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

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

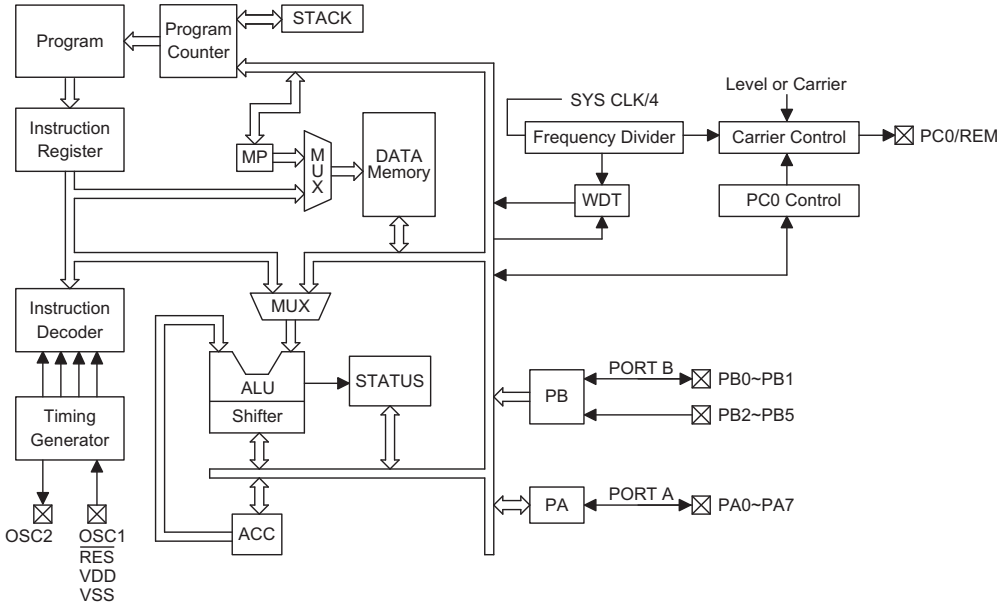
- 工作电压：2.0V ~3.6V
- 10 个双向输入/输出口
- 4 个斯密特触发输入口
- 一个载波输出口（1/2 或 1/3 占空比）
- 内置晶体和 RC 振荡电路
- 看门狗定时器
- 1K×14 程序存储器（EPROM）
- 32×8 数据存储器（RAM）
- HALT 和唤醒功能以降低功耗
- 62 条指令
- 系统频率为 4MHz 时，指令周期为 1μs
- 所有指令在 1 或 2 个指令周期内完成
- 查表指令，表格内容字长 14 位
- 一层堆栈
- 位操作指令
- 低电压复位功能
- 20-pin 的 SSOP 封装

概述

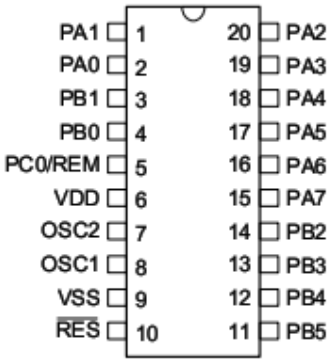
HT48RA0-2/ HT48CA0-2 为八位高性能精简指令集单片机，专为多输入/输出口的产品而设计的。HT48RA0-2 是 OTP 版本，和掩膜版本芯片 HT48CA0-2 在引脚和功能上完全相同。

拥有低功耗、I/O 口稳定性高、定时器功能、振荡选择、省电和唤醒功能、看门狗定时器、蜂鸣器驱动、以及低价位等优势，使此款多功能芯片可以广泛地适用于各种应用，例如红外遥控器及各种电子系统的控制器。

方框图



引脚图



HT48RA0-2/HT48CA0-2
20 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	—	8 位双向输入/输出口，带上拉电阻。每一位都能由软件指令设定为 NMOS 输出或斯密特触发输入。
PB0,PB1	输入/输出	唤醒功能	2 位双向输入/输出口，带上拉电阻。每一位都可在掩膜选项中设置为带唤醒功能的输入口，可由软件指令设定为 NMOS 输出或斯密特触发输入。
PB2~PB5	输入	唤醒功能	4 位斯密特触发输入口，带上拉电阻。每一位都能在掩膜选项中设置为唤醒输入。
PC0/REM	输出	电平或载波	电平或载波输出口。 在掩膜选项中 PC0 可以被设为 CMOS 输出或载波输出口。
VDD	—	—	正电源。
VSS	—	—	负电源，接地。
OSC1 OSC2	输入 输出	晶体振荡或 RC 振荡	OSC1 和 OSC2 连接 RC 或 Crystal（由掩膜选项选择）产生内部系统时钟。在 RC 模式下，OSC2 是一个系统时钟四分频的输出端（NMOS 漏极开路输出）。
$\overline{RES}$	输入	—	斯密特触发复位输入端，低电平有效。

极限参数

电源供应电压 …… V_{SS} – 0.3V～Vss +4.0V
端口输入电压 …… V_{SS} – 0.3V～V_{DD} +0.3V
端口总灌电流 …… 150mA
总功耗 …… 500mW

储存温度 …… - 50℃～125℃
工作温度 …… - 40℃～ 85℃
端口总源电流 …… -100mA

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

直流电气特性

Ta=25℃

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压	—	—	2.0	—	3.6	V
I_{DD}	工作电流	3V	无负载 fsys=4MHz	—	0.7	1.5	mA
I_{STB}	静态电流	3V	无负载 暂停模式	—	—	1	μA
V_{IL1}	输入/输出口的低电平输入电压	3V	—	0	—	0.2 V_{DD}	V
V_{IH1}	输入/输出口的高电平输入电压	3V	—	0.8V_{DD}	—	V_{DD}	V
V_{IL2}	低电平输入电压($\overline{RES}$)	3V	—	0	—	0.4V_{DD}	V
V_{IH2}	高电平输入电压($\overline{RES}$)	3V	—	0.9V_{DD}	—	V_{DD}	V
V_{LVR}	低电压复位值	—	温度=25℃*	1.8	1.9	2.0	V
I_{OL}	输入/输出口灌电流	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
I_{OH}	PC0/REM 输出源电流	3V	V_{OH}=0.9V_{DD}	-2	-4	—	mA
R_{PH}	上拉电阻	3V	—	20	60	100	KΩ

注：“*”只保证在 25℃的条件下

交流电气特性

Ta=25℃

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
f_{sys}	系统时钟	3V	—	400	—	4000	KHz
t_{RES}	外部复位低电平的脉宽	—	—	1	—	—	μs
t_{SST}	系统启动延迟周期	—	上电或从 HALT 唤醒	—	1024	—	t_{sys}
t_{LVR}	低电压复位延迟时间	—	—	0.25	1	2	ms

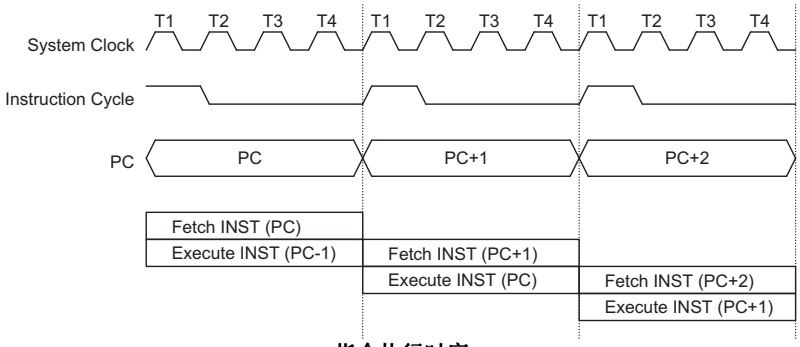
注：$t_{SYS}=1/f_{SYS}$

系统功能说明

指令执行时序

HT48RA0-2/HT48CA0-2 系统可外接晶体振荡器或陶瓷谐振器产生振荡频率。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。一个指令周期包含了四个系统时钟周期。

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



指令执行时序

程序计数器 — PC

10 位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。程序计数器的最大寻址范围为 1024 个地址。

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、初始复位或从一个子程序返回，PC 会通过装载每条指令的相应地址来执行程序转移。

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

程序计数器的低位字节（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。

当一个控制转移指令发生时，就需要有一个附加的空指令周期。

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

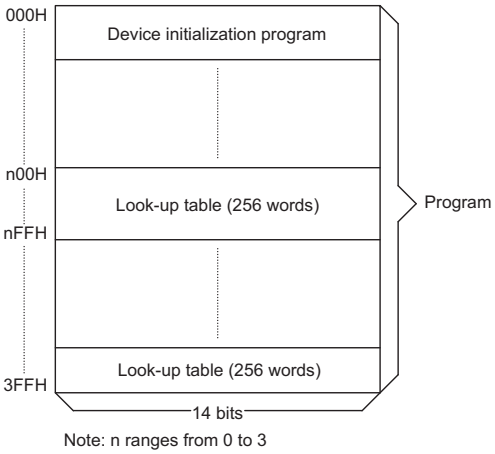
程序计数器

注意：*9 ~ *0 ： 程序计数器位　　S9 ~ S0 ： 堆栈寄存器位

#9 ~ #0 ： 指令代码位　　@7 ~ @0 ： PCL 位

程序存储器 — ROM

程序存储器（ROM）被用来存放要执行的指令的代码。还包括数据和表格。一共为 1024×14 位。可以由程序计数器或表格指针来寻址。



程序存储器

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。
- 表格区
  程序存储器内的任何地址都可被用来作为表格地址使用。查表指令 TABRDC [m]（查当前页，1 页=256 个字节）和 TABRDL [m]（查最后一页）将所指地址的低字节传到指定的存储器中，而高字节传送到 TBLH （08H）。表格中只有低位字节被完整传到到目标地址中，而其它位则传送到 TBLH 的低位，剩余的二位被作为 0 读出。表格中的高位字节 TBLH 为只读寄存器。而表格指针（TBLP；07H）是可以读写的寄存器，表格指针指向要读的表格地址。在访问表格以前，通过对 TBLP 寄存器赋值来指明表格地址。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，表格地址指明的这些位置也可以作为正常的程序存储器来使用。

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

表格区

注：*9 ~ *0 ： 表格地址位　　P9~P8 ： 当前程序计数器位　　@7 ~ @0 ： 表格指针位

堆栈寄存器 — STACK

堆栈寄存器（STACK）是一个用来保存 PC 值的特殊存储单元。HT48RA0-2/HT48CA0-2 的堆栈为 1 级。堆栈寄存器既不是数据存储器的一部分，也不是程序存储器的一部分，而且它既不能读出，也不能写入。堆栈位置由堆栈指针（SP）来确定。堆栈指针（SP）也不能读/写。一旦发生子程序调用时，程序计数器（PC）的值会被压入堆栈，在子程序调用结束时，通过一条返回指令（RET），堆栈将原先压入堆栈的内容弹出，重新装入程序计数器（PC）中。在系统复位后，堆栈指针会指向堆栈顶部。

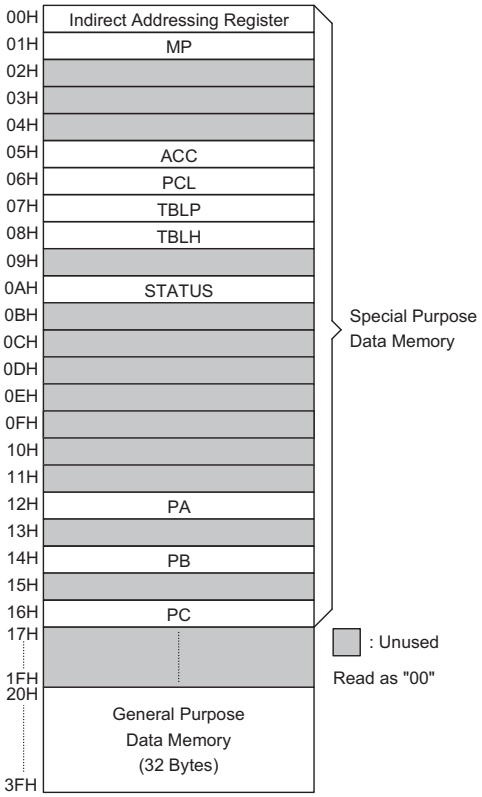
如果堆栈满了，接着又执行一个子程序调用（CALL），那么堆栈会产生溢出，而使第一个进入堆栈的内容将会丢失。（只有最后那个返回地址会被保留着）。

数据存储器 — RAM

数据存储器 RAM 由 42×8 位组成。它可分成两个功能组：特殊功能寄存器和通用数据存储器（32×8）。这两个功能组的大部分单元可以读写，而某些单元只能读，不能写。

特殊功能寄存器包括间接寻址寄存器（00H），间接寻址指针寄存器（MP；01H），累加器（ACC；05H），程序计数器低位字节寄存器（PCL；06H），表格指针寄存器（TBLP；07H），表格高字节寄存器（TBLH；08H），状态寄存器（STATUS；0AH），输入/输出口寄存器（PA；12H，PB；14H，PC；16H）。20H 以前的剩余单元都被保留为将来进一步扩展使用。读取这些被保留单元的值，都将返回 00H。通用数据存储器地址 20H-3FH 作为程序数据和控制信息使用。

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



数据存储器

间接寻址寄存器

地址 00H 是作为间接寻址寄存器。它没有物理结构。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。

间接寻址指针寄存器 MP 的宽度是 7 位，MP 的第 7 位是没有定义的。若读它们将返回“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

累加器（ACC）与算术逻辑单元（ALU）运算相联系。它对应于 RAM 的地址 05H，并能与立即数进行操作，在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（ALU）是执行 8 位算术、逻辑运算的电路，它提供有以下功能：

- 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（INC，DEC）
- 分支判断（SZ，SNZ，SIZ，SDZ…）

ALU 不仅可以储存数据运算的结果，还会改变状态寄存器的值。

状态寄存器 — STATUS

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）、暂停模式标志位（PDF）、看门狗定时器溢出标志位（TO）组成。该寄存器所不仅记录状态信息，而且还可控制运算顺序。

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

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

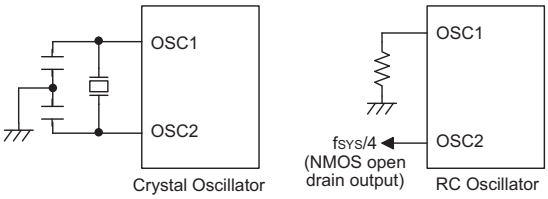
位	符号	功　　能
0	C	如果在加法运算中结果产生了进位，或在减法运算中结果不发生借位,那么 C 被置位；反之，C 被清除。它也可被一个带循环进位指令而影响。
1	AC	在加法运算中低四位产生了向高四位进位，或减法运算中低四位不发生从高四位借位，AC 被置位；反之，AC 被清除。
2	Z	算术运算或逻辑运算的结果为零则 Z 被置位；反之，Z 被清除。
3	OV	如果运算结果向最高位进位，但最高位并不产生进位输出，那么 OV 被置位；反之，OV 被清除。
4	PDF	系统上电或执行了 CLR WDT 指令，PDF 被清除。执行 HALT 指令 PDF 被置位。
5	TO	系统上电或执行了 CLR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

STATUS (0AH)寄存器

振荡电路

HT48RA0-2/HT48CA0-2 有两种振荡电路。

通过掩膜选项选择，HT48RA0-2/HT48CA0-2 可以选择晶体振荡或 RC 振荡模式。两者都为产生系统时钟而设计的。不管用哪种振荡器，其信号都支持系统时钟。在 HALT 模式下，系统时钟会停振，并忽略外部输入信号，由此来节省功耗。



系统振荡器

如果采用 RC 振荡器，那么在 Vss 与 OSC1 之间要接一个外接电阻。其阻值范围为 51KΩ~1MΩ。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 振荡方式是一种低成本方案，但是振荡频率会由于 V_{DD}、温度及芯片自身参量的漂移，而产生误差。因此，对振荡器要求精确度高的场合，RC 振荡器是不适用的。

如果选用晶体振荡器，在 OSC1 与 OSC2 之间连接一个晶体，被用来提供晶体振荡器所要的反馈（Feedback）和相位位移（Phase Shift）。除此以外，不再需要其他外部元件。另外，可在 OSC1 与 OSC2 之间接一个谐振器（Resonator） 来取代晶体振荡器，用来得到参考频率，但是需要外接两个接地电容器。

看门狗定时器

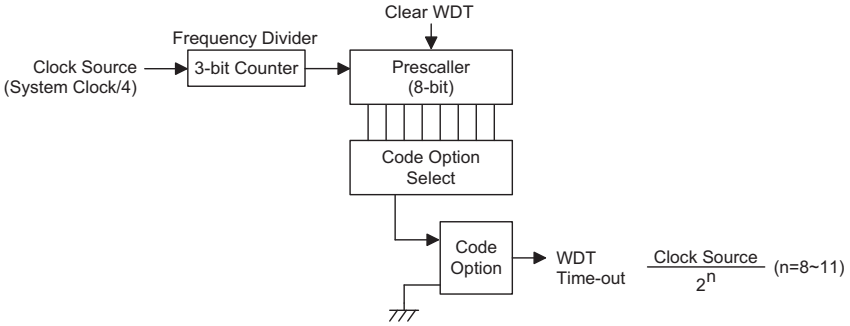
看门狗定时器（WDT）的时钟源是由指令时钟（系统时钟 4 分频）来提供的，时钟源通过分频器和预分频器来设置不同的溢出时间。

$$\text{WDT 溢出时间} = \frac{\text{时钟源}}{2^n}$$

这里 n=8 ~ 11， 由掩膜选项来选择。

看门狗定时器是用来防止程序的不正常运行或是跳到未知或不希望去的地址，而导致不可预见的结果。看门狗定时器可以被掩膜选项选项禁止。如 WDT 被禁止，所有与 WDT 有关的执行都导致一个空操作，而 WDT 会失去保护的功能。在这种情况下，只能由外部逻辑来重新启动系统。

在正常运作下，WDT 溢出会使系统复位并置位 TO 位。有三种方法可以清除 WDT 的值：外部复位（低电平输入到$\overline{\text{RES}}$端），用软件指令和 HALT 指令三种。有两组软件指令：一组是单条指令 CLR WDT，另一组是二条指令组成 CLR WDT1 及 CLR WDT2 。这两组指令中，只能选取其中一种。由掩膜选项中的 CLR WDT 次数选项决定。如果“CLR WDT”被选择（即 CLR WDLT 次数为 1），那么只要执行 CLR WDT 指令就会清除 WDT。在 CLR WDT1 和 CLR WDT2 被选的情况下，（即 CLR WDT 次数为 2），那么要交替执行二条指令才会连续清除 WDT，否则，WDT 会由于超时溢出而使系统复位。



看门狗定时器

暂停模式 — HALT

暂停模式是由 HALT 指令来实现的，执行后情况如下：

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

外部复位或 PB 口的下降沿信号，可使系统脱离暂停模式。外部复位能使系统初始化。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或执行 CLR WDT 指令被清除，而它的置位是由于执行了 HALT 指令。如 WDT 产生溢出，会使 TO 标志位置位，还能产生唤醒使得程序计数的 PC 和堆栈指针 SP 复位。其他状态都保持不变。

PB 端口的唤醒看作为正常运行的继续。PB 口的每一位都可以通过掩膜选项来单独设定为对系统的唤醒。如果唤醒是来自于输入/输出口的信号变化，程序会重新继续执行下一条命令。

当唤醒事件发生时，要花 1024 个 t_{sys}（系统时钟周期）后，系统重新正常运行。这就是说，在唤醒后将插入了一个等待时间。

为了减小功耗，在进入 HALT 模式之前必须要小心处理输入/输出端口引脚。

复 位

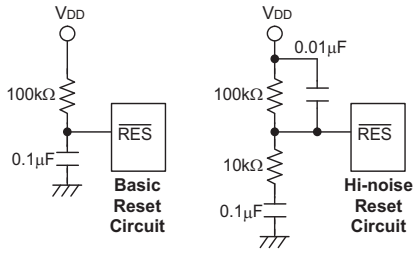
有三种方法可以产生复位：

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

复位发生时，某些寄存器的内容会保持不变，大多数寄存器的值会复位到初始化状态。通过测试 PDF 标志位和 TO 标志位，程序能分辨不同的复位。

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

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



复位电路

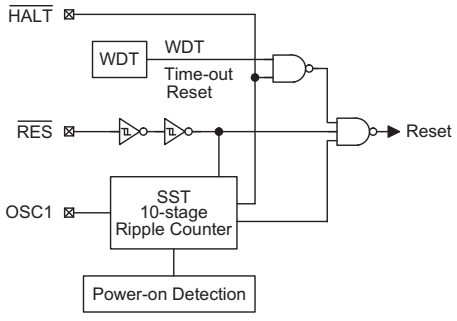
注意：大多数应用中使用基本复位电路即可，如方案中外部噪音或杂迅比较厉害，就请选用高抗杂迅电路

TO	PDF	复位条件
0	0	电源上电复位
u	u	正常运作时由$\overline{\text{RES}}$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

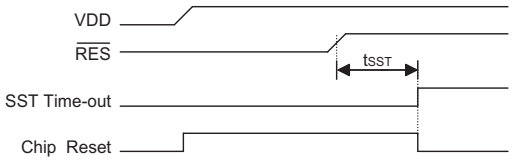
注释：“u”表示未变化。

系统复位时各功能单元的状态如下所示：

程序计数器(PC)	000H
看门狗预分频器	清除
输入/输出端口	输入模式
堆栈指针 SP	指向栈顶
载波输出	低电平



复位电路结构



复位时序

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 超时溢出（正常工作）	RES 端复位（正常工作）	RES 端复位（暂停模式）	WDT 超时溢出（暂停模式）*
PC	000H	000H	000H	000H	000H
MP	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	0011 1111	0011 1111	0011 1111	0011 1111	uuuu uuuu
PC	---- ---1	---- ---1	---- ---1	---- ---1	---- ---u

注意： “U”表示不变化。

“X”表示不确定。

载 波 — Carrier

HT48RA0-2/HT48CA0-2 提供一个载波输出口，其与 PC0 共用一个输入输出口。由掩膜选项决定是载波输出（REM）或电平输出（PC0）。在选择载波输出的情况下，设置 PC0=“0”来允许载波输出，设置 PC0=“1”来禁止载波输出，回到低电平状态。

载波的时钟源是由指令时钟（系统时钟 4 分频）来提供的，通过分频器的处理来产生各种不同的载波频率。

$$\text{载波频率} = \frac{\text{时钟频率源}}{m \times 2^n}$$

这里 m=2 或 3 ，n=0 ~ 3 是由掩膜选项选项来选择。

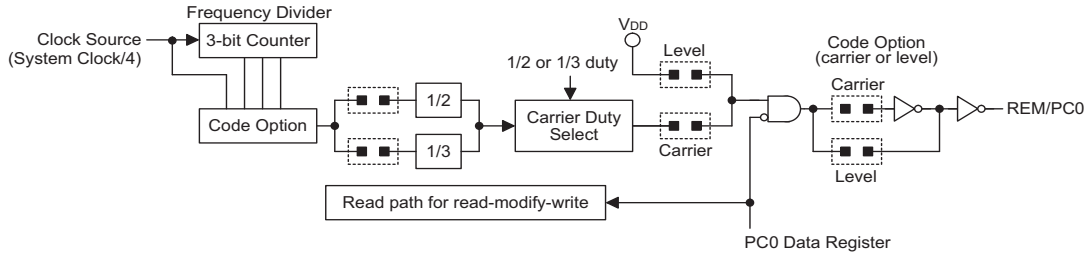
如果 m=2，那么载波输出的占空比是 1/2 ，如果 m=3，那么载波输出的占空比可以由掩膜选项决定为 1/2 或 1/3。（n=0 例外）。

载波输出的占空比的详细的选择如下所示：

m x 2^n	占空比
2，4，8，16	1/2
3	1/3
6，12，24	1/2 或/3

载波频率选择例子如下表所示：

系统频率	载波频率	占空比	m x 2^n
455kHz	37.92kHz	1/3only	3
	56.9kHz	1/2only	2



载波/电平输出

输入/输出端口

HT48RA0-2/HT48CA0-2 提供一个 8 位双向输入/输出端口、一个 6 位输入 2 位输入/输出端口、一个 1 位的输出端口：PA，PB，PC 分别对应到 RAM 的[12H] 、[14H]和[16H]。PA 的每一位可由掩膜选项确定为 NMOS 输出或斯密特触发输入，带上拉电阻。而 PB 的 PB0 ~ PB1 具有 PA 的相同的结构，而 PB2 ~ PB7 只能用于输入（斯密特触发输入，带上拉电阻）。PC 只有一位输出，与载波输出共享一个输出口。如果选择电平输出，那么 PC 是 CMOS 输出端口。

在 PA 和 PB 用于输入时，这些端口不具有锁存功能，输入数据必须在 MOV A，[m] (m=12H 或 14H) 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，直到执行下一个写操作。

在 PA 和 PB0~PB1 用于输入时，需要注意，在从引脚读入数据以前，应该先将相关引脚置“1”。即，要先执行“SET [m].i”(PA: i=0~7,PB: i=0~1)指令来禁止相关的 NMOS 输出，然后用“MOV A，[m]”来获得稳定的数据。

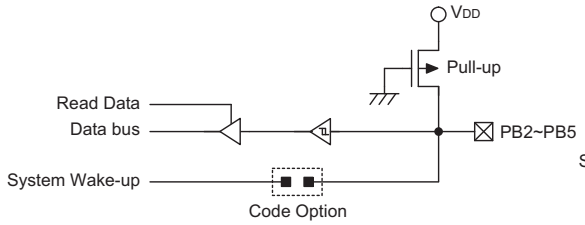
芯片复位后，PA 和 PB 保持高电平输入状态，如果选择电平输出，PC 则保持高电平输出状态。

PA、PB0~PB1、和 PC 的每一位输出锁存都能被 SET [m].i 或 CLR [m].i 指令置位或清除(m=12H，14H 或 16H)。

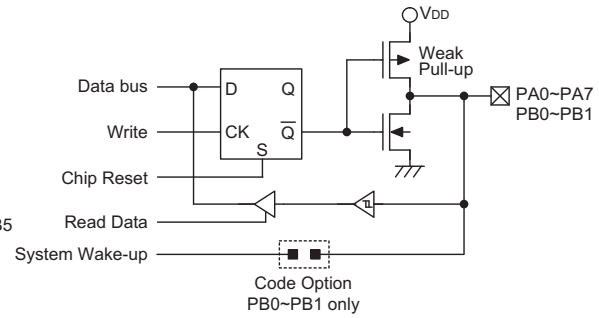
某些指令会首先输入数据然后进行输出操作。例如，SET [m].i ，CLR [m].i，CPL [m]和 CPLA [m]指令，读取输入口的状态到 CPU，执行这个被定义的操作（位操作），然后将结果写回这个锁存器或累加器。

由掩膜选项来选择，PB 口的每一根线都具有唤醒系统的能力。PC 口的高七位在物理上并不提供，读它们时结果为 0，而写入的结果是空操作。

注意：PB 特殊寄存器（14H）的第 6、7 位对于 HT48RA0-2/HT48CA0-2 是没有作用的。任何对此位的读动作，读出的都是“0”。用户如果要将程序从 HT48RA0A/HT48CA0A 或者 HT48RA0-1/HT48CA0-1 上转到 HT48RA0-2/HT48CA0-2，则必须非常小心谨慎。

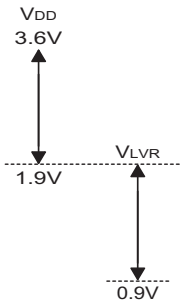


PB 输入口



PA、PB 输入/输出口

低电压复位 — LVR

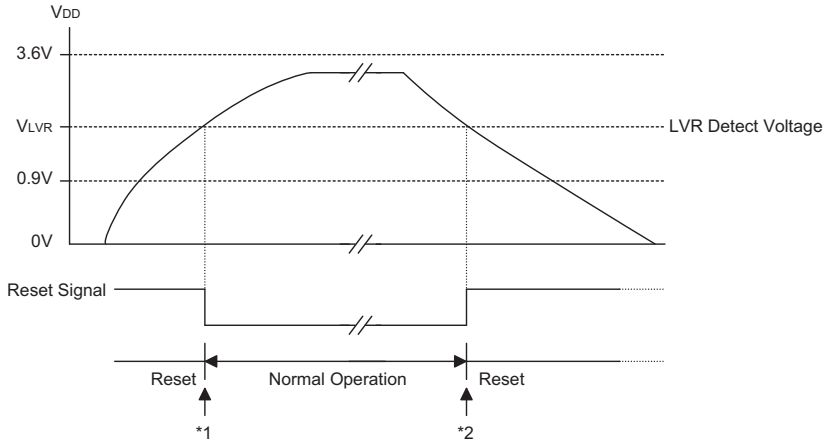


为了监控器件的工作电压，单片机提供低电压复位功能。如果工作电压在 0.9V～V_{LVR} 之间，例如电池电压的变化，那么 LVR 会自动使器件产生内部复位。

LVR 功能说明如下：

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

V_{DD} 与 V_{LVR} 之间的关系如下所示：



低电压复位

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

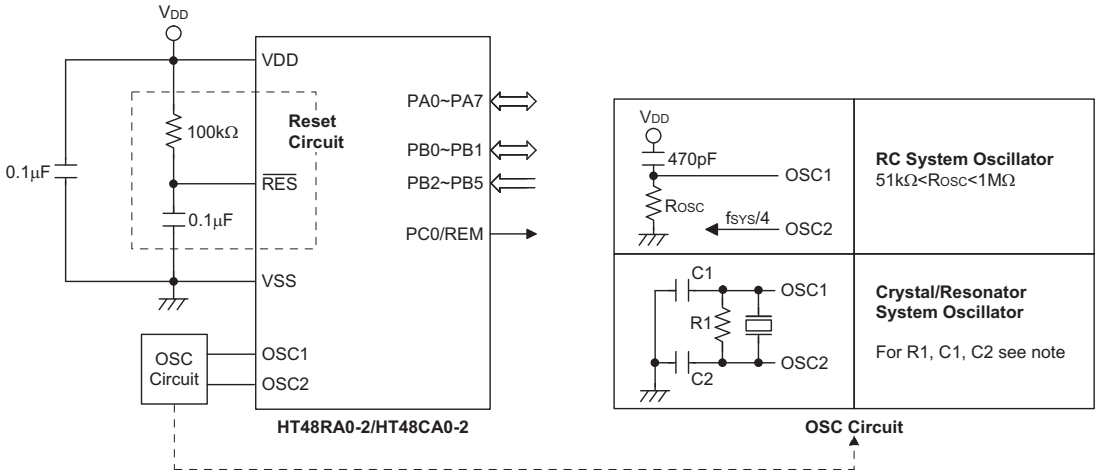
*2：因为低电压状态必须保持 1ms 以上，因此进入复位模式就要有 1ms 的延迟。

掩膜选项

下面的表格列出了 HT48RA0-2/HT48CA0-2 掩膜选项，所有这些选项必须被定义来确保系统正确的功能。

No.	掩膜选项
1	看门狗定时器（WDT）溢出时间的选择： WDT 的溢出周期= $\frac{2^n}{\text{时钟频率源}}$ ，这里 n = 8 ~ 11
2	看门狗定时器（WDT）：打开/关闭
3	清除看门狗定时器（CLR WDT）指令条数， 这个选项定义用指令清除 WDT 的方法。 1 条指令，即用 CLR WDT 指令清除 WDT。2 条指令，即必须要用 CLR WDT1 和 CLR WDT2 两条指令交替使用来清除 WDT。
4	唤醒（Wake-up）选择： 这个选项来定义激活唤醒功能。外部的输入/输出引脚（仅 PB 具有）使芯片从 HALT 状态中唤醒的能力。
5	载波/电平（Carry/Level）输出选择： 这个选项定义 PC0 是作为载波（Carry）输出还是作为电平（Level）输出。
6	载波（Carry）频率选择： 载波频率= $\frac{\text{时钟频率源}}{(2\text{或者}3)\times 2^n}$ ，这里 n =0 ~ 3
7	载波占空比（Carry duty）选择有二种选择：1/2 或 1/3。 如果频率=时钟源频率/（2，4，8 或 16），那么占空比为 1/2。 如果频率=时钟源频率/3，那么占空比为 1/3。 如果频率=时钟源频率/（6，12，或 24），那么占空比为 1/2 或 1/3。
8	振荡器（OSC）类型的选择： 这个选项确定 RC 或晶体振荡器被选择来作为系统时钟。
9	LVR 功能：打开/关闭

应用电路



注：1. 晶体/陶瓷谐振器为系统振荡器

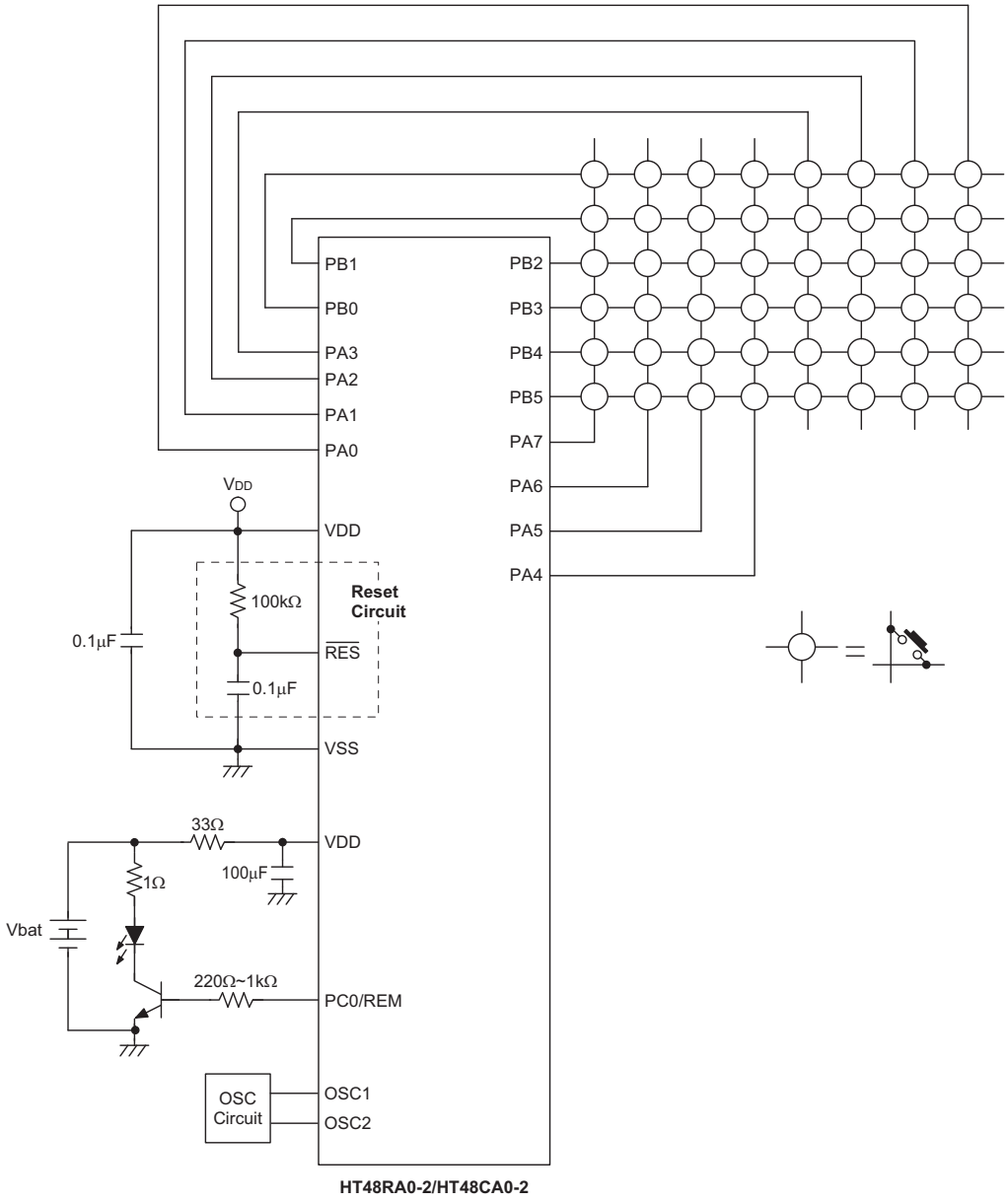
以晶体振荡器而言，仅有部分晶体是需要 C1 与 C2 来校准其振荡精度。而对于陶瓷谐振器来说，基本上都需要 C1 与 C2 来确保可以正常起振。在多数方案中，R1 没有必要使用。当 LVR 功能没有启用，如果要求当电压低于工作电压时晶体必须要停振，就有必要加上电阻 R1。C1 与 C2 的确切数据可以根据晶体/陶瓷谐振器的规格说明来选定。

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 $\overline{\text{RES}}$ 引脚上升到高之前能稳定，并将数值维持在正常允许的数据之内。为了防止噪声干扰，$\overline{\text{RES}}$ 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


PB1
PB0
PA3
PA2
PA1
PA0
PB2
PB3
PB4
PB5
PA7
PA6
PA5
PA4
VDD
VDD
100kΩ
Reset
Circuit
0.1μF
RES
0.1μF
VSS
33Ω
VDD
1Ω
100μF
Vbat
220Ω~1kΩ
PC0/REM
OSC
Circuit
OSC1
OSC2
HT48RA0-2/HT48CA0-2

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 0.5μs 中执行完成，而分支或调用操作则将在 1μs 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
算术运算				
ADD	A,[m]	ACC 与数据存储器相加，结果放入 ACC	1	Z,C,AC,OV
ADDM	A,[m]	ACC 与数据存储器相加，结果放入数据存储器	1(1)	Z,C,AC,OV
ADD	A,x	ACC 与立即数相加，结果放入 ACC	1	Z,C,AC,OV
ADC	A,[m]	ACC 与数据存储器、进位标志相加，结果放入 ACC	1	Z,C,AC,OV
ADCM	A,[m]	ACC 与数据存储器、进位标志相加，结果放入数据存储器	1(1)	Z,C,AC,OV
SUB	A,x	ACC 与立即数相减，结果放入 ACC	1	Z,C,AC,OV
SUB	A,[m]	ACC 与数据存储器相减，结果放入 ACC	1	Z,C,AC,OV
SUBM	A,[m]	ACC 与数据存储器相减，结果放入数据存储器	1(1)	Z,C,AC,OV
SBC	A,[m]	ACC 与数据存储器、进位标志相减，结果放入 ACC	1	Z,C,AC,OV
SBCM	A,[m]	ACC 与数据存储器、进位标志相减，结果放入数据存储器	1(1)	Z,C,AC,OV
DAA	[m]	将加法运算中放入 ACC 的值调整为十进制数，并将结果放入数据存储器	1(1)	C
逻辑运算				
AND	A,[m]	ACC 与数据存储器做“与”运算，结果放入 ACC	1	Z
OR	A,[m]	ACC 与数据存储器做“或”运算，结果放入 ACC	1	Z
XOR	A,[m]	ACC 与数据存储器做“异或”运算，结果放入 ACC	1	Z
ANDM	A,[m]	ACC 与数据存储器做“与”运算，结果放入数据存储器	1(1)	Z
ORM	A,[m]	ACC 与数据存储器做“或”运算，结果放入数据存储器	1(1)	Z
XORM	A,[m]	ACC 与数据存储器做“异或”运算，结果放入数据存储器	1(1)	Z
AND	A,x	ACC 与立即数做“与”运算，结果放入 ACC	1	Z
OR	A,x	ACC 与立即数做“或”运算，结果放入 ACC	1	Z
XOR	A,x	ACC 与立即数做“异或”运算，结果放入 ACC	1	Z
CPL	[m]	对数据存储器取反，结果放入数据存储器	1(1)	Z
CPLA	[m]	对数据存储器取反，结果放入 ACC	1	Z
递增和递减				
INCA	[m]	递增数据存储器，结果放入 ACC	1	Z
INC	[m]	递增数据存储器，结果放入数据存储器	1(1)	Z
DECA	[m]	递减数据存储器，结果放入 ACC	1	Z
DEC	[m]	递减数据存储器，结果放入数据存储器	1(1)	Z
移位				
RRA	[m]	数据存储器右移一位，结果放入 ACC	1	无
RR	[m]	数据存储器右移一位，结果放入数据存储器	1(1)	无
RRCA	[m]	带进位将数据存储器右移一位，结果放入 ACC	1	C
RRC	[m]	带进位将数据存储器右移一位，结果放入数据存储器	1(1)	C
RLA	[m]	数据存储器左移一位，结果放入 ACC	1	无
RL	[m]	数据存储器左移一位，结果放入数据存储器	1(1)	无
RLCA	[m]	带进位将数据存储器左移一位，结果放入 ACC	1	C
RLC	[m]	带进位将数据存储器左移一位，结果放入数据存储器	1(1)	C
数据传送				
MOV	A,[m]	将数据存储器送至 ACC	1	无
MOV	[m],A	将 ACC 送至数据存储器	1(1)	无
MOV	A,x	将立即数送至 ACC	1	无
位运算				
CLR	[m].i	清除数据存储器的位	1(1)	无
SET	[m].i	置位数据存储器的位	1(1)	无

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

影响标志位 Z

CALL addr 子程序调用

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

运算过程： Stack←PC+1

PC ← addr

影响标志位 没有

CLR [m] 清除数据存储器

说明： 本指令将数据存储器内的数值清零。

运算过程： [m] ←00H

影响标志位 没有

CLR [m] . i 将数据存储器的第 i 位清“0”

说明： 本指令将数据存储器内第 i 位值清零。

运算过程： [m].i ←0

影响标志位 没有

CLR WDT 清除看门狗定时器

说明： 本指令清除 WDT 计数器(从 0 开始重新计数)，暂停标志位(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

CLR WDT1 预清除看门狗定时器

说明： 必须搭配 CLR WDT2 一起使用，才可清除 WDT 计时器(从 0 开始重新计数)。当程序只执行过该指令，没有执行 CLR WDT2 时，系统只会不会将暂停标志位(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

CLR WDT2 预清除看门狗定时器

说明： 必须搭配 CLR WDT1 一起使用，才可清除 WDT 计时器(从 0 开始重新计数)。当程序只执行过该指令，没有执行 CLR WDT1 时，系统只会不会将暂停标志位(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

CPL [m] 对数据存储器取反，结果放入数据存储器

说明： 本指令是将数据存储器内保存的数值取反。

运算过程： [m] ← $[\overline{m}]$

影响标志位 Z

CPLA [m] 对数据存储器取反，结果放入累加器

说明： 本指令是将数据存储器内保存的值取反后，结果存放在累加器中。

运算过程： ACC ← $[\overline{m}]$

影响标志位 Z

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”；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] ←(ACC+00H)或

[m] ←(ACC+06H)或

[m] ←(ACC+60H)或

[m] ←(ACC+66H)或

影响标志位 C

DEC [m] 数据存储器的内容减 1，结果放入数据存储器

说明： 本指令将数据存储器内的数值减一再放回数据存储器。

运算过程： [m] ← [m]-1

影响标志位 Z

DECA [m] 数据存储器的内容减 1，结果放入累加器

说明： 本指令将存储器内的数值减一,再放到累加器。

运算过程： ACC ← [m]-1

影响标志位 Z

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： TO ← 0

PDF ← 1

影响标志位 TO，PDF

INC [m] 数据存储器的内容加 1，结果放入数据存储器

说明： 本指令将数据存储器内的数值加一,结果放回数据存储器。

运算过程： [m] ← [m]+1

影响标志位 Z

INCA [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 ← [m]+1

影响标志位 Z

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 没有

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 没有

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 没有

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 没有

NOP 空指令

说明： 本指令不作任何运算，而只将程序计数器加一。

运算过程： PC ← PC+1

影响标志位 没有

OR A, [m] 累加器与数据存储器做“或”运算，结果放入累加器

说明： 本指令是把累加器、数据存储器值做逻辑或，结果放到累加器。

运算过程： ACC←ACC “OR” [m]

影响标志位 Z

OR A, x 累加器与立即数做“或”运算，结果放入累加器

说明： 本指令是把累加器值、立即数做逻辑或，结果放到累加器。

运算过程： ACC←ACC “OR” x

影响标志位 Z

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]←ACC “OR” [m]

影响标志位 Z

RET 从子程序返回

说明： 本指令是将堆栈寄存器中的程序计数器值送回程序计数器。

运算过程： PC ← Stack

影响标志位 没有

RET A, x 从子程序返回，并将立即数放入累加器

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

运算过程： PC ← Stack

ACC ← x

影响标志位 没有

RETI 从中断返回

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

运算过程： PC ← Stack

EMI ← 1

影响标志位 没有

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].(i+1) ←[m].i;(i=0~6)

[m].0 ←[m].7

影响标志位 没有

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.(i+1) ←[m].i; (i=0~6)

ACC.0 ←[m].7

影响标志位 没有

RLC [m] 带进位将数据存储器左移一位，结果放入数据存储器

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

运算过程： [m].(i+1) ←[m].i; (i=0~6)

[m].0 ←C

C ← [m].7

影响标志位 C

RLCA [m] 带进位将数据存储器左移一位，结果放入累加器

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

运算过程： ACC.(i+1) ←[m].i; (i=0~6)

ACC.0 ←C

C ← [m].7

影响标志位 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].i ←[m].(i+1); (i=0~6)

[m].7 ←[m].0

影响标志位 没有

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.i ←[m].(i+1); (i=0~6)

ACC.7 ←[m].0

影响标志位 没有

RRC [m] 带进位将数据存储器右移一位，结果放入数据存储器

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

运算过程： [m].i ←[m].(i+1); (i=0~6)

[m]. 7 ←C

C ← [m].0

影响标志位 C

RRCA [m] 带进位将数据存储器右移一位，结果放入累加器

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

运算过程： ACC.i ←[m].(i+1) ; (i=0~6)

ACC.7 ←C

C ← [m].0

影响标志位 C

SBC A,[m] 累加器与数据存储器、进位标志相减，结果放入累加器

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

运算过程： ACC← ACC - [m] - $\overline{C}$

影响标志位 OV，Z，AC，C

SBCM A,[m] 累加器与数据存储器、进位标志相减，结果放入数据存储器

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

运算过程： [m]←ACC - [m] - $\overline{C}$

影响标志位 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： [m]← [m]-1

如果[m]=0，跳过下一条指令执行再下一条。

影响标志位 没有

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： ACC ←[m]-1

如果 ACC =0，跳过下一条指令执行再下一条。

影响标志位 没有

SET [m] 置位数据存储器

说明： 本指令是把存储器内的数值每个位置为 1。

运算过程： [m] ← FFH

影响标志位 没有

SET [m]. i 将数据存储器的第 i 位置“1”

说明： 本指令是把存储器内的数值的第 i 位置为 1。

运算过程： [m].i ←1

影响标志位 没有

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： [m] ←[m]+1

如果 [m]=0，跳过下一行指令

影响标志位 没有

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： ACC ←[m]+1

` 如果 ACC =0，跳过下一行指令

影响标志位 没有

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 没有

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]

影响标志位 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

影响标志位 OV，Z，AC，C

SUBM A, [m] 累加器与数据存储器相减，结果放入数据存储器

说明： 本指令是把累加器值、存储器值相减，结果放到存储器。

运算过程： [m]←ACC - [m]

影响标志位 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].3~[m].0↔[m].7~[m].4

影响标志位 没有

SWAPA [m] 交换数据存储器的高低字节，结果放入累加器

说明： 本指令是将数据存储器的低四位和高四位互换，再将结果送回累加器。

运算过程： ACC.3~ACC.0← [m].7~[m].4

ACC.7~ACC.4← [m].3~[m].0

影响标志位 没有

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 没有

SZA [m] 数据存储器送至累加器，如果内容为“0”，则跳过下一条指令

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

运算过程： 如果[m] = 0，跳过下一行指令，并 ACC← [m]。

影响标志位 没有

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 没有

TABRDC [m] 读取 ROM 当前页的内容，并送至数据存储器和 TBLH
说明： 本指令是将表格指针指向程序寄存器当前页，将低位送到存储器，高位直接送到 TBLH 寄存器内。
运算过程： [m] ←程序存储器低字节
TBLH←程序存储器高字节
影响标志位 没有

TABRDL [m] 读取 ROM 最后一页的内容，并送至数据存储器和 TBLH
说明： 本指令是将 TABLE 指针指向程序寄存器最后页，将低位送到存储器，高位直接送到 TBLH 寄存器内。
运算过程： [m] ←程序存储器低字节
TBLH←程序存储器高字节
影响标志位 没有

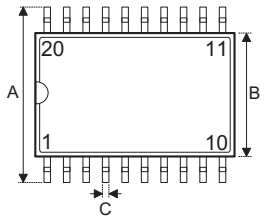
XOR A, [m] 累加器与立即数做“异或”运算，结果放入累加器
说明： 本指令是把累加器值、数据存储器值做逻辑异或，结果放到累加器。
运算过程： ACC←ACC “XOR” [m]
影响标志位 Z

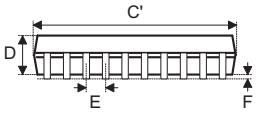
XORM A, [m] 累加器与数据存储器做“异或”运算，结果放入数据存储器
说明： 本指令是把累加器值、数据存储器值做逻辑异或，结果放到数据存储器。
运算过程： [m]←ACC “XOR” [m]
影响标志位 Z

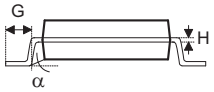
XOR A, x 累加器与数据存储器做“异或”运算，结果放入累加器
说明： 本指令是把累加器值与立即数做逻辑异或，结果放到累加器。
运算过程： ACC←ACC “XOR” x
影响标志位 Z

封装信息

20-pin SSOP (150mil) 外形尺寸



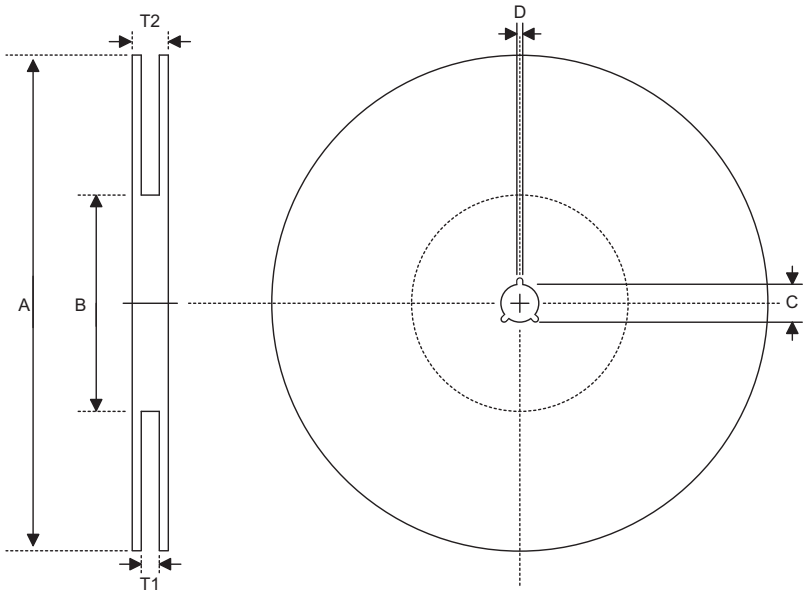




标号	尺寸（mil）		
	Min	**Nom**	**Max**
A	228	--	244
B	150	--	158
C	8	--	12
C’	335	--	347
D	49	--	65
E	--	25	--
F	4	--	10
G	15	--	50
H	7	--	10
α	0°	--	8°

包装带和卷轴规格：

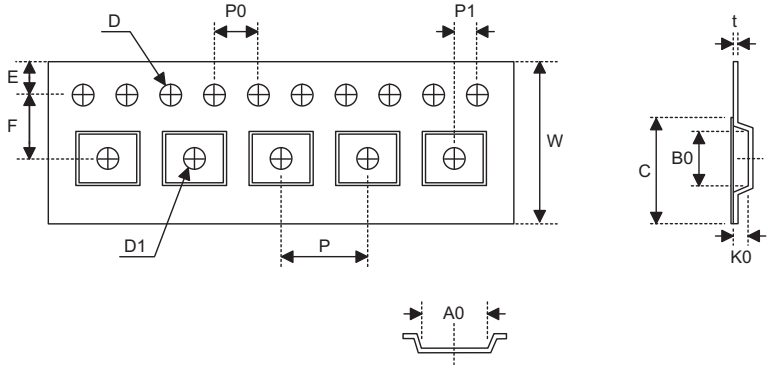
卷轴尺寸：



SSOP 20S(150mil)

标号	描述	尺寸(mm)
A	卷轴外圈直径	330±1.0
B	卷轴内圈直径	62±1.5
C	轴心直径	13.0+0.5 -0.2
D	缝宽	2.0±0.5
T1	轮缘宽	16.8+0.3 -0.2
T2	卷轴宽	22.2±0.2

运输带尺寸：



SSOP 20S(150mil)

标号	描述	尺寸(mm)
W	运输带宽	16.0+0.3 -0.1
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	1.5+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	6.5±0.1
B0	空穴宽	9.0±0.1
K0	空穴深	2.3±0.1
t	传输带厚度	0.30±0.05
C	覆盖带宽度	13.3

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

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

盛扬半导体有限公司（上海业务处）
上海市宜山路 2016 号合川大厦 1 号楼 3 楼 G 室 200103
电话: 86-21-5422-4590
传真: 86-21-5422-4596
网站: www.holtek.com.cn

盛扬半导体有限公司（深圳业务处）
深圳市南山区科技园科技中三路与高新中二道交汇处生产力大楼 A 单元五楼 518057
电话: 0755-8616-9908,8616-9308
传真: 0755-8616-9722

盛扬半导体有限公司（北京业务处）
北京市西城区宣武门西大街甲 129 号金隅大厦 1721 室 100031
电话: 010-6641-0030, 6641-7751, 6641-7752
传真: 010-6641-0125

盛扬半导体有限公司（成都业务处）
成都市东大街 97 号香槟广场 C 座 709 室 610016
电话: 028-6653-6590
传真: 028-6653-6591

Holtek Semiconductor(USA), Inc.（北美业务处）
46712 Fremont Blvd., Fremont, CA 94538
电话: 510-252-9880
传真: 510-252-9885
网站: www.holtek.com