

## 技术相关信息

- [工具信息](#)
- [FAQs](#)
- [应用范例](#)
  - [HA0075S MCU 复位电路和振荡电路的应用范例](#)

## 特性

- 工作电压：  
f<sub>sys</sub>=4MHz: 2.2V~5.5V  
f<sub>sys</sub>=8MHz: 3.3V~5.5V
- 16 个双向输入/输出口
- 2 个外部中断口与输入/输出口共用
- 具有 7 级预分频器及中断功能的 8 位可编程定时/计数器
- 外部 RC 振荡转换电路
- 内置晶体和 RC 振荡器
- 看门狗定时器
- 12 通道电容/电阻型传感器输入
- ROM 程序存储: 2048 × 14
- RAM 数据存储: 120 × 8
- 提供暂停和唤醒功能, 以降低功耗
- V<sub>DD</sub>=5V, 系统时钟为 8MHz 时, 指令周期为 0.5μs
- 所有指令都可在 1 或 2 个指令周期内完成
- 14 位查表指令
- 4 层堆栈
- 位操作指令
- 63 条指令
- 低电压复位功能
- 集成直流 24V 转 5V 的 LDO 稳压器
- 蜂鸣器和灯丝 5V 转 24V 输出电平转换器
- 24 位移位寄存器/锁存器, 用于驱动 VFD 面板 24 栅格/段输出
- 集成 3 线串行 VFD 接口, 用于栅格/段显示控制
- 52-pin QFP 封装形式

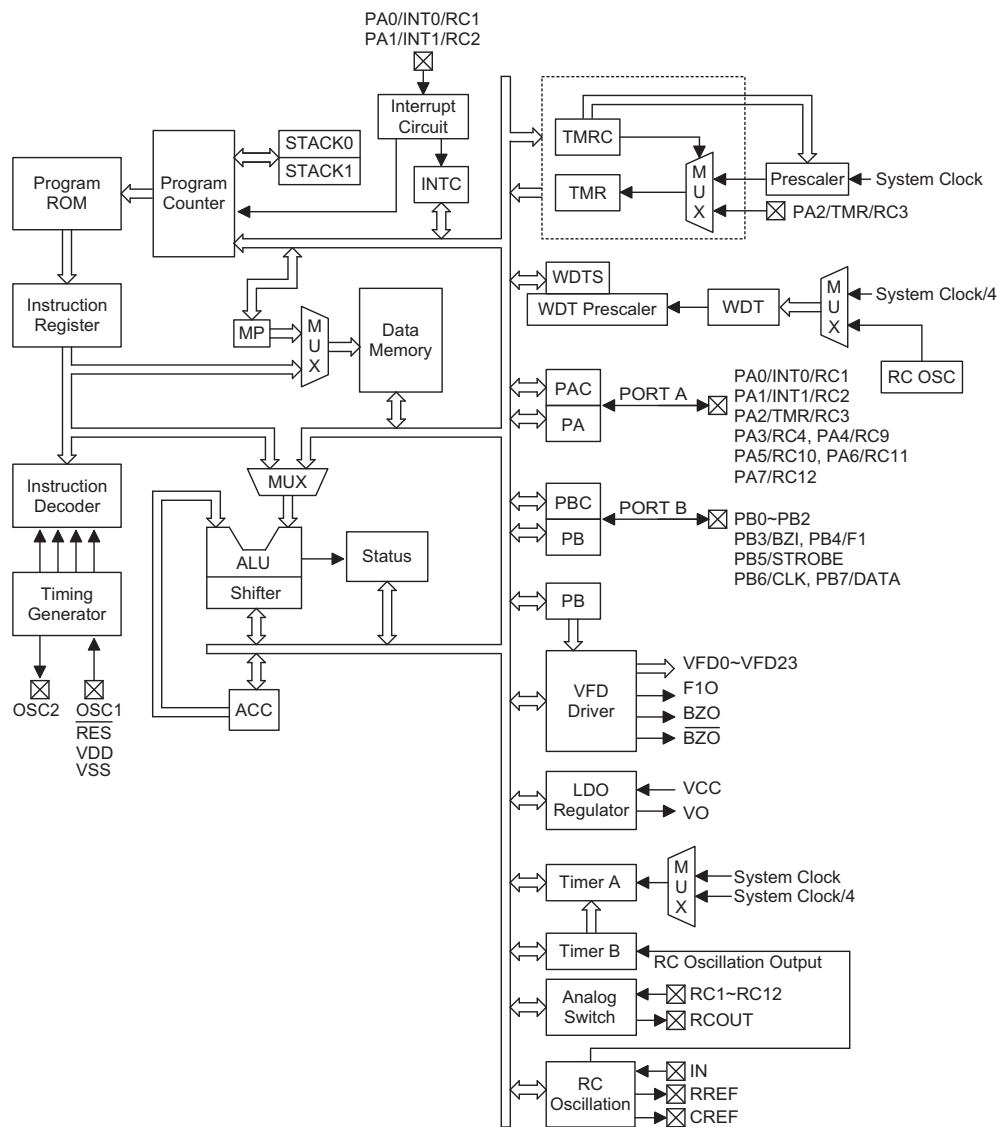
## 概述

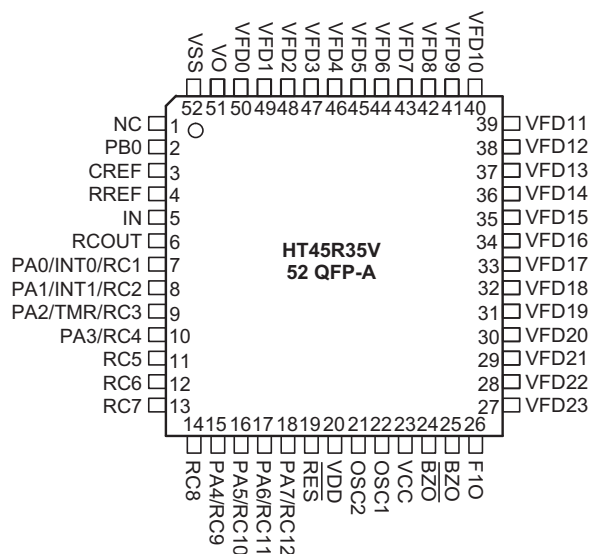
HT45R35V 是一款 C/R-F 型具有 8 位高性能精简指令集的单片机，专门为需要 VFD 功能的产品而设计。

秉承 HOLTEK MCU 一般特性，该单片机带有暂停和唤醒功能，振荡器选项等等，这些都保证使用者的应用只需极少的外部元器件便可实现。

这款单片机专门为直接与 VFD 面板相连的 VFD 应用而设计。集成 C/R-F 功能，外加功耗低、性能良好、I/O 使用灵活、成本低等优势，使这款单片机可以广泛应用于 VFD 相关产品中，例如家电定时产品，各种消费产品，子系统控制器，其他家电应用等方面。

## 方框图



**引脚图**

**引脚说明**

引脚名称	输入/输出	配置选项	说明
PA0/INT0/RC1 PA1/INT1/RC2 PA2/TMR/RC3 PA3/RC4 PA4/RC9 PA5/RC10 PA6/RC11 PA7/RC12	输入/输出	上拉电阻* 唤醒功能	8 位双向输入/输出口。每一位可通过配置选项设置为唤醒输入。可由软件设置为 CMOS 输出或者斯密特触发输入，可通过配置选项选择是否带有上拉电阻。 外部中断输入 INT0 和 INT1 分别与 PA0、PA1 引脚共用。中断使能/除能和中断上升沿/下降沿触发形式可由配置选项设置。PA2 与外部定时器输入脚 TMR 共用。 PA0~PA3 和 PA4~PA7 分别与 RC1~RC4 和 RC9~RC12 引脚共用，他们可分别由配置选项选择。 RC1~RC4 和 RC9~RC12 引脚与电容或者电阻相连。
PB0	输入/输出	上拉电阻*	1 位双向输入/输出口。可由软件设置为 CMOS 输出或斯密特触发输入。可通过配置选项选择是否带有上拉电阻。
PB1~PB2	—	—	这两个引脚为内部输入/输出引脚，没有引出。
PB3/BZI PB4/F1 PB5/STROBW PB6/CLK PB7/DATA	输入/输出	上拉电阻*	PB3~PB7 用于控制 VFD 驱动接口。可通过配置选项选择是否带有上拉电阻。该引脚必须设为输出模式作为 VFD 接口引脚，而不是作为普通的输入/输出口。
RC5~RC8	输入/输出	—	电容或者电阻连接引脚。
RCOUT	输入	—	电容或者电阻连接引脚到 RC OSC。
IN	输入	—	振荡器输入引脚。

引脚名称	输入/输出	配置选项	说明
RREF	输出	—	参考电阻连接引脚。
CREF	输出	—	参考电容连接引脚。
F1O	输出	—	高压灯丝信号输出。
BZO $\overline{\text{BZO}}$	输出	—	高压蜂鸣器互补信号输出。
VO	—	—	LDO 稳压器输出。
VCC	—	—	高压正电源用于驱动 VFD 灯丝、F1O、BZO 和 $\overline{\text{BZO}}$ 输出。在 PCB 上，建议外部使用一个 10 $\mu\text{F}$ 的电容与地相连，以降低雷击电压。
VFD0~VFD23	输出	—	高压栅格/段输出，用于 VFD 面板。
$\overline{\text{RES}}$	输入	—	施密特触发复位输入。低有效。
VSS	—	—	负电源，接地。
VDD	—	—	正电源。
OSC1 OSC2	输入 输出	晶振或者 RC	OSC1, OSC2 连接 RC 或晶体振荡器（由配置选项设定）以产生内部系统时钟。如果选择 RC 振荡器，OSC2 可用于监视系统时钟。它的频率为 1/4 系统时钟。

- 注： 1、\*所有的上拉电阻都由位选项控制。  
 2、PB3~PB7 为 5 个内部引脚，没有引出，控制寄存器必须将这些引脚设为输出。  
 3、PB3~PB7 可以分别设置为带上拉电阻。

## 极限参数

电源供应电压..... $V_{SS}-0.3V\sim V_{SS}+6.0V$	储存温度..... $-50^{\circ}\text{C}\sim 125^{\circ}\text{C}$
端口输入电压..... $V_{SS}-0.3V\sim V_{DD}+0.3V$	工作温度..... $-40^{\circ}\text{C}\sim 85^{\circ}\text{C}$
VCC 供应电压..... 12V~24V	$I_{OL}$ 总电流..... 150mA
$I_{OH}$ 总电流..... -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

$T_a=25^{\circ}\text{C}$

符号	参数	测试条件			最小	典型	最大	单位
		$V_{DD}$	$V_{CC}$	条件				
$V_{DD}$	工作电压	—	—	$f_{SYS}=4\text{MHz}$	2.2	—	5.5	V
				$f_{SYS}=8\text{MHz}$	3.3	—	5.5	V
$I_{DD1}$	工作电流 (Crystal OSC, RC OSC)	3V	—	无负载	—	1	2	mA
				$f_{SYS}=4\text{MHz}$	—	3	5	mA
$I_{DD2}$	工作电流 (Crystal OSC, RC OSC)	5V	—	无负载, $f_{SYS}=8\text{MHz}$	—	4	8	mA

符号	参数	测试条件			最小	典型	最大	单位
		V <sub>DD</sub>	V <sub>CC</sub>	条件				
I <sub>STB1</sub>	静态电流 (WDT 使能)	3V	—	无负载, 系统 HALT	—	—	5	μA
		5V	—		—	—	10	μA
I <sub>STB2</sub>	静态电流 (WDT 除能)	3V	—	无负载, 系统 HALT	—	—	1	μA
		5V	—		—	—	2	μA
V <sub>IL1</sub>	输入/输出口、TMR、 INT0 和 INT1 的低电平 输入电压	—	—	—	0	—	0.3V <sub>DD</sub>	V
V <sub>IH1</sub>	输入/输出口、TMR、 INT0 和 INT1 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( $\overline{\text{RES}}$ )	—	—	—	0	—	0.4V <sub>DD</sub>	V
V <sub>IH2</sub>	高电平输入电压( $\overline{\text{RES}}$ )	—	—	—	0.9V <sub>DD</sub>	—	V <sub>DD</sub>	V
V <sub>LVR</sub>	低电压复位电压	—	—	LVR 使能	2.7	3.0	3.3	V
I <sub>OL</sub>	输入/输出口、RREF 和 CREF 的灌电流	3V	—	V <sub>OL</sub> =0.1V <sub>DD</sub>	4	8	—	mA
		5V	—		10	20	—	mA
I <sub>OH</sub>	输入/输出口、RREF 和 CREF 的源电流	3V	—	V <sub>OH</sub> =0.9V <sub>DD</sub>	-2	-4	—	mA
		5V	—		-5	-10	—	mA
R <sub>PH</sub>	上拉电阻	3V	—	—	20	60	100	kΩ
		5V	—		10	30	50	kΩ
R <sub>PL</sub>	RC1~RC12 下拉电阻	3V	—	—	20	60	100	kΩ
		5V	—		10	30	50	kΩ
V <sub>O</sub>	LDO 输出电压	—	—	—	4.7	5.0	5.3	V
V <sub>CC</sub>	VFD、F10、BZO 和 $\overline{\text{BZO}}$ 输出供电电压	—	—	—	12	—	24	V
I <sub>OUT</sub>	LDO 最大输出电流	—	—	V <sub>CC</sub> ≥5V	10	—	—	mA
ΔVLNR	线性调整率	—	—	V <sub>IN</sub> =(V <sub>OUT</sub> +0.1V) ~24V, I <sub>OUT</sub> =1mA	TBD	0.06	TBD	%/V

符号	参数	测试条件			最小	典型	最大	单位
		V <sub>DD</sub>	V <sub>CC</sub>	条件				
△VLDR	负载调整率	—	—	I <sub>OUT</sub> =100μA 到 20mA, C <sub>OUT</sub> =10pF	TBD	0.16	TBD	%/mA
VDRO	输入输出电压差	—	—	I <sub>OUT</sub> =1mA	25	30	35	mV
I <sub>CC1</sub>	逻辑工作电流 1	5V	18V	无负载, VFD 输出, 所有输出低, CLK=100kHz	—	70	110	μA
			24V		—	TBD	TBD	μA
I <sub>CC2</sub>	逻辑工作电流 2	5V	18V	无负载, VFD 输出, 所有输出高, CLK=100kHz	—	70	110	μA
			24V		—	TBD	TBD	μA
I <sub>CC3</sub>	蜂鸣器工作电流	5V	18V	无负载, BZI 输入 50kHz	—	130	180	μA
			24V		—	TBD	TBD	μA
I <sub>CC4</sub>	灯丝工作电流	5V	18V	无负载, F1 输入 50kHz	—	90	140	μA
			24V		—	TBD	TBD	μA
I <sub>STB</sub>	静态电流 (LDO 开启, WDT 使能/除能)	5V	18V	无负载	—	65 (TBC)	105 (TBC)	μA
			24V		—	TBD	TBD	μA
I <sub>OL2</sub>	F1O 灌电流	5V	18V	V <sub>OL</sub> =0.1V <sub>CC</sub>	2.5	5.0	—	mA
			24V		TBD	TBD	—	mA
I <sub>OH2</sub>	F1O 源电流	5V	18V	V <sub>OH</sub> =0.9V <sub>CC</sub>	-15	-30	—	mA
			24V		TBD	TBD	—	mA
I <sub>OL3</sub>	BZO/BZO 灌电流	5V	18V	V <sub>OL</sub> =0.1V <sub>CC</sub>	15	30	—	mA
			24V		TBD	TBD	—	mA
I <sub>OH3</sub>	BZO/BZO 源电流	5V	18V	V <sub>OH</sub> =0.9V <sub>CC</sub>	-15	-30	—	mA
			24V		TBD	TBD	—	mA
I <sub>OL4</sub>	栅格/段灌电流	5V	18V	V <sub>OL</sub> =0.1V <sub>CC</sub>	2.5	5.0	—	mA
			24V		TBD	TBD	—	mA
I <sub>OH4</sub>	栅格/段源电流	5V	18V	V <sub>OH</sub> =0.9V <sub>CC</sub>	-6	-12	—	mA
			24V		TBD	TBD	—	mA

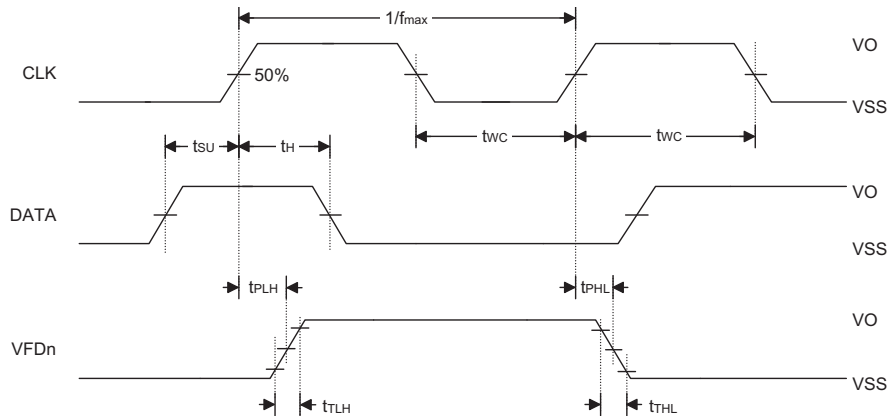
**交流电气特性**

Ta=25°C

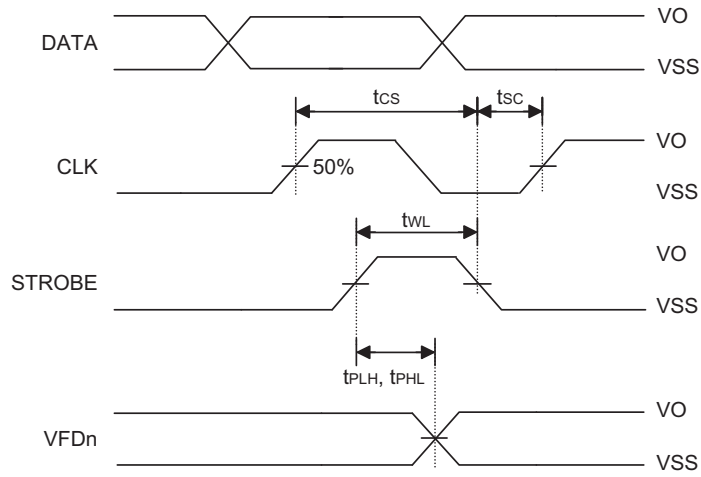
符号	参数	测试条件		最小	典型	最大	单位
		V <sub>DD</sub>	条件				
f <sub>SYS</sub>	系统时钟 (晶体振荡、RC 振荡)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
f <sub>TIMER</sub>	定时器输入频率	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	kHz
t <sub>WDTOSC</sub>	看门狗振荡周期	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t <sub>WDT1</sub>	看门狗溢出周期 (看门狗 RC 振荡器)	3V	看门狗无预分频器	11	23	46	ms
		5V		8	17	33	ms
t <sub>WDT2</sub>	看门狗溢出周期 (系统时钟四分频)	—	看门狗无预分频器	—	1024	—	t <sub>SYS</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	—	从 HALT 模式中唤醒	—	1024	—	t <sub>SYS</sub>
t <sub>INT</sub>	中断脉冲宽度	—	—	1	—	—	μs
t <sub>LVR</sub>	低电压复位时间	—	—	0.25	1	2	ms
t <sub>PHL</sub> , t <sub>PLH</sub>	传播时延 (Clock to VFD Output)	—	V <sub>CC</sub> =15V	—	100	200	ns
	传播时延 (Strobe to VFD Output)	—	V <sub>CC</sub> =15V	—	100	200	ns
t <sub>THL</sub> , t <sub>TLH</sub>	输出转换时间	—	V <sub>CC</sub> =15V	—	40	80	ns
t <sub>SU</sub>	数据建立时间	—	V <sub>CC</sub> =15V	—	10	20	ns
t <sub>CS</sub>	建立时间 (Clock to Strobe)	—	V <sub>CC</sub> =15V	—	10	20	ns
t <sub>H</sub>	保持时间 (Data to Clock)	—	V <sub>CC</sub> =15V	—	10	20	ns
t <sub>SC</sub>	保持时间 (Clock to Strobe)	—	V <sub>CC</sub> =15V	—	75	150	ns
t <sub>R</sub> ,t <sub>F</sub>	时钟输入上升沿或下降 沿时间	—	V <sub>CC</sub> =15V	—	—	20	ns
t <sub>WC</sub>	时钟脉冲宽度	—	V <sub>CC</sub> =15V	—	40	83	ns
t <sub>WL</sub>	选通脉冲宽度	—	V <sub>CC</sub> =15V	—	35	70	ns
f <sub>MAX</sub>	最大时钟输入频率	—	V <sub>CC</sub> =15V	—	8	—	MHz

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

交流波形



数据传播延迟、建立和保持时间



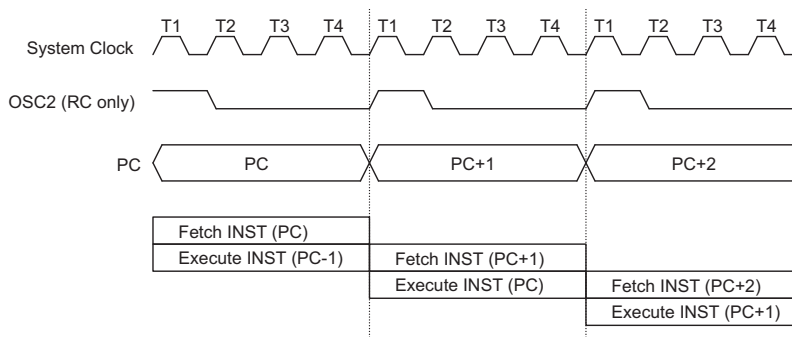
选通脉冲传播延迟、建立和保持时间

## 功能概述

### 执行流程

单片机的系统时钟来源于晶振或者 RC 振荡器，它被细分为四个内部产生的非重叠时序。一个指令周期包含四个系统时钟周期。

指令的抓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抓取指令操作，而在下一个指令周期进行解码与执行该指令。因此，流水线的方式使多数指令在一个指令周期内被有效执行。但如果涉及到的指令要改变程序计数器的内容，就需要两个指令周期去完成这一条指令。



### 执行流程

#### 程序计数器 — PC

程序计数器 (PC) 控制程序存储器 ROM 中指令执行的顺序并且它可寻址整个 ROM 的范围。

执行完一个程序存储器字并抓取一个指令码后，程序计数器的值会加一。然后程序计数器指向下一个指令码的地址。

当执行跳转、条件跳转、加载 PCL 寄存器、子程序调用、初始复位、内部中断、外部中断或者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。

对于条件跳转指令，一旦条件符合，在当前指令执行时取得的下一条指令将会被舍弃，而由一个空指令周期来取代。否则程序将继续执行下一条指令。

程序计数器的低字节 (PCL) 是一个可读写的寄存器。数据直接写入这个寄存器将会有个短跳转。但跳转被限制在存储器的当前页中。

当要执行程序跳转时，会插入一个空指令周期。

模式	程序计数器										
	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
初始化复位	0	0	0	0	0	0	0	0	0	0	0
外部中断 0	0	0	0	0	0	0	0	0	1	0	0
外部中断 1	0	0	0	0	0	0	0	1	0	0	0
定时/计数器溢出	0	0	0	0	0	0	0	1	1	0	0
外部 RC 振荡中断	0	0	0	0	0	0	1	0	0	0	0
条件跳转	程序计数器+2										
装载 PCL	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
跳转, 子程序调用	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
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

S10 ~ S0 : 堆栈寄存器位

#10~#0: 指令代码位

@7 ~ @0 : PCL 位

### 程序存储器

程序存储器用于存储被执行的程序指令。其中也包含数据、表格和中断入口，它的容量为 2048 × 14 位，程序存储器用程序计数器和表格指针来寻址。

程序存储器内部某些地址保留用做特殊用途。

- 地址 000H

该地址为程序初始化保留。在芯片复位之后，程序将跳到这个地址并开始执行。

- 地址 004H

该地址为外部中断 0 服务程序保留。当 INTO 引脚有触发信号输入，如果外部中断使能且堆栈未满的情况下，程序将跳到这个地址并开始执行。

- 地址 008H

该地址为外部中断 1 服务程序保留。当 INT1 引脚有触发信号输入，如果外部中断使能且堆栈未满的情况下，程序将跳到这个地址并开始执行。

- 地址 00CH

该地址为定时/计数器中断服务程序保留。当定时器中断来源于定时/计数器溢出，而定时/计数器中断使能且堆栈未满的情况下，程序将跳到这个地址并开始执行。

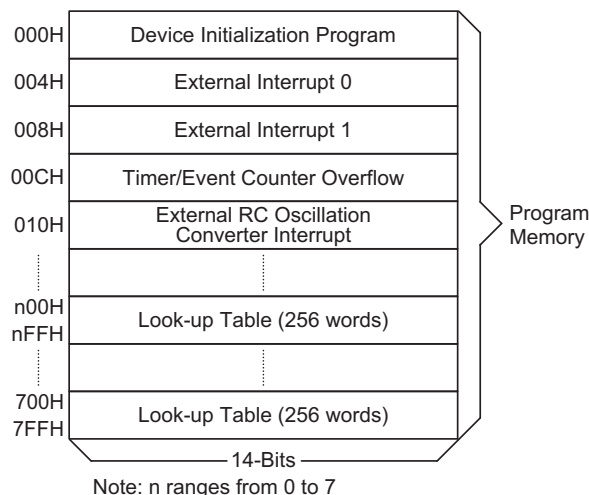
- 地址 010H

该地址为外部 RC 振荡转换中断服务程序保留。当中断来源于外部 RC 振荡，如果中断使能且堆栈未满的情况下，程序将跳到这个地址并开始执行。

- 表格区

程序存储器中的任何地址都可以定义成一个表格。执行“TABRDC [m]”（查当前页表格，1 页 = 256 字）和“TABRDL [m]”指令时，程序存储器中表格数据低字节，将被传送到所指定的数据存储器，而高字节则被传送到 TBLH。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 的最高两位始终为“0”。表格高字节寄存器 TBLH 为只读寄存器。表格指针 TBLP 是可读/写寄存器，用来指明表格地址。使用表格之前，必须先将地址写入表格指针 TBLP 中。TBLH 为只读寄存器，不能重新存储。若主程序和中断服务程序 (ISR) 都使用表格读取指令，中断服务程序可能改变 TBLH 的值，若随后在主程序中再次使用这个值，则会发生错误。因此建议避免主程序和 ISR 同时使用表格读取指令。然而在某些情况下，同时使用表格

读取指令是不可避免的，则在执行任何主程序的表格读取指令前，中断应该先除能。中断不能重新使能直到 TBLH 已经备份。所有与表格相关的指令，都需要两个指令周期去完成操作。根据需求这些区域可以作为正常的程序存储器来使用。



程序存储器

指令	表格区										
	*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                      P10~P8: 当前程序计数位

@7~@0: 表格指针位

**堆栈寄存器 — STACK**

堆栈是一个特殊的存储空间，用来存储程序计数器中的内容。该单片机有 4 层堆栈，它不是数据部分也不是程序空间部分，而且它既不是可读取也不是可写入的。当前层由堆栈指针(SP)加以指示，同样 SP 也是不可读写的。在子程序调用或中断响应服务时，程序计数器的内容被压入到堆栈中。当子程序或中断响应结束时，返回指令(RET 或 RETI)使程序计数器从堆栈中重新得到它以前的值。当一个芯片复位后，堆栈指针将指向堆栈顶部。

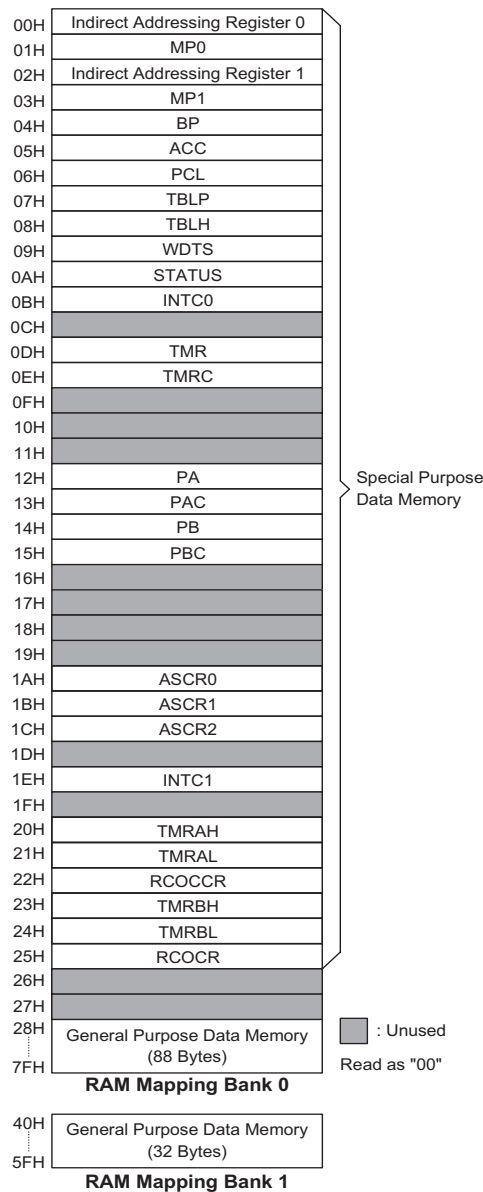
如果堆栈已满，且有非屏蔽的中断发生，那么只有中断请求标志位会被记录下来，中断响应被禁止。当堆栈指针减少(执行 RET 或 RETI)，中断才会被响应。这个特性提供程序设计者简单的方法来预防堆栈溢出。然而即使堆栈已满，CALL 指令仍然可以被执行，造成堆栈溢出将首先进入堆栈的内容将会丢失，只有最后的 4 个返回地址被保留下来。

**数据存储单元 — RAM**

数据存储单元的容量为 146×8 位。数据存储单元被分为两个部分：特殊功能寄存器和通用数据存储单元（120×8）。数据存储单元大多可以进行读/写，而有些是只读。通用数据存储单元在存储区 0 中它的地址从 28H 到 7FH，在存储区 1 中它的地址从 40H 到 5FH，通用数据存储单元常被用于储存数据和指令控制信息。

所有的数据存储单元都可以直接处理算术、逻辑、加、减和移位操作。除了一些特殊位，其它的每个位都可以用“SET [m].i”和“CLR [m].i”操作指令进行置位和复位。数据存储单元还可以通过间接寻址指针寄存器（MP0:01H，MP1:02H）进行间接访问。

存储区 1 必须由间接寻址指针 MP1 和间接寻址寄存器 IAR1 进行间接寻址。MP0 和 IAR0 对任何存储区的直接寻址和间接寻址，都只能存取存储区 0 的数据。



**数据存储单元映射方式**

## 间接寻址寄存器

间接寻址的方式允许使用间接指针操作数据，以取代定义实际存储器地址的直接存储器寻址方法。任何对间接寻址寄存器的操作，将对相关的间接寻址指针所指向的存储器地址产生对应的读/写操作。HT45R35V 提供两个间接寻址寄存器（IAR0 和 IAR1）和两个间接寻址指针（MP0 和 MP1）。需要注意的是，这些间接寻址寄存器并不是实际存在的，间接读取间接寻址寄存器将返回“00H”，而间接写入此寄存器则不做任何操作。

HT45R35V 提供两个间接寻址指针，即 MP0 和 MP1，由于这些指针在数据存储器中像通用寄存器一样被操作，因此提供了一个寻址和数据追踪的有效方法。当对间接寻址寄存器进行任何操作时，单片机指向的实际地址是由间接寻址指针所指向的地址。

间接寻址指针的第 7 位并未使用。然而，必须注意的是当单片机读取该间接寻址指针时，将会读到“1”。

## 存储区指针 — BP

当访问存储区 0 和存储区 1 中的通用数据存储器时，必须确保选中正确的存储器。通用数据存储器被分为两个存储区，即存储区 0 和存储区 1。使用存储区指针来选择相对应的数据存储器区块。如果要访问数据存储器 0 和数据存储器 1 的数据，则必须分别将 BP 设定为“00H”和“01H”，然而，需要注意的是对存储区 1 的数据进行存取只能使用间接寻址指针 MP1 和间接寻址寄存器 IAR1。

任何直接寻址或通过 MP0 和 IAR0 的间接寻址只会对存储区 0 内数据进行存取。复位后，数据存储器指针 BP 指向存储区 0，但是在 HALT 模式下的 WDT 溢出复位，不会改变通用数据存储器的存储区号。

应该注意的是特殊数据存储器不受存储区指针 BP 的影响，也就是说，不论是在哪一个存储区，都能对特殊寄存器进行读写操作。

## 累加器

累加器与算术逻辑单元 ALU 有密切关系。累加器对应于数据存储器地址“05H”，作为运算的立即数据。两个数据寄存器之间的数据传送必须通过累加器。

## 算术逻辑单元 — ALU

算术逻辑单元用于执行 8 位的算术和逻辑运算。ALU 所提供的功能如下：

- 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 在保存数据计算结果的同时也改变了状态寄存器的值。

## 状态寄存器 — STATUS

8 位的状态寄存器由零标志位(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。状态寄存器不仅记录单片机的运行状态，而且还控制操作顺序。

除了 TO 和 PDF 标志外，状态寄存器中的位像其他大部分寄存器一样可以被改变。任何对状态寄存器的写操作都不会改变 TO 或 PDF 标志位。另外，执行不同的指令后，与状态寄存器有关的运算可能会得到不同的结果。TO 标志位只会受系统上电、看门狗溢出、执行“CLR WDT”或“HALT”指令影响。

PDF 标志位只会受执行“HALT”或“CLR WDT”指令或系统上电影响。

Z、OV、AC 和 C 标志位通常反映的是最近一次操作的状态。

另外，进入中断程序或者执行子程序调用时，状态寄存器不会自动压入堆栈保存。如果状态寄存器的内容非常重要且子程序可能改变状态寄存器的话，则程序员必须事先将 STATUS 的值保存好。

位	符号	功能
0	C	当加法运算的结果产生进位，或减法运算的结果没有产生借位时，则 C 被置位，否则 C 被清零，同时 C 也会被循环移位指令所影响。
1	AC	当低半字节加法运算的结果产生进位，或高半字节减法运算的结果没有产生借位时，AC 被置位，否则 AC 被清零。
2	Z	当算术或逻辑运算结果是零时，Z 被置位，否则 Z 被清零
3	OV	当运算结果高两位的进位状态异或结果为 1 时，OV 被置位，否则 OV 被清零。
4	PDF	系统上电或执行“CLR WDT”指令会清零 PDF，而执行“HALT”指令则会置位 PDF。
5	TO	系统上电或执行“CLR WDT”或“HALT”指令会清零 TO，而当 WDT 溢出则会置位 TO。
6~7	—	未用，读取为“0”。

### 状态寄存器 (0AH)

#### 中断

HT45R35V 提供 2 个外部中断、1 个内部 8 位定时/计数器中断和 1 个外部 RC 振荡中断。INTC0 中断控制寄存器 0 和 INTC1 中断控制寄存器 1 包含了用来设置中断使能/除能的中断控制位以及中断请求标志位。

一旦中断子程序被响应，所有其它的中断都会被屏蔽，（通过自动清除 EMI 位）。这种做法可以防止任何进一步的中断嵌套。这时如果有其它的中断发生，只有中断请求标志位会被记录。如果某个中断服务子程序正在执行，此时有另外一个中断要求响应，程序员可以置位 EMI 位和 INTC 相关的位，以便进行中断嵌套。

如果堆栈已满，即使相关中断使能，中断请求仍不会被响应，直到堆栈指针减小为止。如果需要中断立即得到响应，应避免堆栈饱和。

所有中断都具有唤醒功能。当进入中断服务程序，系统会将程序计数器的内容压入堆栈，然后进入程序存储器的特定向量的子程序中。但这时只有程序计数器的内容被压入堆栈。如果累加器或者状态寄存器的内容会被中断服务程序改变，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 INTO 或 INT1 引脚边沿信号触发的。配置选项选择该引脚为中断输入，上升沿触发还是下降沿触发。触发条件满足将请求标志位（EIF0: INTC0 的第 4 位，EIF1: INTC0 的第 5 位）置位。如果中断允许，堆栈未满并且外部中断脚状态改变，将调用位于地址 04H 或 08H 处的外部中断向量子程序。当响应外部中断服务子程序时，中断请求标志位 EIF0 或 EIF1 和总中断控制位 EMI 会被清零，以除能其它中断。

内部定时/计数器中断是由定时/计数器溢出触发，其中断请求标志（TF: INTC0 的第 6 位）会被置位。当中断允许，堆栈未满，将调用位于地址 0CH 的子程序。当响应中断服务子程序时，中断请求标志位 TF 和总中断控制位 EMI 会被清零，以除能其它中断。

外部 RC 振荡中断是由 Timer A 或 Timer B 溢出触发的，其中断请求标志位（RCOCF: INTC1 的第 4 位）会被置位。当中断使能，堆栈未满，RCOCF 位被置位，将调用位于地址 10H 的子程序。当响应中断服务子程序时，中断请求标志位 RCOCF 和总中断控制位 EMI 会被清零，以除能其它中断。

如果某个中断服务子程序正在执行，堆栈未滿，则其它中断将被保留直到执行 RETI 指令或者 EMI 位和相关中断控制位被设为“1”。如果要从中断服务子程序返回，只要执行 RET 或 RETI 指令即可。RETI 会置位 EMI 位以使能中断服务，但是 RET 不会。

位	符号	功能
0	EMI	总中断控制位（1=使能；0=除能）
1	EEI0	外部中断 0 控制位（1=使能；0=除能）
2	EEI1	外部中断 1 控制位（1=使能；0=除能）
3	ETI	定时/计数器中断控制位（1=使能；0=除能）
4	EIF0	外部中断 0 请求标志位（1=有效；0=无效）
5	EIF1	外部中断 1 请求标志位（1=有效；0=无效）
6	TF	内部定时/计数器中断请求标志位（1=有效；0=无效）
7	—	未用，读取为“0”

### INTC0 (0BH) 寄存器

位	符号	功能
0	ERCOCI	外部 RC 振荡中断控制位（1=使能；0=除能）
1~3, 5~7	—	未用，读取位“0”
4	RCOCF	外部 RC 振荡中断请求标志位（1=有效；0=无效）

### INTC1(1EH)寄存器

中断发生在两个连续的 T2 脉冲的上升沿之间，如果相应的中断使能被允许，中断将在后一个 T2 脉冲响应。下表指出在同时提出请求的情况下所提供的优先级。这个可以通过重新设定 EMI 位来加以屏蔽。

中断源	优先级	中断向量
外部中断 0	1	04H
外部中断 1	2	08H
定时/计数器溢出中断	3	0CH
外部 RC 振荡器中断	4	10H

### 中断优先级

EMI、EEI0、EEI1、ETI 和 ERCOCI 都是中断控制使能/除能位。通过除能这些中断使能位，可以屏蔽其它中断请求。然而，一旦中断请求标志位被置位，会一直保留在 INTC1 和 INTC0 寄存器内，直到相应的中断服务子程序执行或软件指令清除。

建议在中断服务子程序中不要使用“CALL 子程序”指令。因为中断通常发生在不可预料的情况或是需要立刻执行的某些应用。假如只剩下一层堆栈且没有控制好中断，当“CALL 子程序”在中断服务子程序中执行时，将破坏原来的控制序列。

### 振荡器配置

不同的振荡器选择可以让使用者在不同的应用需求中获得更大范围的功能。有两种系统时钟可供选择，看门狗定时器也有多种灵活的时钟源选项，给使用者最大的灵活性。所有的振荡器选项都可通过配置选项来选择。

有两种方法产生系统时钟：

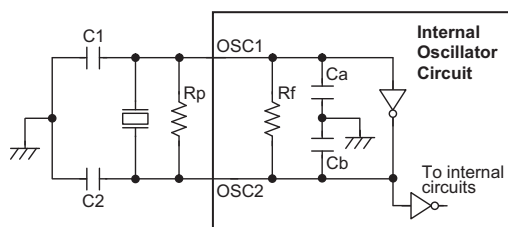
- 外部晶体/陶瓷振荡器
- 外部 RC 振荡器

系统时钟的产生必须通过掩膜选项设置上述两种方法中的一种。

预知更多振荡器的相关应用信息可参考 HOLTEK 网站应用范例 HA0075S。

### 外部晶体/陶瓷振荡器

选择外部晶振配置选项之后，只要简单地将晶体连接至 OSC1 和 OSC2，就会产生震荡所需的相移及反馈，而不需外部电容。然而对于某些晶振和大多陶瓷共振器类型，可能需要加上两个小电容 C1, C2 以保证起振和获得精确的频率。C1 和 C2 的大小则应根据制造商的晶体/陶瓷晶振的规格参数加以选择。大多数的应用场合中，电阻 R<sub>P</sub> 并不需要，但在某些场合中，可能需要 R<sub>P</sub> 以保证振荡器起振。



Note: 1. R<sub>p</sub> is normally not required.  
2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

### 晶体/陶瓷振荡器

在 5V, 25°C 下，内部 Ca、Cb、Rf 的典型值		
Ca	Cb	Rf
TBD	TBD	TBD

### 振荡器内部元件值

晶体振荡器 C1 和 C2 值			
晶体振荡器频率	C1	C2	CL
12MHz 晶振	TBD	TBD	TBD
8MHz 晶振	TBD	TBD	TBD
4MHz 晶振	TBD	TBD	TBD
1MHz 晶振	TBD	TBD	TBD

注： 1、C1 和 C2 数值仅作参考用。  
2、CL 为晶振制造商的指定负载电容值。

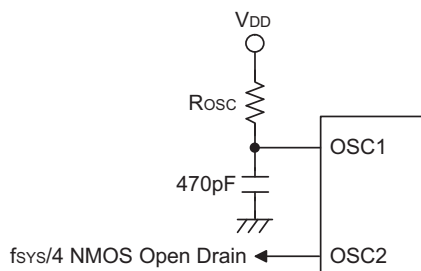
### 晶体振荡器电容的推荐值

谐振器 C1 和 C2 值		
谐振器频率	C1	C2
3.58MHz	TBD	TBD
1 MHz	TBD	TBD
455 MHz	TBD	TBD
注：C1 和 C2 数值仅作参考用		

### 谐振器电容推荐值

#### 外部 RC 振荡器

使用外部 RC 电路作为系统振荡器，需要在 OSC1 和 VDD 之间连接一个阻值在 24kΩ 到 1.5MΩ 之间的电阻，OSC1 与 VSS 之间连接一个电容。产生的系统时钟 4 分频后提供给 OSC2 做输出，用作同步外部元件。由于 OSC2 为 NMOS 开漏输出，当监控 RC 振荡频率时，则需加上拉电阻。虽然此振荡器配置成本较低，但振荡频率会因 VDD、温度和芯片本身的制程而发生变化，因此不适合用于做计时严格或需要精确振荡频率的场合。对于外部电阻 R<sub>OSC</sub> 的阻值，请参考附录章节中典型 RC 振荡器对温度以及对 V<sub>DD</sub> 特性曲线分析。应注意的是系统频率由内部电路和外部电阻 R<sub>OSC</sub> 共同决定，图中显示的外部电容并不会影响振荡器的频率值。



### 外部 RC 振荡器

#### 看门狗振荡器

WDT 振荡器是一个完全独立在芯片上且自由动作的振荡器，它在 5V 时的周期时间典型值为 65μs，且不需外部的器件搭配，由配置选项设置。当单片机进入暂停模式时，系统时钟将停止动作，但 WDT 振荡器继续自由动作且保持有效。因此，为了降低功耗，可在配置选项中将 WDT 振荡器关闭。

#### 看门狗定时器 — WDT

看门狗时钟源可以来自内部振荡器（看门狗振荡器），或者指令时钟（系统时钟的四分频）。由配置选项选择哪个时钟源。看门狗定时器的功能在于防止程序运行故障和程序跳入死循环而导致不可预测的结果。可通过配置选项关闭看门狗定时器。如果看门狗定时器关闭，则所有与 WDT 有关的指令操作都为无效。

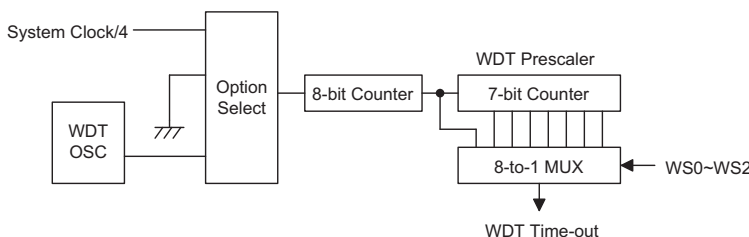
如果使用内部看门狗振荡器，时钟源首先经过 256 分频，在 5V 电压下其溢出周期为 17ms。这个溢出周期会随着 VDD、温度和制作工艺而改变。通过 WDT 预分频器，可以得到更长的溢出周期。设置寄存器 WDTS 的 WS0~WS2 位，可以得到不同的溢出周期。若将 WS0, WS1 和 WS2 位都设置为 1，此时在 5V 的电压下可使溢出周期最大为 2.1s。如果内部看门狗振荡器被禁止，看门狗时钟仍可以来自指令时钟，不同的是当系统进入暂停模式后，指令时钟会停止且 WDT 将失去其保护目的。寄存器 WDTS 的高位元和第三位保留给用户定义标志位使用。

若系统工作在强干扰环境中，建议选用内部 WDT 振荡器，因为 HALT 模式会使系统时钟停止，

使看门狗失去保护功能。

WS2	WS1	WS1	分频系数
0	0	0	1: 1
0	0	1	1: 2
0	1	0	1: 4
0	1	1	1: 8
1	0	0	1: 16
1	0	1	1: 32
1	1	0	1: 64
1	1	1	1: 128

WDTS(09H)寄存器



看门狗定时器

系统在正常运行状态下，WDT 溢出将导致“芯片复位”，且置位 TO 状态标志位。然而，如果系统处于暂停模式，WDT 溢出复位将产生一个“热复位”，只有程序计数器 PC 和堆栈指针复位。清除 WDT 和 WDT 预分频器有外部复位（RES 引脚低电平）、清除看门狗指令或“HALT”指令三种方法。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，由配置选项决定清除看门狗的次数。如果选择“CLR WDT”（一条清除看门狗指令），那么只要执行“CLR WDT”指令就会清除 WDT。如果选择“CLR WDT1”和“CLR WDT2”（两条清除看门狗指令），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

暂停模式仅通过“HALT”指令实现且造成如下结果：

- 系统振荡器停振，但 WDT 看门狗定时器继续工作（如果选择 WDT 振荡器为 WDT 时钟源）。
- RAM 和寄存器内容保持不变。
- WDT 及 WDT 预分频器清除并重新开始计数（如果选择 WDT 振荡器为 WDT 时钟源）。
- 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。

外部复位、系统中断、PA 口下降沿和 WDT 溢出可使系统离开暂停模式。外部复位会使系统初始化，WDT 溢出则会发生“热复位”。这两种方式都会使系统复位，可通过状态寄存器中 TO 和 PDF 位来判断它的复位原因。系统上电或执行清除看门狗的指令，会清零 PDF；执行 HALT 指令，PDF 将被置位。看门狗计数器溢出将会置位 TO 标志并唤醒系统，这种复位会重置程序计数器和堆栈指针，其它都保持原有状态。

PA 口下降沿唤醒和系统中断唤醒可以看成是正常运行的继续。PA 口的每一位都可以由配置选项设置为唤醒功能。如果是由 PA 端口唤醒，程序将在下一条指令处继续执行。如果系统是通过中断唤醒，则有可能发生两种情况。第一种情况是：相关中断除能或者是堆栈已满，则程序会在下一条指令处继续执行。这种情况下，唤醒系统的中断会等到相关中断使能或有堆栈层可以使用之后才执行。第二种情况是：相关中断使能且堆栈未滿，则中断可以马上执行。如果在进入暂停模式之前中断标志位已经被设置为“1”，则相关中断的唤醒功能将无效。不管是哪一种唤醒，一旦有唤醒事件发生，单片机恢复正常运行将需要 1024 个系统时钟周期。唤醒后将插入一个空指令。如果是由中断唤醒系统，则实际的中断子程序执行将延迟一个或数个周期；如果唤醒后接着去执行下一条指令，则它在 1024 个系统周期结束后立刻执行。

为减少功耗，在系统进入暂停模式前，必须处理好输入/输出口状态。

### 复位

单片机共有三种复位方式：

- 正常操作时的  $\overline{\text{RES}}$  复位
- 暂停模式下的  $\overline{\text{RES}}$  复位
- 正常操作时看门狗溢出复位

暂停时看门狗溢出复位与其它种类的复位有些不同，因为看门狗定时器溢出会产生一个“热复位”，只有程序计数器 PC 与堆栈指针 SP 被清 0 及 TO 位被设为 1 外，而系统其它部分都保持原有状态。其它几种复位，寄存器保持不变。当发生相应的复位时，大多寄存器返回到它们原来的初始状态。程序可以通过判断标志位 PDF 和 TO 的值来区分是哪一种复位。

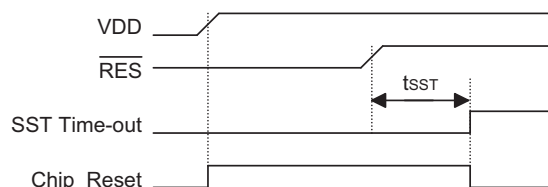
TO	PDF	复位条件
0	0	上电时的 $\overline{\text{RES}}$ 复位
u	u	正常运行时的 $\overline{\text{RES}}$ 复位
0	1	暂停时的 $\overline{\text{RES}}$ 唤醒
1	u	正常运行时的 WDT 溢出复位
1	1	暂停时的 WDT 唤醒

注：“u”代表不改变

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 溢出或者  $\overline{\text{RES}}$  复位）或由暂停状态唤醒时，系统启动定时器（SST）提供了一个额外延迟时间，共 1024 个系统时钟周期。

当系统发生复位时，在复位期间系统会有一个 SST 的延迟周期。暂停模式下的任何唤醒，都会有一个 SST 的系统延迟周期。

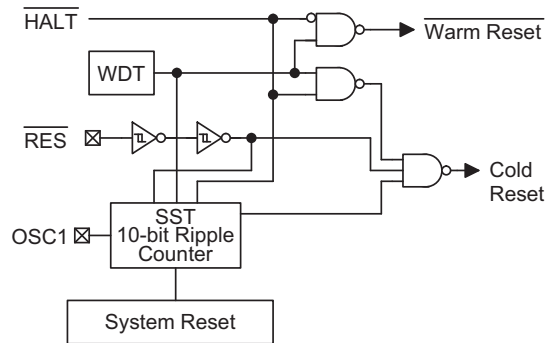
系统复位（包括上电复位、正常操作时 WDT 溢出或者  $\overline{\text{RES}}$  复位）时会额外增加一个加载配置选项的时间。



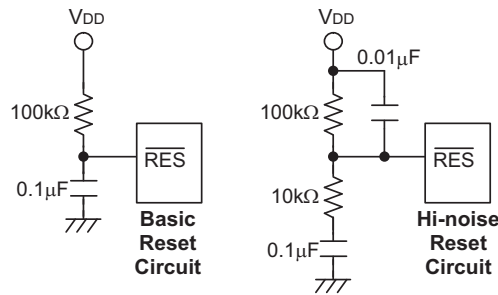
复位时序图

单片机上电复位之后，各功能单元初始化的情形，列于下表。

项目	复位后情况
程序计数器	000H
中断	禁止
预分频器	清除
看门狗定时器	清除，主系统复位后，WDT 重新开始计数
定时/计数器	关闭
输入/输出口	输入模式
堆栈指针	指向堆栈顶端



复位电路结构



复位电路

注：大多数的应用使用上图的基本复位电路，然而当系统在较强干扰的场合工作时，强烈建议使用增强型复位电路。

下表描述了不同复位影响单片机的内部寄存器。

寄存器	复位 (上电复位)	WDT 溢出 (正常运行)	RES (正常运行)	RES (暂停模式)	WDT 溢出 (暂停模式)*
MP0	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
MP1	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
BP	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	000H	000H	000H	000H	000H
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRC	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
ASC0	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
ASC1	---- 1111	---- 1111	---- 1111	---- 1111	---- uuuu
ASC2	--11 1111	--11 1111	--11 1111	--11 1111	--uu uuuu
INTC1	---0 ---0	---0 ---0	---0 ---0	---0 ---0	---u ---u
TMRAH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRAL	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
RCOCCR	0000 1---	0000 1---	0000 1---	0000 1---	uuuu u---
TMRBH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRBL	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
RCOCR	1xxx --00	1xxx --00	1xxx--00	1xxx --00	uuuu --uu

注：“\*”表示热复位。

“u”表示未变化。

“x”表示未确定。

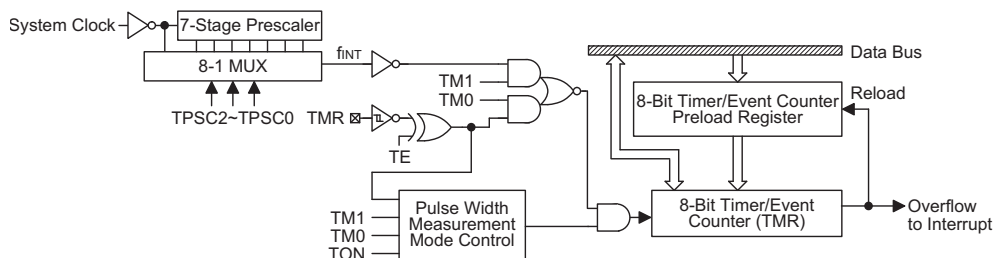
“-”表示未使用。

如果配置选项选择 PA0~PA7 为 RC 输入, PA 数据寄存器和控制寄存器的相关位将无效且读取为 0。

如果配置选项选择 PA0~PA7 为普通输入/输出脚, ASCR0 和 ASCR1 寄存器中位 0~3 将无效, 读取为 0。

## 定时/计数器

HT45R35V 提供 1 个 8 位可编程向上计数的定时/计数器，其时钟源可以来自外部信号输入也可以来自内部系统时钟。如果选择外部时钟源，则允许用户用来计数外部事件、测量时间间隔、测量脉冲宽度或者产生精确时基。使用内部系统时钟允许用户产生一个精确的时基信号。



### 定时/计数器

有两个与定时/计数器相关的寄存器，TMR 和 TMRC。TMR 寄存器有两个物理空间。写数据到 TMR 寄存器实际上是将数据写到 TMR 的预置寄存器，而读取 TMR 寄存器的内容可得到定时/计数器的值。TMRC 是定时/计数器的控制寄存器，此寄存器用来设置定时/计数器的工作模式等。

TM0 和 TM1 位用于定义定时/计数器的工作模式。外部事件计数模式用于计数外部事件，其时钟源为外部 TMR 引脚输入。在定时器模式中，它的时钟源来自  $f_{INT}$ 。外部脉冲宽度测量模式可以测量 TMR 引脚高/低电平的脉冲宽度，其时钟源为内部时钟  $f_{INT}$ 。无论是在外部事件计数模式还是定时器模式下，一旦定时器开始计数，定时/计数器将从当前值开始向上计数到 0FFH。一旦发生溢出，定时器将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位 TF（INTC0 的第五位）。

在脉冲宽度测量模式下，当 TON 与 TE 都为 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）时，定时/计数器就会开始计数直到 TMR 引脚电平恢复，同时 TON 被清零。即使再次发生有效的电平转换，定时/计数器的测量结果仍会被保留下来，直到新的测量开始。换句话说，一次只能测量一个脉冲宽度。如果想继续测量脉宽必须用软件将 TON 置 1。要注意的是，在这种模式下，定时/计数器是跳变触发而不是电平触发。当定时/计数器计数溢出，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）。在外部脉冲宽度测量模式中，脉冲宽度测量完成时，TON 会自动清为 0。但是其它两种工作模式，TON 只能通过指令清零。定时/计数器是唤醒源之一。不管工作在哪种模式下，给 ETI 写 0 都将除能中断。

如果定时/计数器停止计数，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/计数器中。但如果在定时/计数器运行时写入，数据只能写入到预置寄存器中。定时/计数器会继续向上计数直到发生溢出。当读取定时/计数器的值时，计数会被停止，以避免出错。在编程时必须注意，计数停止可能导致计数出错。TMRC 寄存器的位 0~2 用来定义内部时钟预分频级数，定义如下表。

位	符号	功能
0~2	TPSC0~TPSC2	分频系数 TPSC2,TPSC1,TPSC0= 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	TE	定义定时/计数器 TMR 的有效边沿 (0=低到高有效; 1=高到低有效)
4	TON	使能或者除能定时/计数器 (0=除能; 1=使能)
5	—	未使用, 读取为“0”
6 7	TM0 TM1	定义工作模式, TM1, TM0= 01=外部计数模式 (外部时钟) 10=定时模式 (内部时钟) 11=外部脉冲宽度测量模式 00=未使用

### TMRC(0EH)寄存器

#### 外部 RC 振荡转换电路

HT45R35V 提供一个外部 RC 振荡的功能。外部 RC 振荡器包含两个 16 位可编程向上计数的计数器。

当 RCO 位 (RCOCCR 寄存器的第 1 位) 为“1”时, 有四个寄存器与外部 RC 振荡转换电路相关, TMRAL、TMRAH、TMRBL 和 TMRBH。Timer B 的时钟源可以来自外部 RC 振荡器。而 Timer A 的时钟源可以来自系统时钟或者系统时钟 4 分频, 由 RCOCCR 寄存器决定。

有 6 个寄存器与外部 RC 振荡转换电路相关, TMRAH、TMRAL、RCOCCR、TMRBH、TMRBL 和 RCOCCR。定时器 TMRA 时钟源来自内部, 而定时器 TMRB 时钟源来自外部 RC 振荡器。Timer A 还是 Timer B 溢出产生中断, 由 OVB (RCOCCR 寄存器的第 0 位) 位决定。当 Timer A 或 Timer B 溢出时, 中断请求标志位 RCOCF 置位并触发外部 RC 振荡器中断。在外部 RC 振荡器模式下, Timer A 或者 Timer B 溢出时, RCOCON 位被清零并且定时器停止计数。Timer A/Timer B 初始值只有在写 TMRAH/TMRBH 时才会被改变, 同时读取 TMRAH/TMRBH 会得到 Timer A/Timer B 的值。写数据到 TMRAL/TMRBL 只能将数据写入到低字节缓冲器内。然而, 将数据写到 TMRAH/TMRBH 会把指定的数据和低字节缓冲器的内容分别写到 Timer A/Timer B (16 位) 寄存器中。写值到 TMRAH/TMRBH 将会改变 Timer A/Timer B 的值, 而写值到 TMRAL/TMRBL 不会改变 Timer A/Timer B 的值。

读取 TMRAH/TMRBH 的值时, 会将 TMRAL/TMRBL 锁存至低字节缓冲器, 以避免时序出错。读取 TMRAL/TMRBL 的值将返回低字节缓冲器的内容。因此, Timer A/Timer B 的低字节寄存器无法直接读取。必须先读取 TMRAH/TMRBH 的值, 以保证 Timer A/Timer B 的低字节内容锁存至低字节缓冲器。

Timer B 时钟来源于外部电阻和电容组成的振荡器。RCOCCR 寄存器中的 RCOM0、RCOM1 和 RCOM2 位用来定义 Timer A 的时钟源。建议 Timer A 的时钟源使用系统时钟或者指令时钟。如果

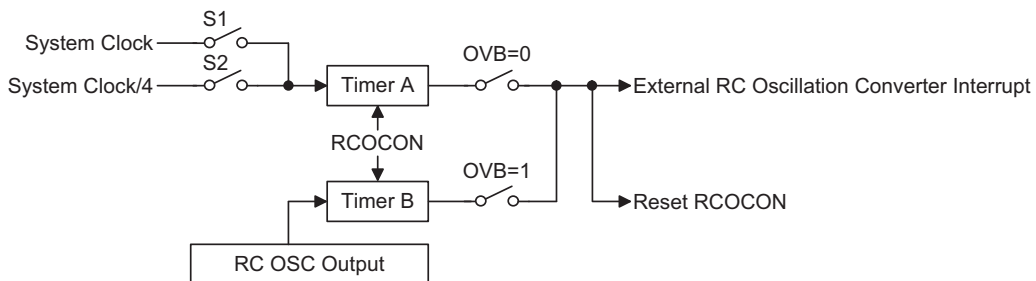
RCOCON 位 (RCOCCR 寄存器的第 4 位) 被置位, Timer A/Timer B 将开始计数直到 Timer A 或者 Timer B 溢出, 并且置位定时/计数器中断请求标志位 RCOCF (INTC1 的第 4 位)。此时, Timer A 和 Timer B 停止计数, RCOCON 位自动清零。如果 RCOCON 位被置位, TMRAH, TMRAL, TMRBH 和 TMRBL 将不可读写。

位	符号	功能
0~2	—	未使用, 读取为 0
3	—	未使用, 可读写
4	RCOCON	打开/关闭外部 RC 振荡转换电路 (0=打开; 1=关闭)
5	RCOM0	定义 Timer A 时钟源, RCOM2,RCOM1,RCOM0= 000=系统时钟 001=系统时钟/4 010=未使用
6	RCOM1	011=未使用
7	RCOM2	100=未使用 101=未使用 110=未使用 111=未使用

RCOCCR(22H)寄存器

位	符号	功能
0	OVB	在外部 RC 振荡器模式下, 该位用于定义定时/计数器中断来自 Timer A 溢出还是 Timer B 溢出。 (0=Timer A 溢出; 1=Timer B 溢出)
1	RCO	定义外部 RC 振荡器模式 (0=关闭外部 RC 振荡转换电路; 1=打开外部 RC 振荡转换电路)
2~3	—	未使用, 读取为 0
4~7	RW	用户自定义的 4 位可读/写寄存器

RCOCR(25H)寄存器



外部 RC 振荡转换电路

外部 RC 振荡模式范例程序 — Timer A 溢出：

```

clr  RCOCCR           ;Timer/Event Counter 0 interrupt vector
mov  a,00000010b     ;enable external RC oscillation mode and set Timer A overflow
mov  RCOCR,a
clr  intc1.4         ;clear External RC Oscillation Converter interrupt request flag
mov  a,low(65536 - 1000) ;Give timer A initial value
mov  tmral,a         ;Timer A count 1000 time and then overflow
mov  a,high(65536 - 1000)
mov  tmrah,a
mov  a,00h           ;Give timer B initial value
mov  tmrbl,a
mov  a,00110000b    ;Timer A clock source=fSYS/4 and timer on
mov  RCOCCR,a
p10:
clr  wdt
snz  intc1.4        ;Polling External RC oscillation Converter interrupt request flag
jmp  p10
clr  intc1.4        ;Cleat Erernal RC oscillation Converter interrupt request flag
                                ;Program continue

```

**模拟开关**

HT45R35V 提供 12 通道的模拟开关 RC1~RC12，与模拟开关相关的三个寄存器是：ASCR0、ASCR1 和 ASCR2。

如果配置选项选择 PA0~PA3 作为普通输入/输出口，则 ASCR0 寄存器的相关位 0~3 将无效，并读取为 0。

位	符号	功能
0	AS1ON	定义 RC1 模拟开关的开或关。AS1ON=0=RC1 通道打开，且与下拉电阻断开 1=RC1 通道关闭，且是否与下拉电阻相连由 ASPLON0 寄存器而定
1	AS2ON	定义 RC2 模拟开关的开或关。AS2ON=0=RC2 通道打开，且与下拉电阻断开 1=RC2 通道关闭，且是否与下拉电阻相连由 ASPLON0 寄存器而定
2	AS3ON	定义 RC3 模拟开关的开或关。AS3ON=0=RC3 通道打开，且与下拉电阻断开 1=RC3 通道关闭，且是否与下拉电阻相连由 ASPLON1 寄存器而定
3	AS4ON	定义 RC4 模拟开关的开或关。AS4ON=0=RC4 通道打开，且与下拉电阻断开 1=RC4 通道关闭，且是否与下拉电阻相连由 ASPLON1 寄存器而定
4	AS5ON	定义 RC5 模拟开关的开或关。AS5ON=0=RC5 通道打开，且与下拉电阻断开 1=RC5 通道关闭，且是否与下拉电阻相连由 ASPLON2 寄存器而定
5	AS6ON	定义 RC6 模拟开关的开或关。AS6ON=0=RC6 通道打开，且与下拉电阻断开 1=RC6 通道关闭，且是否与下拉电阻相连由 ASPLON2 寄存器而定
6	AS7ON	定义 RC7 模拟开关的开或关。AS7ON=0=RC7 通道打开，且与下拉电阻断开 1=RC7 通道关闭，且是否与下拉电阻相连由 ASPLON3 寄存器而定
7	AS8ON	定义 RC8 模拟开关的开或关。AS8ON=0=RC8 通道打开，且与下拉电阻断开 1=RC8 通道关闭，且是否与下拉电阻相连由 ASPLON3 寄存器而定

**ASCR0(1AH)寄存器**

如果配置选项选择 PA4~PA7 作为普通输入/输出口，则 ASCR1 寄存器的相关位 0~3 将无效，并读取为 0。

位	符号	功能
0	AS9ON	定义 RC9 模拟开关的开或关。AS9ON=0=RC9 通道打开，且与下拉电阻断开 1=RC9 通道关闭，且是否与下拉电阻相连由 ASPLON4 寄存器而定
1	AS10ON	定义 RC10 模拟开关的开或关。AS2ON=0=RC10 通道打开，且与下拉电阻断开 1=RC10 通道关闭，且是否与下拉电阻相连由 ASPLON4 寄存器而定
2	AS11ON	定义 RC11 模拟开关的开或关。AS11ON=0=RC11 通道打开，且与下拉电阻断开 1=RC11 通道关闭，且是否与下拉电阻相连由 ASPLON5 寄存器而定
3	AS12ON	定义 RC12 模拟开关的开或关。AS12ON=0=RC12 通道打开，且与下拉电阻断开 1=RC13 通道关闭，且是否与下拉电阻相连由 ASPLON5 寄存器而定
4~7	—	未使用，读取为 0

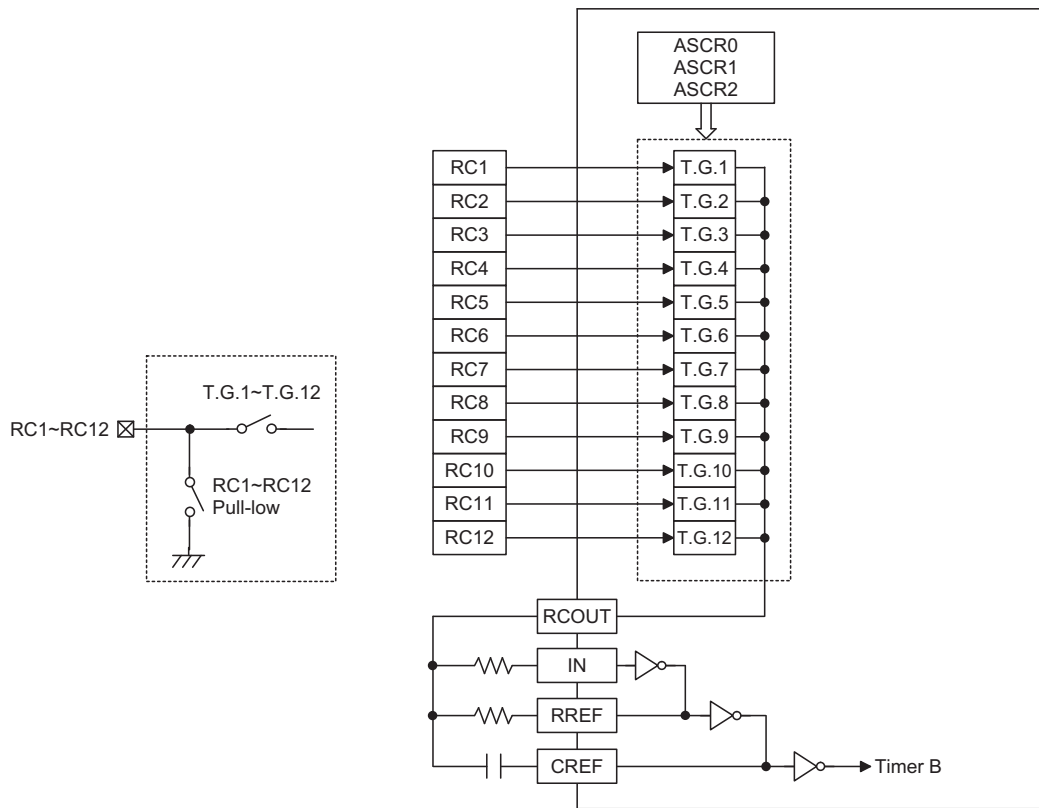
### ASCR1(1BH)寄存器

如果配置选项选择 PA0~PA7 作为普通输入/输出口，则 ASCR2 寄存器的相关位，位 0，位 1，位 4 和位 5 必须被设为 0 以除能 RC1/RC2,RC3/RC4,RC9/RC10 或 RC11/RC12 的下拉电阻。这些位由软件设为 0 或 1。

位	符号	功能
0	ASPLON0	定义 RC1 和 RC2 通道是否与下拉电阻相连。ASPLON0=0=RC1 和 RC2 通道与下拉电阻断开 1=RC1 和 RC2 是否与下拉电阻连接由 RC1 和 RC2 通道的开或关而定 当 ASPLON0=1 且 AS1ON/AS2ON 关闭时，RC1/RC2 通道与下拉电阻相连
1	ASPLON1	定义 RC3 和 RC4 通道是否与下拉电阻相连。ASPLON1=0=RC3 和 RC4 通道与下拉电阻断开 1=RC3 和 RC4 是否与下拉电阻连接由 RC3 和 RC4 通道的开或关而定 当 ASPLON1=1 且 AS3ON/AS4ON 关闭时，RC3/RC4 通道与下拉电阻相连
2	ASPLON2	定义 RC5 和 RC6 通道是否与下拉电阻相连。ASPLON2=0=RC5 和 RC6 通道与下拉电阻断开 1=RC5 和 RC6 是否与下拉电阻连接由 RC5 和 RC6 通道的开或关而定 当 ASPLON2=1 且 AS5ON/AS6ON 关闭时，RC5/RC6 通道与下拉电阻相连
3	ASPLON3	定义 RC7 和 RC8 通道是否与下拉电阻相连。ASPLON3=0=RC7 和 RC8 通道与下拉电阻断开 1=RC7 和 RC8 是否与下拉电阻连接由 RC7 和 RC8 通道的开或关而定 当 ASPLON3=1 且 AS7ON/AS8ON 关闭时，RC7/RC8 通道与下拉电阻相连

4	ASPLON4	定义 RC9 和 RC10 通道是否与下拉电阻相连。ASPLON4=0=RC9 和 RC10 通道与下拉电阻断开 1=RC9 和 RC10 是否与下拉电阻连接由 RC9 和 RC10 通道的开或关而定 当 ASPLON4=1 且 AS9ON/AS10ON 关闭时，RC9/RC10 通道与下拉电阻相连
5	ASPLON5	定义 RC11 和 RC12 通道是否与下拉电阻相连。ASPLON5=0=RC11 和 RC12 通道与下拉电阻断开 1=RC11 和 RC12 是否与下拉电阻连接由 RC11,RC12 通道的开或关而定 当 ASPLON5=1 且 AS11ON/AS12ON 关闭时，RC11/RC12 通道与下拉电阻相连
6~7	—	未使用，读取为 0

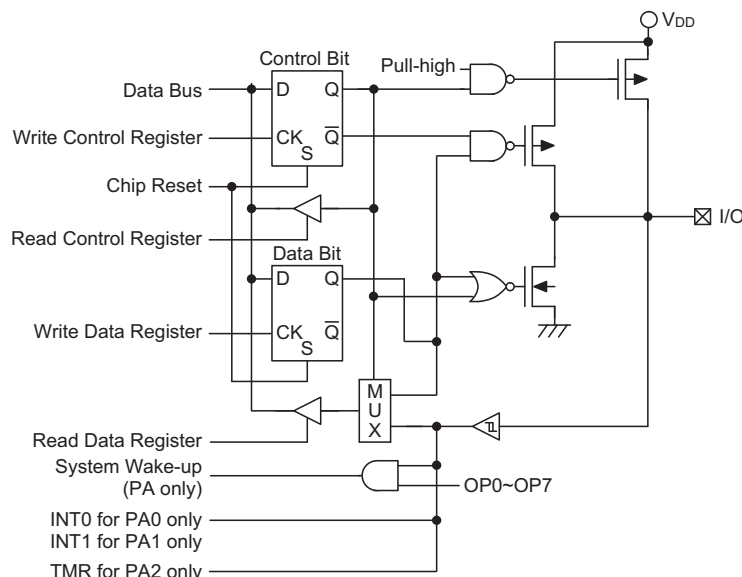
ASCR2(1CH)寄存器



模拟开关

## 输入/输出口

HT45R35V 提供 9 个双向输入/输出口，标示为 PA 和 PB。所有端口都可进行输入/输出操作。作为输入操作时，端口没有锁存功能，也就是输入数据必须在指令“MOV A, [m]” T2 上升沿来前准备好。对于输出操作，端口有锁存功能，而且持续到输出锁存被重写。



## 输入/输出口

每个输入/输出口都有自己的控制寄存器（PAC、PBC）用来控制输入/输出状态。利用此控制寄存器，每一个 CMOS 输出或者施密特触发输入，均可利用软件控制方式加以动态的重新设置。若作为输入时，则对应的控制寄存器位必须设置为“1”。所有输入/输出端口引脚都各自对应于输入/输出端口控制寄存器的某一位。如果控制寄存器位设定为“1”，这时程序指令可以直接读出输入引脚的逻辑状态。如果引脚的控制寄存器位被设定为“0”，则读取的是内部锁存器的值。后者可能会在“读-改-写”指令中发生。

输入/输出口作为输出时，只能采用 CMOS 输出。

系统复位之后，所有输入/输出引脚默认为输入状态，且输入/输出数据及端口控制寄存器都将被设为高电平（上拉电阻选项选择带上拉）或浮空状态。每一个输入/输出锁存位都能用“SET [m].i”及“CLR [m].i”指令置位或清零。

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

PA 的每一个口都具有唤醒系统的能力。

PA 和 PB 所有的输入/输出口都有上拉电阻选项。一旦配置选项选择了上拉电阻选项，则所有输入/输出口都将与上拉电阻相连接。否则，上拉电阻无效。必须注意在输入模式下，如果输入/输出口不选择上拉电阻，输入/输出口将处于浮空状态。

PA0, PA1, PA2 分别与 INT0, INT1 和 TMR 引脚共用。PA0~PA3 和 PA4~PA7 分别与 RC1~RC4 和 RC9~RC12 共用引脚。如果配置选项选择 PA0~PA7 作为 RC 输入脚，则对应的 PA 数据寄存器和 P 控制寄存器的相关位无效。

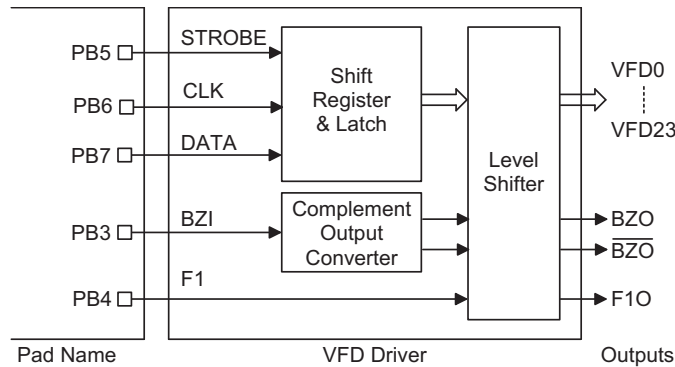
建议软件中将未使用或没有外接的输入输出口设置为输出模式，以避免这些端口在输入浮空状态下造成功率率的损耗。

## VFD 驱动器

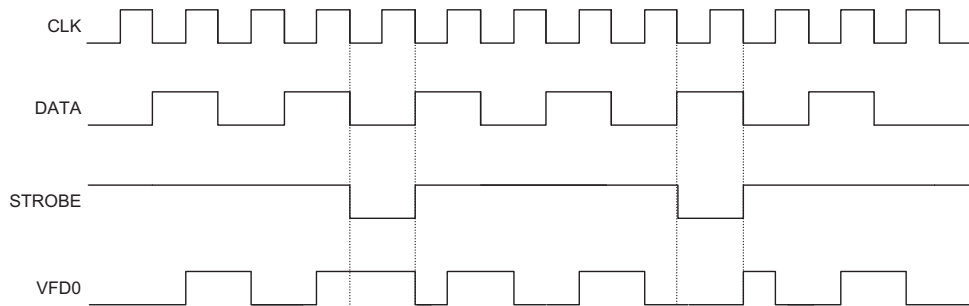
该单片机具有 VFD 功能，用来驱动 VFD 面板的高压灯丝和蜂鸣器。单片机使用串行通讯接口将显示数据传送到 VFD 驱动器中的 24 位移位寄存器。VFD 驱动器将移位寄存器中的数据转成 VFD 驱动信号，并且提供必要的电平转换。数据只能从单片机单向传送到 VFD 驱动器，反之则不行。

### VFD 接口

下图为单片机与 VFD 驱动器之间的五线接口方框图。



VFD 驱动器



VFD 显示控制时序图

单片机与 VFD 驱动器之间的数据传输由三线接口 (CLK, DATA 和 STROBE) 来控制。数据为单向传输方式，且单片机的输入/输出口必选设置为输出。

蜂鸣器的控制输入 BZI 经由 VFD 驱动的转换器转变为对互补输出 BZO 和  $\overline{BZO}$ ，且此对互补输出也将通过电平转换器转换为高电压输出。VFD 驱动器的灯丝控制输入 F1 也将转换为高电压输出 F1O，用于灯丝的开/关控制。

### 24 位移位寄存器/锁存器

单片机与 VFD 驱动器间使用串行数据传输，需首先将数据写入到位于 VFD 驱动的 24 位移位寄存器中。这 24 位中 VFD0~VFD15 口用于控制 VFD 面板的段，VFD16~VFD23 口用于控制 VFD 面板的栅格，控制方式描述如下：

- 通过“DATA”和“CLK”口将数据传送至内部的 24 位移位寄存器，数据在时钟上升沿写入移位寄存器，此数据为 VFD0~VFD23 口要输出的显示数据。只有当 STROBE 口为高时，VFD 输

出才发生改变。而当 STROBE 口为低时，只修改移位寄存器中的数据，VFD 输出保持不变。

- 通过“STROBE”口将移位寄存器中的数据锁存到输出口 VDF0~VDF23，当 STROBE 口为高时，移位寄存器数据将会锁存在 VFD 口上。注意，STROBE 口为电平触发而非边沿触发。

下表为 24 位移位寄存器/锁存器功能表：

Clock	Strobe	Data	VFD0	VFDn
↑	0	×	不变	不变
↑	1	0	0	VFDn-1
↑	1	1	1	VFDn-1
↓	1	1	不变	不变

注意：“×”表示不影响。

“VFDn”表示 VFD1 ~VFD23。

### 编程注意事项

上电之后，所有的输入/输出口将自动设置为输入。然而当 PB2~PB5 用于 VFD 驱动时，单片机上电之后必须将他们设置输出。而 VFD 接口控制口设置为输入将导致错误的 VFD 显示操作。建议上电初始化时，配置选项设置 VFD 控制口带上拉电阻，使得这些引脚保持固定的高电平，直到这些口设置为输出。

### 编程应用范例

以下例子说明了 VFD 显示数据如何通过单片机编程。

```

strobe    equ pb.5
clk       equ pb.6
data     equ pb.7
data_2_register:          ; send data to vfd driver
    mov    a,024d          ; shift register counter
    mov    count,a
    clr    strobe          ; strobe = 0
data_2_register_1:
    clr    clk              ; clk = 0
    set    data             ; data = 1
    snz    vfd_grid.7
    clr    data             ; data = 0
    rlc    vfd_seg1         ; shift data to vfd[7:0]
    rlc    vfd_seg2         ; shift data to vfd[15:8]
    rlc    vfd_grid         ; shift data to vfd[22:16]
    set    clk              ; clk = 1 (rising edge)
    sdz    count
    jmp    data_2_register_1
    set    strobe           ; strobe = 1, vfd output
    ret
    
```

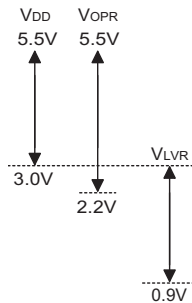
## 低电压复位 — LVR

为了监控器件的工作电压，该单片机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。

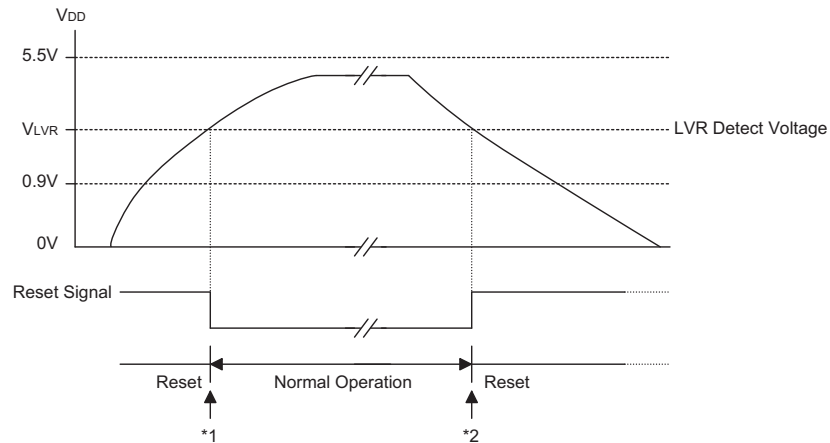
LVR 功能说明如下：

- 低电压 ( $0.9V \sim V_{LVR}$ ) 的状态必须持续  $t_{LVR}$  以上。如果低电压的状态没有持续  $t_{LVR}$  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。

以下为  $V_{DD}$  与  $V_{LVR}$  的关系图



注：VOPR 是在系统时钟为 4MHz 时，使得芯片正常运行的电压范围



### 低电压复位

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

\*2、因为低电压状态必须保持  $t_{LVR}$  以上，因此进入复位模式就要有  $t_{LVR}$  的延迟。

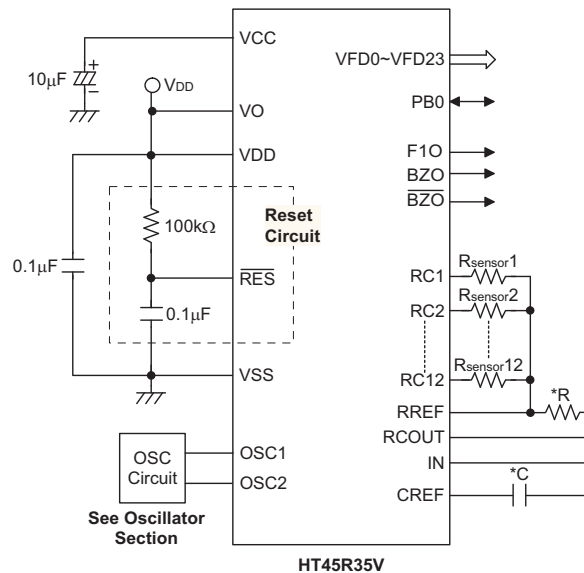
### 配置选项

下表列出了单片机所有配置选项。所有选项必须正确定义，以保证系统正常运行。

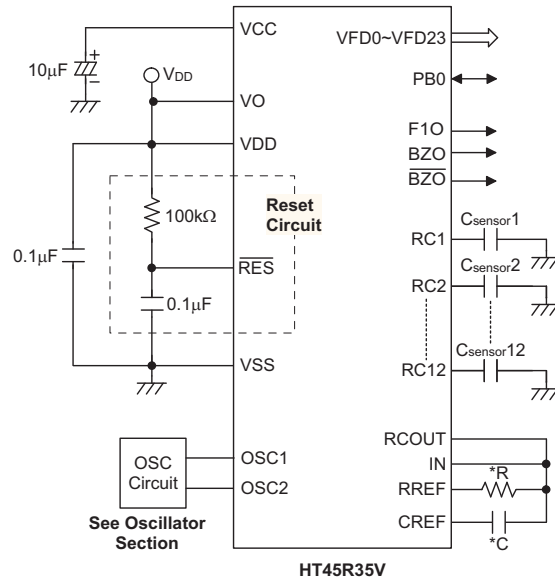
编号	功能	描述
1	PA0~PA7 唤醒(位选)	带或不带唤醒
2	PA0~PA7, PB0 和 PB3~PB7 上拉电阻(位选)	带或不带上拉电阻
3	WDT 时钟源	WDT OSC 或 $f_{SYS}/4$
4	WDT	使能/禁止
5	CLR WDT	1 条或者 2 条指令
6	LVR	使能/禁止
7	OSC	晶体振荡器或者 RC 振荡器
8	INT0 触发边沿	禁止、上升沿、下降沿或两者皆可触发
9	INT1 触发边沿	禁止、上升沿、下降沿或两者皆可触发
10	输入/输出或 RC 连接引脚	PA0 或 RC1, PA1 或 RC2, PA2 或 RC3 PA3 或 RC4, PA4 或 RC9, PA5 或 RC10 PA6 或 RC11, PA7 或 RC12

### 应用电路

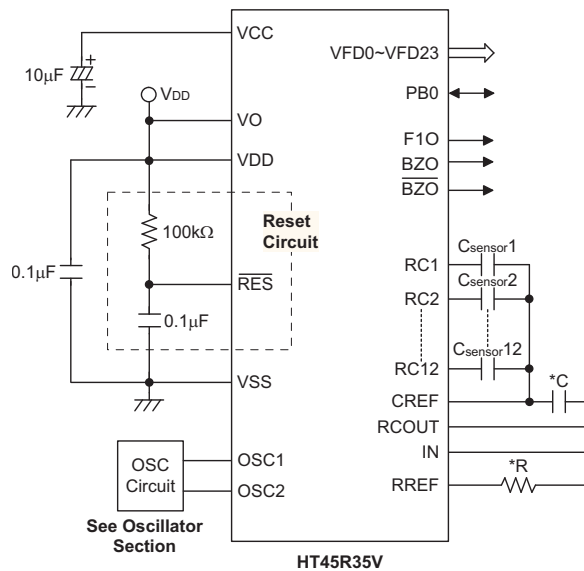
#### R - F 型应用电路



C - F 型应用电路 1



C - F 型应用电路 2



- 注: 1、电阻“\*R”和电容“\*C”需要参考 RC 振荡频率。  
 2、 $R_{\text{sensor}1} \sim R_{\text{sensor}12}$  为电阻传感器。  
 3、 $C_{\text{sensor}1} \sim C_{\text{sensor}12}$  为电容传感器。

## 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 HOLTEK 单片机中，提供了丰富且灵活的指令，共超过六十条，程序设计者可以事半功倍地实现他们的应用。

为了更加容易理解各种各样的指令码，接下来按功能分组介绍它们。

### 指令周期

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

### 数据的传送

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

### 算术运算

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

### 逻辑和移位运算

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

### 分支和控制转换

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

### **位运算**

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

### **查表运算**

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

### **其它运算**

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

## 指令集概要

下表中说明了按功能分类的指令集，用户可以将该表作为基本的指令参考。

### 惯例

x: 立即数                      m: 数据存储器地址  
A: 累加器                      i: 第 0~7 位                      addr: 程序存储器地址

助记符	说明	指令周期	影响标志位
<b>算术运算</b>			
ADD    A,[m]	ACC 与数据存储器相加，结果放入 ACC	1	Z,C,AC,OV
ADDM   A,[m]	ACC 与数据存储器相加，结果放入数据存储器	1 <sup>注</sup>	Z,C,AC,OV
ADD    A, x	ACC 与立即数相加，结果放入 ACC	1	Z,C,AC,OV
ADC    A,[m]	ACC 与数据存储器、进位标志相加，结果放入 ACC	1	Z,C,AC,OV
ADCM   A,[m]	ACC 与数据存储器、进位标志相加，结果放入数据存储器	1 <sup>注</sup>	Z,C,AC,OV
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>注</sup>	Z,C,AC,OV
SBC    A,[m]	ACC 与数据存储器、进位标志的反相减，结果放入 ACC	1	Z,C,AC,OV
SBCM   A,[m]	ACC 与数据存储器、进位标志相减，结果放入数据存储器	1 <sup>注</sup>	Z,C,AC,OV
DAA    [m]	将加法运算中放入 ACC 的值调整为十进制数，并将结果放入数据存储器	1 <sup>注</sup>	C
<b>逻辑运算</b>			
AND    A,[m]	ACC 与数据存储器做“与”运算，结果放入 ACC	1	Z
OR     A,[m]	ACC 与数据存储器做“或”运算，结果放入 ACC	1	Z
XOR    A,[m]	ACC 与数据存储器做“异或”运算，结果放入 ACC	1	Z
ANDM   A,[m]	ACC 与数据存储器做“与”运算，结果放入数据存储器	1 <sup>注</sup>	Z
ORM    A,[m]	ACC 与数据存储器做“或”运算，结果放入数据存储器	1 <sup>注</sup>	Z
XORM   A,[m]	ACC 与数据存储器做“异或”运算，结果放入数据存储器	1 <sup>注</sup>	Z
AND    A, x	ACC 与立即数做“与”运算，结果放入 ACC	1	Z
OR     A, x	ACC 与立即数做“或”运算，结果放入 ACC	1	Z
XOR    A, x	ACC 与立即数做“异或”运算，结果放入 ACC	1	Z
CPL    [m]	对数据存储器取反，结果放入数据存储器	1 <sup>注</sup>	Z
CPLA   [m]	对数据存储器取反，结果放入 ACC	1	Z
<b>递增和递减</b>			
INCA   [m]	递增数据存储器，结果放入 ACC	1	Z
INC    [m]	递增数据存储器，结果放入数据存储器	1 <sup>注</sup>	Z
DECA   [m]	递减数据存储器，结果放入 ACC	1	Z
DEC    [m]	递减数据存储器，结果放入数据存储器	1 <sup>注</sup>	Z
<b>移位</b>			
RRA    [m]	数据存储器右移一位，结果放入 ACC	1	无
RR     [m]	数据存储器右移一位，结果放入数据存储器	1 <sup>注</sup>	无

助记符		说明	指令周期	影响标志位
RRCA	[m]	带进位将数据存储器右移一位, 结果放入 ACC	1	C
RRC	[m]	带进位将数据存储器右移一位, 结果放入数据存储器	1 <sup>注</sup>	C
RLA	[m]	数据存储器左移一位, 结果放入 ACC	1	无
RL	[m]	数据存储器左移一位, 结果放入数据存储器	1 <sup>注</sup>	无
RLCA	[m]	带进位将数据存储器左移一位, 结果放入 ACC	1	C
RLC	[m]	带进位将数据存储器左移一位, 结果放入数据存储器	1 <sup>注</sup>	C
<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>注</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>注</sup>	无
SET	[m].i	置位数据存储器的位	1 <sup>注</sup>	无
<b>转移</b>				
JMP	addr	无条件跳转	2	无
SZ	[m]	如果数据存储器为零, 则跳过下一条指令	1 <sup>注</sup>	无
SZA	[m]	数据存储器送至 ACC, 如果内容为零, 则跳过下一条指令	1 <sup>注</sup>	无
SZ	[m].i	如果数据存储器的第 i 位为零, 则跳过下一条指令	1 <sup>注</sup>	无
SNZ	[m].i	如果数据存储器的第 i 位不为零, 则跳过下一条指令	1 <sup>注</sup>	无
SIZ	[m]	递增数据存储器, 如果结果为零, 则跳过下一条指令	1 <sup>注</sup>	无
SDZ	[m]	递减数据存储器, 如果结果为零, 则跳过下一条指令	1 <sup>注</sup>	无
SIZA	[m]	递增数据存储器, 将结果放入 ACC, 如果结果为零, 则跳过下一条指令	1 <sup>注</sup>	无
SDZA	[m]	递减数据存储器, 将结果放入 ACC, 如果结果为零, 则跳过下一条指令	1 <sup>注</sup>	无
CALL	addr	子程序调用	2	无
RET		从子程序返回	2	无
RET	A, x	从子程序返回, 并将立即数放入 ACC	2	无
RETI		从中断返回	2	无
<b>查表</b>				
TABRDC	[m]	读取当前页的 ROM 内容, 并送至数据存储器 and TBLH	2 <sup>注</sup>	无
TABRDL	[m]	读取最后页的 ROM 内容, 并送至数据存储器 and TBLH	2 <sup>注</sup>	无
<b>其它指令</b>				
NOP		空指令	1	无
CLR	[m]	清除数据存储器	1 <sup>注</sup>	无
SET	[m]	置位数据存储器	1 <sup>注</sup>	无
CLR	WDT	清除看门狗定时器	1	TO,PDF

助记符	说明	指令周期	影响标志位
CLR WDT1	预清除看门狗定时器	1	TO,PDF
CLR WDT2	预清除看门狗定时器	1	TO,PDF
SWAP [m]	交换数据存储器的高低字节，结果放入数据存储器	1 <sup>注</sup>	无
SWAPA [m]	交换数据存储器的高低字节，结果放入 ACC	1	无
HALT	进入暂停模式	1	TO,PDF

**注：** 1、对跳转指令而言，如果比较的结果牵涉到跳转即需 2 个周期，如果没有发生跳转，则只需一个周期。  
 2、任何指令若要改变 PCL 的内容将需要 2 个周期来执行。  
 3、对于“CLR WDT1”或“CLR WDT2”指令而言，TO 和 PDF 标志位也许会受执行结果影响，“CLR WDT1”和“CLR WDT2”被连续地执行后，TO 和 PDF 标志位会被清除，否则 TO 和 PDF 标志位保持不变。

### 指令定义

**ADC A, [m]** Add data memory and carry to the accumulator  
 说明：将指定的数据存储器、累加器内容以及进位标志相加，结果存放到累加器。  
 运算过程： $ACC \leftarrow ACC + [m] + C$   
 影响标志位：OV、Z、AC、C

**ADCM A, [m]** Add the accumulator and carry to the accumulator  
 说明：将指定的数据存储器、累加器内容和进位标志位相加，结果存放到指定的数据存储器。  
 运算过程： $[m] \leftarrow ACC + [m] + C$   
 影响标志位：OV、Z、AC、C

**ADD A, [m]** Add data memory to the accumulator  
 说明：将指定的数据存储器内容和累加器内容相加，结果存放到累加器。  
 运算过程： $ACC \leftarrow ACC + [m]$   
 影响标志位：OV、Z、AC、C

**ADD A, x** Add immediate data to the accumulator  
 说明：将累加器和立即数相加，结果存放到累加器。  
 运算过程： $ACC \leftarrow ACC + x$   
 影响标志位：OV、Z、AC、C

**ADDM A, [m]** Add the accumulator to the data memory  
 说明：将指定的数据存储器内容和累加器内容相加，结果存放到指定的数据存储器。  
 运算过程： $[m] \leftarrow ACC + [m]$   
 影响标志位：OV、Z、AC、C

**AND A, [m]** Logical AND accumulator with data memory  
 说明：将累加器中的数据和指定数据存储器内容做逻辑与，结果存放到累加器。  
 运算过程： $ACC \leftarrow ACC \text{ “AND” } [m]$   
 影响标志位：Z

<b>AND</b>	<b>A, x</b>	Logical AND immediate data to the accumulator
说明:		将累加器中的数据 and 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
<b>ANDM</b>	<b>A, [m]</b>	Logical AND data memory with the accumulator
说明:		将指定数据存储器内容和累加器中的数据做逻辑与, 结果存放到数据存储器。
运算过程:		$[m] \leftarrow ACC \text{ "AND" } [m]$
影响标志位:		Z
<b>CALL</b>	<b>addr</b>	Subroutine call
说明:		无条件的调用指定地址的子程序, 此时程序计数器先加 1 获得下一个要执行的指令地址并压入堆栈, 接着载入指定地址并从新地址执行程序。由于指令需要额外的运算, 所以此指令为 2 个周期。
运算过程:		$Stack \leftarrow Program\ Counter + 1$ $Program\ Counter \leftarrow addr$
影响标志位:		无
<b>CLR</b>	<b>[m]</b>	Clear data memory
说明:		将指定数据存储器的内容清零。
运算过程:		$[m] \leftarrow 00H$
影响标志位:		无
<b>CLR</b>	<b>[m].i</b>	Clear bit of data memory
说明:		将指定数据存储器的 i 位内容清零。
运算过程:		$[m].i \leftarrow 0$
影响标志位:		无
<b>CLR</b>	<b>WDT</b>	Clear Watchdog Timer
说明:		WDT 计数器、暂停标志位 PDF 和看门狗溢出标志位 TO 清零。
运算过程:		$WDT \leftarrow 00H$ $PDF \ \& \ TO \leftarrow 0$
影响标志位:		TO、PDF
<b>CLR</b>	<b>WDT1</b>	Preclear Watchdog Timer
说明:		PDF 和 TO 标志位都被清 0。必须配合 CLR WDT2 一起使用清除 WDT 计时器。当程序仅执行 CLR WDT1, 而没有执行 CLR WDT2 时, PDF 与 TO 保留原状态不变。
运算过程:		$WDT \leftarrow 00H$ $PDF \ \& \ TO \leftarrow 0$
影响标志位:		TO、PDF
<b>CLR</b>	<b>WDT2</b>	Preclear Watchdog Timer
说明:		PDF 和 TO 标志位都被清 0。必须配合 CLR WDT1 一起使用清除 WDT 计时器。

		当程序仅执行 CLR WDT2, 而没有执行 CLR WDT1 时, PDF 与 TO 保留原状态不变。
运算过程:		WDT $\leftarrow$ 00H PDF & TO $\leftarrow$ 0
影响标志位:		TO、PDF
<b>CPL</b> [m]		Complement data memory
说明:		将指定数据存储器中的每一位取逻辑反, 相当于从 1 变 0 或从 0 变 1。
运算过程:		$[m] \leftarrow [\bar{m}]$
影响标志位:		Z
<b>CPLA</b> [m]		Complement data memory
说明:		将指定数据存储器中的每一位取逻辑反, 相当于从 1 变 0 或从 0 变 1, 结果被存放回累加器且数据寄存器的内容保持不变。
运算过程:		ACC $\leftarrow [\bar{m}]$
影响标志位:		Z
<b>DAA</b> [m]		Decimal-Adjust accumulator for addition
说明:		将累加器中的内容转换为 BCD (二进制转成十进制) 码。如果低四位的值大于“9”或 AC=1, 那么 BCD 调整就执行对原值加“6”, 否则原值保持不变; 如果高四位的值大于“9”或 C=1, 那么 BCD 调整就执行对原值加“6”。BCD 转换实质上是根据累加器和标志位执行 00H, 06H, 60H 或 66H 的加法运算, 结果存放于数据存储器。只有进位标志位 C 受影响, 用来指示原始 BCD 的和是否大于 100, 并可以进行双精度十进制数的加法运算。
操作:		$[m] \leftarrow ACC+00H$ 或 $[m] \leftarrow ACC+06H$ $[m] \leftarrow ACC+60H$ 或 $[m] \leftarrow ACC+66H$
影响标志位:		C
<b>DEC</b> [m]		Decrement data memory
说明:		将指定数据存储器的内容减 1。
运算过程:		$[m] \leftarrow [m]-1$
影响标志位:		Z
<b>DECA</b> [m]		Decrement data memory and place result in the accumulator
说明:		将指定数据存储器的内容减 1, 把结果存放回累加器并保持指定数据存储器的内容不变。
运算过程:		ACC $\leftarrow [m]-1$
影响标志位:		Z
<b>HALT</b>		Enter power down mode
说明:		此指令终止程序执行并关掉系统时钟, RAM 和寄存器的内容保持原状态, WDT 计数器和分频器被清“0”, 暂停标志位 PDF 被置位 1, WDT 溢出标志位 TO 被清 0。
运算过程:		TO $\leftarrow$ 0

		PDF $\leftarrow$ 1
影响标志位:		TO、PDF
<b>INC</b> [m]		Increment data memory
说明:		将指定数据存储器的内容加 1。
运算过程:		$[m] \leftarrow [m]+1$
影响标志位:		Z
<b>INCA</b> [m]		Increment data memory and place result in the accumulator
说明:		将指定数据存储器的内容加 1，结果存放回累加器并保持指定的数据存储器内容不变。
运算过程:		$ACC \leftarrow [m]+1$
影响标志位:		Z
<b>JMP</b> addr		Directly jump
说明:		程序计数器的内容无条件地由被指定的地址取代，程序由新的地址继续执行。当新的地址被加载时，必须插入一个空指令周期，所以此指令为 2 个周期的指令。
运算过程:		$PC \leftarrow \text{addr}$
影响标志位:		无
<b>MOV</b> A, [m]		Move data memory to the accumulator
说明:		将指定数据存储器的内容复制到累加器。
运算过程:		$ACC \leftarrow [m]$
影响标志位:		无
<b>MOV</b> A, x		Move immediate data to the accumulator
说明:		将 8 位立即数载入累加器。
运算过程:		$ACC \leftarrow x$
影响标志位:		无
<b>MOV</b> [m], A		Move the accumulator data to memory
说明:		将累加器的内容复制到指定的数据存储器。
运算过程:		$[m] \leftarrow ACC$
影响标志位:		无
<b>NOP</b>		No operation
说明:		空操作，顺序执行下一条指令。
运算过程:		$PC \leftarrow PC+1$
影响标志位:		无
<b>OR</b> A, [m]		Logical OR accumulator with data memory
说明:		将累加器中的数据和指定的数据存储器内容逻辑或，结果存放到累加器。
运算过程:		$ACC \leftarrow ACC \text{ "OR" } [m]$
影响标志位:		Z

<b>OR</b>	<b>A, x</b>	Logical OR immediate data to the accumulator 说明：将累加器中的数据和立即数逻辑或，结果存放到累加器。 运算过程： $ACC \leftarrow ACC \text{ "OR" } x$ 影响标志位： <b>Z</b>
<b>ORM</b>	<b>A, [m]</b>	Logical OR data memory with accumulator 说明：将存在指定数据存储器中的数据和累加器逻辑或，结果放到数据存储器。 运算过程： $[m] \leftarrow ACC \text{ "OR" } [m]$ 影响标志位： <b>Z</b>
<b>RET</b>		Return from subroutine 说明：将堆栈寄存器中的程序计数器值恢复，程序由取回的地址继续执行。 运算过程： $PC \leftarrow Stack$ 影响标志位： <b>无</b>
<b>RET A, x</b>		Return and place immediate data in the accumulator 说明：将堆栈寄存器中的程序计数器值恢复且累加器载入指定的立即数，程序由取回的地址继续执行。 运算过程： $PC \leftarrow Stack$ $ACC \leftarrow x$ 影响标志位： <b>无</b>
<b>RETI</b>		Return from interrupt 说明：将堆栈寄存器中的程序计数器值恢复且中断功能通过设置 <b>EMI</b> 位重新使能。 <b>EMI</b> 是控制中断使能的主控制位。如果在执行 <b>RETI</b> 指令之前还有中断未被相应，则这个中断将在返回主程序之前被相应。 运算过程： $PC \leftarrow Stack$ $EMI \leftarrow 1$ 影响标志位： <b>无</b>
<b>RL</b>	<b>[m]</b>	Rotate data memory left 说明：将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位。 运算过程： $[m].(i+1) \leftarrow [m].i \quad (i=0\sim6)$ $[m].0 \leftarrow [m].7$ 影响标志位： <b>无</b>
<b>RLA</b>	<b>[m]</b>	Rotate data memory left and place result in the accumulator 说明：将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位，结果送到累加器，而指定数据存储器的内容保持不变。 运算过程： $ACC.(i+1) \leftarrow [m].i \quad (i=0\sim6)$ $ACC.0 \leftarrow [m].7$ 影响标志位： <b>无</b>

<b>RLC</b>	<b>[m]</b>	<b>Rotate data memory left through carry</b>
说明:		将指定数据存储器的内容连同进位标志左移 1 位, 第 7 位取代进位标志且原本的进位标志移到第 0 位。
运算过程:		$[m].(i+1) \leftarrow [m].i \quad (i=0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
影响标志位:		C
<b>RLCA</b>	<b>[m]</b>	<b>Rotate left through carry and place result in the accumulator</b>
说明:		将指定数据存储器的内容连同进位标志左移 1 位, 第 7 位取代进位标志且原本的进位标志移到第 0 位, 移位结果送回累加器, 但是指定数据寄存器的内容保持不变。
运算过程:		$ACC.(i+1) \leftarrow [m].i \quad (i=0\sim6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
影响标志位:		C
<b>RR</b>	<b>[m]</b>	<b>Rotate data memory right</b>
说明:		将指定数据存储器的内容循环右移 1 位且第 0 位移到第 7 位。
运算过程:		$[m].i \leftarrow [m].(i+1) \quad (i=0\sim6)$ $[m].7 \leftarrow [m].0,$
影响标志位:		无
<b>RRA</b>	<b>[m]</b>	<b>Rotate right and place result in the accumulator</b>
说明:		将指定数据存储器的内容循环右移 1 位, 第 0 位移到第 7 位, 移位结果存放到累加器, 而指定数据存储器的内容保持不变。
运算过程:		$ACC.i \leftarrow [m].(i+1) \quad (i=0\sim6)$ $ACC.7 \leftarrow [m].0$
影响标志位:		无
<b>RRC</b>	<b>[m]</b>	<b>Rotate data memory right through carry</b>
说明:		将指定数据存储器的内容连同进位标志右移 1 位, 第 0 位取代进位标志且原本的进位标志移到第 7 位。
运算过程:		$[m].i \leftarrow [m].(i+1) \quad (i=0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位:		C
<b>RRCA</b>	<b>[m]</b>	<b>Rotate right through carry and place result in the accumulator</b>
说明:		将指定数据存储器的内容连同进位标志右移 1 位, 第 0 位取代进位标志且原本的进位标志移到第 7 位, 移位结果送回累加器, 但是指定数据寄存器的内容保持不变。

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

影响标志位:        C

**SBC**        **A,[m]**        Subtract data memory and carry from the accumulator

说明:                将累加器减去指定数据存储器的内容以及进位标志的反, 结果存放到累加器。  
                          如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。

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

影响标志位:        OV、Z、AC、C

**SBCM**      **A,[m]**        Subtract data memory and carry from the accumulator

说明:                将累加器减去指定数据存储器的内容以及进位标志的反, 结果存放到数据存储器。  
                          如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。

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

影响标志位:        OV、Z、AC、C

**SDZ**        **[m]**            Skip if decrement data memory is 0

说明:                将指定的数据存储器的内容减 1, 判断是否为 0, 若为 0 则跳过下一条指令, 由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。

运算过程:             $[m] \leftarrow [m] - 1$ , 如果  $[m]=0$  跳过下一条指令执行

影响标志位:        无

**SDZA**      **[m]**            Decrement data memory and place result in ACC, skip if 0

说明:                将指定数据存储器内容减 1, 判断是否为 0, 如果为 0 则跳过下一条指令, 此结果将存放到累加器, 但指定数据存储器内容不变。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。

运算过程:             $ACC \leftarrow [m] - 1$ , 如果  $ACC=0$  跳过下一条指令执行

影响标志位:        无

**SET**        **[m]**            Set data memory

说明:                将指定数据存储器的每一位设置为 1。

运算过程:             $[m] \leftarrow FFH$

影响标志位:        无

**SET**        **[m].i**        Set bit of data memory

说明:                将指定数据存储器的第 i 位设置为 1。

运算过程:             $[m].i \leftarrow 1$

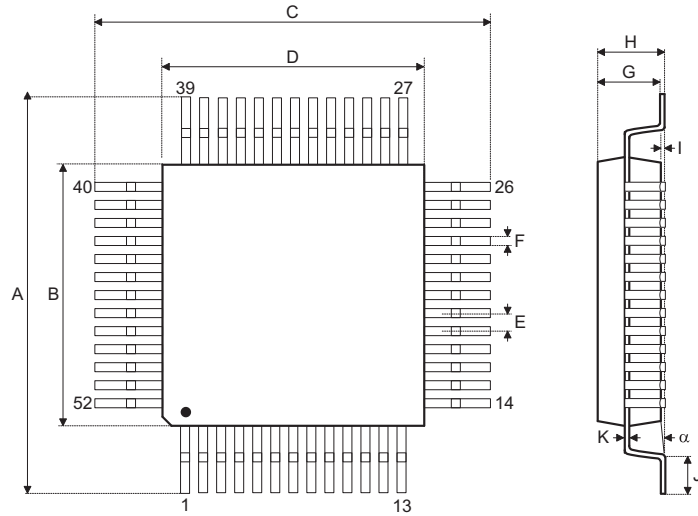
影响标志位:	无
<b>SIZ</b> [m]	Skip if increment data memory is 0
说明:	将指定的数据存储器的内容加 1, 判断是否为 0, 若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。
运算过程:	$[m] \leftarrow [m]+1$ , 如果 $[m]=0$ 跳过下一条指令执行
影响标志位:	无
<b>SIZA</b> [m]	Increment data memory and place result in ACC, skip if 0
说明:	将指定数据存储器的内容加 1, 判断是否为 0, 如果为 0 则跳过下一条指令, 此结果会被存放到累加器, 但是指定数据存储器的内容不变。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。
运算过程:	$ACC \leftarrow [m]+1$ , 如果 $ACC=0$ 跳过下一条指令执行
影响标志位:	无
<b>SNZ</b> [m]. i	Skip if bit I of the data memory is not 0
说明:	判断指定数据存储器的第 i 位, 若不为 0, 则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果为 0, 则程序继续执行下一条指令。
运算过程:	如果 $[m].i \neq 0$ , 跳过下一条指令执行
影响标志位:	无
<b>SUB</b> A, [m]	Subtract data memory from the accumulator
说明:	将累加器的内容减去指定的数据存储器的数据, 把结果存放到累加器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。
运算过程:	$ACC \leftarrow ACC - [m]$
影响标志位:	OV、Z、AC、C
<b>SUBM</b> A, [m]	Subtract data memory from the accumulator
说明:	将累加器的内容减去指定数据存储器的数据, 结果存放到指定的数据存储器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。
运算过程:	$[m] \leftarrow ACC - [m]$
影响标志位:	OV、Z、AC、C
<b>SUB</b> A, x	Subtract immediate data from the accumulator
说明:	将累加器的内容减去立即数, 结果存放到累加器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。
运算过程:	$ACC \leftarrow ACC - x$
影响标志位:	OV、Z、AC、C
<b>SWAP</b> [m]	Swap nibbles within the data memory

说明:	将指定数据存储器的低 4 位和高 4 位互相交换。
运算过程:	$[m].3\sim[m].0 \leftrightarrow [m].7\sim[m].4$
影响标志位:	无
<b>SWAPA</b> [m]	Swap data memory and place result in the accumulator
说明:	将指定数据存储器的低 4 位和高 4 位互相交换，再将结果存放到累加器且指定数据寄存器的数据保持不变。
运算过程:	$ACC.3\sim ACC.0 \leftarrow [m].7\sim[m].4$ $ACC.7\sim ACC.4 \leftarrow [m].3\sim[m].0$
影响标志位:	无
<b>SZ</b> [m]	Skip if data memory is 0
说明:	判断指定数据存储器的内容是否为 0，若为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
运算过程:	如果 $[m] = 0$ ，跳过下一条指令执行
影响标志位:	无
<b>SZA</b> [m]	Move data memory to ACC, skip if 0
说明:	将指定数据存储器内容复制到累加器，并判断指定数据存储器的内容是否为 0，若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
运算过程:	$ACC \leftarrow [m]$ ，如果 $[m] = 0$ ，跳过下一条指令执行
影响标志位:	无
<b>SZ</b> [m]. i	Skip if bit I of the data memory is 0
说明:	判断指定数据存储器的第 i 位是否为 0，若为 0，则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
运算过程:	如果 $[m].i = 0$ ，跳过下一条指令执行
影响标志位:	无
<b>TABRDC</b> [m]	Move the ROM code(current page) to TBLH and data memory
说明:	将表格指针 TBLP 所指的程序代码低字节（当前页）移至指定的数据存储器且将高字节移至 TBLH。
运算过程:	$[m] \leftarrow$ 程序代码（低字节） $TBLH \leftarrow$ 程序代码（高字节）
影响标志位:	无
<b>TABRDL</b> [m]	Move the ROM code(last page) to TBLH and data memory
说明:	将表格指针 TBLP 所指的程序代码低字节（最后一页）移至指定的数据存储器

		且将高字节移至 TBLH。
运算过程:		[m] ←程序代码 (低字节) TBLH←程序代码 (高字节)
影响标志位:		无
<b>XORA, [m]</b>		Logical XOR accumulator with data memory
说明:		将累加器的数据和指定的数据存储器内容逻辑异或, 结果存放到累加器。
运算过程:		ACC←ACC “XOR” [m]
影响标志位:		Z
<b>XORM A, [m]</b>		Logical XOR data memory with accumulator
说明:		将累加器的数据和指定的数据存储器内容逻辑异或, 结果放到数据存储器。
运算过程:		[m]←ACC “XOR” [m]
影响标志位:		Z
<b>XOR A, x</b>		Logical XOR immediate data to the accumulator
说明:		将累加器的数据与立即数逻辑异或, 结果存放到累加器。
运算过程:		ACC←ACC “XOR” x
影响标志位:		Z

封装信息

52-pin QFP (14mm×14mm)外形尺寸



符号	尺寸(单位: mm)		
	最小	正常	最大
A	17.3	—	17.5
B	13.9	—	14.1
C	17.3	—	17.5
D	13.9	—	14.1
E	—	1	—
F	—	0.4	—
G	2.5	—	3.1
H	—	—	3.4
I	—	0.1	—
J	0.73	—	1.03
K	0.1	—	0.2
α	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（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: 86-755-8616-9908, 86-755-8616-9308  
传真: 86-755-8616-9722

**Holtek Semiconductor(USA), Inc.（北美业务处）**

46729 Fremont Blvd., Fremont, CA 94538, USA  
电话: 1-510-252-9880  
传真: 1-510-252-9885  
网站: [www.holtek.com](http://www.holtek.com)

Copyright © 2009 by HOLTEK SEMICONDUCTOR INC.

使用指南中所出现的信息在出版当时相信是正确的，然而盛群对于说明书的使用不负任何责任。文中提到的应用目的仅仅是用来做说明，盛群不保证或表示这些没有进一步修改的应用将是适当的，也不推荐它的产品使用在会由于故障或其它原因可能会对人身造成危害的地方。盛群产品不授权使用于救生、维生从机或系统中做为关键从机。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>