

## 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](#)
  - [HA0016S HT48 MCU读写HT24系列EEPROM的应用范例](#)
  - [HA0018S HT48 MCU 对HT1621 LCD控制器的应用](#)
  - [HA0041S HT48CA0发射HT6221码的应用范例](#)
  - [HA0075S MCU重置电路及振荡电路应用](#)
  - [HA0076S HT48RAx/HT48CAx 软件应用要点](#)
  - [HA0082S HT48xA0-1、HT48xA0-2 Power-on Reset 时序](#)

## 特性

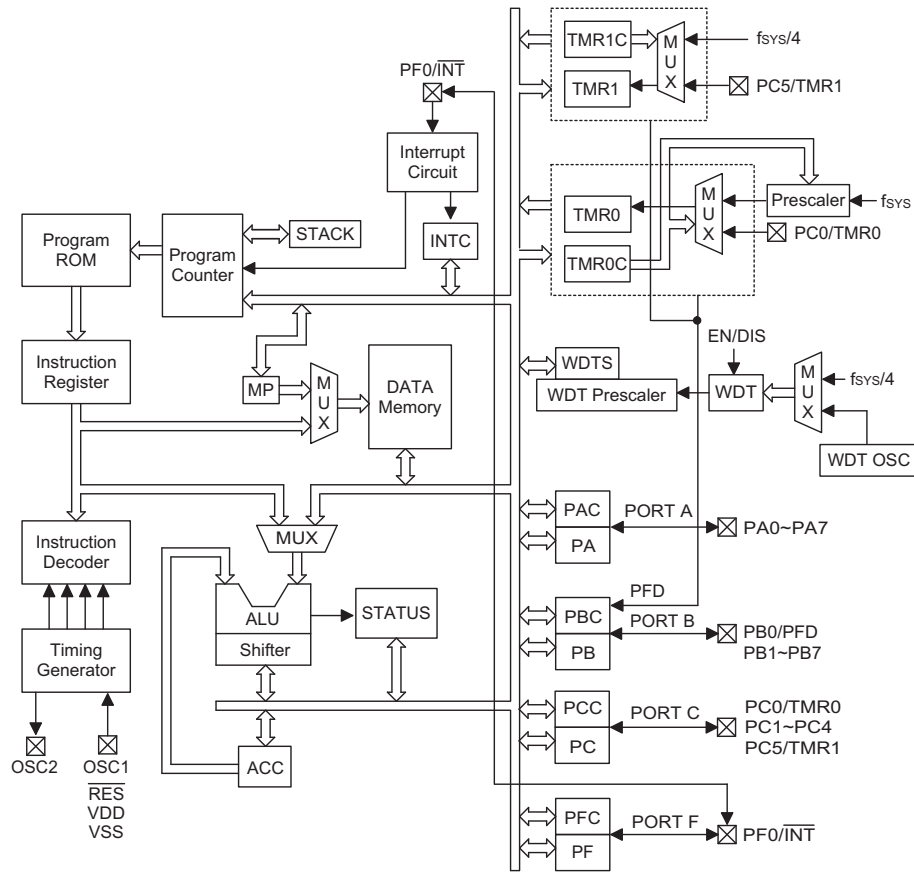
- 工作电压: 2.0V~5.5V
- 最多可有 23 个双向输入/输出口
- 1 个与输入/输出共用引脚的外部中断输入
- 8 位可编程定时/计数器, 带溢出中断及 8 阶预分频器(TMR0)
- 16 位可编程定时/计数器, 带溢出中断(TMR1)
- 内置晶体和 RC 振荡电路
- 看门狗定时器
- 8K×16 位的程序存储器 ROM
- 224×8 位的数据存储器 RAM
- 支持 PFD
- HALT 和唤醒功能来降低功耗
- 8 层硬件堆栈
- 在 VDD=3V, 系统频率为 4MHz 时, 指令周期为 1μs
- 位操作指令
- 查表指令, 表格内容字长 16 位
- 63 条功能强大的指令
- 所有指令执行时间皆为 1 或 2 个指令周期
- 低电压复位功能
- 28-pin SOP/SSOP(209mil)封装

## 概述

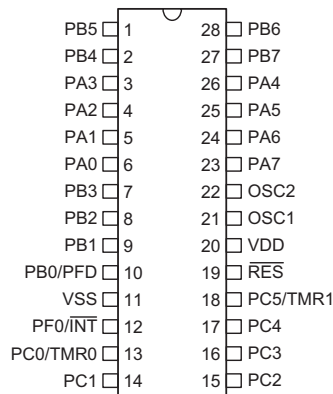
HT48RA1/HT48CA1 是一款八位高性能精简指令集单片机, 专为多输入/输出控制的产品设计。ROM 数据可以存储遥控码。掩膜版芯片 HT48CA1 在引脚和功能方面, 都与 OTP 版芯片 HT48RA1 完全相同。

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, 特别适用于多功能遥控器(URC)。

方框图



引脚图



HT48RA1/HT48CA1  
- 28 SOP-A/SSOP-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	唤醒* 上拉电阻***	8 位双向输入/输出端口，每一位能由掩膜选项设置为唤醒输入。可由软件指令设置为 CMOS 输出或斯密特触发输入，作为输入时可由掩膜选项设置是否带上拉电阻
PB0/PFD PB1~PB7	输入/输出	上拉电阻** PB0 或 PDF	8 位双向输入/输出。可由软件指令设置为 CMOS 输出或斯密特触发输入，作为输入时可由掩膜选项设置是否带上拉电阻 PB0 输出模式可以用作内部 PFD 信号输出，用作可变载波频率信号输出。
PC0/TMR0 PC1~PC4 PC5/TMR1	输入/输出	上拉电阻*	6 位双向输入/输出。可由软件指令设置为 CMOS 输出或斯密特触发输入，作为输入时可由掩膜选项设置是否带或上拉电阻（由上拉电阻选项决定）。PC0 和 PC5 与定时/计数器 0/1 外部输入共用。
PF0/ $\overline{\text{INT}}$	输入/输出	上拉电阻*	双向输入/输出。可由软件设置为 CMOS 输出或斯密特触发输入，作为输入时可由掩膜选项设置是否带或上拉电阻（由上拉电阻选项决定）。外部中断输入与 PF0 共用。
OSC1 OSC2	输入输出	晶体振荡 RC 振荡	OSC1 和 OSC2 连接至 RC 或晶体振荡（由掩膜选项确定）来产生内部系统时钟；在 RC 振荡方式下，OSC2 是系统时钟四分频的输出端。
$\overline{\text{RES}}$	输入	—	斯密特触发复位输入端，低电平有效
VSS	—	—	负电源，接地
VDD	—	—	正电源

注意： \* 选项以位为单位  
 \*\* 选项以位为半字节单位  
 \*\*\* 选项以字节为单位

## 极限参数

电源供应电压	..... $V_{SS}-0.3\text{V} \sim V_{SS}+6.0\text{V}$	储存温度	..... $-50^{\circ}\text{C} \sim 125^{\circ}\text{C}$
端口输入电压	... $V_{SS}-0.3\text{V} \sim V_{DD}+0.3\text{V}$	工作温度	..... $-40^{\circ}\text{C} \sim 85^{\circ}\text{C}$
端口总灌电流	..... 150mA	端口总源电流	..... -100mA
总功耗	..... 500mW		

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°C

符号	参数	测试条件		最小	典型	最大	单位
		V <sub>DD</sub>	条件				
V <sub>DD</sub>	工作电压	—	—	2.0	—	5.5	V
I <sub>DD1</sub>	工作电流	3V	无负载	—	0.6	1.5	mA
		5V	f <sub>sys</sub> =4MHz	—	2	4	
I <sub>DD2</sub>	工作电流(晶体振荡, RC 振荡)	5V	无负载 f <sub>sys</sub> =8MHz	—	4	8	mA
I <sub>STB1</sub>	静态电流 (看门狗打开, 看门狗 RC 振荡器打开)	3V	无负载	—	1.1	5	μA
		5V	暂停模式	—	4	10	
I <sub>STB2</sub>	静态电流 (看门狗关闭)	3V	无负载	—	0.1	1	μA
		5V	暂停模式	—	0.2	2	
V <sub>IL1</sub>	输入/输出口的低电平输入电压	—	—	0	—	0.3V <sub>DD</sub>	V
V <sub>IH1</sub>	输入/输出口的高电平输入电压	—	—	0.7V <sub>DD</sub>	—	V <sub>DD</sub>	V
V <sub>IL2</sub>	低电平输入电压 ( $\overline{\text{RES}}$ )	—	—	0	—	0.4V <sub>DD</sub>	V
V <sub>IH2</sub>	高电平输入电压 ( $\overline{\text{RES}}$ )	—	—	0.9V <sub>DD</sub>	—	V <sub>DD</sub>	V
V <sub>LVR</sub>	低电压复位	—	LVR=2.0V	1.8	1.9	2.0	V
			LVR=3.0V	2.7	3.0	3.3	
I <sub>OL</sub>	输入/输出灌电流	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	4	8	—	mA
		5V		10	20	—	
I <sub>OH</sub>	输入/输出源电流	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	—2	—4	—	mA
		5V		—5	—10	—	
R <sub>PH</sub>	上拉电阻	3V	—	20	60	100	K Ω
		5V	—	10	30	50	

## A.C.特性

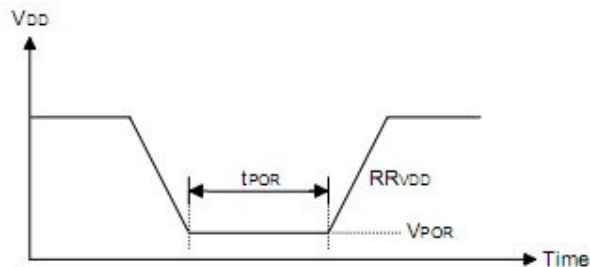
Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V <sub>DD</sub>	条件				
f <sub>SYS1</sub>	系统时钟 (晶体振荡)	—	2.0V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	
f <sub>SYS2</sub>	系统时钟 (RC 振荡)	—	2.0V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	
f <sub>TIMER</sub>	定时器的输入频率 (TMR0/TMR1)	3V	50%占空比	0	—	4000	kHz
		5V		0	—	8000	
t <sub>WDTOSC</sub>	看门狗振荡周期	3V	—	45	90	180	μs
		5V		32	65	130	
t <sub>WDT1</sub>	看门狗定时溢出周期(WDT 振荡)	3V	WDT 无预分频	11	23	46	ms
		5V		8	17	33	
t <sub>WDT2</sub>	看门狗定时溢出周期(F <sub>sys</sub> /4)	—	WDT 无预分频	—	1024	—	t <sub>sys</sub>
t <sub>RES</sub>	外部复位低电平脉冲宽度	—	—	1	—	—	μs
t <sub>SST</sub>	系统启动延时周期	—	上电复位或从暂停状态唤醒	—	1024	—	t <sub>sys</sub>
t <sub>LVR</sub>	低电压复位延时时间	—	—	1	—	—	ms
t <sub>INT</sub>	中断脉冲宽度	—	—	1	—	—	μs

 附注: t<sub>sys</sub> = 1/f<sub>sys</sub>

## 上电复位特性

符号	参数	测试条件		最小	典型	最大	单位
		V <sub>DD</sub>	条件				
V <sub>POR</sub>	完成上电复位的 VDD 起始电压	—	—	—	—	0	mV
RR <sub>VDD</sub>	完成上电复位的 VDD 上升速率	—	—	0.05	—	—	V/ms
t <sub>POR</sub>	完成上电复位时 VDD 停留在 V <sub>POR</sub> 的最小时间	—	—	200	—	—	ms

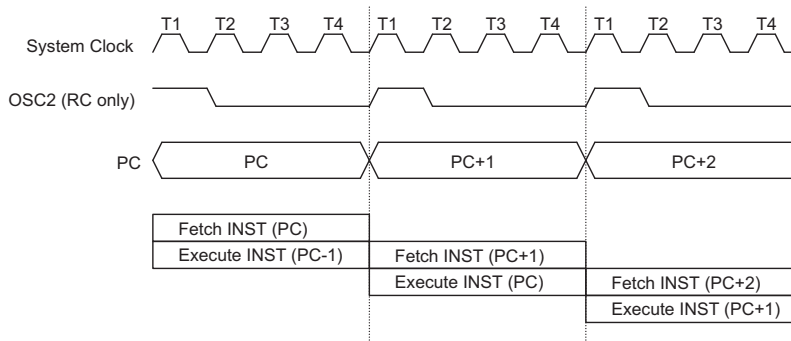


## 系统功能说明

### 指令系统

系统时钟由晶体振荡器或 RC 振荡器产生，系统内部将此频率分为四个不重叠的时钟（一般称为 T 状态，分别为 T1、T2、T3、T4），一个指令周期包含了四个 T 状态，即一个指令周期为四个系统时钟周期。

指令的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，如果涉及到的指令要改变程序计数器（如 JMP，CALL 等），就需要花两个指令周期来完成这一条指令。



指令时序

### 程序计数器（Program Counter）

程序计数器是作为程序存储器寻址之用，控制了程序的流程单片机通过 PC 指向的程序存储器的地址取得一条指令码后，PC 会自动地指向下一条指令的地址（PC 值自动加 1）。

但是若执行的是如下指令：跳转、条件跳跃、读取 PCL 的值、子程序调用、初始复位、内部或外部中断响应、子程序返回等，则 PC 要根据每一条指令获得其相应的地址来控制程序的转向。

比如执行条件跳转指令，一旦条件符合，则在当前执行指令时所获取的指令码不会被执行，并且同时会插入一个空指令周期（dummy cycle），换句话说，相当与执行了一个 NOP 指令（空操作），这样 PC 才会指向正确的指令码的地址；反之，条件不符合时，PC 将指向下一条指令的地址。

PC 的低位（PCL）是可读写的暂存器（06H）。若向 PCL 写入一个值将会产生一个短程的跳跃动作，这个短程跳跃的地址范围是 256 个地址，即在 ROM 的当前页。当发生控制转移时，就会加入一个空指令周期。

模式	程序计数器								
	*12~*8	*7	*6	*5	*4	*3	*2	*1	*0
初始化复位	00000	0	0	0	0	0	0	0	0
外部中断	00000	0	0	0	0	0	1	0	0
定时/计数器 0 溢出中断	00000	0	0	0	0	1	0	0	0
定时/计数器 1 溢出中断	00000	0	0	0	0	1	1	0	0
条件跳跃	PC+2								
装入 PCL	*12~*8	@7	@6	@5	@4	@3	@2	@1	@0
跳转、子程序调用	#12~#8	#7	#6	#5	#4	#3	#2	#1	#0
从子程序返回 (RET, RETI)	S12~S8	S7	S6	S5	S4	S3	S2	S1	S0

程序计数器

附注： \*12~\*0：程序指针  
#12~#0：指令指针

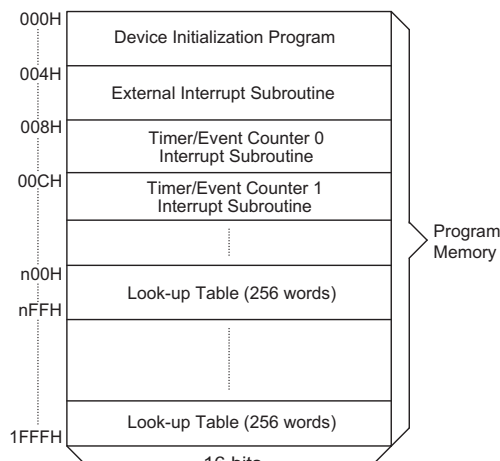
S12~S0：堆栈指针  
@7~@0：PCL 字节

**程序存储器 (Program Memory — ROM)**

程序存储器用来存储要执行的程序指令，也包含数据、表格、中断入口地址，由 8192×16 个位组成，由 PC 和表格指针来确定其地址。

ROM 里面的某些地址是为一些特殊使用而保留的，使用时应加以注意，避免误用，导致程序运行的不正常，以下是说明：

- 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  
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滿，则程序会跳转到 004H 地址开始执行。服务程序。
- 地址 008H  
此地址保留给定时/计数器 0 中断服务使用。如果中断允许且堆栈未滿，则一旦定时/计数器 0 发生溢出时，就能产生中断，程序会从 008H 地址开始执行中断服务程序。
- 地址 00CH  
此地址保留给定时/计数器 1 中断服务使用。如果中断允许且堆栈未滿，则一旦定时/计数器 1 发生溢出时，就能产生中断，程序会从 00CH 地址开始执行中断服务程序。
- 表格区 (Table Location)



**程序存储器**

ROM 内的任何地址都可被用来作为查表地址使用。查表指令为 TABRDC [m] 与 TABRDL [m]。TABRDC [m]是查表当前页的数据 [1 页=256 个字 (word)]。TABRDL [m]是查表最后一页的数据。[m] 为数据被存入的地址。在执行 TABRDC [m]指令 (或 TABRDL [m] 指令) 后，将会传送当前页 (或最后一页) 上的一个字的低位字节到[m]，而这个字的高位字节传送到 TBLH (08H)。只有表格中的低位字节被定义到目标地址中，而高位字节传送到表格的高位字节寄存器 (TBLH)。TBLH 为只读寄存器。而表格指针 (TBLP; 07H) 是可以读写的寄存器，用来指明表格地址。在访问表格以前，通过对 TBLP 寄存器赋值来指明表格低位地址。高位字节寄存器 TBLH 只能读出，不能写入。如果主程序和中断服务程序 (ISR) 同时使用查表指令，那么主程序读取的高位字节 (即存放于高位字节寄存器 TBLH 之中) 可能会被中断服务程序的查表指令改写而产生错误。因此，应该避免主程序和中断服务程序 (ISR) 同时使用查表指令。但是，如果主程序和中断服务程序必须同时使用查表指令的话，那么，主程序在使用查表指令之前，必须先关闭所有使用查表指令的相关中断，直到高位字节寄存器 TBLH 的内容被备份好再开放这些中断。查表指令要花两个指令周期来完成这一条指令的操作。按照用户的需要，表格地址这些位置可以作为正常的程序存储器来使用。

指令	表格地址								
	*12~*8	*7	*6	*5	*4	*3	*2	*1	*0
TABRDC [m]	TBHP	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	11111	@7	@6	@5	@4	@3	@2	@1	@0

**表格区**

附注： \*12~\*0: 表格地址位                      @7~@0: 表格指针位

## 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。HT48RA1/HT48CA1 有 8 层堆栈，堆栈寄存器既不是数据存储器的一部分，又不是程序存储器的一部分，而且也不可读不可写。堆栈的使用是通过堆栈指针 SP 来实现的，而堆栈指针 SP 也是不可读不可写的。一旦发生了子程序的调用或是中断响应，则程序计数器 PC 的内容会被压入堆栈中。在子程序调用或中断响应结束时（执行指令 RET 或 RETI），程序计数器 PC 的值会从堆栈中还原。在系统复位后，堆栈指针 SP 会指向堆栈的顶端。

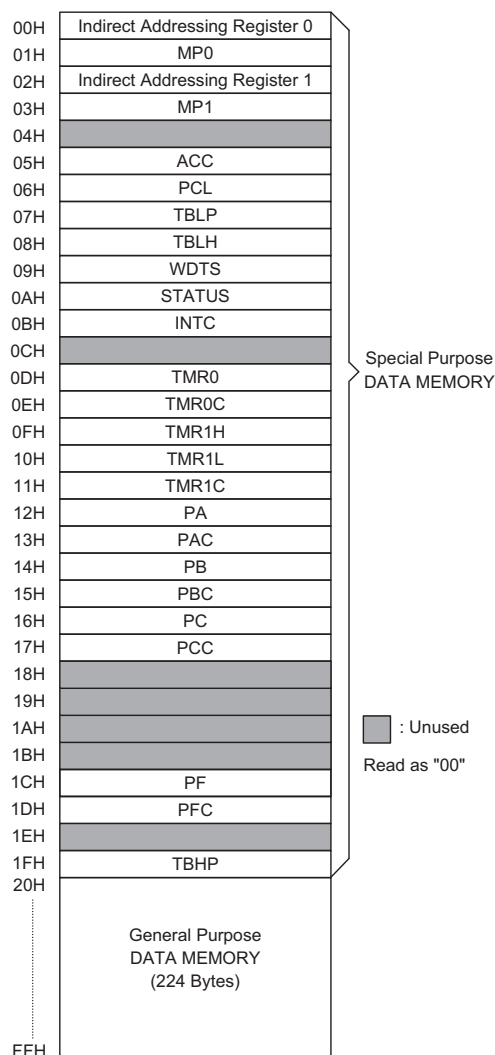
当堆栈已满，而此时又发生的中断请求，则这个中断的请求标志会被记录下来，该中断服务仍被禁止。一旦堆栈指针 SP 发生了递减（由于 RET 或 RETI）则会响应此未被服务的中断。这个功能就可防止堆栈溢出，使得编程者更易于使用该结构。同样，如果堆栈已满，而随后又执行了 CALL 指令，此时会发生堆栈溢出，并且第一个返回地址将会丢失（只有最近的 8 个返回地址会被保存）。

## 数据存储器—RAM

数据存储器 (RAM) 由  $249 \times 8$  个位组成，包含特别功能寄存器、通用数据寄存器 ( $224 \times 8$ ) 两个不同功能的区间。这些空间大多是可读可写的，只有少部分是只读的。

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 高字节 (TMR1H; 0FH)，定时/计数器 1 低字节 (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)，间接寻址寄存器 (MP0; 01H, MP1; 03H)，累加器 (ACC; 05H)，表格指针 (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)，看门狗选项设置寄存器 (WDTS; 09H)，输入/输出寄存器 (PA; 12H, PB; 14H, PC; 16H, PF; 1CH)，输入/输出控制寄存器 (PAC; 13H, PBC; 15H, PCC; 17H, PFC; 1DH)。在 20H 以前的剩余单元都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 的值。通用数据存储器的地址从 20H~FFH，作为数据和控制信息使用。

所有的 RAM 都可以直接执行算术、逻辑、递增、递减和移位等运算。除了一些少数指定的位之外，RAM 的每个位都可以由 SET[m].i 和 CLR[m].i 指令来置位和复位。这些 RAM 地址可以通过间接寻址寄存器 MP0(01H)或 MP1(03H) 来存取。



数据存储器

### 间接寻址寄存器 (Indirect Addressing Register)

地址 00H 和 02H 作为间接寻址寄存器。它们都没有实际的物理结构。任何对[00H]和[02H]的读/写操作，都会访问由 MP0[01H]和 MP1[03H]所指向的 RAM 单元。间接地读取 00H 或 02H 单元，将会返回 00H，而间接地写入 00H 或 02H 单元，则不会产生任何操作。

间接寻址寄存器 MP0 和 MP1 都是 8 位寄存器。

### 累加器 (ACC)

累加器 (ACC) 与算术逻辑单元 (ALU) 有关，同样也是对应至 RAM 的地址 05H，作为运算的立即数据，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) 为执行八位算术及逻辑运算的电路，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 (RL, RR, RLC, RRC)
- 递增及递减 (INC, DEC)
- 分支判断 (SZ, SNZ, SIZ, SDZ 等)

ALU 不仅可以储存数据运算的结果，还可以改变状态寄存器。

### 状态寄存器—STATUS

状态寄存器 (0AH) 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)，暂停标志位 (PDF)，看门狗定时器溢出标志位 (TO) 组成。该寄存器不仅记录状态信息，而且还控制运算顺序。

除了 TO 和 PDF 以外，状态寄存器中的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 PDF 标志位。但是与状态寄存器有关的操作会导致状态寄存器的改变。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 PDF。

Z, OV, AC 和 C 标志位反映了最近一次的运算状态。

进入中断程序或执行子程序调用时，状态寄存器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。它也可被一个带进位循环移位指令影响。
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，或那么 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，或执行 HALT 指令，TO 被清除。WDT 定时溢出，TO 被置位。
—	6,7	未定义，读出为零

状态寄存器

## 中 断—INT

本单片机提供一个外部中断和内部定时/计数器中断。中断控制寄存器 (INTC; 0BH) 包含了中断控制位, 用来设置中断允许/禁止及中断请求标志。

一旦有中断子程序被服务, 所有其它的中断将被禁止 (通过清除 EMI 位)。这种机制能防止中断嵌套。这时如有其它中断请求发生, 这个中断请求的标志会被记录下来。如果在一个中断服务程序中有另一个中断需要服务的话, 程序员可以设置 EMI 位及 INTC 所对应的位来允许中断嵌套服务。如果堆栈已满, 该中断请求将不会被响应。即使相关的中断被允许, 也要到堆栈指针发生递减时才会响应。如果需要立即得到中断服务, 则必须避免让堆栈饱和。

所有的中断都具有唤醒功能。当一个中断被服务时, 会产生一个控制传送: 通过将程序计数器 (PC) 压入堆栈, 然后转移到中断服务程序的入口。只有程序计数器的内容能压入堆栈。如果寄存器和状态寄存器的内容会被中断服务程序改变, 从而破坏主程序的预定控制, 那么程序员必须事先将这些数据保存起来。

外部中断是由 INT 脚上的下降沿触发的, 相关的中断请求位 (EIF, INTC 的第 4 位) 被置位。当中断允许, 堆栈也没有满, 一个外部中断触发时, 那么将会产生地址 04H 的子程序调用。中断请求标志 (EIF) 和 EMI 位也将被清除来禁止另外的中断发生。

内部定时/计数器 0 中断是通过置位定时/计数器 0 中断请求标志位 (TOF, INTC 的第 5 位) 来初始化的, 中断的请求是由定时器溢出产生的。当中断允许, 堆栈又未滿, 并且 TOF 已被置位, 就会产生地址 08H 的子程序调用。该中断请求标志位 (TOF) 被复位并且 EMI 位也将被清除, 以便禁止其他中断。

内部定时/计数器 1 中断是通过置位定时/计数器中断请求标志位 (T1F, INTC 的第 6 位) 来初始化的, 中断的请求是由定时器溢出产生的。当中断允许, 堆栈又未滿, 并且 T1F 已被置位, 就会产生地址 0CH 的子程序调用。该中断请求标志位 (T1F) 被复位并且 EMI 位也将被清除, 以便禁止其他中断。

单片机在执行中断子程序期间, 其他的中断响应会被暂停, 直到执行 RETI 指令或是 EMI 位和相关的中断控制位都被置为 1 (当堆栈是未滿时)。若要从中断子程序返回时, 只要执行 RET 或 RETI 指令即可。RETI 指令将会自动置位 EMI 来再次允许中断服务, 而 RET 则不能自动置位 EMI。

如果中断在内部二个连续的 T2 脉冲上升沿间发生, 而且中断响应被允许的话, 那么在第二个 T2 脉冲, 该中断会被服务。如果同时发生中断服务请求, 那么下列表中列出了中断服务优先等级。这种优先级也可以通过 EMI 位的复位来屏蔽。

中断控制寄存器 (INTC) 由定时/计数器 0 中断请求标志位 (TOF), 定时/计数器 1 中断请求标志位 (T1F), 外部中断请求标志位 (EIF), 定时/计数器 0 允许位 (ETOI), 定时/计数器 1 允许位 (ET1I), 外部中断允许位 (EEI), 和主中断控制允许位 (EMI) 组成。EMI、EEI、ETOI 和 ET1I 都是用来控制中断的允许/禁止状态的。这些位防止正在进行的中断服务中的中断请求。一旦中断请求标志位被置位 (TOF, T1F, EIF), 它们将在 INTC 寄存器中被保留下来, 直到相关的中断被服务或由软件指令来清除。

建议不要在中断子程序中使用“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。基于上述情况, 如果只剩下一个堆栈, 若此时中断不能很好地被控制, 而且在这个中断服务程中又执行了 CALL 子程序调用, 则会造成堆栈溢出, 而破坏原先的控制序列。

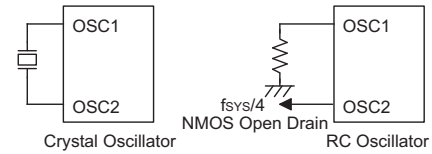
INTC (0BH)	位	符号	功 能
	0	EMI	总中断控制位(1=允许, 0=禁止)
	1	EEI	外部中断控制位(1=允许, 0=禁止)
	2	ETOI	定时/计数器 0 中断控制位(1=允许, 0=禁止)
	3	ET1I	定时/计数器 1 中断控制位(1=允许, 0=禁止)
	4	EIF	外部中断请求标志位(1=有, 0=无)
	5	TOF	定时/计数器 0 中断请求位(1=有, 0=无)
	6	T1F	定时/计数器 1 中断请求位(1=有, 0=无)
7	—	未使用位, 读数为零	

中断控制寄存器—INTC (0BH)

中断源	优先级	中断
外部中断	1	04H
定时/计数器 0 中断	2	08H
定时/计数器 1 中断	3	0CH

### 振荡器

HT48RA1/HT48CA1 有 2 种振荡电路。这 2 种振荡器都是针对系统时钟而设计的，分别是外部 RC 振荡、外部晶体振荡。不管所选的是哪一种振荡器，其信号都可以支持系统的时钟，可由掩膜选项设置。进入 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。



系统振荡器

如果使用外部 RC 型振荡器，在 OSC1 和 VSS 之间需要一个外部电阻，其阻值范围为 100kΩ ~ 820kΩ。在 OSC2 端可获得系统频率四分频信号，用于同步外部逻辑电路。RC 振荡方式是一种低成本方案，可是，振荡频率会随着 V<sub>DD</sub>、温度和制造漂移而不同。因此，在用于需要非常精确振荡频率的计时操作场合，我们并不建议使用 RC 型振荡器。

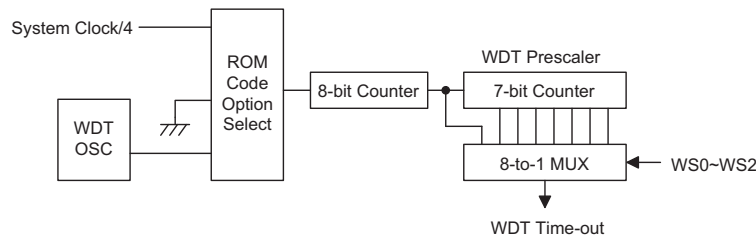
如果选用的是晶体振荡器，那么在 OSC1 和 OSC2 之间需要连接一个晶体，用来提供晶体振荡器所需要的反馈和相移。另外，在 OSC1 和 OSC2 之间还可以用谐振器代替晶体振荡器，来产生系统时钟，但是在 OSC1 和 OSC2 需要多连接两个电容器至地。

WDT 振荡器是 IC 内部 RC 型振荡器，不需要任何外部元件。即使在系统进入暂停模式，系统时钟被停止，但这个 RC 振荡器仍会运作，其振荡周期大约为 90 μs。在掩膜时，如欲节省电源，可在掩膜选项中关闭 WDT 振荡器。

### 看门狗计时器(Watchdog Timer)

WDT 的时钟源来自看门狗振荡器 (WDT 振荡器) 或是指令时钟 (系统时钟 4 分频)，由掩膜选项设置。看门狗主要用来避免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。看门狗可用掩膜选项设置为打开或关闭，如果在关闭状态，所有的 WDT 指令都是没有作用的。

如果选择了内部 WDT 振荡器 (RC 振荡周期一般为 90μs/3V) 的话，这个频率会先除以 256 (8 级) 产生 23ms/3V 的溢出时间。这个溢出时间会因为温度，V<sub>DD</sub>，以及芯片参数的漂移而变化。如果使用 WDT 的预分频器，则可实现延长 WDT 溢出时间。设置 WS2, WS1, WS0 (WDTS 的第 2、1、0 位) 会产生不同的溢出时间。举例来说，如果 WS2, WS1, WS0 的位都为 1，其分频级数最大为 1: 128，得到最长的 WDT 溢出周期 2.9s/3V。如果 WDT 振荡器被禁止，那么 WDT 的时钟来源可为指令时钟，其运作与 WDT 振荡器一样。但当在 HALT 状态时，WDT 会停止计数而失去保护功能。在这种情况下，只能由外部逻辑复位来重新启动系统。WDTS 的高四位及其第 3 位保留给用户定义标志来使用，程序员可以利用这些标志来指示某些特殊的状态。



看门狗定时器

如果单片机工作在干扰很大的环境中，那么建议使用片内的 RC 振荡器 (WDT OSC)，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。

在正常运作下，WDT 溢出会使系统复位并设置 TO 状态位。但在 HALT 模式下，溢出只产生一个热复位，并只能使程序计数器 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），用清除看门狗指令和 HALT 指令三种。清除看门狗指令有“CLR WDT”和“CLR WDT1”及“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”（即 CLR WDT 次数为 1），那么只要执行 CLR WDT 指令就会清除 WDT。若选择 CLR WDT1 和 CLR WDT2 的情况下（即 CLR WDT 次数为 2），那么要两条指令交替使用才会清除 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

**WDTS 寄存器**

### 暂停模式 (HALT)

暂停模式是由 HALT 指令来实现的，系统状态如下：

- 关闭系统振荡器，但 WDT 振荡器依然工作(如果 WDT 振荡器被选择)。
- RAM 及寄存器的内容保持不变。
- WDT 被清除，并重新计数（如果 WDT 的时钟是来自 WDT 振荡器）。
- 所有的输入/输出口都保持其原先状态。
- 置位 PDF 标志位，清除 TO 标志位。

外部复位、中断或 PA 口下降沿信号或 WDT 溢出均可使系统脱离暂停状态。外部复位能使系统初始化，而 WDT 溢出能执行“热复位”。通过检测 TO 和 PDF 标志，即可了解系统复位的原因。PDF 标志位是由系统上电复位和执行 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，使 TO 标志位置位，同时产生唤醒，使得程序计数的 PC 和堆栈指针复位。其他都保持原状态。

PA 口的唤醒和中断的方式看作为正常运行，PA 口的每一位都可以通过掩膜选项来设定为唤醒功能。如果唤醒是来自于输入/输出信号的变化，程序会继续执行下一条命令。如果唤醒是来自中断的话，则会产生二种情况：如果相关的中断被禁止或中断是允许的，但堆栈已满，那么程序将继续执行下条指令，如果中断允许并且堆栈未满，那么这个中断响应就发生了。当唤醒事件发生时，要花 1024tsys（系统时钟周期）后，系统重新正常运行。这就是说，在唤醒后被插入了一个等待时间。如果唤醒是来自于中断响应，那么实际的中断程序执行就被延迟了一个以上的周期。但是如果唤醒导致下一条指令执行，那么在一个等待周期结束后指令就立即被执行。进入 HALT 模式前，如果中断请求标志位被置“1”，那么相关的中断唤醒功能被禁止。

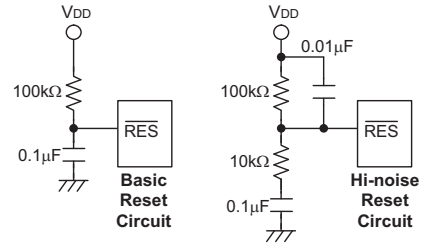
为了减小功耗，在进入 HALT 模式之前必须要小心处理输入/输出口的状态。

**复位 (RESET)**

有三种方法可以产生复位

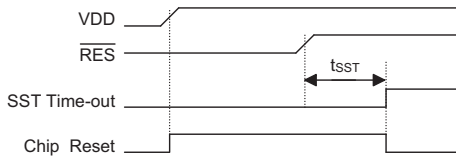
- 在正常运行时由RES脚产生复位。
- HALT 期间由RES脚产生复位。
- 正常运行时，WDT 溢出复位。

暂停模式中的看门狗定时器溢出与其它系统复位状况不同，因为看门狗定时器溢出会执行热复位，热复位只复位程序计数器 PC 和堆栈指针 SP，而系统其它部分都保持原有状态。在其它复位状态下，某些寄存器不会改变。在初始复位时，大部分寄存器会复位成初始的状态。通过检测 PDF 和 TO 标志，即可判断出各种不同的复位原因。

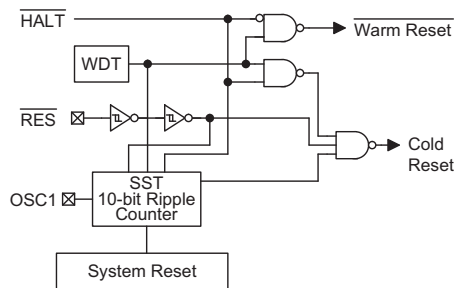


**复位电路**

注意：大多数应用中使用基本复位电路即可，如方案中外部噪音或杂讯比较厉害，就请选用高抗杂讯电路



**复位时序**



**复位电路结构**

TO	PDF	复位条件
0	0	电源上电复位
u	u	正常运作时由RES发生复位
0	1	由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

注意：u 表示不变

为了保证系统振荡器起振并稳定运行，SST（系统启动定时器）当系统复位时（上电、WDT 定时器溢出或是RES引脚复位）或是 HALT 模式唤醒时，它会提供额外的 1024 个系统时钟周期的延迟。

系统复位时，SST 被加到复位延时中。任何来自 HALT 的唤醒都将产生 SST 延迟。

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	禁止
预分频器	清除
看门狗定时器	清除，复位后看门狗定时器开始计数
定时/计数器	关闭
输入/输出口	输入模式
堆栈指针	指向堆栈的顶端

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 (正常工作)	$\overline{\text{RES}}$ 端复位 (正常工作)	$\overline{\text{RES}}$ 端复位 (暂停模式)	WDT 溢出 (暂停模式)*
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PC	0000H	0000H	0000H	0000H	0000H
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMRO	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRO0	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

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

HT48RA1/HT48CA1 提供两个定时/计数器。定时/计数器 0 包含一个 8 位可编程的向上计数的计数器，并且其时钟来源可以是外部信号输入、系统时钟。定时/计数器 1 则包含一个 16 位的可编程的向上计数的计数器，且其时钟来源可来自外部信号输入、指令时钟/4。

外部时钟输入，允许用户去计数外部事件，测量时间长度或脉冲宽度或产生一个精确的时基信号。

定时/计数器 0 可以产生 PFD 信号，时钟源来自外部或者内部时钟，PFD 频率 =  $f_{INT}/[2 \times (256-N)]$ 。

有两个寄存器与定时/计数器 0 相关联，即 TMR0 ([0DH]) 和 TMR0C ([0EH])。当处于定时/计数器 0 计数模式时 (TOON)，写 TMR0 会将初始值装入到定时/计数器 0 的预置寄存器中。每次写 TMR0 动作，都会改变预置寄存器。读 TMR0 则会获得定时/计数器 0 的内容。TMR0C 是定时/计数器控制寄存器，可以定义工作模式、计数与否、边沿触发。

TOM0 和 TOM1 位定义工作模式。外部事件计数模式用来记录外部事件，它的时钟来自外部 TMR0 引脚。定时器模式是一个常用功能，时钟源来自  $f_{INT}$  时钟。脉冲宽度测量模式能用来测量外部引脚(TMR0)上的高电平或低电平的宽度。计数是基于  $f_{INT}$  时钟。

在外部事件计数或定时器模式中，一旦定时/计数器开始计数，它将会从当前定时/计数器中的数值开始向上计数到 0FFH。一旦产生溢出，计数器会从定时/计数器 0 预置寄存器重新装载，并置位中断请求状态位 (TOF ; INTC 的第 5 位)。

在脉冲宽度测量中，将 TOON 和 TOE 置为“1”，如果 TMR0 接收到上升沿（或下降沿，如果 TOE 位被清零），就开始计数，直到 TMR0S 返回到原来的电平，并复位 TOON 位。测量的结果被保留在定时/计数器 0 中，即使电平跳变再一次发生也不会改变。换句话说，一次只能测量一个脉宽。直到 TOON 再次被置位，这样再次收到跳变信号，测量会再次执行。要注意在这个操作模式中，定时/计数器 0 的启动计数不是根据逻辑电平，而是依据信号的边沿跳变触发。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，并引发中断请求，这种情况与其另外两个模式一样。

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

在定时/计数器 0 为关闭的状态下，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 已经开始运行，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。

读取定时/计数器 0 (TMR0) 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。

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。

符号	位	功能
TOPSC0 TOPSC1 TOPSC2	0 1 2	定义预分频器级数, TOPSC2, TOPSC1, TOPSC0= 000: $f_{INT} = f_{sys}/2$ 或者 $f_{RTC}/2$ 001: $f_{INT} = f_{sys}/4$ 或者 $f_{RTC}/4$ 010: $f_{INT} = f_{sys}/8$ 或者 $f_{RTC}/8$ 011: $f_{INT} = f_{sys}/16$ 或者 $f_{RTC}/16$ 100: $f_{INT} = f_{sys}/32$ 或者 $f_{RTC}/32$ 101: $f_{INT} = f_{sys}/64$ 或者 $f_{RTC}/64$ 110: $f_{INT} = f_{sys}/128$ 或者 $f_{RTC}/128$ 111: $f_{INT} = f_{sys}/256$ 或者 $f_{RTC}/256$
TOE	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)
TOON	4	打开/关闭定时/计数器( 1=打开, 0=关闭)
—	5	未用, 读出为 0
TOM0 TOM1	6 7	定义工作模式 (TOM0, TOM1) 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 (0EH) 寄存器

有三个寄存器与定时/计数器 1 相关联, 即 TMR1H ([0FH]) 和 TMR1L ([10H]); TMR1C ([11H])。若写入 TMR1L 只能将数据写入低字节内部缓冲器 (8 bit) 中, 但若写入的是 TMR1H 则可将数据和低字节内部缓冲器的内容写入 TMR1H 和 TMR1L 的加载寄存器之中。每一次对 TMR1H 的写操作都会改变定时/计数器 1 加载寄存器的内容。若读取 TMR1H 则将锁存 TMR1H 的内容并将 TMR1L 传送至低字节内部缓冲器之中, 以避免发生计时错误。然而, 若读取 TMR1L, 则只读回低字节内部缓冲器的内容。换言之, 定时/计数器 1 的低字节数据并不能直接读取。若欲读取该低字节的数据, 必须先读取 TMR1H, 以便将定时/计数器的低字节数据传送至内部低字节缓冲器之中。TMR1C 是定时/计数器 1 控制寄存器, 它可定义: 工作模式、计数功能打开或关闭、计数的触发沿。

T1M0 和 T1M1 位定义工作模式。外部事件计数模式用来记录外部事件, 它的时钟来自外部 TMR1 引脚。定时器模式是一个常用功能, 时钟源来自指令时钟。脉冲宽度测量模式能用来测量外部引脚 (TMR1) 上的高电平或低电平的宽度。计数是基于指令时钟。

在外部事件计数或定时器模式中, 一旦定时/计数器 1 开始计数, 它将会从当前定时/计数器 1 中的数值开始向上计数到 0FFFFH。一旦产生溢出, 计数器会从定时/计数器 1 预置寄存器重新装载, 并置位中断请求状态位 (T1F ; INTC 的第 6 位)。

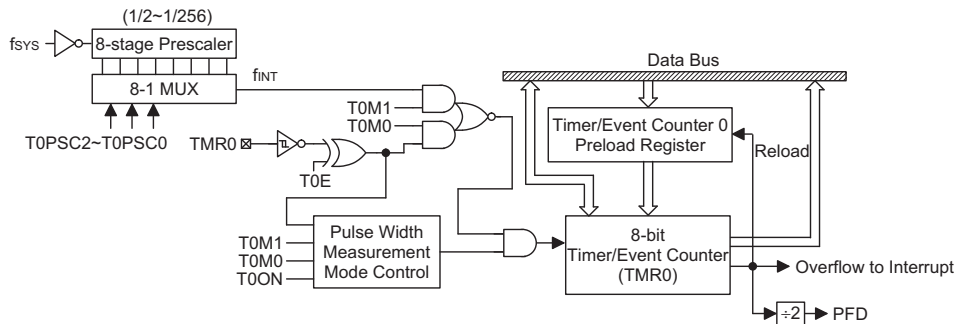
在脉冲宽度测量中, 将 T1ON 和 T1E 置为“1”, 如果 TMR1 接收到上升沿 (或下降沿, 如果 T1E 位被清零), 就开始计数, 直到 TMR1 返回到原来的电平, 并复位 T1ON 位。测量的结果被保留在定时/计数器 1 中, 即使电平跳变再一次发生也不会改变。换句话说, 一次只能测量一个脉宽。直到 T1ON 再次被置位, 这样再次收到跳变信号, 测量会再次执行。要注意在这个操作模式中, 定时/计数器 1 的启动计数不是根据逻辑电平, 而是依据信号的边沿跳变触发。一旦发生计数器溢出, 计数器会从定时/计数器 1 的预置寄存器重新装入, 并引发中断请求, 这种情况与其另外两个模式一样。

要使得计数运行, 只要将定时器启动位 (T1ON; TMR1C 的第 4 位) 置 1。在脉宽测量模式中, T1ON 在测量周期结束后自动被清零。但在另外两个模式中, T1ON 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响应。

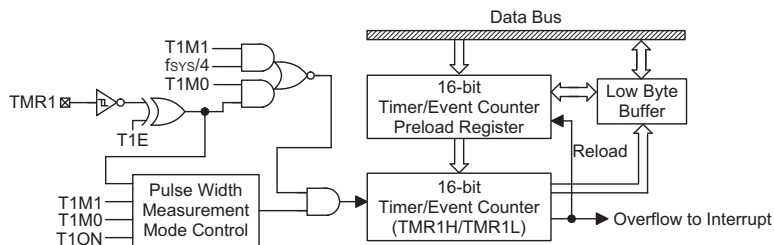
在定时/计数器 1 为关闭的状态下, 写数据到定时/计数器 1 的预置寄存器之中, 同时也会将数据装入定时/计数器 1 中。但若是定时/计数器 1 已经开始运行, 写到定时/计数器 1 的数据只会被保留在定时/计数器 1 的预置寄存器中, 直到定时/计数器 1 发生计数溢出为止, 再由预置寄存器加载新的值。读取定时/计数器 1 (TMR1) 时, 计数会被停止, 以避免发生错误; 计数停止会导致计数错误, 程序员必须注意到这一点。

符号	位	功能
—	0 ~ 2	未定义, 读数为“0”
T1E	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)
T1ON	4	打开/关闭定时/计数器(1=打开, 0=关闭)
—	5	未用, 读数为 0
T1M0 T1M1	6 7	定义操作模式(T1M1,T1M0) 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

**TMR1C (11H) 寄存器**

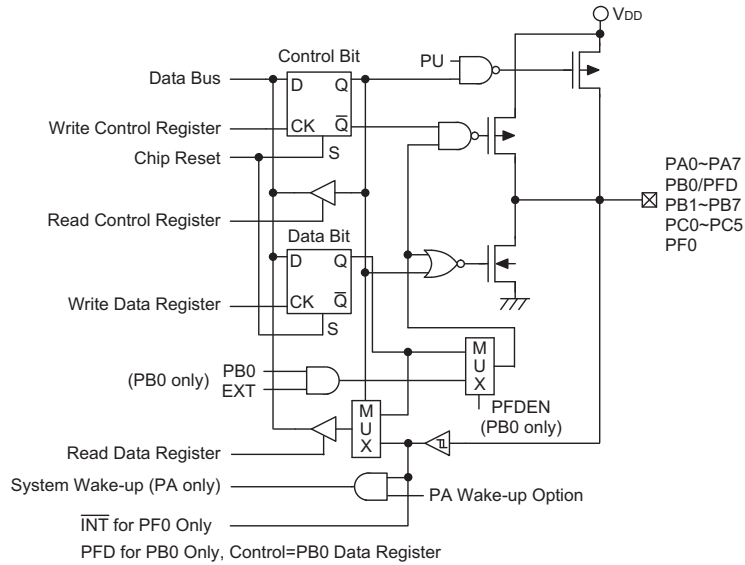


**定时/计数器 0**



**定时/计数器 1**

输入/输出口



输入/输出口

单片机具有 23 个双向输入输出口，标号从 PA 到 PC 以及 PF，其分别对应的 RAM 的[12H], [14H], [16H]和[1CH]。所有的输入/输出口都能被作为输入或输出使用。就输入而言这些口不具有锁存功能，即，输入数据必须在“MOV A, [m]” (m=12H,14H, 16H 或 1CH) 指令的 T2 上升沿被准备好。对输出而言，所有的数据被锁存并保持不变，直到输出锁存器重新被改写。

每个 I/O 口都有其自己的控制寄存器 (PAC, PBC, PCC, PFC)，用来控制输入/输出的设置。使用控制寄存器，可对 CMOS 输出或带或不带上拉电阻的斯密特触发输入在软件下动态地进行改变。要设置为输入功能，相应的控制寄存器必须写“1”。信号源的输入也取决于控制寄存器。如果控制寄存器的某位值为“1”那么输入信号是读取自这个引脚 (PAD) 的状态，但是如果控制寄存器的某位值为“0”，那么锁存器的内容将会被送到内部总线。后者，会在“读改写”指令中发生。

对于输出功能，只能设置为 CMOS 输出。这些控制寄存器是对应于内存的 13H, 15H, 17H 和 1DH 地址。

芯片复位后，这些输入/输出口都会是高电平（掩膜选择上拉电阻）或浮空状态（掩膜选择无上拉电阻）。每一个输入/输出锁存位都能被 SET [m].i 或 CLR [m].i 指令置位或清零，(m=12H, 14H, 16H 或 1CH.)

某些指令会首先输入数据然后进行输出操作。例如，SET [m].i，CLR [m].i, CPL [m]和 CPLA [m] 指令，读取输入口的状态到 CPU，执行这个操作（位操作），然后将数据写回锁存器或累加器。

PA 的每一个口都具有唤醒系统的能力。PC 口的高 2 位和 PF 口的高 7 位在物理上是不存在的；读这些位将返回“0”，而写这些位结果为无操作。请看应用注释。

PB0 和 PFD 管脚复用。如果选择为 PFD 输出，PB0 在输出模式时的输出信号将是 PFD 信号。在输入模式始终保持它的原来的功能。PF0 和 PC0 分别与 INT 和 TMR0 管脚共用引脚。INT 信号直接从 PF0 输入。PFD 信号（输出模式）仅仅由 PB0 数据寄存器控制。

PB0/PFD 的真值表格如下：

PBC(15H) Bit0	I	O	O	O
PB0/PFD 选项	×	PB0	PFD	PFD
PB0(14H) Bit0	×	D	0	1
PB0 管脚状态	I	D	0	PFD

注释：I：输入；O：输出；D：数据；

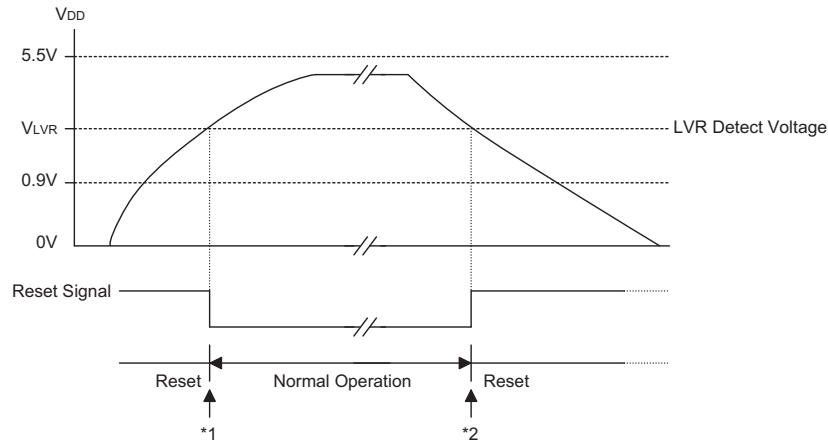
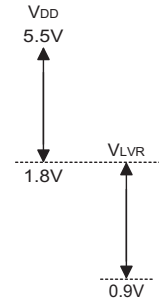
### 低电压复位 (LVR)

为了监控器件的工作电压，单片机提供低电压复位电路。如果器件的工作电压在  $0.9V \sim V_{LVR}$  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 具有下列功能说明：

- 低电压 ( $0.9$  伏  $\sim V_{LVR}$  伏) 的状态必须持续  $1ms$  以上。如果低电压的状态没持续  $1ms$  以上，那么 LVR 会忽视它而不去执行复位功能。
- LVR 通过与  $\overline{RES}$  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

注意：

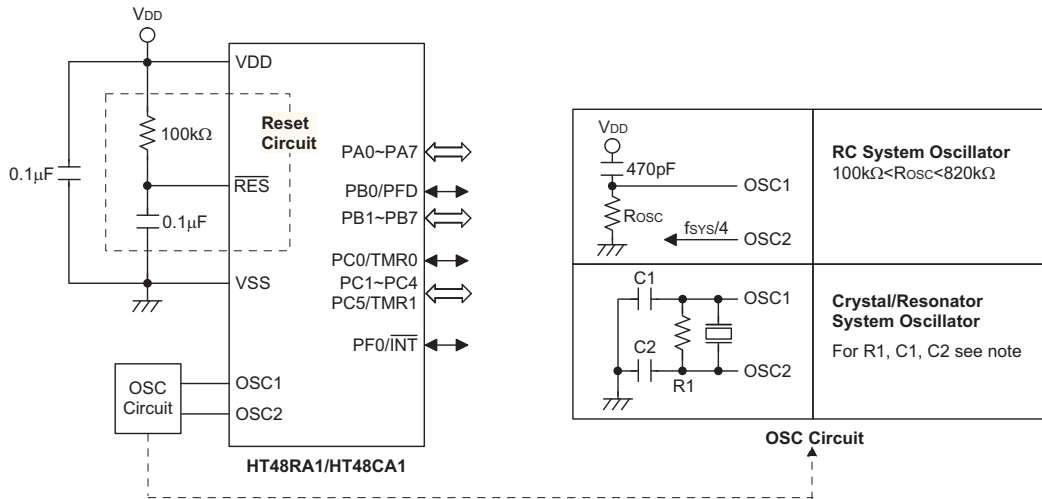
- \*1: 要保证系统振荡器起振并稳定运行，在系统进入正常运行以前，SST 提供额 外的 1024 个系统时钟周期的延迟。
- \*2: 因为低电压状态必须保持  $1ms$  以上，因此进入复位模式就要有  $1ms$  的延迟。

**掩膜选项**

下表示出了这种单片机的各种类型的掩膜选项。所有的掩膜选项必须正确定义。

选	项
PA0~PA7 唤醒选项：	有/没有
PC 端口上拉电阻：	有/没有
PA 端口上拉电阻：	有/没有，按字节单位选择
PF 端口上拉电阻：	有/没有
PB (PB0~PB3) (PB4~PB7) 上拉电阻：	有/没有，按半字节单位选择
PB0 或者 PFD	
CLR WDT 指令	
系统振荡器：	外部 RC 振荡/外部晶体振荡
WDT :	打开或关闭
WDT 时钟来源：	WDTOSC 或者 Fsys/4
LVR:	打开/关闭
LVR 电压：	2.0V/3.0V

应用电路



注：1. 晶体/陶瓷谐振器为系统振荡器

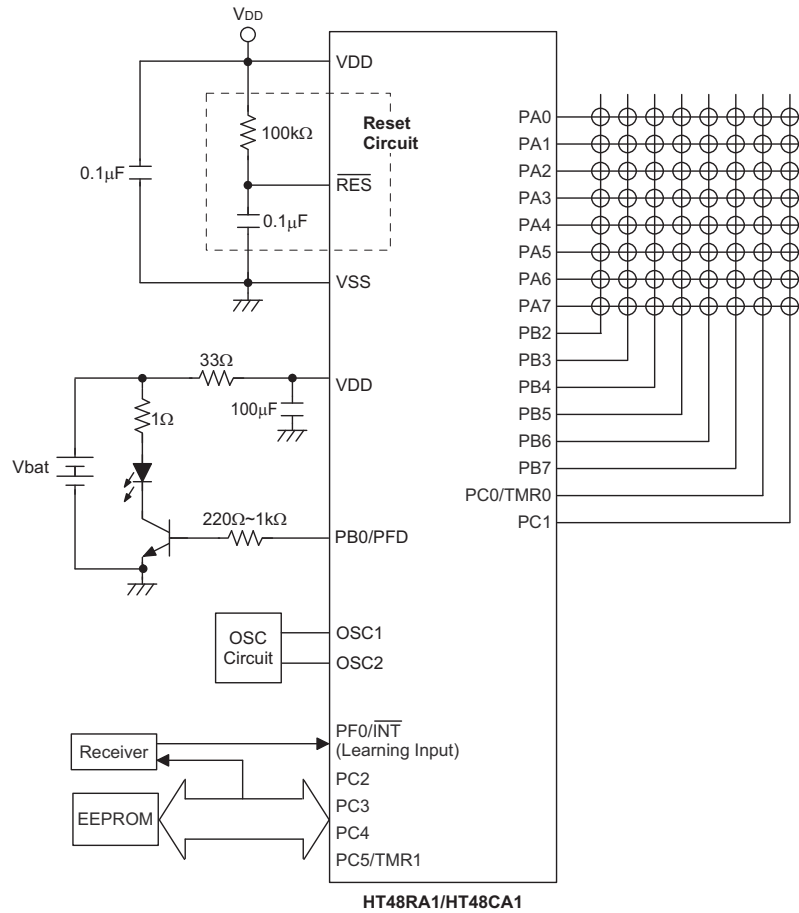
以晶体振荡器而言，仅有部分晶体是需要 C1 与 C2 来校准其振荡精度。而对于陶瓷谐振器来说，基本上都需要 C1 与 C2 来确保可以正常起振。在多数方案中，R1 没有必要使用。当 LVR 功能没有启用，如果要求当电压低于工作电压时晶体必须要停振，就有必要加上电阻 R1。C1 与 C2 的确切数据可以根据晶体/陶瓷谐振器的规格说明来选定。

2. 复位电路

复位电路的电阻及电容值的选取应确保工作电压 VDD 在 RES 引脚上到高之前能稳定，并将数值维持在正常允许的数据之内。为了防止噪声干扰，RES 脚的引线应尽可能地越短越好。

3. 对于应用中从复位电路进入的噪声干扰处理及振荡器外部器件的细节，可以参看应用范例 HA0075S。

图例



## 指令集介绍

### 指令集

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，提供了丰富且易变通的指令，共超过六十条，程序设计师可以事半功倍地实现他们的应用。

为了更加容易了解各式各样的指令码，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 $\mu$ s 中执行完成，而分支或调用操作则将在 1 $\mu$ 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”。对于跳转命令必须注意的是，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接收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。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**

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规划尤其有用，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设定一块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。

**指令集摘要**

下列表格是按照指令功能来分类描述的，可作为基本指令的参考，其使用了如下惯例。

表格惯例：

x: 立即数

m: 数据存储器地址

A: 累加器

I: 0~7 号位

Addr: 程序存储器地址

助记符	说明	指令周期	影响标志位
<b>算术运算</b>			
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 <sup>(1)</sup>	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 <sup>(1)</sup>	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 <sup>(1)</sup>	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 <sup>(1)</sup>	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 <sup>(1)</sup>	C
<b>逻辑运算</b>			
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 <sup>(1)</sup>	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 <sup>(1)</sup>	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 <sup>(1)</sup>	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 <sup>(1)</sup>	Z
CPLA [m]	对数据存储器取反, 结果放入 ACC	1	Z
<b>递增和递减</b>			
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 <sup>(1)</sup>	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 <sup>(1)</sup>	Z
<b>移位</b>			
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 <sup>(1)</sup>	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 <sup>(1)</sup>	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 <sup>(1)</sup>	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 <sup>(1)</sup>	C
<b>数据传送</b>			
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 <sup>(1)</sup>	无
MOV A,x	将立即数送至 ACC	1	无
<b>位运算</b>			
CLR [m].i	清除数据存储器的位	1 <sup>(1)</sup>	无
SET [m].i	置位数据存储器的位	1 <sup>(1)</sup>	无

助记符	说明	指令周期	影响标志位
<b>转移</b>			
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 <sup>(2)</sup>	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 <sup>(2)</sup>	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 <sup>(2)</sup>	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 <sup>(2)</sup>	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 <sup>(3)</sup>	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 <sup>(3)</sup>	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 <sup>(2)</sup>	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 <sup>(2)</sup>	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
<b>查表</b>			
TABRDC [m]	读取当前页的 ROM 内容, 并送至数据存储器 and TBLH	2 <sup>(1)</sup>	无
TABRDL [m]	读取最后页的 ROM 内容, 并送至数据存储器 and TBLH	2 <sup>(1)</sup>	无
<b>其它指令</b>			
NOP	空指令	1	无
CLR [m]	清除数据存储器	1 <sup>(1)</sup>	无
SET [m]	置位数据存储器	1 <sup>(1)</sup>	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 <sup>(4)</sup> ,PDF <sup>(4)</sup>
CLR WDT2	预清除看门狗定时器	1	TO <sup>(4)</sup> ,PDF <sup>(4)</sup>
SWAP [m]	交换数据存储器的高低字节, 结果放入数据存储器	1 <sup>(1)</sup>	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,PDF

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 PDF 标志位; 否则不会影响 TO 和 PDF 标志位。

## 指令描述

- 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： $ACC \leftarrow ACC + [m] + C$   
 影响标志位 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： $[m] \leftarrow ACC + [m] + C$   
 影响标志位 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： $ACC \leftarrow ACC + [m]$   
 影响标志位 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： $ACC \leftarrow ACC + x$   
 影响标志位 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： $[m] \leftarrow ACC + [m]$   
 影响标志位 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： $ACC \leftarrow ACC \text{ “AND” } [m]$   
 影响标志位 Z
- 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： $ACC \leftarrow ACC \text{ “AND” } x$   
 影响标志位 Z
- 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： $[m] \leftarrow ACC \text{ “AND” } [m]$   
 影响标志位 Z

<b>CALL</b>	<b>addr</b>	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		没有
<b>CLR</b>	<b>[m]</b>	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		没有
<b>CLR</b>	<b>[m].i</b>	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		没有
<b>CLR</b>	<b>WDT</b>	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(PDF)和看门狗溢出标志位(TO)也被清零。
运算过程:		WDT ← 00H PDF&TO ← 0
影响标志位		TO, PDF
<b>CLR</b>	<b>WDT1</b>	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(PDF)和计数溢出位(TO)清零, PDF 与 TO 保留原状态不变。
运算过程:		WDT ← 00H PDF&TO ← 0
影响标志位		TO, PDF
<b>CLR</b>	<b>WDT2</b>	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(PDF)和计数溢出位(TO)清零, PDF 与 TO 保留原状态不变。
运算过程:		WDT ← 00H PDF&TO ← 0
影响标志位		TO, PDF
<b>CPL</b>	<b>[m]</b>	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		Z

<b>CPLA</b>	[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:		$ACC \leftarrow [\bar{m}]$
影响标志位		Z
<b>DAA</b>	[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”；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”。本质上，就是基于累加器的值和标志位的基础上，通过对 BCD 加 00H，06H，60H 或 66H 进行十进制调整。只有进位标志位(C)受该指令的影响，它标志原 BCD 码之和是否大于 100，它保证了通过十进制数进行乘法计算的精确度。
操作		[m] $\leftarrow$ (ACC+00H)或 [m] $\leftarrow$ (ACC+06H)或 [m] $\leftarrow$ (ACC+60H)或 [m] $\leftarrow$ (ACC+66H)或
影响标志位		C
<b>DEC</b>	[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] $\leftarrow$ [m]-1
影响标志位		Z
<b>DECA</b>	[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:		$ACC \leftarrow [m]-1$
影响标志位		Z
<b>HALT</b>		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(PDF)被设为 1，WDT 计数溢出位(TO)被清为 0。
运算过程:		TO $\leftarrow$ 0 PDF $\leftarrow$ 1
影响标志位		TO, PDF
<b>INC</b>	[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] $\leftarrow$ [m]+1
影响标志位		Z
<b>INCA</b>	[m]	数据存储器的内容加 1，结果放入累加器
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:		$ACC \leftarrow [m]+1$
影响标志位		Z

<b>JMP</b>	<b>addr</b>	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:		$PC \leftarrow \text{addr}$
影响标志位		没有
<b>MOV</b>	<b>A, [m]</b>	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:		$ACC \leftarrow [m]$
影响标志位		没有
<b>MOV</b>	<b>A, x</b>	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:		$ACC \leftarrow x$
影响标志位		没有
<b>MOV</b>	<b>[m], A</b>	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:		$[m] \leftarrow ACC$
影响标志位		没有
<b>NOP</b>		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:		$PC \leftarrow PC+1$
影响标志位		没有
<b>OR</b>	<b>A, [m]</b>	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:		$ACC \leftarrow ACC \text{ "OR" } [m]$
影响标志位		Z
<b>OR</b>	<b>A, x</b>	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:		$ACC \leftarrow ACC \text{ "OR" } x$
影响标志位		Z
<b>ORM</b>	<b>A, [m]</b>	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:		$[m] \leftarrow ACC \text{ "OR" } [m]$
影响标志位		Z
<b>RET</b>		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:		$PC \leftarrow \text{Stack}$
影响标志位		没有

<b>RET</b>	<b>A, x</b>	<p>从子程序返回，并将立即数放入累加器</p> <p>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。</p> <p>运算过程：  <math>PC \leftarrow Stack</math>  <math>ACC \leftarrow x</math></p> <p>影响标志位 没有</p>
<b>RETI</b>		<p>从中断返回</p> <p>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。</p> <p>运算过程：  <math>PC \leftarrow Stack</math>  <math>EMI \leftarrow 1</math></p> <p>影响标志位 没有</p>
<b>RL</b>	<b>[m]</b>	<p>数据存储器左移一位，结果放入数据存储器</p> <p>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。</p> <p>运算过程：  <math>[m].(i+1) \leftarrow [m].i; (i=0\sim 6)</math>  <math>[m].0 \leftarrow [m].7</math></p> <p>影响标志位 没有</p>
<b>RLA</b>	<b>[m]</b>	<p>数据存储器左移一位，结果放入累加器</p> <p>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。</p> <p>运算过程：  <math>ACC.(i+1) \leftarrow [m].i; (i=0\sim 6)</math>  <math>ACC.0 \leftarrow [m].7</math></p> <p>影响标志位 没有</p>
<b>RLC</b>	<b>[m]</b>	<p>带进位将数据存储器左移一位，结果放入数据存储器</p> <p>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。</p> <p>运算过程：  <math>[m].(i+1) \leftarrow [m].i; (i=0\sim 6)</math>  <math>[m].0 \leftarrow C</math>  <math>C \leftarrow [m].7</math></p> <p>影响标志位 C</p>
<b>RLCA</b>	<b>[m]</b>	<p>带进位将数据存储器左移一位，结果放入累加器</p> <p>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。</p> <p>运算过程：  <math>ACC.(i+1) \leftarrow [m].i; (i=0\sim 6)</math>  <math>ACC.0 \leftarrow C</math>  <math>C \leftarrow [m].7</math></p> <p>影响标志位 C</p>

<b>RR</b>	<b>[m]</b>	<p>数据存储器右移一位，结果放入数据存储器</p> <p>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。</p> <p>运算过程：<math>[m].i \leftarrow [m].(i+1); (i=0\sim 6)</math>  <math>[m].7 \leftarrow [m].0</math></p> <p>影响标志位 没有</p>
<b>RRA</b>	<b>[m]</b>	<p>数据存储器右移一位，结果放入累加器</p> <p>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。</p> <p>运算过程：<math>ACC.i \leftarrow [m].(i+1); (i=0\sim 6)</math>  <math>ACC.7 \leftarrow [m].0</math></p> <p>影响标志位 没有</p>
<b>RRC</b>	<b>[m]</b>	<p>带进位将数据存储器右移一位，结果放入数据存储器</p> <p>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。</p> <p>运算过程：<math>[m].i \leftarrow [m].(i+1); (i=0\sim 6)</math>  <math>[m].7 \leftarrow C</math>  <math>C \leftarrow [m].0</math></p> <p>影响标志位 C</p>
<b>RRCA</b>	<b>[m]</b>	<p>带进位将数据存储器右移一位，结果放入累加器</p> <p>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。</p> <p>运算过程：<math>ACC.i \leftarrow [m].(i+1); (i=0\sim 6)</math>  <math>ACC.7 \leftarrow C</math>  <math>C \leftarrow [m].0</math></p> <p>影响标志位 C</p>
<b>SBC</b>	<b>A,[m]</b>	<p>累加器与数据存储器、进位标志相减，结果放入累加器</p> <p>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。</p> <p>运算过程：<math>ACC \leftarrow ACC - [m] - \bar{C}</math></p> <p>影响标志位 OV, Z, AC, C</p>
<b>SBCM</b>	<b>A,[m]</b>	<p>累加器与数据存储器、进位标志相减，结果放入数据存储器</p> <p>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。</p> <p>运算过程：<math>[m] \leftarrow ACC - [m] - \bar{C}</math></p> <p>影响标志位 OV, Z, AC, C</p>
<b>SDZ</b>	<b>[m]</b>	<p>数据存储器减 1，如果结果为“0”，则跳过下一条指令</p> <p>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)。</p> <p>运算过程：<math>[m] \leftarrow [m] - 1</math>          如果 <math>[m]=0</math>，跳过下一条指令执行再下一条。</p> <p>影响标志位 没有</p>

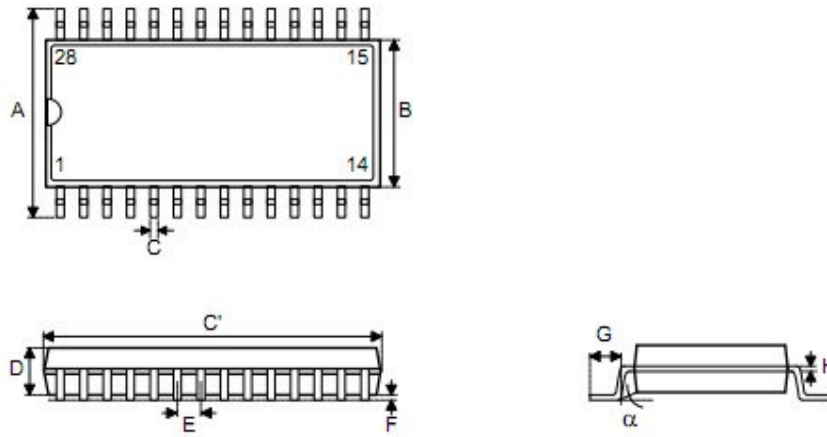
<b>SDZA</b>	<b>[m]</b>	<p>数据存储器减 1，将结果放入累加器，如果结果为“0”，则跳过下一条指令</p> <p>说明：本指令是把数据存储器内的数值减 1，判断是否为 0，为 0 则跳过下一行指令并将减完后数据存储器内的数值送到累加器，而数据存储器内的值不变，即若结果为 0，放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以取得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。</p> <p>运算过程：<math>ACC \leftarrow [m]-1</math></p> <p>如果 <math>ACC = 0</math>，跳过下一条指令执行再下一条。</p> <p>影响标志位 没有</p>
<b>SET</b>	<b>[m]</b>	<p>置位数据存储器</p> <p>说明：本指令是把存储器内的数值每个位置为 1。</p> <p>运算过程：<math>[m] \leftarrow FFH</math></p> <p>影响标志位 没有</p>
<b>SET</b>	<b>[m]. i</b>	<p>将数据存储器的第 i 位置“1”</p> <p>说明：本指令是把存储器内的数值的第 i 位置为 1。</p> <p>运算过程：<math>[m].i \leftarrow 1</math></p> <p>影响标志位 没有</p>
<b>SIZ</b>	<b>[m]</b>	<p>数据存储器加 1，如果结果为“0”，则跳过下一条指令</p> <p>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)。</p> <p>运算过程：<math>[m] \leftarrow [m]+1</math></p> <p>如果 <math>[m]=0</math>，跳过下一行指令</p> <p>影响标志位 没有</p>
<b>SIZA</b>		<p>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</p> <p>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)。</p> <p>运算过程：<math>ACC \leftarrow [m]+1</math></p> <p>如果 <math>ACC = 0</math>，跳过下一行指令</p> <p>影响标志位 没有</p>
<b>SNZ</b>	<b>[m]. i</b>	<p>如果数据存储器的第 i 位不为“0”，则跳过下一条指令</p> <p>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)。</p> <p>运算过程：如果 <math>[m].i \neq 0</math>，跳过下一行指令。</p> <p>影响标志位 没有</p>

<b>SUB</b>	<b>A, [m]</b>	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:		$ACC \leftarrow ACC - [m]$
影响标志位		OV, Z, AC, C
<b>SUB</b>	<b>A, x</b>	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:		$ACC \leftarrow ACC - x$
影响标志位		OV, Z, AC, C
<b>SUBM</b>	<b>A, [m]</b>	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:		$[m] \leftarrow ACC - [m]$
影响标志位		OV, Z, AC, C
<b>SWAP</b>	<b>[m]</b>	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:		$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
影响标志位		没有
<b>SWAPA</b>	<b>[m]</b>	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:		$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
影响标志位		没有
<b>SZ</b>	<b>[m]</b>	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		没有
<b>SZA</b>	<b>[m]</b>	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 $ACC \leftarrow [m]$ 。
影响标志位		没有

<b>SZ</b>	<b>[m]. i</b>	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		没有
<b>TABRDC</b>	<b>[m]</b>	读取 ROM 当前页的内容，并送至数据存储器 and 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] ← 程序存储器低字节 TBLH ← 程序存储器高字节
影响标志位		没有
<b>TABRDL</b>	<b>[m]</b>	读取 ROM 最后一页的内容，并送至数据存储器 and 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] ← 程序存储器低字节 TBLH ← 程序存储器高字节
影响标志位		没有
<b>XOR</b>	<b>A, [m]</b>	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		Z
<b>XORM</b>	<b>A, [m]</b>	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		Z
<b>XOR</b>	<b>A, x</b>	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		Z

封装信息

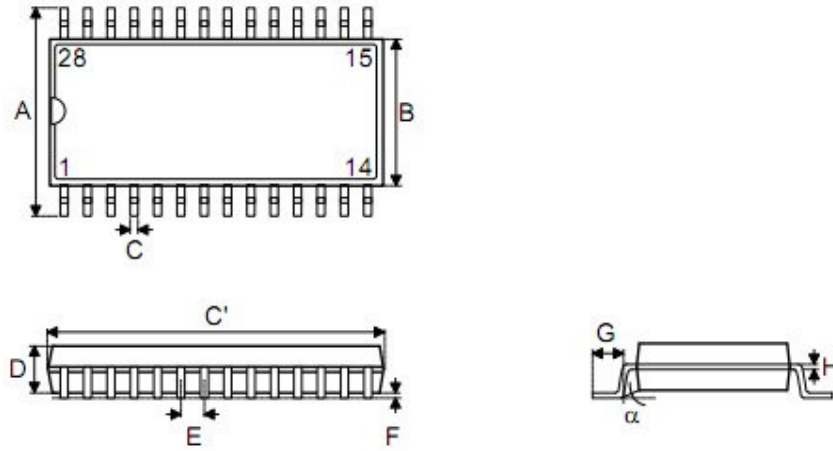
28-pin SOP (300mil)外形尺寸



• MS-013

标号	尺寸 (mil)		
	Min	Nom	Max
A	393	--	419
B	256	--	300
C	12	--	20
C'	697	--	713
D	--	--	104
E	--	50	--
F	4	--	12
G	16	--	50
H	8	--	13
α	0°	--	8°

28pin SSOP (209mil)外形尺寸

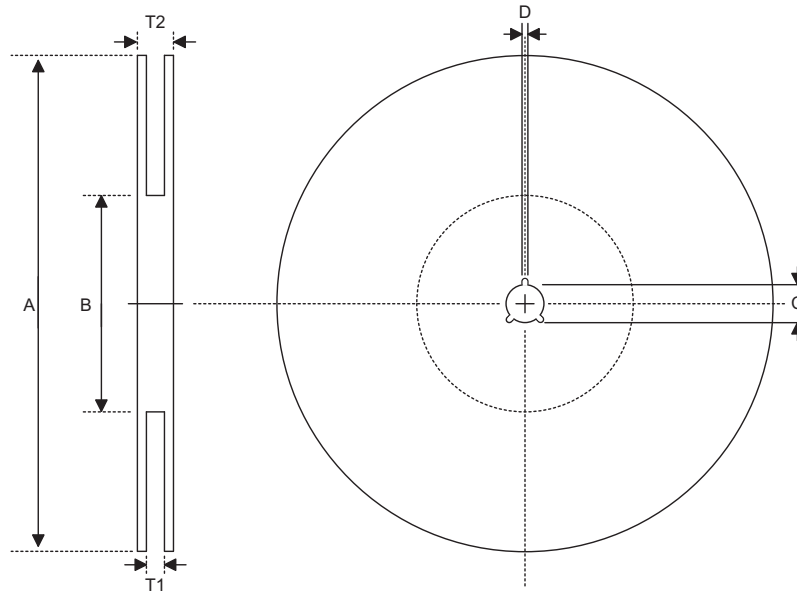


• MS-150

标号	尺寸 (mil)		
	Min	Nom	Max
A	7.40	--	8.20
B	5.00	--	5.60
C	0.22	--	0.33
C'	9.90	--	10.50
D	--	--	2.00
E	--	0.65	--
F	0.05	--	--
G	0.55	--	0.95
H	0.09	--	0.21
$\alpha$	0°	--	8°

包装带和卷轴规格:

卷轴尺寸:



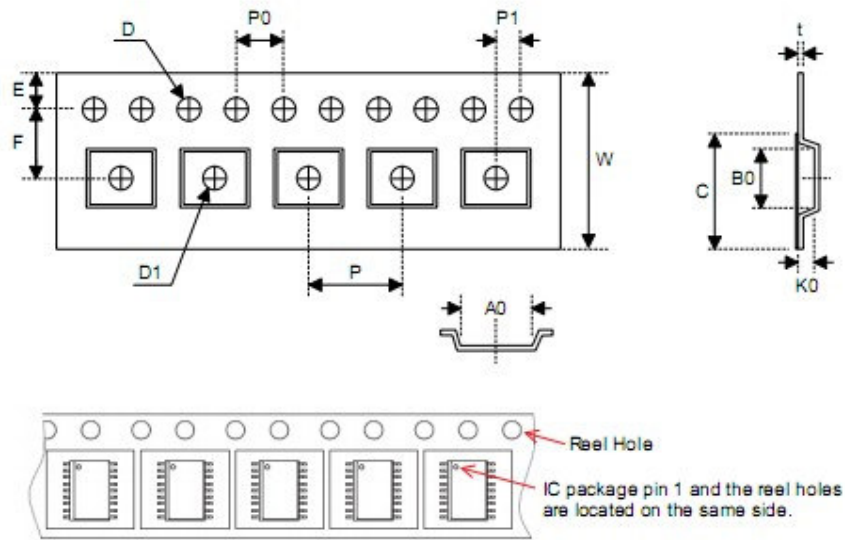
SOP 28W(300mil)

标号	描述	尺寸(mm)
A	卷轴外圈直径	330±1.0
B	卷轴内圈直径	100±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

SSOP 28S(209mil)

标号	描述	尺寸(mm)
A	卷轴外圈直径	330±1.0
B	卷轴内圈直径	100±1.5
C	轴心直径	13.0+0.5 -0.2
D	缝宽	2.0±0.5
T1	轮缘宽	28.4+0.3 -0.2
T2	卷轴宽	31.1(max.)

运输带尺寸:



SOP 28W(300mil)

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	1.5+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	10.85±0.1
B0	空穴宽	18.34±0.1
K0	空穴深	2.97±0.1
t	传输带厚度	0.35±0.01
C	覆盖带宽度	21.3±0.1

SSOP 28S(209mil)

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	1.5+0.1
D1	空穴中之小孔直径	1.5+0.25
P0	穿孔间距	4.0±0.2
P1	空穴至穿孔距离(长度)	2.0±0.1
A0	空穴长	8.4±0.1
B0	空穴宽	10.65±0.1
K0	空穴深	2.4±0.1
t	传输带厚度	0.30±0.05
C	覆盖带宽度	21.3±0.1

**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](http://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）**

上海宜山路2016号1号楼3楼G室 201103  
电话: 021-5422-4590  
传真: 021-5422-4705  
网站: [www.holtek.com.cn](http://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

**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](http://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>