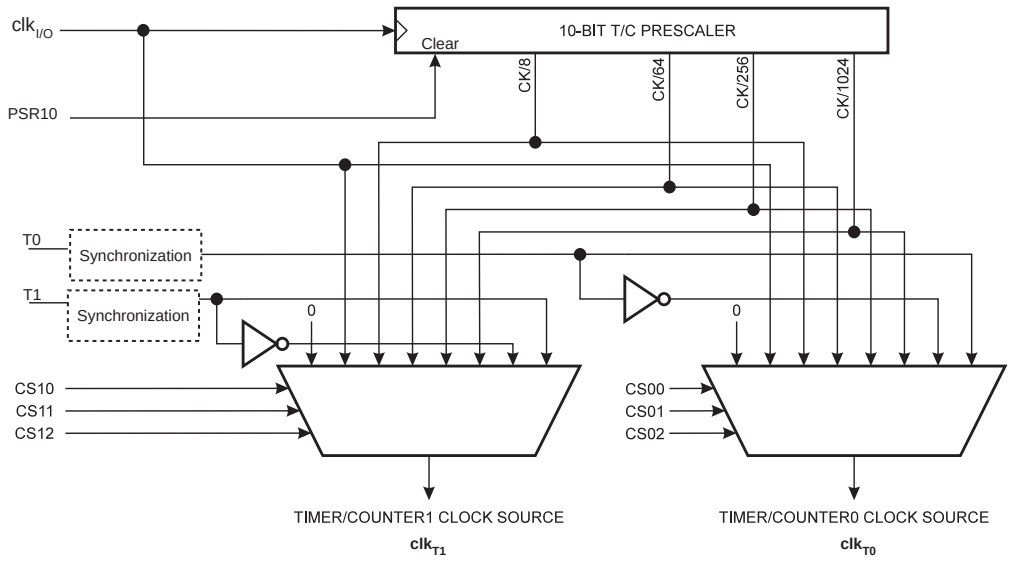
由于引脚上同步与边沿监测电路的存在，引脚 T1/T0 上的电平变化需要延时 2.5 到 3.5 个系统时钟周期才能使计数器进行更新。

禁止或使能时钟输入必须在 T1/T0 保持稳定至少一个系统时钟周期后才能进行，否则有产生错误 T/C 时钟脉冲的危险。

为保证正确的采样，外部时钟脉冲宽度必须大于一个系统时钟周期。在占空比为 50% 时外部时钟频率必须小于系统时钟频率的一半 ($f_{ExtClk} < f_{clk\_I/O}/2$)。由于边沿检测器使用的是采样这一方法，它能检测到的外部时钟最多是其采样频率的一半 (Nyquist 采样定理)。然而，由于振荡器（晶体、谐振器与电容）本身误差带来的系统时钟频率及占空比的差异，建议外部时钟的最高频率不要大于 $f_{clk\_I/O}/2.5$。

外部时钟源不送入预分频器。

Figure 38. T/C0 与 T/C1 预分频器 (1)



Note: 1. 输入引脚 (T1/T0) 的同步逻辑见 Figure 37.

通用 T/C 控制寄存器 - GTCCR

Bit	7	6	5	4	3	2	1	0	
	—	–	–	–	–	–	—	PSR10	GTCCR
读 / 写	R	R	R	R	R	R	R	R/W	
初始值	0	0	0	0	0	0	0	0	

- **Bits 7..1 – Res: 保留**

保留位，返回值为零。

- **Bit 0 – PSR10: T/C1 与 T/C0 预分频器复位**

置位时 T/C1 与 T/C0 的预分频器复位，操作完成后这一位由硬件自动清零。注意，由于 T/C1 与 T/C0 共用预分频器，因此预分频器复位对两个定时器均有影响。

16 位定时器 / 计数器 1

16位的T/C可以实现精确的程序定时(事件管理)、波形产生和信号测量。其主要特点如下:

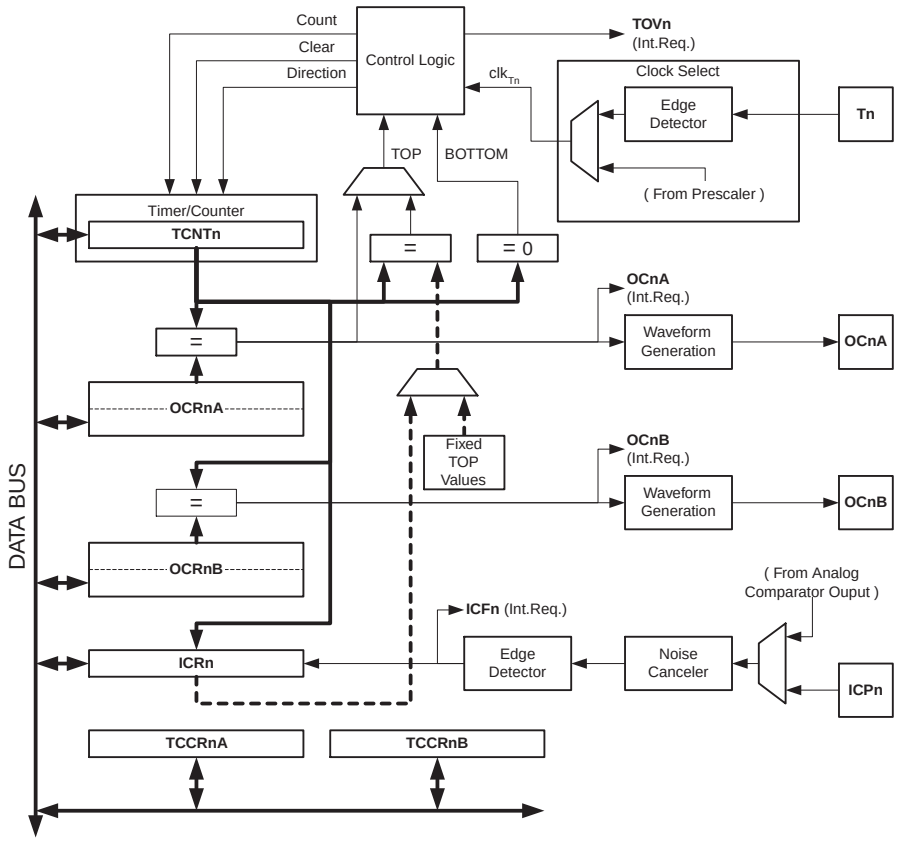
- **真正的 16 位设计 (即允许 16 位的 PWM)**
- **2 个独立的输出比较单元**
- **双缓冲的输出比较寄存器**
- **一个输入捕捉单元**
- **输入捕捉噪声抑制器**
- **比较匹配发生时清除寄存器 (自动重载)**
- **无干扰脉冲，相位正确的 PWM**
- **可变的 PWM 周期**
- **频率发生器**
- **外部事件计数器**
- **4 个独立的中断源 (TOV1、OCF1A、 OCF1B 与 ICF1)**

综述

本节大多数的寄存器和位定义以通用的方式表示。小写 “n” 表示 T/C 序号，小写 “x” 表示输出比较通道号。但是在写程序时要用完整的、精确的名称。如用 TCNT1 表示访问 T/C1 计数器值等。

16 位 T/C 的简化框图示于 Figure 39。I/O 引脚的实际位置请参见 P2“ATtiny2313 引脚配置 ”。CPU可访问的I/O寄存器，包括I/O位和I/O引脚以粗体表示。器件具体I/O寄存器与位定位见 P97“16 位定时器 / 计数器寄存器说明 ”。

Figure 39. 16 位 T/C 框图 (1)



Note: 1. 请参考 P2Figure 1， P56Table 25 和 P61Table 31 以获得 T/C1 的引脚定义。

寄存器

定时器 / 计数器 TCNT1、输出比较寄存器 OCR1A/B 与输入捕捉寄存器 ICR1 均为 16 位寄存器。访问 16 位寄存器必须通过特殊的步骤，详见 P78“ 访问 16 位寄存器 ”。 T/C 控制寄存器 TCCR1A/B 为 8 位寄存器，没有 CPU 访问的限制。中断请求 (图中简写为

Int.Req.) 信号在中断标志寄存器 TIFR1 都有反映。所有中断都可以由中断屏蔽寄存器 TIMSK 单独控制。图中未给出 TIFR 与 TIMSK。

T/C可由内部时钟通过预分频器或通过由T1引脚输入的外部时钟驱动。引发T/C数值增加(或减少)的时钟源及其有效沿由时钟选择逻辑模块控制。没有选择时钟源时 T/C 处于停止状态。时钟选择逻辑模块的输出称为 clk_{T1}。

双缓冲输出比较寄存器 OCR1A/B 一直与 T/C 的值做比较。波形发生器用比较结果产生 PWM 或在输出比较引脚 OC1A/B 输出可变频率的信号。参见 P84 “ 输出比较单元 ” 。比较匹配结果还可置位比较匹配标志 OCF1A/B，用来产生输出比较中断请求。

当输入捕捉引脚 ICP1 或模拟比较器输入引脚 (见 P141 “ 模拟比较器 ”) 有输入捕捉事件产生 (边沿触发) 时，当时的 T/C 值被传输到输入捕捉寄存器保存起来。输入捕捉单元包括一个数字滤波单元 (噪声消除器) 以降低噪声干扰。

在某些操作模式下， TOP 值或 T/C 的最大值可由 OCR1A 寄存器、 ICR1 寄存器，或一些固定数据来定义。在 PWM 模式下用 OCR1A 作为 TOP 值时， OCR1A 寄存器不能用作 PWM 输出。但此时 OCR1A 是双向缓冲的， TOP 值可在运行过程中得到改变。当需要一个固定的 TOP 值时可以使用 ICR1 寄存器 ,从而释放 OCR1A 来用作 PWM 的输出。

定义

以下定义适用于本节：

Table 42. 定义

BOTTOM	计数器计到 0x0000 时即达到 BOTTOM
MAX	计数器计到 0xFFFF (十进制的 65535) 时即达到 MAX
TOP	计数器计到计数序列的最大值时即达到 TOP。 TOP 值可以为固定值 0x00FF、0x01FF 或 0x03FF ,或是存储于寄存器 OCR1A 或 ICR1 里的数值 ,具体有赖于工作模式

兼容性

16位T/C是从以前版本的16位AVRT/C改进和升级得来的。它在如下方面与以前的版本完全兼容：

- 包括定时器中断寄存器在内的所有 16 位 T/C 相关的 I/O 寄存器的地址
- 包括定时器中断寄存器在内的所有 16 位 T/C 相关的寄存器位定位
- 中断向量

下列控制位名称已改，但具有相同的功能与寄存器单元：

- PWM10 改为 WGM10
- PWM11 改为 WGM11
- CTC1 改为 WGM12

16 位 T/C 控制寄存器中添加了下列位：

- TCCR1C 中加入 FOC1A 与 FOC1B
- TCCR1B 中加入 WGM13

16 位 T/C 的一些改进在某些特殊情况下将影响兼容性。

访问 16 位寄存器

TCNT1、OCR1A/B与ICR1是AVR CPU 通过8位数据总线可以访问的16位寄存器。读写16位寄存器需要两次操作。每个16位计时器都有一个8位临时寄存器用来存放其高8位数据。每个 16 位定时器所属的 16 位寄存器共用相同的临时寄存器。访问低字节会触发 16 位读或写操作。当 CPU 写入数据到 16 位寄存器的低字节时，写入的 8 位数据与存放在临时寄存器中的高 8 位数据组成一个 16 位数据，同步写入到 16 位寄存器中。当 CPU 读取 16 位寄存器的低字节时，高字节内容在读低字节操作的同时被放置于临时辅助寄存器中。

并非所有的 16 位访问都涉及临时寄存器。对 OCR1A/B 寄存器的读操作就不涉及临时寄存器。

写 16 位寄存器时，应先写入该寄存器的高位字节。而读 16 位寄存器时应先读取该寄存器的低位字节。

下面的例程说明了如何访问 16 位定时器寄存器。前提是假设不会发生更新临时寄存器内容的中断。同样的原则也适用于对 OCR1A/B 与 ICR1 寄存器的访问。使用 “C” 语言时，编译器会自动处理 16 位操作。

汇编代码例程 (1)

```
...
; 设置 TCNT1 为 0x01FF
ldi r17,0x01
ldi r16,0xFF
out TCNT1H,r17
out TCNT1L,r16
; 将 TCNT1 读入 r17:r16
in  r16,TCNT1L
in  r17,TCNT1H
...
```

C 代码例程 (1)

```
unsigned int i;
...
/* 设置 TCNT1 为 0x01FF */
TCNT1 = 0x1FF;
/* 将 TCNT1 读入 i */
i = TCNT1;
...
```

Note: 1. 本代码假定已经包含了合适的头文件。
当 I/O 寄存器为扩展 I/O 寄存器时，必须用诸如 “LDS” 、“STS”、“SBRS”、“SBRC”、“SBR” 与 “CBR” 等可访问扩展 I/O 寄存器的指令代替 “IN”、“OUT”、“SBIS”、“SBIC”、“CBI” 与 “SBI” 指令。

汇编代码例程中 r17:r16 寄存器对保存的是 TCNT1 的写入数据。

注意到 16 位寄存器的访问是一个基本操作是非常重要的。在对 16 位寄存器操作时，最好首先屏蔽中断响应，防止在主程序读写 16 位寄存器的两条指令之间发生这样的中断：它也访问同样的寄存器或其他的 16 位寄存器，从而更改了临时寄存器。如果这种情况发生，那么中断返回后临时寄存器中的内容已经改变，造成主程序对 16 位寄存器的读写错误。

下面的例程给出了读取 TCNT1 寄存器内容的基本操作。对 OCR1A/B 或 ICR1 的读操作可以使用相同的方法。

汇编代码例程 (1)

```
TIM16_ReadTCNT1:
  ; 保存全局中断标志
  in  r18,SREG
  ; 禁用中断
  cli
  ; 将 TCNT1 读入 r17:r16
  in  r16,TCNT1L
  in  r17,TCNT1H
  ; 恢复全局中断标志
  out SREG,r18
  ret
```

C 代码例程 (1)

```
unsigned int TIM16_ReadTCNT1( void )
{
  unsigned char sreg;
  unsigned int i;
  /* 保存全局中断标志 */
  sreg = SREG;
  /* 禁用中断 */
  _CLI();
  /* 将 TCNT1 读入 i */
  i = TCNT1;
  /* 恢复全局中断标志 */
  SREG = sreg;
  return i;
}
```

Note: 1. 本代码假定已经包含了合适的头文件。
本代码假定已经包含了合适的头文件。当 I/O 寄存器为扩展 I/O 寄存器时，必须用诸如 “LDS” 、“STS”、“SBRS”、“SBRC”、“SBR” 与 “CBR” 等可访问扩展 I/O 寄存器的指令代替 “IN”、 “OUT”、 “SBIS”、“SBIC”、 “CBI” 与 “SBI” 指令。

汇编代码例程中 TCNT1 的返回值在 r17:r16 寄存器对中。

下面的例程给出了写 TCNT1 寄存器的基本操作。对 OCR1A/B 或 ICR1 的写操作可以使用相同的方法。

汇编代码例程 (1)

```
TIM16_WriteTCNT1:
  ; 保存全局中断标志
  in  r18,SREG
  ; 禁用中断
  cli
  ; 设置 TCNT1 到 r17:r16
  out TCNT1H,r17
  out TCNT1L,r16
  ; 恢复全局中断标志
  out SREG,r18
  ret
```

C 代码例程 (1)

```
void TIM16_WriteTCNT1 ( unsigned int i )
{
  unsigned char sreg;
  unsigned int i;
  /* 保存全局中断标志 */
  sreg = SREG;
  /* 禁用中断 */
  _CLI();
  /* 设置 TCNT1 到 i */
  TCNT1 = i;
  /* 恢复全局中断标志 */
  SREG = sreg;
}
```

Note: 1. 本代码假定已经包含了合适的头文件。
本代码假定已经包含了合适的头文件。当 I/O 寄存器为扩展 I/O 寄存器时，必须用诸如 “LDS” 、“STS”、“SBRS”、“SBRC”、“SBR” 与 “CBR” 等可访问扩展 I/O 寄存器的指令代替 “IN”、 “OUT”、 “SBIS”、“SBIC”、 “CBI” 与 “SBI” 指令。

汇编代码例程中 r17:r16 寄存器对保存的是 TCNT1 的写入数据。

重复利用暂存器的高字节

如果对不只一个 16 位寄存器写入数据而且所有的寄存器高字节相同，则只需写一次高字节。前面讲到的基本操作在这种情况下同样适用。

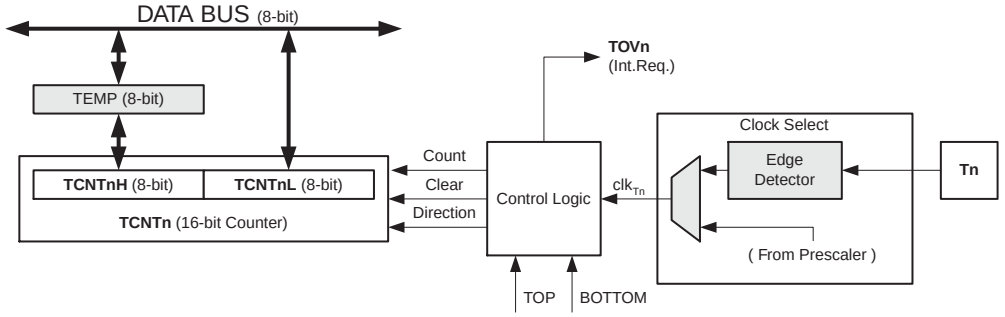
定时器 / 计数器的时钟源

T/C 时钟源可以来自内部，也可来自外部，由位于 T/C 控制寄存器 B(TCCR1B) 的时钟选择位 (CS12:0) 决定。时钟源与预分频器的描述见 P75“T/C0 与 T/C1 预分频器 ”。

计数器单元

16 位 T/C 的主要部分是可编程的 16 位双向计数器单元。Figure 40 给出了计数器与其外围电路方框图。

Figure 40. 计数器单元方框图



信号描述 (内部信号) :

Count	TCNT1 加 1 或减 1
Direction	确定是加操作还是减操作
Clear	TCNT1 清零
clk_{T1}	定时器 / 计数器时钟信号
TOP	表示 TCNT1 计数器到达最大值
BOTTOM	表示 TCNT1 计数器到达最小值 (0)

16 位计数器映射到两个 8 位 I/O 存储器位置：TCNT1H 为高 8 位，TCNT1L 为低 8 位。CPU 只能间接访问 TCNT1H 寄存器。CPU 访问 TCNT1H 时，实际访问的是临时寄存器 (TEMP)。读取 TCNT1L 时，临时寄存器的内容更新为 TCNT1H 的数值；而对 TCNT1L 执行写操作时，TCNT1H 被临时寄存器的内容所更新。这就使 CPU 可以在一个时钟周期里通过 8 位数据总线完成对 16 位计数器的读、写操作。此外还需要注意计数器在运行时的一些特殊情况。在这些特殊情况下对 TCNT1 写入数据会带来未知的结果。在合适的章节会对这些特殊情况进行具体描述。

根据工作模式的不同，在每一个 clk_{T1} 时钟到来时，计数器进行清零、加 1 或减 1 操作。clk_{T1} 由时钟选择位 CS12:0 设定。当 CS12:0= 0 时，计数器停止计数。不过 CPU 对 TCNT1 的读取与 clk_{T1} 是否存在无关。CPU 写操作比计数器清零和其他操作的优先级都高。

计数器的计数序列取决于寄存器 TCCR1A 和 TCCR1B 中标志位 WGM13:0 的设置。计数器的运行 (计数) 方式与通过 OC1x 输出的波形发生方式有很紧密的关系。计数序列与波形产生的详细描述请参见 P87“ 工作模式 ”。

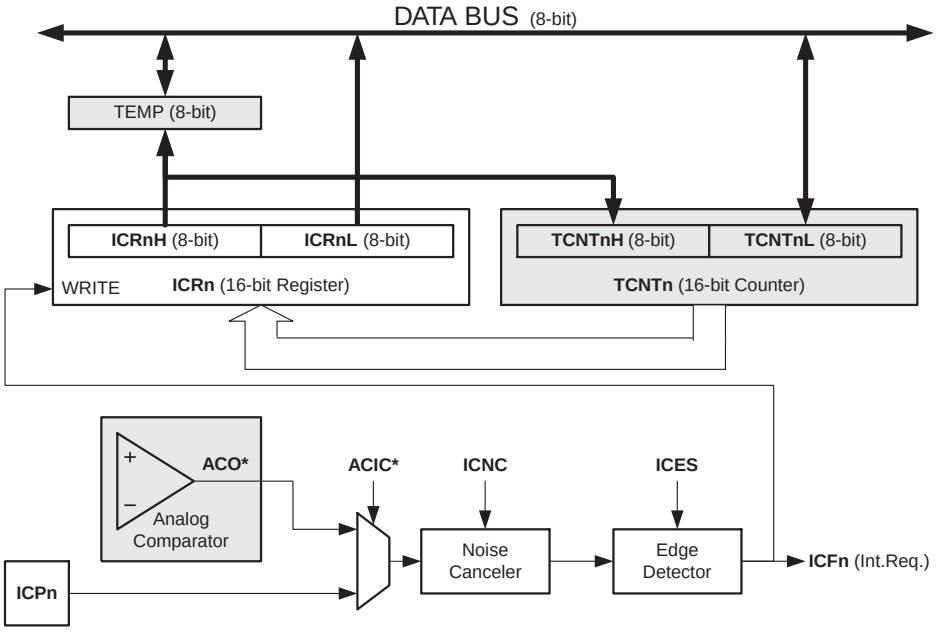
通过 WGM13:0 确定了计数器的工作模式之后，TOV1 的置位方式也就确定了。TOV1 可以用来产生 CPU 中断。

输入捕捉单元

T/C 的输入捕捉单元可用来捕获外部事件，并为其赋予时间标记以说明此时间的发生时刻。外部事件发生的触发信号由引脚 ICP1 输入，也可通过模拟比较器单元来实现。时间标记可用来计算频率、占空比及信号的其它特征，以及为事件创建日志。

输入捕捉单元方框图见 Figure 41。图中不直接属于输入捕捉单元的部分用阴影表示。寄存器与位中的小写 “n” 表示定时器 / 计数器编号。

Figure 41. 输入捕捉单元方框图



当引脚 ICP1 上的逻辑电平（事件）发生了变化，或模拟比较器输出 ACO 电平发生了变化，并且这个电平变化为边沿检测器所证实，输入捕捉即被激发：16 位的 TCNT1 数据被拷贝到输入捕捉寄存器 ICR1，同时输入捕捉标志位 ICF1 置位。如果此时 ICIE1 = 1，输入捕捉标志将产生输入捕捉中断。中断执行时 ICF1 自动清零，或者也可通过软件在其对应的 I/O 位置写入逻辑 "1" 清零。

读取 ICR1 时要先读低字节 ICR1L，然后再读高字节 ICR1H。读低字节时，高字节被复制到高字节临时寄存器 TEMP。 CPU 读取 ICR1H 时将访问 TEMP 寄存器。

对 ICR1 寄存器的写访问只存在于波形产生模式。此时 ICR1 被用作计数器的 TOP 值。写 ICR1 之前首先要设置 WGM13:0 以允许这个操作。对 ICR1 寄存器进行写操作时必须先将高字节写入 ICR1H I/O 位置，然后再将低字节写入 ICR1L。

请参见 P78“ 访问 16 位寄存器 ” 以了解更多的关于如何访问 16 位寄存器的信息。

输入捕捉触发源

输入捕捉单元的主要触发源是 ICP1。T/C1 还可用模拟比较输出作为输入捕捉单元的触发源。用户必须通过设置模拟比较控制与状态寄存器 ACSR 的模拟比较输入捕捉位 ACIC 来做到这一点。要注意的是，改变触发源有可能造成一次输入捕捉。因此在改变触发源后必须对输入捕捉标志执行一次清零操作以避免出现错误的结果。

ICP1与ACO的采样方式与T1引脚是相同的(P75Figure 37),使用的边沿检测器也一样。但是使能噪声抑制器后，在边沿检测器前会加入额外的逻辑电路并引入 4 个系统时钟周期的延迟。要注意的是，除去使用 ICR1 定义 TOP 的波形产生模式外， T/C 中的噪声抑制器与边沿检测器总是使能的。

输入捕捉也可以通过软件控制引脚 ICP1 的方式来触发。

噪声抑制器

噪声抑制器通过一个简单的数字滤波方案提高系统抗噪性。它对输入触发信号进行 4 次采样。只有当 4 次采样值相等时其输出才会送入边沿检测器。

置位 TCCR1B 的 ICNC1 将使能噪声抑制器。使能噪声抑制器后，在输入发生变化到 ICR1 得到更新之间将会有额外的 4 个系统时钟周期的延时。噪声抑制器使用的是系统时钟，因而不受预分频器的影响。

使用输入捕捉单元

使用输入捕捉单元的最大问题就是分配足够的处理器资源来处理输入事件。事件的时间间隔是关键。如果处理器在下一次事件出现之前没有读取 ICR1 的数据，ICR1 就会被新值覆盖，从而无法得到正确的捕捉结果。

使用输入捕捉中断时，中断程序应尽可能早的读取 ICR1 寄存器。尽管输入捕捉中断优先级相对较高，但最大中断响应时间与其它正在运行的中断程序所需的时间相关。

在任何输入捕捉工作模式下都不推荐在操作过程中改变 TOP 值。

测量外部信号的占空比时要求每次捕捉后都要改变触发沿。因此读取 ICR1 后必须尽快改变敏感的信号边沿。改变边沿后，ICF1 必须由软件清零 (在对应的 I/O 位置写 "1")。若仅需测量频率，且使用了中断发生，则不需对 ICF1 进行软件清零。

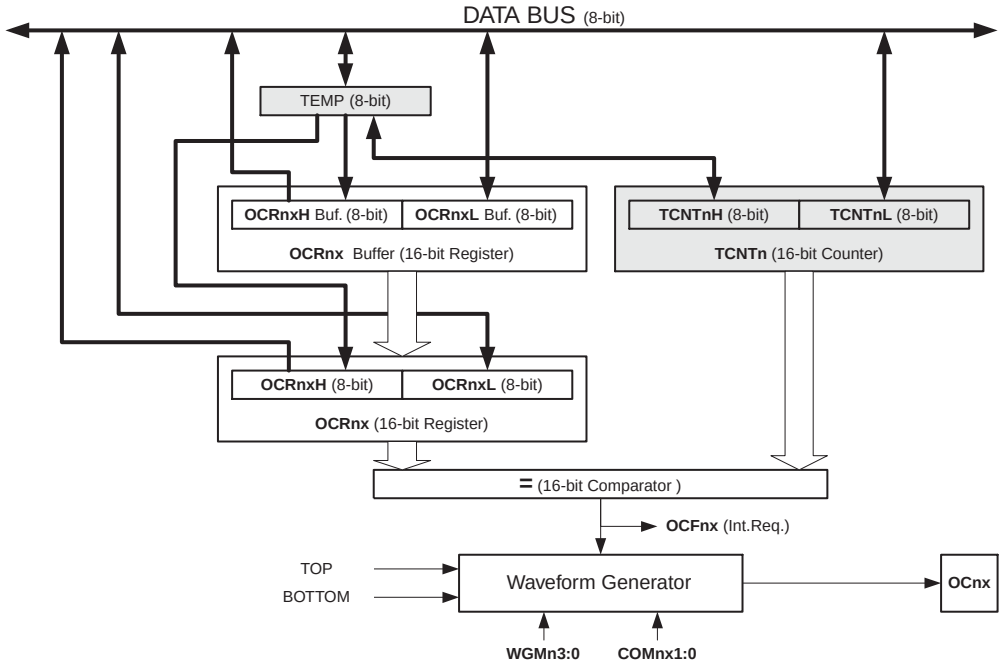
输出比较单元

16 位比较器持续比较 TCNT1 与 OCR1x 的内容，一旦发现它们相等，比较器立即产生一个匹配信号。然后 OCF1x 在下一个定时器时钟置位。如果此时 OCIE1x = 1，OCF1x 置位将引发输出比较中断。中断执行时 OCF1x 标志自动清零，或者通过软件在其相应的 I/O 位置写入逻辑 "1" 也可以清零。根据 WGM13:0 与 COM1x1:0 的不同设置，波形发生器用匹配信号生成不同的波形。波形发生器利用 TOP 和 BOTTOM 信号处理在某些模式下对极值的操作 (P87 " 工作模式 ")。

输出比较单元 A 的一个特质是定义 T/C 的 TOP 值 (即计数器的分辨率)。此外，TOP 值还用来定义通过波形发生器产生的波形的周期。

Figure 42 给出输出比较单元的方框图。寄存器与位上的小写 "n" 表示器件编号 (n = 1 表示 T/C1)，"x" 表示输出比较单元 (A/B)。框图中非输出比较单元部分用阴影表示。

Figure 42. 输出比较单元方框图



当 T/C 工作在 12 种 PWM 模式种的任意一种时，OCR1x 寄存器为双缓冲寄存器；而在正常工作模式和匹配时清零模式 (CTC) 双缓冲功能是禁止的。双缓冲可以实现 OCR1x 寄存器对 TOP 或 BOTTOM 的同步更新，防止产生不对称的 PWM 波形，消除毛刺。

访问 OCR1x 寄存器看起来很复杂，其实不然。使能双缓冲功能时，CPU 访问的是 OCR1x 缓冲寄存器；禁止双缓冲功能时 CPU 访问的则是 OCR1x 本身。OCR1x(缓冲或比较) 寄存器的内容只有写操作才能将其改变 (T/C 不会自动将此寄存器更新为 TCNT1 或 ICR1

的内容），所以 OCR1x 不用通过 TEMP 读取。但是象其他 16 位寄存器一样首先读取低字节是一个好习惯。由于比较是连续进行的，因此在写 OCR1x 时必须通过 TEMP 寄存器来实现。首先需要写入的是高字节 OCR1xH。当 CPU 将数据写入高字节的 I/O 地址时，TEMP 寄存器的内容即得到更新。接下来写低字节 OCR1xL。在此同时，位于 TEMP 寄存器的高字节数据被拷贝到 OCR1x 缓冲器，或是 OCR1x 比较寄存器。

请参见 P78“ 访问 16 位寄存器 ” 以了解更多的关于如何访问 16 位寄存器的信息。

强制输出比较

工作于非 PWM 模式时，可以通过对强制输出比较位 FOC1x 写 ”1” 的方式来产生比较匹配。强制比较匹配不会置位 OCF1x 标志，也不会重载 / 清零定时器，但是 OC1x 引脚将被更新，好象真的发生了比较匹配一样 (COMx1:0 决定 OC1x 是置位、清零，还是交替变化)。

写 TCNT1 将阻止比较匹配

CPU对TCNT1寄存器的写操作会阻止比较匹配的发生。这个特性可以用来将OCR1x初始化为与 TCNT1 相同的数值而不触发中断。

使用输出比较匹配单元

由于在任意模式下写 TCNT1 都将在下一个定时器时钟周期里阻止比较匹配，在使用输出比较时改变TCNT1就会有风险，不管T/C是否在运行。若写入TCNT1的数值等于OCR1x，比较匹配就被忽略了，造成不正确的波形发生结果。在 PWM 模式下，当 TOP 为可变数值时，不要赋予 TCNT1 和 TOP 相等的数值。否则会丢失一次比较匹配，计数器也将计到0xFFFF。类似地，在计数器进行降序计数时不要对TCNT1写入等于BOTTOM的数据。

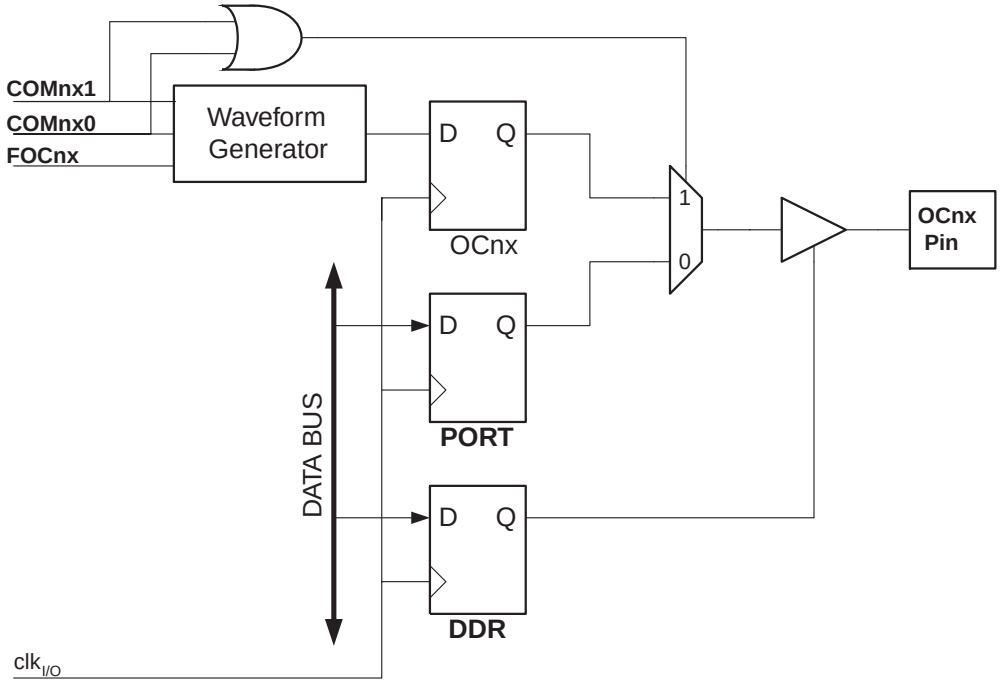
OC1x 的设置应该在设置数据方向寄存器之前完成。最简单的设置 OC1x 的方法是在普通模式下利用强制输出比较 FOC1x。即使在改变波形发生模式时 OC1x 寄存器也会一直保持它的数值。

COM1x1:0 和比较数据都不是双缓冲的。 COM1x1:0 的改变将立即生效。

比较匹配输出单元

比较匹配模式控制位 COM1x1:0 具有双重功能。波形发生器利用 COM1x1:0 来确定下一次比较匹配发生时的输出比较 OC1x 状态； COM1x1:0 还控制 OC1x 引脚输出的来源。Figure 43 为受 COM1x1:0 设置影响的逻辑的简化原理图。I/O 寄存器、I/O 位和 I/O 引脚以粗体表示。图中只给出了受 COM1x1:0 影响的通用 I/O 端口控制寄存器 (DDR 和 PORT)。谈及 OC1x 状态时指的是内部 OC1x 寄存器，而不是 OC1x 引脚的状态。系统复位时 COM1x 寄存器复位为 "0”。

Figure 43. 比较匹配输出单元原理图



只要 COM1x1:0 不全为零，波形发生器的输出比较功能就会重载 OC1x 的通用 I/O 口功能。但是 OC1x 引脚的方向仍旧受控于数据方向寄存器 (DDR)。从 OC1x 引脚输出有效信号之前必须通过数据方向寄存器的 DDR_OC1x 将此引脚设置为输出。一般情况下功能重载与波形发生器的工作模式无关，但也由一些例外，详见 Table 43、Table 44 与 Table 45 。

输出比较逻辑的设计允许 OC1x 在输出之前首先进行初始化。要注意某些 COM1x1:0 设置在某些特定的工作模式下是保留的，如 P97 “16 位定时器 / 计数器寄存器说明 ” 。

COM1x1:0 不影响输入捕捉单元。

比较输出模式和波形产生

波形发生器利用 COM1x1:0 的方法在普通模式、 CTC 模式和 PWM 模式下有所区别。对于所有的模式 ,设置 COM1x1:0 = 0 表明比较匹配发生时波形发生器不会操作 OC1x 寄存器。非 PWM 模式的比较输出请参见 P97Table 43 ；快速 PWM 的比较输出于 P97Table 44 ；相位修正 PWM 的比较输出于 P98Table 45。

改变 COM1x1:0 将影响写入数据后的第一次比较匹配。对于非 PWM 模式，可以通过使用 FOC1x 来立即产生效果。

工作模式

工作模式 - T/C 和输出比较引脚的行为 - 由波形发生模式 (WGM13:0) 及比较输出模式 (COM1x1:0) 的控制位决定。比较输出模式对计数序列没有影响，而波形产生模式对计数序列则有影响。 COM1x1:0 控制 PWM 输出是否为反极性。非 PWM 模式时 COM1x1:0 控制输出是否应该在比较匹配发生时置位、清零，或是电平取反 (P86 “ 比较匹配输出单元 ”)。

具体的时序信息请参考 P95“ 定时器 / 计数器时序图 ”。

普通模式

普通模式 (WGM13:0 = 0) 为最简单的工作模式。在此模式下计数器不停地累加。计到最大值后 (MAX = 0xFFFF) 由于数值溢出计数器简单地返回到最小值 0x0000 重新开始。在TCNT1为零的同一个定时器时钟里T/C溢出标志TOV1置位。此时TOV1有点象第17位 ,只是只能置位，不会清零。但由于定时器中断服务程序能够自动清零 TOV1，因此可以通过软件提高定时器的分辨率。在普通模式下没有什么需要特殊考虑的 ,用户可以随时写入新的计数器数值。

在普通模式下输入捕捉单元很容易使用。要注意的是外部事件的最大时间间隔不能超过计数器的分辨率。如果事件间隔太长 ,必须使用定时器溢出中断或预分频器来扩展输入捕捉单元的分辨率。

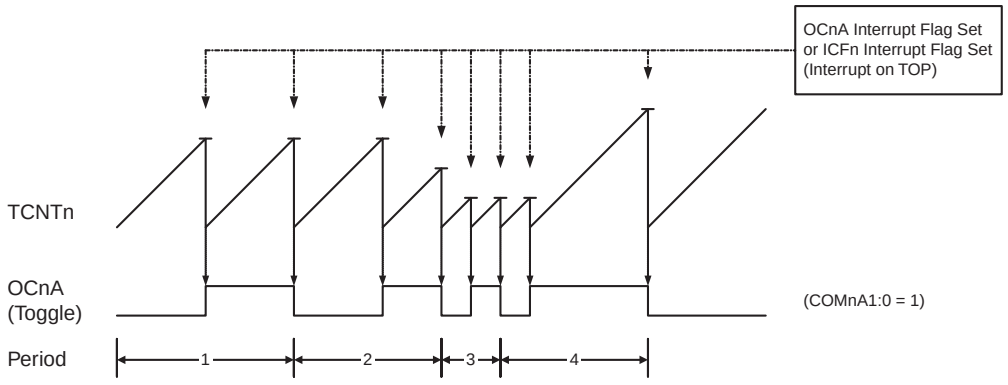
输出比较单元可以用来产生中断。但是不推荐在普通模式下利用输出比较来产生波形 ,因为会占用太多的 CPU 时间。

CTC(比较匹配时清除定时器) 模式

在 CTC 模式 (WGM13:0 = 4 或 12) 里 OCR1A 或 ICR1 寄存器用于调节计数器的分辨率。当计数器的数值 TCNT1 等于 OCR1A(WGM13:0 = 4) 或等于 ICR1 (WGM13:0 = 12) 时计数器清零。 OCR1A 或 ICR1 定义了计数器的 TOP 值，亦即计数器的分辨率。这个模式使得用户可以很容易地控制比较匹配输出的频率，也简化了外部事件计数的操作。

CTC 模式的时序图为 P87Figure 44。计数器数值 TCNT1 一直累加到 TCNT1 与 OCR1A 或 ICR1 匹配，然后 TCNT1 清零。

Figure 44. CTC 模式的时序图



利用 OCF1A 或 ICF1 标志可以在计数器数值达到 TOP 时产生中断。在中断服务程序里可以更新 TOP 的数值。由于 CTC 模式没有双缓冲功能 ,在计数器以无预分频器或很低的预分频器工作的时候将 TOP 更改为接近 BOTTOM 的数值时要小心。如果写入的 OCR1A 或 ICR1 的数值小于当前 TCNT1 的数值 ,计数器将丢失一次比较匹配。在下一次比较匹配发

生之前，计数器不得不先计数到最大值 0xFFFF，然后再从 0x0000 开始计数到 OCR1A 或 ICR1。在许多情况下，这一特性并非我们所希望的。替代的方法是使用快速 PWM 模式，该模式使用 OCR1A 定义 TOP 值 (WGM13:0 = 15)，因为此时 OCR1A 为双缓冲。

为了在CTC模式下得到波形输出，可以设置OC1A在每次比较匹配发生时改变逻辑电平。这可以通过设置 COM1A1:0 = 1 来完成。在期望获得 OC1A 输出之前，首先要将其端口设置为输出(DDR_OC1A = 1)。波形发生器能够产生的最大频率为 $f_{OC2} = f_{clk\_I/O}/2$ (OCR1A = 0x0000)。频率由如下公式确定：

$$f_{OCnA} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$$

变量 N 代表预分频因子 (1、 8、 64、 256 或 1024)。

在普通模式下，TOV1 标志的置位发生在计数器从 MAX 变为 0x0000 的定时器时钟周期。

快速 PWM 模式

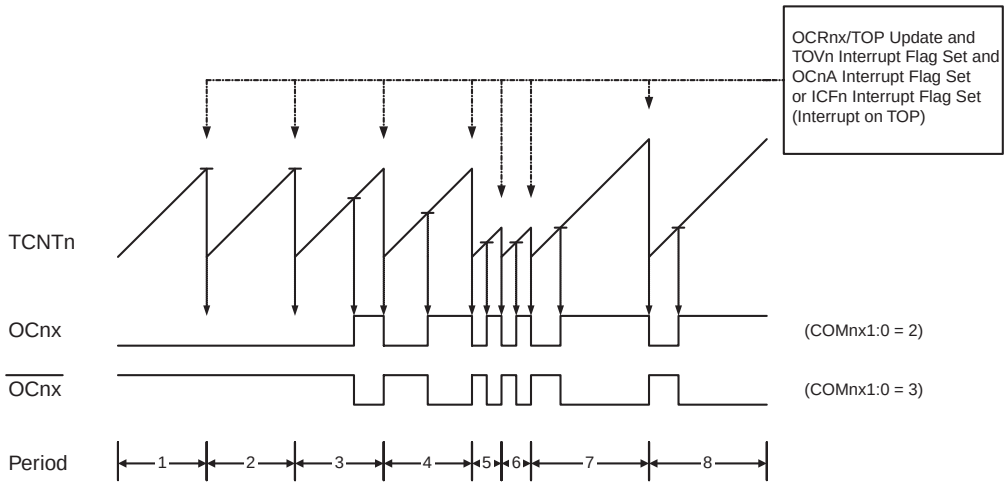
快速 PWM 模式 (WGM13:0 = 5、 6、7、 14 或 15) 可用来产生高频的 PWM 波形。快速 PWM 模式与其他 PWM 模式的不同之处是其单边斜坡工作方式。计数器从 BOTTOM 计到 TOP ,然后立即回到 BOTTOM 重新开始。对于普通的比较输出模式 ,输出比较引脚 OC1x 在 TCNT1 与 OCR1x 匹配时置位，在 TOP 时清零；对于反向比较输出模式， OCR1x 的动作正好相反。由于使用了单边斜坡模式 ,快速 PWM 模式的工作频率比使用双斜坡的相位修正 PWM 模式高一倍。此高频操作特性使得快速 PWM 模式十分适合于功率调节 ,整流和 DAC 应用。高频可以减小外部元器件 (电感 ,电容) 的物理尺寸 ,从而降低系统成本。

工作于快速 PWM 模式时 ,PWM 分辨率可固定为 8、9 或 10 位 ,也可由 ICR1 或 OCR1A 定义。最小分辨率为 2 比特 (ICR1 或 OCR1A 设为 0x0003)，最大分辨率为 16 位 (ICR1 或 OCR1A 设为 MAX)。 PWM 分辨率位数可用下式计算：

$$R_{FPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

工作于快速 PWM 模式时 ,计数器的数值一直累加到固定数值 0x00FF、0x01FF、0x03FF (WGM13:0 = 5、6 或 7)、ICR1 (WGM13:0 = 14) 或 OCR1A (WGM13:0 = 15) ,然后在后面的一个时钟周期清零。具体的时序图为 Figure 45。图中给出了当使用 OCR1A 或 ICR1 来定义 TOP 值时的快速 PWM 模式。图中柱状的 TCNT1 表示这是单边斜坡操作。方框图同时包含了普通的 PWM 输出以及反向 PWM 输出。 TCNT1 斜坡上的短水平线表示 OCR1x 和 TCNT1 的匹配比较。比较匹配后 OC1x 中断标志置位。

Figure 45. 快速 PWM 模式时序图



计时器数值达到 TOP 时 T/C 溢出标志 TOV1 置位。另外若 TOP 值是由 OCR1A 或 ICR1 定义的 ,则 OC1A 或 ICF1 标志将与 TOV1 在同一个时钟周期置位。如果中断使能，可以在中断服务程序里来更新 TOP 以及比较数据。

改变TOP值时必须保证新的TOP值不小于所有比较寄存器的数值。否则TCNT1与OCR1x不会出现比较匹配。使用固定的 TOP 值时，向任意 OCR1x 寄存器写入数据时未使用的位将屏蔽为 "0”。

定义 TOP 值时更新 ICR1 与 OCR1A 的步骤时不同的。ICR1 寄存器不是双缓冲寄存器。这意味着当计数器以无预分频器或很低的预分频工作的时候，给 ICR1 赋予一个小的数值时存在着新写入的 ICR1 数值比 TCNT1 当前值小的危险。结果是计数器将丢失一次比较匹配。在下一次比较匹配发生之前，计数器不得不先计数到最大值 0xFFFF，然后再从 0x0000 开始计数 ,直到比较匹配出现。而 OCR1A 寄存器则是双缓冲寄存器。这一特性决定 OCR1A 可以随时写入。写入的数据被放入 OCR1A 缓冲寄存器。在 TCNT1 与 TOP 匹配后的下一个时钟周期，OCR1A 比较寄存器的内容被缓冲寄存器的数据所更新。在同一个时钟周期 TCNT1 被清零，而 TOV1 标志被设置。

使用固定 TOP 值时最好使用 ICR1 寄存器定义 TOP。这样 OCR1A 就可以用于在 OC1A 输出 PWM 波。但是，如果 PWM 基频不断变化 (通过改变 TOP 值)， OCR1A 的双缓冲特性使其更适合于这个应用。

工作于快速PWM模式时 ,比较单元可以在OC1x引脚上输出PWM波形。设置COM1x1:0为 2 可以产生普通的 PWM 信号；为 3 则可以产生反向 PWM 波形 (参见 P97Table 43)。此外，要真正从物理引脚上输出信号还必须将 OC1x 的数据方向 DDR_OC1x 设置为输出。产生 PWM 波形的机理是 OC1x 寄存器在 OCR1x 与 TCNT1 匹配时置位 (或清零)，以及在计数器清零 (从 TOP 变为 BOTTOM) 的那一个定时器时钟周期清零 (或置位)。

输出的 PWM 频率可以通过如下公式计算得到：

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot (1 + TOP)}$$

变量 N 代表分频因子 (1、8、64、 256 或 1024)。

OCR1x 寄存器为极限值时说明了快速 PWM 模式的一些特殊情况。若 OCR1x 等于 BOTTOM(0x0000) ,输出为出现在第TOP+1个定时器时钟周期的窄脉冲；OCR1x为TOP时，根据 COM1x1:0 的设定，输出恒为高电平或低电平。

通过设定 OC1A 在比较匹配时进行逻辑电平取反 (COM1A1:0 = 1)，可以得到占空比为 50% 的周期信号。这只适用于 OCR1A 用来定义 TOP 值的情况 (WGM13:0 = 15)。OCR1A 为 0(0x0000) 时信号有最高频率 $f_{oc2} = f_{clk\_I/O}/2$。这个特性类似于 CTC 模式下的 OC1A 取反操作，不同之处在于快速 PWM 模式具有双缓冲。

相位修正 PWM 模式

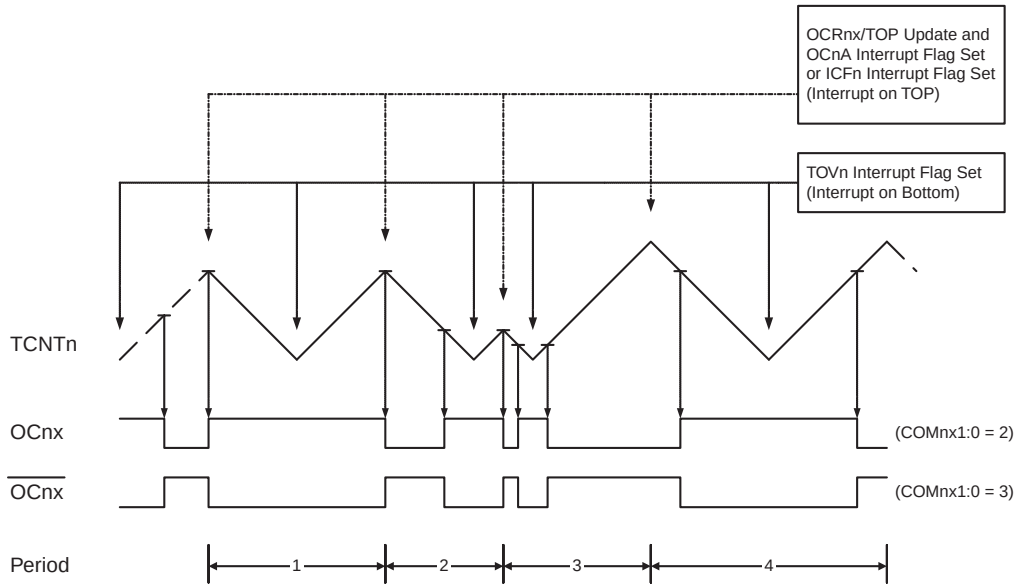
相位修正 PWM 模式 (WGM13:0 = 1、2、3、10 或 11) 为用户提供了一个获得高精度的、相位准确的 PWM 波形的方法。与相位和频率修正模式类似，此模式基于双斜坡操作。计时器重复地从 BOTTOM 计到 TOP，然后又从 TOP 倒退回到 BOTTOM。在一般的比较输出模式下，当计时器往 TOP 计数时若 TCNT1 与 OCR1x 匹配，OC1x 将清零为低电平；而在计时器往 BOTTOM 计数时若 TCNT1 与 OCR1x 匹配， OC1x 将置位为高电平。工作于反向比较输出时则正好相反。与单斜坡操作相比，双斜坡操作可获得的最大频率要小。但其对称特性十分适合于电机控制。

相位修正 PWM 模式的 PWM 分辨率固定为 8、 9 或 10 位，或由 ICR1 或 OCR1A 定义。最小分辨率为 2 比特 (ICR1 或 OCR1A 设为 0x0003)，最大分辨率为 16 位 (ICR1 或 OCR1A 设为 MAX)。 PWM 分辨率位数可用下式计算：

$$R_{PCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

工作于相位修正 PWM 模式时，计数器的数值一直累加到固定值 0x00FF、 0x01FF、0x03FF (WGM13:0 = 1、2 或 3)、ICR1 (WGM13:0 = 10) 或 OCR1A (WGM13:0 = 11)，然后改变计数方向。在一个定时器时钟里 TCNT1 值等于 TOP 值。具体的时序图为 Figure 46。图中给出了当使用 OCR1A 或 ICR1 来定义 TOP 值时的相位修正 PWM 模式。图中柱状的 TCNT1 表示这是双边斜坡操作。方框图同时包含了普通的 PWM 输出以及反向 PWM 输出。TCNT1 斜坡上的短水平线表示 OCR1x 和 TCNT1 的匹配比较。比较匹配后 OC1x 中断标志置位。

Figure 46. 相位修正 PWM 模式的时序图



计时器数值达到 BOTTOM 时 T/C 溢出标志 TOV1 置位。若 TOP 由 OCR1A 或 ICR1 定义，在 OCR1x 寄存器通过双缓冲方式得到更新的同一个时钟周期里 OC1A 或 ICF1 标志置位。标志置位后即可产生中断。

改变 TOP 值时必须保证新的 TOP 值不小于所有比较寄存器的数值。否则 TCNT1 与 OCR1x 不会出现比较匹配。使用固定的 TOP 值时，向任意 OCR1x 寄存器写入数据时未使用的位将屏蔽为 "0"。在 Figure 46 给出的第三个周期中，在 T/C 运行于相位修正模式时改变 TOP 值导致了不对称输出。其原因在于 OCR1x 寄存器的更新时间。由于 OCR1x 的更新时刻为定时器 / 计数器达到 TOP 之时，因此 PWM 的循环周期起始于此，也终止于此。就是说，下降斜坡的长度取决于上一个 TOP 值，而上升斜坡的长度取决于新的 TOP 值。若这两个值不同，一个周期内两个斜坡长度不同，输出也就不对称了。

若要在 T/C 运行时改变 TOP 值，最好用相位与频率修正模式代替相位修正模式。若 TOP 保持不变，那么这两种工作模式实际没有区别。

工作于相位修正 PWM 模式时，比较单元可以在 OC1x 引脚输出 PWM 波形。设置 COM1x1:0 为 2 可以产生普通的 PWM，设置 COM1x1:0 为 3 可以产生反向 PWM (参见 P97Table 44)。要真正从物理引脚上输出信号还必须将OC1x的数据方向DDR_OC1x设置为输出。OCR1x 和 TCNT1 比较匹配发生时 OC1x 寄存器将产生相应的清零或置位操作，从而产生 PWM 波形。工作于相位修正模式时 PWM 频率可由如下公式获得：

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP}$$

变量 N 表示预分频因子 (1、8、64、256 或 1024)。

OCR1x 寄存器处于极值时表明了相位修正 PWM 模式的一些特殊情况。在普通 PWM 模式下，若 OCR1x 等于 BOTTOM，输出一直保持为低电平；若 OCR1x 等于 TOP，输出则保持为高电平。反向 PWM 模式正好相反。

相位和频率修正 PWM 模式

相位与频率修正 PWM 模式 (WGM13:0 = 8 或 9) - 以下简称相频修正 PWM 模式 - 可以产生高精度的、相位与频率都准确的 PWM 波形。与相位修正模式类似，相频修正 PWM 模式基于双斜坡操作。计时器重复地从 BOTTOM 计到 TOP，然后又从 TOP 倒退回到 BOTTOM。在一般的比较输出模式下，当计时器往TOP计数时若TCNT1与OCR1x匹配，OC1x将清零为低电平；而在计时器往BOTTOM计数时TCNT1与OCR1x匹配，OC1x将置位为高电平。工作于反向输出比较时则正好相反。与单斜坡操作相比，双斜坡操作可获得的最大频率要小。但其对称特性十分适合于电机控制。

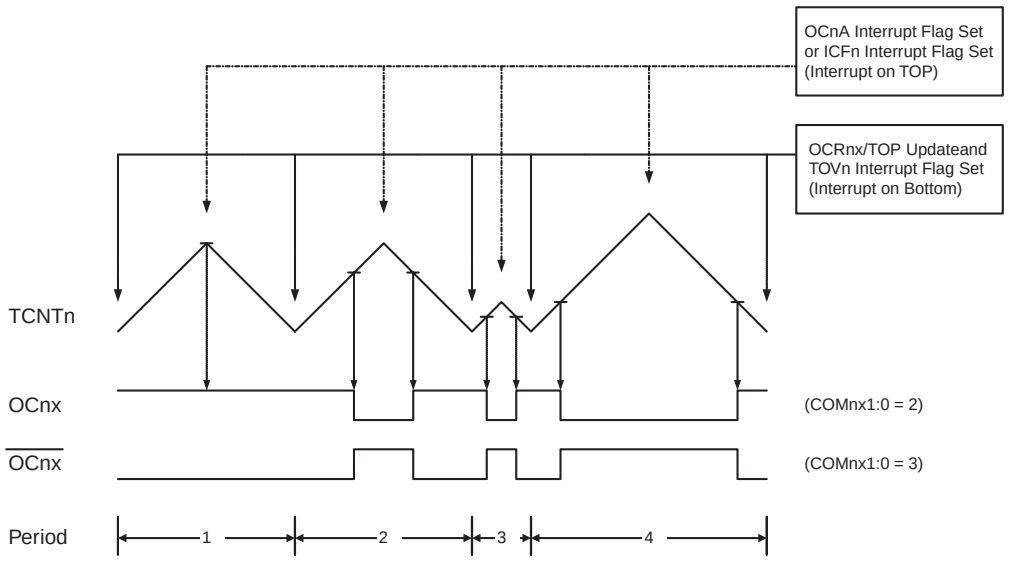
相频修正修正 PWM 模式与相位修正 PWM 模式的主要区别在于 OCR1x 寄存器的更新时间，详见 Figure 46 与 Figure 47。

相频修正修正 PWM 模式的 PWM 分辨率可由 ICR1 或 OCR1A 定义。最小分辨率为 2 比特 (ICR1 或 OCR1A 设为 0x0003)，最大分辨率为 16 位 (ICR1 或 OCR1A 设为 MAX)。PWM 分辨率位数可用下式计算：

$$R_{PFCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

工作于相频修正PWM模式时，计数器的数值一直累加到ICR1 (WGM13:0 = 8)或OCR1A (WGM13:0 = 9)，然后改变计数方向。在一个定时器时钟里TCNT1值等于TOP值。具体的时序图为 Figure 47。图中给出了当使用 OCR1A 或 ICR1 来定义 TOP 值时的相频修正PWM模式。图中柱状的TCNT1表示这是双边斜坡操作。方框图同时包含了普通的PWM输出以及反向 PWM 输出。TCNT1 斜坡上的短水平线表示 OCR1x 和 TCNT1 的匹配比较。比较匹配发生时， OC1x 中断标志将被置位。

Figure 47. 相位与频率修正 PWM 模式的时序图



在 OCR1x 寄存器通过双缓冲方式得到更新的同一个时钟周期里 T/C 溢出标志 TOV1 置位。若 TOP 由 OCR1A 或 ICR1 定义，则当 TCNT1 达到 TOP 值时 OC1A 或 ICF1 置位。这些中断标志位可用来在每次计数器达到 TOP 或 BOTTOM 时产生中断。

改变TOP值时必须保证新的TOP值不小于所有比较寄存器的数值。否则TCNT1与OCR1x不会产生比较匹配。

如 Figure 47 所示，与相位修正模式形成对照的是，相频修正 PWM 模式生成的输出在所有的周期中均为对称信号。这是由于 OCR1x 在 BOTTOM 得到更新，上升与下降斜坡长度始终相等。因此输出脉冲为对称的，确保了频率是正确的。

使用固定 TOP 值时最好使用 ICR1 寄存器定义 TOP。这样 OCR1A 就可以用于在 OC1A 输出 PWM 波。但是，如果 PWM 基频不断变化 (通过改变 TOP 值)， OCR1A 的双缓冲特性使其更适合于这个应用。

工作于相频修正 PWM 模式时，比较单元可以在 OC1x 引脚上输出 PWM 波形。设置 COM1x1:0 为 2 可以产生普通的 PWM 信号；为 3 则可以产生反向 PWM 波形。(参见 P98Table 45)。要想真正输出信号还必须将OC1x的数据方向设置为输出。产生PWM波形的机理是 OC1x 寄存器在 OCR1x 与升序记数的 TCNT1 匹配时置位 (或清零) ,与降序记数的 TCNT1 匹配时清零 (或置位)。输出的 PWM 频率可以通过如下公式计算得到：

$$f_{OCnxPFCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP}$$

变量 N 代表分频因子 (1、8、64、 256 或 1024)。

OCR1x 寄存器处于极值时说明了相频修正 PWM 模式的一些特殊情况。在普通 PWM 模式下，若 OCR1x 等于 BOTTOM，输出一直保持为低电平；若 OCR1x 等于 TOP，则输出保持为高电平。反向 PWM 模式则正好相反。

定时器 / 计数器时序图

定时器 / 计数器为同步电路，因而时钟 clk_{T1} 表示为时钟使能信号。图中说明了何时设置中断标志及何时使用 OCR1x 缓冲器中的数据更新 OCR1x 寄存器（工作于双缓冲器模式时）。Figure 48 给出了置位 OCF1x 的时序图。

Figure 48. T/C 时序图 ， OCF1x 置位，无预分频器

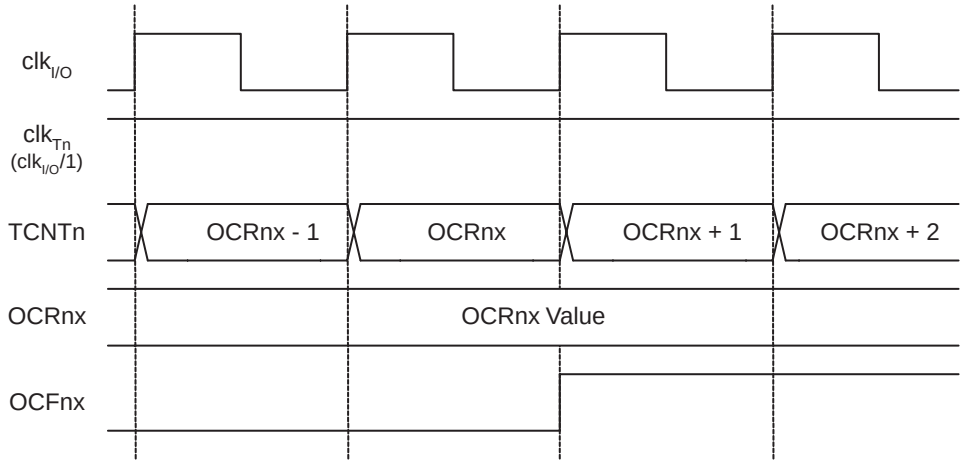


Figure 49 给出相同的时钟数据，但预分频使能。

Figure 49. T/C 时序图，置位 OCF1x，预分频器为 $f_{clk\_I/O}/8$

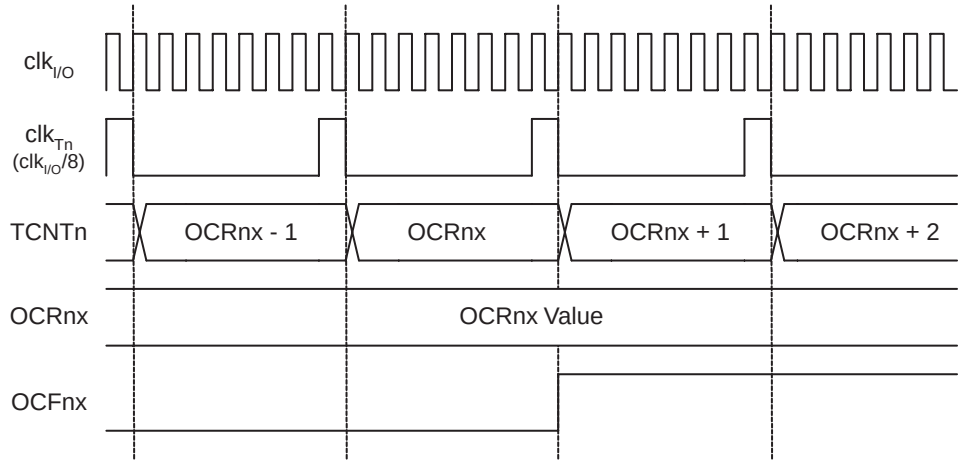


Figure 50 给出工作在不同模式下接近 TOP 值时的计数序列。工作于相频修正 PWM 模式时， OCR1x 寄存器在 BOTTOM 被更新。时序图相同，但 TOP 需要用 BOTTOM 代替，BOTTOM+1 代替 TOP-1 ,等等。同样的命名规则也适用于那些在 BOTTOM 置位 TOV1 标志的工作模式。

Figure 50. T/C 时序图，无预分频器

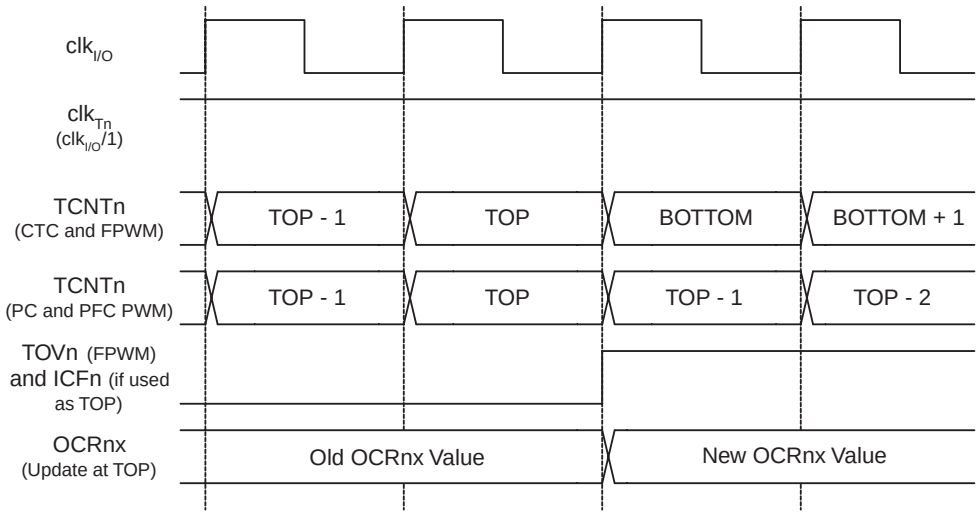
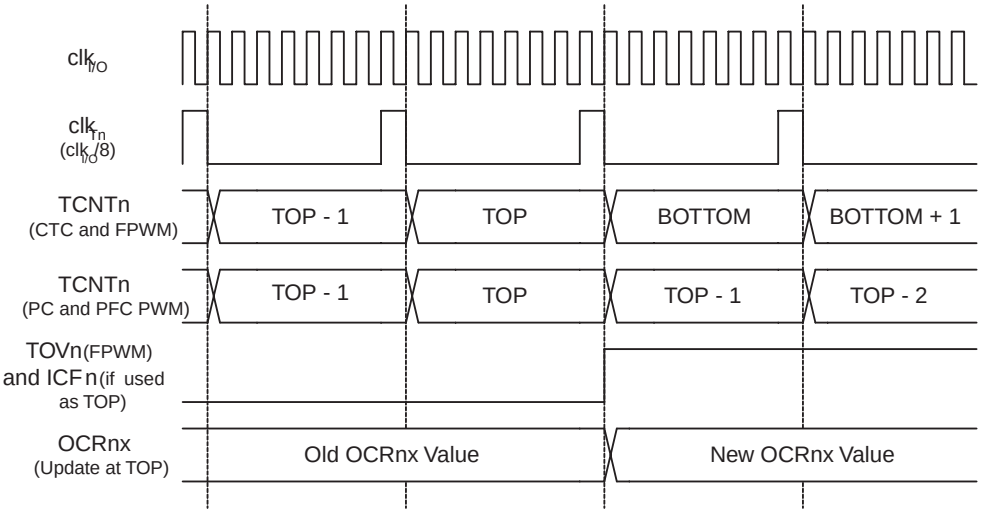


Figure 51 给出相同的时钟数据，但预分频使能。

Figure 51. T/C 时序图，预分频器为 $f_{clk\_I/O}/8$



16 位定时器 / 计数器寄存器说明

T/C1 控制寄存器 A - TCCR1A

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
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	

- **Bit 7:6 – COM1A1:0: 通道 A 的比较输出模式**
- **Bit 5:4 – COM1B1:0: 通道 B 的比较输出模式**

COM1A1:0、COM1B1:0 分别控制 OC1A 、OC1B 的状态。如果 COM1A1:0 的一位或两位被写入 "1”，OC1A 输出功能将取代 I/O 端口功能。如果 COM1B1:0 的一位或两位被写入 "1”,OC1B输出功能将取代I/O端口功能。此时OC1A或OC1B相应的输出引脚数据方向控制必须置位以使能输出驱动器。

OC1A(或 OC1B) 与物理引脚相连时 ,COM1x1:0 的功能由 WGM13:0 的设置决定。Table 43 给出当 WGM13:0 设置为普通模式与 CTC 模式 (非 PWM) 时 COM1x1:0 的功能定义。

Table 43. 比较输出模式，非 PWM

COM1A1/COM1B1	COM1A0/COM1B0	说明
0	0	普通端口操作，不与 OC1A/OC1B 连接
0	1	比较匹配时 OC1A/OC1B 电平取反
1	0	比较匹配时清零 OC1A/OC1B(输出低电平)
1	1	比较匹配时置位 OC1A/OC1B(输出高电平)

Table 44 给出 WGM13:0 设置为快速 PWM 模式时 COM1x1:0 的功能定义。

Table 44. 比较输出模式，快速 PWM(1)

COM1A1/COM1B1	COM1A0/COM1B0	说明
0	0	普通端口操作，不与 OC1A/OC1B 连接
0	1	WGM13=0: 普通端口操作，不与 OC1A/OC1B 连接 WGM13=1: 比较匹配时 OC1A 电平取反，OC1B 保留
1	0	比较匹配时清零 OC1A/OC1B，在 TOP 时置位 OC1A/OC1B
1	1	比较匹配时置位 OC1A/OC1B，在 TOP 时清零 OC1A/OC1B

Note: 1. 当 OCR1A/OCR1B 等于 TOP 且 COM1A1/COM1B1 置位时，比较匹配被忽略，但 OC1A/OC1B 的置位 / 清零操作有效。详见 P89 “ 快速 PWM 模式 ” 。

Table 45给出当WGM13:0设置为相位修正PWM模式或相频修正PWM模式时COM1x1:0的功能定义。

Table 45. 比较输出模式，相位修正及相频修正 PWM 模式 (1)

COM1A1/COM1B1	COM1A0/COM1B0	说明
0	0	普通端口操作，不与 OC1A/OC1B 连接
0	1	WGM13=0: 普通端口操作，不与 OC1A/OC1B 连接 WGM13=1: 比较匹配时 OC1A 电平取反，OC1B/OCnC 保留
1	0	升序记数时比较匹配将清零 OC1A/OC1B ，降序记数时比较匹配将置位 OC1A/OC1B
1	1	升序记数时比较匹配将置位 OC1A/OC1B，降序记数时比较匹配将清零 OC1A/OC1B

Note: 1. OCR1A/OCR1B 等于 TOP 且 COM1A1/COM1B1 置位是一个特殊情况。详细信息请参见 P91 “ 相位修正 PWM 模式 ” 。

- **Bit 1:0 – WGM11:0: 波形发生模式**

这两位与位于 TCCR1B 寄存器的 WGM13:2 相结合，用于控制计数器的计数序列——计数器计数的上限值和确定波形发生器的工作模式，见 Table 46。T/C 支持的工作模式有：普通模式（计数器），比较匹配时清零定时器 (CTC) 模式，及三种脉宽调制 (PWM) 模式 (P87 “ 工作模式 ”)。

Table 46. 波形产生模式的位描述 (1)

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	定时器 / 计数器工作模式	TOP	OCR1x 更新时刻	TOV1 置位时刻
0	0	0	0	0	普通模式	0xFFFF	立即更新	MAX
1	0	0	0	1	8 位相位修正 PWM	0x00FF	TOP	BOTTOM
2	0	0	1	0	9 位相位修正 PWM	0x01FF	TOP	BOTTOM
3	0	0	1	1	10 位相位修正 PWM	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	立即更新	MAX
5	0	1	0	1	8 位快速 PWM	0x00FF	TOP	TOP
6	0	1	1	0	9 位快速 PWM	0x01FF	TOP	TOP
7	0	1	1	1	10 位快速 PWM	0x03FF	TOP	TOP
8	1	0	0	0	相位与频率修正 PWM	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	相位与频率修正 PWM	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	相位修正 PWM	ICR1	TOP	BOTTOM
11	1	0	1	1	相位修正 PWM	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	立即更新	MAX
13	1	1	0	1	保留	–	–	–
14	1	1	1	0	快速 PWM	ICR1	TOP	TOP
15	1	1	1	1	快速 PWM	OCR1A	TOP	TOP

Note: 1. CTC1 和 PWM11:0 的定义已经不再使用了，要使用 WGM12:0。但是两个版本的功能和位置是兼容的。

T/C1 控制寄存器 B - TCCR1B

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
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	

- **Bit 7 – ICNC1: 输入捕捉噪声抑制器**

置位 ICNC1 将使能输入捕捉噪声抑制功能。此时外部引脚 ICP1 的输入被滤波。其作用是从 ICP1 引脚连续进行 4 次采样。如果 4 个采样值都相等，那么信号送入边沿检测器。因此使能该功能使得输入捕捉被延迟了 4 个时钟周期。

- **Bit 6 – ICES1: 输入捕捉触发沿选择**

该位选择使用 ICP1 上的哪个边沿触发捕获事件。ICES1 为 "0" 选择的是下降沿触发输入捕捉； ICES1 为 "1" 选择的是逻辑电平的上升沿触发输入捕捉。

按照 ICES1 的设置捕获到一个事件后 ,计数器的数值被复制到 ICR1 寄存器。捕获事件还会置为 ICF1。如果此时中断使能，输入捕捉事件即被触发。

当 ICR1 用作 TOP 值 (见 TCCR1A 与 TCCR1B 寄存器中 WGM13:0 位的描述) 时 ,ICP1 与输入捕捉功能脱开，从而输入捕捉功能被禁用。

- **Bit 5 – 保留**

该位保留。为保证与将来器件的兼容性，写 TCCR1B 时，该位必须写入 "0"。

- **Bit 4:3 – WGM13:2: 波形发生模式**

见 TCCR1A 寄存器中的描述。

- **Bit 2:0 – CS12:0: 时钟选择**

这 3 位用于选择 T/C 的时钟源，见 Figure 48 与 Figure 49。

Table 47. 时钟选择位描述

CS12	CS11	CS10	说明
0	0	0	无时钟源 (T/C 停止)
0	0	1	$clk_{I/O}$/1 (无预分频)
0	1	0	$clk_{I/O}$/8 (来自预分频器)
0	1	1	$clk_{I/O}$/64 (来自预分频器)
1	0	0	$clk_{I/O}$/256 (来自预分频器)
1	0	1	$clk_{I/O}$/1024 (来自预分频器)
1	1	0	外部 T1 引脚，下降沿驱动
1	1	1	外部 T1 引脚，上升沿驱动

选择使用外部时钟源后，即使 T1 引脚被定义为输出，其 1 引脚上的逻辑信号电平变化仍然会驱动 T/C1 计数，这个特性允许用户通过软件来控制计数。

T/C1 控制寄存器 C - TCCR1C

Bit	7	6	5	4	3	2	1	0	
	FOC1A	FOC1B	–	–	–	–	–	–	TCCR1C
读 / 写	W	W	R	R	R	R	R	R	
初始值	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC1A: 通道 A 强制输出比较**
- **Bit 6 – FOC1B: 通道 B 强制输出比较**

FOC1A/FOC1B位只在WGM13:0位被设置为非PWM模式时才有效。为与将来器件兼容，当在 PWM 模式下对 TCCR1A 进行写操作时，该位必须置 1。对 FOC1A/FOC1B 写 "1" 将强制波形发生器产生一次成功的比较匹配，并使波形发生器依据 COM1x1:0 的设置而改变 OC1A/OC1B 的输出状态。 FOC1A/FOC1B 的作用如同一个选通信号， COM1x1:0 的设置才是最终确定比较匹配结果的因素。

FOC1A/FOC1B 选通信号不会产生任何中断请求，也不会对计数器清零，象使用 OCR1A 为 TOP 值的 CTC 工作模式那样。

FOC1A/FOC1B 的读返回值总为零。

T/C1 - TCNT1H 与 TCNT1L

Bit	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								TCNT1H
	TCNT1[7:0]								TCNT1L
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	

TCNT1H与TCNT1L组成了T/C1的数据寄存器TCNT1。通过它们可以直接对定时器/计数器单元的 16 位计数器进行读写访问。为保证 CPU 对高字节与低字节的同时读写，必须使用一个 8 位临时高字节寄存器 TEMP。TEMP 是所有的 16 位寄存器共用的，详见 P78 “ 访问 16 位寄存器 ”。

在计数器运行期间修改TCNT1的内容有可能丢失一次TCNT1与OCR1x的比较匹配操作。

写 TCNT1 寄存器将在下一个定时器周期阻塞比较匹配。

输出比较寄存器 1A - OCR1AH 与 OCR1AL

Bit	7	6	5	4	3	2	1	0	
	OCR1A[15:8]								OCR1AH
	OCR1A[7:0]								OCR1AL
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	

输出比较寄存器 1B - OCR1BH 与 OCR1BL

Bit	7	6	5	4	3	2	1	0	
	OCR1B[15:8]								OCR1BH
	OCR1B[7:0]								OCR1BL
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	

该寄存器中的 16 位数据与 TCNT1 寄存器中的计数值进行连续的比较，一旦数据匹配，将产生一个输出比较中断，或改变 OC1x 的输出逻辑电平。

输出比较寄存器长度为 16 位。为保证 CPU 对高字节与低字节的同时读写，必须使用一个 8 位临时高字节寄存器 TEMP。TEMP 是所有的 16 位寄存器共用的，详见 P78 “ 访问 16 位寄存器 ” 。

输入捕捉寄存器 1 - ICR1H 与 ICR1L

Bit	7	6	5	4	3	2	1	0	
	ICR1[15:8]								ICR1H
	ICR1[7:0]								ICR1L
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	

当外部引脚 ICP1(或 T/C1 的模拟比较器) 有输入捕捉触发信号产生时，计数器 TCNT1 中的值写入 ICR1 中。 ICR1 的设定值可作为计数器的 TOP 值。

输入捕捉寄存器长度为 16 位。为保证 CPU 对高字节与低字节的同时读写，必须使用一个 8 位临时高字节寄存器 TEMP。TEMP 是所有的 16 位寄存器共用的，详见 P78 “ 访问 16 位寄存器 ” 。

定时器 / 计数器中断屏蔽寄存器 - TIMSK

Bit	7	6	5	4	3	2	1	0	
	TOIE1	OCIE1A	OCIE1B	–	ICIE1	OCIE0B	TOIE0	OCIE0A	TIMSK
读 / 写	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

- **Bit 7 – TOIE1:T/C1 溢出中断使能**

当该位被设为 "1”，且状态寄存器中的 I 位被设为 ”1” 时， T/C1 的溢出中断使能。一旦 TIFR 上的 TOV1 置位， CPU 即开始执行 T/C1 溢出中断服务程序 (P42 “ 中断 ”)。

- **Bit 6 – OCIE1A: T/C1 输出比较 A 匹配中断使能**

当该位被设为 "1”，且状态寄存器中的 I 位被设为 "1” 时，使能 T/C1 的输出比较 A 匹配中断使能。一旦 TIFR1 上的 OCF1A 置位， CPU 即开始执行 T/C1 输出比较 A 匹配中断服务程序 (P42 “ 中断 ”)。

- **Bit 5 – OCIE1B: T/C1 输出比较 B 匹配中断使能**

当该位被设为 "1”，且状态寄存器中的 I 位被设为 "1” 时，使能 T/C1 的输出比较 B 匹配中断使能。一旦 TIFR1 上的 OCF1B 置位， CPU 即开始执行 T/C1 输出比较 B 匹配中断服务程序 (P42 “ 中断 ”)。

- **Bit 3 – ICIE1: T/C1 输入捕捉中断使能**

当该位被设为 "1”，且状态寄存器中的 I 位被设为 "1” 时， T/C1 的输入捕捉中断使能。一旦 TIFR1 的 ICF1 置位， CPU 即开始执行 T/C1 输入捕捉中断服务程序 (P42 “ 中断 ”) 。

T/C 中断标志寄存器 - TIFR

Bit	7	6	5	4	3	2	1	0	
	TOV1	OCF1A	OCF1B	–	ICF1	OCF0B	TOV0	OCF0A	TIFR
读 / 写	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

- **Bit 7 – TOV1: T/C1 溢出标志**

该位的设置与 T/C1 的工作方式有关。工作于普通模式和 CTC 模式时 ,T/C1 溢出时 TOV1 置位。对工作在其它模式下的 TOV1 标志位置位，见 P99Table 46。

执行溢出中断服务程序时 TOV1 自动清零。也可以对其写入逻辑 "1” 来清除该标志位。

- **Bit 6 – OCF1A: T/C1 输出比较 A 匹配标志位**

当 TCNT1 与 OCR1A 匹配成功时，该位被设为 "1”。

强制输出比较 (FOC1A) 不会置位 OCF1A。

执行强制输出比较匹配 A 中断服务程序时 OCF1A 自动清零。也可以对其写入逻辑 "1” 来清除该标志位。

- **Bit 5 – OCF1B: T/C1 输出比较 B 匹配标志位**

当 TCNT1 与 OCR1B 匹配成功时，该位被设为 "1”。

强制输出比较 (FOC1B) 不会置位 OCF1B。

执行强制输出比较匹配 B 中断服务程序时 OCF1B 自动清零。也可以对其写入逻辑 "1” 来清除该标志位。

- **Bit 3 – ICF1: T/C1 输入捕捉标志位**

外部引脚 ICP1 出现捕捉事件时 ICF1 置位。此外，当 ICR1 作为计数器的 TOP 值时，一旦计数器值达到 TOP， ICF1 也置位。

执行输入捕捉中断服务程序时 ICF1 自动清零。也可以对其写入逻辑 "1” 来清除该标志位。

USART

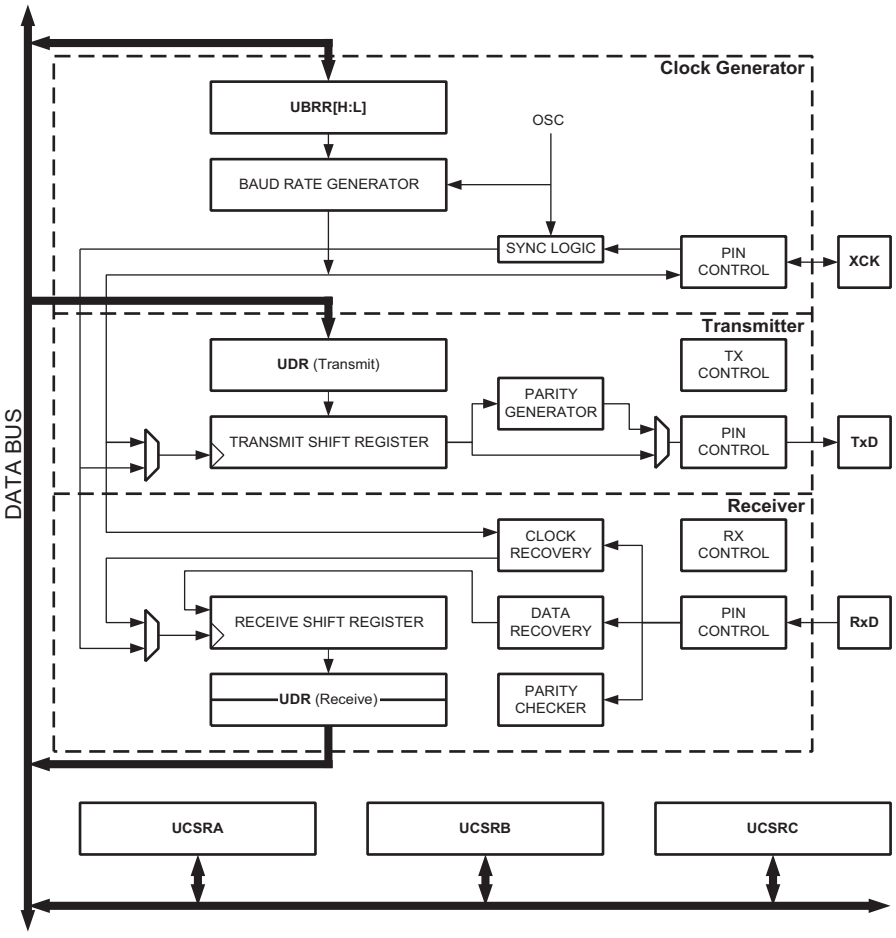
通用同步和异步串行接收器和转发器 (USART) 是一个高度灵活的串行通讯设备。主要特点为：

- **全双工操作 (独立的串行接收和发送寄存器)**
- **异步或同步操作**
- **主机或从机提供时钟的同步操作**
- **高精度的波特率发生器**
- **支持 5, 6, 7, 8, 或 9 个数据位和 1 个或 2 个停止位**
- **硬件支持的奇偶校验操作**
- **数据过速检测**
- **帧错误检测**
- **噪声滤波，包括错误的起始位检测，以及数字低通滤波器**
- **三个独立的中断：发送结束中断，发送数据寄存器空中断，以及接收结束中断**
- **多处理器通讯模式**
- **倍速异步通讯模式**

综述

Figure 52 为简化的 USART 转发器。CPU 可以访问的 I/O 寄存器和 I/O 引脚以粗体表示。

Figure 52. USART 方框图 (1)



Note: 1. 请参考 P2Figure 1，P55Table 29 和 P53Table 26 了解 USART 的引脚分布。

虚线框将 USART 分为了三个主要部分：时钟发生器，发送器和接收器。控制寄存器由三个单元共享。时钟发生器包括同步从机操作用来与外部输入时钟进行同步的逻辑，以及波特率发生器。XCK (发送器时钟) 引脚用于同步发送模式。发送器包括单个写缓冲器，串行移位寄存器，奇偶发生器以及处理不同的帧格式所需的控制逻辑。写缓冲器可以保持连续发送数据而不会在数据帧之间引入延迟。由于接收器具有时钟和数据恢复单元，它是 USART 模块中最复杂的。恢复单元用于异步数据的接收。除了恢复单元，接收器还包括奇偶校验，控制逻辑，移位寄存器和两个接收缓冲器 UDR。接收器支持与发送器相同的帧格式，而且可以检测帧错误，数据过速和奇偶校验错误。

AVR USART 和 AVR UART - 兼容性

USART 在如下方面与 AVR UART 完全兼容：

- 所有 USART 寄存器的位定义
- 波特率发生器
- 发送器操作
- 发送缓冲器的功能
- 接收器操作

然而，接收器缓冲器有两个方面的改进，在某些特殊情况下会影响兼容性：

- 增加了一个缓冲器。两个缓冲器的操作好象是一个循环的 FIFO。因此对于每个接收到的数据只能读一次！更重要的是错误标志 FE 和 DOR，以及第 9 个数据位 RXB8 与数据一起存放于接收缓冲器。因此必须在读取 UDR 寄存器之前访问状态标志位。否则将丢失错误状态。
- 现在接收移位寄存器可以作为第三级缓冲了。其意义是在两个缓冲器都没有空的时候，将数据保存于串行移位寄存器之中 (参见 Figure 52)，直到检测到新的起始位。从而增强了 USART 抵抗数据过速 (DOR) 的能力。

下面的控制位的名称做了改动，但是功能和在寄存器中的位置并没有改变 :

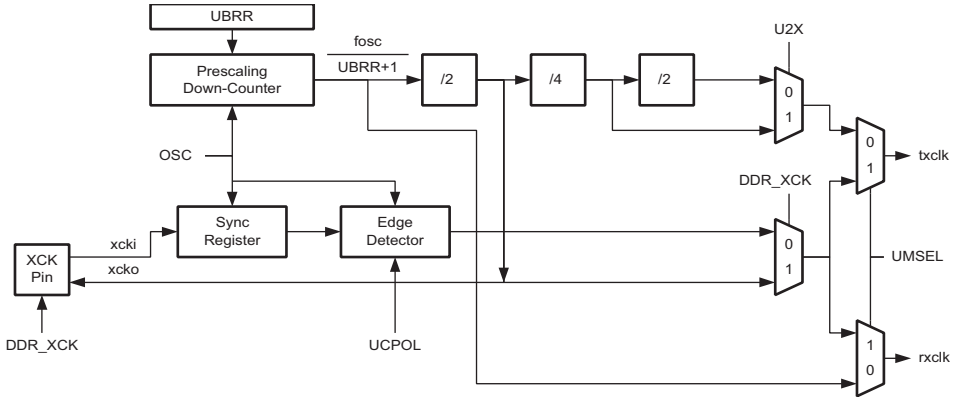
- CHR9 改变为 UCSZ2
- OR 改变为 DOR

时钟产生

时钟产生逻辑为发送器和接收器产生基础时钟。 USART 支持 4 种模式的时钟 : 正常的异步模式，倍速的异步模式，主机同步模式，以及从机同步模式。 USART 控制和状态寄存器 C (UCSRC) 用于选择异步模式和同步模式。倍速模式 (只适用于异步模式) 受控于 UCSRA 寄存器的 U2X。使用同步模式 (UMSEL = 1) 时，XCK 的数据方向寄存器 (DDR_XCK)决定时钟源是由内部产生(主机模式)还是由外部生产(从机模式)。仅在同步模式下 XCK 有效。

Figure 53 为时钟产生逻辑的框图。

Figure 53. 时钟产生逻辑框图



信号说明：

txclk 发送器时钟 (内部信号)

rxclk 接收器基础时钟 (内部信号)

xcki 由 XCK 引脚输入 (内部信号) , 用于同步从机操作

xcko 输出到 XCK 引脚的时钟 (内部信号). 用于同步主机操作

fosc XTAL 频率 (系统时钟)

片内时钟产生 - 波特率发生器

内部时钟用于异步模式与同步主机模式，请参见 Figure 53。

USART 的波特率寄存器 UBRR 和降序计数器相连接，一起构成可编程的预分频器或波特率发生器。降序计数器对系统时钟计数，当其计数到零或 UBRRL 寄存器被写时，会自动装入 UBRR 寄存器的值。当计数到零时产生一个时钟，该时钟作为波特率发生器的输出时钟，输出时钟的频率为 f_{osc}/(UBRR+1)。发生器对波特率发生器的输出时钟进行 2、 8 或 16 的分频，具体情况取决于工作模式。波特率发生器的输出被直接用于接收器与数据恢复单元。数据恢复单元使用了一个有2、8或16个状态的状态机，具体状态数由UMSEL、U2X 与 DDR_XCK 位设定的工作模式决定。

Table 48给出了计算波特率(位/秒)以及计算每一种使用内部时钟源工作模式的UBRR值的公式。

Table 48. 波特率计算公式

使用模式	波特率的计算公式 (1)	UBRR 值的计算公式
异步正常模式 (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(\boldsymbol{UBRR}+1)}$	$\boldsymbol{UBRR} = \frac{f_{OSC}}{16BAUD} - 1$
异步倍速模式 (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(\boldsymbol{UBRR}+1)}$	$\boldsymbol{UBRR} = \frac{f_{OSC}}{8BAUD} - 1$
同步主机模式	$BAUD = \frac{f_{OSC}}{2(\boldsymbol{UBRR}+1)}$	$\boldsymbol{UBRR} = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. 波特率定义为每秒的位传输速度 (bps)。

BAUD 波特率 (bps)

f_{OSC} 系统时钟频率

UBRR UBRRH 与 UBRRL 的数值 (0-4095)

Table 56 给出了在某些系统时钟频率下对应的 UBRR 数值。

倍速操作 (U2X)

通过设定 UCSRA 寄存器的 U2X 可以使传输速率加倍。该位只对异步工作模式有效。当工作在同步模式时，设置该位为 "0”。

设置该位把波特率分频器的分频值从 16 降到 8，使异步通信的传输速率加倍。此时接收器只使用一半的采样数对数据进行采样及时钟恢复，因此在该模式下需要更精确的系统时钟与更精确的波特率设置。发送器则没有这个要求。

外部时钟

同步从机操作模式由外部时钟驱动，如 Figure 53 所示。

输入到 XCK 引脚的外部时钟由同步寄存器进行采样，用以提高稳定性。同步寄存器的输出通过一个边沿检测器，然后应用于发送器与接收器。这一过程引入了两个 CPU 时钟周期的延时，因此外部 XCK 的最大时钟频率由以下公式限制：

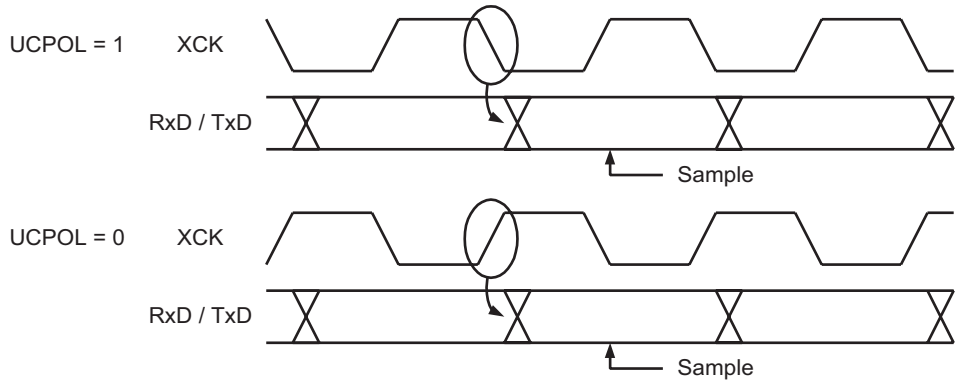
$$f_{XCK} < \frac{f_{OSC}}{4}$$

要注意 f_{osc} 由系统时钟的稳定性决定，为了防止因频率漂移而丢失数据，建议保留足够的裕量。

同步时钟操作

使用同步模式时 (UMSEL = 1)XCK 引脚被用于时钟输入 (从机模式) 或时钟输出 (主机模式)。时钟的边沿、数据的采样与数据的变化之间的关系的基本规律是：在改变数据输出端 TxD 的 XCK 时钟的相反边沿对数据输入端 RxD 进行采样。

Figure 54. 同步模式时的 XCK 时序 .



UCRSC 寄存器的 UCPOL 位确定使用 XCK 时钟的哪个边沿对数据进行采样和改变输出数据。如 Figure 54 所示，当 UCPOL=0 时，在 XCK 的上升沿改变输出数据，在 XCK 的下降沿进行数据采样；当 UCPOL=1 时 ,在 XCK 的下降沿改变输出数据 ,在 XCK 的上升沿进行数据采样。

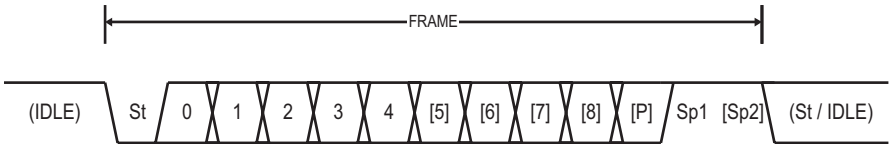
帧格式

串行数据帧由数据字加上同步位 (开始位与停止位) 以及用于纠错的奇偶校验位构成。USART 接受以下 30 种组合的数据帧格式：

- 1 个起始位
- 5、6、7、8 或 9 个数据位
- 无校验位、奇校验或偶校验位
- 1 或 2 个停止位

数据帧以起始位开始；紧接着是数据字的最低位 , 数据字最多可以有 9 个数据位 , 以数据的最高位结束。如果使能了校验位，校验位将紧接着数据位，最后是结束位。当一个完整的数据帧传输后，可以立即传输下一个新的数据帧，或使传输线处于空闲状态。 Figure 55 所示为可能的数据帧结构组合。括号中的位是可选的。

Figure 55. 帧格式



St 起始位，总是为低电平

(n) 数据位 (0 ~ 8)

P 校验位，可以为奇校验或偶校验

Sp 停止位，总是为高电平

IDLE 通讯线上没有数据传输 (RxD 或 TxD)，线路空闲时必须为高电平

数据帧的结构由 UCSRB 和 UCSRC 寄存器中的 UCSZ2:0、UPM1:0 与 USBS 设定。接收与发送使用相同的设置。设置的任何改变都可能破坏正在进行的数据传送与接收。

USART的字长位UCSZ2:0确定了数据帧的数据位数；校验模式位UPM1:0用于使能与决定校验的类型； USBS 位设置帧有一位或两位结束位。接收器忽略第二个停止位，因此帧错误 (FE) 只在第一个结束位为 "0” 时被检测到。

计算奇偶校验位

校验位的计算是对数据的各个位进行异或运算。如果选择了奇校验，则异或结果还需要取反。校验位与数据位的关系如下：:

$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$
$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

P_{even} 偶校验结果

P_{odd} 奇校验位结果

d_n 第 n 个数据位

校验位处于最后一个数据位与第一个停止位之间。

USART 初始化

进行通信之前首先要对 USART 进行初始化。初始化过程通常包括波特率的设定，帧结构的设定，以及根据需要使能接收器或发送器。对于中断驱动的 USART 操作，在初始化时首先要清零全局中断标志位 (全局中断被屏蔽)。

重新改变 USART 的设置应该在没有数据传输的情况下进行。 TXC 标志位可以用来检验一个数据帧的发送是否已经完成， RXC 标志位可以用来检验接收缓冲器中是否还有数据未读出。在每次发送数据之前 (在写发送数据寄存器 UDR 前)TXC 标志位必须清零。

以下是 USART 初始化程序示例。例程采用了轮询（中断被禁用）的异步操作，而且帧结构是固定的。波特率作为函数参数给出。在汇编程序里波特率参数保存于寄存器 r17:r16。

汇编代码例程 (1)

```
USART_Init:
  ; 设置波特率
  out   UBRRH, r17
  out   UBRRL, r16
  ; 接收器与发送器使能
  ldi   r16, (1<<RXEN)|(1<<TXEN)
  out   UCSRB,r16
  ;设置帧格式： 8 个数据位， 2 个停止位
  ldi   r16, (1<<USBS)|(3<<UCSZ0)
  out   UCSRC,r16
  ret
```

C 代码例程 (1)

```
void USART_Init( unsigned int baud )
{
  /* 设置波特率*/
  UBRRH = (unsigned char)(baud>>8);
  UBRRL = (unsigned char)baud;
  /* 接收器与发送器使能*/
  UCSRB = (1<<RXEN)|(1<<TXEN);
  /* 设置帧格式： 8 个数据位， 2 个停止位 */
  UCSRC = (1<<USBS)|(3<<UCSZ0);
}
```

Note: 1. 本代码假定已经包含了合适的头文件。
当 I/O 寄存器为扩展 I/O 寄存器时，必须用诸如 “LDS” 、“STS”、“SBRS”、“SBRC”、“SBR” 与 “CBR” 等可访问扩展 I/O 寄存器的指令代替 “IN”、“OUT”、“SBIS”、“SBIC”、“CBI” 与 “SBI” 指令。

更高级的初始化程序可将帧格式作为参数、禁止中断等等。然而许多应用程序使用固定的波特率与控制寄存器。此时初始化代码可以直接放在主程序中，或与其它 I/O 模块的初始化代码组合到一起。

发送数据 - USART 发送器

置位 UCSRB 寄存器的发送允许位 TXEN 将使能 USART 的数据发送。使能后 TxD 引脚的通用 I/O 功能即被 USART 功能所取代，成为发送器的串行输出引脚。发送数据之前要设置好波特率、工作模式与帧结构。如果使用同步发送模式，施加于 XCK 引脚上的时钟信号即为数据发送的时钟。

以 5 到 8 个数据位的方式发送帧

将需要发送的数据加载到发送缓存器将启动数据发送。加载过程即为 CPU 对 UDR 寄存器的写操作。当移位寄存器可以发送新一帧数据时，缓冲的数据将转移到移位寄存器。当移位寄存器处于空闲状态（没有正在进行的数据传输），或前一帧数据的最后一个停止位传送结束，它将加载新的数据。一旦移位寄存器加载了新的数据，就会按照设定的波特率完成数据的发送。

以下程序给出一个对 UDRE 标志采用轮询方式发送数据的例子。当发送的数据少于 8 位时，写入 UDR 相应位置的高几位将被忽略。当然，执行本段代码之前首先要初始化 USART。在汇编代码中要发送的数据存放于 R16。

汇编代码例程 (1)

```
USART_Transmit:
  ; 等待发送缓冲器为空
  sbis  UCSRA,UDRE
  rjmp  USART_Transmit
  ; 将数据放入缓冲器，发送数据
  out   UDR,r16
  ret
```

C 代码例程 (1)

```
void USART_Transmit( unsigned char data )
{
  /* 等待发送缓冲器为空 */
  while ( !( UCSRA & (1<<UDRE)) )
        ;
  /* 将数据放入缓冲器，发送数据*/
  UDR = data;
}
```

Note: 1. 本代码假定已经包含了合适的头文件。
当 I/O 寄存器为扩展 I/O 寄存器时，必须用诸如 “LDS” 、“STS”、“SBRS”、“SBRC”、“SBR” 与 “CBR” 等可访问扩展 I/O 寄存器的指令代替 “IN”、“OUT”、“SBIS”、“SBIC”、“CBI” 与 “SBI” 指令。

这个程序只是在载入新的要发送的数据前，通过检测 UDRE 标志等待发送缓冲器为空。如果使用了数据寄存器空中断，则数据写入缓冲器的操作在中断程序中进行。

以 9 个数据位的方式发送帧

如果发送 9 位数据的数据帧 (UCSZ = 7)，应先将数据的第 9 位写入寄存器 UCSRB 的 TXB8，然后再将低8位数据写入发送数据寄存器UDR。以下程序给出发送9位数据的数据帧例子。在汇编代码中要发送的数据存放在 R17:R16 寄存器中。

汇编代码例程 (1)(2)

```
USART_Transmit:
  ; 等待发送缓冲器为空
  sbis  UCSRA,UDRE
  rjmp  USART_Transmit
  ; 将第 9 位从 r17 中复制到 TXB8
  cbi   UCSRB,TXB8
  sbrc  r17,0
  sbi   UCSRB,TXB8
  ; 将低 8 位数据放入缓冲器，发送数据
  out   UDR,r16
  ret
```

C 代码例程 (1)(2)

```
void USART_Transmit( unsigned int data )
{
  /* 等待发送缓冲器为空 */
  while ( !( UCSRA & (1<<UDRE))) )
        ;
  /* 将第 9 位复制到 TXB8 */
  UCSRB &= ~(1<<TXB8);
  if ( data & 0x0100 )
    UCSRB |= (1<<TXB8);
  /* 将数据放入缓冲器，发送数据 */
  UDR = data;
}
```

Notes:
1. 这些函数均为通用函数。如果 UCSRB 的内容在应用中是固定的，函数可以进一步优化。例如，初始化后只使用 UCSRB 寄存器的 TXB8 位。
2. 本代码假定已经包含了合适的头文件。
   当 I/O 寄存器为扩展 I/O 寄存器时，必须用诸如 “LDS” 、“STS”、“SBRS”、“SBRC”、“SBR” 与 “CBR” 等可访问扩展 I/O 寄存器的指令代替 “IN”、“OUT”、“SBIS”、“SBIC”、“CBI” 与 “SBI” 指令。

第 9 位数据在多机通信中用于表示地址帧，在同步通信中可以用于协议处理。

发送器标志和中断

USART 发送器有两个标志位:USART 数据寄存器空标志 UDRE 及传输结束标志 TXC ,两个标志位都可以产生中断。

数据寄存器空 UDRE 标志位表示发送缓冲器是否可以接受一个新的数据。该位在发送缓冲器空时被置 "1”；当发送缓冲器包含需要发送的数据时清零。为与将来的器件兼容，写 UCSRA 寄存器时该位要写 "0”。

当 UCSRB 寄存器中的数据寄存器空中断使能位 UDRIE 为 "1” 时，只要 UDRE 被置位 (且全局中断使能) ,就将产生 USART 数据寄存器空中断请求。对寄存器 UDR 执行写操作将清零 UDRE。当采用中断方式的传输数据时 ,在数据寄存器空中断服务程序中必须写一个新的数据到 UDR 以清零 UDRE ；或者是禁止数据寄存器空中断。否则一旦该中断程序结束，一个新的中断将再次产生。

当整个数据帧移出发送移位寄存器 ,同时发送缓冲器中又没有新的数据时 ,发送结束标志 TXC 置位。TXC 在传送结束中断执行时自动清零 ,也可在该位写 "1” 来清零。TXC 标志位对于采用如 RS-485 标准的半双工通信接口十分有用。在这些应用里，一旦传送完毕，应用程序必须释放通信总线并进入接收状态。

当 UCSRB 上的发送结束中断使能位 TXCIE 与全局中断使能位均被置为 "1” 时 ,随着 TXC 标志位的置位， USART 发送结束中断将被执行。一旦进入中断服务程序， TXC 标志位即被自动清零，中断处理程序不必执行 TXC 清零操作。

产生奇偶校验位

奇偶校验产生电路为串行数据帧生成相应的校验位。校验位使能 (UPM1 = 1) 时，发送控制逻辑电路会在数据的最后一位与第一个停止位之间插入奇偶校验位。

禁止发送器

TXEN 清零后 ,只有等到所有的数据发送完成后发送器才能够真正禁止 ,即发送移位寄存器与发送缓冲寄存器中没有要传送的数据。发送器禁止后 ,TxD 引脚恢复其通用 I/O 功能。

数据接收 - USART 接收器

置位 UCSRB 寄存器的接收允许位 (RXEN) 即可启动 USART 接收器。接收器使能后 RxD 的普通引脚功能被 USART 功能所取代，成为接收器的串行输入口。进行数据接收之前首先要设置好波特率、操作模式及帧格式。如果使用同步操作，XCK 引脚上的时钟被用为传输时钟。

以 5 到 8 个数据位的方式接收帧

一旦接收器检测到一个有效的起始位，便开始接收数据。起始位后的每一位数据都将以所设定的波特率或 XCK 时钟进行接收，直到收到一帧数据的第一个停止位。接收到的数据被送入接收移位寄存器。第二个停止位会被接收器忽略。接收到第一个停止位后，接收移位寄存器就包含了一个完整的数据帧。这时移位寄存器中的内容将被转移到接收缓冲器中。通过读取 UDR 就可以获得接收缓冲器的内容的。

以下程序给出一个对 RXC 标志采用轮询方式接收数据的例子。当数据帧少于 8 位时，从 UDR 读取的相应的高几位为 0。当然，执行本段代码之前首先要初始化 USART。

汇编代码例程 (1)

```
USART_Receive:
  ; 等待接收数据
  sbis  UCSRA, RXC
  rjmp  USART_Receive
  ; 从缓冲器中获取并返回数据
  in    r16, UDR
  ret
```

C 代码例程 (1)

```
unsigned char USART_Receive( void )
{
  /* 等待接收数据*/
  while ( !(UCSRA & (1<<RXC)) )
        ;
  /* 从缓冲器中获取并返回数据*/
  return UDR;
}
```

Note: 1. 本代码假定已经包含了相应的头文件。
当 I/O 寄存器为扩展 I/O 寄存器时，必须用诸如 “LDS” 、“STS”、“SBRS”、“SBRC”、“SBR” 与 “CBR” 等可访问扩展 I/O 寄存器的指令代替 “IN”、“OUT”、“SBIS”、“SBIC”、“CBI” 与 “SBI” 指令。

在读缓冲器并返回之前，函数通过检查 RXC 标志来等待数据送入接收缓冲器。

以 9 个数据位的方式发送帧

如果设定了 9 位数据的数据帧 (UCSZ=7)，在从 UDR 读取低 8 位**之前**必须首先读取寄存器 UCSRB 的 RXB8 以获得第 9 位数据。这个规则同样适用于状态标志位 FE、 DOR 及 UPE。状态通过读取UCSRA获得，数据通过UDR获得。读取UDR存储单元会改变接收缓冲器 FIFO 的状态，进而改变同样存储在 FIFO 中的 TXB8、FE、 DOR 及 UPE 位。

接下来的代码示例展示了一个简单的USART接收函数，说明如何处理9位数据及状态位。

汇编代码例程 (1)

```
USART_Receive:
  ; 等待接收数据
  sbis  UCSRA, RXC
  rjmp  USART_Receive
  ; 从缓冲器中获得状态、第 9 位及数据
  in    r18, UCSRA
  in    r17, UCSRB
  in    r16, UDR
  ; 如果出错，返回 -1
  andi  r18,(1<<FE)|(1<<DOR)|(1<<UPE)
  breq  USART_ReceiveNoError
  ldi   r17, HIGH(-1)
  ldi   r16, LOW(-1)
USART_ReceiveNoError:
  ;过滤第 9 位数据，然后返回
  lsr   r17
  andi  r17, 0x01
  ret
```

C 代码例程 (1)

```
unsigned int USART_Receive( void )
{
  unsigned char status, resh, resl;
  /* 等待接收数据 */
  while ( !(UCSRA & (1<<RXC)) )
        ;
  /* 从缓冲器中获得状态、第 9 位及数据 */
  status = UCSRA;
  resh = UCSRB;
  resl = UDR;
  /* 如果出错，返回 -1 */
  if ( status & (1<<FE)|(1<<DOR)|(1<<UPE) )
    return -1;
  /* 过滤第 9 位数据，然后返回*/
  resh = (resh >> 1) & 0x01;
  return ((resh << 8) | resl);
}
```

Note: 1. 本代码假定已经包含了相应的头文件。
当 I/O 寄存器为扩展 I/O 寄存器时，必须用诸如 “LDS” 、“STS”、“SBRS”、“SBRC”、“SBR” 与 “CBR” 等可访问扩展 I/O 寄存器的指令代替 “IN”、“OUT”、“SBIS”、“SBIC”、“CBI” 与 “SBI” 指令。

上述例子在进行任何计算之前将所有的 I/O 寄存器的内容读到寄存器文件中。这种方法优化了对接收缓冲器的利用。它尽可能早地释放了缓冲器以接收新的数据。

接收完成标志和中断

USART 接收器有一个标志用来指明接收器的状态。

接收结束标志 (RXC) 用来说明接收缓冲器中是否有未读出的数据。当接收缓冲器中有未读出的数据时，此位为 1，当接收缓冲器空时为 0(即不包含未读出的数据)。如果接收器被禁止 (RXEN = 0)，接收缓冲器会被刷新，从而使 RXC 清零。

置位 UCSRB 的接收结束中断使能位 (RXCIE) 后，只要 RXC 标志置位 (且全局中断只能) 就会产生 USART 接收结束中断。使用中断方式进行数据接收时，数据接收结束中断服务程序程序必须从 UDR 读取数据以清 RXC 标志，否则只要中断处理程序一结束，一个新的中断就会产生。

接收器错误标志

USART 接收器有三个错误标志: : 帧错误 (FE)、数据溢出 (DOR) 及奇偶校验错 (UPE)。它们都位于寄存器 UCSRA。错误标志与数据帧一起保存在接收缓冲器中。由于读取 UDR 会改变缓冲器， UCSRA 的内容必须在读接收缓冲器 (UDR) 之前读入。错误标志的另一个同一性是它们都不能通过软件写操作来修改。但是为了保证与将来产品的兼容性，对执行写操作是必须对这些错误标志所在的位置写 "0“。所有的错误标志都不能产生中断。

帧错误标志 (FE) 表明了存储在接收缓冲器中的下一个可读帧的第一个停止位的状态。停止位正确 (为 1) 则 FE 标志为 0，否则 FE 标志为 1。这个标志可用来检测同步丢失、传输中断，也可用于协议处理。UCSRC 中 USBS 位的设置不影响 FE 标志位，因为除了第一位，接收器忽略所有其他的停止位。为了与以后的器件相兼容，写 UCSRA 时这一位必须置 0。

数据溢出标志 (DOR) 表明由于接收缓冲器满造成了数据丢失。当接收缓冲器满 (包含了两个数据)，接收移位寄存器又有数据，若此时检测到一个新的起始位，数据溢出就产生了。DOR 标志位置位即表明在最近一次读取 UDR 和下一次读取 UDR 之间丢失了一个或更多的数据帧。为了与以后的器件相兼容，写 UCSRA 时这一位必须置 0。当数据帧成功地从移位寄存器转入接收缓冲器后， DOR 标志被清零。

奇偶校验错标志 (UPE) 指出，接收缓冲器中的下一帧数据在接收时有奇偶错误。如果不使能奇偶校验，那么 UPE 位应清零。为了与以后的器件相兼容，写 UCSRA 时这一位必须置 0。细节请参照 P109“ 计算奇偶校验位 ” 及 P117“ 奇偶校验 ” 。

奇偶校验

奇偶校验模式位UPM1置位将启动奇偶校验器。校验的模式(偶校验还是奇校验)由UPM0确定。奇偶校验使能后，校验器将计算输入数据的奇偶并把结果与数据帧的奇偶位进行比较。校验结果将与数据和停止位一起存储在接收缓冲器中。这样就可以通过读取奇偶校验错误标志位 (UPE) 来检查接收的帧中是否有奇偶错误。

如果下一个从接收缓冲器中读出的数据有奇偶错误，并且奇偶校验使能 (UPM1 = 1)，则 UPE 置位。直到接收缓冲器 (UDR) 被读取，这一位一直有效。

禁止接收器

与发送器对比，禁止接收器即刻起作用。正在接收的数据将丢失。禁止接收器 (RXEN 清零) 后，接收器将不再占用 RxD 引脚；接收缓冲器 FIFO 也会被刷新。缓冲器中的数据将丢失。

刷新接收缓冲器

禁止接收器时缓冲器 FIFO 被刷新，缓冲器被清空。导致未读出的数据丢失。如果由于出错而必须在正常操作下刷新缓冲器，则需要一直读取 UDR 直到 RXC 标志清零。下面的代码展示了如何刷新接收缓冲器。

汇编代码例程 (1)

```
USART_Flush:
  sbis  UCSRA, RXC
  ret
  in    r16, UDR
  rjmp  USART_Flush
```

C 代码例程 (1)

```
void USART_Flush( void )
{
  unsigned char dummy;
  while ( UCSRA & (1<<RXC) ) dummy = UDR;
}
```

Note: 1. 本代码假定已经包含了相应的头文件。
当 I/O 寄存器为扩展 I/O 寄存器时，必须用诸如 “LDS” 、“STS”、“SBRS”、“SBRC”、“SBR” 与 “CBR” 等可访问扩展 I/O 寄存器的指令代替 “IN”、“OUT”、“SBIS”、“SBIC”、“CBI” 与 “SBI” 指令。

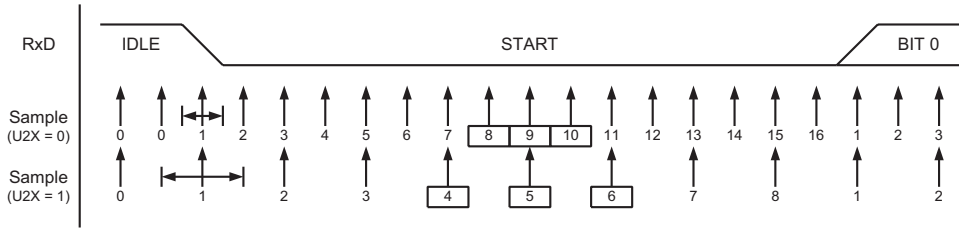
接收异步数据

USART 有一个时钟恢复单元和数据恢复单元用来处理异步数据接收。时钟恢复逻辑用于同步从 RxD 引脚输入的异步串行数据和内部的波特率时钟。数据恢复逻辑采集数据，并通过一低通滤波器过滤所输入的每一位数据，从而提高接收器的抗干扰性能。异步接收的工作范围依赖于内部波特率时钟的精度、帧输入的速率及一帧所包含的位数。

恢复异步时钟

时钟恢复逻辑将输入的串行数据帧与内部时钟同步起来。 Figure 56 展示了对输入数据帧起始位的采样过程。普通工作模式下采样率是波特率的 16 倍，倍速工作模式下则为波特率的 8 倍。水平箭头表示由于采样而造成的同步的变化。使用倍速模式 (U2X = 1) 时同步变化时间更长。 RxD 线空闲 (即没有任何通讯活动) 时，采样值为 0。

Figure 56. 起始位采样

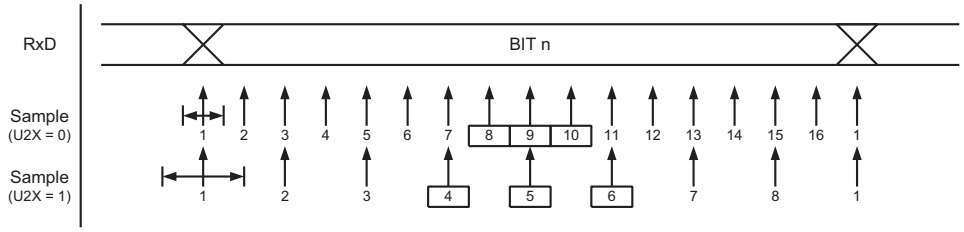


当时钟恢复电路检测到 RxD 线上一个由高 (空闲) 到低 (开始) 的电平跳变时，起始位检测序列即被启动。如图所示，我们用采样 1 表示第一个 0 采样。然后，时钟恢复逻辑用采样 8、 9、 10(普通模式)，或采样 4、 5、 6(倍速模式)，来判断是否接收到一个正确的起始位。如果这三个采样中的两个或更多个是逻辑高电平 (多数表决)，起始位会被视为毛刺噪声而被拒绝接受，接收器等待下一个由高到低的电平转换。如果检测到一个有效的起始位，时钟恢复逻辑即被同步并开始接收数据。每一个起始位都会引发同样的同步过程。

恢复异步数据

接收时钟与起始位同步之后，数据恢复工作可开始了。数据恢复单元使用一个状态机来接收每一个数据位。这个状态机在普通模式下具有 16 个状态，在倍速模式下具有 8 个状态。Figure 57 演示了对数据位和奇偶位的采样。每个采样点都被赋予了一个数字，这个数字等于数据恢复单元当前的状态序号。

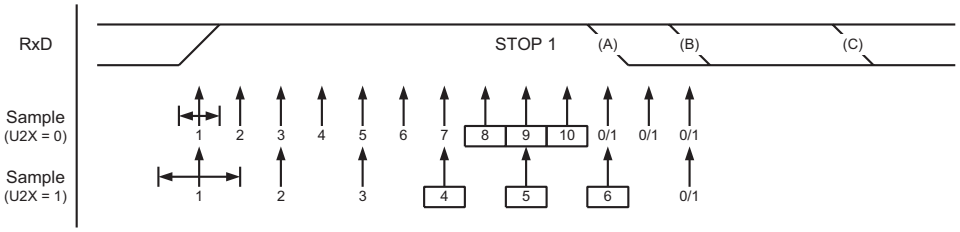
Figure 57. 数据及奇偶位的采样



确定接收到的数据位的逻辑电平的方法为多数表决法。表决对象即为三个在数据位中心获得的采样。为了强调这些采样，图中采样序号被包含小方框中。多数表决是这样工作的 : 如果有 2 个或所有 3 个采样值都是高电平，那么接收位就为逻辑 1。如果 2 个或所有 3个采样值都是低电平，那么接收位就被为逻辑0。对从RxD引脚输入的信号来说，多数表决的作用就象是一个低通滤波。数据恢复过程重复进行，直到接收到一个完整的数据帧。其中也包含了第一个停止位。接收器将忽略其他的停止位。

Figure 58 说明了停止位的采样，以及下一帧信号起始位最早可能出现的情况。

Figure 58. 停止位及下一个起始位采样



多数表决对停止位同样有效。若停止位为逻辑 0，那么帧错误标志 FE 置位。

如果电平再一次出现了从高到低的跳变，说明紧接着上一个数据帧来了新的数据帧。在普通模式中，第一个低电平的采样点可以发生在 Figure 58 的 A 点。在倍速工作模式下第一个低电平采样点必须延迟到 B 点， C 点则为完整停止位的结束位置。对起始位的及早检测将影响接收器的工作范围。

异步工作范围

接收器的工作范围取决于接收到的数据速率及内部波特率之间的不匹配程度。如果发送器以过快或过慢的比特率传输数据帧，或者接收器内部产生的波特率没有相同的频率 (见 Table 49)，那么接收器就无法与起始位同步。

下面的公式可用来计算数据输入速率与内部接收器波特率的比值。

$$R_{slow} = \frac{(D+1)S}{S-1+D\cdot S+S_F} \qquad R_{fast} = \frac{(D+2)S}{(D+1)S+S_M}$$

D 字符长度及奇偶位长度的总和 (D = 5 到 10 位)

S 每一位的采样数。普通模式下 S = 16，倍速模式下 S = 8

S_F 用于多数表决的第一个采样序号。普通模式下 S_F = 8 ，倍速模式下 S_F = 4

S_M 用于多数表决的中间采样序号。普通模式下 S_M = 9 ，倍速模式下 S_M = 5

R_{slow} 是可接受的、最慢的数据输入速率与接收器波特率的比值； R_{fast} 是可接受的、最快的数据输入速率与接收器波特率的比值。

Table 49 和 Table 50 列出了容许的最大接收器波特率误差。需要注意的是，普通模式下波特率允许有更大的变化范围。

Table 49. 普通模式下推荐的最大接收器波特率误差范围 (U2X = 0)

D # (数据 + 奇偶位)	R_{slow} (%)	R_{fast} (%)	最大的总误差 (%)	推荐的最大接收器误差 (%)
5	93.20	106.67	+6.67/-6.8	± 3.0
6	94.12	105.79	+5.79/-5.88	± 2.5
7	94.81	105.11	+5.11/-5.19	± 2.0
8	95.36	104.58	+4.58/-4.54	± 2.0
9	95.81	104.14	+4.14/-4.19	± 1.5
10	96.17	103.78	+3.78/-3.83	± 1.5

Table 50. 倍速率模式下推荐的最大接收器波特率误差范围 (U2X = 1)

D # (数据 + 奇偶位)	R_{slow} (%)	R_{fast} (%)	最大的总误差 (%)	推荐的最大接收器误差 (%)
5	94.12	105.66	+5.66/-5.88	± 2.5
6	94.92	104.92	+4.92/-5.08	± 2.0
7	95.52	104,35	+4.35/-4.48	± 1.5
8	96.00	103.90	+3.90/-4.00	± 1.5
9	96.39	103.53	+3.53/-3.61	± 1.5
10	96.70	103.23	+3.23/-3.30	± 1.0

上述推荐的最大接收波特率误差是在假定接收器和发送器对最大总误差具有同等贡献的前提下得出的。

产生接收器波特率误差的可能原因有两个。首先，接收器系统时钟 (XTAL) 的稳定性于电压范围及工作温度有关。使用晶振来产生系统时钟时一般不会有此问题，但对于谐振器而言，根据谐振器不同的误差容限，系统时钟可能有超过 2% 的偏差。第二个误差的原因就好控制多了。波特率发生器不一定能够通过对系统时钟的分频得到恰好的波特率。此时可以调整 UBRR 值，使得误差低至可以接受。

多处理器通讯模式

置位 UCSRA 的多处理器通信模式位 (MPCM) 可以对 USART 接收器接收到的数据帧进行过滤。那些没有地址信息的帧将被忽略，也不会存入接收缓冲器。在一个多处理器系统中，处理器通过同样的串行总线进行通信，这种过滤有效的减少了需要 CPU 处理的数据帧的数量。MPCM 位的设置不影响发送器的工作，但在使用多处理器通信模式的系统中，它的使用方法会有所不同。

如果接收器所接收的数据帧长度为 5 到 8 位，那么第一个停止位表示这一帧包含的是数据还是地址信息。如果接收器所接收的数据帧长度为 9 位，那么由第 9 位 (RXB8) 来确定是数据还是地址信息。如果确定帧类型的位 (第一个停止位或第 9 个数据位) 为 1，那么这是地址帧，否则为数据帧。

在多处理器通信模式下，多个从处理器可以从一个主处理器接收数据。首先要通过解码地址帧来确定所寻址的是哪一个处理器。如果寻址到某一个处理器，它将正常接收后续的数据，而其他的从处理器会忽略这些帧直到接收到另一个地址帧。

使用 MPCM

对于一个作为主机的处理器来说，它可以使用 9 位数据帧格式 (UCSZ = 7)。如果传输的是一个地址帧 (TXB8 = 1) 就将第 9 位 (TXB8) 置 1，如果是一个数据帧 (TXB = 0) 就将它清零。在这种帧格式下，从处理器必须工作于 9 位数据帧格式。

下面即为在多处理器通信模式下进行数据交换的步骤：

1. 所有从处理器都工作在多处理器通信模式 (UCSRA 寄存器的 MPCM 置位)。
2. 主处理器发送地址帧后，所有从处理器都会接收并读取此帧。从处理器 UCSRA 寄存器的 RXC 正常置位。
3. 每一个从处理器都会读取 UDR 寄存器的内容已确定自己是否被选中。如果选中，就清零 UCSRA 的 MPCM 位，否则它将等待下一个地址字节的到来，并保持 MPCM 为 1。
4. 被寻址的从处理器将接收所有的数据帧，直到收到一个新的地址帧。而那些保持 MPCM 位为 1 的从处理器将忽略这些数据。
5. 被寻址的处理器接收到最后一个数据帧后，它将置位 MPCM，并等待主处理器发送下一个地址帧。然后第 2 步之后的步骤重复进行。

使用 5 至 8 比特的帧格式是可以的，但是不实际，因为接收器必须在使用 n 和 n+1 帧格式之间进行切换。由于接收器和发送器使用相同的字符长度设置，这种设置使得全双工操作变得很困难。如果使用 5 至 8 比特的帧格式，发送器应该设置两个停止位 (USBS = 1)，其中的第一个停止位被用于判断帧类型。

不要使用读 - 修改 - 写指令 (SBI 和 CBI) 来操作 MPCM 位。 MPCM 和 TXC 标志使用相同的 I/O 单元，使用 SBI 或 CBI 指令可能会不小心将它清零。

USART 寄存器说明

USART I/O 数据寄存器 - UDR

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDR (读)
	TXB[7:0]								UDR (写)
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	

USART 发送数据缓冲寄存器和 USART 接收数据缓冲寄存器共享相同的 I/O 地址，称为 USART 数据寄存器或 UDR。将数据写入 UDR 时实际操作的是发送数据缓冲器存器 (TXB)，读 UDR 时实际返回的是接收数据缓冲寄存器 (RXB) 的内容。

在 5、6、7 比特字长模式下，未使用的高位被发送器忽略，而接收器则将它们设置为 0。

只有当 UCSRA 寄存器的 UDRE 标志置位后才可以对发送缓冲器进行写操作。如果 UDRE 没有置位，那么写入 UDR 的数据会被 USART 发送器忽略。当数据写入发送缓冲器后，

若移位寄存器为空，发送器将把数据加载到发送移位寄存器。然后数据串行地从 TxD 引脚输出。

接收缓冲器包括一个两级 FIFO，一旦接收缓冲器被寻址 FIFO 就会改变它的状态。因此不要对这一存储单元使用读 - 修改 - 写指令 (SBI 和 CBI)。使用位查询指令 (SBIC 和 SBIS) 时也要小心，因为这也有可能改变 FIFO 的状态。

USART 控制和状态寄存器 A - UCSRA

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM	UCSRA
读 / 写	R	R/W	R	R	R	R	R/W	R/W	
初始值	0	0	1	0	0	0	0	0	

- **Bit 7 – RXC: USART 接收结束**

接收缓冲器中有未读出的数据时 RXC 置位，否则清零。接收器禁止时，接收缓冲器被刷新，导致 RXC 清零。 RXC 标志可用来产生接收结束中断 (见对 RXCIE 位的描述)。

- **Bit 6 – TXC: USART 发送结束**

发送移位缓冲器中的数据被送出，且当发送缓冲器 (UDR) 为空时 TXC 置位。执行发送结束中断时 TXC 标志自动清零，也可以通过写 1 进行清除操作。TXC 标志可用来产生发送结束中断 (见对 TXCIE 位的描述)。

- **Bit 5 – UDRE: USART 数据寄存器空**

UDRE标志指出发送缓冲器(UDR)是否准备好接收新数据。UDRE为1说明缓冲器为空，已准备好进行数据接收。UDRE标志可用来产生数据寄存器空中断(见对UDRIE位的描述)。

复位后 UDRE 置位，表明发送器已经就绪。

- **Bit 4 – FE: 帧错误**

如果接收缓冲器接收到的下一个字符有帧错误，即接收缓冲器中的下一个字符的第一个停止位为 0，那么 FE 置位。这一位一直有效直到接收缓冲器 (UDR) 被读取。当接收到的停止位为 1 时， FE 标志为 0。对 UCSRA 进行写入时，这一位要写 0。

- **Bit 3 – DOR: 数据过速**

数据过速时 DOR 置位。当接收缓冲器满 (包含了两个数据) ，接收移位寄存器又有数据，若此时检测到一个新的起始位，数据溢出就产生了。这一位一直有效直到接收缓冲器 (UDR) 被读取。对 UCSRA 进行写入时，这一位要写 0。

- **Bit 2 – UPE: USART 奇偶校验错误**

当奇偶校验使能 (UPM1 = 1)，且接收缓冲器中所接收到的下一个字符有奇偶校验错误时 UPE 置位。这一位一直有效直到接收缓冲器 (UDR) 被读取。对 UCSRA 进行写入时，这一位要写 0。

- **Bit 1 – U2X: 倍速发送**

这一位仅对异步操作有影响。使用同步操作时将此位清零。

此位置 1 可将波特率分频因子从 16 降到 8，从而有效的将异步通信模式的传输速率加倍。

- **Bit 0 – MPCM: 多处理器通信模式**

设置此位将启动多处理器通信模式。 MPCM 置位后， USART 接收器接收到的那些不包含地址信息的输入帧都将被忽略。发送器不受MPCM设置的影响。详细信息请参考 P121“多处理器通讯模式”。

USART 控制和状态寄存器 B - UCSRB

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
读 / 写	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
初始值	0	0	0	0	0	0	0	0	

- **Bit 7 – RXCIE: 接收结束中断使能**

置位后使能 RXC 中断。当 RXCIE 为 1，全局中断标志位 SREG 置位， UCSRA 寄存器的 RXC 亦为 1 时可以产生 USART 接收结束中断。

- **Bit 6 – TXCIE: 发送结束中断使能**

置位后使能 TXC 中断。当 TXCIE 为 1，全局中断标志位 SREG 置位，UCSRA 寄存器的 TXC 亦为 1 时可以产生 USART 发送结束中断。

- **Bit 5 – UDRIE: USART 数据寄存器空中断使能**

置位后使能 UDRE 中断。当 UDRIE 为 1，全局中断标志位 SREG 置位，UCSRA 寄存器的 UDRE 亦为 1 时可以产生 USART 数据寄存器空中断。

- **Bit 4 – RXEN: 接收使能**

置位后将启动 USART 接收器。 RxD 引脚的通用端口功能被 USART 功能所取代。禁止接收器将刷新接收缓冲器，并使 FE、 DOR 及 UPE 标志无效。

- **Bit 3 – TXEN: 发送使能**

置位后将启动将启动 USART 发送器。TxD 引脚的通用端口功能被 USART 功能所取代。TXEN 清零后，只有等到所有的数据发送完成后发送器才能够真正禁止，即发送移位寄存器与发送缓冲寄存器中没有要传送的数据。发送器禁止后，TxD 引脚恢复其通用 I/O 功能。

- **Bit 2 – UCSZ2: 字符长度**

UCSZ2与UCSRC寄存器的UCSZ1:0结合在一起可以设置数据帧所包含的数据位数(字符长度)。

- **Bit 1 – RXB8: 接收数据位 8**

对 9 位串行帧进行操作时， RXB8 是第 9 个数据位。读取 UDR 包含的低位数据之前首先要读取 RXB8。

- **Bit 0 – TXB8: 发送数据位 8**

对 9 位串行帧进行操作时，TXB8 是第 9 个数据位。写 UDR 之前首先要对它进行写操作。

USART 控制和状态寄存器 C - UCSRC

Bit	7	6	5	4	3	2	1	0	
	–	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
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	1	1	0	

- **Bit 6 – UMSEL: USART 模式选择**

通过这一位来选择同步或异步工作模式。

Table 51. UMSEL 设置

UMSEL	模式
0	异步操作
1	同步操作

- **Bit 5:4 – UPM1:0: 奇偶校验模式**

这两位设置奇偶校验的模式并使能奇偶校验。如果使能了奇偶校验，那么在发送数据，发送器都会自动产生并发送奇偶校验位。对每一个接收到的数据，接收器都会产生一奇偶值，并与 UPM0 所设置的值进行比较。如果不匹配，那么就将 UCSRA 中的 UPE 置位。

Table 52. UPM 设置

UPM1	UPM0	奇偶模式
0	0	禁止
0	1	保留
1	0	偶校验
1	1	奇校验

- **Bit 3 – USBS: 停止位选择**

通过这一位可以设置停止位的位数。接收器忽略这一位的设置。

Table 53. USBS 设置

USBS	停止位位数
0	1
1	2

- **Bit 2:1 – UCSZ1:0: 字符长度**

UCSZ1:0与UCSRB寄存器的 UCSZ2结合在一起可以设置数据帧包含的数据位数(字符长度)，见 P126Table 54。

Table 54. UCSZ 设置

UCSZ2	UCSZ1	UCSZ0	字符长度
0	0	0	5 位
0	0	1	6 位
0	1	0	7 位
0	1	1	8 位
1	0	0	保留
1	0	1	保留
1	1	0	保留
1	1	1	9 位

- **Bit 0 – UCPOL: 时钟极性**

这一位仅用于同步工作模式。使用异步模式时，将这一位清零。UCPOL 设置了输出数据的改变和输入数据采样，以及同步时钟 XCK 之间的关系。

Table 55. UCPOL 设置

UCPOL	发送数据的改变 (TxD 引脚的输出)	接收数据的采样 (RxD 引脚的输入)
0	XCK 上升沿	XCK 下降沿
1	XCK 下降沿	XCK 上升沿

USART 波特率寄存器 - UBRRL 与 UBRRH

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRR[11:8]				UBRRH
	UBRR[7:0]								UBRRL
	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	0	
	0	0	0	0	0	0	0	0	

- **Bit 15:12 – 保留**

这些位是为以后的使用而保留的。为了与以后的器件兼容，写 UBRRH 时将这些位清零。

- **Bit 11:0 – UBRR11:0: USART 波特率寄存器**

这个 12 位的寄存器包含了 USART 的波特率信息。其中 UBRRH 包含了 USART 波特率高 4 位，UBRRL 包含了低 8 位。波特率的改变将造成正在进行的数据传输受到破坏。写 UBRRL 将立即更新波特率分频器。

设置波特率的例子

对标准晶振及谐振器频率来说，异步模式下最常用的波特率可通过 Table 56 中 UBRR 的设置来产生。表中的粗体数据表示由此产生的波特率与目标波特率的偏差不超过 0.5%。更高的误差也是可以接受的，但发送器的抗噪性会降低，特别是需要传输大量数据时 (参看 P119“ 异步工作范围 ”)。误差可以通过如下公式计算：

$$\text{Error}[\%] = \left(\frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1\right) \bullet 100\%$$

Table 56. 通用振荡器频率下设置 UBRR 的例子

波特率 (bps)	f_{osc} = 1.0000 MHz				f_{osc} = 1.8432 MHz				f_{osc} = 2.0000 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	误差	UBRR	误差	UBRR	误差	UBRR	误差	UBRR	误差	UBRR	误差
2400	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
9600	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
14.4k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
19.2k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
38.4k	1	-18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	-7.0%
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
76.8k	–	–	1	-18.6%	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%
115.2k	–	–	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%
230.4k	–	–	–	–	–	–	0	0.0%	–	–	–	–
250k	–	–	–	–	–	–	–	–	–	–	0	0.0%
最大 (1)	62.5 kbps		125 kbps		115.2 kbps		230.4 kbps		125 kbps		250 kbps	

1. UBRR = 0, 误差 = 0.0%

Table 57. 通用振荡器频率下设置 UBRR 的例子

波特率 (bps)	f_{osc} = 3.6864 MHz				f_{osc} = 4.0000 MHz				f_{osc} = 7.3728 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	误差	UBRR	误差	UBRR	误差	UBRR	误差	UBRR	误差	UBRR	误差
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
14.4k	15	0.0%	31	0.0%	16	2.1%	34	-0.8%	31	0.0%	63	0.0%
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
28.8k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
38.4k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
57.6k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
76.8k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
250k	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%
0.5M	–	–	0	-7.8%	–	–	0	0.0%	0	-7.8%	1	-7.8%
1M	–	–	–	–	–	–	–	–	–	–	0	-7.8%
最大 (1)	230.4 kbps		460.8 kbps		250 kbps		0.5 Mbps		460.8 kbps		921.6 kbps	

1. UBRR = 0, 误差 = 0.0%

Table 58. 通用振荡器频率下设置 UBRR 的例子

波特率 (bps)	f_{osc} = 8.0000 MHz				f_{osc} = 11.0592 MHz				f_{osc} = 14.7456 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	误差	UBRR	误差	UBRR	误差	UBRR	误差	UBRR	误差	UBRR	误差
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
最大 (1)	0.5 Mbps		1 Mbps		691.2 kbps		1.3824 Mbps		921.6 kbps		1.8432 Mbps	

1. UBRR = 0, 误差 = 0.0%

Table 59.
通用振荡器频率下设置 UBRR 的例子

	f_{osc} = 16.0000 MHz			
	U2X = 0		U2X = 1	
波特率 (bps)	UBRR	误差	UBRR	误差
2400	416	-0.1%	832	0.0%
4800	207	0.2%	416	-0.1%
9600	103	0.2%	207	0.2%
14.4k	68	0.6%	138	-0.1%
19.2k	51	0.2%	103	0.2%
28.8k	34	-0.8%	68	0.6%
38.4k	25	0.2%	51	0.2%
57.6k	16	2.1%	34	-0.8%
76.8k	12	0.2%	25	0.2%
115.2k	8	-3.5%	16	2.1%
230.4k	3	8.5%	8	-3.5%
250k	3	0.0%	7	0.0%
0.5M	1	0.0%	3	0.0%
1M	0	0.0%	1	0.0%
最大 (1)	1 Mbps		2 Mbps	

1. UBRR = 0, 误差 = 0.0%

通用串行接口 - USI

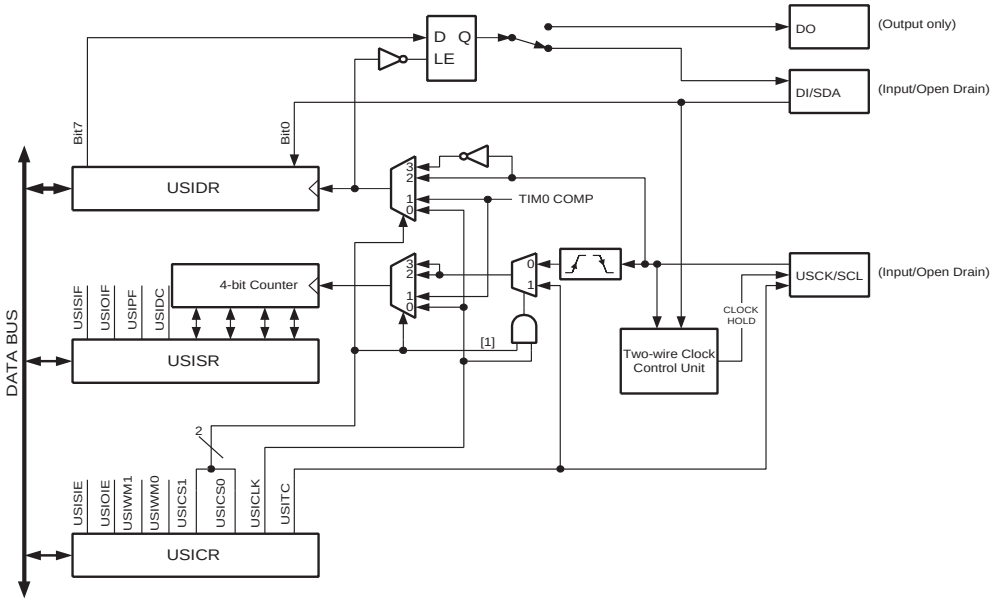
通用串行接口，或 USI，提供了进行串行通信所需的最基本的硬件资源。和最小化的控制软件结合之后，USI 可以提供比使用纯软件高得多的传输速率并使用更少的代码空间。可以使用中断来最小化处理器的工作量。 USI 的主要特性是：

- **两线同步数据传输 (主机或从机，$f_{SCLmax} = f_{CK}/16$)**
- **三线同步数据传输 (主机或从机，$f_{SCKmax} = f_{CK}/4$)**
- **数据接收中断**
- **可以从空闲模式唤醒**
- **两线模式下可从所有的睡眠模式唤醒，包括掉电模式**
- **两线启动条件检测器有中断能力**

概述

Figure 59.给出了一个简化的USI框图。实际的I/O管脚请参看 P2“ATtiny2313引脚配置”。CPU 可寻址的 I/O 寄存器，包括位和引脚，以粗体显示。器件专用 I/O 寄存器及位说明在P137“USI 寄存器说明 ”。

Figure 59. 通用串行接口框图



通过数据总线可直接访问 8 位的移位寄存器，这个寄存器包含正在发送和接收的数据。由于这个寄存器没有缓冲，因此要尽快读出其中的数据以保证数据不丢失。依据不同的工作模式，最高位与两个输出引脚之一相连。在串行寄存器输出和输出引脚之间有一个透明的锁存器，它的作用是将数据输出延迟到和用于数据输入采样的时钟沿相反的时钟沿。串行输入总是经过数据输入引脚 (DI) 进行采样，而与配置无关。

通过数据总线还可以访问 4 比特的计数器，并产生溢出中断。串行寄存器和计数器由相同的时钟驱动，从而定时器可以对接收或发送的比特数进行计数，并在传输结束时产生中断。当选择了外部时钟时，计数器将对时钟的上下两个沿进行计数。此时计数器统计沿的数目而不是比特数。有三种可选的时钟源：USCK 引脚、定时器 0 溢出或通过软件产生。

检测到起始条件后，两线时钟控制单元可以产生中断。它也可以在检测到起始条件或计数器溢出之后通过拉低时钟引脚来产生等待状态。

功能描述

三线模式

USI的三线工作模式与串行外设接口(SPI)模式0和模式1兼容，但没有从机选择(SS)功能。不过这可以通过软件来实现。这种模式下的引脚名称为 DI、 DO 和 USCK。

Figure 60. 三线模式简化框图

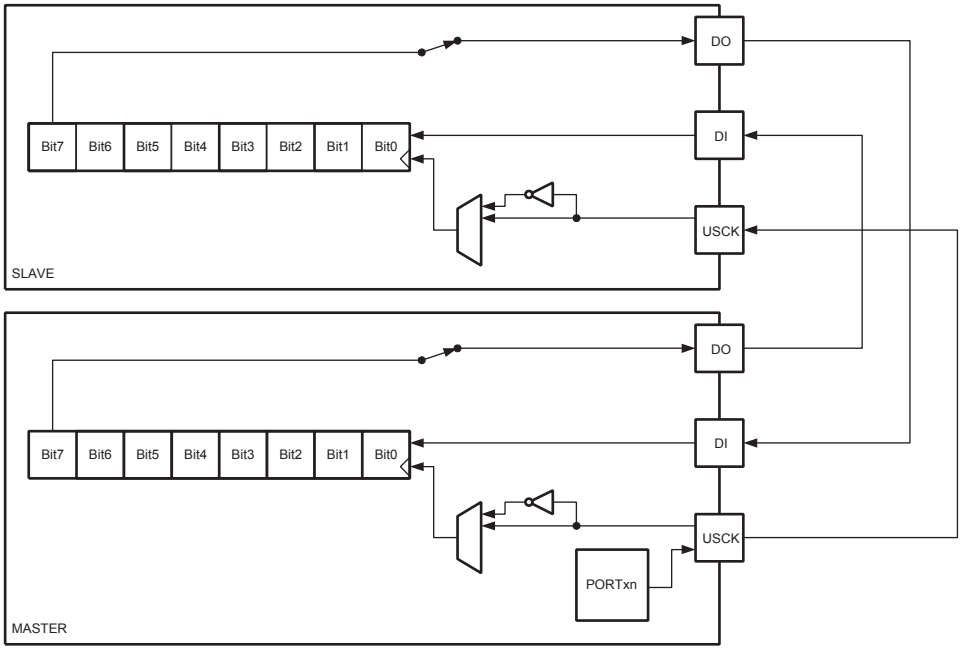


Figure 60 给出了两个工作于三线模式的 USI 单元，一个为主机，另一个为从机。两个移位寄存器的连接方式使得 8 个 USCK 时钟之后，两个寄存器中的数据相互交换。同样的时钟还驱动 USI 的 4 位计数器。因此计数器溢出 (中断) 标志 USIOI 可用来判断传输何时完成。这个时钟可以由两种方式产生 : 一是由主机软件通过操作端口寄存器来操作 USCK 引脚，二是置位 USICR 寄存器的 USITC。

Figure 61. 三线模式时序图

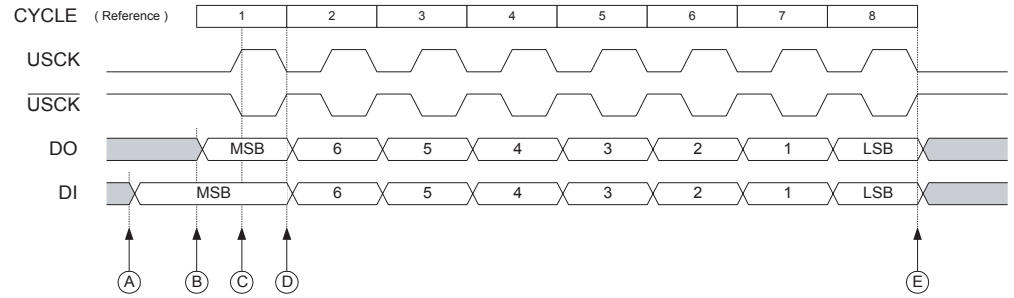


Figure 61. 给出了三线模式的时序。图的顶端是 USCK 的参考周期。在每一个这样的周期里都有一个比特的数据转移到 USI 移位寄存器 (USIDR) 中。 USCK 时序展示了两种外部时钟模式。工作于外部时钟模式 0(USICS0 = 0) 时， DI 在时钟的上升沿采样，输出 DO 在下降沿改变 (数据寄存器移动一位)，外部时钟模式 1(USICS0 = 1) 使用与模式 0 相反的时钟沿，即在在下降沿进行数据采样，在上升沿改变输出。USI 时钟模式对应于 SPI 数据模式 0 和模式 1。

由 Figure 61. 时序图可以看出，总线传输包括以下步骤 :

1. 通过向串行数据寄存器中写入需要发送的数据来准备数据输出。通过设置数据方向寄存器中的对应位来启动数据输出。注意， A 与 B 点之间并没有什么特殊的顺

序，但是它们都必须早于数据采样点 C 点之前至少半个 USCK 周期。这点必须得到保障，以保证数据准备所需条件得到满足。 4 位计数器被重置为 0。

2. 主机通过软件改变USCK两次(C与D)来产生一个时钟脉冲。主从设备的输入引脚(DI)上的值由USI在第一个沿(C)进行采样；数据输出则在其相对的沿(D)改变。4位计数器将对两个沿进行计数。
3. 在一个完整的寄存器 (字节) 传输过程中，第 2 步将重复 8 次。
4. 8 个时钟脉冲 (16 个时钟沿) 之后 ,计数器溢出 ,表明传输完成。传输的数据必须在下一次传输开始之前得到处理。如果处理器处于空闲模式，那么溢出中断会将它唤醒。依据通讯协议，从机现在可以将它的输出置为高阻状态。

SPI 主机工作例子

接下来的代码说明了如何将 USI 模块当作 SPI 主机来使用：

```
SPITransfer:
    out    USIDR,r16
    ldi    r16,(1<<USIOIF)
    out    USISR,r16
    ldi    r16,(1<<USIWM0)|(1<<USICS1)|(1<<USICLK)|(1<<USITC)
SPITransfer_loop:
    out    USICR,r16
    sbis   USISR,USIOIF
    rjmp   SPITransfer_loop
    in     r16,USIDR
    ret
```

这段代码仅使用了 8 条指令 (+ ret)，非常优化。示例代码假定 DO 及 USCK 引脚已经通过设置 DDRE 寄存器成为输出引脚。调用函数之前， r16 寄存器包含了要送到从机的数据，传输结束之后， r16 寄存器包含了从从机接收回来的数据。

第二和第三条指令是清 USI 计数器溢出标志及 USI 计数器。第四第五条指令设置三线模式、上升沿移位寄存器时钟、在 USITC 选通时进行计数以及触发 USCK。循环将重复运行 16 次。

下面的代码则演示了在最大速率 (fsck = fck/2) 下如何将 USI 模块作为 SPI 主机来使用：

```
SPITransfer_Fast:

    out    USIDR,r16
    ldi    r16,(1<<USIWM0)|(0<<USICS0)|(1<<USITC)
    ldi    r17,(1<<USIWM0)|(0<<USICS0)|(1<<USITC)|(1<<USICLK)

    out    USICR,r16 ; MSB
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16 ; LSB
    out    USICR,r17

    in     r16,USIDR
ret
```

SPI 从机工作例子

接下来的代码展示了如何将 USI 模块作为 SPI 从机来使用：

```
init:
    ldi    r16,(1<<USIWM0)|(1<<USICS1)
    out    USICR,r16
...
SlaveSPITransfer:
    out    USIDR,r16
    ldi    r16,(1<<USIOIF)
    out    USISR,r16
SlaveSPITransfer_loop:
    sbis   USISR,USIOIF
    rjmp   SlaveSPITransfer_loop
    in     r16,USIDR
    ret
```

这段代码仅使用了 8 条指令 (+ ret)，非常优化。示例代码假定 DO 及 USCK 引脚已经通过设置 DDR 寄存器分别成为输出和输入引脚。调用函数之前， r16 寄存器包含了要送到主机的数据，传输结束之后， r16 寄存器包含了从主机接收回来的数据。

开头的两条指令为初始化指令，仅需执行一次。这些指令用来设置三线模式及上升沿沿移位寄存器时钟。循环一直重复到 USI 计数器溢出标志位置位。

两线模式

USI 两线模式兼容 IC 间 (TWI) 总线协议，但没有输出转换速率限制及输入噪声滤波器。这种模式下的引脚名为 SCL 和 SDA。

Figure 62. 两线模式框图

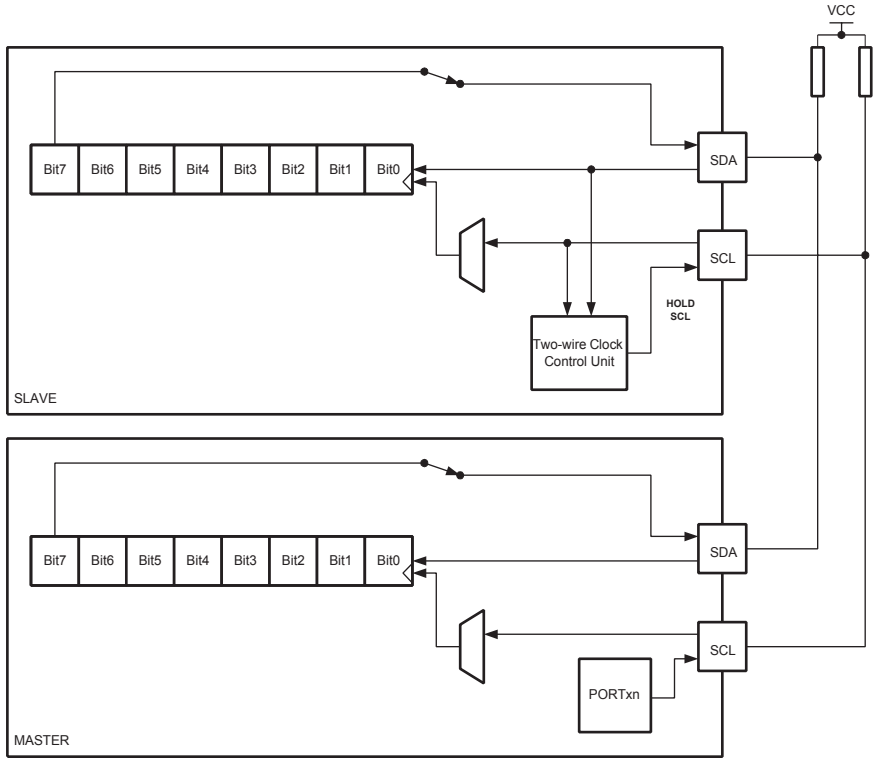
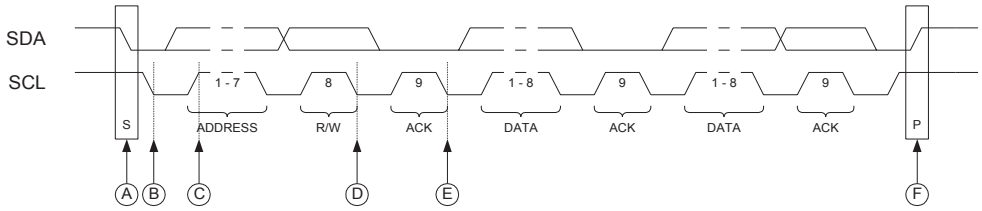


Figure 62 展示了两个工作在两线模式下的 USI 单元，一个为主机，另一个为从机。由于系统的工作在很大程度上取决于所使用的通信方案，这里仅仅给出了物理层。在这一层中，主机和从机的主要区别是：串行时钟由主机产生，只有从机使用时钟控制单元。时钟的产生必须由软件实现，但移位操作是自动完成的。数据传输过程中只有下降沿触发数据移位。通过强制 SCL 时钟为低，从机可以在传输启动或结束时插入等待状态。这说明，产生上升沿之后，主机必须检查 SCL 线是否已经释放。

由于时钟同时驱动计数器计数，计数器溢出可用来判断传输是否结束。主机通过 PORT 寄存器使 USCK 引脚产生时钟。

物理层不确定数据的方向。数据流控制由协议决定，如 TWI 总线协议。

Figure 63. 两线模式典型时序图



根据此时序图 (Figure 63.)，总线传输包括以下步骤：

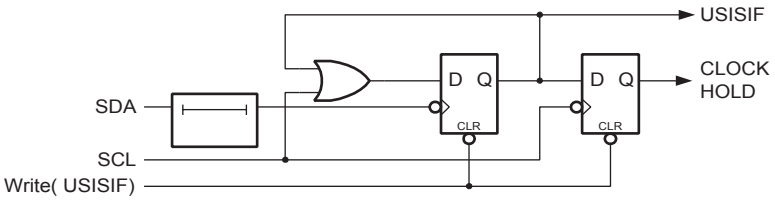
1. 当SCL为高时(A)，主机可通过将SDA强制拉低来产生一个起始条件。有两种方法可使 SDA 强制为低：一种是对移位寄存器的第 7 位写 0，另一种是将 PORT 寄存器中的对应位置 0。在此之前还需要通过数据方向寄存器将相关引脚设置为输 出。

从机起始条件监测逻辑 (Figure 64.) 检测到起始条件后置位 USISIF 标志。如果有必要，可通过此标志产生中断。

2. 此外，当主机强制在 SCL 线产生下降沿 (B) 后，起始条件检测器将保持 SCL 为低。这可将从机从睡眠状态唤醒，或在设置移位寄存器接收地址之前完成其他任务。这个操作通过清起始条件标志及复位计数器来实现。
3. 主机设置第一个需要传输的位，并释放 SCL 线 (C)。从机在 SCL 时钟上升沿对数据进行采样并将数据转移到串行寄存器中。
4. 当包含从机地址及数据方向 (读或写) 的 8 位数据全都传输完完毕之后，从机计数器溢出， SCL 被强制置为低 (D)。如果这个从机不是主机所寻址的设备，它将释放 SCL 线并等待下一个起始条件。
5. 如果从机被寻址，那么在 SCL 被再次拉低之前 (在释放 SCL(D) 之前计数器要达到 14)，在应答期间它将保持 SDA 线为低。 R/W 位决定是主机还是从机输出数据。若 R/W 为 1，主机执行读操作 (即从设备驱动 SDA 线)。在应答 (E) 之后从机可以保持 SCL 线为低。
6. 现在可以在同一方向传输多个字节的数据了，直到主机发出停止条件 (F)，或者产生一个新的起始条件。

如果从机无法接收更多的数据，那么它不要应答最后接收到的数据。主机进行读操作时，接收到最后一个字节后，它必须强制应答位为低来结束读操作。

Figure 64. 起始条件监测器逻辑电路图



起始条件检测器

Figure 64. 为起始条件监测器。SDA 被延迟 (50 到 300 ns) , 以保证 SCL 的有效采样。起始条件监测器仅在两线模式下使能。

起始条件监测器工作在异步模式，因此可以将处理器从掉电睡眠模式唤醒。但是，通讯协议可能对 SCL 的保持时间有限制。在这种情况下就必须考虑由 CKSEL 熔丝位确定的晶振启动时间，见 P21“ 时钟系统及其分布 ”。

其他 USI 用法

如果 USI 不用于串行通讯，由于其设计上的灵活性，可以用来完成其他工作。

半双工异步数据传输

在三线模式下使用移位寄存器可以实现比软件方案更紧凑、更高效的 UART 功能。

4 位计数器

这个 4 比特计数器可作为独立的计数器来使用，并可产生溢出中断。要注意的是，如果计数器由外部时钟源提供时钟，那么两个时钟边沿都会使其计数。

12 位定时器 / 计数器

将 USI 的 4 比特计数器与定时器 / 计数器 0 结合起来使用可得到一个 12 位的计数器。

边沿触发的外部中断

把计数器的值设到最大 (F)，可实现一个额外的外部中断。溢出标志位及中断使能位都为此外部中断服务。可通过 USICS1 来选择这一特性。

软件中断

计数器溢出中断还可用作软件中断。

USI 寄存器说明

USI 数据寄存器 - USIDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	USIDR
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	

USI 串行寄存器没有缓冲 , 寻址数据寄存器 USIDR 时 , 实际操作的是串行寄存器。若在写寄存器的同一个周期产生了一个串行时钟，寄存器将被更新，但不会进行移位操作。移位操作依赖于 USICS1..0 的设置，可由外部时钟沿、定时器 / 计数器 0 比较匹配或直接通过软件使用 USICLK 来控制。即使没有选择任何连接模式 (USIWM1..0 = 0)，移位寄存器仍然可以使用外部数据输入 (DI/SDA) 及外部时钟输入 (USCK/SCL)。

输出引脚 (DO 或 SDA，由连接模式确定) 通过输出锁存器与数据寄存器的最高位 (位 7) 相连。选择了外部时钟源时 (USICS1 = 1) , 输出锁存器在串行时钟的前半个周期打开 (透明)，如果使用的是内部时钟源 (USICS1 = 0)，它将一直打开。锁存器打开时写入新的 MSB 会立即在输出引脚反映出来。锁存器保证输入数据的采样时间与输出数据的改变发生在相反的时钟沿。

为了输出移位寄存器的数据，必须利用方向寄存器将对应的引脚设置为输出。

USI 状态寄存器 - USISR

Bit	7	6	5	4	3	2	1	0	
	USISIF	USIOIF	USIPF	USIDC	USICNT3	USICNT2	USICNT1	USICNT0	USISR
读 / 写	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

状态寄存器包括中断标志、线状态标志及计数器值。

- **Bit 7 – USISIF: 起始状态中断标志**

在两线模式下，检测到起始条件将置位 USISIF 标志。当使用输出禁止模式或三线模式，并且 (USICSx = 0b11 & USICLK = 0) 或 (USICS = 0b10 & USICLK = 0) 时， SCK 引脚的任意电平变化都会使此标志置位。

USISIF 置位时，若 USICR 寄存器的 USISIE，以及全局中断使能标志也置位，则会产生中断。 USISIF 标志位清零的唯一方式是对其写入逻辑 1。在两线模式下清除这一标志会释放由于检测到起始条件而保持的 USCL。

起始条件中断可把处理器从所有睡眠状态唤醒。

- **Bit 6 – USIOIF: 计数器溢出中断标志**

4位计数器溢出(即从15跳变到0)时此标志置位。若USICR寄存器的USIOIE，以及全局中断使能标志也置位，则会产生中断。 USIOIF 标志位清零的唯一方式是对其写入逻辑 1。在两线模式下清这一标志会释放由于计数器溢出而保持的 SCL。

溢出中断可以将处理器从所有睡眠模式唤醒。

- **Bit 5 – USIPF: 停止状态标志**

在两线模式下，如果检测到停止状态， USIPF 标志置位。 USIOIF 标志位清零的方法是对其写入逻辑 1。注意这不是一个中断标志位。在进行总线主机仲裁时，可使用这个标志。

- **Bit 4 – USIDC: 数据输出冲突**

如果移位寄存器中的位 7 与物理引脚所对应的值不同， USIDC 置位。此标志仅在两线模式下有效。在进行总线主机仲裁时，可使用这个标志。

- **Bits 3..0 – USICNT3..0: 计数器值**

反映的是 4 位计数器的当前值。 CPU 可以直接读写这几位。

使计数器计数的时钟源有外部时钟边沿检测器、定时器 / 计数器 0 比较匹配及通过软件使用 USICLK 或 USITC 产生的时钟源。时钟源的确定由 USICS1..0 设置。外部时钟操作另有一个特性，即可以通过写 USITC 来产生时钟。方法是设置外部时钟源 (USICS1 = 1) 时将 USICLK 置 1。

即使没有选择任何连接模式 (USIWM1..0 = 0)，计数器仍然可以使用外部时钟输入 (USCK/SCL)。

USI 控制寄存器 - USICR

Bit	7	6	5	4	3	2	1	0	
	USISIE	**USIOIE**	**USIWM1**	**USIWM0**	**USICS1**	**USICS0**	**USICLK**	**USITC**	**USICR**
读 / 写	R/W	R/W	R/W	R/W	R/W	R/W	W	W	
初始值	0	0	0	0	0	0	0	0	

控制寄存器包括中断使能控制、连接模式设置、时钟选择设置及时钟选择信号。

- **Bit 7 – USISIE: 起始条件中断使能**

将此位置 1 可使能起始条件检测中断。如果 USISIE 及全局中断使能标志位置位时产生了一个这样的中断，那么中断将立即执行。

- **Bit 6 – USIOIE: 计数器溢出中断使能**

此标志为 1 可产生计数器溢出中断。如果 USIOIE 及全局中断使能标志位置位时产生了一个这样的中断，那么中断将立即执行。

- **Bit 5..4 – USIWM1..0: 连接模式**

这几位用来设置连接模式。基本上只有输出功能受这几位的影响。数据及时钟输入不受所选模式的影响，总是保持同样的功能。因此，即使输出被禁止，计数器和移位寄存器照样可由外部提供时钟，也可以进行数据输入采样。 USIWM1..0 与 USI 操作的关系在 P139Table 60 中有简要介绍。

Table 60. USIWM1..0 与 USI 操作之间的关系

USIWM1	USIWM0	说明
0	0	输出，时钟保持，起始检测器禁止。引脚以普通端口方式工作。
0	1	三线模式。使用 DO、 DI 及 USCK 引脚 在这种模式下*数据输出* (DO) 功能取代了普通端口 IO 功能，但对应的 DDR 仍然控制这数据方向。端口引脚设置为输入时，引脚的上拉电阻由 PORT 位来控制。 *数据输入* (DI) 及*串行时钟* (USCK) 功能不影响正常的端口功能。作为主机工作时，软件通过操作 PORT 寄存器来产生时钟脉冲，同时数据方向设为输出。 USICR 寄存器中的 USITC 位可用作这一目的。
1	0	两线模式。使用 SDA (DI) 及 SCL (USCK) 引脚 (1)。 *串行数据* (SDA) 及*串行时钟* (SCL) 引脚是双向的，且使用集电极开路输出驱动器。通过设置 DDR 寄存器中相应的位来启动输出驱动器。 SDA 引脚的驱动器使能时，如果移位寄存器的输出或 PORT 寄存器对应的位为 0，那么输出驱动器会把 SDA 线强制拉低。否则 SDA 线将不被驱动 (即将它释放)。 SCL 引脚输出驱动器使能时，如果 PORT 寄存器中的对应位为 0，或者由于起始检测器的作用， SCL 线被强制置低。否则 SCL 线将不被驱动。 当起始检测器检测到起始条件且输出允许时， SCL 被拉低。清起始条件标志 (USISIF) 将释放此口线。启动这一模式不会影响 SDA 及 SCL 引脚的输入。在两线模式下 SDA 及 SCL 引脚的上拉无效。
1	1	两线模式。使用 SDA 及 SCL 引脚 两线模式下的一些操作在上面已有所描述，除了以下这点：当发生计数器溢出时 SCL 线保持为低，并一直保持到计数器溢出标志位 (USIOIF) 被清零。

Note: 1. 为了避免混淆，DI 及 USCK 引脚分别被重命名为*串行数据* (SDA) 及*串行时钟* (SCL)。

- **Bit 3..2 – USICS1..0: 时钟源选择**

通过这几位可以设置移位寄存器及计数器的时钟源。数据输出锁存器保证在使用外部时钟 (USCK/SCL) 时，输出数据的改变与输入数据 (DI/SDA) 的采样发生在相反的时钟沿。如果选择了软件方式或定时器 / 计数器 0 比较匹配作为时钟，输出锁存器即成为是透明的，输出可以立即改变。 USICS1..0 为 0 时软件方式使能。此时，向 USICLK 写 1 就可以同时给移位寄存器和计数器提供时钟。对于外部时钟 (USICS1 = 1)， USICLK 位不再用作选通信号，而是通过 USITC 在外部时钟及软件时钟之间进行选择。

Table 61 给出了 USICS1..0 及 USICLK 的设置与移位寄存器及 4 位计数器所使用的时钟之间的关系。

Table 61. USICS1..0 与 USICLK 的设置

USICS1	USICS0	USICLK	移位寄存器时钟源	4 位计数器时钟源
0	0	0	无时钟	无时钟
0	0	1	软件时钟 (USICLK)	软件时钟 (USICLK)
0	1	X	定时器 / 计数器 0 比较匹配	定时器 / 计数器 0 比较匹配
1	0	0	外部时钟，上升沿	外部时钟，上升及下降沿
1	1	0	外部时钟，下降沿	外部时钟，上升及下降沿
1	0	1	外部时钟，上升沿	软件时钟 (USITC)
1	1	1	外部时钟，下降沿	软件时钟 (USITC)

- **Bit 1 – USICLK: 时钟选通**

若 USICS1..0 为 0，置位 USICLK 将使移位寄存器进行一次移位，计数器累加一。此即为软件时钟。在同一指令周期里输出立即得到更新。移入移位寄存器的值在上一个指令周期得以采样。这一位的读出值为 0。

使用外部时钟时 (USICS1 = 1)， USICLK 的功能由时钟选通变为时钟选择寄存器。在这种情况下设置 USICLK 将选择 USITC 作为软件时钟源来驱动 4 位计数器 (见 Table 61)。

- **Bit 0 – USITC: 交替变换时钟端口引脚**

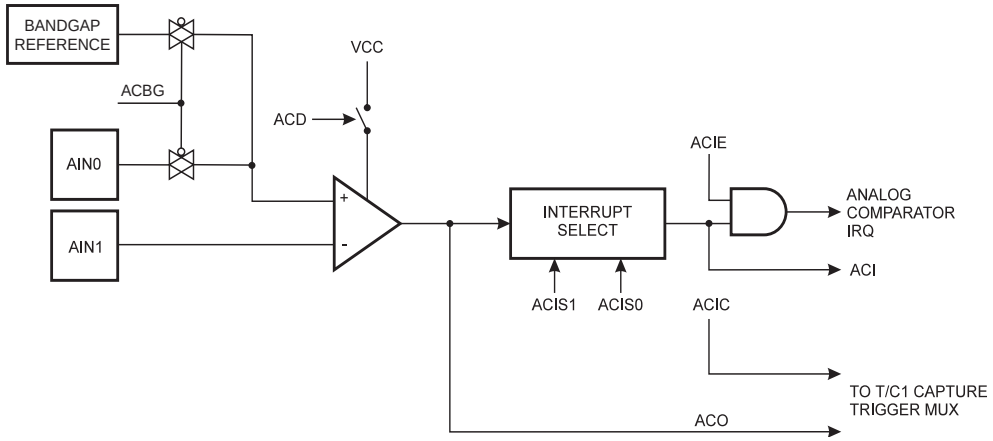
USITC 置位将使 USCK/SCL 出现 0、1 的交替变换。方向寄存器的设置不影响触发，但需要置位 DDRE4 使相应的端口成为输出。这个特性为主机实现提供了一个产生时钟的简单办法。这一位的读返回值为 0。

选用外部时钟 (USICS1 = 1) 并且 USICLK 置 1 时，对 USITC 进行写入将直接驱动 4 位计数器。作为主机工作时，这样可以更早地知道传输何时结束。

模拟比较器

模拟比较器对正极 AIN0 的值与负极 AIN1 的值进行比较。当 AIN0 上的电压比负极 AIN1 上的电压要高时，模拟比较器的输出 ACO 即置位。比较器的输出可用来触发定时器 / 计数器 1 的输入捕捉功能。此外，比较器还可触发自己专有的、独立的中断。用户可以选择比较器是以上升沿、下降沿还是交替变化的边沿来触发中断。Figure 65 为比较器及其外围逻辑电路的框图。

Figure 65. 模拟比较器框图



模拟比较器控制及状态寄存器 - ACSR

Bit	7	6	5	4	3	2	1	0	
	ACD	**ACBG**	**ACO**	**ACI**	**ACIE**	**ACIC**	**ACIS1**	**ACIS0**	**ACSR**
读 / 写	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	N/A	0	0	0	0	0	

- **Bit 7 – ACD: 模拟比较器禁用**

ACD 置位时，模拟比较器的电源被切断。可以在任何时候设置此位来关掉模拟比较器。这可以减少器件工作模式及空闲模式下的功耗。改变 ACD 位时，必须清零 ACSR 寄存器的 ACIE 位来禁止模拟比较器中断。否则 ACD 改变时可能会产生中断。

- **Bit 6 – ACBG: 模拟比较器的能隙基准源选择**

ACBG 置 "1" 后，模拟比较器的正极输入由固定能隙基准源所取代。ACBG 清零，AIN0 作为模拟比较器的正极输入，见 P36 “ 片内基准电压 ” 。

- **Bit 5 – ACO: 模拟比较器输出**

模拟比较器的输出为同步信号，直接连到 ACO。同步加入 1 - 2 时钟周期的延迟。

- **Bit 4 – ACI: 模拟比较器中断标志**

当比较器的输出事件触发了由 ACIS1 及 ACIS0 定义的中断模式时，ACI 由硬件置位。如果 ACIE 和 SREG 寄存器的全局中断标志 I 也置位，那么模拟比较器中断服务程序即得以执行，同时 ACI 被硬件清零。 ACI 也可以通过写 1 来清零。

- **Bit 3 – ACIE: 模拟比较器中断使能**

当 ACIE 位被置 1 且状态寄存器中的全局中断标志 I 也被置位时，模拟比较器中断被激活。否则中断被禁止。

- **Bit 2 – ACIC: 模拟比较器输入捕获使能**

当该位置位， T/C1 的输入捕获功能被模拟比较器触发。此时比较器输出直接与输入捕获前端逻辑连接，使比较器利用 T/C1 输入捕获中断的噪声抑制模式与边沿选择特性。当该位清零，模拟比较器与输入捕获功能间没有连接。为使比较器触发 T/C1 输入捕获中断，定时器屏蔽寄存器 (TIMSK) 的 ICIE1 位必须置位。

- **Bits 1, 0 – ACIS1, ACIS0: 模拟比较器中断模式选择**

这两位确定哪个事件可以触发模拟比较器中断。 Table 62 给出了不同的设置。

Table 62. ACIS1/ACIS0 设置

ACIS1	ACIS0	中断模式
0	0	比较器输出变化即可触发中断
0	1	保留
1	0	比较器输出的下降沿产生中断
1	1	比较器输出的上升沿产生中断

当改变 ACIS1/ACIS0 位时，必须通过清除 ACSR 寄存器中断使能位禁用模拟比较中断。否则当位变化时可能会出现中断。

数字输入禁用寄存器 - DIDR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	–	AIN1D	AIN0D	DIDR
读 / 写	R	R	R	R	R	R	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

- **Bit 1, 0 – AIN1D, AIN0D: AIN1, AIN0 数字输入禁用**

当该位写 "1”，AIN1/0 引脚的数字输入缓冲禁用。相应的引脚寄存器位为零。当 AIN1/0 引脚输入为模拟信号且不使用数字输入，该位写 "1” 以降低数字输入缓冲的功耗。

debugWIRE 片上调试系统

特性

- 完全的程序流控制
- 仿真芯片上除 RESET 引脚外所有的模拟和数字功能
- 实时操作
- 支持符号调试 (C 与汇编级，或其它 HLL)
- 没有限制的程序断点数 (使用软件断点)
- 非插入式操作
- 与实际器件相同的电气特性
- 自动配置系统
- 高速操作
- 编程非易失性存储器

概述

debugWIRE 片上调试系统使用单线双向接口来控制程序流，在 CPU 中执行 AVR 指令，对不同的非易失性存储器进行编程。

物理接口

当 debugWIRE 使能熔丝位 DWEN 被编程且锁定位未编程时，目标器件中的 debugWIRE 系统被激活。RESET 端口引脚配置为上拉使能的线与 (开漏) 双向 I/O，成为目标与仿真器间的联系通路。

Figure 66. debugWIRE 设置

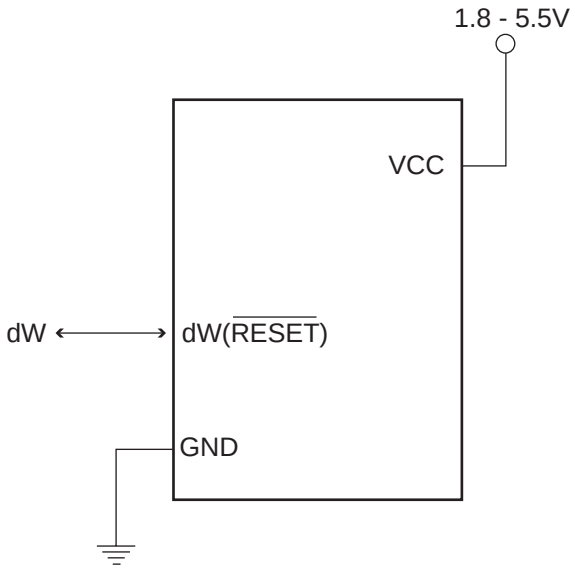


Figure 66 给出 debugWIRE 使能的目标 MCU 及仿真连接器的示意图。系统时钟不受 debugWIRE 的影响，只由 CKSEL 熔丝位决定。

设计使用 debugWIRE 的系统时，必须进行下面的检查：

- dW/(RESET) 的上拉电阻不得小于 10kΩ。debugWIRE 并不需要上拉电阻。
- 将 RESET 引脚与 V_{CC} 直接连接将无法工作。
- 使用 debugWIRE 时必须断开与 RESET 引脚连接的电容。
- 必须断开所有的外部复位源。

软件断点

debugWIRE 通过 AVR 断点指令来设置程序存储器断点。在 AVR Studio® 设置一个断点将在程序存储器中插入 BREAK 指令。被 BREAK 指令所替代的指令将被保存。程序继续运行时，保存的指令得到执行，然后继续执行其他指令。断点也可以通过在程序中插入 BREAK 指令进行手工设置。

每次断点改变后 Flash 必须要重新编程。这由 AVR Studio® 通过 debugWIRE 接口自动处理。断点的使用会降低 Flash 数据记忆时间。调试用的器件不能发给最终客户。

debugWIRE 的局限

debugWIRE 通讯引脚 (dW) 与外部复位 (RESET) 共用同一引脚。因此使能 debugWIRE 之后，系统不支持外部复位源。

当程序在 CPU 中全速运行时， debugWIRE 系统精确的仿真所有的 I/O 口功能；当 CPU 停止工作时，通过调试器访问某些 I/O 寄存器时要注意。详见 debugWIRE 文档。

DWEN 熔丝位的编程使部分时钟系统在所有的休眠模式下都保持运行。这会增加器件休眠模式的功耗。因此不使用 debugWire 时应该禁用 DWEN 熔丝位。

I/O 存储器中与 debugWIRE 相关的寄存器

下面说明在 debugWire 中用到的寄存器。

debugWire 数据寄存器 - DWDR

Bit	7	6	5	4	3	2	1	0	
	DWDR[7:0]								DWDR
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	

DWDR 寄存器为在 MCU 中运行的程序与调试器提供了通信通路。该寄存器只能由 debugWIRE 访问且不能在通常操作中作为通用寄存器使用。

Flash 自编程

器件为通过 MCU 本身来下载和上载程序代码提供了一个自编程机制。自编程可以使用任何器件可用的数据接口和相关的协议来获得代码并把代码 (程序) 写入程序存储器。

程序存储器的更新以页的方式进行。在用临时页缓冲器存储的数据对一页存储器进行编程之前首先要将这一页擦除。 SPM 指令以一次一个字的方式将数据写入临时页缓冲器。临时页缓冲器的写入可以在页擦除命令之前完成，也可以在页擦除和页写操作之间完成。

方案 1，在页擦除前填充缓冲器：

- 填充临时页缓冲器
- 执行页擦除操作
- 执行页写操作

方案 2，在页擦除之后填充缓冲器：

- 执行页擦除操作
- 填充临时页缓冲器
- 执行页写操作

如果只需要改变页中的一部分，擦除前必须保存页中的其它部分 (例如保存于临时页缓冲器之中)，再重新写入。使用方案 1 时， Boot Loader 提供了有效的读 - 该 - 写操作，允许用户先读页，做必要的改变，再写回修改后的数据；若使用方案 2，由于页已被擦除，因此不可能在加载数据时读取旧的数据。临时页缓冲器可以进行随机访问。进行页擦除与页写操作时要确保地址是相同的。

通过 SPM 完成页擦除

执行页擦除操作首先需要设置 Z 指针的地址信息，然后将 “00000011” 写入 SPMCSR，最后在其后的四个时钟周期内执行 SPM。R1 和 R0 中的数据被忽略。页地址必须写入 Z 寄存器的 PCPAGE。 Z 指针的其他位被忽略。

- 在擦除操作过程中 CPU 停止。

写临时缓冲区 (页加载)

写一个指令字首先需要设置 Z 指针的地址信息，以及将指令字写入 R1:R0，然后将 “00000001” 写入 SPMCSR，最后在其后的四个时钟周期内执行 SPM。 Z 寄存器中 PCWORD的内容用来寻址临时缓冲区。页写操作完成，或置位 SPMCSR寄存器的CTPB位将使临时缓冲区自动擦除。系统复位也会擦除临时缓冲区。但是如果不清除临时缓冲区就只能对每个地址进行一次写操作。

如果在 SPM 页加载操作过程中对 EEPROM 执行了写操作，则所有加载的数据都将丢失。

执行页写操作

执行页写操作首先需要设置 Z 指针的地址信息，然后将 “00000101” 写入 SPMCSR，最后在其后的四个时钟周期内执行 SPM。R1 和 R0 中的数据被忽略。页地址必须写入 Z 寄存器的 PCPAGE。 Z 指针的其他位被忽略。

- 页写过程中 CPU 停止。

在自编程过程中寻址 Flash

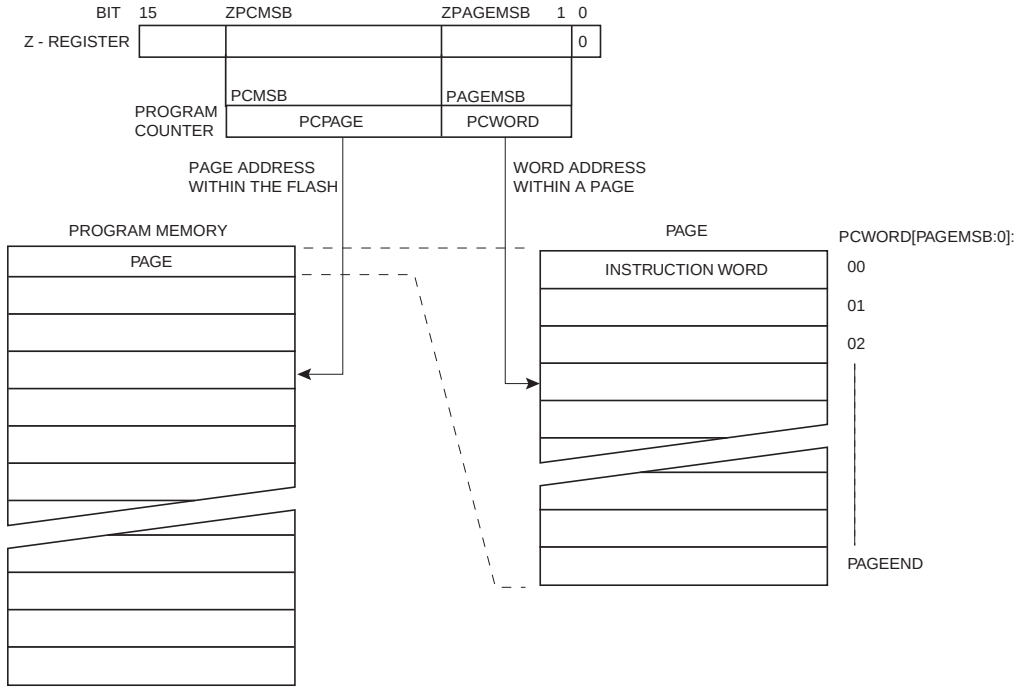
Z 指针用来寻址 SPM 命令。

Bit	15	14	13	12	11	10	9	8
ZH (R31)	Z15	Z14	Z13	Z12	Z11	Z10	Z9	Z8
ZL (R30)	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0
	7	6	5	4	3	2	1	0

由于 Flash 存储器是以页的形式组织 (见 P152Table 69) 起来的 , 程序计数器可看作由两个部分构成 : 其一为实现页内寻址的低位部分 ; 其次为实现页寻址的高位部分 , 如 Figure 67 所示。由于页擦除和页写操作的寻址是相互独立的 , 因此保证 Boot Loader 软件在页擦除和页写操作时寻址相同的页是最重要的。

LPM指令也使用Z指针来保存地址。由于这个指令的寻址逐字节地进行 , 所以Z指针的LSB位 (位 Z0) 也使用到了。

Figure 67. SPM 的寻址 (1)



Note: 1. Figure 67 中所用的不同的变量在 P152Table 69 列出。

存贮程序存储器 (SPM) 控制和状态寄存器 - SPMCSR

SPMCSR 包括了控制 Boot Loader 操作所需的控制位。

Bit	7	6	5	4	3	2	1	0	
	–	–	–	CTPB	RFLB	PGWRT	PGERS	SELFPRGEN	SPMCSR
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	

- **Bits 7..5 – Res: 保留**

在 ATtiny2313 中为保留位，读返回值为 "0”。

- **Bit 4 – CTPB: 清除临时页缓冲**

当临时页缓冲器写入时， CTPB 位写入，临时页缓冲器将清除，数据将丢失。

- **Bit 3 – RFLB: 读熔丝位与锁定位**

SPMCSR 寄存器中的 RFLB 与 SELFPRGEN 位置位后的三个时钟周期内执行 LPM 指令，将锁定位或熔丝位读入目的寄存器 (由 Z 指针的 Z0 位决定) , 见 P148“EEPROM 写操作阻止对 SPMCSR 寄存器的写操作 ” 。

- **Bit 2 – PGWRT: 页写入**

如果这一位和 SELFPRGEN 同时置位，发生于紧接着的四个时钟周期内的 SPM 指令执行页写功能 , 将临时缓冲器中存储的数据写入 Flash。页地址取自 Z 指针的高位部分。R1 和 R0 的数据则被忽略。页写操作完成，或在四个时钟周期内没有 SPM 指令被执行时，PGWRT 自动清零。在整个页写操作过程中 CPU 停止。

- **Bit 1 – PGERS: 页擦除**

如果这一位和 SELFPRGEN 同时置位，发生于紧接着的四个时钟周期内的 SPM 指令执行页擦除功能。页地址取自 Z 指针的高位部分。 R1 和 R0 的数据则被忽略。页擦除操作完成，或在四个时钟周期内没有 SPM 指令被执行时， PGERS 自动清零。在整个页擦除操作过程中 CPU 停止。

- **Bit 0 – SELFPRGEN: 自编程使能**

这一位使能紧接着的四个时钟周期内的 SPM 指令。如果将这一位和 CTPB、 RFLB、PGWRT 或 PGERS 之一同时置位 , 则如上所述 , 接下来的 SPM 指令将有特殊的含义。如果只有 SELFPRGEN 置位，那么接下来的 SPM 指令将把 R1:R0 中的数据存储到由 Z 指针确定的临时页缓冲器。Z 指针的 LSB 被忽略。SPM 指令完成 , 或在四个时钟周期内没有 SPM 指令被执行时， SPMEN 自动清零。在页擦除和页写过程中 SELFPRGEN 保持为高直到操作完成。

在低五位中写入除 “10001”、“01001”、“00101”、 “00011” 或 “00001” 之外的任何组合都无效。

EEPROM 写操作阻止对 SPMCSR 寄存器的写操作

EEPROM 写操作会阻塞对 Flash 的编程，也会阻塞对熔丝位和锁定位的读操作。建议用户在对 SPMCSR 寄存器进行写操作之前首先检查 EECR 寄存器的状态位 EEWE，确保此位已被清除。

通过软件读取熔丝位和锁定位

熔丝位和锁定位可以通过软件读取。读锁定位时，需要将 0x0001 赋予给 Z 指针并且置位 SPMCSR 寄存器的 RFLB 和 SELFPRGEN。在 RFLB 和 SELFPRGEN 置位后的三个 CPU 周期内执行的 LPM 指令将把锁定位的值将加载到目的寄存器。读锁定位操作结束，或者在三个 CPU 周期内没有 执行 LPM 指令，或在四个 CPU 周期内没有执行 SPM 指令，RFLB 和 SELFPRGEN 位将自动硬件清零。RFLB 和 SELFPRGEN 清零后，LPM 将按照指令手册中所描述的那样工作。

Bit	7	6	5	4	3	2	1	0
Rd	–	–	–	–	–	–	LB2	LB1

读取熔丝位低字节的算法和上述读取锁定位的算法类似。要读取熔丝位低字节，需要将 0x0000 赋予给 Z 指针并且置位 SPMCSR 寄存器的 RFLB 和 SELFPRGEN。在 SPMCSR 操作之后的三个 CPU 周期内执行的 LPM 指令将把熔丝位低位字节的值 (FLB) 加载到目的寄存器。更详细的说明及熔丝位低位字节映射的细节请参见 P152Table 68 。 .

Bit	7	6	5	4	3	2	1	0
Rd	FLB7	FLB6	FLB5	FLB4	FLB3	FLB2	FLB1	FLB0

类似的，读取熔丝位高位字节时，需要将 0x0003 赋予给 Z 指针并且置位 SPMCSR 寄存器的 RFLB 和 SELFPRGEN。在 SPMCSR 操作之后的三个 CPU 周期内执行的 LPM 指令将把熔丝位高位字节的值 (FHB) 加载到目的寄存器。

Bit	7	6	5	4	3	2	1	0
Rd	FHB7	FHB6	FHB5	FHB4	FHB3	FHB2	FHB1	FHB0

被编程的熔丝位 / 锁定位的读返回值为 "0”。未被编程的熔丝位 / 锁定位的读返回值为 "1”。

防止 Flash 损毁

V_{CC} 低于工作电压时 ,CPU 和 Flash 正常工作无法保证 ,Flash 的内容可能受到破坏。这个问题在板级系统的独立 Flash 中一样存在。所以也要采用同样的解决方案。

电压太低时有两种情况可以破坏 Flash 内容。第一，Flash 写过程需要一个最低电压。第二，电压太低时 CPU 本身会错误地执行指令。

通过遵循以下设计建议可以避免 Flash 被破坏 (采用其中之一就足够了) :

1. 电源电压不足期间，保持 AVR RESET 为低：采用的方式为：如果工作电压与检测电平相匹配，可以使能 BOD 功能；否则可以使用外部复位保护电路。如果在写操作进行中发生了复位，只要电源电压足够，写操作还会完成。
2. 低电压期间保持 AVR 内核处于掉电休眠模式。这样可以防止 CPU 解码并执行指令，有效地保护 SPMCSR 寄存器，从而保护 Flash 被无意识得修改掉。

使用 SPM 时的 Flash 编程时间

片内校准的 RC 振荡器用于 Flash 寻址时序控制。Table 63 给出了 CPU 访问 Flash 的典型编程时间。

Table 63. SPM 编程时间

符号	最小编程时间	最大编程时间
Flash 写操作 (通过 SPM 实现页擦除、页写、及写锁定位)	3.7 ms	4.5 ms

存储器编程

程序存储器和数据存储器锁定位

ATtiny2313 提供了 2 个锁定位 , 根据其被编程 (“0”) 还是没有被编程 (“1”) 的情况可以获得 Table 65 列出的附加性能。锁定位只能通过芯片擦除命令擦写为 “1”。

Table 64. 锁定位字节 (1)

锁定位字节	位号	描述	默认值
	7	–	1 (未编程)
	6	–	1 (未编程)
	5	–	1 (未编程)
	4	–	1 (未编程)
	3	–	1 (未编程)
	2	–	1 (未编程)
LB2	1	锁定位	1 (未编程)
LB1	0	锁定位	1 (未编程)

Note: 1. “1” 表示未编程 , “0” 表示已编程。

Table 65. 锁定位保护模式 (1)(2)

存储器锁定位			保护类型
LB 模式	LB2	LB1	
1	1	1	没有使能存储器保护特性
2	1	0	在并行和串行编程模式中 Flash 和 EEPROM 的进一步编程被禁止，熔丝位被锁定。(1)
3	0	0	在并行和串行编程模式中 Flash 和 EEPROM 的进一步编程及验证被禁止，锁定位和熔丝位被锁定 (1)

Notes: 1. 在编程 LB1 和 LB2 前先编程熔丝位和 Boot 锁定位。
2. “1” 表示未被编程 , “0” 表示已编程。

熔丝位

ATtiny2313 有三个熔丝位字节。Table 67 和 Table 68 简单地描述了所有熔丝位的功能以及他们是如何映射到熔丝字节的。如果熔丝位被编程则读返回值为 “0”。

Table 66. 熔丝位扩展字节

熔丝位扩展字节	位号	描述	默认值
	7	–	1 (未编程)
	6	–	1 (未编程)
	5	–	1 (未编程)
	4	–	1 (未编程)
	3	–	1 (未编程)
	2	–	1 (未编程)
	1	–	1 (未编程)
SELFPRGEN	0	自编程使能	1 (未编程)

Table 67. 熔丝位高位字节

熔丝位高位字节	位号	描述	默认值
DWEN(3)	7	debugWIRE 使能	1 (未编程)
EESAVE	6	执行芯片擦除时 EEPROM 的内容保留	1 (未编程，EEPROM 内容不保留)
SPIEN(1)	5	使能串行程序和数据下载	0 (已编程，SPI 编程使能)
WDTON(2)	4	看门狗定时器一直启用	1 (未编程)
BODLEVEL2(4)	3	BOD 触发电平	1 (未编程)
BODLEVEL1(4)	2	BOD 触发电平	1 (未编程)
BODLEVEL0(4)	1	BOD 触发电平	1 (未编程)
RSTDISBL(5)	0	外部复位禁用	1 (未编程)

Note:
1. 在串行编程模式下 SPIEN 熔丝位不可访问。
2. 详见 P40“ 看门狗定时器控制寄存器 - WDTCSR” 。
3. 出厂时不对 DWEN 熔丝位编程。对 DWEN 熔丝位编程后使时钟系统的某些部分在睡眠模式下运行，这会增加功耗。
4. P33Table 16 给出 BODLEVEL 熔丝位具体设置。
5. RSTDISBL 熔丝位的说明，请见 P51“ 端口 A 的第二功能 ” 。

Table 68. 熔丝位低位字节

熔丝位低位字节	位号	描述	默认值
CKDIV8	7	时钟 8 分频	0 (被编程)
CKOUT	6	时钟输出	1 (未编程)
SUT1	5	选择启动时间	1 (未编程)(1)
SUT0	4	选择启动时间	0 (被编程)(1)
CKSEL3	3	选择时钟源	0 (被编程)(2)
CKSEL2	2	选择时钟源	0 (被编程)(2)
CKSEL1	1	选择时钟源	1 (未编程)(2)
CKSEL0	0	选择时钟源	0 (被编程)(2)

Note: 1. 对于默认时钟源，SUT1..0 的默认值给出最大的启动时间。详细内容见 P32Table 15。
2. CKSEL3..0 的默认设置导致了片内 RC 振荡器运行于 8 MHz。

熔丝位的状态不受芯片擦除命令的影响。如果锁定位 1(LB1) 被编程则熔丝位被锁定。在编程锁定位前先编程熔丝位。

熔丝位的锁存

器件进入编程模式时熔丝位的值被锁存。其间熔丝位的改变不会生效，直到器件退出编程模式。不过这不适用于 EESAVE 熔丝位。它一旦被编程立即起作用。在正常工作模式中器件上电时熔丝位也被锁存。

标识字节

所有的 Atmel 微控制器都具有一个三字节的标识代码用来区分器件型号。这个代码可以通过串行和并行模式读取，也可以在芯片被锁定时读取。这三个字节分别存储于三个独立的地址空间。

对 ATtiny2313 其标识字节为：

1. 0x000: 0x1E (表示由 Atmel 公司生产 I)
2. 0x001: 0x91 (表示芯片包含 2KB Flash 存储器)
3. 0x002: 0x0A (当 0x001 字节的内容为 0x91 时表示这是 ATmega48)

校准字节

ATtiny2313 内部 RC 振荡器保存两个不同校准值。这两个字节位于标识地址空间 0x000 与 0x0001 的高位字节，分别代表 4 与 8 MHz 。在复位期间， 4 MHz 值的字节被自动写入 OSCCAL 寄存器以确保校准的 RC 振荡器频率的正确性。

页尺寸

Table 69. 页中的字号与 Flash 中的页号

Flash 尺寸	页尺寸	PCWORD	页号	PCPAGE	PCMSB
1K 字 (2K 字节)	16 字	PC[3:0]	64	PC[9:4]	9

Table 70. 页中的字号与 EEPROM 中的页号

EEPROM 尺寸	页尺寸	PCWORD	页号	PCPAGE	EEAMSB
128 字节	4 字节	EEA[1:0]	32	EEA[6:2]	6

并行编程参数，引脚映射及命令

这部分描述了如何对 ATtiny2313 的 Flash 程序存储器、 EEPROM 数据存储器、存储锁定位及熔丝位进行并行编程和校验。除非另有说明，否则脉冲宽度至少为 250 ns。

信号名称

在这一节 ATtiny2313 的相关引脚以并行编程信号的名称进行引用 , 如 Figure 68 与 Table 71 所示。表中没有描述的引脚沿用原来的称谓。

XA1/XA0 决定了给 XTAL1 引脚一个正脉冲时所执行的操作。具体编码请见 Table 73。

给 $\overline{WR}$ 或 $\overline{OE}$ 输入脉冲时所加载的命令决定了要执行的操作。具体命令请参见 Table 74。

Figure 68. 并行编程

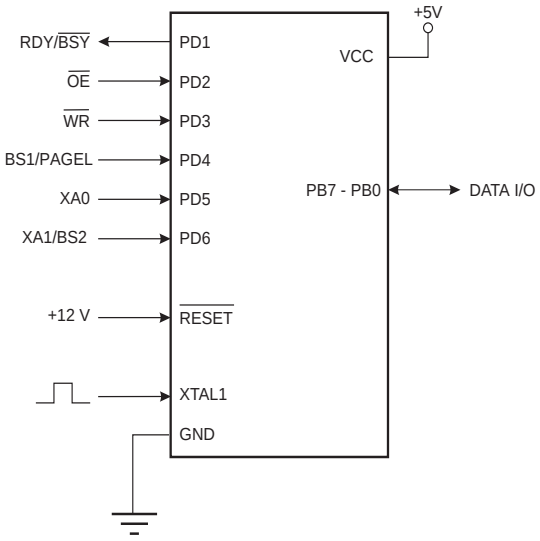


Table 71. 引脚名称映射

编程模式信号的名称	引脚名称	I/O	功能
RDY/$\overline{BSY}$	PD1	O	0: 设备忙于编程 , 1: 设备等待新的命令。
$\overline{OE}$	PD2	I	输出使能 (低电平有效).
$\overline{WR}$	PD3	I	写脉冲 (低电平有效).
BS1/PAGEL	PD4	I	字节选择 1(“0” 选择低位字节 , “1” 选择高位字节).
XA0	PD5	I	XTAL 动作位 0
XA1/BS2	PD6	I	XTAL 动作位 1 字节选择 2 (“0” 选择低位字节 , “1” 选择第二个高位字节)
DATA I/O	PB7-0	I/O	双向数据总线 ($\overline{OE}$ 为低时输出)

Table 72. 进入编程模式所需要的引脚数据

引脚	符号	数值
XA1	Prog_enable[3]	0
XA0	Prog_enable[2]	0
BS1	Prog_enable[1]	0
$\overline{WR}$	Prog_enable[0]	0

Table 73. XA1 和 XA0 的编码

XA1	XA0	给 XTAL1 施加脉冲激发的操作
0	0	加载 Flash 或 EEPROM 地址 (通过 BS1 确定是高位还是低位字节)
0	1	加载数据 (通过 BS1 决定是高位还是低位 Flash 数据字节)
1	0	加载命令
1	1	无操作，空闲

Table 74. 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	读标识字节和校准字节
0000 0100	读熔丝位和锁定位
0000 0010	读 Flash
0000 0011	读 EEPROM

串行编程引脚映射

Table 75. 串行编程时的引脚映射

符号	引脚	I/O	描述
MOSI	PB5	I	串行数据输入
MISO	PB6	O	串行数据输出 t
SCK	PB7	I	串行时钟

并行编程

进入编程模式

通过下面的算法进入并行编程模式：

1. 在 V_{CC} 及 GND 之间提供 4.5 - 5.5V 的电压。
2. 将 $\overline{\text{RESET}}$ 拉低，并至少改变 XTAL1 电平 6 次。
3. 将 P153Table 72 中列出的的 Prog_enable 引脚置为 "0000" , 并等待至少 100 ns。
4. 给 $\overline{\text{RESET}}$ 提供 11.5 - 12.5V 的电压。在向 $\overline{\text{RESET}}$ 提供 +12V 电压后的 100 ns 内，Prog_enable 引脚的任何行为都会导致芯片无法进入编程模式。
5. 在发送新命令之前至少等待 50 µs。

高效编程的几点考虑

在编程过程中，加载的命令及地址保持不变。为了实现高效的编程应考虑以下因素：

- 对多个存储单元进行读或写操作时，命令仅需加载一次。
- 当需要写入的数据为 0xFF 时可以跳过，因为这就是执行全片擦除命令后 Flash 及 EEPROM(除非 EESAVE 熔丝位被编程) 的内容。

- 只有在编程或读取Flash及EEPROM中新的256字时才需要用到地址高位字节。在读标识字节时也需考虑这一点。

芯片擦除

芯片擦除操作会擦除 Flash 及 EEPROM(1) 存储器以及锁定位。程序存储器没有擦除结束之前锁定位不会复位。全片擦除不影响熔丝位。芯片擦除命令必须在编程 Flash 与 / 或 EEPROM 之前完成。

Note: 1. 如果 EESAVE 熔丝位被编程，那么在芯片擦除时 EEPRPOM 不受影响。

加载 “ 芯片擦除 ” 命令的过程：

1. 将 XA1、 XA0 置为 “10” 以启动命令加载。
2. 将 BS1 置为 “0”。
3. DATA 赋值为 “1000 0000”。这是芯片擦除命令。
4. 给 XTAL1 提供一个正脉冲，进行命令加载。
5. 给 $\overline{WR}$ 提供一个负脉冲，启动芯片擦除。 RDY/$\overline{BSY}$ 变低。
6. 等待 RDY/$\overline{BSY}$ 变高，然后才能加载新的命令。

对 Flash 进行编程

Flash 是以页的形式组织起来的，如 P152Table 69 所示。编程 Flash 时，程序数据被锁存到页缓冲区中。这样一整页的程序数据可以同时得到编程。下面的步骤描述了如何对 Flash 进行编程：

A. 加载 “ 写 Flash” 命令：

1. 将 XA1、 XA0 置为 “10”，启动命令加载。
2. 将 BS1 置 "0”。
3. DATA 赋值为 “0001 0000”，这是写 Flash 命令。
4. 给 XTAL1 提供一个正脉冲以加载命令。

B. 加载地址低位字节 ：

1. 将 XA1、 XA0 置为 “00”，启动地址加载。
2. 将 BS1 置 “0”，选择低位地址。
3. DATA 赋值为地址低位字节 (0x00 - 0xFF)。
4. 给 XTAL1 提供一个正脉冲，加载地址低位字节。

C. 加载数据低位字节 ：

1. 将 XA1、 XA0 置为 “01”，启动数据加载。
2. DATA 赋值为数据低位字节 (0x00 - 0xFF)。
3. 给 XTAL1 提供一个正脉冲，加载数据字节。

D. 加载数据高位字节 ：

1. 将 BS1 置为 “1”，选择数据高位字节。
2. 将 XA1、 XA0 置为 “01”，启动数据加载。
3. DATA 赋值为数据高位字节 (0x00 - 0xFF)。
4. 给 XTAL1 提供一个正脉冲，进行数据字节加载。

E. 锁存数据：

1. 将 BS1 置为 “1”，选择数据高位字节。
2. 给 PAGEL 提供一个正脉冲，锁存数据 (见 Figure 70 信号波形)。

F. 重复 B 到 E 操作，直到整个缓冲区填满或此页中所有的数据都已加载。

地址信息中的低位用于页内寻址，高位用于 FLASH 页的寻址，详见 P157Figure 69。如果页内寻址少于 8 位 (页地址 < 256) ，那么进行页写操作时地址低字节中的高位用于页寻址。

G. 加载地址高位字节：

1. 将 XA1、 XA0 置为 “00”，启动地址加载操作。
2. 将 BS1 置为 “1”，选择高位地址。
3. DATA 赋值为地址高位字节 (0x00 - 0xFF)。
4. 给 XTAL1 提供一个正脉冲，加载地址高位字节。

H. 编程一页数据：

1. 给 $\overline{WR}$ 提供一个负脉冲，对整页数据进行编程， RDY/$\overline{BSY}$ 变低。

等待 RDY/$\overline{BSY}$ 变高 (见 Figure 70 信号波形)。

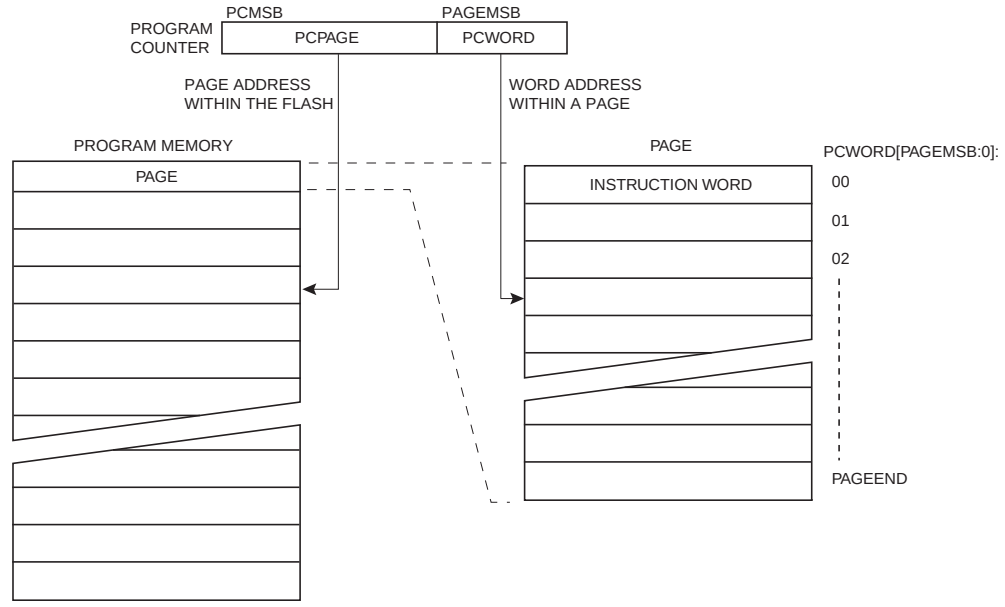
I. 重复 B 到 H 的操作，直到整个 Flash 编程结束或者所有的数据都被编程。

J. 结束页编程：

1. 1. 将 XA1、 XA0 置为 “10”，启动命令加载操作。
2. DATA 赋值为 “0000 0000”，这是不操作指令。

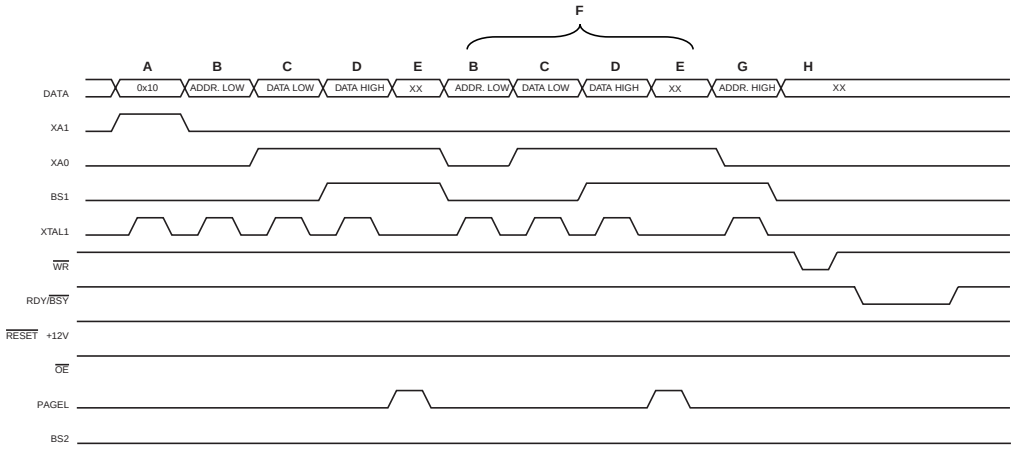
3. 给 XTAL1 提供一个正脉冲，加载命令，内部写信号复位。

Figure 69. 对以页为组织单位的 Flash 进行寻址 (1)



Note: 1. PCPAGE 及 PCWORD 列于 P152Table 69。

Figure 70. Flash 编程波形 (1)



Note: 1. 不用考虑 "XX"，各个大写字母对应于前面描述的 Flash 编程阶段。

对 EEPROM 进行编程

如 P152Table 70 所示， EEPROM 也以页为单位。编程 EEPROM 时，编程数据锁存于页缓冲区中。这样可以同时对一页数据进行编程。 EEPROM 数据存储器编程算法如下 (命令、地址及数据加载的细节请参见 P156“ 对 Flash 进行编程 ”) :

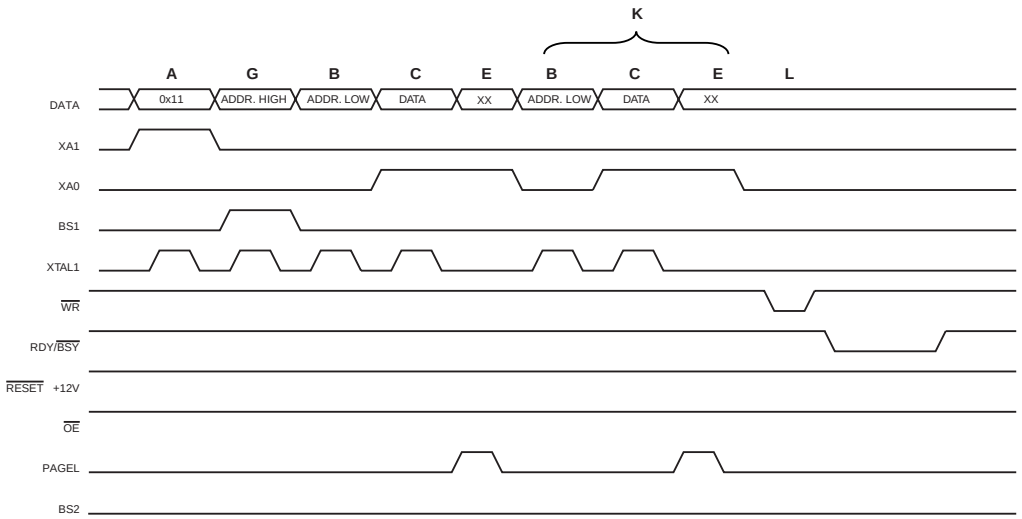
1. A：加载命令 “0001 0001”。
2. G：加载地址高位字节 (0x00 - 0xFF)。
3. B：加载地址低位字节 (0x00 - 0xFF)。
4. C：加载数据 (0x00 - 0xFF)。
5. E：锁存数据 (给 PAGEL 提供一个正脉冲)。

K：重复步骤 3 到 5，直到整个缓冲区填满。

L：对 EEPROM 页进行编程：

1. 将 BS 置 “0”。
2. 给 $\overline{WR}$ 提供一个负脉冲，开始对 EEPROM 页进行编程， RDY/$\overline{BSY}$ 变低。
3. 等到 RDY/$\overline{BSY}$ 变高再对下一页进行编程 (见 Figure 71 信号波形)。

Figure 71. EEPROM 编程波形



读取 Flash 的内容

读 Flash 存储器的过程如下 (命令及地址加载细节见 P156“ 对 Flash 进行编程 ”) :

1. A：加载命令 “0000 0010”。
2. G：加载地址高位字节 (0x00 - 0xFF)。
3. B：加载地址低位字节 (0x00 - 0xFF)。
4. 将 $\overline{OE}$ 置 “0”， BS1 置 “0”，然后从 DATA 读出 Flash 字的低位字节。
5. 将 BS 置 “1”，然后从 DATA 读出 Flash 字的高位字节。
6. 将 $\overline{OE}$ 置 “1”。

读取 EEPROM 的内容

读存储器的步骤如下 (命令及地址加载细节见 P156“ 对 Flash 进行编程 ”) :

1. A : 加载命令 “0000 0011”。
2. G : 加载地址高位字节 (0x00 - 0xFF)。
3. B : 加载地址低位字节 (0x00 - 0xFF)。
4. 将 $\overline{OE}$ 置 "0”， BS1 置 "0”，然后从 DATA 读出 EEPROM 数据字节。
5. 将 $\overline{OE}$ 置 ”1”。

对熔丝低位进行编程

对熔丝低位的编程步骤如下 (命令及数据装在细节见 P156“ 对 Flash 进行编程 ”) :

1. A : 加载命令 “0100 0000”。
2. C : 加载数据低字节，若某一位为 "0” 表示需要进行编程，否则需要擦除。
3. 给 $\overline{WR}$ 提供一个负脉冲，并等待 RDY/$\overline{BSY}$ 变高。

对熔丝高位进行编程

对熔丝高位的编程步骤如下 (命令及数据装在细节见 P156“ 对 Flash 进行编程 ”) :

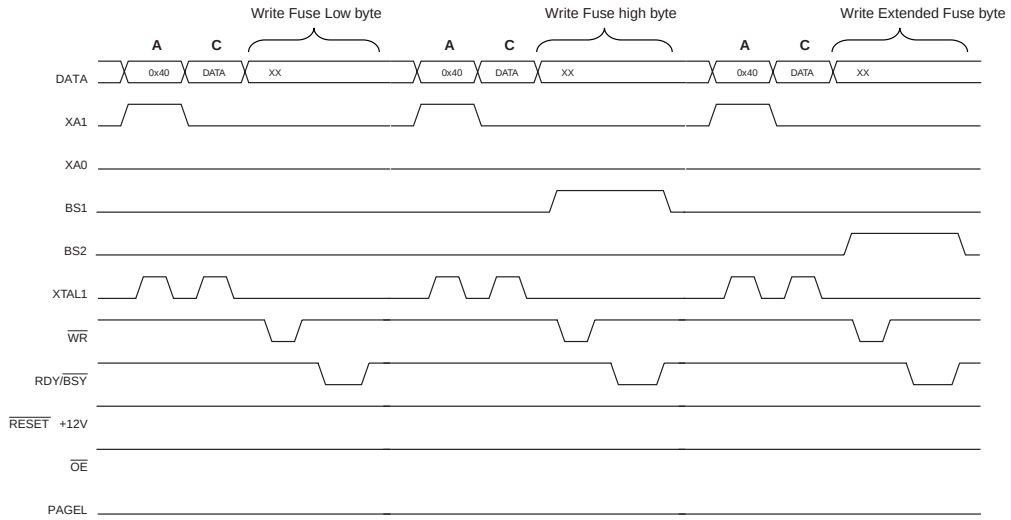
1. A : 加载命令 “0100 0000”。
2. C : 加载数据高字节，若某一位为 “0” 表示需要进行编程，否则需要擦除。
3. 将 BS1 置 “1”、 BS2 置 “0”，选择高位数据字节。
4. 给 $\overline{WR}$ 提供一个负脉冲并等待 RDY/$\overline{BSY}$ 变高。
5. 将 BS1 置 “0”，选择低位字节。

对扩展熔丝位进行编程

对扩展熔丝位的编程步骤如下 (命令及数据装在细节见 P156“ 对 Flash 进行编程 ”) :

1. A : 加载命令 “0100 0000”。
2. C : 加载数据低字节。若某一位为 “0” 表示需要进行编程，否则需要擦除。
3. 3. 将 BS1 置 “0”、 BS2 置 “1”，选择扩展数据字节。
4. 4. 给 $\overline{WR}$ 提供一个负脉冲并等待 RDY/$\overline{BSY}$ 变高。
5. 5. 将 BS2 置 “0”，选择低位字节。

Figure 72. 熔丝位编程波形



锁定位编程

锁定位编程步骤如下 (命令及数据装在细节见 P156“ 对 Flash 进行编程 ”) :

1. A : 加载命令 “0010 0000”。
2. C.: 加载数据低字节 , 位 n 为 “0” 表示此锁定位需要编程。如果芯片已经处于 LB 模式 3(LB1 及 LB2 被编程)，那么就不可能通过外部编程命令对锁定位进行编程。

3. 给 $\overline{WR}$ 提供一个负脉冲并等待 RDY/$\overline{BSY}$ 变高。

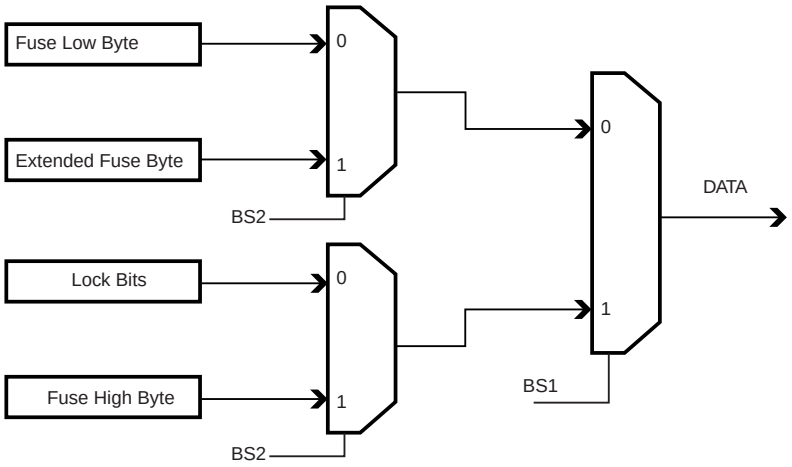
锁定位只能通过芯片擦除命令来清除。

读取熔丝位及锁定位

读取熔丝位及锁定位的步骤如下 (命令加载细节见 P156“ 对 Flash 进行编程 ”) :

1. A : 加载命令 “0000 0100”。
2. 将 $\overline{OE}$、BS2 和 BS1 置 “0” , 然后从 DATA 读取熔丝低位的状态 (“0” 表示已编程)。
3. 将 $\overline{OE}$ 置 “0” , BS2 和 BS1 置 “1” , 然后从 DATA 读取熔丝高位的状态 (“0” 表示已编程)。
4. 将 $\overline{OE}$ 和 BS1 置 “0” , BS2 置 “1” , 然后从 DATA 读取扩展熔丝位的状态 (“0” 表示已编程)。
5. 将 $\overline{OE}$ 置 “0” , BS2 置 “0” , BS1 置 “1” , 然后从 DATA 读取锁定位的状态 (“0” 表示已编程)。
6. 将 $\overline{OE}$ 置 “1”。

Figure 73. 读操作过程中 BS1、 BS2 与熔丝位及锁定位的对应关系



读取标识字节

读取标识字节的算法如下 (命令及地址加载参考 P156“ 对 Flash 进行编程 ”) :

1. A : 加载命令 “0000 1000”。
2. B : 加载地址低字节 0x00 - 0x02。
3. 将 $\overline{OE}$ 置 “0” , BS1 置 “1” , 然后从 DATA 读取标识字节。
4. 将 $\overline{OE}$ 置 “1”。

读取校准字节

读取校准字节的算法如下 (命令及地址加载参考 P156“ 对 Flash 进行编程 ”) :

1. A：加载命令 “0000 1000”。
2. B：加载地址低字节 0x00。
3. 将 $\overline{OE}$ 置 “0”，BS1 置 “1”，然后从 DATA 读取校准字节。
4. 将 $\overline{OE}$ 置 “1”。

并行编程特性

Figure 74. 并行编程时序，包括一些常规的时序要求

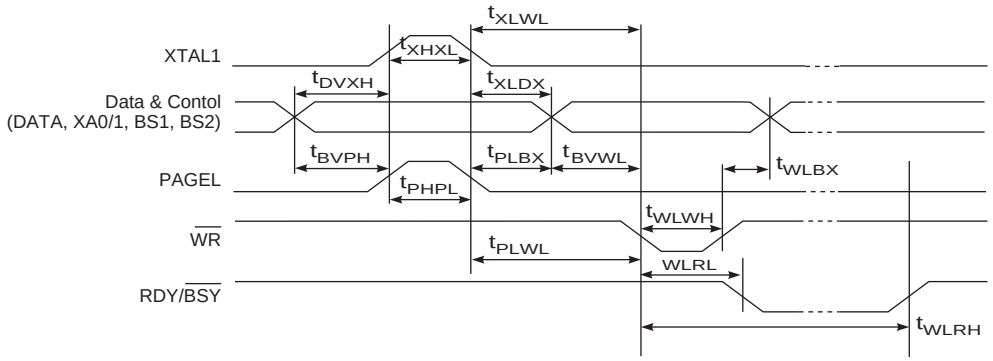
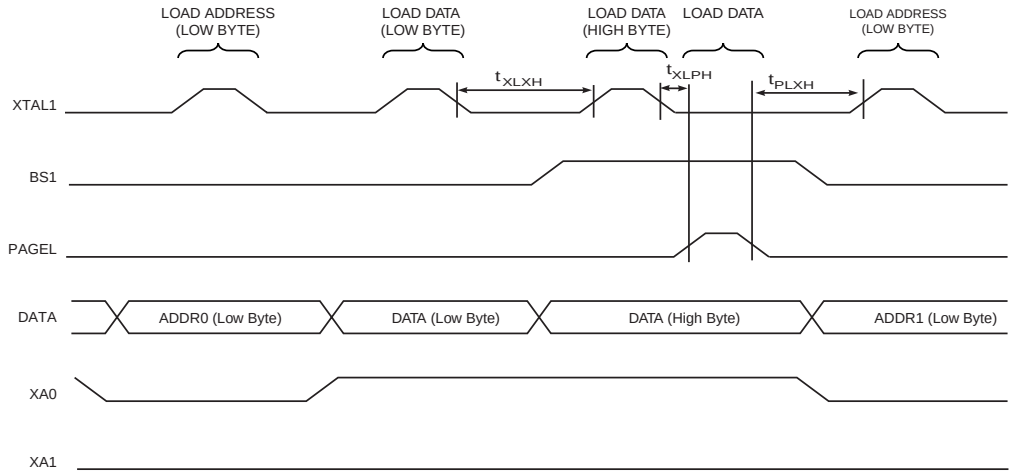
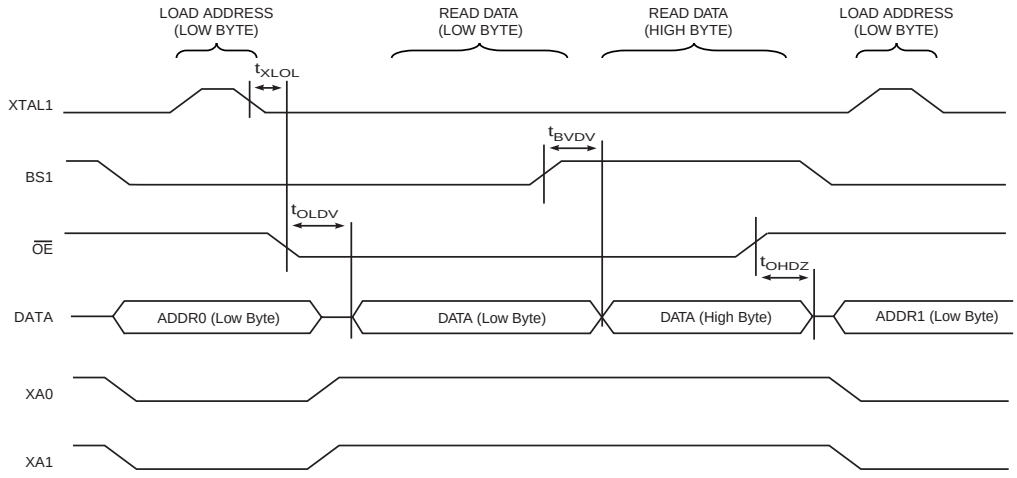


Figure 75. 并行编程时序，有时序要求的加载序列 (1)



Note: 1. Figure 74 给出的时序要求 (t_{DVXH}、 t_{XHXL} 及 t_{XLDX}) 也适用于加载操作。

Figure 76. 并行编程时序，有时序要求的读序列 (同一页)[1]



Note: 1. Figure 74 给出的时序要求 (即 t_{DVXH}、 t_{XHXL} 及 t_{XLDX}) 也适用于读操作。

Table 76. 并行编程参数 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}	在 XTAL1 为高之前数据及控制有效	67			ns
t_{XLXH}	从 XTAL1 低到 XTAL1 高	200			ns
t_{XHXL}	XTAL1 为高时的脉宽	150			ns
t_{XLDX}	XTAL1 为低之后数据及控制保持	67			ns
t_{XLWL}	从 XTAL1 低到 $\overline{WR}$ 低	0			ns
t_{XLPH}	从 XTAL1 低到 PAGEL 高	0			ns
t_{PLXH}	从 PAGEL 低到 XTAL1 高	150			ns
t_{BVPH}	PAGEL 为高之前 BS1 有效	67			ns
t_{PHPL}	PAGEL 为高时的脉宽	150			ns
t_{PLBX}	PAGEL 为低之后 BS1 保持	67			ns
t_{WLBX}	$\overline{WR}$ 为低之后 BS2/1 保持	67			ns
t_{PLWL}	从 PAGEL 低到 $\overline{WR}$ 为低	67			ns
t_{BVWL}	BS1 有效至 $\overline{WR}$ 为低	67			ns
t_{WLWH}	$\overline{WR}$ 为低时的脉宽	150			ns
t_{WLRL}	从 $\overline{WR}$ 低到 RDY/$\overline{BSY}$ 为低	0		1	µs
t_{WLRH}	从 $\overline{WR}$ 低到 RDY/$\overline{BSY}$ 为高 [1]	3.7		4.5	ms
t_{WLRH_CE}	从 $\overline{WR}$ 低到 RDY/$\overline{BSY}$ 为高，芯片擦除操作 [2]	7.5		9	ms
t_{XLOL}	从 XTAL1 低到 $\overline{OE}$ 为低	0			ns

Table 76. 并行编程参数 V_{CC} = 5V ± 10%

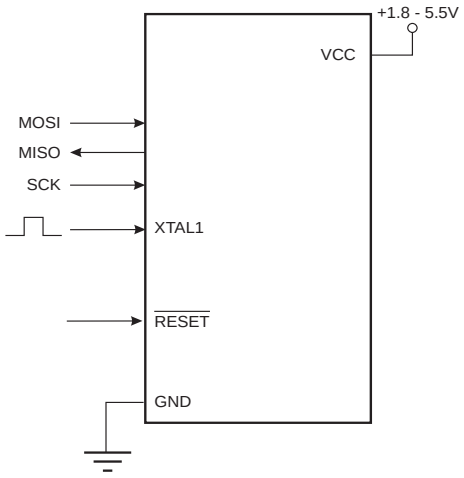
符号	参数	最小值	典型值	最大值	单位
t_{BVDV}	BS1 有效至 DATA 有效	0		250	ns
t_{OLDV}	从 $\overline{OE}$ 低到 DATA 有效			250	ns
t_{OHDZ}	从 $\overline{OE}$ 低到 DATA 为三态			250	ns

Notes: 1. 在进行 Flash、 EEPROM、熔丝位及锁定位写操作时 t_{WLRH} 有效。
2. 在执行芯片擦除操作时 t_{WLRH_CE} 有效。

串行下载

当 $\overline{RESET}$ 为低电平时，可以通过串行 SPI 总线对 Flash 及 EEPROM 进行编程。串行接口包括 SCK、 MOSI(输入) 及 MISO(输出)。 $\overline{RESET}$ 为低之后，应在执行编程 / 擦除操作之前执行编程允许指令。 P154Table 75 列出了 SPI 编程所需引脚的映射。不是所有的器件都使用 SPI 引脚专用于内部 SPI 接口。

Figure 77. 串行编程及校验 (1)



Notes: 1. 如果芯片由片内振荡器提供时钟，那么就不用在 XTAL1 引脚上连接时钟源。
2. V_{CC} - 0.3V < AVCC < V_{CC} + 0.3V，但是 AVCC 必须在 1.8 - 5.5V 范围内。

编程 EEPROM 时， MCU 在自定时的编程操作中会插入一个自动擦除周期，从而无需执行芯片擦除命令。芯片擦除操作将程序存储器及 EEPROM 的内容都擦除为 0xFF。

时钟通过 CKSEL 熔丝位确定。串行时钟 (SCK) 的最小低电平时间和最小高电平时间要满足如下要求：

低 :> f_{ck} < 12 MHz 时为 2 个 CPU 时钟周期，f_{ck} >= 12 MHz 时为 3 个 CPU 时钟周期

高 :> f_{ck} < 12 MHz 时为 2 个 CPU 时钟周期，f_{ck} >= 12 MHz 时为 3 个 CPU 时钟周期

串行编程算法

向 ATtiny2313 串行写入数据时，数据在 SCK 的上升沿得以锁存。

从 ATtiny2313 读取数据时，数据在 SCK 的下降沿输出。时序细节见 Figure 78、 Figure 79 与 Table 79。

在串行编程模式下对 ATtiny2313 进行编程及校验时，应遵循以下的步骤 (见 P165Table 78 中的 4 字节指令格式) :

1. 上电顺序 :
   在 RESET 及 SCK 为 "0" 时，向 V_{CC} 及 GND 供电。在一些系统中，编程器不能保证在上电时 SCK 保持为低。在这种情况下， SCK 拉低之后应在 RESET 加一正脉冲，而且这个脉冲至少要维持 2 个 CPU 时钟周期。
2. 上电之后等待至少 20 ms ,然后向 MOSI 引脚输入串行编程使能指令以使能串行编程。
3. 通信不同步将造成串行编程指令不工作。同步之后，在发送编程使能指令的第三个字节时，第二个字节的内容 (0x53) 将被反馈回来。不论反馈的内容正确与否，指令的 4 个字节必须全部传输。如果 0x53 未被反馈，则需要向 RESET 提供一个正脉冲以开始新的编程使能指令。
4. Flash 的编程以一次一页的方式进行。在执行加载程序存储页指令时 ,通过 4 LSB 的地址信息，数据以字节为单位加载到存储页。为保证加载的正确性，应先向给定地址传送数据低字节，之后是高字节。程序存储页通过地址的高 8 位以及写程序存储器页指令获得数据。如果不使用查询的方式，那么在操作下一页数据之前应等待至少 t_{WD_FLASH} 的时间 (见 P165Table 77.)。在 Flash 写操作完成之前访问串行编程接口会导致编程错误。
5. **A:** 提供了地址及数据信息之后 ,适合的写指令将以字节为单位对 EEPROM 编程。EEPROM 存储单元总是在写入新数据之前自动擦除。如果不使用查询的方式，那么在操作下一页数据之前应等待至少 t_{WD_EEPROM} 的时间 (见 P165Table 77.)。对于全片擦除之后的芯片，数据为 0xFF 的不需要编程。
   B:EEPROM 以页为单位编程。存储器页由 Load EEPROM 存储器页指令一次给出 2 LSB 的地址与数据，下载一个字节。 EEPROM 存储器页保存是通过载入写 EEPROM存储器页指令及 5 MSB地址来完成。当使用EEPROM页访问字节时 ,只需改变 Load EEPROM 存储器页指令即可。其余部分保持不变。如果不使用查询的方式 ,那么在操作下一页数据之前应等待至少 t_{WD_FLASH} 的时间 (见 P165Table 77)。对于全片擦除之后的芯片，数据为 0xFF 的不需要编程。
6. 可通过读指令来校验任何一个存储单元的内容。数据从串行输出口 MISO 输出。
7. 编程结束后可以将 RESET 拉高开始正常操作。
8. 下电序列 (如果需要) :
   将 RESET 置 “1”。
   切断 V_{CC}。

Table 77. 写下一个 Flash 或 EEPROM 单元之前的最小等待时间

符号	最小等待时间
t_{WD_FLASH}	4.5 ms
t_{WD_EEPROM}	4.0 ms
t_{WD_ERASE}	4.0 ms
t_{WD_FUSE}	4.5 ms

Figure 78. 串行编程波形图

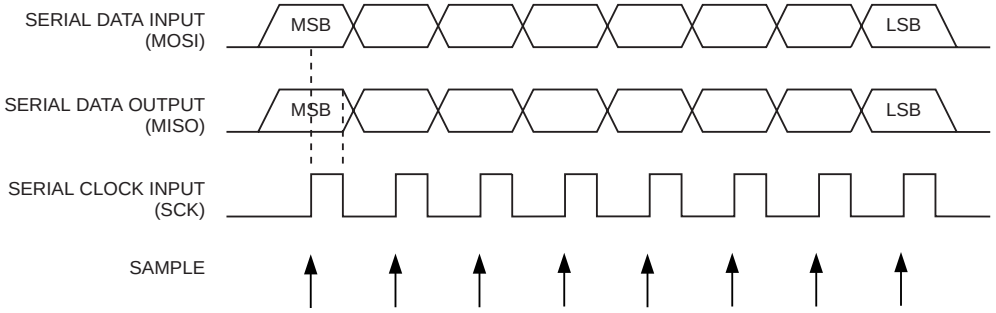


Table 78. 串行编程指令集

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	$\overline{\text{RESET}}$ 拉低后使能串行编程。
全片擦除	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	擦除 EEPROM 及 Flash。
读程序存储器	0010 **H**000	0000 00**aa**	**bbbb bbbb**	**oooo oooo**	从字地址为 **a**:**b** 的程序存储器读取 **H**(高或低字节) 数据的 **o**。
加载程序存储器页	0100 **H**000	000x xxxx	xxxx **bbbb**	**iiii iiii**	向字地址为 **b** 的程序存储页 **H**(高或低字节) 写入数据 **i**。应先写低字节再写高字节。
写程序存储器页	0100 1100	0000 00**aa**	**bbbb** xxxx	xxxx xxxx	在地址 **a**:**b** 加载程序存储页。
读 EEPROM 存储器	1010 0000	000x xxxx	x**bbb bbbb**	**oooo oooo**	从 EEPROM 的地址 **a**:**b** 处读出数据 **o**。
写 EEPROM 存储器	1100 0000	000x xxxx	x**bbb bbbb**	**iiii iiii**	向 EEPROM 地址 **b** 处中写入数据 **i**。
加载 EEPROM 存储器页 (页寻址)	1100 0001	0000 0000	0000 00**bb**	**iiii iiii**	将数据 **i** 加载到 EEPROM 存储器页缓冲区。数。据加载完毕后对 EEPROM 页进行编程
写EEPROM存储器页(页寻址)	1100 0010	00xx xxxx	x**bbb bb**00	xxxx xxxx	对地址为 **b** 的 EEPROM 执行页写操作。
读锁定位	0101 1000	0000 0000	xxxx xxxx	xx**oo oooo**	读锁定位。 ”0” 为已编程 , "1” 为未编程。细节见 P150Table 64。
写锁定位	1010 1100	111x xxxx	xxxx xxxx	11**ii iiii**	写锁定位。写 “0” 表示编程锁定位。细节见 P150Table 64。
读标识字节	0011 0000	000x xxxx	xxxx xx**bb**	**oooo oooo**	从地址 **b** 读取标识字节 **o**。

Table 78. 串行编程指令集

指令	指令格式				操作
	字节 1	字节 2	字节 3	字节 4	
写熔丝位	1010 1100	1010 0000	xxxx xxxx	**iiii iiii**	“0” 表示已编程，“1” 表示未编程。.
写高熔丝位	1010 1100	1010 1000	xxxx xxxx	**iiii iiii**	“0” 表示已编程，“1” 表示未编程。
写扩展熔丝位	1010 1100	1010 0100	xxxx xxxx	xxxx xxx**i**	“0” 表示已编程，“1” 表示未编程。
读熔丝位	0101 0000	0000 0000	xxxx xxxx	**oooo oooo**	读熔丝位。“0” 表示已编程，“1” 表示未编程。
读高熔丝位	0101 1000	0000 1000	xxxx xxxx	**oooo oooo**	读熔丝高位。“0” 表示已编程，“1” 表示未编程。
读扩展熔丝位	0101 0000	0000 1000	xxxx xxxx	**oooo oooo**	读扩展熔丝位。“0” 表示已编程，“1” 表示未编程。
读校准字节	0011 1000	000x xxxx	0000 000**b**	**oooo oooo**	在地址 **b** 读校准字节。
查询 RDY/$\overline{\text{BSY}}$	1111 0000	0000 0000	xxxx xxxx	xxxx xxx**o**	**o** = “1” 表示编程操作正在进行。等到它为 “0” 后可以执行其他命令。

Note: **a** = 地址高位， **b** = 地址低位， **H** = 0 - 低字节， 1 - 高字节 ，**o** = 数据输出，**i** = 数据输入，x = 任意值

串行编程特性参数

Figure 79. 串行编程时序图

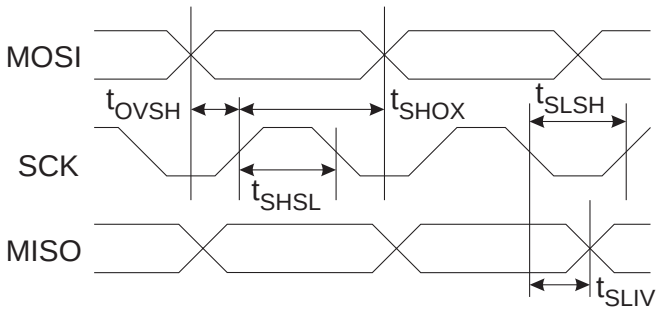


Table 79. 串行编程特性参数 , T_A = -40°C ~ 85°C, V_{CC} = 2.7V - 5.5V (在无特别指出时)

符号	参数	最小值	典型值	最大值	单位
$1/t_{CLCL}$	振荡器频率 (ATtiny2313L)	0		8	MHz
t_{CLCL}	振荡器周期 (ATtiny2313L)	125			ns
$1/t_{CLCL}$	振荡器频率 (ATtiny2313, V_{CC} = 4.5V - 5.5V)	0		16	MHz
t_{CLCL}	振荡器周期 (ATtiny2313, V_{CC} = 4.5V - 5.5V)	67			ns
t_{SHSL}	SCK 脉宽为高	2 t_{CLCL}*			ns
t_{SLSH}	SCK 脉宽为低	2 t_{CLCL}*			ns
t_{OVSH}	MOSI 设置到 SCK 为高	t_{CLCL}			ns
t_{SHOX}	SCK 为高后 MOSI 保持	2 t_{CLCL}			ns
t_{SLIV}	SCK 低到 MISO 有效	TBD	TBD	TBD	ns

Note: 1. f_{ck} < 12 MHz 时 2 t_{CLCL} ， f_{ck} >= 12 MHz 时 3 t_{CLCL}

电气特性

绝对极限值 *

工作温度	-55°C ~ +125°C
存储温度	-65°C ~ +150°C
各个引脚对地的电压，除了 $\overline{\text{RESET}}$	-0.5V ~ V_{CC}+0.5V
$\overline{\text{RESET}}$ 引脚对地的电压	-0.5V ~ +13.0V
最大工作电压	6.0V
每个 I/O 引脚上的直流电流	40.0 mA
V_{CC} 与 GND 引脚上的直流电流	200.0 mA

*NOTICE: 如果强制芯片在超出“ 绝对极限值” 表中所列的条件之下工作可能造成器件的永久损坏。这仅是工作应力的极限。并不表示器件可以工作于表中所列条件之下，或是那些超越工作范围明确规定的其他条件之下。长时间工作于绝对极限值可能会影响器件的寿命。

直流特性

T_A = -40°C ~ 85°C, V_{CC} = 1.8V ~ 5.5V (无特别说明)(1)

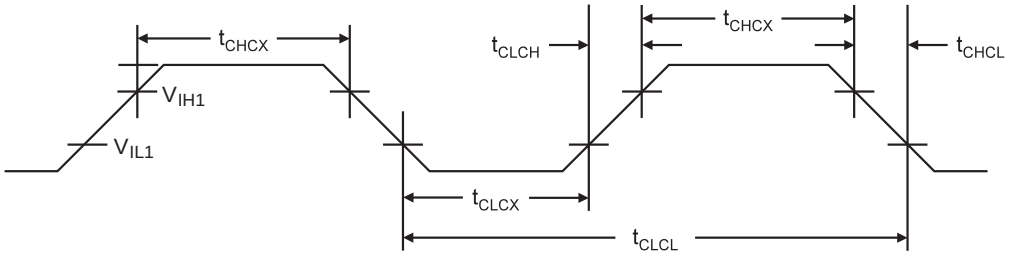
符号	参数	条件	最小值	典型值	最大值	单位
V_{IL}	输入低电压		-0.5		$0.2V_{CC}$	V
V_{IH}	输入高电压	除 $\overline{\text{RESET}}$ 引脚	$0.6V_{CC}$(3)		V_{CC} +0.5	V
V_{IH2}	输入高电压	$\overline{\text{RESET}}$ 引脚	$0.9V_{CC}$(3)		V_{CC} +0.5	V
V_{OL}	输出低电压 (4) (端口 B)	I_{OL} = 10 mA, V_{CC} = 5V I_{OL} = 5 mA, V_{CC} = 3V			0.7 0.5	V V
V_{OH}	输出高电压 (5) (端口 B)	I_{OH} = -10 mA, V_{CC} = 5V I_{OH} = -5 mA, V_{CC} = 3V	4.2 2.5			V V
I_{IL}	输入漏电流， I/O 引脚	Vcc = 5.5V, 引脚低 (绝对值)			1	µA
I_{IH}	输入漏电流， I/O 引脚	Vcc = 5.5V, 引脚低 (绝对值)			1	µA
R_{RST}	Reset 引脚上拉电阻		30		60	kΩ
R_{pu}	I/O 引脚上拉电阻		20		50	kΩ
I_{CC}	电源电流	工作于 1MHz, V_{CC} = 2V			0.35	mA
		工作于 4MHz, V_{CC} = 3V			2	mA
		工作于 8MHz, V_{CC} = 5V			6	mA
		空闲 1MHz, V_{CC} = 2V		0.08	0.2	mA
		空闲 4MHz, V_{CC} = 3V		0.41	1	mA
		空闲 8MHz, V_{CC} = 5V		1.6	3	mA
	掉电模式	WDT 使能 , V_{CC} = 3V		< 3	6	µA
		WDT 禁用 , V_{CC} = 3V		< 0.5	2	µA

Notes: 1. 数据表中的所有直流特性均基于仿真及与其使用同一制造工艺的其他 AVR 微控制器的特性值。仅是设计的目标值，实际值在对实际芯片的测试后会更新。。
2. “ 最大值 ” 表示保证引脚读取数值为低时的最高值。
3. “ 最小值 ” 表示保证引脚读取数值为高时的最低值。

4. 虽然在稳定状态条件(非瞬态)下每个I/O端口都可以吸收比测试条件下更多的电流(20 mA ,V_{CC} = 5V；10 mA，V_{CC} = 3V) ，但是仍需要遵循以下要求：
   1] 所有端口的 IOL 总和不应超过 60 mA。
   如果 IOL 超过了测试条件， VOL 可能超过相关指标。不保证引脚可以吸收比列于此处的测试条件更大的电流。
5. 虽然在稳定状态条件 (非瞬态) 下每个 I/O 端口都可以输出比测试条件下更多的电流 (20 mA ,V_{CC} = 5V ； 10 mA，V_{CC} = 3V) ，但是需要遵循以下要求：
   1] 所有端口的 IOH 总和不应超过 60 mA。
   如果 IOH 超过了测试条件， VOH 可能超过相关指标。不保证引脚可以输出比列于此处的测试条件更大的电流。

外部时钟波形

Figure 80. 外部时钟波形



外部时钟驱动

Table 80. 外部时钟驱动 (估计值)

		V_{CC}=1.8-5.5V				V_{CC}=4.5-5.5V		
符号	参数	最小值	最大值	最小值	最大值	最小值	最大值	单位
$1/t_{CLCL}$	振荡器频率	0	1	0	8	0	16	MHz
t_{CLCL}	时钟周期	1000		125		62.5		ns
t_{CHCX}	高电平时间	400		50		25		ns
t_{CLCX}	低电平时间	400		50		25		ns
t_{CLCH}	上升时间		2.0		1.6		0.5	µs
t_{CHCL}	下降时间		2.0		1.6		0.5	µs
Δt_{CLCL}	时钟周期的变化		2		2		2	%

最大速度与 V_{CC} 的关系

最高频率由 V_{CC} 决定，见 Figure 81 与 Figure 82。最高频率与 V_{CC} 关系曲线在 1.8V < V_{CC} < 2.7V 及 2.7V < V_{CC} < 4.5V 为线性。

Figure 81. ATtiny2313V 中最高频率与 V_{CC} 的关系

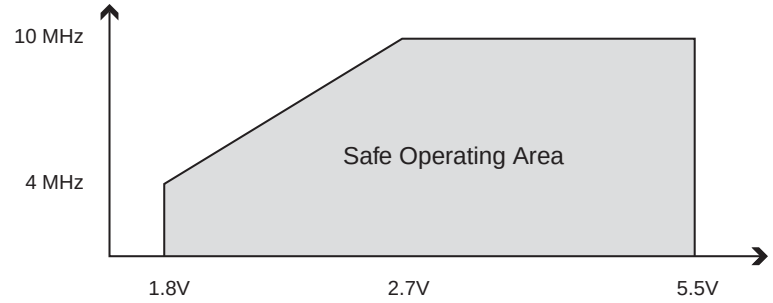
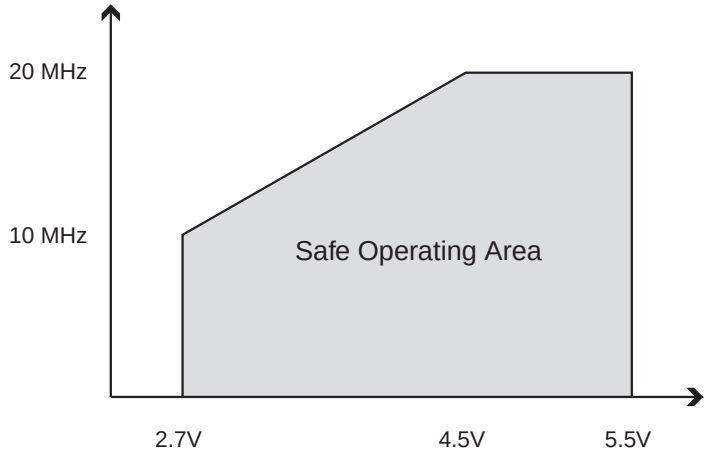


Figure 82. ATtiny2313 中最高频率与 V_{CC} 的关系



ATtiny2313 典型特性

下面的图表给出了器件的典型性能。在产生过程中并不测试这些数据。全部电流测量数据都是在所有的 I/O 引脚配置为输入且内部上拉电阻使能的条件下测得的。时钟源为外部正弦波发生器产生的满幅值正弦波。

掉电模式下的功耗与选择的时钟无关。

耗电流与多个因素有关：工作电压、工作频率、 I/O 的负载、 I/O 引脚开关速率、执行的代码及环境温度。最主要的因素是工作电压和工作频率。

容性负载 I/O 的输出电流可通过公式 C_L*V_{CC}*f 进行估算， C_L = 负载电容， V_{CC} = 工作电压， f = I/O 引脚平均开关频率。

器件的特性化是在比测试极限频率更高的频率进行的。但是不保证器件能够正常在高于订货信息表给出的工作频率。

看门狗使能的掉电模式和看门狗禁止的掉电模式之间的电流差值即为开关看门狗定时器所需的电流。

值为估计值。

工作电流

Figure 83. 工作电流和工作频率 (0.1 - 1.0 MHz) 的关系

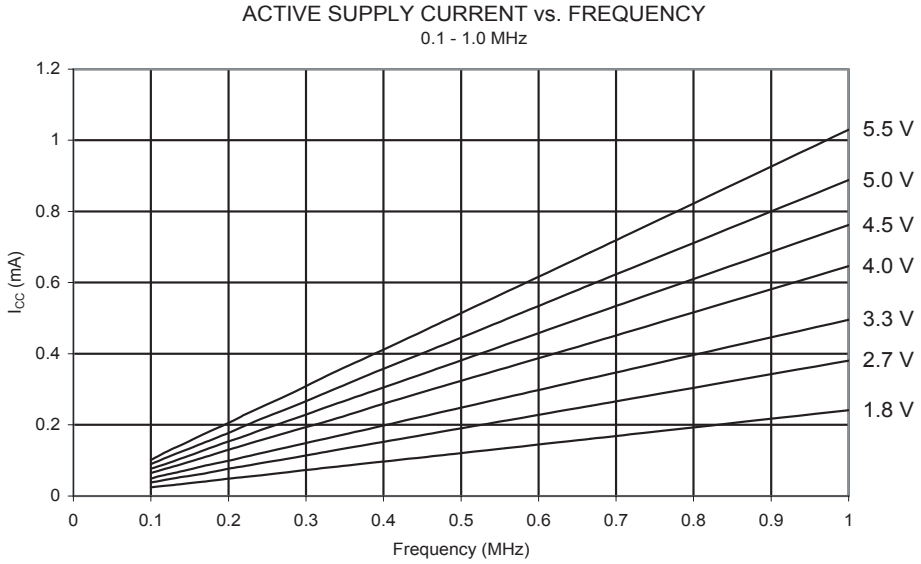


Figure 84. 工作电流和工作频率 (1 - 20 MHz) 的关系

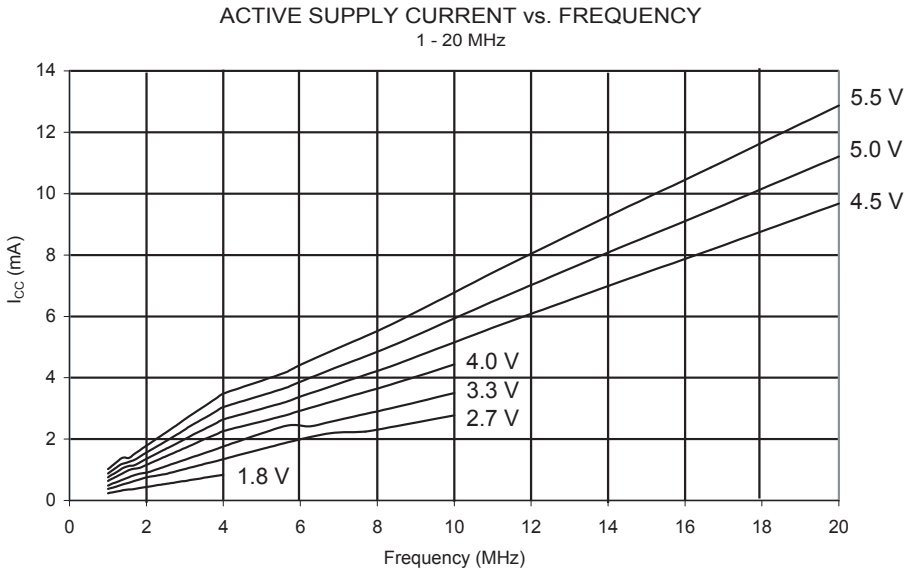


Figure 85. 工作电流和 V_{CC} 的关系 (内部 RC 振荡器，8 MHz)

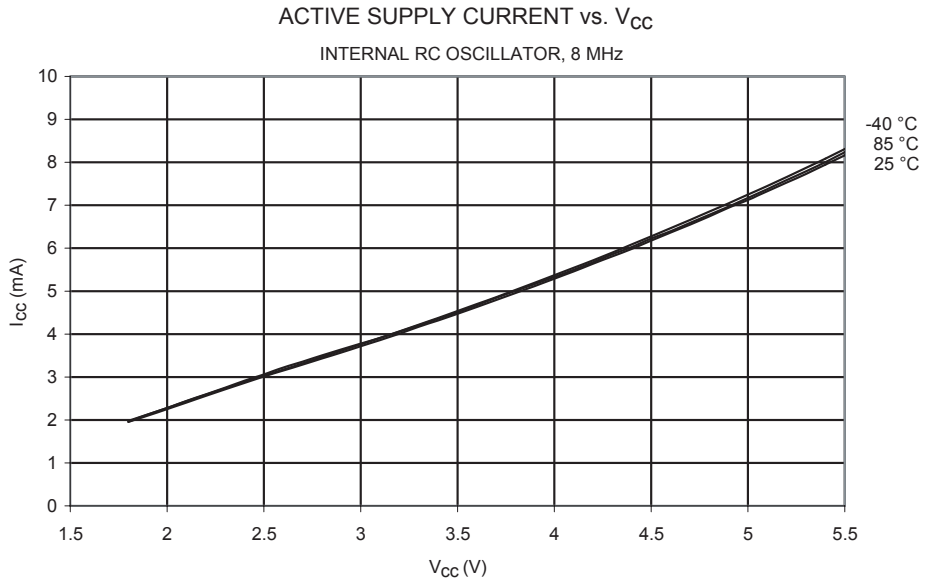


Figure 86. 工作电流和 V_{CC} 的关系 (内部 RC 振荡器 ，4 MHz)

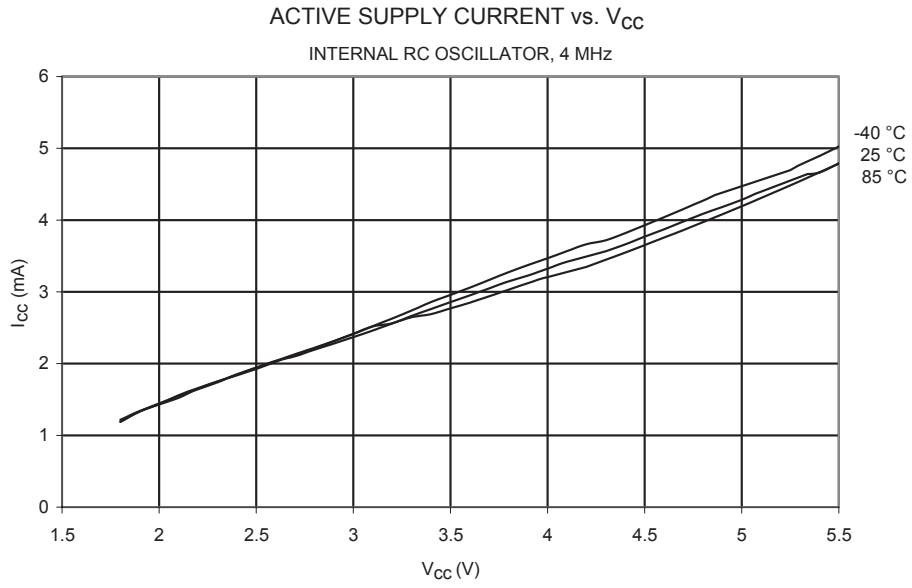


Figure 87. 工作电流和 V_{CC} 的关系 (内部 RC 振荡器 ，1 MHz)

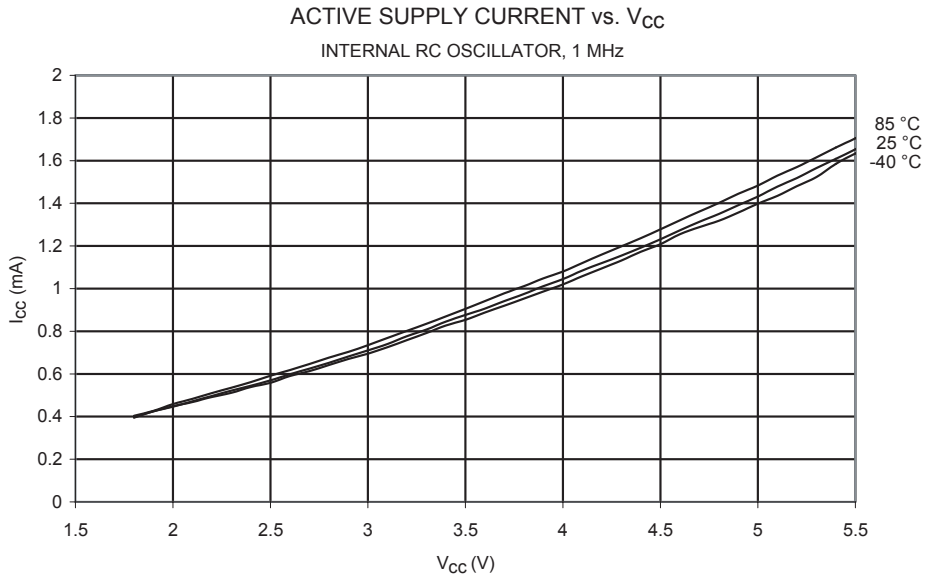


Figure 88. 工作电流和 V_{CC} 的关系 (内部 RC 振荡器 ， 0.5 MHz)

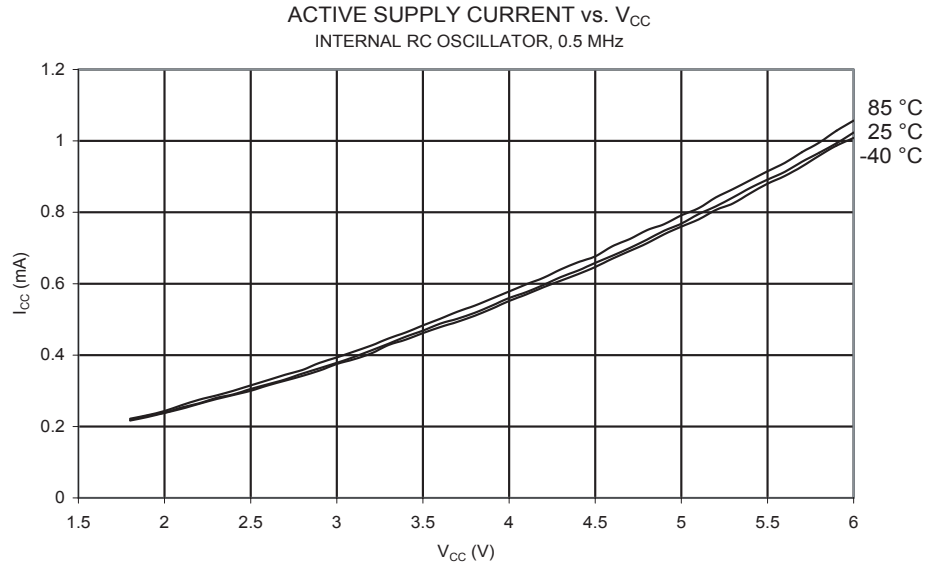
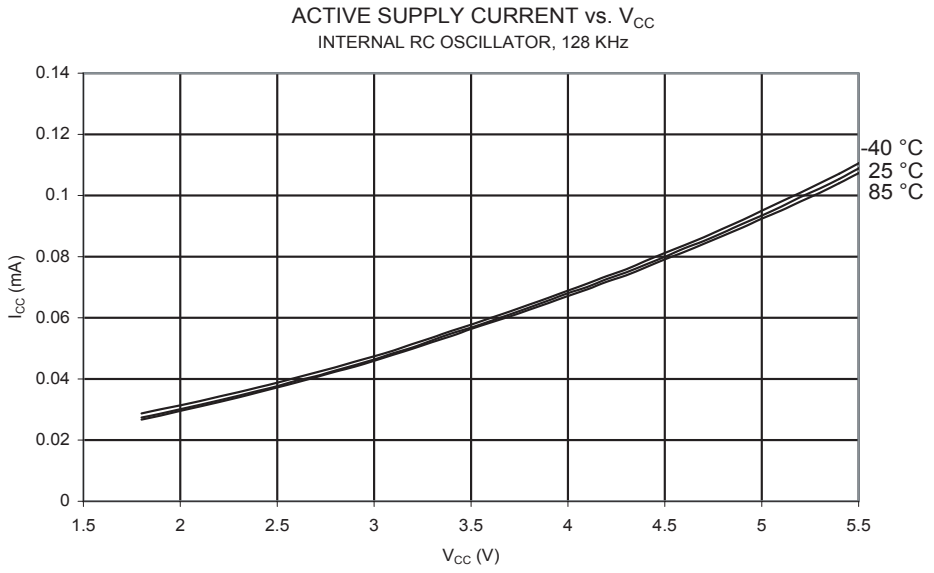


Figure 89. 工作电流和 V_{CC} 的关系 (内部 RC 振荡器 ， 128 KHz)



空闲模式电流

Figure 90. 空闲模式电流和工作频率 (0.1 - 1.0 MHz) 的关系

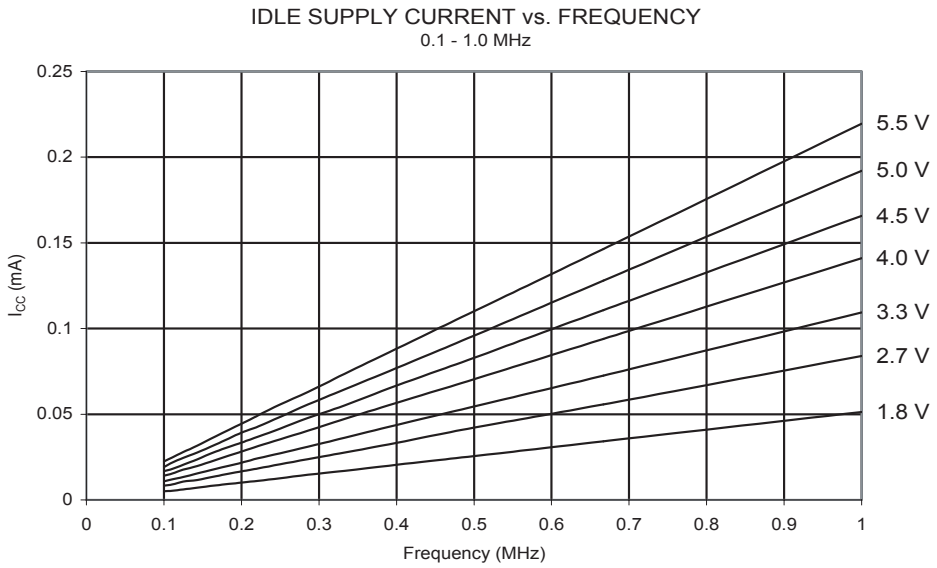


Figure 91. 空闲模式电流和工作频率 (1 - 20 MHz) 的关系

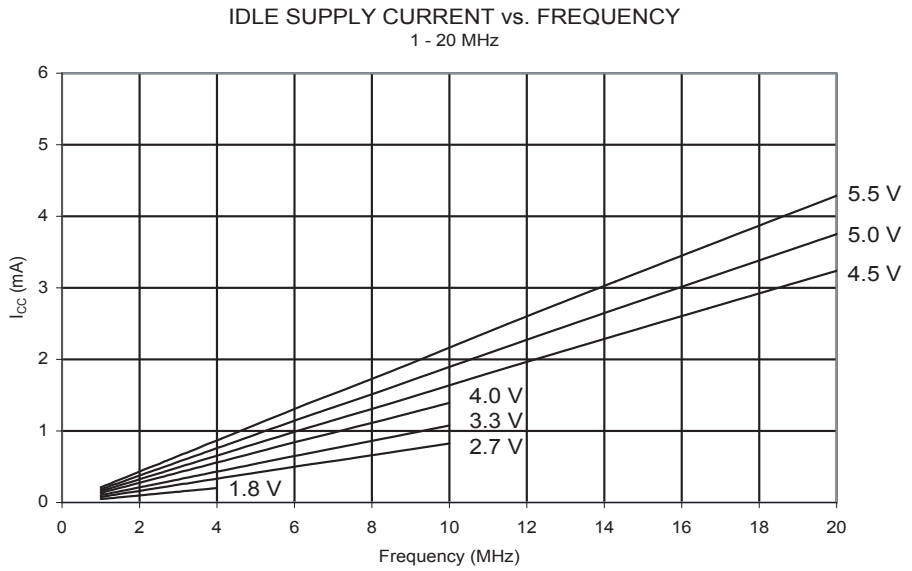


Figure 92. 空闲模式电流和 V_{CC} 的关系 (内部 RC 振荡器 ， 8 MHz)

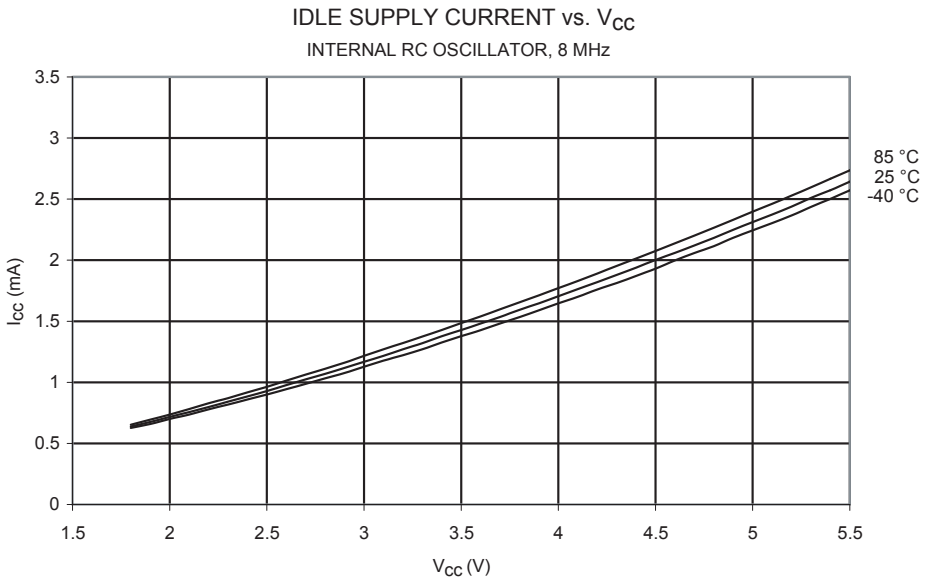


Figure 93. 空闲模式电流和 V_{CC} 的关系 (内部 RC 振荡器 ， 4 MHz)

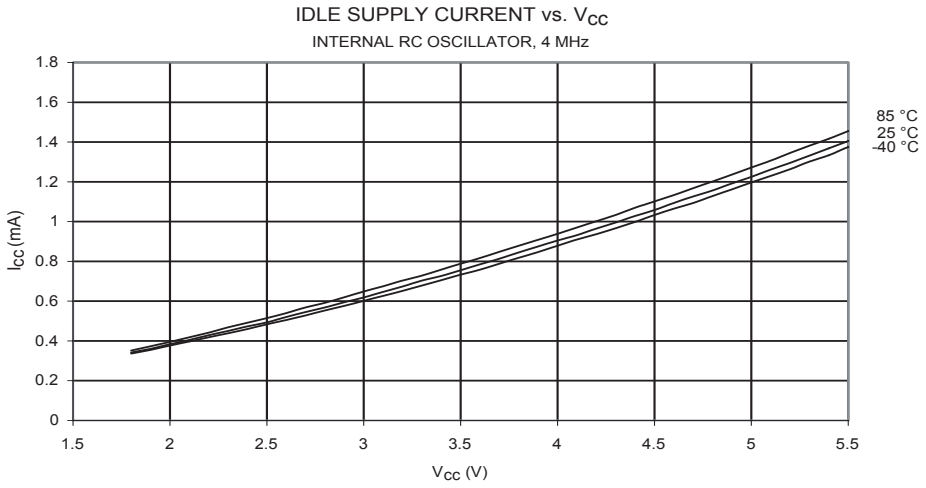


Figure 94. 空闲模式电流和 V_{CC} 的关系 (内部 RC 振荡器 ， 1 MHz)

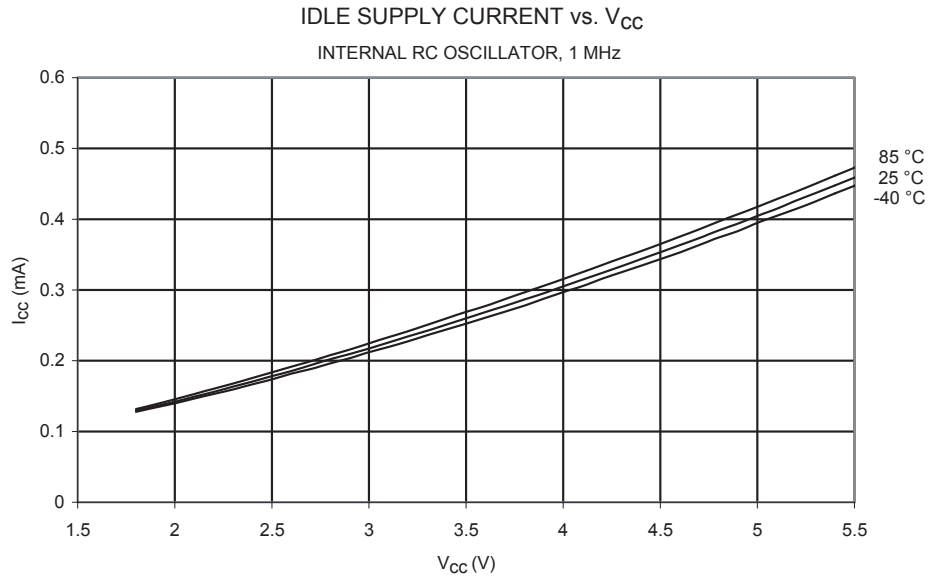


Figure 95. 空闲模式电流和 V_{CC} 的关系 (内部 RC 振荡器 ， 0.5 MHz)

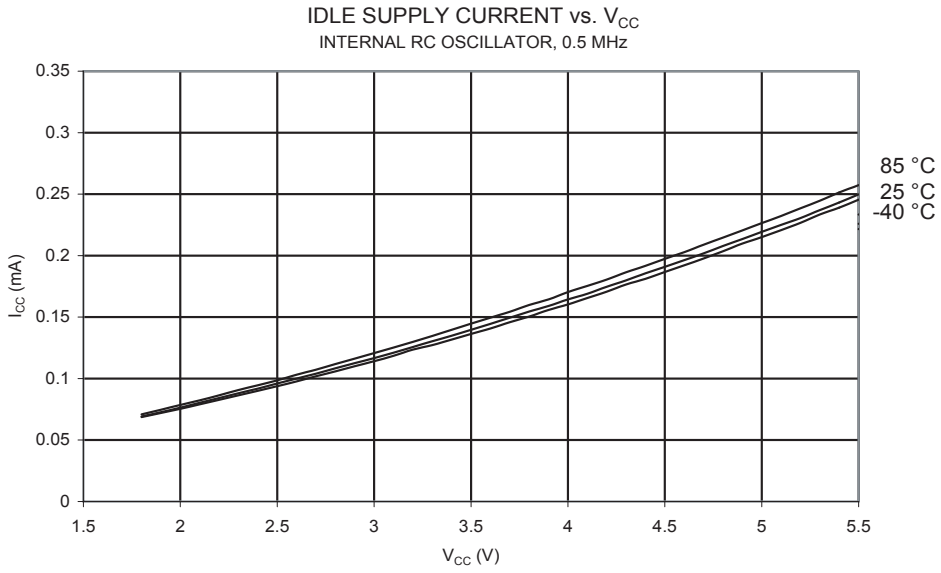
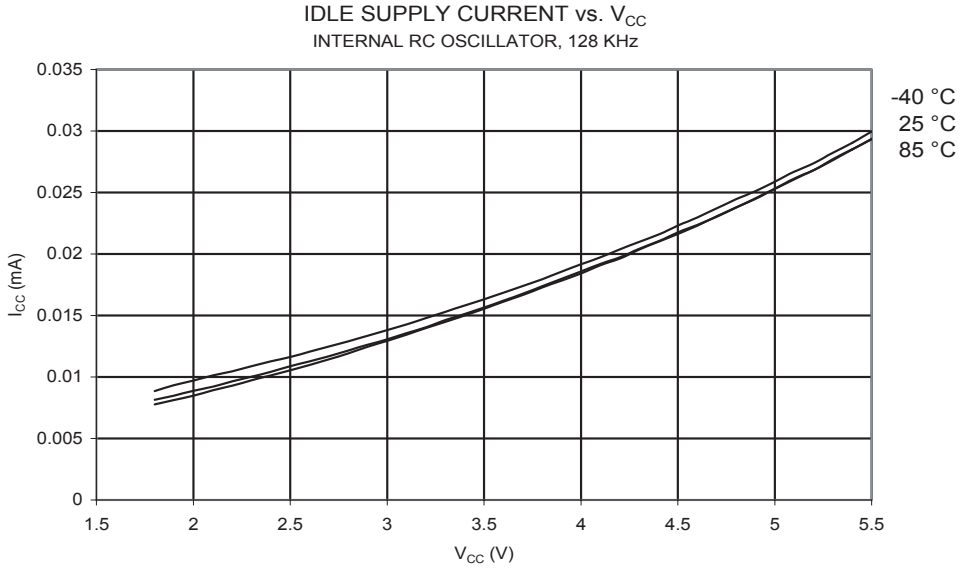


Figure 96. 空闲模式电流和 V_{CC} 的关系 (内部 RC 振荡器 ， 128 KHz)



掉电模式电流

Figure 97. 掉电模式电流和 V_{CC} 的关系 (看门狗定时器禁用)

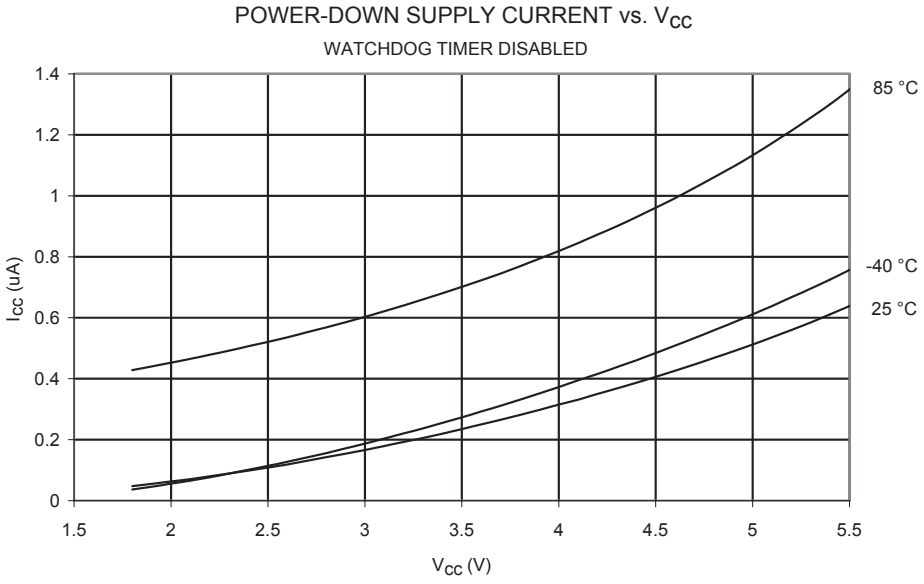
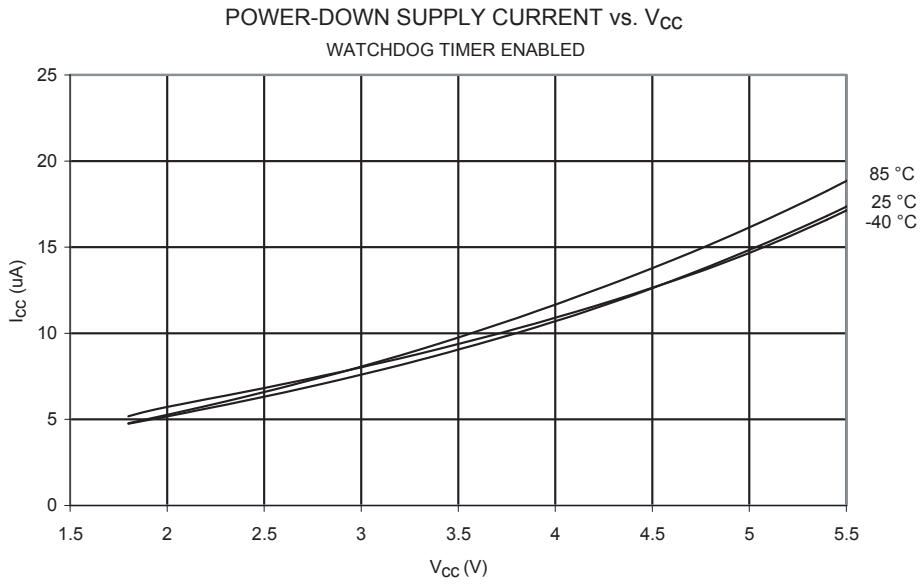
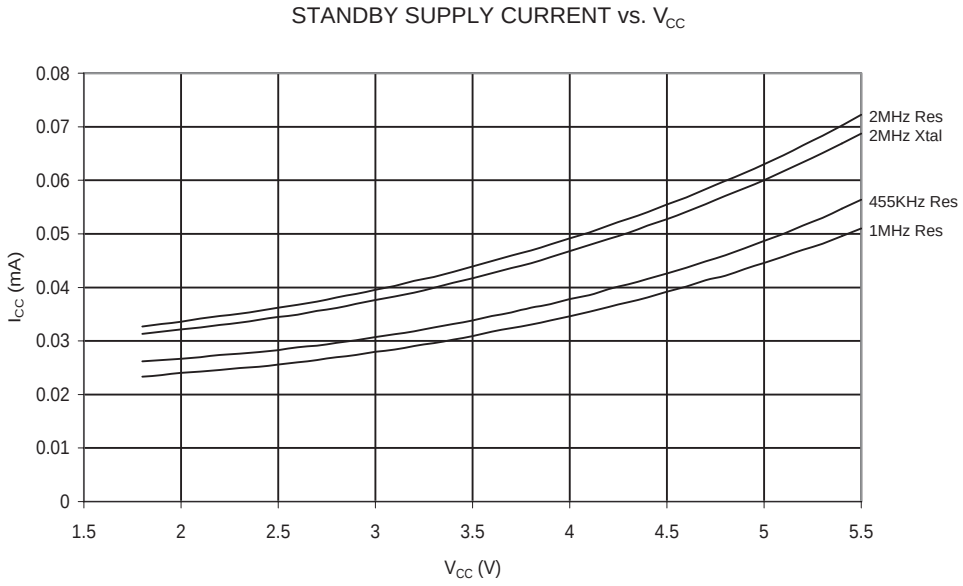


Figure 98. 掉电模式电流和 V_{CC} 的关系 (看门狗定时器使能)



Standby 模式电流

Figure 99. Standby 模式电流和 V_{CC} 的关系



引脚上拉

Figure 100. I/O 引脚上拉电阻电流和输入电压的关系 (V_{CC} = 5V)

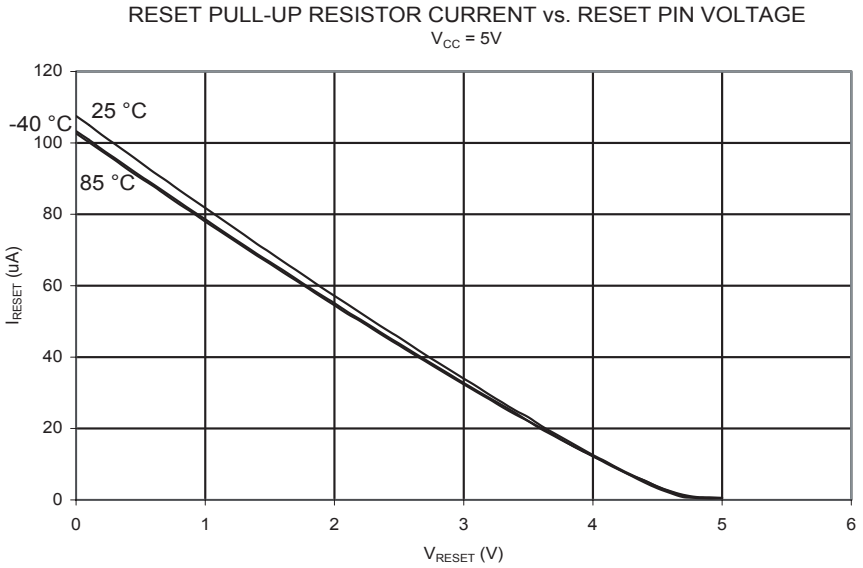


Figure 101. I/O 引脚上拉电阻电流和输入电压的关系 (V_{CC} = 2.7V)

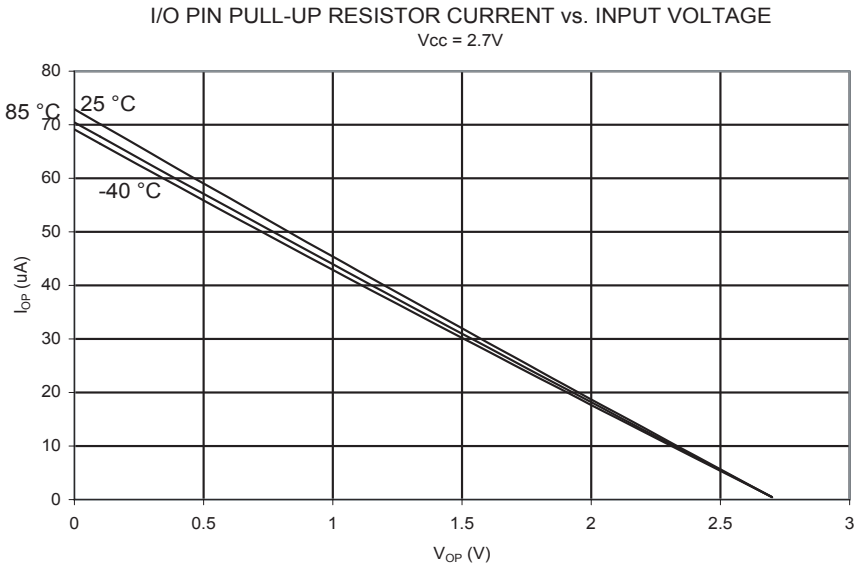


Figure 102. 复位 (Reset) 引脚上拉电阻电流和 Reset 引脚电压的关系 (V_{CC} = 5V)

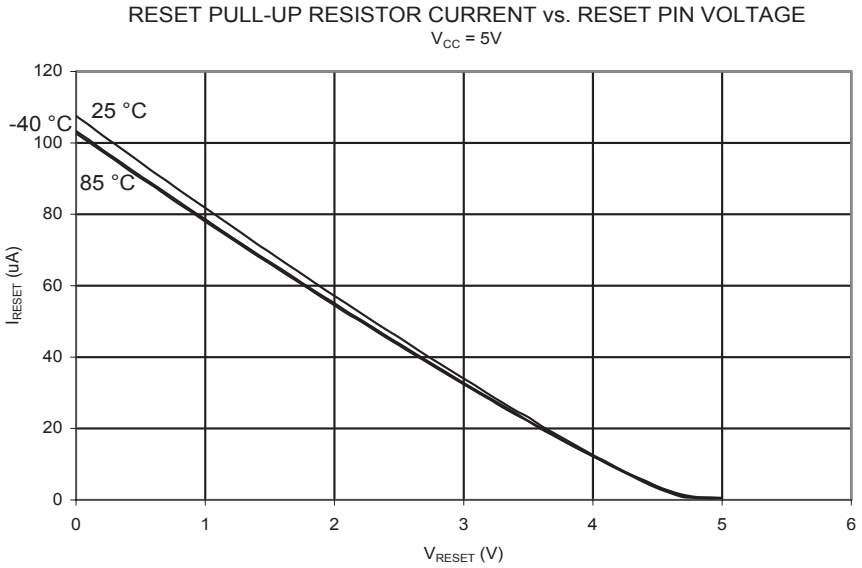
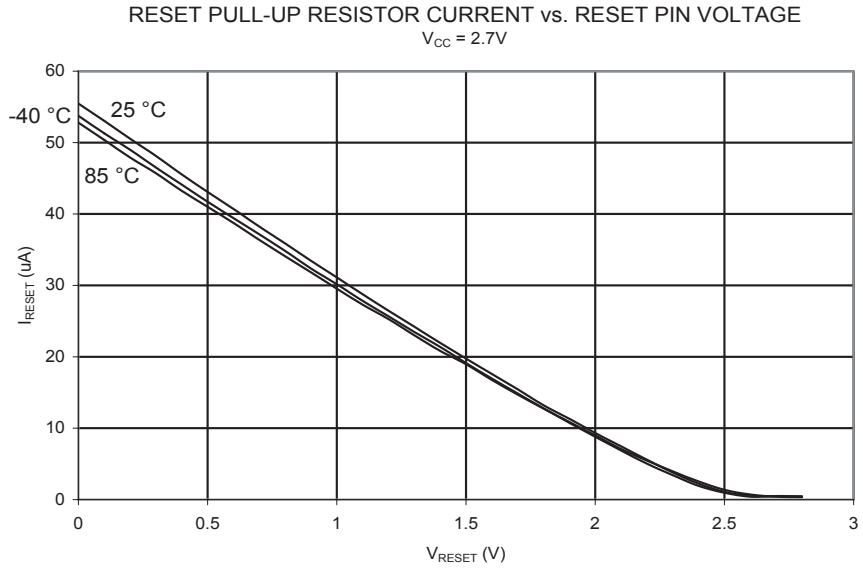


Figure 103. 复位 (Reset) 引脚上拉电阻电流和 Reset 引脚电压的关系 (V_{CC} = 2.7V)



驱动能力

Figure 104. I/O 引脚源电流和输出电压的关系 (V_{CC} = 5V)

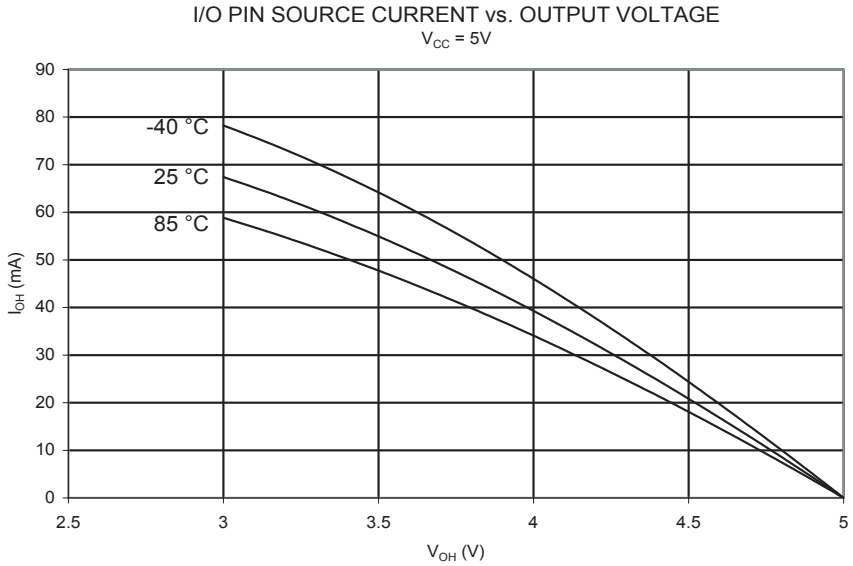


Figure 105. I/O 引脚源电流和输出电压的关系 (V_{CC} = 2.7V)

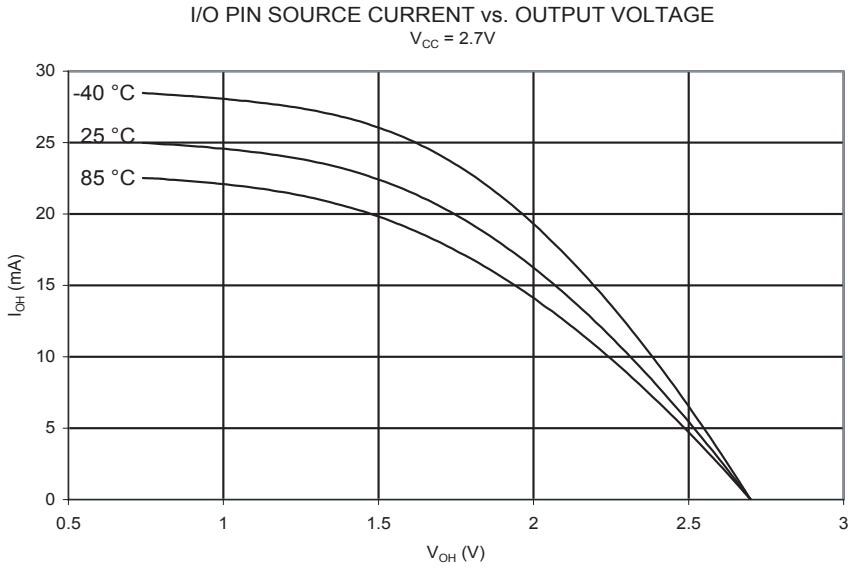


Figure 106. I/O 引脚源电流和输出电压的关系 (V_{CC} = 1.8V)

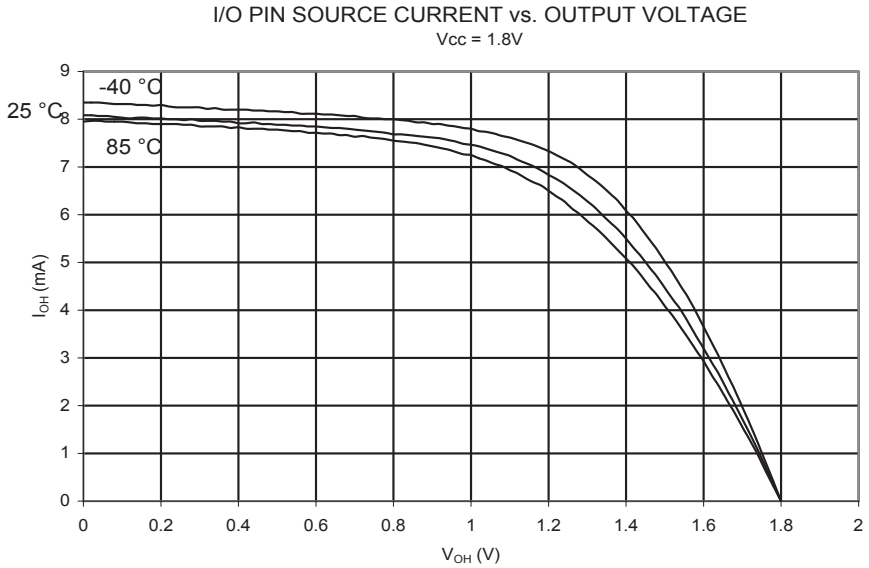


Figure 107. I/O 引脚吸收电流和输出电压的关系 (V_{CC} = 5V)

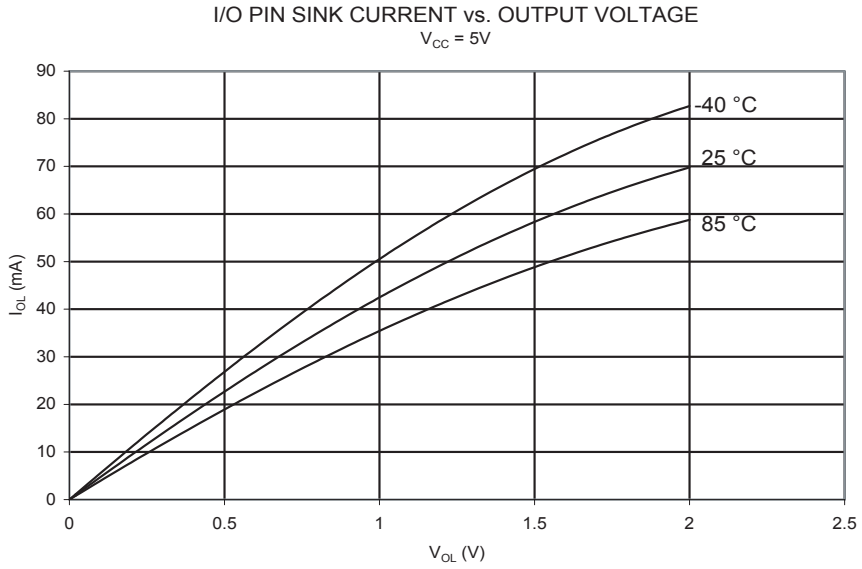


Figure 108. I/O 引脚吸收电流和输出电压的关系 (V_{CC} = 2.7V)

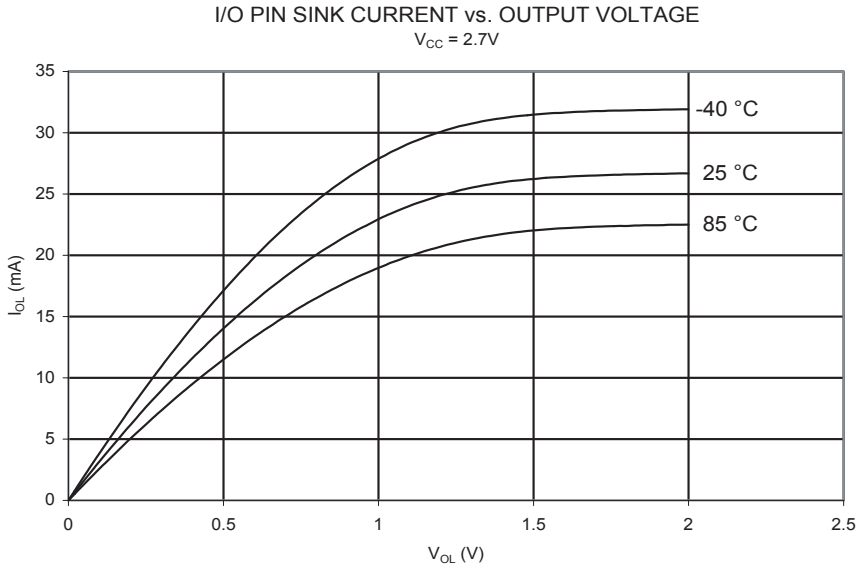


Figure 109. I/O 引脚吸收电流和输出电压的关系 (V_{CC} = 1.8V)

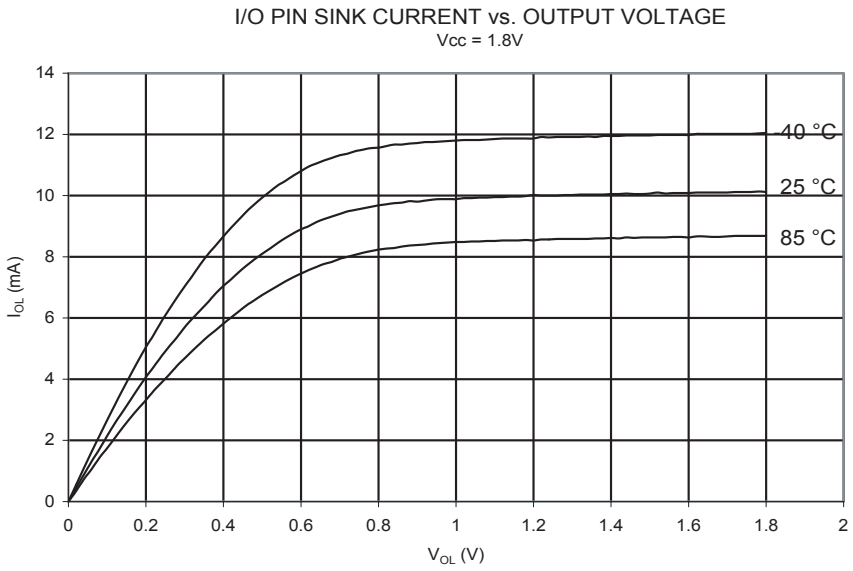


Figure 110. 复位 I/O 引脚源电流和输出电压的关系 (V_{CC} = 5V)

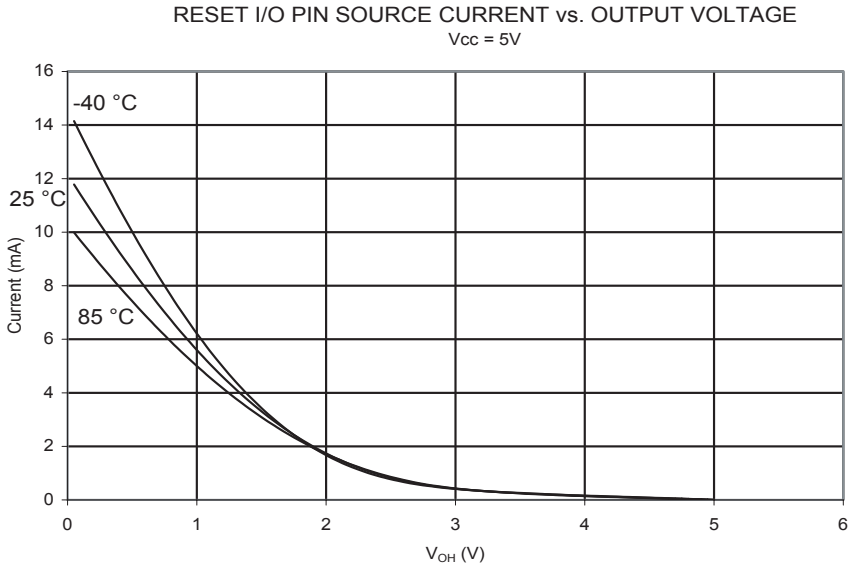


Figure 111. 复位 I/O 引脚源电流和输出电压的关系 (V_{CC} = 2.7V)

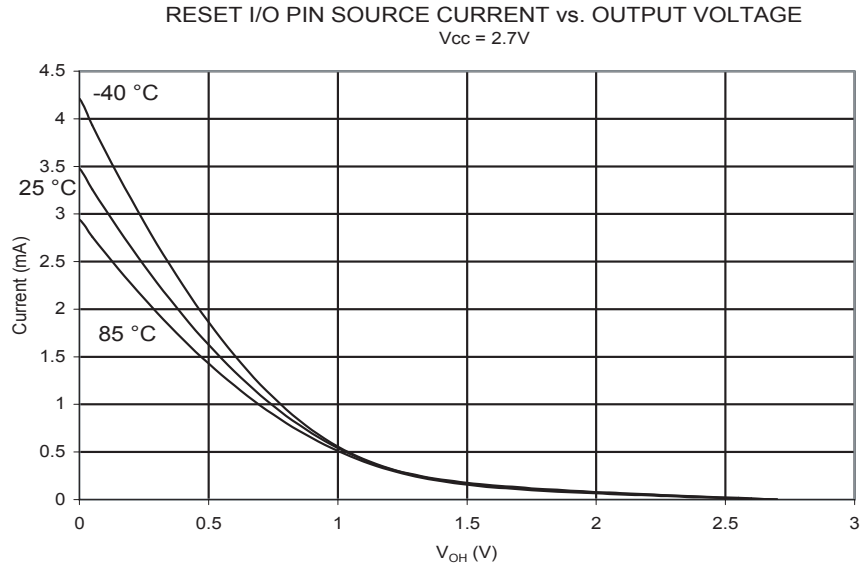


Figure 112. 复位 I/O 引脚源电流和输出电压的关系 (V_{CC} = 1.8V)

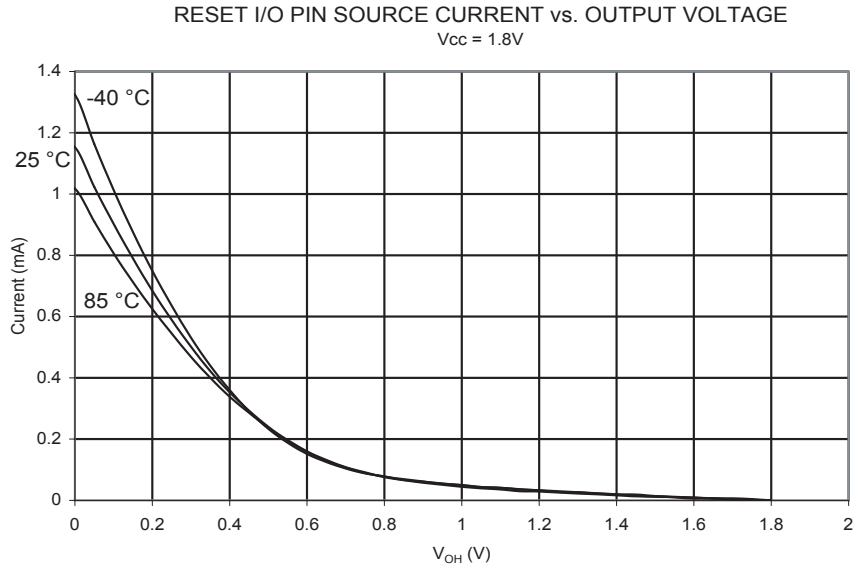


Figure 113. 复位 I/O 引脚吸收电流和输出电压的关系 (V_{CC} = 5V)

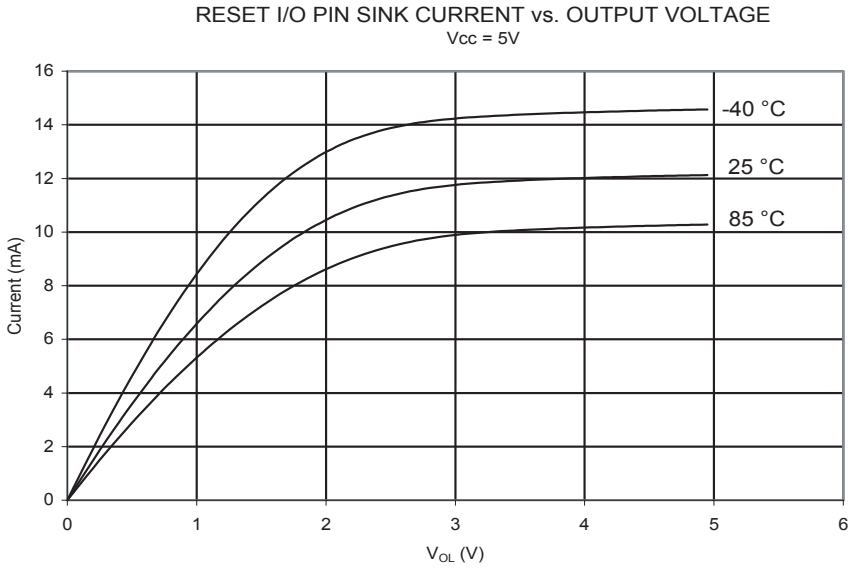


Figure 114. 复位 I/O 引脚吸收电流和输出电压的关系 (V_{CC} = 2.7V)

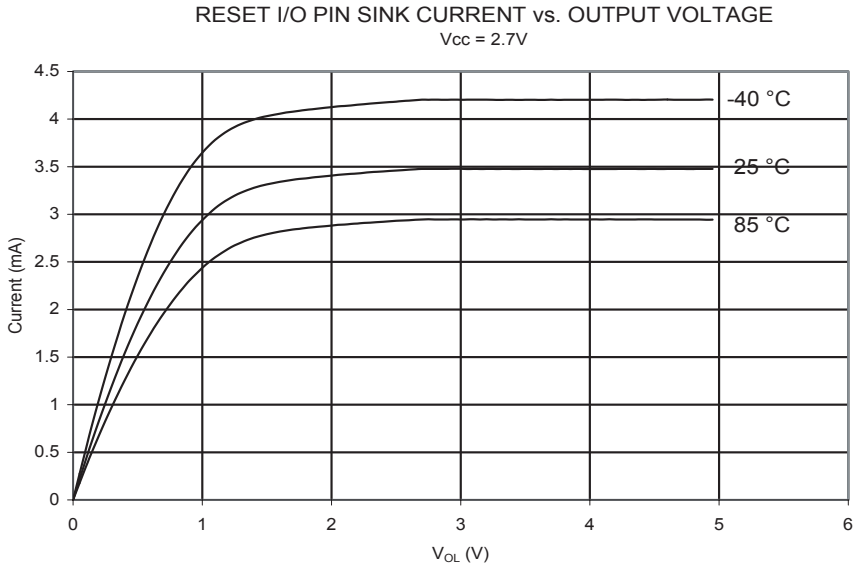
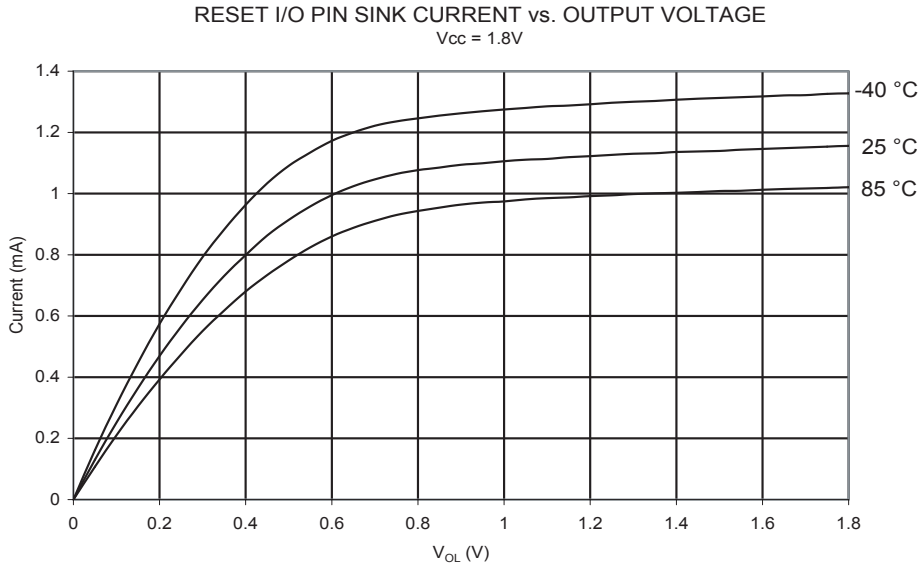


Figure 115. 复位 I/O 引脚吸收电流和输出电压的关系 (V_{CC} = 1.8V)



引脚门限及滞后

Figure 116. I/O 引脚输入门限电压和 V_{CC} 的关系 (V_{IH}, I/O 引脚读出值为 “1”)

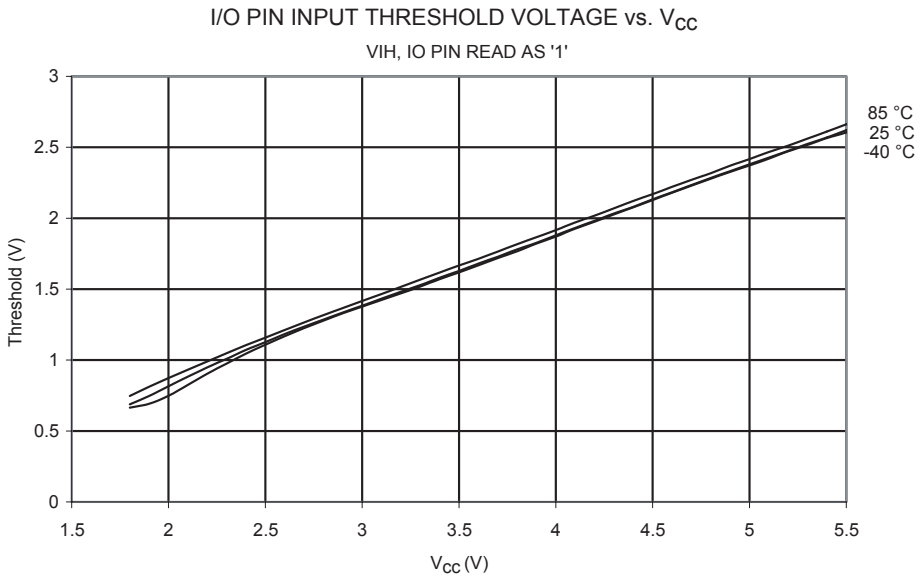


Figure 117. I/O 引脚输入门限电压和 V_{CC} 的关系 (V_{IL}, I/O 引脚读出值为 “0”)

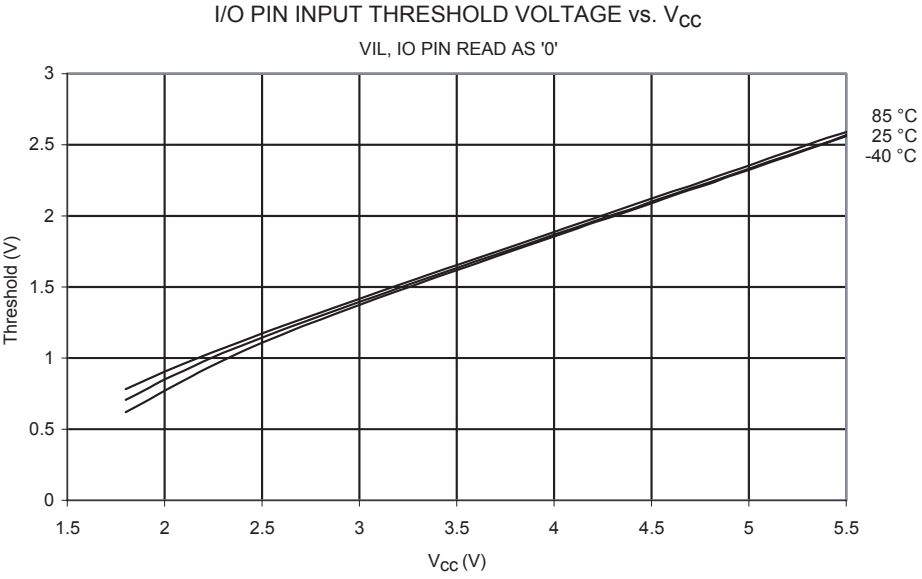


Figure 118. I/O 引脚输入迟滞和 V_{CC} 的关系

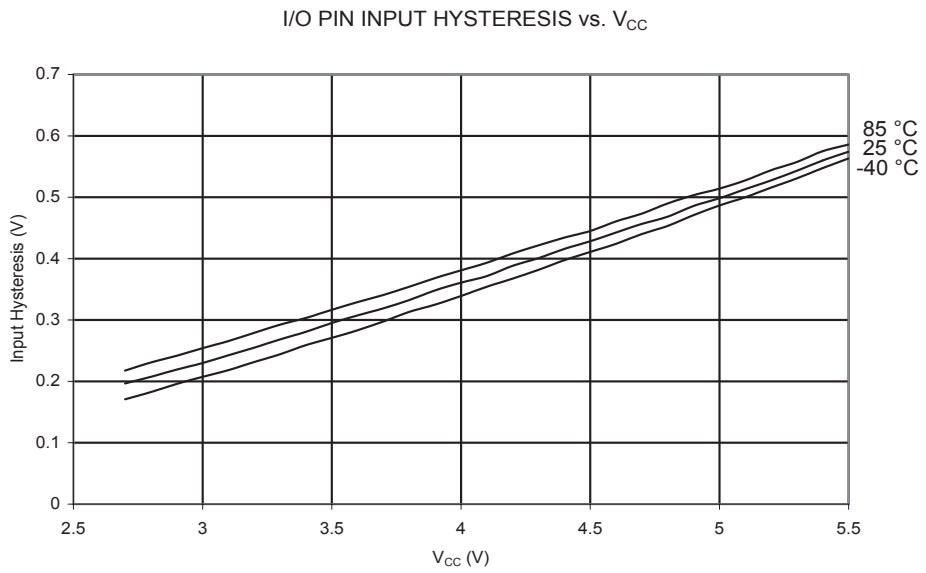


Figure 119. Reset I/O 输入门限电压和 V_{CC} 的关系 (V_{IH},Reset 引脚读出值为 “1”)

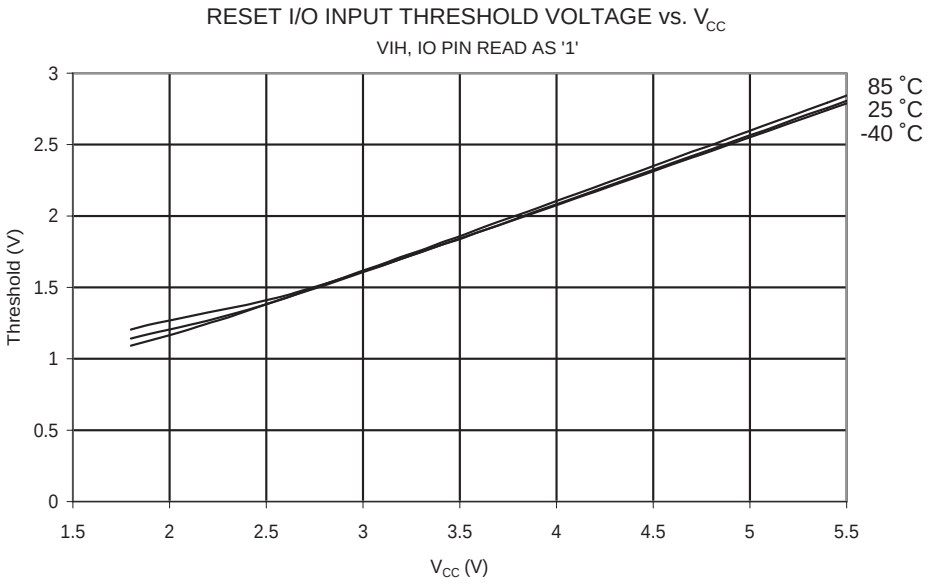


Figure 120. Reset I/O 输入门限电压和 V_{CC} 的关系 (V_{IL},Reset 引脚读出值为 “0”)

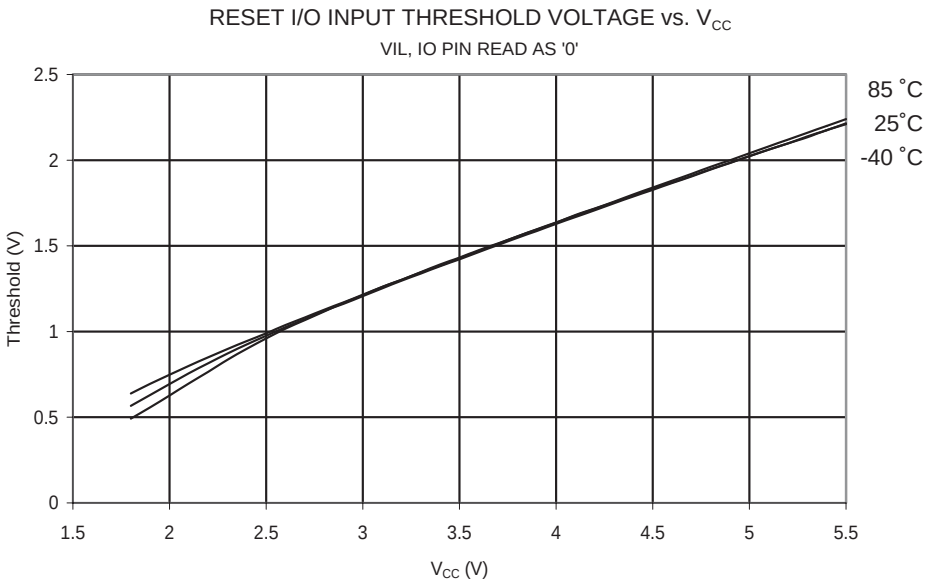


Figure 121. Reset I/O 输入迟滞和 V_{CC} 的关系

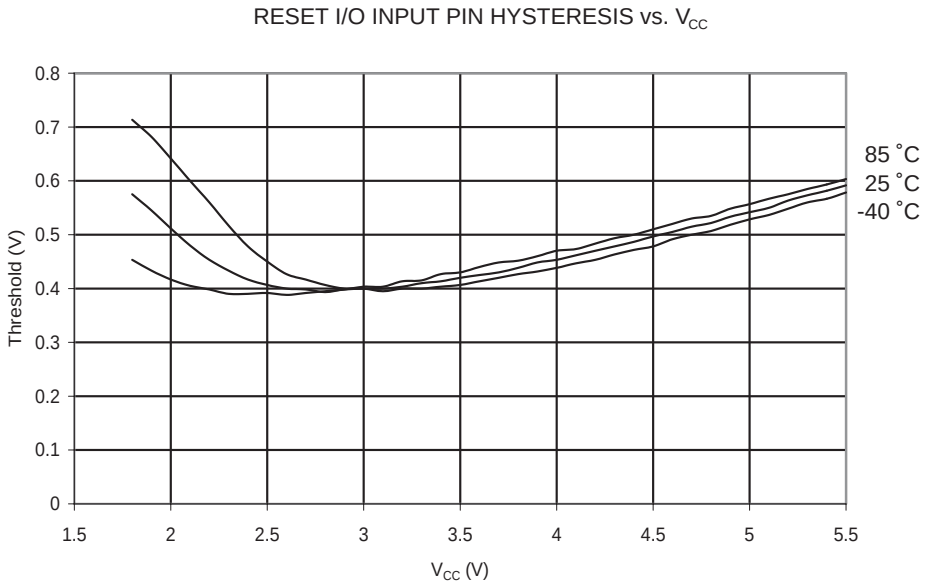


Figure 122. Reset 输入门限电压和 V_{CC} 的关系 (V_{IH},Reset 引脚读出值为 “1”)

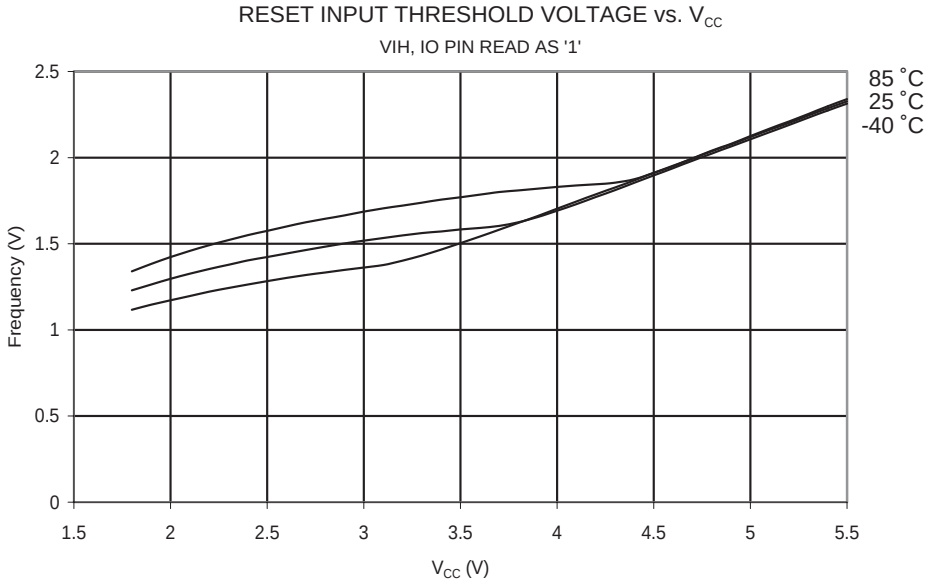


Figure 123. Reset 输入门限电压和 V_{CC} 的关系 (V_{IL},Reset 引脚读出值为 “0”)

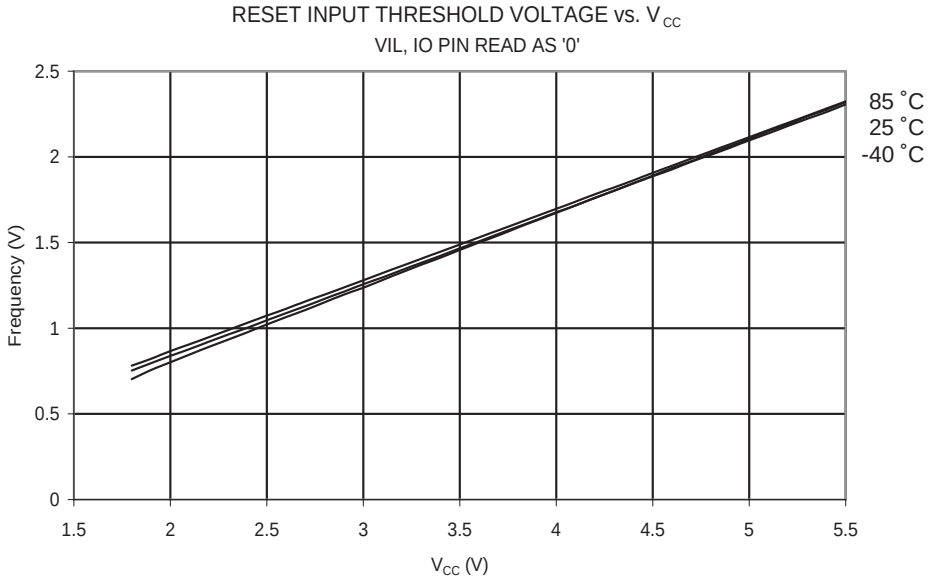
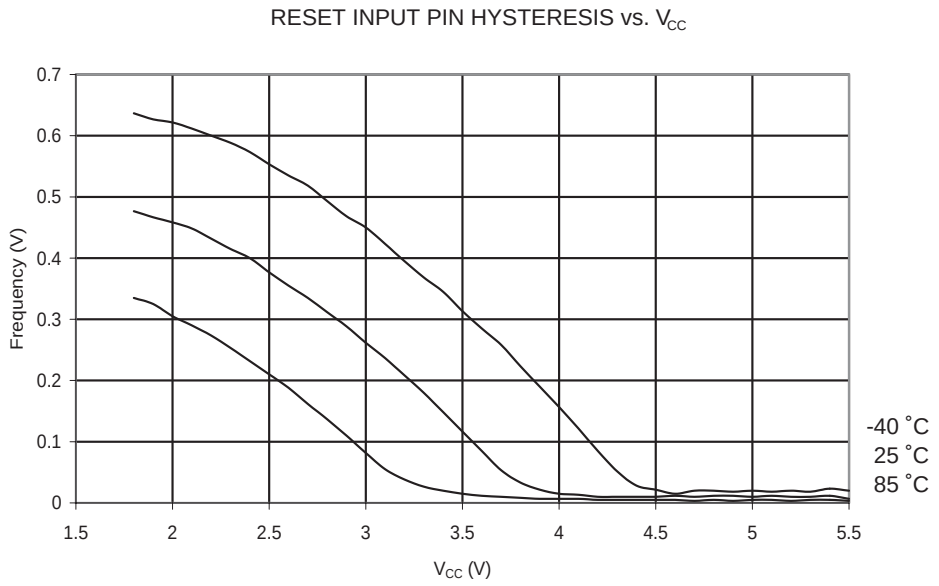


Figure 124. Reset 输入迟滞和 V_{CC} 的关系



BOD 门限值与模拟比较器偏移量

Figure 125. BOD 门限值和温度的关系 (BOD 电平为 4.3V)

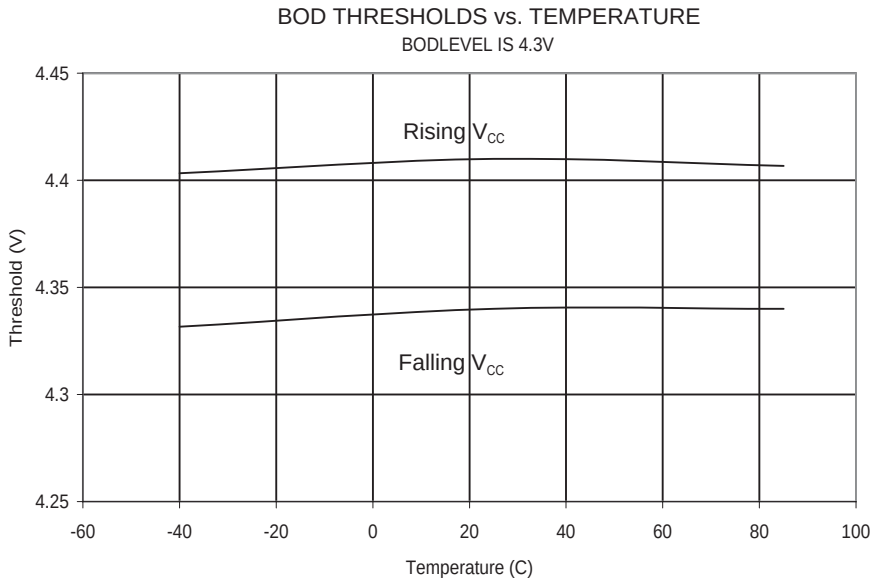


Figure 126. BOD 门限值和温度的关系 (BOD 电平为 2.7V)

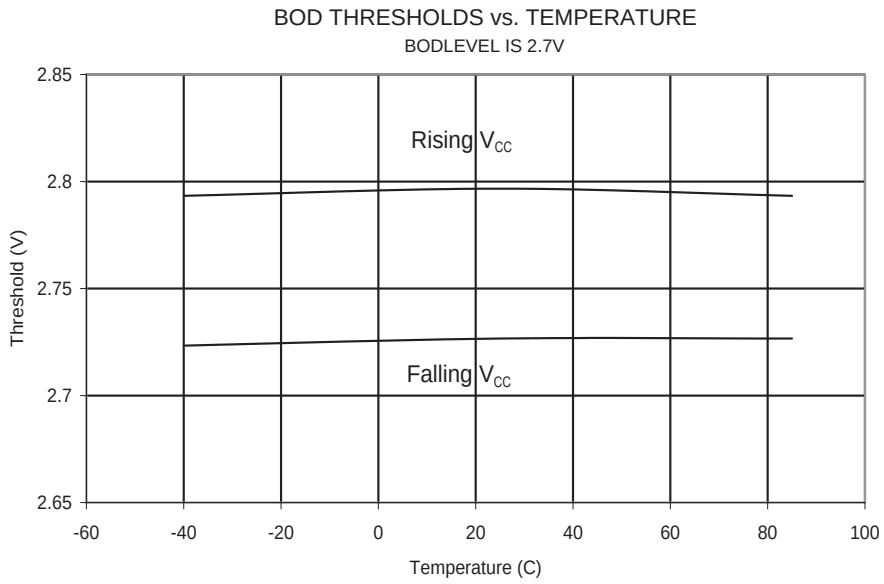


Figure 127. BOD 门限值和温度的关系 (BOD 电平为 1.8V)

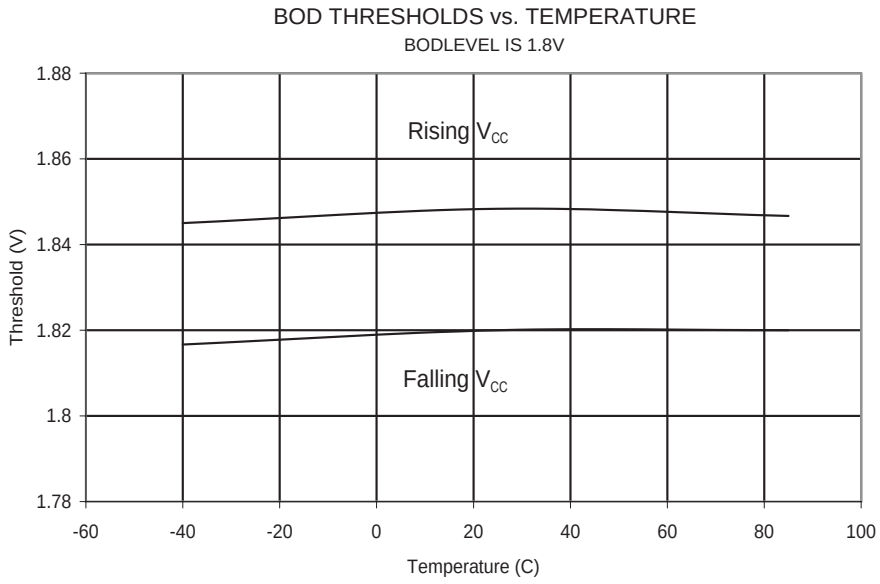
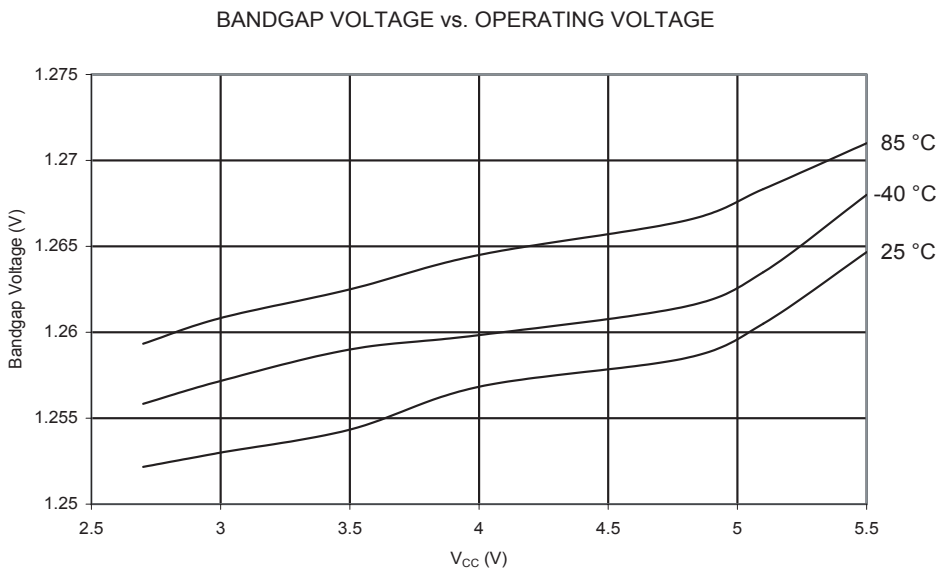


Figure 128. 能隙电压和 V_{CC} 的关系



片内振荡器速度

Figure 129. 看门狗振荡器频率与 V_{CC} 的关系

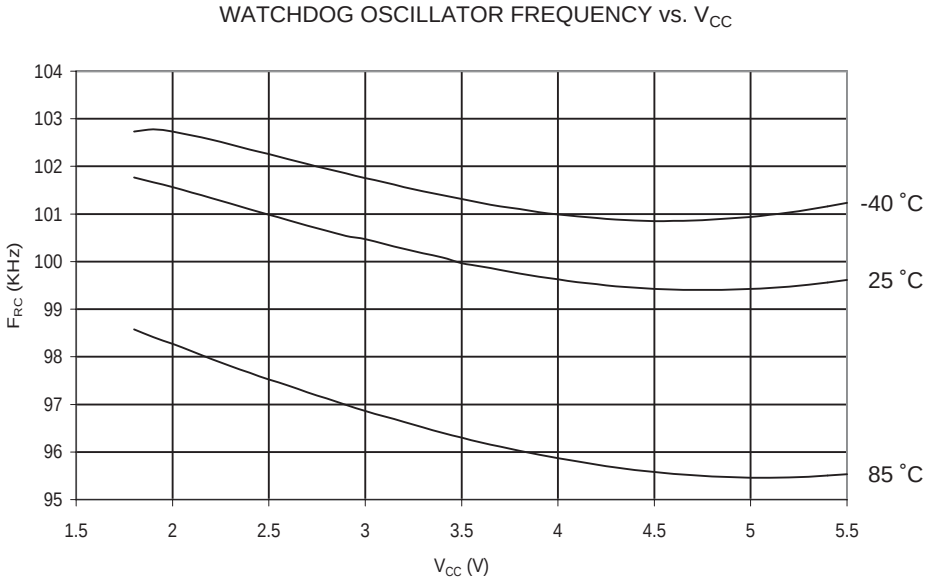


Figure 130. 看门狗振荡器频率与温度的关系

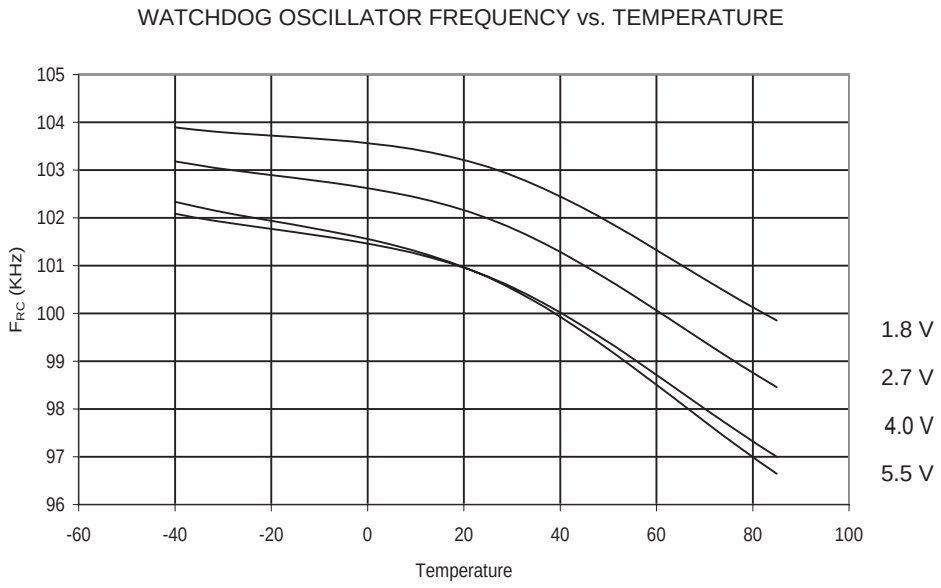


Figure 131. 校准的 8 MHz RC 振荡器频率与温度的关系

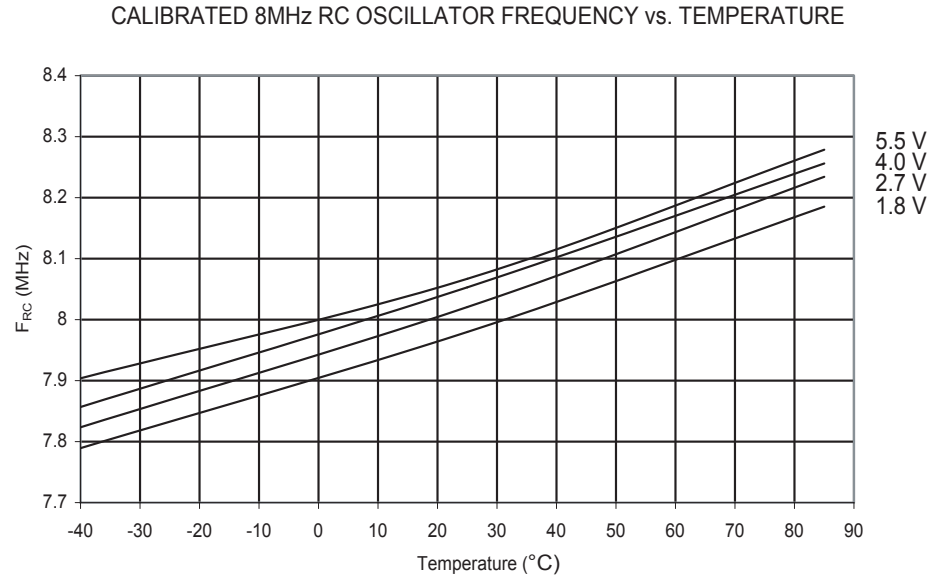


Figure 132. 校准的 8 MHz RC 振荡器频率与 V_{CC} 的关系

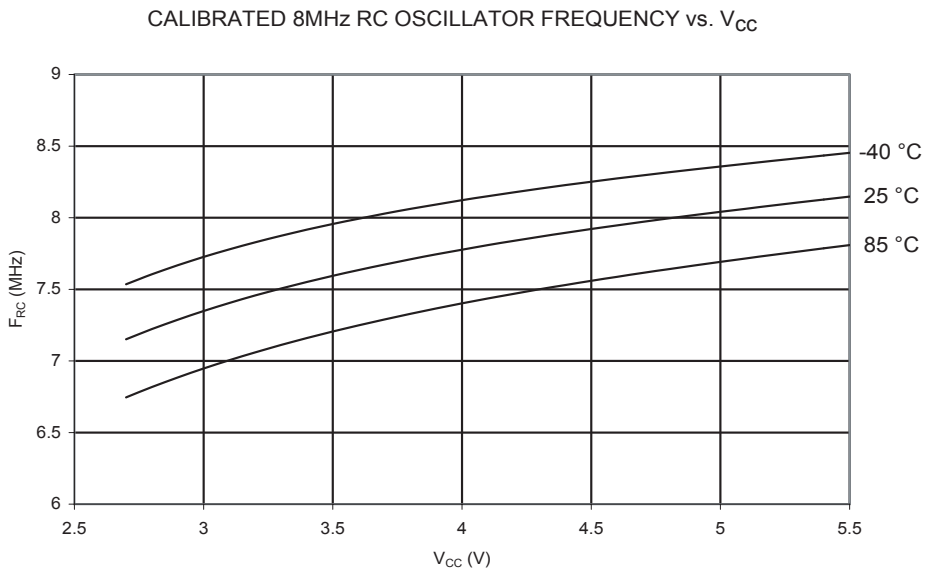


Figure 133. 校准的 8 MHz RC 振荡器频率与 Osccal 值的关系

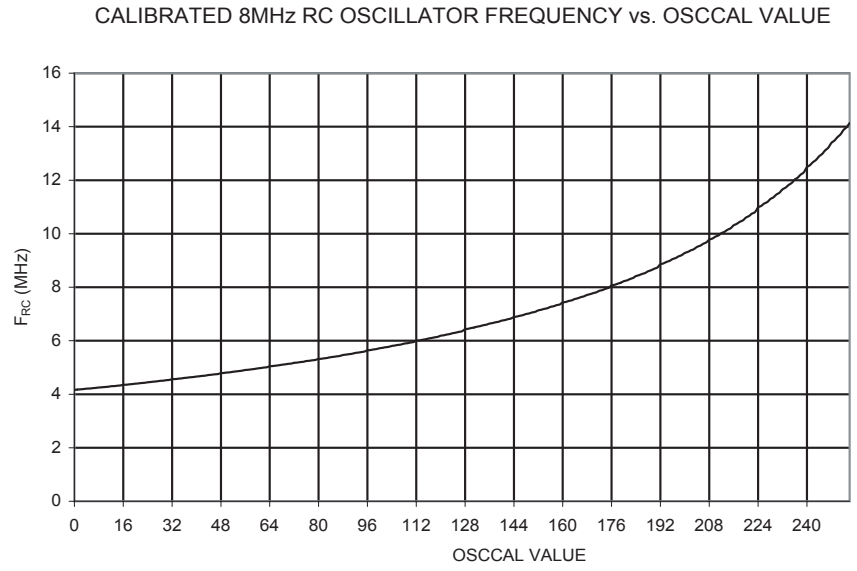


Figure 134. 校准的 4 MHz RC 振荡器频率与温度的关系 I

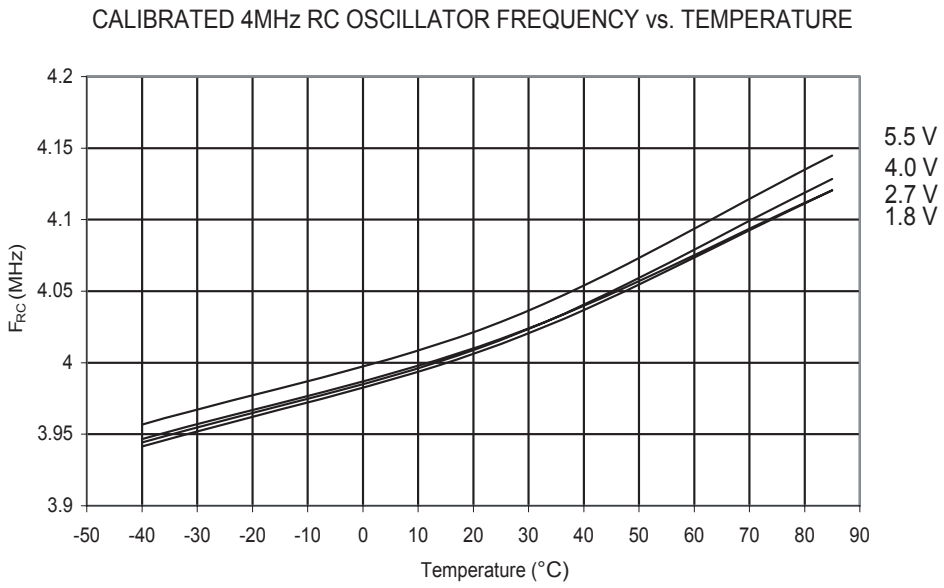


Figure 135. 校准的 4 MHz RC 振荡器频率与 V_{CC} 的关系

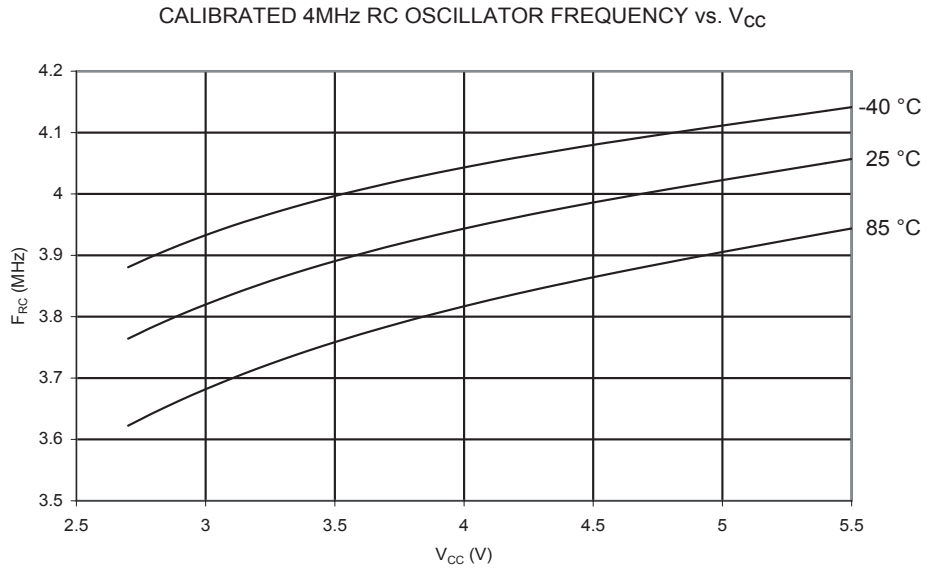
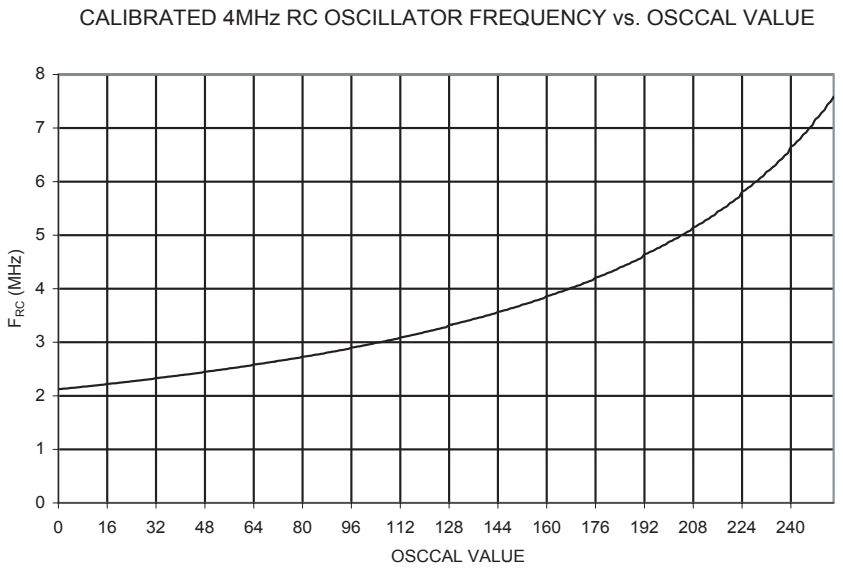


Figure 136. 校准的 4 MHz RC 振荡器频率与 Osccal 值的关系



外设电流

Figure 137. BOD 电流和 V_{CC} 的关系

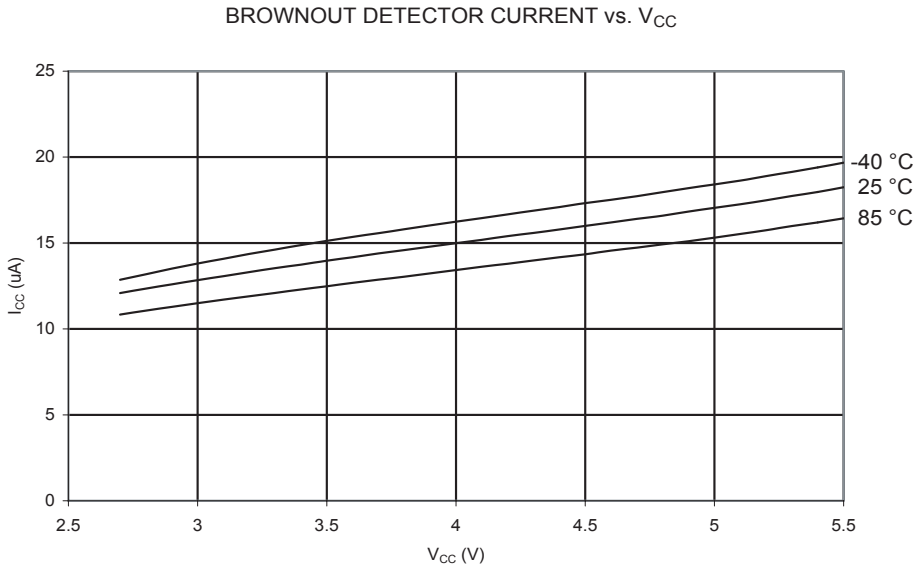


Figure 138. 模拟比较器电流和 V_{CC} 的关系

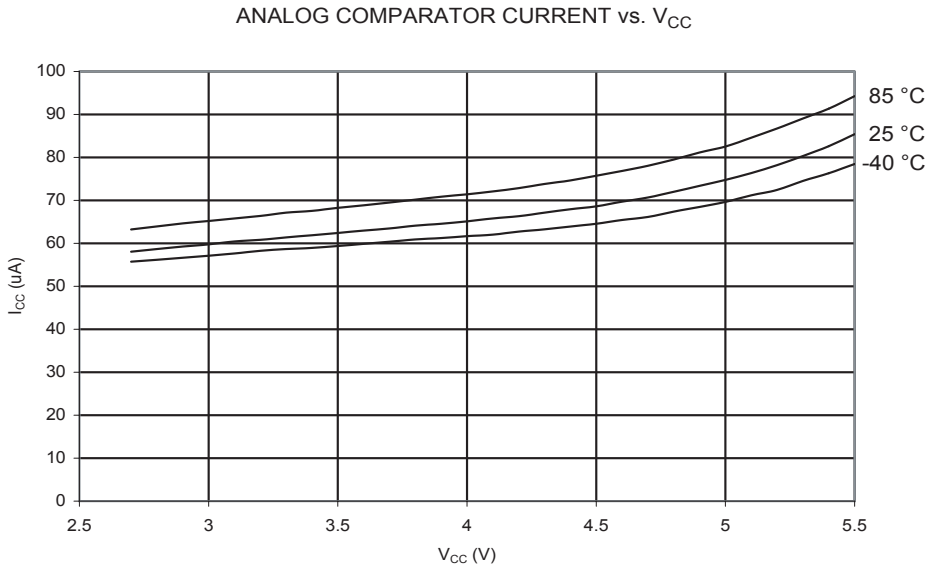
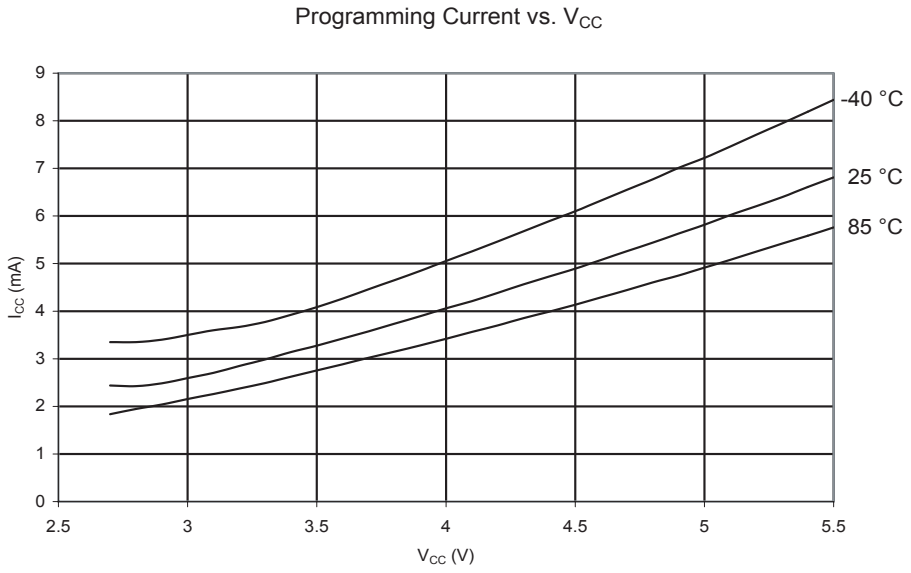


Figure 139. 编程电路和 V_{CC} 的关系



复位电流消耗及复位脉宽

Figure 140. 复位电流和 V_{CC} 的关系 (0.1 - 1.0 MHz，包括流经复位上拉电阻的电流)

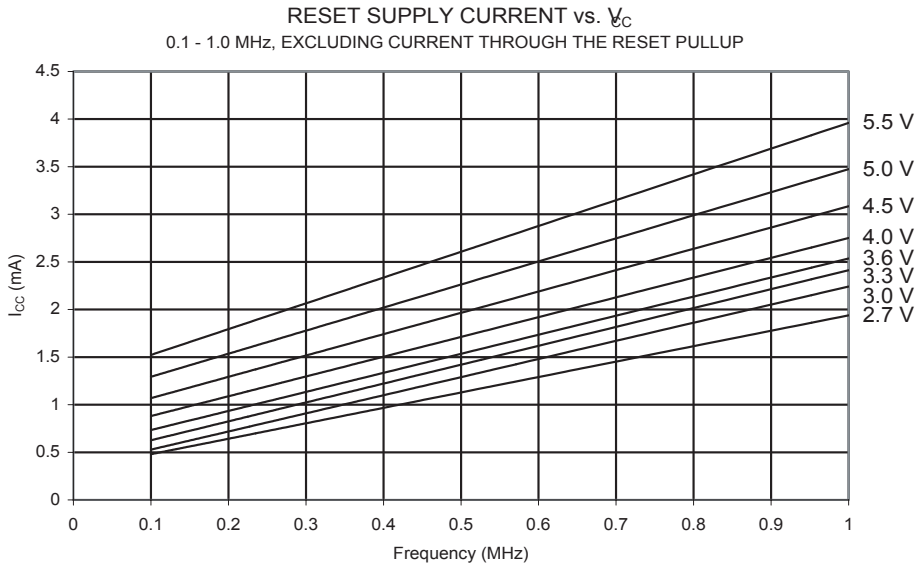


Figure 141. 复位电流和 V_{CC} 的关系 (1 - 20 MHz，包括流经复位上拉电阻的电流)

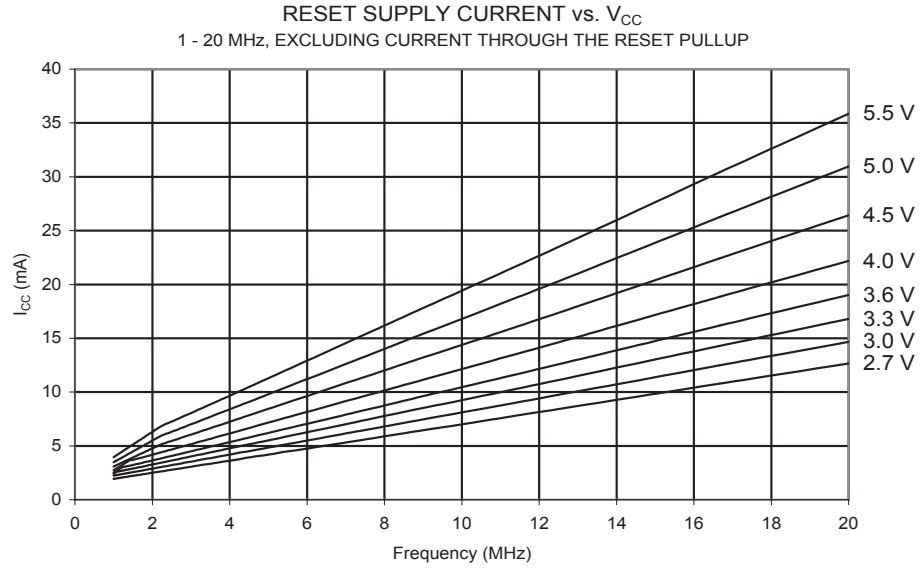
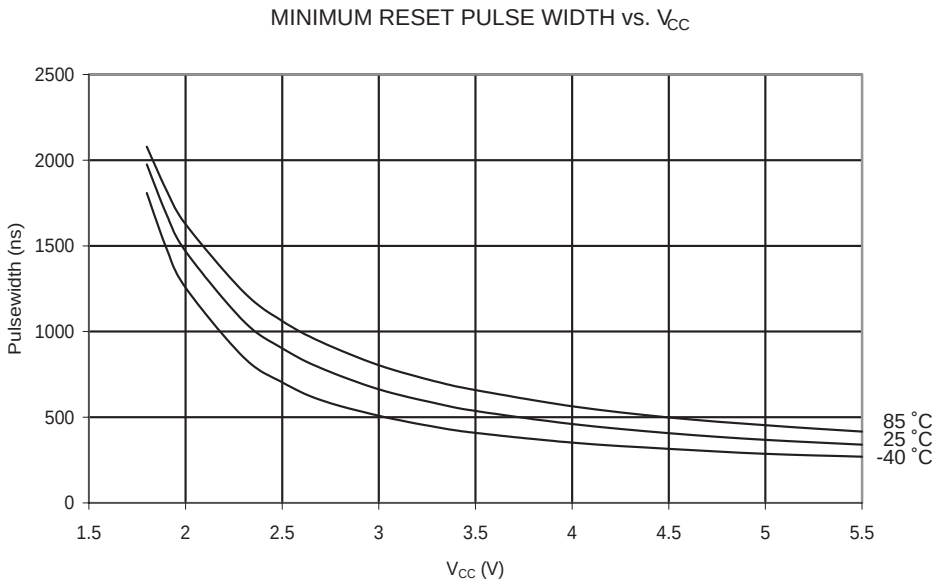


Figure 142. 最小复位脉宽和 V_{CC} 的关系



寄存器概述

地址	名称	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0	页码
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	7
0x3E (0x5E)	保留	–	–	–	–	–	–	–	–	
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	10
0x3C (0x5C)	OCR0B	T/C0 比较寄存器 B								73
0x3B (0x5B)	GIMSK	INT1	INT0	PCIE	–	–	–	–	–	58
0x3A (0x5A)	EIFR	INTF1	INTF0	PCIF	–	–	–	–	–	59
0x39 (0x59)	TIMSK	TOIE1	OCIE1A	OCIE1B	–	ICIE1	OCIE0B	TOIE0	OCIE0A	74, 102
0x38 (0x58)	TIFR	TOV1	OCF1A	OCF1B	–	ICF1	OCF0B	TOV0	OCF0A	74
0x37 (0x57)	SPMCSR	–	–	–	CTPB	RFLB	PGWRT	PGERS	SELFPRGEN	147
0x36 (0x56)	OCR0A	T/C0 比较寄存器 A								73
0x35 (0x55)	MCUCR	PUD	SM1	SE	SM0	ISC11	ISC10	ISC01	ISC00	51
0x34 (0x54)	MCUSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	35
0x33 (0x53)	TCCR0B	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	72
0x32 (0x52)	TCNT0	T/C0 (8 位)								73
0x31 (0x51)	OSCCAL	–	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	25
0x30 (0x50)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	69
0x2F (0x4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1BO	–	–	WGM11	WGM10	97
0x2E (0x4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	100
0x2D (0x4D)	TCNT1H	T/C1 – 计数器寄存器高字节								101
0x2C (0x4C)	TCNT1L	T/C1 – 计数器寄存器低字节								101
0x2B (0x4B)	OCR1AH	T/C1 – 比较寄存器 A 高字节								101
0x2A (0x4A)	OCR1AL	T/C1 – 比较寄存器 A 低字节								101
0x29 (0x49)	OCR1BH	T/C1 – 比较寄存器 B 高字节								102
0x28 (0x48)	OCR1BL	T/C1 – 比较寄存器 B 低字节								102
0x27 (0x47)	保留	–	–	–	–	–	–	–	–	
0x26 (0x46)	CLKPR	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	27
0x25 (0x45)	ICR1H	T/C1 - 输入捕获寄存器高字节								102
0x24 (0x44)	ICR1L	T/C1 - 输入捕获寄存器低字节								102
0x23 (0x43)	GTCCR	–	–	–	–	–	–	–	PSR10	76
0x22 (0x42)	TCCR1C	FOC1A	FOC1B	–	–	–	–	–	–	100
0x21 (0x41)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	40
0x20 (0x40)	PCMSK	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	59
0x1F (0x3F)	保留	–	–	–	–	–	–	–	–	
0x1E (0x3E)	EEAR	–	EEPROM 地址寄存器							15
0x1D (0x3D)	EEDR	EEPROM 数据寄存器								16
0x1C (0x3C)	EECR	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	16
0x1B (0x3B)	PORTA	–	–	–	–	–	PORTR2	PORTA1	PORTA0	56
0x1A (0x3A)	DDRA	–	–	–	–	–	DDA2	DDA1	DDA0	56
0x19 (0x39)	PINA	–	–	–	–	–	PINA2	PINA1	PINA0	56
0x18 (0x38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	56
0x17 (0x37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	56
0x16 (0x36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	56
0x15 (0x35)	GPIOR2	通用功能 I/O 寄存器 2								20
0x14 (0x34)	GPIOR1	通用功能 I/O 寄存器 1								20
0x13 (0x33)	GPIOR0	通用功能 I/O 寄存器 0								20
0x12 (0x32)	PORTD	–	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	56
0x11 (0x31)	DDRD	–	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	56
0x10 (0x30)	PIND	–	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	56
0x0F (0x2F)	USIDR	USI 数据寄存器								137
0x0E (0x2E)	USISR	USISIF	USIOIF	USIPF	USIDC	USICNT3	USICNT2	USICNT1	USICNT0	137
0x0D (0x2D)	USICR	USISIE	USIOIE	USIWM1	USIWM0	USICS1	USICS0	USICLK	USITC	138
0x0C (0x2C)	UDR	UART 数据寄存器 (8 位)								121
0x0B (0x2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM	122
0x0A (0x2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	124
0x09 (0x29)	UBRRL	UBRRH[7:0]								126
0x08 (0x28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	141
0x07 (0x27)	保留	–	–	–	–	–	–	–	–	
0x06 (0x26)	保留	–	–	–	–	–	–	–	–	
0x05 (0x25)	保留	–	–	–	–	–	–	–	–	
0x04 (0x24)	保留	–	–	–	–	–	–	–	–	
0x03 (0x23)	UCSRC	–	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	125
0x02 (0x22)	UBRRH	–	–	–	–	UBRRH[11:8]				126
0x01 (0x21)	DIDR	–	–	–	–	–	–	AIN1D	AIN0D	142
0x00 (0x20)	保留	–	–	–	–	–	–	–	–	

Note:
1. 为了和未来的设备兼容，如果需要保留位应该被写 "0”。保留的 I/O 地址不可以写入。
2. 通过 SBI 和 CBI 指令可直接对地址从 0x00 - 0x1F 的 I/O 寄存器进行位寻址。在这些寄存器中，单个位的值可以通过 SBIS 和 SBIC 指令查询。
3. 一些状态标志可以通过写入逻辑 "1" 来清空。需要注意的是，不同于大多数其他的 AVR，CBI 和 SBI 指令只对一些特殊位有效，因 此可以对那些包含标志位的寄存器进行操作。CBI 和 SBI 指令可使用的范围只能是地址从 0x00 - 0x1F 的寄存器。
4. 当使用特殊的I/O操作命令IN和OUT时，从0x00 - 0x3F的I/O地址会被用到。使用LD 和ST命令可以像操作普通数据空间一样对 I/O 寄存器进行寻址，这时要增加地址 0x20 。

指令集概述

指令	操作数	说明	操作	标志	# 时钟数
算术和逻辑指令					
ADD	Rd, Rr	无进位加法	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	带进位加法	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	立即数与字相加	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	无进位减法	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	减立即数	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	带进位减法	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	带进位减立即数	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	从字中减立即数	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	逻辑与	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	与立即数的逻辑与操作	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	逻辑或	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	与立即数的逻辑或操作	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	异或	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	1 的补码	Rd ← 0xFF − Rd	Z,C,N,V	1
NEG	Rd	2 的补码	Rd ← 0x00 − Rd	Z,C,N,V,H	1
SBR	Rd,K	设置寄存器的位	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	寄存器位清零	Rd ← Rd • (0xFF - K)	Z,N,V	1
INC	Rd	加一操作	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	减一操作	Rd ← Rd − 1	Z,N,V	1
TST	Rd	测试是否为零或负	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	寄存器清零	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	寄存器置位	Rd ← 0xFF	None	1
跳转指令					
RJMP	k	相对跳转	PC ← PC + k + 1	None	2
IJMP		间接跳转到 (Z)	PC ← Z	None	2
RCALL	k	相对子程序调用	PC ← PC + k + 1	None	3
ICALL		间接调用 (Z)	PC ← Z	None	3
RET		子程序返回	PC ← STACK	None	4
RETI		中断返回	PC ← STACK	I	4
CPSE	Rd,Rr	比较，相等则跳过下一条指令	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd,Rr	比较	Rd − Rr	Z, N,V,C,H	1
CPC	Rd,Rr	带进位比较	Rd − Rr − C	Z, N,V,C,H	1
CPI	Rd,K	与立即数比较	Rd − K	Z, N,V,C,H	1
SBRC	Rr, b	寄存器位为 "0" 则跳过下一条指令	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	寄存器位为 "1" 则跳过下一条指令	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	I/O 寄存器位为 "0" 则跳过下一条指令	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	I/O 寄存器位为 "1" 则跳过下一条指令	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	状态寄存器位为 "1" 则跳过下一条指令	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	状态寄存器位为 "0" 则跳过下一条指令	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	相等则跳转	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	不相等则跳转	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	进位位为 "1" 则跳转	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	进位位为 "0" 则跳转	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	大于或等于则跳转	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	小于则跳转	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	负则跳转	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	正则跳转	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	有符号数大于或等于则跳转	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	有符号数负则跳转	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	半进位位为 "1" 则跳转	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	半进位位为 "0" 则跳转	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	T 为 "1" 则跳转	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	T 为 "0" 则跳转	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	溢出标志为 "1" 则跳转	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	溢出标志为 "0" 则跳转	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	中断使能则跳转	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	中断禁止则跳转	if (I = 0) then PC ← PC + k + 1	None	1/2
位和位测试指令					
SBI	P,b	I/O 寄存器位置位	I/O(P,b) ← 1	None	2
CBI	P,b	I/O 寄存器位清零	I/O(P,b) ← 0	None	2
LSL	Rd	逻辑左移	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	逻辑右移	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	带进位循环左移	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1

指令	操作数	说明	操作	标志	# 时钟数
ROR	Rd	带进位循环右移	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=0..6	Z,C,N,V	1
SWAP	Rd	高低半字节交换	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	标志置位	SREG(s) ← 1	SREG(s)	1
BCLR	s	标志清零	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	从寄存器将位赋给 T	T ← Rr(b)	T	1
BLD	Rd, b	将 T 赋给寄存器位	Rd(b) ← T	None	1
SEC		进位位置位	C ← 1	C	1
CLC		进位位清零	C ← 0	C	1
SEN		负标志位置位	N ← 1	N	1
CLN		负标志位清零	N ← 0	N	1
SEZ		零标志位置位	Z ← 1	Z	1
CLZ		零标志位清零	Z ← 0	Z	1
SEI		全局中断使能	I ← 1	I	1
CLI		全局中断禁用	I ← 0	I	1
SES		符号测试标志位置位	S ← 1	S	1
CLS		符号测试标志位清零	S ← 0	S	1
SEV		2 的补码溢出标志置位	V ← 1	V	1
CLV		2 的补码溢出标志清零	V ← 0	V	1
SET		SREG 的 T 置位	T ← 1	T	1
CLT		SREG 的 T 清零	T ← 0	T	1
SEH		SREG 的半进位标志置位	H ← 1	H	1
CLH		SREG 的半进位标志清零	H ← 0	H	1
数据传送指令					
MOV	Rd, Rr	寄存器间复制	Rd ← Rr	None	1
MOVW	Rd, Rr	复制寄存器字	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	加载立即数	Rd ← K	None	1
LD	Rd, X	加载间接寻址数据	Rd ← (X)	None	2
LD	Rd, X+	加载间接寻址数据，然后地址加一	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	地址减一后加载间接寻址数据	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	加载间接寻址数据	Rd ← (Y)	None	2
LD	Rd, Y+	加载间接寻址数据，然后地址加一	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	地址减一后加载间接寻址数据	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	加载带偏移量的间接寻址数据	Rd ← (Y + q)	None	2
LD	Rd, Z	加载间接寻址数据	Rd ← (Z)	None	2
LD	Rd, Z+	加载间接寻址数据，然后地址加一	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	地址减一后加载间接寻址数据	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	加载带偏移量的间接寻址数据	Rd ← (Z + q)	None	2
LDS	Rd, k	从 SRAM 加载数据	Rd ← (k)	None	2
ST	X, Rr	以间接寻址方式存储数据	(X) ← Rr	None	2
ST	X+, Rr	以间接寻址方式存储数据，然后地址加一	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	地址减一后以间接寻址方式存储数据	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	加载间接寻址数据	(Y) ← Rr	None	2
ST	Y+, Rr	加载间接寻址数据，然后地址加一	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	地址减一后加载间接寻址数据	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	加载带偏移量的间接寻址数据	(Y + q) ← Rr	None	2
ST	Z, Rr	加载间接寻址数据	(Z) ← Rr	None	2
ST	Z+, Rr	加载间接寻址数据，然后地址加一	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	地址减一后加载间接寻址数据	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	加载带偏移量的间接寻址数据	(Z + q) ← Rr	None	2
STS	k, Rr	从 SRAM 加载数据	(k) ← Rr	None	2
LPM		加载程序空间的数据	R0 ← (Z)	None	3
LPM	Rd, Z	加载程序空间的数据	Rd ← (Z)	None	3
LPM	Rd, Z+	加载程序空间的数据，然后地址加一	Rd ← (Z), Z ← Z+1	None	3
SPM		保存程序空间的数据	(Z) ← R1:R0	None	-
IN	Rd, P	从 I/O 端口读数据	Rd ← P	None	1
OUT	P, Rr	想 I/O 端口输出数据	P ← Rr	None	1
PUSH	Rr	将寄存器推入堆栈	STACK ← Rr	None	2
POP	Rd	将寄存器弹出堆栈	Rd ← STACK	None	2
MCU 控制指令					
NOP		空操作		None	1
SLEEP		休眠	(对 Sleep 功能，见 specific)	None	1
WDR		复位看门狗	(对 WDR/timer 功能，见 specific)	None	1
BREAK		终止	仅对片上调试	None	N/A

产品信息

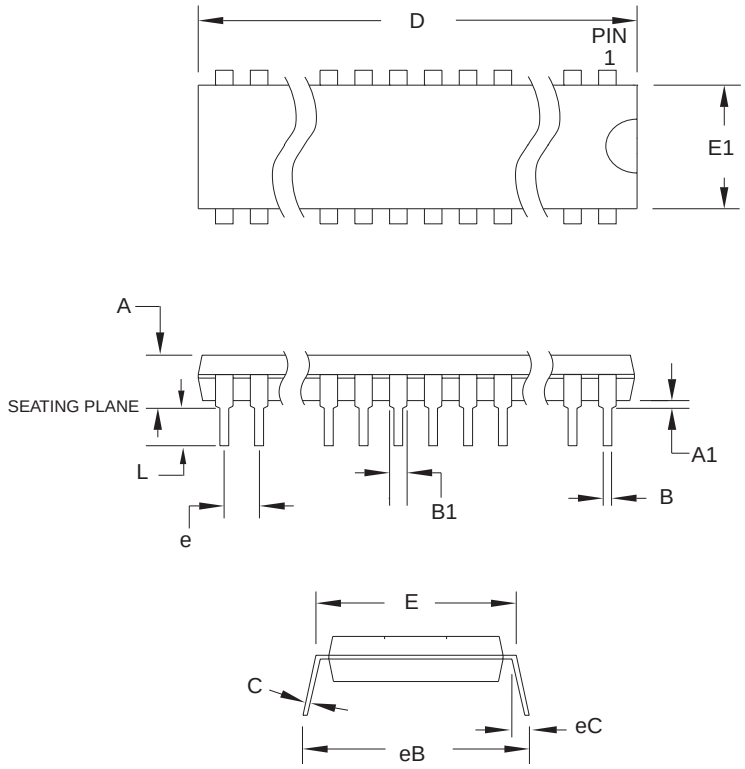
速度 (MHz)	所需电源	产品号	封装 (1)	工作范围
10(3)	1.8 - 5.5V	ATtiny2313V-10PI ATtiny2313V-10PJ(2) ATtiny2313V-10SI ATtiny2313V-10SJ(2)	20P3 20P3 20S 20S	工业级 (-40°C - 85°C)
20(3)	2.7 - 5.5V	ATtiny2313-20PI ATtiny2313-20PJ(2) ATtiny2313-20SI ATtiny2313-20SJ(2)	20P3 20P3 20S 20S	工业级 (-40°C - 85°C)

Note: 1. 产品也可以 wafer 的形式提供，订货信息细节以及最小定货量请与 Atmel 当地机构联系。
2. 可选无铅封装。
3. 参见 P170Figure 81 与 P170Figure 82。

封装类型	
20P3	20 引线， 0.300" 宽， PDIP 封装
20S	20 引线， 0.300" 宽， SOIC 封装

封装信息

20P3



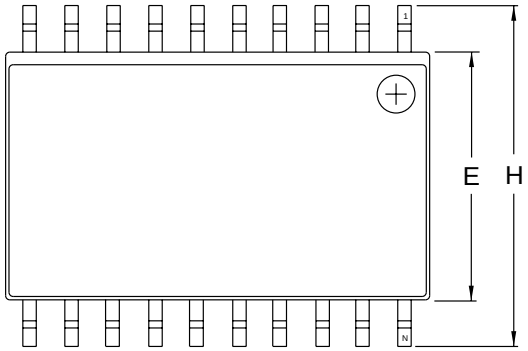
COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	5.334	
A1	0.381	–	–	
D	25.493	–	25.984	Note 2
E	7.620	–	8.255	
E1	6.096	–	7.112	Note 2
B	0.356	–	0.559	
B1	1.270	–	1.551	
L	2.921	–	3.810	
C	0.203	–	0.356	
eB	–	–	10.922	
eC	0.000	–	1.524	
e	2.540 TYP			

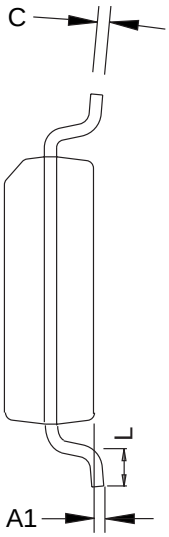
Notes: 1. This package conforms to JEDEC reference MS-001, Variation AD.
2. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25 mm (0.010").

1/12/04

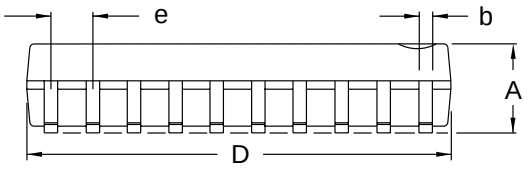
	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**20P3**, 20-lead (0.300"/7.62 mm Wide) Plastic Dual Inline Package (PDIP)	20P3	C



Top View



End View



Side View

COMMON DIMENSIONS
(Unit of Measure = inches)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.0926		0.1043	
A1	0.0040		0.0118	
b	0.0130		0.0200	4
C	0.0091		0.0125	
D	0.4961		0.5118	1
E	0.2914		0.2992	2
H	0.3940		0.4190	
L	0.0160		0.050	3
e	0.050 BSC			

Notes:
1. This drawing is for general information only; refer to JEDEC Drawing MS-013, Variation AC for additional information.
2. Dimension "D" does not include mold Flash, protrusions or gate burrs. Mold Flash, protrusions and gate burrs shall not exceed 0.15 mm (0.006") per side.
3. Dimension "E" does not include inter-lead Flash or protrusion. Inter-lead Flash and protrusions shall not exceed 0.25 mm (0.010") per side.
4. "L" is the length of the terminal for soldering to a substrate.
5. The lead width "b", as measured 0.36 mm (0.014") or greater above the seating plane, shall not exceed a maximum value of 0.61 mm (0.024") per side.

1/9/02

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**20S2**, 20-lead, 0.300" Wide Body, Plastic Gull Wing Small Outline Package (SOIC)	20S2	A

勘误表

本节的版本号与 ATtiny2313 器件的版本号相同。

ATtiny2313 Rev B

- **在擦除操作后读返回值错误。**
- **并行编程不工作。**
- **看门狗定时器中断禁止。**

1. 在擦除操作后读返回值错误

当电源电压低于 2.7 V 时，被擦除的 EEPROM 地址读返回值为 0x00。

解决方法：

如果要在擦除操作之后进行读操作，请使用基本写操作，数据为 0xFF。这同样可以达到擦除的目的。在任何情况下写操作都是正确的。

2. 并行编程不工作

并行编程功能不正确。因此在下列模式下，可能要对器件重新编程：

– 系统内编程禁用 (SPIEN 未编程)。

– 复位禁用 (RSTDISBL 已编程)。

解决方法：

串行编程仍正常工作。为避免上述两种模式，器件可由串行编程。

3. 看门狗定时器中断禁用

若在新的时间溢出前看门狗定时器标志未清零，看门狗将被禁用，中断标志自动清零。这只会出现在只有中断模式中。若将看门狗配置为看门狗时间溢出后复位中断，器件工作正确。

解决方法：

保证在新的看门狗时间溢出前有足够的时间处理时间溢出事件。可通过选择足够长的溢出时间周期来解决。

ATtiny2313 Rev A

版本 A 无。

ATtiny2313 数据手册改变日志

本节所指的页号为本文的相关页码；修订号则为文档的修订号。

从版本 Rev. 2514E-04/04 到版本 Rev. 2514F-07/04 的改动

1. 更新 P1“ 工作电压 ” 。
2. 更新 P51“ 端口 B 的第二功能 ” 。
3. 更新 P152“ 校准字节 ” 。
4. 将 P152Table 69 与 P152Table 70 移到 P152“ 页尺寸 ” 。
5. 更新 P164“ 串行编程算法 ” 。
6. 更新 P165Table 78。
7. 更新 P168“ 直流特性 ” 。
8. 更新 P171“ATtiny2313 典型特性 ” 。
9. 文档中将以下引脚的事件改变：PCINT15 为 PCINT7, EEMWE 为 EEMPE 及 EEWE 为 EEPE。

从版本 Rev. 2514D-03/04 到版本 Rev. 2514E-04/04 的改动

1. 速度等级改变：
   - 12MHz 为 10MHz
   - 24MHz 为 20MHz
2. 更新 P2Figure 1。
3. 更新 P206“ 产品信息 ” 。
4. 更新 P170“ 最大速度与 V_{CC} 的关系 ” 。
5. 更新 P171“ATtiny2313 典型特性 ” 。

从版本 Rev. 2514C-12/03 到版本 Rev. 2514D-03/04 的改动

1. 更新 P21Table 2。
2. 替换 P37“ 看门狗定时器 ” 。
3. 添加 P170“ 最大速度与 V_{CC} 的关系 ” 。
4. 更新 P164“ 串行编程算法 ” 。
5. 将 P198Figure 137 中 mA 改为 µA
6. 更新 P206“ 产品信息 ” 。
   删除 MLF 封装选项。
7. 更新封装图 P207“20P3” 。
8. 更新 C 代码例程。
9. 将自编程使能 SPMEN 改为 to SELFPRGEN。

从版本 Rev. 2514B-09/03 到版本 Rev. 2514C-12/03 的改动

1. 更新 P24“ 标定的片内 RC 振荡器 ” 。

从版本 Rev. 2514A-09/03 到版本 Rev. 2514B-09/03 的改动

1. 在印刷中将UART 改为USART且更新P1“产品特性” 中速度等级与功耗估计 。
2. 更新 P2“ 引脚配置 ” 。
3. 更新 P32Table 15 与 P169Table 80。
4. 更新 P164“ 串行编程算法 ” 中第 5 条。
5. 更新 P168“ 电气特性 ” 。

6. 更新 P170Figure 81 并添加 P170Figure 82。
7. 在 P202“ 寄存器概述 ” 中将 SFIOR 变为 GTCCR。
8. 更新 P206“ 产品信息 ” 。
9. 在 P209“ 勘误表 ” 中加入新的勘误表。

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe

Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia

Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan

9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards

Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive

Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/ High Speed Converters/RF Data-com

Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests

www.atmel.com/literature

Printed on recycled paper.

2543F–AVR–07/04