

## 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 © 2007 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](#)
  - [HA0005S HT48 & HT46 MCU 用软件执行 I2C 总线的控制功能的方法](#)
  - [HA0047S HT46 MCU 的 PWM 的应用范例](#)

## 特性

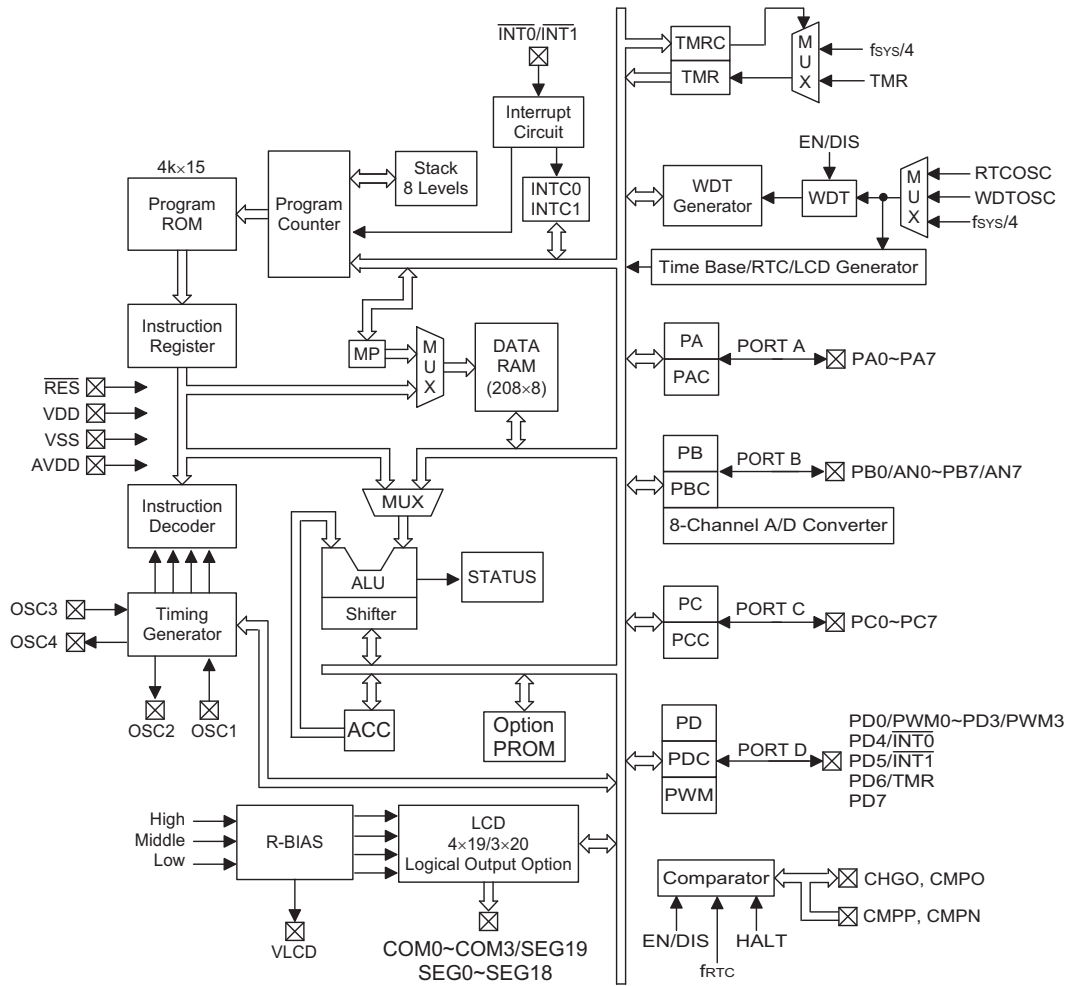
- 工作电压：
  - $f_{SYS}=4\text{MHz}$ : 2.2V~5.5V
  - $f_{SYS}=8\text{MHz}$ : 3.3V~5.5V
- 工作频率：外部 RC 振荡或晶体振荡
- 32.768kHz 晶体振荡器可用作计时
- 看门狗定时器
- 1 个 16 位定时/计数器，具有溢出中断 (TMR)
- 时基发生器 (时钟来源：32.768kHz) 和 RTC 中断
- 4K×15 程序存储器
- 208×8 数据存储器
- 最多可有 32 个双向输入/输出口 (与  $\overline{\text{INT0}}$ 、 $\overline{\text{INT1}}$ 、TMR、AN0~AN7、PWM0~PWM3 共用引脚)
- 8 层硬件堆栈
- 当 VDD=5V，系统频率为 8MHz 时，指令周期为 0.5 $\mu\text{s}$
- 2 个外部中断 (上升/下降沿触发)
- 1 个比较器
- LCD: 20×3 或 19×4, 1/3bias 时有 12 个口可由掩膜选项设置为逻辑输出 (以 4 个为一组进行设置，其中有 8 个口可大电流灌入)
- 内置 R 型偏压发生器
- 8 通道 8 位解析度的 A/D 转换器
- 4 通道 PWM 输出
- 56-pin SSOP, 100-pin QFP 封装

## 概述

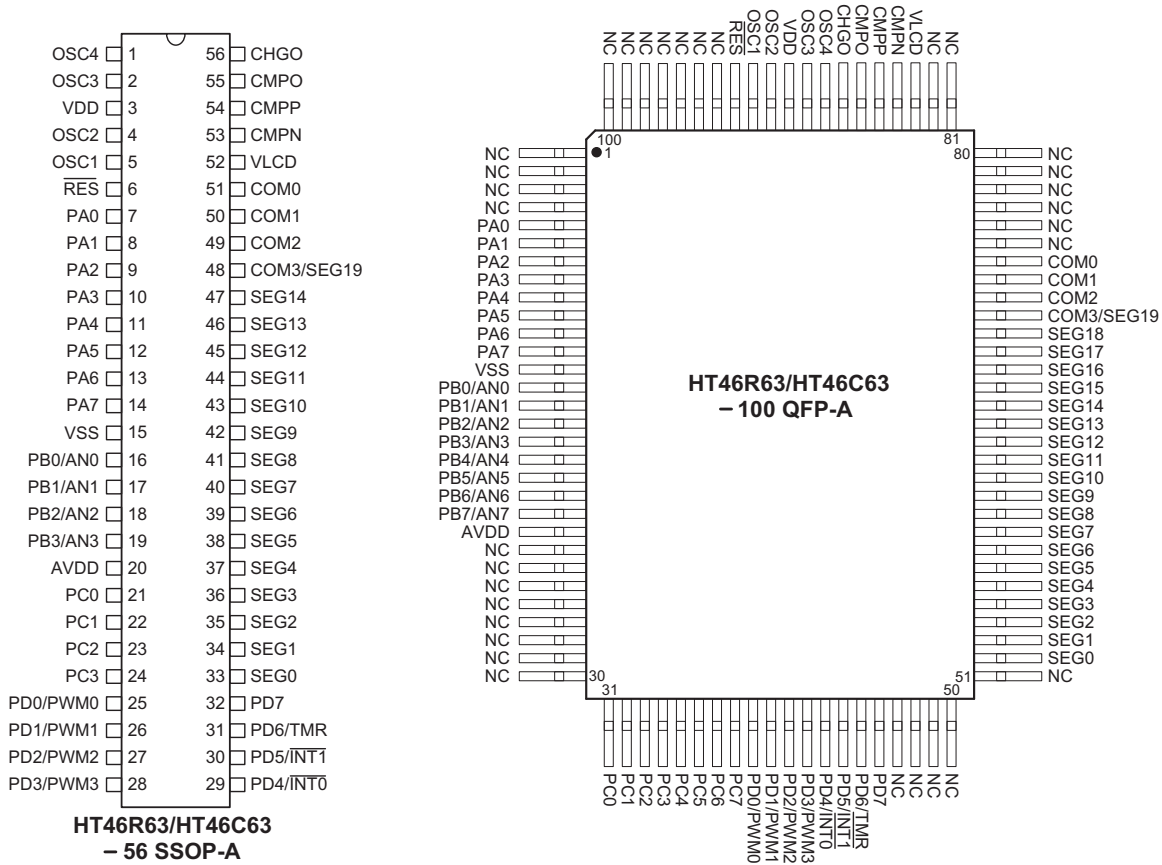
HT46R63/HT46C63 是 8 位高性能精简指令集单片机，专门为需要 A/D 转换和 LCD 显示的产品而设计。掩膜版本 HT46C63 与 OTP 版本 HT46R63 引脚和功能完全相同。

低功耗、I/O 使用灵活、计数器、振荡类型选择、多通道 A/D 转换、脉冲测量功能、暂停和唤醒功能，以及 LCD 显示功能，使这款单片机可以广泛应用于需要 A/D 转换和 LCD 显示的产品中，例如电子测量仪器、环境监控、手持式测量工具、马达控制等工业和家庭系统中。

方框图



引脚图



## 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	每一位可由掩膜选项设置是否带上拉电阻，每一位输入/输出口都有相关的控制寄存器（PAC）。PA 的每一位可由掩膜选项设置为唤醒输入。可由软件设置为斯密特触发输入或 CMOS 输出。
PB0/AN0~ PB7/AN7	输入/输出	上拉电阻	每一位可由掩膜选项设置是否带上拉电阻，每一位输入/输出口都有相关的控制寄存器（PBC）。可由软件设置为斯密特触发输入或 CMOS 输出。PB 的每一位与 A/D 输入共用引脚。
PC0~PC7	输入/输出	上拉电阻	每一位可由掩膜选项设置是否带上拉电阻，每一位输入/输出口都有相关的控制寄存器（PCC）。可由软件设置为斯密特触发输入或 CMOS 输出。
PD0/PWM0~ PD3/PWM3, PD4/ $\overline{\text{INT0}}$ , PD5/ $\overline{\text{INT1}}$ , PD6/TMR, PD7	输入/输出	上拉电阻 PWM 中断 上升与/或下降沿	每一位可由掩膜选项设置是否带上拉电阻，每一位输入/输出口都有相关的控制寄存器（PDC）。可由软件设置为斯密特触发输入或 CMOS 输出。PD0~PD3 可设置为 PWM 输出。 $\overline{\text{INT0}}/\overline{\text{INT1}}$ 可选择为上升/下降沿触发。
OSC1 OSC2	输入 输出	RC 或晶体	在 OSC1 和 VDD 之间接电阻或 OSC1 和 OSC2 之间接晶体可产生系统时钟。
OSC3 OSC4	输入 输出	—	在引脚 OSC3 和 OSC4 之间接 32768Hz 晶体，可产生提供系统计时的 RTC 时钟信号。
CMPN	输入	—	比较器的负极输入。
CMPP	输入	—	比较器的正极输入。
CMPO	输出	—	比较器输出。
CHGO	输出	—	带 32768Hz 载波的比较器输出。
VDD	—	—	正电源。
AVDD	—	—	A/D 转换器的正电源，AVDD 需要外接至 VDD。
VSS	—	—	负电源，接地。
$\overline{\text{RES}}$	输入	—	斯密特触发复位输入。
VLCD	输入/输出	—	LCD 驱动的最高电压，需要通过电阻接至 VDD。
SEG0~SEG18	输出	SEG0~SEG18 逻辑 CMOS 输出	LCD segment 驱动信号输出；SEG7~SEG10 可掩膜选择为逻辑输出口；SEG11~SEG14，SEG15~SEG18 可掩膜选择为大电流灌入的逻辑输出口。
COM0~COM3 /SEG19	输出	COM3 或 SEG19	LCD common 驱动信号输出。

## 极限参数

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

储存温度..... $-50^{\circ}\text{C} \sim 125^{\circ}\text{C}$   
 工作温度..... $-40^{\circ}\text{C} \sim 85^{\circ}\text{C}$

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

## 直流电气特性

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				
V <sub>DD</sub>	工作电压	—	f <sub>sys</sub> =4MHz	2.2	—	5.5	V
			f <sub>sys</sub> =8MHz	3.3	—	5.5	V
I <sub>DD1</sub>	工作电流 (晶体振荡、RC 振荡)	3V	无负载, f <sub>sys</sub> =4MHz	—	1	2	mA
		5V	ADC 关闭	—	3	5	
I <sub>DD2</sub>	工作电流 (晶体振荡、RC 振荡)	3V	无负载, f <sub>sys</sub> =8MHz	—	1	2	mA
		5V	ADC 关闭	—	3	5	
I <sub>STB1</sub>	静态电流 (*f <sub>s</sub> =WDT 振荡)	3V	无负载, 系统 HALT	—	—	5	μA
		5V	LCD 关闭	—	—	20	
I <sub>STB2</sub>	静态电流 (*f <sub>s</sub> =f <sub>sys</sub> )	3V	无负载, 系统 HALT	—	—	1	μA
		5V	LCD 关闭	—	—	2	
I <sub>STB3</sub>	静态电流 (*f <sub>s</sub> =RTC 振荡)	3V	无负载, 系统 HALT	—	—	5	μA
		5V	LCD 关闭	—	—	15	
I <sub>STB4</sub>	静态电流 (*f <sub>s</sub> =RTC 振荡)	3V	无负载, 系统 HALT, HALT 时 LCD 打开, 电阻型偏压, VLCD=VDD (低偏压电流)	10	12	16	μA
		5V		20	24	32	
I <sub>STB5</sub>	静态电流 (*f <sub>s</sub> =RTC 振荡)	3V	无负载, 系统 HALT, HALT 时 LCD 打开, 电阻型偏压, VLCD=VDD (中偏压电流)	16	20	26	μA
		5V		32	40	52	
I <sub>STB6</sub>	静态电流 (*f <sub>s</sub> =RTC 振荡)	3V	无负载, 系统 HALT, HALT 时 LCD 打开, 电阻型偏压, VLCD=VDD (高偏压电流)	38	52	68	μA
		5V		76	104	136	
V <sub>IL1</sub>	输入/输出口的低电平 输入电压	—	—	0	—	0.3V <sub>DD</sub>	V
V <sub>IH1</sub>	输入/输出口的高电平 输入电压	—	—	0.7V <sub>DD</sub>	—	V <sub>DD</sub>	V
V <sub>IL2</sub>	低电平输入电压( $\overline{RES}$ )	—	—	0	—	0.4 V <sub>DD</sub>	V
V <sub>IH2</sub>	高电平输入电压( $\overline{RES}$ )	—	—	0.9 V <sub>DD</sub>	—	V <sub>DD</sub>	V
V <sub>LCD</sub>	LCD 最高电压	—	—	0	—	V <sub>DD</sub>	V
I <sub>OH1</sub>	输入/输出源电流	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-2	-4	—	mA
		5V		-5	-8	—	
I <sub>OL1</sub>	输入/输出灌电流	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	6	12	—	mA
		5V		10	25	—	

符号	参数	测试条件		最小	典型	最大	单位
		V <sub>DD</sub>	条件				
I <sub>OH2</sub>	SEG7~18 源电流	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-2	-4	—	mA
		5V		-4	-8	—	
I <sub>OL2</sub>	SEG7~10 灌电流	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	8	—	—	mA
		5V		16	—	—	
I <sub>OL3</sub>	SEG11~18 灌电流	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	16	—	—	mA
		5V		32	—	—	
I <sub>OHTOTAL</sub>	输入/输出口总源电流	—	—	—	—	-100	mA
I <sub>OLTOTAL</sub>	输入/输出口总灌电流	—	—	—	—	100	mA
R <sub>PH</sub>	上拉电阻(输入/输出)	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
V <sub>OS</sub>	比较器输入偏置电压	—	—	-10	—	10	mV
V <sub>I</sub>	比较器输入电压范围	—	—	0.2	—	V <sub>DD</sub> -0.8	V
V <sub>AD</sub>	A/D 输入电压	—	—	0	—	V <sub>DD</sub>	V
E <sub>AD</sub>	A/D 转换误差	—	—	—	±0.5	±1	LSB
I <sub>ADC</sub>	A/D 转换电路打开时增加的系统功耗	3V	—	—	0.5	1	mA
		5V	—	—	1.5	3	

注：有关“f<sub>s</sub>”的具体说明请参阅 WDT 的时钟选择。

## 交流电气特性

T<sub>a</sub>=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V <sub>DD</sub>	条件				
f <sub>SYS1</sub>	系统时钟（晶体振荡）	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	
f <sub>SYS2</sub>	系统时钟 （32768Hz 晶体振荡）	—	2.2V~5.5V	—	32768	—	Hz
f <sub>TIMER</sub>	定时器输入频率	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	
t <sub>WDTOSC</sub>	看门狗振荡器	3V	—	45	90	180	μs
		5V		32	65	130	
t <sub>WDT</sub>	看门狗溢出周期 （WDT 振荡器）	—	—	—	2 <sup>16</sup>	—	t <sub>WDTOSC</sub>
	看门狗溢出周期 （f <sub>SYS</sub> /4）	—	—	—	2 <sup>18</sup>	—	t <sub>SYS</sub>
	看门狗溢出周期 （32768Hz）	—	—	—	2 <sup>16</sup>	—	t <sub>RTCOSC</sub>
t <sub>RES</sub>	外部复位低电平脉宽	—	—	1	—	—	μs
t <sub>SST</sub>	系统启动延迟时间	—	上电或从 HALT 状态唤醒	—	1024	—	*t <sub>SYS</sub>
t <sub>INT</sub>	中断脉冲宽度	—	—	1	—	—	μs
t <sub>AD</sub>	A/D 时钟周期	—	—	1	—	—	μs
t <sub>ADC</sub>	A/D 转换时间	—	—	64	—	—	t <sub>AD</sub>
t <sub>ADCS</sub>	A/D 采样时间	—	—	—	32	—	t <sub>AD</sub>
t <sub>COMP</sub>	比较器响应时间	—	—	—	—	3	μs

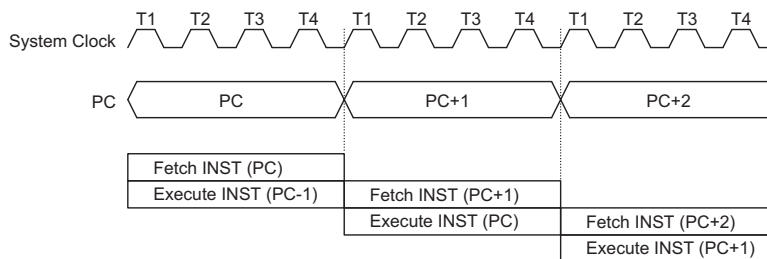
注：\*t<sub>SYS</sub> = 1/f<sub>SYS</sub>

## 系统功能说明

### 指令执行时序

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

指令的读取和执行是以流水线方式进行的，这种方式在一个指令周期进行读取指令操作，而在下一个指令周期进行解码与执行该指令。因此，流水线方式使多数指令能在一个周期内执行完成。但如果涉及到的指令要改变程序计数器的值，就需要花两个指令周期来完成这一条指令。



指令执行时序

### 程序计数器 — PC

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

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

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

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

当遇到控制转移指令时，系统也会插入一个空指令周期。

模式	程序计数器								
	*11~*8	*7	*6	*5	*4	*3	*2	*1	*0
初始化复位	0000	0	0	0	0	0	0	0	0
外部中断 0	0000	0	0	0	0	0	1	0	0
外部中断 1	0000	0	0	0	0	1	0	0	0
定时/计数器溢出	0000	0	0	0	0	1	1	0	0
时基溢出	0000	0	0	0	1	0	0	0	0
A/D 转换中断	0000	0	0	0	1	0	1	0	0
RTC 中断	0000	0	0	0	1	1	0	0	0
条件跳跃	Program Counter+2								
装载 PCL	@11~@8	@7	@6	@5	@4	@3	@2	@1	@0
跳转, 子程序调用	#11~#8	#7	#6	#5	#4	#3	#2	#1	#0
子程序返回(RET、RETI)	S11~S8	S7	S6	S5	S4	S3	S2	S1	S0

程序计数器

注：\*11~\*0：程序计数器位  
#11~#0：指令代码位

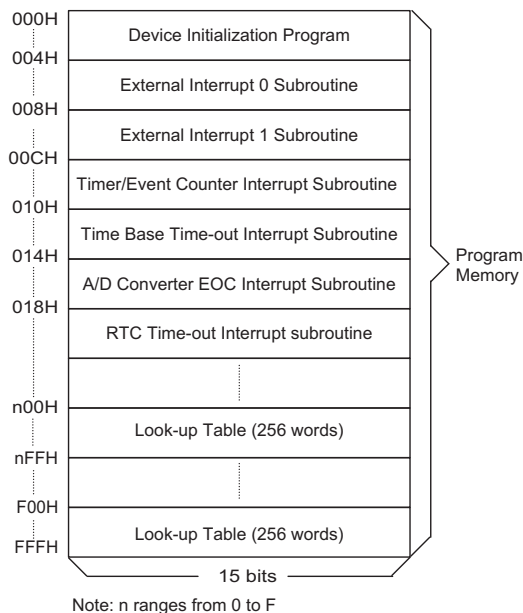
S11~S0：堆栈寄存器位  
@7~@0：PCL 位

### 程序存储器 — PROM

程序存储器用来存放要执行的指令代码，以及一些数据、表格和中断入口。程序存储器有 4096×15 位，程序存储器空间可以用程序计数器或表格指针进行寻址。

以下列出的程序存储器地址是系统专为特殊用途而保留的：

- 地址 000H  
该地址为程序初始化保留。系统复位后，程序总是从 000H 开始执行。
- 地址 004H  
该地址为外部中断 0 服务程序保留。当  $\overline{INT0}$  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  
该地址为外部中断 1 服务程序保留。当  $\overline{INT1}$  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 008H 地址开始执行。
- 地址 00CH  
该地址为定时/计数器中断服务程序保留。当定时/计数器溢出，如果中断允许且堆栈未滿，则程序会跳转到 00CH 地址开始执行。
- 地址 010H  
该地址为时基中断服务程序保留。当时基发生溢出，如果中断允许且堆栈未滿，则程序会跳转到 010H 地址开始执行。
- 地址 014H  
该地址为 A/D 转换中断服务程序保留。当 A/D 转换完成，如果中断允许且堆栈未滿，则程序会跳转到 014H 地址开始执行。
- 地址 018H  
该地址为 RTC 中断服务程序保留。当 RTC 发生溢出，如果中断允许且堆栈未滿，则程序会跳转到 018H 地址开始执行。
- 表格区



程序存储器

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

指令	表格区											
	*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：\*11 ~ \*0 : 表格地址字节      P11 ~ P8 : 当前程序指针字节      @7 ~ @0 : 表格指针字节

## 堆栈寄存器 — STACK

堆栈寄存器是特殊的存储器空间，用来保存 Program Counter 的值。HT46x63 有 8 层堆栈，堆栈寄存器既不是数据存储器的一部分，也不是程序存储器的一部分，而且它既不能读出，也不能写入。堆栈的使用是通过堆栈指针（SP）来实现的，堆栈指针也不能读出或写入。当发生子程序调用或中断响应时，程序计数器（Program Counter）的值会被压入堆栈；在子程序调用结束或中断响应结束时（执行指令 RET 或 RETI），堆栈将原先压入堆栈的内容弹出，重新装入程序计数器中。在系统复位后，堆栈指针会指向堆栈顶部。

如果堆栈已满，并且发生了不可屏蔽的中断，那么只有中断请求标志会被记录下来，而中断响应会被抑制，直到堆栈指针（执行 RET 或 RETI 指令）发生递减，中断才会被响应。这个功能可以防止堆栈溢出，使得程序员易于使用这种结构。同样，如果堆栈已满，并且发生了子程序调用，那么堆栈会发生溢出，首先进入堆栈的内容将会丢失，只有最后的 8 个返回地址会被保留。

## 数据存储器 — RAM

数据存储器由 239×8 位组成，分为两个功能区间：特殊功能寄存器和通用数据存储器（208×8），数据存储器单元大多数是可读/写的，但有些只读的。

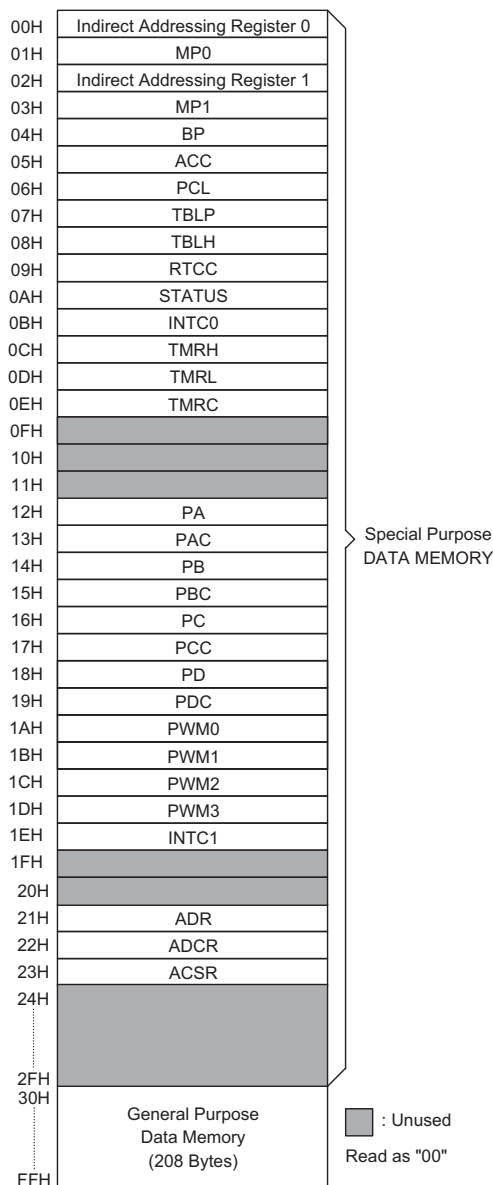
特殊功能寄存器包括间接寻址寄存器（R0: 00H, R1: 02H），间接寻址指针寄存器（MP0: 01H, MP1: 03H），存储器段指针（BP: 04H），累加器（ACC: 05H），程序计数器低字节寄存器（PCL: 06H），表格指针寄存器（TBLP: 07H），表格内容高字节寄存器（TBLH: 08H），RTC 控制寄存器（RTCC: 09H），状态寄存器（STATUS: 0AH），中断控制寄存器（INTC0: 0BH），定时/计数器高字节（TMRH; 0CH），定时/计数器低字节（TMRL; 0DH），定时/计数器控制寄存器（TMRC; 0EH），输入/输出寄存器（PA: 12H, PB: 14H, PC: 16H, PD: 18H），输入/输出控制寄存器（PAC: 13H, PBC: 15H, PCC: 17H, PDC: 19H），PWM0（1AH），PWM1（1BH），PWM2（1CH），PWM3（1DH），INTC1（1EH），A/D 转换结果寄存器（ADR: 21H），A/D 控制寄存器（ADCR: 22H），A/D 时钟设置寄存器（ACSR: 23H）。其余在 30H 之前的空间保留给系统以后扩展使用，读取这些地址的返回值为“00H”。通用数据寄存器地址从 30H 到 FFH，用来存储数据和控制信息。

所有的数据存储器单元都能直接执行算术、逻辑、递增、递减和循环操作。除了一些特殊位外，数据存储器的每一位都可通过“SET[m].i”置位或由“CLR[m].i”复位。而且都可以通过间接寻址指针（MP0; 01H/MP1; 03H）进行间接寻址。

## 间接寻址寄存器

地址 00H 和 02H 是间接寻址寄存器，并无实际的物理区存在。任何对[00H]或[02H]的读/写操作，都是访问由 MP0（01H）MP1（03H）或所指向的 RAM 单元。间接读取地址 00H 或 02H 得到的值为 00H，间接写入此地址，不会产生任何操作。

间接寻址指针 MP0(01H)和 MP1(03H)是 8 位寄存器。只有 MP1/R1 能寻址 LCD 显示存储器(BP=1)。



数据存储器

### 存储器段指针

存储器段指针 (Bank Pointer) 可以用来指向不同的 RAM 空间。当 BP=0 时, 表示对 RAM Bank 0 操作; 当 BP=1 时, 表示对 LCD RAM 操作 (只能用 MP1/R1 进行间接寻址)。除了 RAM Bank 0 之外, 其它所有 RAM Bank 中, 地址 40H 之前的空间都不存在。

### 累加器

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

### 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

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) 和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, 而且还控制操作顺序。

符号	位	功能
C	0	如果在加法运算中结果产生了进位或在减法运算中结果不产生借位, 则 C 被置位; 反之, C 被清除。它也可被循环移位指令影响。
AC	1	如果在加法运算中低 4 位产生了进位或减法运算中低 4 位不产生借位, 则 AC 被置位; 反之, AC 被清除。
Z	2	如果算术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, 则 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” 或 “HALT” 指令, TO 被清除; WDT 定时溢出, TO 被置位。
—	6	未用, 读数为 “0”
—	7	未用, 读数为 “0”

**STATUS(0AH) 寄存器**

除了 PDF 和 TO 标志外, 状态寄存器的其它位都可以用指令改变。任何对状态寄存器的写操作都不会改变 PDF 和 TO 的值。对状态寄存器的操作可能会导致与预期不一样的结果。TO 标志只受系统上电、看门狗溢出、“CLR WDT” 指令或 “HALT” 指令的影响。PDF 标志只受系统上电、“CLR WDT” 指令或 “HALT” 指令的影响。

标志位 Z、OV、AC 和 C 反映的是最近一次操作的状态。

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

## 中断

HT46x63 提供两个外部中断、一个内部定时/计数器中断、一个时基溢出中断、一个 A/D 转换结束中断和一个 RTC 中断。中断控制寄存器 0 (INTC0; 0BH) 和中断控制寄存器 1 (INTC1; 1EH) 包含了中断控制位和中断请求标志，中断控制位用来设置中断允许/禁止。

寄存器	位	符号	功 能
INTC0 (0BH)	0	EMI	总中断控制位 (1=允许; 0=禁止)
	1	EEI0	外部中断 0 控制位 (1=允许; 0=禁止)
	2	EEI1	外部中断 1 控制位 (1=允许; 0=禁止)
	3	ETI	定时/计数器中断控制位 (1=允许; 0=禁止)
	4	EIF0	外部中断 0 请求标志 (1=有; 0=无)
	5	EIF1	外部中断 1 请求标志 (1=有; 0=无)
	6	TF	定时/计数器中断请求标志 (1=有; 0=无)
	7	—	只作内部测试用 使用时必须写入 '0'; 否则会发生不可预知的错误

INTC0(0BH) 寄存器

寄存器	位	符号	功 能
INTC1 (1EH)	0	ETBI	时基中断控制位 (1=允许; 0=禁止)
	1	EADI	A/D 转换中断控制位 (1=允许; 0=禁止)
	2	ERTI	实时时钟中断控制位 (1=允许; 0=禁止)
	3	—	未用, 读出为 "0"
	4	TBF	时基中断请求标志 (1=有; 0=无)
	5	ADF	A/D 转换结束中断请求标志 (1=有; 0=无)
	6	RTF	实时时钟中断请求标志 (1=有; 0=无)
	7	—	未用, 读出为 "0"

INTC1(1EH) 寄存器

只要有中断子程序被服务，其余的中断全部都被自动禁止（通过清除 EMI 位），这种做法的目的在于防止中断嵌套。这时如果有其它中断发生，只有中断请求标志会被记录下来。如果在中断服务程序中有另一个中断需要响应，程序员可以置位 EMI、INTC0 和 INTC1 所对应的位，以便进行中断嵌套。如果堆栈已满，则中断并不会被响应，一直到堆栈指针 (SP) 发生递减后才会响应。如果需要中断立即得到响应，应避免堆栈饱和。

所有的中断都具有唤醒能力。当中断被服务，系统会将程序计数器的内容压入堆栈，然后再跳转至中断服务程序的入口。但这时只有程序计数器的内容被压入堆栈，如果其它寄存器和状态寄存器的内容会被中断程序改变，从而会破坏主程序的控制流程的话，程序员应该事先将这些数据保存起来。

外部中断是由  $\overline{INT0}/\overline{INT1}$  引脚上上升沿信号或者下降沿信号或者电平变化边沿信号（包括上升沿和下降沿）触发的，其中断请求标志位 (EIF0/EIF1; INTC0 的第 4、5 位) 会被置位。如果中断允许，且堆栈未满，当发生外部中断时，会产生地址 004H/008H 的子程序调用；而中断请求标志 EIF0/EIF1 和总中断控制位 EMI 会被清除，以禁止其它中断响应。

内部定时/计数器中断是由定时/计数器溢出触发的，其中断请求标志 (TF; INTC0 的第 6 位) 会被置位。如果中断允许，且堆栈未满，当发生定时/计数器中断时，会产生地址 00CH 的子程序调用；而中断请求标志 TF 和总中断控制位 EMI 会被清除，以禁止其它中断响应。

时基中断是由时基溢出触发的，其中断请求标志 (TBF; INTC1 的第 4 位) 会被置位。如果中断允许，且堆栈未满，当发生时基中断时，会产生地址 010H 的子程序调用；而中断请求标志 TBF 和总中断控制位 EMI 会被清除，以禁止其它中断响应。

A/D 转换中断是由 A/D 转换完成触发的，其中断请求标志 (ADF; INTC1 的第 5 位) 会被置位。如果中断允许，且堆栈未满，当发生 A/D 转换中断时，会产生地址 014H 的子程序调用；而中断请求标志位 ADF 和总中断控制位 EMI 会被清除，以禁止其它中断响应。

RTC 中断是由 RTC 溢出触发的，其中断请求标志（RTF；INTC1 的第 6 位）会被置位。如果中断允许，且堆栈未滿，当发生 RTC 中断时，会产生地址 018H 的子程序调用；而中断请求标志位 RTF 和总中断控制位 EMI 会被清除，以禁止其它中断响应。

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

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

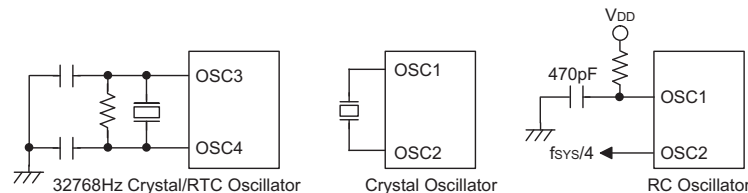
No.	中断源	优先级	中断向量
A	外部中断 0	1	004H
B	外部中断 1	2	008H
C	定时/计数器中断	3	00CH
D	时基中断	4	010H
E	A/D 转换结束中断	5	014H
F	RTC 中断	6	018H

中断控制寄存器（INTC0/INTC1），由外部中断 0/1 请求标志（EIF0/EIF1）、定时/计数器中断请求标志（TF）、时基中断请求标志（TBF）、A/D 转换中断请求标志（ADF）、RTC 中断请求标志（RTF）、外部中断允许（EEI0/EEI1）、定时/计数器中断允许（ETI）、时基中断允许（ERI）、A/D 转换中断允许（EADI）和总中断允许（EMI）组成，其对应于数据存储地址 0BH/1EH。EMI、EEI0、EEI1、ETI、ETBI、EADI 和 ERTI 用来控制中断的允许/禁止状态的。这些控制位可以用来屏蔽正在进行中断服务程序时发生的其它中断请求。一旦中断请求标志（EIF0、EIF1、TF、EIF、ADF、HIF）被置位，会一直保留在 INTC0 和 INTC1 寄存器中，直到中断被响应或用软件指令清除为止。

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

## 振荡电路

HT46x63 有四种振荡方式提供给系统。其中外部 RC 振荡和外部晶体振荡可提供系统时钟，可以通过掩膜选项设定。HALT 模式会停止系统振荡器，并忽视任何外部信号以降低功耗。另一种是 32768Hz 的晶体振荡，此频率提供给实时时钟；还有一种为内置 12kHz 的 RC 振荡器，做为 WDTOSC。



系统振荡器

注：当使用 32.768kHz 的晶振时，外部的电阻电容不是必须的。但在精确的 RTC 应用场合，电阻电容可以用来消除晶振制造带来的误差。

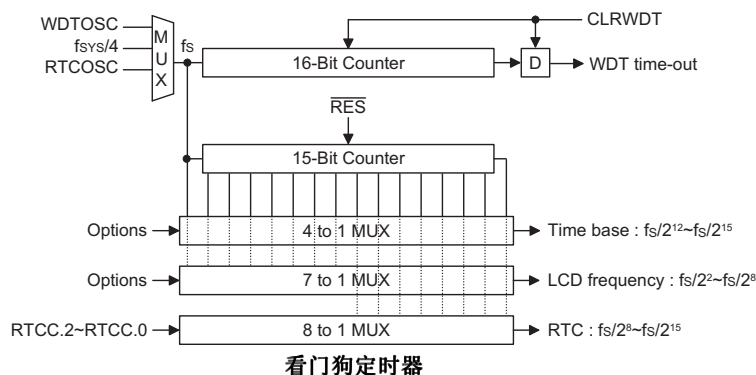
如果选用外部 RC 振荡方式，在 OSC1 与 VSS 之间需要接一个外部电阻，其阻值为 24kΩ 到 1MΩ；而 OSC2 上会输出系统频率的 4 分频信号，可用于同步外部逻辑。

如果选用晶体振荡方式，在 OSC1 和 OSC2 之间需要连接一个晶体，用来提供晶体振荡器所需的反馈和相移，除此之外，不再需要其它外部元件。另外，在 OSC1 和 OSC2 之间也可使用谐振器来取代晶体振荡器，但是在 OSC1 和 OSC2 需要多连接两个电容。

如果需要使用 RTCOSC，那么在 XT1 与 XT2 之间需要接一个晶体提供振荡器所需的反馈和相移，除此之外，不需要其它外部器件。

## 看门狗定时器 — WDT

WDT(LCD、RTC 和 Time Base)的时钟来源可由掩膜选项设置为晶体振荡 (32.768kHz: RTCOSC) 或指令时钟 (系统时钟 4 分频:  $f_{SYS}/4$ ) 或 RC 振荡器 (12kHz: WDTOSC)。看门狗定时器主要用来防止程序运行故障和程序跳入一死循环而导致不可预测的结果。看门狗定时器可由掩膜选项设置为打开或关闭, 如果在关闭状态, 所有与 WDT 有关的指令操作都是没有作用的。WDT 的溢出时间周期为  $2^{16}/f_s$ ,  $f_s$  指的是 WDT、Time Base、RTC 以及 LCD 的时钟频率。



如果 WDT 时钟源为内部 WDT 振荡, 溢出周期会因为温度、VDD 以及芯片参数的变化而变化。如果选择 WDTOSC 或 RTCOSC 做为 WDT 的时钟源, 那么它们在系统处于暂停模式时仍然可以振荡(由掩膜选择设定)。一旦使用 RTCOSC(正常情况下周期为 31.25 $\mu$ s)做为 WDT 的时钟来源, 则该频率直接  $2^{16}$  分频可获得一个大约 2s 的溢出周期。

如果 WDT 的时钟源为指令时钟, 则在 HALT 状态时, WDT 会停止计数而失去保护功能; 此时只能靠外部逻辑复位来重新启动系统。如果系统运用在强干扰的环境中, 建议选用内部 WDT 振荡器, 因为 HALT 模式会使系统时钟停止, 看门狗也就失去了保护的功能。

在正常运行时, WDT 溢出会使系统复位并置位 TO 标志; 但在 HALT 模式下, WDT 溢出只产生“热复位”, 只有程序计数器 Program Counter 和堆栈指针 SP 被复位。要清除 WDT 的值可以有三种方法: 外部复位(低电平输入到  $\overline{RES}$  端)、清除看门狗指令或 HALT 指令。清除看门狗指令有“CLR WDT”和“CLR WDT1”、“CLR WDT2”二组指令。这两组指令中, 只能选择其中一组, 由掩膜选项决定。如 果选择“CLR WDT”, 那么只要执行“CLR WDT”指令就会清除 WDT。如果选择“CLR WDT1”和“CLR WDT2”, 那么二条指令要交替使用才会清除 WDT, 否则, WDT 会由于溢出而使系统复位。RTC 振荡器需要设置为自动快速起振 (auto-speed-up) 振荡器。在 RTC 振荡器振荡稳定后, 需要关闭自动快速起振。

## 时基 — Time Base

时基指的是用一个周期性的溢出来产生一个有规律的内部中断。溢出周期的范围为  $f_s/2^{12} \sim f_s/2^{15}$ , 由掩膜选项确定。当时基发生溢出, 如果中断允许, 且堆栈未满, 那么就会产生一个地址 010H 的子程序调用。

## 实时时钟 — RTC

实时时钟(RTC)的工作情况和时基一样。它是用来提供一个有规律的内部中断。它的溢出周期范围为  $f_s/2^8 \sim f_s/2^{15}$ , 可通过软件编程实现。当 RTC 发生溢出, 如果中断允许, 且堆栈未满, 那么就会产生一个地址 018H 的子程序调用。寄存器 RTCC 用来确定 RTC 时钟来源的分频系数。RTCC.7~RTCC.3 不能使用。

RTCC.2	RTCC.1	RTCC.0	RTC 实时时钟分频级数
0	0	0	$2^8$
0	0	1	$2^9$
0	1	0	$2^{10}$
0	1	1	$2^{11}$
1	0	0	$2^{12}$
1	0	1	$2^{13}$
1	1	0	$2^{14}$
1	1	1	$2^{15}$

### 暂停模式 — HALT

暂停模式是由 HALT 指令来实现的，暂停模式时系统状态如下：

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

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

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

当发生唤醒，系统需要额外花费  $1024t_{\text{SYS}}$ （系统时钟周期）的时间，才能重新正常运行，也就是说，唤醒之后会插入一个等待周期。如果唤醒是由中断产生的话，则实际中断子程序的执行会延迟一个以上的周期。如果唤醒导致下一条指令执行，那么在等待周期执行完成之后，会立即执行该指令。

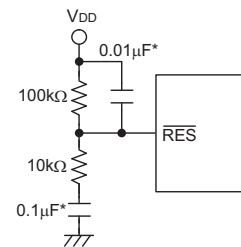
为减小功耗，在进入暂停模式之前，应小心处理所有的输入/输出口状态。在 HALT 模式下，32.768kHz 的晶体可由掩膜选项设置为运行或停止。

### 复位

总共有三种方法会产生初始复位：

- 正常运行时由 RES 引脚发生复位。
- 在暂停模式由 RES 引脚发生复位。
- 正常运行时由看门狗定时器溢出发生复位。

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



复位电路

注：“\*”连线应该尽量靠近 RES

TO	PDF	复位原因
0	0	上电时 RES 发生复位
u	u	正常运行时 $\overline{\text{RES}}$ 发生复位
0	1	暂停模式下 $\overline{\text{RES}}$ 发生复位
1	u	正常运行时 WDT 溢出
1	1	暂停模式下 WDT 溢出

注：“u”表示不变

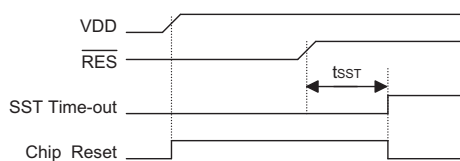
为了保证系统振荡器起振并稳定运行，系统复位（包括上电复位、WDT 溢出或由  $\overline{\text{RES}}$  端复位）或由暂停状态唤醒时，系统启动定时器（SST）提供了一个额外的延迟时间，共 1024 个系统时钟周期。

系统复位时，SST 会被加在复位延时中；由暂停模式唤醒也会加入 SST 延迟。

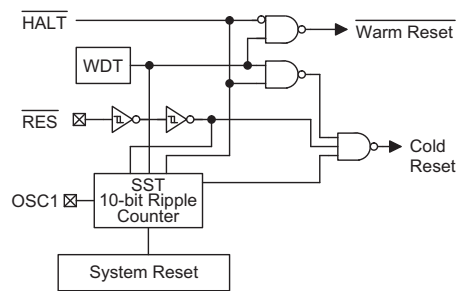
系统复位（包括上电复位、正常运行时 WDT 溢出或由  $\overline{\text{RES}}$  端复位）需要额外增加一个加载掩膜选项（Option）的时间。

系统复位时各功能单元的状态如下所示：

Program Counter	000H
中断	禁止
WDT	清除，在主系统复位后，WDT 开始计数
定时/计数器	停止
输入/输出端口	输入模式
堆栈指针 SP	指向堆栈顶部



复位时序图



复位配置

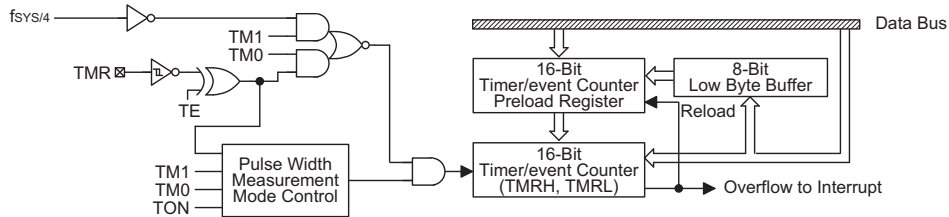
有关寄存器的状态如下:

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*
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCH.PCL	000H	000H	000H	000H	000H
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
RTCC	--xx x111	--xx x111	--xx x111	--xx x111	--uu uuuu
STATUS	--00 xxxx	--lu uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu
TMRL	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRC	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
PWM0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PWM1	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PWM2	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PWM3	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	0--- -100	0--- -100	0--- -100	0--- -100	u--- -uuu

注: 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

HT46x63 提供一个 16 位可编程向上计数的定时/计数器。定时/计数器的时钟来源可以是外部信号输入或内部时钟。



定时/计数器

如果用做内部定时器方式，则时钟源为  $f_{SYS}/4$ 。外部信号输入可以用来计数外部事件、测量时间间隔、测量脉冲宽度或产生一个精确的时基信号。

有三个与定时/计数器有关的寄存器，TMRH (0CH)、TMRL (0DH) 和 TMRC (0EH)。写入 TMRL 只能将数据写到低字节缓冲器，而写入 TMRH 会把指定数据和低字节缓冲器的数据写到预置寄存器（16 位）中，定时/计数器预置寄存器的内容只有在写入 TMRH 时才会被改变而写 TMRL 不会改变预置寄存器的值。

读取 TMRH 会把 TMRH 的内容送至目标单元，而 TMRL 的值被送至低字节缓冲器中。读 TMRL 将读取低字节缓冲器的值。换言之，定时/计数器的低字节内容是无法直接读取的。必须先读取 TMRH，将定时/计数器的低字节内容送至低字节缓冲器。TMRC 是定时/计数器控制寄存器，用来定义定时/计数器一些选项。

名称	位	功能
—	0~2	未用，读出为“0”
TE	3	定义定时/计数器 TMR 的触发方式 在事件计数模式 (TM1, TM0) = (0, 1): 1: 在下降沿计数 0: 在上升沿计数 在脉冲宽度测量模式 (TM1, TM0) = (1, 1): 1: 在上升沿开始计数，下降沿停止计数 0: 在下降沿开始计数，上升沿停止计数
TON	4	打开/关闭定时/计数器 (1=打开, 0=关闭)
—	5	未用，读出为“0”
TM0 TM1	6 7	定义工作模式: 01=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(0EH) 寄存器

TM0、TM1 用来定义定时/计数器的工作模式。外部事件计数模式是用来记录外部事件的，其时钟来源为外部 TMR 引脚输入。定时器模式是一个常用模式，其时钟来源为内部时钟  $f_{SYS}/4$ 。脉宽测量模式可以测量 TMR 引脚高/低电平的脉冲宽度，其时钟来源为内部时钟  $f_{SYS}/4$ 。

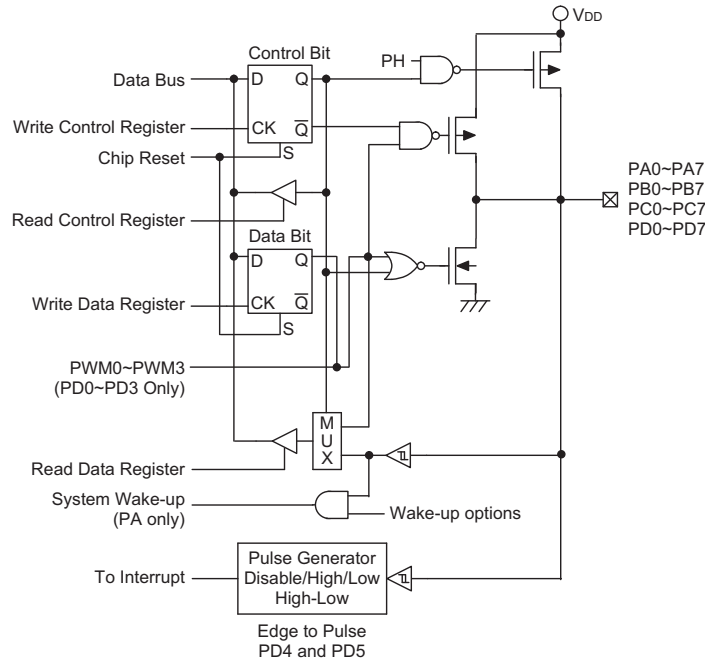
无论是定时模式还是外部事件计数模式，一旦开始计数，定时/计数器会从寄存器当前值向上计到 0FFFFH。一旦发生溢出，定时/计数器会从预置寄存器中重新加载初值，并开始计数；同时置位中断请求标志 (TF; INTC0 的第 5 位)。

在脉宽测量模式，当 TON 与 TE 是 1 时，只要 TMR 引脚有一个上升沿信号（如果 TE 是 0，则为下降沿信号），定时/计数器就会开始计数，直到 TMR 脚电平恢复，同时 TON 被清零。测量的结果会保存在寄存器中，直到有新的测量开始。换句话说，一次只能测量一个脉冲宽度。重新置位 TON 后，可以继续测量。注意，在该模式下，定时/计数器是跳变触发而不是电平触发。当计数器溢出时，定时/计数器会从预置寄存器中重新加载初值，并置位中断请求标志，这与其它两种模式一样。要启动计数器，只要置位 TON（TMRC 的第 4 位）。在脉宽测量模式下，TON 在测量结束后会被自动清除；但在另外两种模式中，TON 只能由指令来清除。定时/计数器溢出可以做为唤醒信号。不管是什么模式，只要写 0 到 ETI 即可禁止定时/计数器中断服务。

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

**输入/输出口**

HT46x63 有 32 位双向输入/输出口，记为 PA、PB、PC 和 PD，其分别对应 RAM 地址[12H]，[14H]、[16H]和[18H]，所有端口都可以进行输入/输出操作。输入时，端口没有锁存功能，输入信号必须在 MOV A, [m]（m=12H、14H、16H 或 18H）指令的 T2 上升沿到来前准备好；输出时，端口有锁存功能，端口上的数据会保持不变直到执行下一个写入操作。



输入/输出口

每个输入/输出口都有一个控制寄存器（PAC, PBC, PCC, PDC），用来控制输入/输出状态。利用控制寄存器，可对 CMOS 输出、带或不带上拉电阻的斯密特触发输入通过软件动态地进行改变。做为输入时，对应的控制寄存器应设置为“1”。输入信号来源也取决于控制寄存器，如果控制寄存器的值为“1”，那么读取的是引脚状态；如控制寄存器的值为“0”，则读取的是内部锁存器的值。后者可能会在‘读-修改-写’指令中发生。做为输出时，只能采用 CMOS 输出。控制寄存器对应 RAM 地址 13H、15H、17H、19H。

系统复位之后，这些输入/输出口会是高电平或浮空状态（由上拉电阻选项决定）。每一个输入/输出锁存位都能用“SET [m].i”或“CLR [m].i”指令置位或清除（m=12H、14H、16H 或 18H）。

有些指令会先输入数据，然后进行输出操作。例如：“SET [m].i”，“CLR [m].i”，“CPL [m]”，“CPLA[m]”这些指令会先将整个端口状态读入 CPU 中，接着执行所定义的运算（位操作），然后再将结果写入锁存器或累加器中。

PA 的每一个口都具有唤醒系统的能力。每一个输入/输出口都可以由掩膜选项来选择为带上拉电阻。

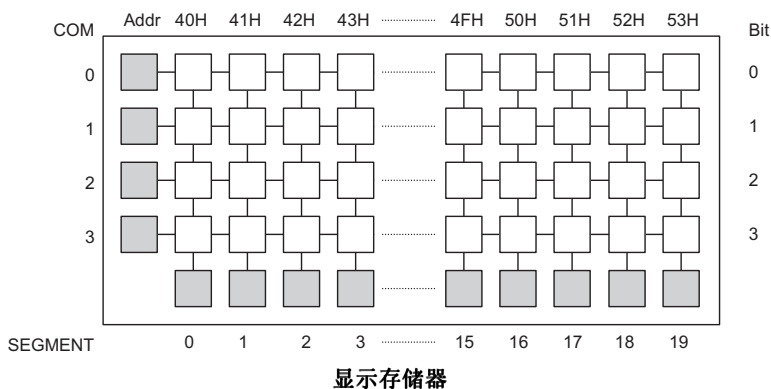
## 比较器

在 46x63 中提供一个比较器，可由掩膜选项设置为打开/关闭。其输入为 CMPP (+) 和 CMPN (-)，输出为 CMPO 和 CHGO。当 CMPN 的输入电压小于 CMPP 的输入电压时，CMPO 输出为  $V_{DD}$ ；当 CMPN 的输入电压大于 CMPP 的输入电压时，CMPO 的输出为  $V_{SS}$ 。如果选用了 32768Hz RTC 振荡器，则 CHGO 输出为带 32768Hz 载波的 CMPO 信号。

该比较器可由掩膜选项设置为关闭。在暂停模式下，会关闭比较器以降低功耗。一旦比较器关闭，则 CHGO 与 CMPO 保持为  $V_{SS}$ 。

### LCD 显示存储器

HT46x63 为 LCD 显示提供一个嵌入式数据存储器区域。这个区域位于第一段数据存储器 (RAM Bank 1) 的 40H 到 53H 单元。存储器段指针 Bank Pointer (BP; RAM 的 04H 单元) 是通用存储器 LCD 显示存储器之间切换的开关。当 BP 被置“1”，任何数据写入 40H~53H (用 MP1 和 R1 间接寻址访问) 将会影响 LCD 的显示。当 BP 被清“0”，任何数据写入 40H~53H 意味着访问一般意义上的数据存储器。LCD 显示存储器能被读出和写入，但是只能通过间接寻址模式，使用 MP1 来进行。当数据被写入显示数据区域，这些数据自动地被 LCD 驱动器读取来产生相应的 LCD 驱动信号。把“1”或“0”写入显示存储器的相应位，可以控制显示或不显示。图示表达了 HT46x63 显示存储器和 LCD 显示模块之间的映射关系。

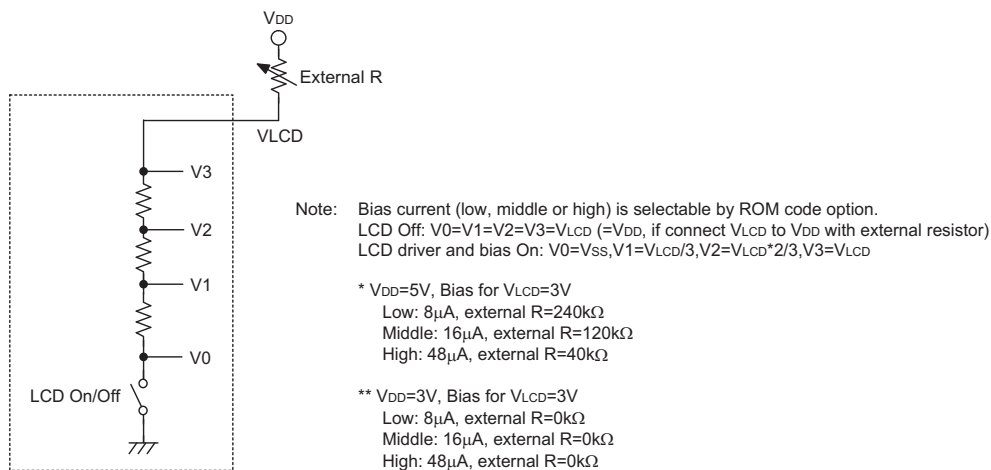


### LCD 驱动输出

HT46x63 LCD 驱动器的输出数目可以由掩膜选项确定为 20×3 或 19×4 (即 1/3 或 1/4 占空比)。LCD 驱动器的偏压种类为“R”型，不需要外接电容器。LCD 可掩膜选择确定“在暂停时 LCD 打开”或“在暂停时 LCD 关闭”。

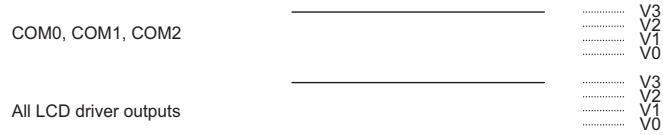
SEG7~SEG18 可掩膜选择为逻辑输出。掩膜选择时 SEG7~SEG10, SEG11~SEG14, SEG15~SEG18 为各自独立的组。一旦 LCD 设置为逻辑输出，LCD 存储区的 bit0 将控制相关 segment 口的输出状况，如下图。

RAM 内容	Segment 输出
Bit0 = 0	V <sub>SS</sub>
Bit0 = 1	V <sub>DD</sub>

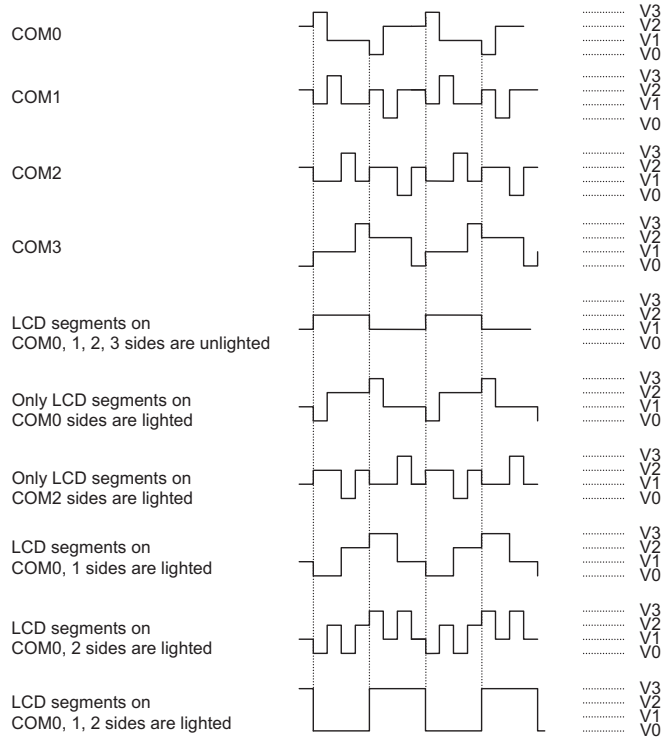


**LCD 偏压框图和应用电路**

**During a reset pulse**



**Normal operation mode**



**HALT mode (LCD off at HALT)**



**LCD 驱动输出 (1/4duty, 1/3bias)**

注：若 LCD 在暂停模式开启，则 LCD 的输出决定于 LCD 显示存储器  
若 LCD 在暂停模式关闭，则  $V3 = V2 = V1 = V0 = V_{DD}$

## A/D 转换

HT46x63 有 8 个通道、8 位解析度（7 位精度）的 A/D 转换器。其参考电压为 AVDD，AVDD 引脚必须连接到 VDD。如果电源负载过大或电源线连接不合理造成电源波动和干扰，都会影响到 A/D 转换的精度。工作电压的抖动和干扰，与 A/D 转换有关的寄存器有 3 个，ADR (21H)、ADCR (22H) 和 ACSR (23H)。ADR 是 A/D 转换结果寄存器，并且是只读寄存器。当 A/D 转换完成后，ADR 应被读出以获取转换结果。ADCR 是 A/D 转换控制寄存器，用来定义 A/D 转换通道数、模拟通道选择、A/D 转换开始控制和完成标志。如果要进行 A/D 转换，要先定义好 PB 口的设置，选择转换的模拟通道，然后给 START 控制位一个上升沿信号和一个下降沿信号（0→1→0）。完成 A/D 转换后， $\overline{EOC}$  位会被清除，并且产生 A/D 转换中断（如果 A/D 转换允许）。ACSR 是 A/D 时钟设定寄存器，用来选择 A/D 的时钟来源。

A/D 转换控制寄存器用来控制 A/D 转换。ADCR 的第 2 位~第 0 位用来选择模拟信号输入通道，总共有 8 个通道可以选择。ADCR 的第 5 位~第 3 位用来设置 PB 的工作模式，PB 可以做为模拟信号的输入通道，或是数字输入/输出口，即由这 3 位来决定。如果 PB 选择为模拟输入，则其输入/输出功能和对应的 I/O 口线的上拉电阻将无效。 $\overline{EOC}$  位（ADCR 的第 6 位）是 A/D 转换结束标志位。通过检测这个标志位可以知道 A/D 转换是否结束。ADCR 的起始位 START 用来开启 A/D 转换，给 START 位一个下降沿信号可以开始 A/D 转换。为了确保 A/D 转换顺利完成，START 位应保持为“0”，直到  $\overline{EOC}$  位变为“0”（A/D 转换完成信号）。

ACSR 的第 7 位是用来内部检测的，不提供给用户使用。ACSR 的第 1 位和第 0 位用来选择 A/D 转换的时钟来源。

当 A/D 转换完成时，A/D 中断标志被置位。当 START 标志被置为“1”时， $\overline{EOC}$  也由“0”置为“1”。A/D 转换初始化注意事项：

每次改变模拟通道选择位后都应注意初始化 A/D 转换器，否则  $\overline{EOC}$  可能处于不确定状态。在模拟通道选择位改变的 10 个指令周期内将 START 置 1 后清 0 来初始化 A/D 转换器。模拟通道选择位都清 0，可以不初始化 A/D。

符号 (ADSR)	位	功能
ADCS0 ADCS1	0 1	ADCS1, ADCS0: 选择 A/D 转换时钟源 0, 0=系统时钟/2 0, 1=系统时钟/8 1, 0=系统时钟/32 1, 1=未定义
CMPC	2	比较器控制 (*) 0: 关闭 1: 打开
—	3~6	未用, 读出为“0”
TEST	7	仅用于测试, 不可使用

ACSR(23H) 寄存器

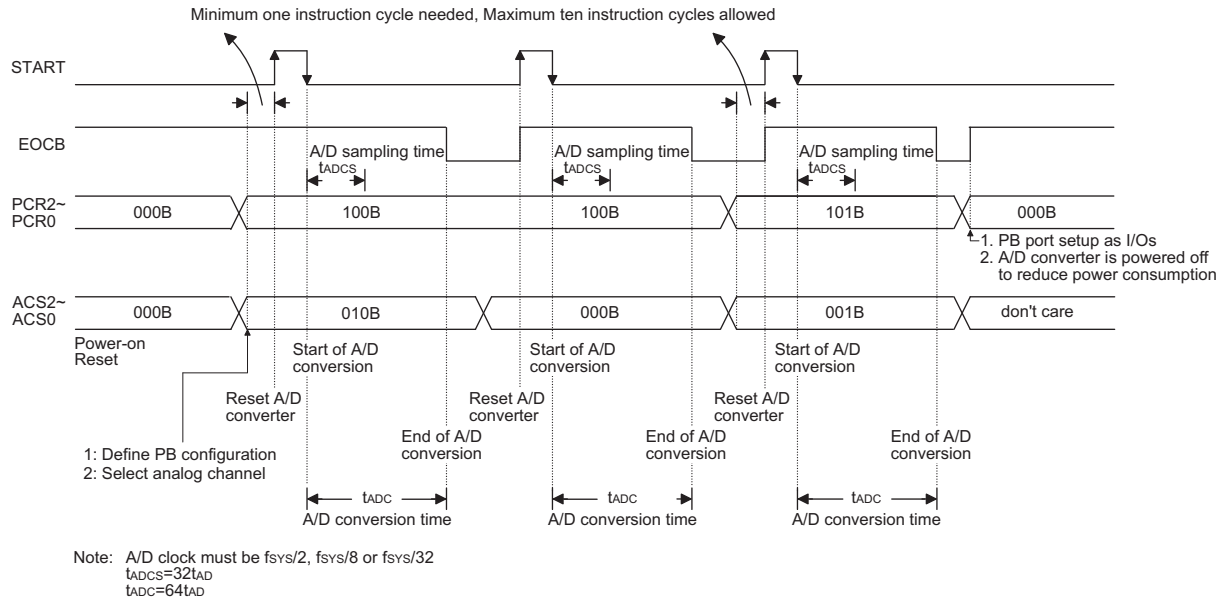
注：“\*”在复位后，此位为 0

符号(ADCR)	位	功能
ACS0 ACS1 ACS2	0 1 2	ACS2, ACS1, ACS0: 选择模拟输入通道 0, 0, 0: AN0 0, 0, 1: AN1 0, 1, 0: AN2 0, 1, 1: AN3 1, 0, 0: AN4 1, 0, 1: AN5 1, 1, 0: AN6 1, 1, 1: AN7
PCR0 PCR1 PCR2	3 4 5	PCR2, PCR1, PCR0: 定义 PB 口的设置 0, 0, 0: PB7, PB6, PB5, PB4, PB3, PB2, PB1, PB0 0, 0, 1: PB7, PB6, PB5, PB4, PB3, PB2, PB1, AN0 0, 1, 0: PB7, PB6, PB5, PB4, PB3, PB2, AN1, AN0 0, 1, 1: PB7, PB6, PB5, PB4, PB3, AN2, AN1, AN0 1, 0, 0: PB7, PB6, PB5, PB4, AN3, AN2, AN1, AN0 1, 0, 1: PB7, PB6, PB5, AN4, AN3, AN2, AN1, AN0 1, 1, 0: PB7, PB6, AN5, AN4, AN3, AN2, AN1, AN0 1, 1, 1: AN7, AN6, AN5, AN4, AN3, AN2, AN1, AN0
$\overline{\text{EOC}}$	6	A/D 转换结束标志(0: A/D 转换结束) 每次 BIT3-5 状态的改变都必须通过 START 信号来初始化 A/D 转换器, 否则 $\overline{\text{EOC}}$ 可能会处于不确定状态, 具体可参照“A/D 转换初始化注意事项”
START	7	A/D 转换起始控制位 0→1→0: 开始; 0→1: A/D 转换复位并且置 $\overline{\text{EOC}}$ 为“1”

ADCR(22H) 寄存器

寄存器	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADR	D7	D6	D5	D4	D3	D2	D1	D0

ADR(21H) 的寄存器



A/D 转换时序图

下面举两个例子说明如何启动和实现 A/D 转换。第一个例子不断扫描 ADCR 寄存器的  $\overline{EOC}$  位来判断 A/D 转换是否完成；而第二个例子直接用中断的方法来判断 A/D 转换是否完成。

例 1：通过扫描  $\overline{EOC}$  位判断 A/D 转换是否完成。

```

clr    EADI           ;禁止A/D中断
mov    a,00000001B
mov    ACSR,a        ; 设置ACSR寄存器，选择fsys/8做为A/D转换时钟
mov    a,00100000B   ; 在ADCR寄存器中设置Port PB0~PB3做为A/D输入
mov    ADCR,a        ; 设置AN0进行A/D转换
:
:
:                   ; 当模拟通道选择位改变后，START信号（0-1-0）必须在10个
:                   ; 指令周期内发出

```

Start\_conversion:

```

clr    START
set    START         ; A/D转换复位
clr    START         ; 开始A/D转换

```

Polling\_EOC:

```

sz     EOC           ; 扫描ADCR寄存器的 $\overline{EOC}$ 位判断A/D转换是否完成
jmp    polling_EOC  ; 继续扫描
mov    a,ADR        ; 从ADR寄存器读取A/D转换的结果
mov    adr_buffer,a ; 将结果放入用户定义的寄存器中
:
:
jmp    start_conversion ; 开始下一次A/D转换

```

例 2: 用中断方法判断 A/D 转换是否完成。

```

clr    EADI          ; 禁止A/D中断
mov    a,00000001B
mov    ACSR,a        ; 设置ACSR寄存器, 选择fsys/8做为A/D转换时钟
mov    a,00100000B
mov    ADCR,a        ; 在ADCR寄存器中设置Port PB0~PB3做为A/D输入
:      ; 设置AN0进行A/D转换
:
:      ; 当模拟通道选择位改变后, START信号(0-1-0)必须在10个
:      ; 指令周期内发出

start_conversion:
clr    START
set    START         ; A/D转换复位
clr    START         ; 开始A/D转换
clr    ADF           ; 清除AD中断请求标志
set    EADI          ; 打开 A/D 中断
set    EMI           ; 打开总中断
:
:

; 中断服务子程序
ADC_ISR:
mov    acc_stack,a   ; 将ACC保存到用户定义的寄存器中
mov    a,STATUS
mov    status_stack,a ; 将STATUS保存到用户定义的寄存器中
:
:
mov    a,ADR         ; 从ADR寄存器读取A/D转换的结果
mov    adr_buffer,a ; 将结果放入用户定义的寄存器中
clr    START
set    START         ; A/D转换复位
clr    START         ; 开始A/D转换
:
:

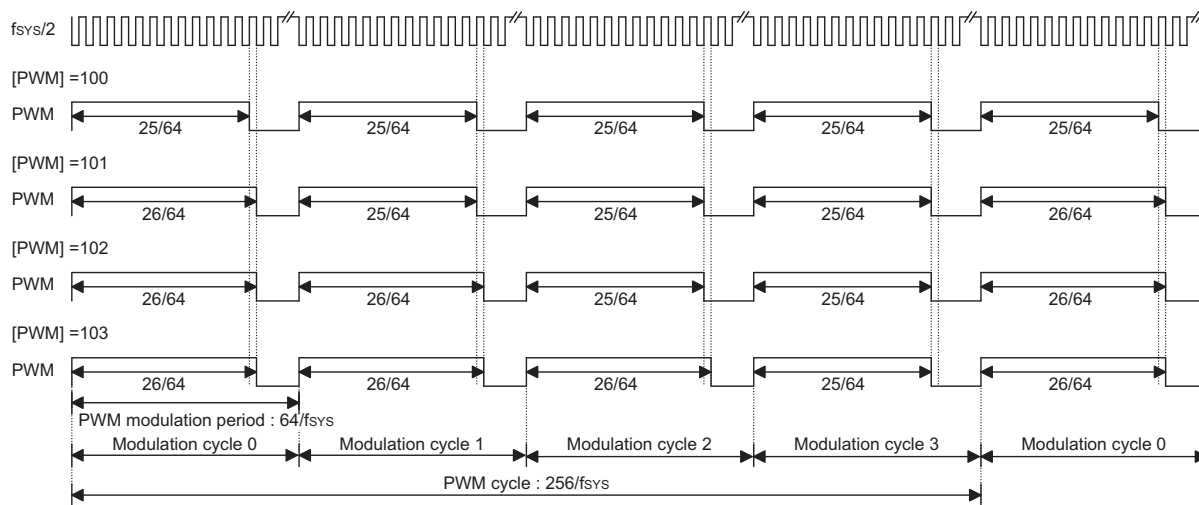
EXIT_INT_ISR:
mov    a,status_stack
mov    STATUS,a      ; 将STATUS从暂存器中读出
mov    a,acc_stack   ; 将ACC从暂存器中读出
reti

```

**PWM**

HT46x63 有 4 个通道 (6+2) 位的 PWM 输出, 与 PD0~PD3 共用引脚。PWM 通道由相应的数据寄存器来控制, 它们使用 8 阶的 PWM 计数器 (stage1~stage8:  $f_{SYS}/2^1 \sim f_{SYS}/2^8$ )。PWM 计数器的时钟来源为系统时钟 ( $f_{SYS}$ )。PWM 寄存器是 8 位的寄存器。PWM 的输出波形如图所示。一旦 PDi ( $i = 0 \sim 3$ ) 选择为 PWMi 输出, 并且 PDi 为输出模式 (PDi=“0”), 则向 PDi 寄存器写“1”能够产生 PWMi 输出, 向 PDi 寄存器写“0”会使 PDi 输出保持为“0”。PWM 的调制频率, PWM 的周期频率以及 PWM 的占空比总结见下表。

PWM 调制频率	PWM 周期频率	PWM 占空比
$f_{SYS}/64$	$f_{SYS}/256$	$[PWM]/256$


**PWM 模式**

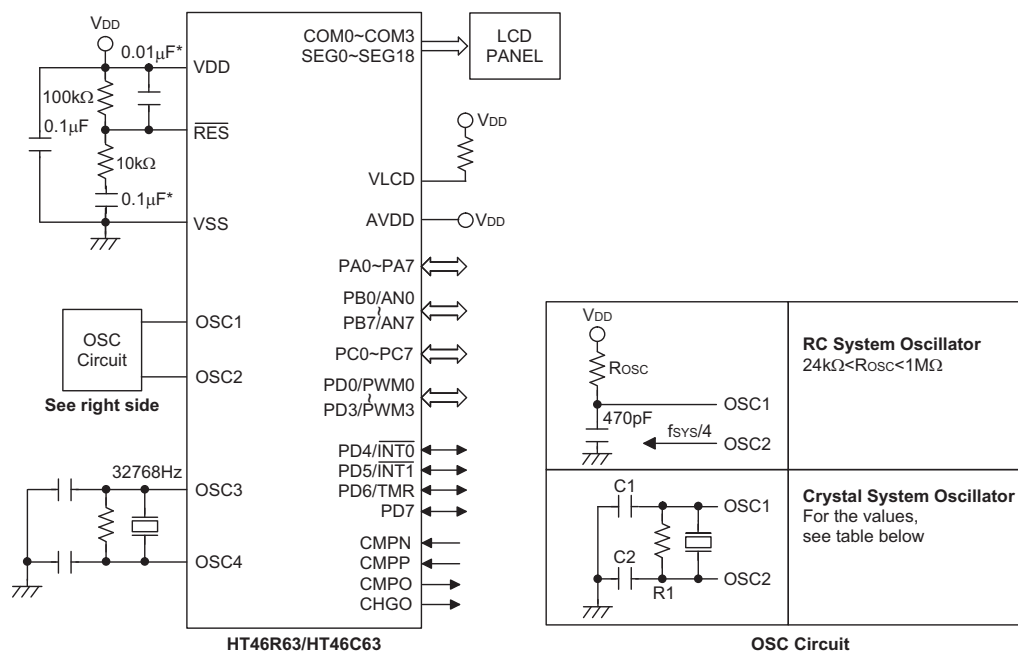
## 掩膜选项

下表列出了所有掩膜选项。所有选项必须正确定义，以保证系统正常运行。

编号	选项
1	PA 唤醒功能打开/关闭 (1/0) 选择
2	WDT/LCD/RTC/Time Base 时钟来源 (fs): RTCOSC (32768Hz 晶体), T1 或 WDTOSC (*1)
3	清除 WDT 指令条数选择: 1/2
4	WDT 打开/关闭
5	PA 口上拉电阻选择 (4bits 一组 (0~3/4~7))
6	PB 口上拉电阻选择 (4bits 一组 (0~3/4~7))
7	PC 口上拉电阻选择 (4bits 一组 (0~3/4~7))
8	PD 口上拉电阻选择 (4bits 一组 (0~3/4~7))
9	INT0 或 INT1 触发沿: 禁止, 上升沿触发, 下降沿触发, 上升沿或下降沿触发
10	COM3 或 SEG19 (1/4 或 1/3duty)
11	暂停模式下 LCD 的开/关
12	比较器打开/关闭
13	PWMi 功能选择 (位选择)
14	时基周期选择: $f_s/2^{12} \sim f_s/2^{15}$
15	SEG7~SEG18 逻辑输出或 LCD 输出: (每 4bits 一组 (SEG7~SEG10/SEG11~SEG14/SEG15~SEG18))
16	系统时钟: 外部 RC 振荡/外部晶体振荡
17	暂停模式下 RTCOSC (32768Hz) 或 WDTOSC 停止/运行
18	LCD 偏置电流选择: 低/中/高电流
19	LCD 驱动输出时钟选择 总共有 7 种 LCD 驱动输出频率可以选择: $f_s/2^2 \sim f_s/2^8$ ; 其中 $f_s$ 是由 Option 设置的时钟源。

注: “\*1” 在暂停模式下 T1 停止; RTCOSC(32768Hz)/WDTOSC 可由第 17 项掩膜选择在暂停模式时是否工作。

## 应用电路



下表是不同晶体频率时，C1、C2 和 R1 的不同取值。

晶体或共振器	C1、C2	R1
4MHz 晶体	0pF	10kΩ
4MHz 共振器	10pF	12kΩ
3.58MHz 晶体	0pF	10kΩ
3.58MHz 共振器	25pF	10kΩ
2MHz 晶体和共振器	25pF	10kΩ
1MHz 晶体	35pF	27kΩ
480kHz 共振器	300pF	9.1kΩ
455kHz 共振器	300pF	10kΩ
429kHz 共振器	300pF	10kΩ

R1 的作用是在低电压的时候确保关闭振荡，此低电压值低于单片机的最低工作电压。

注：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內。

“\*” 为了避免噪声干扰，连接 RES 引脚的线请尽可能地短

## 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： $Stack \leftarrow Program Counter + 1$   
 $Program Counter \leftarrow addr$   
 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
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
0	0	—	—	—	—

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

影响标志位

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

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

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:  $[m] \leftarrow [\bar{m}]$

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:  $ACC \leftarrow [\bar{m}]$

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。结果存放到数据存储器中，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] 数据存储器的内容减 1，结果放入数据存储器  
 说明： 本指令将数据存储器内的数值减一再放回数据存储器。

运算过程： $[m] \leftarrow [m] - 1$

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

**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： $ACC \leftarrow [m] - 1$

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： $Program\ Counter \leftarrow Program\ Counter + 1$

$PDF \leftarrow 1$

$TO \leftarrow 0$

影响标志位

TO	PDF	OV	Z	AC	C
0	1	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

**INCA [m]** 数据存储器的内容加 1，结果放入数据存储器  
 说明：本指令是将存储器内的数值加一，结果放到累加器。  
 运算过程： $ACC \leftarrow [m]+1$   
 影响标志位

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： $Program Counter \leftarrow addr$   
 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： $ACC \leftarrow [m]$   
 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： $ACC \leftarrow x$   
 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： $[m] \leftarrow ACC$   
 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:                 $\text{Program Counter} \leftarrow \text{Program Counter} + 1$   
 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:                 $\text{ACC} \leftarrow \text{ACC} \text{ "OR" } [m]$   
 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:                 $\text{ACC} \leftarrow \text{ACC} \text{ "OR" } x$   
 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:                 $[m] \leftarrow \text{ACC} \text{ "OR" } [m]$   
 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:                 $\text{Program Counter} \leftarrow \text{Stack}$   
 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:                 $\text{Program Counter} \leftarrow \text{Stack}$   
                                $\text{ACC} \leftarrow x$

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:                 $\text{Program Counter} \leftarrow \text{Stack}$   
 $\text{EMI} \leftarrow 1$

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:                 $[\text{m}].0 \leftarrow [\text{m}].7, [\text{m}].(i+1) \leftarrow [\text{m}].i; (i=0\sim6)$

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:                 $\text{ACC}.0 \leftarrow [\text{m}].7, \text{ACC}.(i+1) \leftarrow [\text{m}].i; (i=0\sim6)$

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:                 $[\text{m}].(i+1) \leftarrow [\text{m}].i; (i=0\sim6)$

$[\text{m}].0 \leftarrow \text{C}$

$\text{C} \leftarrow [\text{m}].7$

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:                 $\text{ACC}.(i+1) \leftarrow [\text{m}].i; (i=0\sim6)$

$\text{ACC}.0 \leftarrow \text{C}$

$\text{C} \leftarrow [\text{m}].7$

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
—	—	—	—	—	√

**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： $[m] \leftarrow ACC + [\overline{m}] + C$

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；  $[m] \leftarrow [m]+1$   
 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；  $ACC \leftarrow ([m]+1)$   
 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  $[m].i \neq 0$ ，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：           $ACC \leftarrow ACC + [\bar{m}] + 1$   
 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：           $ACC \leftarrow ACC + \bar{x} + 1$   
 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： $[m] \leftarrow ACC + [\overline{m}] + 1$   
 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，将低字节送到存储器，高字节直接送到 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，将低字节送到存储器，高字节直接送到 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

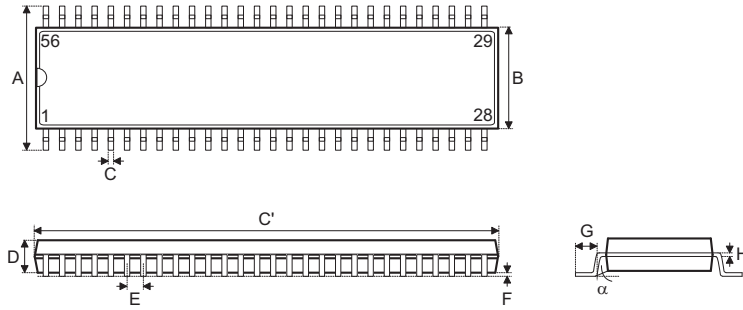
TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

**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
—	—	—	√	—	—

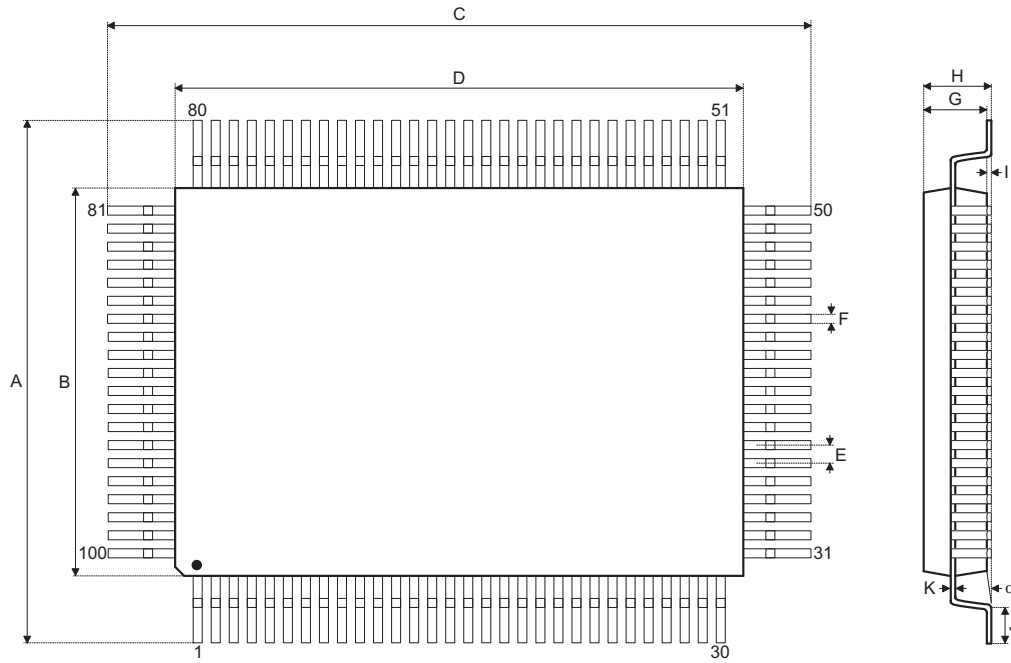
封装信息

56-pin SSOP (300mil) 外形尺寸



符号	尺寸 (单位: mil)		
	最小	正常	最大
A	395	—	420
B	291	—	299
C	8	—	12
C'	720	—	730
D	89	—	99
E	—	25	—
F	4	—	10
G	25	—	35
H	4	—	12
$\alpha$	0°	—	8°

100-pin QFP (14×20)外形尺寸



符号	尺寸 (单位: mm)		
	最小	正常	最大
A	18.50	—	19.20
B	13.90	—	14.10
C	24.50	—	25.20
D	19.90	—	20.10
E	—	0.65	—
F	—	0.30	—
G	2.50	—	3.10
H	—	—	3.40
I	—	0.10	—
J	1	—	1.40
K	0.10	—	0.20
$\alpha$	0°	—	7°

**盛群半导体股份有限公司（总公司）**

新竹市科学工业园区研新二路3号  
电话: 886-3-563-1999  
传真: 886-3-563-1189  
网站: [www.holtek.com.tw](http://www.holtek.com.tw)

**盛群半导体股份有限公司（台北业务处）**

台北市南港区园区街3之2号4楼之2  
电话: 886-2-2655-7070  
传真: 886-2-2655-7373  
传真: 886-2-2655-7383 (International sales hotline)

**盛扬半导体有限公司（上海业务处）**

上海宜山路889号2号楼7楼 200233  
电话: 021-6485-5560  
传真: 021-6485-0313  
网站: [www.holtek.com.cn](http://www.holtek.com.cn)

**盛扬半导体有限公司（深圳业务处）**

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

**盛扬半导体有限公司（北京业务处）**

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

**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](http://www.holtek.com)

Copyright © 2007 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>