

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](#)
 - [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](#)
 - [HA0011S HT48 & HT46 MCU 键盘扫描程序](#)
 - [HA0013S HT48 & HT46 MCU LCM 接口设计](#)
 - [HA0075S MCU 复位和晶体振荡应用范例](#)
 - [HA0101S HT46R12 在电磁炉中的应用](#)

特性

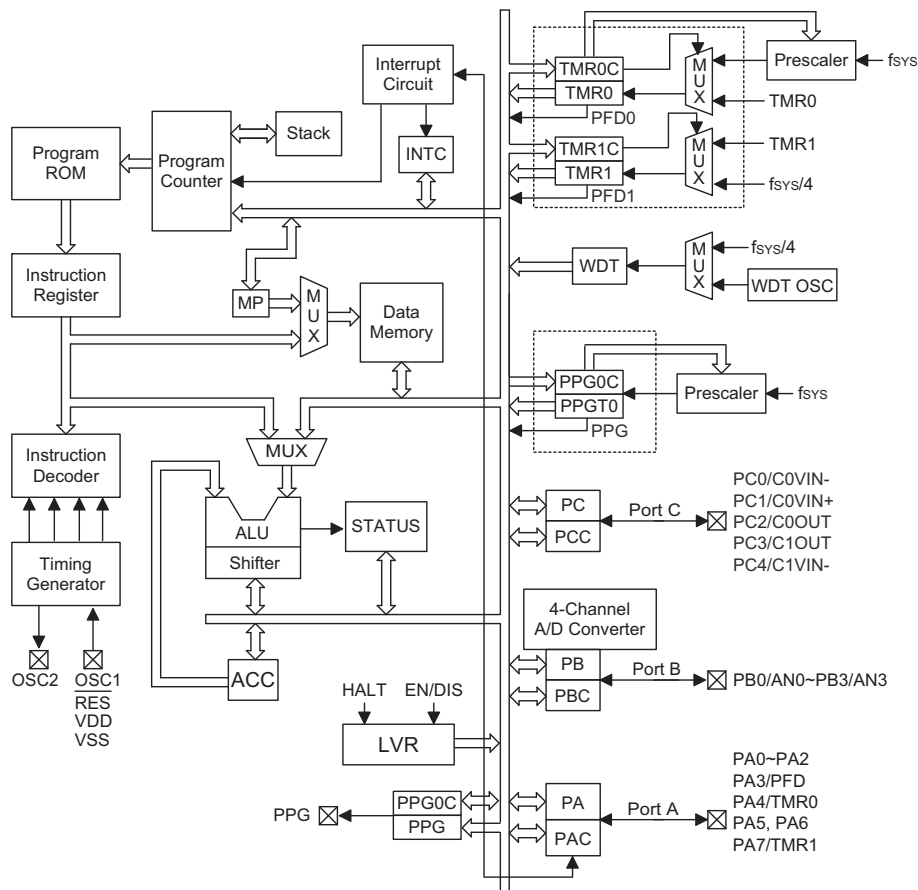
- 工作电压：
 - $f_{\text{SYS}}=4\text{MHz}$: 2.2V~5.5V
 - $f_{\text{SYS}}=8\text{MHz}$: 3.3V~5.5V
- 最多可有 17 个双向输入/输出口
- 2 组 8 位可编程定时/计数器，具有溢出中断和 7 级预分频器
- 1 组可编程脉冲发生器输出，带一组 8 位计数器，可设置预分频，输出电平可设置高或是低有效
- 内置晶体和 RC 振荡电路
- 看门狗定时器
- 2048×14 程序存储器 ROM
- 88×8 数据存储器 RAM
- 具有 PFD 功能，可用于发声
- HALT 和唤醒功能可降低功耗
- 在 $V_{\text{DD}}=5\text{V}$ ，系统频率为 8MHz 时，指令周期为 0.5 μs
- 8 层硬件堆栈
- 4 通道 9 位解析度的 A/D 转换器
- 两组比较器，可发生中断
- 位操作指令
- 查表指令，表格内容字长 14 位
- 63 条指令
- 指令执行时间为 1 或 2 个指令周期
- 低电压复位功能
- 24-pin SKDIP/SOP 封装

概述

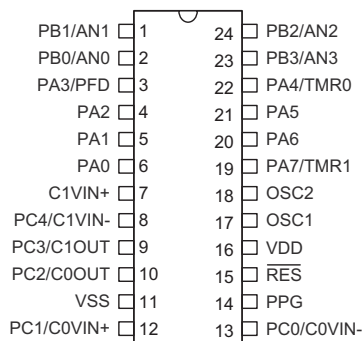
HT46R12A 是 8 位高性能精简指令集单片机，专门为需要 A/D 转换的产品而设计，例如传感器信号输入。低功耗、I/O 使用灵活、可编程分频器、计数器、振荡类型选择、多通道 A/D 转换、暂停和唤醒功能，使这款单片机可以广泛应用于带传感器的 A/D 转换产品中。

这款 MCU 提供两组比较器及一组可编程脉冲发生器的功能，使它尤其适合应用在电磁炉等家用电器产品中。

方框图



引脚图



HT46R12A
— 24 SKDIP-A/SOP-A

引脚说明

引脚名称	输入/输出	掩膜选项	说明
PA0 ~PA2 PA3/PFD PA4/TMR0 PA5、PA6 PA7/TMR1	输入/输出	上拉电阻 唤醒功能 PA3 或 PFD	8 位双向输入/输出口。每一位可由掩膜选项设置为唤醒输入。可由软件设置为 CMOS 输出、带或不带上拉电阻（由上拉电阻选项决定：位选择）的斯密特触发输入。PA3、PA4、PA7 分别与 PFD、TMR0、TMR1 共用引脚。
PB0/AN0 PB1/AN1 PB2/AN2 PB3/AN3	输入/输出	上拉电阻	4 位双向输入/输出口。每一位都具有可由软件设置为 CMOS 输出、带或不带上拉电阻（由上拉电阻选项决定：端口选择）的斯密特触发输入、或 A/D 输入。一旦 PB 有一个口做为 A/D 输入（由软件设置），则其输入/输出功能和上拉电阻会自动失效。
PC0/C0VIN- PC1/C0VIN+ PC2/C0OUT PC3/C1OUT PC4/C1VIN- C1VIN+	输入/输出	上拉电阻 输入/输出 比较器	5 位双向输入/输出口。可由软件设置为 CMOS 输出、带或不带上拉电阻（由上拉电阻选项决定：端口选择）的斯密特触发输入。 C0VIN+, C0VIN-, C0OUT 与 PC1, PC0, PC2 分别共用引脚。一旦比较器 0 使能, PC2 控制寄存器只能设置为输入, 并且 PC0/PC1/PC2 的带上拉电阻功能将自动关闭, 而 PC0/PC1 保持其输入/输出功能。比较器 0 是否使能是由软件指令来决定的。 C1VIN+, C1VIN-是比较器 1 的输入口, C1OUT 及 C1VIN-与 PC3、PC4 共用引脚。一旦比较器 1 使能, PC3 的控制寄存器仅能设置成输入状态, PC3 及 PC4 所带的上拉输入功能会立即自动失效, PC4 保持其输入/输出功能。比较器 1 是否使能是由软件指令来决定的。 PC1/COVIN+同时也是外部中断输入口, 不管是做为比较器的输入口还是做为普通 IO 口使用, 只要在这个引脚上发生一个下降沿即可以触发中断。
PPG	输出	—	可编程脉冲发生器的输出口, 在上电复位时它的端口状态为浮态。PPG0 输出端状态（输出高或低脉冲）是由相关选项配置来定的。
OSC1 OSC2	输入 输出	晶体或 RC	OSC1、OSC2 连接 RC 或晶体(由掩膜选项确定)以产生内部系统时钟。在 RC 振荡方式下, OSC2 是系统时钟四分频的输出口。
RES	输入	—	斯密特触发复位输入, 低电平有效。
VDD	—	—	正电源。
VSS	—	—	负电源, 接地。

极限参数

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

注：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 _{DD}	条件				
V _{DD}	工作电压	—	f _{SYS} =4MHz	2.2	—	5.5	V
		—	f _{SYS} =8MHz	3.3	—	5.5	V
I _{DD1}	工作电流（晶体振荡）	3V	无负载, f _{SYS} =4MHz	—	0.6	1.5	mA
		5V	ADC 关闭	—	2	4	mA
I _{DD2}	工作电流（RC 振荡）	3V	无负载, f _{SYS} =4MHz	—	0.8	1.5	mA
		5V	ADC 关闭	—	2.5	4	mA
I _{DD3}	工作电流 （晶体振荡, RC 振荡）	5V	无负载, f _{SYS} =8MHz ADC 关闭	—	4	8	mA
I _{STB1}	静态电流（看门狗打开）	3V	无负载, 系统 HALT	—	—	5	μA
		5V		—	—	10	μA
I _{STB2}	静态电流（看门狗关闭）	3V	无负载, 系统 HALT	—	—	1	μA
		5V		—	—	2	μA
V _{IL1}	输入/输出、TMR0、 TMR1 的低电平输入电压	—	—	0	—	0.3 V _{DD}	V
V _{IH1}	输入/输出、TMR0、 TMR1 的高电平输入电压	—	—	0.7V _{DD}	—	V _{DD}	V
V _{IL2}	低电平输入电压（ $\overline{\text{RES}}$ ）	—	—	0	—	0.4 V _{DD}	V
V _{IH2}	高电平输入电压（ $\overline{\text{RES}}$ ）	—	—	0.9 V _{DD}	—	V _{DD}	V
V _{LVR}	低电压复位	—	—	2.7	3	3.3	V
I _{OL}	输入/输出及 PPG 口灌电 流	3V	V _{OL} =0.1V _{DD}	4	8	—	mA
		5V		10	20	—	mA
I _{OH}	输入/输出及 PPG 口源电 流	3V	V _{OH} =0.9V _{DD}	-2	-4	—	mA
		5V		-5	-10	—	mA
R _{PH}	上拉电阻	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
V _{AD}	A/D 输入电压	—	—	0	—	V _{DD}	V
E _{AD}	A/D 转换误差	—	—	—	±0.5	±1	LSB
I _{ADC}	打开 ADC 增加的功耗	3V	—	—	0.5	1	mA
		5V	—	—	1.5	3	mA

注意：如果比较器的输入电压不为 V_{DD} 或 V_{SS}，这将会增加 I_{DD}/I_{STB} 电流。因为不管比较器使能或除能，比较器输入口与 IO 口复用了输入功能。

在 5V 的工作电压下，当比较器输入口电压为 2.5V 时，每个比较器输入脚上有典型的耗电流为 500μA。

交流电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
f _{SYS}	系统时钟	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
f _{TIMER}	定时器输入频率 (TMR0/TMR1)	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	kHz
t _{WDTOSC}	看门狗振荡器周期	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t _{RES}	外部复位低电平脉宽	—	—	1	—	—	μs
t _{SST}	系统启动延迟时间	—	上电或从 HALT 状态 唤醒	—	1024	—	*t _{SYS}
t _{INT}	中断脉冲宽度	—	—	1	—	—	μs
t _{AD}	A/D 时钟周期	—	—	1	—	—	μs
t _{ADC}	A/D 转换时间	—	—	—	76	—	t _{AD}
t _{ADCS}	A/D 采样时间	—	—	—	32	—	t _{AD}

 注: *t_{SYS} = 1/f_{SYS}
比较器的电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
	比较器工作电压	—	—	2.2	—	5.5	V
	比较器工作电流	5V	—	—	—	200	μA
V _{OPOS1}	比较器输入偏移电压	5V	—	-10	—	10	mV
V _{OPOS2}	比较器输入偏移电压	5V	通过校正	-2	—	2	mV
V _{CM}	比较器共模输入电压范围	—	—	VSS	—	V _{DD} -1.4	V
t _{PD}	比较器响应时间	—	以 10 mV 为跨度	—	—	2	μs

注意: 如果比较器的输入电压不为 VDD 或 VSS, 这将会增加 I_{DD}/I_{STB} 电流。因为不管比较器使能或除能, 比较器输入口与 IO 口复用了输入功能。

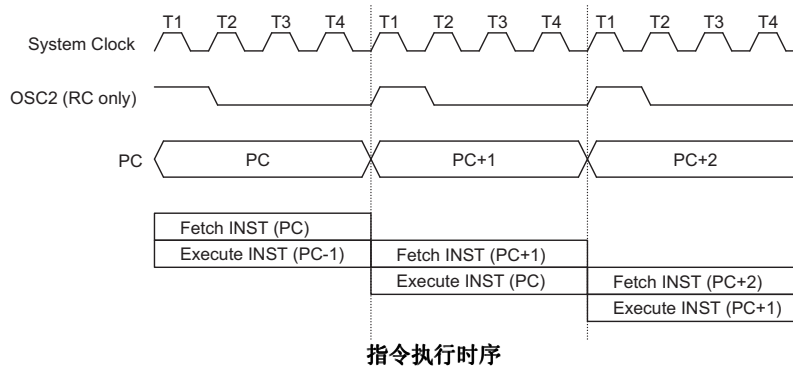
在 5V 的工作电压下, 当比较器输入口电压为 2.5V 时, 每个比较器输入脚上有典型的耗电流为 500μA。

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。



程序计数器 — PC

程序计数器(PC)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 赋值、子程序调用、初始化复位、内部中断、外部中断、子程序返回等操作时，PC 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										
	*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
比较器 0 中断	0	0	0	0	0	0	0	0	1	0	0
比较器 1 中断	0	0	0	0	0	0	0	1	0	0	0
外部中断(来自 PC1)	0	0	0	0	0	0	0	1	1	0	0
定时/计数器 0 中断	0	0	0	0	0	0	1	0	0	0	0
定时/计数器 1 中断	0	0	0	0	0	0	1	0	1	0	0
A/D 转换中断	0	0	0	0	0	0	1	1	0	0	0
条件跳跃	PC+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： *10 ~ *0 : 程序计数器位
#10 ~ #0 : 指令代码位

S10 ~ S0 : 堆栈寄存器位
@7 ~ @0 : PCL 位

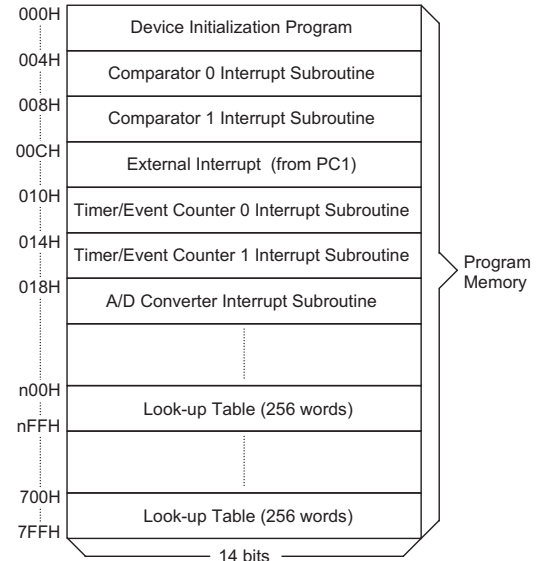
程序存储器 — ROM

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 2048×14 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
该地址为比较器 0 中断服务程序保留。当比较器 0 输出口产生下降沿，如果中断允许且堆栈未满，则程序会跳转到 004H 地址开始执行。
- 地址 008H
该地址为比较器 1 中断服务程序保留。当比较器 1 输出口产生下降沿，如果中断允许且堆栈未满，则程序会跳转到 008H 地址开始执行。
- 地址 00CH
该地址为外部中断服务程序保留。当 PC1 口产生一具下降沿，如果中断允许且堆栈未满，则程序会跳转到 00CH 地址开始执行。
- 地址 010H
该地址为定时/计数器 0 中断服务程序保留。当定时/计数器 0 溢出，如果中断允许且堆栈未满，则程序会跳转到 010H 地址开始执行。
- 地址 014H
该地址为定时/计数器 1 中断服务程序保留。当定时/计数器 1 溢出，如果中断允许且堆栈未满，则程序会跳转到 014H 地址开始执行。
- 地址 018H
该地址 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满，则程序会跳转到 018H 地址开始执行。
- 表格区

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 2 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。



程序存储器

指令	表格区										
	*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：*10~*0：表格地址位
@7~@0：表格指针位
P10~P8：当前程序指针位

堆栈寄存器 — STACK

堆栈寄存器是特殊的存储器空间，用来保存 PC 的值。HT46R12A 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（PC）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 115×8 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（88×8），数据存储器单元大多数是可读/写的，但有些只读的。

在地址 28H 以前未用到的空间是用来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 28H 到 7FH，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

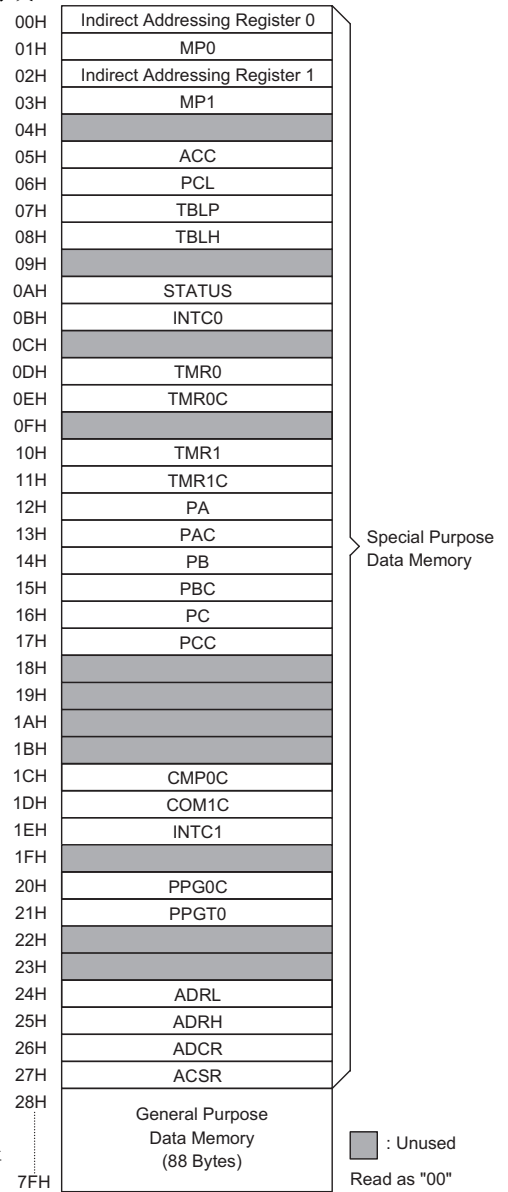
地址 00H 和 02H 是间接寻址寄存器，并无实际的物理区存在。任何对[00H]或[02H]的读/写操作，都是访问由 MP0（01H）MP1（03H）或所指向的 RAM 单元。间接读取地址 00H 或 02H 得到的值为 00H，间接写入此地址，不会产生任何操作。间接寻址寄存器之间不支持数据传送功能。

间接寻址指针寄存器 MP0（01H）和 MP1（03H）都是 7 位寄存器，被用来访问由间接寻址寄存器所对应的 RAM。

间接寻址指针寄存器 MP0 与 MP1 是 7 位寄存器，它们的第 7 位均未定义，如果读取时返回值为“1”。任何对 MP0 及 MP1 的写操作仅会改变相应寄存器的低 7 位。

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。



数据存储器

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, SD）

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。

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。与状态寄存器相关的一些操作会按预期分配给 STATUS 寄存器相应的值。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。

符号	位	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。
1	AC	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。
2	Z	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。
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，则 OV 被置位；反之，OV 被清除。
4	PDF	系统上电或执行“CLR WDT”指令，PDF 被清除；执行“HALT”指令，PDF 被置位。
5	TO	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。
6,7	—	未用，读出为“0”

STATUS(0AH) 寄存器

中断

HT46R12A 提供二个内部定时/计数器 0/1 中断、二个比较器中断、一个 A/D 转换中断。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。

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。当有中断被服务，系统会将程序计数器值压入堆栈，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。

比较器 0 中断是由比较器 0 输出端产生一个下降沿触发的，其中断请求标志位 (COF; INTC0 的第 4 位) 会被置位。如果中断允许，且堆栈未满，当发生比较器 0 中断时，会产生地址 04H 的子程序调用；而中断请求标志 (COF) 和总中断控制位 EMI 会被清除，以禁止其它中断响应。

比较器 1 中断是由比较器 1 输出端产生一个下降沿触发的，其中断请求标志位 (C1F; INTC0 的第 5 位) 会被置位。如果中断允许，且堆栈未满，当发生比较器 1 中断时，会产生地址 08H 的子程序调用；而中断请求标志 (C1F) 和总中断控制位 EMI 会被清除，以禁止其它中断响应。

外部中断是由 PC1 端口上产生一个下降沿触发的，其中断请求标志位 (EIF; INTC0 的第 6 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 0CH 的子程序调用；而中断请求标志 (EIF) 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 0 中断是由定时/计数器 0 溢出触发的，其中断请求标志 (TOF; INTC1 的第 4 位) 会被置位。如果中断允许，且堆栈未满，当发生定时/计数器 0 中断时，会产生地址 10H 的子程序调用；而中断请求标志 (TOF) 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 1 中断是由定时/计数器 1 溢出触发的，其中断请求标志 (T1F; INTC1 的第 5 位) 会被置位。如果中断允许，且堆栈未满，当发生定时/计数器 1 中断时，会产生地址 014H 的子程序调用；而中断请求标志 (T1F) 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 6 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 018H 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。

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，此时堆栈未满）。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。

位	符号	功 能
0	EMI	总中断控制位 (1=允许; 0=禁止)
1	EC0I	比较器 0 控制位 (1=允许; 0=禁止)
2	EC1I	比较器 1 控制位 (1=允许; 0=禁止)
3	E EI	外部中断控制位 (1=允许; 0=禁止)
4	COF	比较器 0 请求标志 (1=有; 0=无)
5	C1F	比较器 1 请求标志 (1=有; 0=无)
6	EIF	外部中断请求标志 (1=有; 0=无)
7	—	保留位，读取为“0”

INTC0(0BH) 寄存器

位	符号	功能
0	ET0I	定时/计数器 0 中断控制位 (1=允许; 0=禁止)
1	ET1I	定时/计数器 1 控制位 (1=允许; 0=禁止)
2	EADI	A/D 转换中断 1 控制位 (1=允许; 0=禁止)
3	—	未定义
4	T0F	定时/计数器 0 中断请求标志 (1=有; 0=无)
5	T1F	定时/计数器 1 请求标志 (1=有; 0=无)
6	ADF	A/D 转换中断请求标志 (1=有; 0=无)
7	—	未定义

INTC1(1EH) 寄存器

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。

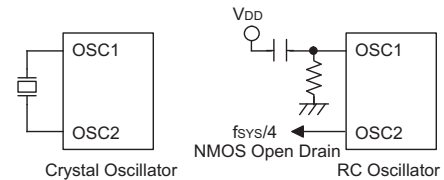
中断源	优先级	中断向量
比较器 0 中断	1	004H
比较器 1 中断	2	008H
外部中断-来自 PC1	3	00CH
定时/计数器 0 中断	4	010H
定时/计数器 1 中断	5	014H
A/D 转换中断	6	018H

EMI、EC0I、EC1I、EEI、ET0I、ET1I、和 EADI 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(C0F、C1F、EIF、T0F、T1F、ADF)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

HT46R12A 有两种振荡方式，外部 RC 振荡和外部晶体振荡，可以通过掩膜选项设定，不管选用哪一种振荡方式，其信号都可以做为系统时钟。HALT 模式会停止系统振荡器，并忽视任何外部信号以降低功耗。



系统振荡器

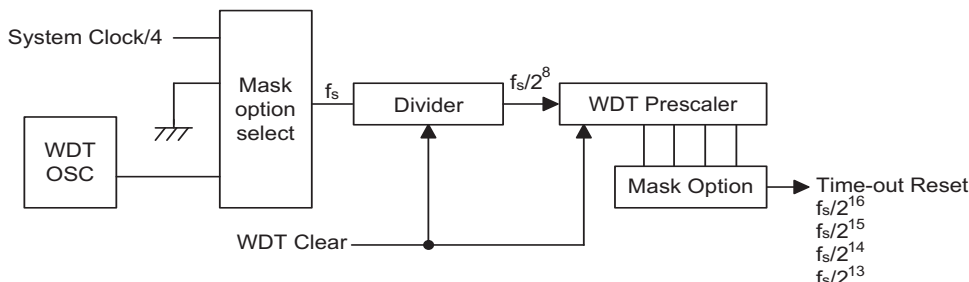
如果选用外部 RC 振荡方式，在 OSC1 与 VSS 之间需要接一个外部电阻，其阻值为 24kΩ 到 1MΩ；而 OSC2 上会输出带上拉的系统频率的 4 分频信号，可用于同步外部逻辑。RC 振荡方式是一种低成本方案，但是，RC 振荡频率会随着 VDD、温度和芯片自身参数的漂移而产生误差。因此，在需要精确振荡频率做为计时操作的场合，并不适合使用 RC 振荡方式。

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(如果振荡频率小于 1MHz)。当系统进入省电模式后，系统振荡器将会停振以降低功耗。

WDT 振荡器是一个内部 RC 振荡器，并不需要连接任何外部元件。当系统进入暂停模式时，系统时钟会停止，但 WDT 振荡器会继续工作，其振荡周期大约为 65μs/5V。如果要降低功耗，可在掩膜选项中关闭 WDT 振荡器。

看门狗定时器

看门狗定时器的时钟来源有两种：专门的内部 RC 振荡(看门狗振荡器)或指令时钟(系统时钟 4 分频)，由掩膜选项设置。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。



看门狗定时器

为得到一个实用的 WDT 溢出周期，WDT 频率(通过掩膜选项：WDT time out)可设置 $2^{13} \sim 2^{16}$ 的分频系数。如果 WDT 时钟源为内部 WDT 振荡，最小的 WDT 溢出周期大约是 600mS。溢出时间会因为温度、VDD 以及芯片参数的变化而变化。如果再用 WDT 预分频器，则可以得到更长的溢出周期。如果 WDT 的溢出时间(time out)选为 2^{16} ，最大的溢出时间可达到 4.7s。

WDT 时钟源除了使用内部 WDT 振荡器输出外，还可以使用指令时钟(系统时钟 4 分频)，不同点只是在 HALT 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 PC 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。

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，但如果选择 WDT 振荡器，WDT 振荡器会继续振荡。
- 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 PC 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。如果是由输入/输出口唤醒，程序会从下一条指令开始运行。如果是由中断唤醒，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 $1024t_{SYS}$ (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。

复位

总共有三种方法会产生初始复位：

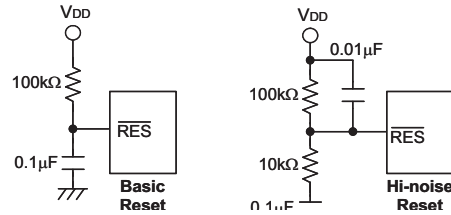
- 正常运行时由RES引脚发生复位。
- 在暂停模式下由RES引脚发生复位。
- 正常运行时由看门狗定时器溢出发生复位。

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PC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

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

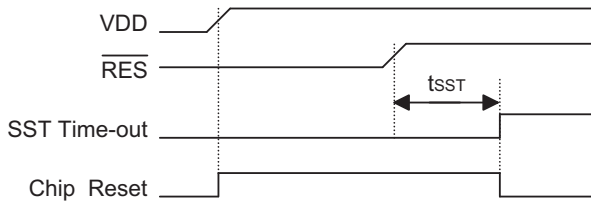
TO	PDF	复位原因
0	0	上电时RES发生复位
u	u	正常运行时RES发生复位
0	1	暂停模式下RES发生复位
1	u	正常运行时WDT溢出
1	1	暂停模式下WDT溢出

注：“u”表示不变

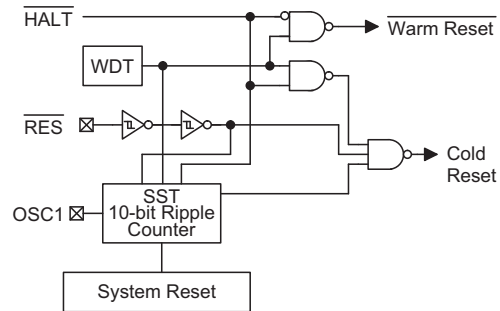
为了保证系统振荡器起振并稳定运行，系统复位(包括上电复位、WDT溢出或由RES端复位)或由暂停状态唤醒时，系统启动定时器(SST)提供了一个额外的延迟时间，共1024个系统时钟周期。

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RES端复位)需要额外增加一个加载掩膜选项(Option)的时间。



复位时序



复位电路结构

系统复位时各功能单元的状态如下所示：

PC	000H
中断	禁止
预分频、除频	清除
WDT	清除，在主系统复位后，WDT开始计数
定时/计数器	停止
PPG计数器	停止
PPG输出	浮态
输入/输出口	输入模式
堆栈指针 SP	指向堆栈顶部

有关寄存器的状态如下：

寄存器	复位 (上电复位)	WDT 溢出 (正常运行)	RES复位 (正常运行)	RES复位 (暂停模式)	WDT 溢出 (暂停模式)*
MP0	1xxx xxxx	1uuu uuuu	1uuu uuuu	1uuu uuuu	1uuu uuuu
MP1	1xxx xxxx	1uuu uuuu	1uuu uuuu	1uuu uuuu	1uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PC	000H	000H	000H	000H	000H
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR1	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR1C	00-0 1---	00-0 1---	00-0 1---	00-0 1---	uu-u u---
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	---- 1111	---- 1111	---- 1111	---- 1111	---- uuuu
PBC	---- 1111	---- 1111	---- 1111	---- 1111	---- uuuu
PC	---1 1111	---1 1111	---1 1111	---1 1111	---u uuuu
PCC	---1 1111	---1 1111	---1 1111	---1 1111	---u uuuu
INTC1	-000 -000	-000 -000	-000 -000	-000 -000	-uuu uuuu
PPG0C	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PPGT0	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
CMP0C	-000 1000	-000 1000	-000 1000	-000 1000	-uuu uuuu
CMP1C	-000 1000	-000 1000	-000 1000	-000 1000	-uuu uuuu
ADRL	x--- ----	x--- ----	x--- ----	x--- ----	u--- ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	---- -00	---- -00	---- -00	---- -00	---- -uu

- 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

HT46R12A 有二个定时/计数器 (TMR0, TMR1)。定时/计数器 0 是一个 8 位可编程向上计数器, 它的时钟来源可以是外部信号输入或内部系统时钟, 选内部时钟时它的时钟来源就是 f_{SYS} 。定时/计数器 1 也是一个 8 位可编程向上计数器, 它的时钟来源可以是外部信号输入或内部系统时钟, 选内部时钟时它的时钟来源就是 f_{SYS} 的 4 分频。外部时钟输入允许用户去计算外部事件, 测量时间间隔或脉宽、或产生一个精确的时基信号; 而使用内部时钟的话, 则允许用户去产生一个精确的时基信号。

如果用做内部定时器方式, 即系统时钟来自于 f_{SYS} , 这时计数器仅做为一个参考时基。当计数器的时钟信号来自外部信号输入时则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。

有四个与定时/计数器有关的寄存器, TMR0(0DH)、TMR0C(0EH)、TMR1(10H)和 TMR1C(11H)。写入 TMR0/TMR1 会将初始值装入到 TMR0/TMR1 的相应预置寄存器中, 而读 TMR0/TMR1 则会获得定时/计数器的内容。TMR0C 和 TMR1C 是定时/计数器控制寄存器, 其定义某些选项。

T0M0 和 T0M1 (T1M0 和 T1M1) 位, 定义工作模式。外部事件计数模式用来记录外部事件, 时钟来源由外部 (TMR0, TMR1) 引脚输入。定时器模式是作为一个普通的定时器功能, 时钟来源为内部时钟。脉冲宽度测量模式能用来测量外部引脚 (TMR0, TMR1) 上的高电平或低电平的宽度, 时钟来源是被选中的内部时钟。

在外部事件计数或定时器模式中, 定时/计数器 0/1 会从当前定时/计数器中的数值开始向上计数, 到 0FFH 结束。如果产生溢出, 计数器会从定时/计数器预置寄存器重新装载计数值并且同时产生相应的中断请求标志 (T0F; INTC1 第 4 位, T1F; INTC1 的第 5 位)。

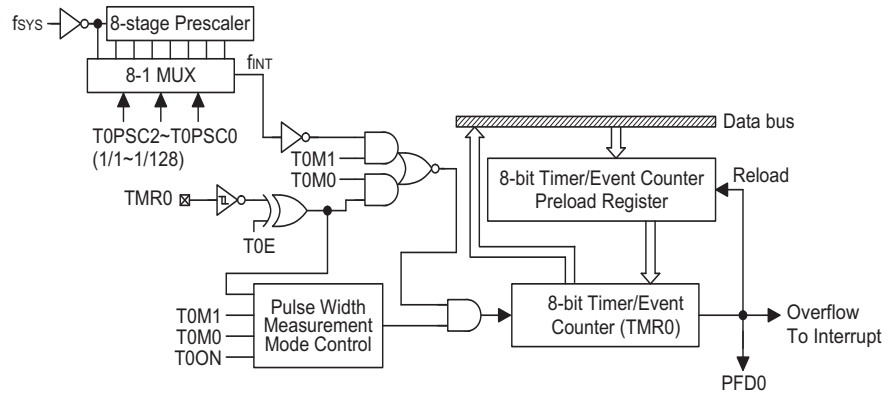
在脉冲宽度测量模式时, 其 T0ON/T1ON 和 T0E/T1E 位皆为 1 时, 如果引脚 TMR0/TMR1 接收到一个上升沿信号 (如 T0E/T1E 值为 0, 则为下降沿信号) 时, 计数器会开始计数直至 TMR0/TMR1 引脚回到原来的电平为止, 并且会将 T0ON/T1ON 清零, 只有这种模式 T0ON/T1ON 会自动清零, 其它模式 T0ON/T1ON 位只可以用指令清除。计数器停止计数, 测量的结果则保留在定时/计数器之中, 而且即使再收到一个跳变信号也不会改变。换句话说, 脉冲宽度测量模式一次只能测量一个脉冲。只要 T0ON/T1ON 位又被置位, 则当引脚 TMR0/TMR1 接到跳变脉冲, 测量周期会再次执行下去。在脉冲宽度测量模式中, 定时/计数器并不会根据逻辑电平来计数, 其根据的标准为信号的跳变沿。一旦发生计数器溢出, 计数器会从定时/计数器加载寄存器重新装入, 同时还会发出中断请求, 这个情况和外部事件计数模式和定时器模式一样。

要启动计数运作, 只要将定时器启动 ON 位 (T0ON; TMR0C 的第 4 位; T1ON; TMR1C 的第四位) 被置成为 1。在脉宽测量模式中 T0ON/T1ON 在测量周期结束后自动被清除。但在另外两个模式中, T0ON/T1ON 只能由指令来复位。定时/计数器的溢出是唤醒的信号源之一。并且可由掩膜选项, 设定 PA3 用作为 PFD 输出 (可编程分频器)。掩膜只有一个 PFD 信号 (PFD0 或 PFD1) 提供给 PA3。不管处于何种模式, 若写 0 到 ET0I 或 ET1I 位即可禁止相应的中断服务。当 PFD 功能被选时, 执行 “CLR [PA].3” 指令来禁止 PFD 输出和执行 “SET [PA].3” 指令来使能 PFD 输出。

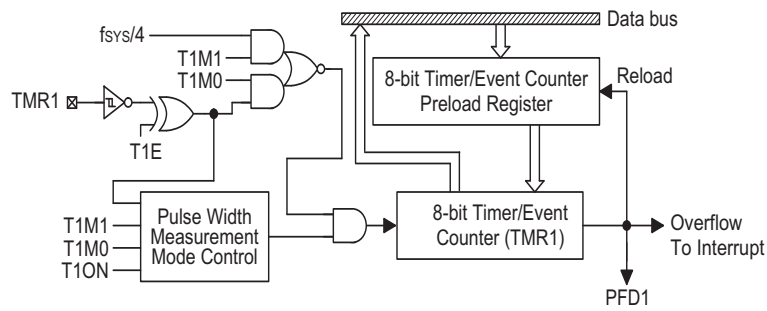
在定时/计数器为关闭的状态下, 写数据到定时/计数器的预置寄存器之中, 同时也会将数据装入定时/计数器中。但是若定时/计数器出于工作状态, 写到定时/计数器的数据只会被保留在定时/计数器的预置寄存器中, 直到定时/计数器发生计数溢出才会将数据从预置寄存器加载到定时/计数器寄存器中。如果要读取定时/计数器数据 (读 TMR0/TMR1), 计数会被停止。计数停止会导致计数错误, 所以程序员必须注意到这一点。

强烈建议首先装载一个指定的值到 TMR0/TMR1 寄存器, 然后启动定时/计数器作正常的运作。因为 TMR0/TMR1 的初始值是不确定的。鉴于定时/计数器的配置, 在任何要使用定时/计数器的时候, 为了避免不可预料的结果, 第一次开启然后关闭定时/计数器, 用户都必须特别注意。在这以后定时/计数器开始正常工作。

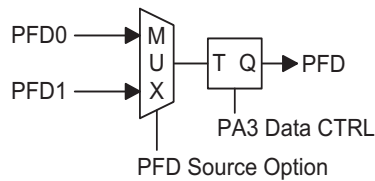
TMR0C 的第 0~2 位用来定义内部时钟预分频级数, 定义如下表所示。定时/计数器的溢出信号可做为 PFD 输出。



定时/计数器 0



定时/计数器 1



PFD 信号源选择

位	符号 (TMRC)	功能
0 1 2	TOPSC0 TOPSC1 TOPSC2	定义预分频器级数, TOPSC0、TOPSC1、TOPSC2= 000: $f_{INT}=f_{SYS}$ 001: $f_{INT}=f_{SYS}/2$ 010: $f_{INT}=f_{SYS}/4$ 011: $f_{INT}=f_{SYS}/8$ 100: $f_{INT}=f_{SYS}/16$ 101: $f_{INT}=f_{SYS}/32$ 110: $f_{INT}=f_{SYS}/64$ 111: $f_{INT}=f_{SYS}/128$
3	T0E	定义定时/计数器 TMR0 的触发方式 在事件计数模式 (T0M1, T0M0) = (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 (T0M1, T0M0) = (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
4	T0ON	打开/关闭定时/计数器(1=打开, 0=关闭)
5	—	未用, 读出为“0”
6 7	T0M0 T0M1	定义工作模式(T0M1,T0M0): 01=事件计数模式(外部时钟) 10=定时模式(内部时钟) 11=脉冲宽度测量模式(外部时钟) 00 =未用

TMR0C(0EH) 寄存器

位	符号 (TMRC)	功能
0~2	—	未用, 读出为“0”
3	T1E	定义定时/计数器 TMR1 的触发方式 在事件计数模式 (T1M1, T1M0) = (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 (T1M1, T1M0) = (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
4	T1ON	打开/关闭定时/计数器(1=打开, 0=关闭)
5	—	未用, 读出为“0”
6 7	T1M0 T1M1	定义工作模式(T1M1,T1M0): 01=事件计数模式(外部时钟) 10=定时模式(内部时钟) 11=脉冲宽度测量模式(外部时钟) 00 =未用

TMR1C(11H) 寄存器

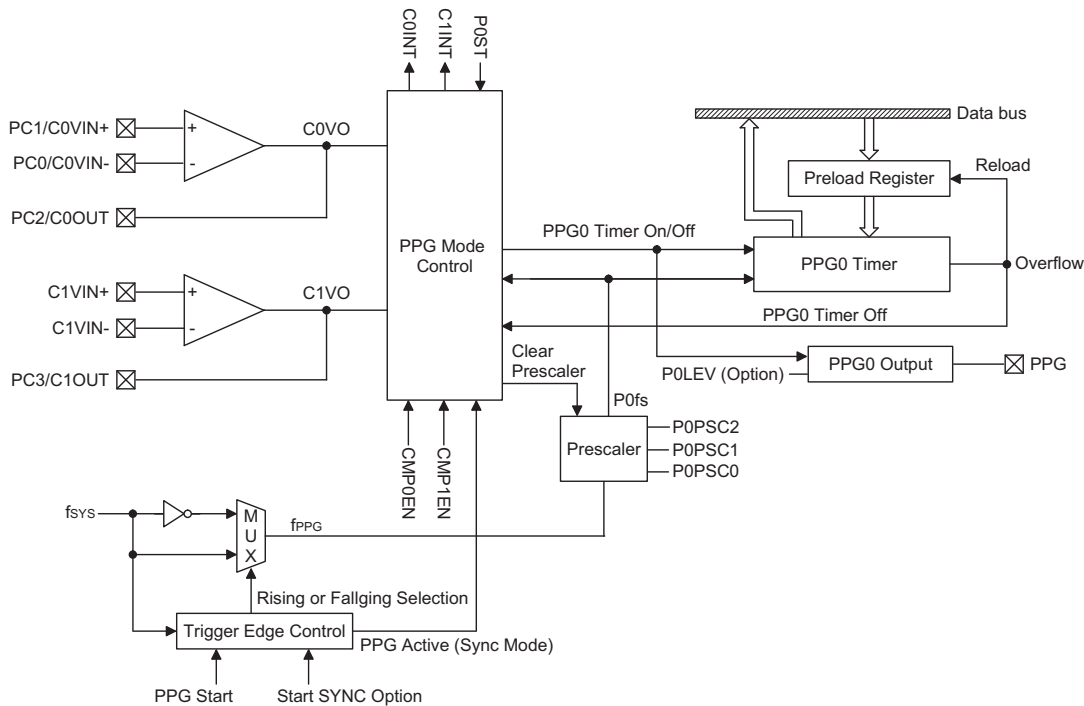
可编程脉冲发生器—PPG

HT46R12A 提供一组 8 位的 PPG 输出通道。PPG 的可编程周期范围是 $256 \times T$ ，决定 PPG 输出脉冲宽度的 T 可能是： $1/f_{SYS}$ 、 $2/f_{SYS}$ 、 $4/f_{SYS}$ 、 $8/f_{SYS}$ 、 $16/f_{SYS}$ 、 $32/f_{SYS}$ 、 $64/f_{SYS}$ 、 $128/f_{SYS}$ 。当 PPG 模块有一个下降沿信号输入，PPG 就会输出一个脉冲。它的触发信号可以由软件指令决定来自于比较器或是软件指令位。在系统频率为 4M 的情况下 PPG 的输出脉宽范围为 0.25us~8.192ms。可以通过设置极性控制位 (P0LEV) 来决定 PPG 输出有效脉冲是高电平或是低电平 (腌模选项设置)。当寄存器 PPGT0 内写入值为 00H 的时候相应 PPG 的输出脉宽为 $256 \times T$ 。

• PPG0 功能描述

PPG0 模块是由 PPG0 计数器、PPG 模块控制器及两组比较器三部分组成。其中 PPG0 计数器又是由一个预分频器、一个 8 位向上计数器及一个 8 位预载寄存器组成。可编程脉冲发生器 (PPG) 一启动，相应计数器就从预载寄存器中的值开始向上计数至到内部数据由 "FFH→00H" 结束计数，一旦计数器溢出，预载寄存器里的值将会自动装载到计数器中，同时会产生一个信号去停止 PPG 的输出。当 PPG0 计数器溢出的同时软件触发位 (POST) 也将会被清 0。

与 PPG 功能相关的寄存器有 2 个分别是：控制寄存器 PPG0C、计数器预载寄存器 PPGT0。控制寄存器中 PPG0C 定义 PPG0 的输入控制模式 (触发信号) 包括：比较器是否使能、计数器的预分频系数 (范围是： $1/f_{SYS}$ 、 $2/f_{SYS}$ 、 $4/f_{SYS}$ 、 $8/f_{SYS}$ 、 $16/f_{SYS}$ 、 $32/f_{SYS}$ 、 $64/f_{SYS}$ 、 $128/f_{SYS}$)、停止 PPG 输出的信号是否来自 C0VO、启动 PPG 计数器的信号是否来自 C1VO、控制 PPG0 的软指令触发位状态以控制相应计数器的启动与停止。PPGT0 是 PPG0 计数器的预载寄存器，这个寄存器里的值决定了输出脉冲的宽度。



PPG 方框图

• PPG0C 控制寄存器

位	7	6	5	4	3	2	1	0
PPG0C (20H)	POST	PORSEN	POSPEN	POPSC2	POPSC1	POPSC0	CMP1EN	CMP0EN
POR value	0	0	0	0	0	0	0	0

CMP0EN: 使能或不使能比较器 0 输出 (0=不使能, 1=使能)

CMP1EN: 使能或不使能比较器 1 输出 (0=不使能, 1=使能)

POPSC2, POPSC1, POPSC0: 选择 PPG0 计数器的预分频系数

POSPEN: 使能或不使能停止 PPG0 输出的信号来自于 C0VO 触发 (0=不使能, 1=使能)

PORSEN: 使能或不使能启动 PPG0 输出的信号来自于 C1VO 触发 (0=不使能, 1=使能)

POST: PPG0 的软指令触发位 (0=停止 PPG0 输出, 1=启动 PPG0 输出)

CMP0EN 与 CMP1EN 做为控制位来控制比较器使能与否:

当 CMP0EN 被置“0”时, 比较器 0 不工作, PC0/C0VIN-, PC1/C0VIN+, PC2/C0OUT 都是做为普通 I/O 口功能 (GPIO); 如果 CMP0EN 被置“1”, 比较器 0 功能选中, PC0/C0VIN-, PC1/C0VIN+, PC2/C0OUT 仍可做为输入口。

如果 CMP1EN 被置“0”时, 比较器 1 不使能, PC3/C1OUT 都是做为普通 I/O 口功能 (GPIO), 如果 CMP1EN 被置“1”, 比较器 1 功能选中, PC3 口仍可做为输入口来用。

任何停止 PPG 输出的动作诸如 PPG 计数器溢出、软件指令停止 (POST 由 1 变成 0) 均可导致以下情况的发生:

停止并清除 PPG 预分频 (预分频指的是其相应的计数器, 不是 PPG0C 中的 POPSC[2: 0]设置)

PPG 计数器数据重新载入

POST 会被清 0

PPG0 截止

PPG0 的启动延时 $\leq 0.5 \times (1/f_{sys})$, 当在掩膜选项中选择 SYNC with clock 时, 下一个系统周期的上升沿或下降沿均可触发 PPG 脉冲的输出。PPG 功能启动之后, PPG 输出使能, 一旦第一个边沿 (系统频率的上升或下降沿) 来到, 它的预分频寄存器便开始计数。第一个触发完成之后, 接下来的时钟沿即跟随保持如: 一旦 PPG 的第一个启动是由一个下降沿触发, 那么接下来的 PPG 都是由下降沿触发至到 PPG 停止输出, 反之亦然。

图 1: 在 PPG 启动之后第一个触发信号是下降沿, PPG 计数器此后都是下降沿触发至到 PPG 截止。

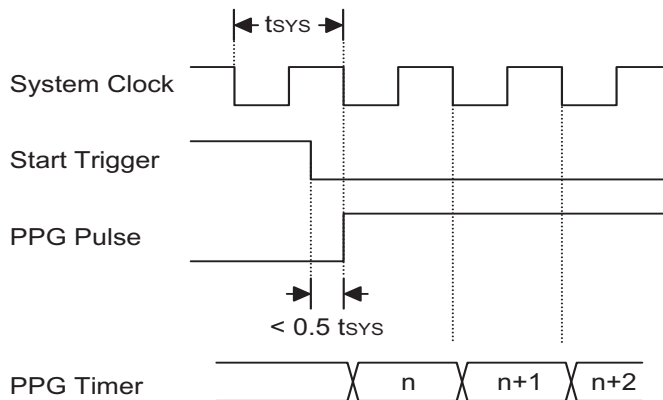
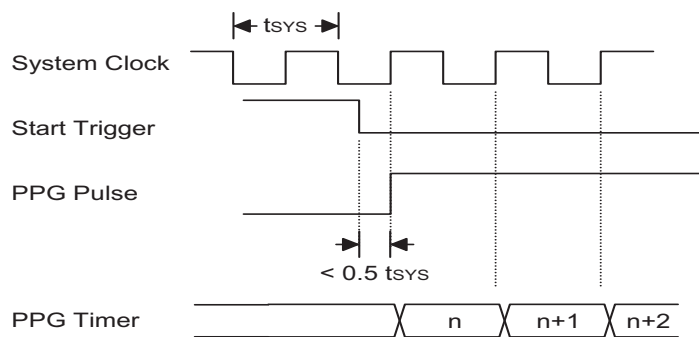


图 2: 在 PPG 启动之后第一个触发信号是上升沿, PPG 计数器此后都是上升沿触发至到 PPG 截止。



PPG0C: CMP0EN、CMP1EN 为比较器的使能控制位

CMP0EN	描述
0	比较器 0 不使能。PC0/C0VIN-, PC1/C0VIN+, PC2/C0OUT 为普通 I/O 口 (GPIO)
1	比较器 0 使能。PC0/C0VIN-, PC1/C0VIN+ 是比较器 0 的输入口, PC2/C0OUT 是比较器 0 的输出口, PC2 的输出功能除能, 上拉电阻失效。

CMP1EN	描述
0	比较器 1 不使能。PC3/C1OUT 为普通 I/O 口 (GPIO)
1	比较器 1 使能。PC3/C1OUT 是比较器 1 的输出口。PC3 的输出功能除能, 上拉电阻失效。

PPG0 控制寄存器 PPG0C 的第 4~2 位是用来选取 PPG0 计数器时钟的预分频系数。

PPG0C: PPG0 计数器预分频位

P0PSC2	P0PSC1	P0PSC0	定义分频系数
0	0	0	$P0fs=f_{SYS}$
0	0	1	$P0fs=f_{SYS}/2$
0	1	0	$P0fs=f_{SYS}/4$
0	1	1	$P0fs=f_{SYS}/8$
1	0	0	$P0fs=f_{SYS}/16$
1	0	1	$P0fs=f_{SYS}/32$
1	1	0	$P0fs=f_{SYS}/64$
1	1	1	$P0fs=f_{SYS}/128$

P0SPEN 是决定 PPG0 的停止信号源是否是来自 C0VO 输出的下降沿信号, 如果是, 那么 PC2 或 C0VO 一旦有下降沿产生 PPG0 输出就会被停止。P0RSEN 是决定 PPG0 的启动信号是否来自 C1VO 输出的下降沿触发, 如果是那么一旦 C1VO 或 PC3 口有一个下降沿产生 PPG0 计数器就会开始计数。在比较器 0 或比较器 1 使能的情况下, 使用者可能通过将 PC2 或 PC3 设置为输入状态来读取 C0VO 或 C1VO 的状态。

P0SPEN	描述
0	停止 PPG0 计数器的信号不采用 C0VO 触发。 PPG0 计数器的停止仅由软件指令位 (POST) 来控制。
1	停止 PPG0 计数器的信号来自 C0VO 下降沿。 PPG0 计数器的停止信号来自 C0VO 的下降沿或是软指令控制位 (POST=0)

PORSEN	描述
0	启动 PPG0 计数器的信号不采用 C1VO 触发。 PPG0 模块只能软件指令位触发启动。
1	启动 PPG0 计数器的信号来自于 C1VO 触发。 PPG0 模块的启动信号来自于 C1VO 输出的下降沿或是软指令控制位(POST=1)

POST 是一个软指令控制位，如果这个位被置“1”，PPG0 计数器会立即开始计数，当计数器溢出，这位会自动被清 0 同时 PPG0 计数器停止计数。如果这位被置“0”，PPG0 计数器会立即停止计数。在 PPG0 计数过程中，如果 C1VO、PC3 口有下降沿产生或是 POST 被置“1”，PPG0 计数器将不受其影响，也就是说此时来自 C1VO、PC3、POST 的再启动信号无效。POST 也可以做为 PPG0 计数器的一个状态标志位使用。

PPG0 模组输出的有效电平是由掩膜选项来定，如果 POLEV 被置“0”，则 PPG 输出有效电平为高电平，反之为低电平。

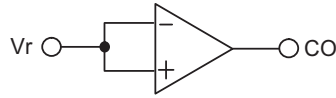
另外一个功能，即 PPG 计数器是否与系统时钟同步的问题，这也是可以由掩膜选项来设置。

启动 PPG 功能的操作：

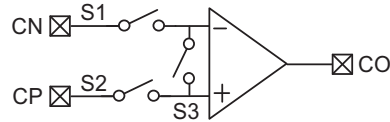
- 设置 PPG0 输出有效电平—由掩膜选项来定。
- 设置 PPG0 功能的控制信号来源—PORSEN, POSPEN
- 决定 PPG0 输出脉冲宽度—写数据到 PPGT0 及设置好 PPG0 的预除频 (POPSC2: POPSC0)
- 确定 PPG0 计数器时钟是否与系统时钟同步 (由掩膜选项 POTSYN 位来定)
- 当 PPG0 由 C1VO 的下降沿或由软件指令 (POST 设置为 1) 触发时，PPG0 将会在预载寄存器内数据的基础上开始计数。当 PPG0 收到一个来自 C0VO 下降沿信号或软件指令 (POST 设置为 0) 或 PPG0 计数器计数溢出时，PPG0 输出将被截止。

比较器

PPG 比较器的输入偏置电压可以通过调整共模输入来得到调整。



调整步骤如下：



- 置 CnCOFM=1，选择偏置电压补偿模式 - 关闭开关 S3。
- 置位 CnCRS 位，选择参考电压输入引脚 - 关闭开关 S1 或 S2。
- 调整位 CnCOF0~CnCOF3 直到输出状态发生变化。
- 置 CnCOFM=0，选择正常操作模式。

位	符号	功能	POR
0	C0COF0	比较器输入偏置电压补偿控制位	1000B
1	C0COF1		
2	C0COF2		
3	C0COF3		
4	C0CRS	比较器输入偏置电压补偿参考选择位 1/0: 选择 CP/CN 作为参考输入	0
5	C0COFM	输入偏置电压补偿模式和比较器模式选择 1: 输入偏置电压补偿模式 0: 比较器	0
6	C0CMPOP	比较器输出; 正逻辑	0
7	—	未使用位, 读取为“0”	0

CMP0C (1BH) 寄存器

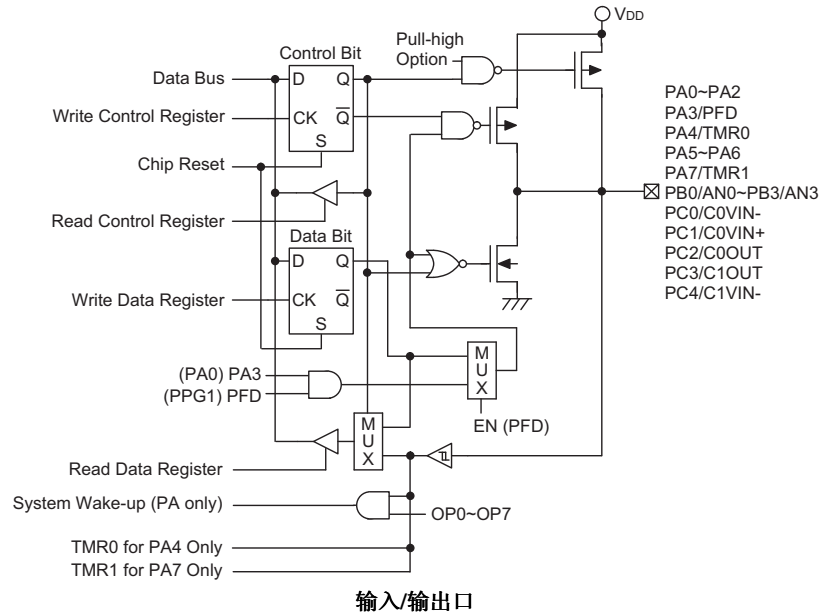
位	符号	功能	POR
0	C1COF0	比较器输入偏置电压补偿控制位	1000B
1	C1COF1		
2	C1COF2		
3	C1COF3		
4	C1CRS	比较器输入偏置电压补偿参考选择位 1/0: 选择 CP/CN 作为参考输入	0
5	C1COFM	输入偏置电压补偿模式和比较器模式选择 1: 输入偏置电压补偿模式 0: 比较器	0
6	C1CMPOP	比较器输出; 正逻辑	0
7	—	未使用位, 读取为“0”	0

CMP1C (1CH) 寄存器

输入/输出口

HT46R12A 有 16 个双向输入/输出口，记为 PA、PB、PC，其分别对应 RAM 地址[12H]、[14H]和[16H]，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，也就是说，输入信号必须在 MOV A, [m](m=12H、14H、16H)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。

每个输入/输出口都有一个控制寄存器(PAC, PBC, PCC)，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。



输入/输出口

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。

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/输出锁定位都能用“SET [m].i”或“CLR [m].i”指令置位或清除 (m=12H、14H、16H)。

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。所有的输入/输出口都有上拉电阻选项。一旦选择了上拉电阻选项，输入/输出口就加了上拉电阻。如果不选择上拉电阻，必须在输入模式下，输入/输出口会产生浮空状态。

PA3、PA4 和 PA7 分别与 PFD、TMR0 和 TMR1 共用引脚。PC0、PC1、PC2、PC3 及 PC4 分别与 COVIN-、COVIN+、COOUT、C1OUT 及 C1VIN-共用引脚。

PA3 与 PFD 共用引脚，如果选择 PFD 功能，则 PA3 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，而在输入模式始终保持其原来的功能。一旦选择 PFD 功能，PFD 的输出信号只受 PA3 数据寄存器控制。向 PA3 数据寄存器写入“1”，则输出 PFD 信号；向 PA3 数据寄存器写入“0”，则 PA3 输出为“0”。PA3 的输入/输出功能如下所示：

I/O 模式	I/P (正常)	O/P (正常)	I/P (PFD)	O/P (PFD)
PA3	逻辑输入	逻辑输出	逻辑输入	PFD (定时/计数器开启)

注：PFD 的输出频率是定时/计数器溢出频率的 1/2.

建议用软件将未使用 and 没有外接的输入/输出口设置为输出模式，以防止这些端口在输入浮空时增加系统的功耗。

PFD (PFD0 或 PFD1) 输出与 PA3 共享引脚，一旦选择 PFD 功能，向 PA3 数据寄存器写入“1” (“SET PA.3”) 将使能 PFD 输出；向 PA3 数据寄存器写入“0” (“CLR PA.3”) 将停止 PFD 输出，且 PA3 输出为“0”。

PFD 的控制信号和输出频率如下所示：

定时/计数器	定时/计数器预置值	PA3 数据寄存器	PA3 引脚状态	PFD 输出频率
关闭	X	0	0	X
关闭	X	1	U	X
开启	N	0	0	X
开启	N	1	PFD	$f_{TMR}/[2 \times (M-N)]$

注：“X”表示未定义

“U”表示未知

“M” PFD0 时等于“256”，PFD1 时等于“65536”

“N”定时/计数器初始值

“ f_{TMR} ”定时/计数器输入频率

A/D 转换

HT46R12A 有 4 个通道、9 位解析度(8 位精度)的 A/D 转换器。其参考电压为 VDD。与 A/D 转换有关的寄存器有 4 个: ADRL(24H)、ADRH(25H)、ADCR(26H)和 ACSR(27H)。ADRH 和 ADRL 是 A/D 转换结果的高字节和低字节寄存器,是只读寄存器。当完成 A/D 转换后,可从 ADRH 和 ADRL 读取 A/D 转换结果。ADCR 是 A/D 转换控制寄存器,用来定义 A/D 通道数量、模拟输入通道选择、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,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 时钟控制寄存器,用来选择 A/D 的时钟来源。

符号(ACSR)	位	功能
ADCS0 ADCS1	0 1	选择 A/D 转换时钟源: 00=系统时钟/2 01=系统时钟/8 10=系统时钟/32 11=未定义
—	2~6	未用, 读出为“0”
TEST	7	只做为内部测试用

ACSR(27H) 寄存器

A/D 转换控制寄存器用来控制 A/D 转换。ADCR 的第 2~0 位用来选择模拟输入通道,总共有 4 个通道可以选择。ADCR 的第 5~3 位用来设置 PB 的工作模式, PB 可以做为模拟输入通道,或是数字输入/输出口,由这 3 位来决定。

PCR2	PCR1	PCR0	3	2	1	0
0	0	0	PB3	PB2	PB1	PB0
0	0	1	PB3	PB2	PB1	AN0
0	1	0	PB3	PB2	AN1	AN0
0	1	1	PB3	AN2	AN1	AN0
1	x	x	AN3	AN2	AN1	AN0

PB 口配置

如果 PB 选择为模拟输入,则其输入/输出功能和上拉电阻将失效,而 A/D 转换电路会被使能。EOCB 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 START 位用来开启 A/D 转换,给 START 位一个上升沿信号和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 EOCB 位变为“0”(A/D 转换完成信号)。

ACSR 的第 7 位是内部测试用的,用户不能使用。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 中断请求标志被置位。当 START 标志由“0”置为“1”时,EOCB 也置为“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要注意初始化 A/D 转换器,否则 EOCB 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。

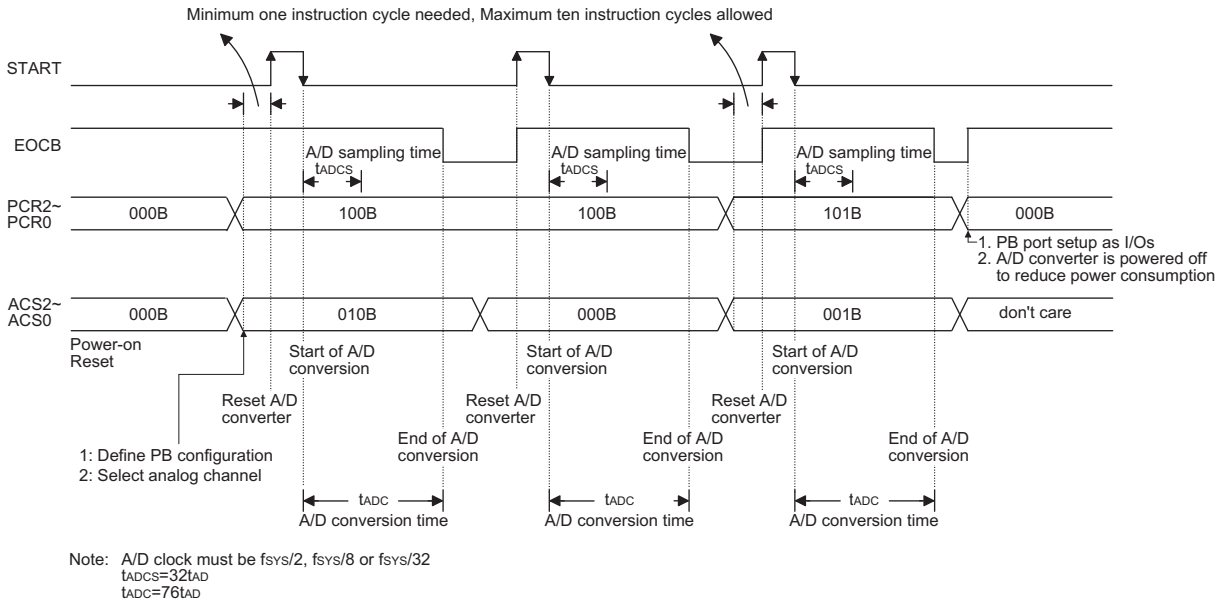
符号(ADCR)	位	功能
ACS0 ACS1 ACS2	0 1 2	选择模拟输入通道 0, 0, 0: AN0 0, 0, 1: AN1 0, 1, 0: AN2 0, 1, 1: AN3 1,x, x: 未定义, 不能使用
PCR0 PCR1 PCR2	3 4 5	定义 PB 口的设置 如果 PCR0、PCR1 和 PCR2 都为 0, 则 A/D 转换电路被关闭以减小功耗
EOCB	6	A/D 转换结束标志(0: A/D 转换结束) 每次 BIT3-5 状态的改变都必须通过 START 信号来初始化 A/D 转换器, 否则 EOCB 可能会处于不确定状态, 具体可参照“A/D 转换初始化注意事项”
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 EOCB 为“1”

ADCR(26H) 寄存器

寄存器	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRL(24H)	D0	—	—	—	—	—	—	—
ADRH(25H)	D8	D7	D6	D5	D4	D3	D2	D1

注: D0~D8 是 A/D 转换结果的低位~高位

ADRL(24H) 寄存器 ADRH(25H)寄存器



A/D 转换时序

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 EOCB 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 EOCB 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     EOCB          ; 扫描ADCR寄存器的EOCB位判断A/D转换是否完成
jmp    polling_EOC  ; 继续扫描
mov    a,ADRH        ; 从ADRH寄存器读取A/D转换结果的高位字节
mov    adrh_buffer,a ; 将结果放入用户定义的寄存器中
mov    a,ADRL        ; 从ADRL寄存器读取A/D转换结果的低位字节
mov    adrl_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   ; 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
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,ADRH        ; 从ADRH寄存器读取A/D转换结果的高位字节
mov    adrh_buffer,a ; 将结果放入用户定义的寄存器中
mov    a,ADRL        ; 从ADRL寄存器读取A/D转换结果的低位字节
mov    adrl_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

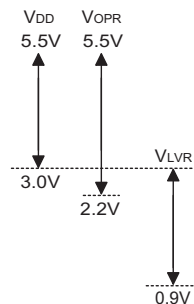
```

低电压复位—LVR

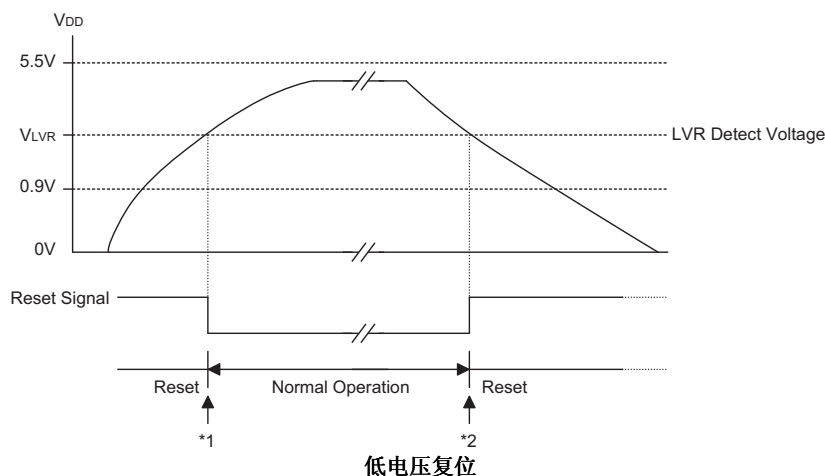
为了监控器件的工作电压，HT46R12A 提供低电压复位功能。如果器件的工作电压在 $0.9V \sim V_{LVR}$ 之间，例如电池电压的变化，那么 LVR 会自动使器件产生内部复位。

LVR 功能说明如下：

- 低电压($0.9V \sim V_{LVR}$)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：



注： V_{OPR} 是在系统时钟为 4MHz 时，使得芯片正常运行的电压值



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。

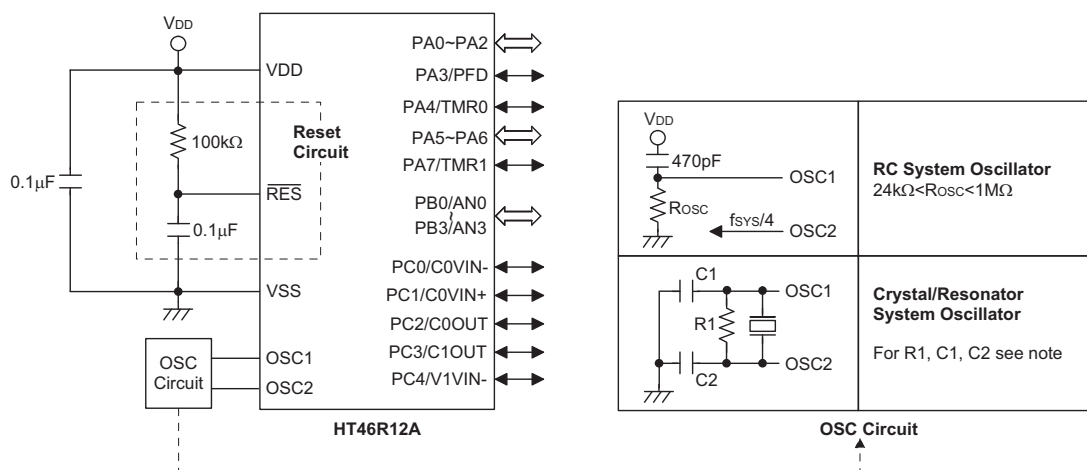
*2：因为低电压状态必须保持 1ms 以上，因此进入复位模式就要有 1ms 的延迟。

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。

选项
振荡类型选择。 该选项用来定义系统时钟来源为 RC 振荡还是晶体振荡。
WDT 时钟源选择。 内部 WDT 振荡 RC、 $f_{SYS}/4$ 或是不使能 WDT 功能
清除 WDT 指令选择。 该选项用来定义清除 WDT 的指令条数。“1 条指令”表示“CLR WDT”就能清除 WDT；“2 条指令”表示要同时使用“CLR WDT1”和“CLR WDT2”才能清除 WDT。
WDT 预分频选择。 有四种分频选择： $f_{SYS}/2^{13}$ 、 $f_{SYS}/2^{14}$ 、 $f_{SYS}/2^{15}$ 、 $f_{SYS}/2^{16}$
唤醒选择。该选项用来定义唤醒功能，外部输入/输出口(只有 PA)都具有将系统从 HALT 模式中唤醒的能力。
上拉选择。该选项用来定义输入/输出口做为输入时，是否带有内部上拉电阻；PA0~PA7 可以按位定义。
上拉选择。该选项用来定义输入/输出口做为输入时，是否带有内部上拉电阻；PB0~PB3 可以按位定义。
上拉选择。该选项用来定义输入/输出口做为输入时，是否带有内部上拉电阻；PC0~PC4 可以按位定义。
I/O 或其它功能选择。 PA3/PFD：PA3 可以做为 I/O 口或是 PFD 输出口
PFD 选择：如果 PA3 做为 PFD 输出口，会有两种可能：PFD0 或是 PFD1 输出。PFD0 与 PFD1 分别是由定时/计数器 0 或定时/计数器 1 计数溢出来驱动。
低电压复位功能：打开/关闭
PPG0 输出电平位准选择 (POLEV)：这个选项决定 PPG0 输出电平。是高电平有效还是低电平有效。如果这位被置“0”，则 PPG0 输出有效脉冲为高，反之为低。
PPG0 计数器时钟同步选择 (P0TSYN)：PPG0 计数器启动是否与 PPG 时钟同步

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 确切数据可以根据晶体/陶瓷谐振器的规格说明来选定。

2，复位电路

复位电路的电阻及电容值的选取应确保工作电压 VDD 在 RES 引脚上升到高之前能稳定，并将数值维持在正常允许的数据之内。为了防止噪声干扰，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。

指令集

简介

任何单片机成功运作的核心在于它的指令集，此指令集为一组程序指令码，用来指导单片机如何去执行指定的工作。在盛群单片机中，提供了丰富且灵活的指令，共超过六十条，程序设计者可以事半功倍地实现他们的应用。

为了更加容易理解各种各样的指令码，接下来按功能分组介绍它们。

指令周期

大部分的操作均只需要一个指令周期来执行。分支、调用或查表则需要两个指令周期。一个指令周期相当于四个系统时钟周期，因此如果在 8MHz 的系统时钟振荡器下，大部分的操作将在 0.5 μ s 中执行完成，而分支或调用操作则将在 1 μ s 中执行完成。虽然需要两个指令周期的指令通常指的是 JMP、CALL、RET、RETI 和查表指令，但如果牵涉到程序计数器低字节寄存器 PCL 也将多花费一个周期去加以执行。即指令改变 PCL 的内容进而导致直接跳转至新地址时，需要多一个周期去执行，例如“CLR PCL”或“MOV PCL, A”指令。对于跳转指令必须注意的是，如果比较的结果牵涉到跳转动作将多花费一个周期，如果没有则需一个周期即可。

数据的传送

单片机程序中数据传送是使用最为频繁的操作之一，使用三种 MOV 的指令，数据不但可以从寄存器转移至累加器(反之亦然)，而且能够直接移动立即数到累加器。数据传送最重要的应用之一是从输入端口接收数据或者传送数据到输出端口。

算术运算

算术运算和数据处理是大部分单片机应用所必需具备的能力，在盛群单片机内部的指令集中，可直接实现加与减的运算。当加法的结果超出 255 或减法的结果少于 0 时，要注意正确的处理进位和借位的问题。INC、INCA、DEC 和 DECA 指令提供了对一个指定地址的值加一或减一的功能。

逻辑和移位运算

标准逻辑运算例如 AND、OR、XOR 和 CPL 全都包含在盛群单片机内部的指令集中。大多数牵涉到数据运算的指令，数据的传送必须通过累加器。在所有逻辑数据运算中，如果运算结果为零，则零标志位将被置位，另外逻辑数据运用形式还有移位指令，例如 RR、RL、RRC 和 RLC 提供了向左或向右移动一位的方法。不同的移位指令可满足不同的应用需要。移位指令常用于串行端口的程序应用，数据可从内部寄存器转移至进位标志位，而此位则可被检验，移位运算还可应用在乘法与除法的运算组成中。

分支和控制的转换

程序分支是采取使用 JMP 指令跳转至指定地址或使用 CALL 指令调用子程序的形式，两者之不同在于当子程序被执行完毕后，程序必须马上返回原来的地址。这个动作是由放置在子程序里的返回指令 RET 来实现，它可使程序跳回 CALL 指令之后的地址。在 JMP 指令中，程序则只是跳到一个指定的地址而已，并不需如 CALL 指令般跳回。一个非常有用的分支指令是条件跳转，跳转条件是由数据存储器或指定位来加以决定。遵循跳转条件，程序将继续执行下一条指令或略过且跳转至接下来的指令。这些分支指令是程序走向的关键，跳转条件可能是外部开关输入，或者是内部数据位的值。

位运算

提供数据存储器中单个位的运算指令是盛群单片机的特性之一。这特性对于输出端口位的设置尤其有用，其中个别的位或端口的引脚可以使用“SET [m].i”或“CLR [m].i”指令来设定其为高位或低位。如果没有这特性，程序设计师必须先读入输入输出的 8 位数据，处理这些数据，然后再输出正确的新数据。这种读入-修改-写出的过程现在则被位运算指令所取代。

查表运算

数据的储存通常由寄存器完成，然而当处理大量固定的数据时，它的存储量常常造成对个别存储器的不便。为了改善此问题，盛群单片机允许在程序存储器中建立一个表格作为数据可直接存储的区域，只需要一组简易的指令即可对数据进行查表。

其它运算

除了上述功能指令外，其它指令还包括用于省电的“HALT”指令和使程序在极端电压或电磁环境下仍能正常工作的看门狗定时器控制指令。这些指令的使用则请查阅相关的章节。

指令集概要

下列表格根据功能描述了指令集的分类，利用下表列出的惯例可以作为基本的指令参考。

表格惯例：

x: 立即数

m: 数据存储器地址

A: 累加器

i: 第 0~7 位

addr: 程序存储器地址

助记符	说明	指令周期	影响标志位
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

助记符	说明	指令周期	影响标志位
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
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 and TBLH	2 ⁽¹⁾	无
TABRDL [m]	读取最后页的 ROM 内容，并送至数据存储器 and 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	进入暂停模式	1	TO,PDF

注： 1、对跳转指令而言，如果比较的结果牵涉到跳转即需 2 个周期，如果没有发生跳转，则只需一个周期。

2、任何指令若要改变 PCL 的内容将需要 2 个周期来执行。

3、对于“CLR WDT1”或“CLR WDT2”指令而言，TO 和 PDF 标志位也许会受执行结果影响，“CLR WDT1”和“CLR WDT2”被连续地执行后，TO 和 PDF 标志位会被清除，除此之外 TO 和 PDF 标志位保持不变。

指令定义

- ADC A, [m]** 累加器与数据存储器、进位标志相加，结果放入累加器
 说明：本指令把累加器、数据存储器值以及进位标志相加，结果存放到累加器。
 运算过程： $ACC \leftarrow ACC + [m] + C$
 影响标志位：OV, Z, AC, C
- ADCM A, [m]** 累加器与数据存储器、进位标志相加，结果放入数据存储器
 说明：本指令把累加器、数据存储器值以及进位标志相加，结果存放到存储器。
 运算过程： $[m] \leftarrow ACC + [m] + C$
 影响标志位：OV, Z, AC, C
- ADD A, [m]** 累加器与数据存储器相加，结果放入累加器
 说明：本指令把累加器、数据存储器值相加，结果存放到累加器。
 运算过程： $ACC \leftarrow ACC + [m]$
 影响标志位：OV, Z, AC, C
- ADD A, x** 累加器与立即数相加，结果放入累加器
 说明：本指令把累加器值和立即数相加，结果存放到累加器。
 运算过程： $ACC \leftarrow ACC + x$
 影响标志位：OV, Z, AC, C
- ADDM A, [m]** 累加器与数据存储器相加，结果放入数据存储器
 说明：本指令把累加器、数据存储器值相加，结果存放到数据存储器。
 运算过程： $[m] \leftarrow ACC + [m]$
 影响标志位：OV, Z, AC, C
- AND A, [m]** 累加器与数据存储器做“与”运算，结果放入累加器
 说明：本指令把累加器值、数据存储器值做逻辑与，结果存放到累加器。
 运算过程： $ACC \leftarrow ACC \text{ “AND” } [m]$
 影响标志位：Z
- AND A, x** 累加器与立即数做“与”运算，结果放入累加器
 说明：本指令把累加器值、立即数做逻辑与，结果存放到累加器。
 运算过程： $ACC \leftarrow ACC \text{ “AND” } x$
 影响标志位：Z
- ANDM A, [m]** 累加器与数据存储器做“与”运算，结果放入数据存储器
 说明：本指令把累加器值、数据存储器值做逻辑与，结果存放到数据存储器。
 运算过程： $[m] \leftarrow ACC \text{ “AND” } [m]$
 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: $Stack \leftarrow PC+1$
 $PC \leftarrow addr$

影响标志位: 无

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: 无

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: 无

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

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

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

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

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”，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: 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: Z

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: 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: $PC \leftarrow PC + 1$
 $PDF \leftarrow 1$
 $TO \leftarrow 0$

影响标志位: TO, PDF

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: $[m] \leftarrow [m] + 1$

影响标志位: Z

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: $ACC \leftarrow [m] + 1$

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:		$PC \leftarrow \text{addr}$
影响标志位:		无
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:		无
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:		无
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:		无
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:		无
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:		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:		$ACC \leftarrow ACC \text{ "OR" } x$
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:		$[m] \leftarrow ACC \text{ "OR" } [m]$
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:		$PC \leftarrow \text{Stack}$
影响标志位:		无

RET	A, x	<p>从子程序返回，并将立即数放入累加器</p> <p>说明：本指令是将堆栈寄存器中的程序计数器值送回程序计数器，并将立即数送回累加器。</p> <p>运算过程：$PC \leftarrow Stack$ $ACC \leftarrow x$</p> <p>影响标志位：无</p>
RETI		<p>从中断返回</p> <p>说明：本指令是将堆栈寄存器中的程序计数器值送回程序计数器，与 RET 不同的是它使用在中断程序结束返回时，它还会将中断控制寄存器 INTC 的 0 位(EMI)中断允许位置 1，允许中断服务。</p> <p>运算过程：$PC \leftarrow Stack$ $EMI \leftarrow 1$</p> <p>影响标志位：无</p>
RL	[m]	<p>数据存储器左移一位，结果放入数据存储器</p> <p>说明：本指令是将数据存储器内的数值左移一位，第 7 位移到第 0 位，结果送回数据存储器。</p> <p>运算过程：$[m].0 \leftarrow [m].7, [m].(i+1) \leftarrow [m].i; (i=0\sim6)$</p> <p>影响标志位：无</p>
RLA	[m]	<p>数据存储器左移一位，结果放入累加器</p> <p>说明：本指令是将存储器内的数值左移一位，第 7 位移到第 0 位，结果送到累加器，而数据存储器内的数值不变。</p> <p>运算过程：$ACC.0 \leftarrow [m].7, ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$</p> <p>影响标志位：无</p>
RLC	[m]	<p>带进位将数据存储器左移一位，结果放入数据存储器</p> <p>说明：本指令是将存储器内的数值与进位标志左移一位，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。</p> <p>运算过程：$[m].(i+1) \leftarrow [m].i; (i=0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$</p> <p>影响标志位：C</p>
RLCA	[m]	<p>带进位将数据存储器左移一位，结果放入累加器</p> <p>说明：本指令是将存储器内的数值与进位标志左移一位，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。</p> <p>运算过程：$ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$</p> <p>影响标志位：C</p>
RR	[m]	<p>数据存储器右移一位，结果放入数据存储器</p> <p>说明：本指令是将存储器内的数值循环右移，第 0 位移到第 7 位，结果送回数据存储器。</p> <p>运算过程：$[m].7 \leftarrow [m].0, [m].i \leftarrow [m].(i+1); (i=0\sim6)$</p> <p>影响标志位：无</p>

RRA	[m]	<p>数据存储器右移一位，结果放入累加器</p> <p>说明：本指令是将数据存储器内的数值循环右移，第 0 位移到第 7 位，结果送回累加器，而数据存储器内的数值不变。</p> <p>运算过程：$ACC.7 \leftarrow [m].0, ACC.i \leftarrow [m].(i+1); \quad (i=0\sim6)$</p> <p>影响标志位：无</p>
RRC	[m]	<p>带进位将数据存储器右移一位，结果放入数据存储器</p> <p>说明：本指令是将存储器内的数值加进位标志循环右移，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。</p> <p>运算过程：$[m].i \leftarrow [m].(i+1); \quad (i=0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$</p> <p>影响标志位：C</p>
RRCA	[m]	<p>带进位将数据存储器右移一位，结果放入累加器</p> <p>说明：本指令是将数据存储器内的数值加进位标志循环右移，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。</p> <p>运算过程：$ACC.i \leftarrow [m].(i+1); \quad (i=0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$</p> <p>影响标志位：C</p>
SBC	A,$\overline{[m]}$	<p>累加器与数据存储器、进位标志相减，结果放入累加器</p> <p>说明：本指令是把累加器值减去数据存储器值以及进位标志的取反，结果放到累加器。</p> <p>运算过程：$ACC \leftarrow ACC + [\overline{m}] + C$</p> <p>影响标志位：OV, Z, AC, C</p>
SBCM	A,$\overline{[m]}$	<p>累加器与数据存储器、进位标志相减，结果放入数据存储器</p> <p>说明：本指令是把累加器值减去数据存储器值以及进位标志取反，结果放到数据存储器。</p> <p>运算过程：$[m] \leftarrow ACC + [\overline{m}] + C$</p> <p>影响标志位：OV, Z, AC, C</p>
SDZ	[m]	<p>数据存储器减 1，如果结果为“0”，则跳过下一条指令</p> <p>说明：本指令是把数据存储器内的数值减 1，判断是否为 0，若为 0 则跳过下一条指令，即如果结果为零，放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以取得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。</p> <p>运算过程：如果 $[m]-1=0$，跳过下一条指令执行再下一条。</p> <p>影响标志位：无</p>
SDZA	[m]	<p>数据存储器减 1，将结果放入累加器，如果结果为“0”，则跳过下一条指令</p> <p>说明：本指令是把数据存储器内的数值减 1，判断是否为 0，为 0 则跳过下一行指令并将减完后数据存储器内的数值送到累加器，而数据存储器内的值不变，即若结果为 0，放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以取得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。</p>

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: 无

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: 无

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: 无

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: 无

SIZE 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: 无

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: 无

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: $ACC \leftarrow ACC + \overline{[m]} + 1$

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: $ACC \leftarrow ACC + \overline{x} + 1$

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： $[m] \leftarrow ACC + [\overline{m}] + 1$ 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： $[m].7 \sim [m].4 \leftrightarrow [m].3 \sim [m].0$ 影响标志位：无
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：无
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：无
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：无
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：无
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： $[m] \leftarrow$ 程序存储器低字节 $TBLH \leftarrow$ 程序存储器高字节 影响标志位：无
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: 无

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

24-pin SKDIP (300mil)外形尺寸

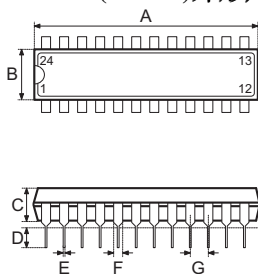


Fig 1. Full Lead Packages

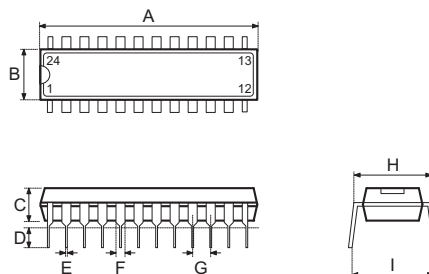


Fig 2. 1/2 Lead Packages

• MS-001d (见 fig 1)

标号	尺寸 (mil)		
	Min	Nom	Max
A	1230	--	1280
B	240	--	280
C	115	--	195
D	115	--	150
E	14	--	22
F	45	--	70
G	--	100	--
H	300	--	325
I	--	--	430

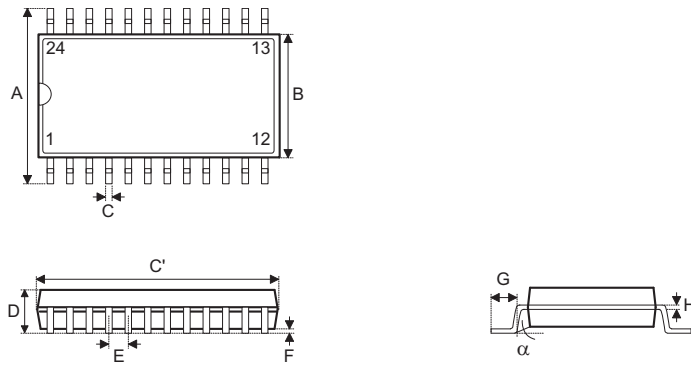
• MS-001d (见 fig 2)

标号	尺寸 (mil)		
	Min	Nom	Max
A	1160	--	1195
B	240	--	280
C	115	--	195
D	115	--	150
E	14	--	22
F	45	--	70
G	--	100	--
H	300	--	325
I	--	--	430

• MO-095a (见 fig 2)

标号	尺寸 (mil)		
	Min	Nom	Max
A	1145	--	1185
B	275	--	295
C	120	--	150
D	110	--	150
E	14	--	22
F	45	--	60
G	--	100	--
H	300	--	325
I	--	--	430

24-pin SOP (300mil)外形尺寸

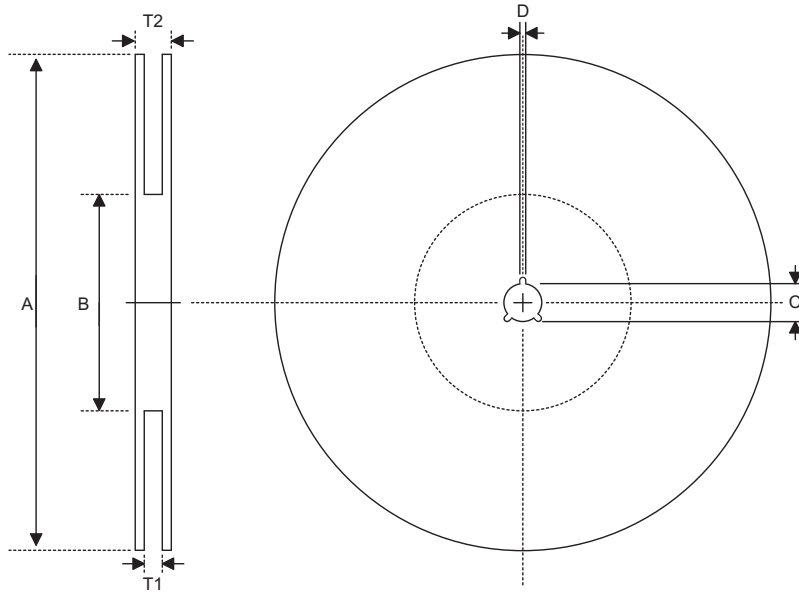


• MS-013

标号	尺寸 (mil)		
	Min	Nom	Max
A	393	--	419
B	256	--	300
C	12	--	20
C'	598	--	613
D	--	--	104
E	--	50	--
F	4	--	12
G	16	--	50
H	8	--	13
α	0°	--	8°

包装带和卷轴规格

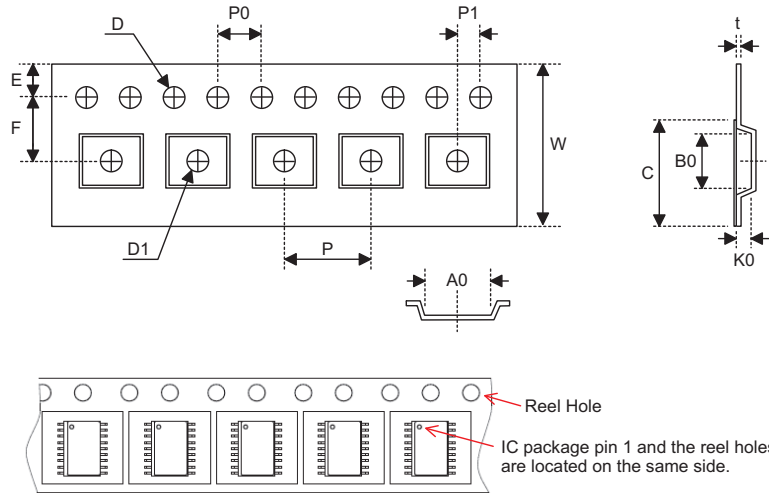
卷轴尺寸



SOP 24W

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 ^{+0.5/-0.2}
D	缝宽	2.0±0.5
T1	轮缘宽	24.8 ^{+0.3/-0.2}
T2	卷轴宽	30.2±0.2

运输带尺寸



SOP 24W

标号	描述	尺寸(mm)
W	运输带宽	24.0±0.3
P	空穴间距	12.0±0.1
E	穿孔位置	1.75±0.1
F	空穴至穿孔距离 (宽度)	11.5±0.1
D	穿孔直径	1.55 ^{+0.10/-0.00}
D1	空穴中之小孔直径	1.50 ^{+0.25/-0.00}
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	15.9±0.1
K0	空穴深	3.1±0.1
t	传输带厚度	0.35±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号香榭广场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>