68HC(9)08JL3 单片机

1. 概述

MC68HC(9)08JL3 是 MC68HC08 家族中高性能、低价位的一员。基于用户定义的集成电路(CSIC)的设计思想，68HC08 单片机家族使用增强型 68HC08 CPU 配以各种 I/O 模块和不同大小及类型的存储器，组成不同的单片机系列。每种单片机都有若干种封装形式。

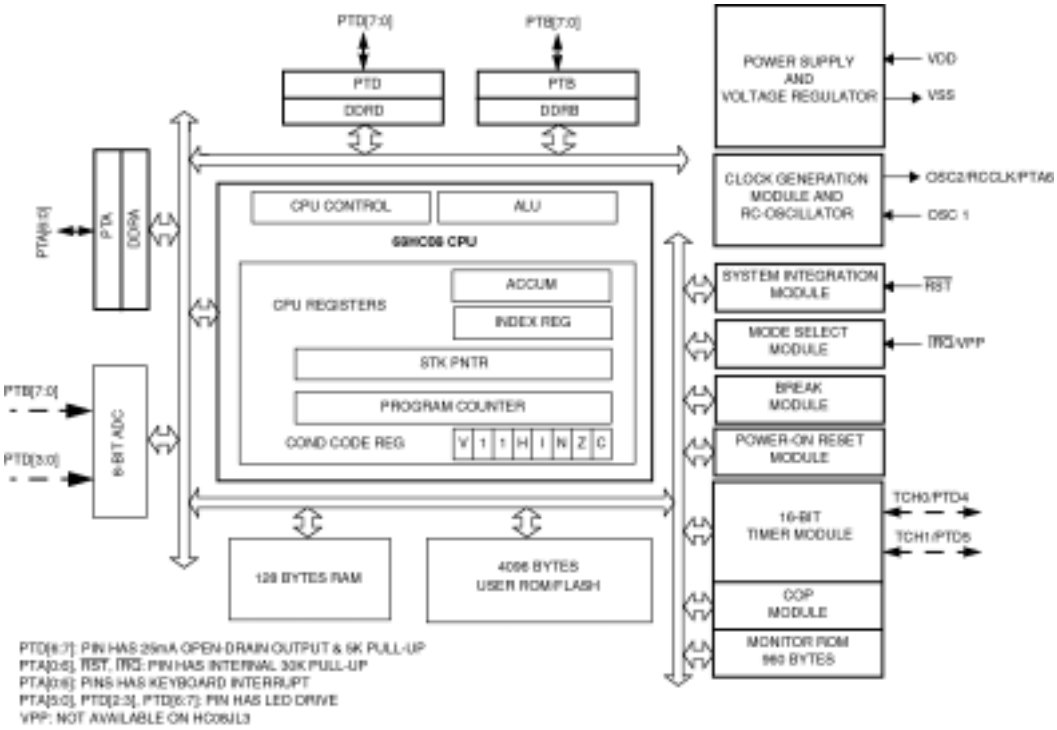

图 1-1　68HC(9)08JL3 单片机内部框图

2. 性能

- VDD＝5V±10%或 3V±10%
- 与 68HC05 目标码向上兼容
- 内部总线速度 8MHz
- 4096 字节闪速存储器(Flash)或 ROM
- 内带 960 字节监控与自检程序
- 128 字节 RAM
- 12 路 8 位 A/D
- 7 个键盘中断位
- 可编程低电压复位

- 可选用 RC 振荡器或石英振荡器
- 有 28、20、16 引脚三种封装形式
- 28 引脚的有 23 位 I/O
  - 其中 12 路 A/D 不做 A/D 也可定义成普通 I/O
  - 10 个 LED 驱动输出
  - 2 路有 25mA 漏级开路式可编程上拉电阻输出
  - 2 路输入捕捉或输出比较或 PWM
  - 7 个键盘中断位
- 20 引脚封装有 15 位 I/O
  - 其中 10 路 A/D 不做 A/D 也可定义成普通 I/O
  - 4 个 LED 驱动输出
  - 2 路有 25mA 漏级开路式可编程上拉电阻输出
  - 2 路输入捕捉或输出比较或 PWM
  - 1 个键盘中断位(当选用 RC 振荡器)
- 16 引脚封装片有 11 个 I/O 端口
  - 其中 8 路 A/D 不做 A/D 也可定义成普通 I/O
  - 1 路输入捕捉或输出比较或 PWM
  - 1 个键盘中断位（当选用 RC 振荡器)
- 有 FLASH/ROM 加密位
- 在片编程（ICP，In-Circuit-Programming）
- 有非法指令复位或中断，非法寻址中断或复位
- 内带看门狗(COP)
- 全静态设计，有 WAIT、STOP 模式

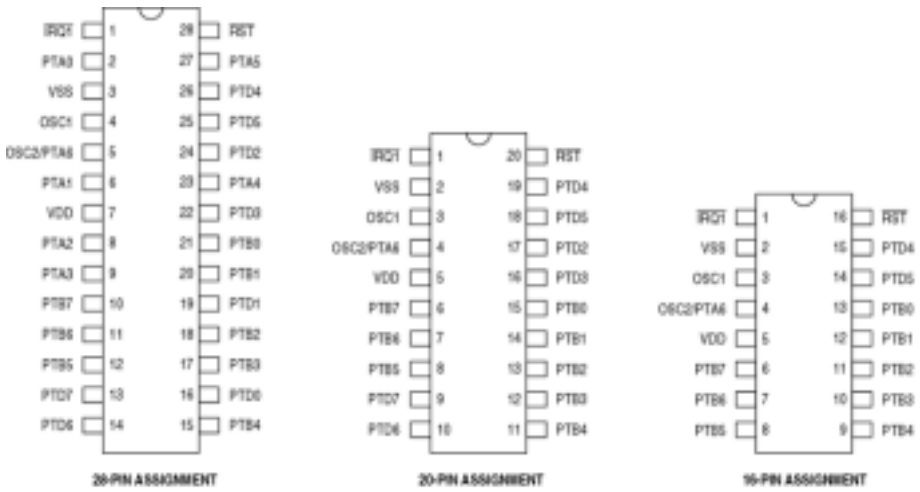

图 2-1　68HC(9)08JL3 单片机外部引脚

68HC08CPU 与 68HC05CPU 机器码兼容，寻址方式由 05 的 8 种扩大到 16 种，指令更丰富。总线速度由 05 的 2M 提高到 8M，故总体性能是 05CPU 的 5 倍。08CPU 较 05CPU 最大的改进在于堆栈指针 SP 不再是固定的，而是可以在 64K 寻址空间内滚动。因而可以使用 C 语言编译器。

表 2-1　内存分配

$0000~$003F	I/O 寄存器，实际可读/写 I/O 控制寄存器 27 个
$0080~$00FF	128 字节 RAM
$EC00~$FBFF	ROM/FLASH(4096 字节)
$FC00~FDFF	监控 ROM(512 字节)
$FE00~$FE0F	辅助 I/O 寄存器，包括断点控制，FLASH 编程控制等
$FE10~FFCF	监控 ROM(448 字节)
$FFD0~$FFFF	24 个中断向量(48 字节)
辅助 I/O 寄存器内容如下	
$FE00	断点状态寄存器 BSR
$FE01	复位状态寄存器 RSR
$FE03	断点标志控制寄存器 BFCR
$FE04	中断状态寄存器 1
$FE05	中断状态寄存器 2
$FE06	中断状态寄存器 3
$FE08	FLASH 控制寄存器 FLCR
$FE09	FLASH 块保护寄存器 FLBPR
$FE0A	FLASH 测式控制寄存器 FLTCR
$FE0C、$FE0D	断点地址寄存器
$FE0E	断点状态和控制寄存器
$FFFF	写该寄存器完成 COP 控制动作

3. 闪速存储器（FLASH Memory）

MC68HC08JL3 集成有片上的闪速存储器，并且有在片的电荷泵来产生编程和擦除电压，所以只需要单一的外部电源就可以实现读出、写入和擦除的全部操作。

JL3 片上的闪速存储器包括一个 4096 字节的块作为用户存储器，外加 32 字节的小块供用户中断向量使用。这两块的地址范围分别为$EC00－$FBFF 和$FFE0－$FFFF。它们可以象正常的 RAM 和 ROM 一样读出，而写入和擦除操作是通过**闪速控制寄存器(FLCR)**中的控制位来完成的。

		位 7	6	5	4	3	2	1	0	
FLCR	读	0	0	0	0	HVEN	MASS	ERASE	PGM	$FE08
	写									
复位		0	0	0	0	0	0	0	0	

HVEN－高压允许位

这个可读写位用来在擦除和写入操作中将来自片内电荷泵的高压加到闪速存储器阵列上。只有当 PGM 或 ERASE 为 1 且后接擦除或写入/校验序列时，才能置位此位。

1 = 内部电荷泵打开，允许高压加到存储器阵列上；

0 = 内部电荷泵关闭，禁止高压加到存储器阵列上。

MASS－整体擦除控制位

当 ERASE 置位时，这个可读写位将闪速存储器配置成整体或块擦除方式。

1 = 选择整体擦除方式

0 = 选择块擦除方式

ERASE－擦除控制位

这个可读写位将闪速存储器配置成擦除方式。这一位与 PGM 位不应同时置位。

1 = 选择擦除操作方式

0 = 不选择擦除操作方式

PGM－编程写入控制位

这个可读写位将闪速存储器配置成写入方式。这一位与 ERASE 位不应同时置位。

1 = 选择写入操作方式

0 = 不选择写入操作方式

在对闪速存储器进行擦除时，有两种方式可供选择：一是对 64 字节大小的块进行块擦除 (Block Erase)；二是对整个闪速存储器区进行整体擦除 (Mass Erase)。块擦除可以对 4K 范围内任意 64 字节边界(即$XX00，$XX40，$XX80，$XXC0)开始的块单独进行擦除。而出于安全性考虑，另外 32 字节的用户中断向量块只能进行整体擦除。

块擦除的步骤如下：

(1) 在闪速控制寄存器中置位 ERASE，清除 MASS 位；

(2) 向要擦除的 64 字节块范围内的任意地址写入任意数据；

(3) 等待一段时间 t_{nvs}（10us）；

(4) 置位 HVEN 位；

(5) 等待一段时间 t_{ERASE}（1ms）；

(6) 清除 ERASE 位；

(7) 等待一段时间 t_{nvh}（5us）；

(8) 清除 HVEN 位；

(9) 等待 t_{rcv}（1us 之后），可以再次对存储器进行读访问。

整体擦除的步骤如下：

(1) 在闪速控制寄存器中置位 ERASE 和 MASS 位；

(2) 向闪速存储器范围内的任意地址写入任意数据；

(3) 等待一段时间 t_{nvs}（10us）；

(4) 置位 HVEN 位；

(5) 等待一段时间 t_{ERASE}（4ms）；

(6) 清除 ERASE 位；

(7) 等待一段时间 t_{nvh}（100us）；

(8) 清除 HVEN 位；

(9) 等待 t_{rcv}（1us 之后），可以再次对存储器进行读访问。

对闪速存储器的编程写入是以行为基础进行的，每一行包括从 32 字节地址边界起始的 32 个字节。对某一行的编程写入步骤如下：

(1) 在闪速控制寄存器中置位 PGM 位；

(2) 向要编程的行范围内的任意地址写入任意数据；

(3) 等待一段时间 t_{nvs}（10us）；

(4) 置位 HVEN 位；

(5) 等待一段时间 t_{pgs}（5us）；

(6) 向要编程的字节地址写入要编程的数据；

(7) 等待一段时间 t_{PROG}（30us）；

(8) 重复步骤 6 和 7 直到这一行中所有字节都编程完毕；
(9) 清除 PGM 位；
(10) 等待一段时间 t_{nvh}（5us）；
(11) 清除 HVEN 位；
(12) 等待 t_{rcv}（1us 之后），可以再次对存储器进行读访问。

可以重复类似上面的操作序列从而对整个闪速存储器进行编程。需要注意的是，由于在编程和擦除时不能对闪速存储器区进行读出，所以擦除和编程操作的程序代码也不能位于这个存储区内。在各个步骤之间的空闲时间内，可以进行任何与此无关的其他操作。

虽然片内的电荷泵给我们的擦除和编程带来方便，但同时也带来了由于系统工作不正常而造成对闪速存储器误写入或误擦除的可能性。为此，专门设置了一个**闪速保护寄存器(FLSPR)**，其内容决定了被保护的存储器区域。当某区域受到保护时，在对该区域的擦除和写入操作中将不允许置位 HVEN 控制位，从而防止这种意外的擦除和写入。

		位 7	6	5	4	3	2	1	位 0	
FLSPR	读	BPR7	BPR6	BPR5	BPR4	BPR3	BPR2	BPR1	BPR0	$FE09
	写									
复位		0	0	0	0	0	0	0	0	

BPR[7:1]指定了被保护存储器的起始地址的第 12 位－第 6 位。BPR[0]未使用。
$00－$60 = 整个闪速存储器区域都被保护
$62 = 被保护范围：$EC40－$FFFF
$64 = 被保护范围：$EC80－$FFFF
$68 = 被保护范围：$ECC0－$FFFF
$DE = 被保护范围：$FBC0－$FFFF
$FE = 被保护范围：$FFC0－$FFFF
$FF = 整个闪速存储器区域都不被保护，可以被写入和擦除

4. 随机存取存储器（RAM）

从$0080 到$00FF 的地址范围是 RAM 区，它属于整个地址空间的第 0 页。通过快速而有效的直接寻址可以访问到所有第 0 页的数据，因此提供了一个存储那些访问频率较高的全局变量的理想场所。同时，上电复位后，栈指针初始化为$00FF，所以栈区也处于这个区域内。每次进入中断服务例程时，栈区将扩大 5 个字节；而调用子例程时，将占用两个字节来存储返回地址。在进行嵌套调用时，尤其要注意由于栈区的膨胀而覆盖有用数据的可能性。

MC68HC08JL3 具有 16 位的栈指针，因此允许栈区处于 64K 地址空间的任意位置。必须注意的是，为保证操作正确，栈指针只应该指向 RAM 区域。

5. 配置寄存器（CONFIG）

配置寄存器使用户在对 MCU 初始化的过程中，具有多种选择。它可以允许或禁止以下选项：

- STOP 方式恢复时间（32 2OSCOUT 周期或 4096 2OSCOUT 周期）
- STOP 指令
- 计算机工作正常（COP）模块
- COP 复位周期 COPRS，$(2^{13}-2^4)*2OSCOUT$ 或$(2^{18}-2^4)*2OSCOUT$

- 允许 LVI 电路
- 选择 LVI 电压

配置寄存器在每次复位后只能写入一次。考虑到以上选项都对 MCU 的操作有影响，所以最好在复位后立即对配置寄存器进行写入。

在 ROM 版本的 MCU 中，CONFIG 模块实际上是 MOR(掩模选项寄存器)。以下对各控制位的说明也适用于 EPROM/OTPROM 版本的 MCU。

		位 7	6	5	4	3	2	1	位 0	
CONFIG2	读 写	IRQPUD	保留	保留	LVIT1	LVIT0	保留	保留	保留	$001E
复位：		0	0	0	Not affected		0	0	0	
POR：		0	0	0	0	0	0	0	0	

IRQPUD－IRQB1 上拉控制位

1 = 断开内部上拉；

0 = 在 IRQB1 和 V_{DD} 之间连接内部上拉。

关于 LVI 控制位的详细信息请参看第 12 部分。

		位 7	6	5	4	3	2	1	位 0	
CONFIG1	读	COPRS	保留	保留	LVID	保留	SSREC	STOP	COPD	$001F
	写									
复位：		0	0	0	0	0	0	0	0	

COPRS－COP 复位周期选择位

1 = COP 复位周期为$(2^{13}-2^4)*2$OSCOUT；

0 = COP 复位周期为$(2^{18}-2^4)*2$OSCOUT。

SSREC－STOP 方式快速恢复位

1 = 从 STOP 方式恢复时间为 32 2OSCOUT 周期；

0 = 从 STOP 方式恢复时间为 4096 2OSCOUT 周期。

STOP－允许 STOP 指令

1 = 允许 STOP 指令；

0 = 视 STOP 为非法指令。

COPD－COP 禁止位

1 = 禁止 COP 模块；

0 = 允许 COP 模块。

LVID－LVI 禁止位

1 = 禁止 LVI 模块；

0 = 允许 LVI 模块。

6. 68HC08 CPU

6.1 概述

本节描述 68HC08 CPU，该 CPU 与 68HC05CPU 目标码全部向上兼容，详见 CPU08 参考手册(“The CPU08 Reference Manual”Motorola 技术文件号 CPU08 RM/AD)。

6.2 CPU08 的主要特征

- 与 MC68HC05 系列目标码完全向上兼容
- 64K 字节程序/数据存储器空间
- 8 MHz CPU 内部总线频率
- 16 种寻址方式，相对于 HC05 增加了 5 种 *(详见 68HC08 参考手册)*
- 可扩展的内部总线定义，用于寻址超过 64K 字节的地址空间
- 用于指令操作的 16 位索引寄存器
- 16 位堆栈指针和相应栈操作指令
- 不使用累加器的存储器之间的数据移动
- 快速 8 位乘法和 16 位除法指令
- BCD 码指令进一步增强
- 内部总线灵活多变可适应的 CPU 增强外设如 DMA 控制器
- 完全的静态低电压/低功耗设计

6.3 68HC08 CPU 内部寄存器

68HC08CPU 内部有：8 位累加器，16 位间址寄存器，可寻址整个 64K 空间，H 是高位，X 是低位。堆栈指针也是 16 位寄存器，复位时栈指针指向$00FF，RSP 指令复位栈指针低位到 FF，而高位不受影响。PUSH 指令使栈指针减 1，而 PULL 指令使栈指针加 1。有带有 8 位和 16 位偏移量的栈指针间址的寻址方式。

程序计数器是 16 位的，一般指向下一条指令。复位时，程序计数器的值是$FFFE 和$FFFF。

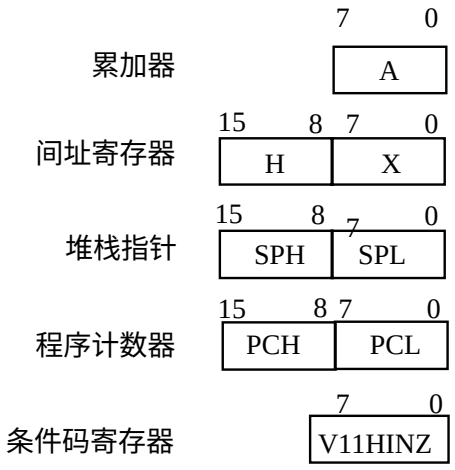

图 6-1　68HC08 CPU 内部寄存器

条件码寄存器 CCR

✧ V - bit7 是溢出标志

当二进制补码溢出时置位。有符号跳转指令 BGT、BGE、BLE 和 BLT 使用该标志。

✧ Bit6 和 bit5 永远为 1

✧ H - bit4 半进位标志

执行 ADD 和 ADC 指令时，当累加器 bit3 向 bit4 有进位时该位置位。用于 BCD 运算。做 DAA 十进制调整时，利用 H 和 C 的状态来判断是否调整。

✧ I - bit3 中断屏蔽位

1＝中断禁止　　0＝中断允许

复位时，该位置 1，可用 CLI 指令开中断。中断响应时，CPU 将除 H 以外的寄存器推入堆栈，然后执行中断服务子程序，遇到 RTI 指令时，从栈中恢复包括 CCR 在内的各寄存器。当然也包括这一位的状态。注意在中断服务子程序中如用到 H 寄存器的话，

不要忘了 PUSH 和 PULL 指令。

- ✧ N - bit2 为负标志，运算结果为负时置位。
- ✧ Z - bit1 为零标志，数据或运算结果为 0 时置位。
- ✧ C - bit0 进位/借位标志，加法在 bit7 上有进位，减法在 bit7 上有借位，该位置位。一些指令如位测试、跳转、移位指令等影响该标志。

7. 振荡器

本节主要介绍 MC68HC(9)08JL3 的振荡器部分的结构，外引脚以及所产生的信号。

这种微控制器有两种可选的振荡器类型：RC 振荡器和石英晶体振荡器，具体采用哪一种，是在掩模时确定的。两种类型振荡器的片内和片外电路均不相同。

石英晶体振荡器采用石英共振或陶瓷共振来提供精确的时钟频率。在实际工作时，这种振荡器有两个外部引脚，即振荡放大器输入端 OSC1 和振荡放大器输出端 OSC2。需要在这两个引脚间加入晶振、反馈电阻和电容，以实现稳定的工作频率。

RC 振荡器依靠外部一对电阻和电容与片内电路来提供精度为 10%的时钟频率，要求用在这里的电阻和电容自己的精度为 1%。按这种方式连接的振荡器精度和稳定度都要比石英晶体振荡器为低，但它只用到了一个外部引脚，即 OSC1。原 OSC2 引脚可以设定为通用 I/O 引脚 PT6 或者 RC 振荡器输出时钟 RCCLK。

这两种振荡器外部接法如下图所示：

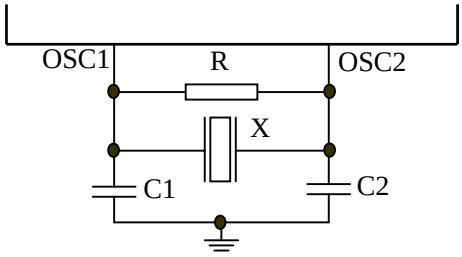

图 7-1 石英晶体振荡器

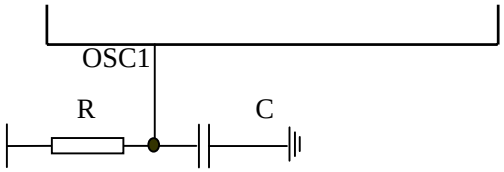

图 7-2 RC 振荡器

振荡器电路的输出 2OSCOUT 为 SIM 单元提供时钟，并决定 COP 的时钟周期。对石英晶体振荡器，2OSCOUT 与 XTALCLK 相连，对 RC 振荡器，2OSCOUT 与 RCCLK 相连。将 2OSCOUT 二分频后，得到 OSCOUT，用于产生 CPU 和其他模块的总线时钟。

当 CPU 进入低功耗 STOP 模式时，XTALCLK 和 RCCLK 以及 OSCCLK 都被禁止，而在 WAIT 或 BREAK 模式时，时钟不受影响。

8. 系统集成模块 (SIM)

8.1 简介

本节介绍系统集成模块 SIM (System Integration Module)。SIM 与 CPU 一同控制了 MCU 的所有操作，正是它才使 CPU 能够进行异常顺序的处理。SIM 支持总计达 24 个的各种内部或外部中断，主要功能如下：

- CPU 及外设的总线时钟产生和控制
  - ✧ Stop/Wait/Reset/Break 各种模式的进入和恢复
  - ✧ 内部时钟控制
- 主复位控制，包括上电复位(POR)和 COP 超时
- 中断控制
  - ✧ 应答时序
  - ✧ 仲裁控制时序
  - ✧ 向量地址生成
- CPU 允许/禁止时序
- 可扩展到 128 个中断源

8.2 SIM 总线时钟的产生和控制

总线时钟由 OSCOUT 二分频得到，也就是振荡器时钟(2OSCOUT)的四分频。

当上电复位模块产生复位信号时，在 4096 个 2OSCOUT 时钟周期(POR 等待时间)内，提供给 CPU 和外设的总线时钟保持无效。在这个等待时间结束之后，IBUS(Internal Bus)时钟才能启动，以确保系统的正常复位。

当复位、break 或中断使系统脱离 Stop 模式时，SIM 要等待 4096 或 32 个(可选) 2OSCOUT 周期，才恢复 CPU 和外设的时钟。在 Wait 模式下情况稍微复杂一些，CPU 时钟会被禁止掉，但其他模块的时钟是否禁止视具体情况而定，某些模块甚至可以编程来设定。

8.3 复位和系统初始化

MCU 有五种不同的复位产生方式：

- 上电复位 (POR)
- 外部复位引脚 ($\overline{\text{RST}}$)
- COP
- 非法指令
- 非法地址

所有这些复位方式引导向量$FFFE~FFFF，并产生内部复位信号 IRST。IRST 使所有寄存器恢复到默认值，并使所有模块恢复到初始状态。内部复位会清除 SIM 计数器，而外部复位不会。每个复位都会在复位状态寄存器 RSR 中置起相应位。

$\overline{\text{RST}}$ 引脚内部有上拉电阻。当该引脚被拉低 67 个以上的 2OSCCLK 时钟时，RSR 寄存器的 PIN 位会置起，表示这时的中断源不是上电复位。

内部中断源除了上电复位以外，都会使外部的 $\overline{\text{RST}}$ 引脚拉低 32 个 2OSCOUT 时钟，以使外设合适的复位。

上电复位时，系统在等待 4096 个 2OSCCLK 时钟的振荡器稳定时间后才将时钟加给

CPU 和其他模块，在振荡器未稳定前，$\overline{\text{RST}}$ 保持低电平。同时置起 RSR 寄存器的 POR 位，并清除其他位。上电复位时还产生 POR 脉冲和内部复位信号。

COP 是另一种内部复位源。一个到 SIM 的输入被保留为 COP 复位信号。COP 寄存器的溢出会导致内部复位并置起 RSR 寄存器的 COP 位，以及把 $\overline{\text{RST}}$ 引脚拉低。若要防止 COP 复位，可在复位后立即往$FFFF 写一个任意值，以清除 COP 计数器和 SIM 计数器的 12 到 5 级。用来驱动 COP 计数器的 SIM 计数器的输出周期为 16～4096 个 2OSCOUT 时钟。

SIM 检查来自 CPU 的信号以监测非法指令，发现非法指令后，RSR 寄存器的 ILOP 位会被置起并发生复位。如果屏蔽设置寄存器的 stop 使能位 STOP 是逻辑零，SIM 会认为 STOP 指令是非法指令并复位系统。

当 CPU 试图从不合法的地址去读取指令时，SIM 会产生非法地址复位，并将 RSR 寄存器的 ILAD 位置起。但从非法的地址读数据并不会引起这种复位。

当电源电压低于低电压禁止模块(LVI)的阈值时，LVI 就会通知 SIM 产生低电压复位。发生低电压复位时，RSR 寄存器的 LVI 被置起，其他操作与上电复位类似：SIM 计数器的 4096 个 2OSCCLK 周期内，外部复位引脚被拉低；64 个 2OSCOUT 周期后，CPU 和存储器被释放以获取复位向量。

8.4 SIM 计数器

SIM 计数器的作用是在上电复位时或从 STOP 模式恢复时，在内部总线时钟建立前使振荡器有时间稳定下来。SIM 计数器的时钟源为 2OSCOUT，前 12 级计数器用于计数，第 13 级用于清除 SIM 计数器，并为 COP 提供时钟。在上电复位时，上电复位模块用 PORRST 信号通知 SIM 模块，在 SIM 计数器计数完毕，SIM 模块完成初始化后，振荡器就能够驱动总线时钟了。

外部复位对 SIM 计数器无效。在所有复位后，SIM 计数器都处于自由运行状态。

8.5 异常控制

正常运行的程序可被以下三种异常情况所改变：

- 中断
  - ✧ 可屏蔽硬件 CPU 中断
  - ✧ 不可屏蔽软件中断指令

中断可以临时改变当前程序的运行状态以相应一个特定的事件。当中断发生时，SIM 会锁存该中断请求，并在中断处理的开始进行中断仲裁。CPU 用仲裁的结果来确定应当引导哪个中断向量。一旦一个中断请求被 SIM 锁存，其他中断请求，无论优先级，在再次打开中断屏蔽前都不能获得响应。中断处理开始时，CPU 保存所有寄存器的值(为保持同 68HC05 系列的一致性，H 寄存器的值并没有入栈)，置起中断屏蔽位(I 位)，以防止其他中断打入。中断返回指令 RTI 则恢复所有寄存器的值，并使被中断的程序继续执行。

硬件中断并不马上打断当前执行的指令，它是在该指令完成后，由 SIM 所有挂起的中断请求，如果中断没有被屏蔽，并且对应的局部使能位是逻辑 1，中断就得到处理。否则执行下一条指令。软件中断是靠 SWI 软中断指令发出的，在处理上它与硬件中断唯一不同之处在于它是不可屏蔽的。

中断状态寄存器有三个，当中断发生时，寄存器的对应位就会置起。这些寄存器的地址为$FE04～06。

- 复位

所有的复位源有相同的优先级，因此无需仲裁。

- 断点中断

断点模块在可编程断点处，可产生中断请求来中断正常的程序流。发生断点中断时，SIM 使 CPU 装载 SWI 向量从而令其进入断点状态。

当用户合适设置了断点标志控制寄存器(BFCR)的断点清除标志使能(BCFE)位后，在断点模式中，就可以随意清除其他模块的标志寄存器。如果 BCFE 位没有置起，这些标志寄存器的内容就受到保护。

8.6 低功耗模式

WAIT 和 STOP 指令都使 MCU 进入低功耗模式－在此模式中，SIM 切断 CPU 的时钟源。这两条指令都清除 CCR 寄存器的中断屏蔽位。

在 WAIT 模式，CPU 时钟停止但外设时钟仍在运行，其他模块的运行视具体情况而定。中断、断点和 COP 都可将 CPU 从 WAIT 模式中激活。

在 STOP 模式，CPU 和外设的时钟都停止运行，SIM 计数器被清除，系统时钟被禁止。中断、断点和 COP 都可将 CPU 从 STOP 模式中激活。STOP 模式的恢复时间可用 CONFIG 寄存器的 SSREC 位来设置，如 SSREC＝0，恢复时间为 4096 个 2OSCOUT 时钟周期；如 SSREC＝1，恢复时间缩短到 32 个周期。CPU 开始从 STOP 模式恢复时，SIM 计数器启动计时。

8.7 SIM 寄存器

断点状态寄存器 BSR

该寄存器仅在 bit1 有效。Bit1 为 SBSW，它设置在 CPU 退出断点中断后，是否还返回 STOP 或 WAIT 模式。当 SBSW＝0 时，返回 STOP 或 WAIT 模式，否则不返回。

复位状态寄存器 RSR

该寄存器包含了标识上次复位源的六个标志位，读操作清除寄存器的内容。上电复位置起 POR 位并清除寄存器的所有其他位。逻辑 1 表示上次复位由对应复位源引起。

		位 7	6	5	4	3	2	1	位 0	
RSR	读	POR	PIN	COP	ILOP	ILAD	MODRST	LVI	0	$FE01
POR:		1	0	0	0	0	0	0	0	

MODRST 表示复位源为监视模式入口模块。

断点标志控制寄存器 BFCR

该寄存器仅在 bit7 有效。Bit7 为断点清除标志使能位 BCFE。BCFE＝1 时允许在断点模式时清除状态位。

9. 系统操作正常监视模块 COP

COP 是一个 6 位的自由运行计数器，时钟来源于 12 位 SIM 计数器的输出。在运行 $2^{18}-2^4$ 个 2OSCOUT 时钟周期后发生溢出，可以产生复位信号，防止程序进入不可预料的操作。使用 8MHz 石英晶振时，COP 溢出周期为 32.775ms。可以周期的写地址$FFFF 来清除 COP 计数器，以禁止 COP 进行复位。

COP 复位将外部复位引脚拉低 32 个 2OSCOUT 时钟，并置起 RSR 寄存器的 COP 位。

上电复位电路在 4096 个 2OSCOUT 时钟后清除 SIM 计数器。内部复位清除 SIM 计数

器和 COP 计数器。COP 控制寄存器位于$FFFF(覆盖了复位向量的地址)，写入任何内容均清除 COP 计数器，读操作则返回复位向量的低字节。

10. 监控 ROM（MON）

所谓监控 ROM（Monitor ROM）是指 MCU 出厂前固化在地址范围$FC00－$FDFF 和$FE10－$FFCF 处的固件，其中包含了有关系统检测、FLASH 编程以及串行通信等功能的代码。这就使得 MCU 多了一种不同于正常用户方式的特殊操作方式，称为监控方式。在特定条件下，MCU 可以不进入正常的用户方式，而是进入到监控方式中，这时从$FEFE、$FEFF 处取得复位向量，执行固件代码。监控 ROM 可以通过单一的一条信号线与主机进行串行通信，接收和执行预先定义的主机命令如读写存储器、执行程序等，并返回结果。适当运用监控方式和这些主机命令，能够完成一些特殊功能，例如：

- 下装代码到 RAM 或 FLASH 存储器中；
- 执行 RAM 或 FLASH 中的程序代码；
- FLASH 存储器的加密；
- FLASH 存储器擦除/写入/校验；
- 与主计算机进行标准的不归零（NRZ）传号/空号串行通信，波特率 4800－28.8k；
- 在片编程（ICP，In-Circuit-Programming）；
- 用户方式 FLASH 编程（UMFP，User Mode FLASH Programming）。

通常，监控方式可以通过以下三种途径之一来进入：

- 复位时在 IRQ 引脚上加高压 V_{HI} (V_{HI}=1.4－2 V_{DD}) 并置 PTB0－PTB3 引脚为适当值，此时进入正常的监控方式，适用于系统检测、下装和执行 RAM 程序以及用串行编程器进行 FLASH 编程写入。
- 复位时用户方式复位向量$FFFE、$FFFF 为空时，MCU 进入监控方式，如果 MCU 已经安装在 PCB 上，通过这种方式可以进行在片编程（ICP）。
- 从用户方式下执行监控 ROM 程序，可以进行用户方式 FLASH 编程（UMFP），同样能完成现场的软件升级。

11. 断点模块（BREAK）

断点模块可以在程序执行到预定地址时产生断点中断，从而暂时中断正常程序流，进入到一个后台程序中。在这个程序中可以访问存储器和 I/O 寄存器，进行一系列调试操作。

以下两种情况都可以引发断点中断：

- CPU 产生的地址（程序计数器所包含的地址）与断点地址寄存器的内容相等时；
- 通过软件向断点状态和控制寄存器的 BRKA 位写入逻辑“ 1 ”时。

当这两种情况之一发生时，断点模块就产生一个断点信号（BKPT）给 SIM，SIM 则使 CPU 在结束当前指令后，将一条 SWI 指令装入内部指令寄存器作为下一条指令执行。这样就如同发生一个软件中断，$FFFC 和$FFFD（在监控模式下为$FEFC 和$FEFD）指定了中断服务例程的起始地址；在断点服务例程中执行 RTI 指令，就结束了断点中断，使 MCU 回复到正常的程序流程。

SIM 还控制着在断点方式下能否清除其他模块中包含的状态标志。用户可以通过正确设置断点标志控制寄存器（BFCR）中的 BCFE 位，来选择这些状态标志在断点方式下是否受到保护。通常保护这些状态标志允许各种状态寄存器在断点方式下可以被随意读写，而不至于在退出断点方式时丢失状态信息。

在断点方式下，定时器计数停止，当 RST 引脚上加有 V_{HI} 时，COP 也被禁止。

在低功耗 WAIT 和 STOP 方式下，发生断点中断时，将退出低功耗方式，同时断点状态寄存器（BSR）中的 SBSW 被置位。若要想在断点返回之后继续回到原来的低功耗方式，则有必要在断点服务例程中判读 SBSW 标志，并决定是否要将栈中的返回地址减一，以使它重新指向 WAIT 或 STOP 指令。详见第 8 部分有关 SIM 寄存器的叙述。

有三个寄存器用来控制断点模块的操作，它们是：**断点状态和控制寄存器(BRKSCR)、断点地址寄存器高位(BRKH)**和**断点地址寄存器低位(BRKL)。**

		位 7	6	5	4	3	2	1	位 0	
BRKSCR	读	BRKE	BRKA	0	0	0	0	0	0	$FE0E
	写									
复位：		0	0	0	0	0	0	0	0	

BRKE－断点允许位

1 = 允许在 16 位断点地址匹配时发生断点中断；

0 = 禁止断点中断。

BRKA－断点激活位

这个读/写位在断点地址匹配时被置位；向其中写入“ 1 ”也会产生断点中断。在离开断点服务例程前，应向此位写入“ 0 ”清除此位。

1 = 发生断点地址匹配；

0 = 未发生断点地址匹配。

		位 7	6	5	4	3	2	1	位 0	
BRKH	读	位 15	14	13	12	11	10	9	位 8	$FE0C
	写									
BRKL	读	位 7	6	5	4	3	2	1	位 0	$FE0D
	写									
复位		0	0	0	0	0	0	0	0	

12. 低电压禁止（LVI）

低电压禁止模块的作用就是监测加在 V_{DD} 上的供电电压，并在 V_{DD} 低于某个预定电压值 LVI_{TRIP} 时，认为发生电源故障，产生中断信号并强制系统复位。用户可以通过 CONFIG1 & CONFIG2 寄存器中的三位来选择是否允许这种功能，并设定 LVI_{TRIP} 电压值。

		位 7	6	5	4	3	2	1	位 0	
CONFIG1	读	COPRS	保留	保留	LVID	保留	SSREC	STOP	COPD	$001F
	写									
复位：		0	0	0	0	0	0	0	0	
CONFIG2	读	IRQPUD	保留	保留	LVIT1	LVIT0	保留	保留	保留	$001E
	写									
复位：		0	0	0	Not affected		0	0	0	
POR：		0	0	0	0	0	0	0	0	

LVID－LVI 禁止位

1 = 禁止 LVI 模块；

0 = 允许 LVI 模块。

LVIT1，LVIT0－LVI 动作电压选择位，见下表：

LVIT1	LVIT0	LVI_{TRIP}
0	0	1.6V
0	1	2.4V
1	0	4.0V
1	1	保留

若 LVI 被允许，它将在低功耗 WAIT 和 STOP 方式下继续工作。

13. 并口（I/O）

MC68HC(9)08JL3 共有 3 个输入/输出并口，共 23 个引脚。其中，PORTA 有 7 个引脚，PORTB、PORTD 各有 8 个引脚。3 个并口除了可以作为普通的输入/输出端口外，均具有其它的一些特殊功能。当作为普通的输入/输出端口使用时，这些并口和一般的 05 的并口一样，有它们各自的数据寄存器和方向寄存器。

PORTA 的 7 个引脚可以作为键盘中断（KBI）使用。当作为输入口使用时，各引脚可以软件设置使用内部的上拉电阻，此外，PTA0~PTA5 具有 LED 驱动能力。
PORTA 的数据寄存器用来读写端口。读端口时，总是读取引脚上的电平，和输入/输出设置无关。

PORTA 方向寄存器(DDRA)用来控制 PORTA 的每个引脚的输入（0）/输出（1）。

键盘中断控制寄存器（KBAIER）的 KBIE6~KBIE0 位控制 PORTA 的引脚作为外部中断使用。

PORTA 输入用上拉电阻使能寄存器（PTAPUE）软件设置每个引脚具有一个内部的上拉电阻，它要求 PORTA 的方向寄存器（DDRA）的相应位设置为输入方式。

PORTB 除作为普通 I/O 外，可以作为 A/D 变换器使用。

PTB 和 DDRB 的使用同 PTA 和 DDRA 一样。

PORTD 的功能比较多。8 个引脚除作为 I/O 外，0~3 引脚具有 A/D 变换功能；2、3、6 和 7 引脚具有 LED 驱动能力；6、7 引脚具有大电流（25mA）驱动能力和 5k 的上拉电阻。此外，4、5 引脚可以作为定时器的输入/输出引脚使用。

PTD 和 DDRD 的使用同 PTA 和 DDRA。

PORTD 控制寄存器（PDC）的 bit1、bit0 控制 PTD7、PTD6 是否使用上拉电阻，这两位的设置和 DDRD 的 bit7、bit6 无关。bit3、bit2 设置 PTD6、PTD7 位具有缓慢沿变化的大电流驱动能力。

14. 定时器接口模块（TIM）

TIM 是一个双通道的定时器，具有输入捕捉、输出比较和脉宽调制（PWM）功能。TIM 具有以下特征：

- 双通道输入捕捉（有效沿可以是上升沿、下降沿或者两者均可）/输出比较（可置位、清除或者切换输出比较引脚的电平）功能；
- 带缓冲或者不带缓冲的 PWM 信号产生功能；
- TIM 时钟可以软件选择：可使用内部总线时钟产生的 7 种分频频率或者使用外部时钟源；
- 16-bit 计数器可以自由运行或者取模运行；
- 具有计数器停止和复位位；
- 模块化设计，可以扩展到 8 个通道。

图 14-1 是 TIM 的框图。TIM 的核心是一个 16bit 的计数器（TCNTH：TCNTL）。它可以自由运行，也可以取模运行，为输入捕捉和输出比较提供时间基准。TMODH：TMODL 提供了取模运行时的模数。

TIM 的时钟源可以取自 CPU 内部总线时钟的 7 种分频后的频率，或者外部时钟 TCLK。这种选择用 TIM 状态控制寄存器 TSC 的 PS[2：0]位实现。

在输入捕捉方式下，TIM 能够捕捉到外部事件发生的时间。当有效的沿跳变在输入捕捉引脚上发生时，TIM 锁存当前计数器中的值到相应的 TIM 通道寄存器（TCHxH：TCHxL）。这种沿跳变是可以用软件选择的。

在输出比较方式下，TIM 可以产生特定极性、特定宽度和特定频率的脉冲。当计数器中的值达到输出比较寄存器中的值时，TIM 清除、设置或者切换该通道引脚上的电平。

输出比较分为不带缓冲和带缓冲两种。

不带缓冲的输出比较，正如上面所说的，改变输出比较要在输出比较寄存器写入新的值。这种异步的写入会导致在两个定时器溢出周期内，输出比较功能不正常。例如，在计数器未达到旧的输出比较值之前，写入一个计数器已走过的新的比较值，会在当前的定时器溢出周期内没有任何输出比较发生。解决这个问题可以采用以下的方法：当要写入一个比当前的旧比较值小的值时，在输出比较中断服务程序中写入新值；当要写入一个更大的值时，在溢出中断服务程序中写入此值。

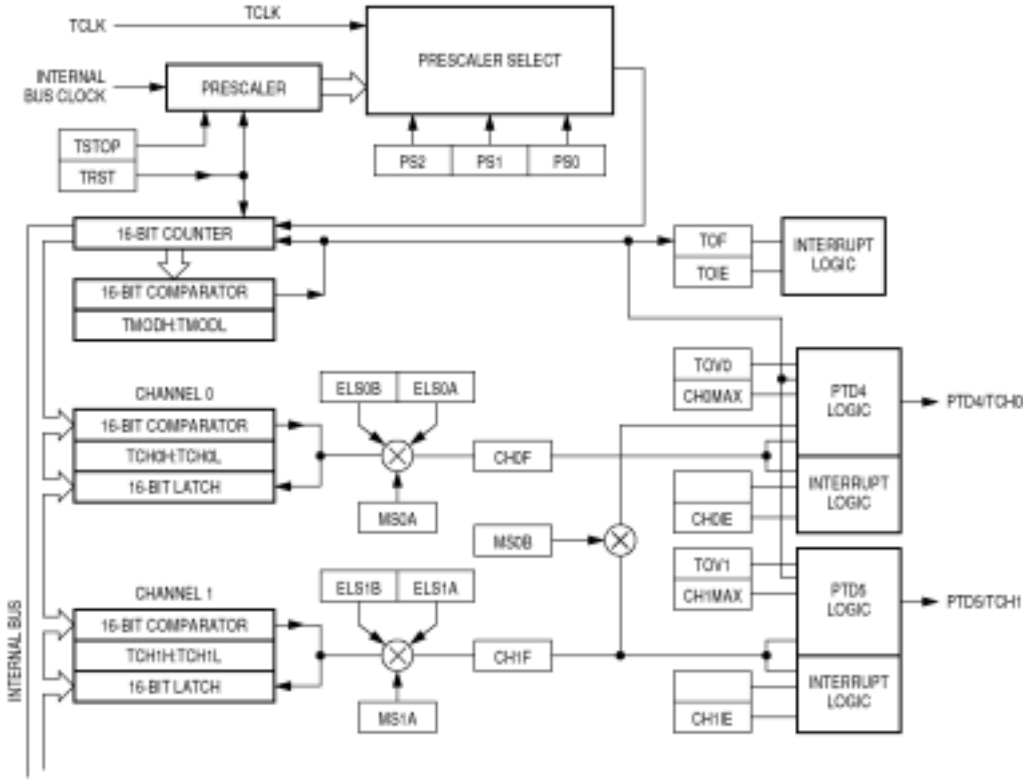

图 14-1　TIM 框图

MC68HC(9)08JL3 有带缓冲的输出比较功能。通过设置 TSC0 中的 MS0B 位，将通道 0 和 1 联系起来，在 PTD4/TCH0 引脚上输出这个输出比较信号，PTD5/TCH1 这时作为普通的 I/O 使用。两个通道的输出比较寄存器在相邻的两个溢出周期内，依次轮流控制输出比

较功能。先是通道 0 控制一个溢出周期，然后是通道 1 控制。这样就可以在其中一个通道控制时，来改写另一个通道的输出比较寄存器，这样就不会出现在不带缓冲的输出比较功能中改写输出比较寄存器时出现的问题。
如果在使用输出比较功能的同时，在计数器溢出时，切换输出引脚上的电平，就可以产生 PWM 信号。模数寄存器（TMODH：TMODL）中的值决定了 PWM 信号的周期。当计数器中的值达到模数寄存器中的值时，TIM 就切换该通道输出引脚上的电平。相应地，输出比较寄存器中的值决定了 PWM 信号的脉宽，如图 14-2 所示。

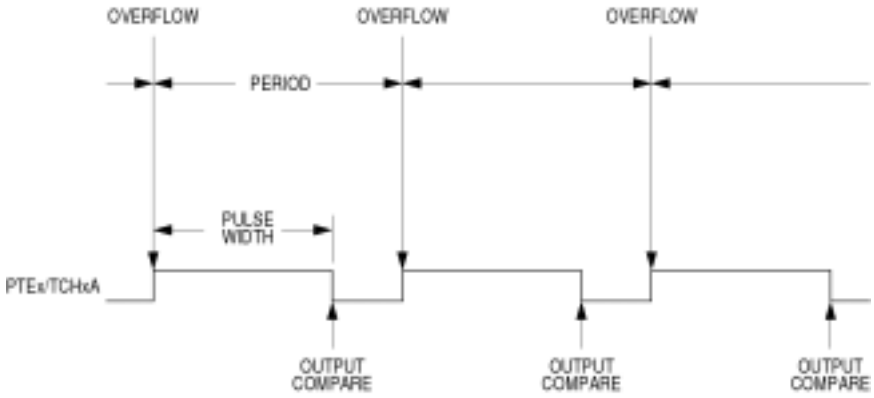

图 14-2　PWM 周期和脉宽

由于 PWM 的发生，用到了 TIM 的输出比较功能，所以，对于 PWM 来说，同样存在修改输出比较寄存器时带来的输出脉冲暂时不稳定的情况。因此，在 MC68HC(9)08JL3 中，也有不带缓冲的 PWM 和带缓冲的 PWM 之分。它们的操作和前面的输出比较功能是一样的，这里不再重复。唯一的区别，在于通道状态控制寄存器（TSCx）中的 TOVx 位设置的不同，PWM 方式下，要是该位有效，使得时钟溢出时，切换输出引脚上的电平。

PWM 初始化顺序如下：

1. TIM 状态控制寄存器 TSC：设置 TSTOP 位，使 TIM 停止；设置 TRST 位，使 TIM 复位；
2. 在 TMODH:TMODL 中写入 PWM 的周期（模数）；
3. 在 TIM 通道寄存器（TCHxH:TCHxL）中写入 PWM 脉宽；
4. 在 TIM 通道状态控制寄存器（TSCx）中：写入 0:1（无缓冲的输出比较或者 PWM 信号）或者 1:0（缓冲的输出比较或者 PWM 信号）到模式选择位 MSxB:MSxA；TOVx 写入 1，在溢出时，切换输出引脚上的电平；在沿/电平选择位，ELSxB:ELSxA，写入 1:0（清除输出引脚点平）或者 1:1（设置输出引脚电平）。
5. 在 TSC 中清除 TSTOP 位。

TIM 可以向 CPU 产生以下的中断请求：

- TIM 溢出标志（TOF）：当 TIM 计数器中的指到达模数寄存器的值后，计数器翻转到$0000，同时置位 TOF。如果 TIM 溢出中断请求是允许的（TOIE=1），TIM 同时

向 CPU 请求溢出中断。TOF 和 TOIE 在 TIM 状态控制寄存器 TSC 中。

- TIM 通道标志（CH1F:CH0F）：当 TIM 的通道 x 发生输入捕捉或者输出比较事件时，TIM 置位 CHxF。该中断请求受允许位 CHxIE 的控制。当 ChxIE=1 时，中断允许。CHxF 和 ChxIE 在通道 x 的状态控制寄存器 TSCx 中。

WAIT 方式下的 TIM：WAIT 指令使 CPU 进入一种低功耗的空闲状态。在该方式下，TIM 仍然在运行，但 TIM 的寄存器对 CPU 是不可访问的。任何 CPU 允许的 TIM 中断，可以使 CPU 跳出 WAIT 方式。如果在 CPU 处于 WAIT 中断期间，不需要 TIM，可以在运行 WAIT 指令之前，停止 TIM 的运行。这样，可以进一步降低系统的功耗。

Break 中断期间的 TIM：break 中断将使 TIM 计数器停止运行。系统集成模块（SIM）控制其它模块的状态位在 break 状态下是否被清除。Break 标志控制寄存器（BFCR）中的 BCFE 位置位使得软件可以在 break 状态期间清除各状态位。一旦某状态位在 break 期间被清除，在 CPU 退出 break 状态时，该状态位将仍然保持清除状态。为了保护这些状态位，可以在 BCFE 位写入 0。这种状态下，软件可以读写 I/O 寄存器，而不影响各状态寄存器。

TIM 对外的引脚包括：

和 PORTD 共享两个引脚，PTD4/TCH0 和 PTD5/TCH1；

外部时钟输入引脚 TCLK。

		位 7	6	5	4	3	2	1	位 0	
TSC	读	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0	$0020
	写				TRST					
复位：		0	0	1	0	0	0	0	0	

图 14-3　TIM 状态控制寄存器（TSC）

TOF — TIM 溢出标志位。1 表示发生了计数器溢出。

TOIE — TIM 溢出中断允许位。1 使得溢出中断允许。

TSTOP — TIM 停止位。1 使得 TIM 计数器停止。

TRST — TIM 复位位。1 使得 TIM 计数器清零。

PS[2:0] — TIM 时钟源选择。选择关系表 14-1 所示。

表 14-1　TIM 的时钟源选择关系

PS[2:0]	TIM 时钟源
000	内部总线时钟÷1
001	内部总线时钟÷2
010	内部总线时钟÷4
011	内部总线时钟÷8
100	内部总线时钟÷16
101	内部总线时钟÷32
110	内部总线时钟÷64
111	TCLK

TIM 通道状态控制寄存器（TSC0:TSC1）如图 14-4 所示。

CHxF — 通道 x 的标志位。1 表示有输入捕捉或者输出比较事件在该通道发生。

CHxIE — 通道 x 的中断允许位。1 使得该通道可以向 CPU 发中断请求。

MSxB，MSxA — 模式选择位，参照表 14-2。

ELSxB，ELSxA — 跳变沿/电平选择，参照表 14-2。

TOVx — 溢出时切换引脚电平。1 使得当通道 x 的引脚电平在 TIM 计数器发生溢出时被切换。

CHxMAX — 通道 x 最大占空比选择。1 使得该通道 PWM 波形拥有最大的占空比。该位的设置在下一个 TIM 时钟周期开始起作用，如图 14-5 所示。

TOVx — 溢出时切换引脚电平。1 使得当通道 x 的引脚电平在 TIM 计数器发生溢出时被切换。

CHxMAX — 通道 x 最大占空比选择。1 使得该通道 PWM 波形拥有最大的占空比。该位的设置在下一个 TIM 时钟周期开始起作用，如图 14-5 所示。

		位 7	6	5	4	3	2	1	位 0	
TCNTH	读	位 15	14	13	12	11	10	9	位 8	$0021
	写									
TCNTL	读	位 7	6	5	4	3	2	1	位 0	$0022
	写									
复位：		0	0	0	0	0	0	0	0	

图 14-4　TIM 通道状态控制寄存器 (TSC0:TSC1)

表 14-2 模式、跳变沿选择

MSxB：MSxA	ELSxB：ELSxA	模式	选择的配置
X0	00	Output Preset	Pin under port Control; Initial Output Level High
X1	00		Pin under Port Control; Initial Output Level Low
00	01	Input Capture	Capture on Rising Edge Only
00	10		Capture on Falling Edge Only
00	11		Capture on Rising or Falling Edge
01	01	Output Compare or PWM	Toggle Output on Compare
01	10		Clear Output on Compare
01	11		Set Output on Compare
1X	01	Buffered Output Compare or Buffered PWM	Toggle Output on Compare
1X	10		Clear Output on Compare
1X	11		Set Output on Compare

		位 7	6	5	4	3	2	1	位 0	
TMODH	读	位 15	14	13	12	11	10	9	位 8	$0023
	写									
TMODL	读	位 7	6	5	4	3	2	1	位 0	$0024
	写									
复位		1	1	1	1	1	1	1	1	

图 14-5　CHxMAx 延迟

15. ADC 转换器

MC68HC(9)08JL3 包含有一个 8 位的 12 路 ADC 转换器。　其主要特性为:

12 路可复用的 ADC 通道
结果为连续线性近似
8 位的分辨率
单一或连续转换
具有转换结束标志或中断功能
可选的 ADC 时钟

ADC 的 12 路输入引脚分别为 PTB7~PTB0，PTD3~PTD0。可以选择其中的任何一条腿作为 ADC 的电压输入，ADC 通过近似连续的计数器值来标定转换结果。其精度为 8 位。当转换结束，ADC 将结果放入 ADC 数据寄存器，并且设立标志或产生中断。

PTB7~PTB0，PTD3~PTD0 可作为通常 I/O 引腿或 ADC 的输入，通过设定(ADC 状态控制寄存器，$003C)来选择哪一路作为 ADC 输入。一旦设定为 ADC 输入，其作为普通 I/O 功能失效，这时，向此位的端口寄存器或 DDR 写入无效，而读时，倘若 DDR 相应位为 0，读入为 0;而相应位为 1，读出的将是以前的锁存值。这时，其它未设为 ADC 输入的引腿保持其正常的 I/O 功能。

ADC 转换为线性转换，高于或等于 Vss，其结果为$FF， 低于或等于 Vdd，其结果为$00。每次转换要 16 个 ADC 内部时钟。ADC 从写 ADSCR 信号的第一个 ADC 内部时钟的上升沿开始转换。倘若 ADC 内部时钟为 1MHz，其完成一次转换耗时 16us。

当连续转换时，ADC 连续对所选通道进行转换，并对 ADC 的数据寄存器进行每次更新，而不管读出与否。转换通过 ADCO 位清零来终止。每次转换完后 COCO 位(ADC 状态控制寄存器，$003C)置一，可以通过写 ADC 状态控制寄存器，$003C 或读 ADC 数据寄存器来清零。

当 AIEN 置 1，允许 ADC 转换结束中断。而且 COCO 位为 0，就会发生中断。但是允许中断时，COCO 位并不是作为转换结束标志。

低功耗下:

WAIT 模式下，ADC 正常工作，ADC 中断可以将 CPU 从 WAIT 模式中唤醒。倘若不希望 ADC 中断将其唤醒，可以在执行 WAIT 指令前，把 ADC 状态控制寄存器的 CH[4-0]设为 1 来关闭 ADC。

STOP 模式下，ADC 停止工作，放弃当前的转换。当程序一旦退出 STOP 模式，ADC 转换继续进行。在退出 STOP 模式之后，继续转换之前会有一个转换时间来稳定转换电路。

有关的寄存器如下:

ADC 状态控制寄存器(ADSCR)

地址:$003C Bit 7 6 5 4 3 2 1 0

读取	COCO	AIEN	ADCO	CH4	CH3	CH2	CH1	CH0
写入								
复位:	0	0	0	0	1	1	1	1

COCO - 转换结束位

当 AIEN 位为 0，COCO 位只读，每次转换完后 COCO 位置 1。可以通过写 ADC 状态控制寄存器或读 ADC 数据寄存器来清零。复位清零。

1=转换结束(AIEN=0)
0=转换未结束(AIEN=0)
0=中断允许(AIEN=1)
AIEN - ADC 中断允许位
当这位为一，ADC 转换结束后中断发生。 当写 ADC 状态控制寄存器或读 ADC 数据寄存器时信号清零。复位清零。
1=ADC 中断允许
0=ADC 中断禁止
ADCO - ADC 连续转换位
当为 1 时，ADC 连续转换，每次更新。为 0 时，进行一次转换。 复位清零。
1=连续 ADC 转换
0=单次 ADC 转换
ADCH[4:0] - ADC 通道选择位
ADCH4~0,。用于选择 ADC 通道。特别注意的就是当引脚同时为模拟输入和数字输入切换时产生的开关躁声。
当所有位设为 1 时，关闭 ADC。这样减少 MCU 的消耗。复位时均为 1。

ADC 数据寄存器(ADR)

8 位的转换结果。每次转换后更新。

地址:$003D Bit 7 6 5 4 3 2 1 0

读取	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0

ADC 时钟寄存器(ADICLK)

用于选择 ADC 时钟

地址:$003E Bit 7 6 5 4 3 2 1 0

读/写	ADIV2	ADIV1	ADIV0					
复位:	0	0	0	0	0	0	0	0

ADIV2~0 用于选择 ADC 内部时钟的分频率，ADC 时钟应设在在 1MHz 左右。

16. 键盘中断模块

键盘中断模块（KBI）通过 PTA0-PTA6 引脚提供了 7 个独立可屏蔽的外部中断。其主要特性：

- ✧ 7 个键盘中断引脚及各自的键盘中断允许位和一个键盘中断屏蔽位。
- ✧ 当引脚设定为输入时，软件可设定其上拉电阻。
- ✧ 软件设定其触发方式：沿触发或沿及电平触发。
- ✧ 可从低功耗模式中唤醒。

向键盘中断允许寄存器（KBIE6-KBIE0）位写可独立的允许或禁止端口 A 腿的键盘中断功能。允许端口 A 的键盘中断意味着允许其上拉设备。向端口 A 的上拉允许寄存器的(PTAPUE6-PTAPUE0)位写可以独立的允许或禁止其上拉设备。当一个逻辑 0 在允许的键盘中断腿上出现时，就锁存了一个键盘中断请求。

只有当一个或多个键盘中断腿在复位为高又变低以后，才会锁存一个键盘中断请求。键盘状态和控制寄存器中的 MODE 位来控制键盘中断的触发模式。

倘若键盘中断仅仅为沿触发，倘若在已有一个键盘引脚为低的情况下，又来一个下降沿，这时，此下降沿不会产生键盘请求。为了防止这种情况的发生，可以软件上把已变低的引脚的键盘中断功能禁止掉。

倘若键盘中断为沿触发和低电平触发，一个中断请求可以在引脚为低的情况下一直保持下去。

MODE 位如果为 1，键盘中断为沿触发和低电平触发，以下的两个动作都可以用于清除键盘中断请求。

1．取得中断向量以及软件清除-取得中断向量会产生一个中断应答信号用来清除键盘中断请求。而软件可以通过向键盘状态和控制寄存器 KBSCR 的 ACK 位写 1 来产生中断应答信号。ACK 通常在软件循环检测键盘中断腿，并且要求软件清除键盘中断请求时非常有用。在中断例程之前优先写 ACK 位可以防止由于躁声引起的虚假中断。设置 ACK 位并不影响随后在键盘中断腿上发生的变化。在写了 ACK 信号之后发生的一个下降沿会锁存另一个中断请求。倘若中断屏蔽位 IMASK 为零，CPU 从$FFEO~$FFFE1 处开始执行。

2．所有允许的键盘中断腿都返回 1-只要任何的键盘中断腿为 0， 键盘中断请求就会保持。

取得中断向量以及软件清除和所有允许的键盘中断腿都返回 1 可以以任何顺序发生。

倘若 MODE 位为 0，键盘中断仅仅为沿触发。 当 MODE 位为 0 中断向量的取得以及软件清除会立即清除键盘中断请求。

RESET 清除键盘中断请求和 MODE 位，即使在有一个键盘中断腿为 0 时也如此。

键盘标志位(KEYF)在键盘状态和控制寄存器可用于查看是否存在一个键盘中断。KEYF 位不受 IMASK 位的影响，这一点在查询时特别有效。

为了获得键盘中断腿上的逻辑电平，需要禁止上拉设备，使用数据方向寄存器把引脚设为输入，然后再读。

注意:设置键盘中断允许位(KBIEx)会强制相应的键盘中断腿为输入，而不管数据方向寄存器。

键盘初始化:

当一个键盘中断允许时，它要使内部的上拉达到 1 需要一定的时间。这样的话，就有可能会发生虚假中断。

为了防止这一点:

- 通过设置键盘状态和控制寄存器中的 IMASK 位来屏蔽键盘中断。
- 向 DDRAx 位写入来设定端口为输入并允许上拉。
- 通过设定键盘中断允许寄存器中恰当的 KBIEx 位来允许 KBI 腿。
- 向键盘状态和控制寄存器中的 ACK 位写 1 来清除虚假中断。
- 清除 IMASK 位。

一个沿触发的引腿可以在引腿一允许的情况下立即反应，而一个沿触发或电平触发的引腿则要在一段时延之后才会反应，这段时延取决与外部负载。

键盘状态和控制寄存器(KBSCR)

其主要功能为标志键盘中断请求;回应键盘中断请求;屏蔽键盘中断请求;控制键盘中断触发模式

		位 7	6	5	4	3	2	1	位 0	
KBSCR	读	0	0	0	0	KEYF	0	IMASK	MODE	$001A
	写						ACK			
复位：		0	0	0	0	0	0	0	0	

Bits7-无用,只读，其为 0

KEYF - 键盘标志位

只读，当存在一个键盘中断时为 1。RESET 清零。

1=存在键盘中断

0=不存在键盘中断

ACK - 键盘中断请求回应

只写，写 1 清除键盘中断请求。读时为 0，RESET 清零。

IMASK - 键盘中断屏蔽位

可读写，用于屏蔽键盘中断。 RESET 清零。

1=键盘中断屏蔽

0=键盘中断未屏蔽

MODE - 键盘中断触发模式

可读写，控制键盘中断的触发模式。RESET 清零

1=键盘中断为沿触发和低电平触发

0=键盘中断为沿触发

键盘中断允许寄存器(KBIER)

		位 7	6	5	4	3	2	1	位 0	
KBIER	读		KBIE6	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0	$001B
	写									
复位：		0	0	0	0	0	0	0	0	

键盘中断允许寄存器允许或禁止端口 A 的键盘中断位。

KBIE6-KBIE0 - 端口 A 键盘中断允许位

可读写，分别允许或禁止相应的键盘中断腿。RESET 清零。

1=允许相应的键盘中断腿

0=禁止相应的键盘中断腿