

## 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 © 2008 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](#)
  - [HA0013S HT48 & HT46 MCU LCM 接口设计](#)
  - [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](#)
  - [HA0049S HT1380 的读写控制](#)
  - [HA0075S MCU 复位和振荡电路应用范例](#)

## 特性

- 工作电压：
  - fsys = 4MHz : 2.2V – 5.5V
  - fsys = 8MHz : 3.3V – 5.5V
- 13 个双向输入/输出
- 一个与输入/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
- 看门狗定时器
- 程序存储器 ROM
  - HT48R05A-1/HT48C05 : 512×14
  - HT48R06A-1/HT48C06 : 1024×14
  - HT48R08A-1 : 2048×14
- 数据存储器 RAM
  - HT48R05A-1/HT48C05 : 32×8
  - HT48R06A-1/HT48C06 : 64×8
  - HT48R08A-1 : 96×8
- 蜂鸣器驱动并支持 PFD
- HALT 和唤醒功能可降低功耗
- 在 V<sub>DD</sub>=5V，系统时钟为 8MHz 时，指令周期为 0.5 μs
- 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
- 2 层硬件堆栈
- 位操作指令
- 强大的指令
  - HT48R05A-1/HT48C05 :62 条
  - HT48R06A-1/HT48C06/HT48R08A-1 :63 条
- 低电压复位功能
- 16-pin SSOP/NSOP 封装
- 18-pin DIP/SOP 封装

## 概述

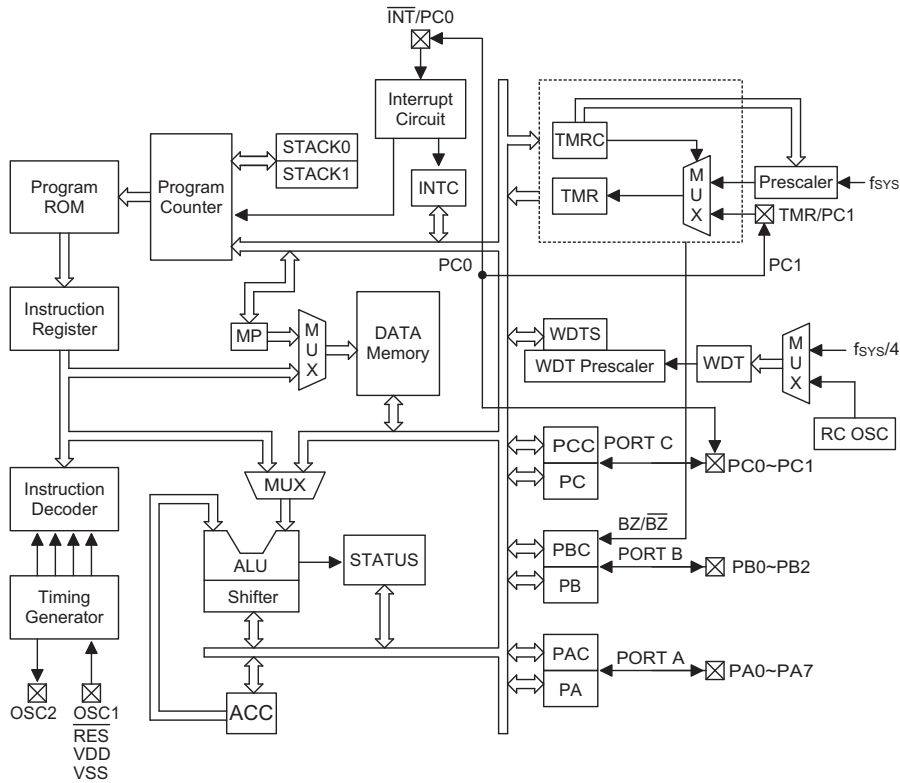
HT48R05A-1/HT48C05, HT48R06A-1/HT48C06 和 HT48R08A-1 是一款八位高性能精简指令集单片机，专为经济型多输入输出控制的产品设计。掩膜版芯片 HT48C05 和 HT48C06 在引脚和功能方面，都与 OTP 版芯片 HT48R05A-1 和 HT48R06A-1 完全相同。

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。

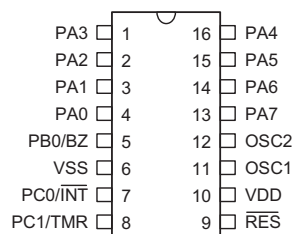
## 选型表

Part No.	VDD	Program Memory	Data Memory	I/O	Timer	Int.	PFD	Stack	Package Types
HT48R05A-1 HT48C05	2.2V~5.5V	0.5K×14	32×8	13	8-bit×1	2	√	2	16SSOP/NSOP, 18DIP/SOP
HT48R06A-1 HT48C06	2.2V~5.5V	1K×14	64×8	13	8-bit×1	2	√	2	16SSOP/NSOP, 18DIP/SOP
HT48R08A-1	2.2V~5.5V	2K×14	96×8	13	8-bit×1	2	√	2	16SSOP/NSOP, 18DIP/SOP

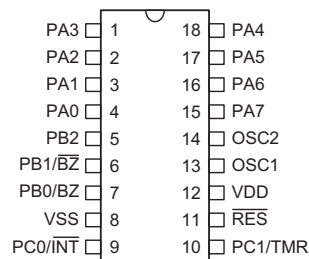
### 方框图



### 引脚图



HT48R05A-1/HT48C05  
HT48R06A-1/HT48C06  
HT48R08A-1  
— 16 SSOP-A/NSOP-A



HT48R05A-1/HT48C05  
HT48R06A-1/HT48C06  
HT48R08A-1  
— 18 DIP-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	双向 8 位输入/输出口。每一位能被掩膜选项设置为唤醒输入。软件指令确定 CMOS 输出，或带上拉电阻的斯密特触发输入（由上拉电阻选项确定）。
PB0/BZ PB1/ $\overline{\text{BZ}}$ PB2	输入/输出	上拉电阻* 输入/输出 或 BZ/BZ	双向 3 位输入/输出口。软件指令确定 CMOS 输出，或带上拉电阻的斯密特触发输入（由上拉电阻选项确定）。PB0 和 PB1 与 BZ 和 $\overline{\text{BZ}}$ 共用一个引脚。一旦 PB0 和 PB1 被选为蜂鸣器输出，输出信号来自内部 PFD 发生器（与定时/计数器共用）。
VSS	—	—	负电源，接地。
PC0/ $\overline{\text{INT}}$ PC1/TMR	输入/输出	上拉电阻*	双向输入/输出口。软件指令确定 CMOS 输出，或带上拉电阻的斯密特触发输入（由上拉电阻选项确定）。外部中断和定时器输入与 PC0 和 PC1 共用一个引脚。外部中断输入信号下降沿有效。
$\overline{\text{RES}}$	输入	—	斯密特触发复位输入端，低电平有效。
VDD	—	—	正电源。
OSC1 OSC2	输入 输出	晶体振荡 或 RC 振荡	OSC1 和 OSC2 采用 RC 振荡或晶体振荡（由掩膜选项确定）提供系统时钟。在 RC 方式下 OSC2 是一个系统时钟四分频的输出口。

\*所有的上拉电阻由一个选择位控制。

## 极限参数

电源供应电压 ..... V<sub>SS</sub> -0.3V 至 V<sub>SS</sub> +6.0V  
 端口输入电压 ..... V<sub>SS</sub> - 0.3V 至 V<sub>DD</sub> +0.3V  
 I<sub>OL</sub> 总电流.....150mA  
 总消耗电流.....500mW

储存温度 ..... -50°C 至 125°C  
 工作温度 ..... -40°C 至 85°C  
 I<sub>OH</sub> 总电流..... -100mA

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
		VDD	条件				
VDD	工作电压	—	f <sub>sys</sub> =4MHZ	2.2	—	5.5	V
		—	f <sub>sys</sub> =8MHZ	3.3	—	5.5	V
IDD1	工作电流 (晶体振荡)	3V	无负载	—	0.6	1.5	mA
		5V	F <sub>sys</sub> =4MHZ	—	2	4	
IDD2	工作电流 (RC 振荡)	3V	无负载	—	0.8	1.5	mA
		5V	f <sub>sys</sub> =4MHZ	—	2.5	4	
IDD3	工作电流 (晶体振荡, RC 振荡)	5V	无负载 f <sub>sys</sub> =8MHZ	—	4	8	mA
ISTB1	静态电流 (看门狗打开)	3V	无负载	—	—	5	μ A
		5V	暂停模式	—	—	10	
ISTB2	静态电流 (看门狗关闭)	3V	无负载	—	—	1	μ A
		5V	暂停模式	—	—	2	
VIL1	输入/输出、TMR 和 $\overline{\text{INT}}$ 的低电平输入 电压,	—	—	0	—	0.3V <sub>DD</sub>	V
VIH1	输入/输出、TMR 和 $\overline{\text{INT}}$ 的高电平输入 电压,	—	—	0.7V <sub>DD</sub>	—	V <sub>DD</sub>	V
VIL2	低电平输入电压 (RES)	—	—	0	—	0.4V <sub>DD</sub>	V
VIH2	高电平输入电压 (RES)	—	—	0.9V <sub>DD</sub>	—	V <sub>DD</sub>	V
VLVR	低电压复位	—	LVR 打开	2.7	3.0	3.3	V
IOL	输入/输出灌电流	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	4	8	—	mA
		5V		10	20	—	
IOH	输入/输出源电流	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-2	-4	—	mA
		5V		-5	-10	—	
RPH	上拉电阻	3V	—	20	60	100	kΩ
		5V		10	30	50	

A.C.特性

Ta=25°C

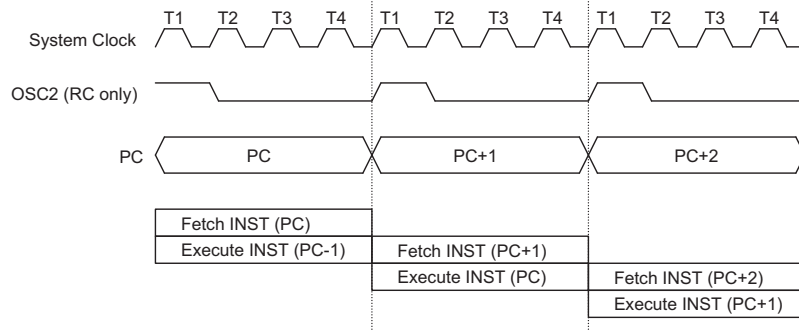
符号	参数	测试条件		最小	典型	最大	单位
		VDD	条件				
f <sub>SYS</sub>	系统时钟（晶体振荡，RC振荡）	—	2.2V~5.5V	400	—	4000	KHz
		—	3.3V~5.5V	400	—	8000	
f <sub>TIMER</sub>	定时器输入频率(TMR)	—	2.2V~5.5V	0	—	4000	KHz
		—	3.3V~5.5V	0	—	8000	
t <sub>WDTOSC</sub>	看门狗振荡器周期	3V	—	45	90	180	μs
		5V		32	65	130	
t <sub>WDT1</sub>	看门狗溢出周期(RC)	3V	WDT 无预分频	11	23	46	ms
		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	—	tsys
t <sub>RES</sub>	外部复位低电平脉宽	—	—	1	—	—	μs
t <sub>SST</sub>	系统启动延时周期	—	HALT 模式唤醒	—	1024	—	tsys
t <sub>INT</sub>	中断脉冲宽度	—	—	1	—	—	μs
t <sub>LVR</sub>	低电压复位周期	—	—	0.25	1	2	ms

备注: t<sub>sys</sub>=1/f<sub>sys</sub>

## 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。系统内部对此频率进行四分频，产生四个不重叠的时钟周期。一个指令周期包括四个系统时钟周期。



### 操作流程

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

### 程序计数器 (PC)

程序计数器控制存放在程序存储器中的要被执行的指令序列。程序计数器可寻址程序存储器的所有地址。

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

当执行一条跳转指令，条件跳转指令，装载 PCL 寄存器，子程序调用，初始复位，内部中断，外部中断，或从一个子程序返回，PC 会通过装载相应的地址来执行程序转移。

通过指令实现条件跳转，一旦条件满足，那么在当前指令执行期间取出的下一条指令会被放弃，而替代它的是一条空指令周期(dummy cycle)来获取正确的指令，接着就执行这条指令。否则就执行下一条指令。程序计数器的低位字节 (PCL; 06H) 是可读写的寄存器。将数据赋值到 PCL 会执行一个短跳转。这种跳转只能在 256 个地址范围内。当一个控制转移发生时，系统也会插入一个空指令周期。

模 式	程序计数器										
	*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
外部中断	0	0	0	0	0	0	0	0	1	0	0
定时/计数器	0	0	0	0	0	0	0	1	0	0	0
条件跳转	程序计数器+2										
装载 PCL	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
跳转, 子程序调用	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
从子程序返回	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

### 程序计数器

注意: \*10 ~ \*0 : 程序计数器位                      S10 ~ S0 : 堆栈寄存器位  
 #10 ~ #0 : 指令代码位                              @7 ~ @0 : PCL 位  
 对于 HT48R05A-1/HT48C05, 程序计数器有 9 位, 从 \*8 ~ \*0  
 对于 HT48R06A-1/HT48C06, 程序计数器有 10 位, 从 \*9 ~ \*0  
 对于 HT48R08A-1, 程序计数器有 11 位, 从 \*10 ~ \*0

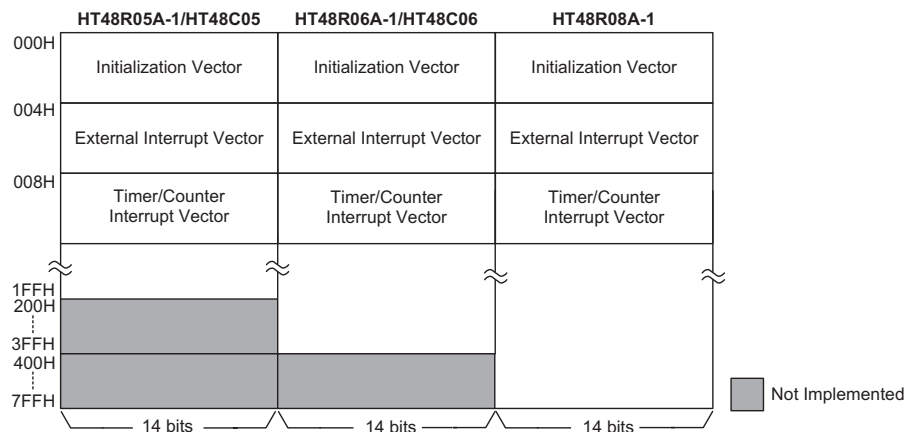
### 程序存储器 (ROM)

程序存储器被用来存放要执行的指令代码。还包括数据、中断向量。一共为  $512 \times 14$  位 (HT48R05A-1/HT48C05) 或者是  $1024 \times 14$  位 (HT48R06A-1/HT48C06) 或者是  $2048 \times 14$  位 (HT48R08A-1)。可以由程序计数器或表格指针来寻址。

在程序存储器中某几个地址被保留作为特殊用途。

- 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  
该地址保留给外部中断服务使用。当中断是开放的，且堆栈又未滿，则一旦  $\overline{\text{INT}}$  端被下降沿触发触发，就能产生中断，程序会从 004H 地址开始执行外部中断服务程序。
- 地址 008H  
该地址保留给定时/计数器中断服务使用。当中断是开放的，且堆栈又未滿，则一旦定时/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。

• 表格区  
程序存储器内的任何地址都可被用来作为查表使用。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] 为数据被存放的地址，但是对于 HT48R05A-1/HT48C05 来说，此语句无效。[m] 为数据存放的地址。在执行 TABRDC [m]指令后，将会传送当前页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送到目标地址中，而表格中的高位字节的其它位被传送到 TBLH 的低部位，剩余的二位作为 0 读出。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。TBLH 只能读出而不能存储。如果主程序和 ISR (中断服务程序) 二者都使用查表指令，那么在主程序中的 TBLH 的内容可能会被 ISR 中的查表指令改变而产生错误。换句话说，应该避免在主程序和 ISR (中断服务程序) 中同时使用查表指令。但是，如果主程序和 ISR (中断服务程序) 二者都必需使用查表指令，那么中断应该在查表指令前被禁止，直到 TBLH 被备份好。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，这些区域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										
	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
TABRDC [m]	P10	P9	P8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

**表格区**

注意: \*10 ~ \*0 : 表格地址                      P10~P8 : 当前程序计数器  
 @7 ~ @0 : 表格指针  
 对于 HT48R05A-1/HT48C05, 表格地址有 9 位, 从\*8 ~ \*0  
 对于 HT48R06A-1/HT48C06, 表格地址有 10 位, 从\*9 ~ \*0  
 对于 HT48R08A-1, 表格地址有 11 位, 从\*10 ~ \*0



### 间接寻址寄存器

地址 00H 是作为间接寻址寄存器。它没有实际的物理空间。任何对[00H]的读/写操作，都会访问由 MP(01H)所指向的 RAM 单元。间接地读取[00H]，将会返回 00H，而间接地写入 00H 单元，则不会产生任何结果。

间接寻址指针 MP 的宽度是 7 位，MP 的第 7 位是没有定义的。若读它，将返回“1”。而任何对 MP 写的操作只能将低 7 位数据传送到 MP。

### 累加器 (ACC)

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

### 算术逻辑单元 (ALU)

算术逻辑单元是执行 8 位算术逻辑运算的电路。它提供如下的功能：

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

算术逻辑单元 ALU 不仅会保存运算的结果而且会改变状态寄存器。

### 状态寄存器 (STATUS)

8 位的状态寄存器 (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运算会导致状态寄存器的改变。系统上电，看门狗定时器溢出或执行“CLR WDT”或“HALT”指令，能改变看门狗定时器溢出标志位(TO)。系统上电，或执行“CLR WDT”或“HALT”指令，能改变暂停标志位(PDF)。

Z, OV, AC 和 C 标志位都反映了当前的运算状态。

位	符号	功 能
0	C	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 被清零。它也可被一个循环移位指令而影响。
1	AC	在加法运算中低四位产生了进位或减法运算中在低四位不产生借位, AC 被置位; 反之, AC 被清零。
2	Z	算术运算或逻辑运算的结果为零则 Z 被置位; 反之, Z 被清零。
3	OV	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 被清零。
4	PDF	系统上电或执行了 CLR WDT 指令, PDF 被清零。执行 HALT 指令 PDF 被置位。
5	TO	系统上电或执行了 CLR WDT 指令或 HALT 指令, TO 被清零。WDT 溢出, TO 被置位。
6~7	—	未定义, 读为零

### 状态寄存器 (0AH)

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

## 中断 (INT)

单片机提供一个外部中断和内部定时器/计数器中断。中断控制寄存器 (INTC; 0BH) 包含了中断控制位, 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 (通过清零 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。

位	符号	功 能
0	EMI	主中断(全局)控制位 (1=允许, 0=禁止)
1	E EI	外部中断控制位 (1=允许, 0=禁止)
2	ETI	定时/计数器中断控制位 (1=允许, 0=禁止)
3,6~ 7	—	未使用位, 读为零
4	EIF	外部中断请求标志位 (1=有, 0=无)
5	TF	定时/计数器中断请求位 (1=有, 0=无)

### 中断控制寄存器 (0BH)

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 (PC) 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  $\overline{\text{INT}}$  脚上的电平由高到低的变化触发的, 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, 将会产生地址 04H 的子程序调用。中断请求标志 (EIF) 位和 EMI 位也将被清零来禁止中断的嵌套。

内部定时/计数器中断发生时, 会设置定时/计数器中断请求标志位 (TF, INTC 的第五位), 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 TF 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 (TF) 被复位并且 EMI 位将被清零, 以便禁止中断的嵌套。

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 RETI 指令被执行或是 EMI 位和相关的中断控制位都被置为 1 (堆栈未滿时)。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 位来允许中断服务, 而 RET 则不能自动置位 EMI。

如果中断在二个连续的 T2 脉冲的上升沿间发生, 如果中断响应被允许的话, 那么在二个 T2 脉冲后, 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。

NO.	中 断 源	优 先 级	中 断
a	外部中断	1	04H
b	定时/计数器溢出	2	08H

中断控制寄存器 (INTC) 其 RAM 地址是 0BH, 由定时/计数器中断请求标志位 (TF)、外部中断请求标志位 (EIF)、定时/计数器允许位 (ETI)、外部中断允许位 (EEI) 和主中断控制允许位 (EMI) 组成。EMI、EEI 和 ETI 是用来控制中断的允许/禁止的状态的。这些位防止正在进行中断服务中的中断请求。一旦中断请求标志位 (EEI, ETI) 被置位, 它们将在 INTC 中被保留下来, 直到相关的中断被服务或由软件指令来清零。

建议不要在中断子程序中使用“CALL”指令来调用子程序, 因为它可能会破坏原来的控制序列, 而中断经常随机发生或某一个确定的应用程序可能要求立即服务。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序中又执行了 CALL 子程序调用, 则会造成堆栈溢出而破坏原先的控制序列。

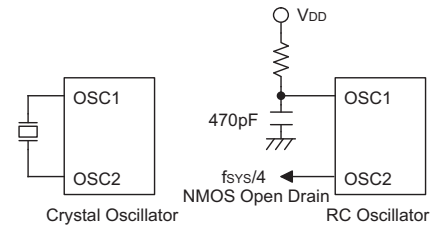
## 振荡器

通过掩膜选项，可以选择外部晶体振荡或外部 RC 振荡。两者都可作为系统时钟。不管用哪种振荡器，其信号都支持系统时钟。HALT 模式会停止系统振荡器并忽略任何外部信号，由此来节省功耗。

如果采用 RC 振荡方式，那么在 VDD 与 OSC1 之间要接一个外部电阻。其阻值范围为  $24\text{K}\Omega \sim 1\text{M}\Omega$ 。在 OSC2 端可获得系统时钟四分频信号，它可以用于同步系统外部逻辑。RC 振荡器是一种低成本方案，但是振荡频率由于 VDD，温度及芯片自身参量的漂移而产生误差。因此，对计时敏感的场所，要求精确度高的振荡器频率，RC 振荡器是不适用的。

如果选用晶体振荡方式，在 OSC1 与 OSC2 之间连接一个晶体，用来提供晶体振荡器所要的反馈（Feedback）和相位位移（Phase Shift）。除此以外，不再需要其他外部元件。另外，在 OSC1 与 OSC2 之间接一个谐振器（Resonator）来取代晶体振荡器用来得到参考频率，但是需要在 OSC1 与 OSC2 外接二个电容器（如果振荡器频率小于  $1\text{MHz}$ ）。

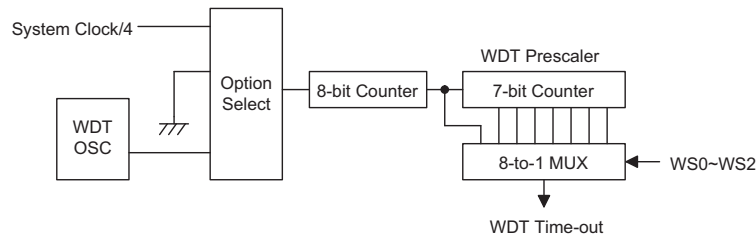
WDT 看门狗振荡器是一个芯片内部自由振荡的 RC 振荡器，不需连接外部元件。甚至系统进入省电模式时，系统时钟停止了，但 WDT 振荡器仍然依大约  $65\mu\text{s}/5\text{V}$  的周期工作。由掩膜选项，WDT 振荡器能停振来节省功耗。



## 看门狗定时器 (WDT)

WDT 的时钟源是由一个专用的 RC 振荡器 (WDT 振荡器) 或是由指令周期 (系统时钟 4 分频) 来实现的, 由掩膜选项来决定的。WDT 是用来防止程序不正常运行或是跳到未知或不希望去的地址, 而导致不可预见的结果。WDT 可以被掩膜选项禁止。如果 WDT 定时器被禁止, 所有与 WDT 有关的操作都是空操作。

如果设置了内部 WDT 振荡器 (以 65us/5V 为周期的 RC 振荡器) 的话, WDT 的值会先除以 256 (8 级) 来产生大约 17ms/5V 的溢出时间, 这个溢出时间会因为温度, VDD, 以及芯片参数的变化而变化。如果启动 WDT 的预分频器, 则可实现更长的溢出时间。写数据到 WS2, WS1, WS0 (WDTS 的 2、1、0 位) 会产生不同的溢出时间, 举例来说, 如果 WS2, WS1, WS0 的值都为 1, 其分频级数最大, 为 1: 128, 溢出时间最长约 2.1s/5V。如果 WDT 被禁止, 那么 WDT 的时钟来源可来自指令时钟, 其运作与 WDT 振荡器一样。但当在 HALT 状态时, 来源于指令时钟的 WDT 会在暂停模式时停止计数并失去保护功能。在这种情况下, 只能由外部逻辑来重新启动系统。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, 那么强烈建议使用片内 RC 振荡器 (WDT OSC), 因为 HALT 模式会使系统时钟终止运作。

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, 溢出只产生一个“热复位”, 只能使 PC 程序计数器和堆栈指针 SP 复位到零。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), 用软件指令和 HALT 指令三种。软件指令由 CLR WDT 和另一组指令 CLR WDT1 及 CLR WDT2 组成。这两组指令中, 只能选取其中一种。选择的方式由掩膜选项的 CLR WDT 次数选项决定。如果“CLR WDT”被选择 (即 CLR WDT 次数为 1), 那么只要执行 CLR WDT 指令就会清除 WDT。在 CLR WDT1 和 CLR WDT2 被选择的情况下 (即 CLR WDT 次数为 2), 那么要执行二条指令才会清除 WDT。否则, WDT 会由于溢出而使系统复位。

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

看门狗定时器预置寄存器(09H)

## 暂停模式 (HALT)

暂停模式是由 HALT 指令来实现的，产生如下结果：

- 关闭系统振荡器，但 WDT 振荡器继续工作（如果 WDT 时钟来源是 WDT 振荡器）。
- RAM 及寄存器的内容保持不变。
- WDT 被清除并再次重新计数（如果 WDT 时钟来源是 WDT 振荡器）。
- 所有的输入/输出都保持其原先状态。
- PDF 标志位被置位，TO 标志位被清零。

由于外部复位、中断、外部输入一个下降沿的信号到 PA 口或 WDT 溢出，可使系统脱离暂停状态。外部复位能使系统初始化而 WDT 溢出使系统“热复位”。测试 TO 和 PDF 状态后，系统复位的原因就可以被确定。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。如果 WDT 产生溢出，会使 TO 标志位置位，还能产生唤醒使得程序计数的 PC 和堆栈指针 SP 复位。其他都保持原状态。

PA 口的唤醒和中断方式被看作为正常运行。PA 口的每一位都可以通过掩膜选项来单独设定为对系统的唤醒。如果唤醒是来自于输入/输出信号的变化，程序会重新继续执行下一条命令。如果唤醒是来自于中断，那么有两种情况可能发生：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滿，那么这个中断响应就发生了。如果在进入 HALT 模式以前，中断请求标志位被置“1”，那么相关的中断唤醒功能被禁止。一旦唤醒事件发生，要花 1024tsys（系统时钟周期），系统重新正常运行。换句话说，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或一个以上的周期。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。

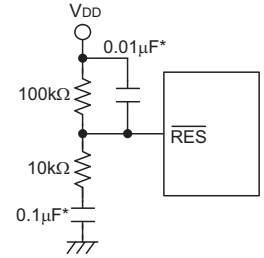
为了省电，在进入 HALT 模式之前必须要小心处理输入/输出状态。

### 复位 (RESET)

有三种方法可以产生复位:

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

在 HALT 期间 WDT 溢出是不同于其它的复位操作条件, 因为它可执行“热复位”, 结果只能使程序计数器 PC 和堆栈指针 SP 复位, 而别的寄存器均保持原来的状态。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。

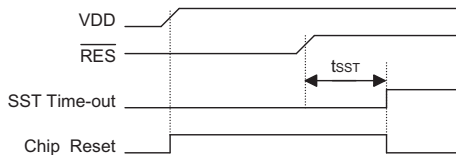


复位电路  
注意: “\*” 为了避免噪声干扰, 连接 RES 引脚的线应尽量短

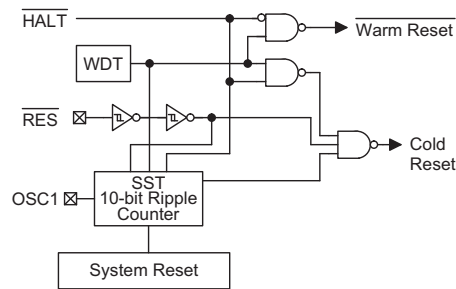
TO	PDF	复位条件
0	0	电源上电复位
u	u	正常运作时由 $\overline{\text{RES}}$ 复位
0	1	$\overline{\text{RES}}$ 时唤醒暂停模式
1	u	正常运作时发生看门狗定时器超时
1	1	由看门狗定时器唤醒暂停模式

注释: “u” 表示未变化。

为保证系统振荡器起振并稳定运行, 那么 SST (系统启动定时器) 当系统复位 (上电, WDT 定时溢出或  $\overline{\text{RES}}$ ) 或从 HALT 状态被唤醒时, 提供额外延迟 1024 个系统时钟脉冲。



复位时序



复位电路结构

当系统复位时, SST 延迟被加到复位周期中。任何来自 HALT 的唤醒都将允许 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 0/1	关闭
输入/输出口	输入模式
堆栈指针 SP	指向堆栈顶端

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
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
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC	--00 -000	--00 -000	--00 -000	--00 -000	--uu -uuu
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
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	---- -111	---- -111	---- -111	---- -111	---- -uuu
PBC	---- -111	---- -111	---- -111	---- -111	---- -uuu
PC	---- --11	---- --11	---- --11	---- --11	---- --uu
PCC	---- --11	---- --11	---- --11	---- --11	---- --uu

注意：“\*”表示“热复位。”

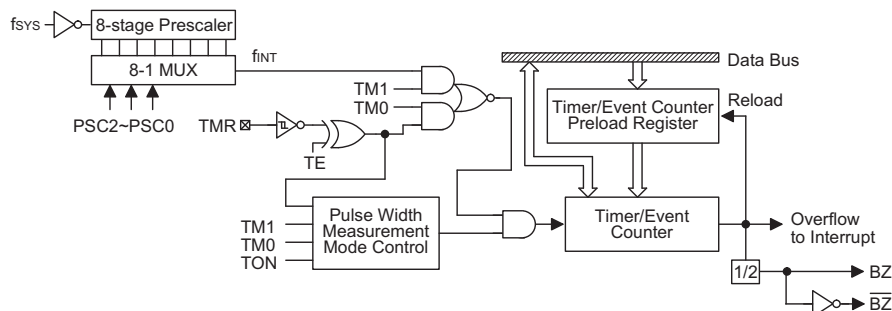
“U”表示不变化。

“×”表示不确定。

## 定时/计数器

这个单片机提供一个定时/计数器。定时/计数器包含一个 8 位可编程的向上计数的计数器，时钟可以来自外部的时钟源或是系统时钟。

在采用外部时钟输入的情况下，允许用户计量外部事件、测量时间间隔或脉冲宽度、或产生一个精确的时基信号。该定时/计数器使用外部时钟或内部时钟可以产生 PFD 信号，PFD 信号频率可由等式  $f_{INT} / [2 \times (256 - N)]$  计算。



## 定时/计数器

有两个寄存器与定时/计数器相关联，即 TMR ([0DH]) 和 TMRC ([0EH])。有两个物理寄存器对应 TMR 的位置。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 则会获得定时/计数器的内容。TMRC 是定时/计数器控制寄存器。

TM0 和 TM1 位定义操作模式。外部事件计数模式用来记录外部事件，这意味着时钟来自外部 TMR 引脚。定时器模式是作为一个普通的定时器功能，时钟源来自 fINT 时钟。脉冲宽度测量模式能用来计算外部引脚 TMR 上的高电平或低电平的宽度。计数是基于 fINT 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计数到 FFH。一旦产生溢出，计数器会从定时/计数器预置寄存器重新装载并且同时产生相应的中断请求状态位 (TF ; INTC 的第 5 位)。

在脉冲宽度测量中，将 TON 和 TE 置为“1”，如果 TMR 接收到从低到高的电平跳变（或从高到低的变化，如果 TE 位被清零），就开始计数直到 TMR 返回到原来的电平并且复位 TON 位。测量的结果被保留在定时/计数器中，甚至电平跳变再一次发生也不会改变。换句话说，一次只能测量一个周期。直到 TON 再次被置位，只要再接到跳变信号，那么测量过程会再次执行。要注意在这个操作模式中，定时/计数器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（TON；TMRC 的第 4 位）置 1。在脉宽测量模式中，TON 在测量周期结束后自动被清零。但在另外两个模式中，TON 只能由指令来复位。定时/计数器的溢出是唤醒的信号之一。不管任何模式，若写 0 到 ETI 位即可禁止相应的中断服务。

在定时/计数器为关闭的状态下，写数据到定时/计数器的预置寄存器之中，同时也会将数据装入定时/计数器中。但若是定时/计数器已经开启，写到定时/计数器的数据只会被保留在定时/计数器的预置寄存器中，直到定时/计数器发生计数溢出为止，再由预置寄存器加载新的值。当定时/计数器的数据被读取时，会禁止时钟输入以防出错。因为禁止时钟输入可能导致计数错误，所以程序员必须仔细加以考虑。

TMRC 的 0~2 位被用于定义定时/计数器的内部时钟源的预分频级数。定义如表所示。定时/计数器的溢出信号被用于产生驱动蜂鸣器的 PFD 信号。

位	符号	功能
0-2	PSC0~PSC2	定义预分频器级数，PSC2, PSC1, PSC0= 000: $f_{INT} = f_{sys}/2$ 001: $f_{INT} = f_{sys}/4$ 010: $f_{INT} = f_{sys}/8$ 011: $f_{INT} = f_{sys}/16$ 100: $f_{INT} = f_{sys}/32$ 101: $f_{INT} = f_{sys}/64$ 110: $f_{INT} = f_{sys}/128$ 111: $f_{INT} = f_{sys}/256$
3	TE	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)
4	TON	打开/关闭定时/计数器(1=打开，0=关闭)
5	—	未用，读出为 0
6 7	TM0 TM1	定义操作模式 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

**TMRC 寄存器 (0EH)**

### 输入/输出口

单片机具有 13 个双向输入输出口，标号从 PA 到 PC，其分别对应的 RAM 的[12H]，[14H]和[16H]。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 或 16H) 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。

每个输入输出口都有其自己的控制寄存器（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在软件下动态地进行改变。作为输入时，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，可以在“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 地址。

芯片复位后，这些输入/输出口都会是高电平或浮空状态（取决于上拉电阻的选项）。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 或 16H。）

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 6 位和 PB 口的高 5 位在物理上是不存在的；读这些位将返回“0”，而写入则是一个空操作。请看应用注释。

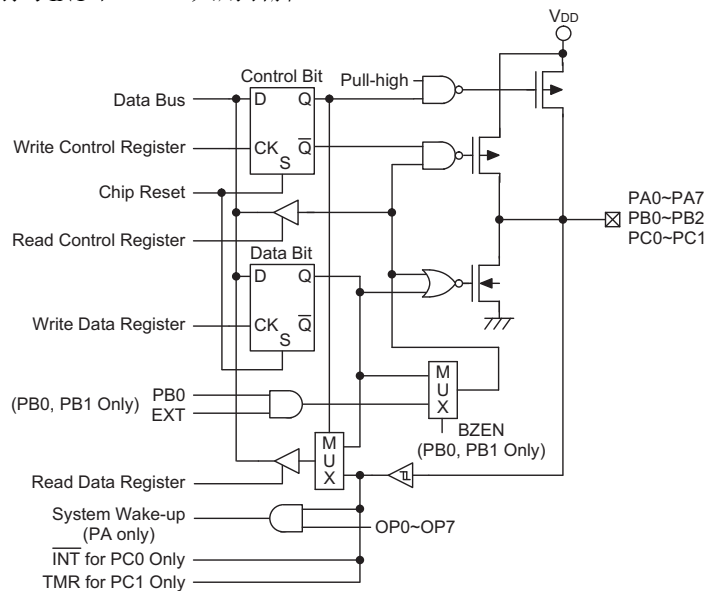
所有的输入/输出口都可以有上拉电阻的选择。一旦选择上拉电阻，所有的输入/输出口都具有上拉电阻。必须要注意的是：没有上拉电阻的输入/输出口工作在输入模式会产生浮空状态。

PB0 和 PB1 分别与 BZ 和 BZ 共用引脚。如果 BZ/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/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	I	I	O	O	O	O	O	O
PB1 输入/输出	I	O	O	O	I	I	I	O	O	O
PB0/PB1 模式	×	C	B	B	C	B	B	C	B	B
PB0 数据	×	×	0	1	D	0	1	D <sub>0</sub>	0	1
PB1 数据	×	D	×	×	×	×	×	D <sub>1</sub>	×	×
PB0 Pad 状态	I	I	I	I	D	0	B	D <sub>0</sub>	0	B
PB1 Pad 状态	I	D	0	B	I	I	I	D <sub>1</sub>	0	B

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 BZ；×：任意值；  
C：CMOS 输出；

PC0 和 PC1 分别与 INT 和 TMR 共用引脚。



### 输入/输出口

为了避免在浮空状态下功耗太大，建议将未用的或没有连结到外部的输入/输出口由软件指令设置成输出引脚。

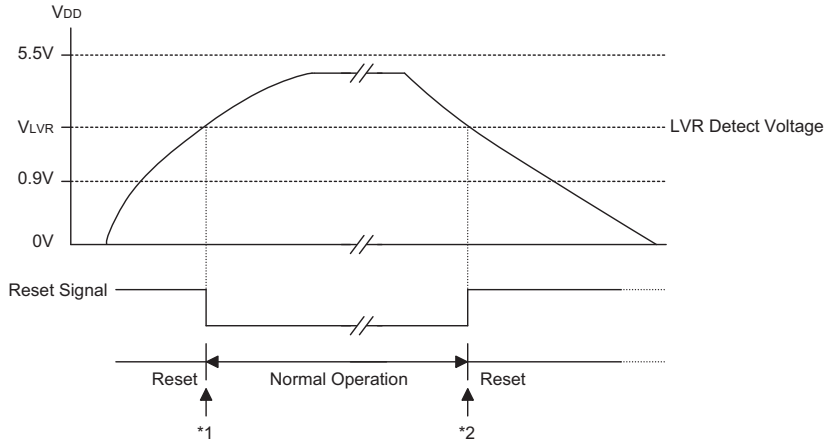
### 低电压复位 — LVR

为了监控器件的工作电压，单片机提供低电压复位功能。如果工作电压在 0.9V~VLVR 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~VLVR)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  $\overline{\text{RES}}$  信号的“或”的功能来执行系统复位。

VDD 与 VLVR 之间的关系如下所示：



### 低电压复位

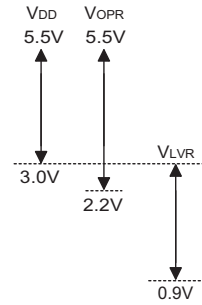
注：

- \*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

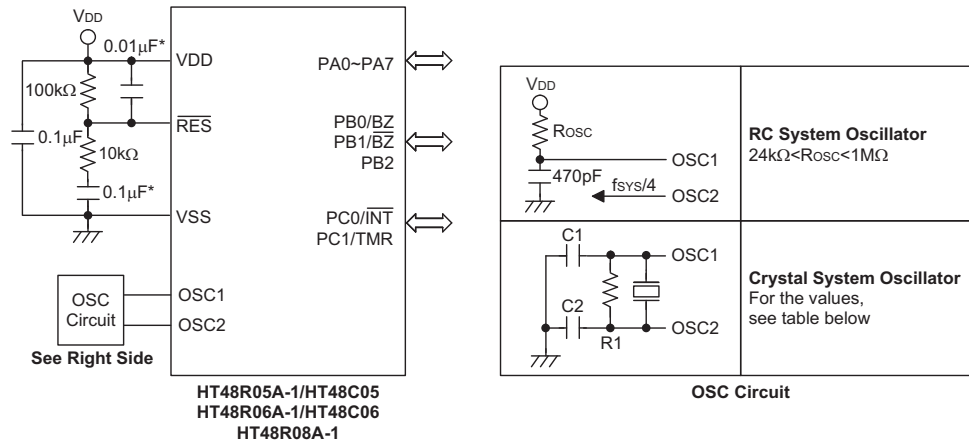
以下的表格，列出了这种单片机的各种类型的掩膜选项。所有的掩膜选项必须正确定义。

编号	选 项
1	WDT 时钟源：WDT 振荡或 fsys/4
2	WDT 打开/关闭
3	LVR 打开/关闭
4	清除看门狗指令条数：1 或 2 条清除 WDT
5	系统振荡选择：外部 RC 振荡/外部晶体振荡
6	上拉电阻 (PA~PC)：无上拉电阻或有上拉电阻
7	BZ 打开/关闭
8	PA0~PA7 唤醒功能：有/没有



注：V<sub>OPR</sub> 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:

电阻和电容值选取的原则是使 VDD 保持稳定并在  $\overline{RES}$  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  $\overline{RES}$  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 $\Omega$
4MHz 谐振器	10pF	12k $\Omega$
3.58MHz 晶振	0pF	10k $\Omega$
3.58MHz 谐振器	25pF	10k $\Omega$
2MHz 晶振和谐振器	25pF	10k $\Omega$
1MHz 晶振	35pF	27k $\Omega$
480KHz 谐振器	300pF	9.1k $\Omega$
455KHz 谐振器	300pF	10k $\Omega$
429KHz 谐振器	300pF	10k $\Omega$

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。

## 指令集摘要

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	2	无
SZ	[m]	1 <sup>(2)</sup>	无
SZA	[m]	1 <sup>(2)</sup>	无
SZ	[m].i	1 <sup>(2)</sup>	无
SNZ	[m].i	1 <sup>(2)</sup>	无
SIZ	[m]	1 <sup>(3)</sup>	无
SDZ	[m]	1 <sup>(3)</sup>	无
SIZA	[m]	1 <sup>(2)</sup>	无
SDZA	[m]	1 <sup>(2)</sup>	无
CALL	addr	2	无
RET		2	无
RET	A,x	2	无
RETI		2	无
<b>查表</b>			
TABRDC	[m]	2 <sup>(1)</sup>	无
TABRDL	[m]	2 <sup>(1)</sup>	无
<b>其它指令</b>			
NOP		1	无
CLR	[m]	1 <sup>(1)</sup>	无
SET	[m]	1 <sup>(1)</sup>	无
CLR	WDT	1	TO,PDF
CLR	WDT1	1	TO <sup>(4)</sup> ,PDF <sup>(4)</sup>
CLR	WDT2	1	TO <sup>(4)</sup> ,PDF <sup>(4)</sup>
SWAP	[m]	1 <sup>(1)</sup>	无
SWAPA	[m]	1	无
HALT		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 CLR 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。

**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

TO	PDF	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

TO	PDF	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

TO	PDF	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

TO	PDF	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，结果存放到数据存储器。  
 运算过程： $[m] \leftarrow ACC + [m]$   
 影响标志位

TO	PDF	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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

**AND A, x** 累加器与立即数做“与”运算，结果放入累加器  
 说明： 本指令把累加器值、立即数做逻辑与，结果存放到累加器。  
 运算过程： $ACC \leftarrow ACC \text{ “AND” } x$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

**ANDM A, [m]** 累加器与数据存储器做“与”运算，结果放入数据存储器  
 说明： 本指令把累加器值、数据存储器值做逻辑与，结果存放到数据存储器。  
 运算过程： $[m] \leftarrow ACC \text{ “AND” } [m]$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

**CALL addr** 子程序调用  
 说明： 本指令直接调用地址所在处的子程序，此时程序计数器加一，将此程序计数器值存到堆栈寄存器中，再将子程序所在处的地址存放到程序计数器中。  
 运算过程： $Stack \leftarrow PC+1$   
 $PC \leftarrow addr$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**CLR [m]** 清除数据存储器  
 说明： 本指令将数据存储器内的数值清零。  
 运算过程： $[m] \leftarrow 00H$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**CLR [m].i** 将数据存储器的第 i 位清“0”  
 说明： 本指令将数据存储器内第 i 位值清零。  
 运算过程： $[m].i \leftarrow 0$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**CLRWDT** 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： $WDT \leftarrow 00H$   
 $PDF \& TO \leftarrow 0$   
 影响标志位

TO	PDF	OV	Z	AC	C
0	0	—	—	—	—

**CLRWDT1** 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:  $WDT \leftarrow 00H^*$   
 $PDF \& TO \leftarrow 0^*$

影响标志位

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

**CLRWDT2** 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:  $WDT \leftarrow 00H^*$   
 $PDF \& TO \leftarrow 0^*$

影响标志位

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

**CPL** [m] 对数据存储器取反, 结果放入数据存储器

说明: 本指令是将数据存储器内保存的数值取反。

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

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

**CPLA** [m] 对数据存储器取反, 结果放入累加器

说明: 本指令是将数据存储器内保存的值取反后, 结果存放在累加器中。

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

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

**DAA [m]** 将加法运算后放入累加器的值调整为十进制数，并将结果放入数据存储器  
 说明 本指令将累加器高低四位分别调整为 BCD 码。如果低四位的值大于“9”或 AC=1，那么 BCD 调整就执行对原值加“6”，并且内部进位标志  $AC1 = \overline{AC}$ ，即 AC 求反；否则原值保持不变。如果高四位的值大于“9”或 C=1，那么 BCD 调整就执行对原值加“6”再加 AC1，并把 C 置位；否则 BCD 调整就执行对原值加 AC1，C 的值保持不变。结果存放在数据存储器中，只有进位标志位(C)受影响。

操作 如果  $ACC.3 \sim ACC.0 > 9$  或  $AC=1$   
 那么  $[m].3 \sim [m].0 \leftarrow (ACC.3 \sim ACC.0) + 6$ ， $AC1 = \overline{AC}$   
 否则  $[m].3 \sim [m].0 \leftarrow (ACC.3 \sim ACC.0)$ ， $AC1 = 0$   
 并且  
 如果  $ACC.7 \sim ACC.4 + AC1 > 9$  或  $C=1$   
 那么  $[m].7 \sim [m].4 \leftarrow (ACC.7 \sim ACC.4) + 6 + AC1$ ， $C=1$   
 否则  $[m].7 \sim [m].4 \leftarrow (ACC.7 \sim ACC.4) + AC1$ ， $C=C$

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

**DEC [m]** 数据存储器内容减 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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

**DECA [m]** 数据存储器内容减 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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

运算过程： $PC \leftarrow PC + 1$   
 $PDF \leftarrow 1$   
 $TO \leftarrow 0$

影响标志位

TO	PDF	OV	Z	AC	C
0	1	—	—	—	—

**INC**     **[m]**     数据存储器内容加 1，结果放入数据存储器  
 说明：         本指令将数据存储器内的数值加一，结果放回数据存储器。  
 运算过程：     [m] ← [m]+1  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

**INCA**   **[m]**     数据存储器内容加 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 ← [m]+1  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

**JMP**     **addr**    无条件跳转  
 说明：         本指令是将要跳到的目的地直接放到程序计数器内。  
 运算过程：     PC ← addr  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**MOV**    **A, [m]**    将数据存储器送至累加器  
 说明：         本指令是将数据存储器内的数值送到累加器内。  
 运算过程：     ACC ← [m]  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**MOV**    **A, x**     将立即数送至累加器  
 说明：         本指令是将立即数送到累加器内。  
 运算过程：     ACC ← x  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**MOV**    **[m], A**    将累加器送至数据存储器  
 说明：         本指令是将累加器值送到数据存储器内。  
 运算过程：     [m] ← ACC  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**NOP**      空指令

说明:            本指令不作任何运算, 而只将程序计数器加一。  
运算过程:         $PC \leftarrow PC+1$   
影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**OR**      **A, [m]**      累加器与数据存储器做“或”运算, 结果放入累加器  
说明:            本指令是把累加器、数据存储器值做逻辑或, 结果放到累加器。  
运算过程:         $ACC \leftarrow ACC \text{ "OR" } [m]$   
影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

**OR**      **A, x**      累加器与立即数做“或”运算, 结果放入累加器  
说明:            本指令是把累加器值、立即数做逻辑或, 结果放到累加器。  
运算过程:         $ACC \leftarrow ACC \text{ "OR" } x$   
影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

**ORM**    **A, [m]**      累加器与数据存储器做“或”运算, 结果放入数据存储器  
说明:            本指令是把累加器值、存储器值做逻辑或, 结果放到数据存储器。  
运算过程:         $[m] \leftarrow ACC \text{ "OR" } [m]$   
影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

**RET**                    从子程序返回  
说明:            本指令是将堆栈寄存器中的程序计数器值送回程序计数器。  
运算过程:         $PC \leftarrow Stack$   
影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**RET**      **A, x**      从子程序返回, 并将立即数放入累加器  
说明:            本指令是将堆栈寄存器中的程序计数器值送回程序计数器, 并将立即数送回累加器。  
运算过程:         $PC \leftarrow Stack$   
                       $ACC \leftarrow x$   
影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

运算过程:                 $PC \leftarrow Stack$   
                                $EMI \leftarrow 1$

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**RL**            **[m]**                    数据存储器左移一位, 结果放入数据存储器  
 说明:                    本指令是将数据存储器内的数值左移一位, 第 7 位移到第 0 位, 结果送回数据存储器。

运算过程:                 $[m].0 \leftarrow [m].7, [m].(i+1) \leftarrow [m].i; \quad (i=0\sim6)$

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

运算过程:                 $ACC.0 \leftarrow [m].7, ACC.(i+1) \leftarrow [m].i; \quad (i=0\sim6)$

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**RLC[m]**                带进位将数据存储器左移一位, 结果放入数据存储器  
 说明:                    本指令是将存储器内的数值与进位标志左移一位, 第 7 位取代进位标志, 进位标志移到第 0 位, 结果送回数据存储器。

运算过程:                 $[m].(i+1) \leftarrow [m].i; \quad (i=0\sim6)$

$[m].0 \leftarrow C$   
 $C \leftarrow [m].7$

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

**RLCA**    **[m]**                    带进位将数据存储器左移一位, 结果放入累加器  
 说明:                    本指令是将存储器内的数值与进位标志左移一位, 第七位取代进位标志, 进位标志移到第 0 位, 结果送回累加器。

运算过程:                 $ACC.(i+1) \leftarrow [m].i; \quad (i=0\sim6)$

$ACC.0 \leftarrow C$   
 $C \leftarrow [m].7$

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

**RR** [m] 数据存储器右移一位，结果放入数据存储器  
 说明：本指令是将存储器内的数值循环右移，第 0 位移到第 7 位，结果送回数据存储器。  
 运算过程： $[m].7 \leftarrow [m].0, [m].i \leftarrow [m].(i+1); (i=0\sim6)$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**RRA** [m] 数据存储器右移一位，结果放入累加器  
 说明：本指令是将数据存储器内的数值循环右移，第 0 位移到第 7 位，结果送回累加器，而数据存储器内的数值不变。  
 运算过程： $ACC.7 \leftarrow [m].0, ACC.i \leftarrow [m].(i+1); (i=0\sim6)$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**RRC** [m] 带进位将数据存储器右移一位，结果放入数据存储器  
 说明：本指令是将存储器内的数值加进位标志循环右移，第 0 位取代进位标志，进位标志移到第 7 位，结果送回存储器。  
 运算过程： $[m].i \leftarrow [m].(i+1); (i=0\sim6)$   
 $[m].7 \leftarrow C$   
 $C \leftarrow [m].0$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

**RRC A** [m] 带进位将数据存储器右移一位，结果放入累加器  
 说明：本指令是将数据存储器内的数值加进位标志循环右移，第 0 位取代进位标志，进位标志移到第 7 位，结果送回累加器，数据存储器内的数值不变。  
 运算过程： $ACC.i \leftarrow [m].(i+1); (i=0\sim6)$   
 $ACC.7 \leftarrow C$   
 $C \leftarrow [m].0$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

**SBC** A,[m] 累加器与数据存储器、进位标志相减，结果放入累加器  
 说明：本指令是把累加器值减去数据存储器值以及进位标志的取反，结果放到累加器。  
 运算过程： $ACC \leftarrow ACC + [\overline{m}] + C$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

**SBCM A,[m]** 累加器与数据存储器、进位标志相减，结果放入数据存储器  
 说明： 本指令是把累加器值减去数据存储器值以及进位标志取反，结果放到数据存储器。  
 运算过程： $[m] \leftarrow ACC + [\bar{m}] + C$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

**SDZ [m]** 数据存储器减 1，如果结果为“0”，则跳过下一条指令  
 说明： 本指令是把数据存储器内的数值减 1，判断是否为 0，若为 0 则跳过下一条指令，即如果结果为零，放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以取得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。  
 运算过程： 如果 $[m]-1=0$ ，跳过下一条指令执行再下一条。  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**SDZA [m]** 数据存储器减 1，将结果放入累加器，如果结果为“0”，则跳过下一条指令  
 说明： 本指令是把数据存储器内的数值减 1，判断是否为 0，为 0 则跳过下一行指令并将减完后数据存储器内的数值送到累加器，而数据存储器内的值不变，即若结果为 0，放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以取得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。  
 运算过程： 如果 $[m]-1=0$ ，跳过下一条指令执行再下一条。  
 $ACC \leftarrow ([m]-1)$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**SET [m]** 置位数据存储器  
 说明： 本指令是把存储器内的数值每个位置为 1。  
 运算过程： $[m] \leftarrow FFH$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**SET [m]. i** 将数据存储器的第 i 位置“1”  
 说明： 本指令是把存储器内的数值的第 i 位置为 1。  
 运算过程： $[m].i \leftarrow 1$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

运算过程：     如果 ([m]+1=0)，跳过下一行指令； [m] ← [m]+1  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

运算过程：     如果 [m]+1=0，跳过下一行指令； ACC ← ([m]+1)  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

运算过程：     如果 [m].i≠0，跳过下一行指令。  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**SUB**         A, [m]     累加器与数据存储器相减，结果放入累加器  
 说明：         本指令是把累加器值、数据存储器值相减，结果放到累加器。

运算过程：     ACC ← ACC + [m] + 1  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

**SUB**         A, x         累加器与立即数相减，结果放入累加器  
 说明：         本指令是把累加器值、立即数相减，结果放到累加器。

运算过程：     ACC ← ACC + x + 1  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

**SUBM A, [m]** 累加器与数据存储器相减，结果放入数据存储器  
 说明： 本指令是把累加器值、存储器值相减，结果放到存储器。  
 运算过程： $[m] \leftarrow ACC + [\bar{m}] + 1$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

**SWAP [m]** 交换数据存储器的高低字节，结果放入数据存储器  
 说明： 本指令是将数据存储器的低四位和高四位互换，再将结果送回数据存储器。  
 运算过程： $[m].7 \sim [m].4 \leftrightarrow [m].3 \sim [m].0$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**SWAPA [m]** 交换数据存储器的高低字节，结果放入累加器  
 说明： 本指令是将数据存储器的低四位和高四位互换，再将结果送回累加器。  
 运算过程： $ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$   
 $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**SZ [m]** 如果数据存储器为“0”，则跳过下一条指令  
 说明： 本指令是判断数据存储器内的数值是否为0，为0则跳过下一行指令，即放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。  
 运算过程： 如果  $[m] = 0$ ，跳过下一行指令。  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**SZA [m]** 数据存储器送至累加器，如果内容为“0”，则跳过下一条指令  
 说明： 本指令是判断存储器内的数值是否为0，若为0则跳过下一行指令，即放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以得正确的指令(二个指令周期)。并把存储器内值送到累加器，而存储器的值保持不变。否则执行下一条指令(一个指令周期)。  
 运算过程： 如果  $[m] = 0$ ，跳过下一行指令，并  $ACC \leftarrow [m]$ 。  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**SZ**      **[m].i**    如果数据存储器的第 i 位为“0”，则跳过下一条指令  
 说明：      本指令是判断存储器内第 i 位值是否为 0，若为 0 则跳过下一行指令，即放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。  
 运算过程：    如果 [m].i=0，跳过下一行指令。  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**TABRDC [m]**    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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**TABRDL [m]**    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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**XOR**    **A, [m]**    累加器与立即数做“异或”运算，结果放入累加器  
 说明：      本指令是把累加器值、 数据存储器值做逻辑异或，结果放到累加器。  
 运算过程：    ACC←ACC “XOR” [m]  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

**XORM**   **A, [m]**    累加器与数据存储器做“异或”运算，结果放入数据存储器  
 说明：      本指令是把累加器值、数据存储器值做逻辑异或，结果放到数据存储器。  
 运算过程：    [m]←ACC “XOR” [m]  
 影响标志位

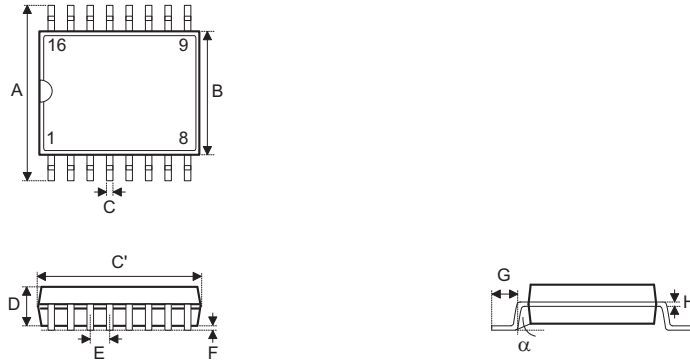
TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

**XOR**    **A, x**      累加器与数据存储器做“异或”运算，结果放入累加器  
 说明：      本指令是把累加器值与立即数做逻辑异或，结果放到累加器。  
 运算过程：    ACC←ACC “XOR” x  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

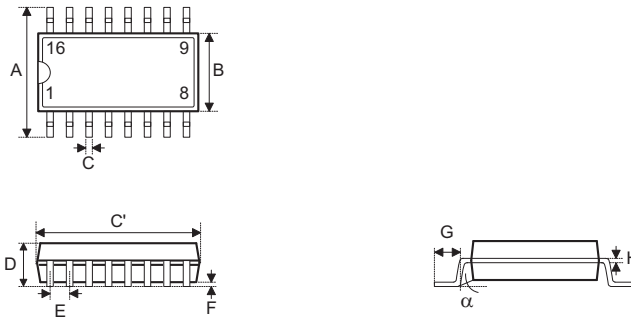
封装信息:

16-pin SSOP (150mil)外形尺寸



标号	尺寸 (mil)		
	Min	Nom	Max
A	228	--	244
B	150	--	157
C	8	--	12
C'	189	--	197
D	54	--	60
E	--	25	--
F	4	--	10
G	22	--	28
H	7	--	10
$\alpha$	0°	--	8°

16-pin NSOP (150mil)外形尺寸



标号	尺寸 (mil)		
	Min	Nom	Max
A	228	--	244
B	150	--	157
C	12	--	20
C'	386	--	394
D	--	--	69
E	--	50	--
F	4	--	10
G	16	--	50
H	7	--	10
$\alpha$	0°	--	8°

18-pin DIP (300mil)外形尺寸

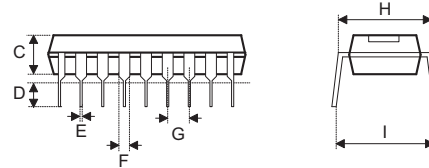
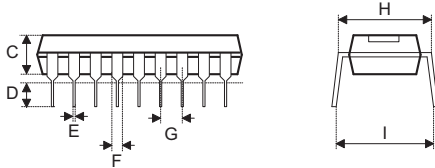
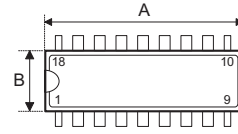
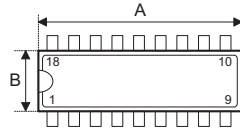


Fig1. Full Lead Packages

Fig2. 1/2 Lead Packages

●MS\_001d (fig1)

标号	尺寸 (mil)		
	Min	Nom	Max
A	880	--	920
B	240	--	280
C	115	--	195
D	115	--	150
E	14	--	22
F	45	--	70
G	--	100	--
H	300	--	325
I	--	--	430

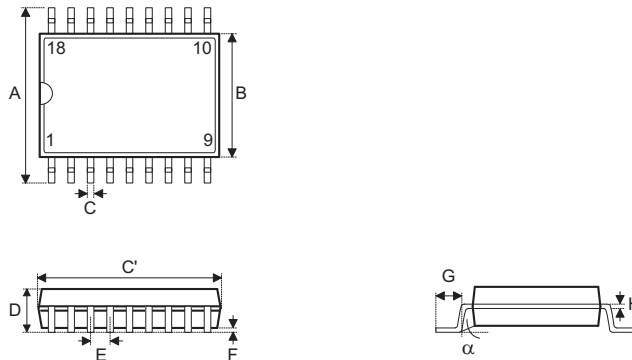
●MS\_001d (fig1)

标号	尺寸 (mil)		
	Min	Nom	Max
A	845	--	880
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)		
	Min	Nom	Max
A	845	--	885
B	275	--	295
C	120	--	150
D	110	--	150
E	14	--	22
F	45	--	60
G	--	100	--
H	300	--	325
I	--	--	430

18-pin SOP (300mil)外形尺寸

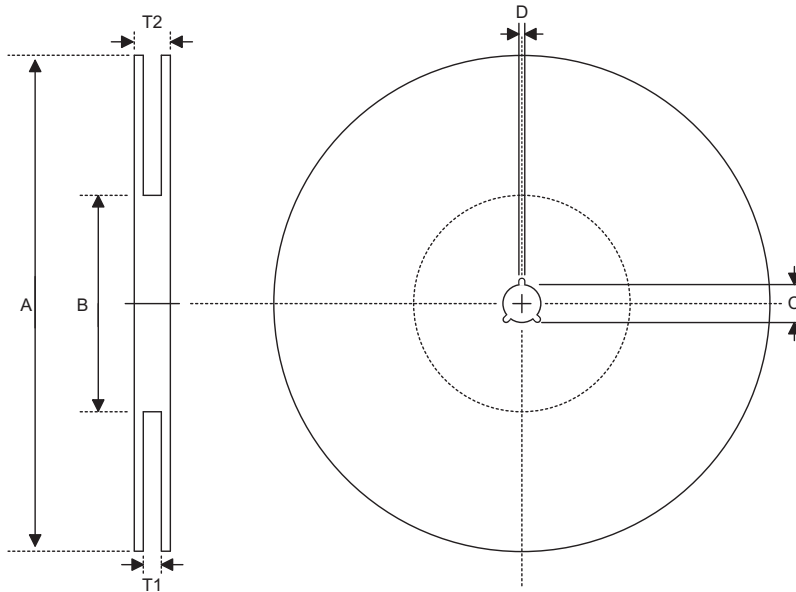


●MS\_013

标号	尺寸 (mil)		
	Min	Nom	Max
A	393	--	419
B	256	--	300
C	12	--	20
C'	447	--	463
D	--	--	104
E	--	50	--
F	4	--	12
G	16	--	50
H	8	--	13
$\alpha$	0°	--	8°

包装带和卷轴规格:

卷轴尺寸:



SSOP 16S

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 <sup>+0.5/-0.2</sup>
D	缝宽	2.0±0.5
T1	轮缘宽	12.8 <sup>+0.3/-0.2</sup>
T2	卷轴宽	18.2±0.2

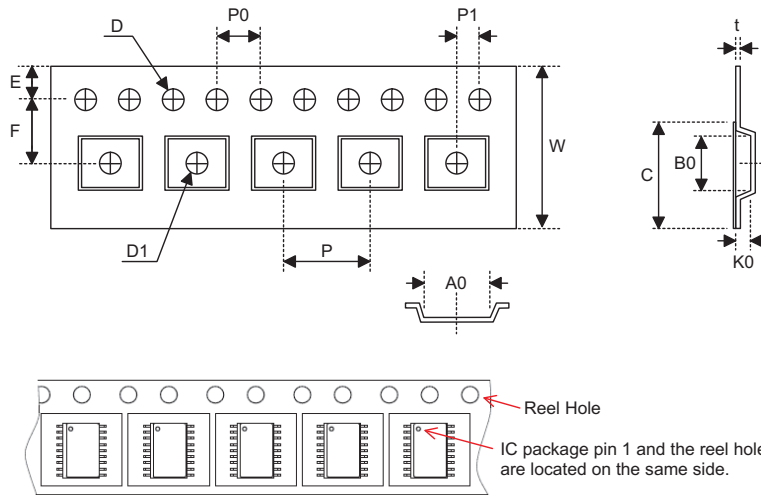
SOP 16N (150mil)

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 <sup>+0.5/-0.2</sup>
D	缝宽	2.0±0.5
T1	轮缘宽	16.8 <sup>+0.3/-0.2</sup>
T2	卷轴宽	22.2±0.2

SOP 18W

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 <sup>+0.5/-0.2</sup>
D	缝宽	2.0±0.5
T1	轮缘宽	24.8 <sup>+0.3/-0.2</sup>
T2	卷轴宽	30.2±0.2

运输带尺寸:



SSOP 16S

标号	描述	尺寸(mm)
W	运输带宽	12.0 <sup>+0.3/-0.1</sup>
P	空穴间距	8.0±0.1
E	穿孔位置	1.75±0.1
F	空穴至穿孔距离 (宽度)	5.5±0.1
D	穿孔直径	1.55±0.1
D1	空穴中之小孔直径	1.5 <sup>+0.25/-0.0</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	6.4±0.1
B0	空穴宽	5.2±0.1
K0	空穴深	2.1±0.1
t	传输带厚度	0.30±0.05
C	覆盖带宽度	9.3±0.1

SOP 16N (150mil)

标号	描述	尺寸(mm)
W	运输带宽	16.0±0.3
P	空穴间距	8.0±0.1
E	穿孔位置	1.75±0.1
F	空穴至穿孔距离 (宽度)	7.5±0.1
D	穿孔直径	1.55 <sup>+0.17/-0.0</sup>
D1	空穴中之小孔直径	1.5 <sup>+0.25/-0.0</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	6.5±0.1
B0	空穴宽	10.3±0.1
K0	空穴深	2.1±0.1
t	传输带厚度	0.3±0.05
C	覆盖带宽度	13.3±0.1

SOP 18W

标号	描述	尺寸(mm)
W	运输带宽	24.0 <sup>+0.3/-0.1</sup>
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)	11.5±0.1
D	穿孔直径	1.5±0.1
D1	空穴中之小孔直径	1.5 <sup>+0.25/-0.0</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	10.9±0.1
B0	空穴宽	12.0±0.1
K0	空穴深	2.8±0.1
t	传输带厚度	0.3±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

**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 © 2008 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>