特性

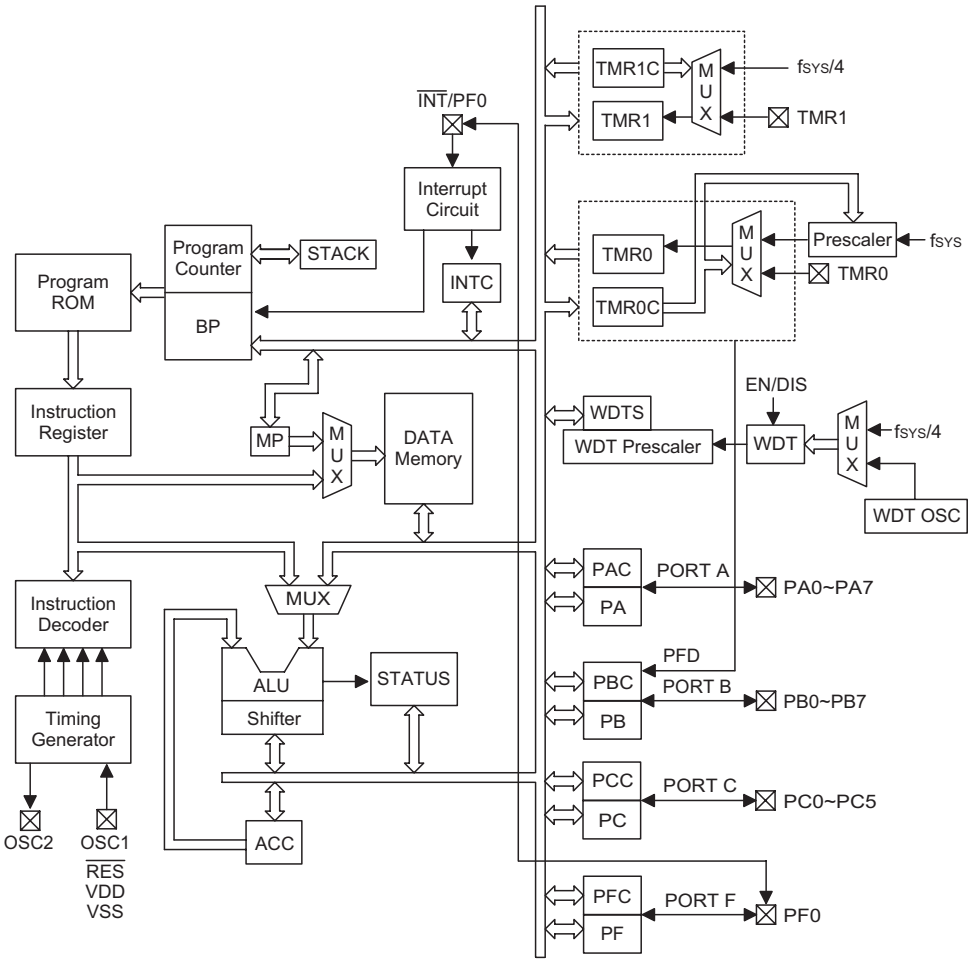
- 工作电压：2.2V ~3.6V
- 最多有 23 个双向输入/输出口
- 1 个与输入/输出口共用引脚的外部中断输入
- 8 位可编程定时/计数器，具有溢出中断以及 8 位预分频器
- 16 位可编程定时/计数器，具有溢出中断
- 内置晶体和 RC 振荡电路
- 看门狗定时器
- 24K×16 程序数据存储器 ROM（8K×16bits ×3banks）
- 224×8 数据存储器 RAM
- PFD 输出
- HALT 和唤醒功能以降低功耗
- 8 层堆栈
- VDD=3V 时，系统时钟为 4MHz，指令周期为 1μs
- 位操作指令
- 16 位的查表指令
- 63 条指令
- 所有的指令在 1 到 2 个指令周期里完成
- 28-pin 的 SKDIP/SOP 封装

概述

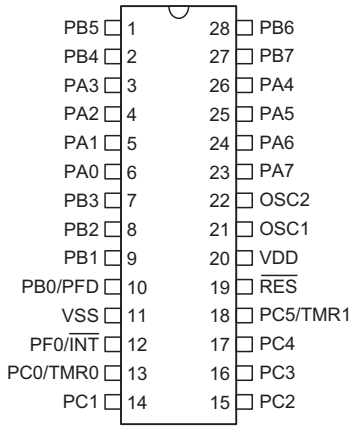
HT48CA3 为八位高性能精简指令集单片机，专为多输入/输出口的产品而设计的。程序存储器 ROM 存放遥控控制代码。此款单片机是掩膜版本，它和 OTP 版本芯片 HT48RA3 在引脚和功能上完全相同。

拥有低功耗、I/O 口稳定性高、定时器功能、振荡选择、省电和唤醒功能、看门狗定时器、蜂鸣器驱动、以及低价位等优势，使此款多功能芯片可以广泛地适用于各种应用，例如工业控制、消费类产品、子系统控制器，特别适用于例如通用红外遥控器（URC）等的产品应用。

方框图


TMR1C
TMR1
M
U
X
fSYS/4
TMR1
INT/PF0
Interrupt Circuit
Program Counter
STACK
Program ROM
BP
INTC
TMR0
TMR0C
Prescaler
fSYS
TMR0
EN/DIS
WDTS
WDT Prescaler
WDT
fSYS/4
WDT OSC
Instruction Register
MP
DATA Memory
Instruction Decoder
MUX
ALU
Shifter
STATUS
Timing Generator
ACC
OSC2
OSC1
RES
VDD
VSS
PAC
PA
PORT A
PA0~PA7
PFD
PBC
PB
PORT B
PB0~PB7
PCC
PC
PORT C
PC0~PC5
PFC
PF
PORT F
PF0

引脚图



HT48CA3
– 28 SKDIP-A/SOP-A

引脚说明

引脚名称	输入/输出	掩膜选项	说　　明
$\overline{\text{RES}}$	输入	—	斯密特触发复位输入端，低电平有效。
PA0~PA7	输入/输出	唤醒功能/上拉电阻***	8 位双向输入/输出端口。每一位都可在掩膜选项中设置为带唤醒功能的输入口，可由软件指令设定为 CMOS 输出或有/无上拉电阻的斯密特触发输入。每个输入/输出引脚的上拉电阻都是可选择的。
PB0/PFD PB1~PB7	输入/输出	上拉电阻** PB0 或 PFD	8 位双向输入/输出端口。可由软件指令设定为 CMOS 输出或有/无上拉电阻的斯密特触发输入。每个输入/输出引脚的上拉电阻都是可选择的。PB0 的输出模式可被用于 PFD 的输出口并用于多种频率的载波信号。
VSS	—	—	负电源，接地。
PC0/TMR0 PC1~PC4 PC5/TMR1	输入/输出	上拉电阻*	6 位双向输入/输出端口。可由软件指令设定为 CMOS 输出或有/无上拉电阻的斯密特触发输入。每个输入/输出引脚的上拉电阻都是可选择的。PC0、PC5 和 TMR0、TMR1 共用引脚
PF0/$\overline{\text{INT}}$	输入/输出	上拉电阻*	1 位双向输入/输出口。可由软件指令设定为 CMOS 输出或有/无上拉电阻的斯密特触发输入。每个输入/输出引脚的上拉电阻都是可选择的。PF0 和 $\overline{\text{INT}}$ 功能共用引脚
VDD	—	—	正电源
OSC1 OSC2	输入 输出	晶体振荡 或 RC 振荡	OSC1，OSC2 可连接 RC 电路或晶体（由硬件决定）以产生内部的系统时钟。在选择 RC 方案时，OSC2 是系统时钟四分频的输出端。

注意：　*：位选择
　　**：半字节选择
　　***：字节选择

极限参数*

电源供应电压……Vss – 0.3V～Vss +4.0V　　储存温度 …… - 50℃～125℃
端口输入电压……V_{SS} – 0.3V～V_{DD} +0.3V　　工作温度 …… - 40℃～ 85℃

注意：这里只强调额定功率，超过极限参数所规定的范围将对芯片造成损害，无法预期芯片在上述标示范围外的工作状态，而且若长期在标示范围外的条件下工作，可能影响芯片的可靠性。

D.C.特性 **Ta=25℃**

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压	—	—	2.2	—	3.6	V
I_{DD}	工作电流	3V	无负载 fsys=4MHZ	—	3	5	mA
I_{STB1}	静态电流(看门狗打开)	3V	无负载 暂停模式	—	5	10	μA
I_{STB2}	静态电流(看门狗关闭)	3V	无负载 暂停模式	—	0.1	1	μA
V_{IL1}	输入/输出口的低电平输入电压	—	—	0	—	0.2 V_{DD}	V
V_{IH1}	输入/输出口的高电平输入电压	—	—	0.8V_{DD}	—	V_{DD}	V
V_{IL2}	低电平输入电压($\overline{RES}$)	—	—	0	—	0.4V_{DD}	V
V_{IH2}	高电平输入电压($\overline{RES}$)	—	—	0.9V_{DD}	—	V_{DD}	V
I_{OL}	输入/输出口灌电流	3V	VOL=0.1V_{DD}	5	10	—	mA
I_{OH1}	输入/输出口源电流	3V	VOH=0.9V_{DD}	-2	-5	—	mA
I_{OH2}	输入/输出口源电流	3V	VOH=0.8V_{DD}	-4	-8	—	mA
R_{PH}	上拉电阻	3V	—	40	60	80	KΩ

A.C.特性 **Ta=25℃**

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
f_{sys}	系统时钟	3V	—	400	—	4000	KHZ
f_{timer}	定时器输入频率(TMR0/TMR1)	3V	50%占空比	0	—	4000	KHz
t_{WDTOSC}	看门狗振荡器	3V	—	45	90	180	μs
t_{WDT1}	看门狗溢出周期（WDT 振荡）	3V	WDT 无预分频	11.5	23	46	ms
t_{WDT2}	看门狗溢出周期（f_{sys}/4）	3V	WDT 无预分频	—	1024	—	t_{sys}
$t_{\overline{RES}}$	外部复位低电平脉宽	—	—	1	—	—	μs
t_{SST}	系统启动延迟周期	—	上电或 HALT 唤醒	—	1024	—	t_{sys}
$t_{\overline{INT}}$	中断脉冲宽度	—	—	1	—	—	μs
t_{RES}	外部复位低电平脉冲宽度	—	—	1	—	—	μs

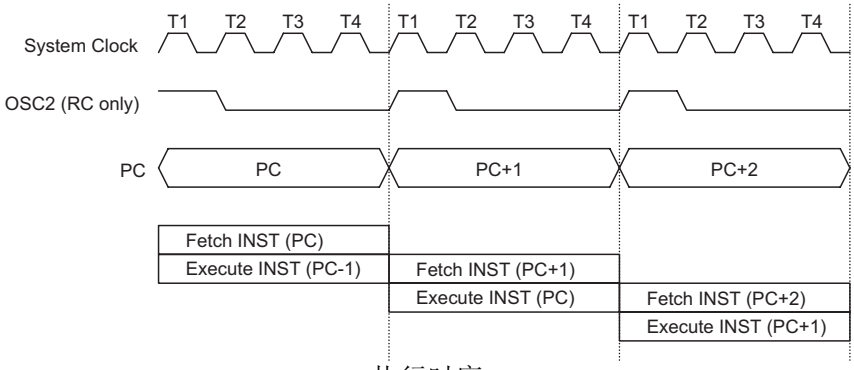
注：$t_{SYS}=1/f_{SYS}$

系统功能说明

指令系统

单片机系统时钟来自于晶体振荡或 RC 振荡。系统时钟被分割成四个不重叠的时钟。一个指令周期由 4 个系统时钟周期组成。

指令读取与执行是以流水线方式来进行的。这种方式允许在一个指令周期进行读取指令操作，而在下一个指令周期里进行解码与执行该指令。这种流水线方式能在一个指令周期里有效地执行一个指令。但是，如果指令会改变程序计数器的值，就需要花两个指令周期来完成这一条指令。



执行时序

程序计数器（PC）

10 位的程序计数器控制程序存储器 ROM 中指令执行的顺序。

通过访问一个程序存储单元来取出指令代码后，PC 的值便会加 1。然后程序计数器便会指向下一条指令代码所在的程序存储单元。

当执行一条跳转指令、条件跳转指令、装载 PCL 寄存器、子程序调用、初始复位或从一个子程序返回，PC 会通过装载每条指令的相应地址来执行程序转移。

通过指令实现条件跳转，一旦条件满足，那么在当前指令执行期间取出的下一条指令会被放弃，而替代它的是一个空指令周期(dummy cycle)来获取正确的指令，接着就执行这条指令。否则就执行下一条指令。

程序计数器的低字节（PCL；06H）是可读写的寄存器。将数据赋值到 PCL 会执行一个短跳转。这种跳转只能在当前 ROM 页地址范围内。

当一个控制转移指令发生时，就需要有一个附加的空指令周期。

模　　式	程序计数器								
	***14~*8**	***7**	***6**	***5**	***4**	***3**	***2**	***1**	***0**
初始化复位	0000000	0	0	0	0	0	0	0	0
外部中断	0000000	0	0	0	0	0	1	0	0
定时/计数器 0 中断	0000000	0	0	0	0	1	0	0	0
定时/计数器 1 中断	0000000	0	0	0	0	1	1	0	0
条件跳转	*14~*13，(*12~*0+2)：(在当前的 BANK)								
装载 PCL	*14~*8	@7	@6	@5	@4	@3	@2	@1	@0
跳转，子程序调用	BP6~5,#12~#8	#7	#6	#5	#4	#3	#2	#1	#0
从子程序返回	S14~S8	S7	S6	S5	S4	S3	S2	S1	S0

程序计数器

注意：　*14 ~ *0 ：程序计数器位　　S14 ~ S0 ：堆栈寄存器位

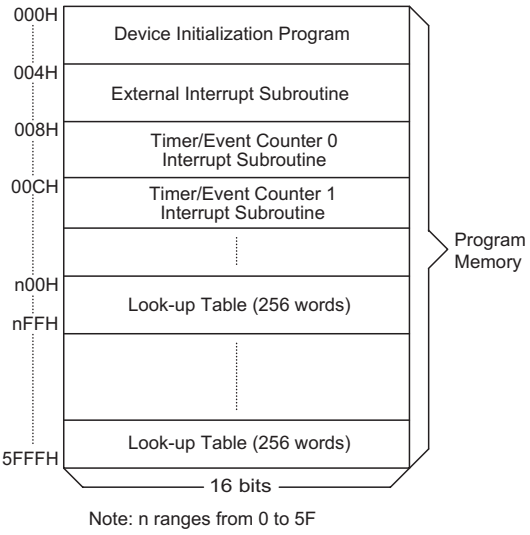
#14 ~ #0 ：指令代码位　　@7 ~ @0 ：PCL 位

1 bank：8K Words

程序存储器（ROM）

程序存储器（ROM）被用来存放要执行的指令的代码。还包括数据和表格以及中断相应。被组织为 8192×16 位×3banks。可以由程序计数器或表格指针来寻址。

在程序存储器中某几个地址被保留作为特殊用途：



Note: n ranges from 0 to 5F

- 地址 000H
  此地址保留给程序初始化之用。当系统复位时，程序会从 000H 地址开始执行。
- 地址 004H
  这个地址是为外部中断服务程序保留的。如果中断输入引脚是有效的，中断是允许的，并且堆栈没有满，那么程序就会从 004H 地址开始执行。
- 地址 008H
  这个地址是为定时/计数器 0 中断服务程序保留的。如果定时/计数器 0 溢出引起了时间中断，中断是允许的，并且堆栈没有满，那么程序就会从 008H 开始执行。
- 地址 00CH
  这个地址是为定时/计数器 1 中断服务程序保留的。如果定时/计数器 1 溢出引起了时间中断，中断是允许的，并且堆栈没有满，那么程序就会从 00CH 开始执行。
- 表格区
  程序存储器内的任何地址都可被用来作为表格地址使用。查表指令 TABRDC [m]（查当前页，1 页=256 个字节，页由 TBHP 确定）和 TABRDL [m]（查最后一页）将所指地址的低字节传到指定的存储器中，而高字节传送到 TBLH （08H）。表格内容中低八位被完整传到到目标地址中，而高八位则传送到 TBLH。表格中的高位字节 TBLH 为只读寄存器。而高位表格指针 TBHP（1FH），低位表格指针 TBLP（07H）是可以读写的寄存器，表格指针指向要读的表格地址。在访问表格以前，通过对 TBHP 和 TBLP 寄存器赋值来指明表格地址。TBLH 是只读的寄存器，并且不能保存它的值，如果主程序和 ISR（中断服务程序）都使用了查表指令，那么主程序中的查表指令的 TBLH 的结果很可能被 ISR 中的查表命令改变。这样将产生错误。因此，应该避免在主程序和 ISR 中都使用查表指令的情况。然而，如果同时在主程序和 ISR 中使用查表指令，就要避免 ISR 中的查表指令优先执行。因该在主程序中的 TBLH 没有备份之前不要开启中断。查表指令要花两个指令周期来完成这一条指令的操作。

指　令	表格地址								
	***14~*8**	***7**	***6**	***5**	***4**	***3**	***2**	***1**	***0**
TABRDC [m]	TBHP	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1011111	@7	@6	@5	@4	@3	@2	@1	@0

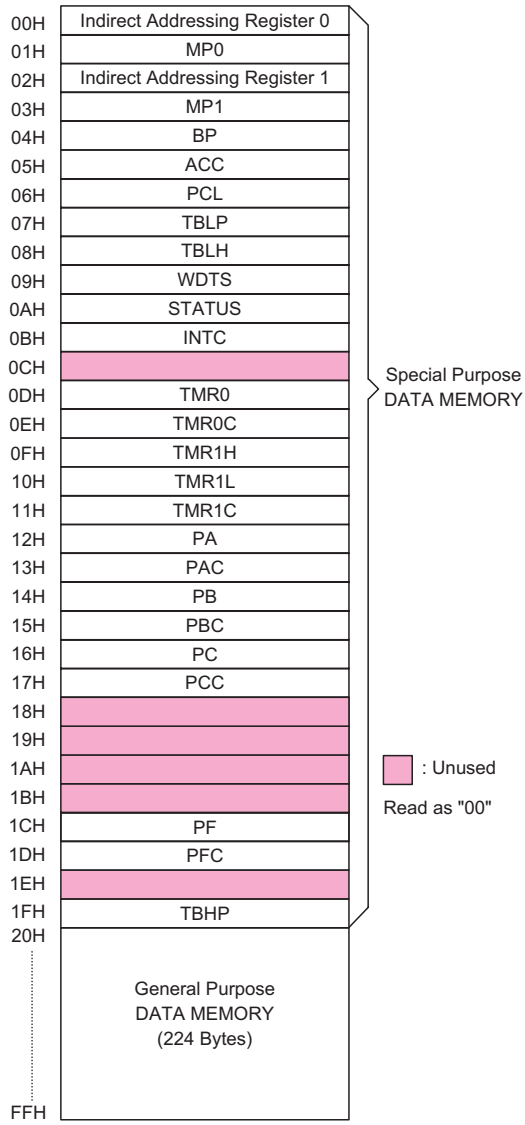
表格区

注意：*14 ~ *0 ： 表格地址位　　@7 ~ @0 ： 表格指针位

堆栈寄存器（STACK）

堆栈寄存器（STACK）是一个用来保存 PC 值的特殊存储单元。HT48CA3 的堆栈为 8 级。堆栈寄存器既不是数据存储器的一部分，也不是程序存储器的一部分，而且它既不能读出，也不能写入。当前堆栈位置由堆栈指针（SP）来确定。堆栈指针（SP）也不能读/写。一旦发生子程序调用时，当前程序计数器（PC）的值会被压入堆栈，在子程序调用结束时, 通过一条返回指令（RET 或 RETI），堆栈将原先压入堆栈的内容弹出，重新装入程序计数器（PC）中。在系统复位后，堆栈指针会指向堆栈顶部。

如果堆栈满了，并且一个没有屏蔽的中断发生了，中断请求的标志位会被记录，但是相应信号会抑制。当堆栈指针被消耗（通过 RET 或 RETI），中断服务程序才开始执行。这个防止中断溢出的特点使得程序员使用指令变得比较容易。在一个简单的情况下，如果堆栈满了，接着又执行一个子程序调用（CALL），那么堆栈会产生溢出，而使第一个进入堆栈的内容将会丢失。（只有最后 8 个返回地址会被保留着）。



数据存储器（RAM）

数据存储器 RAM 由 250×8 位组成。它可分成两个功能组：特殊功能寄存器和通用数据存储器（224×8）。这两个功能组的大部分单元可以读写，而某些单元只能读，不能写。

特殊功能寄存器包括间接寻址寄存器（R0：00H，R1：02H），定时/计数器 0（TMR0：0DH），定时/计数器 0 控制寄存器（TMR0C：0EH），定时/计数器 1 高位寄存器（TMR1H：0FH），定时/计数器 1 低字节寄存器（TMR1L：10H），定时/计数器 1 控制寄存器（TMR1C：11H），程序计数器低字节寄存器（PCL：06H），间接寻址指针寄存器（MP0：01H，MP1：03H），累加器（ACC：05H），表格指针寄存器（TBLP：07H，TBHP：1FH），表格高字节寄存器（TBLH：08H），状态寄存器（STATUS：0AH），中断控制寄存器（INTC：0BH），看门狗设置寄存器（WDTS：09H），输入/输出口寄存器（PA：12H，PB：14H，PC：16H，PF：1CH）以及输入/输出控制寄存器（PAC：13H，PBC：15H，PCC：17H，PFC：1DH）。20H 以前的剩余单元都被保留为将来进一步扩展使用。读取这些被保留单元的值，都将返回 00H。通用数据存储器地址 20H-FFH 作为程序数据和控制信息使用。

所有的 RAM 区单元都能直接执行算术，逻辑，递增，递减和移位等运算。除了某些特殊的位以外，RAM 中的每一位都可以由 SET [m].i 和 CLR [m].i 指令来置位和清零。它们都可通过间接寻址指针寄存器（MP0：01H，MP1：03H）间接寻址来存取。

间接寻址寄存器

地址 00H 和 02H 是作为间接寻址寄存器。它们没有在物理上实现。任何对[00H]（[02H]）的读/写操作，都会访问由 MP0(MP1)所指向的 RAM 单元。间接地读取[00H]（[02H]），将会返回 00H（[02H]），而间接地写入[00H]（[02H]）单元，则不会产生任何结果。

间接寻址指针寄存器 MP0，MP1 是 8 位寄存器。

累加器 （ACC）

累加器（ACC）与算术逻辑单元（ALU）运算相联系。它对应于 RAM 的地址 05H，并能与立即数进行操作，在两个存储器间的数据传送都必须经过累加器。

算术逻辑单元 （ALU）

算术逻辑单元是执行 8 位算术逻辑运算的电路。它提供如下的功能：

- 算术运算 （ADD，ADC，SUB，SBC，DAA)
- 逻辑运算 （AND，OR，XOR，CPL)
- 移位运算 （RL，RR，RLC，RRC ）
- 递增和递减运算（INC，DEC ）
- 分支转移 （SZ，SNZ，SIZ，SDZ 等）

算术逻辑单元 ALU 不仅会保存运算的结果而且会改变状态寄存器。

状态寄存器 （STATUS）

状态寄存器（STATUS：0AH）为 8 位寄存器，由零标志位（Z）、进位标志位（C）、辅助进位标志位（AC）、溢出标志位（OV）、暂停模式标志位（PD）、看门狗定时器溢出标志位（TO）组成。该寄存器所不仅记录状态信息，而且还可控制运算顺序。

除了 TO 和 PD 以外，状态寄存器中其他位都可用指令来改变，这种情况与其它寄存器一样。任何写到状态寄存器的数据不会改变 TO 或 PD 标志位。但是与状态寄存器有关的运算会导致状态寄存器的变化。

系统上电，看门狗定时器溢出，执行 HALT 指令，或清除看门狗定时器都能改变 TO 和 PD 标志位。 Z、OV、AC 和 C 标志位都反映了当前的运算状态。

另外，执行子程序调用时，状态寄存器的内容不会自动压入堆栈。如果状态寄存器的内容是重要的，而且子程序会改变状态寄存器的内容，那么程序员必须事先将其保存好，以免被破坏。

符号	位	功　　能
C	0	如果在加法运算中结果产生了进位，或在减法运算中结果不发生借位,那么 C 被置位；反之，C 被清除。它也可被一个带循环进位指令而影响。
AC	1	在加法运算中低四位产生了向高四位进位，或减法运算中低四位不发生从高四位借位，AC 被置位；反之，AC 被清除。
Z	2	算术运算或逻辑运算的结果为零则 Z 被置位；反之，Z 被清除。
OV	3	如果运算结果向最高位进位，但最高位并不产生进位输出，那么 OV 被置位；反之，OV 被清除。
PD	4	系统上电或执行了 CLR WDT 指令，PD 被清除。执行 HALT 指令 PD 被置位。
TO	5	系统上电或执行了 CLR WDT 指令或执行了 HALT 指令，TO 被清除。WDT 定时溢出，TO 被置位。
—	6	未定义，读出为零
—	7	未定义，读出为零

中断

该单片机提供了一个外部的中断和两个内部定时器/计数器的中断。中断控制寄存器（INTC：0BH），包含了中断控制位和中断请求标志，中断控制位用来设置中断允许/禁止。。

当一个中断服务程序运行时，所有其他的中断将被屏蔽（通过清除 EMI 标志位），这样可防止多层的中断嵌套。在这期间，别的中断到来只记录它的中断请求标志。如果某一个中断要求中断服务程序响应，则 EMI 位和响应的 INTC 位要被置 1 以允许中断嵌套。如果堆栈满了，则中断请求不被响应，哪怕这个中断是被允许的，直到 SP 递减后才响应这个中断。如果必须响应这个中断，那么就必须防止堆栈饱和。

所有的中断都有唤醒功能。在响应中断时，先把程序计数器压入堆栈，然后通过一个跳转子程序转到程序存储器的一个特定的地址。只有程序计数器被压入堆栈。如果其它寄存器和状态寄存器的内容会被中断程序改变，从而会破坏主程序的控制流程的话，程序员应该事先将这些数据保存起来。

外部中断由 $\overline{\text{INT}}$ 引脚上的下降沿触发，同时中断请求标志（EIF：INTC 的第 4 位）被置 1。当中断被允许，且堆栈没有满，此时发生外部中断事件，就会调用在 04H 的子程序。中断请求标志（EIF）和 EMI 位就会被清除，来禁止响应其他中断。

定时/计数器 0 中断由定时/计数器 0 的溢出而产生，并置位中断请求标志（T0F；INTC 的第 5 位）。当中断是允许的，堆栈没有满，并且 T0F 被置 1，就会调用在 08H 的子程序。中断请求标志（T0F）和 EMI 位就会被清除，以禁止其他的中断响应。

定时/计数器 1 中断由定时/计数器 1 的溢出而产生，并置位中断请求标志（T1F；INTC 的第 6 位）。当中断是允许的，堆栈没有满，并且 T1F 被置 1，就会调用在 0CH 的子程序。中断请求标志（T1F）和 EMI 位就会被清除以禁止其他的中断。

在执行中断子程序时，在“RETI”指令被执行或 EMI 位被置 1 之前，别的中断请求信号将一直被保持（如果堆栈没有满的话）。通过调用“RETI”或“RET”，可以从中断子程序返回。RETI 将把 EMI 置 1 以允许中断服务，而 RET 则不能。

中断在两个连续的 T2 的上升沿之间产生，如果允许响应中断，那么将在后面的两个连续的 T2 脉冲期间被服务。下面的表提供了当中断同时发生时的优先级顺序

编号	中断源	优先级	向量
A	外部中断	1	04H
B	定时/计数器 0 中断	2	08H
C	定时/计数器 1 中断	3	0CH

定时/计数器 0/1 中断请求标志（T0F/T1F），外部中断请求标志（EIF），定时/计数器 0/1 中断允许位（ET0/ET1），外部中断允许位（EEI）和中断允许控制位（EMI）组成了数据存储器 RAM 中的中断控制寄存器（INTC）。EMI，EEI，ET0I 和 ET1I 用来控制是否允许中断。这些位防止在服务一个中断时响应另一个中断。一旦中断请求标志（T0F，T1F，EIF）被置 1，它们将一直保存在 INTC 中，直到这些中断被服务

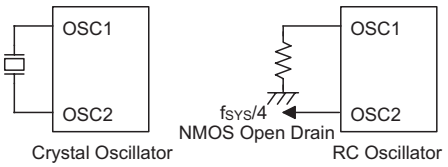
或通过软件指令把它们清除掉。

建议在中断子程序中不要使用“CALL ”指令。中断经常在不可预料的情况下发生并且在某些应用下要立即被服务。如果只剩下一层堆栈并且中断没有被很好的控制，一旦在中断子程序中使用 CALL 则原来的控制顺序就会变的很危险。

寄存器	位编号	标号	功能
INTC（0BH）	0	EMI	控制全部的中断（1 允许；0 禁止）
	1	EEI	控制外部中断（1 允许；0 禁止）
	2	ET0I	控制定时/计数器 0 中断（1 允许；0 禁止）
	3	ET1I	控制定时/计数器 1 中断（1 允许；0 禁止）
	4	EIF	外部中断请求标志（1 有效；0 无效）
	5	T0F	定时/计数器 0 中断请求标志（1 有效；0 无效）
	6	T1F	定时/计数器 1 中断请求标志（1 有效；0 无效）
	7	—	无效位，读出为 0

振荡器

HT48CA3 有两种振荡电路。通过掩膜选项选择，HT48CA3 可以选择晶振或 RC 振荡模式。两者都为产生系统时钟而设计的。不管用哪种振荡器，其信号都支持系统时钟。在暂停模式下，系统时钟会停振，并忽略外部输入信号，由此来节省功耗。



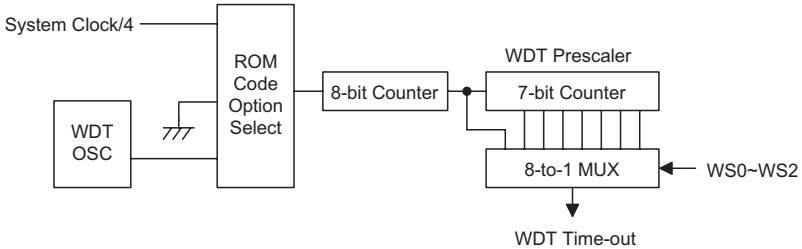
若采用 RC 振荡器，则在 VSS 与 OSC1 之间要接一个外接电阻。其阻值范围为 100KΩ~820KΩ。在 OSC2 端可获得系统时钟四分频信号，它可以用于同步系统外部逻辑。RC 振荡器是一种有效的方案，但是振荡频率由于 V_{DD}、温度及芯片自身参量的漂移，而产生误差。因此，对振荡器要求精确度高的场合，RC 振荡器是不适用的。

如果选用晶体振荡器，在 OSC1 与 OSC2 之间连接一个晶体，被用来提供晶体振荡器所要的反馈（Feedback）和相位位移（Phase Shift)。除此以外，不再需要其他外部元件。另外，可在 OSC1 与 OSC2 之间接一个谐振器（Resonator） 来取代晶体振荡器，用来得到参考频率，但是需要外接二个电容器。

看门狗振荡器时一个自由运行的片内的 RC 振荡器，不需要任何外部的元件。甚至在暂停模式下，系统时钟已经停止了，而看门狗振荡器还是以大约 90μs 的周期在运行。设置掩膜选项可打开/关闭看门狗振荡器。

看门狗定时器 （WDT）

看门狗定时器（WDT）的时钟源是由专用的 RC 振荡器（WDT 振荡器），指令时钟（系统时钟 4 分频）来提供的，由掩膜选项来决定。



看门狗定时器是用来防止程序的不正常运行或是跳到未知或不希望去的地址，而导致不可预见的结果。设置掩膜选项可打开/关闭看门狗定时器。如关闭 WDT，所有与 WDT 有关的执行都不会有操作。

一旦选择了内部的 WDT 振荡器（通常是在 3V 电压时周期为 90μs 的 RC 振荡器），先经过 256 分频（8 阶）以获得一个溢出周期为 23ms/3V。这个溢出周期随着 V_{DD}、温度及芯片自身参量的变化而变化。通过使用 WDT 预分频器，可以得到更长的溢出周期。通过写 WS2，WS1，WS0（WDTS 的 2、1、0 位）可以产生不同的溢出时间。如果 WS2，WS1，WS0 都被置 1，则分频的比率是 1：128，得到最大溢出周期 2.9s/3V。如果 WDT 振荡器被关闭，则 WDT 的时钟来源是指令时钟，并且同样的操作，但是在暂停模式下 WDT 将停止计数，从而失去了保护作用。在这种状态下只能通过外部逻辑来复位。WDTS 的高半字节和第三位留给用户使用，以指明某些特定的标志位。

WS2	WS1	WS0	分频率
0	0	0	1：1
0	0	1	1：2
0	1	0	1：4
0	1	1	1：8
1	0	0	1：16
1	0	1	1：32
1	1	0	1：64
1	1	1	1：128

如果系统工作在一个噪声比较大的环境下，建议使用片内的 RC 振荡器（WDT 振荡器）。

在正常运作下，WDT 溢出会产生“芯片复位”（chip reset）并置位 TO 位。在暂停模式下则会产生“热复位”（warm reset），只有 PC 和 SP 被清零。有三种方法可以清除 WDT 的预分频器值：外部复位（低电平输入到$\overline{RES}$端），用软件指令和 HALT 指令三种。有二组软件指令。一组是单条指令 CLR WDT，另一组是二条指令组成 CLR WDT1 及 CLR WDT2 。这两组指令中，只能选取其中一种。选择的方式由掩膜选项中的 CLR WDT 次数选项决定。如果“CLR WDT”被选择（即 CLR WDLT 次数为 1），那么只要执行 CLR WDT 指令就会清除 WDT。在 CLR WDT1 和 CLR WDT2 被选的情况下，（即 CLR WDT 次数为 2），那么要交替执行二条指令才会连续清除 WDT，否则，WDT 会由于超时溢出而使系统复位。

暂停模式 （HALT）

暂停模式是由 HALT 指令来实现的，执行后情况如下：

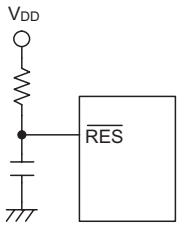
- 关闭系统振荡器和停止 WDT。
- RAM 及寄存器的内容保持不变。
- WDT 预分频器被清除。
- 所有的输入/输出端口都保持其原先状态。
- PD 标志位被置位，TO 标志位被清除。

外部复位、中断、PA 口的下降沿信号或看门狗溢出，可使系统脱离暂停模式。外部复位能使系统初始化，看门狗溢出导致“热复位”（warm reset）。测试 TO 和 PD 状态后，系统复位的原因就可以被确定。PD 标志位是由系统上电复位或执行 CLR WDT 指令被清除，而它的置位是由于执行了 HALT 指令。如 WDT 产生溢出，会使 TO 标志位置位，还能产生唤醒系统，使得程序计数的 PC 和堆栈指针 SP 复位，其他状态都保持不变。

PA 端口的唤醒看作为正常运行的继续。PA 口的每一位都可以通过掩膜选项来单独设定为对系统的唤醒。如果唤醒是来自于输入/输出口的信号变化，程序会继续执行下一条指令。如果是由中断唤醒的，则会执行两个流程。如果中断被禁止了或者堆栈是满的，则程序将继续执行下一条指令。如果中断是允许的并且堆栈没有满，则和平时一样响应中断。如果在进入暂停模式之前一个中断请求标志被置 1，则后面到来的中断将失去唤醒功能。当唤醒事件发生时，要花 1024 个 t_{sys}（系统时钟周期）后，系统重新正常运行。这就是说，在唤醒后将插入了一个等待时间。如果是由中断响应信号唤醒的则中断子程序将被延迟一个或多个周期才被执行。

为了减小功耗，在进入暂停模式之前必须要小心处理输入/输出端口。

复 位（RESET）



有三种方法可以产生复位

- 在正常运行时期由 $\overline{RES}$ 脚产生复位。
- 在 HALT 期间$\overline{RES}$脚产生复位。
- 正常运行时，WDT 溢出复位。

在 HALT 模式下的 WDT 溢出而引起的复位和其他条件下复位不同，产生一个“热复位”，仅仅复位 PC 和 SP，其他电路不变。在其他的复位条件下某些寄存器不发生变化，而大多数寄存器则恢复到“初始状态”。通过测试 PD 标志位和 TO 标志位，能分辨不同的复位原因。

TO	PD	复位条件
0	0	电源上电复位
u	u	正常运作时由$\overline{\text{RES}}$发生复位
0	1	由$\overline{\text{RES}}$唤醒暂停模式
1	u	正常运作时发生看门狗定时器超时
1	1	由看门狗定时器唤醒暂停模式

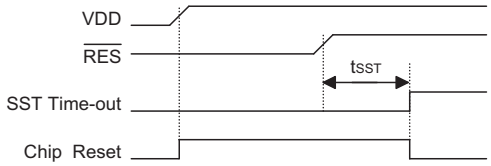
注释：“u”表示未变化。

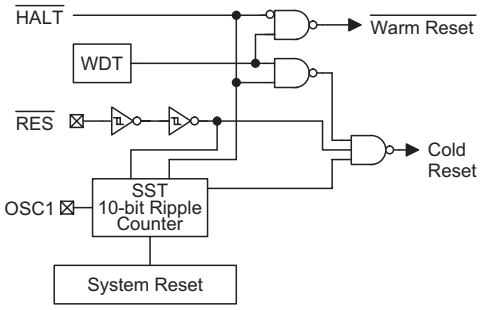
为了保证系统振荡器起振并稳定运行，系统复位(包括上电复位、看门狗定时器溢出或由$\overline{\text{RES}}$端复位)或由暂停状态唤醒时，系统启动定时器(SST)提供了一个额外的延迟时间，共 1024 个系统时钟周期。

系统复位时，SST 会被加在复位延时中。由暂停模式唤醒也会启动 SST 延迟，但是复位来自$\overline{\text{RES}}$引脚时，将没有 SST 延时。

系统复位时各功能单元的状态如下所示：

程序计数器 PC	000H
中断	关闭
预分频器	清除
WDT 预分频器	清除
输入/输出端口	输入模式
堆栈指针 SP	指向堆栈顶部
定时/事件计数器	关闭





特殊功能寄存状态概述表

寄存器	上电复位	正常运行期间		暂停模式	
		WDT 溢出	RES 端复位	RES 端复位	WDT 溢出*
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PC	000H	000H	000H	000H	000H
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu uuuu
STATUS	--00 xxxx	--lu uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC	--00 -000	--00 -000	--00 -000	--00 -000	--uu uuuu
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR1H	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1L	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	00-0 1---	00-0 1---	00-0 1---	00-0 1---	uu-u u---
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	--11 1111	--11 1111	--11 1111	--11 1111	--uu uuuu
PCC	--11 1111	--11 1111	--11 1111	--11 1111	--uu uuuu
PF	---- ---1	---- ---1	---- ---1	---- ---1	---- ---u
PFC	---- ---1	---- ---1	---- ---1	---- ---1	---- ---u
TBHP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu

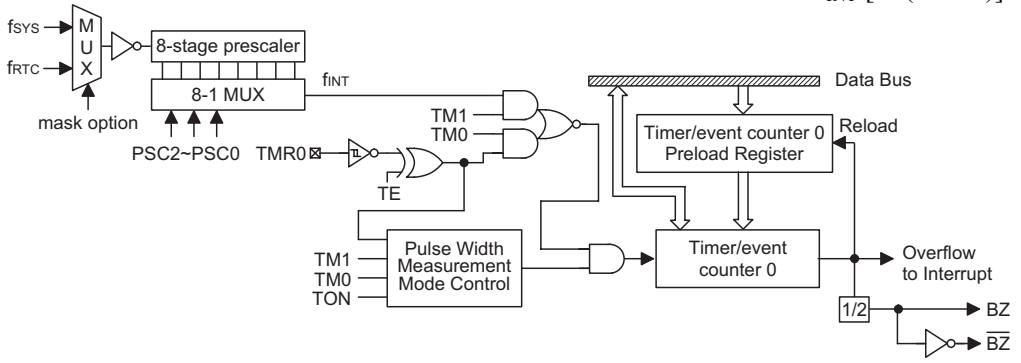
注意：1.“*”表示热复位。
2.“U”表示不变化。
3.“X”表示不确定。

定时/计数器

系统有两个定时/计数器。定时/计数器 0 是一个 8 位可编程递增计数器，它的时钟来源可以是外部信号输入或系统时钟。定时/计数器 1 是一个 16 位可编程递增计数器，时钟来源可以是外部信号输入或系统时钟的四分频。

这两个定时/计数器，若使用外部时钟输入的话，则允许用户对外部时间计数，测量时间、脉宽，或者产生精确的时基；若使用内部时钟来源的话，则允许用户产生精确的时基信号。

只有定时计数器 0 能够产生 PFD 信号，可使用外部或内部时钟。PFD 频率=f_{INT}/[2×(256-N)]。



定时/计数器 0

定时/计数器 0 有 2 个相关的寄存器：TMR0（0DH），TMR0C（0EH）。在定时/计数器 0 的工作状态中（TON=1），写 TMR0 仅仅会把数据写入预置寄存器中，而读 TMR0 则会获得定时/计数器 0 的内容。TMR0C

是定时/计数器 0 的控制寄存器，它定义工作模式、关闭/打开计数、上/下降沿计数。

TM0，TM1 位定义了工作模式。外部事件计数模式用于计数外部事件，时钟来源来自于外部引脚（TMR0）输入。通用内部计时模式，时钟来自于 f_{INT} 时钟。脉宽测量模式可以测量外部信号的高电平或低电平的时间长度（TMR0）。计数则基于 f_{sys} 时钟。

在定时或计数模式中，一旦定时/计数器 0 开始计数，它将从定时/计数器 0 的当前内容开始计数直到 FFH。一旦溢出，定时/计数器从预置寄存器中重载数据，并且同时设置相应的中断请求标志（T0F：INTC 的第 5 位）。

在脉宽测量模式中，TON 和 TE 是相同的，一旦 TMR0 收到一个从上升沿（TE=0 则是下降沿），定时/计数器 0 就开始计数，直到 TMR0 回到原来的状态，并且复位 TON。测量的结果保存在定时/计数器 0，即使再次发生跳变也不改变。也就是说，只可以执行一个测量周期。直到 TON 被置位，再次发生跳变则再次执行测量功能。注意， 在这种模式下，定时/计数器 0 的启动计数不是根据逻辑电平，而是依据信号的边缘跳变触发。一旦发生计数器溢出，计数器会从定时/计数器 0 的预置寄存器重新装入，并引发出中断请求，这种情况与其另外两个模式一样。

符号	位	功能
PSC0~PSC2	0-2	定义预分频器级数，PSC2，PSC1，PSC0= 000：fINT= fsys/2 001：fINT= fsys/4 010：fINT= fsys/8 011：fINT= fsys/16 100：fINT= fsys/32 101：fINT= fsys/64 110：fINT= fsys/128 111：fINT= fsys/256
TE	3	定义定时/计数器 0 的触发方式 （0=上升沿作用，1=下降沿作用）
TON	4	打开/关闭定时/计数器(1=打开，0=关闭)
—	5	未用，读出为 0
TM0 TM1	6 7	定义工作模式 01=外部事件计数模式(外部时钟) 10=定时模式(内部时钟) 11=脉冲宽度测量模式 00 =未用

TMR0C 寄存器

要使得计数运行，只要将定时器启动位（TON；TMR0C 的第 4 位）置 1。在脉宽测量模式中，TON 在测量周期结束后自动被清零。但在另外两个模式中，TON 只能由指令来复位。定时/计数器 0 的溢出是唤醒暂停模式的信号之一。不管任何模式，若写 0 到 ET0I 位即可禁止相应的中断服务。

在定时/计数器 0 为关闭（OFF）的状态下，写数据到定时/计数器 0 的预置寄存器之中，同时也会将数据装入定时/计数器 0 中。但若是定时/计数器 0 已经开启，写到定时/计数器 0 的数据只会被保留在定时/计数器 0 的预置寄存器中，直到定时/计数器 0 发生计数溢出为止，再由预置寄存器加载新的值。

当定时/计数器 0（TMR0）的数据被读取时，会禁止时钟输入以防出错。因为禁止时钟输入可能导致计数错误，所以程序员必须仔细加以考虑。

TMR0C 的 0~2 位被用于定义定时/计数器的内部时钟源的预分频级数。定义如表所示。

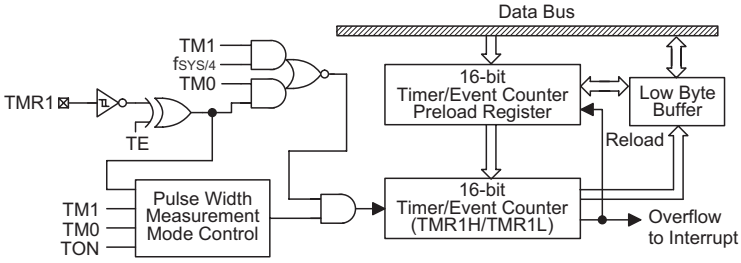
定时/计数器 1 有 3 个相关的寄存器：TMR1H（0FH），TMR1L（10H），TMR1C（11H）。写 TMR1L 把数据放入低位的缓冲区里，而写 TMR1H 则把高位和低位的数据同时放入 TMR1H，TMR1L 的预置寄存器里。预置寄存器因为写 TMR1H 的操作而改变。读 TMR1H 会读出 TMR1H 的值同时把 TMR1L 的值放入低位的缓冲区里。读 TMR1L 的操作则从缓冲区里读出值。TMR1C 是定时/计数器控制寄存器，定义工作的模式，是否允许计数以及在哪个跳变沿计数。

TM0，TM1 定义工作模式。外部事件计数模式用于计数外部事件，意味时钟来源来自于外部引脚（TMR1）。作为普通的定时器的记时功能则时钟来自于指令时钟。脉宽测量模式可以测量外部信号的高电平或低电平的时间长度（TMR1）。计数则基于指令时钟。

在定时或计数模式中，一旦定时/计数器 1 开始计数，它将从定时/计数器 1 的当前内容开始计数直到

FFFFH。一旦溢出，定时/计数器从预置寄存器中重载数据，并且同时设置相应的中断请求标志（T1F：INTC 的第 6 位）。

在脉宽测量模式中，TON 和 TE 是相同的，一旦 TMR1 收到一个上升沿（TE=0 则下降沿），定时/计数器就开始计数，直到 TMR1 回到原来的状态，并且复位 TON。测量的结果保存在定时/计数器 1，即使再次发生跳变也不改变。也就是说，只可以执行一个测量周期。直到 TON 被置位，再次发生跳变则再次执行测量功能。注意，再这种模式下，定时/计数器 1 的启动计数不是根据逻辑电平，而是依据信号的边缘跳变触发。一旦发生计数器溢出，计数器会从定时/计数器 1 的预置寄存器重新装入，并引发出中断请求，这种情况与其另外两个模式一样。



定时/计数器 1

要使得计数运行，只要将定时器启动位（TON；TMR1C 的第 4 位）置 1。在脉宽测量模式中，TON 在测量周期结束后自动被清零。但在另外两个模式中，TON 只能由指令来复位。定时/计数器 1 的溢出是唤醒的信号之一。不管任何模式，若写 0 到 ET1I 位即可禁止相应的中断服务。

符号	位	功能
—	0-2	未用，读出为 0
TE	3	定义定时/计数器 TMR 的触发方式 （0=上升沿作用，1=下降沿作用）
TON	4	打开/关闭定时/计数器(1=打开，0=关闭)
—	5	未用，读出为 0
TM0 TM1	6 7	定义工作模式 01=外部事件计数模式(外部时钟) 10=定时模式(内部时钟) 11=脉冲宽度测量模式 00 =未用

TMR1C 寄存器

在定时/计数器 1 为关闭（OFF）的状态下，写数据到定时/计数器 1 的预置寄存器之中，同时也会将数据装入定时/计数器 1 中。但若是定时/计数器 1 已经开启，写到定时/计数器 1 的数据只会被保留在定时/计数器 1 的预置寄存器中，直到定时/计数器 1 发生计数溢出为止，再由预置寄存器加载新的值。

当定时/计数器 1（TMR1H）的数据被读取时，会禁止时钟输入以防出错。因为禁止时钟输入可能导致计数错误，所以程序员必须仔细加以考虑。

TMR1C 的定义如表所示。

输入/输出端口

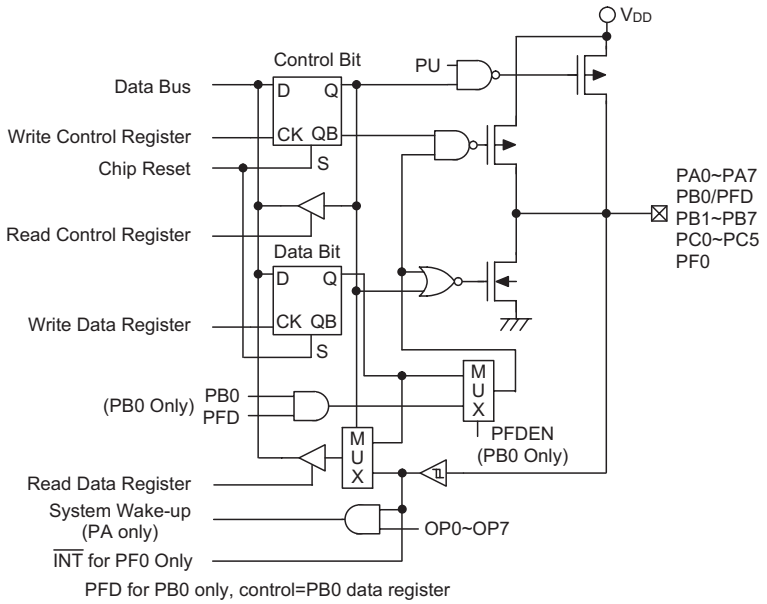
HT48CA3 有 23 个双向输入/输出口，标号为 PA 到 PC 以及 PF。映射到 RAM 的地址为[12H]，[14H]，[16H]以及[1CH]。所有的输入/输出口都可以用作输入输出功能。作输入时，这些输入/输出端口没有锁存的，也就是说，输入信号必须在指令“MOV A，[M]（M=12H，14H，16H，1CH）”的 T2 上升沿被读入。作输出时，所有的输入/输出端口是有锁存的，所有的数据被保存起来一直到下一个写操作。

每一个输入/输出端口都有自己的控制寄存器（PAC，PBC，PCC，PFC）来控制输入输出的设置。通过这些寄存器，用软件指令设置 CMOS 输出或带有/不带有（根据掩膜选项）上拉电阻的施密特触发输入。如果要作为输入，则相关的控制寄存器的位要设置为 1。输入信号来源也取决于控制寄存器，如果控制寄存器的值为“1”，那么读入的是读取引脚状态；如控制寄存器的值为“0”，则读取的是内部锁存器值。后者会在‘读-修改-写’指令中可能发生。作输出时，CMOS 输出是唯一的选择。这些控制寄存器 RAM 地址是地址 13H，15H，17H，1DH。

系统复位之后，这些输入/输出口都会是高电平或浮空状态(由上拉电阻选项决定)。每一个输入/输出锁

存位都能用 SET [m].i 或 CLR [m].i 指令置位或清除（m=12H，14H 或 16H）。

有些指令会先输入数据，然后进行输出操作。例如：“SET [m].i”，“CLR [m].i”，“CPL [m]”，“CPLA[m]”这些指令会先将整个端口状态读入 CPU 中，接着执行所定义的运算(位操作)，然后再将执行的结果写入锁存器或累加器中。



PA 端口的每一个输入/输出口都有唤醒暂停模式的功能。PC 端口的最高两位和 PF 口的七位在物理上并不存在，读回的值是 0，而对它们的写操作则没有任何作用。每一个端口的上拉电阻都可以通过掩膜选项设置。

PB0 和 EXT 信号复用一个管脚，如果选择了 EXT，则输出模式下输出的信号将是 EXT 信号。输入模式总是保持它原来的功能。PF0 和 PC0 与 $\overline{\text{INT}}$ 和 TMR 复用，$\overline{\text{INT}}$ 信号直接连接在 PF0 上。PFD 输出信号（在输出模式下）仅仅由 PB0 数据寄存器控制。下表是 PB0/PFD 的真值表。

PB0 和 PFD 信号复用一个管脚，如果选择了 PFD，则输出模式下输出的信号将是 PFD 信号。输入模式总是保持它原来的功能。PC0 与 TMR 复用，PF0 和 INT 复用。INTB 信号直接连接在 PF0 上。PFD 输出信号（在输出模式下）仅仅由 PB0 数据寄存器控制。下表是 PB0/PFD 的真值表。

PBC(15H)BIT0	I	O	O	O
PB0/PFD 模式	×	PB0	PFD	PFD
PB0 数据	×	D	0	1
PB0 Pad 状态	I	D	0	PFD

注释：I：输入；O：输出；D：数据

BP（BANK POINTER）

由 BANK 指针来控制程序指针 PC 所指向的程序代码应在哪一个 BANK。一个 BANK 为 8K×16 位 ROM 地址空间，当执行 JMP 和 CALL 指令时 BANK 指针的内容会被装入程序计数器。程序计数器是 15 位的寄存器，用来确定要执行的指令的地址。

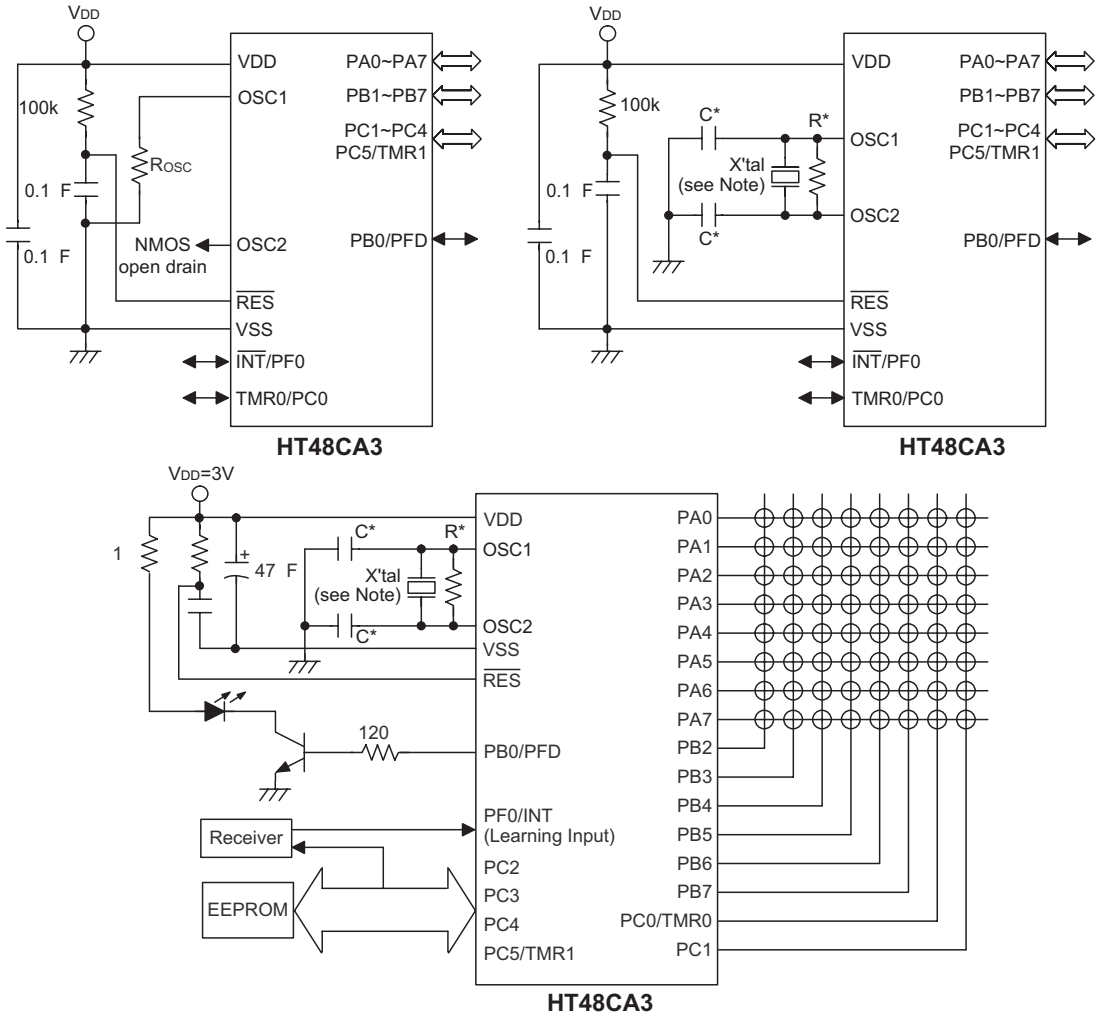
当调用子程序或中断事件发生，程序计数器的内容被保存进堆栈寄存器。如果发生了返回子程序的事件，就从堆栈中恢复程序计数器的内容。

掩膜选项（Mask option）

下面的表格列出了 HT48CA3 掩膜选项，所有这些选项必须被定义来确保系统正确的功能。

功　　能
PA 唤醒功能 有/没有
PC 上拉电阻 有/没有
PA 上拉电阻 有/没有（字节选择）
PF 上拉电阻 有/没有
PB 上拉电阻 有/没有（PB0~PB3，PB4~PB7）：半位选择
PB0/PFD
清除看门狗指令条数：1/2 条
系统振荡器：RC 振荡或晶体振荡
WDT　有/没有
看门狗定时器时钟来源：WDTOSC 或系统时钟 4 分频（T1D）

应用电路图



附注：复位电路所要的电阻和电容器应该这样来设计：在$\overline{RES}$返回高电平以前，V_{DD}稳定地保持在工作电压范围内。

下表所示为根据不同的晶振值选择 R*/C*

晶体振荡或谐振器	C*	R*
4MHz 晶振	0pF	10k Ω
4MHz 谐振器（3 脚）	0pF	12k Ω
4MHz 谐振器（2 脚）	10pF	12k Ω
3.58MHz 晶振	0pF	10k Ω
3.58MHz 谐振器（2 脚）	25pF	10k Ω
2MHz 晶振和谐振器（2 脚）	25pF	10k Ω
1MHz 晶振	35pF	27k Ω
429KHz 谐振器	300pF	10k Ω
455KHz 谐振器	300pF	10k Ω
480KHz 谐振器	300pF	9.1k Ω

指令集摘要

助记符		说明	指令周期	影响标志位
算术运算				
ADD	A,[m]	ACC 与数据存储器相加，结果放入 ACC	1	Z,C,AC,OV
ADDM	A,[m]	ACC 与数据存储器相加，结果放入数据存储器	1(1)	Z,C,AC,OV
ADD	A,x	ACC 与立即数相加，结果放入 ACC	1	Z,C,AC,OV
ADC	A,[m]	ACC 与数据存储器、进位标志相加，结果放入 ACC	1	Z,C,AC,OV
ADCM	A,[m]	ACC 与数据存储器、进位标志相加，结果放入数据存储器	1(1)	Z,C,AC,OV
SUB	A,x	ACC 与立即数相减，结果放入 ACC	1	Z,C,AC,OV
SUB	A,[m]	ACC 与数据存储器相减，结果放入 ACC	1	Z,C,AC,OV
SUBM	A,[m]	ACC 与数据存储器相减，结果放入数据存储器	1(1)	Z,C,AC,OV
SBC	A,[m]	ACC 与数据存储器、进位标志相减，结果放入 ACC	1	Z,C,AC,OV
SBCM	A,[m]	ACC 与数据存储器、进位标志相减，结果放入数据存储器	1(1)	Z,C,AC,OV
DAA	[m]	将加法运算中放入 ACC 的值调整为十进制数，并将结果放入数据存储器	1(1)	C
逻辑运算				
AND	A,[m]	ACC 与数据存储器做“与”运算，结果放入 ACC	1	Z
OR	A,[m]	ACC 与数据存储器做“或”运算，结果放入 ACC	1	Z
XOR	A,[m]	ACC 与数据存储器做“异或”运算，结果放入 ACC	1	Z
ANDM	A,[m]	ACC 与数据存储器做“与”运算，结果放入数据存储器	1(1)	Z
ORM	A,[m]	ACC 与数据存储器做“或”运算，结果放入数据存储器	1(1)	Z
XORM	A,[m]	ACC 与数据存储器做“异或”运算，结果放入数据存储器	1(1)	Z
AND	A,x	ACC 与立即数做“与”运算，结果放入 ACC	1	Z
OR	A,x	ACC 与立即数做“或”运算，结果放入 ACC	1	Z
XOR	A,x	ACC 与立即数做“异或”运算，结果放入 ACC	1	Z
CPL	[m]	对数据存储器取反，结果放入数据存储器	1(1)	Z
CPLA	[m]	对数据存储器取反，结果放入 ACC	1	Z
递增和递减				
INCA	[m]	递增数据存储器，结果放入 ACC	1	Z
INC	[m]	递增数据存储器，结果放入数据存储器	1(1)	Z
DECA	[m]	递减数据存储器，结果放入 ACC	1	Z
DEC	[m]	递减数据存储器，结果放入数据存储器	1(1)	Z
移位				
RRA	[m]	数据存储器右移一位，结果放入 ACC	1	无
RR	[m]	数据存储器右移一位，结果放入数据存储器	1(1)	无
RRCA	[m]	带进位将数据存储器右移一位，结果放入 ACC	1	C
RRC	[m]	带进位将数据存储器右移一位，结果放入数据存储器	1(1)	C
RLA	[m]	数据存储器左移一位，结果放入 ACC	1	无
RL	[m]	数据存储器左移一位，结果放入数据存储器	1(1)	无
RLCA	[m]	带进位将数据存储器左移一位，结果放入 ACC	1	C
RLC	[m]	带进位将数据存储器左移一位，结果放入数据存储器	1(1)	C
数据传送				
MOV	A,[m]	将数据存储器送至 ACC	1	无
MOV	[m],A	将 ACC 送至数据存储器	1(1)	无
MOV	A,x	将立即数送至 ACC	1	无
位运算				
CLR	[m].i	清除数据存储器的位	1(1)	无
SET	[m].i	置位数据存储器的位	1(1)	无

助记符		说明	指令周期	影响标志位
转移				
JMP	addr	无条件跳转	2	无
SZ	[m]	如果数据存储器为零，则跳过下一条指令	1(2)	无
SZA	[m]	数据存储器送至 ACC，如果内容为零，则跳过下一条指令	1(2)	无
SZ	[m].i	如果数据存储器的第 i 位为零，则跳过下一条指令	1(2)	无
SNZ	[m].i	如果数据存储器的第 i 位不为零，则跳过下一条指令	1(2)	无
SIZ	[m]	递增数据存储器，如果结果为零，则跳过下一条指令	1(3)	无
SDZ	[m]	递减数据存储器，如果结果为零，则跳过下一条指令	1(3)	无
SIZA	[m]	递增数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令	1(2)	无
SDZA	[m]	递减数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令	1(2)	无
CALL	addr	子程序调用	2	无
RET		从子程序返回	2	无
RET	A,x	从子程序返回，并将立即数放入 ACC	2	无
RETI		从中断返回	2	无
查表				
TABRDC	[m]	读取当前页的 ROM 内容，并送至数据存储器和 TBLH	2(1)	无
TABRDL	[m]	读取最后页的 ROM 内容，并送至数据存储器和 TBLH	2(1)	无
其它指令				
NOP		空指令	1	无
CLR	[m]	清除数据存储器	1(1)	无
SET	[m]	置位数据存储器	1(1)	无
CLR	WDT	清除看门狗定时器	1	TO,PD
CLR	WDT1	预清除看门狗定时器	1	TO(4),PD(4)
CLR	WDT2	预清除看门狗定时器	1	TO(4),PD(4)
SWAP	[m]	交换数据存储器的高低字节，结果放入数据存储器	1(1)	无
SWAPA	[m]	交换数据存储器的高低字节，结果放入 ACC	1	无
HALT		进入暂停省电模式	1	TO,PD

注： x：立即数
m：数据存储器地址
A：累加器
i：第 0~7 位
addr：程序存储器地址
√：影响标志位
—：不影响标志位
(1)：如果数据是加载到 PCL 寄存器，则指令执行周期会被延长一个指令周期(四个系统时钟)。
(2)：如果满足跳跃条件，则指令执行周期会被延长一个指令周期(四个系统时钟)；否则指令执行周期不会被延长。
(3)：(1)和(2)
(4)：如果执行 CLW WDT1 或 CLR WDT2 指令后，看门狗定时器被清除，则会影响 TO 和 PD 标志位；否则不会影响 TO 和 PD 标志位。

ADC A, [m] 累加器与数据存储器、进位标志相加，结果放入累加器

说明： 本指令把累加器、数据存储器值以及进位标志相加，结果存放到累加器。

运算过程： ACC←ACC+[m]+C

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

ADCM A, [m] 累加器与数据存储器、进位标志相加，结果放入数据存储器

说明： 本指令把累加器、数据存储器值以及进位标志相加，结果存放到存储器。

运算过程： [m]←ACC+[m]+C

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

ADD A, [m] 累加器与数据存储器相加，结果放入累加器

说明： 本指令把累加器、数据存储器值相加，结果存放到累加器。

运算过程： ACC←ACC+[m]

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

ADD A, x 累加器与立即数相加，结果放入累加器

说明： 本指令把累加器值和立即数相加，结果存放到累加器。

运算过程： ACC←ACC+x

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

ADDM A, [m] 累加器与数据存储器相加，结果放入数据存储器

说明： 本指令把累加器、数据存储器值相加，结果放到数据存储器。

运算过程： [m]←ACC+[m]

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

AND A, [m] 累加器与数据存储器做“与”运算，结果放入累加器

说明： 本指令把累加器值、数据存储器值做逻辑与，结果存放到累加器。

运算过程： ACC←ACC “AND” [m]

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

AND A, x 累加器与立即数做“与”运算，结果放入累加器

说明： 本指令把累加器值、立即数做逻辑与，结果存放到累加器。

运算过程： ACC←ACC “AND” x

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

ANDM A, [m] 累加器与数据存储器做“与”运算，结果放入数据存储器

说明： 本指令把累加器值、数据存储器值做逻辑与，结果放到数据存储器。

运算过程： ACC←ACC “AND” [m]

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

CALL addr 子程序调用

说明： 本指令直接调用地址所在处的子程序，此时程序计数器加一，将此程序计数器值存到堆栈寄存器中，再将子程序所在处的地址存放到程序计数器中。

运算过程： Stack←PC+1

PC ← addr

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

CLR [m] 清除数据存储器

说明： 本指令将数据存储器内的数值清零。

运算过程： [m] ←00H

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

CLR [m] . i 将数据存储器的第 i 位清“0”

说明： 本指令将数据存储器内第 i 位值清零。

运算过程： [m].i ←0

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

CLR WDT 清除看门狗定时器

说明： 本指令清除 WDT 计数器(从 0 开始重新计数)，暂停标志位(PD)和看门狗溢出标志位(TO)也被清零。

运算过程： WDT ← 00H

PD&TO ← 0

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	0	0	—	—	—	—

CLR WDT1 预清除看门狗定时器

说明： 必须搭配 CLR WDT2 一起使用，才可清除 WDT 计时器(从 0 开始重新计数)。当程序只执行过该指令，没有执行 CLR WDT2 时，系统只会不会将暂停标志位(PD)和计数溢出位(TO)清零，PD 与 TO 保留原状态不变。

运算过程： WDT ← 00H*

PD&TO ← 0*

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	0*	0*	—	—	—	—

CLR WDT2 预清除看门狗定时器

说明： 必须搭配 CLR WDT1 一起使用，才可清除 WDT 计时器(从 0 开始重新计数)。当程序只执行过该指令，没有执行 CLR WDT1 时，系统只会不会将暂停标志位(PD)和计数溢出位(TO)清零，PD 与 TO 保留原状态不变。

运算过程： WDT ← 00H*

PD&TO ← 0*

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	0*	0*	—	—	—	—

CPL [m] 对数据存储器取反，结果放入数据存储器

说明： 本指令是将数据存储器内保存的数值取反。

运算过程： [m] ← $[\overline{m}]$

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

CPLA [m] 对数据存储器取反，结果放入累加器

说明： 本指令是将数据存储器内保存的值取反后，结果存放在累加器中。

运算过程： ACC ← $[\overline{m}]$

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

DAA [m] 将加法运算后放入累加器的值调整为十进制数，并将结果放入数据存储器

说明 本指令将累加器高低四位分别调整为 BCD 码。如果低四位的值大于“9”或 AC=1，那么 BCD 调整就执行对原值加“6”，并且内部进位标志 AC1=$\overline{AC}$，即 AC 求反；否则原值保持不变。如果高四位的值大于“9”或 C=1，那么 BCD 调整就执行对原值加“6”再加 AC1，并把 C 置位；否则 BCD 调整就执行对原值加 AC1，C 的值保持不变。结果存放到数据存储器中，只有进位标志位(C)受影响。

操作 如果 ACC.3~ACC.0 > 9 或 AC=1

那么 [m].3~[m].0 ←(ACC.3~ACC.0)+6，AC1=$\overline{AC}$

否则 [m].3~[m].0 ←(ACC.3~ACC.0)，AC1=0

并且

如果 ACC.7~ACC.4+AC1 > 9 或 C=1

那么 [m].7~[m].4 ←(ACC.7~ACC.4)+6+ AC1，C=1

否则 [m].7~[m].4 ←(ACC.7~ACC.4)+ AC1，C=C

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	√

DEC [m] 数据存储器的内容减 1，结果放入数据存储器

说明： 本指令将数据存储器内的数值减一再放回数据存储器。

运算过程： [m] ← [m]-1

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

DECA [m] 数据存储器的内容减 1，结果放入累加器

说明： 本指令将存储器内的数值减一,再放到累加器。

运算过程： ACC ← [m]-1

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

HALT 进入暂停模式

说明： 本指令终止程序执行并关掉系统时钟，RAM 和寄存器内的数值保持原状态，WDT 计数器清“0”，暂停标志位(PD)被设为 1， WDT 计数溢出位(TO)被清为 0。

运算过程： PC ← PC+1

PD ← 1

TO ← 0

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	0	1	—	—	—	—

INC [m] 数据存储器的内容加 1，结果放入数据存储器

说明： 本指令将数据存储器内的数值加一,结果放回数据存储器。

运算过程： [m] ← [m]+1

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

INCA [m] 数据存储器的内容加 1，结果放入数据存储器

说明： 本指令是将存储器内的数值加一,结果放到累加器。

运算过程： ACC ← [m]+1

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

JMP addr 无条件跳转

说明： 本指令是将要跳到的目的地直接放到程序计数器内。

运算过程： PC ← addr

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

MOV A, [m] 将数据存储器送至累加器

说明： 本指令是将数据存储器内的数值送到累加器内。

运算过程： ACC ← [m]

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

MOV A, x 将立即数送至累加器

说明： 本指令是将立即数送到累加器内。

运算过程： ACC ← x

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

MOV [m], A 将累加器送至数据存储器

说明： 本指令是将累加器值送到数据存储器内。

运算过程： [m] ← ACC

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

NOP 空指令

说明： 本指令不作任何运算，而只将程序计数器加一。

运算过程： PC ← PC+1

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

OR A, [m] 累加器与数据存储器做“或”运算，结果放入累加器

说明： 本指令是把累加器、数据存储器值做逻辑或，结果放到累加器。

运算过程： ACC←ACC “OR” [m]

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

OR A, x 累加器与立即数做“或”运算，结果放入累加器

说明： 本指令是把累加器值、立即数做逻辑或，结果放到累加器。

运算过程： ACC←ACC “OR” x

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

ORM A, [m] 累加器与数据存储器做“或”运算，结果放入数据存储器

说明： 本指令是把累加器值、存储器值做逻辑或，结果放到数据存储器。

运算过程： ACC←ACC “OR” [m]

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

RET 从子程序返回

说明： 本指令是将堆栈寄存器中的程序计数器值送回程序计数器。

运算过程： PC ← Stack

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

RET A, x 从子程序返回，并将立即数放入累加器

说明： 本指令是将堆栈寄存器中的程序计数器值送回程序计数器，并将立即数送回累加器。

运算过程： PC ← Stack

ACC ← x

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

RETI 从中断返回

说明： 本指令是将堆栈寄存器中的程序计数器值送回程序计数器，与 RET 不同的是它使用在中断程序结束返回时，它还会将中断控制寄存器 INTC 的 0 位(EMI)中断允许位置 1，允许中断服务。

运算过程： PC ← Stack

EMI ← 1

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

RL [m] 数据存储器左移一位，结果放入数据存储器

说明： 本指令是将数据存储器内的数值左移一位，第 7 位移到第 0 位，结果送回数据存储器。

运算过程： [m].0 ←[m].7, [m].(i+1) ←[m].i :i=0~6

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

RLA [m] 数据存储器左移一位，结果放入累加器

说明： 本指令是将存储器内的数值左移一位，第 7 位移到第 0 位，结果送到累加器，而数据存储器内的数值不变。

运算过程： ACC.0 ←[m].7, ACC.(i+1) ←[m].i :i=0~6

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

RLC [m] 带进位将数据存储器左移一位，结果放入数据存储器

说明： 本指令是将存储器内的数值与进位位左移一位，第 7 位取代进位标志，进位标志移到第 0 位，结果送回数据存储器。

运算过程： [m].(i+1) ←[m].i :i=0~6

[m].0 ←C

C ← [m].7

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	√

RLCA [m] 带进位将数据存储器左移一位，结果放入累加器

说明： 本指令是将存储器内的数值与进位位左移一位，第七位取代进位标志，进位标志移到第 0 位，结果送回累加器。

运算过程： ACC.(i+1) ←[m].i :i=0~6

ACC.0 ←C

C ← [m].7

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	√

RR [m] 数据存储器右移一位，结果放入数据存储器

说明： 本指令是将存储器内的数值循环右移，第 0 位移到第 7 位，结果送回数据存储器。

运算过程： [m].7 ←[m].0, [m].i ←[m].(i+1) : i=0~6

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

RRA [m] 数据存储器右移一位，结果放入累加器

说明： 本指令是将数据存储器内的数值循环右移，第 0 位移到第 7 位，结果送回累加器，而数据存储器内的数值不变。

运算过程： ACC.7 ←[m].0, ACC.i ←[m].(i+1) :i=0~6

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

RRC [m] 带进位将数据存储器右移一位，结果放入数据存储器

说明： 本指令是将存储器内的数值加进位位循环右移，第 0 位取代进位标志，进位标志移到第 7 位，结果送回存储器。

运算过程： [m].i ←[m].(i+1) :i=0~6

[m]. 7 ←C

C ← [m].0

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	√

RRCA [m] 带进位将数据存储器右移一位，结果放入累加器

说明： 本指令是将数据存储器内的数值加进位位循环右移，第 0 位取代进位标志，进位标志移到第 7 位，结果送回累加器，数据存储器内的数值不变。

运算过程： ACC.i ←[m].(i+1)：i=0~6

ACC.7 ←C

C ← [m].0

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	√

SBC A,[m] 累加器与数据存储器、进位标志相减，结果放入累加器

说明： 本指令是把累加器值减去数据存储器值以及进位标志的取反，结果放到累加器。

运算过程： ACC←ACC+[$\overline{m}$]+ C

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

SBCM A,[m] 累加器与数据存储器、进位标志相减，结果放入数据存储器

说明： 本指令是把累加器值减去数据存储器值以及进位标志取反，结果放到数据存储器。

运算过程： [m]←ACC+[$\overline{m}$]+C

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

SDZ [m] 数据存储器减 1，如果结果为“0”，则跳过下一条指令

说明： 本指令是把数据存储器内的数值减 1，判断是否为 0，若为 0 则跳过下一条指令，即如果结果为零，放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以取得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。

运算过程： 如果[m]-1=0，跳过下一条指令执行再下一条。

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SDZA [m] 数据存储器减 1，将结果放入累加器，如果结果为“0”，则跳过下一条指令

说明： 本指令是把数据存储器内的数值减 1，判断是否为 0, 为 0 则跳过下一行指令并将减完后数据存储器内的数值送到累加器,而数据存储器内的值不变，即若结果为 0，放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以取得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。

运算过程： 如果[m]-1=0，跳过下一条指令执行再下一条。

ACC ←([m]-1)

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SET [m] 置位数据存储器

说明： 本指令是把存储器内的数值每个位置为 1。

运算过程： [m] ← FFH

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SET [m]. i 将数据存储器的第 i 位置“1”

说明： 本指令是把存储器内的数值的第 i 位置为 1。

运算过程： [m].i ←1

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SIZ [m] 数据存储器加 1，如果结果为“0”，则跳过下一条指令

说明： 本指令是把数据存储器内的数值加 1，判断是否为 0。若为 0，跳过下一条指令，即放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以取得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。

运算过程： 如果 ([m]+1=0)，跳过下一行指令；[m] ←[m]+1

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SIZA 数据存储器加 1，将结果放入累加器，如果结果为“0”，则跳过下一条指令

说明： 本指令是把数据存储器内的数值加 1，判断是否为 0, 若为 0 跳过下一条指令，即放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以取得正确的指令(二个指令周期)，并将加完后存储器内的数值送到累加器，而数据存储器的值保持不变。否则执行下一条指令(一个指令周期)。

运算过程： 如果[m]+1=0，跳过下一行指令；ACC ←([m]+1)

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SNZ [m]. i 如果数据存储器的第 i 位不为“0”，则跳过下一条指令

说明： 本指令是判断数据存储器内的数值的第 i 位, 若不为 0，则程序计数器再加 1, 跳过下一行指令，放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以取得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。

运算过程： 如果 [m].i≠0，跳过下一行指令。

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SUB A, [m] 累加器与数据存储器相减，结果放入累加器

说明： 本指令是把累加器值、数据存储器值相减，结果放到累加器。

运算过程： ACC←ACC+[$\overline{m}$]+1

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

SUB A, x 累加器与立即数相减，结果放入累加器

说明： 本指令是把累加器值、立即数相减，结果放到累加器。

运算过程： ACC←ACC+$\overline{x}$+1

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

SUBM A, [m] 累加器与数据存储器相减，结果放入数据存储器

说明： 本指令是把累加器值、存储器值相减，结果放到存储器。

运算过程： [m]←ACC+[$\overline{m}$]+1

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

SWAP [m] 交换数据存储器的高低字节，结果放入数据存储器

说明： 本指令是将数据存储器的低四位和高四位互换,再将结果送回数据存储器。

运算过程： [m].7~[m].4↔[m].3~[m].0

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SWAPA [m] 交换数据存储器的高低字节，结果放入累加器

说明： 本指令是将数据存储器的低四位和高四位互换，再将结果送回累加器。

运算过程： ACC.3~ACC.0← [m].7~[m].4

ACC.7~ACC.4← [m].3~[m].0

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SZ [m] 如果数据存储器为“0”，则跳过下一条指令

说明： 本指令是判断数据存储器内的数值是否为 0，为 0 则跳过下一行指令，即放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。

运算过程： 如果 [m] = 0, 跳过下一行指令。

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SZA [m] 数据存储器送至累加器，如果内容为“0”，则跳过下一条指令

说明： 本指令是判断存储器内的数值是否为 0，若为 0 则跳过下一行指令，即放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以得正确的指令(二个指令周期)。并把存储器内值送到累加器，而存储器的值保持不变。否则执行下一条指令(一个指令周期)。

运算过程： 如果[m] = 0，跳过下一行指令，并 ACC← [m]。

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SZ [m]. i 如果数据存储器的第 i 位为“0”，则跳过下一条指令

说明： 本指令是判断存储器内第 i 位值是否为 0，若为 0 则跳过下一行指令，即放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。

运算过程： 如果 [m].i = 0，跳过下一行指令。

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

TABRDC [m] 读取 ROM 当前页的内容，并送至数据存储器和 TBLH

说明： 本指令是将表格指针指向程序寄存器当前页，将低位送到存储器，高位直接送到 TBLH 寄存器内。

运算过程： [m] ←程序存储器低四位
TBLH←程序存储器高四位

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

TABRDL [m] 读取 ROM 最后一页的内容，并送至数据存储器和 TBLH

说明： 本指令是将 TABLE 指针指向程序寄存器最后页，将低位送到存储器，高位直接送到 TBLH 寄存器内。

运算过程： [m] ←程序存储器低四位
TBLH←程序存储器高四位

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

XOR A, [m] 累加器与立即数做“异或”运算，结果放入累加器

说明： 本指令是把累加器值、数据存储器值做逻辑异或，结果放到累加器。

运算过程： ACC←ACC “XOR” [m]

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

XORM A, [m] 累加器与数据存储器做“异或”运算，结果放入数据存储器

说明： 本指令是把累加器值、数据存储器值做逻辑异或，结果放到数据存储器。

运算过程： [m]←ACC “XOR” [m]

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

XOR A, x 累加器与数据存储器做“异或”运算，结果放入累加器

说明： 本指令是把累加器值与立即数做逻辑异或，结果放到累加器。

运算过程： ACC←ACC “XOR” x

影响标志位

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

本文译至 July 16, 2003 的 HT48CA3.pdf 文档

盛群半导体股份有限公司（总公司）
台湾新竹市科学工业园区研新二路 3 号
电话: 886-3-563-1999
传真: 886-3-563-1189
网站: www.holtek.com.tw

盛群半导体股份有限公司（业务处）
台湾台北市南港区园区街 3 之 2 号 4 楼之 2
电话: 886-2-2655-7070
传真: 886-2-2655-7373
传真: 886-2-2655-7383 (International sales hotline)

盛扬半导体（上海）有限公司
上海宜山路 889 号 2 号楼 7 楼 200233
电话: 021-6485-5560
传真: 021-6485-0313
网站: www.holtek.com.cn

盛群半导体（香港）有限公司
香港九龙长沙湾道 777-779 号天安工业大厦 3 楼 A 座
电话: 852-2-745-8288
传真: 852-2-742-8657

Holmate Semiconductor, Inc.
46712 Fremont Blvd., Fremont, CA 94538
电话: 510-252-9880
传真: 510-252-9885
网站: www.holmate.com