

盛群知识产权政策

专利权

盛群半导体公司在全球各地区已核准和申请中之专利权至少有 160 件以上，享有绝对之合法权益。与盛群公司 MCU 或其它产品有关的专利权并未被同意授权使用，任何经由不当手段侵害盛群公司专利权之公司、组织或个人，盛群将采取一切可能的法律行动，遏止侵权者不当的侵权行为，并追讨盛群公司因侵权行为所受之损失、或侵权者所得之不法利益。

商标权

盛群之名称和标识、Holtek 标识、HT-IDE、HT-ICE、Marvel Speech、Music Micro、Adlib Micro、Magic Voice、Green Dialer、PagerPro、Q-Voice、Turbo Voice、EasyVoice 和 HandyWriter 都是盛群半导体公司在台湾地区和其它国家的注册商标。

著作权

Copyright © 2009 by HOLTEK SEMICONDUCTOR INC.

规格书中所出现的信息在出版当时相信是正确的，然而盛群对于规格内容的使用不负责任。文中提到的应用其目的仅仅是用来做说明，盛群不保证或不表示这些应用没有更深入的修改就能适用，也不推荐它的产品使用在会由于故障或其它原因可能会对人身造成危害的地方。盛群产品不授权使用于救生、维生器件或系统中做为关键器件。盛群拥有不事先通知而修改产品的权利，对于最新的信息，请参考我们的网址 <http://www.holtek.com.tw>; <http://www.holtek.com.cn>

技术相关信息

- [工具信息](#)
- [FAQs](#)
- [应用范例](#)
 - [HA0003S HT48 & HT46 MCU 与 HT93LC46 的通信](#)
 - [HA0004S HT48 & HT46 MCU UART 的软件实现方法](#)
 - [HA0013S HT48 & HT46 MCU LCM 接口设计](#)
 - [HA0021S HT48 MCU 输入/输出的使用](#)
 - [HA0085S 8 位 Pseudo-Random Number Generator](#)

特性

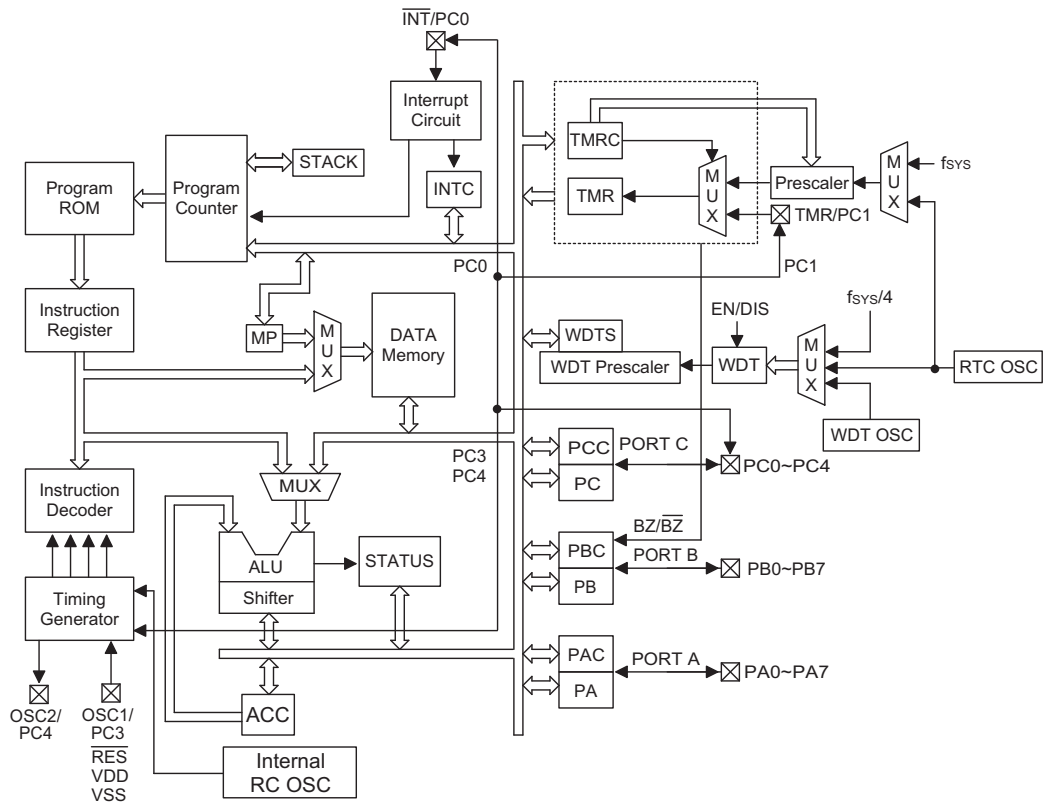
- 工作电压：
 - fsys = 4MHz: 2.2V~5.5V
 - fsys = 8MHz: 3.3V~5.5V
- 低电压复位功能
- 最多可有 21 个双向输入/输出口
- 1 个与输入/输出共用引脚的外部中断输入
- 8 位可编程定时/计数器，具有溢出中断和 8 级预分频器
- 内置晶体和 RC 振荡电路、内置 RC 振荡
- 可接 32768Hz 晶振用于计时
- 看门狗定时器
- 1024×14 程序存储器 ROM
- 64×8 数据存储器 RAM
- 一组蜂鸣器驱动并支持 PFD
- HALT 和唤醒功能可减少功耗
- 在 VDD=5V，系统频率为 8MHz 时，指令周期为 0.5μs
- 所有指令在 1 或 2 个指令周期内完成
- 查表指令，表格内容字长 14 位
- 4 层硬件堆栈
- 位操作指令
- 63 条指令
- 24-pin SKDIP/SOP 封装

概述

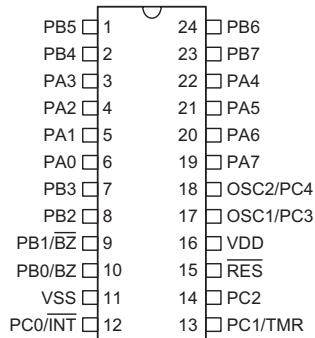
HT48R10A-1/HT48C10 是一款八位高性能精简指令集单片机，专为多输入输出控制的产品设计。掩膜版芯片 HT48C10-1 在引脚和功能方面，都与 OTP 版芯片 HT48R10A-1 完全相同。

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、子系统控制器等。

方框图



引脚图



HT48R10A-1/HT48C10-1
-24 SKDIP-A/SOP-A

引脚说明

引脚名称	输入/输出	掩膜选项	说明
PA0~PA7	输入/输出	上拉电阻* 唤醒功能 CMOS/斯密特触发输入	8 位双向输入/输出口。每一位可由掩膜选项设置为唤醒输入。可由软件设置为 CMOS 输出、带上拉电阻的斯密特触发输入或 CMOS 输入(由上拉电阻选项决定)。
PB0/BZ PB1/ \overline{BZ} PB2~PB7	输入/输出	上拉电阻* 输入/输出或 BZ/ \overline{BZ}	8 位双向输入/输出口。可由软件设置为 CMOS 输出或带上拉电阻的斯密特触发输入(由上拉电阻选项决定)。 PB0、PB1 分别与 BZ、 \overline{BZ} 共用引脚。当 PB0、PB1 设定为蜂鸣器输出时, 输出信号由内部的 PFD 发生器提供(PFD 频率由定时计数器控制)。
VSS	—	—	负电源, 接地。
PC0/ \overline{INT} PC1/TMR PC2	输入/输出	上拉电阻*	双向输入/输出口。可由软件设置为 CMOS 输出或带上拉电阻的斯密特触发输入(由上拉电阻选项决定)。PC0、PC1 分别与外部中断输入、计数器外部输入共用引脚。外部中断输入信号下降沿有效。
\overline{RES}	输入	—	斯密特触发复位输入端, 低电平有效。
VDD	—	—	正电源。
OSC1/PC3 OSC2/PC4	输入 输出	晶体振荡或 RC 振荡或 Int.RC+输入/ 输出或 Int.RC+RTC	由掩膜选项决定 OSC1、OSC2 采用 RC 振荡或晶体振荡提供系统时钟。在 RC 振荡模式下, OSC2 是系统时钟四分频的输出口。这两个口还可做为 RTC 振荡器连接端(32768Hz)或输入/输出口。在这两种模式下, 系统时钟来自内部 RC 振荡, 其频率有四种选择: 3.2MHz, 1.6MHz, 800kHz, 400kHz。如果选择做为输入/输出口, 则带有上拉电阻; 否则 PC3 和 PC4 只能用作内部寄存器(不带上拉电阻)。

“*” 输入/输出(PA、PB、PC)的上拉电阻选择是以整个端口(8 位)为单位的。

极限参数

电源供应电压..... $V_{SS}-0.3V \sim V_{SS}+6.0V$
 端口输入电压..... $V_{SS}-0.3V \sim V_{DD}+0.3V$

储藏温度..... $-50^{\circ}C \sim 125^{\circ}C$
 工作温度..... $-40^{\circ}C \sim 85^{\circ}C$

注: 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°C

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	—	f _{SYS} =4MHz	2.2	—	5.5	V
V _{DD}	工作电压	—	f _{SYS} =8MHz	3.3	—	5.5	V
I _{DD1}	工作电流(晶体振荡)	3V	无负载	—	0.6	1.5	mA
		5V	f _{SYS} =4MHz	—	2	4	mA
I _{DD2}	工作电流(RC 振荡)	3V	无负载	—	0.8	1.5	mA
		5V	f _{SYS} =4MHz	—	2.5	4	mA
I _{DD3}	工作电流(晶体振荡, RC 振荡)	5V	无负载 f _{SYS} =8MHz	—	4	8	mA
I _{STB1}	静态电流(看门狗打开, RTC 关闭)	3V	无负载	—	—	5	μA
		5V	暂停模式	—	—	10	μA
I _{STB2}	静态电流(看门狗关闭, RTC 关闭)	3V	无负载	—	—	1	μA
		5V	暂停模式	—	—	2	μA
I _{STB3}	静态电流(看门狗关闭, RTC 打开)	3V	无负载	—	—	5	μA
		5V	暂停模式	—	—	10	μA
V _{IL1}	输入/输出口的低电平输入电压	—	—	0	—	0.3V _{DD}	V
V _{IH1}	输入/输出口的高电平输入电压	—	—	0.7V _{DD}	—	V _{DD}	V
V _{IL2}	低电平输入电压($\overline{\text{RES}}$)	—	—	0	—	0.4V _{DD}	V
V _{IH2}	高电平输入电压($\overline{\text{RES}}$)	—	—	0.9V _{DD}	—	V _{DD}	V
V _{LVR}	低电压复位	—	LVR 打开	2.7	3.0	3.3	V
I _{OL}	输入/输出灌电流	3V	V _{OL} =0.1V _{DD}	4	8	—	mA
		5V	V _{OL} =0.1V _{DD}	10	20	—	mA
I _{OH}	输入/输出源电流	3V	V _{OH} =0.9V _{DD}	-2	-4	—	mA
		5V	V _{OH} =0.9V _{DD}	-5	-10	—	mA
R _{PH}	上拉电阻	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ

A.C.特性

Ta=25°C

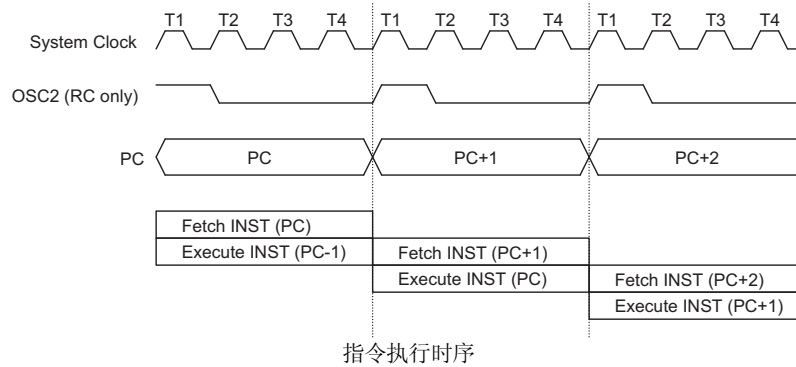
符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
f _{SYS1}	系统时钟(晶体振荡)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	
f _{SYS2}	系统时钟(RC 振荡)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	
f _{SYS3}	系统时钟 (内置 RC 振荡)	5V	3.2MHz	1800	—	5400	kHz
			1.6MHz	900	—	2700	
			800kHz	450	—	1350	
			400kHz	225	—	675	
f _{TIMER}	定时器输入频率(TMR)	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	
t _{WDTOSC}	看门狗振荡器周期	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t _{WDT1}	看门狗溢出周期(WDT 振荡)	3V	WDT 无预分频	11	23	46	ms
		5V		8	17	33	ms
t _{WDT2}	看门狗溢出周期(系统时钟)	—	WDT 无预分频	—	1024	—	t _{sys}
t _{WDT3}	看门狗溢出周期(RTC 振荡)	—	WDT 无预分频	—	7.812	—	ms
t _{RES}	外部复位低电平脉冲宽度	—	—	1	—	—	μs
t _{SST}	系统启动延时周期	—	从 HALT 模式唤醒	—	1024	—	t _{sys}
t _{INT}	中断脉冲宽度	—	—	1	—	—	μs

系统功能说明

指令系统

单片机的系统时钟由晶体振荡器或 RC 振荡器产生。该时钟在芯片内部被分成四个互不重叠的时钟周期。一个指令周期包括四个系统时钟周期。

指令的读取和执行是以流水线方式进行的，这种方式在一个指令周期进行读取指令操作，而在下一个指令周期进行解码与执行该指令。因此，流水线方式使多数指令能在一个周期内执行完成。但如果涉及到的指令要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。



程序计数器 — PC

程序计数器(PC)控制程序存储器 ROM 中指令执行的顺序，它可寻址整个 ROM 的范围。

取得指令码以后，程序计数器会自动加一，指向下一个指令码的地址。但如果执行跳转、条件跳跃、向 PCL 赋值、子程序调用、初始化复位、内部中断、外部中断、子程序返回等操作时，PC 会载入与指令相关的地址而非下一条指令地址。

当遇到条件跳跃指令且符合条件时，当前指令执行过程中读取的下一条指令会被丢弃，取而代之的是一个空指令周期，随后才能取得正确的指令。

程序计数器的低字节(PCL)是一个可读写的寄存器(06H)。对 PCL 赋值将产生一个短跳转动作，跳转的范围为当前页 256 个地址。

当遇到控制转移指令时，系统也会插入一个空指令周期。

模式	程序计数器									
	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
初始化复位	0	0	0	0	0	0	0	0	0	0
外部中断	0	0	0	0	0	0	0	1	0	0
定时/计数器溢出中断	0	0	0	0	0	0	1	0	0	0
条件跳跃	PC+2									
装载 PCL	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
跳转, 子程序调用	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
从子程序返回	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

程序计数器

注: *9 ~ *0 : 程序计数器位
#9 ~ #0 : 指令代码位

S9 ~ S0 : 堆栈寄存器位
@7 ~ @0 : PCL 位

程序存储器 — ROM

程序存储器用来存放要执行的指令代码，以及一些数据、表格和中断入口。程序存储器有 1024×14 位，程序存储器空间可以用程序计数器或表格指针进行寻址。

以下列出的程序存储器地址是系统专为特殊用途而保留的：

- 地址 000H

该地址为程序初始化保留。系统复位后，程序总是从 000H 开始执行。

- 地址 004H

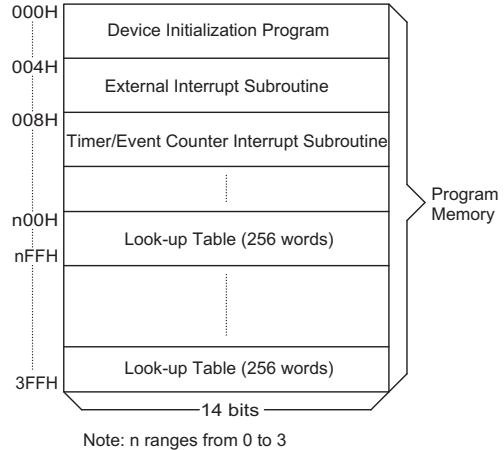
该地址为外部中断服务程序保留。当 $\overline{\text{INT}}$ 引脚有触发信号输入，如果中断允许且堆栈未满，则程序会跳转到 004H 地址开始执行。

- 地址 008H

该地址为定时/计数器中断服务程序保留。当定时/计数器溢出，如果中断允许且堆栈未满，则程序会跳转到 008H 地址开始执行。

- 表格区

ROM 空间的任何地址都可做为查表使用。查表指令“TABRDC [m]” (查当前页表格，1 页=256 个字) 和“TABRDL [m]” (查最后页表格)，会把表格内容低字节传送给[m]，而表格内容高字节传送到 TBLH 寄存器(08H)。只有表格内容的低字节被传送到目标地址中，而高字节被传送到表格内容高字节寄存器 TBLH，并且 TBLH 的高 2 位始终为“0”。表格内容高字节寄存器 TBLH 是只读寄存器。表格指针(TBLP)是可读/写寄存器(07H)，用来指明表格地址。在查表之前，要先将表格地址写入 TBLP 中。如果主程序和中断服务程序(ISR)都用到查表指令，主程序中 TBLH 的值可能会因为 ISR 中执行的查表指令而发生变化，产生错误。也就是说，要避免在主程序和中断服务程序中都使用查表指令。但如果必须这样做的话，我们可以在查表指令前先将中断禁止，在保存了 TBLH 的值后再开放中断以避免发生错误。所有与表格有关的指令都需要两个指令周期的执行时间。这里提到的表格区都可以做为正常的程序存储器来使用。



指令	表格地址									
	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
TABRDC [m]	P9	P8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	@7	@6	@5	@4	@3	@2	@1	@0

表格区

注：*9~*0：表格地址字节

@7~@0：表格指针字节

P9~P8：当前程序指针字节

堆栈寄存器 — STACK

堆栈寄存器是特殊的存储器空间，用来保存 PC 的值。此型号单片机有 4 层堆栈，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。堆栈的使用是通过堆栈指针(SP)来实现的，堆栈指针也不能读出或写入。当发生子程序调用或中断响应时，程序计数器(PC)的值会被压入堆栈；在子程序调用结束或中断响应结束时(执行指令 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中。在系统复位后，堆栈指针会指向堆栈顶部。

如果堆栈已满，并且发生了不可屏蔽的中断，那么只有中断请求标志将会被记录下来，而中断响应会被抑制，一直到堆栈指针(由 RET 或 RETI)发生递减，中断服务才会被响应。这个功能可以防止堆栈溢出，使得程序员易于使用这种结构。同样，如果堆栈已满，并且发生了子程序调用，那么堆栈会发生溢出，首先进入堆栈的内容将会丢失，只有最后的 4 个返回地址会被保留。

数据存储器 — RAM

数据存储器由 81×8 位组成，分为两个功能区间：特殊功能寄存器和通用数据存储区(64×8)，这些空间多数是可读/写的，但也有只读的。

特殊功能寄存器包括间接寻址寄存器(00H)，定时/计数器(TMR; 0DH)，定时/计数器控制寄存器(TMRC; 0EH)，程序计数器低字节寄存器(PCL; 06H)，间接寻址指针寄存器(MP; 01H)，累加器(ACC; 05H)，表格指针(TBLP; 07H)，表格内容高字节寄存器(TBLH; 08H)，状态寄存器(STATUS; 0AH)，中断控制寄存器(INTC; 0BH)，看门狗选项设置寄存器(WDTS; 09H)，输入/输出寄存器(PA; 12H, PB; 14H, PC; 16H)和输入/输出控制寄存器(PAC; 13H, PBC; 15H, PCC; 17H)。其余在 40H 之前的空间保留给系统以后扩展使用，读取这些地址的返回值为“00H”。通用数据寄存器地址从 40H 到 7FH，用来存储数据和控制信息。

所有的数据存储器空间都能直接执行算术、逻辑、递增、递减和循环操作。除了一些特殊位外，数据存储器的每一位都可由“SET[m].i”置位或由“CLR[m].i”复位，而且都可以通过 MP 间接寻址。

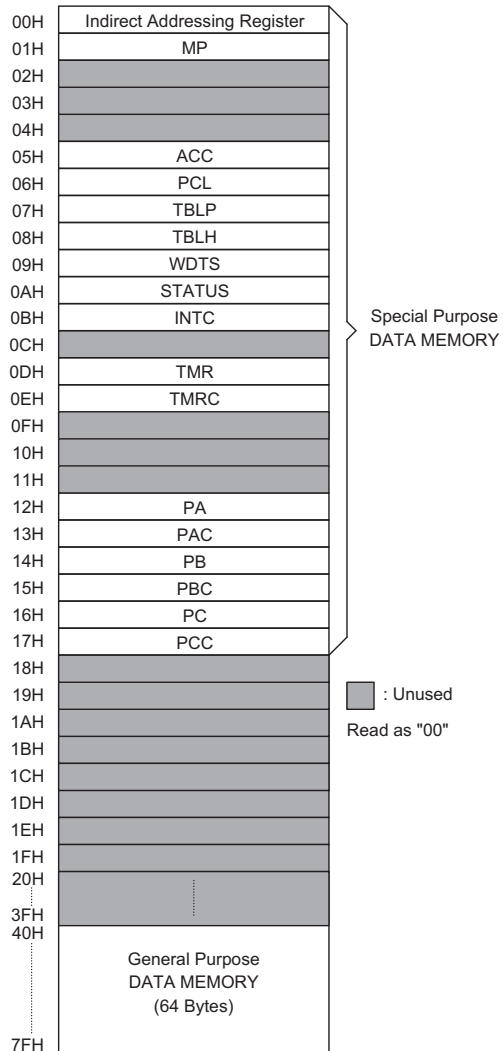
间接寻址寄存器

地址 00H 是一个间接寻址寄存器，并无实际的物理区存在。任何对[00H]的读/写操作，都是访问由 MP(01H)所指向的 RAM 单元。间接读取此地址得到的值为 00H，间接写入此地址，不会产生任何操作。

间接寻址指针 MP(01H)是一个 7 位寄存器。MP 的最高位未定义，读取该位得到的结果是“1”。任何对 MP 的写操作只能将低 7 位数据写入 MP 中。

累加器

累加器(ACC)与算术逻辑单元(ALU)有密切关系。它对应于 RAM 地址 05H，做为运算的立即数据。存储器之间的数据传送必须经过 ACC。



算术逻辑单元 — ALU

算术逻辑单元(ALU)是执行 8 位算术、逻辑运算的电路，它提供有下列的功能：

- 算术运算(ADD, ADC, SUB, SBC, DAA)
- 逻辑运算(AND, OR, XOR, CPL)
- 移位运算(PL, RR, RLC, RRC)
- 递增和递减(INC, DEC)
- 分支判断(SZ, SNZ, SIZ, SDZ...)

ALU 不仅可以储存数据运算的结果，还会改变状态寄存器的值。

状态寄存器 — STATUS

8 位的状态寄存器(0AH)，包含零标志(Z)、进位标志(C)、辅助进位标志(AC)、溢出标志(OV)、暂停标志(PDF)和看门狗定时器溢出标志(TO)。它可以用来记录状态信息和控制操作流程。

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产生借位,那么 C 被置位; 反之, C 被清除。它也可被循环移位指令影响。
AC	1	在加法运算中低 4 位产生了进位或减法运算中低 4 位不产生借位, 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
PDF	4	系统上电或执行了 CLR WDT 指令, PDF 被清除。执行 HALT 指令 PDF 被置位。
TO	5	系统上电或执行了 CLR WDT 指令, 或 HALT 指令, TO 被清除。WDT 定时溢出, TO 被置位。
—	6	未用, 读出为 “0”
—	7	未用, 读出为 “0”

状态寄存器 (0AH)

除了 PDF 和 TO 标志外，状态寄存器的其它位都可以用指令改变。而任何对状态寄存器的写操作都不会改变 PDF 和 TO 的值。对状态寄存器的操作可能会导致与预期不一样的结果。PDF 和 TO 标志只受系统上电、看门狗溢出、清除看门狗指令“CLR WDT”、暂停指令“HALT”的影响。标志位 Z、OV、AC 和 C 反映的是最近一次操作的状态。

在进入中断程序或子程序调用时，状态寄存器不会被自动压入堆栈。如果状态寄存器的内容是重要的，而且子程序会影响状态寄存器的内容，那么程序员必须事先将 STATUS 的值保存好。

中断

HT48R10A-1/HT48C10-1 提供了一个外部中断和一个内部定时器/计数器中断。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。

只要有中断子程序被服务, 其余的中断全部都被自动禁止(通过清除 EMI 位), 这种做法的目的在于防止中断嵌套。这时如果有其它中断发生, 只有中断请求标志会被记录下来。如果在中断服务程序中有另一个中断需要响应, 程序员可以置位 EMI 及 INTC 所对应的位, 以便进行中断嵌套。如果堆栈已满, 则中断并不会被响应, 一直到堆栈指针(SP)发生递减后才会响应。如果需要中断立即得到响应, 应避免堆栈饱和。

所有的外部或内部中断都具有唤醒能力。当有中断被服务, 系统会将程序计数器值压入堆栈, 然后再跳转至中断服务程序的入口。但这时只有程序计数器的内容被压入堆栈, 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。

外部中断是由 $\overline{\text{INT}}$ 脚下降沿信号触发的, 其中断请求标志位(EIF; INTC 的第 4 位)会被置位。如果中断允许, 且堆栈未满, 当发生外部中断时, 会产生地址 04H 的子程序调用; 而中断请求标志 EIF 和 EMI 会被清除, 以禁止其它中断响应。

内部定时/计数器中断是由定时/计数器溢出触发的, 其中断请求标志(TF; INTC 的第 5 位)会被置位。如果中断允许, 且堆栈未满, 当发生定时/计数器中断时, 会产生地址 08H 的子程序调用; 而中断请求标志 TF 和 EMI 会被清除, 以禁止其它中断响应。

在执行中断子程序期间, 其它的中断请求会被屏蔽, 直到执行 RETI 指令或置位 EMI 和相关中断控制位为止(此时堆栈未满)。如果要从中断子程序返回, 只要执行 RET 或 RETI 指令即可。其中, RETI 指令会自动置位 EMI, 以允许中断服务, 而 RET 则不会。

如果中断在两个连续的 T2 脉冲的上升沿之间发生, 且中断响应允许, 那么在下两个 T2 脉冲之间, 该中断会被服务。如果同时发生中断请求, 其优先级如下表示; 也可以通过设定各中断相关的控制位来改变优先级。

编号	中断源	优先级	中断向量
a	外部中断	1	04H
b	定时/计数器中断	2	08H

中断控制寄存器(INTC), 是由外部中断请求标志(EIF)、外部中断允许(EEI)、定时/计数器中断请求标志(TF)、定时/计数器中断允许(ETI)以及总中断允许(EMI)组成的, 其应于数据存储地址 0BH。EMI、EEI、ETI 用来控制中断的允许/禁止状态, 这些控制位可以用来屏蔽正在进行中断服务程序时发生的其它中断请求。一旦中断请求标志(TF、EIF)被置位, 会一直保留在 INTC 寄存器中, 直到中断被响应或用软件指令清除为止。

建议不要在中断服务程序中使用“CALL”指令来调用子程序。因为中断随时都可能发生, 而且需要立刻给予响应。如果只剩下一层堆栈, 而中断不能被很好地控制, 原先的控制序列很可能因为在中断子程序中执行“CALL”指令而使堆栈溢出而发生混乱。

寄存器	位	符号	功能
INTC (0BH)	0	EMI	总中断控制(1=允许; 0=禁止)
	1	EEI	外部中断控制(1=允许; 0=禁止)
	2	ETI	定时/计数器中断控制(1=允许; 0=禁止)
	3	—	未用, 读出为“0”
	4	EIF	外部中断请求标志(1=有; 0=无)
	5	TF	定时/计数器中断请求标志(1=有; 0=无)
	6	—	未用, 读出为“0”
	7	—	未用, 读出为“0”

INTC 寄存器 (0BH)

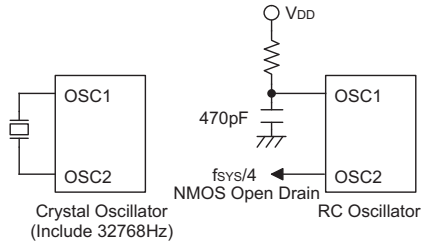
振荡电路

HT48R10A-1/HT48C10-1 有三种振荡方式：外部晶体振荡、外部 RC 振荡和内部 RC 振荡。不管选用哪一种振荡方式，其信号都可以做为系统时钟，可由掩膜选项设置。HALT 模式会停止系统振荡器，并忽视任何外部信号以降低功耗。

如果选用外部 RC 型振荡方式，在 OSC1 和 VDD 之间需要接一个外部电阻，其电阻值可为 24 kΩ 到 1MΩ；而 OSC2 上会输出系统频率的四分频信号，可用于同步外部逻辑。RC 振荡方式是一种低成本方案，但是，RC 振荡频率会随着 VDD、温度和芯片自身参数的漂移而产生误差。因此，在需要精确振荡频率做为计时操作的场合，我们并不建议使用 RC 型振荡器。

如果选用晶体振荡方式，在 OSC1 和 OSC2 之间需要连接一个晶体，用来提供晶体振荡器所需的反馈和相移，除此之外，不再需要其它外部元件。另外，在 OSC1 和 OSC2 之间也可使用谐振器来取代晶体振荡器，但是在 OSC1 和 OSC2 需要多连接两个电容。如果选用内部 RC 振荡方式，OSC1 和 OSC2 可用作通用输入/输出口或做为 32768HZ 的晶体连接口。内部 RC 振荡频率可为 3.2MHz, 1.6 MHz, 800kHz, 400kHz(由掩膜选项设定)。

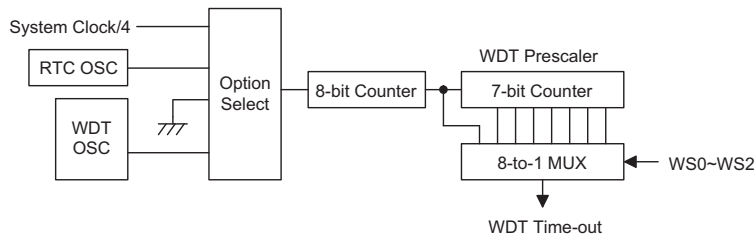
WDT 振荡器是一个内部 RC 振荡器，并不需要连接任何外部元件。当系统进入暂停模式时，系统时钟会停止，但 WDT 振荡器会继续工作(其振荡周期大约为 65μs)。如果要降低功耗，可在掩膜选项中关闭 WDT 振荡器。



看门狗定时器 — WDT

看门狗定时器的时钟来源有三种：看门狗振荡器、RTC 振荡器或指令时钟(系统时钟 4 分频)，由掩膜选项设置。看门狗定时器主要用来防止程序运行故障和程序跳入一死循环而导致不可预测的结果。看门狗定时器由掩膜选项设置为打开或关闭，如果在关闭状态，所有与 WDT 有关指令操作都是没有作用的。RTC 时钟只有在内部 RC+RTC 模式时才能工作。

如果 WDT 时钟源为内部 WDT 振荡器输出(RC 振荡周期一般为 65μs)，它会先经过一个 256(8 级)的分频电路得到大约 17ms/5V 的溢出周期，这个周期会随温度、VDD 和芯片参数的漂移而变化。如果要得到更长的 WDT 溢出周期，可以使用 WDT 预分频器。设置 WS2、WS1、WS0(WDTS 的 2、1、0 位)，可以得到不同的分频级数。如果 WS2、WS1、WS0 的值都为 1，则 WDT 的溢出周期被是原来的 128 倍，得到一个大约 2.2s/5V 的 WDT 溢出周期。WDT 时钟源除了使用内部 WDT 振荡器输出外，还可以使用指令时钟(系统时钟 4 分频)，只是在 HALT 时，WDT 会停止计数而失去保护功能；此时只能靠外部逻辑复位来重新启动系统。WDTS 的 3~7 位是保留给用户来定义自己的状态字。



看门狗定时器

如果系统运用在强干扰的环境中，建议选用内部 WDT 振荡器或 32kHz 的晶体振荡器(RTC 振荡器)，因为 HALT 模式会使系统时钟停止，看门狗也就失去了保护的功能。

在正常运行时，WDT 溢出会使系统复位并置位 TO 标志；但在 HALT 模式下，WDT 溢出只产生一个热复位，只能使程序计数器 PC 和堆栈指针 SP 复位。要清除 WDT 的值(包括 WDT 预分频器)可以有三种方法：外部复位(低电平输入到 RES 端)、清除看门狗指令和 HALT 指令。清除看门狗指令有“CLR WDT”和“CLR WDT1”、“CLR WDT2”二组指令。这两组指令中，只能选取其中一种，由掩膜选项决定。如果选择“CLR WDT”，那么只要执行“CLR WDT”指令就会清除 WDT。如果选择“CLR WDT1”和“CLR WDT2”，那么二条指令要交替使用才会清除 WDT，否则，WDT 会由于溢出而使系统复位。

WS2	WS1	WS0	分频
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

暂停模式 — HALT

暂停模式是由 HALT 指令来实现的，HALT 时系统状态如下：

- 系统振荡器关闭，但 WDT 振荡器会继续运行(如果选择 WDT 振荡器)；
- RAM 和寄存器内容保持不变；
- WDT 被清除并重新开始计数(如果 WDT 时钟来源为 WDT 振荡器)；
- 所有输入/输出口都保持其原有状态；
- 置位 PDF 标志，清除 TO 标志；

以下操作可以使系统离开暂停模式：外部复位、中断、PA 口下降沿信号或看门狗定时器溢出。其中，外部复位会使系统初始化，WDT 溢出则会发生热复位。通过检测 TO 和 PDF 标志，即可了解系统复位的原因。PDF 标志可由系统上电或是执行“CLR WDT”指令清除，由 HALT 指令置位。TO 标志由 WDT 溢出置位，同时产生唤醒，但只有程序计数器 PC 和堆栈指针 SP 被复位，其它都保持其原有的状态。

PA 口唤醒和中断唤醒可做为正常运行的继续。PA 口的每一位都可以由掩膜选项设置为唤醒功能。如果由输入/输出口唤醒，程序会从下一条指令开始运行。但如果是从中断唤醒的话，可能会发生两种情况：如果中断禁止或中断允许但堆栈已满，程序将会从下一条指令重新开始运行；如果中断允许且堆栈未滿，则会产生一般中断响应。如果在进入 HALT 模式之前，中断请求标志位已被置“1”，则中断相关的唤醒功能被禁止。

当发生唤醒，系统必需额外花费 $1024t_{sys}$ (系统时钟周期)的时间，才能重新正常运行，也就是说，唤醒之后会插入一个等待周期。如果唤醒是由中断产生的话，则实际中断子程序的执行会延迟大约一个以上的周期。如果唤醒导致下一条指令执行，那么在等待周期执行完成之后，会立即执行该指令。

为减小功耗，在进入暂停模式之前，应小心处理所有的输入/输出管脚。RTC 振荡器在 HALT 模式下会继续运行(如果选用 RTC 振荡器)。

有关寄存器的状态如下：

寄存器	复位 (上电复位)	WDT 溢出 (正常运行)	RES复位 (正常运行)	RES复位 (暂停模式)	WDT 溢出 (暂停模式)*
TMR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRC	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
Program Counter	000H	000H	000H	000H	000H
MP	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC	--00 -000	--00 -000	--00 -000	--00 -000	--uu -uuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu uuuu
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	---1 1111	---1 1111	---1 1111	---1 1111	---u uuuu
PCC	---1 1111	---1 1111	---1 1111	---1 1111	---u uuuu

注：“*”表示“热复位” “u”表示“不变” “x”表示“未知”

定时/计数器

HT48R10A-1/HT48C10-1 有一个 8 位可编程向上计数的定时/计数器。定时/计数器的时钟来源可以是外部信号输入、系统时钟或 RTC 时钟。

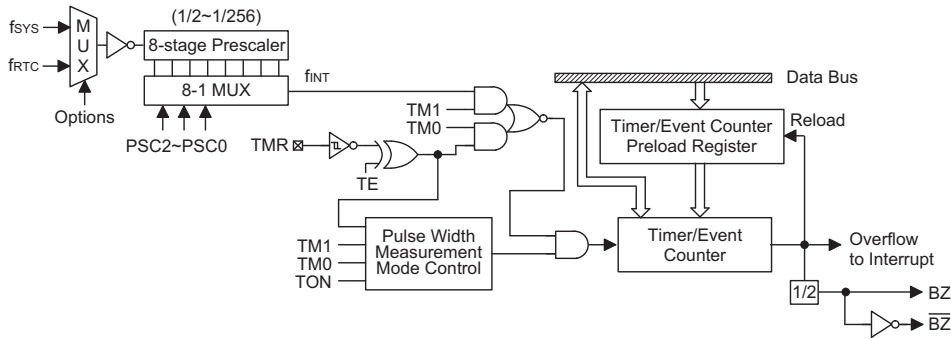
符号	位	功能
PSC0~PSC2	0-2	定义预分频器级数, PSC2、PSC1、PSC0= 000: $f_{INT} = f_{SYS}/2$ 或 $f_{RTC}/2$ 001: $f_{INT} = f_{SYS}/4$ 或 $f_{RTC}/4$ 010: $f_{INT} = f_{SYS}/8$ 或 $f_{RTC}/8$ 011: $f_{INT} = f_{SYS}/16$ 或 $f_{RTC}/16$ 100: $f_{INT} = f_{SYS}/32$ 或 $f_{RTC}/32$ 101: $f_{INT} = f_{SYS}/64$ 或 $f_{RTC}/64$ 110: $f_{INT} = f_{SYS}/128$ 或 $f_{RTC}/128$ 111: $f_{INT} = f_{SYS}/256$ 或 $f_{RTC}/256$
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=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

TMRC 寄存器 (0EH)

当使用内部时钟源时, 定时/计数器有两个时钟源, 可由掩膜时选项设置为系统时钟 f_{SYS} (任何情况下可选)或 RTC 时钟 f_{RTC} (仅在系统振荡为内部 RC+RTC 模式时可选)。外部信号输入可以用来计数外部事件、测量时间间隔、测量脉冲宽度或者产生一个时基信号。内部时钟源输入可以产生一个精确的时基。

定时/计数器可以产生 PFD 信号, PFD 信号频率为: $f_{INT}/[2 \times (256-N)]$ 。

有两个与定时/计数器有关的寄存器, TMR([0DH])和 TMRC([0EH])。TMR 寄存器有两个物理空间; 写入 TMR 会将初始值装入到定时/计数器的预置寄存器中, 而读 TMR 则会取得定时/计数器的内容。TMRC 是定时/计数器控制寄存器, 它可以定义定时/计数器的工作模式。



定时/计数器

TM0、TM1 用来定义定时/计数器的工作模式。外部事件计数模式是用来记录外部事件的，其时钟来源为外部 TMR 引脚输入。定时器模式是一个常用模式，其时钟来源为内部时钟。脉宽测量模式可以测量 TMR 引脚高/低电平的宽度，其时钟来源为内部时钟。

无论是定时模式还是外部事件计数模式，一旦开始计数，定时/计数器会从寄存器当前值向上计到 0FFH。一旦发生溢出，定时/计数器会从预置寄存器中重新加载初值，并开始计数；同时置位中断请求标志 (TF; INTC 的第 5 位)。在脉宽测量模式，TON 与 TE 是 1 时，只要 TMR 引脚有一个上升沿信号 (TE 是 0 为下降沿信号)，定时/计数器就会开始计数，直到 TMR 脚电平恢复，同时 TON 被清零。测量的结果会保存在寄存器中，直到有新的测量开始。换句话说，一次只能测量一个脉冲宽度。重新置位 TON 后，可以继续测量。注意，在该模式下，定时/计数器是跳变触发而不是电平触发。当计数器溢出时，会从定时/计数器的预置寄存器中重新加载初值，而中断的处理方式与其它两种模式一样。要启动计数器，只要置位 TON (TMRC 的第 4 位)。在脉宽测量模式下，TON 在测量结束后会被自动清除；但在另外两种模式中，TON 只能由指令来清除。定时/计数器溢出可以做为唤醒信号。不管是什么模式，只要写 0 到 ETI 位即可禁止定时/计数器中断服务。

在定时/计数器停止计数时，写数据到定时/计数器的预置寄存器中，同时会将该数据写入到定时/计数器。但如果在定时/计数器工作时这么做，数据只能写入到预置寄存器中，直到发生溢出时才会将数据从预置寄存器加载到定时/计数器寄存器。读取定时/计数器时，计数会被停止，以避免发生错误；计数停止会导致计数错误，程序员必须注意到这一点。

TMRC 的第 0~2 位用来定义内部时钟预分频级数，定义如上表所示。定时/计数器的溢出信号可用来产生 PFD 信号以驱动蜂鸣器。

输入/输出口

HT48R10A-1/HT48C10-1 有 21 个双向输入/输出口，记为 PA~PC，分别对应 RAM 地址[12H]、[14H]和[16H]。所有输入/输出口都可以进行输入/输出操作。输入时，口没有锁存功能，输入信号必须在 MOV A, [m](m=12H, 14H 或 16H)指令的 T2 上升沿到来前准备好；输出时，口有锁存功能，口上的数据会保持不变直到执行下一个写入操作。

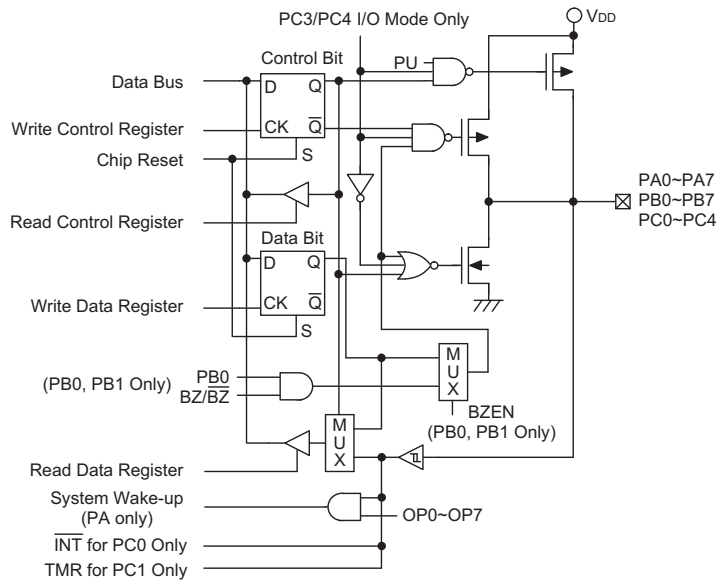
每一个输入/输出口都有一个控制寄存器(PAC, PBC, PCC)，用来控制输入/输出模式。使用控制寄存器，可对 CMOS 输出或斯密特触发输入通过软件动态地进行改变。做为输入时，对应的控制寄存器应设置为“1”。输入信号来源也取决于控制寄存器，如果控制寄存器的值为“1”，那么读入的是读取引脚状态；如控制寄存器的值为“0”，则读取的是内部锁存器值。后者会在‘读-修改-写’指令中可能发生。做为输出时，只能采用 CMOS 输出。控制寄存器对应 RAM 的 13H, 15H, 17H。

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 的每一个口都可具有唤醒功能。PC 的高 3 位没有实际的物理结构。读取这 3 位会返回“0”，而写入则是一个空操作。

所有的输入/输出口都有上拉电阻选项(字节操作)。一旦选择了上拉选项，所有的输入/输出口就都加了上拉电阻。如果不选择上拉电阻，必须注意在输入模式下，若输入/输出口会产生浮空状态。



输入/输出口

PB0、PB1 与 BZ、 $\overline{\text{BZ}}$ 共用引脚。如果选择为 BZ/ $\overline{\text{BZ}}$ 输出，PB0/PB1 在输出模式时的输出将是 PFD 信号，PFD 由定时/计数器的溢出信号产生。在输入模式始终保持其原来的功能。选择 BZ/ $\overline{\text{BZ}}$ 后，蜂鸣器输出信号只受 PB0 数据寄存器控制。PB0/PB1 的输入/输出功能如下所示。

PB0 输入/输出	I	I	I	I	O	O	O	O	O	O	O	O
PB1 输入/输出	I	O	O	O	I	I	I	O	O	O	O	O
PB0 模式	×	×	×	×	C	B	B	C	B	B	B	B
PB1 模式	×	C	B	B	×	×	×	C	C	C	B	B
PB0 数据	×	×	0	1	D	0	1	D0	0	1	0	1
PB1 数据	×	D	×	×	×	×	×	D1	D	D	×	×
PB0 Pad 状态	I	I	I	I	D	0	B	D0	0	B	0	B
PB1 Pad 状态	I	D	0	B	I	I	I	D1	D	D	0	B

注： I: 输入; O: 输出; D, D0, D1 : 数据;

B: 蜂鸣器选项, BZ 或 $\overline{\text{BZ}}$; x: 任意值;

C: CMOS 输出;

PC0、PC1 与 $\overline{\text{INT}}$ 、TMR 共用引脚。

在系统振荡为“内部 RC+输入/输出”时，PC3、PC4 与 OSC1、OSC2 共用引脚。一旦选择“内部 RC+输入/输出”模式后，PC3 和 PC4 可用作通用输入/输出口；否则，PC3 和 PC4 没有上拉电阻和输入输出功能。

我们建议用软件将未使用和没有外接的输入/输出口置为输出模式，以防止这些口在输入浮空时增加系统的功耗。

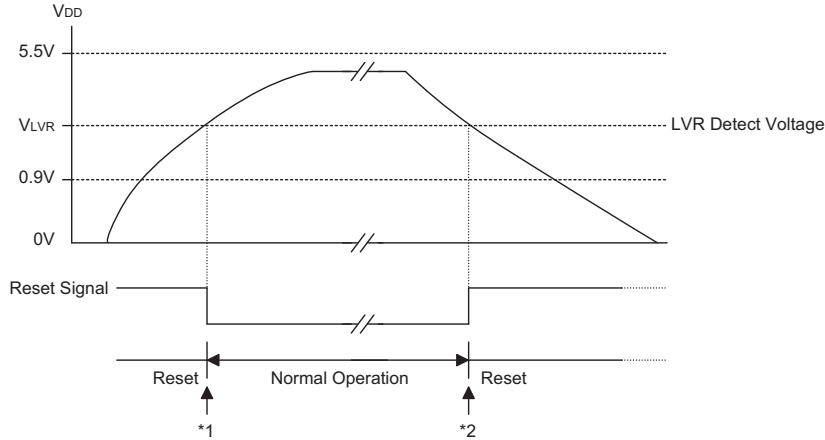
低电压复位 — LVR

为了监控器件的工作电压，HT48R10A-1/HT48C10-1 提供低电压复位功能。如果工作电压在 $0.9V \sim V_{LVR}$ 之间，例如电池电压的变化，那么 LVR 会自动使器件产生内部复位。

LVR 功能说明如下：

- 低电压($0.9V \sim V_{LVR}$)的状态必须持续 1ms 以上。如果低电压的状态没有持续 1ms 以上，那么 LVR 会忽视它而不去执行复位功能。
- LVR 通过与外部 \overline{RES} 信号的“或”的功能来执行系统复位。

V_{DD} 与 V_{LVR} 之间的关系如下所示：



低电压复位

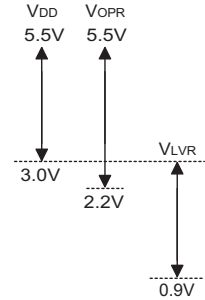
注： *1: 要保证系统振荡器起振并稳定运行，在系统进入正常运行以前，SST 提供额外的 1024 个系统时钟周期的延迟。

*2: 因为低电压状态必须保持 1ms 以上，因此进入复位模式就要有 1ms 的延迟。

掩膜选项

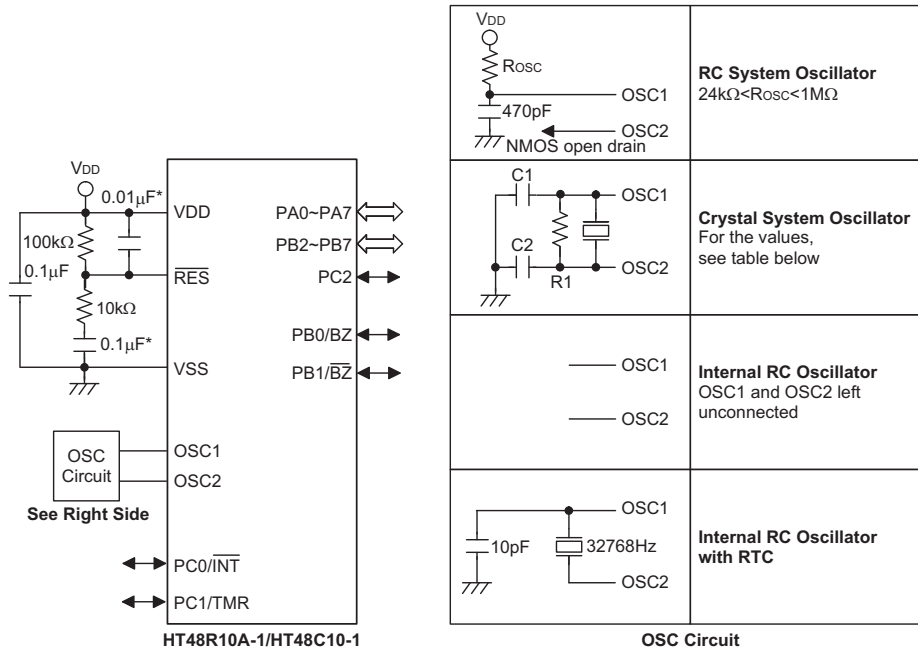
下表标出了所有掩膜选项。在系统运行前必须定义所有选项的值。

项目	选择
1	WDT 时钟源: WDT 振荡或 $f_{SYS}/4$ 或 RTC 振荡或关闭
2	清除看门狗指令条数: 1 或 2 条指令
3	定时/计数器时钟源: f_{SYS} 或 RTC 振荡
4	PA 唤醒功能 (位): 有/没有
5	PA 端口: CMOS/斯密特输入
6	PA、PB、PC 端口上拉电阻: 有/没有
7	BZ/ \overline{BZ} : 打开/关闭
8	LVR: 打开/关闭
9	系统振荡选择: 外部 RC 振荡, 外部晶体振荡, 内部 RC+RTC 或内部 RC+PC3/PC4
10	内部 RC 振荡频率选择: 3.2MHz, 1.6MHz, 800kHz 或 400kHz



注: V_{OPR} 为系统时钟 4MHz 时一般芯片正常工作电压的范围。

应用电路



注意：电阻和电容值选取的原则是使 VDD 保持稳定并在 $\overline{\text{RES}}$ 置为高以前把工作电压保持在允许的范围內。
“*” 为了避免噪声干扰，连接 $\overline{\text{RES}}$ 引脚的线请尽可能地短

下表所示为根据不同的晶振值选择 R1、C1、C2

晶体振荡或谐振器	C1, C2	R1
4MHz 晶振	0pF	10k Ω
4MHz 谐振器	10pF	12k Ω
3.58MHz 晶振	0pF	10k Ω
3.58MHz 谐振器	25pF	10k Ω
2MHz 晶振和谐振器	25pF	10k Ω
1MHz 晶振	35pF	27k Ω
480KHz 谐振器	300pF	9.1k Ω
455KHz 谐振器	300pF	10k Ω
429KHz 谐振器	300pF	10k Ω

电阻 R1 保证了在低电压状态下，晶振被关闭。这里的低电压，是指低于 MCU 正常工作电压范围。请注意，当启动了 LVR 功能，R1 可以不接。

指令集

简介

任何单片机成功运作的核心在于它的指令集，此指令集为一组程序指令码，用来指导单片机如何去执行指定的工作。在盛群单片机中，提供了丰富且灵活的指令，共超过六十条，程序设计者可以事半功倍地实现他们的应用。

为了更加容易理解各种各样的指令码，接下来按功能分组介绍它们。

指令周期

大部分的操作均只需要一个指令周期来执行。分支、调用或查表则需要两个指令周期。一个指令周期相当于四个系统时钟周期，因此如果在 8MHz 的系统时钟振荡器下，大部分的操作将在 0.5 μ s 中执行完成，而分支或调用操作则将在 1 μ s 中执行完成。虽然需要两个指令周期的指令通常指的是 JMP、CALL、RET、RETI 和查表指令，但如果牵涉到程序计数器低字节寄存器 PCL 也将多花费一个周期去加以执行。即指令改变 PCL 的内容进而导致直接跳转至新地址时，需要多一个周期去执行，例如“CLR PCL”或“MOV PCL,A”指令。对于跳转指令必须注意的是，如果比较的结果牵涉到跳转动作将多花费一个周期，如果没有则需一个周期即可。

数据的传送

单片机程序中数据传送是使用最为频繁的操作之一，使用三种 MOV 的指令，数据不但可以从寄存器转移至累加器(反之亦然)，而且能够直接移动立即数到累加器。数据传送最重要的应用之一是从输入端口接收数据或者传送数据到输出端口。

算术运算

算术运算和数据处理是大部分单片机应用所必需具备的能力，在盛群单片机内部的指令集中，可直接实现加与减的运算。当加法的结果超出 255 或减法的结果少于 0 时，要注意正确的处理进位和借位的问题。INC、INCA、DEC 和 DECA 指令提供了对一个指定地址的值加一或减一的功能。

逻辑和移位运算

标准逻辑运算例如 AND、OR、XOR 和 CPL 全都包含在盛群单片机内部的指令集中。大多数牵涉到数据运算的指令，数据的传送必须通过累加器。在所有逻辑数据运算中，如果运算结果为零，则零标志位将被置位，另外逻辑数据运用形式还有移位指令，例如 RR、RL、RRC 和 RLC 提供了向左或向右移动一位的方法。不同的移位指令可满足不同的应用需要。移位指令常用于串行端口的程序应用，数据可从内部寄存器转移至进位标志位，而此位则可被检验，移位运算还可应用在乘法与除法的运算组成中。

分支和控制的转换

程序分支是采取使用 JMP 指令跳转至指定地址或使用 CALL 指令调用子程序的形式，两者之不同在于当子程序被执行完毕后，程序必须马上返回原来的地址。这个动作是由放置在子程序里的返回指令 RET 来实现，它可使程序跳回 CALL 指令之后的地址。在 JMP 指令中，程序则只是跳到一个指定的地址而已，并不需如 CALL 指令般跳回。一个非常有用的分支指令是条件跳转，跳转条件是由数据存储器或指定位来加以决定。遵循跳转条件，程序将继续执行下一条指令或略过且跳转至接下来的指令。这些分支指令是程序走向的关键，跳转条件可能是外部开关输入，或者是内部数据位的值。

位运算

提供数据存储器中单个位的运算指令是盛群单片机的特性之一。这特性对于输出端口位的设置尤其有用，其中个别的位或端口的引脚可以使用“SET [m].i”或“CLR [m].i”指令来设定其为高位或低位。如果没有这特性，程序设计师必须先读入输入输出的 8 位数据，处理这些数据，然后再输出正确的新数据。这种读入-修改-写出的过程现在则被位运算指令所取代。

查表运算

数据的储存通常由寄存器完成，然而当处理大量固定的数据时，它的存储量常常造成对个别存储器的不便。为了改善此问题，盛群单片机允许在程序存储器中建立一个表格作为数据可直接存储的区域，只需要一组简易的指令即可对数据进行查表。

其它运算

除了上述功能指令外，其它指令还包括用于省电的“HALT”指令和使程序在极端电压或电磁环境下仍能正常工作的看门狗定时器控制指令。这些指令的使用则请查阅相关的章节。

指令集概要

惯例

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

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 ^注	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 ^注	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 ^注	Z,C,AC,OV
SBC A,[m]	ACC 与数据存储器、进位标志的反相减, 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 ^注	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 ^注	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 ^注	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 ^注	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 ^注	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 ^注	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 ^注	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 ^注	Z

助记符	说明	指令周期	影响标志位
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 ^注	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 ^注	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 ^注	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 ^注	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 ^注	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 ^注	无
SET [m].i	置位数据存储器的位	1 ^注	无
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 ^注	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 ^注	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 ^注	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 ^注	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 ^注	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 ^注	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 ^注	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 ^注	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 and TBLH	2 ^注	无
TABRDL [m]	读取最后页的 ROM 内容, 并送至数据存储器 and TBLH	2 ^注	无
其它指令			
NOP	空指令	1	无
CLR [m]	清除数据存储器	1 ^注	无
SET [m]	置位数据存储器	1 ^注	无
CLR WDT	清除看门狗定时器	1	TO,PDF
CLR WDT1	预清除看门狗定时器	1	TO,PDF
CLR WDT2	预清除看门狗定时器	1	TO,PDF
SWAP [m]	交换数据存储器的高低字节, 结果放入数据存储器	1 ^注	无
SWAPA [m]	交换数据存储器的高低字节, 结果放入 ACC	1	无
HALT	进入暂停模式	1	TO,PDF

- 注： 1、对跳转指令而言，如果比较的结果牵涉到跳转即需 2 个周期，如果没有发生跳转，则只需一个周期。
- 2、任何指令若要改变 PCL 的内容将需要 2 个周期来执行。
- 3、对于“CLR WDT1”或“CLR WDT2”指令而言，TO 和 PDF 标志位也许会受执行结果影响，“CLR WDT1”和 “CLR WDT2”被连续地执行后，TO 和 PDF 标志位会被清除，除此之外 TO 和 PDF 标志位保持不变。

指令定义

ADC A, [m]	Add data memory and carry to the accumulator
说明:	将指定的数据存储器、累加器内容以及进位标志相加，结果存放到累加器。
运算过程:	$ACC \leftarrow ACC + [m] + C$
影响标志位:	OV、Z、AC、C
ADCM A, [m]	Add the accumulator and carry to the accumulator
说明:	将指定的数据存储器、累加器内容和进位标志位相加，结果存放到指定的数据存储器。
运算过程:	$[m] \leftarrow ACC + [m] + C$
影响标志位:	OV、Z、AC、C
ADD A, [m]	Add data memory to the accumulator
说明:	将指定的数据存储器和累加器内容相加，结果存放到累加器。
运算过程:	$ACC \leftarrow ACC + [m]$
影响标志位:	OV、Z、AC、C
ADD A, x	Add immediate data to the accumulator
说明:	将累加器和立即数相加，结果存放到累加器。
运算过程:	$ACC \leftarrow ACC + x$
影响标志位:	OV、Z、AC、C
ADDM A, [m]	Add the accumulator to the data memory
说明:	将指定的数据存储器和累加器内容相加，结果存放到指定的数据存储器。
运算过程:	$[m] \leftarrow ACC + [m]$
影响标志位:	OV、Z、AC、C
AND A, [m]	Logical AND accumulator with data memory
说明:	将累加器中的数据和指定数据存储器内容做逻辑与，结果存放到累加器。
运算过程:	$ACC \leftarrow ACC \text{ "AND" } [m]$
影响标志位:	Z
AND A, x	Logical AND immediate data to the accumulator
说明:	将累加器中的数据和立即数做逻辑与，结果存放到累加器。
运算过程:	$ACC \leftarrow ACC \text{ "AND" } x$
影响标志位:	Z

ANDM	A, [m]	Logical AND data memory with the accumulator
说明:		将指定数据存储器和累加器中的数据做逻辑与, 结果存放到数据存储器。
运算过程:		$[m] \leftarrow ACC \text{ "AND" } [m]$
影响标志位:		Z
CALL	addr	Subroutine call
说明:		无条件的调用指定地址的子程序, 此时程序计数器先加 1 获得下一个要执行的指令地址并压入堆栈, 接着载入指定地址并从新地址执行程序。由于指令需要额外的运算, 所以此指令为 2 个周期。
运算过程:		$Stack \leftarrow Program\ Counter + 1$ $Program\ Counter \leftarrow addr$
影响标志位:		无
CLR [m]		Clear data memory
说明:		将指定数据存储器的内容清零。
运算过程:		$[m] \leftarrow 00H$
影响标志位:		无
CLR [m].i		Clear bit of data memory
说明:		将指定数据存储器的 i 位内容清零。
运算过程:		$[m].i \leftarrow 0$
影响标志位:		无
CLR WDT		Clear Watchdog Timer
说明:		WDT 计数器、暂停标志位 PDF 和看门狗溢出标志位 TO 清零。
运算过程:		$WDT \leftarrow 00H$ $PDF \ \& \ TO \leftarrow 0$
影响标志位:		TO、PDF
CLR WDT1		Preclear Watchdog Timer
说明:		PDF 和 TO 标志位都被清 0。必须配合 CLR WDT2 一起使用清除 WDT 计时器。当程序仅执行 CLR WDT1, 而没有执行 CLR WDT2 时, PDF 与 TO 保留原状态不变。
运算过程:		$WDT \leftarrow 00H$ $PDF \ \& \ TO \leftarrow 0$
影响标志位:		TO、PDF

CLR WDT2	Preclear Watchdog Timer
说明:	PDF 和 TO 标志位都被清 0。必须配合 CLR WDT1 一起使用清除 WDT 计时器。当程序仅执行 CLR WDT2, 而没有执行 CLR WDT1 时, PDF 与 TO 保留原状态不变。
运算过程:	WDT \leftarrow 00H PDF & TO \leftarrow 0
影响标志位:	TO、PDF
CPL [m]	Complement data memory
说明:	将指定数据存储器中的每一位取逻辑反, 相当于从 1 变 0 或从 0 变 1。
运算过程:	$[m] \leftarrow \overline{[m]}$
影响标志位:	Z
CPLA [m]	Complement data memory
说明:	将指定数据存储器中的每一位取逻辑反, 相当于从 1 变 0 或从 0 变 1, 结果被存放回累加器且数据寄存器的内容保持不变。
运算过程:	ACC $\leftarrow \overline{[m]}$
影响标志位:	Z
DAA [m]	Decimal-Adjust accumulator for addition
说明:	将累加器中的内容转换为 BCD (二进制转成十进制) 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”, 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”。BCD 转换实质上是根据累加器和标志位执行 00H, 06H, 60H 或 66H 的加法运算, 结果存放和数据存储器。只有进位标志位 C 受影响, 用来指示原始 BCD 的和是否大于 100, 并可以进行双精度十进制数的加法运算。
操作:	$[m] \leftarrow \text{ACC}+00\text{H}$ 或 $[m] \leftarrow \text{ACC}+06\text{H}$ $[m] \leftarrow \text{ACC}+60\text{H}$ 或 $[m] \leftarrow \text{ACC}+66\text{H}$
影响标志位:	C
DEC [m]	Decrement data memory
说明:	将指定数据存储器的内容减 1。
运算过程:	$[m] \leftarrow [m]-1$
影响标志位:	Z
DECA [m]	Decrement data memory and place result in the accumulator
说明:	将指定数据存储器的内容减 1, 把结果存放回累加器并保持指定数据存储器的内容不变。
运算过程:	ACC $\leftarrow [m]-1$
影响标志位:	Z

HALT	Enter power down mode
说明:	此指令终止程序执行并关掉系统时钟, RAM 和寄存器的内容保持原状态, WDT 计数器和分频器被清“0”, 暂停标志位 PDF 被置位 1, WDT 溢出标志位 TO 被清 0。
运算过程:	PDF \leftarrow 1 TO \leftarrow 0
影响标志位:	TO、PDF
INC [m]	Increment data memory
说明:	将指定数据存储器的内容加 1。
运算过程:	[m] \leftarrow [m]+1
影响标志位:	Z
INCA [m]	Increment data memory and place result in the accumulator
说明:	将指定数据存储器的内容加 1, 结果存放回累加器并保持指定的数据存储器内容不变。
运算过程:	ACC \leftarrow [m]+1
影响标志位:	Z
JMP addr	Directly jump
说明:	程序计数器的内容无条件地由被指定的地址取代, 程序由新的地址继续执行。当新的地址被加载时, 必须插入一个空指令周期, 所以此指令为 2 个周期的指令。
运算过程:	PC \leftarrow addr
影响标志位:	无
MOVA, [m]	Move data memory to the accumulator
说明:	将指定数据存储器的内容复制到累加器。
运算过程:	ACC \leftarrow [m]
影响标志位:	无
MOVA, x	Move immediate data to the accumulator
说明:	将 8 位立即数载入累加器。
运算过程:	ACC \leftarrow x
影响标志位:	无
MOV [m], A	Move the accumulator data to memory
说明:	将累加器的内容复制到指定的数据存储器。
运算过程:	[m] \leftarrow ACC
影响标志位:	无

NOP	No operation
说明:	空操作, 顺序执行下一条指令。
运算过程:	$PC \leftarrow PC+1$
影响标志位:	无
OR A, [m]	Logical OR accumulator with data memory
说明:	将累加器中的数据和指定的数据存储器内容逻辑或, 结果存放到累加器。
运算过程:	$ACC \leftarrow ACC \text{ "OR" } [m]$
影响标志位:	Z
OR A, x	Logical OR immediate data to the accumulator
说明:	将累加器中的数据和立即数逻辑或, 结果存放到累加器。
运算过程:	$ACC \leftarrow ACC \text{ "OR" } x$
影响标志位:	Z
ORM A, [m]	Logical OR data memory with accumulator
说明:	将存在指定数据存储器中的数据和累加器逻辑或, 结果放到数据存储器。
运算过程:	$[m] \leftarrow ACC \text{ "OR" } [m]$
影响标志位:	Z
RET	Return from subroutine
说明:	将堆栈寄存器中的程序计数器值恢复, 程序由取回的地址继续执行。
运算过程:	$PC \leftarrow Stack$
影响标志位:	无
RET A, x	Return and place immediate data in the accumulator
说明:	将堆栈寄存器中的程序计数器值恢复且累加器载入指定的立即数, 程序由取回的地址继续执行。
运算过程:	$PC \leftarrow Stack$ $ACC \leftarrow x$
影响标志位:	无
RETI	Return from interrupt
说明:	将堆栈寄存器中的程序计数器值恢复且中断功能通过设置 EMI 位重新使能。EMI 是控制中断使能的主控制位。如果在执行 RETI 指令之前还有中断未被相应, 则这个中断将在返回主程序之前被相应。
运算过程:	$PC \leftarrow Stack$ $EMI \leftarrow 1$
影响标志位:	无

RL [m]	Rotate data memory left
说明:	将指定数据存储器的内容左移 1 位, 且第 7 位移到第 0 位。
运算过程:	$[m].(i+1) \leftarrow [m].i \quad (i=0\sim6)$ $[m].0 \leftarrow [m].7$
影响标志位:	无
RLA [m]	Rotate data memory left and place result in the accumulator
说明:	将指定数据存储器的内容左移 1 位, 且第 7 位移到第 0 位, 结果送到累加器, 而指定数据存储器的内容保持不变。
运算过程:	$ACC.(i+1) \leftarrow [m].i \quad (i=0\sim6)$ $ACC.0 \leftarrow [m].7$
影响标志位:	无
RLC [m]	Rotate data memory left through carry
说明:	将指定数据存储器的内容连同进位标志左移 1 位, 第 7 位取代进位标志且原本的进位标志移到第 0 位。
运算过程:	$[m].(i+1) \leftarrow [m].i \quad (i=0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
影响标志位:	C
RLCA [m]	Rotate left through carry and place result in the accumulator
说明:	将指定数据存储器的内容连同进位标志左移 1 位, 第 7 位取代进位标志且原本的进位标志移到第 0 位, 移位结果送回累加器, 但是指定数据寄存器的内容保持不变。
运算过程:	$ACC.(i+1) \leftarrow [m].i \quad (i=0\sim6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
影响标志位:	C
RR [m]	Rotate data memory right
说明:	将指定数据存储器的内容循环右移 1 位且第 0 位移到第 7 位。
运算过程:	$[m].i \leftarrow [m].(i+1) \quad (i=0\sim6)$ $[m].7 \leftarrow [m].0,$
影响标志位:	无

RRA [m]	Rotate right and place result in the accumulator
说明:	将指定数据存储器的内容循环右移 1 位, 第 0 位移到第 7 位, 移位结果存放累加器, 而指定数据存储器的内容保持不变。
运算过程:	$ACC.i \leftarrow [m].(i+1) \quad (i=0\sim6)$ $ACC.7 \leftarrow [m].0$
影响标志位:	无
RRC [m]	Rotate data memory right through carry
说明:	将指定数据存储器的内容连同进位标志右移 1 位, 第 0 位取代进位标志且原本的进位标志移到第 7 位。
运算过程:	$[m].i \leftarrow [m].(i+1) \quad (i=0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位:	C
RRCA [m]	Rotate right through carry and place result in the accumulator
说明:	将指定数据存储器的内容连同进位标志右移 1 位, 第 0 位取代进位标志且原本的进位标志移到第 7 位, 移位结果送回累加器, 但是指定数据寄存器的内容保持不变。
运算过程:	$ACC.i \leftarrow [m].(i+1) \quad (i=0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位:	C
SBC A,[m]	Subtract data memory and carry from the accumulator
说明:	将累加器减去指定数据存储器的内容以及进位标志的反, 结果存放累加器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。
运算过程:	$ACC \leftarrow ACC - [m] - \bar{C}$
影响标志位:	OV、Z、AC、C

SBCMA,[m] Subtract data memory and carry from the accumulator

说明: 将累加器减去指定数据存储器的内容以及进位标志的反, 结果存放到数据存储器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。

运算过程: $ACC \leftarrow ACC - [m] - \overline{C}$

影响标志位: OV、Z、AC、C

SDZ [m] Skip if decrement data memory is 0

说明: 将指定的数据存储器的内容减 1, 判断是否为 0, 若为 0 则跳过下一条指令, 由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。

运算过程: $[m] \leftarrow [m] - 1$, 如果 $[m]=0$ 跳过下一条指令执行

影响标志位: 无

SDZA [m] Decrement data memory and place result in ACC, skip if 0

说明: 将指定数据存储器内容减 1, 判断是否为 0, 如果为 0 则跳过下一条指令, 此结果将存放到累加器, 但指定数据存储器内容不变。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。

运算过程: $ACC \leftarrow [m] - 1$, 如果 $ACC=0$ 跳过下一条指令执行

影响标志位: 无

SET [m] Set data memory

说明: 将指定数据存储器的每一位设置为 1。

运算过程: $[m] \leftarrow FFH$

影响标志位: 无

SET [m].i Set bit of data memory

说明: 将指定数据存储器的第 i 位设置为 1。

运算过程: $[m].i \leftarrow 1$

影响标志位: 无

SIZ [m] Skip if increment data memory is 0

说明: 将指定的数据存储器的内容加 1, 判断是否为 0, 若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。

运算过程: $[m] \leftarrow [m] + 1$, 如果 $[m]=0$ 跳过下一条指令执行

影响标志位: 无

SIZA [m]	Increment data memory and place result in ACC, skip if 0
说明:	将指定数据存储器的内容加 1, 判断是否为 0, 如果为 0 则跳过下一条指令, 此结果会被存放到累加器, 但是指定数据存储器的内容不变。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。
运算过程:	$ACC \leftarrow [m] + 1$, 如果 $ACC = 0$ 跳过下一条指令执行
影响标志位:	无
SNZ [m].i	Skip if bit I of the data memory is not 0
说明:	判断指定数据存储器的第 i 位, 若不为 0, 则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果为 0, 则程序继续执行下一条指令。
运算过程:	如果 $[m].i \neq 0$, 跳过下一条指令执行
影响标志位:	无
SUB A, [m]	Subtract data memory from the accumulator
说明:	将累加器的内容减去指定的数据存储器的数据, 把结果存放到累加器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。
运算过程:	$ACC \leftarrow ACC - [m]$
影响标志位:	OV、Z、AC、C
SUBM A, [m]	Subtract data memory from the accumulator
说明:	将累加器的内容减去指定数据存储器的数据, 结果存放到指定的数据存储器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。
运算过程:	$[m] \leftarrow ACC - [m]$
影响标志位:	OV、Z、AC、C
SUB A, x	Subtract immediate data from the accumulator
说明:	将累加器的内容减去立即数, 结果存放到累加器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。
运算过程:	$ACC \leftarrow ACC - x$
影响标志位:	OV、Z、AC、C
SWAP [m]	Swap nibbles within the data memory
说明:	将指定数据存储器的低 4 位和高 4 位互相交换。
运算过程:	$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
影响标志位:	无

SWAPA	[m]	Swap data memory and place result in the accumulator
说明:		将指定数据存储器的低 4 位和高 4 位互相交换, 再将结果存放到累加器且指定数据寄存器的数据保持不变。
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:		无
SZ	[m]	Skip if data memory is 0
说明:		判断指定数据存储器的内容是否为 0, 若为 0, 则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。
运算过程:		如果[m]=0, 跳过下一条指令执行
影响标志位:		无
SZA	[m]	Move data memory to ACC, skip if 0
说明:		将指定存储器内容复制到累加器, 并判断指定数据存储器的内容是否为 0, 若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。
运算过程:		ACC←[m], 如果[m]=0, 跳过下一条指令执行
影响标志位:		无
SZ	[m].i	Skip if bit I of the data memory is 0
说明:		判断指定数据存储器的第 i 位是否为 0, 若为 0, 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。
运算过程:		如果[m].i=0, 跳过下一条指令执行
影响标志位:		无
TABRDC	[m]	Move the ROM code(current page) to TBLH and data memory
说明:		将表格指针 TBLP 所指的程序代码低字节(当前页)移至指定的存储器且将高字节移至 TBLH。
运算过程:		[m] ←程序代码(低字节) TBLH←程序代码(高字节)
影响标志位:		无
TABRDL	[m]	Move the ROM code(last page) to TBLH and data memory
说明:		将表格指针 TBLP 所指的程序代码低字节(最后一页)移至指定的存储器且将高字节移至 TBLH。
运算过程:		[m] ←程序代码(低字节) TBLH←程序代码(高字节)
影响标志位:		无

XOR A, [m] Logical XOR accumulator with data memory

说明: 将累加器的数据和指定的数据存储器内容逻辑异或, 结果存放到累加器。

运算过程: $ACC \leftarrow ACC \text{ "XOR" } [m]$

影响标志位: Z

XORMA, [m] Logical XOR data memory with accumulator

说明: 将累加器的数据和指定的数据存储器内容逻辑异或, 结果放到数据存储器。

运算过程: $[m] \leftarrow ACC \text{ "XOR" } [m]$

影响标志位: Z

XOR A, x Logical XOR immediate data to the accumulator

说明: 将累加器的数据与立即数逻辑异或, 结果存放到累加器。

运算过程: $ACC \leftarrow ACC \text{ "XOR" } x$

影响标志位: Z

封装信息

24-pin SKDIP (300mil)外形尺寸

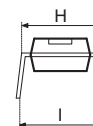
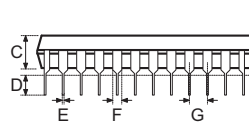
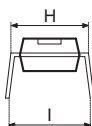
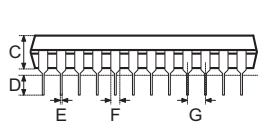
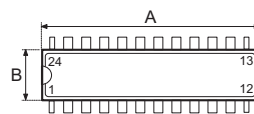
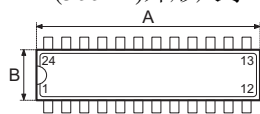


Fig1. Full Lead Packages

Fig2. 1/2 Lead Packages

•MS_001d (Fig1)

标号	尺寸(mil)		
	最小	典型	最大
A	1230	—	1280
B	240	—	280
C	115	—	195
D	115	—	150
E	14	—	22
F	45	—	70
G	—	100	—
H	300	—	325
I	—	—	430

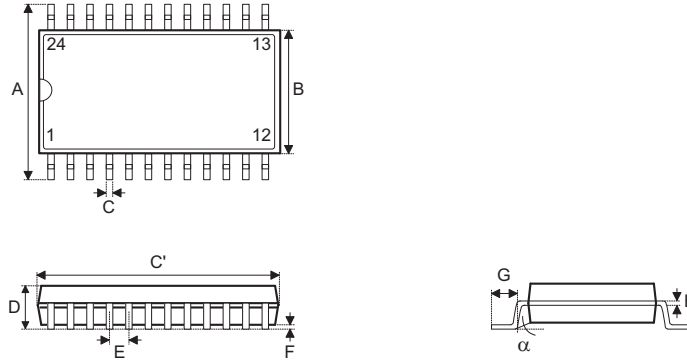
•MS_001d (Fig2)

标号	尺寸(mil)		
	最小	典型	最大
A	1160	—	1195
B	240	—	280
C	115	—	195
D	115	—	150
E	14	—	22
F	45	—	70
G	—	100	—
H	300	—	325
I	—	—	430

•MO_095a (Fig2)

标号	尺寸(mil)		
	最小	典型	最大
A	1145	—	1185
B	275	—	295
C	120	—	150
D	110	—	150
E	14	—	22
F	45	—	60
G	—	100	—
H	300	—	325
I	—	—	430

24-pin SOP (300mil)外形尺寸

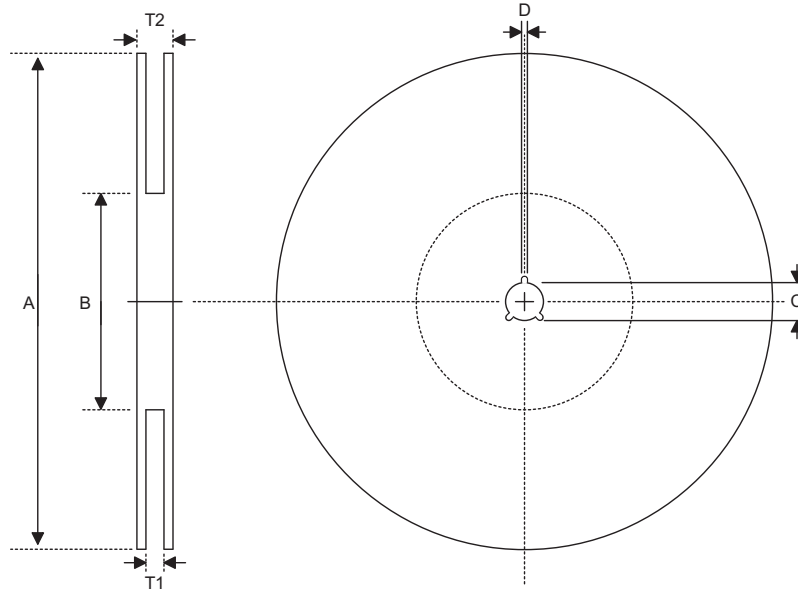


●MS_013

标号	尺寸(mil)		
	最小	典型	最大
A	393	—	419
B	256	—	300
C	12	—	20
C'	598	—	613
D	—	—	104
E	—	50	—
F	4	—	12
G	16	—	50
H	8	—	13
α	0°	—	8°

包装带和卷轴规格:

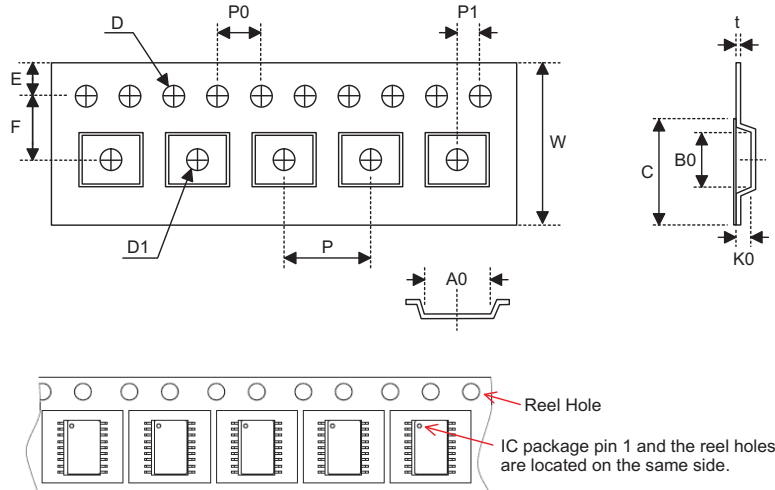
卷轴尺寸:



SOP 24W

标号	描述	尺寸(mm)
A	卷轴外圈直径	330.0±1.0
B	卷轴内圈直径	100.0±1.5
C	轴心直径	13.0 ^{+0.5/-0.2}
D	缝宽	2.0±0.5
T1	轮缘宽	24.8 ^{+0.3/-0.2}
T2	卷轴宽	30.2±0.2

运输带尺寸:



S0P 24W

标号	描述	尺寸(mm)
W	运输带宽	24.0±0.3
P	空穴间距	12.0±0.1
E	穿孔位置	1.75±0.1
F	空穴至穿孔距离(宽度)	11.5±0.1
D	穿孔直径	1.55±0.1
D1	空穴中之小孔直径	1.5±0.25
P0	穿孔间距	4.0±0.1
P1	空穴至穿孔距离(长度)	2.0±0.1
A0	空穴长	10.9±0.1
B0	空穴宽	15.9±0.1
K0	空穴深	3.1±0.1
t	传输带厚度	0.35±0.05
C	覆盖带宽度	21.3

盛群半导体股份有限公司（总公司）

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（台北业务处）

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)

盛扬半导体有限公司（上海业务处）

上海市宜山路2016号合川大厦1号楼3楼G室 200103
电话: 86-21-5422-4590
传真: 86-21-5422-4596
网站: www.holtek.com.cn

盛扬半导体有限公司（深圳业务处）

深圳市南山区科技园科技中三路与高新中二道交汇处生产力大楼A单元五楼 518057
电话: 0755-8616-9908, 8616-9308
传真: 0755-8616-9722

盛扬半导体有限公司（北京业务处）

北京市西城区宣武门西大街甲129号金隅大厦1721室 100031
电话: 010-6641-0030, 6641-7751, 6641-7752
传真: 010-6641-0125

盛扬半导体有限公司（成都业务处）

成都市东大街97号香槟广场C座709室 610016
电话: 028-6653-6590
传真: 028-6653-6591

Holtek Semiconductor(USA), Inc.（北美业务处）

46712 Fremont Blvd., Fremont, CA 94538
电话: 510-252-9880
传真: 510-252-9885
网站: www.holtek.com

Copyright © 2009 by HOLTEK SEMICONDUCTOR INC.

使用指南中所出现的信息在出版当时相信是正确的，然而盛群对于说明书的使用不负任何责任。文中提到的应用目的仅仅是用来做说明，盛群不保证或表示这些没有进一步修改的应用将是适当的，也不推荐它的产品使用在会由于故障或其它原因可能会对人身造成危害的地方。盛群产品不授权使用于救生、维生器件或系统中做为关键器件。盛群拥有不事先通知而修改产品的权利，对于最新的信息，请参考我们的网址 <http://www.holtek.com.tw>