

## 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](#)
  - [HA0055S 2<sup>12</sup> Decoder \(8+4, 对应 HT12E\)](#)

## 特性

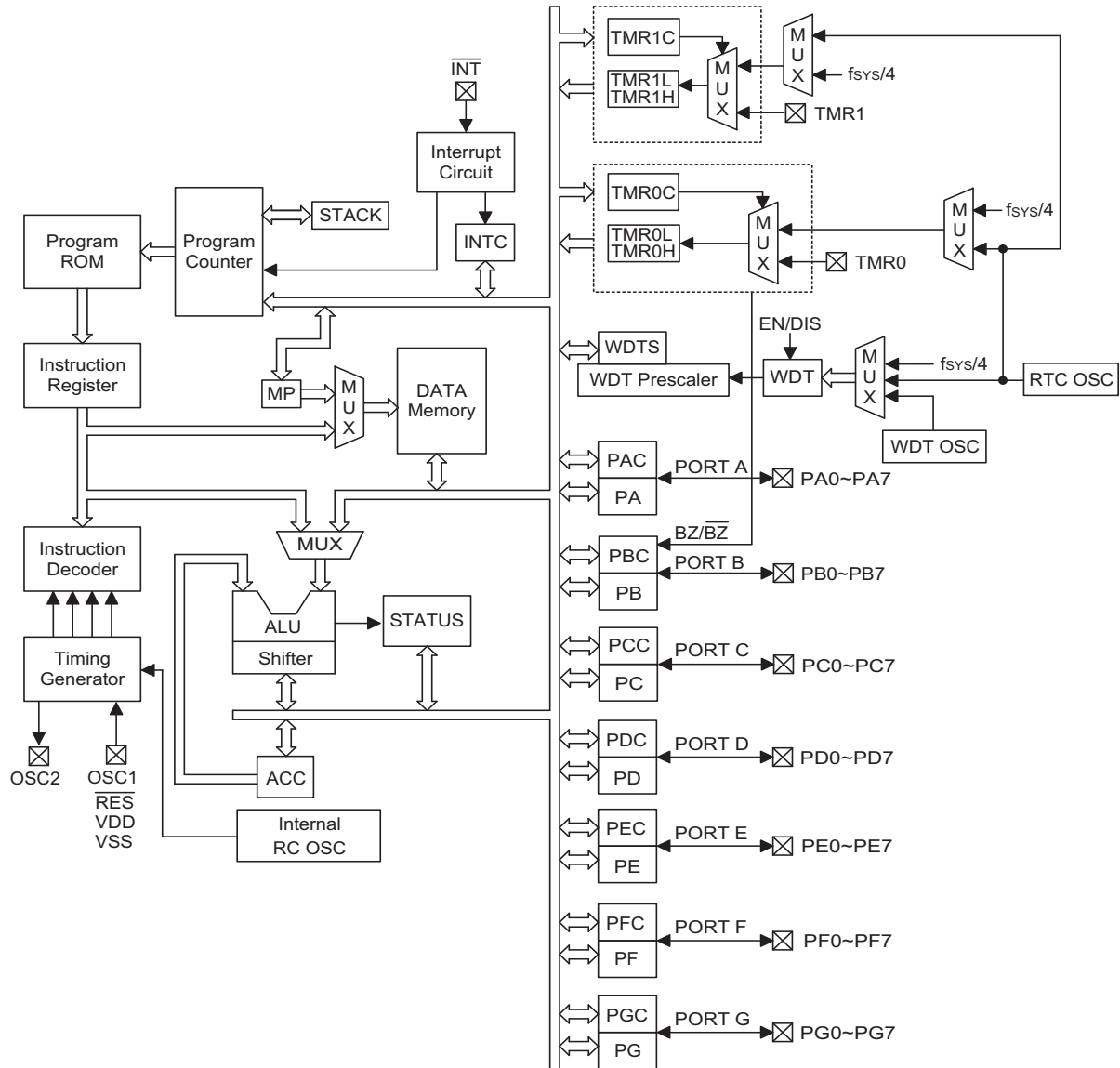
- 工作电压:
  - fsys = 4MHz: 2.2V~5.5V
  - fsys = 8MHz: 3.3V~5.5V
- 低电压复位功能
- 最多有 56 个双向输入/输出端口
- 1 个外部中断输入端口
- 2 个 16 位可编程定时/计数器, 具有溢出中断
- 内置晶体和 RC 振荡电路、内置 RC 振荡
- 可接 32768Hz 晶振用于计时
- 看门狗定时器
- 4096×15 位的程序存储器 ROM
- 224×8 数据存储器 RAM
- 一对蜂鸣器驱动并支持 PFD
- HALT 和唤醒功能来降低功耗
- 16 层硬件堆栈
- 在 VDD=5V, 系统频率为 8MHz 时, 指令周期为 0.5μs
- 位操作指令
- 查表指令, 表格内容字长 15 位
- 63 条指令
- 所有指令在 1 个或 2 个指令周期内完成
- 48-pin SSOP, 64-pin LQFP 的封装

## 概述

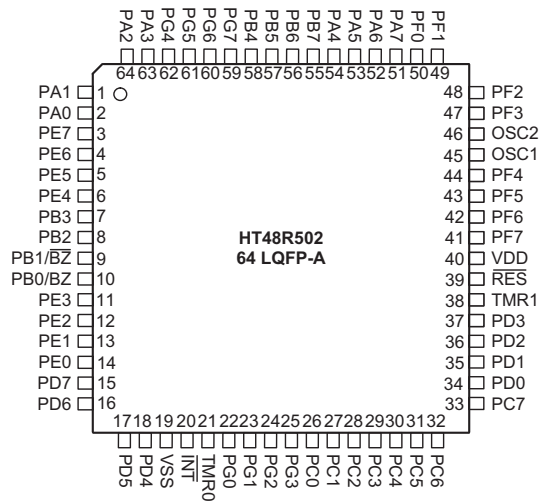
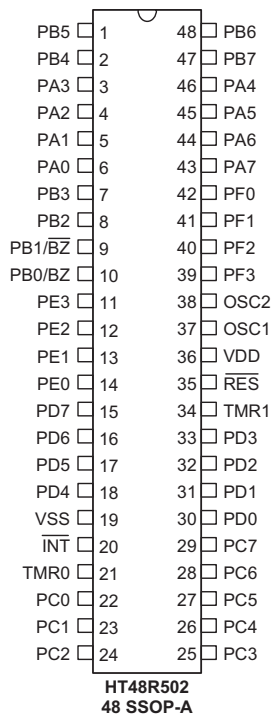
HT48R502 是一款 8 位高性能精简指令集单片机, 专为多输入/输出控制的产品设计。

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



引脚排列



## 引脚说明

引脚名称	输入/输出	掩膜选项	说明
PA0~PA7	输入/输出	唤醒功能 上拉电阻* CMOS/斯密特触发输入	8 位双向输入/输出口 每一位能由掩膜选项设置为唤醒输入。软件指令设置为 CMOS 输出或斯密特触发输入，作为 CMOS 输入时可由掩膜选项设置是否带上拉电阻（由上拉电阻选项决定）
PB0/BZ PB1/ $\overline{BZ}$ PB2~PB7	输入/输出	上拉电阻* 输出\输入或者 BZ/ $\overline{BZ}$	8 位双向输入/输出口 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（由上拉电阻选项决定） PB0 和 PB1 是与 BZ 和 $\overline{BZ}$ 共用引脚。一旦 PB0 和 PB1 选为驱动蜂鸣器输出，它的输出信号则由内部的 PFD 发生器（由定时/计数器 0 编程决定）提供
V <sub>SS</sub>	—	—	负电源，接地。
$\overline{INT}$	输入	—	外部中断斯密特触发输入无上拉电阻，在下降沿时被触发
TMR0	输入	—	定时/计数器 0 斯密特触发输入
TMR1	输入	—	定时/计数器 1 斯密特触发输入
PC0~PC7	输入/输出	上拉电阻*	8 位双向输入/输出口 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（由上拉电阻选项决定）
$\overline{RES}$	输入	—	斯密特触发复位输入端，低电平有效
V <sub>DD</sub>	—	—	正电源
OSC1 OSC2	输入 输出	晶体振荡/ RC 振荡/ RTC 振荡	OSC1 和 OSC2 连接至 RC 网络或晶体振荡（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；这两个引脚也可以通过掩膜来选择作为 RTC 振荡器（32768 Hz），这种情况下，系统时钟由内部的 RC 振荡器提供，频率可有四种选择（3.2 MHz，1.6 MHz，800 kHz，400 kHz）
PD0~PD7	输入/输出	上拉电阻*	8 位双向输入/输出口 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（由上拉电阻选项决定）
PE0~PE7	输入/输出	上拉电阻*	8 位双向输入/输出口 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（由上拉电阻选项决定）
PF0~PF7	输入/输出	上拉电阻*	8 位双向输入/输出口 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（由上拉电阻选项决定）
PG0~PG7	输入/输出	上拉电阻*	8 位双向输入/输出口 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（由上拉电阻选项决定）

**注意：** “\*” 所有输入/输出(PA、PB、PC、PD、PE、PF、PG)的上拉电阻由一个选择位控制。

PA 端口的 CMOS 或斯密特触发选择也是以整个端口(8 位)为单位的。

## 极限参数

电源提供电压	.....	$V_{SS}-0.3V \sim V_{SS}+6.0V$	储存温度	.....	$-50^{\circ}C \sim 125^{\circ}C$
端口输入电压	.....	$V_{SS}-0.3V \sim V_{DD}+0.3V$	工作温度	.....	$-40^{\circ}C \sim 85^{\circ}C$
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。

## 直流电气特性

 $T_a=25^{\circ}C$ 

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
		—	$F_{sys}=8MHz$	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	
$I_{DD2}$	工作电流 (RC 振荡)	3V	无负载	—	0.8	1.5	mA
		5V	$f_{sys}=4MHz$	—	2.5	4	
$I_{DD3}$	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 (看门狗使能, RTC 关闭)	3V	无负载	—	—	5	$\mu A$
		5V	暂停模式	—	—	10	
$I_{STB2}$	静态电流 (看门狗禁止, RTC 关闭)	3V	无负载	—	—	1	$\mu A$
		5V	暂停模式	—	—	2	
$I_{STB3}$	静态电流 (看门狗禁止, RTC 打开)	3V	无负载	—	—	5	$\mu A$
		5V	暂停模式	—	—	10	
$V_{IL1}$	输入/输出口低电平输入电压	—	—	0	—	$0.3V_{DD}$	V
$V_{IH1}$	输入/输出口高电平输入电压	—	—	$0.7V_{DD}$	—	$V_{DD}$	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
$V_{LVR}$	低电压复位	—	LVR 打开	2.7	3.0	3.3	V
$I_{OL}$	输入/输出口灌电流	3V	$V_{OL}=0.1V_{DD}$	4	8	—	mA
		5V	$V_{OL}=0.1V_{DD}$	10	20	—	
$I_{OH}$	输入/输出口源电流	3V	$V_{OH}=0.9V_{DD}$	-2	-4	—	mA
		5V	$V_{OH}=0.9V_{DD}$	-5	-10	—	
$R_{PH}$	上拉电阻	3V	—	20	60	100	K $\Omega$
		5V	—	10	30	50	

**交流电气特性**

Ta=25°C

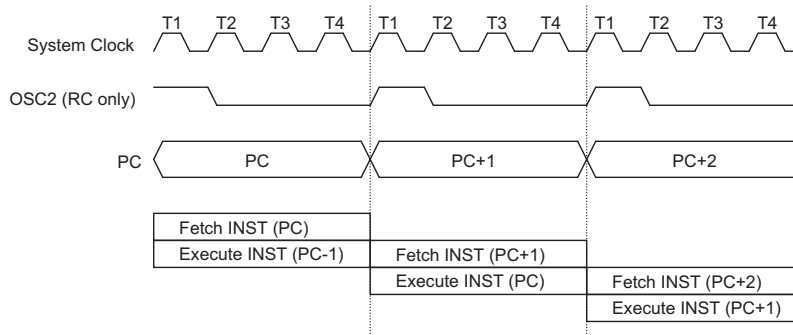
符号	参数	测试条件		最小	典型	最大	单位
		V <sub>DD</sub>	条件				
f <sub>SYS1</sub>	系统时钟（晶体振荡/RC振荡）	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	
f <sub>SYS2</sub>	系统时钟(内部 RC 振荡)	5V	3.2MHz	1800	—	5400	kHz
			1.6MHz	900	—	2700	
			800KHz	450	—	1350	
			400KHz	225	—	675	
f <sub>TIMER</sub>	定时/计数器输入频率（TMR）	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	
t <sub>WDTOSC</sub>	看门狗振荡器周期	3V	—	45	90	180	μ s
		5V	—	32	65	130	
t <sub>WDT1</sub>	看门狗溢出周期(WDT 振荡)	3V	WDT 无预分频	11	23	46	m s
		5V		8	17	33	
t <sub>WDT2</sub>	看门狗溢出周期(系统时钟)	—	WDT 无预分频	—	1024	—	t <sub>SYS</sub>
t <sub>WDT3</sub>	看门狗定时周期(RTC 振荡)	—	WDT 无预分频	—	7.812	—	m s
t <sub>RES</sub>	外部复位低电平脉冲宽度	—	—	1	—	—	μ s
t <sub>SST</sub>	系统启动延时周期	—	从 HALT 唤醒	—	1024	—	t <sub>SYS</sub>
t <sub>INT</sub>	中断脉冲宽度	—	—	1	—	—	μ s
T <sub>LVR</sub>	低电压复位宽度	—	—	0.25	1	2	ms

 注：t<sub>SYS</sub> = 1/f<sub>SYS</sub>

## 系统功能说明

### 指令执行时序

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。



指令执行时序

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。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。

### 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、条件跳转、读取寄存器的值、子程序调用、子程序返回、初始复位、内部或外部中断响应、中断返回等，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的指令周期（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，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，这个短程跳跃的地址范围为 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											
	*11	*10	*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	0	0
外部中断	0	0	0	0	0	0	0	0	0	1	0	0
定时/计数器 0 溢出	0	0	0	0	0	0	0	0	1	0	0	0
定时/计数器 1 溢出	0	0	0	0	0	0	0	0	1	1	0	0
条件跳转	PC+2											
装载 PCL	*11	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
跳转，子程序调用	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
从子程序返回	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

程序计数器

注意：\*11 ~ \*0：程序计数器位  
#11 ~ #0：指令代码位

S11 ~ S0：堆栈寄存器位  
@7 ~ @0：PCL 位

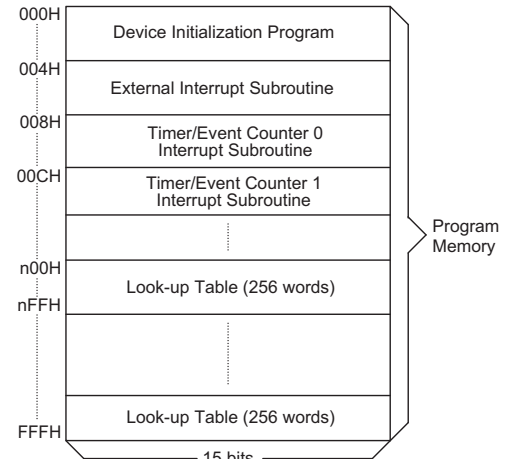
**程序存储器 (Program Memory — ROM)**

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，有 4096×15 位组成，程序存储器空间可以用程序计数器或表格指针进行寻址。

程序寄存器中的某些地址是为一些特殊使用而保留的：

- 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。
- 地址 008H  
此地址保留给定时/计数器 0 中断服务使用。当中断是开放的，且堆栈又未滿，一旦定时/计数器发生溢出时，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。当中断允许，堆栈未滿，一旦定时/计数器发生溢出时，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)

程序存储器内的任何地址都可被用来作为查表地址使用。指令 TABRDC [m] (查表当前页的数据，1 页=256 字) 和 TABRDL [m] (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指定的数据寄存器  $[m]$ ，而这个字的高位字节传送到 TBLH (08H)。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。



程序存储器

指令	表格地址											
	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
TABRDC[m]	P11	P10	P9	P8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL[m]	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

表格区

注意：\*11 ~ \*0 : 表格地址位                      P11 ~ P8 : 当前程序计数器位  
 @7 ~ @0 : 表格指针位



## 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) 的宽度为 8 位, 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。该寄存器用来记录状态信息, 和控制操作流程。

除了 TO 和 PDF 标志位以外, 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。另外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, 执行清除看门狗定时器或 HALT 指令都将改变标志位 TO, 执行 HALT 指令, 清除看门狗定时器或系统上电将影响 PDF 标志位。

Z, OV, AC 和 C 标志位都反映了最近一次的操作状态。

另外, 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, 那么 OV 被置位; 反之, OV 被清除。
PDF	4	系统上电或执行 CLR WDT 指令, PDF 被清除。 执行 HALT 指令 PDF 被置位。
TO	5	系统上电, 执行 CLR WDT 指令或 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

## 状态寄存器 (0AH)

## 中断

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，用来设置中断使能/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，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 位将会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 的第五位）初始化，中断的请求是由定时器溢出产生。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。

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	ET0I	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, 读出为零	

中断控制寄存器—INTC（0BH）

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 的第六位）初始化，中断的请求是由定时器溢出产生。当中断允许，堆栈未滿，并且 T1F 已被置位时，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禁止，直到执行 RETI 指令或是 EMI 位和相关中断控制位都被置为 1（假如堆栈未滿）。若要从中断子程序返回时，调用 RET 或 RETI 指令即可。RETI 指令将会置位 EMI 来允许中断服务，而 RET 不能。

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，必须遵循下列表的中断服务优先等级。这些优先级也可以通过 EMI 位的复位来屏蔽。

NO	中断源	优先级	中断矢量
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

中断控制寄存器（INTC）由定时/计数器 0/1 中断请求标志位（TOF/T1F），外部中断请求标志位（EIF），定时/计数器 0/1 使能位（ET0I/ET1I），外部中断使能位（EEI），和主中断控制使能位（EMI）组成。中断控制寄存器位于数据寄存器 0BH 单元。EMI、EEI、ET0I 和 ET1I 都是用来控制中断的使能/禁止状态。这些位防止中断已经响应的中断服务程序。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，直到相关的中断服务程序返回或由软件指令来清除相关的中断请求标志位。

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，而且在许多应用中需要立刻给予响应。基于上述情况，如果只剩下一个堆栈，中断使能也不是很好地被控制，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。

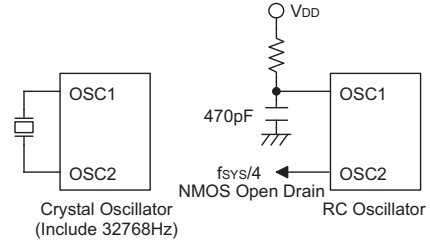
## 振荡电路

HT48R502 有三种振荡电路。这三种振荡电路都是针对系统时钟而设计的，分别是外部 RC 振荡器、外部晶体振荡器以及内部 RC 振荡器，可掩膜选项设置。不管所选的是哪一种振荡器，都可以提供系统时钟。进入暂停模式会使系统时钟停止工作，还可以忽视外部信号的，以降低功耗。

如果使用 RC 振荡器，在 OSC1 和  $V_{DD}$  之间需要一个外部电阻，其阻值范围为  $24k\Omega$  至  $1M\Omega$ 。在 OSC2 端系统时钟四分频，用于同步外部逻辑。RC 型振荡方式是一种低成本的方式，但是，振荡频率会随着  $V_{DD}$ 、温度和芯片自身参数的漂移而变化。因此，在用于非常精确计时的场合，建议不要使用 RC 型振荡器。

如果选用晶体振荡器，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，不再需要其他元器件。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，但是需要在 OSC1 和 OSC2 连接两个电容器接地。如果选用内部的 RC 振荡器，那么 OSC1 和 OSC2 可以选择作为 32768Hz 晶体振荡器 (RTC OSC)。内部的 RC 振荡器的频率可以选择 3.2MHz、1.6MHz、800kHz 和 400kHz（在掩膜时选择）。

WDT 振荡器是芯片内部自带的 RC 型振荡器，不需要任何外部元器件。即使在系统进入暂停模式，系统时钟被停止，但这个 RC 振荡器仍会工作（振荡周期大约为  $65\mu s/5V$ ）。为了节省电源，可在掩膜选项中禁止 WDT 振荡器。



系统振荡器

### 看门狗计时器(Watchdog Timer)

WDT 的时钟源有三种：RC 振荡器（WDT 振荡器）、RTC 振荡器或指令时钟（系统时钟 4 分频），由掩膜选项来决定。看门狗定时器主要用来避免程序运行失常和程序跳入一死循环而导致不可预测的结果。看门狗定时器可用掩膜来设置为禁止。如果在关闭状态，所有与 WDT 相关的指令操作都是没有作用的。只有选择“内部 RC 振荡器+RTC”模式时，实时时钟才是有效。

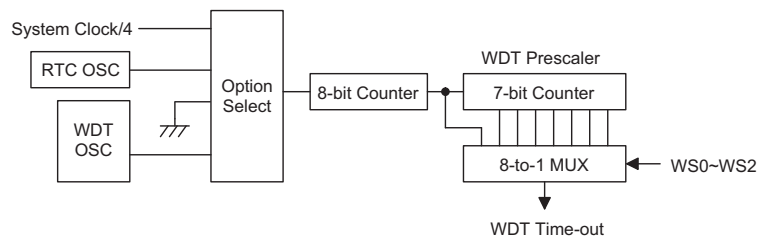
一旦选择了内部 WDT 振荡器（以  $65\mu S/5V$  为周期的 RC 振荡器），首先对时钟频率要进行 256（8 阶）预分频，得到  $17ms/5V$  的溢出周期。溢出周期受温度，VDD 以及芯片自身的参数影响。通过使用 WDT 预分频器，可以得到更长的溢出周期。设置 WS2, WS1, WS0（WDT 的第 2、1、0 位）可以获得不同的溢出周期。如果 WS2, WS1, WS0 都是 1，则分频比为最大 1: 128，溢出周期是  $2.1s/5V$ 。如果 WDT 振荡器禁止，时钟源是指令时钟，只是在 HALT 模式下 WDT 停止，看门狗失去保护的作用，除此以外其余的方面和 WDT 振荡器是一样的。在这种状态下只有通过外部逻辑来复位。WDTS 的高半字节和第 3 位可以由用户自定义标志位。

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

看门狗定时器预分频寄存器

如果单片机工作在干扰很大的环境中，强烈建议使用片内的 RC 振荡器（WDT OSC）或是 32kHz 晶体振荡器（RTC OSC），因为 HALT 模式会使系统时钟停止运作。

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，溢出只产生热复位，只能使 PC 程序计数器和堆栈指针 SP 复位到零。清除 WDT 的值（包括 WDT 预分频值）有外部复位（低电平输入到  $\overline{RES}$  端），软件指令和 HALT 指令三种方法。软件指令由“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”（即 CLR WDT 次数为 1），那么只要执行 CLR WDT 指令就会清除 WDT。在选择 CLR WDT1 和 CLR WDT2 的情况下（即 CLR WDT 次数为 2），那么要交替执行两条指令才会清除 WDT，否则，WDT 会由于溢出而使系统复位。



看门狗定时器

## 暂停模式 (HALT)

暂停模式是由 HALT 指令来实现，有如下结果：

- 系统振荡器关闭，但 WDT 振荡器会继续工作（如果选择 WDT 振荡器）
- RAM 和寄存器内容保持不变
- WDT 和 WDT 预分频器被清除，并重新计数（如果 WDT 的时钟是来自 WDT 振荡器）
- 所有的输入/输出端口都保持其原先状态
- PDF 标志位被置位，TO 标志位被清除

外部复位、中断或 PA 端口下降沿信号或 WDT 溢出均可使系统脱离暂停模式。外部复位能使系统初始化，而 WDT 溢出能执行“热复位”。通过检查 TO 和 PDF 标志，即可了解系统复位的原因。PDF 标志位在系统上电复位和执行 CLR WDT 指令时清 0，执行 HALT 指令时 PDF 置 1。TO 标志位当 WDT 溢出时置 1，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 端口唤醒和中断唤醒可作为正常运行的继续，PA 端口的每一位都可以通过掩膜选项来单独设定为唤醒功能。如果唤醒是来自于输入/输出端口的信号变化，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一条指令；如果中断允许并且堆栈未满，那么这个中断响应就发生了。如果在进入暂停模式前，中断标志位已经被置位 1，则中断唤醒功能被禁止。当唤醒发生时，要 1024 tsys（系统时钟周期）后，系统重新正常运行。换句话说，唤醒后要插入了一个等待周期。如果唤醒是来自于中断响应，那么实际的中断程序执行就要延迟一个以上的周期。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。

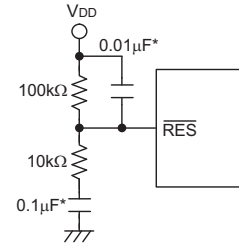
为了减小功耗，在进入 HALT 模式之前必须小心处理输入/输出端口的状态。但在 HALT 模式下 RTC 振荡器仍然运作（如果使能 RTC 振荡器）。

### 复位 (RESET)

有三种方法可以产生复位

- 在正常运行期间由  $\overline{\text{RES}}$  脚产生复位
- 在 HALT 期间由  $\overline{\text{RES}}$  脚产生复位
- 在正常运行期间由 WDT 溢出复位

在 HALT 期间 WDT 溢出复位是不同于其它的复位操作,因为它可执行“热复位”,结果只使程序计数器 PC 和堆栈指针 SP 复位,而别的电路均保持原来的状态。在其他复位条件下,某些寄存器保持不变。当复位条件被满足时,极大多数的寄存器被复位到“初始状态”,通过测试 PDF 标志位和 TO 标志位,程序能分辨不同的系统复位原因。



复位电路

注意：“\*”为了避免噪声干扰,连接  $\overline{\text{RES}}$  引脚的线应尽量短

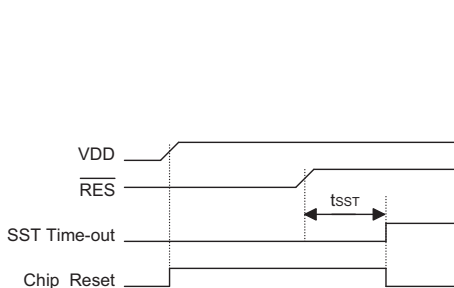
TO	PDF	复位条件
0	0	电源上电复位
u	u	正常运行时由 $\overline{\text{RES}}$ 发生复位
0	1	由 $\overline{\text{RES}}$ 唤醒暂停模式
1	u	正常运作时发生看门狗定时器超时
1	1	由看门狗定时器唤醒暂停模式

注: u 表示不变

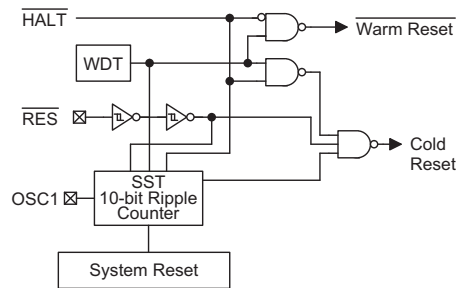
为了保证系统振荡器起振并稳定运行,系统复位(包括上电复位、看门狗定时器溢出或由  $\overline{\text{RES}}$  端复位)或由暂停状态唤醒时,系统启动定时器(SST)提供了一个额外的延迟时间,共 1024 个系统时钟周期。

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。

当系统上电、正常运行时 WDT 溢出或  $\overline{\text{RES}}$  脚复位,系统需要额外增加一个加载 Option 的时间。



复位时序



复位电路结构

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

有关寄存器的状态如下：

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*
Program Counter	000H	000H	000H	000H	000H
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
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu 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
TMR0H	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0L	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMROC	00-0 1---	00-0 1---	00-0 1---	00-0 1---	uu-u u---
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	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PE	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PF	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PG	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PGC	1111 1111	1111 1111	1111 1111	1111 1111	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**

HT48R502 提供两个定时/计数器 (TMR0, TMR1)。定时/计数器 0 包含一个 16 位可编程的加计数的计数器, 并且其时钟来源可以是外部信号输入、系统时钟的四分频或是 RTC 时钟。

定时/计数器 1 包含一个 16 位的可编程的加计数的计数器, 且其时钟来源可来至外部信号输入、系统时钟的四分频或是 RTC 时钟。

如果采用内部时钟源, 那么该单片机有两种参考时基提供给定时/计数器 0。这个内部时钟源来自  $f_{TID}$  (通常选择这个) 或  $f_{RTC}$  (当系统时钟选内部的 RC+RTC 模式时), 由掩膜选项决定。外部时钟输入时, 允许用户计数外部事件, 测量时间间隔或脉冲宽度或产生一个精确的时基信号。

有三个与定时/计数器 0 相关的寄存器, TMR0H (0CH), TMR0L (0DH), TMR0C (0EH)。若写入 TMR0L 只能将数据写入低字节内部缓冲器 (8 bit) 中, 但若写入的是 TMR0H 则可将数据和低字节内部缓冲器的内容分别写入 TMR0H 和 TMR0L 的预置寄存器之中。每一次对 TMR0H 的写操作都会改变定时/计数器 0 预置寄存器的内容。若读取 TMR0H 则将锁存 TMR0H 的内容并将 TMR0L 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 TMR0L, 则只读回低字节内部缓冲器的内容。换言之, 定时/计数器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 TMR0H, 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。TMR0C 是定时/计数器 0 制寄存器, 它可定义一些操作模式, 计数的允许或禁止以及计数的触发沿。

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 =  $f_{INT}/[2 \times (65536-N)]$ 。

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分别写入 TMR1H 和 TMR1L 的预置寄存器之中。每一次对 TMR1H 的写操作都会改变定时/计数器 1 预置寄存器的内容。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, 它可定义一些操作模式, 计数的允许或禁止以及计数的触发边沿。

TOM0, TOM1 (TMR0C), T1M0, T1M1 (TMR1C) 位定义了工作模式。计数器模式用来对外部事件进行计数, 意味着在这种模式下时钟来源来自于外部引脚输入 (TMR0/TMR1)。定时模式时钟源来自于指令时钟或 RTC 时钟 (Timer0/Timer1)。脉宽测量模式可以用来外部信号 (TMR0/TMR1) 的高电平/低电平的宽度。计数基于指令时钟或 RTC 时钟 (Timer0/Timer1)。

无论在定时或计数模式, 一旦定时/计数器 0/1 开始计数, 它将从当前值计数到 FFFFH。当发生溢出时, 计数器根据定时/计数器 0/1 的预置寄存器重置计数器, 并且产生一个中断请求信号 (TOF/T1F; INTC 的 5/6 位)。

符号	位	功能
—	0~2, 5	未用, 读数为 0
TOE	3	定义定时/计数器 TMR0 沿触发方式 (0=上升沿作用, 1=下降沿作用)
TOON	4	使能/禁止定时/计数器(1=使能, 0=禁止)
TOM0 TOM1	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

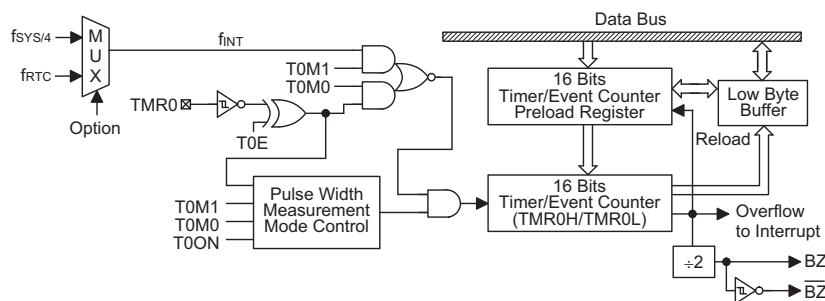
**TMR0C 寄存器 (0EH)**

符号	位	功能
—	0 ~ 2, 5	未定义, 读数为“0”
T1E	3	定义定时/计数器 TMR1 沿触发方式 (0=上升沿作用, 1=下降沿作用)
T1ON	4	使能/禁止定时/计数器(1=使能, 0=禁止)
T1M0 T1M1	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

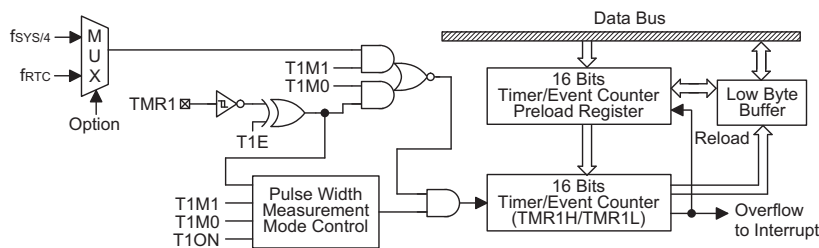
**TMR1C 寄存器 (11H)**

在脉宽测量模式中, 当 T0ON/T1ON 和 T0E/T1E 被置 1, 一旦 TMR0/TMR1 接受到一个上升沿触发(如果 T0E/T1E 位是 0 则是下降沿触发), 则开始计数, 当 TMR0/TMR1 恢复到原来的电平时, 则停止计数。并且复位 T0ON/T1ON。在下次测量开始以前, 测量的结果保存在定时/计数器 0/1 里。换句话说, 一次只能测量一个脉冲宽度。如果要再次测量, 则要把 T0ON/T1ON 位置 1。注意, 在该工作模式下, 不是电平触发而是边沿触发。当计数器溢出, 则计数器 0/1 从定时/计数器 0/1 的预置寄存器中重载初值, 并产生中断请求。要启动计数器, 则定时器 ON 位 (T0ON: TMR0C 的第 4 位; T1ON: TMR1C 的第 4 位) 要被置 1。在脉宽测量模式中, 在每个测量周期结束后 T0ON/T1ON 都会自动被复位。但是在其他两种模式中, T0ON/T1ON 只能通过软件指令来复位。定时/计数器 0/1 溢出中断是唤醒暂停模式源之一。无论在什么工作模式下, 把 ETI0/ETI1 复位都会禁止定时/计数器的中断响应。

在定时/计数器 0/1 为关闭的状态下, 写数据到定时/计数器 0/1 的预置寄存器, 同时也会将数据装入定时/计数器 0/1 中。但若是定时/计数器 0/1 已经运行, 写到定时/计数器 0/1 的数据只会被保留在定时/计数器 0/1 的预置寄存器中, 直到定时/计数器 0/1 发生计数溢出为止, 再由预置寄存器加载新的值。当读取定时/计数器 0/1 (读 TMR0/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。



**定时/计数器 0**



**定时/计数器 1**

输入/输出口

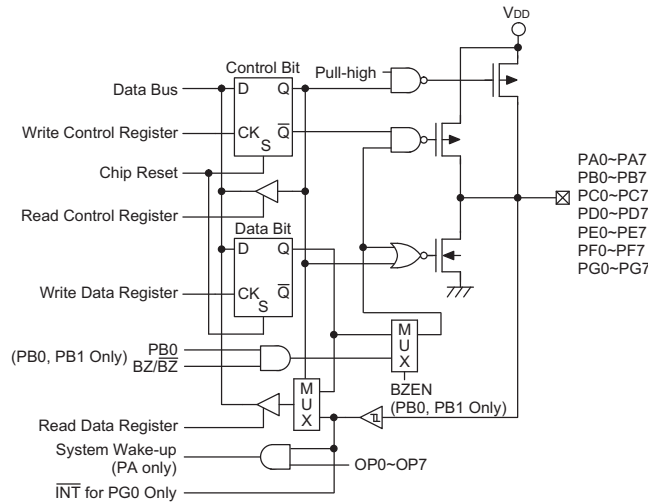
HT48R502 具有 56 个双向输入输出口，标号从 PA 到 PG，其分别对应的 RAM 的[12H], [14H], [16H], [18H], [1AH], [1CH]和[1EH]。所有的输入/输出口都能被作为输入或输出使用。输入时，端口不具有锁存功能，即输入数据必须在“MOV A, [m]” (m=12H, 14H, 16H, 18H, 1AH, 1CH 或 1EH) 指令的 T2 上升沿被准备好。输出时，所有的数据被锁存并保持不变，直到执行下一个写入操作。

每个 I/O 端口都有各自的控制寄存器 (PAC, PBC, PCC, PDC, PEC, PFC, PGC)，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可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”。输入信号来源也取决于控制寄存器。如果控制寄存器的某位值为“1”那么读取的是这个引脚的状态，如果控制寄存器的某位值为“0”，则读取的是内部锁存器的值。后者可能在“读—修改—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数据寄存器的 13H, 15H, 17H, 19H, 1BH, 1DH 和 1FH 地址。

系统复位后，这些输入/输出口都会是高电平或浮空状态（取决于上拉电阻选项）。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, 18H, 1AH, 1CH 或 1EH。)

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。



输入/输出口

PA 的每一个口都具有唤醒系统的能力。

所有的输入/输出口都可以有上拉电阻的选项。一旦选择上拉电阻，所有的输入/输出口都具有上拉电阻。必须要注意的是：没有上拉电阻的输入/输出口工作在输入模式会产生浮空状态。

PB0 和 PB1 分别与 BZ 和  $\overline{BZ}$  管脚复用。如果 BZ/ $\overline{BZ}$  的选项被选择，PB0/PB1 在输出模式时的输出信号将是由定时/计数器 0 的溢出信号产生的 PFD 信号。在输入模式始终保持它的原来的功能。一旦 BZ/ $\overline{BZ}$  的选项被选择，蜂鸣器的输出信号只受 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	O	O	O	O	O	O	O
PB1 输入/输出	I	O	I	I	I	O	O	O	O
PB0 模式	×	×	C	B	B	C	B	B	B
PB1 模式	×	C	×	×	×	C	C	C	B
PB0 数据	×	×	D	0	1	D <sub>0</sub>	0	1	0
PB1 数据	×	D	×	×	×	D <sub>1</sub>	D	D	×
PB0 引脚状态	I	I	D	0	B	D <sub>0</sub>	0	B	0
PB1 引脚状态	I	D	I	I	I	D <sub>1</sub>	D	D	0

注释：“I”输入，“O”输出，“D, D<sub>0</sub>, D<sub>1</sub>”：数据  
 “B”蜂鸣器的选项，BZ 或  $\overline{BZ}$ ；“×”：任意值  
 “C”：CMOS 输出

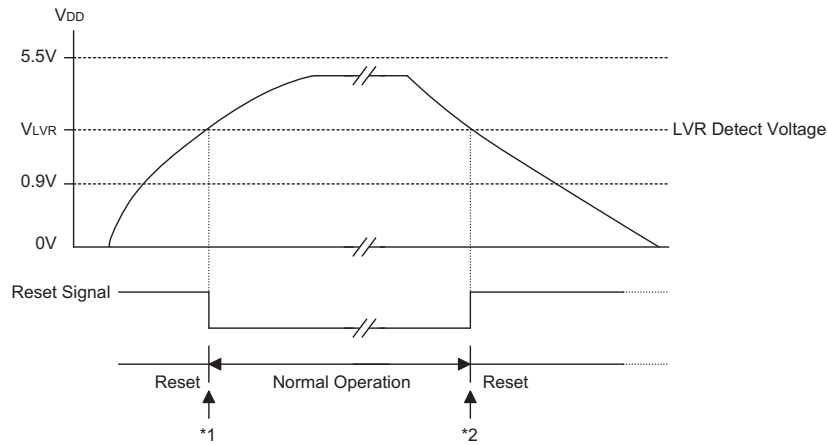
### 低电压复位 (LVR)

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 RES 信号的“或”函数功能来执行系统复位。

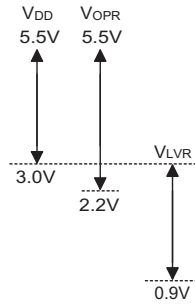
$V_{DD}$  与  $V_{LVR}$  之间的关系如下所示：



### 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。



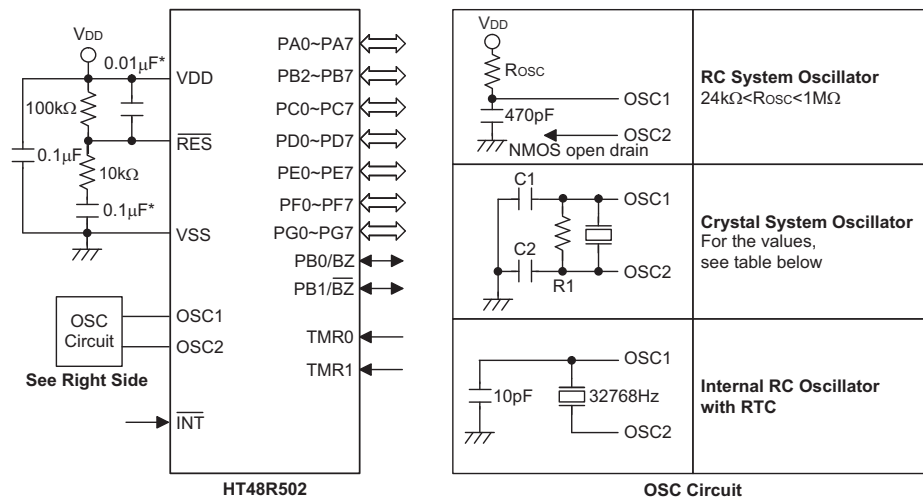
注： $V_{OPR}$  为系统时钟  $4MHz$  时一般芯片正常工作电压的范围。

**掩膜选项**

下表列出所有的掩膜选项。在系统运行前必须定义所有选项的值。

编号	选 项
1	WDT 时钟源: WDT 振荡或 f <sub>sys</sub> / 4 或 RTC 振荡或禁止
2	清除看门狗指令: 1 条或 2 条指令
3	定时/计数器 0 时钟来源: f <sub>sys</sub> 或 RTC 振荡
4	定时/计数器 1 时钟来源: f <sub>sys</sub> /4 或 RTC 振荡
5	PA 位唤醒功能: 使能/禁止
6	PA 端口 CMOS/斯密特输入
7	PA、PB、PC、PD、PE、PF、PG 上拉电阻使能/禁止
8	系统振荡器: 外部 RC、外部晶体振荡、内部 RC+RTC
9	内部 RC 振荡频率选择: 3.2MHz, 1.6MHz, 800kHz 或 400kHz
10	LVR 使能/禁止
11	BZ 选项使能/禁止

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 RES 置为高以前把工作电压保持在允许的范围内。  
“\*” 为了避免噪声干扰，连接 RES 引脚的线请尽可能地短

下表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
8MHz 晶振 & 谐振器	12pf	5.1 k Ω
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 & 谐振器	25pF	10k Ω
1MHz 晶振	68pF	10k 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 作用是保证在低电压状态下，晶振被关闭。这里的低电压，是指低于 MCU 正常工作电压范围。需要注意的是，当启用了 LVR 功能，R1 可以移除。

## 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 $\mu$ 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”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出口的 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
<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>注</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>注</sup>	Z,C,AC,OV
SUBA,x	ACC 与立即数相减, 结果放入 ACC	1	Z,C,AC,OV
SUBA,[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>注</sup>	Z,C,AC,OV
SBCA,[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 <sup>注</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>注</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>注</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>注</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>注</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>注</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>注</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>注</sup>	Z
助记符	说明	指令周期	影响标志位
<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>注</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>注</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>注</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>注</sup>	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 <sup>注</sup>	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 <sup>注</sup>	无
SET	[m].i	置位数据存储器的位	1 <sup>注</sup>	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 <sup>注</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>注</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>注</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>注</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>注</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>注</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>注</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>注</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
查表				
TABRDC	[m]	读取当前页的 ROM 内容, 并送至数据存储器 and TBLH	2 <sup>注</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>注</sup>	无
其它指令				
NOP		空指令	1	无
CLR	[m]	清除数据存储器	1 <sup>注</sup>	无
SET	[m]	置位数据存储器	1 <sup>注</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,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 <sup>注</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内容和累加器内容相加，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内容和累加器内容相加，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：将指定数据存储器内容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

<b>CALL</b>	<b>addr</b>	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 ← Program Counter+1 Program Counter ← addr
影响标志位:		无
<b>CLR</b>	<b>[m]</b>	Clear data memory
说明:		将指定数据存储器的内容清零。
运算过程:		[m] ← 00H
影响标志位:		无
<b>CLR</b>	<b>[m].i</b>	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 ← 0
影响标志位:		无
<b>CLR</b>	<b>WDT</b>	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 ← 00H PDF & TO ← 0
影响标志位:		TO、PDF
<b>CLR</b>	<b>WDT1</b>	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 ← 00H PDF & TO ← 0
影响标志位:		TO、PDF
<b>CLR</b>	<b>WDT2</b>	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 ← 00H PDF & TO ← 0
影响标志位:		TO、PDF
<b>CPL</b>	<b>[m]</b>	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] ← $\overline{[m]}$
影响标志位:		Z

<b>CPL</b>	<b>A</b>	<b>[m]</b>	<b>Complement data memory</b>
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
<b>DAA</b>	<b>[m]</b>		<b>Decimal-Adjust accumulator for addition</b>
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, 结果存放回数据存储器。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:			$[m] \leftarrow ACC+00H$ 或 $[m] \leftarrow ACC+06H$ $[m] \leftarrow ACC+60H$ 或 $[m] \leftarrow ACC+66H$
影响标志位:			C
<b>DEC</b>	<b>[m]</b>		<b>Decrement data memory</b>
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
<b>DECA</b>	<b>[m]</b>		<b>Decrement data memory and place result in the accumulator</b>
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
<b>HALT</b>			<b>Enter power down mode</b>
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
<b>INC</b>	<b>[m]</b>		<b>Increment data memory</b>
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
<b>INCA</b>	<b>[m]</b>		<b>Increment data memory and place result in the accumulator</b>
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

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

<b>RET</b>	<b>A, x</b>	<b>Return and place immediate data in the accumulator</b>
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 ← Stack ACC ← x
影响标志位:		无
<b>RETI</b>		<b>Return from interrupt</b>
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 ← Stack EMI ← 1
影响标志位:		无
<b>RL</b>	<b>[m]</b>	<b>Rotate data memory left</b>
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) ← [m].i (i=0~6) [m].0 ← [m].7
影响标志位:		无
<b>RLA</b>	<b>[m]</b>	<b>Rotate data memory left and place result in the accumulator</b>
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) ← [m].i (i=0~6) ACC.0 ← [m].7
影响标志位:		无
<b>RLC</b>	<b>[m]</b>	<b>Rotate data memory left through carry</b>
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) ← [m].i (i=0~6) [m].0 ← C C ← [m].7
影响标志位:		C
<b>RLCA</b>	<b>[m]</b>	<b>Rotate left through carry and place result in the accumulator</b>
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) ← [m].i (i=0~6) ACC.0 ← C C ← [m].7
影响标志位:	<b>CRR</b>	<b>[m] Rotate data memory right</b>
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 ← [m].(i+1) (i=0~6) [m].7 ← [m].0,
影响标志位:		无

<b>RRA</b>	<b>[m]</b>	<b>Rotate right and place result in the accumulator</b>
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:		$ACC.i \leftarrow [m].(i+1) \quad (i=0\sim6)$ $ACC.7 \leftarrow [m].0$
影响标志位:		无
<b>RRC</b>	<b>[m]</b>	<b>Rotate data memory right through carry</b>
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
<b>RRCA</b>	<b>[m]</b>	<b>Rotate right through carry and place result in the accumulator</b>
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
<b>SBC</b>	<b>A,[m]</b>	<b>Subtract data memory and carry from the accumulator</b>
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:		$ACC \leftarrow ACC - [m] - \overline{C}$
影响标志位:		OV、Z、AC、C
<b>SBCM</b>	<b>A,[m]</b>	<b>Subtract data memory and carry from the accumulator</b>
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
<b>SDZ</b>	<b>[m]</b>	<b>Skip if decrement data memory is 0</b>
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

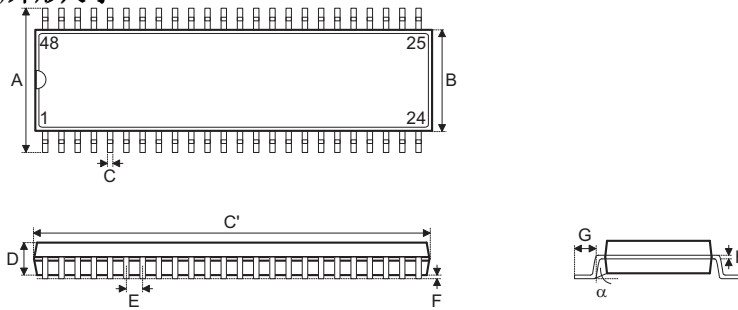
<b>SDZA</b> [m]	Decrement data memory and place result in ACC,skip if 0
说明:	将指定数据存储器内容减 1, 判断是否为 0, 如果为 0 则跳过下一条指令, 此结果将存放到累加器, 但指定数据存储器内容不变。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。
运算过程:	$ACC \leftarrow [m]-1$ , 如果 $ACC=0$ 跳过下一条指令执行
影响标志位:	无
<b>SET</b> [m]	Set data memory
说明:	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
<b>SET</b> [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
<b>SIZ</b> [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
<b>SIZA</b> [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
<b>SNZ</b> [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
<b>SUB</b> 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

<b>SUBM</b>	<b>A, [m]</b>	<b>Subtract data memory from the accumulator</b>
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	
<b>SUB</b>	<b>A, x</b>	<b>Subtract immediate data from the accumulator</b>
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	
<b>SWAP</b>	<b>[m]</b>	<b>Swap nibbles within the data memory</b>
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	
<b>SWAPA</b>	<b>[m]</b>	<b>Swap data memory and place result in the accumulator</b>
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:	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$	
影响标志位:	无	
<b>SZ</b>	<b>[m]</b>	<b>Skip if data memory is 0</b>
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	
<b>SZA</b>	<b>[m]</b>	<b>Move data memory to ACC, skip if 0</b>
说明:	将指定数据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:	$ACC \leftarrow [m]$ , 如果 $[m] = 0$ , 跳过下一条指令执行	
影响标志位:	无	
<b>SZ</b>	<b>[m].i</b>	<b>Skip if bit I of the data memory is 0</b>
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	

<b>TABRDC [m]</b>	<b>Move the ROM code(current page) to TBLH and data memory</b>
说明:	将表格指针 TBLP 所指的程序代码低字节（当前页）移至指定的数据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
<b>TABRDL [m]</b>	<b>Move the ROM code(last page) to TBLH and data memory</b>
说明:	将表格指针 TBLP 所指的程序代码低字节（最后一页）移至指定的数据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
<b>XOR A, [m]</b>	<b>Logical XOR accumulator with data memory</b>
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 ← ACC “XOR” [m]
影响标志位:	Z
<b>XORM A, [m]</b>	<b>Logical XOR data memory with accumulator</b>
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] ← ACC “XOR” [m]
影响标志位:	Z
<b>XOR A, x</b>	<b>Logical XOR immediate data to the accumulator</b>
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 ← ACC “XOR” x
影响标志位:	Z

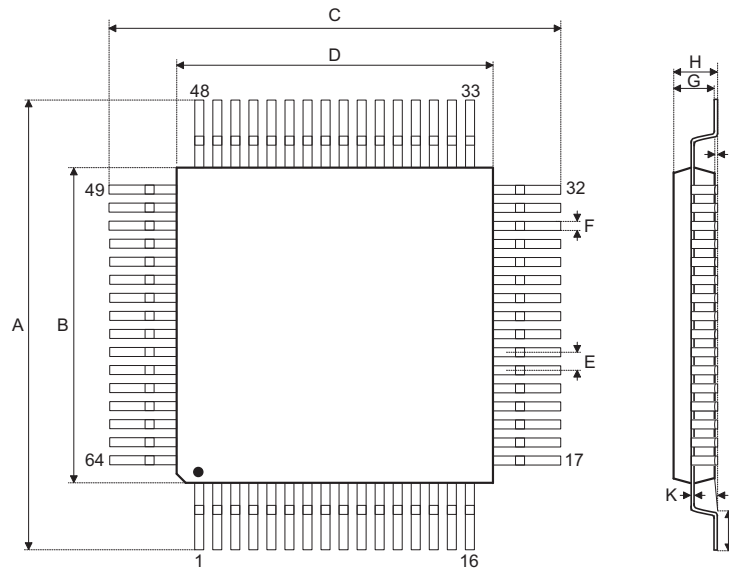
封装信息

48pin SSOP (300mil)外形尺寸



标号	尺寸 (mil)		
	最小值	典型值	最大值
A	395	--	420
B	291	--	299
C	8	--	12
C'	613	--	637
D	85	--	99
E	--	25	--
F	4	--	10
G	25	--	35
H	4	--	12
α	0°	--	8°

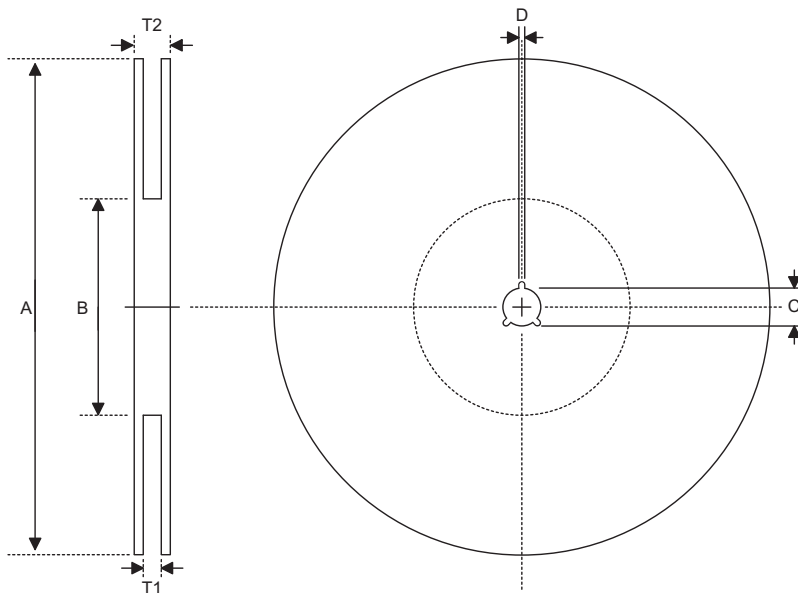
64pin LQFP (7mm x 7mm)外形尺寸



标号	尺寸 (mm)		
	最小值	典型值	最大值
A	8.9	--	9.1
B	6.9	--	7.1
C	8.9	--	9.1
D	6.9	--	7.1
E	--	0.4	--
F	0.13	--	0.23
G	1.35	--	1.45
H	--	--	1.6
I	0.05	--	0.15
J	0.45	--	0.75
K	0.09	--	0.2
α	0°	--	7°

包装带和卷轴规格:

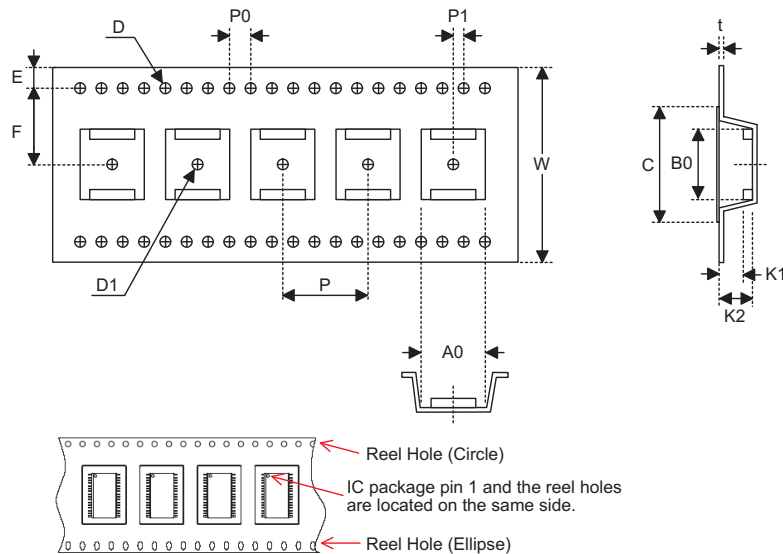
卷轴尺寸:



SSOP 48W

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±0.1
C	轴心直径	13.0 <sup>+0.5/-0.2</sup>
D	缝宽	2.0±0.5
T1	轮缘宽	32.2 <sup>+0.3/-0.2</sup>
T2	卷轴宽	38.2±0.2

运输带尺寸:



SS0P 48W

标号	描述	尺寸(mm)
W	运输带宽	32.0±0.3
P	空穴间距	16.0±0.1
E	穿孔位置	1.75±0.1
F	空穴至穿孔距离 (宽度)	14.2±0.1
D	穿孔直径	2.0Min.
D1	空穴中之小孔直径	1.5 <sup>+0.25/-0.00</sup>
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	12.0±0.1
B0	空穴宽	16.20±0.1
K1	空穴深	2.4±0.1
K2	空穴深	3.2±0.1
t	传输带厚度	0.35±0.05
C	覆盖带宽度	25.5±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

**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: 021-54224590  
传真: 021-54224596  
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: 0755-8616-9533  
ISDN: 0755-8615-6181

**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

**Holmate Semiconductor, 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.holmate.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>