《AVR单片机C语言开发入门与典型实例》（修订版）

作者：华清远见

第 2 章 ATmega128（L）单片机硬件结构

本章目标

本章重点介绍 ATMEL 公司 AVR 单片机系列中性能最好的一款单片机 ATmega128（L）的硬件结构，主要内容包括以下两个方面。

- ATmega128（L）的内核、时钟、电源管理
- ATmega128（L）的内部资源和外部总线接口

2.1 ATmega128（L）的内核

为了提高性能和并行性，ATmega128（L）的 CPU 集成了算术逻辑单元（ALU）、状态寄存器 SREG、通用工作寄存器组、地址指针寄存器（X、Y 和 Z）、堆栈指针寄存器和 RAM 页 Z 选择寄存器 RAMPZ。

- ATmega128（L）的算术逻辑单元与 32 个通用工作寄存器直接相连，可以完成寄存器和寄存器或寄存器和立即数之间的操作。这些操作分为 3 类：算术、逻辑和位操作，这些操作都可以在一个系统时钟周期内完成。此外，ALU 还支持无符号数、有符号数和浮点数的硬件乘法操作。
- 状态寄存器 SREG 的每一位都为一个标志状态位，代表着不同的含义，包含了最近执行的指令的结果信息。这些信息经常被用来改变程序流程以及实现条件操作。这样，在多数情况下就不需要专门的比较指令了，从而使系统运行更快，代码效率更高。
- 在 ATmega128（L）中，32 个 8 位通用工作寄存器 R0～R31 组成一个通用工作寄存器组，而且大多数的工作寄存器组操作指令都能够在单个周期内直接访问所有的工作寄存器。
- 地址指针寄存器就是指 3 个 16 位的地址指针寄存器 X、Y 和 Z，由通用寄存器 R26～R31 两两合并组成，常用于间接寻址的操作中。通过地址指针寄存器 X、Y 和 Z，我们可以在整个数据空间实现间接寻址的操作。
- 堆栈指针主要用来保存临时数据、局部变量和中断或子程序的返回地址。同时，为了方便使用堆栈指针，ATmega128（L）使用了一个 16 位的栈指针寄存器 SP 用来指示和保存栈顶部地址。
- 由于 ATmega128 不支持超过 64KB 的存储器，而 ATmega128（L）的 Flash 程序存储器为 128KB，因此在 ATmega128（L）中设置了 RAM 页 Z 选择寄存器 RAMPZ 来协助 ELPM/SPM 指令决定访问哪一个程序存储器页，即 RAMPZ 寄存器用于指定由 Z 指针间接寻址访问的地址单元位于哪一个 64KB 页中。

2.2 ATmega128（L）的存储器

ATmega128（L）的存储器分为内部存储器和外部存储器两部分。其中，内部存储器为 ATmega128（L）自身带有的，而外部存储器则需要我们自己扩展。

2.2.1 ATmega128（L）的内部存储器

ATmega128（L）的内部存储器主要包括一下 3 部分：

- 支持可在线 ISP 编程和可在应用 IAP 自编程的 Falsh 程序存储器；
- 内部数据存储器 SRAM；
- E^2PROM。

此外，由于 I/O 空间有很多特殊功能的寄存器，我们也可以认为 I/O 寄存器为内部存储器的一部分。

（1）ATmega128（L）的 Flash 程序存储器的使用寿命最少为 1 万次的擦写循环，总共 128KB，用于存放程序指令代码和常量表。由于大部分的 ATmega128（L）指令为 16 位宽（少数为 32 位宽），并且 ATmega128（L）的程序计数器 PC 为 16 位宽，因此 Flash 程序存储器结构为 64K × 16，同时我们可以寻址整个 64K × 16 程序存储器空间。

（2）在普通模式下，内部数据存储器 SRAM 共有 4096 个字节，由低端开始的 4352 个地址依次分配给 32 个通用工作寄存器、64 个 I/O 寄存器、160 个扩展 I/O 寄存器和 4096 字节的内部 SRAM。

内部数据 SRAM 的寻址方式主要有以下 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；
- 带后增量的间接寻址。

约定：在本书中，用 Rx 表示工作寄存器的地址；$xx 表示 I/O 寄存器的地址（非 SRAM 空间的映射地址）；$xxxx 表示 SRAM 空间地址（也可指 I/O 寄存器在 SRAM 空间的映射地址）。

（3）ATmega128（L）的 E^2PROM 使用寿命至少为 10 万次的擦写循环，编程时间为 8448 个周期（典型值为 8.5ms），总共 4KB，组成一个单独的非易失性数据存储器空间，每一个字节都能被读取或写入（读写是以字节为单位的），并且与 Flash 程序存储器和数据存储器 SRAM 相互独立。

在 ATmega128（L）中，CPU 使用专门的指令对 E^2PROM 进行访问操作，同时通过 SPI、JTAG 和并行编程方式也能对 E^2PROM 读和编程写入操作，

当我们向 E^2PROM 写入数据时，必须遵照一个规范的顺序：

① 等待 EEWE 位变为 0；

② 等待 SPMCSR 寄存器中的 SPMEN 位变为 0；

③ 写新的 E^2PROM 单元地址到地址寄存器 EEARH 和 EEARL；

④ 写新的数据到数据寄存器 EEDR；

⑤ 置位 EEMWE 位，同时将 EEWE 位清零；

⑥ 在 EEMWE 置位后的 4 个时钟周期内置位 EEWE。

（4）ATmega128（L）所有的 I/O 及外围设备的控制寄存器和数据寄存器都分别在 I/O 寄存器空间中，同时部分 I/O 寄存器也被映射到 SRAM 空间中。附录 A 总结归纳了 ATmega128（L）的所有 I/O 寄存器的地址空间分配（包括 I/O 空间地址和 SRAM 空间地址）、名称和功能，其中，部分寄存器只有 SRAM 空间地址。

2.2.2 ATmega128（L）的外部存储器

与传统的 8051 单片机支持外部存储器扩展一样，ATmega128（L）具备外扩存储器的并行接口，并且它能够满足单片机在外扩各种高速或低速的并行接口器件的需要，例如，并行接口的 SRAM 存储器、Flash 存储器，以及并行接口的 LCD 器件、AD 和 DA 器件。

当置位 SRE 位时，ATmega128（L）将使能外部存储器扩展接口 XMEM。此时，ATmega128（L）的一些 I/O 引脚（PA 口、PC 口和 PG0～PG2）将使用其第二功能，即 XMEM 中的地址总线、数据总线和控制信号线，从而构成外部存储器并行扩展接口：

- PA 口作为复用的 8 位数据总线和地址总线低 8 位 AD[7:0]；
- PC 口作为地址总线的高 8 位 A[15:8]，此 8 位地址线可以选择和设定，即可以选择其中几个 I/O 引脚作为地址线；
- PG2 作为地址锁存信号 ALE；
- PG1 作为数据读锁存信号 $\overline{RD}$；
- PG0 作为数据写使能信号 $\overline{WR}$。

利用外部存储器扩展接口 XMEM 外扩 SRAM 存储器的典型电路如图 2.1 所示，而且 XMEM 会自动检测 CPU 正在访问的存储器是内部存储器还是外部存储器。

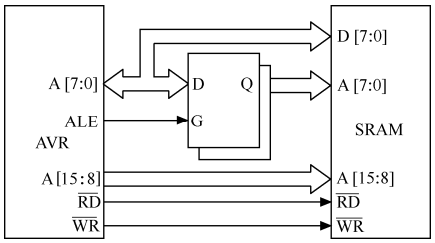

图 2.1 ATmega128（L）外扩 SRAM

由于 ATmega128（L）总的 SRAM 空间寻址能力为 64KB，外部扩展 SRAM 和内部 SRAM 是统一编址的，而且外部扩展的 SRAM 的起始地址紧接在内部 SRAM 之后，因此，在普通模式下，外部可扩展的 SRAM 的实际容量为 65536(64KB) − 32 − 64 − 160 − 4096 字节，即 61184 字节。其中，32 为通用工作寄存器数目（32 字节）；64 为 I/O 空间寄存器（64 字节）；160 为扩展 I/O 寄存器（160 字节）；4096 为内部 SRAM（4096 字节）。

2.3 ATmega128（L）的系统时钟及电源管理

2.3.1 系统时钟

在 ATmega128（L）中主要有 5 种时钟信号，如图 2.2 所示。

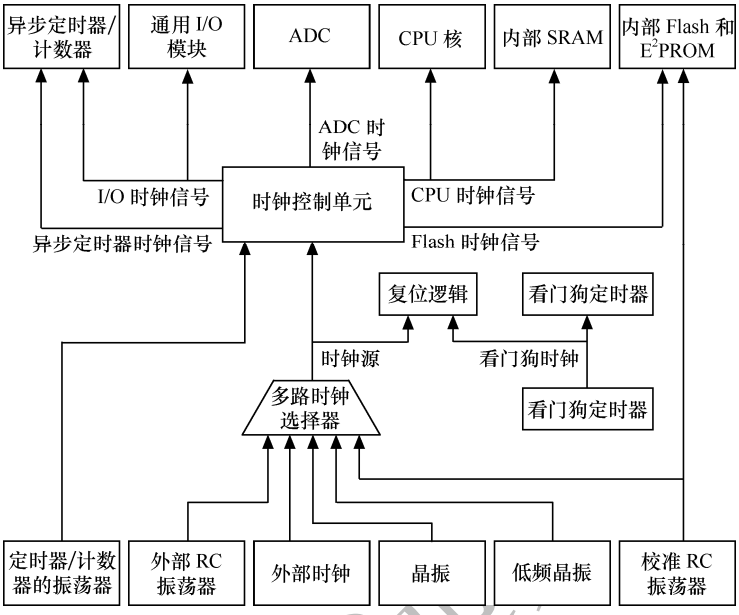

图 2.2 ATmega128（L）的时钟系统

- CPU 时钟信号 CLK_{CPU}：CPU 时钟信号 CLK_{CPU} 连接到与 ATmega128（L）的 CPU 核心，如通用工作寄存器文件、状态寄存器以及保存堆栈指针的数据存储器。如果终止 CPU 时钟信号则将使 CPU 核心停止工作和计算。
- I/O 时钟信号 $CLK_{I/O}$：I/O 时钟信号 $CLK_{I/O}$ 用于主要的 I/O 模块，如定时器/计数器、SPI 和 USART 等。此外，I/O 时钟信号还用于外部中断模块和两线接口 TWI 模块。
- Flash 时钟信号 CLK_{Flash}：Flash 时钟信号 CLK_{Flash} 控制 Flash 接口的操作，此时钟信号通常与 CPU 时钟信号是同步的，即当 CPU 时钟信号处于有效时，Flash 时钟信号同时处于有效状态。
- 异步定时器时钟信号 CLK_{ASY}：异步定时器时钟信号 CLK_{ASY} 用于驱动异步定时器/计数器，这样，即使在休眠模式下，异步定时器/计数器仍可作为实时时钟处于工作状态。
- ADC 时钟信号 CLK_{ADC}：ADC 时钟信号 CLK_{ADC} 是为了提高 ATmega128（L）的 ADC 转换精度而设计的专门时钟信号。这样，我们可以在 ADC 工作的时候停止 CPU 时钟信号 CLK_{CPU} 和 I/O 时钟信号 $CLK_{I/O}$，从而降低数字电路产生的噪声达到提高 ADC 转换精度的目的。

ATmega128（L）有 5 种类型的时钟源，如表 2.1 所示，并且这些时钟源可以通过 ATmega128（L）的 Flash 熔丝位 CKSEL 编程设置。设置好系统时钟源后，时钟源的脉冲再输入到 ATmega128（L）内部的时钟发生器，进而产生上述的 5 种时钟信号。

表 2.1 ATmega128（L）的系统时钟源

熔丝位 CKSEL3～0	系统时钟源
1111-1010	外部晶振/陶瓷振荡器
1001	外部低频晶振
1000-0101	外部 RC 振荡器
0100-0001	标准的内部 RC 振荡器
0000	外部时钟

注意 对所有熔丝位，“1”表示未编程，“0”表示已编程。

（1）当使用外部晶振（外部晶体或陶瓷振荡器）作为系统时钟源时，推荐采用如图 2.3 所示的连接方式。在图 2.3 中，ATmega128（L）片内振荡器的反相放大器输入端 XTAL1 和输出端 XTAL2 与外部一个石英晶体或陶瓷振荡器的两端相连。

在图 2.3 中，不管外部连接的是石英晶体或陶瓷振荡器，电容 C1 和 C2 的大小必须要相同。

（2）当单片机系统的工作频率不是很高时，我们可以使用外部低频晶振（即 32.768kHz 钟表上所用的晶振）作为系统的时钟源，此时熔丝位 CKSEL3～0 必须设置为 1001，从而选择使用低频晶振的时钟源。当对熔丝位 CKOPT 进行编程（1 表示未编程，0 表示已编程）后，我们就可以使用芯片内部与 XTAL1 和 XTAL2 引脚连接的电容（电容值为 36pF），此时就不再需要外部电容 C1 和 C2。

（3）当 ATmega128（L）应用在一些对定时精度要求不高的场合，我们可以使用外部 RC 振荡器，如图 2.4 所示。其中，RC 振荡器的频率可以由式 $f=1/(3RC)$ 粗略估算。

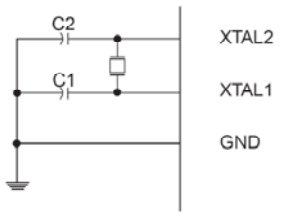

图 2.3　外部晶振作为系统时钟源

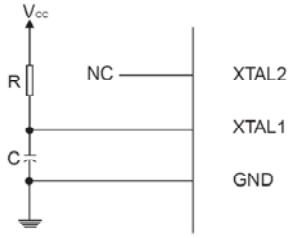

图 2.4　外部 RC 振荡器作为系统时钟源

当使用外部 RC 振荡器时，我们必须从 4 种工作模式中为其选择合适的一种，这主要由系统的工作频率决定，如表 2.2 所示。

表 2.2　　工作频率选择

熔丝位 CKSEL3～0	外部振荡器工作频率范围（MHz）	内部振荡器工作频率（MHz）
0001	≤0.9	1.0
0010	0.9～3.0	2.0
0011	3.0～8.0	4.0
0100	8.0～12.0	8.0

（4）常温下（25℃），当工作电源为 5V 时，ATmega128（L）内部集成的 RC 振荡器可以提供固定的 1.0、2.0、4.0 或 8.0MHz 时钟信号作为系统时钟源。我们可以通过熔丝位 CKSEL 选择这 4 个不同频率的时钟信号中的一个作为系统时钟源，如表 2.2 所示。此时，引脚 XTAL1 和 XTAL2 不再需要连接任何外部元件，系统就拥有一个时钟源。

（5）当整个系统已经有时钟源存在时，我们可以把此时钟作为单片机 ATmega128（L）的时钟源，即 ATmega128（L）使用外部时钟源作为系统时钟，此时熔丝位 CKSEL 必须编程为 0000。外部时钟信号从引脚 XTAL1 输入，XTAL2 引脚没有外接任何元器件（NC 指没有连接）。此外，通过对熔丝位 CKOPT 的编程，我们可以使用芯片内部引脚 XTAL1 与地之间的 36pF 电容。

2.3.2　电源管理与休眠模式

对 ATmega128（L）来说，管理其电源就是为其选择合适的休眠模式。在 ATmega128（L）中，休眠模式意味着应用程序将关掉单片机中未使用的模块，从而达到降低功耗的目的。ATmega128（L）提供了不同的休眠模式以满足不同应用场合对降低功耗的要求。具体来说，ATmega128（L）提供了以下 6 种休眠模式。

- 空闲模式（IDLE）；

- ADC 噪声抑制模式（ADC Noise Reduction）；
- 掉电模式（Power-Down）；
- 省电模式（Power-Save）；
- 待机模式（Standby）；
- 扩展待机模式（Extended Standby）。

当寄存器 MCUCR 中的 SE 位置位后，应用程序执行 SLEEP 指令将使 ATmega128（L）进入上述 6 种休眠模式的一种，而对这 6 种休眠模式进行选择是通过寄存器 MCUCR 中的 SM2～SM0 位来实现的。

在 ATmega128（L）处于休眠模式时，一个使能的中断信号，即唤醒源，如表 2.3 所示，可以把 ATmega128（L）从休眠模式中唤醒。但是，单片机在一个设定的时钟周期内其 CPU 仍然没有开始工作，直到单片机的时钟信号进入稳定状态，此后 CPU 开始进入相应的中断服务程序，最后从 SLEEP 指令后的那条指令重新开始执行指令。

表 2.3　　不同休眠模式下的唤醒源

休 眠 模 式	唤　醒　源					
	INT7:0	TWI 地址匹配	定时器 0	SPM/E^{2}PROM 准备好	ADC	其他 I/O
空闲模式	√	√	√	√	√	√
ADC 噪声抑制模式	√	√	√	√	√	
掉电模式	√	√				
省电模式	√	√	√			
待机模式	√	√				
扩展待机模式	√	√	√			

注意　（1）待机模式和扩展待机模式只有在系统时钟源为外部晶振或陶瓷振荡器时才有效；（2）作为唤醒源的 INT7:0 中的 INT7:4 必须为电平触发。

休眠模式与 ATmega128（L）的最低功耗密切相关。一般来说，为了使 ATmega128（L）系统达到最低功耗，我们要尽可能的利用上述的休眠模式，使尽可能少的模块工作，而不必要的功能则加以禁止，通常以下几个模块要加以考虑：

- ADC；
- 模拟比较器；
- 电压检测（BROWN-OUT）电路；
- 内部电压参考源；
- 看门狗定时器；
- 端口引脚；
- JTAG 接口与片内调试系统。

（1）当 ADC 处于工作状态并在 ATmega128（L）进入休眠模式前没有禁止时，那么在所有的休眠模式下它都处在工作状态。为了降低功耗，我们应该在 ATmega128（L）进入休眠模式之前禁止 ADC。

注意　ADC 关闭后再打开后的第一个 A/D 转换是一个预启动转换，转换结果应该舍弃。

（2）在进入空闲模式和 ADC 噪声抑制模式之前，如果不使用模拟比较器，我们应将其关闭。在其他 4 种休眠模式下，模拟比较器是自动关闭的。此外，还要注意，当模拟比较器被设置成使用内部参考电源时，那么在进入所有的休眠模式之前都应将其关闭，否则内部参考电源在所有的休眠模式下都将处于工作状态。

（3）当 ATmega128（L）系统中不需要使用电压检测（BROWN-OUT）电路，则应关闭该模块。如果电压检测（BROWN-OUT）电路被使能，那么在所有休眠模式下该模块都处于工作状态，从而消耗电能。

（4）前面介绍的 ADC、模拟比较器和电压检测（BROWN-OUT）电路在工作时都将使用内部电压参考源。当这 3 个模块停止工作了，就应该将内部电压参考源关闭。当这 3 个模块再次工作时，应该先启动内部电压参考源。

（5）当看门狗被使能并在 ATmega128（L）进入休眠模式前没有禁止时，它将在所有休眠模式下都处于工作状态，从而一直消耗电能。在深度的休眠模式下，它将成为主要的功耗之一。因此当实际系统不需要看门狗时，应将该模块关闭以降低功耗。

（6）在进入休眠模式之前，所有的端口引脚都应该设置为最低功耗方式，尤其要避免使用输出引脚驱动电阻性负载。此外在休眠模式下，由于时钟信号 $CLK_{I/O}$ 和 CLK_{ADC} 都被停止了，因此输入缓冲区也停止了工作，从而减少了输入逻辑电路消耗不必要的电能。

（7）当通过熔丝位 OCDEN 使能片内调试系统，且进入掉电或省电模式时，主时钟源仍然处于工作状态，从而使总的电能消耗大大增加。在掉电或省电模式下，这将成为主要的功耗之一，为此可以通过禁止熔丝位 OCDEN 或 JTAGEN 或置位 MCUCSR 寄存器中的 JTD 位避免该情况的发生。

2.4 ATmega128（L）的复位及中断

2.4.1 ATmega128（L）的复位逻辑

当 ATmega128（L）复位时，所有的 I/O 寄存器都被设置为初始值，同时程序从复位中断向量处开始执行。复位向量处的指令必须是绝对跳转指令 JMP，以使程序跳转到相应的复位处理程序执行。如果在应用程序中不需要使能任何中断，则中断向量表位置可以放置正常的程序代码。

ATmega128（L）的复位逻辑如图 2.5 所示。

当有一个复位信号产生后，即使系统时钟源停止工作，ATmega128（L）的所有 I/O 端口都会立即被设置为初始值，而不需要任何时钟的帮助。当所有的复位信号消失之后，延时计数器将被激活，以保证 ATmega128（L）正常工作之前系统电源电压达到稳定的状态。

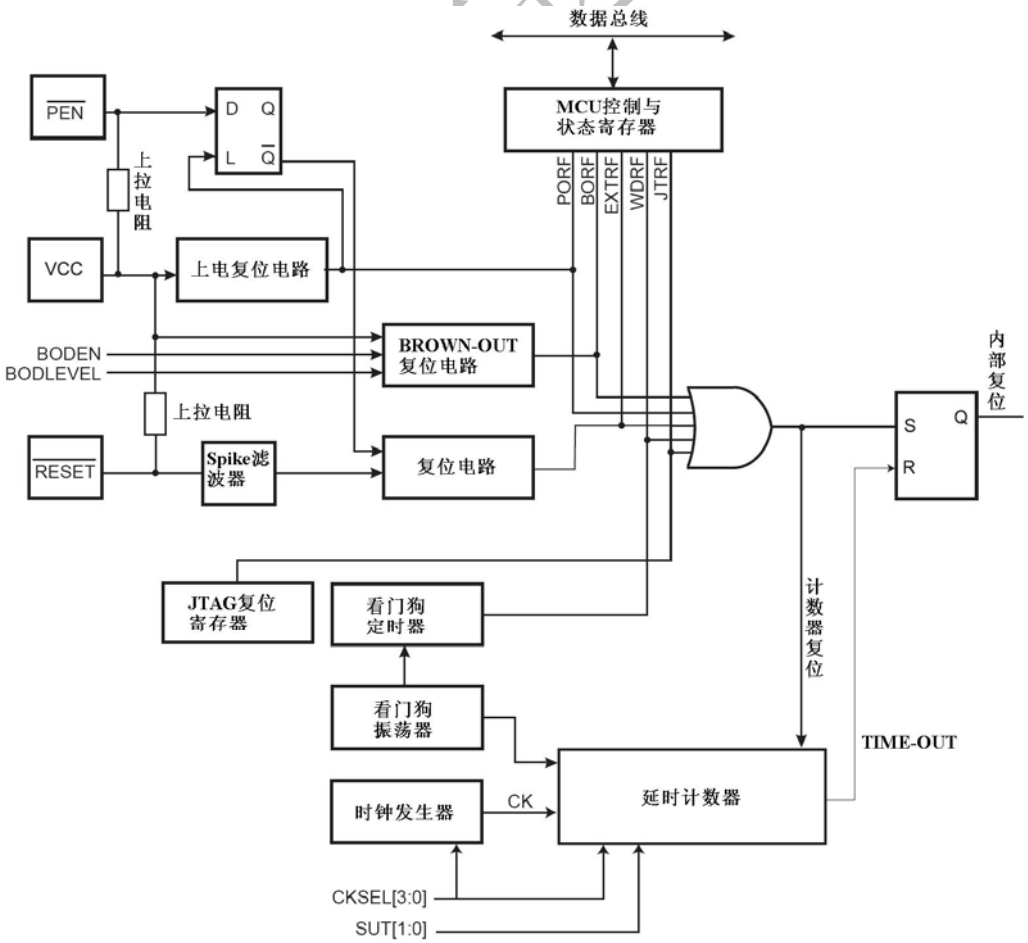

图 2.5 ATmega128（L）的复位逻辑

2.4.2 ATmega128（L）的中断及中断响应

ATmega128(L)总共有35个外部以及内部中断源，相应的中断向量位于Flash程序存储器空间地址0x0000～0x0045处，如表2.4所示。所有的中断向量则构成了ATmega128（L）的中断向量表，并且处于低地址的中断向量所对应的中断源拥有高优先级。这也意味着复位RESET拥有最高优先级。

表 2.4 ATmega128（L）的中断源及相应的中断向量

中断向量号	中断向量地址	中 断 源	中 断 含 义
1	$0000	RESET	上电、外部、掉电检测、看门狗以及 JTAG AVR 复位
2	$0002	INT0	外部中断请求 0
3	$0004	INT1	外部中断请求 1
4	$0006	INT2	外部中断请求 2
5	$0008	INT3	外部中断请求 3
6	$000A	INT4	外部中断请求 4
7	$000C	INT5	外部中断请求 5
8	$000E	INT6	外部中断请求 6
9	$0010	INT7	外部中断请求 7
10	$0012	TIMER2 COMP	定时器/计数器 2 比较匹配
11	$0014	TIMER2 OVF	定时器/计数器 2 溢出
12	$0016	TIMER1 CAPT	定时器/计数器 1 捕捉
13	$0018	TIMER1 COMPA	定时器/计数器 1 比较匹配 A
14	$001A	TIMER1 COMPB	定时器/计数器 1 比较匹配 B
15	$001C	TIMER1 OVF	定时器/计数器 1 溢出
16	$001E	TIMER0 COMP	定时器/计数器 0 比较匹配
17	$0020	TIMER0 OVF	定时器/计数器 0 溢出
18	$0022	SPI，STC	SPI 串行传输完成
19	$0024	USART0，RX	USART0，Rx 数据接收完成
20	$0026	USART0，UDRE	USART0 数据寄存器空
21	$0028	USART0，TX	USART0，Tx 数据发送完成
22	$002A	ADC	ADC 转换完成
23	$002C	EE READY	E^2PROM 准备好
24	$002E	ANALOG COMP	模拟比较器
25	$0030	TIMER1 COMPC	定时器/计数器 1 比较匹配 C
26	$0032	TIMER3 CAPT	定时器/计数器 3 捕捉
27	$0034	TIMER3 COMPA	定时器/计数器 3 比较匹配 A
28	$0036	TIMER3 COMPB	定时器/计数器 3 比较匹配 B
29	$0038	TIMER3 COMPC	定时器/计数器 3 比较匹配 C
30	$003A	TIMER3 OVF	定时器/计数器 3 溢出
31	$003C	USART1，RX	USART1，Rx 数据接收完成
32	$003E	USART1，UDRE	USART1 数据寄存器空
33	$0040	USART1，TX	USART1，Tx 数据发送完成
34	$0042	TWI	两线串行接口
35	$0044	SPM READY	保存程序存储器内容准备好

中断响应为 ATmega128（L）在连续执行指令的过程中碰到需要紧急处理的事件提供了方便。在中断响应的过程中，ATmega128（L）暂时中止执行当前的程序，转去执行中断服务程序，以对紧急事件进行处理。事件处理结束后，ATmega128（L）将自动返回原来的程序执行。

> **注意** 在单片机中，我们称触发中断的事件为“中断源”，而相应的处理程序则称为“中断服务程序”。

同样，我们也就可以理解如果在 ATmega128（L）执行中断服务程序的过程中，又有更紧急的事件发生，则 ATmega128（L）要去执行相应的中断服务程序。

上述两种情况区别在于：在后一种情况下，CPU 将返回前一个中断服务中发生新的更紧急的中断时的程序执行，这一过程称为“中断嵌套”。

下面我们对中断及中断嵌套做几点说明（我们已假定相应中断被使能）。

- 中断源产生中断请求将置位相应的中断标志位。接着，CPU 在执行完当前指令后，在无特殊情况（后面的中断响应过程分析将介绍此特殊情况）下将响应该中断请求，这时硬件将自动的把运行的当前程序的断点地址压入堆栈，即保护断点，然后进入相应的中断服务程序。

> **注意** 断点处的有关信息，如工作寄存器、累加器、程序状态字 PSW、数据指针等，必须由软件来保存（如有需要），这一过程称之为保护现场。

- 中断服务程序是中断处理的具体过程，在程序中是以子程序形式出现，但不能由应用程序直接调用。
- CPU 响应中断后不会自动关闭中断，而保护现场及后面的恢复现场是不允许发生中断的，否则将破坏现场信息。这时就要在现场保护和恢复操作之前关闭单片机的中断。此时，CPU 不响应中断请求。待现场保护和恢复完成之后，如有必要使能新的中断，再开放 ATmega128（L）的中断，从而构成中断嵌套。
- 恢复现场是指将先前在断点处保护起来的有关信息从堆栈中弹出，以确保返回原来程序（中断被响应时 CPU 执行完的指令的下一条指令）继续执行。恢复现场也必须由软件来完成。

在实际的单片机应用中，下列场合经常要考虑使用中断：

- 实现 CPU 与外部设备的速度配合。这主要用在慢速的外部设备与高速的 CPU 进行数据交换的场合。当两者进行数据交换时，我们先启动外部慢速设备，使其做好数据交换的准备，同时 CPU 去执行原来的程序。待外部设备准备好数据之后，就向 CPU 发出请求。这时 CPU 响应中断，在中断服务程序与外部设备交换信息。交换完成后，CPU 回到原来程序继续执行原程序。
- 实现实时控制。这尤其表现在工业自动化上，必须要求对突发事件作出响应。如果采用周期性查询方式检测突发事件，则事件在可能在一个查询周期（比如 100ms）之后才得到响应，这显然是不可取的。这时就要采用中断方式。

> **注意** 实时控制本身就是被控对象可以随时向计算机发出请求并及时得到处理，以确保被控对象工作在期望状态之下。

- 故障的及时处理、精确控制等场合。

2.5 ATmega128（L）的定时器/计数器

ATmega128（L）共有 4 个定时器/计数器：

- 两个 8 位定时器/计数器 Timer/Counter0 和 Timer/Counter2，记为 T/C0 和 T/C2；
- 两个 16 位定时器/计数器 Timer/Counter1 和 Timer/Counter3，记为 T/C1 和 T/C3。

上述 4 个定时器/计数器除了能够实现一般的定时和计数功能外，还具有捕捉、比较匹配、脉宽调制 PWM 输出和实时时钟计数等强大的功能。

2.5.1 8 位定时器/计数器 T/C0

ATmega128（L）中的 T/C0 是一个 8 位的通用多功能定时器/计数器，其主要特点有以下几个方面。

- 单通道计数器；
- 比较匹配发生时清 0 定时器（自动加载，Auto Reload）；
- 无干扰（Glitch-Free）的相位修正 PWM 输出；
- 频率发生器；• 10 位时钟预分频器 1；
- 溢出和比较匹配中断源（TOV0 和 OCF0）；
- 允许使用外部引脚的 32kHz 手表晶振作为独立的时钟源，即实时时钟源 RTC。

在介绍 T/C0 的功能模块之前，我们先给出几个术语的定义，如表 2.5 所示。注意，这些术语的定义只适合对 8 位定时器/计数器的介绍，后面介绍 16 位定时器/计数器 T/C1 和 T/C3 时将会重新定义这几个术语。

表 2.5　　3 个有用的术语

术　　语	含　　义
BOTTOM	当定时器/计数器的值为 0x00 时即达到 BOTTOM
MAX	当定时器/计数器的值为 0xFF 时即达到 MAX
TOP	当定时器/计数器的值为 0xFF 或寄存器 OCR0 中的值时即达到 TOP

下面我们重点介绍 T/C0 的工作模式。

T/C0 的工作模式指的是 T/C0 和 T/C0 的比较匹配输出引脚 OC0 的行为，主要由波形产生模式选择位 WGM01 位和 WGM00 位以及比较匹配输出模式选择位 COM01 位和 COM00 位来控制。T/C0 的工作模式主要有以下 4 种：

- WGM01 位～WGM00 位=0b00 时的普通模式（Normal Mode）；
- WGM01 位～WGM00 位=0b01 时的相位修正 PWM 模式（Phase Correct PWM Mode）；
- WGM01 位～WGM00 位=0b10 时的比较匹配清零定时器 CTC 模式（Clear Timer on Compare Match Mode）；
- WGM01 位～WGM00 位=0b11 时的快速 PWM 模式（Fast PWM Mode）。

（1）WGM01 位～WGM00 位=0b00 时，T/C0 工作在普通模式。这是 T/C0 最简单的工作模式在此模式下，计数器不停地向上计数，计到最大值后（TOP=0xFF）计数器简单地返回到最小值 0x00 重新开始向上计数。当寄存器 TCNT0 为零时，在同一个定时器时钟周期里，T/C0 溢出中断标志位 TOV0 位置位。此时 TOV0 位有点像 T/C0 的第 9 位，只是只能置位，不会清 0。但由于在定时器中断服务程序中，TOV0 位能够由硬件自动清 0，因此可以通过软件提高 T/C0 的分辨率，使其成为一个 9 位计数器。

在普通模式下没有什么特殊情况需要考虑的，应用程序可以随时向寄存器 TCNT0 写入新的计数器数值。此外，比较匹配输出单元可以用来产生中断，但不推荐在普通模式下利用比较匹配输出产生波形，因为这会占用太多的 CPU 时间。

（2）WGM01 位～WGM00 位=0b01 时，T/C0 工作在相位修正 PWM 模式。在此模式下，T/C0 可以产生高精度相位修正的 PWM 波形。当 T/C0 工作在相位修正 PWM 模式下，T/C0 为双向计数器：T/C0 重复地先从 BOTTOM 计数到 MAX（正向计数），然后再从 MAX 开始倒退计数到 BOTTOM（反向计数）。

注意在普通相位 PWM 输出和反向 PWM 输出过程中，当 T/C0 正向计数或反向计数引脚 OC0 上输出电平（即 PWM 波）的行为，如图 2.6 所示。

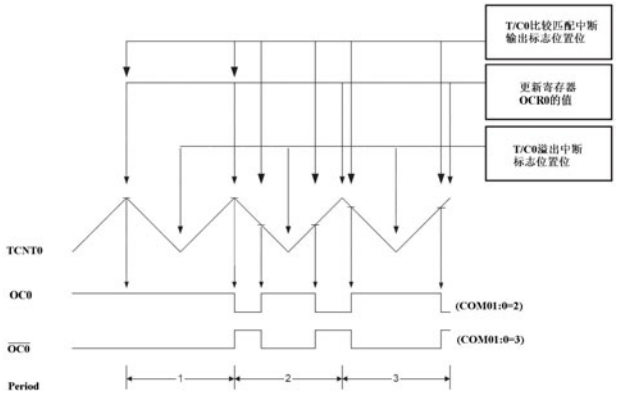

图 2.6　相位修正 PWM 模式

- 设置普通相位 PWM 输出（COM01 位～COM00 位=0b10），在正向计数过程中，当寄存器 TCNT0 中的计数值与寄存器 OCR0 中的值相等时，比较匹配发生，引脚 OC0 上输出电平清 0；在反向计数过程中，当寄存器 TCNT0 中的计数值与寄存器 OCR0 中的值相等时，比较匹配发生，引脚 OC0 上输出电平置位。
- 设置反向 PWM 输出（COM01 位～COM00 位=0b11），在正向计数过程中，当寄存器 TCNT0 中的计数值与寄存器 OCR0 中的值相等时，比较匹配发生，引脚 OC0 上输出电平置位；在反向计数过程中，当寄存器 TCNT0 中的计数值与寄存器 OCR0 中的值相等时，比较匹配发生，引脚 OC0 上输出电平清 0。

注意 在图 2.6 及下面的图 2.7 和图 2.8 中，TCNT0 上的小横条表示寄存器 OCR0 和寄存器 TCNT0 的比较匹配。

（3）WGM01 位～WGM00 位=0b10 时，T/C0 工作在比较匹配清零定时器 CTC 模式，如图 2.7 所示。在 CTC 模式下，我们要注意以下几点。

- 在 CTC 模式下，T/C0 为一个单向加 1 计数器并且用寄存器 OCR0 设置计数器的分辨率。当寄存器 TCNT0 中的值与寄存器 OCR0 的值相等时，比较匹配发生，同时寄存器 TCNT0 清 0。紧接着，TCNT0 将继续向上加 1 计数。
- 寄存器 OCR0 中保存计数器的 TOP 值，即计数器的分辨率。此外，通过设置寄存器 OCR0 的值，应用程序可以方便的控制比较匹配输出的频率。这一点也大大方便了外部事件的计数。

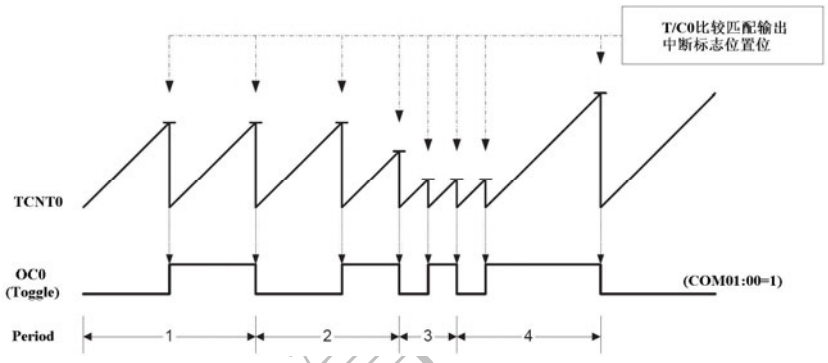

图 2.7　比较匹配清零定时器 CTC 模式

- 在每次寄存器 TCNT0 和 OCR0 中的值相等时，比较匹配输出中断标志位 OCF0 置位。此时，如中断被使能，则可以在相应中断服务中更新 TOP 值，即寄存器 OCR0 中的值。
- 与普通模式相同，在 CTC 模式下，当计数器的计数值由 MAX 变为 BOTTOM 时，T/C0 溢出中断标志位 TOV0 置位。

（4）WGM01 位～WGM00 位=0b11 时，T/C0 工作在快速 PWM 模式，此时可以产生高频率的 PWM 波形。这里我们要注意在普通相位 PWM 输出和反向 PWM 输出过程中，引脚 OC0 上输出电平（即 PWM 波）的行为，如图 2.8 所示。

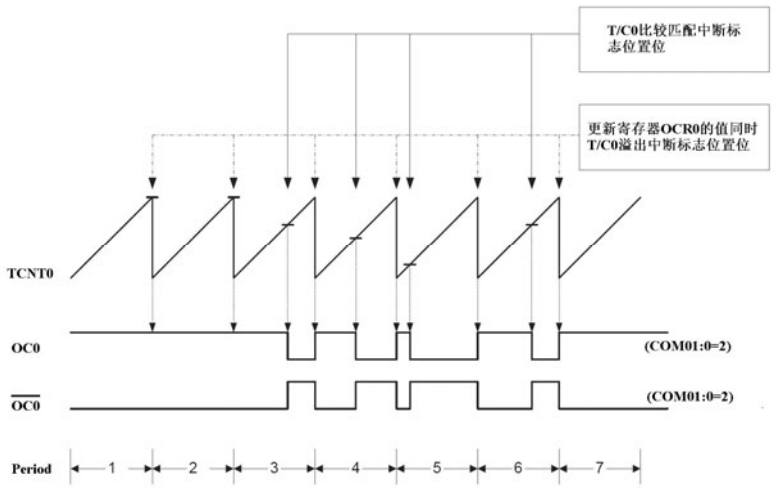

图 2.8　快速 PWM 模式

- 设置普通相位 PWM 输出（COM01 位～COM00 位=0b10），在计数过程中，当寄存器 TCNT0 中的计数值与寄存器 OCR0 中的值相等时，比较匹配发生，引脚 OC0 上输出电平清 0；而当计数值由 MAX 跳变到 BOTTOM 时，引脚 OC0 上输出电平置位。
- 设置反向 PWM 输出（COM01 位～COM00 位=0b11），在计数过程中，当寄存器 TCNT0 中的计数值与寄存器 OCR0 中的值相等时，比较匹配发生，引脚 OC0 上输出电平置位；而当计数值由 MAX 跳变到 BOTTOM 时，引脚 OC0 上输出电平清 0。
- 在快速 PWM 模式下，T/C0 只作为单向加 1 计数器，从 BOTTOM 计数到 MAX，然后再从 BOTTOM 开始加 1 计数，即该模式下产生的 PWM 波是三角形波，而其他 PWM 模式下的 PWM 波为等腰三角形。
- 快速 PWM 模式中只有正向计数，从而使得该模式可以产生比相位修正 PWM 模式频率高一倍的 PWM 波，因此快速 PWM 模式适合于功率调节、整流和 DAC 等场合。
- 当计数器的值达到 MAX 时，T/C0 溢出中断标志位 TOV0 位置位。此时，如果相应中断使能，则可以在相应的中断服务程序中修改寄存器 OCR0 中的值。

2.5.2 8 位定时器/计数器 T/C2

在 ATmega128（L）中，T/C2 也是一个 8 位的通用多功能定时器/计数器，但与 T/C0 有所不同，其主要特点有以下几个方面。

- 单通道计数器；
- 比较匹配发生时清零定时器（自动加载，Auto Reload）；
- 无干扰（Glitch-Free）的相位修正 PWM 输出；
- 频率发生器；
- 10 位时钟预分频器 2；
- 溢出和比较匹配中断源（TOV2 和 OCF2）。

T/C2 的工作模式指的是 T/C2 和 T/C2 的比较匹配输出引脚 OC2 的行为，主要由波形产生模式选择位 WGM21 位和 WGM20 位以及比较匹配输出模式选择位 COM21 位和 COM20 位来控制。T/C2 的工作模式主要有以下 4 种：

- WGM21 位～WGM20 位为 0b00 时的普通模式（Normal Mode）；
- WGM21 位～WGM20 位为 0b01 时的相位修正 PWM 模式（Phase Correct PWM Mode）；
- WGM21 位～WGM20 位为 0b10 时的比较匹配清零定时器 CTC 模式（Clear Timer on Compare Match Mode）；
- WGM21 位～WGM20 位为 0b11 时的快速 PWM 模式（Fast PWM Mode）。

由于 T/C2 的工作模式与 T/C1 基本上完全一样，这里就不再详细介绍了。读者可以参阅 2.8.2 小节中 T/C1 工作模式的介绍，不明白的地方可以阅读器件的手册。

2.5.3 16 位定时器/计数器 T/C1 和 T/C3

在 ATmega128（L）中，T/C1 和 T/C3 是两个功能几乎完全相同的 16 位定时器/计数器，可以实现精确的程序定时（事件管理）、波形产生和信号测量，其主要特点有以下几个方面。

- 真正的 16 位设计，即允许 16 位的 PWM；
- 3 个独立的比较匹配输出单元；
- 双缓冲的比较匹配输出寄存器；
- 一个输入捕捉单元；
- 输入捕捉噪声抑制；
- 比较匹配发生时清零计数器（自动加载，Auto Reload）；
- 无干扰（Glitch-Free）的相位修正 PWM 输出；

- 周期可调的 PWM 输出；
- 频率发生器；• 外部事件计数器；
- 10 个独立的中断源（TOV1、OCF1A、OCF1B、OCF1C、ICF1、TOV3、OCF3A、OCF3B、OCF3C 和 ICF3）。

> **注意** 在下面介绍中，当涉及寄存器或寄存器中的位时，n=1 或 3，表示是 T/C1 和 T/C3 都具有的寄存器和位。

在介绍 T/Cn 的功能模块之前，我们先给出几个术语的定义，如表 2.6 所示。

表 2.6　　3 个有用的术语

术　　语	含　　义
BOTTOM	当定时器/计数器的值为 0x0000 时即达到 BOTTOM
MAX	当定时器/计数器的值为 0xFFFF 时即达到 MAX
TOP	计数器计到设定的最大值时即达到 TOP。TOP 值可以为固定值 0x00FF、0x01FF 或 0x03FF，或是寄存器 OCRnA 或 ICRn 中的值，具体取决于工作模式

T/Cn 的工作模式指的是 T/Cn 和 T/Cn 的比较匹配输出引脚的行为，主要由波形产生模式选择位 WGMn3 位～WGMn0 位以及比较匹配输出模式选择位 COMnx1 位和 COMnx0 位来控制。根据 T/Cn 的 WGMn3 位～WGMn0 位的值的组合，T/Cn 共有 5 大类，16 个不同的工作模式。

- WGMn3 位～WGMn0 位=0 时的普通模式（Normal Mode）；
- WGMn3 位～WGMn0 位=1、2、3、10 或 11 时的相位修正 PWM 模式（Phase Correct PWM Mode）；
- WGMn3 位～WGMn0 位=4 或 12 时的比较匹配清零定时器 CTC 模式（Clear Timer on Compare Match Mode）；
- WGMn3 位～WGMn0 位=5、6、7、14 或 15 时的快速 PWM 模式（Fast PWM Mode）。
- WGMn3 位～WGMn0 位=8 或 9 时的相位频率修正 PWM 模式（Phase and Frequency Correct PWM Mode）。

（1）WGMn3 位～WGMn0 位=0 时，T/Cn 工作在普通模式。这是 T/Cn 最简单的工作模式。在此模式下，计数器不停地向上计数，达到最大值后（TOP=0xFFFF）计数器返回到最小值 0x0000 重新开始向上计数。当寄存器 TCNTn 为 0 时，在同一个定时器时钟周期里，T/Cn 溢出中断标志位 TOVn 位置位。此时 TOVn 位有点像 T/Cn 的第 17 位，只是只能置位，不会清 0。但由于在定时器中断服务程序中，TOVn 位能够由硬件自动清 0，因此可以通过软件提高 T/Cn 的分辨率，使其成为一个 17 位计数器。

（2）WGMn3 位～WGMn0 位=1、2、3、10 或 11 时，T/Cn 工作在相位修正 PWM 模式。在此模式下，T/Cn 可以产生高精度相位修正的 PWM 波形。

当 T/Cn 工作在相位修正 PWM 模式下，T/Cn 为双向计数器：T/Cn 重复的先从 BOTTOM 计数到 TOP（正向计数），然后再从 TOP 开始倒退计数到 BOTTOM（反向计数）。我们要注意在普通相位 PWM 输出和反向 PWM 输出过程中，当 T/Cn 正向计数或反向计数引脚 OCnx 上输出电平的行为，如图 2.9 所示。

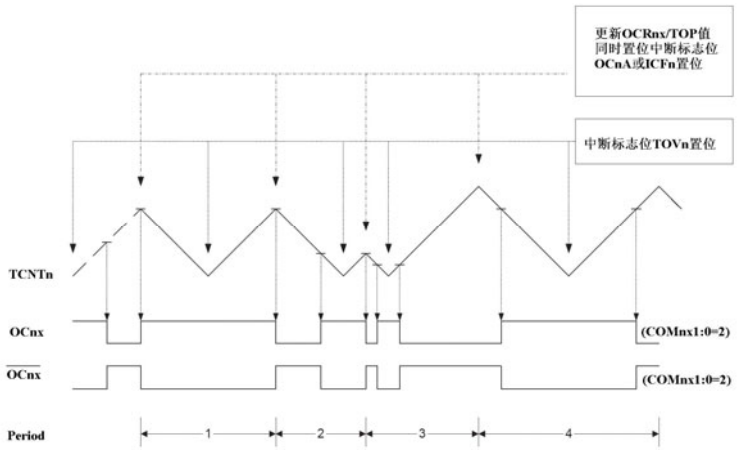

图 2.9　T/Cn 的相位修正 PWM 模式

- 设置普通相位 PWM 输出（COMnx1 位～COMnx0 位=0b10），在正向计数过程中，当寄存器 TCNTn 中的计数值与寄存器 OCRnx 中的值相等时，比较匹配发生，引脚 OCnx 上输出电平清 0；在反向计数过程中，当寄存器 TCNTn 中的计数值与寄存器 OCRnx 中的值相等时，比较匹配发生，引脚 OCnx 上输出电平置位。
- 设置反向 PWM 输出（COMnx1 位～COMnx0 位=0b11），在正向计数过程中，当寄存器 TCNTn 中的计数值与寄存器 OCRnx 中的值相等时，比较匹配发生，引脚 OCnx 上输出电平置位；在反向计数过程中，当寄存器 TCNTn 中的计数值与寄存器 OCRnx 中的值相等时，比较匹配发生，引脚 OCnx 上输出电平清 0。

注意 在图 2.9 及下面的图 2.10、图 2.11 和图 2.12 中，TCNTn 上的小横条表示寄存器 OCRnx 和寄存器 TCNTn 的比较匹配。

（3）WGMn3 位～WGMn0 位=4 或 12 时，T/Cn 工作在比较匹配清 0 定时器 CTC 模式，如图 2.10 所示。在 CTC 模式下，我们要注意以下几点。

- T/Cn 为一个单向加 1 计数器并且用寄存器 OCRnA 或 ICRn 设置计数器的分辨率。当寄存器 TCNTn 中的值与寄存器 OCRnA 的值（WGMn3 位～WGMn0 位=4）或寄存器 ICRn 中的值（WGMn3 位～WGMn0 位=12）相等时，比较匹配发生，寄存器 TCNTn 清 0。紧接着，TCNTn 将继续向上加 1 计数。

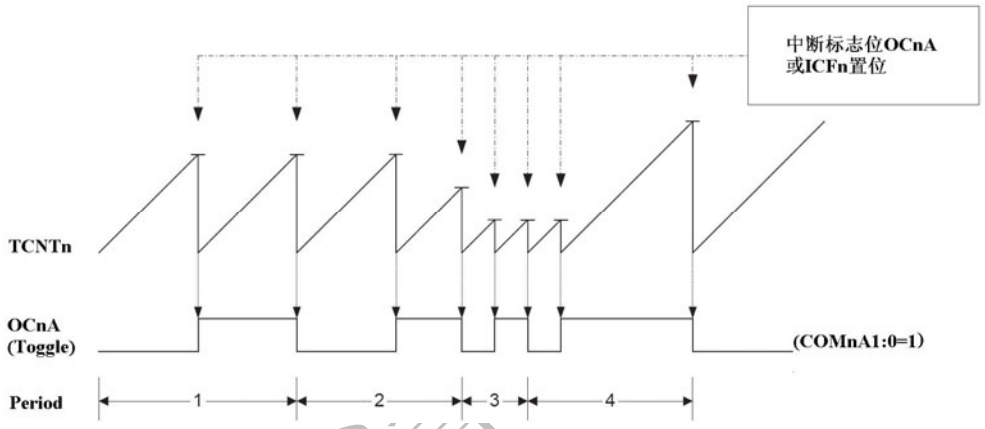

图 2.10　T/Cn 的比较匹配清零定时器 CTC 模式

- 寄存器 OCRnA 或 ICRn 中保存计数器的 TOP 值，即计数器的分辨率。此外，通过设置寄存器 OCRnA 或 ICRn 中的值，应用程序可以方便的控制比较匹配输出的频率。这一点也大大方便了外部事件的计数。
- 在每次寄存器 TCNTn 和 OCRnA 或 ICRn 中的 TOP 值相等时，中断标志位 OCFnA 位或 ICFn 位置位。此时，如中断被使能，则可以在相应中断服务中更新 TOP 值，即寄存器 OCRnA 或 ICRn 中的值。

（4）WGMn3 位～WGMn0 位=5、6、7、14 或 15 时，T/Cn 工作在快速 PWM 模式下。

此时可以产生高频率的 PWM 波形。这里我们要注意在普通相位 PWM 输出和反向 PWM 输出过程中，引脚 OCnx 上输出电平（即 PWM 波）的行为，如图 2.11 所示。

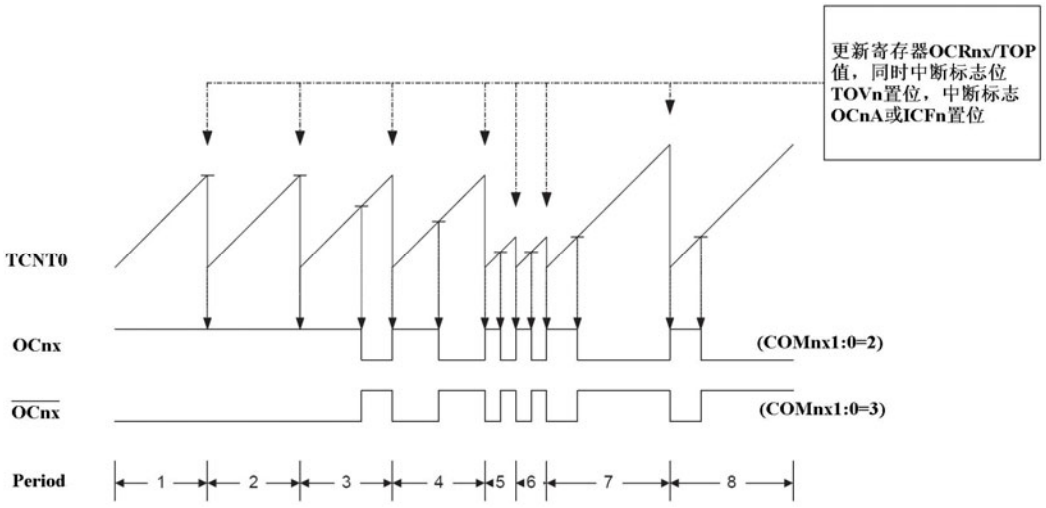

图 2.11　T/Cn 的快速 PWM 模式

- 设置普通相位 PWM 输出（COMnx1 位～COMnx0 位=0b10），在计数过程中，当寄存器 TCNTn 中的计数值与寄存器 OCRnx 中的值相等时，比较匹配发生，引脚 OCnx 上输出电平清 0；而当计数值由 TOP 跳变到 BOTTOM 时，引脚 OCnx 上输出电平置位。
- 设置反向 PWM 输出（COMnx1 位～COMnx0 位=0b11），在计数过程中，当寄存器 TCNTn 中的计数值与寄存器 OCRnx 中的值相等时，比较匹配发生，引脚 OCnx 上输出电平置位；而当计数值由 TOP 跳变到 BOTTOM 时，引脚 OCnx 上输出电平清 0。

在快速 PWM 模式下，我们要注意以下几点。

- T/Cn 只作为单向加 1 计数器，从 BOTTOM 计数到 TOP，然后再从 BOTTOM 开始加 1 计数，即该模式下产生的 PWM 波是三角形波，而其他 PWM 模式下的 PWM 波为等腰三角形。
- 快速 PWM 模式中只有正向计数，从而使得该模式可以产生比相位修正 PWM 模式频率高一倍的 PWM 波，因此快速 PWM 模式适合于功率调节、整流和 DAC 等场合。
- 在快速 PWM 模式下产生的 PWM 的精度（即 TOP 值）可以为固定的 8 位（TOP：0x00FF，WGMn3 位～WGMn0 位=5）、9 位（TOP：0x01FF，WGMn3 位～WGMn0 位=6）或 10 位（TOP：0x03FF，WGMn3 位～WGMn0 位=7）或由寄存器 ICRn（WGMn3 位～WGMn0 位=14）或 OCRnA（WGMn3 位～WGMn0 位=15）定义。
- 当计数器的值达到 MAX 时，T/Cn 溢出中断标志位 TOVn 位置位。另外若 TOP 值是由寄存器 OCRnA 或 ICRn 来定义的，则中断标志位 OCnA 位或 ICFn 位将与 TOVn 位在同一个时钟周期置位。此时，如果相应中断使能，则可以在相应的中断服务程序中修改 TOP 值以及处理数据。

（5）WGMn3 位～WGMn0 位=8 或 9 时，T/Cn 工作在相位频率修正 PWM 模式。在此模式下，T/Cn 可以产生高精度相位和频率修正的 PWM 波形。

当 T/Cn 工作在相位频率修正 PWM 模式下，T/Cn 为双向计数器：T/Cn 重复的先从 BOTTOM 计数到 TOP（正向计数），然后再从 TOP 开始倒退计数到 BOTTOM（反向计数）。这里我们要注意在普通相位 PWM 输出和反向 PWM 输出过程中，当 T/Cn 正向计数或反向计数引脚 OCnx 上输出电平（即 PWM 波）的行为，如图 2.12 所示。

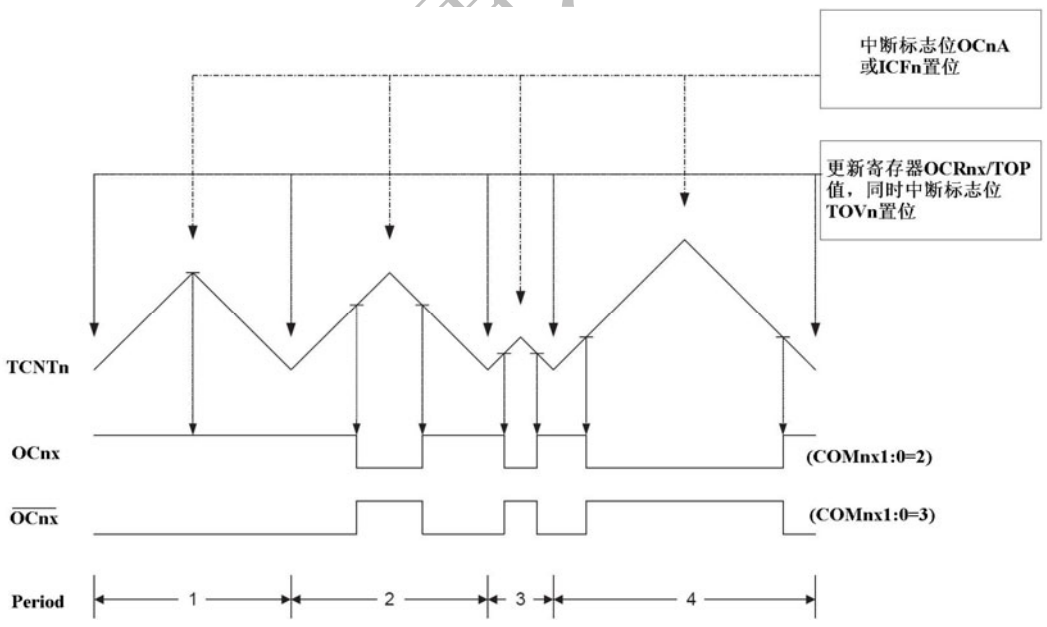

图 2.12　T/Cn 的相位频率修正 PWM 模式

- 设置普通相位 PWM 输出（COMnx1 位～COMnx0 位=0b10），在正向计数过程中，当寄存器 TCNTn 中的计数值与寄存器 OCRnx 中的值相等时，比较匹配发生，引脚 OCnx 上输出电平清 0；在反向计数过程中，当寄存器 TCNTn 中的计数值与寄存器 OCRnx 中的值相等时，比较匹配发生，引脚 OCnx 上输出电平置位。
- 设置反向 PWM 输出（COMnx1 位～COMnx0 位=0b11），在正向计数过程中，当寄存器 TCNTn 中的计数值与寄存器 OCRnx 中的值相等时，比较匹配发生，引脚 OCnx 上输出电平置位；在反向计数过程中，当寄存器 TCNTn 中的计数值与寄存器 OCRnx 中的值相等时，比较匹配发生，引脚 OCnx 上输出电平清 0。

2.6 ATmega128（L）的总线接口

2.6.1 同步外设接口 SPI

ATmega128（L）中的 SPI 接口如图 2.13 所示。

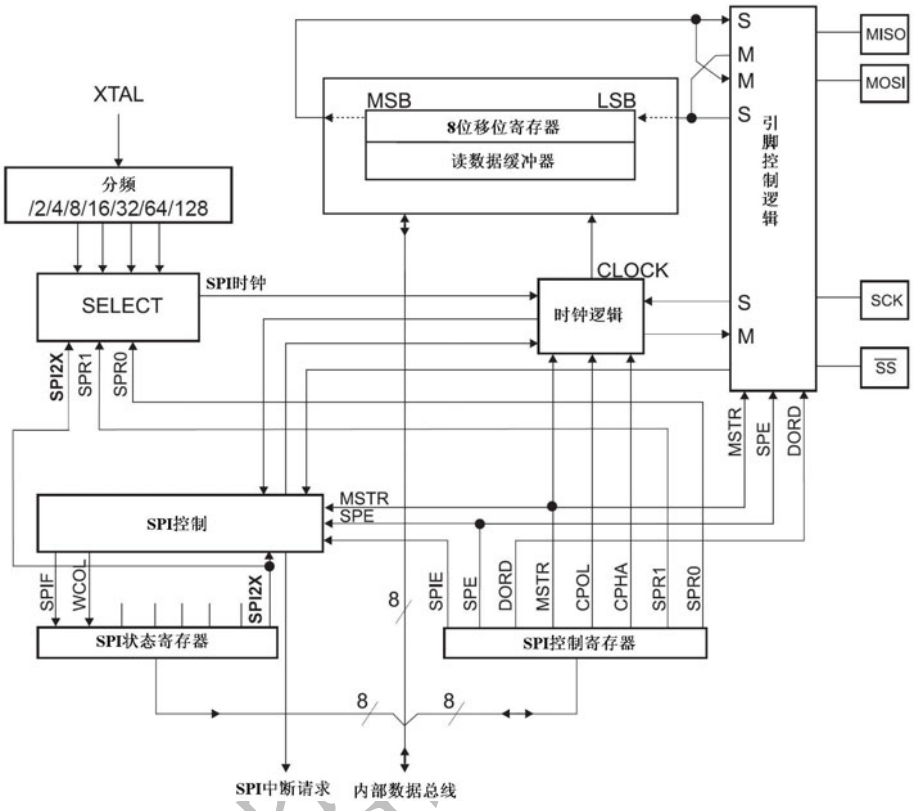

图 2.13 ATmega128（L）中的 SPI 接口

其主要特点如下：

- 全双工、3 线同步数据传输；
- 主机或从机工作模式；
- 数据发送时，可设置低位（LSB）首先发送或高位（MSB）首先发送；
- 7 种可编程的比特率，即每秒传送多少位的数据；
- 数据传输结束触发中断；
- 写冲突标志检测；
- 可以从空闲模式唤醒；
- 主机工作模式时具有倍速模式（CK/2）。

当利用 SPI 为 ATmega128（L）与其他外设或芯片传输数据时，整个数据传输系统由主机和从机两部分构成，如图 2.14 所示。

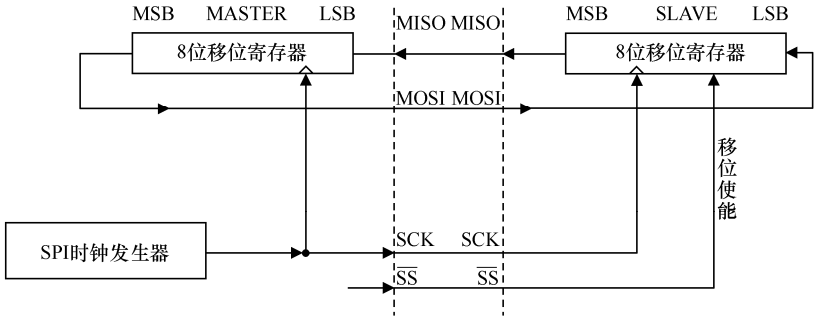

图 2.14　SPI 数据传输系统

下面对图 2.14 做两点说明：

- 整个 SPI 传输系统分为主机和从机两大部分，主要由主从双方的移位寄存器和主机 SPI 时钟发生器组成，且主机为 SPI 数据传输的控制方，即数据传输由主机发起。对单片机 ATmega128（L）来说，其 SPI 既可以作为传输系统的主机，也可以作为从机。
- 当利用 SPI 传输数据时，主机先将其引脚 $\overline{SS}$ 的电平拉低，作为同步数据传输的初始化信号去通知从机进入传输状态。接着，主机的 SPI 时钟发生器开始产生同步时钟信号 SCK，而预先放在两个移位寄存器中的数据就在 SCK 的驱动下开始传输，即主机的数据由引脚 MOSI 进入从机，同时从机数据由引脚 MISO 进入主机，从而实现主从机之间的数据交换。当数据传输完成后，主机将引脚 $\overline{SS}$ 电平拉高，表示数据传输结束。

当 ATmega128（L）的 SPI 接口工作在不同模式时，其数据传输过程如下。

- 当 SPI 的工作模式为主机模式时，引脚 $\overline{SS}$ 不受硬件电路的控制，而是在 SPI 开始数据传输之前由主机的应用程序将其拉为低电平。此后，随着 8 位的数据写入主机的寄存器 SPDR，主机 SPI 接口将启动时钟发生器，并且数据开始移位 8 次由引脚 MOSI 进入从机，而从机中的数据则由引脚 MISO 进入主机。当 8 位数据传输完成后，SPI 时钟发生器将停止工作，同时置位中断标志位 SPIF 位。
- 当 SPI 的工作模式为从机模式时，引脚 $\overline{SS}$ 由外部电路控制。当引脚 $\overline{SS}$ 电平被拉高时，引脚 MISO 为三态并且 SPI 接口处于休眠中。此时，从机的应用程序可以更新从机寄存器 SPDR，而不会造成数据才 SCK 驱动下移出。当引脚 $\overline{SS}$ 电平被拉低时，寄存器 SPDR 中的数据在 SCK 驱动下移出并且在一个字节移出后置位中断标志位 SPIF。

在 ATmega128（L）的 SPI 硬件电路中，在发送方向上只有一个缓冲器，而在接收方向上有两个缓冲器，即在数据发送时应用程序一定要等到移位过程全部结束后才能对 SPI 数据寄存器 SPDR 执行写操作，而在接收数据时，则需要在下一个字符移位过程结束前访问 SPI 数据寄存器 SPDR 读取当前接收到的字符，否则第一个字节数据将丢失。

2.6.2　通用同步/异步串行接口 USART0 和 USART1

ATmega128（L）的通用同步和异步串行接口有两个：USART0 和 USART1（下面记为统称为 USART）。这两个接口都是高度灵活的串行通信设备，其主要特点有以下几点。

- 全双工操作方式（相互独立的串行数据接收寄存器和数据发送寄存器）；
- 支持异步或同步操作；
- 可以是主机或从机提供同步操作所需的时钟；
- 高精度的波特率发生器；
- 支持含有 5、6、7、8 或 9 个数据位和 1 个或 2 个停止位的串行数据帧结构；
- 硬件支持的奇偶校验位发生和检验；
- 数据过速检测（Data OverRun Detection）；
- 帧错误检测；
- 具有错误起始位检测功能的噪声滤波器和数字低通滤波器；
- 三个独立的中断：发送完成中断，发送数据寄存器空中断以及接收完成中断；
- 支持多处理器通信；
- 倍速异步通信模式。

ATmega128（L）的全双工通用同步/异步串行接口 USART 的结构如图 2.15 所示。在图 2.15 中。

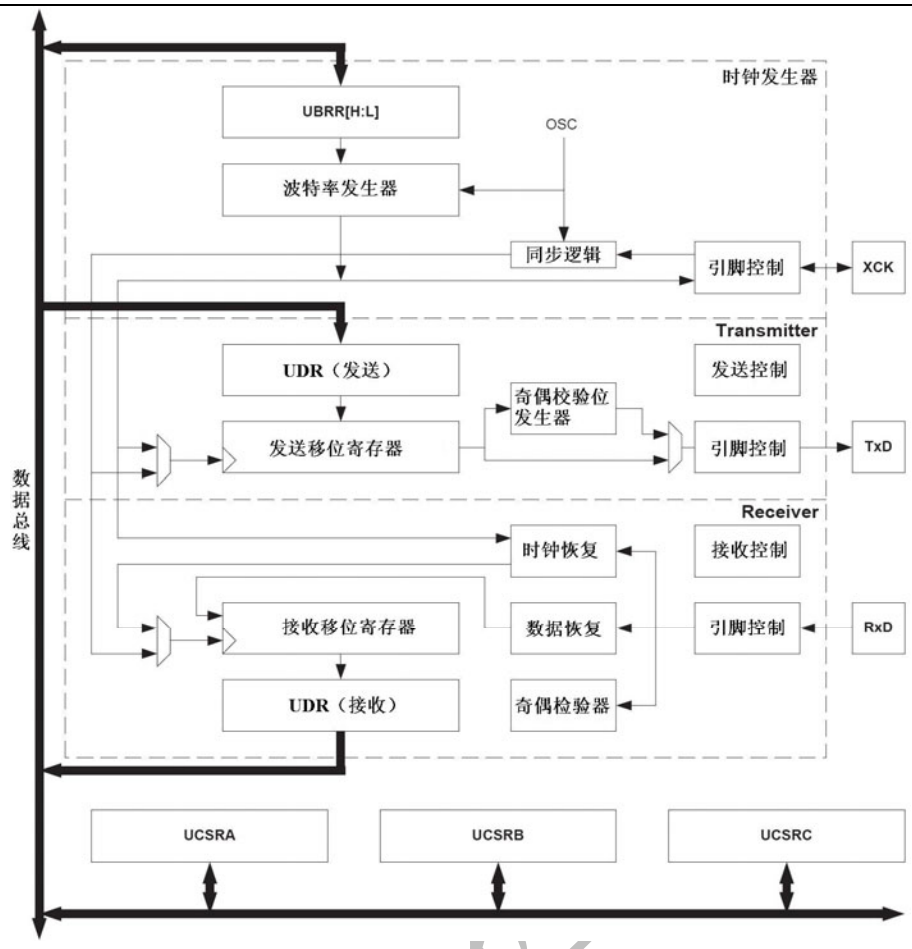

图 2.15　ATmega128（L）的全双工通用同步/异步串行接口 USART 的结构

- USART 接口可以分为 3 大部分：时钟发生器、数据发送器和数据接收器，如图中虚线框所示。
- 时钟发生器由同步逻辑电路和波特率发生器组成。其中同步逻辑电路被用在同步从模式下的外部时钟输入的场合。此外发送时钟引脚 XCK 仅被用在同步发送模式下。
- 数据发送器由一个单独的写缓冲器（即发送 UDR，使连续发送数据时不会在数据帧之间引入延迟）、串行移位寄存器、奇偶校验位发生器和处理不同数据帧的控制逻辑电路构成。
- 接收器是整个 USART 接口中最复杂的部分，其最重要的构成部分是重新获得时钟和数据单元。其中，重新获得数据单元可以用来接收异步数据。除了重新获得时钟和数据单元之外，数据接收器还包括奇偶检验器、相应的控制逻辑、接收移位寄存器和两级的接收缓冲器（即接收 UDR）。注意，数据接收器不仅支持与数据发送器相同的数据帧，而且还具有帧错误检测、数据过速检测以及检验错误检测的功能。

对 ATmega128（L）的 USART 接口来说，时钟发生逻辑为其数据发送器和数据接收器提供基本的时钟，并且 USART 接口支持如下 4 种时钟工作模式：普通异步模式、倍速异步模式、主机同步模式和从机同步模式。

USART 接口中的波特率寄存器 UBRR 和预分频向下计数器相连接，一起构成可编程的预分频器或波特率发生器。

预分频向下计数器对系统时钟计数，当其计数到零或寄存器 UBRRL 被写时，会自动装入寄存器 UBRR 的值。此外，当预分频向下计数器计数到零时会产生一个时钟，该时钟可以作为波特率发生器的输出时钟，且输出时钟的频率为 *fosc*/(UBRR + 1)。当 USART 接口工作在不同模式时，数据发送器再对该输出时钟进行 2、8 或 16 的分频。

表 2.7 给出了计算波特率 BAUD（位/秒，bps）以及计算每一种使用内部时钟源的工作模式的寄存器 UBRR 值（寄存器 UBRRH 和 UBRRL 的值，0～4095）的公式，其中 *fosc* 为系统时钟频率。

表 2.7　　波特率和 UBRR 值的计算公式

时钟工作模式	波特率计算公式	寄存器 UBRR 值计算公式

普通异步模式	$UBRR=\frac{fosc}{2BAUD}-1$	$UBRR=\frac{fosc}{16BAUD}-1$
倍速异步模式	$BAUD=\frac{fosc}{8(UBRR+1)}$	$UBRR=\frac{fosc}{8BAUD}-1$
主机同步模式	$BAUD=\frac{fosc}{2(UBRR+1)}$	$UBRR=\frac{fosc}{2BAUD}-1$

通过置位寄存器 UCSRA 中的 U2X 位可以使数据传输速率加倍，但该位只对时钟异步工作模式有效。当时钟工作在同步模式时，设置该位为 0。

ATmega128（L）的 USART 接口的一个数据帧由数据字（最多可以有 9 位）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，支持由以下位的 30 种有效组合的数据帧，如图 2.16 所示。

- 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。

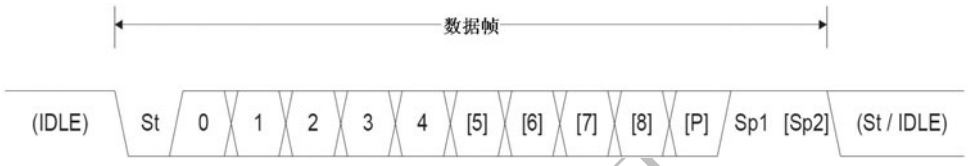

图 2.16　USART 接口的有效数据帧结构

在图 2.16 中，一个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 个数据位并以数据的最高位结束。如果使用校验位，校验位将紧接着数据位，最后是 1 或 2 个停止位。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，或使传输线路 RxD 或 TxD 处于空闲状态。

图 2.16 中符号的含义为（“[ ]”内的内容表示此项是可选的）：

- 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：传输线路 RxD 或 TxD 没有数据传输，注意线路空闲时必须保持高电平。

USART 接口的数据帧的结构由寄存器 UCSRB 和 UCSRC 中的 UCSZ2 位～UCSZ0 位、 UPM1 位～UPM0 位及 USBS 位设置，并且接收与发送的数据帧使用相同的设置。注意这些位的任何改变都可能破坏正在进行的数据传送与接收。

- UCSZ2 位～UCSZ0 位用于设置 USART 接口的数据字中的数据位数；
- UPM1 位～UPM0 位用于使能检验和选择校验的类型（奇检验或偶检验）；
- USBS 位用于选择数据帧中有一位或两位停止位。这里值得注意的是数据接收器忽略第二个停止位，因此帧错误只可能在第一个停止位为 0 时被检测到。

USART 接口的奇偶检验位的值是通过把数据字的各个位进行异或运算，再把结果同 0 或 1 进行异或运算得到的，即：

$$Peven=d_{n-1}\oplus\cdots\oplus d_3\oplus d_2\oplus d_1\oplus d_0\oplus 0$$
$$Podd=d_{n-1}\oplus\cdots\oplus d_3\oplus d_2\oplus d_1\oplus d_0\oplus 0$$

在上面两式中：

- Peven：偶校验位值；
- Podd：奇校验位值；
- d_{n-1}：数据字的第 $n-1$ 个数据位（$n=5\sim 9$）。

2.6.3　两线串行 TWI 总线接口

ATmega128（L）提供了实现标准两线串行总线通信接口 TWI，也即 I^2C 总线接口，如图 2.17 所示，其主要特点有以下几点。

- 只需两根线的强大而灵活的通信接口；
- 支持主机和从机操作；
- 器件可以工作于发送器模式或接收器模式；
- 7 位地址空间允许有 128 个从机；
- 支持多主机模式；
- 高达 400kHz 的数据传输率；
- 斜率受控的输出驱动器；
- 可以抑制总线尖峰的噪声抑制器；
- 完全可编程的从机地址以及公共地址；
- 睡眠时地址匹配可以唤醒 AVR。

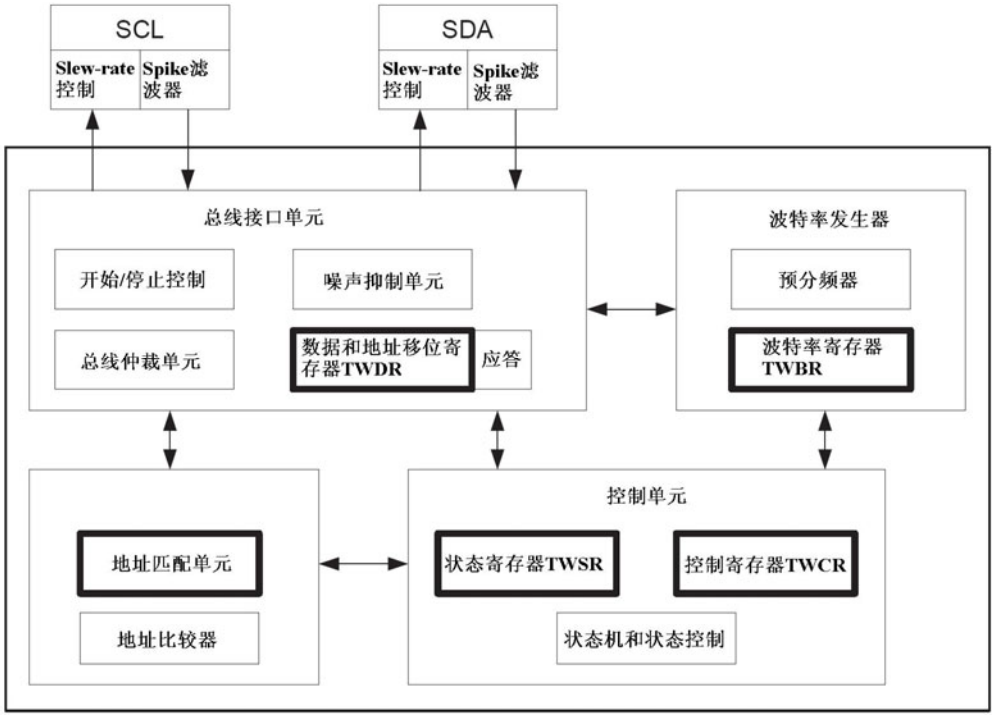

图 2.17　I^2C 总线接口结构

从图 2.17 可知，ATmega128（L）的 TWI 总线接口的主要功能单元有：引脚 SCL 和 SDA、波特率发生器、总线接口单元、地址匹配单元和控制单元。

（1）引脚 SCL 与 SDA 为 ATmega128（L）的 TWI 接口引脚。引脚的输出驱动器包含一个波形斜率控制器以满足 TWI 规范；引脚的输入部分包括噪声抑制单元以去除小于 50ns 的毛刺。当相应的端口设置为引脚 SCL 与 SDA 时，可以使能 I/O 口内部的上拉电阻，这样就可以省掉外部的上拉电阻。

（2）TWI 工作于主机模式时，比特率发生器控制时钟信号 SCL 的周期，即：

$$\text{SCL频率} = \frac{fosc}{16 + 2 \times \text{TWBR} \times 4^{\text{TWPS}}}$$

其中：*fosc* 为系统时钟频率；TWBR 为 TWI 比特率寄存器的值；TWPS 为 TWI 状态寄存器预分频的值。

（3）总线接口单元包括数据与地址移位寄存器 TWDR，控制/停止控制器和总线仲裁单元及噪声抑制单元：

- 寄存器 TWDR 用于存放发送或接收的数据或地址。除了 8 位的 TWDR，总线接口单元还有一个寄存器，包含了用于发送或接收应答的(N)ACK。注意这个(N)ACK 寄存器不能由程序直接访问。当接收数据时，它可以通过 TWI 控制寄存器 TWCR 来置位或清 0；在发送数据时，(N)ACK 值由 TWCR 的设置决定。
- 控制/停止控制器负责产生和检测 TWI 总线上的 START、REPEATED START 与 STOP 状态。即使在 ATmega128（L）处于休眠状态下，控制/停止控制器仍然能够检测 TWI 总线上的 START/STOP 条件，当检测到自己被 TWI 总线上的主机寻址时，控制器将把 ATmega128（L）从休眠状态唤醒。

- 如果 TWI 以主机模式启动了数据传输，总线仲裁单元将持续监听总线，以判断是否可以通过仲裁获得总线控制权。如果总线仲裁单元检测到自己在总线仲裁中丢失了总线控制权，则通知 TWI 控制单元执行正确的动作，并产生合适的状态码。

（4）地址匹配单元检测从总线上接收到的地址是否与寄存器 TWAR 中的 7 位地址相匹配。当寄存器 TWAR 中的 TWGCE 为 1 时，从总线接收到的地址也会与广播地址进行比较。一旦地址匹配成功，控制单元将得到通知以进行正确地响应。即使 ATmega128（L）处于休眠中，地址匹配单元仍可继续工作。一旦某个主机寻址到这个单元，就可以将 ATmega128（L）从休眠唤醒。

在 TWI 由于地址匹配将 ATmega128（L）从掉电状态唤醒期间，如有其他中断发生，TWI 将放弃操作并返回到空闲状态。如果这会引起其他问题，那么在进入掉电休眠模式之前需要确保只有 TWI 地址匹配中断被使能。

（5）控制单元监听 TWI 总线，并根据 TWI 控制寄存器 TWCR 的设置作出相应的响应。当 TWI 总线上产生需要应用程序干预处理的事件时，TWI 中断标志位 TWINT 置位，紧接着在下一个时钟周期，TWI 状态寄存器 TWSR 被表示这个事件的状态码字所更新。在其他时间里，TWSR 的内容为一个表示无事件发生的特殊状态字。一旦 TWINT 标志位置位，时钟线 SCL 即被拉低，TWI 总线上的数据传输将被暂停，用户程序开始处理事件。

由于 ATmega128（L）的 TWI 总线接口是与 I^2C 总线接口兼容的两线串行总线，因此关于 TWI 协议，如数据帧格式和数据传输模式等，读者可以参考 I^2C 总线的描述。

ATmega128（L）的 TWI 接口是面向字节和基于中断的。所有的总线事件，如接收到一个字节或发送了一个 START 信号等，都会产生一个 TWI 中断。

由于 TWI 接口是基于中断的，因此 TWI 接口在字节发送和接收过程中，不需要应用程序的干预。寄存器 TWCR 中的 TWIE 位和 SREG 寄存器的全局中断使能位一起决定了应用程序是否响应 TWINT 标志位产生的中断请求。如果 TWIE 位被清 0，应用程序只能采用查询 TWINT 位的方法来检测 TWI 总线状态。

当 TWINT 位置位时，表示 TWI 接口完成了当前的操作，在等待应用程序的响应。此时寄存器 TWSR 包含了表明当前 TWI 总线状态的值。应用程序可以读取 TWCR 的状态码，判别此时 TWI 总线的状态是否正确，并通过设置寄存器 TWCR 和 TWDR 以决定在下一个 TWI 总线周期 TWI 接口应该如何工作。

2.7 本章小结

在本章中，我们介绍了 ATmega128（L）单片机的知识，重点介绍了 ATmega128（L）的内核、时钟、电源管理、内部资源和外部总线接口（SPI、USART 和 TWI 总线接口），并对一些内容做了适当的取舍。读者在阅读过程中，理解上如果遇到困难，可以阅读 ATMEL 公司提供的 ATmega128（L）的器件手册（中文版或英文版）。

联系方式

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218

广州地址：广州市天河区中山大道 268 号天河广场 3 层，电话：020-28916067